Windows カスタムライブラリ

説明書

Ｃ／Ｃ++ 用レギュラーＤＬＬ
（AjrCst32.dll ／ AjrCst64.dll）編

（ 1.6.8.6版 ）

－　項目一覧　－

1. 概　要 ... 1
2. 基本タイプ定義 ... 3
3. メッセージマップ ... 4
4. カスタムコントロール ... 36
5. タイムチャート・グラフ表示コントロール（AjcCtrlTmChart クラス） ... 38
6. ３Ｄ／２Ｄグラフィック・コントロール（AjcCtrl3dGraph クラス） ... 75
7. ＶＴ－１００エミュレーション・ウインド コントロール（AjcCtrlVT100 クラス） ... 144
8. 数値入力コントロール（AjcCtrlInpVal クラス） ... 202
9. ログファイル出力コントロール（AjcCtrlLogFile クラス） ... 218
10. 棒グラフ／折れ線グラフ表示コントロール（AjcCtrlBarGraph クラス） ... 230
11. 拡張リストボックス・コントロール（AjcCtrlListBox クラス） ... 256
12. 通信コントロールで共通な内容の説明 ... 281
13. シリアル通信 ... 287
14. ソケット(TCP/IP)サーバ機能 ... 367
15. ソケット(TCP/IP)クライアント機能 ... 399
16. ダイアログボックス、コントロール群とメニューのカラー設定 ... 429
17. ダイアログボックス項目／ウインド項目のアクセス ... 455
18. ウインドサポートＡＰＩ ... 490
19. 拡張コンボボックス ... 498
20. ラジオボタンのグループ操作 ... 509
21. 他プロセス内のメモリ操作 ... 514
22. 拡張ツールチップ ... 524
23. コンソール入出力 ... 551
24. プロファイル・アクセス ... 557
25. ＩＮＩファイル・アクセス ... 580
26. レジストリ・アクセス ... 585
27. ファイル名／フォルダ名の取得 ... 598
28. ボリュームラベル操作 ... 602
29. ファイル検索 ... 604
30. ファイル／フォルダ操作 ... 610
31. テキストファイル・アクセス ... 632
32. 平衡２分木（ＡＶＬ木） ... 655
33. 平衡２分木（文字列キー） ... 670
34. 線形リスト制御 ... 682
35. 双方向リスト制御 ... 696
36. スレッド間メールデータ通信 ... 707
37. リングバッファ制御 ... 717
38. Ｃ言語の字句分解 ... 723
39. Ｃ言語のプリコンパイル ... 737
40. 変数管理 ... 759
41. 文字列操作 ... 771
42. 文字列プール ... 800
43. 日付／時刻 ... 804
44. 時間計測 ... 812
45. イメージの描画 ... 814
46. ＬＺＨ書庫データのアクセス ... 819
47. ビットマップ操作 ... 827
48. ＤＩＢセクション ... 831
49. テキスト描画 ... 841
50. 時計表示 ... 859
51. XMODEM／YMODEM ファイル転送プロトコル ... 863
52. 印 刷 ... 883
53. 高速フーリエ変換 ... 901
54. モニタ情報 ... 907
55. ヒープソート ... 913
56. 演算 ... 914
57. ３Ｄテストデータ生成 ... 932
58. バイトストリーム分離 ... 934

59. 状態遷移制御 945
60. ＣＲＣ／チェックサム 957
61. ＨＥＸデータ処理 963
62. フォント処理とフォント選択ダイアログ 967
63. その他 977
64. ＭＦＣでの利用 995
65. Windows API の不具合 1011
66. 問い合わせ先 1013

－　目次　－

1.　概　要 1
1.1.　依存ファイル 1
1.2.　実行可能な Windows バージョン 1
1.3.　バイト文字バージョンとワイド文字(UNICODE)バージョン 2
1.4.　マルチバイト文字 2
1.5.　文字列比較モード 2
1.6.　プロファイルへの記録による設定内容の永続化 2
1.7.　バージョン更新時の注意事項 2
2.　基本タイプ定義 3
3.　メッセージマップ 4
3.1.　ウインドプロシージャのメッセージマップ 4
3.1.1.　メッセージマップテーブル 6
3.1.2.　メッセージアクション関数 6
3.1.3.　ウインドプロシージャ名 7
3.1.4.　複数のコントロールを一括処理 7
3.2.　ダイアログプロシージャのメッセージマップ 8
3.2.1.　メッセージマップテーブル 10
3.2.2.　メッセージアクション関数 10
3.2.3.　ダイアログプロシージャ名 11
3.2.4.　複数のコントロールを一括処理 11
3.3.　動的なメッセージコードのメッセージマップ 12
3.4.　ウインドのサブクラス化 12
3.4.1.　サブクラス化後の UNICODE ウインド属性 15
3.4.2.　サブクラス化によるテキストメッセージの受け渡し 15
3.5.　サポートＡＰＩ 16
3.5.1.　ウインドプロシージャのサブクラス化(MAjcMmpSetSubclass) 16
3.5.2.　ウインドプロシージャのサブクラス解除(MAjcMmpClrSubclass) 16
3.5.3.　ウインドプロシージャのサブクラスを全て解除(MAjcMmpDelSubclasses) 16
3.5.4.　デフォルトウインドプロシージャの設定(MAjcMmpSetDefWndProc) 17
3.5.5.　オリジナルウインドプロシージャの呼び出し(AjcMmpSetSubclass) 17
3.5.6.　サブクラス化情報の取得(AjcMmpGetSubclassInfo) 17
3.6.　サンプルプログラム 18
3.6.1.　SW_SubClass1（多重にサブクラス化） 18
3.6.2.　SW_SubClass2（複数のウインドを１つのウインドプロシージャでサブクラス化） 23
3.6.3.　SW_SubClass3（サブクラス化してメッセージをインターセプト） 25
3.6.4.　SW_SubClass4（サブクラス化してメッセージをトラップ） 28
3.6.5.　SW_Broadcast1～3（ブロードキャストメッセージ） 30
4.　カスタムコントロール 36
4.1.　言語設定 36
4.2.　ダイアログボックスでの利用 37
5.　タイムチャート・グラフ表示コントロール（AjcCtrlTmChart クラス） 38
5.1.　機能概要 38
5.1.1.　ポップアップメニュー 38
5.1.2.　レンジ設定 39
5.1.3.　ワンタッチでレンジ設定 39
5.1.4.　レンジ自動調整 39
5.1.5.　ドラッグ操作によるレンジ設定 39
5.1.6.　オフセット設定 40
5.1.7.　その他の設定 40
5.1.8.　フィルタの設定と表示／非表示 41
5.1.9.　波形の補間表示設定 41
5.1.10.　波形の補間表示 42
5.1.11.　横線描画 43
5.1.12.　縦線描画 43
5.1.13.　プロット周期表示 44
5.1.14.　時間計測 44
5.1.15.　描画時間情報表示 45
5.1.16.　ファイルやディレクトリのドラッグ＆ドロップ 45

5.1.17. 表示の高速化 45
5.2. コントロールのスタイル 46
5.3. プロパティ構造体 46
5.4. キャプション文字列によるプロパティの設定 47
5.5. テキストの取得と設定 47
5.6. プロパティの永続化 48
5.7. サポートＡＰＩ 49
5.7.1. グラフ表示停止／開始(AjcTchStop/ AjcTchStart) 50
5.7.2. データ破棄(AjcTchPurge) 50
5.7.3. データ投与(AjcTchPutRealData) 50
5.7.4. コントロールの外枠表示 (AjcTchShowBorder) 50
5.7.5. フィルタ (チェックボックス) の表示／非表示 (AjcTchShowFilter) 51
5.7.6. スケールの表示／非表示 (AjcTchShowScale) 51
5.7.7. プロパティ設定／取得 (AjcTch{Set/Get}Prop) 51
5.7.8. グラフ・レンジの設定／取得(AjcTch{Set/Get}{Real | Int}Range) 51
5.7.9. グラフ・レンジを自動調整(AjcTchAdjustRange) 52
5.7.10. バッファサイズの設定 (AjcTchSetBufSize) 52
5.7.11. データ項目数の設定 (AjcTchSetItemNumber) 52
5.7.12. 平均化個数の設定 (AjcTchSetAveNumber) 52
5.7.13. グラフ横軸スケール幅の設定 (AjcTchSetTimeScale) 53
5.7.14. ビットマップデータ取得 (AjcTchGetBitmap) 53
5.7.15. 右クリック通知設定 (AjcTchSetNtcRClk) 53
5.7.16. プロファイルからプロパティ値読出し／書込み (AjcTchLoadProp / AjcTchLoadPropEx) 54
5.7.17. ポップアップメニューの許可／禁止 (AjcTchEnablePopupMenu) 54
5.7.18. ツールチップの設定／取得 (AjcTchSetTipText / AjcTchGetTipText) 55
5.7.19. ツールチップ表示条件の設定／取得(AjcTch{Set|Get}TipShowAlways) 55
5.7.20. フィルタチェックボックス・ツールチップの設定／取得 (AjcTch{Set/Get}ChkBoxTipText) 55
5.7.21. フィルタチェックボックス・ツールチップ表示条件の設定／取得(AjcTch{Set|Get}ChkBoxTipShowAlways) 55
5.7.22. 最大結線長の設定／取得 (AjcTch{Set/Get}MaxLineDist) 56
5.7.23. スクロール位置の設定／取得 (AjcTch{Set/Get}ScrollPos) 56
5.7.24. フィルタの設定／取得 (AjcTch{Set/Get}Filter) 56
5.7.25. 横線の属性設定 (AjcTchSetHLineAtt) 57
5.7.26. 横線の描画位置設定 (AjcTchSetHLinePos) 57
5.7.27. 横線描画の許可／禁止(AjcTchEnableHLine) 57
5.7.28. 縦線の描画 (AjcTchSetVLine) 58
5.7.29. 縦線描画の許可／禁止(AjcTchEnableVLine) 58
5.7.30. プロット点描画(AjcTchSetPoint) 58
5.7.31. プロット点の許可／禁止(AjcTchEnablePoint) 58
5.7.32. ドロップされたファイル名取得(AjcTchGetDroppedFile) 59
5.7.33. ドロップされたディレクトリ名取得(AjcTchGetDroppedDir[Ex]) 59
5.7.34. タイトル文字列の設定(AjcTchSetTitleText) 59
5.7.35. 波形補間表示情報の設定／取得(AjcTch{Set/Get}IpInfo) 60
5.7.36. 画面表示の一時停止／再開 (AjcTchPause) 60
5.7.37. 描画時間計測情報の許可／禁止 (AjcTchEnableMesDraw) 60
5.7.38. プロット周期計測値の表示(AjcTchMesPeriShow) 61
5.7.39. プロット周期計測をリセット (AjcTchMesPeriReset) 61
5.7.40. プロット周期計測値取得 (AjcTchMesPeriGet) 61
5.7.41. 時間計測ゲージ情報取得 (AjcTchMesPeriGet) 61
5.7.42. 設定情報の永続化 (AjcTchLoadPermInfo[Ex] / AjcTchSavePermInfo[Ex]) 62
5.7.43. テキスト描画フォント設定(AjcTchSetTextFont) 62
5.7.44. テキスト描画(AjcTchTextOut) 63
5.7.45. 書式テキスト描画(AjcTchPrintF) 63
5.7.46. 描画テキスト取得(AjcTchGetText) 63
5.7.47. 描画テキスト消去(AjcTchClear[All]Text) 63
5.7.48. 全てのデータ消去(AjcTchClear) 64
5.7.49. イベント通知用コールバック関数の設定(AjcTchSetEvtCallBack) 64
5.8. 通知情報の取得ＡＰＩ 65
5.8.1. グラフレンジ情報の取得 (AjcTchGetNtcRng) 65
5.8.2. スクロール位置の取得 (AjcTchGetNtcScrPos) 65
5.8.3. ドロップしたファイル数の取得 (AjcTchGetNtcFiles) 65

5.8.4. ドロップしたディレクトリ数の取得（AjcTchGetNtcDirs） 65
5.8.5. 右クリック通知情報の取得（AjcTchGetNtcRClk） 66
5.9. コントロールの通知メッセージ 67
5.9.1. グラフレンジ通知（AJCTCN_RANGE） 67
5.9.2. スクロール位置通知（AJCTCN_SCRPOS） 67
5.9.3. データクリアー通知（AJCTCN_CLEAR） 67
5.9.4. ダブルクリック通知（AJC3DGN_DBLCLK） 68
5.9.5. ファイルドロップ通知（AJCTCN_DROPFILE） 68
5.9.6. ディレクトリドロップ通知（AJCTCN_DROPDIR） 68
5.9.7. 右クリック通知（AJCTCN_RCLICK） 68
5.10. サンプルプログラム 69
5.10.1. SW_TimeChart（２つのタイムチャート） 69
6. ３Ｄ／２Ｄグラフィック・コントロール（AjcCtrl3dGraph クラス） 75
6.1. プロットデータと図形描画データ 76
6.2. 機能概要 77
6.2.1. ポップアップメニュー 77
6.2.2. レンジ設定 78
6.2.3. ワンタッチでレンジ設定 78
6.2.4. レンジ自動調整 78
6.2.5. アスペクト比の設定 79
6.2.6. フィルタ機能 79
6.2.7. 視点の設定 79
6.2.8. スケールの表示 80
6.2.9. 描画時間情報表示 81
6.2.10. 奥行き表現 81
6.2.11. ２Ｄグラフモード 82
6.2.12. 表示の高速化 83
6.2.13. 任意の平面定義と平面上への図形描画 83
6.2.14. ファイルやディレクトリのドラッグ＆ドロップ 84
6.3. コントロールのスタイル 84
6.4. ベクトルデータ構造体 85
6.5. コントロールのプロパティ構造体 85
6.6. キャプション文字列によるプロパティの設定 86
6.7. テキストの取得と設定 87
6.8. プロパティの永続化 87
6.9. サポートＡＰＩ 88
6.9.1. ３Ｄグラフモード初期化(AjcG3dInit[V]) ・・・ ３Ｄモード専用 90
6.9.2. プロパティ設定／取得（AjcG3d{Set/Get}Prop） ・・・ ３Ｄモード専用 90
6.9.3. レンジ自動調整(AjcG3dAdjustRange) ・・・ ３Ｄモード専用 90
6.9.4. レンジ設定（AjcG3dSetRange） ・・・ ３Ｄモード専用 91
6.9.5. 各座標軸の中心位置（原点）設定(AjcG3dSetCenter) ・・・ ３Ｄモード専用 91
6.9.6. 各座標軸の長さ設定(AjcG3dSetWidth) ・・・ ３Ｄモード専用 91
6.9.7. 各軸のレンジ幅を同じにする（AjcG3dSetSameRangeWidth） ・・・ ３Ｄモード専用 91
6.9.8. レンジ設定された領域のアスペクトを設定する（AjcG3dSetFixedAspect） ・・・ ３Ｄモード専用 92
6.9.9. 表示色設定(AjcG3dSetColor) ・・・ ３Ｄモード専用 92
6.9.10. プロットデータ数設定(AjcG3dSetPlotNumber) ・・・ ３Ｄモード専用 92
6.9.11. プロットデータのピクセルサイズ設定(AjcG3dSetPlotSize) ・・・ ３Ｄモード専用 93
6.9.12. プロットデータ投与(AjcG3dPutPlotData) ・・・ ３Ｄモード専用 93
6.9.13. プロットデータ数取得(AjcG3dGetPlotDataCount) ・・・ ３Ｄモード専用 93
6.9.14. ピクセル描画(AjcG3dPixel) ・・・ ３Ｄモード専用 94
6.9.15. ライン始点設定(AjcG3dMoveTo) ・・・ ３Ｄモード専用 94
6.9.16. ライン終点設定 - ライン／矢印描画(AjcG3dMoveTo) ・・・ ３Ｄモード専用 94
6.9.17. ライン／矢印描画(AjcG3dLine) ・・・ ３Ｄモード専用 95
6.9.18. 三角形描画(AjcG3dTriangle) ・・・ ３Ｄモード専用 95
6.9.19. 四角形描画(AjcG3dSuare) ・・・ ３Ｄモード専用 95
6.9.20. 立方体／長方体描画(AjcG3dCube) ・・・ ３Ｄモード専用 96
6.9.21. 球／楕球描画(AjcG3dSphere) ・・・ ３Ｄモード専用 96
6.9.22. ３Ｄ空間上に任意の平面を定義(AjcG3dDefPlane) ・・・ ３Ｄモード専用 97
6.9.23. 視点設定(３軸回転指定, AjcG3dSetAngle) ・・・ ３Ｄモード専用 98
6.9.24. 視点設定(視点ベクトル指定, AjcG3dSetAngleV) ・・・ ３Ｄモード専用 98

6.9.25. 視点をＸ－Ｙ平面に設定(AjcG3dSetAngleXY) ・・・ ３Ｄモード専用 98
6.9.26. 視点をＸ－Ｚ平面に設定(AjcG3dSetAngleXZ) ・・・ ３Ｄモード専用 99
6.9.27. 視点をＹ－Ｚ平面に設定(AjcG3dSetAngleYZ) ・・・ ３Ｄモード専用 99
6.9.28. 視点を３Ｄイメージに設定(AjcG3dSetAngle3D) ・・・ ３Ｄモード専用 99
6.9.29. 平面アングル設定 (AjcG3dSetPlane) ・・・ ３Ｄモード専用 99
6.9.30. プロファイルからプロパティ値読出し／書き込み (AjcG3d{Load/Save}Prop[Ex]) ・・・ ３Ｄモード専用 100
6.9.31. ビットマップ取得 (AjcG3dGetBitmap) ・・・ ３Ｄモード専用 100
6.9.32. テキスト描画フォント設定(AjcG3dSetTextFont) ・・・ ３Ｄモード専用 101
6.9.33. テキスト描画(AjcG3dTextOut[V]) ・・・ ３Ｄモード専用 101
6.9.34. 書式テキスト描画(AjcG3dPrintF[V]) ・・・ ３Ｄモード専用 102
6.9.35. 描画テキスト取得(AjcG3dGetText) ・・・ ３Ｄモード専用 102
6.9.36. 図形描画データ消去(AjcG3dClear[All]Shape) ・・・ ３Ｄモード専用 102
6.9.37. プロットデータ消去(AjcG3dClear[All]Plot) ・・・ ３Ｄモード専用 102
6.9.38. 描画テキスト消去(AjcG3dClear[All]Text) ・・・ ３Ｄモード専用 103
6.9.39. 描画データ消去(AjcG3dClear[All]Data) ・・・ ３Ｄモード専用 103
6.9.40. 全てのデータ消去(AjcG3dClear) ・・・ ３Ｄモード専用 103
6.9.41. ２Ｄグラフモード初期化(AjcG2dInit[V]) ・・・ ２Ｄモード専用 104
6.9.42. 平面アングル設定(AjcG2dSetPlane) ・・・ ２Ｄモード専用 104
6.9.43. プロパティ設定／取得 (AjcG2d{Set/Get}Prop) ・・・ ２Ｄモード専用 105
6.9.44. レンジ自動調整(AjcG2dAdjustRange) ・・・ ２Ｄモード専用 105
6.9.45. 各軸のレンジ設定 (AjcG2dSetRange) ・・・ ２Ｄモード専用 105
6.9.46. 各座標軸の中心位置設定(AjcG2dSetCenter) ・・・ ２Ｄモード専用 105
6.9.47. 各軸の幅 (半径) 設定(AjcG2dSetWidth) ・・・ ２Ｄモード専用 106
6.9.48. 各軸のレンジ幅を同じにする (AjcG2dSetSameRangeWidth) ・・・ ２Ｄモード専用 106
6.9.49. レンジ設定された領域のアスペクトを設定する (AjcG2dSetFixedAspect) ・・・２Ｄモード専用 107
6.9.50. 表示色設定(AjcG2dSetColor) ・・・ ２Ｄモード専用 107
6.9.51. プロットデータ数設定(AjcG2dSetPlotNumber) ・・・ ２Ｄモード専用 107
6.9.52. プロットデータのピクセルサイズ設定(AjcG2dSetPlotSize) ・・・ ２Ｄモード専用 108
6.9.53. プロットデータ投与(AjcG2dPutPlotData) ・・・ ２Ｄモード専用 108
6.9.54. プロットデータ数取得(AjcG3dGetPlotDataCount) ・・・ ２Ｄモード専用 108
6.9.55. プロファイルからプロパティ値読出し／書き込み (AjcG2dLoadProp[Ex]) ・・・ ２Ｄモード専用 109
6.9.56. ボーダー色で囲まれた部分の塗りつぶし(AjcG2dFillB) ・・・ ２Ｄモード専用 109
6.9.57. 連続する白色部分の塗りつぶし(AjcG2dFillS) ・・・ ２Ｄモード専用 109
6.9.58. ピクセルの表示色取得 (AjcG2dGetPixel) ・・・ ２Ｄモード専用 110
6.9.59. ビットマップデータ取得 (AjcG2dGetBitmap) ・・・ ２Ｄモード専用 110
6.9.60. テキスト描画フォント設定(AjcG2dSetTextFont) ・・・ ２Ｄモード専用 110
6.9.61. テキスト描画(AjcG2dTextOut[V]) ・・・ ２Ｄモード専用 111
6.9.62. 書式テキスト描画(AjcG2dPrintF[V]) ・・・ ２Ｄモード専用 112
6.9.63. 描画テキスト取得(AjcG2dGetText) ・・・ ２Ｄモード専用 112
6.9.64. 図形描画データ消去(AjcG2dClearAll) ・・・ ２Ｄモード専用 112
6.9.65. プロットデータ消去(AjcG2dClear[All]Plot) ・・・ ２Ｄモード専用 112
6.9.66. 描画テキスト消去(AjcG2dClear[All]Text) ・・・ ２Ｄモード専用 113
6.9.67. 描画データ消去(AjcG2dClear[All]Data) ・・・ ２Ｄモード専用 113
6.9.68. 全てのデータ消去(AjcG3dClear) ・・・ ２Ｄモード専用 113
6.9.69. 平面にピクセル描画(AjcG2dPixel) 113
6.9.70. 平面のライン始点設定(AjcG2dMoveTo) 114
6.9.71. 平面のライン終点設定 - ライン／矢印描画(AjcG2dMoveTo) 114
6.9.72. 平面にライン／矢印描画(AjcG2dLine) 114
6.9.73. 平面に三角形描画(AjcG2dTriangle) 115
6.9.74. 平面に四角形描画(AjcG2dSquare) 115
6.9.75. 平面に長方形描画(AjcG2dRectangle) 115
6.9.76. 平面に円／楕円描画(AjcG2dEllipse) 116
6.9.77. 平面に星形描画(AjcG2dStar[V][Ex]) 116
6.9.78. 右クリック通知設定 (AjcG3dSetNtcRClk) 117
6.9.79. ポップアップメニューの許可／禁止 (AjcG3dEnablePopupMenu) 117
6.9.80. ツールチップの設定／取得 (AjcG3d{Set/Get}TipText) 117
6.9.81. ツールチップ表示条件の設定／取得(AjcG3d{Set|Get}TipShowAlways) 118
6.9.82. フィルタ・チェックボックス・ツールチップの設定／取得 (AjcG3d{Set/Get}ChkBoxTipText) 118
6.9.83. フィルタ・チェックボックス・ツールチップ表示条件の設定／取得(AjcG3d{Set|Get}ChkBoxTipShowAlways) ... 118
6.9.84. フィルタの設定／取得 (AjcG3d{Set/Get}Filter) 118

6.9.85. ドロップされたファイル名取得(AjcG3dGetDroppedFile) ... 119
6.9.86. ドロップされたディレクトリ名取得(AjcG3dGetDroppedDir[Ex]) ... 119
6.9.87. タイトル文字列の設定 (AjcG3dSetTitleText) ... 119
6.9.88. 画面表示の停止／再開 (AjcG3dPause) ... 120
6.9.89. 描画時間計測情報の許可／禁止 (AjcG3dEnableMesDraw) ... 120
6.9.90. イベント通知用コールバック関数の設定(AjcTG3dSetEvtCallBack) ... 120
6.10. 通知情報の取得ＡＰＩ ... 121
6.10.1. 視点角度情報の取得 (AjcG3dGetNtcRotTheta) ... 121
6.10.2. プロットリスト情報の取得 (AjcG3dGetNtcList) ... 121
6.10.3. データクリアー情報の取得 (AjcG3dGetNtcClrFact) ... 121
6.10.4. ドロップしたファイル数の取得 (AjcG3dGetNtcFiles) ... 122
6.10.5. ドロップしたディレクトリ数の取得 (AjcG3dGetNtcDirs) ... 122
6.10.6. 右クリック情報の取得 (AjcG3dGetNtcRClk) ... 122
6.11. コントロールの通知メッセージ ... 122
6.11.1. 視点変化通知 (AJC3DGN_ROTTHETA) ... 123
6.11.2. プロットリスト通知 (AJC3DGN_PLOTLIST) ... 123
6.11.3. データクリアー通知 (AJC3DGN_CLEAR) ... 124
6.11.4. ダブルクリック通知 (AJC3DGN_DBLCLK) ... 124
6.11.5. ファイルドロップ通知 (AJC3DGN_DROPFILE) ... 124
6.11.6. ディレクトリドロップ通知 (AJC3DGN_DROPDIR) ... 124
6.11.7. 右クリック通知 (AJC３DGN_RCLICK) ... 124
6.12. サンプルプログラム ... 125
6.12.1. SW_3DGraphic1 (３Ｄグラフィックの描画と軌道演算) ... 125
6.13. SW_3DGraphic2 (２D グラフィック) ... 136
6.14. SW_3DGraphic3 (視点設定) ... 141
7. ＶＴ－１００エミュレーション・ウインド コントロール (AjcCtrlVT100 クラス) ... 144
7.1. 機能概要 ... 144
7.1.1. ポップアップメニュー ... 144
7.1.2. 文字列の検索 ... 145
7.1.3. テキストの選択とコピー ... 146
7.1.4. フォントの設定 ... 146
7.1.5. その他の設定 ... 146
7.1.6. ファイルへセーブ ... 147
7.1.7. 表示の高速化 ... 147
7.1.8. 描画処理の高速化 ... 148
7.1.9. ＡＮＳＩエスケープコード ... 148
7.1.10. キー操作 ... 148
7.1.11. 描画メモリとカーソル位置 ... 149
7.1.12. スクロールアウトした行のバッファリング ... 149
7.1.13. オートスクロール ... 150
7.1.14. ＶＲＡＭとバッファを識別して表示 ... 150
7.1.15. 描画テキストの一時保留 ... 150
7.1.16. マージン ... 151
7.1.17. 表示内容をファイルへ出力 ... 151
7.1.18. ファイルやディレクトリのドロップ ... 151
7.1.19. 固定ピッチ表示 ... 152
7.1.20. 画面クリアーボタン ... 152
7.2. 制御コードとＡＮＳＩエスケープコード ... 153
7.3. コントロールのスタイル ... 154
7.4. コントロールのプロパティ構造体 ... 155
7.5. キャプション文字列によるプロパティの設定 ... 156
7.6. テキストの取得と設定 ... 156
7.7. WM_SETFONT について ... 157
7.8. プロパティの永続化 ... 157
7.9. サポートＡＰＩ ... 158
7.9.1. プロパティ設定／取得(AjcVth{Set/Get}Prop) ... 160
7.9.2. プロパティ・リセット(AjcVthResetProp) ... 160
7.9.3. ＶＲＡＭにフィットしたウインドサイズ取得(AjcVthGetVramFitSize) ... 160
7.9.4. １文字描画(AjcVthPutChar) ... 160
7.9.5. テキスト描画 (AjcVthPutText) ... 161

7.9.6. 日本語ＥＵＣコードテキスト描画 (AjcVthPutTextEUC) 161
7.9.7. ＵＴＦ－８コードテキスト描画 (AjcVthPutTextUTF8) 161
7.9.8. 文字コードを自動判別してテキスト描画 (AjcVthPutTextAuto) 161
7.9.9. 書式テキスト描画(AjcVthPrintF) 162
7.9.10. タイムスタンプ描画(AjcVthTimeStamp) 162
7.9.11. バイナリデータの１６進ダンプ描画(AjcVthHexDump) 162
7.9.12. カーソル位置設定／取得(AjcVthLocate / AjcVthGetCursorPos) 163
7.9.13. テキスト描画用パレット選択／選択パレット番号取得(AjcVth{Set/Get}Color) 163
7.9.14. 文字背景色用パレット選択／選択パレット番号取得(AjcVth{Set/Get}BkColor) 163
7.9.15. パレットの色コード設定／取得(AjcVth{Set/Get}Palette) 164
7.9.16. ウインドの背景色設定／取得(AjcVth{Set/Get}WndtBkColor) 164
7.9.17. 部分テキストを選択 (AjcVthSelect) 164
7.9.18. 全てのテキストを選択 (AjcVthSelectAll) 164
7.9.19. 選択テキストをクリップボードへコピー (AjcVthCopyText) 165
7.9.20. ダイアログによるフォント設定 (AjcVthSetFont) 165
7.9.21. キャレット表示／非表示 (AjcVthShowCaret) 165
7.9.22. 全テキストクリアー (AjcVthClear[AllText]) 165
7.9.23. ドロップされたファイル名取得(AjcVthGetDroppedFile) 166
7.9.24. ドロップされたディレクトリ名取得(AjcVthGetDroppedDir[Ex]) 166
7.9.25. 右クリック通知設定 (AjcVthSetNtcRClk) 166
7.9.26. ポップアップメニューの許可／禁止 (AjcVthEnablePopupMenu) 167
7.9.27. ポップアップメニュー項目の有効化／無効化 (AjcVth{Sel/Exc}MenuItems) 167
7.9.28. テキストの取得(AjcVthGetText) 168
7.9.29. テキスト選択の取得(AjcVthIsSelected) 168
7.9.30. 選択されているテキストの取得(AjcVthGetSelectedText) 168
7.9.31. ダブルクリックした行位置のテキスト取得(AjcVthGetDblClickedLine[Ex]) 168
7.9.32. ツールチップの設定／取得 (AjcVth{Set/Get}TipText) 169
7.9.33. ツールチップ表示条件の設定／取得(AjcVth{Set|Get} TipShowAlways) 169
7.9.34. フォント，行間スペースの設定／取得(AjcVth{Set/Get}FontInfo) 169
7.9.35. 文字サイズ／行の高さ取得(AjcVthGetCharInfo) 169
7.9.36. ウインドに表示可能な行数の取得(AjcVthGetLinesPerWindow) 170
7.9.37. バッファに格納されている有効な行数の取得(AjcVthGetValidLines) 170
7.9.38. ウインド先頭行の位置取得 (AjcVthGetIxOfWndTopLine) 170
7.9.39. タイトル文字列の設定 (AjcVthSetTitleText) 170
7.9.40. プロファイルからプロパティ値読出し／書き込み (AjcVth{Load/Save}Prop) 171
7.9.41. テキストをファイルへ書き込み (AjcVthSaveTextToFile) 172
7.9.42. テキストをＨＴＭＬファイルへの書き込み (AjcVthSaveHtmlToFile) 172
7.9.43. ＥＳＣシーケンスを付加し全テキストをファイルへ書き込み (AjcVthSaveAllTextWithEsc) 172
7.9.44. バッファに格納されている行数の取得 (AjcVthGetLineCount) 172
7.9.45. マウスカーソル位置の行番号と文字位置取得 (AjcVthGetCursorPosInfo) 173
7.9.46. 指定行位置の行テキスト取得 (AjcVthGetLineText) 173
7.9.47. 縦スクロール位置の設定／取得 (AjcVth{Set/Get}VScrollPos) 173
7.9.48. 横スクロール位置の設定／取得 (AjcVth{Set/Get}HScrollPos) 174
7.9.49. 表示ウインドサイズ (行数, 文字数) 取得 (AjcVthGetWindowSize) 174
7.9.50. 文字列の検索 (AjcVthSearchBelow) 174
7.9.51. 画面表示の停止／再開 (AjcVthPause) 175
7.9.52. 設定情報の永続化 (AjcVthLoadPermInfo[Ex] / AjcVthSavePermInfo) 175
7.9.53. 文字列検索情報永続化プロファイルセクション名設定 (AjcVthSetFindProfileSect) 176
7.9.54. 文字列検索キーの設定 (AjcVthSetFindKey) 176
7.9.55. メニューキーの設定 (AjcVthSetMenuKey) 176
7.9.56. ベント通知用コールバック関数の設定(AjcVthSetEvtCallBack) 177
7.10. 通知情報の取得ＡＰＩ 178
7.10.1. 入力キーコードの取得 (AjcVthGetNtcKey) 178
7.10.2. キー押し続けによる繰り返し情報の取得 (AjcVthGetNtcKeyRep) 178
7.10.3. ドロップしたファイル数の取得 (AjcVthGetNtcFiles) 179
7.10.4. ドロップしたディレクトリ数の取得 (AjcVthGetNtcDirs) 179
7.10.5. 右クリック情報の取得 (AjcVthGetNtcRClk) 179
7.10.6. ダブルクリック情報の取得 (AjcVthGetNtcDblClk) 179
7.10.7. 文字サイズ情報の変化通知時の行の高さ取得 (AjcVthGetNtcCyLine) 180
7.10.8. 横スクロール通知時の左端桁位置取得 (AjcVthGetNtcLeft) 180

7.10.9. 縦スクロール通知時の上端行位置取得（AjcVthGetNtcTop） ... 180
7.11. 通知メッセージ ... 181
7.11.1. ダブルクリック通知（AJCVTHN_DBLCLK） ... 181
7.11.2. キー入力通知（AJCVTHN_KEYIN） ... 181
7.11.3. 拡張キー押下通知（AJCVTHN_VKEYIN） ... 182
7.11.4. 拡張キー離し通知（AJCVTHN_VKEYOUT） ... 182
7.11.5. ファイルドロップ通知（AJCVTHN_DROPFILE） ... 182
7.11.6. ディレクトリドロップ通知（AJCVTHN_DROPDIR） ... 183
7.11.7. 文字サイズ情報の変化通知（AJCVTHN_CHARINFO） ... 183
7.11.8. 横スクロール通知（AJCVTHN_HSCROLL） ... 183
7.11.9. 縦スクロール通知（AJCVTHN_VSCROLL） ... 183
7.11.10. 右クリック通知（AJCVTHN_RCLICK） ... 184
7.11.11. 画面クリアー通知（AJCVTHN_CLEAR） ... 184
7.12. サンプルプログラム ... 185
7.13. SW_VT100（VT100 エミュレーションウインド） ... 185
7.14. SW_VT100__MesTime（VT100 描画とウインド表示時間の計測） ... 193
8. 数値入力コントロール（AjcCtrlInpVal クラス） ... 202
8.1. ツールチップテキスト ... 203
8.1.1. コントロールのスタイル ... 204
8.2. コントロールのプロパティ構造体 ... 204
8.3. キャプション文字列によるプロパティの設定 ... 205
8.4. テキストの取得と設定 ... 205
8.5. サポートＡＰＩ ... 206
8.5.1. テキスト表示書式の設定（ AjcIvSetTextFormat ） ... 207
8.5.2. プロパティの設定／取得取得（ AjcIv{Set/Get}Prop ） ... 207
8.5.3. コントロール外枠の表示色設定／取得（ AjcIv{Set/Get}BorderColor ） ... 207
8.5.4. 数値入力時のブリンク色設定／取得（AjcIv{Set/Get}BlinkColor ） ... 207
8.5.5. スライダのウインドハンドル取得（ AjcIvGetSilderHandle ） ... 207
8.5.6. テキストボックスのウインドハンドル取得（AjcIvGetTxtHandle ） ... 208
8.5.7. ボタンのウインドハンドル取得（ AjcIvGetBtnHandle） ... 208
8.5.8. スピンボタンのウインドハンドル取得（ AjcIvGetSpnHandle ） ... 208
8.5.9. テキストボックスのサイズ（桁数）設定／取得（ AjcIv{Set/Get}TxtLen ） ... 208
8.5.10. 右ボタン操作通知設定（ AjcIvSetNtcRClk ） ... 209
8.5.11. 値の設定／取得（ AjcIvGetValue ） ... 209
8.5.12. 数値範囲の設定（ AjcIvSetRange ） ... 210
8.5.13. 数値最小単位値の設定（ AjcIvSetSldUnit ） ... 210
8.5.14. スライダ・ページサイズの設定（ AjcIvSetSldPage ） ... 210
8.5.15. スピンボタンによる増減値設定（ AjcIvSetRealSpnStep ） ... 210
8.5.16. 数値の精度設定（ AjcIvSetPrecision ） ... 211
8.5.17. ツールチップの設定／取得（ AjcIvSetTipText ） ... 211
8.5.18. ツールチップ表示条件の設定／取得(AjcIv{Set|Get}TipShowAlways) ... 211
8.5.19. デフォルトツールチップテキストの許可／禁止（ AjcIvEnaDefTipText） ... 211
8.5.20. 数値編集状態の取得（AjcIvGetEditState） ... 212
8.5.21. 設定情報の永続化（AjcIvLoadPermInfo / AjcIvSavePermInfo） ... 212
8.5.22. ベント通知用コールバック関数の設定(AjcIvSetEvtCallBack) ... 212
8.6. 通知情報の取得ＡＰＩ ... 213
8.6.1. 右クリック情報の取得（AjcIvGetNtcRClk） ... 213
8.7. 設定／取得メッセージ ... 214
8.8. 通知メッセージ ... 215
8.8.1. 整数モードでの数値設定通知（ AJCIVN_INTVALUE ） ... 215
8.8.2. 実数モードでの数値設定通知（ AJCIVN_REALVALUE ） ... 215
8.8.3. 右クリック通知（AJCIVN_RCLICK） ... 215
8.9. サンプルプログラム ... 216
8.10. SW_INPVAL（ウインドの透明度設定） ... 216
9. ログファイル出力コントロール（AjcCtrlLogFile クラス） ... 218
9.1. コントロールのスタイル ... 219
9.2. プロパティの永続化 ... 219
9.3. サポートＡＰＩ ... 220
9.3.1. ログファイル出力先ディレクトリパスの設定／取得(AjcLgf{Set/Get}Dir) ... 221
9.3.2. テキスト文字コード設定／取得(AjcLgf{Set/Get}TextEncode) ... 221

9.3.3. ログ出力開始(AjcLgfStart) 222
9.3.4. ログ出力停止(AjcLgfStop) 222
9.3.5. ログ出力可能状態取得(AjcLgfIsActive) 222
9.3.6. ログファイルへのテキスト出力(AjcLgfPutText) 222
9.3.7. ログファイルへの書式テキスト出力(AjcLgfPrintF) 223
9.3.8. タイムスタンプ形式の設定／取得(AjcLgf{Set/Get}TimeStampFormat) 223
9.3.9. ログファイル名の接頭語設定／取得 (AjcLgf{Set/Get}LogFileNamePrefix) 223
9.3.10. ログファイル名の拡張子設定／取得 (AjcLgf{Set/Get}LogFileExtention) 224
9.3.11. プロファイルからプロパティ値読み出し (AjcLgfLoadProp) 224
9.3.12. プロファイルへプロパティ値を記録 (AjcLgfSaveProp) 224
9.3.13. ベント通知用コールバック関数の設定(AjcLgfSetEvtCallBack) 225
9.4. コントロールの通知メッセージ 226
9.4.1. ログ出力開始通知 (AJCLGFN_START) 226
9.4.2. ログ出力停止通知 (AJCLGFN_STOP) 226
9.4.3. テキスト文字コード設定変更通知 (AJCLGFN_ENCODE) 226
9.5. サンプルプログラム 227
9.5.1. SW_LogFile (ログファイル出力) 227
10. 棒グラフ／折れ線グラフ表示コントロール (AjcCtrlBarGraph クラス) 230
10.1. 機能概要 230
10.1.1. ポップアップメニュー 230
10.1.2. レンジ／ベース値設定 231
10.1.3. フィルタ機能 231
10.1.4. 折れ線グラフ 231
10.1.5. ファイルやディレクトリのドラッグ＆ドロップ 232
10.2. コントロールのスタイル 232
10.3. プロパティ構造体 233
10.4. キャプション文字列によるプロパティの設定 234
10.5. テキストの取得と設定 234
10.6. プロパティの永続化 235
10.7. サポートＡＰＩ 236
10.7.1. データ投与(AjcBarPut{Real/Int}Data) 237
10.7.2. 棒タイトル名設定(AjcBarSetBarTtl) 237
10.7.3. 縦軸単位名設定(AjcBarSetVUnit) 237
10.7.4. データクリアー(AjcBarPurge) 237
10.7.5. 外枠の表示色と表示／非表示(AjcBarShowBorder) 238
10.7.6. フィルタ表示／非表示(AjcBarShowFilter) 238
10.7.7. プロパティ設定／取得(AjcBar{Set/Get}Prop) 238
10.7.8. レンジ設定／取得(AjcBar{Set/Get}RealRange) 238
10.7.9. ベース値設定／取得(AjcBar{Set/Get}{Real/Int}Base) 239
10.7.10. バッファサイズ設定(AjcBarSetBufSize) 239
10.7.11. データ項目数設定(AjcBarSetItemNumber) 239
10.7.12. スケール値表示域の幅設定(AjcBarSetScaleWidth) 239
10.7.13. 棒の幅設定(AjcBarSetBarWidth) 240
10.7.14. 棒表示域の幅設定(AjcBarSetItemWidth) 240
10.7.15. 棒タイトルの最大行数設定(AjcBarSetTtlLines) 240
10.7.16. ビットマップデータ取得 (AjcTchGetBitmap) 240
10.7.17. プロファイルからプロパティ値読み出し (AjcBarLoadProp / AjcBarLoadPropEx) 241
10.7.18. 右クリック通知設定 (AjcBarSetNtcRClk) 241
10.7.19. ポップアップメニューの許可／禁止 (AjcTchEnablePopupMenu) 242
10.7.20. ツールチップの設定／取得 (AjcBar{Set|Get}TipText) 242
10.7.21. ツールチップ表示条件の設定／取得(AjcBar{Set|Get}TipShowAlways) 242
10.7.22. フィルタ・チェックボックス・ツールチップの設定／取得 (AjcBar{Set/Get}ChkBoxTipText) 242
10.7.23. フィルタチェックボックスのツールチップ表示条件 設定／取得(AjcBar{Set|Get}ChkBoxTipShowAlways) 243
10.7.24. スクロール位置の設定／取得 (AjcBar{Set/Get}ScrollPos) 243
10.7.25. フィルタの設定／取得 (AjcBar{Set/Get}Filter) 243
10.7.26. 横線の属性設定 (AjcBarSetHLineAtt) 244
10.7.27. 横線の描画位置設定 (AjcBarSetHLinePos) 244
10.7.28. 横線描画の許可／禁止(AjcBarEnableHLine) 244
10.7.29. 文字サイズの取得 (AjcBarGetCharSize) 245
10.7.30. ドロップされたファイル名取得(AjcBarGetDroppedFile) 245

10.7.31. ドロップされたディレクトリ名取得(AjcBarGetDroppedDir[Ex]) 245
10.7.32. タイトル文字列の設定 (AjcBarSetTitleText) 246
10.7.33. テキスト描画フォント設定(AjcBarSetTextFont) 246
10.7.34. テキスト描画(AjcBarTextOut) 246
10.7.35. 書式テキスト描画(AjcBarPrintF) 247
10.7.36. 描画テキスト取得(AjcBarGetText) 247
10.7.37. 描画テキスト消去(AjcBarClear[All]Text) 247
10.7.38. 全てのデータ消去(AjcBarClear) 247
10.7.39. ベント通知用コールバック関数の設定(AjcBarSetEvtCallBack) 248
10.8. 通知情報の取得ＡＰＩ 249
10.8.1. グラフレンジ情報の取得 (AjcBarGetNtcRng) 249
10.8.2. ドロップしたファイル数の取得 (AjcBarGetNtcFiles) 249
10.8.3. ドロップしたディレクトリ数の取得 (AjcBarGetNtcDirs) 249
10.8.4. 右クリック情報の取得 (AjcBarGetNtcRClk) 249
10.9. コントロールの通知メッセージ 250
10.9.1. レンジ通知 (AJCBARN_RANGE) 250
10.9.2. スクロール位置通知 (AJCBARN_SCRPOS) 250
10.9.3. ダブルクリック通知 (AJCBARN_DBLCLK) 250
10.9.4. ファイルドロップ通知 (AJCBARN_DROPFILE) 251
10.9.5. ディレクトリドロップ通知 (AJCBARN_DROPDIR) 251
10.9.6. 右クリック通知 (AJCBARN_RCLICK) 251
10.10. サンプルプログラム 252
10.10.1. SW_BarGraph (棒グラフの表示サンプル) 252
11. 拡張リストボックス・コントロール (AjcCtrlListBox クラス) 256
11.1. コントロールのスタイル 257
11.2. 機能別スタイル値 258
11.3. リストボックスへ設定するスタイル 258
11.4. リストボックス項目の永続化 259
11.5. サポートＡＰＩ 259
11.5.1. 全リストボックス項目データの配列取得(AjcLbxGetAllItems) 260
11.5.2. 取得した全リストボックス項目のメモリ解放(AjcLbxRelAllItems) 260
11.5.3. 選択されているリストボックス項目データの配列取得(AjcLbxGetSelectedItems) 260
11.5.4. 取得した選択済リストボックス項目群のメモリ解放(AjcLbxRelSelectedItems) 260
11.5.5. プロファイルから全リストボックス項目の読出し／書込み(AjcLbx{Load/Save}Items) 261
11.5.6. プロファイルから全リストボックス項目の読出し／書込み(AjcLbx{Load/Save}PermInfo[Ex]) 261
11.5.7. ツールチップの設定／取得(AjcLbx{Set/Get}TipText) 261
11.5.8. ツールチップ表示条件の設定／取得(AjcLbx{Set|Get}TipShowAlways) 262
11.5.9. リストボックス項目の追加(AjcLbxAddString) 262
11.5.10. リストボックス項目の挿入(AjcLbxInsertString) 262
11.5.11. リストボックス項目の削除(AjcLbxDeleteString) 263
11.5.12. リストボックス項目の検索(AjcLbxFindString) 263
11.5.13. リストボックスの項目数取得(AjcLbxGetCount) 263
11.5.14. 選択されているリストボックス項目のインデクス取得(AjcLbxGetCurSel) 263
11.5.15. リストボックス項目の選択状態取得(AjcLbxGetSel) 263
11.5.16. 選択されているリストボックス項目の個数取得(AjcLbxGetSelCount) 264
11.5.17. リストボックス項目の取得(AjcLbxGetText) 264
11.5.18. リストボックス項目の文字列長取得(AjcLbxGetTextLen) 264
11.5.19. 全リストボックス項目の消去(AjcLbxResetContent) 264
11.5.20. リストボックス項目の選択 (文字列指定) (AjcLbxSelectString) 264
11.5.21. リストボックス項目数の設定(AjcLbxSetCount) 265
11.5.22. リストボックス項目の選択 (インデクス指定) (AjcLbxSetCurSel) 265
11.5.23. リストボックス項目の選択／非選択状態設定(AjcLbxSetSel) 265
11.5.24. リストボックス項目に関連付けた数値の設定／取得(AjcLbx{Set/Get}ItemData) 265
11.5.25. 相対アドレス変換時のベースディレクトリパス設定(AjcLbxSetBasePath) 266
11.5.26. 右クリック通知設定 (AjcLbxSetNtcRClk) 266
11.5.27. ポップアップメニューの許可／禁止 (AjcLbxEnablePopupMenu) 266
11.5.28. ポップアップメニュー項目の有効化／無効化 (AjcVth{Sel/Exc}MenuItems) 267
11.5.29. ファイル選択時のフィルタ，デフォルト拡張子の設定 (AjcLbxSetFileFilter) 267
11.5.30. ドロップされたファイル名取得(AjcLbxGetDroppedFile) 268
11.5.31. ドロップされたディレクトリ名取得(AjcLbxGetDroppedDir[Ex]) 268

11.5.32. 削除された項目の文字列取得(AjcLbxGetRemovedItem) ... 268
11.5.33. リストボックス項目の高さ設定／取得（AjcLbx{Set/Get}Height) ... 269
11.5.34. リストボックスのハンドル取得（AjcLbxGetLstHandle) ... 269
11.5.35. リストボックスのコントロールＩＤ取得（AjcLbxGetLstHandle) ... 269
11.5.36. ベント通知用コールバック関数の設定(AjcLbxSetEvtCallBack) ... 270
11.6. 通知情報の取得ＡＰＩ ... 271
11.6.1. リストボックスへ設定するスタイル取得（AjcLbxGetListStyle） ... 271
11.6.2. ドロップしたファイル数の取得（AjcLbxGetNtcFiles） ... 271
11.6.3. ドロップしたディレクトリ数の取得（AjcLbxGetNtcDirs） ... 271
11.6.4. 削除した項目数の取得（AjcLbxGetNtcRemovedItems） ... 271
11.6.5. 右クリック情報の取得（AjcLbxGetNtcRClk） ... 271
11.7. 通知メッセージ ... 272
11.7.1. リストボックスの設定スタイル問い合わせ（AJCLBXN_LISTSTYLE） ... 272
11.7.2. ダブルクリック通知（AJCLBXN_DBLCLK） ... 272
11.7.3. 選択キャンセル通知（AJCLBXN_SELCANCEL） ... 272
11.7.4. 選択変更通知（AJCLBXN_SELCHANGE） ... 273
11.7.5. フォーカス取得通知（AJCLBXN_SETFOCUS） ... 273
11.7.6. フォーカス喪失通知（AJCLBXN_KILLFOCUS） ... 273
11.7.7. 右クリック通知（AJCLBXN_RCLICK） ... 273
11.7.8. 項目削除通知（AJCLBXN_REMOVED） ... 273
11.7.9. ディレクトリドロップ通知（AJCLBXN_DROPDIR） ... 274
11.7.10. ファイルドロップ通知（AJCLBXN_DROPFILE） ... 274
11.7.11. メモリ不足通知（AJCLBXN_ERRSPACE） ... 274
11.8. サンプルプログラム ... 275
11.8.1. SW_ListBox（リストボックス操作サンプル） ... 275
12. 通信コントロールで共通な内容の説明 ... 281
12.1. チャンクデータ ... 281
12.2. テキストデータ ... 281
12.3. データ区切りの認識 ... 281
12.4. テキストデータのマルチバイト文字分断抑止 ... 282
12.5. 受信通知イベント ... 282
12.5.1. テキスト チャンク・データ受信通知（AJC*XXX*_EV_RXCHUNK） ... 283
12.5.2. バイナリ チャンク・データ受信通知（AJC*XXX*_EV_RXCHUNK） ... 283
12.5.3. テキスト受信通知（AJC*XXX*_EV_RXTEXT） ... 283
12.5.4. ＥＳＣシーケンス受信通知（AJC*XXX*_EV_RXESC） ... 284
12.5.5. パケットデータ受信通知（AJC*XXX*_EV_RXPKT） ... 284
12.5.6. パケット外テキスト受信通知（AJCXXX_EV_RXNOPKT） ... 285
12.5.7. バイトペアによるワード(14Bit)データ受信通知（AJC*XXX*_EV_RXWORD14） ... 285
12.5.8. パケットフレーム外での透過制御バイト（DLE) ... 285
12.6. 送受信動作 ... 286
12.7. 送受信テキストデータの文字コード ... 286
13. シリアル通信 ... 287
13.1. イベントの通知方法 ... 287
13.1.1. ユーザが その場でイベントの発生を待ち受ける場合 ... 287
13.1.2. ユーザのウインドへ、ウインドメッセージとして通知する場合 ... 288
13.2. ＤＣＢとタイムアウト情報 ... 289
13.3. イベント一覧 ... 290
13.4. COM ポート通信 ... 291
13.5. メールスロット通信 ... 291
13.6. ソケット（TCP/IP クライアント）通信 ... 292
13.7. ソケット（TCP/IP）サーバ通信 ... 292
13.8. 各通信リソースを切り替えて通信 ... 293
13.9. サポートＡＰＩ ... 294
13.9.1. インスタンス生成（AjcScpCreate） ... 296
13.9.2. インスタンス消去（AjcScpDelete） ... 297
13.9.3. 実行モード設定（AjcScpSetMode） ... 297
13.9.4. COM ポートオープン［ポートパラメタ指定］(AjcScpOpen) ... 298
13.9.5. COM ポートオープン［DCB で詳細指定］(AjcScpOpenEx) ... 298
13.9.6. メールスロット オープン（AjcScpOpenSlot） ... 298
13.9.7. ソケット（TCP/IP クライアント）通信 オープン（AjcScpOpenSock） ... 299

13.9.8. ソケット（TCP/IP サーバ）通信 オープン（AjcScpOpenServ） 299
13.9.9. ポートオープン［選択済通信リソース］(AjcScpOpenDefault) 300
13.9.10. ポートートオープン［通信リソース選択］(AjcScpOpenSelect) 300
13.9.11. ポートクローズ（AjcScpClose） 300
13.9.12. サーバ停止（AjcScpServStopt） 301
13.9.13. 通信ポート種別設定／取得（AjcScp{Set/Get}SelectedPort） 301
13.9.14. ＣＯＭポートオープン状態取得（AjcScpIsOpened） 301
13.9.15. 1 文字送信（AjcScpSendChar） 302
13.9.16. バイトペアによるワード（14Bit）データ送信（Low byte first）(AjcScpSendWord14LF) 302
13.9.17. バイトペアによるワード（14Bit）データ送信（High byte first）(AjcScpSendWord14HF) 302
13.9.18. １４ビットデータのバイト順設定／取得（AjcScp{Set/Get}ByteSeqRxWord14） 303
13.9.19. テキストデータ送信（AjcScpSendText） 303
13.9.20. バイナリデータ送信（AjcScpSendBinData） 303
13.9.21. パケットデータ送信（AjcScpSendPacket） 304
13.9.22. ブレーク信号送出／停止（AjcScpSendBreak） 304
13.9.23. イベントマスク設定／取得（AjcScp{Set/Get}EvtMask） 304
13.9.24. ＤＴＲ信号設定（AjcScpSetDTR） 304
13.9.25. ＲＴＳ信号設定（AjcScpSetRTS） 305
13.9.26. 入力信号（DSR, CTS, RING, RLSD）状態取得（AjcScpGetSigState） 305
13.9.27. 送信待ちデータバイト数取得（AjcScpGetTxBytes） 305
13.9.28. 全受信済データ破棄（AjcScpPurgeRecvData） 305
13.9.29. 全送信待ちデータ破棄（AjcScpPurgeSendData） 305
13.9.30. 全受信済データと全送信待ちデータ破棄（AjcScpPurgeAllData） 306
13.9.31. イベント発生待ち（AjcScpWaitEvent） 306
13.9.32. クライアントのＩＰアドレス文字列取得（AjcScpvGetIpAddrStr） 306
13.9.33. イベントデータ取得（AjcScpGetEventData） 307
13.9.34. イベントデータ開放（AjcScpRelEventData） 307
13.9.35. パケットフレームを認識する為の制御コード設定／取得（AjcScp{Set/Get}PktCtrlCode） 307
13.9.36. パケットフレーム受信タイムアウト値設定／取得（AjcScp{Set/Get}PktTimeout） 308
13.9.37. ダイアログによる通信パラメタ設定（AjcScpDlgParamEasy） 309
13.9.38. ダイアログによる通信パラメタ設定［詳細］(AjcScpDlgParamDetail) 310
13.9.39. ダイアログ・通信リソースの選択を無効化（AjcScp{Set/Get}GrayRbt） 311
13.9.40. ダイアログの各項目を無効化（AjcScp{Set/Get}GrayItem） 312
13.9.41. ＣＯＭポート情報（ＤＣＢとタイムアウト情報）設定／取得（AjcScp{Set/Get}Param） 314
13.9.42. メールスロット情報設定／取得（AjcScp{Set/Get}SlotParam） 314
13.9.43. ソケット情報設定／取得（AjcScp{Set/Get}SockParam） 315
13.9.44. サーバ情報設定／取得（AjcScp{Set/Get}ServParam） 315
13.9.45. 利用可能なシリアルポート番号の列挙（AjcScpEnumSerialPorts） 316
13.9.46. 通信ポートのパス名を取得（AjcScpGetPortPathName） 316
13.9.47. ポート名称取得（AjcScpGetPortName） 316
13.9.48. ＣＯＭポートのデバイス名称取得（AjcScpGetPortDevName） 317
13.9.49. チャンクデータの受信モード設定／取得（AjcScp{Set/Get}ChunkMode） 317
13.9.50. 受信文字コード種別設定／取得（AjcScp{Set/Get}RxTextCode） 317
13.9.51. 送信文字コード種別設定／取得（AjcScp{Set/Get}TxTextCode） 318
13.9.52. 実際の送受信文字コード種別取得（AjcScpGetActual{Rx/Tx}TextCode） 318
13.9.53. 自メールスロット生成（AjcScpCreateMySlo） 318
13.9.54. 自メールスロット消去（AjcScpDeleteMySlot） 319
13.9.55. 自メールスロットの生成状態取得（AjcScpMySlotIsCreated） 319
13.9.56. 自メールスロットのパス名取得（AjcScpGetMySlotPathName） 319
13.9.57. 自コンピュータ名取得（AjcScpGetMyComputerName） 319
13.9.58. メールスロット名情報設定／取得（AjcScp{Set/Get}MailSlotNames） 320
13.9.59. メールスロット送信制限速度の設定／取得（AjcScp{Set/Get}TxSpeedLimit） 320
13.9.60. 送信停止の認識時間設定（AjcScpSetRecognizeTxStopTime） 320
13.9.61. 受信停止の認識時間設定（AjcScpSetRecognizeRxStopTime） 321
13.10. 基本サンプルコード 322
13.10.1. SW_ScpComC（COM ポート通信, コンソールアプリ） 322
13.10.2. SW_ScpMailSLotC（メールスロット通信, コンソールアプリ） 323
13.10.3. SW_ScpClientC（ソケット(クライアント)通信, コンソールアプリ） 325
13.10.4. SW_ScpServerC（ソケット(サーバ)通信, コンソールアプリ） 327
13.10.5. SW_ScpAny（通信リソースを選択して通信） 329

13.11. サンプルプログラム 332
13.11.1. SW_SerialComPort1（シリアル通信による送受信） 332
13.11.2. SW_SerialComPort2（エコーバック） 341
13.11.3. SW_SerialComPort2C（エコーバック – コンソールアプリ） 344
13.11.4. SW_SerialComPort3（ループバックによるフォルダ構造コピー） 347
13.11.5. SW_SerialComPort4（メールスロット通信） 354
13.11.6. SW_SerialComPort5（テキストとバイナリパケットの多重通信） 357
14. ソケット(TCP/IP)サーバ機能 367
14.1. イベントの通知方法 367
14.2. イベント一覧 370
14.3. サポートＡＰＩ 371
14.3.1. インスタンス生成（AjcSsvCreate） 372
14.3.2. インスタンス消去（AjcSsvDelete） 372
14.3.3. サーバ起動（AjcSsvStart） 372
14.3.4. サーバ停止（AjcSsvStop） 372
14.3.5. サーバ起動状態取得（AjcSsvIsStarted） 373
14.3.6. オプションの設定／取得（AjcSsv{Set/Get}Opt） 373
14.3.7. チャンクデータの通知モード設定／取得（AjcSsv{Set/Get}ChunkMode） 373
14.3.8. イベントマスク設定／取得（AjcSsv{Set/Get}EvtMask） 373
14.3.9. 受信文字コード種別設定／取得（AjcSsv{Set/Get}RxTextCode） 374
14.3.10. 送信文字コード種別設定／取得（AjcSsv{Set/Get}TxTextCode） 374
14.3.11. イベント発生待ち（AjcSsvWaitEvent） 375
14.3.12. イベントデータ取得（AjcSsvGetEventData） 375
14.3.13. イベントデータ開放（AjcSsvRelEventData） 375
14.3.14. イベントが空になるまで待つ（AjcSsvWaitEmptyEvent） 376
14.3.15. ライアント・ハンドル取得（AjcSsvGetEventData） 376
14.3.16. パケットフレーム認識・制御コード設定／取得（AjcSsv{Set/Get}PktCtrlCode） 376
14.3.17. パケット受信タイムアウト値 設定／取得（AjcSsv{Set/Get}PktTimeout） 377
14.3.18. 接続中のクライアント数取得（AjcSsvGetClientCount） 377
14.3.19. クライアントの通算接続回数取得（AjcSsvGetConnectCount） 377
14.3.20. クライアントの接続状態取得（AjcSsvIsClientConnected） 377
14.3.21. 接続中の全クライアント取得（AjcSsvEnumClients） 378
14.3.22. クライアントを切断する（AjcSsvDisconnect） 379
14.3.23. クライアントへ１文字送信（AjcSsvSendChar） 379
14.3.24. クライアントへテキストデータ送信（AjcSsvSendText） 379
14.3.25. クライアントへバイナリデータ送信（AjcSsvSendBinData） 379
14.3.26. クライアントへパケットデータ送信（AjcSsvSendPacket） 380
14.3.27. 全受信済データ破棄（AjcSsvPurgeRecvData） 380
14.3.28. 全送信待ちデータ破棄（AjcSsvPurgeSendData） 380
14.3.29. 全受信済データと全送信待ちデータ破棄（AjcSsvPurgeAllData） 380
14.3.30. クライアントにデータを関連付ける（AjcSsvSetClientData） 381
14.3.31. クライアントに関連付けられたデータを取得する（Ajcssvgetclientdata） 381
14.3.32. クライアント接続順序番号取得（AjcSsvGetSeqNo） 381
14.3.33. クライアントに割り当てられたインデクス取得（AjcSsvGetIndex） 381
14.3.34. クライアントのＩＰアドレス文字列取得（AjcSsvGetIpAddrStr） 382
14.3.35. クライアント接続順序番号取得（AjcSsvGetSeqNo） 382
14.3.36. クライアントスレッドのスレッドＩＤ取得（AjcSsvGetThreadId） 382
14.3.37. クライアントのソケット記述子取得（AjcSsvGetSocket） 382
14.3.38. 実際の送受信文字コード種別取得（AjcSsvGetActual{Rx/Tx}TextCode） 383
14.4. サンプルプログラム 384
14.4.1. SW_SockServer1（送受信テスト） 384
14.4.2. SW_SockServer2（エコーサーバ） 393
14.4.3. SW_SockServer2C（エコーサーバ，コンソールアプリ） 396
15. ソケット(TCP/IP)クライアント機能 399
15.1. 機能概要 399
15.2. イベントの通知方法 399
15.3. イベント一覧 402
15.4. サポートＡＰＩ 403
15.4.1. インスタンス生成（AjcSctCreate） 404
15.4.2. インスタンス消去（AjcSctDelete） 404

15.4.3. 回線接続（AjcSctConnect） ... 404
15.4.4. 回線切断（AjcSctDisconnect） ... 404
15.4.5. 回線状態取得（AjcSctGetState） ... 405
15.4.6. チャンクデータの通知モード設定／取得（AjcSct{Set/Get}ChunkMode） ... 405
15.4.7. イベントマスク設定／取得（AjcSct{Set/Get}EvtMask） ... 405
15.4.8. 受信文字コード種別設定／取得（AjcSct{Set/Get}RxTextCode） ... 406
15.4.9. 送信文字コード種別設定／取得（AjcSct{Set/Get}TxTextCode） ... 406
15.4.10. 実際の送受信文字コード種別取得（AjcSctGetActual{Rx/Tx}TextCode） ... 406
15.4.11. イベント発生待ち（AjcSctWaitEvent） ... 407
15.4.12. イベントデータ取得（AjcSctGetEventData） ... 407
15.4.13. イベントデータ開放（AjcSctRelEventData） ... 407
15.4.14. イベントが空になるまで待つ（AjcSctWaitEmptyEvent） ... 408
15.4.15. パケットフレーム認識・制御コード設定／取得（AjcSct{Set/Get}PktCtrlCode） ... 408
15.4.16. パケット受信タイムアウト値 設定／取得（AjcSct{Set/Get}PktTimeout） ... 408
15.4.17. 1 文字送信（AjcSctSendChar） ... 408
15.4.18. テキストデータ送信（AjcSctSendText） ... 409
15.4.19. バイナリデータ送信（AjcSctSendBinData） ... 409
15.4.20. パケットデータ送信（AjcSctSendPacket） ... 409
15.4.21. 全受信済データ破棄（AjcSctPurgeRecvData） ... 410
15.4.22. 全送信待ちデータ破棄（AjcSctPurgeSendData） ... 410
15.4.23. 全受信済データと全送信待ちデータ破棄（AjcSctPurgeAllData） ... 410
15.5. サンプルプログラム ... 411
15.5.1. SW_SockClient1（送受信テスト） ... 411
15.5.2. SW_SockClient2（エコーバック） ... 418
15.5.3. SW_SockClient2C（エコーバック，コンソールアプリ） ... 421
15.5.4. SW_SockClient3（ループバックによるフォルダ構造コピー） ... 423
16. ダイアログボックス、コントロール群とメニューのカラー設定 ... 429
16.1. サポートＡＰＩ ... 437
16.1.1. セットアップ（AjcDgcSetup[Ex]） ... 438
16.1.2. 背景色とテキスト色の設定(AjcDgcSetColors) ... 438
16.1.3. ビットマップ透明色設定／取得（AjcDgc{Get/Set}BmpTransColor) ... 438
16.1.4. カラー表示禁止（AjcDgcDisablwColorDisplay) ... 439
16.1.5. ダイアログボックスによる背景色とテキスト色設定（AjcDgcSetColorsDlg) ... 439
16.1.6. ボタンのスタイル取得(AjcDgcGetSbcButtonStyle) ... 439
16.1.7. ボタンのスタイル設定(AjcDgcSetSbcButtonStyle) ... 440
16.1.8. ポップアップメニュー(AjcDgcTrackPopupMenu[Ex]) ... 440
16.1.9. フォーカス表示フラグ取得／設定(AjcDgc{Get/Set}ShowFocus) ... 440
16.1.10. メニューフォント取得／設定(AjcDgc{Get/Set}MenuFont) ... 441
16.1.11. 描画色の詳細情報取得／設定(AjcDgc{Get/Set}ColorInfo) ... 441
16.1.12. オプションの設定／取得（AjcDgc{Get/Set}Opt) ... 443
16.1.13. ウインドカラー表示の初期化（AjcDgcInitColorDisplay [Ex]) ... 443
16.2. サンプルプログラム ... 444
16.2.1. SW_DlgColor（ダイアログ，コントロールやメニューの描画色を設定） ... 444
16.2.2. SW_DlgColor2（背景色やテキスト色の動的設定と永続化） ... 453
17. ダイアログボックス項目／ウインド項目のアクセス ... 455
17.1. コントロールへ数値／文字列の設定や取得 ... 455
17.2. グループボックス内のコントロールを 一括アクセス ... 455
17.3. コントロールの設定内容を永続化 ... 455
17.4. サポートＡＰＩ ... 456
17.4.1. ダイアログボックス／ウインド項目の取得（AjcGetDlgItem･･･/ AjcGetCtrl･･･) ... 457
17.4.2. ダイアログボックス項目の設定（AjcSetDlgItem･･･／AjcSepDlgItem･･･／AjcSetCtrl･･･／AjcSepCtrl･･･) ... 459
17.4.3. EDIT コントロールにフォルダやファイルをドロップ可能にする(AjcEnable{DlgItem/Ctrl}ToDrop) ... 463
17.4.4. コントロールの位置／サイズ／**矩形取得**（Ajc{Get/Set}{DlgItem/Ctrl}{Pos/Size/Rect} ... 464
17.4.5. ダイアログ／ウインド項目の有効化／無効化（AjcEnableDlgItem／AjcEnableCtrl) ... 465
17.4.6. ダイアログ／ウインド項目の表示／非表示（AjcShowDlgItem／AjcShowCtrl) ... 465
17.4.7. ダイアログ／ウインド項目の表示／非表示と有効化／無効化（AjcShowAndEnableDlgItem／AjcShowAndEnableCtrl) 465
17.4.8. グループボックス内の全コントロール有効化／無効化（AjcEnableDlgGroup／AjcEnableGroup) ... 466
17.4.9. ウインド内の全コントロール有効化／無効化（AjcEnableCtrlsInWnd) ... 466

17.4.10. グループボックス内の全コントロール表示／非表示［１］(AjcShowDlgGroup／AjcShowGroup) 466
17.4.11. グループボックス内の全コントロール表示／非表示［２］(AjcShowDlgGroupEx／AjcShowGroupEx) 467
17.4.12. グループボックス内の全コントロールの移動(AjcMoveDlgGroupToLoc/Ctl/Org) 467
17.4.13. グループボックス内の全コントロール列挙(AjcDlgItemEnumInGroup／AjcCtrlEnumInGroup) 468
17.4.14. ウインド／グループ内全コントロールの設定内容を永続化(Ajc{Load/Save}{All/Grp}ControlSettings) 468
17.4.15. 各コントロールの永続化属性設定(Ajc{DlgItem/Ctrl}SetPermAtt) 472
17.4.16. グループボックス内全コントロールの永続化属性設定(Ajc{DlgItem/Ctrl}SetPermGrp) 472
17.4.17. 全コントロールの永続化属性解除(AjcDelAllCtrlPermAtt) 473
17.4.18. テキストボックスの永続化(Ajc{DlgItem/Ctrl}{Load/Save}TextBox[Ex]) 473
17.4.19. チェックボックス／ラジオボタンの永続化(Ajc{DlgItem/Ctrl}{Load/Save}ChkBox) 473
17.4.20. コンボボックスの永続化(Ajc{DlgItem/Ctrl}{Load/Save}ComboBox) 474
17.4.21. リストボックスの永続化(Ajc{DlgItem/Ctrl}{Load/Save}ListBox) 475
17.4.22. SW_DlgItem1（ダイアログボックス内の全コントロールの永続化） 476
17.4.23. SW_DlgItem2（ウインド全体／グループボックス単位のコントロールの永続化） 486
18. ウインドサポートＡＰＩ 490
18.1. サポートＡＰＩ 490
18.1.1. プロファイルからウインド位置を読み出す（AjcLoadWndPos) 491
18.1.2. プロファイルへウインド位置を記録する（AjcSaveWndPos) 491
18.1.3. プロファイルからウインド位置とサイズを読み出す（AjcLoadWndRect) 491
18.1.4. プロファイルへウインド位置とサイズを記録する（AjcSaveWndRect) 492
18.1.5. ウインドの一部が隠れている場合 ウインドをモニタ内へ移動する(AjcMoveWndIntoMonitor) 492
18.1.6. ウインド内矩形の一部が隠れている場合 ウインドをモニタ内へ移動する(AjcMoveWndRectIntoMonitor) 492
18.1.7. ウインド全体が隠れている場合 ウインドを原点へ移動する(AjcMoveWndRectIntoMonitor) 492
18.1.8. 全モニタで矩形が見えているかチェックする(AjcIsSeeEntireInMonitors) 493
18.1.9. ウィンドを画面の中央へ移動する（AjcMoveWindowToCenter・・・) 493
18.1.10. 現在フォーカスされているウインドを取得(AjcGetFocusWindow) 493
18.1.11. 指定ウインドをフォーカスする(AjcSetFocusWindow) 494
18.1.12. 通常表示時のウインド矩形情報取得(AjcGetWindowNormalRect) 494
18.1.13. スレッドの全トップレベルウインドへメッセージ送信（AjcSendMessageToThreadWindows) 494
18.1.14. スレッドの全トップレベルウインドと指定された２つのウインドへメッセージ送信
(AjcSendMessageToThreadWindowsEx) 494
18.1.15. スレッドの全トップレベルウインドを許可／禁止(AjcEnableThreadWindows) 495
18.1.16. Windows イベント処理(AjcDoEvent) 495
18.1.17. Windows イベント処理(AjcDoEvent) ・・ ダイアログメッセージ処理付き 495
18.1.18. ウインドハンドルから実行プログラムファイルのパス名を取得する（AjcGetExePathNam) 495
18.1.19. デフォルトウインドプロシージャの実行（AjcDefWindowProc) 496
18.1.20. ウインドプロシージャの呼び出し（AjcCallWindowProc) 496
18.1.21. ウインドのＬＯＮＧ情報設定（AjcSetWindowLongPtr) 496
18.1.22. ウインドのＬＯＮＧ情報取得（AjcGetWindowLongPtr) 496
18.1.23. ウインドの背景色取得（AjcGetWndBkColor) 497
18.1.24. ウインドの背景色のブラシハンドル取得（AjcGetWndBkBurush) 497
19. 拡張コンボボックス 498
19.1. サポートＡＰＩ 498
19.1.1. コンボボックスの拡張（AjcSbcComboBox) 499
19.1.2. コンボボックス項目のテキスト比較方法設定（AjcSbcSetCompExact) 500
19.1.3. プロファイルからコンボボックスのリスト項目群を読み出す（AjcSbcLoadItems) 500
19.1.4. プロファイルへコンボボックスのリスト項目群を記録する（AjcSbcSaveItems) 500
19.1.5. コンボボックス下のエディットコントロールのハンドルを取得する（AjcSbcGetEditCtrlInComboBox) 501
19.1.6. コンボボックス現表示項目を最新項目に設定(AjcSbcSetMostNew) 501
19.1.7. コンボボックスのチップテキスト表示制御（AjcSbcTipCtrl) 501
19.1.8. コンボボックスのチップテキストをファイルの１行目とする(AjcSbcShowFirstLine) 502
19.1.9. 長テキスト・ツールチップ表示の許可／禁止(AjcSbcEnableLongTip) 502
19.1.10. 削除された項目の文字列取得(AjcSbcGetRemovedItem) 502
19.1.11. テキスト文字コード設定(AjcSbcSetTextEncode) 503
19.1.12. テキスト文字コード取得(AjcSbcGetTextEncode) 503
19.2. 通知メッセージ 504
19.2.1. 項目削除通知（AJCCBN_REMOVED） 504
19.2.2. ファイル／フォルダ ドロップ通知（AJCCBN_DROPPED） 504
19.2.3. サンプルプログラム 505
19.2.4. SW_ComboBox.c（拡張コンボボックスの操作サンプル） 505

20. ラジオボタンのグループ操作 509
20.1. サポートＡＰＩ 509
20.1.1. ラジオボタンのグループ操作化（AjcSbcRadioBtns) 509
20.1.2. グループ操作化したラジオボタンの状態取得（AjcSbcGetRbt) 510
20.1.3. グループ操作化したラジオボタンの状態設定（AjcSbcSetRbt) 510
20.1.4. ラジオボタンのハンドル取得（AjcSbcGetRbtHandle) 511
20.2. サンプルプログラム 512
20.2.1. SW_RadioBbutton（ラジオボタングループ化の操作サンプル） 512
21. 他プロセス内のメモリ操作 514
21.1. サポートＡＰＩ 514
21.1.1. 他プロセスにメモリ確保（AjcPmCreateByWnd / AjcPmCreateByPid) 514
21.1.2. 他プロセスにメモリ確保 したメモリを破棄(AjcPmDelete) 514
21.1.3. 他プロセスに確保したメモリへ書き込み(AjcPmWrite) 515
21.1.4. 他プロセスに確保したメモリの読み出し(AjcPmRead) 515
21.1.5. 他プロセスに確保したメモリ先頭からのロケーション位置のアドレス取得(AjcPmGetAddr) 515
21.1.6. 他プロセスのウインドへメッセージ送信(AjcPmSendMessage) 516
21.2. サンプルプログラム 517
21.2.1. SW_ProcessMem（エクスプローラプロセスからのメモリ読み出し） 517
21.2.2. SW_ProcessMemC（エクスプローラプロセスからのメモリ読み出し） 522
22. 拡張ツールチップ 524
22.1. サポートＡＰＩ 525
22.1.1. ツールチップ表示位置モードの設定(AjcTipTextSetTipPosMode) 526
22.1.2. ツールチップの関連付け(AjcTipTextAdd[F]) 527
22.1.3. ツールチップの関連付け(AjcTipTextAdd[F]Ex) 527
22.1.4. ツールチップ表示条件の設定／取得(AjcTipText{Set/Get}ShowAlways) 528
22.1.5. 全ツールチップ表示条件の設定／取得(AjcTipText{Set/Get}ShowForActive) 528
22.1.6. ツールチップの表示／非表示の 設定／取得(AjcTipText{Enable/GetEnableState}) 528
22.1.7. 全ツールチップの表示許可／禁止の 設定／取得(AjcTipText{EnableAll/GetEnableAllState}) 528
22.1.8. ツールチップの関連付け解除(AjcTipTextRemove) 529
22.1.9. 全てのツールチップの関連付け解除(AjcTipTextRemoveAll) 529
22.1.10. コントロール間移動猶予時間設定／取得(AjcTipText{Set|Get}DefementTime) 529
22.1.11. コールバックの設定(AjcTipTextSetCallBack) 530
22.1.12. コールバックの設定(AjcTipTextSetCallBackNoBuf) ・・作業用バッファ未使用 531
22.1.13. ツールチップの関連付け情報取得(AjcTipTextGetInfo) 532
22.1.14. ツールチップの表示(AjcTipTextShow) 532
22.1.15. ツールチップの詳細表示(AjcTipTextShowEx) 533
22.1.16. ツールチップウインドに閉じるマーク(X)の表示設定(AjcTipTextShowCloseMark) 533
22.1.17. ツールチップウインドの移動(AjcTipTextMove) 534
22.1.18. カーソルをツールチップ上へ移動(AjcTipTextMoveCursor) 534
22.1.19. ツールチップ非表示(AjcTipTextHide) 534
22.1.20. チップテキストウインド矩形情報取得(AjcTipTextGetRect) 534
22.1.21. パレット色の設定／取得(AjcTipText{Set|Get}Palette) 535
22.1.22. 文字列によるツールチップウインドの表示サイズ取得(AjcTipTextGetSize) 535
22.1.23. デフォルトの表示色設定／取得(AjcTipText{Set|Get}Def{Text/Border/Bk}Color) 535
22.1.24. デフォルト表示遅延時間の設定／取得(AjcTipText{Set|Get}DefMsDelay) 535
22.1.25. デフォルト表示時間の設定／取得(AjcTipText{Set|Get}DefMsShow) 536
22.1.26. デフォルトフォントの設定／取得(AjcTipText{Set|Get}DefFont) 536
22.1.27. マルチスレッドの許可／禁止(AjcTipTextEnableMultiThread) 536
22.1.28. サブクラス化の通知情報設定(AjcTipTextSetNtcSubclass) 537
22.1.29. サブクラス化の通知情報消去(AjcTipTextClrNtcSubclass) 537
22.1.30. チップコントロール対象のウインドを設定(AJCTIPCTRL_SETHWND) 537
22.1.31. チップコントロールの友達ウインドの設定(AJCTIP_SETFRIEND) 538
22.1.32. 描画イメージ（ビットマップ）の登録（AjcTipTextRegistBitmap) 538
22.1.33. 描画イメージ（ビットマップ）の登録（AjcTxoRegistBitmap) 538
22.1.34. 描画イメージの登録解除（AjcTxoUnregistImage) 539
22.1.35. SW_TipText（チップテキストの表示サンプル） 540
22.2. ツールチップテキストの表示サンプルプログラムです。サンプルプログラム 540
22.2.1. SW_TipTextM（子孫構造を持つコントロールのツールチップ制御） 545
23. コンソール入出力 551
23.1. サポートＡＰＩ 551

23.1.1. コンソールから１行入力（AjcConInput[Ex] / AjcConInputByNtc） 552
23.1.2. 標準出力／標準エラー のモード設定 （AjcSetStdMode)..... 554
23.1.3. 1 文字／文字列／書式文字列をコンソールへ出力（Ajc[Err]{PutC/PutS/PrintF}）..... 554
23.1.4. コンソール情報取得（AjcGetConsoleScreenBufferInfo)..... 555
23.1.5. コンソールカーソル位置設定(AjcSetConsoleCursor..... 555
23.1.6. コンソールカーソル位置取得(AjcGetConsoleCursor)..... 555
23.1.7. コマンドを実行し、出力をファイルへリダイレクト(AjcSystem)..... 555
23.2. サンプルプログラム..... 556
23.2.1. SW_ConInpC（コンソール入力） 556
24. プロファイル・アクセス..... 557
24.1. プロファイルのアクセス手順..... 558
24.2. INI ファイルの UNICODE 化..... 558
24.3. サポートＡＰＩ..... 559
24.3.1. プロファイル・アクセス先設定／取得（AjcSetProfileIsRegistory)..... 560
24.3.2. レジストリの記録方法設定／取得（Ajc{Set/Get}RegOptionVolatile)..... 560
24.3.3. INI ファイルパス設定／取得（Ajc{Set/Get}IniFilePath)..... 560
24.3.4. レジストリ・ルートパスの設定／取得（Ajc{Set/Get}RegRootPath)..... 561
24.3.5. レジストリ・ミドルパス設定／取得（Ajc{Set/Get}RegMidPath)..... 561
24.3.6. プロファイル・アクセス・パス設定（AjcSetProfilePath)..... 561
24.3.7. プロファイル・アクセス先とパス名の取得（AjcGetProfilePath)..... 562
24.3.8. バイナリデータ格納フォルダパス設定／取得（Ajc{Set/Get}BinDataDir)..... 562
24.3.9. プロファイル記録先情報の退避（AjcPushProfileStack)..... 563
24.3.10. プロファイル記録先情報の回復（AjcPopProfileStack)..... 563
24.3.11. プロファイル記録先情報リセット（AjcResetProfileStack)..... 563
24.3.12. プロファイル・読み出し（AjcGetProfile･･･)..... 564
24.3.13. プロファイル・書き込み（AjcPutProfile･･･)..... 565
24.3.14. プロファイル・セクションの削除（AjcDelProfileSect)..... 566
24.3.15. プロファイル・キーの削除（AjcDelProfileKey)..... 566
24.3.16. プロファイル・セクションの消去／クリーンアップ（Ajc{Remove/Cleanup}ProfileSect)..... 566
24.3.17. プロファイル・クローズ（AjcCloseProfile)..... 567
24.3.18. プロファイル・セクション名収集（AjcEnumProfileSect)..... 567
24.3.19. プロファイル・キー名収集（AjcEnumProfileKey)..... 568
24.3.20. プロファイル・アクセス・マクロ 569
24.4. サンプルプログラム..... 570
24.4.1. SW_Profile（プロファイルのアクセスサンプル） 570
25. ＩＮＩファイル・アクセス..... 580
25.1. サポートＡＰＩ..... 580
25.1.1. ＩＮＩファイル・読み出し（AjcGetIniFile･･･)..... 580
25.1.2. ＩＮＩファイル・書き込み（AjcPutIniFile･･･)..... 581
25.1.3. ＩＮＩファイル・セクションの削除（AjcDelIniSect)..... 582
25.1.4. ＩＮＩファイル・キーの削除（AjcDelIniKey)..... 582
25.1.5. ＩＮＩファイルセクションの消去／クリーンアップ（Ajc{Remove/Cleanup}IniFileSect)..... 582
25.1.6. ＩＮＩファイルを UNICODE 化（AjcIniFileToUnicode)..... 582
25.1.7. ＩＮＩファイル・セクション名収集（AjcEnumIniSect)..... 583
25.1.8. ＩＮＩファイル・キー収集（AjcEnumIniKey)..... 583
25.1.9. バイナリデータ格納フォルダパス設定／取得（Ajc{Set/Get}IniBinDir)..... 584
26. レジストリ・アクセス..... 585
26.1. レジストリタイプ..... 585
26.2. サポートＡＰＩ..... 586
26.2.1. レジストリ・読み出し（AjcGetRegFile･･･)..... 587
26.2.2. レジストリ・書き込み（AjcPutRegFile･･･)..... 588
26.2.3. レジストリ・セクションの削除（AjcDelRegSect)..... 589
26.2.4. レジストリ・データの削除（AjcDelRegKey)..... 589
26.2.5. レジストリ・セクションの消去／クリーンアップ(Ajc{Remove/Cleanup}RegSect)..... 589
26.2.6. レジストリ・トップキーの設定／取得（Ajc{Set/Get}RegTopKey)..... 590
26.2.7. レジストリ・キーハンドル・クローズ（AjcCloseRegFile)..... 590
26.2.8. レジストリ・セクション名収集（AjcEnumRegSect)..... 591
26.2.9. レジストリ・キー収集（AjcEnumRegKey)..... 591
26.2.10. レジストリ項目の読み出し（AjcRegGet)..... 592
26.2.11. レジストリ項目の書き込み（AjcRegPut)..... 592

26.2.12. レジストリ・サブキー削除(AjcRegDelSubKey)..... 592
26.2.13. レジストリ項目の削除(AjcRegDelValue)..... 593
26.2.14. レジストリに記録されている環境変数の読み出し (AjcReg[HkCu|HkLm]EnvGet)..... 593
26.2.15. レジストリへ環境変数とその内容を書き込む (AjcReg[HkCu|HkLm]EnvPut)..... 593
26.2.16. レジストリ中の環境変数を削除 (AjcReg{HkCu|HkLm}EnvDel)..... 594
26.2.17. レジストリの環境変数中の指定項目存在チェック (AjcReg{HkCu|HkLm}EnvSrhItem)..... 594
26.2.18. レジストリの環境変数へ項目を追加 (AjcReg{HkCu|HkLn}EnvAddItem)..... 594
26.2.19. レジストリの環境変数から項目を削除 (AjcReg{HkCu|HkLm}EnvDelItem)..... 595
26.2.20. レジストリのPATH環境変数中の指定パス存在チェック (AjcReg{HkCu|HkLm}EnvSrhPathItem)..... 595
26.2.21. レジストリのPATH環境変数へパス項目を追加 (AjcReg{HkCu|HkLm}AddPathItem)..... 595
26.2.22. レジストリのPATH環境変数からパス項目を削除 (AjcReg{HkCu{HkLm}DelPathItem)..... 596
26.2.23. 環境変数の有効化 (AjcRegEnableEnvironment)..... 596
26.2.24. レジストリパスの存在チェック (AjcRegIsPathExist)..... 596
26.2.25. レジストリキーの存在チェック (AjcRegIsKeyExist)..... 597
26.2.26. バイナリデータ格納フォルダパス設定／取得 (Ajc{Set/Get}RegBinDir)..... 597
27. ファイル名／フォルダ名の取得..... 598
27.1. サポートＡＰＩ..... 598
27.1.1. 保存ファイル名取得 (AjcGetSaveFile)..... 599
27.1.2. オープンファイル名取得 (AjcGetOpenFile)..... 599
27.1.3. 複数のオープンファイル名取得 (AjcGetOpenFile)..... 600
27.1.4. AjcGetOpenFilesで取得したファイル名群バッファの開放 (AjcReleaseOpenedFilesArray)..... 600
27.1.5. フォルダ名取得 (AjcGetFolderName)..... 601
27.1.6. 起動したプログラムのフォルダパス名／プログラム名取得 (AjcGetApp{Path | Name})..... 601
27.1.7. ウインドを所有するプロセスの実行ファイル・パス名取得 (AjcGetAppPathByHWnd)..... 601
28. ボリュームラベル操作..... 602
28.1. サポートＡＰＩ..... 602
28.1.1. ドライブのボリュームラベル名取得 (AjcGetVolumeLabel)..... 602
28.1.2. ボリュームラベル名からドライブ種別取得(AjcGetDriveByVolumeLabel)..... 602
28.1.3. ボリューム表現のパス名チェック(AjcIsVolExpPath)..... 603
28.1.4. ボリューム表現のパス名を通常のパス名に変更(AjcVolExpPathToNormalPath)..... 603
28.1.5. 通常のパス名をボリューム表現のパス名に変更(AjcNormalPathToVolExpPath)..... 603
29. ファイル検索..... 604
29.1. サポートＡＰＩ..... 604
29.1.1. サブディレクトリ下を含めたファイル／ディレクトリ検索 (AjcSearchFiles)..... 604
29.1.2. マイコンピュータ内のファイル／ディレクトリ検索 (AjcSearchMyComputer)..... 608
30. ファイル／フォルダ操作..... 610
30.1. サポートＡＰＩ..... 610
30.1.1. フォルダ構造のコピー(AjcCopyFolderStruct..... 611
30.1.2. ファイル群のコピー(AjcCopyFiles)..... 613
30.1.3. ファイルコピー(AjcCopyFile[Ex])..... 614
30.1.4. フォルダ削除／クリーンアップ(Ajc{Remove/Clean}Folder)..... 615
30.1.5. ２つのフォルダ下のファイル群・突き合せリスト列挙 (AjcEnumFilesMatchingList)..... 616
30.1.6. 親ディレクトリのパス取得(AjcGetParentDirectory)..... 618
30.1.7. ディレクトリ作成(AjcCreateDirectory[Ex])..... 618
30.1.8. 多階層ディレクトリ構造の一括作成(AjcCreateDirectoryStruct[Ex])..... 618
30.1.9. パスがワイルドカード[群]の指定に一致するかチェックする(AjcPathMatchSpec)..... 619
30.1.10. パスが指定文字列[群]を含むかチェックする(AjcPathMatchStr)..... 619
30.1.11. パスの末尾が「¥」(or「/」)かチェックする(AjcIsPathBackslash[Ex])..... 619
30.1.12. ２つのパスの間に必ず１つの「¥」を挿入しパスを結合する (AjcPathCat)..... 620
30.1.13. パス末尾の「¥」を除去して比較 (AjcPathCmp)..... 620
30.1.14. ファイル名に使用できない文字を特定の文字に変換する (AjcChangeFNameToCorrect)..... 620
30.1.15. パスが存在するかチェック(AjcPathExists)..... 620
30.1.16. パスがディレクトリかチェック(AjcPathIsDirectory)..... 621
30.1.17. パスが空のディレクトリかチェック(AjcPathIsEmptyDirectory)..... 621
30.1.18. パスがファイルかチェック(AjcPathIsFile)..... 622
30.1.19. ファイルサイズ取得(AjcGetFileSize)..... 622
30.1.20. ファイルタイム取得(AjcGetFileTime1970)..... 622
30.1.21. ファイル比較(AjcFileCompare)..... 622
30.1.22. ファイル／ディレクトリ日時取得(AjcGetFileTime)..... 623
30.1.23. ファイル／ディレクトリ日時設定(AjcSetFileTime / AjcSetFileTimeBySysTime)..... 623

30.1.24. ２つのパスが同一パスかチェックする 624
30.1.25. ２つのパスで、片方がサブパスであるかをチェックする 624
30.2. サンプルプログラム 625
30.2.1. SW_FileDir01（フォルダコピー） 625
30.2.2. SW_FileDir02（ファイル突き合せリストの作成） 629
31. テキストファイル・アクセス 632
31.1. サポートＡＰＩ 633
31.1.1. 入力テキストファイル オープン（AjcFOpen / AjcFOpenShare） 634
31.1.2. 文字列入力（AjcFGetS） 634
31.1.3. 文字コード入力（AjcFGetCode） 636
31.1.4. １バイト／１ワード入力（AjcFGetC） 636
31.1.5. ファイル読み出しポイント退避（AjcFSavePoint） 637
31.1.6. ファイル読み出しポイント回復（AjcFRecvPoint） 637
31.1.7. ファイル読み出しポイント取得（AjcFGetPoint） 637
31.1.8. ファイル読み出しポイント設定（AjcFSetPoint） 637
31.1.9. ファイル読み出しバイトポイント取得（AjcFGetBytePoint） 638
31.1.10. 入力ファイルのＥＯＦチェック（AjcFEof） 638
31.1.11. 出力テキストファイル 生成（AjcFCreate / AjcFCreateShare） 638
31.1.12. ファイルを追記モードでオープン／生成（AjcFAppend / AjcFAppendShare） 639
31.1.13. 文字列出力（AjcFPutS） 639
31.1.14. 文字コード出力（AjcFPutCode） 639
31.1.15. １バイト／１ワード出力（AjcFPutC） 640
31.1.16. 書式文字列出力（AjcFPrintF） 640
31.1.17. テキストファイル出力データフラッシュ（AjcFFlush） 640
31.1.18. テキストファイルクローズ（AjcFClose） 640
31.1.19. 改行コード変換モード設定／取得（AjcF{Set/Get}LfConv） 641
31.1.20. テキスト文字コード種別取得（AjcFGetTec） 641
31.1.21. 入力ファイルのＢＯＭ有無の取得（AjcFGetBOM） 641
31.1.22. ファイル入出力種別取得（AjcFIsModeInput） 642
31.1.23. 書式文字列バッファサイズ設定（AjcFSetFmtLen） 642
31.1.24. システムのファイルハンドル取得（AjcFGetFileHandle） 642
31.1.25. 文字コード種別設定ダイアログ（AjcFTecDialog） 643
31.1.26. システムのファイルハンドルでファイル入力用インスタンス生成（AjcFAttachOpened） 644
31.1.27. システムのファイルハンドルでファイル出力用インスタンス生成（AjcFAttachCreated） 645
31.1.28. ファイルをクローズしないでインスタンス解放（AjcFDetach） 645
31.2. サンプルプログラム 646
31.2.1. SW_TextFile（テキストファイルのアクセスサンプル） 646
31.2.2. SW_TextFileInput （テキストファイルの入力テスト） 654
31.2.3. SW_TextFileOutput（テキストファイルの出力テスト） 654
32. 平衡２分木（ＡＶＬ木） 655
32.1. サポートＡＰＩ 657
32.1.1. インスタンス生成（AjcAvlCreate） 657
32.1.2. インスタンス消去（AjcAvlDelete） 658
32.1.3. コールバックパラメタの設定（AjcAvlSetCbp） 658
32.1.4. ノード挿入（AjcAvlInsNode） 659
32.1.5. データノード挿入（AjcAvlInsDataNode） 659
32.1.6. ノード取得（AjcAvlGetNodeByKey） 659
32.1.7. ノードアドレス取得（AjcAvlGetNodePtr） 660
32.1.8. ノードのキー値取得（AjcAvlGetNodeKey） 660
32.1.9. 先頭ノード取得（AjcAvlGetTopNode） 660
32.1.10. 最終ノード取得（AjcAvlGetLastNode） 660
32.1.11. ノード置換（AjcAvlRepNode） 661
32.1.12. ノード挿入／置換（AjcAvlInsOrRepNode） 661
32.1.13. ノード削除（AjcAvlDelNode） 661
32.1.14. 全ノード削除（AjcAvlDelAllNodes） 661
32.1.15. 文字列ノード挿入（AjcAvlInsStrNode） 662
32.1.16. 文字列ノード取得（AjcAvlInsNode） 662
32.1.17. 文字列ノード挿入／取得（AjcAvlGetStrNode） 662
32.1.18. 文字列ノード削除（AjcAvlDelStrNode） 663
32.1.19. 文字列ノードにデータを関連付ける（AjcAvlSetStrNodeData） 663

32.1.20. 文字列ノードに関連付けたデータを取得（AjcAvlSetStrNodeData） 663
32.1.21. ノード数取得（AjcAvlGetCount） 663
32.1.22. 全ノードの列挙（AjcAvlEnumNodes[Ex]） 664
32.1.23. 全ノードへのポインタ配列生成（AjcAvlCreatePtrArr） 665
32.1.24. 全ノードへのポインタ配列解放（AjcAvlReleasePtrArr） 665
32.1.25. マルチスレッドの許可／禁止（AjcAvlEnableMultiThread） 666
32.2. サンプルプログラム 667
32.2.1. SW_AvlTreeC（ＡＶＬツリー操作サンプル） 667
33. 平衡２分木（文字列キー） 670
33.1. サポートＡＰＩ 671
33.1.1. インスタンス生成（AjcAvsCreate） 671
33.1.2. インスタンス消去（AjcAvsDelete） 672
33.1.3. コールバックパラメタの設定（AjcAvsSetCbp） 672
33.1.4. 文字列比較モードの設定（AjcAvsSetCompMode） 672
33.1.5. ノード挿入（AjcAvsInsNode） 673
33.1.6. ノード取得（AjcAvsGetNode） 673
33.1.7. ノードアドレス取得（AjcAvsGetNodePtr） 673
33.1.8. ノードのキー値取得（AjcAvsGetNodeKey） 673
33.1.9. 先頭ノード取得（AjcAvsGetTopNode） 674
33.1.10. 最終ノード取得（AjcAvsGetLastNode） 674
33.1.11. ノード置換（AjcAvsRepNode） 674
33.1.12. ノード挿入／置換（AjcAvsInsOrRepNode） 674
33.1.13. ノード削除（AjcAvsDelNode） 675
33.1.14. 全ノード削除（AjcAvsDelAllNodes） 675
33.1.15. ノード数取得（AjcAvsGetCount） 675
33.1.16. 全ノードのシーケンシャルな読み出し（AjcAvsEnumNodes[Ex]） 676
33.1.17. 全ノードへのポインタ配列生成（AjcAvlCreatePtrArr） 677
33.1.18. 全ノードへのポインタ配列解放（AjcAvsReleasePtrArr） 677
33.1.19. マルチスレッドの許可／禁止（AjcAvsEnableMultiThread） 678
33.2. サンプルプログラム 679
33.2.1. SW_AvsTreeC（ＡＶＳツリー操作サンプル） 679
34. 線形リスト制御 682
34.1. サポートＡＰＩ 682
34.1.1. インスタンス生成（Ajc?QueCreate） 683
34.1.2. インスタンス消去（Ajc?QueDelete） 683
34.1.3. リストの末尾へノードを追加（Ajc?QueEnque[Ex]） 684
34.1.4. リストの先頭へノードを追加（Ajc?QueEnqTop） 684
34.1.5. リストの先頭ノード取り出し（Ajc?QueDeque） 685
34.1.6. リストの先頭ノード取り出し（Ajc?QueDequeEx） 685
34.1.7. リストの末尾ノード取り出し（Ajc?QueDeqLast） 685
34.1.8. リスト中の全ノード破棄（Ajc?QuePurge） 686
34.1.9. リスト中の個別ノード破棄（Ajc?QueDelAny） 686
34.1.10. ノード数取得（Ajc?QueGetCount） 686
34.1.11. ２つのリストを連結する（Ajc?QueMerge） 687
34.1.12. 先頭ノードデータアドレス取得（Ajc?QueTopNode） 687
34.1.13. 次のノードデータアドレス取得（Ajc?QueNextNode） 687
34.1.14. 全ノードへのポインタ配列生成（Ajc?QueCreatePtrArr） 688
34.1.15. 全ノードへのポインタ配列解放（Ajc?QueReleasePtrArr） 688
34.1.16. マルチスレッドの許可／禁止（Ajc?QueEnableMultiThread） 689
34.2. サンプルプログラム 690
34.2.1. SW_FQue（線形リスト - 固定長データ） 690
34.2.2. SW_VQueC（線形リスト – 可変長データ） 693
35. 双方向リスト制御 696
35.1. サポートＡＰＩ 696
35.1.1. インスタンス生成（AjcXQueCreate） 697
35.1.2. インスタンス消去（AjcXQueDelete） 697
35.1.3. リストの末尾へノードを追加（AjcXQueEnque[Ex]） 697
35.1.4. リストの先頭へノードを追加（AjcXQueEnqTop） 698
35.1.5. ノードを指定ノードの直前に挿入（AjcXQueInsert） 698
35.1.6. 指定ノード削除（AjcXQueRemove） 698

35.1.7. リスト中の全ノード破棄（AjcXQuePurge） ... 698
35.1.8. リスト中の個別ノード破棄（AjcXQueDelAny） ... 699
35.1.9. 先頭ノード取り出し（AjcXQueDequeEx） ... 699
35.1.10. ノード数取得（AjcXQueGetCount） ... 699
35.1.11. ２つのリストを連結する（AjcXQueMerge） ... 700
35.1.12. 先頭ノードデータアドレス取得（AjcXQueTopNode） ... 700
35.1.13. 末尾ノードデータアドレス取得（AjcXQueLastNode） ... 700
35.1.14. 次のノードデータアドレス取得（AjcXQueNextNode） ... 701
35.1.15. 直前のノードデータアドレス取得（AjcXQuePrevNode） ... 701
35.1.16. 全ノードへのポインタ配列生成（AjcXQueCreatePtrArr） ... 702
35.1.17. 全ノードへのポインタ配列解放（AjcXQueReleasePtrArr） ... 702
35.1.18. マルチスレッドの許可／禁止（AjcXQueEnableMultiThread） ... 703
35.2. サンプルプログラム ... 704
35.2.1. SW_XQueC（双方向リスト） ... 704
36. スレッド間メールデータ通信 ... 707
36.1. サポートＡＰＩ ... 707
36.1.1. インスタンス生成（Ajc{F/V}MbxCreate） ... 707
36.1.2. インスタンス消去（Ajc{F/V}MbxDelete） ... 708
36.1.3. メールデータ通知（Ajc{F/V}MbxEnque） ... 708
36.1.4. 優先メールデータ通知（Ajc{F/V}MbxEnqTop） ... 708
36.1.5. メールデータ取り出し（Ajc{F/V}MbxDeque） ... 709
36.1.6. メールデータ取り出し（Ajc{F/V}MbxDeque） ... 709
36.1.7. 全メールデータ破棄（Ajc{F/V}MbxPurge） ... 710
36.1.8. 個別メールデータ破棄（Ajc{F/V}MbxDelAny） ... 710
36.1.9. メールデータ数取得（Ajc{F/V}MbxGetCount） ... 710
36.1.10. 現在のメールデータの総バイト数取得（AjcVMbxGetTotalBytes） ... 711
36.2. サンプルプログラム ... 712
36.2.1. SW_FMbx（スレッド間メールデータ通信 – 固定長データ） ... 712
36.2.2. SW_VMbxC（スレッド間メールデータ通信 – 可変長データ） ... 714
37. リングバッファ制御 ... 717
37.1. サポートＡＰＩ ... 717
37.1.1. インスタンス生成（AjcRngCreate） ... 718
37.1.2. インスタンス消去（AjcRngDelete） ... 718
37.1.3. データ格納（AjcRngPutData，格納できない場合は、格納できる分だけ格納） ... 718
37.1.4. データ格納（AjcRngPutDataEx，格納できない場合は、データ全部を格納しない） ... 719
37.1.5. データ取り出し（AjcRngGetData） ... 719
37.1.6. データ覗き見（AjcRngPeekData） ... 719
37.1.7. 総格納データサイズ取得（AjcRngGetDataSize） ... 719
37.1.8. 格納データ破棄（AjcRngPurge） ... 720
37.1.9. マルチスレッドの許可／禁止（AjcRngEnableMultiThread） ... 720
37.2. サンプルプログラム ... 721
37.2.1. SW_RingBufC（リングバッファのアクセス） ... 721
38. Ｃ言語の字句分解 ... 723
38.1. 空白，コメント，改行や行の継続記号（¥）もトークンとして読み出す ... 723
38.2. プリプロセス文情報 ... 724
38.3. #include 文のヘッダファイル名情報 ... 724
38.4. サポートＡＰＩ ... 725
38.4.1. インスタンス生成（AjcCtkCreate） ... 726
38.4.2. リセット（AjcCtkReset） ... 726
38.4.3. 機能フラグの設定／取得（AjcCtk{Set|Get}Flag） ... 727
38.4.4. コールバック設定（AjcCtkSetCallBack） ... 727
38.4.5. コールバックパラメタ設定（AjcCtkSetCallBackParam） ... 727
38.4.6. インスタンス消去（AjcCtkDelete） ... 727
38.4.7. 字句の読み出し（AjcCtkGetToken） ... 728
38.4.8. 現在の字句情報の取得（AjcCtkPeekToken） ... 730
38.4.9. トークンコードに対応する文字列の取得（AjcCtkGetTokenString） ... 730
38.4.10. インスタンスの複製（AjcCtkGetReplicatedHandle） ... 731
38.5. サンプルプログラム ... 732
38.5.1. SW_Ctk01C（Ｃ言語ソースファイルからコメント削除） ... 732
38.5.2. SW_Ctk02（字句情報のログ表示） ... 734

39. Ｃ言語のプリコンパイル ... 737
39.1. プリコンパイルオプション ... 737
39.1.1. インクルードファイル自動検索（AJCPPC_FLG_AUTO_SEARCH） ... 737
39.1.2. 同一インクルードファイルを１回だけ読み出す（AJCPPC_FLG_ONCE） ... 738
39.1.3. 非生成部分（条件コンパイル＝偽の部分）もファイル出力する（AJCPPC_FLG_GENALL） ... 738
39.2. プリコンパイル結果のファイル出力 ... 739
39.2.1. インクルードファイルの展開／非展開 ... 739
39.2.2. 条件コンパイルの真偽値の表示 ... 740
39.3. トークン情報の形式 ... 741
39.4. サポートＡＰＩ ... 742
39.4.1. インスタンス生成（AjcPpcCreate） ... 742
39.4.2. インスタンス消去（AjcPpcDelete） ... 745
39.4.3. プリコンパイル・オプションの設定／取得（AjcPpc{Set|Get}Option） ... 745
39.4.4. プリコンパイルの実行（AjcPpcCompile） ... 745
39.4.5. プリコンパイル済オブジェクトの取得（AjcPpcGetObject） ... 746
39.4.6. プリコンパイル済オブジェクトの解放（AjcPpcReleaseObject） ... 746
39.4.7. マクロ定義情報の列挙（AjcPpcEnumMacro） ... 746
39.4.8. マクロ定義情報の取得（AjcPpcGetMacroInfo） ... 747
39.4.9. プリコンパイルの中止（AjcPpcStop） ... 747
39.4.10. トークンストリームのファイル出力（AjcPpcTokenStreamToFile） ... 748
39.4.11. トークンストリームをファイル出力する際のテキスト文字コード設定（AjcPpcSetTecAtTokenStreamToFile） ... 748
39.4.12. ソースファイルのテキスト文字コード設定(AjcPpcSetTextEncode) ... 749
39.4.13. ソースファイルのテキスト文字コード取得(AjcPpcGetTextEncode) ... 749
39.4.14. エラーメッセージテキスト取得(AjcPpcGetErrMsgText) ... 750
39.5. サンプルプログラム ... 751
39.5.1. SW_CPrePro（Ｃ言語のプリコンパイル） ... 751
40. 変数管理 ... 759
40.1. 変数情報 ... 759
40.2. サポートＡＰＩ ... 760
40.2.1. インスタンス生成（AjcVmgCreate） ... 761
40.2.2. インスタンス消去（AjcVMgDelete） ... 761
40.2.3. 全変数情報の破棄（AjcVMgPurge） ... 761
40.2.4. 変数の作成（AjcVmgGenVar） ... 761
40.2.5. 変数の属性設定／取得（AjcVmg{Set/Get}Att） ... 762
40.2.6. 変数の削除（AjcVmgDelVar） ... 762
40.2.7. 変数値の設定（AjcVmgSetInteger, AjcVmgSetReal, AjcVmgSetString） ... 763
40.2.8. 変数値の取得（AjcVmgGetInt32, AjcVmgGetInt 64, AjcVmgGetReal, AjcVmgGetString） ... 763
40.2.9. 変数値の交換（AjcVmgSwap） ... 764
40.2.10. 配列要素数の取得（AjcVmgGetArrNum） ... 764
40.2.11. 変数情報の取得（AjcVmgGetNode） ... 764
40.2.12. 登録済変数の列挙（AjcVmgEnumVar） ... 765
40.2.13. 変数のコピー（AjcVmgCopy） ... 765
40.3. サンプルプログラム ... 766
40.3.1. SW_Vmg01C（変数の作成／設定／表示） ... 766
40.3.2. SW_Vmg02C（配列変数） ... 769
41. 文字列操作 ... 771
41.1. 多重文字列 ... 771
41.2. マクロ ... 771
41.3. サポートＡＰＩ ... 772
41.3.1. 比較方法を指定して文字列比較（AjcStrCmpEx） ... 774
41.3.2. 部分文字列検索（AjcStrIStr, 単純バイト比較） ... 774
41.3.3. 部分文字列検索（AjcMbsIStr, マルチバイト比較） ... 775
41.3.4. 複数の文字列を指定して部分文字列検索（AjcStrFind） ... 776
41.3.5. 文字列内のトークンを検索（AjcStrTok[S]） ... 777
41.3.6. マルチバイト文字列内のトークンを検索（AjcMbsTok） ... 777
41.3.7. 区切り文字で区切られた文字列内のトークンを取得（AjcStrTok[S]Ex） ... 778
41.3.8. 文字列を指定文字で囲む（AjcStrEnclose） ... 779
41.3.9. 文字列の囲みを取り除く(AjcStrStripEnclose) ... 779
41.3.10. 書式文字列の生成（AjcSn[V]PrintF） ... 780
41.3.11. 書式文字列生成と１０進数文字列の整数部で特定桁数毎に区切り文字を挿入する（AjcSnPrtSep） ... 780

41.3.12. １０進数文字列の整数部から区切り文字を削除する（AjcStrRmvSepChar）... 780
41.3.13. 引数情報を指定した書式文字列の生成（AjcVSnPrintF）... 781
41.3.14. 多重文字列→文字列ポインタ配列生成（AjcMStrMakeArray）... 782
41.3.15. 文字列ポインタ配列開放(AjcMStrReleaseArray)... 782
41.3.16. 多重文字列中の文字列の個数取得(AjcMStrCount)... 782
41.3.17. 多重文字列の総文字数取得(AjcMStrTotalSize)... 782
41.3.18. 部分文字列置換（AjcRepPartStr）... 783
41.3.19. 文字列の前部空白／不要文字群を除去(AjcStrLTrim[Ex])... 783
41.3.20. 文字列の後部空白／不要文字群を除去(AjcStrRTrim[Ex])... 784
41.3.21. 文字列の両端の空白／不要文字群を除去(AjcStrTrim[Ex])... 784
41.3.22. 文字列中の ESC シーケンスを除去(AjcRemoveEscInStr)... 784
41.3.23. 文字列の整形(AjcStrShaping)... 785
41.3.24. マルチバイト（シフトＪＩＳ）文字列情報の取得(AjcStrChkMbcPart)... 785
41.3.25. ＵＴＦ－８ 文字列情報の取得(AjcStrChkUtf8Part)... 785
41.3.26. 日本語ＥＵＣ文字列情報の取得(AjcStrChkEucPart)... 786
41.3.27. ＵＴＦ８→ マルチバイト（シフトＪＩＳ）文字列変換(AjcUtf8ToMbc[Ex])... 786
41.3.28. マルチバイト（シフトＪＩＳ） →ＵＴＦ８文字列変換(AjcMbcToUtf8[Ex])... 786
41.3.29. 日本語ＥＵＣ → シフトＪＩＳ文字列変換(AjcEucToSJis[Ex])... 787
41.3.30. シフトＪＩＳ → 日本語ＥＵＣ文字列変換(AjcSJisToEuc[Ex])... 787
41.3.31. 文字コード判別（シフトＪＩＳ／日本語ＥＵＣ／ＵＴＦ-８）(AjcStrChkCode)... 787
41.3.32. 文字コード判別（シフトＪＩＳ／日本語ＥＵＣ／ＵＴＦ-８／ＵＴＦ－１６）(AjcStrChkCodeEx)... 788
41.3.33. 文字の長さ（バイト数／ワード数）取得(AjcStrChk{Mbc/Utf8/Euc/U16Le/U16Be}Len)... 788
41.3.34. 文字列の長さ補正(AjcStrAdjustLen)... 788
41.3.35. パスとファイル名を反対にした文字列を作成する（AjcStrCnvRevPath）... 789
41.3.36. パスとファイル名を反対にしたパス名を元に戻す（AjcStrRetRevPath）... 789
41.3.37. 文字列が全て同一文字で構成されているかチェック（AjcStrRetRevPath）... 789
41.3.38. １６進文字列を数値に変換（AjcHexToUI / AjcHexToULL）... 790
41.3.39. ８進文字列を数値に変換（AjcOctToUI / AjcOctToULL）... 790
41.3.40. １６進文字列を数値に変換［１６進文字チェック付］(AjcHexStrTo???)... 790
41.3.41. １０／１６進文字列を数値に変換（AjcAscToInt / AjcAscToLInt）... 790
41.3.42. １０／１６進文字列を実数値に変換（AjcAscToReal）... 791
41.3.43. 文字列中の区切られた複数の数値を取得する（AjcGetSepNumbers）... 791
41.3.44. 接頭語に続くパラメタ取得（AjcGetAfterPrefix）... 792
41.3.45. マルチバイトが分断されないようにマルチバイトテキストを抽出（AjcExtractCompletedText）... 792
41.3.46. マルチバイトが分断されないように抽出したマルチバイトテキストを解放（AjcReleaseCompletedText）... 793
41.3.47. Ｃ言語表記文字列をバイナリ化（AjcCLangStrToBin）... 793
41.3.48. バイナリ文字列Ｃ言語表記文字列化（AjcCLangStrToBin）... 793
41.3.49. テキストの表示桁数取得(AjcTextLen)... 794
41.3.50. 文字の全角／半角チェック(AjcIsBigChar)... 794
41.3.51. UNICODE コードポイント算出(AjcCodePoint)... 794
41.3.52. UNICODE コードポイントからテキスト生成(AjcCodePointToText)... 794
41.3.53. フォント情報(LOGFONT)を文字列に変換(AjcLogFontToText)... 795
41.3.54. 文字列をフォント情報(LOGFONT) に変換(AjcTextToLogFont)... 795
41.4. マルチバイト文字列用インライン関数... 796
41.5. 文字列操作マクロ... 796
41.6. 文字コードチェックマクロ... 798
42. 文字列プール... 800
42.1. サポートＡＰＩ... 800
42.1.1. 文字列プールのインスタンス生成（AjcSplCreate）... 800
42.1.2. 文字列プールのインスタンス消去（AjcSplDelete）... 800
42.1.3. 文字列の比較方法設定／取得（AjcSpl{Set/Get}CompMode）... 801
42.1.4. 文字列プールへ文字列の登録（AjcSplRegist）... 801
42.1.5. 文字列プールから文字列を検索（AjcSplFind）... 802
42.1.6. 部分文字列一致検索（文字列プールの文字列が指定部分文字列と一致するかチェック）(AjcSplPartStrInPool) 802
42.1.7. 部分文字列一致検索（指定文字列が文字列プール中の部分文字列と一致するかチェック）(AjcSplPoolStrInStr) 802
42.1.8. 文字列プールから文字列を削除（AjcSplRemove）... 802
42.1.9. 文字列プールの登録済文字列数取得（AjcSplGetCount）... 803
42.1.10. 文字列プールリセット（AjcSplReset）... 803
42.1.11. 登録済み全文字列の取得（AjcSplEnumStr）... 803
42.1.12. ＡＶＬ２分木ハンドル取得（AjcSplGetAvlHandle）... 803

43. 日付／時刻 ... 804
43.1. サポートＡＰＩ ... 804
43.1.1. 1970/1/1 00:00:00 からの通算秒をシステムタイムに変換(AjcTime1970ToSysTime) ... 804
43.1.2. システムタイムを 1970/1/1 00:00:00 からの通算秒に変換(AjcSysTimeToTime1970) ... 804
43.1.3. システムタイムをローカルタイムに変換(AjcSysTimeToLocalTime) ... 804
43.1.4. 指定秒数進めた（あるいは戻した）日時を求める(AjcTimeUpdate) ... 805
43.1.5. 日付文字列取得(AjcDateStr) ... 805
43.1.6. 時刻文字列取得(AjcTimeStr) ... 805
43.1.7. 日付文字列から SYSTEMTIME 構造体の日付情報を設定(AjcSetDateFromText[Ex]) ... 806
43.1.8. 時刻文字列から SYSTEMTIME 構造体の時刻情報を設定(AjcSetTimeFromText) ... 806
43.1.9. 日時文字列から SYSTEMTIME 構造体の日時情報を設定(AjcSetDateAndTimeFromText[Ex]) ... 807
43.1.10. ランダムな日付と時刻を取得(AjcGetRandomDateAndTime) ... 807
43.2. サンプルプログラム ... 808
43.2.1. SW_DateAndTime1（日時テキストの評価） ... 808
43.2.2. SW_DateAndTime2（ファイルへランダムなタイムスタンプ設定） ... 811
44. 時間計測 ... 812
44.1. サポートＡＰＩ ... 812
44.1.1. 時間計測開始(AjcTimeMeasureStart) ... 812
44.1.2. 周期時間計測(AjcTimeMeasureInterval) ... 812
44.1.3. 経過時間計測(AjcTimeMeasureElapse) ... 812
44.1.4. 時間計測オブジェクト生成(AjcMesTimeCreate) ... 813
44.1.5. 時間計測オブジェクト消去(AjcMesTimeDelete) ... 813
44.1.6. 周期時間計測(AjcMesTimeInterval) ... 813
44.1.7. 経過時間計測(AjcTimeMeasureElapse) ... 813
45. イメージの描画 ... 814
45.1. サポートＡＰＩ ... 814
45.1.1. ファイルからイメージデータ読み出し（AjcImgFuncRead) ... 814
45.1.2. イメージ描画（AjcImgFuncDraw） ... 815
45.1.3. イメージデータ解放（AjcImgFuncRelease) ... 815
45.2. サンプルプログラム ... 816
45.2.1. SW_ShowImage01（イメージファイルの表示） ... 816
46. ＬＺＨ書庫データのアクセス ... 819
46.1. サポートＡＰＩ ... 819
46.1.1. 書庫内のファイル名検索［初回］(AjcLzhFindFirst) ... 820
46.1.2. 書庫内のファイル名検索［２回目以降］(AjcLzhFindNext) ... 821
46.1.3. 書庫内のファイルオープン（AjcLzhOpen) ... 822
46.1.4. 書庫内のファイル読み出し（AjcLzhRead) ... 823
46.1.5. 書庫アクセスのクローズ（AjcLzhClose) ... 823
46.1.6. 検索したファイル情報の取得（AjcLzhGetFileInfo) ... 823
46.1.7. 書庫アクセスのエラーコード取得（AjcLzhGetLastError) ... 824
46.2. サンプルプログラム ... 825
46.2.1. SW_LzhDecodeC（ LZH 書庫ファイルの解凍） ... 825
47. ビットマップ操作 ... 827
47.1. サポートＡＰＩ ... 827
47.1.1. ビットマップの色変更(AjcChangeBitmapColor) ... 827
47.1.2. ビットマップデータをクリップボードへ格納する(AjcSetBitmapToClipboard) ... 827
47.1.3. クリップボードからビットマップデータを取得する(AjcGetBitmapFromClipboard) ... 827
47.1.4. ビットマップファイルを作成する(AjcWriteBitmapToFile) ... 828
47.1.5. ビットマップファイルを読み出してＤＩＢセクションを作成((AjcCreateDibFromFile) ... 828
47.1.6. ビットマップサイズの取得(AjcGetBitmapSize) ... 830
47.1.7. アイコンサイズの取得(AjcGetIconSize) ... 830
48. ＤＩＢセクション ... 831
48.1. サポートＡＰＩ ... 831
48.1.1. ＤＩＢセクションの生成（AjcDibCreate) ... 832
48.1.2. ＤＩＢセクションを特定の色でクリアー（AjcDibClear) ... 832
48.1.3. ＤＩＢセクションへビットマップをコピー（AjcDibCopyBitmap[V]) ... 833
48.1.4. ＤＩＢセクションからＤＩＢセクションへ特定の色範囲の部分だけをコピー（AjcDibCorrectedCopy[V]) ... 834
48.1.5. ＤＩＢセクションからＤＩＢセクションへ特定の色範囲を除く部分だけをコピー（AjcDibExcludedCopy[V]) ... 834
48.1.6. ＤＩＢセクションからＤＩＢセクションへ複数の色範囲を選択してコピー（AjcDibSelectedCopy[V]) ... 835
48.1.7. ＤＩＢセクション内で特定の色範囲のピクセル数をカウントする（AjcDibColorCount[V]) ... 836

48.1.8. 矩形領域の塗りつぶし（AjcDibGetPixel[V]）... 836
48.1.9. ピクセルの取得（AjcDibGetPixel[V]）... 836
48.1.10. ピクセルの設定（AjcDibSetPixel[V]）... 836
48.1.11. ラインの先頭アドレス取得(AjcDibGetLinePtr)... 837
48.1.12. ＤＩＢセクションの生成とビットマップファイルの読み出し（AjcDibReadFileAndCreate）... 837
48.2. サンプルプログラム... 838
48.2.1. SW_DibSect（色の合成）... 838
49. テキスト描画... 841
49.1. 描画域とテキスト描画域，文字列描画域... 841
49.2. イメージ（ビットマップ，アイコン）の描画... 841
49.3. エスケープシーケンス... 842
49.4. 制御文字... 843
49.5. テキストを右端，下端や中央に表示... 843
49.6. タブ('¥t')の扱い... 844
49.7. サポートＡＰＩ... 845
49.7.1. インスタンスの生成（AjcTxoCreate）... 846
49.7.2. インスタンスの消去（AjcTxoDelete）... 846
49.7.3. 描画域のビットマップサイズ設定（AjcTxoSetDcBmpSize)... 846
49.7.4. デフォルトの文字色設定(AjcTxoSetDefTextColor)... 847
49.7.5. デフォルトの文字背景色設定(AjcTxoSetDefBkColor)... 847
49.7.6. 文字列の整列(AjcTxoSetAlign / AjcTxoGetAlign)... 847
49.7.7. 描画位置設定(AjcTxoLocate / AjcTxoSetPos)... 848
49.7.8. 描画位置取得(AjcTxoGetPos)... 848
49.7.9. 書式テキスト描画(AjcTxoPrintF)... 848
49.7.10. テキスト描画(AjcTxoTextOut / AjcTxoTextOutEx)... 849
49.7.11. 文字列の行数取得(AjcTxoGetLineCount)... 849
49.7.12. 文字列描画の矩形サイズ取得(AjcTxoGetExtent[Ex][Ex])... 850
49.7.13. パレットの設定(AjcTxoSetPalette / AjcTxoSetPaletteItem)... 851
49.7.14. パレットの取得(AjcTxoGetPalette / AjcTxoGetPaletteItem)... 851
49.7.15. 文字サイズ取得(AjcTxoGetCharSize)... 851
49.7.16. 行間スペース設定／取得(AjcTxo{Set/Get}LineSpace)... 851
49.7.17. タブステップ設定／取得(AjcTxo{Set/Get}TabStep)... 852
49.7.18. 文字背景モード設定／取得(AjcTxo{Set/Get}BkMode)... 852
49.7.19. ＥＳＣ文字列の列挙(AjcTxoEnumEsc)... 852
49.7.20. ＲＧＢ表現文字列の解析(AjcTxoGetRgbInEsc)... 853
49.7.21. 描画イメージ（ビットマップ）の登録（AjcTxoRegistBitmap)... 853
49.7.22. 描画イメージ（アイコン）の登録（AjcTxoRegistIcon)... 853
49.7.23. 描画イメージの登録解除（AjcTxoUnregistImage)... 854
49.8. サンプルプログラム... 855
49.8.1. SW_TextOut... 855
50. 時計表示... 859
50.1. サポートＡＰＩ... 859
50.1.1. インスタンス生成(AjcWatCreate)... 860
50.1.2. インスタンス消去(AjcWatDelete)... 860
50.1.3. 時計表示(AjcWatShow)... 860
50.1.4. 時計非表示(AjcWatHide)... 861
50.1.5. 時計非表示状態の設定(AjcWatSetShowState)... 861
50.1.6. 時計表示状態の取得(AjcWatGetShowState)... 861
50.1.7. 時計ウインドのハンドル取得(AjcWatSetShowState)... 861
50.1.8. 時計を進める(AjcWatAdvance)... 862
51. XMODEM／YMODEM ファイル転送プロトコル... 863
51.1. ファイル転送手順... 863
51.2. 転送 DATA 形式... 864
51.3. プロトコル種別... 864
51.4. ファイル属性情報... 864
51.5. コンソールアプリで使用する場合... 865
51.6. サポートＡＰＩ... 865
51.6.1. ＸＭＯＤＥＭ／ＹＭＯＤＥＭ通信プロトコル・インスタンス生成（AjcXymCreate)... 866
51.6.2. ＸＭＯＤＥＭ／ＹＭＯＤＥＭ通信プロトコル・インスタンス消去（AjcXymDelete)... 868
51.6.3. 送信時タイマ情報設定／取得（AjcXym{Set/Get}TxTimeInfo)... 868

51.6.4. 受信時タイマ情報設定／取得（AjcXymGetRxTimeInfo）... 868
51.6.5. ファイル送信開始（AjcXymTxStart）... 869
51.6.6. ファイル受信開始（AjcXymRxStart）... 869
51.6.7. ファイル転送中止(AjcXymStop)... 870
51.6.8. シリアル回線からの受信データ投与（AjcXymPutRxChar)... 870
51.6.9. シリアル回線への送信完了を通知（AjcXymTxEnd)... 870
51.6.10. 処理状態取得（AjcXymGetState）... 870
51.7. サンプルプログラム... 871
51.7.1. SW_XYModemTx（XMODEM/YMODEM ファイル送信）... 871
51.7.2. SW_XYModemRx（XMODEM/YMODEM ファイル受信）... 877
52. 印 刷... 883
52.1. 用紙の向き... 883
52.2. 余白情報... 883
52.3. ページ情報... 883
52.4. プリンタ情報... 884
52.5. サポートＡＰＩ... 884
52.5.1. 印刷インスタンス生成（AjcPrnCreate)... 885
52.5.2. 印刷インスタンス消去（AjcPrnDelete)... 885
52.5.3. コールバックパラメタ設定(AjcPrnSetCallbackParam)... 885
52.5.4. ページ印刷開始コールバック設定（AjcPrnSetCallbackQueryPage)... 886
52.5.5. ＤＩＢセクション・ビットマップによる描画コールバック設定(AjcPrnSetCallbackByDibSect)... 886
52.5.6. プリンタＤＣによる描画コールバック設定（AjcPrnSetCallbackByPrinter)... 887
52.5.7. プリンタの選択（AjcPrnSelectDlg）... 887
52.5.8. 印刷ダイアログ（AjcPrnPrintDlg）... 888
52.5.9. 余白サイズ設定（AjcPrnSetMargin)... 889
52.5.10. 印刷開始（AjcPrnStart）... 889
52.5.11. プリンタ設定情報取得（AjcPrnGetInfo)... 889
52.6. サンプルプログラム... 890
52.6.1. SW_PrintText（テキストファイルの印刷）... 890
52.6.2. SW_PrintImage（イメージファイルの印刷 ）... 896
53. 高速フーリエ変換... 901
53.1. サポートＡＰＩ... 901
53.1.1. ＦＦＴ計算インスタンス生成(AjcFftCreate)... 902
53.1.2. ＦＦＴ計算（AjcFftCalc）... 902
53.1.3. ＦＦＴ計算インスタンス消去(AjcFftDelete)... 902
53.1.4. ＦＦＴ計算用三角関数テーブル，ビット反転テーブル作成(AjcFftMakeTable)... 903
53.1.5. ＦＦＴ計算（テーブル指定）(AjcFftCalcByTbl)... 903
53.2. サンプルプログラム... 904
53.2.1. SW_FFT（高速フーリエ変換の演算とグラフ表示）... 904
54. モニタ情報... 907
54.1. サポートＡＰＩ... 907
54.1.1. マルチモニタ全体の矩形情報情報の取得（AjcGetMonitorsRect）... 907
54.1.2. 各モニタ矩形情報情報の取得（AjcGetMonitorsInfo）... 907
54.1.3. プライマリモニタ矩形情報の取得（AjcGetPrimaryMonitorInfo）... 908
54.1.4. ポイントが属するモニタ矩形情報の取得（AjcGetMonitorInfoOfPoint）... 908
54.1.5. ウインドが属するモニタ矩形情報の取得（AjcGetMonitorInfoOfWindow）... 908
54.1.6. ウインドが属するモニタ内に収まるようなウインド情報取得（AjcGetWndInfoToFitOnMonitor）... 909
54.2. サンプルプログラム... 910
54.2.1. SW_MonInfo（モニタ情報表示）... 910
55. ヒープソート... 913
55.1. サポートＡＰＩ... 913
55.1.1. 配列のソート（AjcHeapSort）... 913
56. 演算... 914
56.1. サポートＡＰＩ... 914
56.2. ベクトルデータ構造体... 915
56.2.1. 正弦（AjcSin)... 916
56.2.2. 双曲線・正弦（AjcSinh)... 916
56.2.3. 逆正弦（AjcASin)... 916
56.2.4. 余弦（AjcCos)... 916
56.2.5. 双曲線・余弦(AjcCosh)... 916

56.2.6. 逆余弦(AjcACos) 917
56.2.7. 正接 (AjcTan) 917
56.2.8. 双曲線・正接(AjcTanh) 917
56.2.9. 逆正接(AjcATan) 917
56.2.10. 逆正接(AjcATan2) 917
56.2.11. ３Ｄ・ベクトル加算(AjcV3dAdd) 918
56.2.12. ３Ｄ・ベクトル減算(AjcV3dSub) 918
56.2.13. ３Ｄ・ベクトル乗算(AjcV3dMult) 918
56.2.14. ３Ｄ・ベクトル除算(AjcV3dDiv) 918
56.2.15. ３Ｄ・線ベクトル設定(AjcV3dSetLineVec) 919
56.2.16. ３Ｄ・線分情報設定(AjcV3dSetLinePoint) 919
56.2.17. ３Ｄ・三角形情報設定(AjcV3dSetTriPoint) 919
56.2.18. ３Ｄ・ベクトルの長さ算出(AjcV3dLength) 919
56.2.19. ３Ｄ・線分の中点算出(AjcV3dLineCenter) 919
56.2.20. ３Ｄ・外積算出(AjcV3dOuter) 920
56.2.21. ３Ｄ・内積算出(AjcV3dInner) 920
56.2.22. ３Ｄ・三角形の法線ベクトル算出(AjcV3dPlaneVec) 920
56.2.23. ３Ｄ・単位ベクトル算出(AjcV3dNormal) 920
56.2.24. ３Ｄ・ベクトルの角度算出(AjcV3dTheta) 920
56.2.25. ３Ｄ・ポイントから直線へ垂直な方向ベクトル算出(AjcV3dVertVecP2L) 921
56.2.26. ３Ｄ・ポイントから直線までの長さとベクトル算出(AjcV3dDistP2L) 921
56.2.27. ３Ｄ・ポイント間の距離算出(AjcV3dDistP2P) 921
56.2.28. ３Ｄ・ラインの交点算出(AjcV3dCrossL2L) 921
56.2.29. ３Ｄ・２つのラインの交点算出(AjcV3dCrossL2LEx) 922
56.2.30. ３Ｄ・点との平面の交点算出(AjcV3dCrossP2F) 922
56.2.31. ３Ｄ・同一平面上で線分に直行するベクトル算出(AjcV3dOrthoVecOnPlane) 922
56.2.32. ３Ｄ・ベクトルと行列の掛け算(AjcV3dMultMat) 922
56.2.33. ３Ｄ・ベクトルをＸ軸周りに回転(AjcV3dRotateX) 923
56.2.34. ３Ｄ・ベクトルをＹ軸周りに回転(AjcV3dRotateY) 923
56.2.35. ３Ｄ・ベクトルをＺ軸周りに回転(AjcV3dRotateZ) 923
56.2.36. ３Ｄ・ベクトルを任意の軸周りに回転(AjcV3dRotateAny) 923
56.2.37. ３Ｄ・任意の直角なベクトル算出(AjcV3dAnyOrthoVec) 924
56.2.38. ３Ｄ・平面上の点を平面上で指定角度回転 (AjcV3dRotateOnPlane) 924
56.2.39. ３Ｄ・任意の３点を通る円の中心と半径を求める(AjcV3dCalcCircle[V]) 925
56.2.40. ３Ｄ・球面上の任意の４点から球の中心と半径を求める(AjcV3dCalcSphere [V]) 926
56.3. サンプルプログラム 927
56.3.1. SW_3dVec（３Ｄ円の算出と３Ｄ球の算出） 927
57. ３Ｄテストデータ生成 932
57.1. パラメタ 932
57.2. サポートＡＰＩ 932
57.2.1. インスタンス生成 (AjcSpdCreate) 933
57.2.2. インスタンス消去 (AjcSpdCreate) 933
57.2.3. パラメタ設定／取得 (AjcSpd{Set/Get}Param) 933
57.2.4. 演算 (AjcSpdCalc) 933
58. バイトストリーム分離 934
58.1. サポートＡＰＩ 936
58.1.1. インスタンス生成 (AjcSsepCreate) 937
58.1.2. インスタンス消去 (AjcSsepDelete) 937
58.1.3. データストリーム投与 (AjcSsepPutData) 938
58.1.4. テキストストリーム投与 (AjcSsepPutText) 938
58.1.5. リセット (AjcSsepReset) 939
58.1.6. 生成するイベントコード設定／取得 (AjcSsep{Set/Get}Event) 939
58.1.7. テキストに含める制御コードの設定／取得 (AjcSsepSetIncTxt, AjcSsep{Set/Get}IncTxtMap) 939
58.1.8. パケット・フレーム認識,制御ｺｰﾄﾞ値設定/取得(AjcSsep{Set/Get}Stx / Etx / Dle) 940
58.1.9. パケットフレームのタイムアウト値設定／取得 (AjcSsep{Set/Get}PktTimOut) 940
58.1.10. パケット・フレーム・イメージ生成 (AjcSsepMakePacket) 941
58.1.11. データメモリ開放 (AjcSsepReleasePacket) 941
58.1.12. 文字コード判定用テキスト投与 (AjcSsepPutMbc) 941
58.1.13. テキストの文字コード種別判定 (AjcSsepGetMbcKind) 942
58.1.14. 最後に判定した文字コード種別取得 (AjcSsepGetLastKind) 942

58.1.15. 文字コード判定バッファ リセット（AjcSsepMbcReset） 942
58.1.16. マルチスレッドの許可／禁止（AjcSsepEnableMultiThread） 942
58.2. サンプルプログラム 943
58.2.1. SW_SSepC（バイトストリーム分離） 943
59. 状態遷移制御 945
59.1. 状態遷移表の記述方法 945
59.2. サポートＡＰＩ 946
59.2.1. インスタンス生成（AjcStcCreate） 947
59.2.2. インスタンス消去（AjcStcDelete） 948
59.2.3. タイマ情報設定（AjcStcSetTimerInfo） 948
59.2.4. タイマ・スタート（AjcStcTimerStart） 948
59.2.5. タイマ・ストップ（AjcStcTimerStop） 948
59.2.6. 現在の状態番号設定／取得（AjcStcSetCurrentState） 949
59.2.7. イベントの実行（AjcStcExecEvent） 949
59.2.8. 保留中の全イベント破棄（AjcStcPurge） 949
59.3. サンプルプログラム 950
59.3.1. S_StateCtrl（キッチンタイマ） 950
60. ＣＲＣ／チェックサム 957
60.1. サポートＡＰＩ 957
60.1.1. 部分ＣＲＣ算出（AjcPartCrc??L / AjcPartCrc??R） 958
60.1.2. ＣＲＣ算出（AjcBlkCrc??L / AjcBlkCrc??R） 958
60.1.3. ＸＭＯＤＥＭ－ＣＲＣ／ＦＣＳ算出（AjcBlkXMODEM / AjcBlkCalcFCS） 958
60.1.4. ストリームへＣＲＣ値設定（AjcSetCrc?? ・・・） 959
60.1.5. ストリームへ XMODEM / FCS 設定（AjcSetXMODEM・・・ / AjcSetFCS・・・） 959
60.1.6. ストリームのＣＲＣ値検査（AjcChkCrc??・・・） 960
60.1.7. ストリームの XMODEM / FCS 検査（AjcChkXMODEM・・・ / AjcChkFCS・・・） 960
60.1.8. ストリームのバイト／ワードサム算出（CalcByteSum[N|S], AjcCalcWordSum[N|S]） 961
60.1.9. ストリームのバイト／ワードサム設定（AjcSetByteSum[N|S], AjcSetWordSum[N|S]） 961
60.1.10. ストリームのバイト／ワードサム検査（ChkByteSum[N|S], ChkWordSum[N|S]） 962
61. ＨＥＸデータ処理 963
61.1. サポートＡＰＩ 963
61.1.1. インテルＨＥＸ／モトローラＳ形式のデータをメモリ上に展開する（AjcLoadHexData） 964
61.1.2. インテルＨＥＸ／モトローラＳ形式のＨＥＸファイルのロード（AjcLoadHexFile） 965
61.1.3. インテルＨＥＸ形式のデータをバイナリ形式に変換する（AjcIHexToBin） 966
61.1.4. モトローラＳ形式のデータをバイナリ形式に変換する（AjcSMotToBin） 966
62. フォント処理とフォント選択ダイアログ 967
62.1. サポートＡＰＩ 967
62.1.1. ダイアログボックスによるフォント選択（AjcCfFontDlg） 968
62.1.2. フォント情報，ダイアログ設定値の初期化（AjcCfInitFontInfo, AjcCfInitPermInfo） 969
62.1.3. フォント情報，ダイアログ設定値 → 文字列 変換（AjcCfInfoToText） 970
62.1.4. 文字列 → フォント情報，ダイアログ設定値変換（AjcCfTextToInfo） 970
62.1.5. フォント情報をフォント選択ダイアログ情報に組み入れる（AjcCfMergeToInfo） 970
62.1.6. フォント情報の列挙（AjcCfEnumFonts） 970
62.1.7. フォントフェース名検索（AjcCfFindFont） 971
62.1.8. フォント選択ダイアログのプレビューテキスト設定(AjcCfSetPreviewText) 971
62.1.9. 英字名を含む全フォント情報の生成（AjcCfCreateAllFontInfo） 972
62.1.10. 英字名を含む全フォント情報の解放（AjcCfReleaseAllFontInfo） 972
62.1.11. 英字名を含む全フォントの検索（AjcCfFindFontEx） 973
62.1.12. 英字名を含む全フォントの列挙(AjcCfEnumFontsEx) 973
62.1.13. ポイント数 → ピクセル数変換（AjcCfPointsToPixels） 973
62.1.14. ピクセル数 →ポイント数変換（AjcCfPixelsToPoints） 973
62.2. サンプルプログラム 974
62.2.1. SW_FontDlg（フォント選択） 974
63. その他 977
63.1. サポートＡＰＩ 977
63.1.1. 言語種別設定／取得（Ajc{Set/Get}LangId） 978
63.1.2. アプリケーション名の設定（AjcSetAppName） 978
63.1.3. アプリケーション・アイコンの設定（AjcSetAppIcon） 978
63.1.4. コントロールからの通知時の lParam モード設定(AjcSetCmdWithHdl) 978
63.1.5. GetLastErrot()で得たエラーコードが表す内容のテキストを取得（AjcGetLastErrorText） 979

63.1.6. 符号なし１０進文字列を数値に変換（AjcDecToUI / AjcDecToULL） 979
63.1.7. 指定された、数値を単位数値の整数倍に調整する（AjcAdjustMultValue） 979
63.1.8. 指定された値の絶対値以下で、最も近い１０のｎ乗値を求める（AjcAdjustPow10） 979
63.1.9. 指定された値の絶対値以上で、最も近い「１０ⁿ×Ｎ」を求める（Ajc[Adjust | Round]MultPow10） 980
63.1.10. エンディアン形式変換（AjcExcWord / Long / LLong / Float / Double） 980
63.1.11. バイト単位のメモリ割り当て（AjcMAlloc / AJCMEM） 981
63.1.12. 割り当てたメモリの解放（AjcMFree） 981
63.1.13. メモリ割り当てエラー通知コールバックの設定（AjcSetMemError） 981
63.1.14. メモリブロックの内容交換（AjcMemSwap） 982
63.1.15. 指定データでバッファーを埋める（AjcMemSet{8/16/32}） 982
63.1.16. テキストメモリの確保（AjcTAlloc） 982
63.1.17. テキストメモリの開放（AjcTFree） 982
63.1.18. 明るさを変更した色を算出（AjcChgBright） 983
63.1.19. 色の明るさ算出(AjcRgbBright) 983
63.1.20. 色の輝度算出(AjcRgbIntensity) 983
63.1.21. 輝度比算出(AjcRgbIntensityRatio) 983
63.1.22. 色の差を算出(AjcRgbIntensityRatio) 984
63.1.23. 補色／反対色を算出(AjcComplementaryColor) 984
63.1.24. コマンド引数のポインタ配列作成（AjcParseCmdArgs） 984
63.1.25. コマンド引数ポインタ配列の開放(AjcReleaseCmdARgs) 985
63.1.26. マンド引数のポインタ配列作成・拡張版（AjcParseCmdArgsEx） 985
63.1.27. コマンドライン先頭の引数をスキップする（AjcSkipCmdARgs） 986
63.1.28. ＣＰＵのＩＤ情報取得(AjcGetCpuId) ・・・Win32 のみ 986
63.1.29. ２つの角度値の変移角を求める（AjcDifferenceOfTheta） 986
63.1.30. ＤＬＬのエクスポート情報取得(AjcCreateDllExportInfo) 987
63.1.31. ＤＬＬのエクスポート情報の開放(AjcReleaseDllExportInfo) 987
63.1.32. モジュール（自プロセス／ＤＬＬ）のインポート情報取得(AjcCreateModImportInfo) 988
63.1.33. モジュール（自プロセス／ＤＬＬ）のインポート情報の開放(AjcReleaseModImportInfo) 988
63.1.34. 現在のユーザ（アカウント）のＳＩＤ取得(AjcGetSidStringByCurrentUser) 989
63.1.35. ＳＩＤ文字列からＲＩＤ値取得(AjcGetRidInSid) 989
63.1.36. ２つの矩形の重なり合う部分を取得する（AjcGetDupRect） 989
63.1.37. アスペクト比を維持し拡大縮小した矩形／サイズを求める(AjcAspGetZoomed{Rect/Size/Info}) 990
63.1.38. 親プロセスの情報取得（AjcGetParentProcess） 991
63.1.39. Windows のシャットダウンやリブートを行う(AjcExitWindows[Ex]) 991
63.1.40. デバッガの出力ウインドへ書式文字列出力(AjcTrace) 991
63.1.41. ストック フォントの取得(AjcGetStockFont) 992
63.1.42. フォントハンドル生成(AjcCreateFont) 992
63.1.43. 実行ファイルのバージョン情報取得(AjcGetVerInfo) 993
63.1.44. クリップボードテキストの取得(AjcGetClipboardText) 993
63.1.45. クリップボードテキストの生成(AjcCreateClipboardText) 993
63.1.46. クリップボードテキストの解放(AjcReleaseClipboardText) 993
63.1.47. クリップボードへテキスト格納(AjcPutClipboardText) 994
63.1.48. メニューアイテムの列挙(AjcEnumMenuItems) 994
64. ＭＦＣでの利用 995
64.1. ＭＦＣサンプルプログラム１（S_MFC_01） 996
64.2. ＭＦＣサンプルプログラム２（S_MFC_02） 1001
65. Windows API の不具合 1011
65.1. コンソールから全角文字入力時 _kbhit(), _getch() が誤動作 1011
66. 問い合わせ先 1013

1. 概　要

このプログラムは、Windows 用のカスタムコントロールと、独自のカスタムコントロールや汎用ＡＰＩを収容したライブラリです。
Microsoft・Ｖｉｓｕａｌ Ｓｔｕｄｉｏ（ Visual C/C++ ）でプログラムを作成する事を前提としています。
C++用クラスライブラリについては「AjrCppClass.pdf」を参照してください。
C#（ .NET Framework ）に対応したクラスライブラリについては「AjrMsil.pdf」を参照してください。
本ライブラリは、ユーザが作成した Windows アプリケーションプログラムの部品プログラムとして動作します。

本ライブラリは、作者が過去に自身の仕事用に作成したものを整備し、ドキュメント化したものです。
作者自身は長い間、組み込みソフトの開発を行っていて、ファームウェアのテストや評価用に作成したものです。

ライブラリの用途としては、プログラム開発におけるツール作成用で、さまざまなファンクションが収録されています。
タイムチャート等の各種グラフ表示や拡張ツールチップといった汎用的なファンクションから、Ｃ言語のプリコンパイルや字句分解といった極端にマニアックな用途まであります。

本ライブラリは全ソースプログラム（ＶｉｓｕａｌＳｔｕｄｉｏプロジェクト）を同梱して配布していますので、ユーザ自身でカスタマイズや修正を行い、ライブラリを再ビルドすることができます。(詳細は「AjrCstBuildPack.pdf」を参照してください)

1.1. 依存ファイル

本ライブラリを使用してプログラムを作成する際に、必要となるファイルは、以下のとおりです。

32Bit プラットフォーム（x86）の場合

#	ファイル名	内容	格納場所（インストールフォルダ下）
1	AjrCst*.h Ajc*.h	コンパイル時にインクルードするヘッダファイル	inc
2	AjrCst32.lib	リンケージエディタで指定する LIB ファイル	bin
3	AjrCst32.dll	実行時のダイナミックリンク・ファイル	bin

64Bit プラットフォーム（x64）の場合

#	ファイル名	内容	格納場所（インストールフォルダ下）
1	AjrCst*.h Ajc*.h	コンパイル時にインクルードするヘッダファイル	inc
2	AjrCst64.lib	リンケージエディタで指定する LIB ファイル	bin
3	AjrCst64.dll	実行時のダイナミックリンク・ファイル	bin

1.2. 実行可能な Windows バージョン

本ライブラリは、Windows10/11 の 32 ビット・プラットフォーム(x86)と 64 ビット・プラットフォーム(x64)で実行可能です。

1.3. バイト文字バージョンとワイド文字(UNICODE)バージョン

本ライブラリ中で、文字列を扱う関数では、(Windows-API と同様に) 実際の関数名は、末尾に「A」あるいは「W」を付加した関数名となっています。すあ
コンパイルオプションシンボル「UNICODE」の指定が無い場合は「A」が付加され、「UNICODE」が指定された場合は「W」が付加されます。
例えば、「AjcSnPrintF」の実際の関数名は、「UNICODE」の指定が無い場合は「AjcSnPrintF**A**」、指定された場合は「AjcSnPrintF**W**」となります。

関数名末尾の「A」あるいは「W」は、本ライブラリによりコンパイル時に自動的に付加されますので、ユーザが意識して付加する必要はありません。

但し、明示的にいずれかのバージョンの関数を選択して使用する場合は「A」あるいは「W」を付加した関数名を使用してください。

尚、コンパイルオプションシンボル「UNICODE」が指定された場合は、Ｃランタイムでのワイド文字バージョン指定であるコンパイルオプションシンボル「_UNICODE」を暗黙的に生成します。

コンパイルオプションシンボルは、コマンドラインで「/D」スイッチにより指定します。
コンパイルオプションシンボル「UNICODE」を指定する場合は、コマンドラインで「/DUNICODE」を指定します。

「UNICODE」が指定さた場合、ハングル文字や中国漢字等が混在する完全な UNICODE 文字列を使用できます。
「UNICODE」が指定されない場合は、規定のコードページ (CP_ACP, 日本語の場合はシフト JIS) だけが使用可能です。

多くのＡＰＩで文字列や、文字列格納バッファを指定する場合、文字列／バッファへのポインタと文字列の長さを指定しますが、バイト文字バージョンの場合は文字列の長さをバイト数で指定し、ワイド文字(UNICODE)バージョンの場合は文字列の長さを文字数で指定します。
(厳密には、UNICODE 文字数は (サロゲートペア文字(２ワードで１文字となる文字)も含まれるため) ワード(16Bit)数となります)

main()関数や、**WinMain()**関数は、以下のように記述することにより、バイト文字／ワイド文字バージョンで兼用できます。

```
int AjcMain(int  argc, UTP  argv[]) {・・・}
```

```
int WINAPI AjcWinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow) {・・・}
```

バイト文字バージョンの場合は、「AjcMain」は「main」に、「AjcWinMain」は「WinMain」に変換されます。
ワイド文字バージョンの場合は、「AjcMain」は「wmain」に、「AjcWinMain」は「wWinMain」に変換されます。

1.4. マルチバイト文字

マルチバイト文字は、規定のコードページ (CP_ACP, 規定の Windows ANSI コードページ) として扱います。
日本語の場合は、コードページ 932 (シフト JIS) を意味します。
本書では、規定のコードページの文字を単に「マルチバイト文字」あるいは「S-JIS」として表記します。

1.5. 文字列比較モード

文字列比較モードは、各ＡＰＩで、ばらばらの名称を示していましたが、以下の名称に統一しました。(但し、旧名称も使用可)

名称	内容	旧名称
AJCCMP_EXACT_WIDTH	英大小文字を区別して比較する (”ABC” ≠ “abc”) (”AAA” < “BBB” < “aaa” < “bbb”)	AJCCMP_RECOGNIZE_WIDTH AJCCMP_DISTINGUISH AJCSPL_CMP_DISTINGUISH
AJCCMP_IGNORE_WIDTH	英大小文字を区別しないで比較する (”ABC” = “abc”)	AJCSPL_CMP_IGNORE_WIDTH AJCSPL_CMP_NODISTINGUISH
AJCCMP_MIX	英大小文字を区別し、アルファベット順に並ぶように比較する (”AAA” < “aaa” < “BBB” < “bbb”)	AJCCMP_ALPHABETIC

1.6. プロファイルへの記録による設定内容の永続化

本ライブラリ中のＡＰＩでは、設定内容をプロファイルへ記録し、永続化できるものがあります。
プロファイルの実体は初期化ファイル (.ini ファイル)、あるいは、レジストリです。
プロファイルはライブラリ内で一元的なもので、セクション名とキー名称で各情報を分別します。
プロファイルに関する内容は「プロファイル・アクセス」の章を参照してください。

1.7. バージョン更新時の注意事項

本ライブラリのバージョンを更新した場合は、ライブラリを使用しているプログラムを全て再ビルドしてください。
プログラムを配布する場合は、プログラムをビルドしたバージョンのＤＬＬ(AjrCst32.dll/AjrCst64.dll)を同梱してください。

2. 基本タイプ定義

本ライブラリでは、以下のシンボルを、基本タイプ定義として使用しています。
ユーザシステムでは、これらのシンボルを他の目的で使用しないようにする必要があります。

基本タイプ

#	シンボル	定 義	内 容
1	**VO**	typedef void VO;	void 型
2	**SB**	typedef signed char SB;	符号付き 8 ビット整数
3	**SW**	typedef signed short SW;	〃 １６ビット整数
4	**SL**	typedef signed long SL;	〃 ３２ビット整数
5	**SLL**	typedef signed long long SLL;	〃 ６４ビット整数
6	**SI**	typedef signed int SI;	〃 １６ビット以上の整数
7	**SX**	typedef _W64 signed int SX; (MS-VC, 32Bit 環境) typedef signed __int64 SX; (MS-VC, 64Bit 環境) typedef signed int SX; (VisualStudio 6.0 までのバージョン)	ポインタ値を格納可能なビット数の符号付き整数
8	**UB**	typedef unsigned char UB;	符号無し 8 ビット整数
9	**UW**	typedef unsigned short UW;	〃 １６ビット整数
10	**UL**	typedef unsigned long UL;	〃 ３２ビット整数
11	**ULL**	typedef unsigned long long ULL;	〃 ６４ビット整数
12	**UI**	typedef unsigned int UI;	〃 １６ビット以上の整数
13	**UX**	typedef _W64 unsigned int UX; (MS-VC, 32Bit 環境) typedef unsigned __int64 UX; (MS-VC, 64Bit 環境) typedef unsigned int UX; (VisualStudio 6.0 までのバージョン)	ポインタ値を格納可能なビット数の符号無し整数
14	**BC**	typedef char BC;	バイト文字
15	**WC**	typedef wchar_t WC;	ワイド文字(UNICODE)

ポインタ タイプ

#	シンボル	定 義
16	**VOP**	typedef void * VOP;
17	**SBP**	typedef SB * SBP;
18	**SWP**	typedef SW * SWP;
19	**SLP**	typedef SL * SLP;
20	**SLLP**	typedef SLL * SLLP;
21	**SIP**	typedef SI * SIP;
22	**SXP**	typedef SX * SXP;
23	**UBP**	typedef UB * UBP;
24	**UWP**	typedef UW * UWP;
25	**ULP**	typedef UL * ULP;
26	**ULLP**	typedef ULL * ULLP;
27	**UIP**	typedef UI * UIP;
28	**UXP**	typedef UX * UXP;
29	**BCP**	typedef BC * BCP;
30	**WCP**	typedef WC * WCP;

#	シンボル	定 義
31	**C_VOP**	typedef const void * C_VOP;
32	**C_SBP**	typedef const SB * C_SBP;
33	**C_SWP**	typedef const SW * C_SWP;
34	**C_SLP**	typedef const SL * C_SLP;
35	**C_SLLP**	typedef const SLL * C_SLLP;
36	**C_SIP**	typedef const SI * C_SIP;
37	**C_SXP**	typedef const SX * C_SXP;
38	**C_UBP**	typedef const UB * C_UBP;
39	**C_UWP**	typedef const UW * C_UWP;
40	**C_ULP**	typedef const UL * C_ULP;
41	**C_ULLP**	typedef const ULL * C_ULLP;
42	**C_UIP**	typedef const UI * C_UIP;
43	**C_UXP**	typedef const UX * C_UXP;
44	**C_BCP**	typedef const BC * C_BCP;
45	**C_WCP**	typedef const WC * C_WCP;

テキスト文字／テキストポインタ

#	シンボル	定 義	備考
46	**UT**	バイト文字の場合： typedef BC UT; ワイド文字の場合： typedef WC UT;	ワイド文字の場合とは、コンパイル・オプション・シンボル「UNICODE」が指定された場合のことを意味します。
47	**UTP**	バイト文字の場合： typedef BC * UTP; ワイド文字の場合： typedef WC * UTP;	
48	**C_UTP**	バイト文字の場合： typedef const BC * C_UTP; ワイド文字の場合： typedef const WC * C_UTP;	

3. メッセージマップ

3.1. ウインドプロシージャのメッセージマップ

Ｗｉｎ３２プログラミングでは、ウインドプロシージャを記述する場合、Switch 文のネスト構造となり、記述が煩雑になりがちです。
本ライブラリでは、ウインドプロシージャの記述を簡素化するためのマクロを用意しています。

以下の２つのプログラムは、いずれも同様の処理を行いますが、最初のプログラムは典型的なウインドプロシージャの記述を使用しており、２つ目のプログラムは、本ライブラリのマクロを使用したものです。

典型的なウインドプロシージャの例

```
#include   〈windows.h〉
#include   "resource.h"

HWND    hWndMain;

//===== プロトタイプ定義 ========================================================================//
static  LRESULT CALLBACK WndMainProc(HWND hwnd, UINT msg, WPARAM wParam, LPARAM lParam);

//================================================================================================//
//  WinMain                                                                                       //
//================================================================================================//
int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, LPCTSTR szCmdLine, int iCmdShow)
{
    WNDCLASS    wc;
    MSG         msg;

    //----- メイン・ウインドゥオープン ---------------------------------------------------------//
    wc.style          = 0;
    wc.lpfnWndProc    = WndMainProc;
     ・・・・・
    RegisterClass(&wc);

    hWndMain = CreateWindow(・・・・・);

    ShowWindow(hWndMain, iCmdShow);
    UpdateWindow(hWndMain);

    //----- メッセージループ -------------------------------------------------------------------//
    while (GetMessage(&msg, NULL, 0, 0)) {
        TranslateMessage(&msg);
        DispatchMessage (&msg);
    }

    return (int)msg.wParam ;
}

//================================================================================================//
//  ウインドプロシージャ                                                                          //
//================================================================================================//
static  LRESULT CALLBACK WndMainProc(HWND hwnd, UINT msg, WPARAM wParam, LPARAM lParam)
{
    switch (msg) {
        case WM_CREATE:
             ・・・・・
            return 0;

        case WM_DESTROY:
             ・・・・・
            PostQuitMessage(0);
            return 0;

        case WM_COMMAND:
            switch (LOWORD(wParam)) {
                case IDC_TXT:
                     ・・・・・
                    return 0;

                case IDC_BTN:
                     ・・・・・
                    return 0;
            }
            break;

        case WM_NOTIFY:
            switch (((NMHDR*)lParam)->code) {
                case IDC_SLD_H:
                     ・・・・・
                    return 0;

                case IDC_SLD_V:
                     ・・・・・
                    return 0;
            }
            break;
    }
    return DefWindowProc(hwnd, msg, wParam, lParam);
}
```

本ライブラリのマクロを使用した例

```
#include    <AjrCst32.h>
#include    "resource.h"

HWND    hWndMain;

//===== プロトタイプ定義 ==========================================================================//
AJC_WNDPROC_DEF(Main);      //  前方参照の場合、ウインドプロシージャのプロトタイプ定義要

//=================================================================================================//
//  WinMain                                                                                        //
//=================================================================================================//
int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
{
    WNDCLASS    wc;
    MSG         msg;

    //----- メイン・ウインドゥオープン ---------------------------------------------------------------//
    wc.style            = 0;
    wc.lpfnWndProc      = AJC_WNDPROC_NAME(Main);
    ・・・・・
    RegisterClass (&wc);

    hWndMain = CreateWindow(・・・・・);

    ShowWindow(hWndMain, iCmdShow);
    UpdateWindow(hWndMain);

    //----- メッセージループ -------------------------------------------------------------------------//
    while (GetMessage(&msg, NULL, 0, 0)) {
        TranslateMessage(&msg);
        DispatchMessage (&msg);
    }

    return (int)msg.wParam;
}

//=================================================================================================//
//  ウインドプロシージャ                                                                           //
//=================================================================================================//
AJC_WNDPROC(Main, WM_CREATE     )
{
    ・・・・・
    return 0;
}
//-------------------------------------------------------------------------------------------------//
AJC_WNDPROC(Main, WM_DESTROY    )
{
    ・・・・・
    PostQuitMessage(0);
    return 0;
}
//-------------------------------------------------------------------------------------------------//
AJC_WNDPROC(Main, IDC_TXT       )
{
    ・・・・・
    return 0;
}
//-------------------------------------------------------------------------------------------------//
AJC_WNDPROC(Main, IDC_BTN       )
{
    ・・・・・
    return 0;
}
//-------------------------------------------------------------------------------------------------//
AJC_WNDPROC(Main, IDC_SLD_H     )
{
    ・・・・・
    return 0;
}
//-------------------------------------------------------------------------------------------------//
AJC_WNDPROC(Main, IDC_SLD_V     )
{
    ・・・・・
    return 0;
}
//-------------------------------------------------------------------------------------------------//
AJC_WNDMAP_DEF(Main)
    AJC_WNDMAP_MSG(Main, WM_CREATE     )
    AJC_WNDMAP_MSG(Main, WM_DESTROY    )
    AJC_WNDMAP_CMD(Main, IDC_TXT       )
    AJC_WNDMAP_CMD(Main, IDC_BTN       )
    AJC_WNDMAP_NTC(Main, IDC_SLD_H     )
    AJC_WNDMAP_NTC(Main, IDC_SLD_V     )
AJC_WNDMAP_END
```

一見すると、本ライブラリを使用した方がステップ数が多く長いプログラムに思えますが、各メッセージやコントロールごとにサブ関数が整理される為、プログラムの見通しが良くなります。(メッセージや、コントロールが多くなるほど、この違いは顕著になります)

ウインドプロシージャの記述に関するマクロは、以下のとおりです。

マクロ形式	内容
AJC_WNDPROC_DEF(**WND**)	ウインドプロシージャ定義を生成（ウインドプロシージャのプロトタイプ定義）
AJC_WNDPROC_NAME(**WND**)	ウインドプロシージャの関数名を生成
AJC_WNDPROC(**WND**, XXX)	ウインドメッセージ、あるいは、WM_COMMAND/WM_NOTIFY メッセージにおけるコントロールＩＤに関するアクション関数の定義を生成します
AJC_WNDMAP_DEF(**WND**)	メッセージマップテーブルの定義開始
AJC_WNDMAP_MSG(**WND**, MSG)	メッセージマップテーブルのエントリ定義（ウインドメッセージ）
AJC_WNDMAP_RWM(**WND**, MSG,TEXT(*"アプリ名"*))	動的なメッセージコード用メッセージマップテーブルのエントリ定義
AJC_WNDMAP_CMD(**WND**, ID)	メッセージマップテーブルのエントリ定義（WM_COMMAND のコントロールＩＤ）
AJC_WNDMAP_NTC(**WND**, ID)	〃　　〃　　（WM_NOTIFY のコントロールＩＤ）
AJC_WNDMAP_END	メッセージマップテーブルの定義終了

「WND」は、ウインドプロシージャの識別子であり、任意のシンボルを指定します。
例えば、メインウインドに関するマクロ群ならば「Main」、ログウインドに関するマクロ群ならば「Log」といったユニークなシンボルを指定します。
尚、１つのウインドプロシージャに関するマクロ群の記述は、１つのソースプログラム内で完結しなければなりません。
（内部的に生成される関数や、変数は全て「static」属性を持ちます）

「MSG」は、処理するウインドメッセージの識別子（WM_CREATE や WM_LBUTTONDOWN 等）を指定します。
「ID」は、WM_COMMAND あるいは、WM_NOTIFY メッセージで処理するコントロールの識別子を指定します。
「XXX」は、「AJC_WNDMAP_MSG」「AJC_WNDMAP_CMD」「AJC_WNDMAP_NTC」で指定した「MSG」or「ID」を指定します。

3.1.1. メッセージマップテーブル

処理するメッセージや、コントロールのＩＤをメッセージマップテーブルとして定義します。
メッセージマップテーブルは、「AJC_WNDMAP_MSG」「AJC_WNDMAP_RWM」「AJC_WNDMAP_CMD」「AJC_WNDMAP_NTC」を「AJC_WNDMAP_DEF」～「AJC_WNDMAP_END」でサンドイッチした形で記述します。
「AJC_WNDMAP_MSG」は、ウインドメッセージそのものを処理することを定義します。
「AJC_WNDMAP_CMD」は、WM_COMMAND メッセージの特定のコントロールを処理することを定義します。
「AJC_WNDMAP_NTC」は、WM_NOTIFY メッセージの特定のコントロールを処理することを定義します。
「AJC_WNDMAP_RWM」は、動的なメッセージコードのメッセージ受信用です。（すぐ後で詳細を説明します）

「AJC_WNDMAP_MSG」「AJC_WNDMAP_CMD」「AJC_WNDMAP_RWM」「AJC_WNDMAP_NTC」の定義は順不同でＯＫです。
また、各マクロ単位でまとめる必要もありません。　次の２つの定義は、いずれも同じ効果を持ちます。

```
AJC_WNDMAP_DEF(Main)
    AJC_WNDMAP_MSG(Main, WM_CREATE )
    AJC_WNDMAP_MSG(Main, WM_DESTROY)
    AJC_WNDMAP_CMD(Main, IDOK       )
    AJC_WNDMAP_CMD(Main, IDCANCEL   )
    AJC_WNDMAP_NTC(Main, IDC_SLD_H )
    AJC_WNDMAP_NTC(Main, IDC_SLD_V )
AJC_WNDMAP_END
```

```
AJC_WNDMAP_DEF(Main)
    AJC_WNDMAP_MSG(Main, WM_CREATE )
    AJC_WNDMAP_CMD(Main, IDOK       )
    AJC_WNDMAP_NTC(Main, IDC_SLD_H )
    AJC_WNDMAP_MSG(Main, WM_DESTROY)
    AJC_WNDMAP_CMD(Main, IDCANCEL   )
    AJC_WNDMAP_NTC(Main, IDC_SLD_V )
AJC_WNDMAP_END
```

尚、これらのマクロ（AJC_WNDMAP_DEF～AJC_WNDMAP_END）の記述では、末尾にセミコロン（;）を記述してはなりません。

3.1.2. メッセージアクション関数

各「AJC_WNDMAP_MSG」「AJC_WNDMAP_RWM」「AJC_WNDMAP_CMD」「AJC_WNDMAP_NTC」に対応するアクション関数の定義は、「AJC_WNDPROC」により生成します。　アクション関数の定義は、以下のように生成されます。

```
AJC_WNDPROC(Main, WM_CREATE)
          ↓
static LRESULT CALLBACK AjcWndMainProc_WM_CREATE(HWND hwnd, UI msg, WPARAM wParam, LPARAM lParam)
```

3.1.3. ウインドプロシージャ名

ウインドプロシージャの名称は、「AJC_WNDPROC_NAME」により、以下のように生成されます。

```
AJC_WNDPROC_NAME(Main)
          ↓
    AjcWndMainProc
```

尚、ウインドプロシージャの実体は、「AJC_WNDMAP_DEF」により以下のように生成されます。

```
AJC_WNDMAP_DEF(Main)
          ↓
static LRESULT CALLBACK AjcWndMainProc(HWND hwnd, UI msg, WPARAM wParam, LPARAM lParam){・・・}
```

3.1.4. 複数のコントロールを一括処理

WM_COMMAND や WM_NOTIFY メッセージで、複数のコントロールを一括して処理する場合は、以下のようにします。

```
//=============================================================================================//
//  ウインドプロシージャ                                                                       //
//=============================================================================================//
AJC_WNDPROC(Main, WM_COMMAND    ) // WM_COMMAND メッセージアクション
{
    int cmd = LOWORD(wParam);

    if (cmd >= IDC_CHK_00 && cmd <= IDC_CHK_09) {  // IDC_CHK_00 ～ 09 は、昇順で連番

        ・・・・・(IDC_CHK_00～IDC_CHK_09 に関する処理)・・・・・

    }
    return 0;
}
//---------------------------------------------------------------------------------------------//
AJC_WNDPROC(Main, IDC_TXT       ) // IDC_TXT コントロールのアクション (WM_COMMAND, LOWORD(wParam)== IDC_TXT)
{
    ・・・・・
    return 0;
}
//---------------------------------------------------------------------------------------------//
AJC_WNDMAP_DEF(Main)
    AJC_WNDMAP_MSG(Main, WM_COMMAND    )
    AJC_WNDMAP_CMD(Main, IDC_TXT       )
AJC_WNDMAP_END
```

「AJC_WNDMAP_MSG」で「WM_COMMAND」が指定された場合は、「AJC_WNDMAP_CMD」で指定されたコントロール群のいずれのＩＤとも一致しない場合のみ WM_COMMAND のアクション関数が実行されます。
上記例では「IDC_TXT」のアクション関数を実行した場合は、WM_COMMAND のアクション関数は実行されません。
また、「IDC_TXT」以外の WM_COMMAND メッセージの場合は、全て WM_COMMAND のアクション関数が実行されます。

上記例は、WM_COMMAND に関する例を示していますが、WM_NOTIFY に関しても同様です。
つまり、「AJC_WNDMAP_MSG」で「WM_NOTIFY」が指定された場合は、「AJC_WNDMAP_NTC」で指定されたコントロール群のいずれのＩＤとも一致しない場合のみ WM_NOTIFY のアクション関数が実行されます。

3.2. ダイアログプロシージャのメッセージマップ

ダイアログプロシージャの場合も、基本的にはウインドプロシージャと同様に行います。
但し、マクロの名称がウインドプロシージャの場合と異なります。(マクロ名中の「WND」が「DLG」となります)

以下の２つのプログラムは、いずれも同様の処理を行いますが、最初のプログラムは典型的なダイアログプロシージャの記述を使用しており、２つ目のプログラムは、本ライブラリのマクロを使用したものです。

典型的なダイアログプロシージャの例

```
#include    <windows.h>
#include    "resource.h"

//===== プロトタイプ定義 ========================================================================//
static LRESULT CALLBACK DlgMainProc(HWND hDlg, UINT msg, WPARAM wParam, LPARAM lParam);

//================================================================================================//
//  WinMain                                                                                       //
//================================================================================================//
int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, LPCTSTR szCmdLine, int iCmdShow)
{
    DialogBox(hInstance, MAKEINTRESOURCE(IDD_MAIN), NULL, DlgMainProc);

    return 0;
}
//================================================================================================//
//  ダイアログプロシージャ                                                                        //
//================================================================================================//
static  LRESULT CALLBACK DlgMainProc(HWND hDlg, UINT msg, WPARAM wParam, LPARAM lParam)
{
    BOOL    rc = FALSE;

    switch (msg) {
        case WM_INITDIALOG:
             ・・・・・
            rc = TRUE; break;

        case WM_COMMAND:
            switch (LOWORD(wParam)) {
                case IDOK:
                     ・・・・・
                    EndDialog(hDlg, IDOK);
                    rc = TRUE; break;

                case IDCANCEL:
                     ・・・・・
                    EndDialog(hDlg, IDCANCEL);
                    rc = TRUE; break;
            }
            break;

        case WM_NOTIFY:
            switch (((NMHDR*)lParam)->code) {
                case IDC_SLD_H:
                     ・・・・・
                    rc = TRUE; break;

                case IDC_SLD_V:
                     ・・・・・
                    rc = TRUE; break;
            }
            break;
    }
    return rc;
}
```

本ライブラリのマクロを使用した例

```
#include    <AjrCst32.h>
#include    "resource.h"

//===== プロトタイプ定義 ==========================================================================//
AJC_DLGPROC_DEF(Main);      //  前方参照の場合、ダイログプロシージャのプロトタイプ定義要

//=================================================================================================//
//  WinMain                                                                                        //
//=================================================================================================//
int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
{
    DialogBox(hInstance, MAKEINTRESOURCE(IDD_MAIN), NULL, AJC_DLGPROC_NAME(Main));

    return 0;
}

//=================================================================================================//
//  ダイアログプロシージャ                                                                         //
//=================================================================================================//
AJC_DLGPROC(Main, WM_INITDIALOG )
{
    ・・・・・
    return TRUE;
}
//-------------------------------------------------------------------------------------------------//
AJC_DLGPROC(Main, IDOK          )
{
    ・・・・・
    EndDialog(hDlg, IDOK);
    return TRUE;
}
//-------------------------------------------------------------------------------------------------//
AJC_DLGPROC(Main, IDCANCEL      )
{
    ・・・・・
    EndDialog(hDlg, IDCANCEL);
    return TRUE;
}
//-------------------------------------------------------------------------------------------------//
AJC_DLGPROC(Main, IDC_SLD_H     )
{
    ・・・・・
    return TRUE;
}
//-------------------------------------------------------------------------------------------------//
AJC_DLGPROC(Main, IDC_SLD_V     )
{
    ・・・・・
    return TRUE;
}
//-------------------------------------------------------------------------------------------------//
AJC_DLGMAP_DEF(Main)
    AJC_DLGMAP_MSG(Main, WM_INITDIALOG )
    AJC_DLGMAP_CMD(Main, IDOK          )
    AJC_DLGMAP_CMD(Main, IDCANCEL      )
    AJC_DLGMAP_NTC(Main, IDC_SLD_H     )
    AJC_DLGMAP_NTC(Main, IDC_SLD_V     )
AJC_DLGMAP_END
```

上記例では、モーダル・ダイアログでの使用例ですが、モードレス・ダイアログでも同様に記述できます。

ダイアログプロシージャの記述に関するマクロは、以下のとおりです。

マクロ形式	内容
AJC_DLGPROC_DEF(**DLG**)	ダイアログプロシージャ定義を生成（ダイアログプロシージャのプロトタイプ定義）
AJC_DLGPROC_NAME(**DLG**)	ダイアログプロシージャの関数名を生成
AJC_DLGPROC(**DLG**, XXX)	ウインドメッセージ、あるいは、WM_COMMAND/WM_NOTIFY メッセージのおけるコントロールＩＤに関するアクション関数の定義を生成します
AJC_DLGMAP_DEF(**DLG**)	メッセージマップテーブルの定義開始
AJC_DLGMAP_MSG(**DLG**, MSG)	メッセージマップテーブルのエントリ定義（ウインドメッセージ）
AJC_DLGMAP_RWM(**DLG**, MSG,TEXT(*"アプリ名"*))	動的なメッセージコード用メッセージマップテーブルのエントリ定義
AJC_DLGMAP_CMD(**DLG**, ID)	メッセージマップテーブルのエントリ定義（WM_COMMAND のコントロールＩＤ）
AJC_DLGMAP_NTC(**DLG**, ID)	〃　　〃　　（WM_NOTIFY のコントロールＩＤ）
AJC_DLGMAP_END	メッセージマップテーブルの定義終了

「DLG」は、ダイアログプロシージャの識別子であり、任意のシンボルを指定します。
例えば、メインダイアログに関するマクロ群ならば「Main」、パラメタ設定ダイアログに関するマクロ群ならば「SetParam」といったユニークなシンボルを指定します。
尚、１つのダイアログプロシージャに関するマクロ群の記述は、１つのソースプログラム内で完結しなければなりません。
（内部的に生成される関数や、変数は全て「static」属性を持ちます）

「MSG」は、処理するウインドメッセージの識別子（WM_INITDIALOG や WM_LBUTTONDOWN 等）を指定します。
「ID」は、WM_COMMAND あるいは、WM_NOTIFY メッセージで処理するコントロールの識別子を指定します。
「XXX」は、「AJC_DLGMAP_MSG」「AJC_DLGMAP_CMD」「AJC_DLGMAP_NTC」で指定した「MSG」or「ID」を指定します。

3.2.1. メッセージマップテーブル

処理するメッセージや、コントロールのＩＤをメッセージマップテーブルとして定義します。
メッセージマップテーブルは、「AJC_DLGMAP_MSG」「AJC_DLGMAP_RWM」「AJC_DLGMAP_CMD」「AJC_DLGMAP_NTC」を「AJC_DLGMAP_DEF」～「AJC_DLGMAP_END」でサンドイッチした形で記述します。
「AJC_DLGMAP_MSG」は、ウインドメッセージそのものを処理することを定義します。
「AJC_DLGMAP_CMD」は、WM_COMMAND メッセージの特定のコントロールを処理することを定義します。
「AJC_DLGMAP_NTC」は、WM_NOTIFY メッセージの特定のコントロールを処理することを定義します。
「AJC_DLGMAP_RWM」は、動的なメッセージコードのメッセージ受信用です。（すぐ後で詳細を説明します）

「AJC_DLGMAP_MSG」「AJC_DLGMAP_CMD」「AJC_DLGMAP_RWM」「AJC_DLGMAP_NTC」の定義は順不同でＯＫです。
また、各マクロ単位でまとめる必要もありません。　次の２つの定義は、いずれも同じ効果を持ちます。

```
AJC_DLGMAP_DEF(Main)
    AJC_DLGMAP_MSG(Main, WM_INITDIALOG )
    AJC_DLGMAP_MSG(Main, WM_DESTROY    )
    AJC_DLGMAP_CMD(Main, IDOK          )
    AJC_DLGMAP_CMD(Main, IDCANCEL      )
    AJC_DLGMAP_NTC(Main, IDC_SLD_H     )
    AJC_DLGMAP_NTC(Main, IDC_SLD_V     )
AJC_DLGMAP_END
```

```
AJC_DLGMAP_DEF(Main)
    AJC_DLGMAP_MSG(Main, WM_INITDIALOG )
    AJC_DLGMAP_CMD(Main, IDOK          )
    AJC_DLGMAP_NTC(Main, IDC_SLD_H     )
    AJC_DLGMAP_MSG(Main, WM_DESTROY    )
    AJC_DLGMAP_CMD(Main, IDCANCEL      )
    AJC_DLGMAP_NTC(Main, IDC_SLD_V     )
AJC_DLGMAP_END
```

尚、これらのマクロ（AJC_DLGMAP_DEF～AJC_DLGMAP_END）の記述では、末尾にセミコロン（;）を記述してはなりません。

3.2.2. メッセージアクション関数

各「AJC_DLGMAP_MSG」「AJC_DLGMAP_CMD」「AJC_DLGMAP_RWM」「AJC_DLGMAP_NTC」に対応するアクション関数の定義は、「AJC_DLGPROC」により生成します。　アクション関数の定義は、以下のように生成されます。

```
AJC_DLGPROC(Main, WM_INITDIALOG)
          ↓
static LRESULT CALLBACK AjcDlgMainProc_WM_INITDIALOG(HWND hDlg, UI msg, WPARAM wParam, LPARAM lParam)
```

3.2.3. ダイアログプロシージャ名

ダイアログプロシージャの名称は、「AJC_DLGPROC_NAME」により、以下のように生成されます。

```
AJC_DLGPROC_NAME(Main)
          ↓
    AjcDlgMainProc
```

尚、ダイアログプロシージャの実体は、「AJC_DLGMAP_DEF」により以下のように生成します。

```
AJC_DLGMAP_DEF(Main)
          ↓
static LRESULT CALLBACK AjcDlgMainProc(HWND hDlg, UI msg, WPARAM wParam, LPARAM lParam){・・・}
```

3.2.4. 複数のコントロールを一括処理

WM_COMMAND や WM_NOTIFY メッセージで、複数のコントロールを一括して処理する場合は、以下のようにします。

```
//=============================================================================================//
//  ダイアログプロシージャ                                                                     //
//=============================================================================================//
AJC_DLGPROC(Main, WM_COMMAND    ) // WM_COMMAND メッセージアクション
{
    int cmd = LOWORD(wParam);

    if (cmd >= IDC_CHK_00 && cmd <= IDC_CHK_09) {  // IDC_CHK_00 ～ 09 は、昇順で連番

        ・・・・・(IDC_CHK_00～IDC_CHK_09 に関する処理)・・・・・

    }
    return TRUE;
}
//---------------------------------------------------------------------------------------------//
AJC_DLGPROC(Main, IDC_TXT       ) // IDC_TXT コントロールのアクション (WM_COMMAND, LOWORD(wParam)== IDC_TXT)
{
    ・・・・・
    return TRUE;
}
//---------------------------------------------------------------------------------------------//
AJC_DLGMAP_DEF(Main)
    AJC_DLGMAP_MSG(Main, WM_COMMAND    )
    AJC_DLGMAP_CMD(Main, IDC_TXT       )
AJC_DLGMAP_END
```

「AJC_DLGMAP_MSG」で「WM_COMMAND」が指定された場合は、「AJC_DLGMAP_CMD」で指定されたコントロール群のいずれのＩＤとも一致しない場合のみ WM_COMMAND のアクション関数が実行されます。
上記例では「IDC_TXT」のアクション関数を実行した場合は、WM_COMMAND のアクション関数は実行されません。
また、「IDC_TXT」以外の WM_COMMAND メッセージの場合は、全て WM_COMMAND のアクション関数が実行されます。

上記例は、WM_COMMAND に関する例を示していますが、WM_NOTIFY に関しても同様です。
つまり、「AJC_DLGMAP_MSG」で「WM_NOTIFY」が指定された場合は、「AJC_DLGMAP_NTC」で指定されたコントロール群のいずれのＩＤとも一致しない場合のみ WM_NOTIFY のアクション関数が実行されます。

3.3. 動的なメッセージコードのメッセージマップ

他のウインドとメッセージコードが重複しないように、独自の動的なメッセージコードを使用してメッセージの送受信を行うことができます。
動的なメッセージコードでメッセージを送信する場合は、RegisterWindowMessage()関数により動的に取得したメッセージコードを使用します。

動的なメッセージコードでメッセージを送信する場合は、通常、以下のようなコードとなります。

```
UI msg = RegisterWindowMessage(TEXT("XXXXX"));
・・・・・
PostMessage(HWND_BROADCAST, msg, wParam, lParam); //（あるいは、SendMessage(HWND_BROADCAST, msg, wParam, lParam);
```

一方、動的なメッセージコードでメッセージを受信するプログラムでは、同じ文字定数を指定した「RegisterWindowMessage("XXXXX");」により動的に取得したメッセージコードに反応しなければなりません。
本ライブラリでは、動的なメッセージコードに対応した以下のマクロを用意しています。

マクロ形式	内容
AJC_WNDMAP_RWM(**WND, MSG, TEXT("*XXXXX*")**)	動的メッセージのメッセージマップテーブルのエントリ定義（通常のウインド用）
AJC_DLGMAP_RWM(**DLG, MSG, TEXT("*XXXXX*")**)	動的メッセージのメッセージマップテーブルのエントリ定義（ダイアログボックス用）

上記２つのマクロは、前述した「AJC_WNDMAP_MSG」や「AJC_DLGMAP_MSG」マクロと同様に使用しますが、実際のメッセージコードはTEXT(**"*XXXXX*"**) を引数とした 「RegisterWindowMessage(TEXT(**"*XXXXX*"**))」により動的に取得したメッセージコードとなります。
また、「**MSG**」で指定するローカル・メッセージコードの名称は、#defne 等で定義する必要はありません。（動的なメッセージコードなので、定数として定義することはできません）

動的メッセージコードに関するサンプルプログラムを SW_Broadcast1～3 に示します。

3.4. ウインドのサブクラス化

ウインドをサブクラス化する場合は、以下のようにします。（多重にサブクラス化することも可）

サブクラス化したウインドプロシージャ

```
AJC_WNDMAP_DEF( SUB )
   ・
   ・
   ・
AJC_WNDMAP_END
```

サブクラス設定

```
MAjcMmpSetSubclass( SUB , hwnd );
```

※ *hwnd* ： サブクラス化するウインドのハンド

サブクラス解除

```
MAjcMmpClrSubclass( SUB , hwnd);
```

MAjcMmpSetSubclass()マクロは、サブクラス化を成功した場合はオリジナル・ウインドプロシージャのアドレスを返します。
サブクラス化を失敗した場合は NULL を返します。
MAjcMmpClrSubclass()マクロはサブクラス化解除を成功した場合は TRUE を，失敗した場合は FALSE を返します。

サブクラス化したウインドプロシージャでは、MAjcMmpCallOrgWndProc()/MAjcMmpCallOrgWndProEx()マクロによりオリジナル・ウインドプロシージャを呼び出すようにします。
(MAjcMmpCallOrgWndProc()マクロは、AJC_WNDPROC()マクロで定義したメッセージ処理関数内で記述しなければなりません)

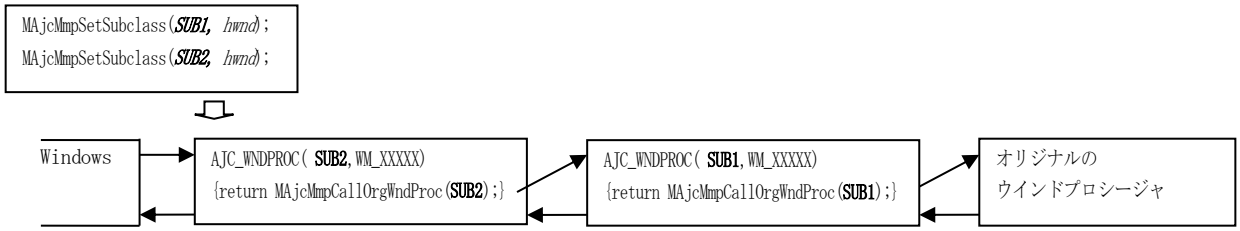
サブクラス化したウインドプロシージャのメッセージ処理

```
AJC_WNDPROC( SUB ,WM_RBUTTONDOWN )
{
   ・
   ・
   ・
  // オリジナルのウインドプロシージャ呼び出し
  return MAjcMmpCallOrgWndProc(SUB);
}
```

```
AJC_WNDPROC( SUB ,WM_RBUTTONDOWN )
{
   ・
   ・
   ・
  // オリジナルのウインドプロシージャ呼び出し
  return MAjcMmpCallOrgWndProcEx(SUB, hwnd, msg, wParam, lParam);
}
```

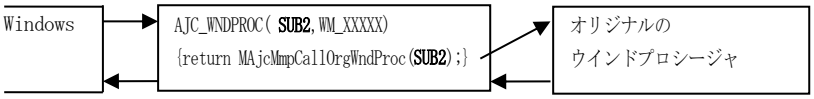
多重にサブクラスした場合は、最後にサブクラスしたプロシージャがメッセージを受けて、順次、前回にサブクラス化したプロシージャが実行されます。

例えば、SUB1，SUB2 で多重にサブクラスした場合、メッセージの流れは、以下のようになります。



サブクラス化する順番と、サブクラスを解除する順番に制限はありません。いきなり途中のサブクラスを解除しても問題ありません。

上記例から、**SUB1** をサブクラス解除「MAjcMmpClrSubclass(**SUB1,** hwnd);」した場合は、以下のようになります。



MAjcMmpCallOrgWndProc()は、MAjcMmpSetSubclass()でサブクラス化した際のオリジナル・ウインドプロシージャを呼び出します。

サブクラス化したウインドプロシージャで、処理されないメッセージやコントロール（AJC_WNDMAP_DEF()～AJC_WNDMAP_END で記述されていないメッセージやコントロール）は、サブクラス化前のオリジナルのウインドプロシージャを呼び出します。

但し、MAjcMmpSetSubclass()マクロの後に、MAjcMmpSetDefWndProc()マクロを実行した場合は、システムＡＰＩの DefWindowProc()を呼び出します。（MAjcMmpSetDefWndProc()については、すぐ後の「メッセージのインターセプト」を参照）

同一ウインドに複数の異なるウインドプロシージャを多重にサブクラス化することはできますが、同一ウインドに同一ウインドプロシージャを複数回サブクラス化することはできません。

※OK

同一ウインド(hWndMain)に異なるウインドプロシージャ(SUB1，SUB2)を多重にサブクラス化

```
HWND  hWndMain;
   ・・・
MAjcMmpSetSubclass( SUB1, hWndMain );
MAjcMmpSetSubclass( SUB2, hWndMain );
   ・・・
AJC_WNDMAP_DEF(SUB1)
   ・・・
AJC_WNDMAP_END

AJC_WNDMAP_DEF(SUB2)
   ・・・
AJC_WNDMAP_END
```

※NG

同一ウインド(hWndMain)に同一ウインドプロシージャ(SUB)を多重にサブクラス化

```
HWND  hWndMain;
   ・・・
MAjcMmpSetSubclass( SUB, hWndMain );
MAjcMmpSetSubclass( SUB, hWndMain );
   ・・・
AJC_WNDMAP_DEF(SUB)
   ・・・
AJC_WNDMAP_END
```

また、同一ウインドに異なるウインドプロシージャを多重にサブクラス化する場合でも、ウインドプロシージャの識別子は異なる名称を指定しなければなりません。

a.c

```
HWND  hWndMain;
   ・・・
MAjcMmpSetSubclass( SUB , hWndMain );
   ・・・
AJC_WNDMAP_DEF(SUB)
   ・・・
AJC_WNDMAP_END
```

b.c

```
extern HWND  hWndMain;
   ・・・
MAjcMmpSetSubclass( SUB ,
hWndMain );
   ・・・
AJC_WNDMAP_DEF(SUB)
   ・・・
AJC_WNDMAP_END
```

⇨

※ＮＧ

コンパイル単位が別なので同じウインドプロシージャの識別子(SUB)は使用可能だが、同一ウインド(hWndMain)を多重サブクラス化する場合は、同一ウインドプロシージャの識別子(SUB)は使用できない。

→SUB1 と SUB2 等、別々の名称ならばＯＫ

多重にサブクラス化した場合の例をサンプルプログラム(SW_SubClass1)に、複数のウインドを１つのウインドプロシージャでサブクラス化する例をサンプルプログラム(SW_SubClass2)に示します。

同じウインドプロシージャの識別子を指定した場合、MAjcMmpSetSubclass()マクロは NULL を返します。

メッセージのインターセプト（横取り）

メッセージを完全にインターセプトし、サブクラス化前の（オリジナルの）ウインドプロシージャを実行しないようにするには、以下のようにします。

1）AJC_WNDPROC()で処理されないメッセージの場合、DefWindowProc()を呼び出すようにする。
DefWindowProc()を呼び出すようにするには、MAjcMmpSetSubclass()マクロの後に、MAjcMmpSetDefWndProc()マクロを実行します。

```
MAjcMmpSetSubclass ( SUB, hwnd );
MAjcMmpSetDefWndProc(SUB, hwnd);
```

2）各メッセージ処理関数で、MAjcMmpCallOrgWndProc()を呼び出さずに、DefWindowProc()を呼び出すか、直接 return する。

```
AJC_WNDPROC( SUB , WM_RBUTTONDOWN )
{
   ・
   ・
   ・
  // メッセージをインターセプトする場合は、、DefWindowProc()を呼び出すか、
  return DefWindowProc(hwnd, msg, wParam, lParam);

  // あるいは、直接 return する
  return 0;
}
```

メッセージをインターセプトした場合の例をサンプルプログラム(SW_SubClass3)に示します。

3.4.1. サブクラス化後の UNICODE ウインド属性

MAjcMmpSetSubclass()でサブクラス化後も、ウインドの UNICODE/ANSI 属性は変化しません。
MAjcMmpSetSubclass()でサブクラス化しても、UNICODE ウインドは UUNICODE 属性のままで、ANSI ウインドは ANSI 属性を維持します。
サンプルプログラム(SW_SubClass1)では、サブクラス化前後のウインド属性を「下のボタンは *XXXXX* Window です」と表示します。

※ウインドの UNICODE/ANSI 属性は、システムの RegisterClass()ＡＰＩに依存します。
RegisterClass**A**()で登録されたウインドクラスは ANSI 属性となり、RegisterClass**W**()で登録されたウインドクラスは UNICODE 属性となります。
システムのウインドクラス（BUTTON や LISTBOX 等）の UNICODE/ANSI 属性は、UNICODE コンパイルオプションに依存します。

3.4.2. サブクラス化によるテキストメッセージの受け渡し

サブクラス化したウインドプロシージャで、メッセージをトラップしテキストを受け取る際は少々注意が必要です。
システムで定義済みメッセージの場合、テキストデータはウインドの UNICODE/ANSI 属性に従います。
例えば、ボタンコントロールに WM_SETTEXT でテキストデータを送ると、サブクラス化したウインドプロシージャでは WM_SETTEXT の lParam に以下の文字列へのポインタが渡されます。

メッセージ送信元	サブクラス化したウインドプロシージャで受ける WM_SETTEXT メッセージの lParam	
	ANSI ウインドの場合	UNICODE ウインドの場合
SendMessageA(hwnd, WM_SETTEXT, 0, “ABC”);	マルチバイト文字列へのポインタ (”ABC”)	UNICODE 文字列へのポインタ(L”ABC”)
SendMessageW(hwnd, WM_SETTEXT, 0, L“ABC”);		

全て検証したわけではありませんが、WM_SETTEXT に限らず、LB_INSERTSTRING 等の他のシステム定義済みメッセージでも同様と思われます。

また、(当然ですが) ユーザ独自のメッセージを使用し、メッセージを送信した場合は、ウインドの UNICODE/ANSI 属性にかかわらず、送信元のテキストデータがそのまま渡されます。

メッセージ送信元	サブクラス化したウインドプロシージャで受ける WM_APP メッセージの lParam	
	ANSI ウインドの場合	UNICODE ウインドの場合
SendMessageA(hwnd, WM_APP, 0, “ABC”);	マルチバイト文字列へのポインタ (”ABC”)	
SendMessageW(hwnd, WM_APP, 0, L“ABC”);	UNICODE 文字列へのポインタ (L”ABC”)	

一方、テキスト取得系のシステム定義済のメッセージでは、取得するテキストはウインドの UNICODE/ANSI 属性に従います。
（メッセージをトラップせずに、システムのデフォルトウインドプロシージャが実行された場合）

メッセージ送信元	メッセージ送信元で受け取るテキスト (buf) の内容	
	ANSI ウインドの場合	UNICODE ウインドの場合
UT buf[N]; SendMessageA(hwnd, WM_GETTEXT, N, buf);	マルチバイト文字列(”*XXXXX*”)	UNICODE 文字列 (L”*XXXXX*”)
UT buf[N]; SendMessageW(hwnd, WM_GETTEXT, N, buf);		

また、(これも当然ですが) サブクラス化したウインドプロシージャがメッセージをトラップする場合は、ウインドプロシージャが指定されたバッファに何を格納するかに依存します。

※SendMessage()以外のシステムＡＰＩ (SetWindowText()や GetWindowText()等) でも同様となります。
これらのＡＰＩは内部で SendMessage()が実行されるものと思われます。

※サブクラス化によるテキストメッセージの受け渡しは、サンプルプログラム(SW_SubClass1)にテストコードを実装しています。

3.5. サポートＡＰＩ

サブクラス化に関するＡＰＩ一覧を以下に示します。

#	関数名	内容	備考
1	MAjcMmpSetSubclass	ウインドプロシージャのサブクラス化	マクロ
2	MAjcMmpClrSubclass	ウインドプロシージャのサブクラス化解除	
3	MAjcMmpDelSubclasses	ウインドプロシージャのサブクラス化を全て解除	
4	MAjcMmpSetDefWndProc	デフォルトウインドプロシージャの設定	
5	MAjcMmpCallOrgWndProc[Ex]	オリジナルウインドプロシージャの呼び出し	
6	AjcMmpGetSubclassInfo	サブクラス化情報の取得	

3.5.1. ウインドプロシージャのサブクラス化(MAjcMmpSetSubclass)

形　式 ： WNDPROC　MAjcMmpSetSubclass (NAME, hWnd)；　(マクロ)

引　数 ： NAME　　- ウインドプロシージャの識別子(AJC_WNDMAP_DEF()マクロで指定した識別子)
hWnd　　- サブクラス化するウインドのハンドル(HWND)

説　明 ： hWnd で指定したウインドをサブクラス化します。
AJC_WNDMAP_DEF(**NAME**)～AJC_WNDMAP_END で定義したウインドプロシージャが、サブクラス化したウインドプロシージャとなります。

戻り値 ： ≠NULL： 成功（元のウインドプロシージャへのポインタ）
＝NULL： 失敗（同一ウインドで既に「NAME」と同一名称でサブクラス化している場合も失敗します）

3.5.2. ウインドプロシージャのサブクラス解除(MAjcMmpClrSubclass)

形　式 ： BOOL　MAjcMmpClrSubclass (NAME, hWnd)；　(マクロ)

引　数 ： NAME　- ウインドプロシージャの識別子(AJC_WNDMAP_DEF()マクロで指定した識別子)
hWnd　- サブクラスを解除するウインドのハンドル(HWND)

説　明 ： hWnd で指定したウインドのサブクラスを解除します。
サブクラス化前のウインドプロシージャが設定されます。

戻り値 ： TRUE ： 成功
FALSE： 失敗

3.5.3. ウインドプロシージャのサブクラスを全て解除(MAjcMmpDelSubclasses)

形　式 ： BOOL　MAjcMmpDelSubclasses (NAME)；　(マクロ)

引　数 ： NAME　- ウインドプロシージャの識別子(AJC_WNDMAP_DEF()マクロで指定した識別子)
hWnd　- サブクラスを解除するウインドのハンドル(HWND)

説　明 ： hWnd で指定したウインドのサブクラスを解除します。
サブクラス化前のウインドプロシージャが設定されます。

戻り値 ： TRUE ： 成功
FALSE： 失敗

3.5.4. デフォルトウインドプロシージャの設定(MAjcMmpSetDefWndProc)

形　式 ： BOOL MAjcMmpSetDefWndProc (NAME, hWnd)； （マクロ）

引　数 ： NAME － ウインドプロシージャの識別子(AJC_WNDMAP_DEF()マクロで指定した識別子)
hWnd － サブクラス化するウインドのハンドル(HWND)

説　明 ： ウインドプロシージャ内で、MAjcMmpCallOrgWndProc[Ex]()を実行した場合、システムＡＰＩの DefWndProc()が呼び出されるように設定します。

戻り値 ： TRUE ： 成功
FALSE： 失敗

3.5.5. オリジナルウインドプロシージャの呼び出し(AjcMmpSetSubclass)

形　式 ： LRESULT MAjcMmpCallOrgWndProc (NAME)； （マクロ）

引　数 ： NAME － ウインドプロシージャの識別子(AJC_WNDMAP_DEF()マクロで指定した識別子)

説　明 ： サブクラス化する前のウインドプロシージャを呼び出します。
但し、MAjcMmpSetDefWndProc()が実行されている場合は、システムＡＰＩの DefWndProc()を呼び出します。
このマクロは、AJC_WNDPROC()マクロで定義したメッセージハンドラ内で記述しなければなりません。

戻り値 ： 呼び出したウインドプロシージャの戻り値

3.5.6. サブクラス化情報の取得(AjcMmpGetSubclassInfo)

形　式 ： UI AjcMmpGetSubclassInfo (HWND hwnd, AJCSBCINFO buf[], UI nBuf)；

引　数 ： hwnd － サブクラス化情報を取得するウインドのハンドル
buf － サブクラス化情報を格納する配列バッファのアドレス（不要時は NULL）
nBuf － 取得するサブクラス化情報の個数（多重数）

説　明 ： pBuf で示すバッファにサブクラス化情報を格納します。
サブクラス化情報（AJCSBCINFO 構造体）の要素は以下の通りです。

#	名称	タイプ	内容
1	pName	C_BCP	プロシージャの名前（AJC_WNDMAP_DEF(name)で指定した名称，バイト文字列）へのポインタ
2	wpNew	WNDPROC	サブクラス化した新ウインドプロシージャ
3	wpOld	WNDPROC	サブクラス化前の元ウインドプロシージャ
4	wpRet	WNDPROC	MAjcMmpCallOrgWndProc[Ex]()で呼び出されるウインドプロシージャ （通常は wpOld と同じだが、MAjcMmpSetDefWndProc()が実行された場合は、DefWndProc()を示す）

buf[0]に最後にサブクラス化したプロシージャの情報が格納され、以降、１つ前にサブクラス化した情報が格納されます。
buf=NULL を指定した場合は、単にサブクラス化の多重数を返します。
nBuf は取得するサブクラス化情報の個数（サブクラスの多重数）を指定します。(buf=NULL 時は無視される)

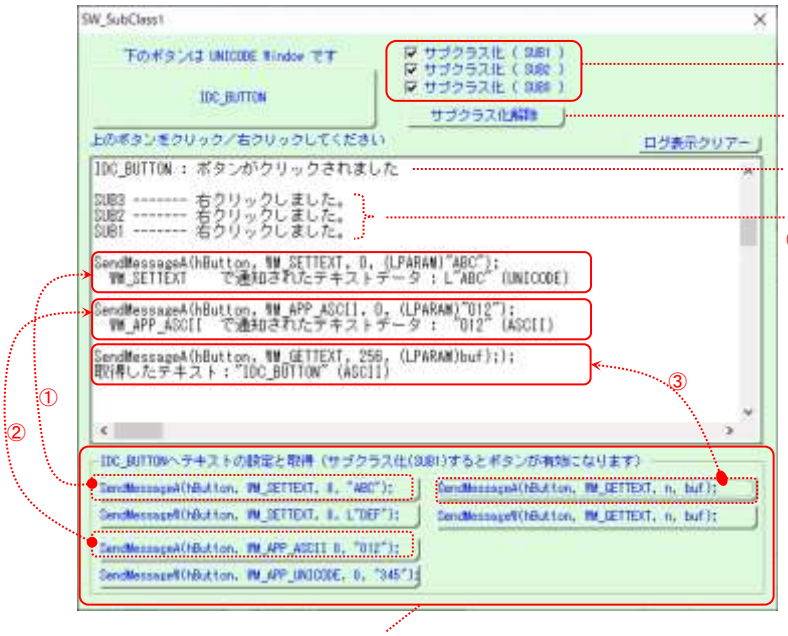
戻り値 ： サブクラス化の多重数（０：サブクラス化無し）

備　考 ： サブクラス化の多重数（最大値）が不明の場合は、最初に buf=NULL を指定し、取得した多重数を nBuf に指定してサブクラス化情報を取得してください。

3.6. サンプルプログラム

3.6.1. SW_SubClass1（多重にサブクラス化）

以下のサンプルプログラムは、ボタン(IDC_BUTTON)をサブクラス化し、右クリックしたら、右クリックした旨をログ表示します。
サブクラス化は、SUB1, SUB2, SUB3(順不同)で最大３重にサブクラス化します。
サブクラス化したプロシージャは、最後にサブクラス化したプロシージャから芋づる式にサブクラス化したウインドプロシージャが実行されます。
ボタンを左クリックした場合は、通常のボタンイベント(BN_CLICKED)が発生した旨をログ表示します。



チェックでサブクラス化し、
チェック解除でサブクラスを解除する

全てのサブクラス化を解除

ボタン左クリック時のログ表示

ボタン右クリック時のログ表示
（サブクラス化したウインドプロシージャからの出力）

ウインドへ送られるテキストの形式(ASCII/UNICODE)と、ウインドから取得されるテキストの形式(ASCII/UNICODE)をテストする。
例えば、
①: UNICODE ウインドへの WM_SETTEXT は、UNICODE に変換されて通知される。（システムが送信先ウインドの属性に合わせて変換する）
②: UNICODE ウインドへの WM_APP・・は、UNICODE に変換されない。
ユーザメッセージの場合は、システムは何も変換しない。このプログラムではメッセージコードを分けてテキスト形式を明示しています。
③: SendMessage**A**(WM_GETTEXT)で取得したテキストは ASCII テキストで取得される。（SendMessage**W**(WM_GETTEXT) の場合は UNICODE で取得される）

```
 1 : //
 2 : //  SW_SubClass1.c
 3 : //
 4 : #include    <AjrCstXX.h>
 5 : #include    <tchar.h>
 6 : #include    "resource.h"
 7 :
 8 : #define     WM_APP_ASCII    (WM_APP + 0)
 9 : #define     WM_APP_UNICODE  (WM_APP + 1)
10 :
11 : //----- 作業領域 ------------------------------------------------------------------------------------------//
12 : static  HINSTANCE       hInst;
13 : static  HWND            hDlgMain;
14 : static  HWND            hButton;
15 : static  HWND            hChk1, hChk2, hChk3;
16 : static  HWND            hVth;
17 : static  UB              fSubClass = 0x00;
18 :
19 : //----- 内部サブ関数 --------------------------------------------------------------------------------------//
20 : AJC_DLGPROC_DEF(Main);
21 : AJC_WNDPROC_DEF(SUB1);
22 : AJC_WNDPROC_DEF(SUB2);
23 : AJC_WNDPROC_DEF(SUB3);
24 : static  VO  ShowButtonFace(HWND hDlg);
25 : static  VO  ShowSubclassInfo(VO);
26 :
27 : //-------------------------------------------------------------------------------------------------------//
28 : int WINAPI AjcWinMain(HINSTANCE hInstance, HINSTANCE hinstPrev, UTP szCmdLine, int iCmdShow)
29 : {
30 :     MSG     msg;
31 :
```

```
 32 :     hInst = hInstance;
 33 :
 34 :     AjcDgcSetup();
 35 :
 36 :     //----- メイン・ダイアログオープン -----------------//
 37 :     hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_MAIN), NULL, AJC_DLGPROC_NAME(Main));
 38 :     ShowWindow(hDlgMain, SW_SHOW);
 39 :
 40 :     //----- メッセージループ ---------------------------//
 41 :     while (GetMessage(&msg, NULL, 0, 0)) {
 42 :         do {
 43 :             if (IsDialogMessage(hDlgMain, &msg)) break;
 44 :             TranslateMessage(&msg);
 45 :             DispatchMessage (&msg);
 46 :         } while (0);
 47 :     }
 48 :
 49 :     return (int)msg.wParam ;
 50 : }
 51 : //=============================================================================================================//
 52 : //                                                                                                             //
 53 : //  ダイアログ・プロシージャ                                                                                   //
 54 : //                                                                                                             //
 55 : //=============================================================================================================//
 56 : //----- ダイアログ初期化 -------------------------------------------------------------------------------------//
 57 : AJC_DLGPROC(Main, WM_INITDIALOG     )
 58 : {
 59 :     hDlgMain = hDlg;
 60 :     hButton  = GetDlgItem(hDlg, IDC_BUTTON);
 61 :     hVth     = GetDlgItem(hDlg, IDC_VTH);
 62 :     hChk1    = GetDlgItem(hDlg, IDC_CHK_SUB1);
 63 :     hChk2    = GetDlgItem(hDlg, IDC_CHK_SUB2);
 64 :     hChk3    = GetDlgItem(hDlg, IDC_CHK_SUB3);
 65 :
 66 :     ShowButtonFace(hDlg);
 67 :
 68 :     return TRUE;
 69 : }
 70 : //----- ウインド破棄 -----------------------------------------------------------------------------------------//
 71 : AJC_DLGPROC(Main, WM_TIMER      )
 72 : {
 73 :     KillTimer(hDlg, 1);
 74 :     AjcVthPrintF(hVth, TEXT("¥n"));
 75 :     return TRUE;
 76 : }
 77 : //----- タイマ -----------------------------------------------------------------------------------------------//
 78 : AJC_DLGPROC(Main, WM_DESTROY        )
 79 : {
 80 :     PostQuitMessage(0);
 81 :     return TRUE;
 82 : }
 83 : //----- IDC_BUTTON -------------------------------------------------------------------------------------------//
 84 : AJC_DLGPROC(Main, IDC_BUTTON        )
 85 : {
 86 :     if (HIWORD(wParam) == BN_CLICKED) {
 87 :         AjcVthPrintF(hVth, TEXT("¥nIDC_BUTTON : ボタン・クリック・イベント(BN_CLICKED)が発生しました¥n¥n"));
 88 :     }
 89 :     return TRUE;
 90 : }
 91 : //----- サブクラス化 ( SUB1 ) --------------------------------------------------------------------------------//
 92 : AJC_DLGPROC(Main, IDC_CHK_SUB1      )
 93 : {
 94 :     if (AjcGetDlgItemChk(hDlg, IDC_CHK_SUB1)) {
 95 :         MAjcMmpSetSubclass(SUB1, hButton);
 96 :         AjcVthPrintF(hVth, TEXT("SUB1 - Subclassed¥n"));
 97 :         AjcEnableDlgGroup(hDlg, IDC_GRP_TEST, TRUE, TRUE);
 98 :     }
 99 :     else {
100 :         MAjcMmpClrSubclass(SUB1, hButton);
101 :         AjcVthPrintF(hVth, TEXT("SUB1 - Un-Subclassed¥n"));
102 :         AjcEnableDlgGroup(hDlg, IDC_GRP_TEST, TRUE, FALSE);
103 :     }
104 :     ShowSubclassInfo();
105 :     return TRUE;
106 : }
107 : //----- サブクラス化 ( SUB2 ) --------------------------------------------------------------------------------//
108 : AJC_DLGPROC(Main, IDC_CHK_SUB2      )
109 : {
110 :     if (AjcGetDlgItemChk(hDlg, IDC_CHK_SUB2)) {
111 :         MAjcMmpSetSubclass(SUB2, hButton);
112 :         AjcVthPrintF(hVth, TEXT("SUB2 - Subclassed¥n"));
113 :     }
114 :     else {
115 :         MAjcMmpClrSubclass(SUB2, hButton);
116 :         AjcVthPrintF(hVth, TEXT("SUB2 - Un-Subclassed¥n"));
117 :     }
118 :     ShowSubclassInfo();
119 :     return TRUE;
120 : }
121 : //----- サブクラス化 ( SUB3 ) --------------------------------------------------------------------------------//
```

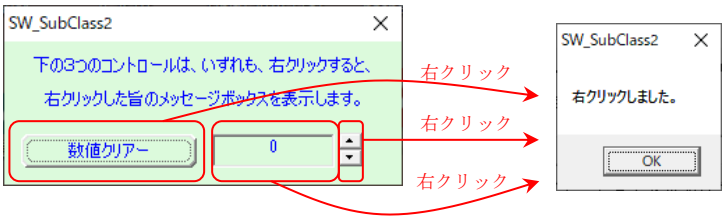
```
122 : AJC_DLGPROC(Main, IDC_CHK_SUB3      )
123 : {
124 :     if (AjcGetDlgItemChk(hDlg, IDC_CHK_SUB3)) {
125 :         MAjcMmpSetSubclass(SUB3, hButton);
126 :         AjcVthPrintF(hVth, TEXT("SUB3 - Subclassed¥n"));
127 :     }
128 :     else {
129 :         MAjcMmpClrSubclass(SUB3, hButton);
130 :         AjcVthPrintF(hVth, TEXT("SUB3 - Un-Subclassed¥n"));
131 :     }
132 :     ShowSubclassInfo();
133 :     return TRUE;
134 : }
135 : //----- サブクラス化解除ボタン ----------------------------------------------------------------------------------//
136 : AJC_DLGPROC(Main, IDC_CMD_CLRSUB    )
137 : {
138 :     MAjcMmpClrSubclass(SUB1, hButton);
139 :     MAjcMmpClrSubclass(SUB2, hButton);
140 :     MAjcMmpClrSubclass(SUB3, hButton);
141 :     AjcSetDlgItemChk(hDlg, IDC_CHK_SUB1, FALSE);
142 :     AjcSetDlgItemChk(hDlg, IDC_CHK_SUB2, FALSE);
143 :     AjcSetDlgItemChk(hDlg, IDC_CHK_SUB3, FALSE);
144 :     ShowSubclassInfo();
145 :     return TRUE;
146 : }
147 : //----- ログ表示クリアーボタン ----------------------------------------------------------------------------------//
148 : AJC_DLGPROC(Main, IDC_CMD_CLRLOG    )
149 : {
150 :     AjcVthClear(hVth);
151 :     return TRUE;
152 : }
153 : //----- キャンセル ----------------------------------------------------------------------------------------------//
154 : AJC_DLGPROC(Main, IDCANCEL          )
155 : {
156 :     DestroyWindow(hDlg);
157 :     return TRUE;
158 : }
159 : //----- SendMessageA(hButton, WM_SETTEXT, 0, (LPARAM)"ABC");実行 -------------------------------------------//
160 : AJC_DLGPROC(Main, IDC_CMD_SETA  )
161 : {
162 :     AjcVthPrintF(hVth, TEXT("SendMessageA(hButton, WM_SETTEXT, 0, (LPARAM)¥"ABC¥");¥n"));
163 :     SendMessageA(hButton, WM_SETTEXT, 0, (LPARAM)"ABC");
164 :     return TRUE;
165 : }
166 : //----- SendMessageW(hButton, WM_SETTEXT, 0, (LPARAM)L"DEF"); -------------------------------------------//
167 : AJC_DLGPROC(Main, IDC_CMD_SETW  )
168 : {
169 :     AjcVthPrintF(hVth, TEXT("SendMessageW(hButton, WM_SETTEXT, 0, (LPARAM)L¥"DEF¥");¥n"));
170 :     SendMessageW(hButton, WM_SETTEXT, 0, (LPARAM)L"DEF");
171 :     return TRUE;
172 : }
173 : //----- SendMessageA(hButton, WM_APP, 0, (LPARAM)"012");実行 ---------------------------------------------//
174 : AJC_DLGPROC(Main, IDC_CMD_USRA  )
175 : {
176 :     AjcVthPrintF(hVth, TEXT("SendMessageA(hButton, WM_APP_ASCII, 0, (LPARAM)¥"012¥");¥n"));
177 :     SendMessageA(hButton, WM_APP_ASCII, 0, (LPARAM)"012");
178 :     return TRUE;
179 : }
180 : //----- SendMessageW(hButton, WM_APP, 0, (LPARAM)"345");実行 ---------------------------------------------//
181 : AJC_DLGPROC(Main, IDC_CMD_USRW  )
182 : {
183 :     AjcVthPrintF(hVth, TEXT("SendMessageW(hButton, WM_APP_UNICODE, 0, (LPARAM)L¥"345¥");¥n"));
184 :     SendMessageW(hButton, WM_APP_UNICODE, 0, (LPARAM)L"345");
185 :     return TRUE;
186 : }
187 : //----- SendMessageA(hButton, WM_GETTEXT, n, (LPARAM)buf);実行 --^^-----------------------------------------//
188 : AJC_DLGPROC(Main, IDC_CMD_GETA  )
189 : {
190 :     BC  buf[256];
191 :     AjcVthPrintF(hVth, TEXT("SendMessageA(hButton, WM_GETTEXT, 256, (LPARAM)buf););¥n"));
192 :     SendMessageA(hButton, WM_GETTEXT, 256, (LPARAM)buf);
193 :     AjcVthPrintFA(hVth,  "取得したテキスト：¥"%s¥" (ASCII)¥n¥n", buf);
194 :     return TRUE;
195 : }
196 : //----- SendMessageW(hButton, WM_SETTEXT, n, (LPARAM)buf); -----------------------------------------------//
197 : AJC_DLGPROC(Main, IDC_CMD_GETW  )
198 : {
199 :     WC  buf[256];
200 :     AjcVthPrintF(hVth, TEXT("SendMessageW(hButton, WM_GETTEXT, 256, (LPARAM)buf);¥n"));
201 :     SendMessageW(hButton, WM_GETTEXT, 256, (LPARAM)buf);
202 :     AjcVthPrintFW(hVth,  L"取得したテキスト：L¥"%s¥" (UNICODE)¥n¥n", buf);
203 :     return TRUE;
204 : }
205 : //-------------------------------------------------------------------------------------------------------------//
206 : AJC_DLGMAP_DEF(Main)
207 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG     )
208 :     AJC_DLGMAP_MSG(Main, WM_DESTROY        )
209 :     AJC_DLGMAP_MSG(Main, WM_TIMER          )
210 :     AJC_DLGMAP_CMD(Main, IDC_BUTTON        )
211 :     AJC_DLGMAP_CMD(Main, IDC_CHK_SUB1      )
```

```
212 :     AJC_DLGMAP_CMD(Main, IDC_CHK_SUB2      )
213 :     AJC_DLGMAP_CMD(Main, IDC_CHK_SUB3      )
214 :     AJC_DLGMAP_CMD(Main, IDC_CMD_CLRSUB    )
215 :     AJC_DLGMAP_CMD(Main, IDC_CMD_CLRLOG    )
216 :     AJC_DLGMAP_CMD(Main, IDCANCEL          )
217 :     AJC_DLGMAP_CMD(Main, IDC_CMD_SETA      )
218 :     AJC_DLGMAP_CMD(Main, IDC_CMD_SETW      )
219 :     AJC_DLGMAP_CMD(Main, IDC_CMD_USRA      )
220 :     AJC_DLGMAP_CMD(Main, IDC_CMD_USRW      )
221 :     AJC_DLGMAP_CMD(Main, IDC_CMD_GETA      )
222 :     AJC_DLGMAP_CMD(Main, IDC_CMD_GETW      )
223 : AJC_DLGMAP_END
224 : //-------------------------------------------------------------------------------------------------------------------//
225 : //  サブクラス１                                                                                                     //
226 : //-------------------------------------------------------------------------------------------------------------------//
227 : //----- WM_RBUTTONDOWN ----------------------------------------------------------------------------------------------//
228 : AJC_WNDPROC(SUB1, WM_RBUTTONDOWN            )
229 : {
230 :     AjcVthPrintF(hVth, TEXT("SUB1 ------- 右クリックしました。¥n"));
231 :     SetTimer(hDlgMain, 1, 10, NULL);
232 :     return MAjcMmpCallOrgWndProc(SUB1);
233 : }
234 : //----- WM_SETTEXT --------------------------------------------------------------------------------------------------//
235 : AJC_WNDPROC(SUB1, WM_SETTEXT                )
236 : {
237 : #ifdef UNICODE
238 :     AjcVthPrintFW(hVth, L"  WM_SETTEXT    で通知されたテキストデータ : L¥"%s¥" (UNICODE)¥n¥n", lParam);
239 : #else
240 :     AjcVthPrintFA(hVth,  "  WM_SETTEXT    で通知されたテキストデータ :  ¥"%s¥" (ASCII)¥n¥n"  , lParam);
241 : #endif
242 :     return 0;
243 : }
244 : //----- WM_APP_ASCII ------------------------------------------------------------------------------------------------//
245 : AJC_WNDPROC(SUB1, WM_APP_ASCII              )
246 : {
247 :     AjcVthPrintFA(hVth,  "  WM_APP_ASCII  で通知されたテキストデータ :  ¥"%s¥" (ASCII)¥n¥n", lParam);
248 :     return 0;
249 : }
250 : //----- WM_APP_UNICODE ----------------------------------------------------------------------------------------------//
251 : AJC_WNDPROC(SUB1, WM_APP_UNICODE            )
252 : {
253 :     AjcVthPrintFW(hVth, L"  WM_APP_UNICODE で通知されたテキストデータ : L¥"%s¥" (UNICODE)¥n¥n", lParam);
254 :     return 0;
255 : }
256 : //-------------------------------------------------------------------------------------------------------------------//
257 : AJC_WNDMAP_DEF(SUB1)
258 :     AJC_WNDMAP_MSG(SUB1, WM_RBUTTONDOWN    )
259 :     AJC_WNDMAP_MSG(SUB1, WM_SETTEXT        )
260 :     AJC_WNDMAP_MSG(SUB1, WM_APP_ASCII      )
261 :     AJC_WNDMAP_MSG(SUB1, WM_APP_UNICODE    )
262 : AJC_WNDMAP_END
263 :
264 : //-------------------------------------------------------------------------------------------------------------------//
265 : //  サブクラス２                                                                                                     //
266 : //-------------------------------------------------------------------------------------------------------------------//
267 : AJC_WNDPROC(SUB2, WM_RBUTTONDOWN            )
268 : {
269 :     AjcVthPrintF(hVth, TEXT("SUB2 ------- 右クリックしました。¥n"));
270 :     SetTimer(hDlgMain, 1, 10, NULL);
271 :     return MAjcMmpCallOrgWndProc(SUB2);
272 : }
273 : //-------------------------------------------------------------------------------------------------------------------//
274 : AJC_WNDMAP_DEF(SUB2)
275 :     AJC_WNDMAP_MSG(SUB2, WM_RBUTTONDOWN    )
276 : AJC_WNDMAP_END
277 :
278 : //-------------------------------------------------------------------------------------------------------------------//
279 : //  サブクラス３                                                                                                     //
280 : //-------------------------------------------------------------------------------------------------------------------//
281 : AJC_WNDPROC(SUB3, WM_RBUTTONDOWN            )
282 : {
283 :     AjcVthPrintF(hVth, TEXT("SUB3 ------- 右クリックしました。¥n"));
284 :     SetTimer(hDlgMain, 1, 10, NULL);
285 :     return MAjcMmpCallOrgWndProc(SUB3);
286 : }
287 : //-------------------------------------------------------------------------------------------------------------------//
288 : AJC_WNDMAP_DEF(SUB3)
289 :     AJC_WNDMAP_MSG(SUB3, WM_RBUTTONDOWN    )
290 : AJC_WNDMAP_END
291 :
292 : //-------------------------------------------------------------------------------------------------------------------//
293 : //  ボタンフェース表示                                                                                               //
294 : //-------------------------------------------------------------------------------------------------------------------//
295 : static  VO  ShowButtonFace(HWND hDlg)
296 : {
297 :     if (IsWindowUnicode(hButton)) AjcSetDlgItemStr(hDlg, IDC_LBL_CHARTYPE, TEXT("下のボタンは UNICODE Window です"));
298 :     else                          AjcSetDlgItemStr(hDlg, IDC_LBL_CHARTYPE, TEXT("下のボタンは ASCII Window です"));
299 : }
300 :
301 : //-------------------------------------------------------------------------------------------------------------------//
```

```
302 : //  サブクラス情報表示                                                                                                  //
303 : //--------------------------------------------------------------------------------------------------------------------//
304 : static  UTP WpName(WNDPROC wp)
305 : {
306 :     static UT txt[128];
307 :     if      (wp == AJC_WNDPROC_NAME(SUB1)) return(TEXT("SUB1"));
308 :     else if (wp == AJC_WNDPROC_NAME(SUB2)) return(TEXT("SUB2"));
309 :     else if (wp == AJC_WNDPROC_NAME(SUB3)) return(TEXT("SUB3"));
310 :     else  AjcSnPrintF(txt, 128, TEXT("%p"), wp);
311 :     return txt;
312 : }
313 : //- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - //
314 : static  VO  ShowSubclassInfo(VO)
315 : {
316 :     PAJCSBCINFO pBuf = NULL;
317 :     UI          i, n;
318 :
319 :     if (n = AjcMmpGetSubclassInfo(hButton, NULL, 0)) {
320 :         AjcVthPrintF(hVth, TEXT("¥n--- Subclass Info. ---¥n"));
321 :         if (pBuf = (PAJCSBCINFO)malloc(sizeof(AJCSBCINFO) * n)) {
322 :             AjcMmpGetSubclassInfo (hButton, pBuf, n);
323 :             for (i = 0; i < n; i++) {
324 :                 AjcVthPrintFA(hVth, pBuf[i].pName);
325 :                 AjcVthPrintF(hVth, TEXT(" : wpOld = %-8s, wpNew = %-8s, WpRet=%s¥n"), WpName(pBuf[i].wpOld),
326 :                                                                                        WpName(pBuf[i].wpNew),
327 :                                                                                        WpName(pBuf[i].wpRet));
328 :             }
329 :             free(pBuf);
330 :         }
331 :         AjcVthPrintF(hVth, TEXT("¥n"));
332 :     }
333 :     else {
334 :         AjcVthPrintF(hVth, TEXT("¥n--- No Subclass Info. ---¥n"));
335 :     }
336 : }
```

3.6.2. SW_SubClass2（複数のウインドを１つのウインドプロシージャでサブクラス化）

以下のサンプルプログラムは、複数のコントロール（ボタン，テキストボックス，スピンボタン）を１つのウインドプロシージャでサブクラス化し、いずれのコントロールも 右クリックすると、その旨のメッセージボックスを表示します。

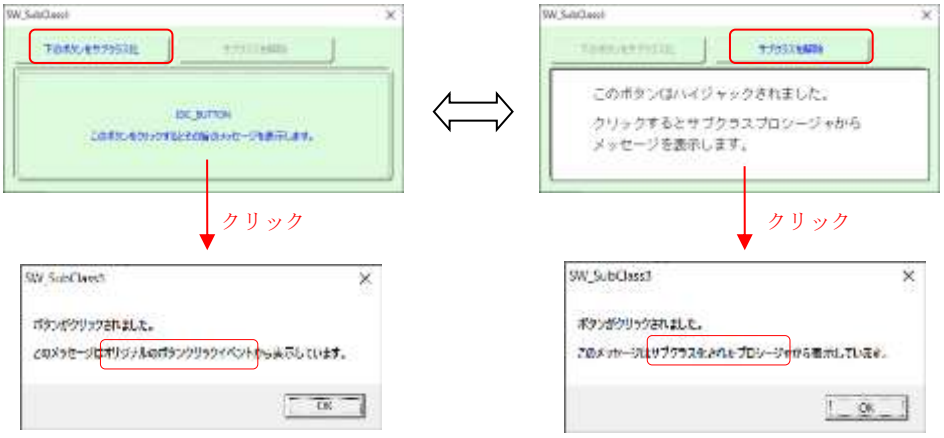



```
 1 : //
 2 : //  SW_SubClass2.c
 3 : //
 4 : #include    <AjrCstXX.h>
 5 : #include    <tchar.h>
 6 : #include    "resource.h"
 7 :
 8 : #define WM_SUBCLASS (WM_APP + 100)
 9 :
10 : //----- 作業領域 ----------------------------------------------------------------------------------------------------//
11 : static  HINSTANCE       hInst;
12 : static  HWND            hDlgMain;
13 : static  HWND            hButton;
14 : static  WNDPROC         WndProcButton = NULL;
15 : static  HWND            hBtn, hTxt, hSpn;
16 :
17 : //----- 内部サブ関数 ------------------------------------------------------------------------------------------------//
18 : AJC_DLGPROC_DEF(Main);
19 : AJC_WNDPROC_DEF(SUB);
20 :
21 : //-------------------------------------------------------------------------------------------------------------------//
22 : int WINAPI AjcWinMain(HINSTANCE hInstance, HINSTANCE hinstPrev, UTP szCmdLine, int iCmdShow)
23 : {
24 :     MSG     msg;
25 :
26 :     hInst = hInstance;
27 :
28 :     AjcDgcSetup();
29 :
30 :     //----- メイン・ダイアログオープン ----------------//
31 :     hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_MAIN), NULL, AJC_DLGPROC_NAME(Main));
32 :     ShowWindow(hDlgMain, SW_SHOW);
33 :
34 :     //----- メッセージループ --------------------------//
35 :     while (GetMessage(&msg, NULL, 0, 0)) {
36 :         do {
37 :             if (IsDialogMessage(hDlgMain, &msg)) break;
38 :             TranslateMessage(&msg);
39 :             DispatchMessage (&msg);
40 :         } while (0);
41 :     }
42 :
43 :     return (int)msg.wParam ;
44 : }
45 : //===================================================================================================================//
46 : //                                                                                                                   //
47 : //  ダイアログ・プロシージャ                                                                                         //
48 : //                                                                                                                   //
49 : //===================================================================================================================//
50 : //----- ダイアログ初期化 --------------------------------------------------------------------------------------------//
51 : AJC_DLGPROC(Main, WM_INITDIALOG     )
52 : {
53 :     UDACCEL acc[1] = {0, -1};
54 :
55 :     hDlgMain = hDlg;
56 :     hBtn     = GetDlgItem(hDlg, IDC_BUTTON );
57 :     hTxt     = GetDlgItem(hDlg, IDC_TEXTBOX);
58 :     hSpn     = GetDlgItem(hDlg, IDC_SPIN   );
59 :
60 :     //  ツールチップ設定
61 :     AjcTipTextAdd(hBtn, TEXT("右の数値をクリアーします（0 を設定します）"));
62 :     AjcTipTextAdd(hTxt, TEXT("数値設定用テキストボックス（-10 ～ 10）"));
63 :     AjcTipTextAdd(hSpn, TEXT("左の数値を増減します"));
64 :     //  スピンボタン初期化
```

```
 65 :     SendMessage(hSpn, UDM_SETRANGE, 0, MAKELONG(-10, 10));
 66 :     SendMessage(hSpn, UDM_SETACCEL, 1, (LPARAM)acc);
 67 :     //  サブクラス化
 68 :     WndProcButton = MAjcMmpSetSubclass(SUB, hBtn);
 69 :     WndProcButton = MAjcMmpSetSubclass(SUB, hTxt);
 70 :     WndProcButton = MAjcMmpSetSubclass(SUB, hSpn);
 71 :     //  ウインドを中央に移動
 72 :     AjcMoveWindowToCenterOfMonitor(hDlg);
 73 :
 74 :     return TRUE;
 75 : }
 76 : //----- ウインド破棄 ----------------------------------------------------------------------------------------------//
 77 : AJC_DLGPROC(Main, WM_DESTROY        )
 78 : {
 79 :     PostQuitMessage(0);
 80 :     return TRUE;
 81 : }
 82 : //----- IDC_BUTTON ------------------------------------------------------------------------------------------------//
 83 : AJC_DLGPROC(Main, IDC_BUTTON        )
 84 : {
 85 :     AjcSetDlgItemSInt(hDlg, IDC_TEXTBOX, 0);
 86 :     return TRUE;
 87 : }
 88 : //----- キャンセル ------------------------------------------------------------------------------------------------//
 89 : AJC_DLGPROC(Main, IDCANCEL          )
 90 : {
 91 :     DestroyWindow(hDlg);
 92 :     return TRUE;
 93 : }
 94 : //-----------------------------------------------------------------------------------------------------------------//
 95 : AJC_DLGMAP_DEF(Main)
 96 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG  )
 97 :     AJC_DLGMAP_MSG(Main, WM_DESTROY     )
 98 :     AJC_DLGMAP_CMD(Main, IDC_BUTTON     )
 99 :     AJC_DLGMAP_CMD(Main, IDCANCEL       )
100 : AJC_DLGMAP_END
101 : //-----------------------------------------------------------------------------------------------------------------//
102 : //  ３つのコントロールをサブクラス化したウインドプロシージャ                                                       //
103 : //-----------------------------------------------------------------------------------------------------------------//
104 : AJC_WNDPROC(SUB, WM_RBUTTONDOWN         )
105 : {
106 :     MessageBox(NULL, TEXT("右クリックしました。"), TEXT("SW_SubClass2"), MB_OK);
107 :     return MAjcMmpCallOrgWndProc(SUB);  //  オリジナルのウインドプロシージャへ続く（インターセプトする場合は０を返す）
108 : }
109 : //-----------------------------------------------------------------------------------------------------------------//
110 : AJC_WNDMAP_DEF(SUB)
111 :     AJC_WNDMAP_MSG(SUB, WM_RBUTTONDOWN  )
112 : AJC_WNDMAP_END
113 :
```

3.6.3. SW_SubClass3（サブクラス化してメッセージをインターセプト）

以下のサンプルプログラムは、ボタン(IDC_BUTTON)をサブクラス化し、ボタンへの全てのメッセージをインターセプト（横取り）します。
「下のボタンをサブクラス化」を押すと、ボタンをサブクラス化し、クリックするとサブクラスプロシージャからメッセージを表示します。
「サブクラスを解除」を押すと、サブクラス化を解除し、通常のボタンイベント(BN_CLICKED)からメッセージを表示します。



67 行の「MAjcMmpSetDefWndProc(SUB, hButton);」を実行することにより、WM_LBUTTONDOWN と WM_PAINT を除く、ボタンへの全てのメッセージはブロックされ、DefWindowProc()へ送られます。
また、117 行と 145 行で「return 0;」としている為、WM_LBUTTONDOWN, WM_PAINT はインターセプトされます。(DefWindowProc()へも送られません)

```
  1 : //
  2 : //  SW_SubClass3.c
  3 : //
  4 : #include    <AjrCstXX.h>
  5 : #include    <tchar.h>
  6 : #include    "resource.h"
  7 :
  8 : //----- 作業領域 ------------------------------------------------------------------------------------------------//
  9 : static  HINSTANCE       hInst;
 10 : static  HWND            hDlgMain;
 11 : static  HWND            hButton;
 12 : static  WNDPROC         WndProcButton = NULL;
 13 :
 14 : //----- 内部サブ関数 --------------------------------------------------------------------------------------------//
 15 : AJC_DLGPROC_DEF(Main);
 16 : AJC_WNDPROC_DEF(SUB);
 17 :
 18 : //-------------------------------------------------------------------------------------------------------------//
 19 : int WINAPI AjcWinMain(HINSTANCE hInstance, HINSTANCE hinstPrev, UTP szCmdLine, int iCmdShow)
 20 : {
 21 :     MSG     msg;
 22 :
 23 :     hInst = hInstance;
 24 :     AjcDgcSetup();
 25 :     //----- メイン・ダイアログオープン ----------------//
 26 :     hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_MAIN), NULL, AJC_DLGPROC_NAME(Main));
 27 :     ShowWindow(hDlgMain, SW_SHOW);
 28 :
 29 :     //----- メッセージループ --------------------------//
 30 :     while (GetMessage(&msg, NULL, 0, 0)) {
 31 :         do {
 32 :             if (IsDialogMessage(hDlgMain, &msg)) break;
 33 :             TranslateMessage(&msg);
 34 :             DispatchMessage (&msg);
 35 :         } while (0);
 36 :     }
 37 :
 38 :     return (int)msg.wParam ;
 39 : }
 40 : //=============================================================================================================//
 41 : //                                                                                                             //
 42 : //  ダイアログ・プロシージャ                                                                                   //
 43 : //                                                                                                             //
 44 : //=============================================================================================================//
 45 : //----- ダイアログ初期化 ----------------------------------------------------------------------------------------//
```

```
AJC_DLGPROC(Main, WM_INITDIALOG     )
{
    hDlgMain = hDlg;
    hButton  = GetDlgItem(hDlg, IDC_BUTTON);

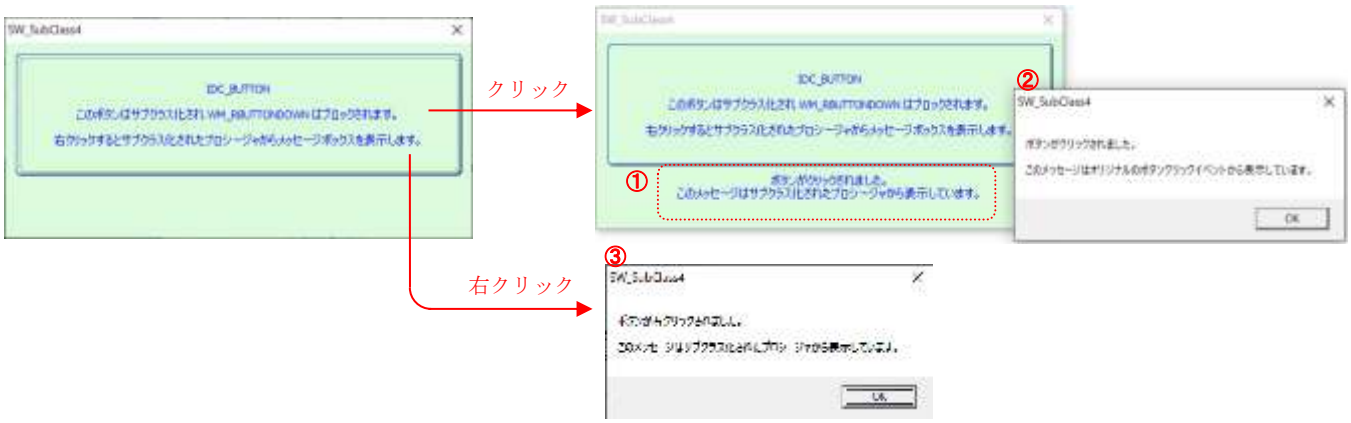
    return TRUE;
}
//----- ウインド破棄 ----------------------------------------------------------------------------------------//
AJC_DLGPROC(Main, WM_DESTROY        )
{
    PostQuitMessage(0);
    return TRUE;
}
//----- IDC_CMD_SUBCLASS ------------------------------------------------------------------------------------//
AJC_DLGPROC(Main, IDC_CMD_SUBCLASS  )
{
    //  サブクラス化
    WndProcButton = MAjcMmpSetSubclass(SUB, hButton);
    InvalidateRect(hButton, NULL, TRUE);
    //  未処理のメッセージ（AJC_WNDMAP_DEF～AJC_WNDMAP_END 間で定義されていないメッセージ）はすべて、
    //  オリジナルのウインドプロシージャを実行しないようにする。(DefWindowProc()を実行するようにする)
    MAjcMmpSetDefWndProc(SUB, hButton);
    //  ボタングレー化
    AjcEnableDlgItem(hDlg, IDC_CMD_SUBCLASS  , FALSE);
    AjcEnableDlgItem(hDlg, IDC_CMD_UNSUBCLASS, TRUE);

    return TRUE;
}
//----- IDC_CMD_UNSUBCLASS ----------------------------------------------------------------------------------//
AJC_DLGPROC(Main, IDC_CMD_UNSUBCLASS)
{
    //  サブクラス化解除
    MAjcMmpClrSubclass(SUB, hButton);
    //  ボタングレー化
    AjcEnableDlgItem(hDlg, IDC_CMD_SUBCLASS  , TRUE);
    AjcEnableDlgItem(hDlg, IDC_CMD_UNSUBCLASS, FALSE);
    //  再描画
    InvalidateRect(hButton, NULL, TRUE);
    return TRUE;
}
//----- IDC_BUTTON ------------------------------------------------------------------------------------------//
AJC_DLGPROC(Main, IDC_BUTTON        )
{
    if (HIWORD(wParam) == BN_CLICKED) {
        MessageBox(hDlg, TEXT("ボタンがクリックされました。¥n¥n")
                         TEXT("このメッセージはオリジナルのボタンクリックイベントから表示しています。"), TEXT("SW_SubClass3"), MB_OK);
    }
    return TRUE;
}
//----- キャンセル ------------------------------------------------------------------------------------------//
AJC_DLGPROC(Main, IDCANCEL          )
{
    DestroyWindow(hDlg);
    return TRUE;
}
//-----------------------------------------------------------------------------------------------------------//
AJC_DLGMAP_DEF(Main)
    AJC_DLGMAP_MSG(Main, WM_INITDIALOG  )
    AJC_DLGMAP_MSG(Main, WM_DESTROY     )
    AJC_DLGMAP_CMD(Main, IDC_CMD_SUBCLASS   )
    AJC_DLGMAP_CMD(Main, IDC_CMD_UNSUBCLASS )
    AJC_DLGMAP_CMD(Main, IDC_BUTTON     )
    AJC_DLGMAP_CMD(Main, IDCANCEL       )
AJC_DLGMAP_END
//-----------------------------------------------------------------------------------------------------------//
//  ボタンをサブクラス化したウインドプロシージャ                                                             //
//-----------------------------------------------------------------------------------------------------------//
AJC_WNDPROC(SUB, WM_LBUTTONDOWN     )
{
    MessageBox(hwnd, TEXT("ボタンがクリックされました。¥n¥n")
                     TEXT("このメッセージはサブクラス化されたプロシージャから表示しています。"), TEXT("SW_SubClass3"), MB_OK);
    return 0;
}
//-----------------------------------------------------------------------------------------------------------//
AJC_WNDPROC(SUB, WM_PAINT           )
{
    PAINTSTRUCT ps;
    HDC         hdc;
    HPEN        hPen;
    HBRUSH      hBru;
```

```
126 :     RECT        r;
127 :     UT          txt1[] = TEXT("このボタンはハイジャックされました。");
128 :     UT          txt2[] = TEXT("クリックするとサブクラスプロシージャから");
129 :     UT          txt3[] = TEXT("メッセージを表示します。");
130 :
131 :     hdc   = BeginPaint(hwnd, &ps);
132 :     hPen = (HPEN  )SelectObject(hdc, GetStockObject(BLACK_PEN));
133 :     hBru = (HBRUSH)SelectObject(hdc, GetStockObject(WHITE_BRUSH));
134 :     GetClientRect(hwnd, &r);
135 :     Rectangle(hdc, r.left, r.top, r.right, r.bottom);
136 :     SelectObject(hdc, hPen);
137 :     SelectObject(hdc, hBru);
138 :
139 :     SetBkMode(hdc, TRANSPARENT);
140 :     TextOut(hdc, 50, 20, txt1, (int)MAjcStrLen(txt1));
141 :     TextOut(hdc, 50, 55, txt2, (int)MAjcStrLen(txt2));
142 :     TextOut(hdc, 50, 80, txt3, (int)MAjcStrLen(txt3));
143 :     EndPaint(hwnd, &ps);
144 :
145 :     return 0;
146 : }
147 : //-------------------------------------------------------------------------------------------------------------------//
148 : AJC_WNDMAP_DEF(SUB)
149 :     AJC_WNDMAP_MSG(SUB, WM_LBUTTONDOWN  )
150 :     AJC_WNDMAP_MSG(SUB, WM_PAINT        )
151 : AJC_WNDMAP_END
152 :
```

3.6.4. SW_SubClass4（サブクラス化してメッセージをトラップ）

以下のサンプルプログラムは、ボタン(IDC_BUTTON)をサブクラス化し、ボタンへの WM_RBUTTONDOWN メッセージをブロックします。
ボタンに WM_LBUTTONDOWN が送られた場合は、ダイアログ下部にその旨、メッセージを表示します。
ボタンに WM_RBUTTONDOWN が送られた場合は、右クリックされた旨のメッセージボックスを表示します。



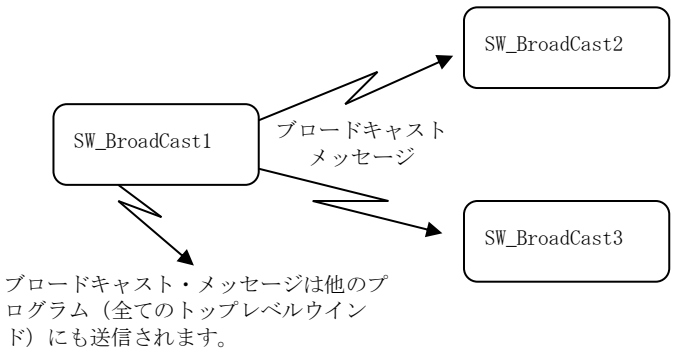
WM_LBUTTONDOWN メッセージは、サブクラスでトラップされ、①のメッセージが表示されます。
90 行の「return MAjcMmpCallOrgWndProc(SUB);」により、WM_LBUTTONDOWN メッセージはボタンへも送られます。これにより IDC_BUTTON のクリックイベントが発生し、②のメッセージボックスが表示されます。
WM_RBUTTONDOWN メッセージは、サブクラスでトラップされ、③のメッセージが表示されます。
97 行で「return 0;」している為、WM_RBUTTONDOWN メッセージはブロックされ、ボタンへ送られません。(DefWindowProc()へも送られません)
以上から、WM_RBUTTONDOWN 以外のメッセージは全てボタンへ送られます。

```
  1 : //
  2 : //  SW_SubClass4.c
  3 : //
  4 : #include    <AjrCstXX.h>
  5 : #include    <tchar.h>
  6 : #include    "resource.h"
  7 :
  8 : //----- 作業領域 ------------------------------------------------------------------------------------------------//
  9 : static  HINSTANCE       hInst;
 10 : static  HWND            hDlgMain;
 11 : static  HWND            hButton;
 12 : static  WNDPROC         WndProcButton = NULL;
 13 :
 14 : //----- 内部サブ関数 --------------------------------------------------------------------------------------------//
 15 : AJC_DLGPROC_DEF(Main);
 16 : AJC_WNDPROC_DEF(SUB);
 17 :
 18 : //-------------------------------------------------------------------------------------------------------------//
 19 : int WINAPI AjcWinMain(HINSTANCE hInstance, HINSTANCE hinstPrev, UTP szCmdLine, int iCmdShow)
 20 : {
 21 :     MSG     msg;
 22 :
 23 :     hInst = hInstance;
 24 :     AjcDgcSetup();
 25 :     //----- メイン・ダイアログオープン ----------------//
 26 :     hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_MAIN), NULL, AJC_DLGPROC_NAME(Main));
 27 :     ShowWindow(hDlgMain, SW_SHOW);
 28 :
 29 :     //----- メッセージループ --------------------------//
 30 :     while (GetMessage(&msg, NULL, 0, 0)) {
 31 :         do {
 32 :             if (IsDialogMessage(hDlgMain, &msg)) break;
 33 :             TranslateMessage(&msg);
 34 :             DispatchMessage (&msg);
 35 :         } while (0);
 36 :     }
 37 :
 38 :     return (int)msg.wParam ;
 39 : }
 40 : //=============================================================================================================//
 41 : //                                                                                                             //
 42 : //  ダイアログ・プロシージャ                                                                                   //
 43 : //                                                                                                             //
 44 : //=============================================================================================================//
```

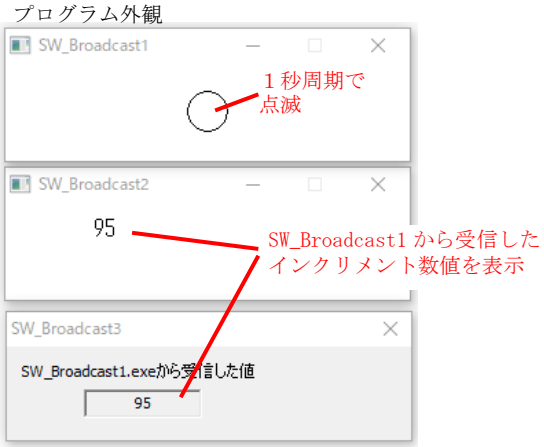
```
45 : //----- ダイアログ初期化 ----------------------------------------------------------------------------------------------//
46 : AJC_DLGPROC(Main, WM_INITDIALOG     )
47 : {
48 :     hDlgMain = hDlg;
49 :     hButton  = GetDlgItem(hDlg, IDC_BUTTON);
50 :
51 :     //  サブクラス化
52 :     WndProcButton = MAjcMmpSetSubclass(SUB, hButton);
53 :
54 :     return TRUE;
55 : }
56 : //----- ウインド破棄 --------------------------------------------------------------------------------------------------//
57 : AJC_DLGPROC(Main, WM_DESTROY        )
58 : {
59 :     PostQuitMessage(0);
60 :     return TRUE;
61 : }
62 : //----- IDC_BUTTON ----------------------------------------------------------------------------------------------------//
63 : AJC_DLGPROC(Main, IDC_BUTTON        )
64 : {
65 :     MessageBox(hDlg, TEXT("ボタンがクリックされました。¥n¥n")
66 :                      TEXT("このメッセージはオリジナルのボタンクリックイベントから表示しています。"), TEXT("SW_SubClass4"), MB_OK);
67 :     AjcSetDlgItemStr(hDlgMain, IDC_LBL_MSG, TEXT(""));
68 :     return TRUE;
69 : }
70 : //----- キャンセル ----------------------------------------------------------------------------------------------------//
71 : AJC_DLGPROC(Main, IDCANCEL          )
72 : {
73 :     DestroyWindow(hDlg);
74 :     return TRUE;
75 : }
76 : //--------------------------------------------------------------------------------------------------------------------//
77 : AJC_DLGMAP_DEF(Main)
78 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG  )
79 :     AJC_DLGMAP_MSG(Main, WM_DESTROY     )
80 :     AJC_DLGMAP_CMD(Main, IDC_BUTTON     )
81 :     AJC_DLGMAP_CMD(Main, IDCANCEL       )
82 : AJC_DLGMAP_END
83 : //--------------------------------------------------------------------------------------------------------------------//
84 : //  ボタンをサブクラス化したウインドプロシージャ                                                                      //
85 : //--------------------------------------------------------------------------------------------------------------------//
86 : AJC_WNDPROC(SUB, WM_LBUTTONDOWN     )
87 : {
88 :     AjcSetDlgItemStr(hDlgMain, IDC_LBL_MSG, TEXT("ボタンがクリックされました。¥n"
89 :                                             TEXT("このメッセージはサブクラス化されたプロシージャから表示しています。")));
90 :     return MAjcMmpCallOrgWndProc(SUB);
91 : }
92 : //--------------------------------------------------------------------------------------------------------------------//
93 : AJC_WNDPROC(SUB, WM_RBUTTONDOWN     )
94 : {
95 :     MessageBox(hwnd, TEXT("ボタンが右クリックされました。¥n¥n")
96 :                      TEXT("このメッセージはサブクラス化されたプロシージャから表示しています。"), TEXT("SW_SubClass4"), MB_OK);
97 :     return 0;
98 : }
99 : //--------------------------------------------------------------------------------------------------------------------//
100 : AJC_WNDMAP_DEF(SUB)
101 :     AJC_WNDMAP_MSG(SUB, WM_LBUTTONDOWN  )
102 :     AJC_WNDMAP_MSG(SUB, WM_RBUTTONDOWN  )
103 : AJC_WNDMAP_END
104 :
```

3.6.5. SW_Broadcast1～3（ブロードキャストメッセージ）

ブロードキャスト・メッセージに関するサンプルプログラムを示します。
このサンプルプログラムは、３つのプログラムで構成されます。
「SW_Broadcast1」は、１秒間隔でインクリメントされた数値をパラメタにして、ブロードキャスト・メッセージを送信します。
「SW_Broadcast2」と「SW_Broadcast3」は、「SW_Broadcast1」から送られてきたブロードキャスト・メッセージを受信し、当該インクリメント数値を表示します。



プログラム外観



SW_BroadCast1.c

```
//
//  SW_Broadcast1.c
//
#include    <AjrCstXX.h>
#include    <tchar.h>

//-------------------------------------------------------------------------------------------------------------//
//  ワーク                                                                                                     //
//-------------------------------------------------------------------------------------------------------------//
HINSTANCE       hInst       = NULL;             //  インスタンスハンドル
HWND            hWndMain    = NULL;             //  ウインドハンドル
UI              MsgRwm      = 0;                //  ブロードキャストメッセージコード
UI              TimeCount   = 0;                //  １秒タイマカウンタ
HBRUSH          hBrush      = NULL;             //  ブラシハンドル

//-------------------------------------------------------------------------------------------------------------//
//  内部サブ関数                                                                                               //
//-------------------------------------------------------------------------------------------------------------//
AJC_WNDPROC_DEF(Main);

//=============================================================================================================//
//                                                                                                             //
//  W i n M a i n                                                                                              //
//                                                                                                             //
//=============================================================================================================//
int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
{
    MSG         msg;
    WNDCLASS    wndclass;

    hInst = hInstance;

    //----- バックウインド生成 -------------------------//
    wndclass.style          = 0;
    wndclass.lpfnWndProc    = AJC_WNDPROC_NAME(Main);
    wndclass.cbClsExtra     = 0;
    wndclass.cbWndExtra     = 0;
    wndclass.hInstance      = hInst;
    wndclass.hIcon          = NULL;
    wndclass.hCursor        = LoadCursor(NULL, IDC_ARROW);
    wndclass.hbrBackground  = (HBRUSH)GetStockObject(WHITE_BRUSH);
    wndclass.lpszMenuName   = NULL;
    wndclass.lpszClassName  = TEXT("SW_Broadcast1");
    RegisterClass(&wndclass);

    hWndMain = CreateWindow(TEXT("SW_Broadcast1"),                      // window class name
                            TEXT("SW_Broadcast1"),                      // window caption
                            WS_OVERLAPPEDWINDOW & ~(WS_THICKFRAME | WS_MAXIMIZEBOX), // window style
                            10,                                         // initial x position
                            10,                                         // initial y position
                            300,                                        // initial x size
                            100,                                        // initial y size
                            NULL,                                       // parent window handle
                            NULL,                                       // window menu handle
                            hInst,                                      // program instance handle
                            NULL);                                      // creation parameters

    ShowWindow(hWndMain, iCmdShow);

    //----- メッセージループ ---------------------------//
    while (GetMessage(&msg, NULL, 0, 0)) {
        do {
            TranslateMessage(&msg);
            DispatchMessage (&msg);
        } while (0);
    }

    return (int)msg.wParam ;
}
//=============================================================================================================//
//                                                                                                             //
//  ウインド・プロシージャ                                                                                     //
//                                                                                                             //
//=============================================================================================================//
//----- WM_CREATE -------------------------------------------------------------------------------------------//
AJC_WNDPROC(Main, WM_CREATE     )
{
```

```
 78 :     MsgRwm = RegisterWindowMessage(TEXT("SW_Broadcast"));
 79 :     hBrush = (HBRUSH)GetStockObject(BLACK_BRUSH);
 80 :     SetTimer(hwnd, 1, 1000, NULL);
 81 :     return 0;
 82 : }
 83 : //----- WM_DESTROY ----------------------------------------------------------------------------------------//
 84 : AJC_WNDPROC(Main, WM_DESTROY    )
 85 : {
 86 :     KillTimer(hwnd, 1);
 87 :     PostQuitMessage(0);
 88 :     return 0;
 89 : }
 90 : //----- WM_TIMER ------------------------------------------------------------------------------------------//
 91 : AJC_WNDPROC(Main, WM_TIMER      )
 92 : {
 93 :     TimeCount++;
 94 :     if (TimeCount & 1) hBrush = GetStockObject(WHITE_BRUSH);
 95 :     else               hBrush = GetStockObject(BLACK_BRUSH);
 96 :     InvalidateRect(hwnd, NULL, FALSE);
 97 :
 98 :     PostMessage(HWND_BROADCAST, MsgRwm, TimeCount, 0);
 99 :
100 :     return 0;
101 : }
102 : //----- WM_PAINT ------------------------------------------------------------------------------------------//
103 : AJC_WNDPROC(Main, WM_PAINT      )
104 : {
105 :     PAINTSTRUCT ps;
106 :     HDC         hdc;
107 :     RECT        r;
108 :     int         x, y;
109 :
110 :     hdc   = BeginPaint(hwnd, &ps);
111 :     GetClientRect(hwnd, &r);
112 :     x = (r.right - r.left) / 2;
113 :     y = (r.bottom - r.top) / 2;
114 :     SelectObject(hdc, hBrush);
115 :     Ellipse (hdc, x - 15, y - 15, x + 15, y + 15);
116 :     EndPaint(hwnd, &ps);
117 :     return 0;
118 : }
119 : //---------------------------------------------------------------------------------------------------------//
120 : AJC_WNDMAP_DEF(Main)
121 :     AJC_WNDMAP_MSG(Main, WM_CREATE  )
122 :     AJC_WNDMAP_MSG(Main, WM_DESTROY )
123 :     AJC_WNDMAP_MSG(Main, WM_TIMER   )
124 :     AJC_WNDMAP_MSG(Main, WM_PAINT   )
125 : AJC_WNDMAP_END
```

SW_BroadCast2.c

```
 1 : //
 2 : //  SW_Broadcast2.c
 3 : //
 4 : #include    <AjrCstXX.h>
 5 : #include    <tchar.h>
 6 :
 7 : //-------------------------------------------------------------------------------------------------------------//
 8 : //  ワーク                                                                                                     //
 9 : //-------------------------------------------------------------------------------------------------------------//
10 : HINSTANCE       hInst       = NULL;         //  インスタンスハンドル
11 : HWND            hWndMain    = NULL;         //  ウインドハンドル
12 : int             Number      = 0;            //  MasMap01A.exe から受信した値
13 :
14 : //-------------------------------------------------------------------------------------------------------------//
15 : //  内部サブ関数                                                                                               //
16 : //-------------------------------------------------------------------------------------------------------------//
17 : AJC_WNDPROC_DEF(Main);
18 :
19 : //=============================================================================================================//
20 : //                                                                                                             //
21 : //  W i n M a i n                                                                                              //
22 : //                                                                                                             //
23 : //=============================================================================================================//
24 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
25 : {
26 :     MSG         msg;
27 :     WNDCLASS    wndclass;
28 :
29 :     hInst = hInstance;
30 :
31 :     //----- バックウインド生成 -------------------------//
32 :     wndclass.style          = 0;
33 :     wndclass.lpfnWndProc    = AJC_WNDPROC_NAME(Main);
34 :     wndclass.cbClsExtra     = 0;
35 :     wndclass.cbWndExtra     = 0;
36 :     wndclass.hInstance      = hInst;
37 :     wndclass.hIcon          = NULL;
38 :     wndclass.hCursor        = LoadCursor(NULL, IDC_ARROW);
39 :     wndclass.hbrBackground  = GetStockObject(WHITE_BRUSH);
40 :     wndclass.lpszMenuName   = NULL;
41 :     wndclass.lpszClassName  = TEXT("SW_Broadcast2");
42 :     RegisterClass(&wndclass);
43 :
44 :     hWndMain = CreateWindow(TEXT("SW_Broadcast2"),    // window class name
45 :                             TEXT("SW_Broadcast2"),    // window caption
46 :                             WS_OVERLAPPEDWINDOW & ~(WS_THICKFRAME | WS_MAXIMIZEBOX),
47 :                             10,                       // initial x position
48 :                             110,                      // initial y position
49 :                             300,                      // initial x size
50 :                             100,                      // initial y size
51 :                             NULL,                     // parent window handle
52 :                             NULL,                     // window menu handle
53 :                             hInst,                    // program instance handle
54 :                             NULL);                    // creation parameters
55 :
56 :     ShowWindow(hWndMain, iCmdShow);
57 :
58 :     //----- メッセージループ ---------------------------//
59 :     while (GetMessage(&msg, NULL, 0, 0)) {
60 :         do {
61 :             TranslateMessage(&msg);
62 :             DispatchMessage (&msg);
63 :         } while (0);
64 :     }
65 :
66 :     return (int)msg.wParam ;
67 : }
68 : //=============================================================================================================//
69 : //                                                                                                             //
70 : //  ウインド・プロシージャ                                                                                     //
71 : //                                                                                                             //
72 : //=============================================================================================================//
73 : //----- WM_CREATE -----------------------------------------------------------------------------------------//
74 : AJC_WNDPROC(Main, WM_CREATE     )
75 : {
76 :     return 0;
77 : }
```

```
 78 : //----- WM_DESTROY ------------------------------------------------------------------------------------------//
 79 : AJC_WNDPROC(Main, WM_DESTROY    )
 80 : {
 81 :     KillTimer(hwnd, 1);
 82 :     PostQuitMessage(0);
 83 :     return 0;
 84 : }
 85 : //----- WM_PAINT -------------------------------------------------------------------------------------------//
 86 : AJC_WNDPROC(Main, WM_PAINT      )
 87 : {
 88 :     PAINTSTRUCT ps;
 89 :     HDC         hdc;
 90 :     UT          txt[16];
 91 :
 92 :     hdc   = BeginPaint(hwnd, &ps);
 93 :     AjcSnPrintF(txt, 16, TEXT("%11d"), Number);
 94 :     TextOut(hdc, 10, 10, txt, (UI)MAjcStrLen(txt));
 95 :     EndPaint(hwnd, &ps);
 96 :     return 0;
 97 : }
 98 : //----- WM_MSGMAP01 ----------------------------------------------------------------------------------------//
 99 : AJC_WNDPROC(Main, WM_MSGMAP01   )
100 : {
101 :     Number = (int)wParam;
102 :     InvalidateRect(hwnd, NULL, FALSE);
103 :     return 0;
104 : }
105 : //-------------------------------------------------------------------------------------------------------------//
106 : AJC_WNDMAP_DEF(Main)
107 :     AJC_WNDMAP_MSG(Main, WM_CREATE  )
108 :     AJC_WNDMAP_MSG(Main, WM_DESTROY )
109 :     AJC_WNDMAP_MSG(Main, WM_PAINT   )
110 :     AJC_WNDMAP_RWM(Main, WM_MSGMAP01, TEXT("SW_Broadcast"))
111 : AJC_WNDMAP_END
```

SW_BroadCast3.c

```
 1 : //
 2 : //  SW_Broadcast3.c
 3 : //
 4 : #include    <AjrCstXX.h>
 5 : #include    <tchar.h>
 6 : #include    "resource.h"
 7 :
 8 : //-------------------------------------------------------------------------------------------------------------------//
 9 : //  ワーク                                                                                                           //
10 : //-------------------------------------------------------------------------------------------------------------------//
11 : HINSTANCE       hInst;                      //  ＤＬＬインスタンスハンドル
12 : HWND            hDlgMain;                   //  ダイアログボックスハンドル
13 :
14 : //-------------------------------------------------------------------------------------------------------------------//
15 : //  内部サブ関数                                                                                                     //
16 : //-------------------------------------------------------------------------------------------------------------------//
17 : AJC_DLGPROC_DEF(Main);
18 :
19 : //===================================================================================================================//
20 : //                                                                                                                   //
21 : //  Ｗｉｎ Ｍａｉｎ                                                                                                  //
22 : //                                                                                                                   //
23 : //===================================================================================================================//
24 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
25 : {
26 :     MSG     msg;
27 :
28 :     //----- メイン・ダイアログオープン -----------------//
29 :     hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_MAIN), NULL, AJC_DLGPROC_NAME(Main));
30 :     ShowWindow(hDlgMain, SW_SHOW);
31 :
32 :     //----- メッセージループ ---------------------------//
33 :     while (GetMessage(&msg, NULL, 0, 0)) {
34 :         do {
35 :             if (IsDialogMessage(hDlgMain, &msg)) break;
36 :             TranslateMessage(&msg);
37 :             DispatchMessage (&msg);
38 :         } while (0);
39 :     }
40 :
41 :     return (int)msg.wParam ;
42 : }
43 : //===================================================================================================================//
44 : //                                                                                                                   //
45 : //  ダイアログ・プロシージャ                                                                                         //
46 : //                                                                                                                   //
47 : //===================================================================================================================//
48 : //----- ダイアログ初期化 -----------------------------------------------------------------------------------------//
49 : AJC_DLGPROC(Main, WM_INITDIALOG )
50 : {
51 :     SetWindowPos(hDlg, NULL, 10, 210, 0, 0, SWP_NOSIZE);
52 :     return TRUE;
53 : }
54 : //----- WM_MSGMAP01 ----------------------------------------------------------------------------------------------//
55 : AJC_DLGPROC(Main, WM_MSGMAP01   )
56 : {
57 :     AjcSetDlgItemUInt(hDlg, IDC_TXT_NUMBER, (UI)wParam);
58 :     return TRUE;
59 : }
60 : //----- キャンセル -----------------------------------------------------------------------------------------------//
61 : AJC_DLGPROC(Main, IDCANCEL      )
62 : {
63 :     PostQuitMessage(0);
64 :     return TRUE;
65 : }
66 : //-------------------------------------------------------------------------------------------------------------------//
67 : AJC_DLGMAP_DEF(Main)
68 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG                      )
69 :     AJC_DLGMAP_RWM(Main, WM_MSGMAP01    , TEXT("SW_Broadcast"))
70 :     AJC_DLGMAP_CMD(Main, IDCANCEL                           )
71 : AJC_DLGMAP_END
```

4. カスタムコントロール

カスタムコントロールとは、テキストボックス（EDIT クラス）や、ボタン（BUTTON クラス）等のように、プログラムのパーツとして動作するウインドを意味します。

カスタムコントロールを作成するには、CreateWindow()やCreateWindowEx()関数を使用し、あらかじめ決められたクラス名を指定します。

```
hWnd = CreateWindow(TEXT("AjcCtrlInpVal"),   // window class name
                    TEXT(""),                // window caption
                    WS_CHILD | WS_VISIBLE,   // window style
                    0, 0, 200, 20,           // window position and size
                    hWndParent,              // parent window handle
                    hMenu,                   // window menu handle
                    hInst,                   // program instance handle
                    NULL);                   // creation parameters
```

本ライブラリでの、カスタムコントロールのクラス名は、以下のとおりです。

#	クラス名	機能
1	AjcCtrlTmChart	タイムチャート・グラフ表示
2	AjcCtrl3dGraph	３Ｄ／２Ｄグラフの描画
3	AjcCtrlVT100	ＶＴ－１００エミュレーションウインド
4	AjcCtrlInpVal	スライダや、スピンボタンによる値（整数／実数）の入力を行います。 値を直接入力することもできます。
5	AjcCtrlLogFile	ログファイル・テキスト出力
6	AjcCtrlBarGraph	棒グラフ／折れ線グラフの表示
7	AjcCtrlListBox	拡張リストボックスのカスタムコントロール

4.1. 言語設定

カスタムコントロールでのポップアップメニューや各種設定ダイアログで表示されるテキストは、日本語 Windows では日本語で表示されますが、その他の Windows では英語で表示されます。

日本語 Windows での表示例



日本語以外の Windows での表示例



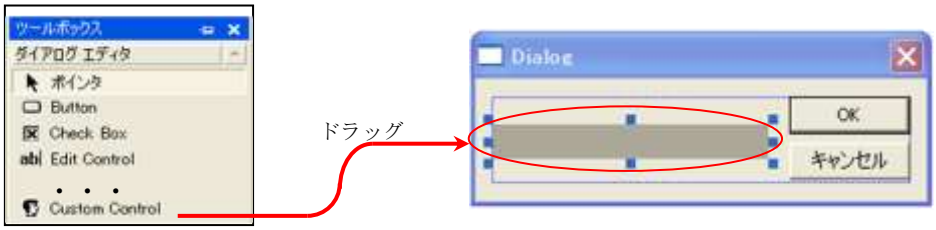
強制的に英語（あるいは日本語）で表示するには、以下の関数を実行します。

```
AjcSetLangId(AJCLID_ENG);  // 英語設定
AjcSetLangId(AJCLID_JPN);  // 日本語設定
```

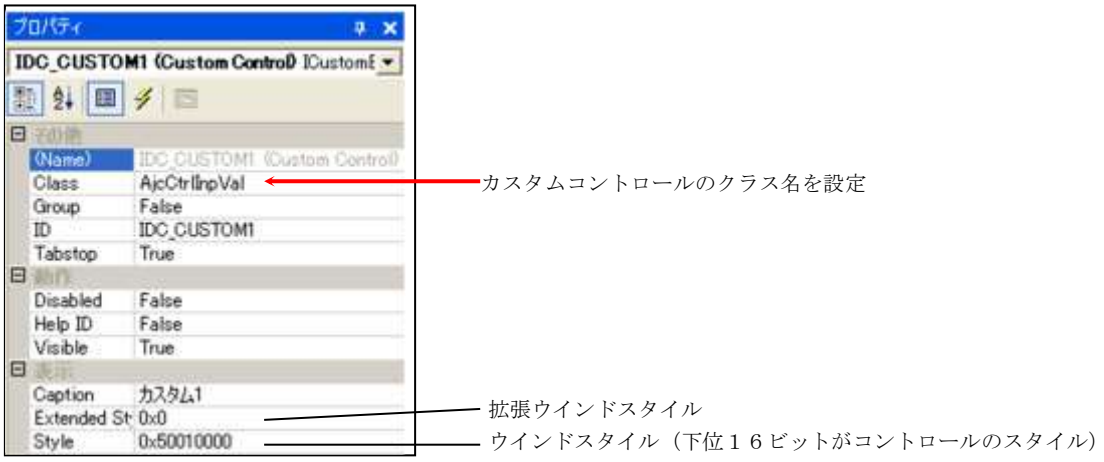
または、レジストリキー（HKEY_CURRENT_USER¥Software¥AjrCstXX¥General）の「Lang」キーを設定することによっても言語を切り替えることができます。（設定は、プログラムを実行する前に行わなければなりません）
"JPN"を設定すると日本語設定に、"ENG"を設定すると英語設定になります。
"AUTO"を設定した場合は、ＰＣの言語設定に従います。

4.2. ダイアログボックスでの利用

ダイアログボックスでカスタムコントロールを使用するには、VisualStudio でダイアログエディタのツールボックスから、ボタンやテキストボックスと同じように「Custom Control」を選択し、ダイアログにドラッグします。



次に、ドラッグしたカスタムコントロールの「Class」プロパティに、使用するカスタムコントロールのクラス名を設定します。



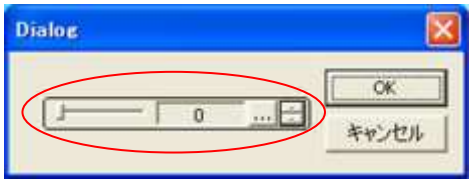
カスタムコントロールの「Style」プロパティは、ビット３１～１６は Windows システムのスタイルで、ビット１５～０はカスタムコントロールのスタイルです。

参考までに、「Style」プロパティのビット３１～１６の内容を以下に示します。（通常は、Bit30, Bit28, Bit16 をセットします）

Bit	名称	内容		Bit	名称	内容
31	WS_POPUP	ポップアップウインド（指定不可）		23	WS_BORDER	細い境界線
30	**WS_CHILD**	**子ウインド（常に指定要）**		22	WS_CAPTION	タイトルバー
29	-			21	WS_VSCROLL	垂直スクロールバー
28	**WS_VISIBLE**	**初期状態で可視**		20	WS_HSCROLL	水平スクロールバー
27	WS_DISABLED	初期状態で使用不可		19	WS_SYSMENU	システムメニュー
26	WS_CLIPSIBLINGS	子ウインド同士のクリッピング		18	WS_TICKFRAME	サイズ変更許可
25	WS_CLIPCHILDREN	子ウインドのクリッピング		17	WS_MINIMIZEBOX	初期状態で最小化
24	-			16	**WS_TABSTOP**	**タブキー ストップ**

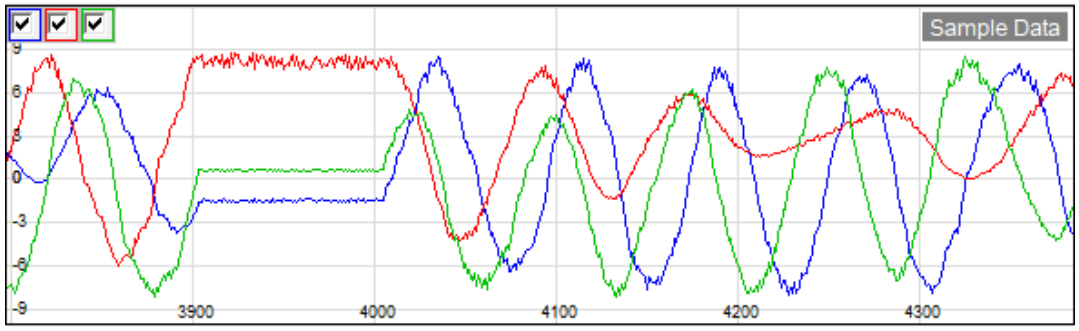
ビット１５～０の設定内容については、各カスタムコントロールの節で説明します。

カスタムコントロールは、ダイアログボックスのデザイン時には、外観は表示されず、単にグレー表示されますが、実行時には以下のように、外観が表示されます。



5. タイムチャート・グラフ表示コントロール（AjcCtrlTmChart クラス）

時間の経過とともに変化する値（例えば、センサ出力の経時変化等）のグラフをリアルタイムに表示するコントロールです。
タイムチャート・グラフ表示コントロールの外観を以下に示します。



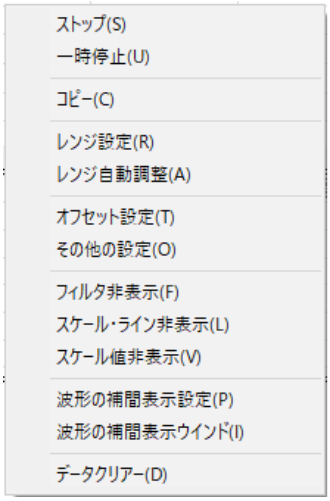
この例では３ヶのデータ項目を、色分けして表示しています。（最大８ヶのデータ項目を表示できます）
縦軸はデータの値を、横軸は時間の経過を意味します。（横軸の目盛りは、経過時間ではなく、データの個数を示します）
最大 100,000 個（デフォルトは 4,096 個）のデータをバッファリングし、スクロールバーでスクロール表示することができます。
データ数がバッファの容量を超えた場合は、古いデータから順に破棄されます。

デフォルトのチャートの表示色は、データ項目の順に、0:**青色**，1:**赤色**，2:**緑色**，3:**水色**、4:**紫色**，5:**黄色**，6:**灰色**，7:**黒色** です。
この表示色は、プロパティ（Item[n].rgb）で変更できます。

5.1. 機能概要

5.1.1. ポップアップメニュー

グラフ上で右クリックすると、以下のポップアップメニューが表示されます。



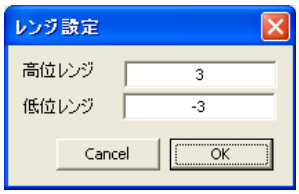
ストップ	：	グラフ表示を停止します。次回は「スタート」メニューとなります。（※1）
一時停止	：	グラフ表示を一時停止します。次回は「再開」メニューとなります。（※2）
コピー	：	グラフ表示内容（ビットマップ）をクリップボードへコピーします。
レンジ設定	：	グラフのレンジ（上限値，下限値）を設定します。
レンジ自動調整	：	プロットデータからグラフのレンジを自動算出して設定します。
オフセット設定	：	プロットデータに加算するオフセット値を設定します。
その他の設定	：	平均化個数，タイムスケール幅，バッファに格納するデータ数を設定します。
フィルタ非表示	：	コントロール左上のフィルタ（チェックボックス）を非表示にします。 次回は「フィルタ表示」メニューに変わります。
スケールライン非表示	：	目盛り線（薄いグレーの線）を非表示にします。 次回は「スケールライン表示」メニューに変わります。
スケール値非表示	：	目盛り数値を非表示にします。 次回は「スケール値表示」メニューに変わります。
波形の補間表示設定	：	波形の補間表示に関するパラメタを設定します
波形の補間表示ウインド	：	波形を補間して表示するウインドを開きます。
データクリアー	：	バッファリングされているデータを全て破棄し、画面をクリアーします。
描画速時間報表示	：	グラフィックイメージの描画時間を計測し表示します (AjcTchEnableMesDraw()で、描画時間計測情報の表示を許可した場合に表示)

※1：ストップ中に投与したプロットデータは破棄されます。
※2：一時停止中に投与したプロットデータは破棄されず、グラフの表示だけが停止します。

コントロールにフォーカスがある状態で、F1 キーを押した場合も ポップアップメニューを表示します。

5.1.2. レンジ設定

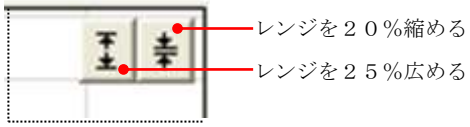
ポップアップメニューで「レンジ設定」を選択すると、以下のダイアログボックスが表示されます。



ここで、レンジ値を入力し、「OK」ボタンを押すと、グラフのレンジが設定されます。
「Cancel」ボタンを押すと設定を中止します。

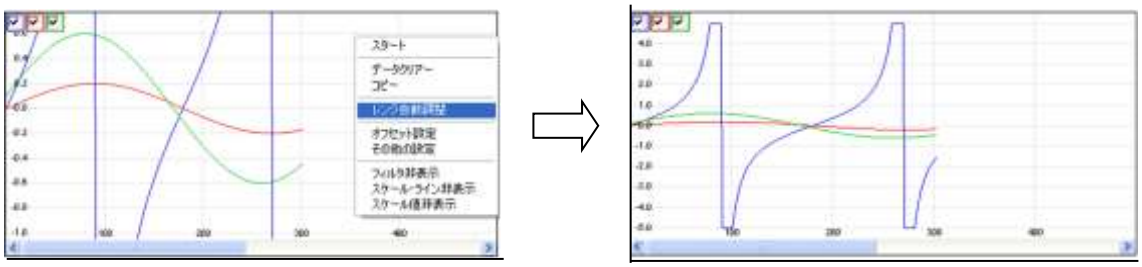
5.1.3. ワンタッチでレンジ設定

マウスカーソルをコントロールの右上隅に置くと、２つのボタンが表示されます。
これらのボタンで、レンジを広めたり、縮めたりすることができます。



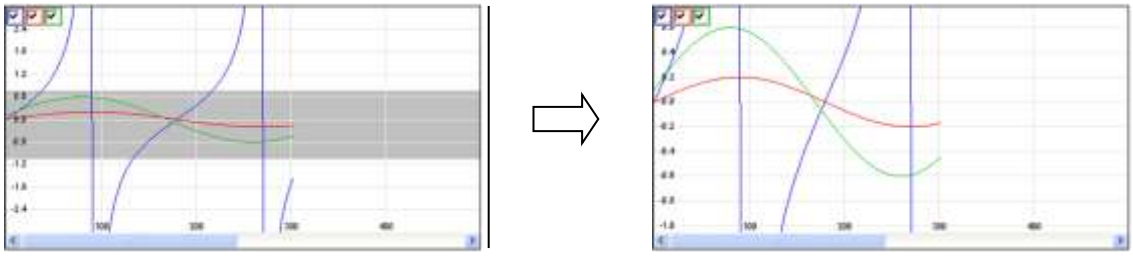
5.1.4. レンジ自動調整

ポップアップメニューで「レンジ自動調整」を選択すると、（フィルタで非表示となっているデータ項目を除く）全てのデータから最小値と最大値を算出し、±５％のマージンを持ってレンジ設定を行います。



5.1.5. ドラッグ操作によるレンジ設定

CTRL キーを押しながら、マウス左ボタンで、レンジ設定したい部分をドラッグすることにより、レンジの設定を行うことができます。



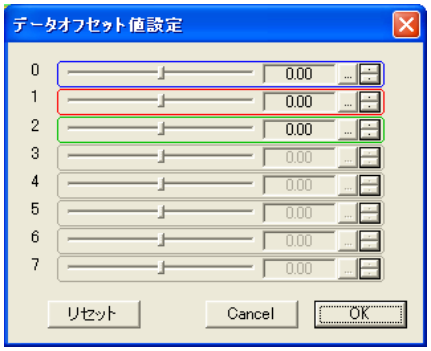
レンジ設定する部分を、CTRL キーを押しながらマウスでドラッグ

CTRL キーを押たまま、マウス左ボタンを離すと、ドラッグした部分がレンジ設定されます。

グラフの上端／下端を越えた部分までドラッグしても、当該ドラッグ範囲がレンジとして設定されます。
CTRL キーを先に離して、マウス左ボタンを離した場合は、レンジ設定は行われません。

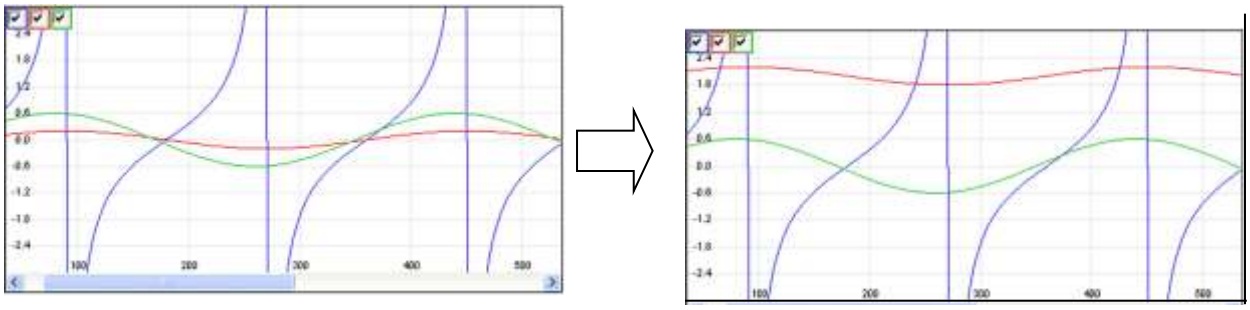
5.1.6. オフセット設定

ポップアップメニューで「オフセット設定」を選択すると、以下のダイアログが表示されます。



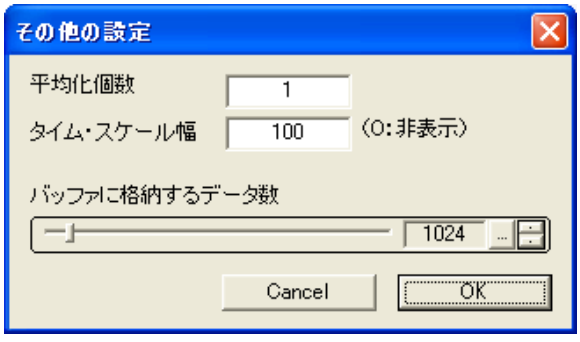
各０～７の項目は、表示されているデータ項目に対応します。（外枠の表示色がグラフ表示色と同じになっています）
ここで、値を設定すると、当該データ項目のデータ値に、設定値を加算した値でグラフが表示されます。
値の設定に追従して設定したオフセット値がグラフに反映されます。
「ＯＫ」ボタンを押すと設定内容が確定します。「Cancel」ボタンを押すと設定内容は破棄され、元のオフセット値に戻ります。
「リセット」ボタンを押すと、全てのオフセット値が「０」に設定されます。

以下の例は、データ項目１（赤色表示のデータ）に、オフセット値として「+2.0」を設定したものです。



5.1.7. その他の設定

ポップアップメニューで「その他の設定」を選択すると、以下のダイアログが表示されます。



平均化個数：
２以上の値を設定すると、コントロールに投与した指定個数のデータの移動平均を算出し、この平均値をバッファに格納します。

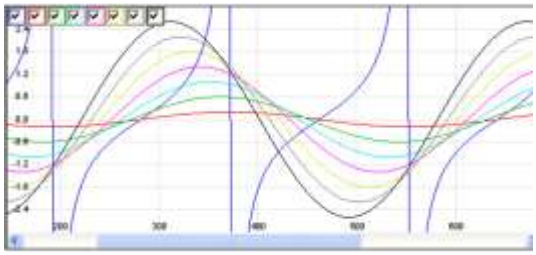
タイムスケール値：
横軸の目盛り表示幅を設定します。

バッファに格納するデータ数：
バッファの容量を、格納するデータ数（データ投与回数）で指定します。

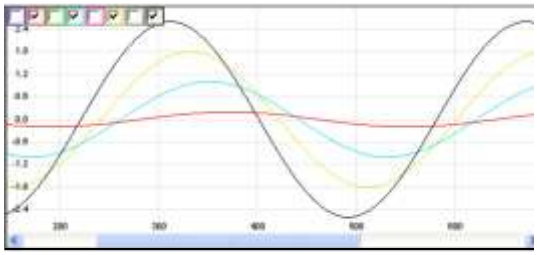
「ＯＫ」ボタンを押すと設定内容を確定します。「Cancel」ボタンを押すと設定を中止します。

5.1.8. フィルタの設定と表示／非表示

カーソルをウインドの左上隅に置くとチェクボックスが表示されます。
左上のチェックボックスは、データ項目の表示フィルタです。チェックを外すと、当該データ項目は非表示となります。
このフィルタチェックボックスは、カーソルをコントロールの左上に置くと表示されます。



８ヶのデータ項目を表示



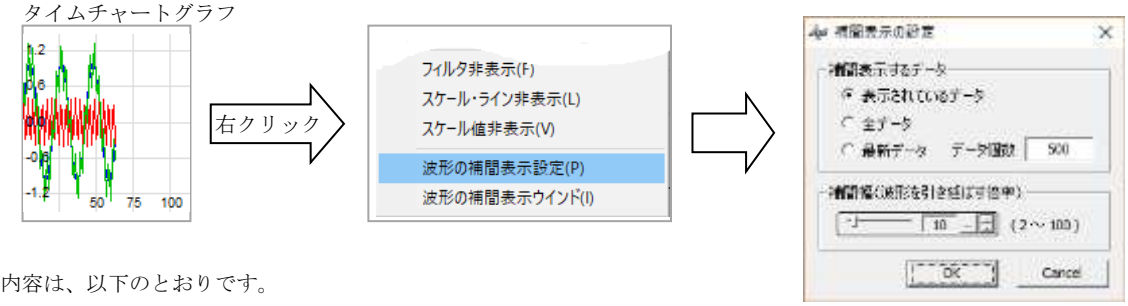
フィルタで、データ項目０，２，４，６を非表示

ポップアップメニューの「フィルタ非表示」／「フィルタ表示」を選択することにより、フィルタの表示を禁止／許可できます。

5.1.9. 波形の補間表示設定

波形の補間表示に関するパラメタを設定するには、タイムチャートグラフを右クリックし、ポップアップメニューから「波形の補間表示設定」を選択します。

タイムチャートグラフ



設定内容は、以下のとおりです。

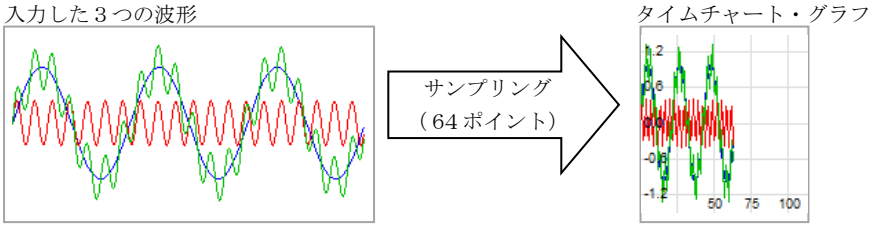
項目		内容
補間表示するデータ（補間表示対象とするデータの選択）	表示されているデータ	タイムチャート・コントロールで表示しているデータ
	全データ	バッファに格納されている全データ
	最新データ	最新のプロットデータ群
データ個数		「最新データ」選択時の、データ個数
補間幅		プロット点の表示間隔（ピクセル数）

5.1.10. 波形の補間表示

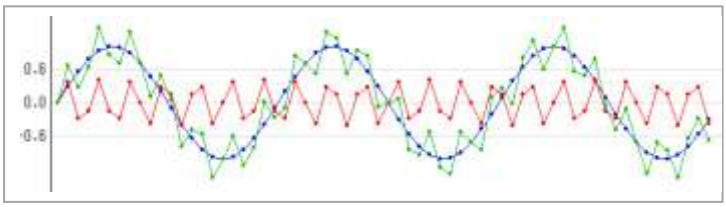
サンプル数が少なくてグラフに波形を正常に表示できない場合、サンプリングしたデータの間を補間することにより、本来の波形を再現して表示することができます。（３次スプライン曲線（サンプリング点を通る曲線）による補間表示）

以下に、波形補間表示の例を示します。

下図は、３つの波形を等間隔にサンプリングし、タイムチャートグラフで表示したものです。

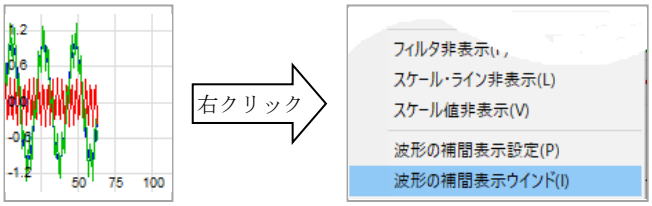


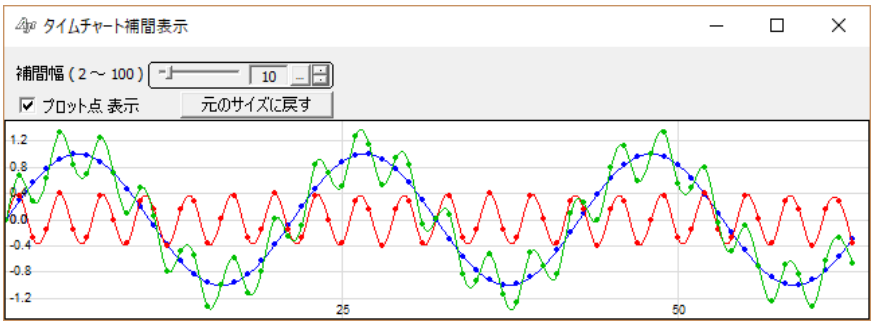
これでは、（タイムチャート・グラフから）元の波形を見ることはできません。
そこで、グラフを横に引き伸ばして、プロット点の間を線で結んでみます。（下図）



低い周波数の波形は、それなりに表示できますが、高い周波数の波形は依然として表示できていません。
（タイムチャート・コントロールには、横に引き伸ばしてプロット点を線で結ぶ機能はありません。上図は別途作成したものです。）

今度は、タイムチャートグラフを右クリックし、ポップアップメニューから「波形の補間表示ウインド」を選択します。





グラフを横に引き伸ばして、プロット点の間を（直線ではなく）曲線で補間したグラフが表示されます。
グラフ上の点は、プロットしたデータ（サンプリングデータ）を示します。
「プロット点表示」のチェックを外すと、プロット点を消去したグラフを表示します。（線だけの表示となります）
「補間幅」はプロット点の表示間隔（ピクセル数）です。（つまり、グラフを横に引き延ばす倍率となります）
「補間幅」を変更すると、プロット点の表示間隔を変更したグラフを再表示します。
「タイムチャート補間表示」ウインドは、初回表示時は（なるべく）元のグラフと同じサイズになるように表示し、以降、自由にサイズを変更することができます。
「元のサイズに戻す」ボタンを押すと、ウインドのサイズを初回に表示した時のサイズに設定し直します。

「タイムチャート補間表示」ウインドは、ポップアップメニューから「波形の補間表示ウインド」を選択した時点のデータを切り取って補間表示します。元のタイムチャートグラフを更新しても、補間表示は更新されません。
補間表示を更新するには、「タイムチャート補間表示」ウインドを一旦閉じて、再度表示し直してください。

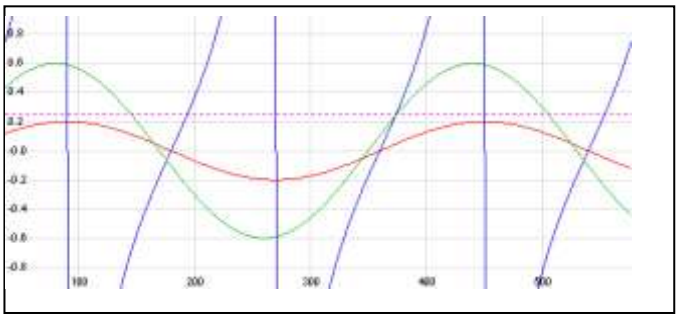
尚、補間表示対象データが少ない（８個未満）の場合は、補間表示できません。

5.1.11. 横線描画

特定のデータ値の位置に（最大８ヶの）横線を描画することができます。
横線の描画は、以下の関数により行ないます。

・AjcTchSetHLineAtt – 横線の属性（線種，色，太さ）の設定
・AjcTchSetHLinePos – 横線の描画位置
・AjcTchEnableHLine – 横線描画の許可／禁止

以下は、＋0.25 の位置に、紫色の点線を引いた例です。



```
AjcTchSetHLineAtt(hWndTc, 0, RGB(255, 0,
                                 255), 1,
                                 AJCTCH_DOT);
AjcTchSetHLinePos(hWndTc, 0, 0.25);
AjcTchEnableHLine(hWndTc, 0, TRUE);
```

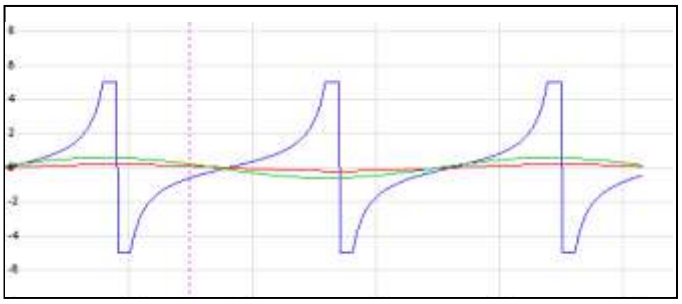
※ 点線等、実線以外の線を描画する場合は、線の太さ＝１でなければなりません。

5.1.12. 縦線描画

最後に格納したデータの位置に縦線を描画することができます。
縦線の描画は、AjcPutRealData() / AjcPutIntData()によりデータ投与後、以下の関数により行ないます。

・AjcTchSetVLine – 縦線の描画
・AjcTchEnableVLine – 縦線描画の許可／禁止

以下は、１５０個目のデータ位置に、紫色の点線を引いた例です。

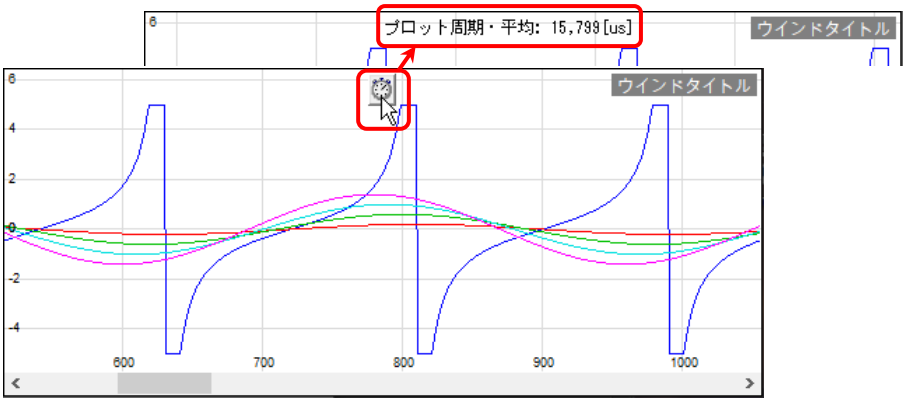


```
AjcTchSetVLine(hWndTc, RGB(255,0,255),
                       1, AJCTCH_DOT);
```

この縦線は、波形と同様にスクロールされます。

5.1.13. プロット周期表示

このコントロールでは、常時プロット周期を計測しています。
プロット周期とは、AjcTchPutIntData()／AjcTchPutRealData()によるデータ投与の間隔を意味します。
ウインド中央上部にカーソルを置くと現れる「 」ボタンを押すと、ボタンを押した時点のプロット周期(平均値)が表示されます。
プロット周期は５秒間だけ表示後、自動的に消えます。



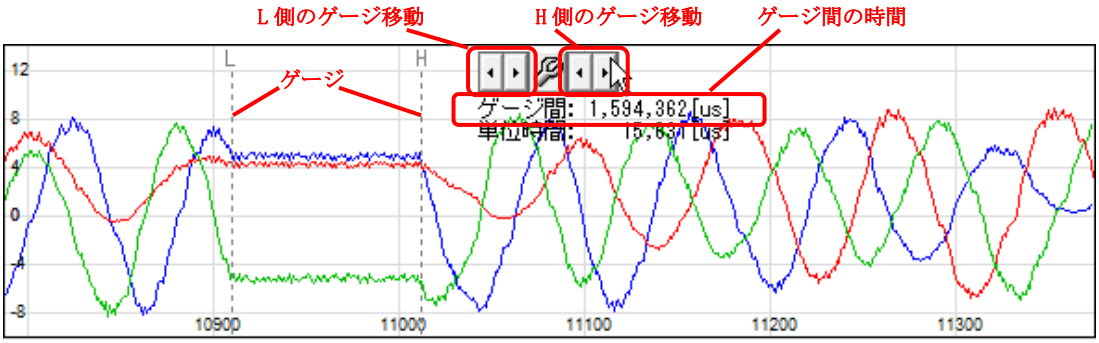
マウスのホイールボタンを押すと、プロット周期の計測をリセットします。

また、以下のＡＰＩでプロット周期の表示やリセットを行うことができます。

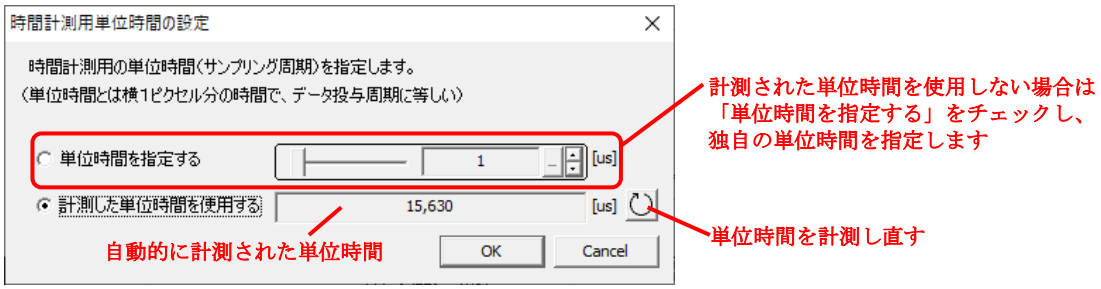
- AjcTchMesPeriShow() － プロット周期の表示
- AjcTchMesPeriReset() － プロット周期のリセット

5.1.14. 時間計測

２つのゲージを表示し、ゲージ間の時間を計測することができます。
Ctrl キーを押しながら、マウスのホイールボタンを押すと２つのゲージ（縦の点線）が表示されます。
ここで、ウインド中央上部にカーソルを置くと現れる２つの「 」ボタンでゲージを移動します。

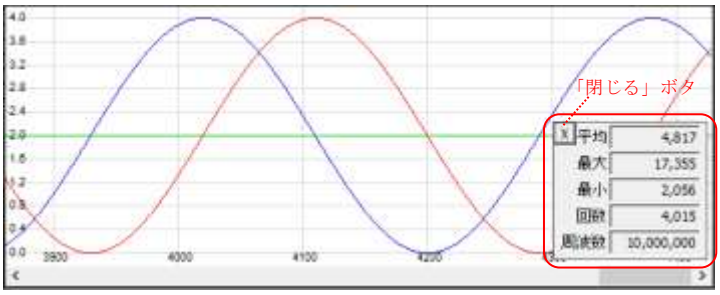


「 」ボタンを押すと、以下のダイアログにより単位時間（プロット周期）を設定することができます。



5.1.15. 描画時間情報表示

タイムチャートグラフを右クリックし、ポップアップメニューから「描画時間情報表示」を選択すると、タイムチャート・イメージの描画時間を計測し、右下に以下のように表示します。



平均：１回のグラフィック描画に要する時間の平均[μｓ]
最大：最大描画時間[μｓ]
最小：最小描画時間[μｓ]
回数：計測回数
周波数：計測周波数[Hz]　（ＰＣで固定なる値）

ウインドのサイズを変更した場合は、計測をやり直します。
短い周期でデータを投与すると、処理が重くなり、タイムチャートイメージをスムーズに表示できなくなります。
処理時間は、描画時間だけではありませんが、少なくとも、データを周期的に投与する場合は、平均値よりも長い周期で投与する必要があります。

※ 描画時間情報を表示するには、AjcTchEnableMesDraw()で、ポップアップメニュー上に「描画時間情報」を表示するように設定する必要があります。

5.1.16. ファイルやディレクトリのドラッグ＆ドロップ

本コントロールにファイルやディレクトリをドラッグ＆ドロップした場合は、WM_COMMAND メッセージにて、親ウインドへ以下の通知を行います。

・AJCTCN_DROPFILE ----- ファイルがドロップされたことを通知
・AJCTCN_DROPDIR ----- フォルダがドロップされたことを通知

これらの通知では、ドロップされたファイルやディレクトリの個数を通知します。
ファイルやディレクトリのパス名は、本コントロールのＡＰＩ「AjcTchGetDroppedFile(),AjcTchGetDroppedDir[Ex]()」にて取得します。

尚、タイムチャート・コントロールでファイルやディレクトリのドラッグ＆ドロップを有効とするには、拡張ウインドスタイルに「WS_EX_ACCEPTFILES」を指定する必要があります。

5.1.17. 表示の高速化

データ投与毎に表示＆スクロールすると表示処理に時間がかかります。
そこで、AjcTchPause()により一定期間の表示を抑止することで表示処理時間を短縮することができます。

```
      ：
AjcTchPause(TRUE);              // 表示停止
AjcTchPutRealData(・・・);      // データ投与
AjcTchPutIntData  (・・・);                    } ①（この間投与したデータは表示されない）
      ：
AjcTchPause(FALSE);             // 表示再開（①の期間に描画したデータを一気に表示）
      ：
```

AjcTchPause(FALSE) を実行すると、それまで表示を停止していたデータを一気に表示します。(全てのデータを表示＆スクロールするわけではなく、最終描画状態だけを表示します)
AjcTchPause(TRUE)～AjcTchPause(FALSE)の間描画していたデータはバッファに蓄えられていますので、通常どおりスクロールして見ることができます。

※AjcTchPause(TRUE)～AjcTchPause(FALSE)で表示を抑止している期間でも、ユーザ操作（ウインドのサイズを変えたり、最小化したタスクバーから戻す、等）によりウインドの再描画が必要な場合は、その瞬間の画面状態が表示されます。

5.2. コントロールのスタイル

タイムチャート・グラフ表示コントロールのスタイルは、以下のとおりです。

名称	ビット	値	内容	備考
AJCTCS_NOBORDER	7	0x0080	コントロールの外枠を表示しない	
AJCTCS_NOSCALELINE	6	0x0040	スケール・ラインを表示しない	
AJCTCS_NOSCALEVALUE	5	0x0020	スケール値を表示しない	
AJCTCS_NOFILTER	4	0x0010	フィルタ（左上のチェックボックス）を表示しない	
AJCTCS_NOSCROLLBAR	3	0x0008	スクロールバー非表示	ウインド生成時のみ有効

尚、タイムチャート・コントロールでファイルやディレクトリのドラッグ＆ドロップを有効とするには、以下の拡張ウインドスタイルを指定する必要があります。

名称	ビット	値	内容
WS_EX_ACCEPTFILES	4	0x0010	ドラッグ＆ドロップでファイルを受け付けるようにする

5.3. プロパティ構造体

タイムチャート・グラフ表示コントロールのプロパティは、以下の構造体で定義されます。

```
#define AJCTC_MAXBUF    16384
#define AJCTC_MAXITEM   8

typedef struct {
    double      ofs;                    //  オフセット（実際のデータ値に加える値）
    COLORREF    rgb;                    //  表示色
} AJCTCITEM, *PAJCTCITEM;

typedef struct {
    double      RngL, RngH;             //  グラフ上下の座標値
    UI          TmScale;                //  タイムスケール表示幅（スケール縦線の幅，０は縦線非表示）
    UI          MaxBuf;                 //  最大データ数
    UI          MaxItem;                //  最大アイテム数
    UI          AveNum;                 //  平均化個数
    AJCTCITEM   Item[AJCTC_MAXITEM];    //  アイテム・プロパティ
} AJCTCPROP, *PAJCTCPROP;
typedef const AJCTCPROP *PCAJCTCPROP;
```

各メンバ変数の内容は、以下のとおりです。

メンバ	内容	規定値	備考
RngL	グラフ低位のレンジ	-1.0	RngL<RngH である必要は無いが、RngL=RngH は不可
RngH	グラフ高位のレンジ	+1.0	
TmScale	横軸の目盛り表示幅	100	
MaxBuf	バッファ容量（２～１００００００）	1024	０を指定した場合は、ウインドサイズに合わせてバッファ容量を設定します
MaxItem	データ項目数（１～８）	3	**※１**
AveNum	平均化個数（１以上）	1	１：移動平均しない ２：移動平均の個数(N)　**（※２）**
ofs	各データ項目のオフセット値	全て 0.0	
rgb	各データ項目の表示色	[0] : 0xFF0000 [1] : 0x0000FF [2] : 0x00C000 [3] : 0xE0E000 [4] : 0xFF00FF [5] : 0x00E0E0 [6] : 0x808080 [7] : 0x000000	Bit23-16 : 青の成分 Bit15- 8 : 緑の成分 Bit 7- 0 : 赤の成分

※１：データ項目数を変更した場合、現在のデータは全て破棄されます。

※２：２以上を指定すると、過去のデータと合わせた(１～N 個の）データを平均化します。(移動平均による遅延は発生しません)

5.4. キャプション文字列によるプロパティの設定

CreateWindow()/CreateWindowEx()の lpszWindowName 引数（ウインドキャプション）あるいは、ダイアログデザイン時におけるコントロールの Caption プロパティにより、タイムチャート・グラフコントロールのプロパティを設定することができます。

パラメタ（キャプション文字列）の形式は、以下のとおりです。([XXX]は、XXX を省略可能であることを意味します)

```
P: [L=fff], [H=fff], [B=nnn], [I=nnn], [A=nnn], [BC=nnn], [0=[nnn][/nnn]], ・・・ [7=[nnn][/nnn]]
```

文字列の先頭は「P:」でなければなりません。(「P:」の直後には空白を置けます)
「fff」は実数で、「nnn」は整数（１６進数の場合は先頭に'0x'を付加）で指定します。
各パラメタはカンマ(,)で区切ります。(カンマの前後には空白を置けます)
各パラメタの指定順序は任意です。

各パラメタの設定内容は、以下のとおりです。

キーワード	内　容
L	低位グラフレンジ
H	高位グラフレンジ
B	バッファ容量（バッファに格納するデータ数）
I	有効なデータ項目数（１～８）
A	平均化個数
BC	コントロール外枠の表示色
0～7	各データ項目の情報を「オフセット値／表示色」の形式で指定

表示色は、１６進数で「0xbbggrr」の形式で指定します（bb:青成分，gg:緑成分，rr:赤成分）

設定例

```
P: L=-1000.0, H=1000.0
```
グラフレンジを、－１０００～＋１０００とする

```
P: B=4096, I=2, A=8
```
バッファに格納するデータ数を４０９６個とし、データ項目数＝２とする。
また、データを、８個毎に平均化する。

```
P: BC=0x0000FF
```
コントロールの外枠を赤色で表示する。

5.5. テキストの取得と設定

テキストを取得した場合は、プロパティ設定内容を表す文字列を返します。
デフォルトでの、取得テキストは、以下のとおりです。

```
P: L=-1, H=1, B=1024, I=3, A=1, BC=0x0, 0=0/0xFF0000, 1=0/0xFF, 2=0/0xC000, 3=0/0xE0E000, 4=0/0xFF00FF,
 5=0/0xE0E0, 6=0/0x808080, 7=0/0x0
```

テキストを設定する場合は、キャプション文字列によるプロパティの設定と同様に扱います。

5.6. プロパティの永続化

設定したプロパティを、プロファイル（.ini ファイル／レジストリ）に記録し、次回起動時に記録されているプロファイルを読み出すことにより、プロパティを永続的に有効とすることができます。

プロパティをプロファイルから読み出すには、ダイアログやウインドの初期化時に、AjcTchLoadProp()を実行します。
尚、初回実行時はプロファイルにプロパティが記録されていない為、AjcTchLoadProp()でプロパティのデフォルト値を指定します。
AjcTchLoadProp()でプロパティのデフォルト値を指定しない場合は、現在設定されているプロパティがデフォルト値となります。

デフォルトプロパティを指定する場合

```
AJCTCPROP DefProp;

DefProp.RngL = 10.0;
DefProp.RngH= 15.0;
   ・・・
AjcTchLoadProp(hwnd, "SectName", &DefProp);
```

デフォルトプロパティを指定しない場合

```
AjcTchSetRealRange(hwnd, 10.0, 15.0);
AjcTchLoadProp(hwnd, "SectName", NULL);
```

プロファイルにプロパティが記録されている場合は、AjcTchLoadProp()により読み出されたプロパティが設定される為、AjcTchSetRealRange()により設定された値は無効となります。（あらかじめ設定されているプロパティ値は初回ロード時のデフォルト値となりま

現在設定されているプロパティをプロファイルに記録するには、ダイアログやウインドの終了時に、AjcTchSaveProp()を実行します。

プロパティをプロファイルに記録

```
AjcTchSaveProp(hwnd, "SectName");
```

デフォルトでは、プロファイルの記録先は、レジストリになります。
プロファイルの記録先を初期化ファイル（自プログラムパス名の拡張子を「.ini」としたファイル）とする場合は、プログラムの最初（AjcTchLoadProp(), AjcTchSaveProp()を実行する前）に「AjcSetProfileIsRegistory(FALSE);」を実行してください。

プロファイルへのアクセスは、AjcGetProfile・・・() と AjcPutProfile・・・() により行います。
これらの関数仕様やプロファイルアクセスに関するその他の情報は、「プロファイル・アクセス」章を参照してください。

尚、AjcTchLoadPropEx()／AjcTchSavePropEx()を使用すれば、プロパティ値に加えて、フィルタ設定とウインドスタイルも永続化できます。

5.7. サポートＡＰＩ

タイムチャート・グラフ表示コントロールのサポートＡＰＩ一覧を以下に示します。

#	関数名	内容	備考
1	AjcTchStop AjcTchStart	グラフの表示を停止／開始	
2	AjcTchPurge	データ破棄	
3	AjcTchPutRealData AjcTchPutIntData	データ投与	
4	AjcTchShowBorder	コントロールの外枠の表示／非表示	
5	AjcTchShowFilter	フィルタ（チェックボックス）の表示／非表示	
6	AjcTchShowScale	スケール（目盛り線／値）の表示／非表示	
7	AjcTch{Set/Get}Prop	プロパティ設定／取得	
8	AjcTch{Set/Get}RealRange	グラフ・レンジの設定／取得	
9	AjcTchAdjustRange	グラフ・レンジを自動調整	
10	AjcTchSetBufSize	バッファサイズの設定	
11	AjcTchSetItemNumber	データ項目数の設定	
12	AjcTchSetAveNumber	平均化個数の設定	
13	AjcTchSetTimeScale	グラフ横軸スケール幅の設定	
14	AjcTchGetBitmap	ビットマップデータ取得	
15	AjcTchSetNtcRClk	右クリック通知設定	
16	AjcTch{Load/Save}Prop[Ex]	プロファイルからプロパティ値読出し／書込み	
17	AjcTchEnablePopupMenu	ポップアップメニューの許可／禁止	
18	AjcTch{Set/Get}TipText	ツールチップの設定／取得	
19	AjcTch{Set/Get}TipShowAlways AjcTchSetTipShowAlwaysAll	ツールチップ表示条件の設定／取得	
20	AjcTch{Set/Get}ChkBoxTipText	フィルタ・チェックボックス・ツールチップの設定／取得	
21	AjcTch{Set/Get}ChkBoxTipShowAlways	フィルタ・チェックボックス・ツールチップ表示条件の設定／取得	
22	AjcTch{Set/Get}MaxLineDist	最大結線長の設定／取得	
23	AjcTch{Set/Get}ScrollPos	スクロール位置の設定／取得	
24	AjcTch{Set/Get}Filter	フィルタの設定／取得	
25	AjcTchSetHLineAtt	横線の属性設定	
26	AjcTchSetHLinePos	横線の描画位置設定	
27	AjcTchEnableHLine	横線描画の許可／禁止	
28	AjcTchSetVLine	縦線の描画	
29	AjcTchEnableVLine	縦線描画の許可／禁止	
30	AjcTchSetPoint	プロット点の描画	
31	AjcTchEnablePoint	プロット点描画の許可／禁止	
32	AjcTchGetDroppedFile	ドロップされたファイル名取得	
33	AjcTchGetDroppedDir[Ex]	ドロップされたディレクトリ名取得	
34	AjcTchSetTitleText	タイトル文字列の設定	
35	AjcTch{Set/Get}IpInfo	波形補間表示情報の設定／取得	
36	AjcTchPause	画面表示の一時停止／再開	
37	AjcTchEnableMesDraw	描画時間計測情報の許可／禁止	
38	AjcTchMesPeriShow	プロット周期計測値の表示	
39	AjcTchMesPeriReset	プロット周期計測をリセット	
40	AjcTchMesPeriGet	プロット周期計測値取得	
41	AjcTchGetGauInfo	時間計測ゲージ情報取得	
42	AjcTch{Load・Save}PermInfo[Ex]	設定情報の永続化	
43	AjcTchSetTextFont	テキスト描画フォント設定	
44	AjcTchTextOut	テキスト描画（ピクセル位置指定）	
45	AjcTchPrintF	書式テキスト描画	
46	AjcTchGetText	描画テキスト取得	
47	AjcTchClear[All]Text	描画テキスト消去	
48	AjcTchClear	全てのデータ（プロットデータ，描画テキストテキスト）消去	
49	AjcTchSetEvtCallBack	イベント通知用コールバック関数の設定	

5.7.1. グラフ表示停止／開始(AjcTchStop/ AjcTchStart)

形　式 ： BOOL　AjcTchStop (HWND hwnd)； --- 停止
BOOL　AjcTchStart(HWND hwnd)； --- 開始

引　数 ： hwnd　- コントロールのウインドハンドル

説　明 ： グラフの表示を停止／開始します。
グラフの表示を停止した場合は、AjcTchStart()を実行するまでは、AjcTchPutRealData()やAjcTchPutIntData()により投与されたデータは無視（破棄）されます。

戻り値 ： TRUE　- 成功
FALSE – 失敗

5.7.2. データ破棄(AjcTchPurge)

形　式 ： BOOL　AjcTchPurge(HWND hwnd)；

引　数 ： hwnd　- コントロールのウインドハンドル

説　明 ： バッファに格納されているプロットデータを全て破棄します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

5.7.3. データ投与(AjcTchPutRealData)

形　式 ： BOOL　AjcTchPutRealData(HWND hwnd, double dat[])； --- 実数データ投与
BOOL　AjcTchPutIntData (HWND hwnd, int　dat[])； --- 整数データ投与

引　数 ： hwnd　- コントロールのウインドハンドル
dat　- グラフに投与するデータ配列のアドレス

説　明 ： コントロールに実数値／整数値でデータを投与します。
dat は、データ配列のアドレスで、配列の要素数は、AjcTchSetItemNumber()で設定されているデータ項目数（１～８）でなければなりません。

戻り値 ： TRUE　- 成功
FALSE - 失敗

5.7.4. コントロールの外枠表示 (AjcTchShowBorder)

形　式 ： BOOL　AjcTchShowBorder(HWND hwnd, BOOL fShow, COLORREF rgb)；

引　数 ： hwnd　- コントロールのウインドハンドル
fShow　- TRUE：表示, FALSE：非表示
rgb　- コントロール外枠の表示色（－１の場合は、表示色を変更しない）

説　明 ： コントロールの外枠を表示／非表示します。
コントロールの外枠は、デフォルトでは、黒色表示となっています。

戻り値 ： TRUE　- 成功
FALSE - エラー

5.7.5. フィルタ（チェックボックス）の表示／非表示 (AjcTchShowFilter)

形　式 ： BOOL　AjcTchShowFilter(HWND hwnd, BOOL fShow, COLORREF rgb)；

引　数 ： hwnd　- コントロールのウインドハンドル
fShow　- TRUE：表示, FALSE：非表示

説　明 ： コントロール左上のフィルタ（チェックボックス）を表示／非表示します。
初期状態での、フィルタ表示は、スタイル設定によります。

戻り値 ： TRUE　- 成功
FALSE - 失敗

5.7.6. スケールの表示／非表示 (AjcTchShowScale)

形　式 ： BOOL　AjcTchShowScale(HWND hwnd, BOOL fLine, BOOL fValue)；

引　数 ： hwnd　- コントロールのウインドハンドル
fLine　- TRUE：スケールライン表示, FALSE：非表示
fValue - TRUE：スケール値表示　　, FALSE：非表示

説　明 ： スケールライン（薄いグレーの目盛り線）とスケール値を表示／非表示します。
初期状態での、スケール表示は、スタイル設定によります。

戻り値 ： TRUE　- 成功
FALSE - 失敗

5.7.7. プロパティ設定／取得 (AjcTch{Set/Get}Prop)

形　式 ： BOOL　AjcTchSetProp(HWND hwnd, PCAJCTCPROP pProp)； -- プロパティ設定
BOOL　AjcTchGetProp(HWND hwnd, PAJCTCPROP　pBuf)； -- プロパティ取得

引　数 ： hwnd　- コントロールのウインドハンドル
pProp　- 設定するプロパティ情報のアドレス
pBuf　- 取得したプロパティ情報を格納するバッファのアドレス

説　明 ： コントロールのプロパティを設定／取得します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

5.7.8. グラフ・レンジの設定／取得(AjcTch{Set/Get}{Real|Int}Range)

形　式 ： BOOL　AjcTchSetRealRange (HWND hwnd, double　low, double　high)； -- グラフのレンジ設定（実数）
BOOL　AjcTchGetRealRange (HWND hwnd, double *low, double *high)； -- グラフのレンジ取得（実数）

BOOL　AjcTchSetIntRange (HWND hwnd, int　　low, int　　high)； -- グラフのレンジ設定（整数）
BOOL　AjcTchGetIntRange (HWND hwnd, int　　*low, int　　*high)； -- グラフのレンジ取得（整数）

引　数 ： hwnd　- コントロールのウインドハンドル
low　- 設定するグラフの低位レンジ値／低位のグラフレンジ値を格納するバッファのアドレス
high　- 設定するグラフの高位レンジ値／高位のグラフレンジ値を格納するバッファのアドレス

説　明 ： グラフのレンジを実数値／整数値で設定／取得します。
low は high より小さい数値である必要はありませんが、同じ値を指定することはできません。

戻り値 ： TRUE　- 成功
FALSE - 失敗

5.7.9. グラフ・レンジを自動調整(AjcTchAdjustRange)

形　式 ： BOOL　AjcTchAdjustRange (HWND hwnd)；

引　数 ： hwnd　- コントロールのウインドハンドル

説　明 ： フィルタで非表示となっているデータ項目を除く、全てのデータから最小値と最大値を算出し、±５％のマージンを持ってグラフのレンジ設定を行います。
バッファにデータが１つも無い場合は、何も行いません。
バッファ中のデータが全て同一値である場合は、当該値の－１～＋１の範囲でレンジを設定します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

5.7.10. バッファサイズの設定 (AjcTchSetBufSize)

形　式 ： BOOL　AjcTchSetBufSize (HWND hwnd, int n)；

引　数 ： hwnd　- コントロールのウインドハンドル
n　　- バッファサイズ（バッファに格納するデータ数, ２～１６３８４）

説　明 ： バッファサイズを、バッファに格納するデータ数で指定します
ｎ＝０とした場合は、ウインドのサイズに合わせてバッファ容量を設定します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

5.7.11. データ項目数の設定 (AjcTchSetItemNumber)

形　式 ： BOOL　AjcTchSetItemNumber (HWND hwnd, int n)；

引　数 ： hwnd　- コントロールのウインドハンドル
n　　- データ項目数（１～８）

説　明 ： データ項目数を設定します。
データ項目数とは、グラフに表示するデータの種別数を意味します。
デフォルトのデータ項目数は、３個です。

戻り値 ： TRUE　- 成功
FALSE - 失敗

5.7.12. 平均化個数の設定 (AjcTchSetAveNumber)

形　式 ： BOOL　AjcTchSetAveNumber (HWND hwnd, int n)；

引　数 ： hwnd　- コントロールのウインドハンドル
n　　- 平均化個数（１以上）

説　明 ： データの平均化個数を設定します。
AjcTchPutRealData()/AjcTchPutIntData()で、コントロールに投与した指定個数のデータの平均を算出し、この平均値をバッファに格納します。
つまり、AjcTchPutRealData()/AjcTchPutIntData()を指定回数コール毎にデータをバッファに格納することになります。

戻り値 ： TRUE　- 成功
FALSE - 失敗

5.7.13. グラフ横軸スケール幅の設定 (AjcTchSetTimeScale)

形　式 ： BOOL　AjcTchSetTimeScale (HWND hwnd, int n)；

引　数 ： hwnd　- コントロールのウインドハンドル
n　　- タイムスケール

説　明 ： タイムスケール（グラフ横軸の目盛り）幅を設定します。
ｎ＝０とした場合は、タイムスケールを表示しません。

戻り値 ： TRUE　- 成功
FALSE - 失敗

5.7.14. ビットマップデータ取得 (AjcTchGetBitmap)

形　式 ： HBITMAP　AjcTchGetBitmap (HWND hwnd)；

引　数 ： hwnd　- コントロールのウインドハンドル

説　明 ： 現在表示しているイメージのビットマップデータを取得します。

戻り値 ： ≠NULL - 成功（ビットマップハンドル）
＝NULL - 失敗

5.7.15. 右クリック通知設定 (AjcTchSetNtcRClk)

形　式 ： BOOL　AjcTchSetNtcRClk(HWND hwnd, BOOL fNtcRClk, UI MsgRBDown, UI MsgRBUp)；

引　数 ： hwnd　　- コントロールのウインドハンドル
fNtcRClk　- 右ボタンの DOWN/UP 通知フラグ（TRUE:通知する，FALSE:通知しない）
MsgRBDown　- 右ボタン押下時の通知メッセージコード　（０の場合は非通知）
MsgRBUp　　- 右ボタン離され時の通知メッセージコード（０の場合は非通知）

説　明 ： コントロールを右クリックした場合に、当該操作を親ウインドへ通知するか否かを設定します。
MsgRBDown，MsgRBUp は、WM_USER+100 以降，WM_APP+500 以降か、RegisterWindowMessage()で取得したコードを指定します。
各引数と、右クリック通知動作は以下のとおりです。

引数			Shift/ Ctrl	メッセージ	wParam	備考
fNtcRClk	MsgRBDown	MsgRBUp				
FALSE (default	－	－	未押下	WM_COMMAND	－（非通知）	ポップアップメニュー表示
			押下		ID + AJCTCN_RCLICK	
TRUE	いずれかが 0 以外		－	MsgRBDown(押下時)	WM_RBUTTONDOWN の wParam	MsgRBDown ＝０の場合は非通知
				MsgRBUp（離し時）	WM_RBUTTONUP　の wParam	MsgRBUp　＝０の場合は非通知
	0	0	－		－（非通知）	－

右クリックの通知を禁止するには、fNtcRClk=TRUE，MsgRBDown=0，MsgRBUp=0 とします。

戻り値 ： TRUE　- 成功
FALSE - 失敗

5.7.16. プロファイルからプロパティ値読出し／書込み (AjcTchLoadProp / AjcTchLoadPropEx)

形　式 ： BOOL AjcTchLoadProp (HWND hwnd, C_UTP pProfileSect, PCAJCTCPROP pDefProp); --- 読み出し
BOOL AjcTchLoadPropEx(HWND hwnd, C_UTP pProfileSect, PCAJCTCPROP pDefProp); --- 〃

BOOL AjcTchSaveProp (HWND hwnd, C_UTP pProfileSect); ------------------------- 書き込み
BOOL AjcTchSavePropEx(HWND hwnd, C_UTP pProfileSect); ------------------------ 〃

引　数 ： hwnd - コントロールのウインドハンドル
pProfileSect - プロファイル・セクション名（文字列）へのポインタ
pDefProp - デフォルトプロパティ値へのポインタ（現在の設定値をデフォルトとする場合は NULL）

説　明 ： AjcTchLoadProp[Ex]は、プロファイル（.ini ファイル／レジストリ）からプロパティ値を読み出して設定します。
AjcTchLoadPropEx は、プロパティ値に加えて、フィルタ設定値とウインドスタイルも読み出して設定します。
pDefProp は、プロファイルに当該プロパティ値が記録されていない場合の、デフォルト・プロパティ値を指定します。
pDefProp=NULL とした場合は、現在設定されているプロパティ値を、デフォルト・プロパティとして扱います。

AjcTchSaveProp[Ex]は、現在設定されているプロパティ値を、プロファイル（.ini ファイル／レジストリ）へ記録します。
AjcTchSavePropEx は、プロパティ値に加えて、フィルタ設定値とウインドスタイルも記録します。

AjcTchLoadProp(),AjcTchSaveProp()でプロファイルへセーブ／ロードする項目は以下のとおりです。

#	内容	キー名称	備考
1	グラフレンジ低位, 高位値	RngL, RngH	
2	横軸の目盛り表示幅	TmScale	バッファ行数
3	バッファ容量	MaxBuf	蓄積可能なデータ数
4	データ項目数	MaxItem	１～８
5	移動平均個数	AveNum	１　：移動平均無し ２～：移動平均個数
6	各データ項目のオフセット値	ofs0～ofs7	実数
7	各データ項目の表示色	rgb0～rgb7	0x00bbggrr

AjcTchLoadPropEx(),AjcTchSavePropEx()では、さらに以下の項目が追加されます。

#	内容	キー名称	備考
1	データ項目選択状態	FilterValue	チェックボックスの内容
2	スタイル値	WndStyle	
3	波形補間表示種別	IpInf.IpKnd	
4	最新データ表示時の最大データ数	IpInf.IpNum	
5	補間表示幅（波形の引き伸ばし倍率）	IpInf.IpWidth	

戻り値 ： TRUE - 成功
FALSE - 失敗

5.7.17. ポップアップメニューの許可／禁止 (AjcTchEnablePopupMenu)

形　式 ： BOOL AjcTchEnablePopupMenu (HWND hwnd, BOOL fEnable);

引　数 ： hwnd - コントロールのウインドハンドル
fEnable - ポップアップメニューの許可(TRUE)／禁止(FALSE)

説　明 ： 右クリックによるポップアップメニューを許可／禁止します。

戻り値 ： TRUE - 成功
FALSE - 失敗

備　考 ： 「AjcTchSetNtcRClk(hwnd, !fEnable, WM_RBUTTONDOWN, WM_RBUTTONUP);」を実行します。

5.7.18. ツールチップの設定／取得 (AjcTchSetTipText / AjcTchGetTipText)

形　式 ： BOOL AjcTchSetTipText(HWND hwnd, C_UTP pTxt); -------- ツールチップの設定
BOOL AjcTchGetTipText(HWND hwnd, UTP pBuf, UI lBuf); - ツールチップの取得

引　数 ： hwnd　　- コントロールのウインドハンドル
pTxt　　- ツールチップ文字列のアドレス（表示しない場合は NULL）
pBuf　　- ツールチップ文字列を格納するバッファのアドレス
lBuf　　- ツールチップ文字列を格納するバッファの文字数

説　明 ： コントロール上にカーソルを置いたときに表示するツールチップ（ツールヒント文字列）を設定／取得します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

5.7.19. ツールチップ表示条件の設定／取得(AjcTch{Set|Get}TipShowAlways)

形　式 ： BOOL AjcTchSetTipShowAlways(HWND hwnd, BOOL fShowAlways); ----- 設定
BOOL AjcTchGetTipShowAlways(HWND hwnd); ----------------------- 取得
BOOL AjcTchGetTipShowAlwaysAll(HWND hwnd, BOOL fShowAlways); --- すべて設定

引　数 ： hwnd　　- コントロールのウインドハンドル
fShowAlways　- ツールチップ表示条件（TRUE:非アクティブ時も表示,　FALSE:非アクティブ時は非表示）

説　明 ： ツールチップの表示条件（自プログラムが非アクティブ時の表示／非表示）を設定／取得します。
AjcTchGetTipShowAlwaysAll()は、フィルタチェックボックスを含めたすべてのツールチップ表示条件を設定します。

戻り値 ： 設定時：TRUE　- 成功　　取得時：ツールチップ表示条件
FALSE - 失敗

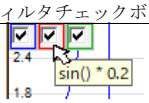
5.7.20. フィルタチェックボックス・ツールチップの設定／取得 (AjcTch{Set/Get}ChkBoxTipText)

形　式 ： BOOL AjcTchSetChkBoxTipText(HWND hwnd, UI n, C_UTP pTxt); --------- 設定
BOOL AjcTchGetChkBoxTipText(HWND hwnd, UI n, UTP pBuf, UI lBuf); -- 取得

引　数 ： hwnd　　- コントロールのウインドハンドル
n　　- チェックボックスのインデクス
pTxt　　- ツールチップ（ツールヒント）文字列のアドレス（表示しない場合は NULL）
pBuf　　- ツールチップ（ツールヒント）文字列を格納するバッファのアドレス
lBuf　　- ツールチップ（ツールヒント）文字列を格納するバッファの文字数

説　明 ： フィルタ・チェックボックス上にカーソルを置いたときに表示するツールヒント文字列を設定／取得します。
ｎは、チェックボックスのインデクスで、左から順に０～７を指定します。

フィルタチェックボックスツールチップ表示例



戻り値 ： TRUE　- 成功
FALSE - 失敗

5.7.21. フィルタチェックボックス・ツールチップ表示条件の設定／取得(AjcTch{Set|Get}ChkBoxTipShowAlways)

形　式 ： BOOL AjcTchSetChkBoxTipShowAlways(HWND hwnd, UI n, BOOL fShowAlways); --- 設定
BOOL AjcTchGetChkBoxTipShowAlways(HWND hwnd, UI n); ---------------------- 取得

引　数 ： hwnd　　- コントロールのウインドハンドル
n　　- チェックボックスのインデクス（0～15）
fShowAlways　- ツールチップ表示条件（TRUE:非アクティブ時も表示,　FALSE:非アクティブ時は非表示）

説　明 ： フィルタチェックボックスのツールチップの表示条件（自プログラムが非アクティブ時の表示／非表示）を設定／取得します。

戻り値 ： 設定時：TRUE　- 成功　　取得時：ツールチップ表示条件
FALSE - 失敗

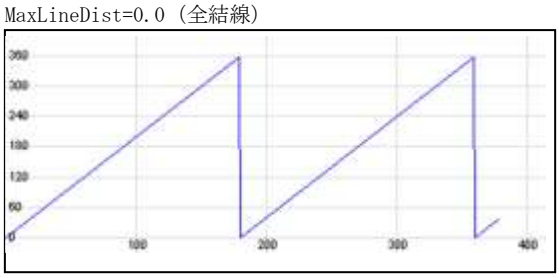
5.7.22. 最大結線長の設定／取得 (AjcTch{Set/Get}MaxLineDist)

形　式 ： BOOL　AjcTchSetMaxLineDist(HWND hwnd, double MaxLineDist); -- 設定
double　AjcTchGetMaxLineDist(HWND hwnd); ---------- 取得

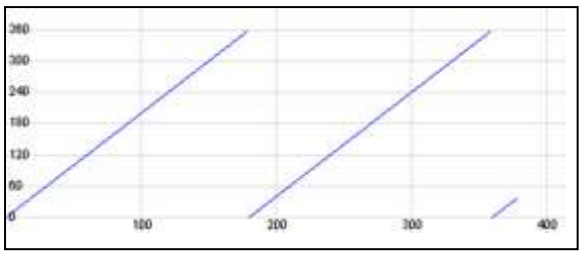
引　数 ： hwnd　　- コントロールのウインドハンドル
MaxLineDist　- 最大結線長（但し、０の場合は全て結線する）

説　明 ： ２つのデータの差により、グラフを結線する最大長を設定／取得します。
２つのデータの差が MaxLineDist より大きい場合は、グラフを結線しないように制御します。
例えば、回転角度（０～３５９度）をグラフ表示する場合、３５９→０度に変化した場合、旧バージョンでは左図のように３５９→０の変化を結線していましたが、例えば、MaxLineDist=180 とすることにより、右図のように３５９→０度の変化は（２つのデータの差が１８０より大きいため）結線しないようになります。

MaxLineDist=0.0（全結線）



MaxLineDist=180



MaxLineDist=0.0（デフォルト）の場合は、全てのデータを結線します。

戻り値 ： 設定時：TRUE　- 成功　　取得時：最大結線長
FALSE - 失敗

5.7.23. スクロール位置の設定／取得 (AjcTch{Set/Get}ScrollPos)

形　式 ： BOOL　AjcTchSetScrollPos(HWND hwnd, int pos); -- スクロール位置の設定
int　AjcTchGetScrollPos(HWND hwnd); ---------- スクロール位置の取得

引　数 ： hwnd　　- コントロールのウインドハンドル

説　明 ： スクロール位置を設定すると、pos で設定された位置までスクロールしてタイムチャートを表示します。
この関数でスクロール位置を設定する場合は、当該タイムチャートグラフのデータ更新を停止している状態で行ってください。
取得時は、現在のスクロール位置（グラフ左端のバッファ先頭からの相対位置）を取得します。

戻り値 ： 設定時：TRUE　- 成功　　取得時：スクロール位置（０～）
FALSE - 失敗

5.7.24. フィルタの設定／取得 (AjcTch{Set/Get}Filter)

形　式 ： BOOL　AjcTchSetFilter(HWND hwnd, UI n, BOOL state);
BOOL　AjcTchGetFilter(HWND hwnd, UI n);

引　数 ： hwnd　　- コントロールのウインドハンドル
n　　- チェックボックスのインデクス
state　　- 設定するフィルタの状態(FALSE:非表示 / TRUE:表示)

説　明 ： フィルタ・チェックボックスの設定（当該項目の表示／非表示の設定）／取得を行います。
ｎは、チェックボックスのインデクスで、左から順に０～７を指定します。

戻り値 ： 設定時：TRUE　- 成功　　取得時：フィルタの状態(FALSE:非表示 / TRUE:表示)
FALSE - 失敗

5.7.25. 横線の属性設定 (AjcTchSetHLineAtt)

形　式 ： BOOL AjcTchSetHLineAtt (HWND hwnd, UI id, COLORREF color, int width, int style);

引　数 ： hwnd - コントロールのウインドハンドル
id - 横線の識別（０～７）
color - 描画色
width - 線の太さ（０～）
style - 線種

説　明 ： タイムチャートに描画する横線の属性を設定します。
「style」は、線の種別であり、以下のいずれかを指定します。

- AJCTCH_SOLID - 実線
- AJCTCH_DASH - 破線
- AJCTCH_DOT - 点線
- AJCTCH_DASHDOT - １点鎖線
- AJCTCH_DASHDOTDOT - ２点鎖線
- AJCTCH_NULL - ヌル線（描画しないのと同じ）

点線等、実線以外の線を描画する場合は、線の太さ＝１(width=1)を指定しなければなりません。

戻り値 ： TRUE - 成功
FALSE - 失敗

5.7.26. 横線の描画位置設定 (AjcTchSetHLinePos)

形　式 ： BOOL AjcTchSetHLinePos (HWND hwnd, UI id, double pos);

引　数 ： hwnd - コントロールのウインドハンドル
id - 横線の識別（０～７）
pos - 描画位置

説　明 ： タイムチャートに描画する横線の位置を設定します。

戻り値 ： TRUE - 成功
FALSE - 失敗

5.7.27. 横線描画の許可／禁止(AjcTchEnableHLine)

形　式 ： BOOL AjcTchEnableHLine (HWND hwnd, UI id, BOOL fEnable);

引　数 ： hwnd - コントロールのウインドハンドル
id - 横線の識別（０～７）
fEnable - 描画指定（TRUE- 描画する，　FALSE- 描画しない）

説　明 ： タイムチャート上に横線を描画するか否かを指定します。

戻り値 ： TRUE - 成功
FALSE - 失敗

5.7.28. 縦線の描画 (AjcTchSetVLine)

形　式 ： BOOL AjcTchSetVLine (HWND hwnd, COLORREF color, int width, int style);

引　数 ： hwnd　　- コントロールのウインドハンドル
color　　- 描画色
width　　- 線の太さ（０～）
style　　- 線種

説　明 ： タイムチャートに描画する横線の属性を設定します。
縦線は、最後に投与したデータ位置に描画されます。
「style」は、線の種別であり、以下のいずれかを指定します。

・AJCTCH_SOLID　　- 実線
・AJCTCH_DASH　　- 破線
・AJCTCH_DOT　　- 点線
・AJCTCH_DASHDOT　　- １点鎖線
・AJCTCH_DASHDOTDOT　- ２点鎖線
・AJCTCH_NULL　　- ヌル線（描画しないのと同じ）

点線等、実線以外の線を描画する場合は、線の太さ＝１(width=1)を指定しなければなりません。

戻り値 ： TRUE　- 成功
FALSE - 失敗

5.7.29. 縦線描画の許可／禁止(AjcTchEnableVLine)

形　式 ： BOOL AjcTchEnableVLine (HWND hwnd, BOOL fEnable);

引　数 ： hwnd　　- コントロールのウインドハンドル
fEnable　　- 描画指定（TRUE- 描画する，　FALSE- 描画しない）

説　明 ： タイムチャート上に縦線を描画するか否かを指定します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

5.7.30. プロット点描画(AjcTchSetPoint)

形　式 ： BOOL AjcTchSetPoint (HWND hwnd, int r);

引　数 ： hwnd　　- コントロールのウインドハンドル
r　　- プロット点の半径（１～２５５）

説　明 ： 最後に投与したデータ位置にプロット点（円）を表示します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

5.7.31. プロット点の許可／禁止(AjcTchEnablePoint)

形　式 ： BOOL AjcTchEnablePoint (HWND hwnd, BOOL fEnable);

引　数 ： hwnd　　- コントロールのウインドハンドル
fEnable　　- 描画指定（TRUE- 描画する，　FALSE- 描画しない）

説　明 ： タイムチャート上にプロット点を描画するか否かを指定します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

5.7.32. ドロップされたファイル名取得(AjcTchGetDroppedFile)

形　式 ： BOOL AjcTchGetDroppedFile (HWND hwnd, UT buf[MAX_PATH]);

引　数 ： hwnd　　- コントロールのウインドハンドル
buf　　- ファイルのパス名を格納するバッファのアドレス

説　明 ： ドラッグ＆ドロップされたファイルのパス名を取得します。
本関数は、通知メッセージ（AJCTCN_DROPFILE）が通知された場合、ドロップされたファイルのパス名を取得するために実行します。
複数のファイルがドロップされた場合は、lParam で通知されたファイルの個数分だけ本関数を実行します。

戻り値 ： TRUE - ファイルのパス名をバッファに格納した
FALSE - ドロップされたファイル名なし

5.7.33. ドロップされたディレクトリ名取得(AjcTchGetDroppedDir[Ex])

形　式 ： BOOL AjcTchGetDroppedDir (HWND hwnd, UT buf[MAX_PATH]);
BOOL AjcTchGetDroppedDirEx (HWND hwnd, UT buf[MAX_PATH] , BOOL fTailIsDelimiter);

引　数 ： hwnd　　- コントロールのウインドハンドル
buf　　- ファイルのパス名を格納するバッファのアドレス
fTailIsDelimiter - TRUE ：ディレクトリパス名の末尾に「¥」を付加する
FALSE：ディレクトリパス名の末尾に「¥」を付加しない

説　明 ： ドラッグ＆ドロップされたディレクトリのパス名を取得します。
本関数は、通知メッセージ（AJCTCN_DROPDIR）が通知された場合、ドロップされたディレクトリのパス名を取得するために実行します。
複数のディレクトリがドロップされた場合は、lParam で通知されたディレクトリの個数分だけ本関数を実行します。

戻り値 ： TRUE - ディレクトリのパス名をバッファに格納した
FALSE - ドロップされたディレクトリ名なし

5.7.34. タイトル文字列の設定(AjcTchSetTitleText)

形　式 ： BOOL AjcTchSetTitleText (HWND hwnd, UTP pTitleText, COLORREF TextColor, COLORREF BackColor, HFONT hFont);

引　数 ： hwnd　　- コントロールのウインドハンドル
pTitleText - タイトルテキストへのポインタ（NULL 指定時は、タイトル非表示）
TextColor - テキスト描画色（－１指定時は前回指定値，デフォルト色は（白））
BackColor - 背景描画色　　（－１指定時は前回指定値，デフォルト色は（グレー））
hFont　　- タイトルテキストのフォント（NULL 指定時はデフォルトフォント）

説　明 ： コントロールウインドの右上にタイトルテキストを描画します。

戻り値 ： TRUE - 成功
FALSE - 失敗

5.7.35. 波形補間表示情報の設定／取得(AjcTch{Set/Get}IpInfo)

形　式 ： BOOL AjcTchSetIpInfo (HWND hwnd, HWND hwnd, PCAJCTCIPINFO pInfo);-- 波形補間表示情報の設定
BOOL AjcTchGetIpInfo(HWND hwnd, PAJCTCIPINFO pBuf); ---------- 波形補間表示情報の取得

引　数 ： hwnd　　- コントロールのウインドハンドル
pInfo　　- 設定する波形補間情報のアドレス
pBuf　　- 取得した波形補間情報を格納するバッファののアドレス

説　明 ： 波形補間情報（波形を補間表示する際のパラメタ）を設定／取得します。
波形補間情報は、以下の構造体です。

```
typedef struct {
    AJCTCIPKND  IpKnd;                  //  波形補間表示種別
    UI          IpNum;                  //  最新データ表示時の最大データ数
    UI          IpWidth;                //  補間表示幅（波形の引き伸ばし倍率）
} AJCTCIPINFO, *PAJCTCIPINFO;
typedef const AJCTCIPINFO *PCAJCTCIPINFO;
```

「IpKnd」は、補間表示するデータの指定であり、以下のいずれかを設定します。
- ・AJCTCIPK_WND - 現在ウインドに表示されているデータ
- ・AJCTCIPK_ALL - バッファに格納されている全データ
- ・AJCTCIPK_NEW – バッファに格納されている最新のデータ（データ個数は「IpNum」で指定）

戻り値 ： TRUE　- 成功
FALSE - 失敗

5.7.36. 画面表示の一時停止／再開 (AjcTchPause)

形　式 ： BOOL AjcTchPause (HWND hwnd, BOOL fPause);

引　数 ： hwnd　　- コントロールのウインドハンドル
fPause　　- 表示停止／再開フラグ（TRUE:停止, FALSE:再開）

説　明 ： 表示を一時停止／再開します。
fPause=TRUE を指定すると表示が一時停止します。表示一時停止中でも、データの投与は有効です。
fPause=FALSE を指定すると、表示を再開します。この時、表示停止中に投与されたデータは一気に表示されます。
（全てのデータを表示&スクロールするわけではなく、最終画面状態だけを表示します）

戻り値 ： TRUE　- 成功
FALSE - 失敗

5.7.37. 描画時間計測情報の許可／禁止 (AjcTchEnableMesDraw)

形　式 ： BOOL AjcTchEnableMesDraw (HWND hwnd, BOOL fEnable);

引　数 ： hwnd　　- コントロールのウインドハンドル
fEnable　- 許可／禁止フラグ（TRUE:許可, FALSE:禁止）

説　明 ： ポップアップメニュー上に「描画時間情報 表示／非表示」を表示するか否かを指定します。
fEnable=TRUE を指定するとポップアップメニュー上に「描画時間情報 表示／非表示」を表示します。
fEnable=FALSE を指定すると、ポップアップメニュー上に「描画時間情報 表示／非表示」を表示しません。

戻り値 ： TRUE　- 成功
FALSE - 失敗

5.7.38. プロット周期計測値の表示(AjcTchMesPeriShow)

形　式　： BOOL　AjcTchMesPeriShow (HWND hwnd, bool fShow);

引　数　： hwnd　　- コントロールのウインドハンドル
fShow　　- プロット周期計測値の表示フラグ（TRUE:表示, FALSE:非表示）

説　明　： プロット周期計測値を表示／非表示します。
fShow=true を指定した場合は、プロット周期計測値をマイクロ秒単位で５秒間表示後、自動的に消えます。

戻り値　： TRUE　- 成功
FALSE - 失敗

5.7.39. プロット周期計測をリセット (AjcTchMesPeriReset)

形　式　： BOOL　AjcTchMesPeriReset(HWND hwnd);

引　数　： hwnd　　- コントロールのウインドハンドル

説　明　： プロット周期の計測をリセットします。(計測をやり直す)

戻り値　： TRUE　- 成功
FALSE - 失敗

5.7.40. プロット周期計測値取得 (AjcTchMesPeriGet)

形　式　： UI　AjcTchMesPeriGet (HWND hwnd);

引　数　： hwnd　　- コントロールのウインドハンドル

説　明　： プロットデータの投与周期の平均値を取得します。
プロットデータの投与周期とは、AjcTchPut{Real/Int}Data()によるデータ投与間隔を意味します。
AjcTchMesPeriReset()が実行された場合は、その時点からのプロットデータの投与周期の平均値となります。

戻り値　： プロットデータ投与周期の平均値[us]

5.7.41. 時間計測ゲージ情報取得 (AjcTchMesPeriGet)

形　式　： UI　AjcTchGetGauInfo (HWND hwnd, PAJCTCGAUINFO pBuf);

引　数　： hwnd　　- コントロールのウインドハンドル
pBuf　　- 時間計測ゲージ情報を格納するバッファのアドレス

説　明　： 時間計測用のゲージ情報を取得します。
pBuf で示すバッファの形式は以下のとおりです。

```
typedef struct {
    BOOL        fGauEnable;         //  ゲージ許可フラグ（ゲージが表示されている状態を示す）
    UI          GauBarL;            //  ゲージ低位置
    UI          GauBarH;            //  ゲージ高位置
    BOOL        fGauUseMes;         //  単位時間値の種別（FALSE：設定値, TRUE：計測値）
    UI          GauSetUTime;        //  単位時間（設定値）[us]
} AJCTCGAUINFO, *PAJCTCGAUINFO;
typedef const AJCTCGAUINFO *PCAJCTCGAUINFO;
```

戻り値　： TRUE　- 成功
FALSE - 失敗

5.7.42. 設定情報の永続化 (AjcTchLoadPermInfo[Ex] / AjcTchSavePermInfo[Ex])

形　式 ： BOOL AjcTchLoadPermInfo (HWND hwnd, C_UTP pProfileSect, C_UTP pKeyPrefix); ---------- 永続化情報の読み出し
BOOL AjcTchLoadPermInfoEx(HWND hwnd, C_UTP pProfileSect, C_UTP pKeyPrefix, UI PermItem); - 〃
BOOL AjcTchSavePermInfo (HWND hwnd); ---------- 永続化情報の書き込み
BOOL AjcTchSavePermInfoEx(HWND hwnd, C_UTP pProfileSect, C_UTP pKeyPrefix, UI PermItem); - 〃

引　数 ： hwnd - コントロールのウインドハンドル
pProfileSect - 永続化情報を記録するプロファイルセクション名のアドレス
pKeyPrefix - 永続化情報を記録するプロファイルキー名先頭部分のアドレス
PermItem - 永続化情報項目の選択 (AJCVTH_PERM_*XXXXX*)

説　明 ： プロファイルから永続化情報の読み出し、あるいは、書き込みを行います。

AjcTchSavePermInfoEx()で PermItem=AJCTCH_PERM_BYLOAD を指定した場合は、AjcTchLoadPermInfoEx()で指定した PermItem が有効となります。

永続化する内容は、以下のとおりです

#	内容	キー末尾名称	備考	項目指定 (AJCTCH_PERM_*XXX*)
1	時間計測ゲージバーの低位値, 高位値	GauBarL, GauBarH		常時永続化　※1
2	単位時間の選択	fGauUseMes	0:指定値, 1:計測値	
3	単位時間	GauSetUTime		
4	データ項目選択状態	MskFilt	チェックボックスの内容	FILTER　※1
5	グラフレンジ低位値, 高位値	RngL, RngH		RANGE
6	バッファ容量	MaxBuf	データ数	OTHER
7	移動平均個数	AveNum		

※1 ： AjcTch{Load/Save}PermInfo()や、Ajc{Load/Save}{All/Grp}ControlSettings()で永続化する項目

戻り値 ： TRUE - 成功
FALSE - 失敗

5.7.43. テキスト描画フォント設定(AjcTchSetTextFont)

形　式 ： HFONT AjcTchSetTextFont (HWND hwnd, HFONT hFont);

引　数 ： hwnd - コントロールのウインドハンドル

説　明 ： AjcTchTextOut ()や、AjcTchPrintF()で描画するテキストのフォントを設定します。

戻り値 ： ≠NULL - 前回のフォントハンドル
＝NULL - 失敗

5.7.44. テキスト描画(AjcTchTextOut)

形　式 ： UI　AjcTchTextOut (HWND hwnd, int x, int y, C_UTP pTxt);

引　数 ： hwnd　- コントロールのウインドハンドル
x　　- 描画ピクセルX位置 (テキストを右隅／中央に表示する場合は、「AJCTXO_RIGHT ± n」or「AJCTXO_CENTER ± n」を指定)
y　　- 描画ピクセルY位置 (テキストを下隅／中央に表示する場合は、「AJCTXO_BOTTOM ± n」or「AJCTXO_CENTER ± n」を指定)
pTxt　- 描画するテキストへのポインタ

説　明 ： 指定したピクセル描画位置へテキストを表示します。
pTxt で指定する描画テキストには、エスケープシーケンス（「テキスト描画」の章参照）を含めることができます。
但し、パレット色（０～７）は、データ項目の描画色と同じとなります。

戻り値 ： ≠0 - テキストキー
＝0 - 失敗

5.7.45. 書式テキスト描画(AjcTchPrintF)

形　式 ： UI　AjcTchPrintF　(HWND hwnd, int x, int y, C_UTP pFmt, ...);

引　数 ： hwnd　- コントロールのウインドハンドル
x　　- 描画ピクセルX位置 (テキストを右隅／中央に表示する場合は、「AJCTXO_RIGHT ± n」or「AJCTXO_CENTER ± n」を指定)
y　　- 描画ピクセルY位置 (テキストを下隅／中央に表示する場合は、「AJCTXO_BOTTOM ± n」or「AJCTXO_CENTER ± n」を指定)
pFmt　- 書式テキストへのポインタ (printf()と同じ)

説　明 ： 指定したピクセル位置へ書式化したテキストを表示します。
描画するテキストには、エスケープシーケンス（「テキスト描画」の章参照）を含めることができます。
但し、パレット色（０～７）は、データ項目の描画色と同じとなります。

戻り値 ： ≠0 - テキストキー
＝0 - 失敗

5.7.46. 描画テキスト取得(AjcTchGetText)

形　式 ： UI　AjcTchGetText (HWND hwnd, UI key, UTP pBuf, UI lBuf);

引　数 ： hwnd　　- コントロールのウインドハンドル
key　　- テキストキー (AjcTchTextOut[V]()や、AjcTchPrintF[V]()の戻り値)
pBuf　　- 取得したテキストを格納するバッファへのポインタ (不要時は NULL)
lBuf　　- 取得したテキストを格納するバッファのバイト数／文字数

説　明 ： AjcTchTextOut()や、AjcTchPrintF()で描画した文字列を取得します。

戻り値 ： ≠0 - 描画テキストのバイト数／文字数
＝0 - 失敗

5.7.47. 描画テキスト消去(AjcTchClear[All]Text)

形　式 ： BOOL　AjcTchClearText　(HWND hwnd, UI key);　// １つの描画テキスト消去
BOOL　AjcTchClearAllText (HWND hwnd);　// すべての描画テキスト消去

引　数 ： hwnd　　- コントロールのウインドハンドル
key　　- テキストキー (AjcTchTextOut[V]()や、AjcTchPrintF[V]()の戻り値)

説　明 ： AjcTchTextOut()や、AjcTchPrintF()で描画した文字列を消去します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

5.7.48. 全てのデータ消去(AjcTchClear)

形　式 ： BOOL AjcTchClear (HWND hwnd);

引　数 ： hwnd　　- コントロールのウインドハンドル

説　明 ： 全てのプロットデータ、描画テキストデータを消去します。

戻り値 ： TRUE - 成功
FALSE - 失敗

5.7.49. イベント通知用コールバック関数の設定(AjcTchSetEvtCallBack)

形　式 ： BOOL AjcTchSetEvtCallBack (HWND hwnd, UX cbp, VO (CALLBACK *cbNtcEvt)(HWND hCtrl, UX wParam, UX lParam, UX cbp));

引　数 ： hwnd　　- コントロールのウインドハンドル
cbp　　- コールバックパラメタ
cbNtcEvt　- イベント通知用コールバック関数（NULL:コールバック関数を無効化）

説　明 ： イベント通知用のコールバック関数を設定します。
通常、イベントは WM_COMMAND メッセージで通知されますが、コールバック関数を設定することにより、コールバック関数の呼び出しにより通知されるようになります。

戻り値 ： TRUE - 成功
FALSE - 失敗

コールバック： コールバック関数の仕様は、以下のとおりです。

cbNtcEvt（イベント通知）

形　式 ： VO CALLBACK *cbNtcEvt*(HWND hCtrl, UX wParam, UX lParam, UX cbp);

引　数 ： hCtrl　- コントロールのウインドハンドル
wPatram - コントロール識別ＩＤ，通知コード（WM_COMMAND による通知と同じ）
lPatram - イベントパラメタ　　（WM_COMMAND による通知と同じ）
cbp　　- コールバックパラメタ

説　明 ： 各種イベントを通知します。（詳細は「コントロールの通知メッセージ」を参照）

戻り値 ： なし

5.8. 通知情報の取得ＡＰＩ

コントロールからの通知メッセージ（WM_COMMAND）に伴うパラメタを取得するＡＰＩ群です。
「AjcSetCmdWithHdl(TRUE);」を実行した場合、各通知メッセージ（WM_COMMAND）の lParam は通知内容に伴うパラメタは通知されず、lParam＝コントロールのウインドハンドルとなります。
この場合、通知内容に伴うパラメタは以下のＡＰＩで取得します。

#	関 数 名	内容	対応する通知	
1	AjcTchGetNtcRng	グラフレンジ情報の取得	AJCTCN_RANGE	グラフレンジ通知
2	AjcTchGetNtcScrPos	スクロール位置の取得	AJCTCN_SCRPOS	スクロール位置通知
3	AjcTchGetNtcFiles	ドロップしたファイル数の取得	AJCTCN_DROPFILE	ファイルドロップ通知
4	AjcTchGetNtcDirs	ドロップしたディレクトリ数の取得	AJCTCN_DROPDIR	ディレクトリロップ通知
5	AjcTchGetNtcRClk	右クリック通知情報の取得	AJCTCN_RCLICK	右クリック通知

5.8.1. グラフレンジ情報の取得（AjcTchGetNtcRng）

形　式　：　PAJCTC_NTC_RANGE　AjcTchGetNtcRng (HHWND hwnd);

引　数　：　hwnd　　- コントロールのウインドハンドル

説　明　：　グラフレンジ通知（AJCTCN_RANGE）時の、グラフレンジ情報を取得します。

戻り値　：　グラフレンジ情報（AJCTC_NTC_RANGE）へのポインタ

5.8.2. スクロール位置の取得（AjcTchGetNtcScrPos）

形　式　：　UI　AjcTchGetNtcScrPos (HHWND hwnd);

引　数　：　hwnd　　- コントロールのウインドハンドル

説　明　：　スクロール位置通知（AJCTCN_SCRPOS）時のスクロール位置を取得します。

戻り値　：　スクロール位置

5.8.3. ドロップしたファイル数の取得（AjcTchGetNtcFiles）

形　式　：　UI　AjcTchGetNtcFiles (HHWND hwnd);

引　数　：　hwnd　　- コントロールのウインドハンドル

説　明　：　ファイルドロップ通知（AJCTCN_DROPFILE）時の、ドロップされたファイルの個数を取得します。

戻り値　：　ドロップされたファイルの個数

5.8.4. ドロップしたディレクトリ数の取得（AjcTchGetNtcDirs）

形　式　：　UI　AjcTchGetNtcDirs (HHWND hwnd);

引　数　：　hwnd　　- コントロールのウインドハンドル

説　明　：　ディレクトリドロップ通知（AJCTCN_DROPDIR）時の、ドロップされたディレクトリの個数を取得します。

戻り値　：　ドロップされたディレクトリの個数

5.8.5. 右クリック通知情報の取得（AjcTchGetNtcRClk）

形　式 ： PAJCTCRCLK AjcTchGetNtcRClk (HHWND hwnd);

引　数 ： hwnd　　- コントロールのウインドハンドル

説　明 ： 右クリック通知（AJCTCN_RCLICK）時の、右クリック通知情報を取得します。

戻り値 ： 右クリック通知情報（AJCTCRCLK）へのポインタ

5.9. コントロールの通知メッセージ

コントロールからの通知メッセージは、WM_COMMAND メッセージにより通知されます。
WM_COMMAND メッセージの wParam には以下の情報が設定されます。

・LOWORD(wParam) - コントロールの識別ＩＤ
・HIWORD(wParam) - 通知メッセージ

通知メッセージコードは、以下のとおりです。

#	通知メッセージコード	内容	備考
1	AJCTCN_RANGE	グラフレンジ通知	
2	AJCTCN_SCRPOS	スクロール位置通知	
3	AJCTCN_CLEAR	データクリアー通知	
4	AJCTCN_DBLCLK	ダブルクリック通知	
5	AJCTCN_DROPFILE	ファイルドロップ通知	
6	AJCTCN_DROPDIR	ディレクトリロップ通知	
7	AJCTCN_RCLICK	右クリック通知	SHIFT/CTRL + 右クリック時

5.9.1. グラフレンジ通知 (AJCTCN_RANGE)

説　明 ： グラフのレンジが変更されたことを通知します。
lParam には、変更後のレンジ情報のアドレスが設定されます。レンジ情報の形式は以下のとおりです。

```
typedef struct {
    double  RngL;       //  グラフレンジ低位値
    double  RngH;       //  グラフレンジ高位値
} AJCTC_NTC_RANGE, *PAJCTC_NTC_RANGE;
```

パラメタ： wParam - コントロール識別ＩＤと、通知メッセージコード（AJCTCN_RANGE）
lParam - レンジ値情報のアドレス（PAJCTC_NTC_RANGE）（MFC 使用時はコントロールのウインドハンドル）

戻り値 ： なし

5.9.2. スクロール位置通知 (AJCTCN_SCRPOS)

説　明 ： スクロールバー操作により、スクロール位置が変更されたことを通知します。

パラメタ： wParam - コントロール識別ＩＤと、通知メッセージコード（AJCTCN_SCRPOS）
lParam - 現在のスクロール位置（MFC 使用時はコントロールのウインドハンドル）

戻り値 ： なし

5.9.3. データクリアー通知 (AJCTCN_CLEAR)

説　明 ： データがクリアーされたことを通知します。

パラメタ： wParam - コントロール識別ＩＤと、通知メッセージコード（AJCTCN_CLEAR）
lParam – コントロールのウインドハンドル

戻り値 ： なし

5.9.4. ダブルクリック通知（AJC3DGN_DBLCLK）

説　明　：　ダブルクリックされたことを通知します。

パラメタ：　wParam – コントロールの識別ＩＤと通知メッセージコード（AJCTCN_DBLCLK）
lParam – ダブルクリック位置（ｘ：下位ワード，ｙ：上位ワード）（MFC 使用時はコントロールのウインドハンドル）

戻り値　：　なし（ゼロを返してください）

5.9.5. ファイルドロップ通知（AJCTCN_DROPFILE）

説　明　：　ドラッグ＆ドロップ操作で、ファイルがドロップされたことを通知します。
本通知では、ドロップされたファイルの個数だけを通知します。
ドロップされたファイルのパス名は、AjcTchGetDroppedFile() により取得します。

パラメタ：　wParam – コントロールの識別ＩＤと通知メッセージコード（AJCTCN_DROPFILE）
lParam – ドロップされたファイルの個数（MFC 使用時はコントロールのウインドハンドル）

戻り値　：　なし（ゼロを返してください）

5.9.6. ディレクトリドロップ通知（AJCTCN_DROPDIR）

説　明　：　ドラッグ＆ドロップ操作で、ディレクトリがドロップされたことを通知します。
本通知では、ドロップされたディレクトリの個数だけを通知します。
ドロップされたディレクトリのパス名は、AjcTchGetDroppedDir[Ex]()により取得します。

パラメタ：　wParam – コントロールの識別ＩＤと通知メッセージコード（AJCTCN_DROPDIR）
lParam – ドロップされたディレクトリの個数（MFC 使用時はコントロールのウインドハンドル）

戻り値　：　なし（ゼロを返してください）

5.9.7. 右クリック通知（AJCTCN_RCLICK）

説　明　：　右クリックしたことを通知します。
右クリック通知の可否については、AjcTchSetNtcRClk()を参照してください。
lParam で以下の右クリック情報を通知します。

```
typedef struct {
    int         x;              //  カーソルｘ座標
    int         y;              //  カーソルｙ座標
    BOOL        fShift;         //  SHIFT キー押下フラグ
    BOOL        fCtrl;          //  CTRL キー押下フラグ
} AJCTCRCLK, *PAJCTCRCLK;
```

パラメタ：　wParam – コントロールの識別ＩＤと通知メッセージコード（AJCTCN_RCLICK）
lParam – 右クリック情報へのポインタ（PAJCTCRCLK）（MFC 使用時はコントロールのウインドハンドル）

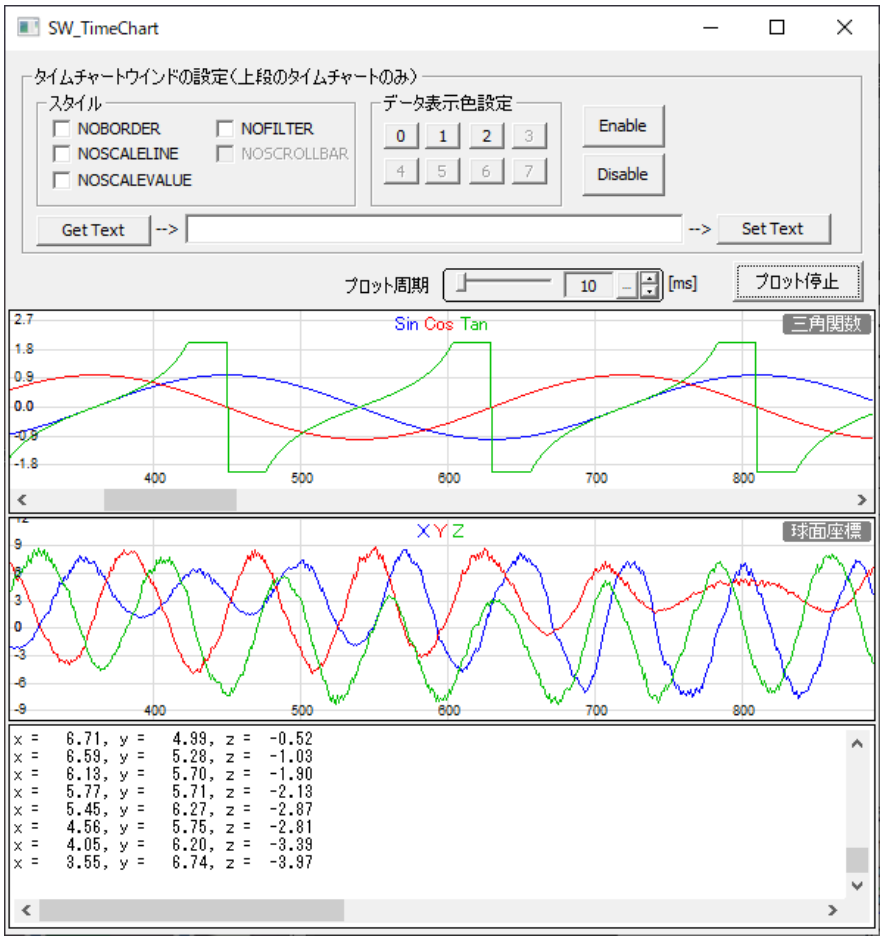
戻り値　：　なし（ゼロを返してください）

5.10. サンプルプログラム

5.10.1. SW_TimeChart（２つのタイムチャート）

このサンプルプログラムは、２つのタイムチャートウインドを表示します。
上段のタイムチャートは、単に、三角関数の値をプロットします。
下段のタイムチャートは、ランダムなサンプルデータをプロットします。

「プロット停止」ボタンでプロットを停止し、上段のタイムチャートのスクロールボタンを動かすと、下段のタイムチャートも連動してスクロールします。




```
 1 : //
 2 : //  SW_TimeChart.c
 3 : //
 4 : #include    <AjrCstXX.h>
 5 : #include    <math.h>
 6 : #include    <tchar.h>
 7 : #include    "resource.h"
 8 :
 9 : //-------------------------------------------------------------------------------------------------------------//
10 : //  マクロ                                                                                                     //
11 : //-------------------------------------------------------------------------------------------------------------//
12 : //  スタイル設定チェックボックス
13 : #define     SET_STYLE(STY)      MAjcSetWindowLong(hWndTc1, GWL_STYLE,                                   ¥
14 :                                 (int)(AjcGetDlgItemChk(hDlg, IDC_CHK_##STY) ?                           ¥
15 :                                     (MAjcGetWindowLong(hWndTc1, GWL_STYLE) | AJCTCS_##STY) :            ¥
16 :                                     (MAjcGetWindowLong(hWndTc1, GWL_STYLE) & ~AJCTCS_##STY)))
17 :
18 : //-------------------------------------------------------------------------------------------------------------//
19 : //  ワーク                                                                                                     //
20 : //-------------------------------------------------------------------------------------------------------------//
21 : HINSTANCE       hInst;                      //  ＤＬＬインスタンスハンドル
22 : HWND            hDlgMain;                   //  ダイアログボックスハンドル
23 : HWND            hWndTc1, hWndTc2;           //  タイムチャート・グラフコントロールのウインドハンドル
24 : HWND            hWndVth;                    //  ログ表示ウインド
25 : double          theta     = 0.0;            //  データ生成角度
26 : int             DlgWidth;                   //  ダイアログボックスの幅
```

```
27 : int             DlgHeight;                    //  ダイアログボックスの高さ
28 : int             ySrtPos;                      //  グラフ表示先頭位置
29 : BOOL            fBusy = FALSE;                //  プロット中フラグ
30 :
31 : //  球の中心と半径
32 : #define     CX      0.2
33 : #define     CY      0.4
34 : #define     CZ      0.1
35 : #define     R       8.0
36 :
37 : //  プロット点演算パラメタ
38 : //                              vCent      radius  noise  xrot  yrot  pitch
39 : static  AJCSPD_PARAM    prm = {{CX, CY, CZ}, R,    7.0,   0.5,  0.8,  9.0};
40 :
41 : //  プロット点演算インスタンス
42 : HAJCSPD     hSpd = {NULL};
43 :
44 : //-------------------------------------------------------------------------------------------------------------//
45 : //  内部サブ関数                                                                                               //
46 : //-------------------------------------------------------------------------------------------------------------//
47 : AJC_DLGPROC_DEF(Main);
48 : static  VO      NtcFromTch(HWND hTch, WPARAM wParam, LPARAM lParam);
49 :
50 : //=============================================================================================================//
51 : //                                                                                                             //
52 : //  W i n M a i n                                                                                              //
53 : //                                                                                                             //
54 : //=============================================================================================================//
55 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
56 : {
57 :     MSG     msg;
58 :
59 :     hInst = hInstance;
60 :
61 :     //----- メイン・ダイアログオープン ----------------//
62 :     hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_MAIN), NULL, AJC_DLGPROC_NAME(Main));
63 :     ShowWindow(hDlgMain, SW_SHOW);
64 :
65 :     //----- メッセージループ --------------------------//
66 :     while (GetMessage(&msg, NULL, 0, 0)) {
67 :         do {
68 :             if (IsDialogMessage(hDlgMain, &msg)) break;
69 :             TranslateMessage(&msg);
70 :             DispatchMessage (&msg);
71 :         } while (0);
72 :     }
73 :
74 :     return (int)msg.wParam ;
75 : }
76 : //=============================================================================================================//
77 : //                                                                                                             //
78 : //  ダイアログ・プロシージャ                                                                                   //
79 : //                                                                                                             //
80 : //=============================================================================================================//
81 : //----- ダイアログ初期化 ----------------------------------------------------------------------------------//
82 : AJC_DLGPROC(Main, WM_INITDIALOG     )
83 : {
84 :     RECT    rd, rt;
85 :     //----- タイムチャートコントロールのハンドル設定 -------//
86 :     hWndTc1 = GetDlgItem(hDlg, IDC_TCH_SINCOS);
87 :     hWndTc2 = GetDlgItem(hDlg, IDC_TCH_SAMPLE);
88 :     hWndVth = GetDlgItem(hDlg, IDC_VTH_LOG   );
89 :     //----- ダイアログサイズ設定 ----------------------------//
90 :     GetWindowRect(hDlg, &rd);
91 :     DlgWidth  = rd.right    - rd.left;
92 :     DlgHeight = rd.bottom - rd.top;
93 :     //----- グラフ表示先頭位置設定 --------------------------//
94 :     GetWindowRect(hWndTc1, &rt);
95 :     MapWindowPoints(NULL, hDlg, (LPPOINT)&rt, 2);
96 :     ySrtPos = rt.top;
97 :     //----- タイトル設定 ------------------------------------//
98 :     AjcTchSetTitleText(hWndTc1, TEXT(" 三角関数 "), -1, -1, NULL);
99 :     AjcTchSetTitleText(hWndTc2, TEXT(" 球面座標 "), -1, -1, NULL);
100 :    //----- ツールチップ設定 --------------------------------//
101 :    AjcTchSetTipText(hWndTc1, TEXT("三角関数(Sin, Cos, Tan)の値をチャート表示"));
102 :    AjcTchSetTipText(hWndTc2, TEXT("球面を移動する球面座標データをチャート表示"));
103 :    //----- プロットデータ演算初期化 ------------------------//
104 :    hSpd = AjcSpdCreate(0);
105 :    AjcSpdSetParam(hSpd, &prm);
106 :    return TRUE;
```

```
107 : }
108 : //----- ウインド破棄 ----------------------------------------------------------------------------------------//
109 : AJC_DLGPROC(Main, WM_DESTROY        )
110 : {
111 :     //  プロットデータ演算終了
112 :     AjcSpdDelete(hSpd);
113 :     //  プログラム終了
114 :     PostQuitMessage(0);
115 :     return TRUE;
116 : }
117 : //----- サイズ変更中 ----------------------------------------------------------------------------------------//
118 : AJC_DLGPROC(Main, WM_SIZING         )
119 : {
120 :     LPRECT  p = (VOP)lParam;
121 :
122 :     if (p->right - p->left < DlgWidth) {
123 :         p->right  = p->left + DlgWidth;
124 :     }
125 :     if (p->bottom - p->top < DlgHeight) {
126 :         p->bottom = p->top  + DlgHeight;
127 :     }
128 :
129 :     return TRUE;
130 : }
131 : //----- サイズ変更 ------------------------------------------------------------------------------------------//
132 : AJC_DLGPROC(Main, WM_SIZE           )
133 : {
134 :     int     w = LOWORD(lParam);
135 :     int     h = HIWORD(lParam);
136 :     int     tw, th;
137 :
138 :     tw = w - 4;
139 :     th = (h - ySrtPos) / 3 - 2;
140 :
141 :     MoveWindow(hWndTc1, 2, ySrtPos               , tw, th, FALSE);
142 :     MoveWindow(hWndTc2, 2, ySrtPos + (th    ) + 2, tw, th, FALSE);
143 :     MoveWindow(hWndVth, 2, ySrtPos + (th * 2) + 4, tw, th, FALSE);
144 :     InvalidateRect(hDlg, NULL, TRUE);
145 :
146 :     return TRUE;
147 : }
148 : //----- タイマ ----------------------------------------------------------------------------------------------//
149 : AJC_DLGPROC(Main, WM_TIMER          )
150 : {
151 :     double  val[3];
152 :
153 :     if (wParam == 1) {
154 :         if (!AjcGetDlgItemChk(hDlg, IDC_CHK_STOP)) {
155 :             //  Sin, Cos, Tan 値データ生成／投与
156 :             val[0] = sin(theta);
157 :             val[1] = cos(theta);
158 :             val[2] = tan(theta);
159 :             val[2] = __min(val[2],  2.0);
160 :             val[2] = __max(val[2], -2.0);
161 :             AjcTchPutRealData(hWndTc1, val);
162 :             theta += 1.0 / 180.0 * AJC_PAI;
163 :             //  ランダムなプロットデータ生成／投与
164 :             AjcSpdCalcV(hSpd, (PAJC3DVEC)&val);
165 :             AjcTchPutRealData(hWndTc2, val);
166 :             //  データをログ表示
167 :             AjcVthPrintF(hWndVth, TEXT("x = %6.2f, y = %6.2f, z = %6.2f¥n"), val[0], val[1], val[2]);
168 :         }
169 :     }
170 :     return TRUE;
171 : }
172 : //----- コマンド（カラー設定）-------------------------------------------------------------------------------//
173 : AJC_DLGPROC(Main, WM_COMMAND        )
174 : {
175 :     int         cmd = LOWORD(wParam);
176 :     int         ix  = cmd - IDC_CMD_COLOR0;
177 :     COLORREF    CustColors[16];
178 :     CHOOSECOLOR cc;
179 :     AJCTCPROP   prop;
180 :
181 :     if (cmd >= IDC_CMD_COLOR0 && cmd <= IDC_CMD_COLOR7) {
182 :         AjcTchGetProp(hWndTc1, &prop);
183 :         memset(&cc, 0, sizeof cc);
184 :         cc.lStructSize  = sizeof(CHOOSECOLOR);
185 :         cc.hwndOwner    = hDlg;
186 :         cc.lpCustColors = CustColors;
```

```
187 :         cc.rgbResult    = prop.Item[ix].rgb;
188 :         cc.Flags        = CC_RGBINIT;
189 :         if (ChooseColor(&cc)) {
190 :             prop.Item[ix].rgb = cc.rgbResult;
191 :             AjcTchSetProp(hWndTc1, &prop);
192 :         }
193 :     }
194 :     return TRUE;
195 : }
196 : //----- プロット開始／停止ボタン ----------------------------------------------------------------------//
197 : AJC_DLGPROC(Main, IDC_CMD_START         )
198 : {
199 :     if (HIWORD(wParam) == BN_CLICKED) {
200 :         if (!fBusy) {
201 :             fBusy = TRUE;
202 :             AjcTchStart(hWndTc1);
203 :             AjcTchStart(hWndTc2);
204 :             SetTimer(hDlg, 1, AjcGetDlgItemUInt(hDlg, IDC_INP_PERIOD), NULL);
205 :             AjcSetDlgItemStr(hDlg, IDC_CMD_START, TEXT("プロット停止"));
206 :         }
207 :         else {
208 :             fBusy = FALSE;
209 :             AjcTchStop(hWndTc1);
210 :             AjcTchStop(hWndTc2);
211 :             KillTimer(hDlg, 1);
212 :             AjcSetDlgItemStr(hDlg, IDC_CMD_START, TEXT("プロット開始"));
213 :         }
214 :     }
215 :     return TRUE;
216 : }
217 : //----- 周期設定 --------------------------------------------------------------------------------------//
218 : AJC_DLGPROC(Main, IDC_INP_PERIOD        )
219 : {
220 :     if (HIWORD(wParam) == AJCIVN_INTVALUE) {
221 :         if (fBusy) {
222 :             SetTimer(hDlg, 1, (int)lParam, NULL);
223 :         }
224 :     }
225 :     return TRUE;
226 : }
227 : //----- Get Text ボタン -------------------------------------------------------------------------------//
228 : AJC_DLGPROC(Main, IDC_CMD_GETTEXT       )
229 : {
230 :     UT      txt[1024];
231 :
232 :     if (HIWORD(wParam) == BN_CLICKED) {
233 :         GetWindowText(hWndTc1, txt, AJCTSIZE(txt));
234 :         AjcSetDlgItemStr(hDlg, IDC_TXT_GETTEXT, txt);
235 :     }
236 :     return TRUE;
237 : }
238 : //----- Set Text ボタン -------------------------------------------------------------------------------//
239 : AJC_DLGPROC(Main, IDC_CMD_SETTEXT       )
240 : {
241 :     UT      txt[1024];
242 :
243 :     if (HIWORD(wParam) == BN_CLICKED) {
244 :         AjcGetDlgItemStr(hDlg, IDC_TXT_GETTEXT, txt, AJCTSIZE(txt));
245 :         SetWindowText(hWndTc1, txt);
246 :     }
247 :     return TRUE;
248 : }
249 : //----- Enable ボタン ---------------------------------------------------------------------------------//
250 : AJC_DLGPROC(Main, IDC_CMD_ENABLE        )
251 : {
252 :     if (HIWORD(wParam) == BN_CLICKED) {
253 :         EnableWindow(hWndTc1, TRUE);
254 :     }
255 :     return TRUE;
256 : }
257 : //----- Disable ボタン --------------------------------------------------------------------------------//
258 : AJC_DLGPROC(Main, IDC_CMD_DISABLE       )
259 : {
260 :     if (HIWORD(wParam) == BN_CLICKED) {
261 :         EnableWindow(hWndTc1, FALSE);
262 :     }
263 :     return TRUE;
264 : }
265 : //----- スタイル設定チェックボックス ------------------------------------------------------------------//
266 : AJC_DLGPROC(Main, IDC_CHK_NOBORDER      )   {SET_STYLE(NOBORDER     );  return TRUE;}
```

```
267 : AJC_DLGPROC(Main, IDC_CHK_NOSCALELINE   )   {SET_STYLE(NOSCALELINE  );  return TRUE;}
268 : AJC_DLGPROC(Main, IDC_CHK_NOSCALEVALUE  )   {SET_STYLE(NOSCALEVALUE );  return TRUE;}
269 : AJC_DLGPROC(Main, IDC_CHK_NOFILTER      )   {SET_STYLE(NOFILTER     );  return TRUE;}
270 : AJC_DLGPROC(Main, IDC_CHK_NOSCROLLBAR   )   {SET_STYLE(NOSCROLLBAR  );  return TRUE;}
271 : //----- タイムチャート・グラフ コントロール(SIN, COS 波形) ---------------------------------------------------//
272 : AJC_DLGPROC(Main, IDC_TCH_SINCOS    )
273 : {
274 :     if (HIWORD(wParam) == AJCTCN_SCRPOS) {
275 :         AjcSetDlgItemChk(hDlg, IDC_CHK_STOP, TRUE);
276 :         AjcTchSetScrollPos(hWndTc2, (UI)lParam);
277 :     }
278 :     else {
279 :         NtcFromTch(hWndTc1, wParam, lParam);
280 :     }
281 :     return TRUE;
282 : }
283 : //----- タイムチャート・グラフ コントロール(サンプルデータ波形) -----------------------------------------------//
284 : AJC_DLGPROC(Main, IDC_TCH_SAMPLE    )
285 : {
286 :     NtcFromTch(hWndTc2, wParam, lParam);
287 :     return TRUE;
288 : }
289 : //-----  「Cancel」  ---------------------------------------------------------------------------------------//
290 : AJC_DLGPROC(Main, IDCANCEL          )
291 : {
292 :     DestroyWindow(hDlg);
293 :     return TRUE;
294 : }
295 : //---------------------------------------------------------------------------------------------------------//
296 : AJC_DLGMAP_DEF(Main)
297 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG          )
298 :     AJC_DLGMAP_MSG(Main, WM_DESTROY             )
299 :     AJC_DLGMAP_MSG(Main, WM_SIZING              )
300 :     AJC_DLGMAP_MSG(Main, WM_SIZE                )
301 :     AJC_DLGMAP_MSG(Main, WM_TIMER               )
302 :     AJC_DLGMAP_MSG(Main, WM_COMMAND             )
303 :
304 :     AJC_DLGMAP_CMD(Main, IDC_CMD_START          )
305 :     AJC_DLGMAP_CMD(Main, IDC_INP_PERIOD         )
306 :     AJC_DLGMAP_CMD(Main, IDC_CMD_GETTEXT        )
307 :     AJC_DLGMAP_CMD(Main, IDC_CMD_SETTEXT        )
308 :     AJC_DLGMAP_CMD(Main, IDC_CMD_ENABLE         )
309 :     AJC_DLGMAP_CMD(Main, IDC_CMD_DISABLE        )
310 :     AJC_DLGMAP_CMD(Main, IDC_CHK_NOBORDER       )
311 :     AJC_DLGMAP_CMD(Main, IDC_CHK_NOSCALELINE    )
312 :     AJC_DLGMAP_CMD(Main, IDC_CHK_NOSCALEVALUE   )
313 :     AJC_DLGMAP_CMD(Main, IDC_CHK_NOFILTER       )
314 :     AJC_DLGMAP_CMD(Main, IDC_CHK_NOSCROLLBAR    )
315 :
316 :     AJC_DLGMAP_CMD(Main, IDC_TCH_SINCOS         )
317 :     AJC_DLGMAP_CMD(Main, IDC_TCH_SAMPLE         )
318 :     AJC_DLGMAP_CMD(Main, IDCANCEL               )
319 : AJC_DLGMAP_END
320 : //---------------------------------------------------------------------------------------------------------//
321 : //  タイムチャートコントロールからの通知処理                                                               //
322 : //---------------------------------------------------------------------------------------------------------//
323 : static  VO      NtcFromTch(HWND hTch, WPARAM wParam, LPARAM lParam)
324 : {
325 :     UT      txt[4096] = {0};
326 :
327 :     switch (HIWORD(wParam)) {
328 :         case AJCTCN_RANGE:                          //  ●レンジ通知
329 :         {   PAJCTC_NTC_RANGE p = (VOP)lParam;
330 :             AjcSnPrintF(txt, AJCTSIZE(txt), TEXT("レンジが設定されました。 (Low = %.3f, High = %.3f)"), p->RngL, p->RngH);
331 :             break;
332 :         }
333 :         case AJCTCN_RCLICK:                         //  ●右クリック通知
334 :         {   PAJCTCRCLK  p = (PAJCTCRCLK)lParam;
335 :             AjcSnPrintF(txt, AJCTSIZE(txt), TEXT("%s%s 右クリック発生 (x = %d, y = %d)"),
336 :                                                                     p->fShift ? TEXT("Shift + ") : TEXT(""),
337 :                                                                     p->fCtrl  ? TEXT("Ctrl + " ) : TEXT(""),
338 :                                                                     p->x, p->y);
339 :             break;
340 :         }
341 :         case AJCTCN_DROPDIR:                        //  ●ディレクトリドロップ
342 :         {   UT      path[MAX_PATH];
343 :             MAjcStrCpy(txt, AJCTSIZE(txt), TEXT("ディレクトリがドロップされました。¥n"));
344 :             while (AjcTchGetDroppedDir(hTch, path)) {
345 :                 MAjcStrCat(txt, AJCTSIZE(txt), TEXT("  "));
346 :                 MAjcStrCat(txt, AJCTSIZE(txt), path);
```

```
347 :                 MAjcStrCat(txt, AJCTSIZE(txt), TEXT("¥n"));
348 :             }
349 :             break;
350 :         }
351 :         case AJCTCN_DROPFILE:                       //  ●ファイルドロップ
352 :         {   UT      path[MAX_PATH];
353 :             MAjcStrCpy(txt, AJCTSIZE(txt), TEXT("ファイルがドロップされました。¥n"));
354 :             while (AjcTchGetDroppedFile(hTch, path)) {
355 :                 MAjcStrCat(txt, AJCTSIZE(txt), TEXT("  "));
356 :                 MAjcStrCat(txt, AJCTSIZE(txt), path);
357 :                 MAjcStrCat(txt, AJCTSIZE(txt), TEXT("¥n"));
358 :             }
359 :             break;
360 :         }
361 :     }
362 :     //  チップ表示
363 :     if (txt[0] != 0) {
364 :         SIZE    sz;
365 :         RECT    r;
366 :         int     w,h;
367 :         GetWindowRect(hTch, &r);
368 :         w = r.right - r.left;
369 :         h = r.bottom - r.top;
370 :         AjcTipTextGetSize(txt, NULL, TRUE, &sz);
371 :         AjcTipTextShow(r.left + (w - sz.cx) / 2, r.top + (h - sz.cy) / 2, txt, -1, NULL);
372 :     }
373 : }
```

6.　３Ｄ／２Ｄグラフィック・コントロール（AjcCtrl3dGraph クラス）

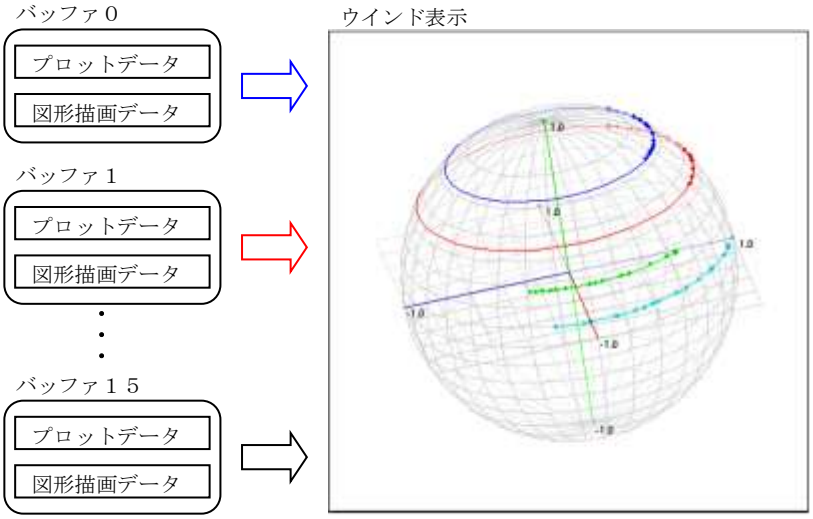
３次元座標系（Ｘ，Ｙ，Ｚ座標）や、２次元座標系（Ｘ，Ｙ座標）上に、球，楕球，立方体，長方体，点やライン等を描画するコントロールです。

このコントロールは、主に実験データのプロッティングや、演算結果をグラフに描画してグラフィカルに確認することを目的としています。(OpenGL や Direct-X のようにシェーデイングやテクスチャマッピングといた高度な機能はありません)

プロットデータや図形描画データは、１６ヶのバッファに分けて格納されます。

各バッファには、バッファ０＝青，バッファ１＝赤・・・といった表示色が割り当てられ、各バッファのデータはそれぞれ割り当てられた色で表示されます。(各バッファの表示色は変更可能です)

また、３Ｄグラフの場合は、視点から見て、中心の手前側と、中心より向こう側のデータを色分けし、遠近感を表現することができます。(下の表示例では、中心から向こう側の色（青と赤）が少し薄く表示されています)



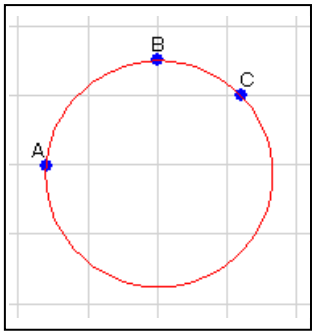
中心で交差する３つのラインは、座標軸(X:青，Y:赤，Z:緑)を意味します。

6.1. プロットデータと図形描画データ

図形描画データとは、図形（球体，円，点や線など）の描画データであり、古いデータを破棄することなく蓄積され続けます。
プロットデータとは、投与した座標位置を点で描画するデータであり、蓄積する個数に制限を設け、制限個数を超えた場合は、古いデータから順次破棄されます。　つまり、プロットデータは、制限された個数の最新データだけを表示することになります。
プロットデータ間を線で結ぶこともできます。

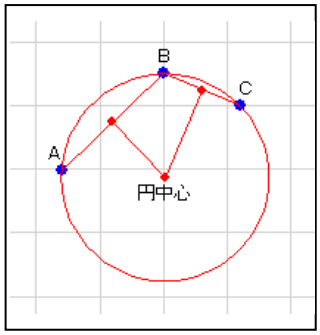
図形の描画は、主に、プロットしたサンプリングデータから、何らかの計算結果を図に表してみることを目的としています。

例えば、（２次元で）サンプリングした３つのプロットデータから、円を求めるとします。
求めた円を、以下のように画面に表示してみることができます。



３つの青い点は、サンプリングされた点を意味します。
この３つの点から円を算出し表示します。

さらに、円算出の過程も表示してみます。



円の中心は、線分Ａ−Ｂと、線分Ｂ−Ｃの中点からの垂線を算出し、
２つの垂線の交点であることを示します。
尚、円の半径は、円中心から３点(A,B,C)のいずれかまでの長さとなりま

参考までに、上記の計算と表示のサンプルコードを示しておきます。

```
#define     BLUE    0
#define     RED     1

・・・・・

    AJC2DVEC    vA = {1234, 350};   //  ３つのプロット点（ここでは定数とする）
    AJC2DVEC    vB = {1250, 365};   //  ・
    AJC2DVEC    vC = {1262, 360};   //  ・
    AJC2DVEC    vo0, vo1;           //  ２つのベクトル(A->B, b->C)
    AJC2DLVEC   vl0, vl1;           //  ２つの線分の中点と垂線
    AJC2DVEC    vCent;              //  円の中心
    double      r;                  //  円の半径

    AjcG2dPixelV(hWndCtrl, BLUE,  &vA, 4);      //  ３つのプロット点表示
    AjcG2dPixelV(hWndCtrl, BLUE,  &vB, 4);      //  ・
    AjcG2dPixelV(hWndCtrl, BLUE,  &vC, 4);      //  ・

    AjcG2dTextOutV(hWndCtrl, AJCG2DTXOMD_ABOVE_LEFT  , &vA, TEXT("A"));                 //  ３点の名前（A，B，C）表示
    AjcG2dTextOutV(hWndCtrl, AJCG2DTXOMD_ABOVE_CENTER, &vB, TEXT("B"));                 //  ・
    AjcG2dTextOutV(hWndCtrl, AJCG2DTXOMD_ABOVE_RIGHT , &vC, TEXT("C"));                 //  ・

    AjcV2dSub (&vB, &vA, &vl0.p);           AjcV2dSub (&vC, &vB, &vl1.p);           //  ２つの線分の中点算出
    AjcV2dMult(&vl0.p, 0.5, &vl0.p);        AjcV2dMult(&vl1.p, 0.5, &vl1.p);        //  ・
    AjcV2dAdd (&vA, &vl0.p, &vl0.p);        AjcV2dAdd (&vB, &vl1.p, &vl1.p);        //  ・

    AjcV2dSub(&vB, &vA, &vo0);              AjcV2dSub(&vC, &vB, &vo1);              //  ２つの中点から垂線を算出
    AjcV2dAnyOrthoVec(&vo0, &vl0.v);        AjcV2dAnyOrthoVec(&vo1, &vl1.v);        //  ・

    AjcV2dCrossL2L(&vl0, &vl1, &vCent);                                             //  ２つの垂線の交点算出

    AjcG2dLineV (hWndCtrl, RED, &vA, &vB);      AjcG2dLineV (hWndCtrl, RED, &vB, &vC);      //  線分 A-B, B-C 表示
    AjcG2dPixelV(hWndCtrl, RED, &vl0.p, 3);     AjcG2dPixelV(hWndCtrl, RED,  &vl1.p, 3);    //  線分の中点表示
    AjcG2dLineV (hWndCtrl, RED, &vl0.p, &vCent);                                    //  ２つの中点からの垂線表示
    AjcG2dLineV (hWndCtrl, RED, &vl1.p, &vCent);                                    //  ・
    AjcG2dPixelV(hWndCtrl, RED, &vCent, 3);                                         //  円の中心表示
    AjcG2dTextOutV(hWndCtrl, AJCG2DTXOMD_BELLOW_CENTER, &vCent, TEXT("円中心"));    //  ・

    r = AjcV2dDistP2P(&vCent, &vA);                                                 //  円の半径算出
    AjcG2dEllipseV(hWndCtrl, RED, &vCent, r, r);                                    //  円描画
```

本ライブラリを使用すれば、上記のような計算／表示をリアルタイムに連続して行うことができます。

6.2. 機能概要

6.2.1. ポップアップメニュー

グラフ上で右クリックすると、以下のポップアップメニューが表示されます。

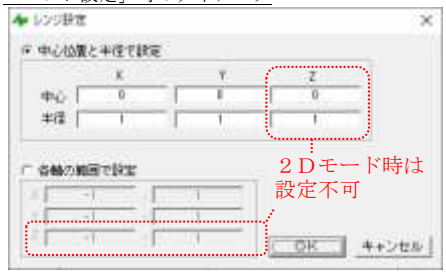
３Ｄモード時のポップアップメニュー



２Ｄモード時のポップアップメニュー



「レンジ設定」時のダイアログ



「任意の平面表示／平面のアングル設定」時のダイアログ



各メニューの内容は、以下のとおりです。

#	メニュー	内容
1	コピー	グラフ表示内容（ビットマップ）をクリップボードへコピーします
2	レンジ設定	グラフのレンジ設定ダイアログを表示します
3	レンジ自動調整	データから最大値／最小値を検索し、レンジを自動的に設定します。
4	各軸のレンジを同一サイズにする	中心位置は変更せずに、各軸のレンジ幅を同一にします。（最大のレンジ幅に合わせる）
5	アスペクト比をウインドサイズに合わせる アスペクト比を１固定にする	アスペクト比を設定します。
6	フィルタ表示／非表示	コントロール左上のフィルタ（チェックボックス）を表示／非表示します
7	Ｘ－Ｙ座標面	Ｘ－Ｙ平面を表示するように視点を設定します
8	Ｘ－Ｚ座標面	Ｘ－Ｚ平面を表示するように視点を設定します
9	Ｙ－Ｚ座標面	Ｙ－Ｚ平面を表示するように視点を設定します
10	３Ｄ座標面	３Ｄイメージを表示するように視点を設定します
11	任意の平面表示／平面のアングル設定 （設定ダイアログ表示）	３Ｄモード時は、任意の平面を表示します。 ２Ｄモード時は、平面のアングルを設定します。 表示平面の種類(X-Y / X-Z / Y-Z)や、横軸／縦軸の向き(昇順／降順)も設定できます。
12	方眼スケール表示	#14～16 で表示設定された平面に、方眼スケールの表示／非表示
13	同心円スケール表示	#14～16 で表示設定された平面に、同心円スケールの表示／非表示
14	球体スケール表示	球体スケールの表示／非表示
15	ＸＹ平面スケール表示	ＸＹ平面に方眼／同心円スケールの表示を許可／禁止
16	ＸＺ平面スケール表示	ＸＺ平面に方眼／同心円スケールの表示を許可／禁止
17	ＹＺ平面スケール表示	ＹＺ平面に方眼／同心円スケールの表示を許可／禁止
18	奥行き表現禁止	奥行き表現（原点より前側と向こう側で表示色を変える）の許可／禁止
19	データクリアー	全てのプロットデータと描画データを破棄します（画面をクリアーします）
20	描画時間情報表示	グラフィックの描画に要する時間を計測し表示します (AjcG3dEnableMesDraw()で、描画時間計測情報の表示を許可した場合に表示)

コントロールにフォーカスがある状態で、F1 キーを押した場合も ポップアップメニューを表示します。

6.2.2. レンジ設定

ポップアップメニューで「レンジ設定」を選択すると、右図のダイアログボックスが表示されます。

ここで、各軸の中心位置と半径／範囲を設定し、ＯＫボタンを押すと、グラフレンジが設定されます。

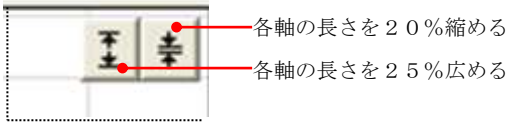
「Cancel」ボタンを押すと、設定を中止します。

「各軸の範囲で設定」を選択し、各軸の最小値と最大値で設定することもできます。



6.2.3. ワンタッチでレンジ設定

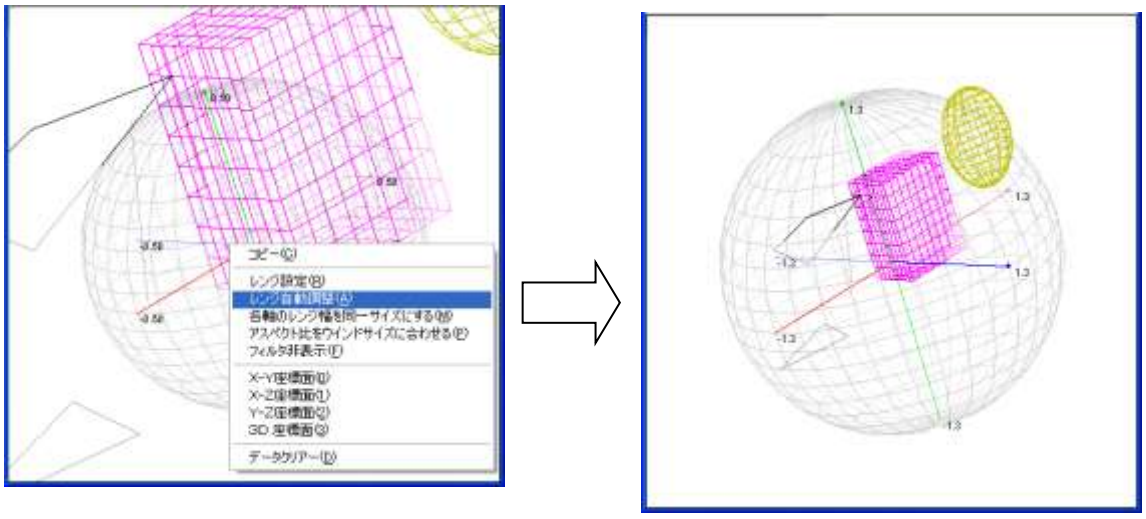
マウスカーソルをコントロールの右上隅に置くと、２つのボタンが表示されます。
これらのボタンで、各軸の長さ（直径）を広めたり、縮めたりすることができます。



6.2.4. レンジ自動調整

ポップアップメニューで「レンジ自動調整」を選択すると、（フィルタで非表示となっている項目を除く）全てのデータから、最大値と最小値を算出し、（中心位置は変わらないように）±５％のマージンを持ってレンジ設定を行います。

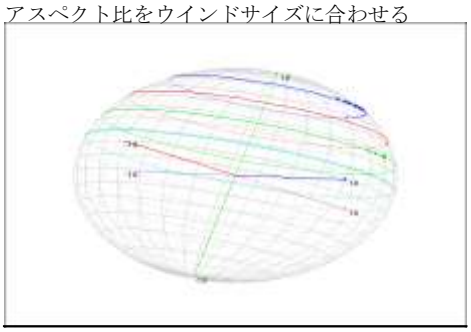
つまり、描画されている全データが視界に入るようにレンジを設定します。



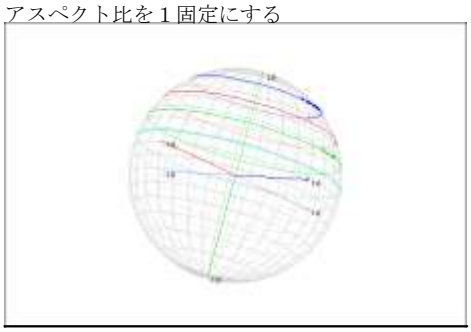
6.2.5. アスペクト比の設定

ポップアップメニューにより、ウインドサイズに合わせてアスペクト比を可変にするか、アスペクト比を１固定(デフォルト))にするかを設定できます。

アスペクト比をウインドサイズに合わせる



アスペクト比を１固定にする



「アスペクト比をウインドサイズに合わせる」を選択した場合は、ウインドの縦／横サイズが異なると、図形が歪んだ形となります。

「アスペクト比を１固定にする」を選択した場合は、アスペクト比を１固定とし、各軸のレンジ（各軸の長さ）を一定にそろえることにより、図形を歪みのない形で表示できます。

アスペクト比は、プロパティ（fAspect1）によっても設定可能です。

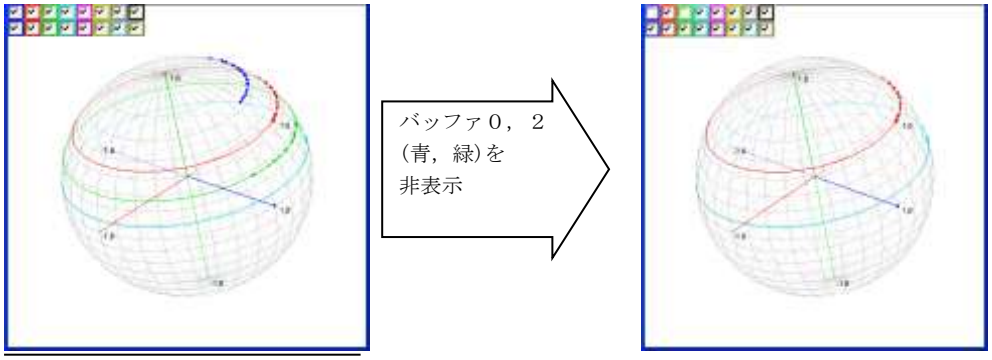
6.2.6. フィルタ機能

カーソルを、ウインドの左上隅に置くとチェクボックスが表示されます。

ウインド左上のチェックボックスは、表示フィルタであり、チェックを外すと当該バッファのデータは非表示となります。

チェックボックスは、左から順に上段がバッファ０～７に、下段が８～１５に対応します。(表示色と同じ色で縁取りされています)

尚、チェックを外しても、座標軸は非表示とはなりません。(座標軸の表示／非表示はスタイル設定によります)



6.2.7. 視点の設定

視点の設定は、ポップアップメニューの「ＸＹ座標面」「ＸＺ座標面」「ＹＺ座標面」「３Ｄ座標面」で特定の視点を設定するか、あるいは、グラフ・ウインド上を、マウスの左ボタンでドラッグすることにより、任意の視点を設定することができます。

マウスで横方向にドラッグすると、表示物体がＸ軸回りに回転します。

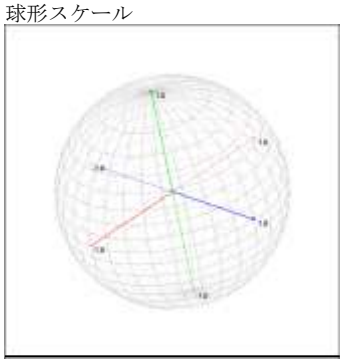
マウスで縦方向にドラッグすると、表示物体がＹ軸回りに回転します。

CTRL キーを押しながら横方向にドラッグすると、表示物体がＺ軸回りに回転します。

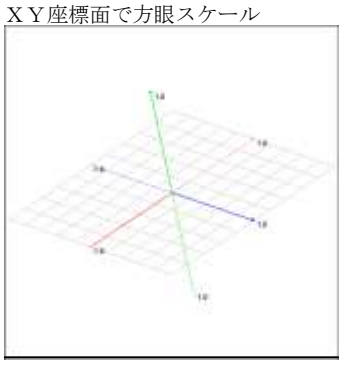
6.2.8. スケールの表示

ポップアップメニューにより、「方眼スケール」「同心(楕)円スケール」「(楕)球形スケール」を選択できます。また、スケール値の表示／非表示も選択可能です。

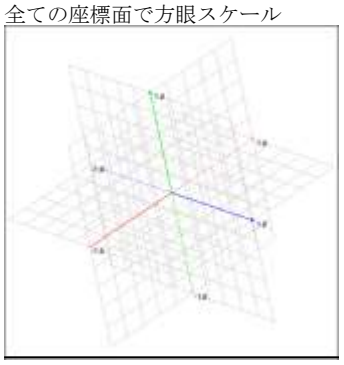
球形スケール



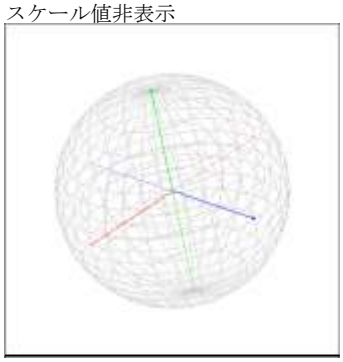
ＸＹ座標面で方眼スケール



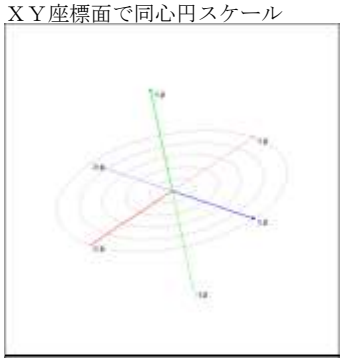
全ての座標面で方眼スケール



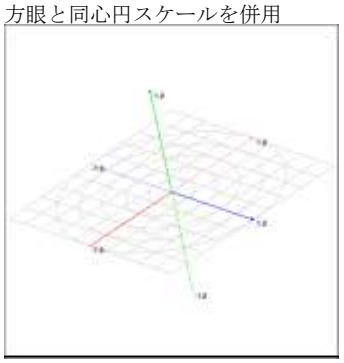
スケール値非表示



ＸＹ座標面で同心円スケール

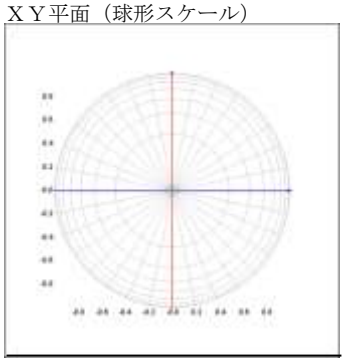


方眼と同心円スケールを併用

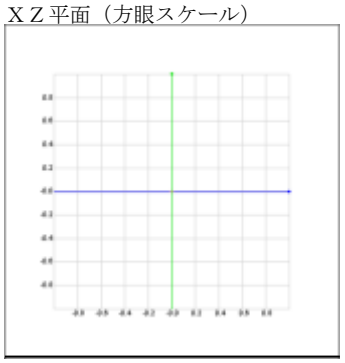


スケール値は、視点がいずれかの座標軸と一致した（つまり、いずれか２軸の平面を見る）場合に限り、中間値も表示されます。その他の視点設定では、各軸の先端値だけを表示します。

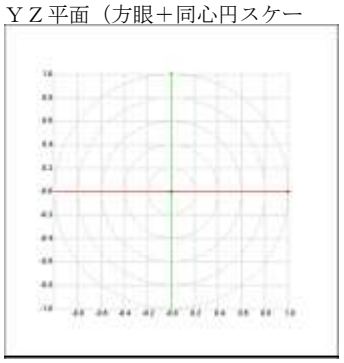
ＸＹ平面（球形スケール）



ＸＺ平面（方眼スケール）



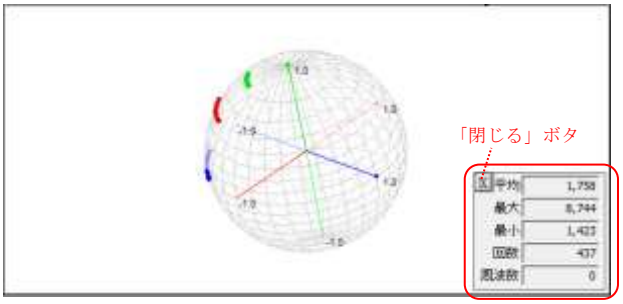
ＹＺ平面（方眼＋同心円スケー



スケールやスケール値の表示は、コントロールのスタイルによっても設定可能です。

6.2.9. 描画時間情報表示

２Ｄ／３Ｄグラフを右クリックし、ポップアップメニューから「描画時間情報表示」を選択すると、２Ｄ／３Ｄプロット・イメージの描画時間を計測し、右下に以下のように表示します。

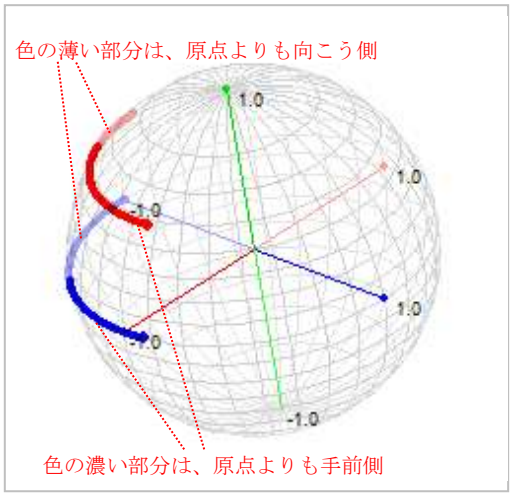


平均：１回の描画に要する時間の平均[μｓ]
最大：最大描画時間[μｓ]
最小：最小描画時間[μｓ]
回数：計測回数
周波数：計測周波数[Hz]　（ＰＣで固定な値）

ウインドのサイズを変更した場合は、計測をやり直します。
短い周期でデータを投与すると、処理が重くなり、２Ｄ／３Ｄプロット・イメージをスムーズに表示できなくなります。
処理時間は、描画時間だけではありませんが、少なくとも、データを周期的に投与する場合は、平均値よりも長い周期で投与する必要があります。

6.2.10. 奥行き表現

原点より向う側のイメージを少し薄く描画することで、原点より手前側か、あるいは、向こう側にあるイメージかを表現します。
ちょうど、原点に垂直な「すりガラス」を立てたようなイメージとなります。



デフォルトの描画色は、以下のとおりです。

データ項目	手前の色	向う側の色
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		

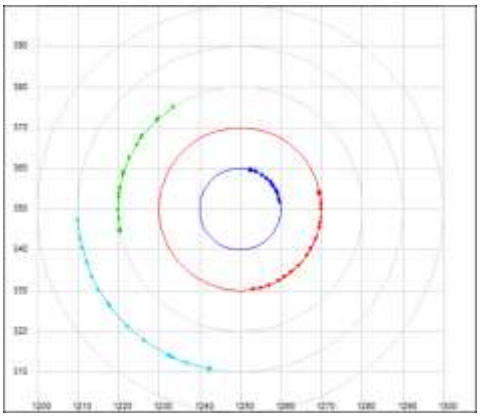
この描画色は、プロパティの Item[n].rgbP/rgbN で変更できます。

ポップアップメニューで「奥行表現禁止」を選択した場合は、原点より手前側でも、向こう側でも、同じ描画（濃い色）となります。

6.2.11.　２Dグラフモード

２Dグラフモードとは、常にX－Y平面を表示するモードであり、プロットデータや描画データは、２次元座標値（ｘとｙ）で指定します。

２Dグラフモードの表示例（横軸はX座標を、縦軸はY座標を示します）



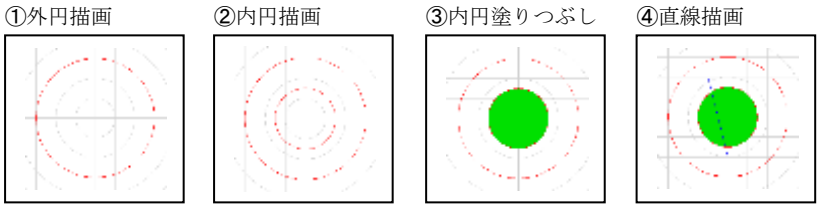
２Dグラフモードを使用するには、最初に AjcG2dInit() を実行します。

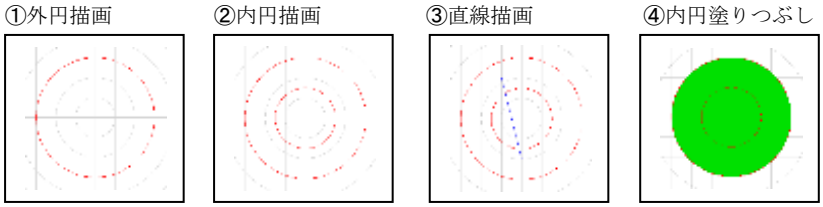
２Dグラフモードでの塗りつぶし

２Dグラフモードでは、塗りつぶしファンクションが使用できます。

AjcG2dFillB() - 閉領域の塗りつぶし
AjcG2dFillS() - 白色部分の塗りつぶし

これらの塗りつぶしファンクションは、描画順に左右される場合がありますので注意が必要です。
以下の例は、外円(赤)、内円(赤)、直線(青)を描画し、内円の中を塗りつぶす例ですが・・





上記例は、内円の中心を指定し、赤で囲まれた閉領域を塗りつぶした例ですが、上段の例は正常に内円が塗りつぶされますが、下段の例では、外円が塗りつぶされてしまいます。
これは、青い直線を上書きすることにより内円の閉領域が壊されたために発生します。

6.2.12. 表示の高速化

データ投与毎に表示＆スクロールすると表示処理に時間がかかります。
そこで、AjcG3dPause()により一定期間の表示を抑止することで表示処理時間を短縮することができます。

```
        ：
AjcG3dPause(TRUE);                  // 表示停止
AjcG3dPutPlotData（・・・);          // データ投与
AjcG3dPixel       (・・・);
AjcG3dLine        (・・・);          ① （この間描画したデータは表示されない）
        ：
AjcG3dPause(FALSE);                 // 表示再開 （①の期間に描画したデータを一気に表示）
        ：
```

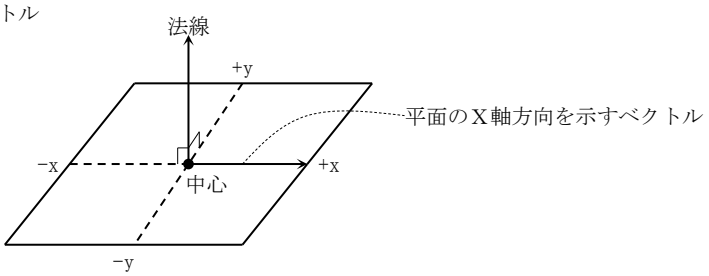
AjcG3dPause(FALSE) を実行すると、それまで表示を停止していたデータを一気に表示します。（全てのデータを表示＆スクロールするわけではなく、最終描画状態だけを表示します）
AjcG3dPause(TRUE)～AjcG3dPause(FALSE)の間描画していたデータはバッファに蓄えられていますので、通常どおりスクロールして見ることができます。

※AjcG3dPause(TRUE)～AjcG3dPause(FALSE)で表示を抑止している期間でも、ユーザ操作（ウインドのサイズを変えたり、最小化したタスクバーから戻す、等）によりウインドの再描画が必要な場合は、その瞬間の画面状態が表示されます。

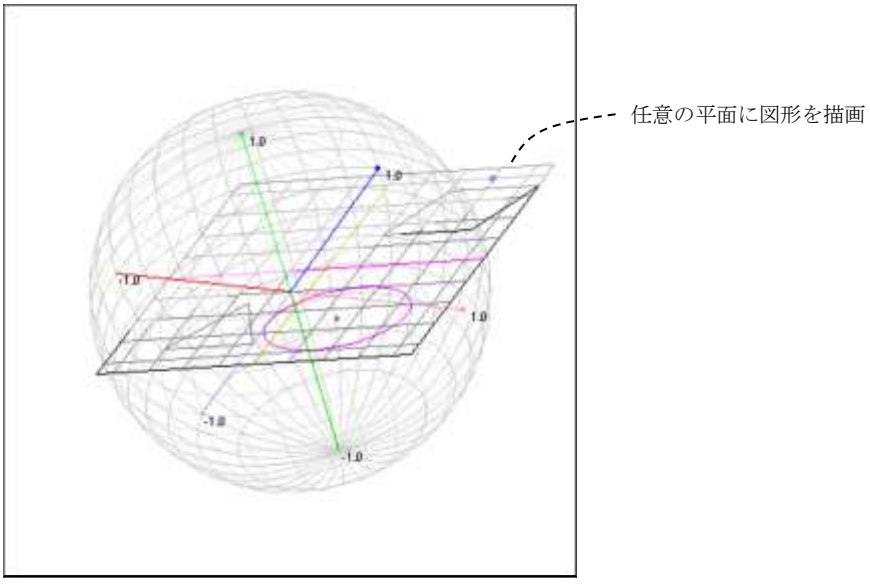
6.2.13. 任意の平面定義と平面上への図形描画

３Ｄ空間上に、任意の平面を定義し、この平面上に図形を描画することができます。
平面は、次の情報で定義されます。

・平面の中心位置と法線（中心位置は平面の原点(0,0)となります）
・平面のＸ軸方向を示すベクトル



定義した平面上には、点，ライン，三角形，四角形，円，楕円を描画することができます。



6.2.14. ファイルやディレクトリのドラッグ＆ドロップ

本コントロールにファイルやディレクトリをドラッグ＆ドロップした場合は、WM_COMMAND メッセージにて、親ウインドへ以下の通知を行います。

・AJC3DGN_DROPFILE ----- ファイルがドロップされたことを通知
・AJC3DGN_DROPDIR ----- フォルダがドロップされたことを通知

これらの通知では、ドロップされたファイルやディレクトリの個数を通知します。
ファイルやディレクトリのパス名は、本コントロールのサポートＡＰＩ AjcG3dGetDroppedFile(),AjcG3dGetDroppedDir[Ex]() にて取得します。

尚、２Ｄ／３Ｄグラフィック・コントロールでファイルやディレクトリのドラッグ＆ドロップを有効とするには、拡張ウインドスタイルに「WS_EX_ACCEPTFILES」を指定する必要があります。

6.3. コントロールのスタイル

３Ｄ／２Ｄグラフィック・コントロールのスタイルは、以下のとおりです。

名称	ビット	値	内容	備考
AJC3DGS_RECTSCALE	15	0x8000	方眼スケール表示	ビット１０～８で指定された平面に表示
AJC3DGS_SCALEVALUE_X	14	0x4000	Ｘ軸スケール値表示	
AJC3DGS_SCALEVALUE_Y	13	0x2000	Ｙ軸スケール値表示	
AJC3DGS_SCALEVALUE_Z	12	0x1000	Ｚ軸スケール値表示	
AJC3DGS_ELPSSCALE	11	0x0800	同心円スケール表示	ビット１０～８で指定された平面に表示
AJC3DGS_SCALE_XY	10	0x0400	ＸＹ平面スケール表示	
AJC3DGS_SCALE_XZ	9	0x0200	ＸＺ平面スケール表示	
AJC3DGS_SCALE_YZ	8	0x0100	ＹＺ平面スケール表示	
AJC3DGS_NODEPTHCTRL	7	0x0080	奥行き表現禁止	３Ｄグラフでパフォーマンスが問題となる場合に禁止してください
AJC3DGS_NOBORDER	6	0x0040	外枠非表示	
AJC3DGS_NOFILTER	5	0x0020	フィルタ非表示	
AJC3DGS_NOANGLE	4	0x0010	視点設定／ドラッグ禁止	
AJC3DGS_SPHERESCALE	3	0x0008	球体スケール表示	
AJC3DGS_SHOWAXIS_X	2	0x0004	Ｘ軸表示	
AJC3DGS_SHOWAXIS_Y	1	0x0002	Ｙ軸表示	
AJC3DGS_SHOWAXIS_Z	0	0x0001	Ｚ軸表示	

合成値

名称	内容
AJC3DGS_SCALEVALUE_XY	(AJC3DGS_SCALEVALUE_X | AJC3DGS_SCALEVALUE_Y)
AJC3DGS_AXIS_XY	(AJC3DGS_SHOWAXIS_X | AJC3DGS_SHOWAXIS_Y)
AJC3DGS_SCALEVALUE	(AJC3DGS_SCALEVALUE_X | AJC3DGS_SCALEVALUE_Y | AJC3DGS_SCALEVALUE_Z)
AJC3DGS_SCALELINE	(AJC3DGS_SCALE_XY | AJC3DGS_SCALE_XZ | AJC3DGS_SCALE_YZ)
AJC3DGS_SCALE	(AJC3DGS_SCALEVALUE | AJC3DGS_SCALELINE)
AJC3DGS_AXIS	(AJC3DGS_SHOWAXIS_X | AJC3DGS_SHOWAXIS_Y | AJC3DGS_SHOWAXIS_Z)
AJC3DGS_3DMODE	(AJC3DGS_SCALEVALUE | AJC3DGS_SPHERESCALE | AJC3DGS_AXIS)
AJC3DGS_2DMODE	AJC3DGS_SCALEVALUE_XY | AJC3DGS_SCALE_XY | AJC3DGS_RECTSCALE)

尚、２Ｄ／３Ｄグラフィック・コントロールでファイルやディレクトリのドラッグ＆ドロップを有効とするには、以下の拡張ウインドスタイルを指定する必要があります。

名称	ビット	値	内容
WS_EX_ACCEPTFILES	4	0x0010	ドラッグ＆ドロップでファイルを受け付けるようにする

6.4. ベクトルデータ構造体

３Ｄ／２Ｄグラフィック・コントロールで使用する、ベクトルデータ構造体の形式は、以下のとおりです。

２Ｄベクトル

```
typedef struct {
    double  x, y;
} AJC2DVEC, *PAJC2DVEC;
typedef const AJC2DVEC *PCAJC2DVEC;
```

３Ｄベクトル

```
typedef struct {
    double  x, y, z;
} AJC3DVEC, *PAJC3DVEC;
typedef const AJC3DVEC *PCAJC3DVEC;
```

３Ｄ始点位置と方向ベクトル

```
typedef struct {
    AJC3DVEC    p, v;
} AJC3DLVEC, *PAJC3DLVEC;
typedef const AJC3DLVEC *PCAJC3DLVEC;
```

p: 始点位置
v: 方向ベクトル

6.5. コントロールのプロパティ構造体

３Ｄ／２Ｄグラフィック・コントロールのプロパティは、以下の構造体で定義されます。

```
//----- アイテム情報 ----------------------------------------------------------------------//
#define AJC3DG_MAXITEM      16

typedef struct {
    UI          MaxPlot;                //  最大プロット数
    COLORREF    rgbP, rgbN;             //  表示色
} AJCG3DITEM, *PAJCG3DITEM;

//----- 軸描画情報 ------------------------------------------------------------------------//
#define AJC3DG_MAXAXIS      3
typedef struct {
    COLORREF    rgbP, rgbN;             //  表示色
} AJC3DGAXIS, *PAJC3DGAXIS;

//------------------------------------------------------------------------------------------//
typedef struct {
    AJC3DVEC    Rot;                    //  回転角度[度]
    AJC3DVEC    Cent;                   //  中心位置
    double      xr, yr, zr;             //  各軸の半径
    double      ratio;                  //  視野に対する半径の割合
    UI          PlotSize;               //  プロット点の大きさ
    UI          PlotSizeE;              //  プロット最終点の大きさ
    BOOL        fPlotLine;              //  プロット線描画フラグ
    BOOL        fAspect1;               //  アスペクト比を１に保つ
    AJC3DGAXIS  axis[AJC3DG_MAXAXIS];   //  軸描画情報
    AJCG3DITEM  Item[AJC3DG_MAXITEM];   //  各アイテムの情報
} AJC3DGPROP, *PAJC3DGPROP;
typedef const AJC3DGPROP *PCAJC3DGPROP;

typedef AJC3DGPROP   AJC2DGPROP;
typedef PAJC3DGPROP  PAJC2DGPROP;
typedef PCAJC3DGPROP PCAJC2DGPROP;
```

各メンバ変数の内容は、以下のとおりです。

メンバ	内容	規定値	備考
Rot	視点設定（各軸回りの回転角度）	全て 0.0	［度］
Cent	各座標の中心位置	全て 0.0	２Ｄモード時は「Cent.z」は未使用
xr, yr, zr	各座標軸の長さ（半径）	全て 1.0	２Ｄモード時は「zr」は未使用
ratio	画面に占めるビューボリュームの割合	0.7	２Ｄモード時は 1.0 に設定
PlotSize	プロットデータのピクセルサイズ	2	
PlotSizeE	プロットデータ（終点）のピクセルサイズ	3	
fPlotLine	プロットデータの結線	1	0:結線しない, 1:結線する
fAspect1	レンジで設定された領域のアスペクト比を１に保つ	1	冒頭の「アスペクト比の設定」を参照
axis	各軸の描画情報	-	
Item	各データの描画情報	-	

各軸の描画情報

メンバ	内容	規定値	備考
rgbP	中心より手前側の表示色	-	
rgbN	中心より向こう側の表示色	-	２Ｄモード時は未使用

各データの描画情報

メンバ	内容	規定値	備考
MaxPlot	プロットデータ数	16	
rgbP	中心より手前側の表示色	-	
rgbN	中心より向こう側の表示色	-	２Ｄモード時は未使用

6.6. キャプション文字列によるプロパティの設定

CreateWindow()/CreateWindowEx()の lpszWindowName 引数（ウインドキャプション）あるいは、ダイアログデザイン時におけるコントロールの Caption プロパティにより、３Ｄ／２Ｄグラフィック・コントロールのプロパティを設定することができます。

パラメタ（キャプション文字列）の形式は、以下のとおりです。([XXX]は、XXX を省略可能であることを意味します)

```
P: [TX=fff], [TY=fff], [TZ=fff], [XC=fff], [YC=fff], [ZC=fff], [XR=fff], [YR=fff], [ZR=fff], [RA=fff], [PS=nnn],
   [PE=nnn], [PL=nnn], [A1=n], [BC=nnn], [X=nnn/nnn], [Y=nnn/nnn], [Z=nnn/nnn],
   [0=nnn/nnn/nnn], ・・・・・, [15=nnn/nnn/nnn]
```

文字列の先頭は「P:」でなければなりません。(「P:」の直後には空白を置けます)
「fff」は実数で、「nnn」は整数（１６進数の場合は先頭に'0x'を付加）で指定します。
各パラメタはカンマ (,) で区切ります。(カンマの前後には空白を置けます)
各パラメタの指定順序は任意です。

各パラメタの設定内容は、以下のとおりです。

キーワード	内　容
TX, TY, TZ	視点設定（各軸回りの回転角度［度］）
XC, YC, ZC	各軸の中心位置の座標値
XR, YR, ZR	各軸の長さ（半径）
RA	画面に占めるビューボリュームの割合（0.1 ～ 1.0）
PS	プロットデータのピクセルサイズ
PE	プロットデータ（終点）のピクセルサイズ
PL	プロットデータの結線（0:結線しない，1:結線する）
A1	レンジで設定された領域のアスペクト設定（0:ウインドサイズに合わせる，1:アスペクト＝１とする）
BC	コントロール外枠の表示色
X～Z	各軸の表示色（ 〈手前側の色〉/〈向こう側の色〉）
0～15	各データのプロット数と表示色（〈プロット数〉/〈手前側の色〉/〈向こう側の色〉）

表示色は、１６進数で「0xbbggrr」の形式で指定します（bb:青成分，gg:緑成分，rr:赤成分）

設定例

```
P: XC=1.0, YC=2.0, ZC=3.0, XR=5.0, YR=5.0, ZR=5.0
```
各軸の中心位置を(1.0, 2.0, 3.0)とし、半径を全て5.0とする

```
P: PS=3, PE=5
```
プロットデータのピクセルサイズを、終点以外＝３、終点＝５とする

```
P: 0=32, 1=32, 2=32, 3=32, 4=32, 5=32, 6=32, 7=32
```
データ項目０～７のプロット数を３２とする

6.7. テキストの取得と設定

テキストを取得した場合は、プロパティ設定内容を表す文字列を返します。
デフォルトでの、取得テキストは、以下のとおりです。

```
P: TX=0, TY=0, TZ=0, XC=0, YC=0, ZC=0, XR=1, YR=1, ZR=1, RA=0.7, PS=2, PE=3, PL=1, A1=1, BC=0x0,
    X=0xFF0000/0xFFA0A0, Y=0xFF/0xA0A0FF, Z=0xE000/0x40FF40,
    0=16/0xFF0000/0xFFA0A0, 1=16/0xFF/0xA0A0FF, 2=16/0xE000/0x40FF40, 3=16/0xE0E000/0xFFFFA0,
    4=16/0xFF00FF/0xFFA0FF, 5=16/0xC0C0/0x80CECE, 6=16/0x808080/0xC0C0C0, 7=16/0x0/0xA0A0A0,
    8=16/0xBE0000/0xBE5A5A , 9=16/0xBE/0x5A5ABE , 10=16/0xBE00/0x5ABE5A, 11=16/0xA0A000/0xA0A05A,
    12=16/0xA000A0/0xDCBEA0, 13=16/0xA0A0/0xA05ABE, 14=16/0xB4B440/0xB4B480, 15=16/0x408080/0x80B4B4
```

テキストを設定する場合は、キャプション文字列によるプロパティの設定と同様に扱います。

6.8. プロパティの永続化

設定したプロパティを、プロファイル（.ini ファイル／レジストリ）に記録し、次回起動時に記録されているプロファイルを読み出すことにより、プロパティを永続的に有効とすることができます。

３Ｄグラフモードの場合

プロパティをプロファイルから読み出すには、ダイアログやウインドの初期化時に、AjcG3dLoadProp()を実行します。
尚、初回実行時はプロファイルにプロパティが記録されていない為、AjcG3dLoadProp()でプロパティのデフォルト値を指定します。
AjcG3dLoadProp()でプロパティのデフォルト値を指定しない場合は、現在設定されているプロパティがデフォルト値となります。

デフォルトプロパティを指定する場合

```
AJC3DGPROP DefProp;

DefProp.Cent.x = 10.0;
DefProp.Cent.y = 15.0;
DefProp.Cent.z = 20.0;
   ・・・
AjcG3dLoadProp(hwnd, "SectName", &DefProp);
```

デフォルトプロパティを指定しない場合

```
AjcG3dSetCenter(hwnd, 10.0, 15.0, 20.0);
AjcG3dLoadProp(hwnd, "SectName", NULL);
```

プロファイルにプロパティが記録されている場合は、AjcG3dLoadProp()により読み出されたプロパティが設定される為、AjcG3dSetCenter()により設定された値は無効となります。(あらかじめ設定されているプロパティ値は初回ロード時のデフォルト値となります)

現在設定されているプロパティをプロファイルに記録するには、ダイアログやウインドの終了時に、AjcG3dSaveProp()を実行します。

プロパティをプロファイルに記録

```
AjcG3dSaveProp(hwnd, "SectName");
```

デフォルトでは、プロファイルの記録先は、レジストリになります。
プロファイルの記録先を初期化ファイル（自プログラムパス名の拡張子を「.ini」としたファイル）とする場合は、プログラムの最初（AjcG3dLoadProp(), AjcG3dSaveProp()を実行する前）に「AjcSetProfileIsRegistory(FALSE);」を実行してください。

プロファイルへのアクセスは、AjcGetProfile･･･() と AjcPutProfile･･･() により行います。
これらの関数仕様やプロファイルアクセスに関するその他の情報は、「プロファイル・アクセス」章を参照してください。

尚、AjcG3dLoadPropEx()／AjcG3dSavePropEx()を使用すれば、プロパティ値に加えて、フィルタ設定とウインドスタイルも永続化できます。

２Ｄグラフモードの場合

２Ｄグラフモードの場合も、３Ｄグラフモードと同様に行いますが、関数の名称だけが異なります。
プロパティをプロファイルから読み出すには、ダイアログやウインドの初期化時に、AjcG2dLoadProp()を実行します。
現在設定されているプロパティをプロファイルに記録するには、ダイアログやウインドの終了時に、AjcG2dSaveProp()を実行します。

尚、AjcG2dLoadPropEx()／AjcG2dSavePropEx()を使用すれば、プロパティ値に加えて、フィルタ設定とウインドスタイルも永続化できます。

6.9. サポートＡＰＩ

３Ｄ／２Ｄグラフィック・コントロールのサポートＡＰＩ一覧を以下に示します。

#	関数名	内容	備考
1	AjcG3dInit	３Ｄグラフモード初期化	
2	AjcG3d{Set/Get}Prop	プロパティ設定／取得	３Ｄグラフモード専用
3	AjcG3dAdjustRange	レンジ自動調整	
4	AjcG3dSetRange	各軸のレンジ設定	
5	AjcG3dSetCenter	各軸の中心位置設定	
6	AjcG3dSetWidth	各軸の幅（半径）設定	
7	AjcG3dSetSameRangeWidth	各軸のレンジ幅を同じにする	
8	AjcG3dSetFixedAspect	レンジ設定された領域のアスペクトを設定する	
9	AjcG3dSetColor	表示色設定	
10	AjcG3dSetPlotNumber	プロットデータ数設定	
11	AjcG3dSetPlotSize	プロットデータのピクセルサイズ設定	
12	AjcG3dPutPlotData[V]	プロットデータ投与	
13	AjcG3dGetPlotDataCount AjcF3dGetPlotMaxCount	プロットデータ数取得	
14	AjcG3dPixel[V]	ピクセル描画	
15	AjcV3dMoveTo[V]	ラインの始点設定	
16	AjcG3dLineTo[V] AjcG3dArrowTo[V]	ラインの終点を設定しライン／矢印描画	
17	AjcG3dLine[V] AjcG3dArrow[V]	ライン／矢印描画	
18	AjcG3dTriangle[V]	三角形描画	
19	AjcG3dSquare[V]	四角形描画	
20	AjcG3dCube[V]	立方体／長方体描画	
21	AjcG3dSphere[V]	球／楕球描画	
22	AjcG3dDefPlane[V]	３Ｄ空間上に任意の平面を定義	
23	AjcG3dSetAngle	視点設定（３軸回転指定）	
24	AjcG3dSetAngle	視点設定（視点ベクトル指定）	
25	AjcG3dSetAngleXY	視点をＸ－Ｙ平面に設定	
26	AjcG3dSetAngleXZ	視点をＸ－Ｚ平面に設定	
27	AjcG3dSetAngleYZ	視点をＹ－Ｚ平面に設定	
28	AjcG3dSetAngle3D	視点を３Ｄイメージに設定	
29	AjcG3dSetPlane	平面アングル設定（視点を任意の平面に設定）	
30	AjcG3dLoadProp[Ex] AjcG3dSaveProp[Ex]	プロファイルからプロパティ値読出し／書き込み	
31	AjcG3dGetBitmap	ビットマップ取得	
32	AjcG3dSetTextFont	テキスト描画フォント設定	
33	AjcG3dTextOut[V]	テキスト描画（ピクセル位置指定）	
34	AjcG3dPrintF[V]	書式テキスト描画	
35	AjcG3dGetText	描画テキスト取得	
36	AjcG3dClear[All]Shape	図形描画データ消去	
37	AjcG3dClear[All]Plot	プロットデータ消去	
38	AjcG3dClear[All]Text	描画テキスト消去	
39	AjcG3dClear[All]Data	描画データ（図形，プロット）消去	
40	AjcG3dClear	全てのデータ（図形，プロット，テキスト）消去	

つづく

つづき

#	関数名	内容	備考
41	AjcG2dInit	２Ｄグラフモード初期化	２Ｄグラフモード専用
42	AjcG2dSetPlane	平面アングル設定	
43	AjcG2d{Set/Get}Prop	プロパティ設定／取得	
44	AjcG2dAdjustRange	レンジ自動調整	
45	AjcG2dSetRange	各軸のレンジ設定	
46	AjcG2dSetCenter	各軸の中心位置設定	
47	AjcG2dSetWidth	各軸の幅（半径）設定	
48	AjcG2dSetSameRangeWidth	各軸のレンジ幅を同じにする	
49	AjcG2dSetFixedAspect	レンジ設定された領域のアスペクトを１にする	
50	AjcG2dSetColor	表示色設定	
51	AjcG2dSetPlotNumber	プロットデータ数設定	
52	AjcG2dSetPlotSize	プロットデータのピクセルサイズ設定	
53	AjcG2dPutPlotData[V]	プロットデータ投与	
54	AjcG2dGetPlotDataCount AjcF2dGetPlotMaxCount	プロットデータ数取得	
55	AjcG2dLoadProp[Ex] AjcG2dSaveProp[Ex]	プロファイルからプロパティ値読出し／書き込み	
56	AjcG2dFillB[V]	ボーダー色で囲まれた部分の塗りつぶし	
57	AjcG2dFillS[V]	連続する白色部分の塗りつぶし	
58	AjcG2dGetPixel[V]	ピクセルの表示色取得	
59	AjcG2dGetBitmap	ビットマップデータ取得	
60	AjcG2dSetTextFont	テキスト描画フォント設定	
61	AjcG2dTextOut[V]	テキスト描画（ピクセル位置指定）	
62	AjcG2dPrintF[V]	書式テキスト描画	
63	AjcG2dGetText	描画テキスト取得	
64	AjcG2dClear[All]Shape	図形描画データクリアー	
65	AjcG2dClear[All]Plot	プロットデータ消去	
66	AjcG2dClear[All]Text	描画テキスト消去	
67	AjcG2dClear[All]Data	描画データ（図形，プロット）削除	
68	AjcG2dClear	全てのデータ（図形，プロット，テキスト）削除	

#	関数名	内容	備考
69	AjcG2dPixel[V]	平面にピクセル描画	２Ｄグラフモードと３Ｄグラフモードで兼用
70	AjcG2dMoveTo[V]	平面に描画するラインの始点設定	
71	AjcG2dLineTo[V] AjcG2dArrowTo[V]	終点を指定し平面にライン／矢印描画	
72	AjcG2dLine[V] AjcG2dArrow[V]	平面にライン／矢印描画	
73	AjcG2dTriangle[V]	〃　三角形描画	
74	AjcG2dSquare[V]	〃　四角形描画	
75	AjcG2dRectangle[V]	〃　長方形描画	
76	AjcG2dEllipse[V]	〃　円／楕円描画	
77	AjcG2dStar[V][Ex]	〃　星形描画	
78	AjcG3dSetNtcRClk	右クリック通知設定	
79	AjcG3dEnablePopupMenu	ポップアップメニューの許可／禁止	
80	AjcG3dSetTipText AjcG3dGetTipText	ツールチップの設定／取得	
81	AjcG3d{Set/Get}TipShowAlways AjcG3dSetTipShowAlwaysAll	ツールチップ表示条件の設定／取得	
82	AjcG3dSetChkBoxTipText AjcG3dGetChkBoxTipText	フィルタ・チェックボックス・ツールチップの設定／取得	
83	AjcG3dSetChkBoxTipShowAlways AjcG3dGetChkBoxTipShowAlways	フィルタ・チェックボックス・ツールチップ表示条件の設定／取得	
84	AjcG3dSetFilter AjcG3dGetFilter	フィルタの設定／取得	
85	AjcG3dGetDroppedFile	ドロップされたファイル名取得	
86	AjcG3dGetDroppedDir[Ex]	ドロップされたディレクトリ名取得	
87	AjcG3dSetTitleText	タイトル文字列の設定	
88	AjcG3dPause	画面表示の停止／再開	
89	AjcG3dEnableMesDraw	描画時間計測情報の許可／禁止	
90	AjcG3dSetEvtCallBack	イベント通知用コールバック関数の設定	

6.9.1. ３Ｄグラフモード初期化(AjcG3dInit[V]) ・・・ ３Ｄモード専用

形 式 : BOOL AjcG3dInit (HWND hwnd, double x1, double y1, double z1, double x2, double y2, double z2, UI style);
BOOL AjcG3dInitV(HWND hwnd, PCAJC3DVEC pLo , PAJC3DVEC pHi , PAJC3DVEC pRot, UI style)

引 数 : hwnd - コントロールのウインドハンドル
x1, y1, z1 ／ pLo - 各軸の低位値（pLo=NULL / pHi=NULL の場合は設定しない）
x2, y2, z2 ／ pHi - 各軸の高位値（pLo=NULL / pHi=NULL の場合は設定しない）
style - コントロールのスタイル(AJC3DGS_*XXXXX*, -1 の場合は設定しない)
pRot - 視点（各軸の回転角度， NULL：設定しない）

説 明 : ３Ｄグラフモードの初期設定を行います。
各座標軸の範囲が x1～x2, y1～y2, z1～z2 ／pLo～pHi に設定されます。
pLo/pHi が NULL の場合は、各座標軸の範囲を設定しません。(キャプション文字列によるパラメタ (XC. YC, ZC, XR, YR, ZR) が有効となります)

style は、任意のコントロール・スタイル(AJC3DGS_*XXXXX*)を指定可能ですが、標準的には「AJC3DGS_3DMODE」を指定します。(AJC3DGS_3DMODE では、球体スケール，スケール値，全座標軸を表示します)
style=-1 とした場合は、スタイルの設定を行いません。

pRot は、視点（各軸の回転角度）を指定します。
pRot が NULL の場合は、視点の設定を行いません。(キャプション文字列によるパラメタ (TX, TY, TZ) が有効となります)
AjcG3dInit()の場合、視点が３Ｄグラフイメージ（x=60°, x=10°, x=45°）に設定されます。

戻り値 : TRUE - 成功
FALSE - 失敗

6.9.2. プロパティ設定／取得 (AjcG3d{Set/Get}Prop) ・・・ ３Ｄモード専用

形 式 : BOOL AjcG3dSetProp(HWND hwnd, PCAJC3DGPROP pProp); - プロパティ設定
BOOL AjcG3dGetProp(HWND hwnd, PAJC3DGPROP pBuf); -- プロパティ取得

引 数 : hwnd - コントロールのウインドハンドル
pProp - 設定するプロパティデータのアドレス
pBuf - 取得したプロパティデータを格納するバッファのアドレス

説 明 : プロパティの設定／取得を行います。
プロパティデータの内容については、「３Ｄ／２Ｄグラフィック・コントロールのプロパティ構造体」を参照してください。

戻り値 : TRUE - 成功
FALSE - 失敗

6.9.3. レンジ自動調整(AjcG3dAdjustRange) ・・・ ３Ｄモード専用

形 式 : BOOL AjcG3dAdjustRange (HWND hwnd);

引 数 : hwnd - コントロールのウインドハンドル

説 明 : （フィルタで非表示となっている項目を除く）全てのデータから、最大値と最小値を算出し、±５％のマージンを持って各座標軸の長さを調整します。
つまり、全てのデータがウインドに表示されるように、グラフのレンジを調整します。
但し、各座標軸の中心位置は変化しません。

戻り値 : TRUE - 成功
FALSE - 失敗

6.9.4. レンジ設定 (AjcG3dSetRange) ・・・ ３Ｄモード専用

形　式 ： BOOL　AjcG3dSetRange (HWND hwnd, double x1, double y1, double z1, double x2, double y2, double z2) ;

引　数 ： hwnd　　- コントロールのウインドハンドル
x1, y1, z1 - 各軸の低位値
x2, y2, z2 - 各軸の高位値

説　明 ： 各座標軸のレンジを設定します。
各座標軸の範囲値は、x1 ＜ x2, y1 ＜ y2, z1 ＜ z2 でなければなりません。

戻り値 ： TRUE　- 成功
FALSE - 失敗

6.9.5. 各座標軸の中心位置（原点）設定(AjcG3dSetCenter) ・・・ ３Ｄモード専用

形　式 ： BOOL　AjcG3dSetCenter (HWND hwnd, double xc, double yc, double zc) ;

引　数 ： hwnd　　- コントロールのウインドハンドル
xc, yc, zc - 各軸の中心位置（原点）

説　明 ： 各座標軸の中心位置を設定します。
各座標軸の長さは、変化しません。

戻り値 ： TRUE　- 成功
FALSE - 失敗

6.9.6. 各座標軸の長さ設定(AjcG3dSetWidth) ・・・ ３Ｄモード専用

形　式 ： BOOL　AjcG3dSetWidth (HWND hwnd, double xr, double yr, double zr) ;

引　数 ： hwnd　　- コントロールのウインドハンドル
xr, yr, zr - 各軸の長さ（半径）

説　明 ： 各座標軸長さ（半径）を設定します。
各座標軸の中心位置は変化しません。

戻り値 ： TRUE　- 成功
FALSE - 失敗

6.9.7. 各軸のレンジ幅を同じにする (AjcG3dSetSameRangeWidth) ・・・ ３Ｄモード専用

形　式 ： BOOL　AjcG3dSetSameRangeWidth (HWND hwnd) ;

引　数 ： hwnd　　- コントロールのウインドハンドル

説　明 ： 中心位置を変更せずに、各軸のレンジ幅を同一に設定します。(最大のレンジ幅に合わせます)

(例)　変更前のレンジ

	レンジ	中心	幅
X	100～500	300	400
Y	300～400	350	100
Z	700～900	800	200



変更後のレンジ

	レンジ	中心	幅
X	100～500	300	400
Y	**150～550**	350	**400**
Z	**400～1200**	800	**400**

戻り値 ： TRUE　- 成功
FALSE - 失敗

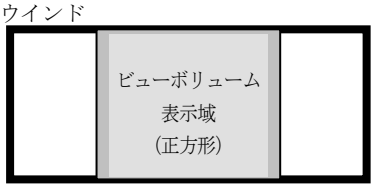
6.9.8. レンジ設定された領域のアスペクトを設定する (AjcG3dSetFixedAspect) ・・・ ３Ｄモード専用

形　式 ： BOOL　AjcG3dSetFixedAspect (HWND hwnd, BOOL fFixed) ;

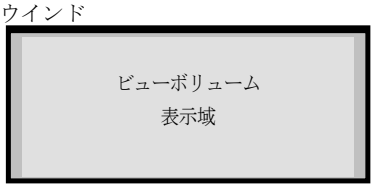
引　数 ： hwnd　　- コントロールのウインドハンドル
fFixed　- TRUE　: レンジ設定された領域のアスペクトを１とする
FALSE : アスペクトをウィンドサイズに合わせて可変にする

説　明 ： fFixed=TRUE を指定した場合、レンジ設定されたビューボリュームを、ウインドの正方形領域にマッピングして描画します。
fFixed=FALSE を指定した場合、レンジ設定されたビューボリュームを、ウインド全体ににマッピングして描画します。

<u>fFixed=TRUE</u>
ウインド



<u>fFixed=FALSE</u>
ウインド



fFixed=TRUE とし、各軸のレンジ（長さ）を同一にすることにより、図形が歪みのない形で表示されます。
fFixed=FALSE とした場合は、ウインドサイズにより、図形がつぶれたり、細長く表示されます。

戻り値 ： TRUE　- 成功
FALSE - 失敗

6.9.9. 表示色設定(AjcG3dSetColor) ・・・ ３Ｄモード専用

形　式 ： BOOL　AjcG3dSetColor (HWND hwnd, UI id, COLORREF rgbP, COLORREF rgbN) ;

引　数 ： hwnd　　- コントロールのウインドハンドル
id　　- データ項目番号（０～１５）
rgbP　　- 視点から見て原点より手前側の表示色
rgbN　　- 視点から見て原点より向こう側の表示色

説　明 ： 「id」で指定されたデータ項目の表示色を設定します。
通常、rgbP は鮮やかな色を指定し、rgbN は薄い色を指定します。
（ex.rgbP=RGB(255, 0, 0)‥赤,　rgbN=(255, 160, 160)‥薄い赤）

尚、「AJC3DGS_ NODEPTHCTRL」スタイルが設定されている場合は、rgbN は無視され、全て rgbP で指定された表示色となります。

戻り値 ： TRUE　- 成功
FALSE - 失敗

6.9.10. プロットデータ数設定(AjcG3dSetPlotNumber) ・・・ ３Ｄモード専用

形　式 ： BOOL　AjcG3dSetPlotNumber (HWND hwnd, UI id, UI PlotNumber) ;

引　数 ： hwnd　　- コントロールのウインドハンドル
id　　- データ項目番号（０～１５）
PlotNumber - プロットデータ数

説　明 ： 「id」で指定されたデータ項目におけるプロットデータの保留個数を設定します。
AjcPutPlotData[V]でプロットデータを投与した際、設定されているプロットデータ数を超える場合は、古いプロットデータが破棄されます。
つまり、プロットデータは、本関数で設定された個数の最新データだけが表示されることになります。

本関数を実行すると、現在保留されているプロットデータは、一旦破棄されます。

戻り値 ： TRUE　- 成功
FALSE - 失敗

6.9.11. プロットデータのピクセルサイズ設定(AjcG3dSetPlotSize) ・・・ ３Ｄモード専用

形　式 ： BOOL　AjcG3dSetPlotSize (HWND hwnd, UI id, UI PixelSize, UI PixelSizeE)；

引　数 ： hwnd　　- コントロールのウインドハンドル
id　　- データ項目番号（０～１５）
PixelSize　- プロットデータの表示ピクセルサイズ
PixelSizeE - 終点プロットデータの表示ピクセルサイズ

説　明 ： 「id」で指定されたデータ項目におけるプロットデータの表示ピクセルサイズを設定します。
「PixelSize」は、終点（最新のプロットデータ）以外の表示ピクセルサイズを指定します。
「PixelSizeE」は、終点（最新のプロットデータ）の表示ピクセルサイズを指定します。
「PixelSize」「PixelSize」とも、０か１を指定した場合は、１ドットでプロットデータを表示します。
２以上を設定した場合は、当該指定値の半径で、塗りつぶし円を表示します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

6.9.12. プロットデータ投与(AjcG3dPutPlotData) ・・・ ３Ｄモード専用

形　式 ： BOOL　AjcG3dPutPlotData (HWND hwnd, UI id, double x, double y, double z)；

引　数 ： hwnd　　- コントロールのウインドハンドル
id　　- データ項目番号（０～１５）
x, y, z　- プロットデータの座標値

説　明 ： プロットデータをバッファに格納します。
プロットデータが満杯である場合は、最古のプロットデータが破棄されます。

戻り値 ： TRUE　- 成功
FALSE - 失敗

6.9.13. プロットデータ数取得(AjcG3dGetPlotDataCount) ・・・ ３Ｄモード専用

形　式 ： UI　　AjcG3dGetPlotDataCount (HWND hwnd, UI id)；
UI　　AjcG3dGetPlotMaxCount　(HWND hwnd)；

引　数 ： hwnd　　- コントロールのウインドハンドル
id　　- データ項目番号（０～１５）

説　明 ： AjcG3dGetPlotDataCoun()は、当該データ項目のプロットデータの個数を取得します。
AjcG3dGetPlotMaxCoun()は、全データ項目で最大のプロットデータの個数を取得します。

戻り値 ： プロットデータ数（エラー時は０を返す）

6.9.14. ピクセル描画(AjcG3dPixel) ・・・ ３Ｄモード専用

形　式 ： BOOL　AjcG3dPixel (HWND hwnd, UI id, double x , double y , double z , UI PixelSize);
BOOL　AjcG3dPixelV(HWND hwnd, UI id, PCAJC3DVEC pPoint, UI PixelSize);

引　数 ： hwnd　　- コントロールのウインドハンドル
id　　- データ項目番号（０～１５）
x, y, z　- ピクセル描画位置
pPoint　- ピクセル描画位置データのアドレス
PixelSize　- 表示ピクセルサイズ

説　明 ： 指定された座標位置に、ピクセル（点）を描画します。
「PixelSize」に０か１を指定した場合は、１ドットでプロットデータを表示します。
２以上を設定した場合は、当該指定値の半径で、塗りつぶし円を表示します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

6.9.15. ライン始点設定(AjcG3dMoveTo) ・・・ ３Ｄモード専用

形　式 ： BOOL　AjcG3dMoveTo (HWND hwnd, UI id, double x, double y, double z);
BOOL　AjcG3dMoveToV(HWND hwnd, UI id, PCAJC3DVEC ps);

引　数 ： hwnd　　- コントロールのウインドハンドル
id　　- データ項目番号（０～１５）
x, y, z　- 線分の始点位置
ps　　- 線分の始点位置データのアドレス

説　明 ： 線分描画の始点を設定します。
線分を描画するには、AjcV3dLineTO()/AjcV3dLineToV()を実行します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

6.9.16. ライン終点設定 - ライン／矢印描画(AjcG3dMoveTo) ・・・ ３Ｄモード専用

形　式 ： BOOL　AjcG3dLineTo　(HWND hwnd, UI id, double x, double y, double z);
BOOL　AjcG3dLineToV (HWND hwnd, UI id, PCAJC3DVEC pe);

BOOL　AjcG3dArrowTo (HWND hwnd, UI id, double x, double y, double z);
BOOL　AjcG3dArrowToV(HWND hwnd, UI id, PCAJC3DVEC pe);

引　数 ： hwnd　　- コントロールのウインドハンドル
id　　- データ項目番号（０～１５）
x, y, z　- 線分の終点位置
pe　　- 線分の終点位置データのアドレス

説　明 ： 線分描画の終点を指定し、線分を描画します。
AjcG3dArrowTo(),AjcG3dArrowToV()は、線分の終点に矢印を描画します。(但し、線分が10ピクセルに満たない場合は矢印を描画しません)
線分の始点は、この前に実行したAjcV3dMoveTo()/AjcV3dMobveToV()で指定した始点か、AjcV3dLineTo()/AjcV3dLineToV()で指定した終点となります。
指定した線分の終点は、新たな線分の始点として設定されます。

戻り値 ： TRUE　- 成功
FALSE - 失敗

6.9.17. ライン／矢印描画(AjcG3dLine) ・・・ ３Ｄモード専用

形　式 ： BOOL　AjcG3dLine　(HWND hwnd, UI id, double x1, double y1, double z1, double x2, double y2, double z2)；
BOOL　AjcG3dLineV (HWND hwnd, UI id, PCAJC3DVEC p1, PCAJC3DVEC p2)；

BOOL　AjcG3dArrow (HWND hwnd, UI id, double x1, double y1, double z1, double x2, double y2, double z2)；
BOOL　AjcG3dArrowV(HWND hwnd, UI id, PCAJC3DVEC p1, PCAJC3DVEC p2)；

引　数 ： hwnd　　- コントロールのウインドハンドル
id　　- データ項目番号（０～１５）
x1, y1, z1 - 線分の始点位置
x2, y2, z2 - 線分の終点位置
p1　　- 線分の始点位置データのアドレス
p2　　- 線分の終点位置データのアドレス

説　明 ： 指定された座標位置に、線分を描画します。
AjcG3dArrow(),AjcG3dArrowV()は、線分の終点に矢印を描画します。(但し、線分が10ピクセルに満たない場合は矢印を描画しません)

戻り値 ： TRUE　- 成功
FALSE - 失敗

6.9.18. 三角形描画(AjcG3dTriangle) ・・・ ３Ｄモード専用

形　式 ： BOOL　AjcG3dTriangle (HWND hwnd, UI id, double x1, double y1, double z1, double x2, double y2, double z2,
double x3, double y3, double z3)；
BOOL　AjcG3dTriangleV(HWND hwnd, UI id, PCAJC3DVEC p1, PCAJC3DVEC p2, PCAJC3DVEC p3)；

引　数 ： hwnd　　- コントロールのウインドハンドル
id　　- データ項目番号（０～１５）
x1, y1, z1, x2, y2, z2, x3, y3, z3 - 三角形の頂点位置
p1, p2, p3 ------------------------ 三角形の頂点位置データのアドレス

説　明 ： 指定された３つの頂点を結んだ三角形を描画します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

6.9.19. 四角形描画(AjcG3dSuare) ・・・ ３Ｄモード専用

形　式 ： BOOL　AjcG3dSquare (HWND hwnd, UI id, double x1, double y1, double z1, double x2, double y2, double z2,
double x3, double y3, double z3, x4, y4, z4)；
BOOL　AjcG3dSquareV(HWND hwnd, UI id, PCAJC3DVEC p1, PCAJC3DVEC p2, PCAJC3DVEC p3, PCAJC3DVEC p4)；

引　数 ： hwnd　　- コントロールのウインドハンドル
id　　- データ項目番号（０～１５）
x1, y1, z1, x2, y2, z2, x3, y3, z3, x4, y4, z4 - 四角形の頂点位置
p1, p2, p3, p4 -------------------------------- 四角形の頂点位置データのアドレス

説　明 ： 指定された４つの頂点を結んだ四角形を描画します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

6.9.20. 立方体／長方体描画(AjcG3dCube) ・・・ ３Ｄモード専用

形　式 ： BOOL　AjcG3dCube (HWND hwnd, UI id, double xc, double yc, double zc, double xr, double yr, double zr, UI division)；
BOOL　AjcG3dCubeV(HWND hwnd, UI id, PCAJC3DVEC pCent, double xr, double yr, double zr, UI division)；

引　数 ： hwnd　　- コントロールのウインドハンドル
id　　- データ項目番号（０～１５）
xc, yc, zc - 立方体／長方体の中心位置
pCent　　- 立方体／長方体の中心位置データのアドレス
xr, yr, zr - 立方体／長方体の大きさ（内接する球／楕球の半径）
division　- 分割数（１～３２）

説　明 ： 立方体／長方体を描画します。
立方体を描画するには、xr, yr, zr を全て同じ値にします。

戻り値 ： TRUE　- 成功
FALSE - 失敗

6.9.21. 球／楕球描画(AjcG3dSphere) ・・・ ３Ｄモード専用

形　式 ： BOOL　AjcG3dSphere (HWND hwnd, UI id, double xc, double yc, double zc, double xr, double yr, double zr, UI slice, UI stack)；
BOOL　AjcG3dSphereV(HWND hwnd, UI id, PCAJC3DVEC pCent, double xr, double yr, double zr, UI slice, UI stack)；

引　数 ： hwnd　　- コントロールのウインドハンドル
id　　- データ項目番号（０～１５）
xc, yc, zc - 球／楕球の中心位置
pCent　　- 球／楕球の中心位置データのアドレス
xr, yr, zr - 球／楕球の半径
slice　　- 水平分割数（２～３２）
stack　　- 垂直分割数（２～３２）

説　明 ： 球／楕球を描画します。
球を描画するには、xr, yr, zr を全て同じ値にします。

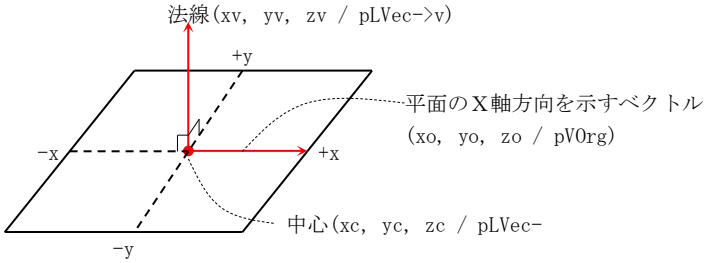
戻り値 ： TRUE　- 成功
FALSE - 失敗

6.9.22.　３Ｄ空間上に任意の平面を定義(AjcG3dDefPlane) ・・・ ３Ｄモード専用

形　式 ： BOOL　AjcG3dDefPlane (HWND hwnd, UI id, double xc, double yc, double zc, double xv, double yv, double zv, double xo, double yo, double zo) ;
BOOL　AjcG3dDefPlaneV(HWND hwnd, UI id, PCAJC3DLVEC pLVec, PCAJC3DVEC pVOrg);

引　数 ： hwnd　　- コントロールのウインドハンドル
id　　- データ項目番号（0～15：当該 id で１個の平面を定義，-1：すべての id(0～15)に同じ平面を定義）
xc，yc，zc - 平面の中心位置（平面の原点(0,0)位置）
xv，yv，zv - 平面の法線ベクトル
xo，yo，zo - 平面上のＸ軸方向を示すベクトル（不要時は全て０）
pLVec　　- 平面の中心位置と法線ベクトルデータのアドレス
pVOrg　　- 平面上のＸ軸方向を示すベクトルデータのアドレス（不要時は NULL）

説　明 ：　３Ｄ空間上に任意の平面を定義します。



xo，yo，zo を全て０（あるいは、pVOrg=NULL）とした場合は、平面のＸ軸方向を示す適当なベクトルを内部で生成します。
（生成するベクトルは任意ですが、平面に原点を中心とした円を描画する場合は、Ｘ軸方向を示すベクトルは関係ないため、この方法でもＯＫです）
定義した平面上の座標系は、xc，yc，zc (or pLVec->p)で指定した位置が原点(0，0)となります。
定義した平面上には、以下の関数により、図形を描画できます。

- AjcG2dPixel
- AjcG2dLine
- AjcG2dMoveTo
- AjcG2dLineTo
- AjcG2dTriangle
- AjcG2dSquare
- AjcG2dRectangle
- AjcG2dEllipse

戻り値 ： TRUE　- 成功
FALSE - 失敗

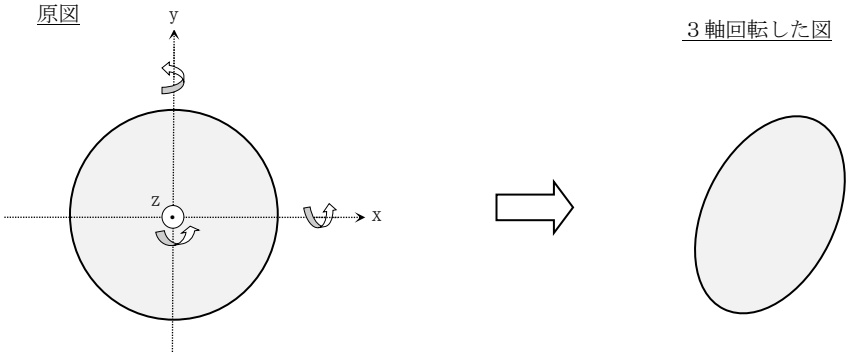
備　考 ：　最大１６個(id=0～15)の平面を定義できますが、別々の平面を定義した場合は、平面を指定する id が表示色となるため、当該平面の描画色は１色になります。
id=0～15 で、全て同じ平面を定義すれば、id は、当該平面への描画色となります。

6.9.23.　視点設定(３軸回転指定，AjcG3dSetAngle) ・・・ ３Ｄモード専用

形　式 ： BOOL　AjcG3dSetAngle (HWND hwnd, double rtx, double rty, double rtz);

引　数 ： hwnd　　- コントロールのウインドハンドル
rtx, rty, rtz - 各軸回りの回転角度［度］

説　明 ： 描画物体を各軸回りに回転することにより、視点を設定します。
原図はＺ軸方向から見た図で、これを基点にＺ，Ｙ，Ｘ軸の順で回転します。



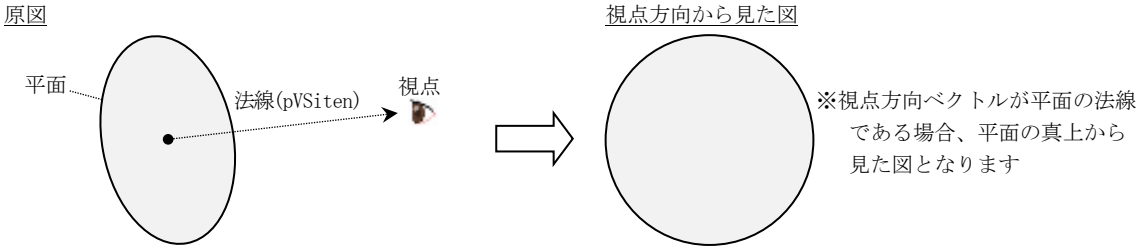
戻り値 ： TRUE　- 成功
FALSE - 失敗

6.9.24.　視点設定(視点ベクトル指定，AjcG3dSetAngleV) ・・・ ３Ｄモード専用

形　式 ： BOOL　AjcG3dSetAngleV (HWND hwnd, PCAJC3DVEC pVSiten);

引　数 ： hwnd　　- コントロールのウインドハンドル
pVSiten　　- 視点方向ベクトルデータのアドレス

説　明 ： 視点方向から見た図となるように、描画物体を各軸回りに回転します。



6.9.25.　視点をＸ－Ｙ平面に設定(AjcG3dSetAngleXY) ・・・ ３Ｄモード専用

形　式 ： BOOL　AjcG3dSetAngleXY (HWND hwnd);

引　数 ： hwnd　　- コントロールのウインドハンドル

説　明 ： Ｘ－Ｙ平面が見えるように視点をＺ軸方向に設定します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

6.9.26. 視点をＸ－Ｚ平面に設定(AjcG3dSetAngleXZ) ・・・ ３Ｄモード専用

形　式 ： BOOL　AjcG3dSetAngleXZ(HWND hwnd);

引　数 ： hwnd　　- コントロールのウインドハンドル

説　明 ： Ｘ－Ｚ平面が見えるように視点をＹ軸方向に設定します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

6.9.27. 視点をＹ－Ｚ平面に設定(AjcG3dSetAngleYZ) ・・・ ３Ｄモード専用

形　式 ： BOOL　AjcG3dSetAngleYZ (HWND hwnd);

引　数 ： hwnd　　- コントロールのウインドハンドル

説　明 ： Ｙ－Ｚ平面が見えるように視点をＸ軸方向に設定します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

6.9.28. 視点を３Ｄイメージに設定(AjcG3dSetAngle3D) ・・・ ３Ｄモード専用

形　式 ： BOOL　AjcG3dSetAngle3D (HWND hwnd);

引　数 ： hwnd　　- コントロールのウインドハンドル

説　明 ： ３Ｄイメージが見えるように、各軸の回転角度を X=60°, Y=10°, Z=45° に設定します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

6.9.29. 平面アングル設定 (AjcG3dSetPlane) ・・・ ３Ｄモード専用

形　式 ： BOOL　AjcG3dSetPlane (HWND hwnd, AJCG3DAXIS_DIR HoriAxis, AJCG3DAXIS_DIR VertAxis);

引　数 ： hwnd　　- コントロールのウインドハンドル
HoriAxis　- 横方向の軸の種別と昇順／降順
VertAxis　- 縦方向の軸の種別と昇順／降順

説　明 ： 平面が見えるように、各軸の回転角度を設定します。
HoriAxis は、横方向にマッピングする軸と、値の昇順／降順を指定します。
VertAxis は、縦方向にマッピングする軸と、値の昇順／降順を指定します。

#	シンボル	内容	備考
1	AJCG3DAXIS_DIR_XP	Ｘ軸を昇順で設定	左端／下端が最小値　→　右端／上端が最大値
2	AJCG3DAXIS_DIR_YP	Ｙ軸を昇順で設定	
3	AJCG3DAXIS_DIR_ZP	Ｚ軸を昇順で設定	
4	AJCG3DAXIS_DIR_XM	Ｘ軸を降順で設定	左端／下端が最大値　→　右端／上端が最小値
5	AJCG3DAXIS_DIR_YM	Ｙ軸を降順で設定	
6	AJCG3DAXIS_DIR_ZM	Ｚ軸を降順で設定	

戻り値 ： TRUE　- 成功
FALSE - 失敗

6.9.30. プロファイルからプロパティ値読出し／書き込み (AjcG3d{Load/Save}Prop[Ex]) ・・・ ３Ｄモード専用

形　式 ： BOOL AjcG3dLoadProp (HWND hwnd, C_UTP pProfileSect, PCAJC3DGPROP pDefProp); --- 読み出し
BOOL AjcG3dLoadPropEx(HWND hwnd, C_UTP pProfileSect, PCAJC3DGPROP pDefProp); --- 〃

BOOL AjcG3dSaveProp (HWND hwnd, C_UTP pProfileSect); ---------- 書き込み
BOOL AjcG3dSavePropEx(HWND hwnd, C_UTP pProfileSect); ---------- 〃

引　数 ： hwnd - コントロールのウインドハンドル
pProfileSect - プロファイル・セクション名（文字列）へのポインタ
pDefProp - デフォルトプロパティ値へのポインタ（現在の設定値をデフォルトとする場合は NULL）

説　明 ： AjcG3dLoadProp[Ex]は、プロファイル（.ini ファイル／レジストリ）からプロパティ値を読み出して設定します。
AjcG3dLoadPropEx は、プロパティ値に加えて、フィルタ設定値とウインドスタイルも読み出して設定します。
pDefProp は、プロファイルに当該プロパティ値が記録されていない場合の、デフォルト・プロパティ値を指定します。
pDefProp=NULL とした場合は、現在設定されているプロパティ値を、デフォルト・プロパティとして扱います。

AjcG3dSaveProp は、現在設定されているプロパティ値を、プロファイル（.ini ファイル／レジストリ）へ記録します。
AjcG3dSavePropEx は、プロパティ値に加えて、フィルタ設定値とウインドスタイルも記録します。

AjcG3dLoadProp(),AjcG3dSaveProp()でプロファイルへセーブ／ロードする項目は以下のとおりです。

#	内容	キー名称	備考
1	各軸の回転角度	RotX, RotY, RotZ	視点位置
2	各軸の中心位置	CentX, CentY, CentZ	
3	各軸の半径	xr, yr, zr	
4	視野に対する半径の割合	ratio	視野の広さ
5	プロット点の大きさ，プロット最終点の大きさ	PlotSize, PlotSizeE	
6	プロット線描画フラグ	fPlotLine	
7	アスペクト比を１に保つ	fAspect1	0:ウインドサイズに合わせる 1: アスペクト比を１に保つ
8	各軸の描画色（原点より手前側）	rgbPX, rgbPY, rgbPZ	
9	各軸の描画色（原点より向う側）	rgbNX, rgbNY, RGNNZ	
10	各データのプロット数	MaxPlot0 ～ MaxPlotF	
11	各データの描画色（原点より手前側）	rgbP0 ～ rgbPF	
12	各データの描画色（原点より向う側）	rgbN0 ～ rgbNF	

AjcG3dLoadPropEx(),AjcG3dSavePropEx()では、さらに以下の項目が追加されます。

#	内容	キー名称	備考
1	データ項目選択状態	FilterValue	チェックボックスの内容
2	スタイル値	WndStyle	
3	平面表示時の横方向の軸と昇順／降順	PlaneAxisH	２Ｄモード時は、ロード時に平面のアングルを設定
4	平面表示時の横方向の軸と昇順／降順	PlaneAxisV	

戻り値 ： TRUE - 成功
FALSE - 失敗

6.9.31. ビットマップ取得 (AjcG3dGetBitmap) ・・・ ３Ｄモード専用

形　式 ： HBITMAP AjcG3dGetBitmap (HWND hwnd);

引　数 ： hwnd - コントロールのウインドハンドル

説　明 ： 現在表示している、３Ｄグラフィック・メージのビットマップデータを取得します。

戻り値 ： ≠NULL - 成功（ビットマップハンドル）
＝NULL - 失敗

6.9.32. テキスト描画フォント設定(AjcG3dSetTextFont) ・・・ ３Ｄモード専用

形　式 ： HFONT　AjcG3dSetTextFont (HWND hwnd, HFONT hFont);

引　数 ： hwnd　　－ コントロールのウインドハンドル

説　明 ： AjcG3dTextOut[V]()や、AjcG3dPrintF[V]()で描画するテキストのフォントを設定します。

戻り値 ： ≠NULL － 前回のフォントハンドル
＝NULL － 失敗

6.9.33. テキスト描画(AjcG3dTextOut[V]) ・・・ ３Ｄモード専用

形　式 ： UI　AjcG3dTextOut (HWND hwnd, int x, int y, C_UTP pTxt);
UI　AjcG3dTextOutV(HWND hwnd, AJCG3DTXOMD md, PCAJC3DVEC pV, C_UTP pTxt);

引　数 ： hwnd　－ コントロールのウインドハンドル
x　　－ 描画ピクセルX位置（テキストを右隅／中央に表示する場合は、「AJCTXO_RIGHT　± n」or「AJCTXO_CENTER　± n」を指定）
y　　－ 描画ピクセルY位置（テキストを下隅／中央に表示する場合は、「AJCTXO_BOTTOM ± n」or「AJCTXO_CENTER　± n」を指定）
md　－ テキスト表示方法
pV　－ 描画位置（３Ｄ座標値で指定）
pTxt　－ 描画するテキストへのポインタ

説　明 ： 指定した描画位置へテキストを表示します。
AjcG3dTextOut()は、ウインド上のピクセル位置を指定してテキストを描画します。
AjcG3dTextOutV()は、グラフィック上の３Ｄ座標位置にテキストを描画します。
md は、テキストの表示方法を以下のシンボルで指定します。

値	シンボル	内容	表示イメージ（青点（ ● ）が描画位置）
0	AJCG3DTXOMD_ RIGHT	描画位置の右側	●表示テキスト
1	AJCG3DTXOMD_ LEFT	描画位置の左側	表示テキスト●
2	AJCG3DTXOMD_ CENTER	描画位置に中央揃え	表示テキスト
3	AJCG3DTXOMD_BELLOW_RIGHT	描画位置の下の右側	●表示テキスト
4	AJCG3DTXOMD_BELLOW_LEFT	描画位置の下の左側	表示テキスト●
5	AJCG3DTXOMD_BELLOW_CENTER	描画位置の下に中央揃え	●表示テキスト
6	AJCG3DTXOMD_ABOVE_RIGHT	描画位置の上の右側	●表示テキスト
7	AJCG3DTXOMD_ABOVE_LEFT	描画位置の上の左側	表示テキスト●
8	AJCG3DTXOMD_ABOVE_CENTER	描画位置の上に中央揃え	●表示テキスト

pTxt で指定する描画テキストには、エスケープシーケンス（「テキスト描画」の章参照）を含めることができます。
但し、パレット色（０～７）は、データ項目の表示色と同じとなります。

戻り値 ： ≠0 － テキストキー
＝0 － 失敗

6.9.34. 書式テキスト描画(AjcG3dPrintF[V]) ・・・ ３Ｄモード専用

形　式 ： UI AjcG3dPrintF (HWND hwnd, int x, int y, C_UTP pFmt, ...);
UI AjcG3dTPrintFV(HWND hwnd, AJCG3DTXOMD md, PCAJC3DVEC pV, C_UTP pFmt, ...);

引　数 ： hwnd - コントロールのウインドハンドル
x - 描画ピクセルX位置（テキストを右隅／中央に表示する場合は、「AJCTXO_RIGHT ± n」or「AJCTXO_CENTER ± n」を指定）
y - 描画ピクセルY位置（テキストを下隅／中央に表示する場合は、「AJCTXO_BOTTOM ± n」or「AJCTXO_CENTER ± n」を指定）
md - テキスト表示方法
pV - 描画位置（３Ｄ座標値で指定）
pFmt - 書式テキストへのポインタ（printf()と同じ）

説　明 ： 指定した描画位置へ書式化したテキストを表示します。
AjcG3dPrintF()は、ウインド上のピクセル位置を指定して書式化されたテキストを描画します。
AjcG3dPrintFV()は、グラフィック上の３Ｄ座標位置に書式化されたテキストを描画します。
md は、AjcG3dTextOut[V]()と同じです。
描画するテキストには、エスケープシーケンス（「テキスト描画」の章参照）を含めることができます。
但し、パレット色（０～７）は、データ項目の表示色と同じとなります。

戻り値 ： ≠0 - テキストキー
＝0 - 失敗

6.9.35. 描画テキスト取得(AjcG3dGetText) ・・・ ３Ｄモード専用

形　式 ： UI AjcG3dGetText (HWND hwnd, UI key, UTP pBuf, UI lBuf);

引　数 ： hwnd - コントロールのウインドハンドル
key - テキストキー（AjcG3dTextOut[V]()や、AjcG3dPrintF[V]()の戻り値）
pBuf - 取得したテキストを格納するバッファへのポインタ（不要時は NULL）
lBuf - 取得したテキストを格納するバッファのバイト数／文字数

説　明 ： AjcG3dTextOut[V]()や、AjcG3dPrintF[V]()で描画した文字列を取得します。

戻り値 ： ≠0 - 描画テキストのバイト数／文字数
＝0 - 失敗

6.9.36. 図形描画データ消去(AjcG3dClear[All]Shape) ・・・ ３Ｄモード専用

形　式 ： BOOL AjcG3dClearShape (HWND hwnd, UI id); // 指定 id の図形描画データ消去
BOOL AjcG3dClearAllShape (HWND hwnd); // すべての図形描画データ消去

引　数 ： hwnd - コントロールのウインドハンドル
id - データ項目番号（０～１５）

説　明 ： 図形描画データを破棄します。

戻り値 ： TRUE - 成功
FALSE - 失敗

6.9.37. プロットデータ消去(AjcG3dClear[All]Plot) ・・・ ３Ｄモード専用

形　式 ： BOOL AjcG3dClearPlot (HWND hwnd, UI id); // 指定 id のプロットデータ消去
BOOL AjcG3dClearAllPlot (VO); // すべてのプロットデータ消去

引　数 ： hwnd - コントロールのウインドハンドル
id - データ項目番号（０～１５）

説　明 ： プロットデータを破棄します。

戻り値 ： TRUE - 成功
FALSE - 失敗

6.9.38. 描画テキスト消去(AjcG3dClear[All]Text) ・・・ ３Ｄモード専用

形　式 ： BOOL AjcG3dClearText (HWND hwnd, UI key); // １つの描画テキスト消去
BOOL AjcG3dClearAllText (HWND hwnd); // すべての描画テキスト消去

引　数 ： hwnd - コントロールのウインドハンドル
key - テキストキー（AjcG3dTextOut[V]()や、AjcG3dPrintF[V]()の戻り値）

説　明 ： AjcG3dTextOut[V]()や、AjcG3dPrintF[V]()で描画した文字列を消去します。

戻り値 ： TRUE - 成功
FALSE - 失敗

6.9.39. 描画データ消去(AjcG3dClear[All]Data) ・・・ ３Ｄモード専用

形　式 ： BOOL AjcG3dClearData (HWND hwnd, UI id); // １つの描画データ消去
BOOL AjcG3dClearAllData(HWND hwnd); // すべての描画データ消去

引　数 ： hwnd - コントロールのウインドハンドル
id - データ項目番号（０～１５）

説　明 ： 図形描画データとプロットデータを消去します。

戻り値 ： TRUE - 成功
FALSE - 失敗

6.9.40. 全てのデータ消去(AjcG3dClear) ・・・ ３Ｄモード専用

形　式 ： BOOL AjcG3dClear (HWND hwnd);

引　数 ： hwnd - コントロールのウインドハンドル

説　明 ： すべての図形描画データ、プロットデータ、描画テキストデータを消去します。

戻り値 ： TRUE - 成功
FALSE - 失敗

6.9.41. ２Ｄグラフモード初期化(AjcG2dInit[V]) ・・・ ２Ｄモード専用

形　式 ： BOOL　AjcG2dInit (HWND hwnd, double x1, double y1, double x2, double y2, UI style);
BOOL　AjcG2dInitV(HWND hwnd, PCAJC2DVEC pLo　　, PCAJC2DVEC pHi　　, UI style);

引　数 ： hwnd　　- コントロールのウインドハンドル
x1, y1 ／ pLo　- 各軸の低位値（pLo=NULL / pHi=NULL の場合は設定しない）
x2, y2 ／ pHi　- 各軸の高位値（pLo=NULL / pHi=NULL の場合は設定しない）
style　　- コントロールのスタイル(AJC3DGS_*XXXXX*, -1 の場合は設定しない)

説　明 ： ２Ｄグラフモードの初期設定を行います。
各座標軸の範囲が x1～x2, y1～y2 ／pLo～pHi に設定されます。
pLo/pHi が NULL の場合は、各座標軸の範囲を設定しません。（キャプション文字列によるパラメタ（XC. YC, XR, YR）が有効となります）

style は、任意のコントロール・スタイル(AJC3DGS_*XXXXX*)を指定可能ですが、標準的には「AJC3DGS_2DMODE」を指定します。(AJC3DGS_2DMODE では、方眼スケールとスケール値を表示します)
style=-1 とした場合は、(「AJC3DGS_NODEPTHCTRL」「AJC3DGS_NOANGLE」を除き）スタイルの設定を行いません。

このＡＰＩを実行すると、強制的に以下の内容が設定されます。
・ビューボリュームの割合＝1.0 が設定されます。
・「AJC3DGS_NODEPTHCTRL」「AJC3DGS_NOANGLE」スタイルが設定されます。
・視点をＺ軸に合わせて、ＸＹ平面が見えるように設定する。

戻り値 ： TRUE　- 成功
FALSE - 失敗

6.9.42. 平面アングル設定(AjcG2dSetPlane) ・・・ ２Ｄモード専用

形　式 ： BOOL　AjcG2dSetPlane (HWND hwnd, AJCG2DAXIS_DIR HoriAxis, AJCG2DAXIS_DIR VertAxis);

引　数 ： hwnd　　- コントロールのウインドハンドル
HoriAxis　- 横方向の軸の種別と昇順／降順
VertAxis　- 縦方向の軸の種別と昇順／降順

説　明 ： 平面がの表示種別（縦／横方向の軸, 値の昇順／降順）を設定します。
HoriAxis は、横方向にマッピングする軸と、値の昇順／降順を指定します。
VertAxis は、縦方向にマッピングする軸と、値の昇順／降順を指定します。

#	シンボル	内容	備考
1	AJCG2DAXIS_DIR_XP	Ｘ軸を昇順で設定	左端／下端が最小値　→　右端／上端が最大値
2	AJCG2DAXIS_DIR_YP	Ｙ軸を昇順で設定	
4	AJCG2DAXIS_DIR_XM	Ｘ軸を降順で設定	左端／下端が最大値　→　右端／上端が最小値
5	AJCG2DAXIS_DIR_YM	Ｙ軸を降順で設定	

戻り値 ： TRUE　- 成功
FALSE - 失敗

6.9.43. プロパティ設定／取得 (AjcG2d{Set/Get}Prop) ・・・ ２Ｄモード専用

形　式 ： BOOL　AjcG2dSetProp(HWND hwnd, PCAJC2DGPROP pProp)； -- プロパティ設定
BOOL　AjcG2dGetProp(HWND hwnd, PAJC2DGPROP　pBuf)； --- プロパティ取得

引　数 ： hwnd　　- コントロールのウインドハンドル
pProp　　- 設定するプロパティデータのアドレス
pBuf　　- 取得したプロパティデータを格納するバッファのアドレス

説　明 ： プロパティの設定／取得を行います。
プロパティデータの内容については、「３Ｄ／２Ｄグラフィック・コントロールのプロパティ構造体」を参照してください。

戻り値 ： TRUE　- 成功
FALSE - 失敗

6.9.44. レンジ自動調整(AjcG2dAdjustRange) ・・・ ２Ｄモード専用

形　式 ： BOOL　AjcG2dAdjustRange (HWND hwnd)；

引　数 ： hwnd　　- コントロールのウインドハンドル

説　明 ： （フィルタで非表示となっている項目を除く）全てのデータから、最大値と最小値を算出し、±５％のマージンを持って各座標軸の長さを調整します。
つまり、全てのデータがウインドに表示されるように、グラフのレンジを調整します。
但し、各座標軸の中心位置は変化しません。

戻り値 ： TRUE　- 成功
FALSE - 失敗

6.9.45. 各軸のレンジ設定 (AjcG2dSetRange) ・・・ ２Ｄモード専用

形　式 ： BOOL　AjcG2dSetRange (HWND hwnd, double x1, double y1, double x2, double y2)；

引　数 ： hwnd　　- コントロールのウインドハンドル
x1, y1　　- 各軸の低位値
x2, y2　　- 各軸の高位値

説　明 ： 各座標軸のレンジを設定します。
各座標軸の範囲値は、x1 < x2, y1 < y2, z1 < z2 でなければなりません。

戻り値 ： TRUE　- 成功
FALSE - 失敗

6.9.46. 各座標軸の中心位置設定(AjcG2dSetCenter) ・・・ ２Ｄモード専用

形　式 ： BOOL　AjcG2dSetCenter (HWND hwnd, double xc, double yc)；

引　数 ： hwnd　　- コントロールのウインドハンドル
xc, yc　　- 各軸の中心位置

説　明 ： 各座標軸の中心位置を設定します。
各座標軸の長さは、変化しません。

戻り値 ： TRUE　- 成功
FALSE - 失敗

6.9.47. 各軸の幅（半径）設定(AjcG2dSetWidth) ・・・ ２Ｄモード専用

形　式 ： BOOL　AjcG2dSetWidth (HWND hwnd, double xr, double yr) ;

引　数 ： hwnd　　- コントロールのウインドハンドル
xr, yr　　- 各軸の長さ（半径）

説　明 ： 各座標軸長さ（半径）を設定します。
各座標軸の中心位置は変化しません。

戻り値 ： TRUE　- 成功
FALSE - 失敗

6.9.48. 各軸のレンジ幅を同じにする (AjcG2dSetSameRangeWidth) ・・・ ２Ｄモード専用

形　式 ： BOOL　AjcG2dSetSameRangeWidth (HWND hwnd) ;

引　数 ： hwnd　　- コントロールのウインドハンドル

説　明 ： 中心位置を変更せずに、各軸のレンジ幅を同一に設定します。(最大のレンジ幅に合わせます)

(例)　変更前のレンジ

	レンジ	中心	幅
X	100～500	300	400
Y	300～400	350	100

⇒

変更後のレンジ

	レンジ	中心	幅
X	100～500	300	400
Y	**150～550**	350	**400**

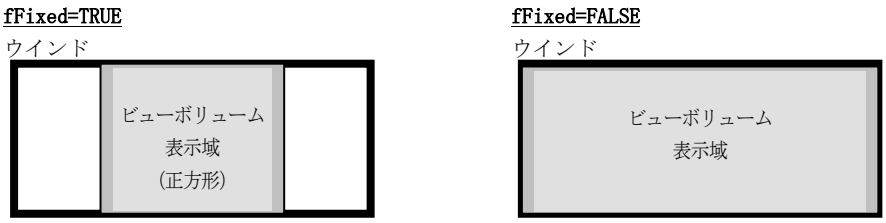
戻り値 ： TRUE　- 成功
FALSE - 失敗

6.9.49. レンジ設定された領域のアスペクトを設定する (AjcG2dSetFixedAspect) ・・・２Ｄモード専用

形　式 ： BOOL　AjcG2dSetFixedAspect (HWND hwnd, BOOL fFixed)；

引　数 ： hwnd　　- コントロールのウインドハンドル
fFixed　- TRUE　：レンジ設定された領域のアスペクトを１とする
FALSE ：アスペクトをウィンドサイズに合わせて可変にする

説　明 ： この関数は、単に、プロパティ「fAspect1」の設定を行います。
fFixed=TRUE を指定した場合、レンジ設定された範囲を、ウインドの正方形領域にマッピングして描画します。
fFixed=FALSE を指定した場合、レンジ設定された範囲を、ウインド全体ににマッピングして描画します。



fFixed=TRUE とし、各軸のレンジ幅を同一にすることにより、円や、正方形が歪まないで表示されます。
fFixed=FALSE とした場合は、ウインドサイズにより、円や、正方形がつぶれたり、細長く表示されます。

戻り値 ： TRUE　- 成功
FALSE - 失敗

6.9.50. 表示色設定(AjcG2dSetColor) ・・・ ２Ｄモード専用

形　式 ： BOOL　AjcG2dSetColor (HWND hwnd, UI id, COLORREF rgb)；

引　数 ： hwnd　　- コントロールのウインドハンドル
id　　- データ項目番号（０～１５）
rgb　　- 表示色

説　明 ： 「id」で指定されたデータ項目の表示色を設定します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

6.9.51. プロットデータ数設定(AjcG2dSetPlotNumber) ・・・ ２Ｄモード専用

形　式 ： BOOL　AjcG2dSetPlotNumber (HWND hwnd, UI id, UI PlotNumber)；

引　数 ： hwnd　　- コントロールのウインドハンドル
id　　- データ項目番号（０～１５）
PlotNumber - プロットデータ数

説　明 ： 「id」で指定されたデータ項目におけるプロットデータの保留個数を設定します。
AjcPutPlotData[V]でプロットデータを投与した際、設定されているプロットデータ数を超える場合は、古いプロットデータが破棄されます。
つまり、プロットデータは、本関数で設定された個数の最新データだけが表示されることになります。

本関数を実行すると、現在保留されているプロットデータは、一旦破棄されます。

戻り値 ： TRUE　- 成功
FALSE - 失敗

6.9.52. プロットデータのピクセルサイズ設定(AjcG2dSetPlotSize) ・・・ ２Ｄモード専用

形　式 ： BOOL　AjcG2dSetPlotSize (HWND hwnd, UI id, UI PixelSize, UI PixelSizeE);

引　数 ： hwnd　　- コントロールのウインドハンドル
id　　- データ項目番号（０～１５）
PixelSize　- プロットデータの表示ピクセルサイズ
PixelSizeE - 終点プロットデータの表示ピクセルサイズ

説　明 ： 「id」で指定されたデータ項目におけるプロットデータの表示ピクセルサイズを設定します。
「PixelSize」は、終点（最新のプロットデータ）以外の表示ピクセルサイズを指定します。
「PixelSizeE」は、終点（最新のプロットデータ）の表示ピクセルサイズを指定します。
「PixelSize」「PixelSize」とも、０か１を指定した場合は、１ドットでプロットデータを表示します。
２以上を設定した場合は、当該指定値の半径で、塗りつぶし円を表示します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

6.9.53. プロットデータ投与(AjcG2dPutPlotData) ・・・ ２Ｄモード専用

形　式 ： BOOL　AjcG2dPutPlotData (HWND hwnd, UI id, double x, double y);

引　数 ： hwnd　　- コントロールのウインドハンドル
id　　- データ項目番号（０～１５）
x, y　　- プロットデータの座標値

説　明 ： プロットデータをバッファに格納します。
プロットデータが満杯である場合は、最古のプロットデータが破棄されます。

戻り値 ： TRUE　- 成功
FALSE - 失敗

6.9.54. プロットデータ数取得(AjcG3dGetPlotDataCount) ・・・ ２Ｄモード専用

形　式 ： UI　　AjcG2dGetPlotDataCount (HWND hwnd, UI id);
UI　　AjcG2dGetPlotMaxCount　(HWND hwnd);

引　数 ： hwnd　　- コントロールのウインドハンドル
id　　- データ項目番号（０～１５）

説　明 ： AjcG2dGetPlotDataCoun()は、当該データ項目のプロットデータの個数を取得します。
AjcG2dGetPlotMaxCoun()は、全データ項目で最大のプロットデータの個数を取得します。

戻り値 ： プロットデータ数（エラー時は０を返す）

6.9.55. プロファイルからプロパティ値読出し／書き込み (AjcG2dLoadProp[Ex]) ・・・ ２Ｄモード専用

形　式 ： BOOL AjcG2dLoadProp (HWND hwnd, C_UTP pProfileSect, PCAJC2DGPROP pDefProp); --- 読み出し
BOOL AjcG2dLoadPropEx(HWND hwnd, C_UTP pProfileSect, PCAJC2DGPROP pDefProp); --- 〃

BOOL AjcG2dSaveProp (HWND hwnd, C_UTP pProfileSect); ------------------------- 書き込み
BOOL AjcG2dSavePropEx(HWND hwnd, C_UTP pProfileSect); ------------------------ 〃

引　数 ： hwnd - コントロールのウインドハンドル
pProfileSect - プロファイル・セクション名へのポインタ
pDefProp - デフォルトプロパティ値へのポインタ（現在の設定値をデフォルトとする場合はNULL）

説　明 ： AjcG2dLoadPropは、プロファイル（.iniファイル／レジストリ）からプロパティ値を読み出して設定します。
AjcG2dLoadPropExは、プロパティ値に加えて、フィルタ設定値とウインドスタイルも読み出して設定します。
pDefPropは、プロファイルに当該プロパティ値が記録されていない場合の、デフォルト・プロパティ値を指定します。
pDefProp=NULLとした場合は、現在設定されているプロパティ値を、デフォルト・プロパティとして扱います。

AjcG2dSavePropは、現在設定されているプロパティ値を、プロファイル（.iniファイル／レジストリ）へ記録します。
AjcG2dSavePropExは、プロパティ値に加えて、フィルタ設定値とウインドスタイルも記録します。

プロファイルへセーブ／ロードする項目はAjcG3dLoadProp[Ex]()/ AjcG3dSaveProp[Ex]()と同じです。

戻り値 ： TRUE - 成功
FALSE - 失敗

6.9.56. ボーダー色で囲まれた部分の塗りつぶし(AjcG2dFillB) ・・・ ２Ｄモード専用

形　式 ： BOOL AjcG2dFillB (HWND hwnd, UI idFill, UI idBorder, double x, double y);
BOOL AjcG2dFillBV(HWND hwnd, UI idFill, UI idBorder, PCAJC2DVEC pPoint);

引　数 ： hwnd - コントロールのウインドハンドル
idFill - 塗りつぶし色（データ項目番号（０～１５）で指定）
idBorder - ボーダー色 （データ項目番号（０～１５）で指定）
x, y - 塗りつぶし位置
pPoint - 塗りつぶし位置データのアドレス

説　明 ： 平面の指定された座標位置(x, y)をボーダー色で囲む閉領域を塗りつぶします。
塗りつぶしは、他の図形描画（直線，三角形，楕円・・・）の後に実行されます。

戻り値 ： TRUE - 成功
FALSE - 失敗

6.9.57. 連続する白色部分の塗りつぶし(AjcG2dFillS) ・・・ ２Ｄモード専用

形　式 ： BOOL AjcG2dFillS (HWND hwnd, UI idFill, double x, double y);
BOOL AjcG2dFillSV(HWND hwnd, UI idFill, PCAJC2DVEC pPoint);

引　数 ： hwnd - コントロールのウインドハンドル
idFill - 塗りつぶし色（データ項目番号（０～１５）で指定）
x, y - 塗りつぶし位置
pPoint - 塗りつぶし位置データのアドレス

説　明 ： 平面の指定された座標位置(x, y)の周囲の連続した白色部分を塗りつぶします。
指定した座標位置(x, y)は、白色でなければなりません。
塗りつぶしは、他の図形描画（直線，三角形，楕円・・・）の後に実行します。

戻り値 ： TRUE - 成功
FALSE - 失敗

6.9.58. ピクセルの表示色取得 (AjcG2dGetPixel) ・・・ ２Ｄモード専用

形　式 ： COLORREF AjcG2dGetPixel (HWND hwnd, double x, double y);
COLORREF AjcG2dGetPixelV(HWND hwnd, PCAJC2DVEC pPoint);

引　数 ： hwnd　　－ コントロールのウインドハンドル
x, y　　－ ピクセル色を取得する位置
pPoint　　－ ピクセル色を取得する位置データのアドレス

説　明 ： 平面の指定された座標位置(x, y)のピクセル色を取得します。

戻り値 ： ≠－１ － ピクセルの色
＝－１ – エラー

6.9.59. ビットマップデータ取得 (AjcG2dGetBitmap) ・・・ ２Ｄモード専用

形　式 ： HBITMAP　AjcG2dGetBitmap (HWND hwnd);

引　数 ： hwnd　　－ コントロールのウインドハンドル

説　明 ： 現在表示している、２Ｄグラフィック・メージのビットマップデータを取得します。

戻り値 ： ≠NULL － 成功（ビットマップハンドル）
＝NULL － 失敗

6.9.60. テキスト描画フォント設定(AjcG2dSetTextFont) ・・・ ２Ｄモード専用

形　式 ： HFONT　AjcG2dSetTextFont (HWND hwnd, HFONT hFont);

引　数 ： hwnd　　－ コントロールのウインドハンドル

説　明 ： AjcG2dtextout[V]()や、AjcG2dPrintF[V]()で描画するテキストのフォントを設定します。

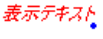
戻り値 ： ≠NULL － 前回のフォントハンドル
＝NULL － 失敗

6.9.61. テキスト描画(AjcG2dTextOut[V]) ・・・ ２Ｄモード専用

形　式 ： UI　AjcG2dTextOut（HWND hwnd, int x, int y, C_UTP pTxt);
UI　AjcG2dTextOutV(HWND hwnd, AJCG2DTXOMD md, PCAJC2DVEC pV, C_UTP pTxt);

引　数 ： hwnd　　－ コントロールのウインドハンドル
x　　－ 描画ピクセルX位置（テキストを右隅／中央に表示する場合は、「AJCTXO_RIGHT ± n」or「AJCTXO_CENTER ± n」を指定）
y　　－ 描画ピクセルＹ位置（テキストを下隅／中央に表示する場合は、「AJCTXO_BOTTOM ± n」or「AJCTXO_CENTER ± n」を指定）
md　　－ テキスト表示方法
pV　　－ 描画位置（２Ｄ座標値で指定）
pTxt　　－ 描画するテキストへのポインタ

説　明 ： 指定した描画位置へテキストを表示します。
AjcG2dTextOut()は、ウインド上のピクセル位置を指定してテキストを描画します。
AjcG2dTextOutV()は、グラフィックの２Ｄ座標位置にテキストを描画します。
md は、テキストの表示方法を以下のシンボルで指定します。

値	シンボル	内容	表示イメージ（青点（ ● ）が描画位置）
0	AJCG2DTXOMD_ RIGHT	描画位置の右側	
1	AJCG2DTXOMD_ LEFT	描画位置の左側	
2	AJCG2DTXOMD_ CENTER	描画位置に中央揃え	
3	AJCG2DTXOMD_BELLOW_RIGHT	描画位置の下の右側	
4	AJCG2DTXOMD_BELLOW_LEFT	描画位置の下の左側	
5	AJCG2DTXOMD_BELLOW_CENTER	描画位置の下に中央揃え	
6	AJCG2DTXOMD_ABOVE_RIGHT	描画位置の上の右側	
7	AJCG2DTXOMD_ABOVE_LEFT	描画位置の上の左側	
8	AJCG2DTXOMD_ABOVE_CENTER	描画位置の上に中央揃え	

pTxt で指定する描画テキストには、エスケープシーケンス（「テキスト描画」の章参照）を含めることができます。
但し、パレット色（０～７）は、データ項目の表示色と同じとなります。

戻り値 ： ≠0 － テキストキー
＝0 － 失敗

6.9.62. 書式テキスト描画(AjcG2dPrintF[V]) ・・・ ２Ｄモード専用

形　式 ： UI AjcG2dPrintF (HWND hwnd, int x, int y, C_BCP pFmt, ...);
UI AjcG2dTPrintFV(HWND hwnd, AJCG2DTXOMD md, PCAJC2DVEC pV, C_BCP pFmt, ...);

引　数 ： hwnd　　- コントロールのウインドハンドル
x　　- 描画ピクセルX位置（テキストを右隅／中央に表示する場合は、「AJCTXO_RIGHT ± n」or「AJCTXO_CENTER ± n」を指定）
y　　- 描画ピクセルY位置（テキストを下隅／中央に表示する場合は、「AJCTXO_BOTTOM ± n」or「AJCTXO_CENTER ± n」を指定）
md　　- テキスト表示方法
pV　　- 描画位置（２Ｄ座標値で指定）
pFmt　　- 書式テキストへのポインタ（printf()と同じ）

説　明 ： 指定した描画位置へテキストを表示します。
AjcG2dPrintF()は、ウインド上のピクセル位置を指定して書式化されたテキストを描画します。
AjcG2dPrintFV()は、グラフィックの２Ｄ座標位置に書式化されたテキストを描画します。
md は、AjcG2dTextOut[V]()と同じです。
pTxt で指定する描画テキストには、エスケープシーケンス（「テキスト描画」の章参照）を含めることができます。

戻り値 ： ≠0 - テキストキー
＝0 - 失敗

6.9.63. 描画テキスト取得(AjcG2dGetText) ・・・ ２Ｄモード専用

形　式 ： UI AjcG2dGetText (HWND hwnd, UI key, BCP pBuf, UI lBuf);

引　数 ： hwnd　　- コントロールのウインドハンドル
key　　- テキストキー（AjcG2dTextOut[V]()や、AjcG2dPrintF[V]()の戻り値）
pBuf　　- 取得したテキストを格納するバッファへのポインタ（不要時は NULL）
lBuf　　- 取得したテキストを格納するバッファのバイト数／文字数

説　明 ： AjcG2dTextOut[V]()や、AjcG2dPrintF[V]()で描画した文字列を取得します。

戻り値 ： ≠0 - 描画テキストのバイト数／文字数
＝0 - 失敗

6.9.64. 図形描画データ消去(AjcG2dClearAll) ・・・ ２Ｄモード専用

形　式 ： BOOL AjcG2dClearShape (HWND hwnd, int id); // 指定 id の図形描画データ消去
BOOL AjcG2dClearAllShape (HWND hwnd); // すべての図形描画データ消去

引　数 ： hwnd　　- コントロールのウインドハンドル

説　明 ： 図形描画データを消去します。

戻り値 ： TRUE - 成功
FALSE - 失敗

6.9.65. プロットデータ消去(AjcG2dClear[All]Plot) ・・・ ２Ｄモード専用

形　式 ： BOOL AjcG2dClearPlot (HWND hwnd, UI id); // 指定 id のプロットデータ消去
BOOL AjcG2dClearAllPlot(HWND hwnd); // すべてのプロットデータ消去

引　数 ： hwnd　　- コントロールのウインドハンドル
id　　- データ項目番号（０～１５）

説　明 ： プロットデータを消去します。

戻り値 ： TRUE - 成功
FALSE - 失敗

6.9.66. 描画テキスト消去(AjcG2dClear[All]Text) ・・・ ２Ｄモード専用

形　式 ： BOOL AjcG2dClearText (HWND hwnd, UI key); // １つの描画テキスト消去
BOOL AjcG2dClearallText (HWND hwnd, UI key); // すべての描画テキスト消去

引　数 ： hwnd - コントロールのウインドハンドル
key - テキストキー（AjcG2dTextOut[V]()や、AjcG2dPrintF[V]()の戻り値）

説　明 ： AjcG2dTextOut[V]()や、AjcG2dPrintF[V]()で描画した文字列を消去します。

戻り値 ： TRUE - 成功
FALSE - 失敗

6.9.67. 描画データ消去(AjcG2dClear[All]Data) ・・・ ２Ｄモード専用

形　式 ： BOOL AjcG2dClearData (HWND hwnd, UI id); // １つの描画データ消去
BOOL AjcG2dClearAllData(HWND hwnd); // すべての描画データ消去

引　数 ： hwnd - コントロールのウインドハンドル
id - データ項目番号（０～１５）

説　明 ： 図形描画データとプロットデータを消去します。

戻り値 ： TRUE - 成功
FALSE - 失敗

6.9.68. 全てのデータ消去(AjcG3dClear) ・・・ ２Ｄモード専用

形　式 ： BOOL AjcG2dClear (HWND hwnd);

引　数 ： hwnd - コントロールのウインドハンドル

説　明 ： すべての図形描画データ、プロットデータ、描画テキストデータを消去します。

戻り値 ： TRUE - 成功
FALSE - 失敗

6.9.69. 平面にピクセル描画(AjcG2dPixel)

形　式 ： BOOL AjcG2dPixel (HWND hwnd, UI id, double x, double y, UI PixelSize);
BOOL AjcG2dPixelV(HWND hwnd, UI id, PCAJC2DVEC pPoint, UI PixelSize);

引　数 ： hwnd - コントロールのウインドハンドル
id - データ項目番号（０～１５）
x, y - ピクセル描画位置
pPoint - ピクセル描画位置データのアドレス
PixelSize - 表示ピクセルサイズ

説　明 ： 平面の指定された座標位置に、ピクセル（点）を描画します。
「PixelSize」に０か１を指定した場合は、１ドットでプロットデータを表示します。
２以上を設定した場合は、当該指定値の半径で、塗りつぶし円を表示します。

戻り値 ： TRUE - 成功
FALSE - 失敗

6.9.70. 平面のライン始点設定(AjcG2dMoveTo)

形　式 ： BOOL　AjcG2dMoveTo (HWND hwnd, UI id, double x, double y) ;
BOOL　AjcG2dMoveToV(HWND hwnd, UI id, PCAJC2DVEC ps);

引　数 ： hwnd　　- コントロールのウインドハンドル
id　　- データ項目番号（０～１５）
x, y, z　- 線分の始点位置
ps　　- 線分の始点位置データのアドレス

説　明 ： 平面に描画する線分描画の始点を設定します。
線分を描画するには、AjcV2dLineTO()/AjcV2dLineToV()を実行します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

6.9.71. 平面のライン終点設定 - ライン／矢印描画(AjcG2dMoveTo)

形　式 ： BOOL　AjcG2dLineTo (HWND hwnd, UI id, double x, double y) ;
BOOL　AjcG2dLineToV(HWND hwnd, UI id, PCAJC2DVEC pe);

BOOL　AjcG2dArrowTo (HWND hwnd, UI id, double x, double y) ;
BOOL　AjcG2dArrowToV(HWND hwnd, UI id, PCAJC2DVEC pe);

引　数 ： hwnd　　- コントロールのウインドハンドル
id　　- データ項目番号（０～１５）
x, y　　- 線分の終点位置
pe　　- 線分の終点位置データのアドレス

説　明 ： 平面に描画する線分描画の終点を指定し、線分を描画します。
AjcG2dArrowTo(),AjcG2dArrowToV()は、線分の終点に矢印を描画します。(但し、線分が10ピクセルに満たない場合は矢印を描画しません)
線分の始点は、この前に実行したAjcV2dMoveTo()/AjcV2dMobveToV()で指定した始点か、AjcV2dLineTo()/AjcV2dLineToV()で指定した終点となります。
指定した線分の終点は、新たな線分の始点として設定されます。

戻り値 ： TRUE　- 成功
FALSE - 失敗

6.9.72. 平面にライン／矢印描画(AjcG2dLine)

形　式 ： BOOL　AjcG2dLine　(HWND hwnd, UI id, double x1, double y1, double x2, double y2) ;
BOOL　AjcG2dLineV (HWND hwnd, UI id, PCAJC2DVEC p1, PCAJC2DVEC p2);

BOOL　AjcG2dArrow (HWND hwnd, UI id, double x1, double y1, double x2, double y2) ;
BOOL　AjcG2dArrowV(HWND hwnd, UI id, PCAJC2DVEC p1, PCAJC2DVEC p2);

引　数 ： hwnd　　- コントロールのウインドハンドル
id　　- データ項目番号（０～１５）
x1, y1　- 線分の始点位置
x2, y2　- 線分の終点位置
p1　　- 線分の始点位置データのアドレス
p2　　- 線分の終点位置データのアドレス

説　明 ： 平面の指定された座標位置に、線分を描画します。
AjcG2dArrow(),AjcG2dArrowV()は、線分の終点に矢印を描画します。(但し、線分が10ピクセルに満たない場合は矢印を描画しません)

戻り値 ： TRUE　- 成功
FALSE - 失敗

6.9.73. 平面に三角形描画(AjcG2dTriangle)

形　式 ： BOOL　AjcG2dTriangle (HWND hwnd, UI id, double x1, double y1, double x2, double y2, double x3, double y3)；
BOOL　AjcG2dTriangleV(HWND hwnd, UI id, PCAJC2DVEC p1, PCAJC2DVEC p2, PCAJC2DVEC p3)；

引　数 ： hwnd　　- コントロールのウインドハンドル
id　　- データ項目番号（０～１５）
x1, y1, x2, y2, x3, y3 - 三角形の頂点位置
p1, p2, p3 ---------- 三角形の頂点位置データのアドレス

説　明 ： 平面上の指定された３つの頂点を結んだ三角形を描画します。

戻り値 ： TRUE - 成功
FALSE - 失敗

6.9.74. 平面に四角形描画(AjcG2dSquare)

形　式 ： BOOL　AjcG2dSquare (HWND hwnd, UI id, double x1, double y1, double x2, double y2, double x3, double y3, double x4, double y4)；
BOOL　AjcG2dSquareV(HWND hwnd, UI id, PCAJC2DVEC p1, PCAJC2DVEC p2, PCAJC2DVEC p3, PCAJC2DVEC p4)；

引　数 ： hwnd　　- コントロールのウインドハンドル
id　　- データ項目番号（０～１５）
x1, y1, x2, y2, x3, y3, x4, y4 - 四角形の頂点位置
p1, p2, p3, p4 ---------- 四角形の頂点位置データのアドレス

説　明 ： 平面上の指定された４つの頂点を結んだ四角形を描画します。

戻り値 ： TRUE - 成功
FALSE - 失敗

6.9.75. 平面に長方形描画(AjcG2dRectangle)

形　式 ： BOOL　AjcG2dRectangle (HWND hwnd, UI id, double x1, double y1, double x2, double y2)；
BOOL　AjcG2dRectangleV(HWND hwnd, UI id, PCAJC2DVEC p1, PCAJC2DVEC p2)；

引　数 ： hwnd　　- コントロールのウインドハンドル
id　　- データ項目番号（０～１５）
x1, y1, x2, y2 - 長方形の対角頂点位置
p1, p2 -------- 長方形の対角頂点位置データのアドレス

説　明 ： 平面上の指定された２つの位置を対角とする長方形を描画します。

戻り値 ： TRUE - 成功
FALSE - エラー

6.9.76. 平面に円／楕円描画(AjcG2dEllipse)

形　式 ： BOOL　AjcG2dEllipse (HWND hwnd, UI id, double xc, double yc, double rx, double ry);
BOOL　AjcG2dEllipseV(HWND hwnd, UI id, PCAJC2DVEC pCent, double rx, double ry);

引　数 ： hwnd　　- コントロールのウインドハンドル
id　　- データ項目番号（０～１５）
xc, yc　　- 円／楕円の中心位置
pCent　　- 円／楕円の中心位置データのアドレス
xr　　- Ｘ軸方向の半径
yr　　- Ｙ軸方向の半径

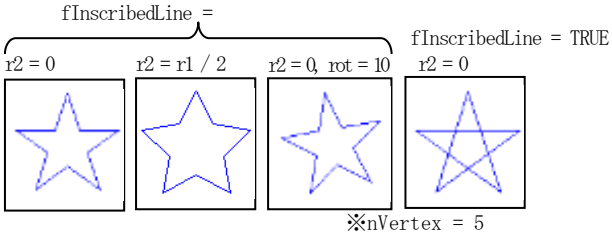
説　明 ： 平面上の指定された中心位置に、円／楕円を描画します。
円を描画するには、xr と yr を同じ値（円の半径）にします。

戻り値 ： TRUE　- 成功
FALSE - 失敗

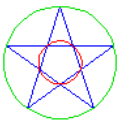
6.9.77. 平面に星形描画(AjcG2dStar[V][Ex])

形　式 ： BOOL AjcG2dStar　(HWND hwnd, UI id, double xc, double yc, double r);
BOOL AjcG2dStarV　(HWND hwnd, UI id, PCAJC2DVEC pCent　, double r);
BOOL AjcG2dStarEx (HWND hwnd, UI id, double xc, double yc, double r1, double r2, UI nVertex, double rot, BOOL fInscribedLine);
BOOL AjcG2dStarVEx(HWND hwnd, UI id, PCAJC2DVEC pCent　, double r1, double r2, UI nVertex, double rot, BOOL fInscribedLine);

引　数 ： hwnd　　- コントロールのウインドハンドル
id　　- データ項目番号（０～１５）
xc, yc　　- 星形の中心位置
pCent　　- 星形の中心位置データのアドレス
r, r1　　- 星形外円の半径
r2　　- 星形内円の半径（0 ： 自動計算）
nVertex　　- 頂点の数（５以上の奇数）
rot　　- 回転角度［度］（星形全体を左回りに回転）
fInscribedLine - 内円に内接する正Ｎ角形描画フラグ（TRUE:Ｎ角形を描画する, FALSE:描画しない）



説　明 ： 平面上の指定された中心位置に、星形を描画します。
r2 = 0 の場合は、外円(右図・緑円)の各頂点を直線で結んだ場合の内円(右図・赤円)の半径を自動計算します。
fInscribedLine =TRUE の場合は、内円に内接する正Ｎ角形を描画します。
AjcG2dStar()/AjcG2dStarV()の場合は、r2 = 0, rot = 0, fInscribedLine = FALSE を仮定します。



戻り値 ： TRUE　- 成功
FALSE - 失敗

6.9.78. 右クリック通知設定 (AjcG3dSetNtcRClk)

形　式 ： BOOL AjcG3dSetNtcRClk(HWND hwnd, BOOL fNtcRClk, UI MsgRBDown, UI MsgRBUp);

引　数 ： hwnd　　- コントロールのウインドハンドル
fNtcRClk　- 右ボタンの DOWN/UP 通知フラグ（TRUE:通知する, FALSE:通知しない）
MsgRBDown　- 右ボタン押下時の通知メッセージコード　（０の場合は非通知）
MsgRBUp　- 右ボタン離され時の通知メッセージコード（０の場合は非通知）

説　明 ： コントロールを右クリックした場合に、当該操作を親ウインドへ通知するか否かを設定します。
MsgRBDown, MsgRBUp は、WM_USER+100 以降, WM_APP+500 以降か、RegisterWindowMessage()で取得したコードを指定します。
各引数と、右クリック通知動作は以下のとおりです。

引数			Shift/ Ctrl	メッセージ	wParam	備考
fNtcRClk	MsgRBDown	MsgRBUp				
FALSE (default	－	－	未押下	WM_COMMAND	－（非通知）	ポップアップメニュー表示
			押下		ID + AJC3DGN_RCLICK	
TRUE	いずれかが０以外		－	MsgRBDown(押下時)	WM_RBUTTONDOWN の wParam	MsgRBDown ＝０の場合は非通知
				MsgRBUp（離し時）	WM_RBUTTONUP　の wParam	MsgRBUp　＝０の場合は非通知
	0	0	－		－（非通知）	－

右クリックの通知を禁止するには、fNtcRClk=TRUE, MsgRBDown=0, MsgRBUp=0 とします。

戻り値 ： TRUE　- 成功
FALSE - 失敗

6.9.79. ポップアップメニューの許可／禁止 (AjcG3dEnablePopupMenu)

形　式 ： BOOL AjcG3dEnablePopupMenu (HWND hwnd, BOOL fEnable);

引　数 ： hwnd　　- コントロールのウインドハンドル
fEnable　- ポップアップメニューの許可(TRUE)／禁止(FALSE)

説　明 ： 右クリックによるポップアップメニューを許可／禁止します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

備　考 ： 「AjcG3dSetNtcRClk(hwnd, !fEnable, WM_RBUTTONDOWN, WM_RBUTTONUP);」を実行します。

6.9.80. ツールチップの設定／取得 (AjcG3d{Set/Get}TipText)

形　式 ： BOOL AjcG3dSetTipText(HWND hwnd, C_UTP pTxt); --------- ツールチップ文字列の設定
BOOL AjcG3dGetTipText(HWND hwnd, UTP pBuf, UI lBuf); -- ツールチップ文字列の取得

引　数 ： hwnd　　- コントロールのウインドハンドル
pTxt　　- ツールチップ（ツールヒント）文字列のアドレス（表示しない場合は NULL）
pBuf　　- ツールチップ（ツールヒント）文字列を格納するバッファのアドレス
lBuf　　- ツールチップ（ツールヒント）文字列を格納するバッファの文字数

説　明 ： コントロール上にカーソルを置いたときに表示するツールヒント文字列を設定／取得します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

6.9.81. ツールチップ表示条件の設定／取得(AjcG3d{Set|Get}TipShowAlways)

形　式 ： BOOL AjcG3dSetTipShowAlways(HWND hwnd, BOOL fShowAlways); --- 設定
BOOL AjcG3dGetTipShowAlways(HWND hwnd); ---------- 取得
BOOL AjcG3dGetTipShowAlwaysAll(HWND hwnd, BOOL fShowAlways); --- すべて設定

引　数 ： hwnd - コントロールのウインドハンドル
fShowAlways - ツールチップ表示条件（TRUE:非アクティブ時も表示, FALSE:非アクティブ時は非表示）

説　明 ： ツールチップの表示条件（自プログラムが非アクティブ時の表示／非表示）を設定／取得します。
AjcG3dGetTipShowAlwaysAll()は、フィルタチェックボックスを含めたすべてのツールチップ表示条件を設定します。

戻り値 ： 設定時：TRUE - 成功　　取得時：ツールチップ表示条件
FALSE - 失敗

6.9.82. フィルタ・チェックボックス・ツールチップの設定／取得 (AjcG3d{Set/Get}ChkBoxTipText)

形　式 ： BOOL AjcG3dSetChkBoxTipText(HWND hwnd, UI n, C_UTP pTxt); ---------- チェックボックスのツールチップ設定
BOOL AjcG3dGetChkBoxTipText(HWND hwnd, UI n, C_UTP pBuf, UI lBuf); - チェックボックスのツールチップ取得

引　数 ： hwnd - コントロールのウインドハンドル
n - チェックボックスのインデクス（0～15）
pTxt - ツールチップ（ツールヒント）文字列のアドレス（表示しない場合はNULL）
pBuf - ツールチップ（ツールヒント）文字列を格納するバッファのアドレス
lBuf - ツールチップ（ツールヒント）文字列を格納するバッファの文字数

説　明 ： フィルタ・チェックボックス上にカーソルを置いたときに表示するツールヒント文字列を設定／取得します。
ｎは、チェックボックスのインデクスで、左から順に０～７(上段)，８～１５(下段)を指定します。

戻り値 ： TRUE - 成功
FALSE - 失敗

6.9.83. フィルタ・チェックボックス・ツールチップ表示条件の設定／取得(AjcG3d{Set|Get}ChkBoxTipShowAlways)

形　式 ： BOOL AjcG3dSetChkBoxTipShowAlways(HWND hwnd, UI n, BOOL fShowAlways); --- 設定
BOOL AjcG3dGetChkBoxTipShowAlways(HWND hwnd, UI n); ---------- 取得
引　数 ： hwnd - コントロールのウインドハンドル
n - チェックボックスのインデクス（0～15）
fShowAlways - ツールチップ表示条件（TRUE:非アクティブ時も表示, FALSE:非アクティブ時は非表示）

説　明 ： フィルタチェックボックス・ツールチップの表示条件（自プログラムが非アクティブ時の表示／非表示）を設定／取得します。
ｎは、チェックボックスのインデクスで、左から順に０～７(上段)，８～１５(下段)を指定します。

戻り値 ： 設定時：TRUE - 成功　　取得時：非アクティブ時のツールチップ表示条件
FALSE - 失敗

6.9.84. フィルタの設定／取得 (AjcG3d{Set/Get}Filter)

形　式 ： BOOL AjcG3dSetFilter(HWND hwnd, UI n, BOOL state);
BOOL AjcG3dGetFilter(HWND hwnd, UI n);

引　数 ： hwnd - コントロールのウインドハンドル
n - チェックボックスのインデクス
state - 設定するフィルタの状態(FALSE:非表示 / TRUE:表示)

説　明 ： フィルタ・チェックボックスの設定（当該項目の表示／非表示の設定）を行います。
ｎは、チェックボックスのインデクスで、左から順に０～７(上段)，８～１５(下段)を指定します。

戻り値 ： 設定時：TRUE - 成功　　取得時：フィルタの状態(FALSE:非表示 / TRUE:表示)
FALSE - 失敗

6.9.85. ドロップされたファイル名取得(AjcG3dGetDroppedFile)

形　式 ： BOOL AjcG3dGetDroppedFile (HWND hwnd, UT buf[MAX_PATH]);

引　数 ： hwnd　　- コントロールのウインドハンドル
buf　　- ファイルのパス名を格納するバッファのアドレス

説　明 ： ドラッグ＆ドロップされたファイルのパス名を取得します。
本関数は、通知メッセージ（AJC3DGN_DROPFILE）が通知された場合、ドロップされたファイルのパス名を取得するために実行します。
本ＡＰＩにより全てのドロップされたファイル名を取得済の場合は、FALSE を返します。

戻り値 ： TRUE - ファイルのパス名をバッファに格納した
FALSE - ドロップされたファイル名なし（ドロップしたファイル数を超えて実行された）

6.9.86. ドロップされたディレクトリ名取得(AjcG3dGetDroppedDir[Ex])

形　式 ： BOOL AjcG3dGetDroppedDir (HWND hwnd, UT buf[MAX_PATH]);
BOOL AjcG3dGetDroppedDirEx (HWND hwnd, UT buf[MAX_PATH], BOOL fTailIsDelimiter);

引　数 ： hwnd　　- コントロールのウインドハンドル
buf　　- ファイルのパス名を格納するバッファのアドレス
fTailIsDelimiter - TRUE ：ディレクトリパス名の末尾に「¥」を付加する
FALSE：ディレクトリパス名の末尾に「¥」を付加しない

説　明 ： ドラッグ＆ドロップされたディレクトリのパス名を取得します。
本関数は、通知メッセージ（AJC3DGN_DROPDIR）が通知された場合、ドロップされたディレクトリのパス名を取得するために実行します。
本ＡＰＩにより全てのドロップされたディレクトリ名を取得済の場合は、FALSE を返します。

戻り値 ： TRUE - ディレクトリのパス名をバッファに格納した
FALSE - ドロップされたディレクトリ名なし（ドロップしたディレクトリ数を超えて実行された）

6.9.87. タイトル文字列の設定 (AjcG3dSetTitleText)

形　式 ： BOOL AjcG3dSetTitleText (HWND hwnd, C_UTP pTitleText, COLORREF TextColor, COLORREF BackColor, HFONT hFont);

引　数 ： hwnd　　- コントロールのウインドハンドル
pTitleText　- タイトルテキストへのポインタ（NULL 指定時は、タイトル非表示）
TextColor　- テキスト描画色（－１指定時は前回設定値，デフォルト色＝白）
BackColor　- テキスト背景色（－１指定時は前回設定値，デフォルト色＝グレー）
hFont　　- タイトルテキストのフォントハンドル（NULL 指定時はデフォルトのフォント）

説　明 ： コントロールウインドの右上にタイトル文字列を表示します。

戻り値 ： TRUE - 成功
FALSE - 失敗

6.9.88. 画面表示の停止／再開 (AjcG3dPause)

形　式 ： BOOL AjcG3dPause (HWND hwnd, BOOL fPause);

引　数 ： hwnd　　- コントロールのウインドハンドル
fPause　- 表示停止／再開フラグ（TRUE:停止, FALSE:再開）

説　明 ： 表示を停止／再開します。
fPause=TRUE を指定すると表示が停止します。表示停止中でも、描画データの投与は有効です。
fPause=FALSE を指定すると、表示を再開します。この時、表示停止中に投与された描画データは一気に表示されます。
（全てのデータを表示&スクロールせずに、最終画面状態だけを表示します）

戻り値 ： TRUE　- 成功
FALSE - 失敗

6.9.89. 描画時間計測情報の許可／禁止 (AjcG3dEnableMesDraw)

形　式 ： BOOL AjcG3dEnableMesDraw (HWND hwnd, BOOL fEnable);

引　数 ： hwnd　　- コントロールのウインドハンドル
fEnable　- 許可／禁止フラグ（TRUE:許可, FALSE:禁止）

説　明 ： ポップアップメニュー上に「描画時間情報 表示／非表示」を表示するか否かを指定します。
fEnable=TRUE を指定するとポップアップメニュー上に「描画時間情報 表示／非表示」を表示します。
fEnable=FALSE を指定すると、ポップアップメニュー上に「描画時間情報 表示／非表示」を表示しません。

戻り値 ： TRUE　- 成功
FALSE - 失敗

6.9.90. イベント通知用コールバック関数の設定(AjcTG3dSetEvtCallBack)

形　式 ： BOOL AjcG3dSetEvtCallBack (HWND hwnd, UX cbp, VO (CALLBACK *cbNtcEvt)(HWND hCtrl, UX wParam, UX lParam, UX cbp));

引　数 ： hwnd　　- コントロールのウインドハンドル
cbp　　- コールバックパラメタ
cbNtcEvt　- イベント通知用コールバック関数（NULL:コールバック関数を無効化）

説　明 ： イベント通知用のコールバック関数を設定します。
通常、イベントは WM_COMMAND メッセージで通知されますが、コールバック関数を設定することにより、コールバック関数の呼び出しにより通知されるようになります。

戻り値 ： TRUE　- 成功
FALSE - 失敗

コールバック： コールバック関数の仕様は、以下のとおりです。

cbNtcEvt（イベント通知）

形　式 ： VO CALLBACK *cbNtcEvt*(HWND hCtrl, UX wParam, UX lParam, UX cbp)；

引　数 ： hCtrl　- コントロールのウインドハンドル
wPatram - コントロール識別ＩＤ, 通知コード（WM_COMMAND による通知と同じ）
lPatram - イベントパラメタ　　（WM_COMMAND による通知と同じ）
cbp　　- コールバックパラメタ

説　明 ： 各種イベントを通知します。（詳細は「コントロールの通知メッセージ」を参照）

戻り値 ： なし

6.10. 通知情報の取得ＡＰＩ

コントロールからの通知メッセージ（WM_COMMAND）に伴うパラメタを取得するＡＰＩ群です。
「AjcSetCmdWithHdl(TRUE);」を実行した場合、各通知メッセージ（WM_COMMAND）の lParam は通知内容に伴うパラメタは通知されず、lParam＝コントロールのウインドハンドルとなります。
この場合、通知内容に伴うパラメタは以下のＡＰＩで取得します。

#	関 数 名	内容	対応する通知
1	AjcG3dGetNtcRotTheta	視点角度（視点ベクトル）情報の取得	AJC3DGN_ROTTHETA　視点角度通知
2	AjcG3dGetNtcList	プロットリスト情報の取得	AJC3DGN_PLOTLIST　プロットリスト通知
3	AjcG3dGetNtcClrFact	クリアー内容の取得	AJC3DGN_CLEAR　データクリアー通知
4	AjcG3dGetNtcFiles	ドロップしたファイル数の取得	AJC3DGN_DROPFILE　ファイルドロップ通知
5	AjcG3dGetNtcDirs	ドロップしたディレクトリ数の取得	AJC3DGN_DROPDIR　ディレクトリドロップ通知
6	AjcG3dGetNtcRClk	右クリック通知情報の取得	AJC3DGN_RCLICK　右クリック通知

6.10.1. 視点角度情報の取得（AjcG3dGetNtcRotTheta）

形　式　：　PAJC3DVEC　AjcG3dGetNtcRotTheta (HHWND hwnd);

引　数　：　hwnd　　- コントロールのウインドハンドル

説　明　：　視点角度通知（AJC3DGN_ROTTHETA）時の、視点ベクトル情報を取得します。

戻り値　：　視点ベクトル情報へのポインタ

6.10.2. プロットリスト情報の取得（AjcG3dGetNtcList）

形　式　：　PAJC3DGPLOTLIST AjcG3dGetNtcList (HHWND hwnd);

引　数　：　hwnd　　- コントロールのウインドハンドル

説　明　：　プロットリスト通知（AJC3DGN_PLOTLIST）時の、プロットリスト情報を取得します。

戻り値　：　プロットリスト情報（AJC3DGPLOTLIST）へのポインタ

6.10.3. データクリアー情報の取得（AjcG3dGetNtcClrFact）

形　式　：　UI　AjcG3dGetNtcClrFact (HHWND hwnd);

引　数　：　hwnd　　- コントロールのウインドハンドル

説　明　：　データクリアー通知（AJC3DGN_CLEAR）時の、クリアー内容を取得します。

戻り値　：　クリアーされたデータ種別（以下の合成値）

- AJC3DG_CLEAR_PLOT(=0x01) ： プロットデータ
- AJC3DG_CLEAR_DATA(=0x02) ： 描画データ

6.10.4. ドロップしたファイル数の取得（AjcG3dGetNtcFiles）

形　式 ： UI　AjcG3dGetNtcFiles (HHWND hwnd);

引　数 ： hwnd　　- コントロールのウインドハンドル

説　明 ： ファイルドロップ通知（AJC3DGN_DROPFILE）時の、ドロップされたファイルの個数を取得します。

戻り値 ： ドロップされたファイルの個数

6.10.5. ドロップしたディレクトリ数の取得（AjcG3dGetNtcDirs）

形　式 ： UI　AjcG3dGetNtcDirs (HHWND hwnd);

引　数 ： hwnd　　- コントロールのウインドハンドル

説　明 ： ディレクトリドロップ通知（AJC3DGN_DROPDIR）時の、ドロップされたディレクトリの個数を取得します。

戻り値 ： ドロップされたディレクトリの個数

6.10.6. 右クリック情報の取得（AjcG3dGetNtcRClk）

形　式 ： PAJC3DGRCLK　AjcG3dGetNtcRClk (HHWND hwnd);

引　数 ： hwnd　　- コントロールのウインドハンドル

説　明 ： 右クリック通知（AJC3DGN_RCLICK）時の、右クリック情報を取得します。

戻り値 ： 右クリック情報（AJC3DGRCLK）へのポインタ

6.11. コントロールの通知メッセージ

３Ｄ／２Ｄグラフィック・コントロールからの通知メッセージは、WM_COMMAND メッセージにより通知されます。
WM_COMMAND メッセージの wParam には以下の情報が設定されます。

・LOWORD(wParam) - コントロールの識別ＩＤ
・HIWORD(wParam) - 通知メッセージコード

通知メッセージコードは、以下のとおりです。

#	通知メッセージコード	内容	備考
1	AJC3DGN_ROTTHETA	視点角度通知	
2	AJC3DGN_PLOTLIST	プロットリスト通知	本コントロールをダイアログ項目とする場合は、使用できません。
3	AJC3DGN_CLEAR	データクリアー通知	
4	AJC3DGN_DBLCLK	ダブルクリック通知	
5	AJC3DGN_DROPFILE	ファイルドロップ通知	
6	AJC3DGN_DROPDIR	ディレクトリドロップ通知	
7	AJC3DGN_RCLICK	右クリック通知	SHIFT/CTRL + 右クリック時

6.11.1. 視点変化通知（ AJC3DGN_ROTTHETA ）

説　明 ： 視点位置が変化したことを通知します。

パラメタ： wParam - コントロールの識別ＩＤと通知メッセージコード（AJC3DGN_ROTTHETA）
lParam - 視点角度データのアドレス(PAJC3DVEC) （MFC 使用時はコントロールのウインドハンドル）
メンバ(x, y, z)は、各軸回りの回転角度［度］を示します。

戻り値 ： なし

6.11.2. プロットリスト通知（ AJC3DGN_PLOTLIST ）

説　明 ： グラフィックウインド上で、プロット点を Shift キーを押しながら左クリックした場合、当該プロット点の情報が通知されます。
クリックした位置にプロット点が存在しない場合は、本イベントは発生しません。
クリックした位置に複数のプロット点が存在する場合は、各バッファデータにおいて、最大６４ヶまでを通知します。
（６４ヶを超える場合、実際のプロット点の個数は「max」メンバ変数に設定されます）

lParam は、以下の形式で、プロット情報のアドレスを示します。

```
#define AJC3DGMAX_PLOTLIST  64      // 最大プロット情報通知数

typedef struct {
    UI          id;             //  ＩＤ＃（バッファ種別, 0 ～ 15）
    UI          ix;             //  プロットデータ：バッファ先頭からの位置(0～),  描画データ：-1）
    AJC3DVEC    vec;            //  ３Ｄベクトル値
} AJC3DGPLITEM, *PAJC3DGPLITEM;
typedef const AJC3DGPLITEM *PCAJC3DGPLITEM;

typedef struct {
    UI      max;    //  実際のプロット情報の個数
    UI      num;    //  通知するプロット情報の個数（１～AJC3DGMAX_PLOTLIST）
    AJC3DGPLITEM    lst[AJC3DGMAX_PLOTLIST];
} AJC3DGPLOTLIST, *PAJC3DGPLOTLIST;
typedef const AJC3DGPLOTLIST *PCAJC3DGPLOTLIST;
```

パラメタ： wParam - コントロールの識別ＩＤと通知メッセージコード（AJC3DGN_PLOTLIST）
lParam - プロット点情報のアドレス(**PAJC3DGPLOTLIST**) （MFC 使用時はコントロールのウインドハンドル）

戻り値 ： なし

6.11.3. データクリアー通知（ AJC3DGN_CLEAR）

説　明　：　プロットデータ／描画データがクリアーされたことを通知します。

パラメタ：　wParam - コントロールの識別ＩＤと通知メッセージコード（AJC3DGN_CLEAR）
lParam - クリアーされたデータ種別（以下の合成値）（MFC 使用時はコントロールのウインドハンドル）
・AJC3DG_CLEAR_PLOT(=0x01) ： プロットデータ
・AJC3DG_CLEAR_DATA(=0x02) ：描画データ

戻り値　：　なし

6.11.4. ダブルクリック通知（AJC3DGN_DBLCLK）

説　明　：　ダブルクリックされたことを通知します。

パラメタ：　wParam - コントロールの識別ＩＤと通知メッセージコード（AJC3DGN_DBLCLK）
lParam - ダブルクリック位置（ｘ：下位ワード，ｙ：上位ワード）（MFC 使用時はコントロールのウインドハンドル）

戻り値　：　なし（ゼロを返してください）

6.11.5. ファイルドロップ通知（AJC3DGN_DROPFILE）

説　明　：　ドラッグ＆ドロップ操作で、ファイルがドロップされたことを通知します。
本通知では、ドロップされたファイルの個数だけを通知します。
AjcG3dGetDroppedFile() によりドロップされたファイルのパス名を取得できます。

パラメタ：　wParam - コントロールの識別ＩＤと通知メッセージコード（AJC3DGN_DROPFILE）
lParam - ドロップされたファイルの個数（MFC 使用時はコントロールのウインドハンドル）

戻り値　：　なし（ゼロを返してください）

6.11.6. ディレクトリドロップ通知（AJC3DGN_DROPDIR）

説　明　：　ドラッグ＆ドロップ操作で、ディレクトリがドロップされたことを通知します。
本通知では、ドロップされたディレクトリの個数だけを通知します。
AjcG3dGetDroppedDir[Ex]()で、ドロップされたディレクトリのパス名をり取得できます。

パラメタ：　wParam - コントロールの識別ＩＤと通知メッセージコード（AJC3DGN_DROPDIR）
lParam - ドロップされたディレクトリの個数（MFC 使用時はコントロールのウインドハンドル）

戻り値　：　なし（ゼロを返してください）

6.11.7. 右クリック通知（AJC３DGN_RCLICK）

説　明　：　右クリックしたことを通知します。
右クリック通知の可否については、AjcG3dSetNtcRClk()を参照してください。
lParam で以下の右クリック情報を通知します。

```
typedef struct {
    int         x;              //  カーソルｘ座標
    int         y;              //  カーソルｙ座標
    BOOL        fShift;         //  SHIFT キー押下フラグ
    BOOL        fCtrl;          //  CTRL キー押下フラグ
} AJC3DGRCLK, *PAJC3DGRCLK;
```

パラメタ：　wParam - コントロールの識別ＩＤと通知メッセージコード（AJC3DGN_RCLICK）
lParam - 右クリック情報へのポインタ（PAJC3DGRCLK）（MFC 使用時はコントロールのウインドハンドル）

戻り値　：　なし（ゼロを返してください）

6.12. サンプルプログラム

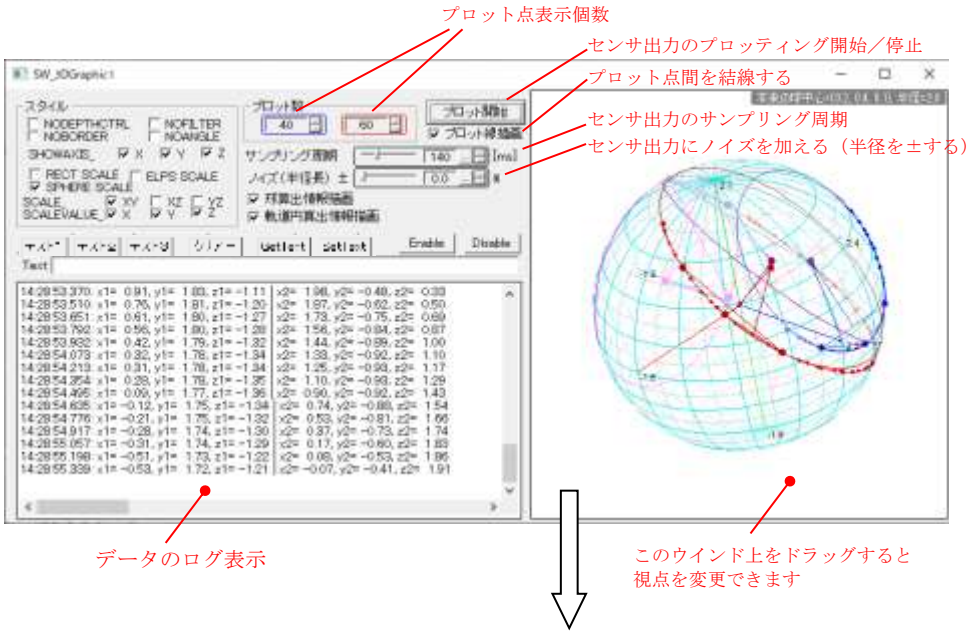
6.12.1. SW_3DGraphic1（３Dグラフィックの描画と軌道演算）

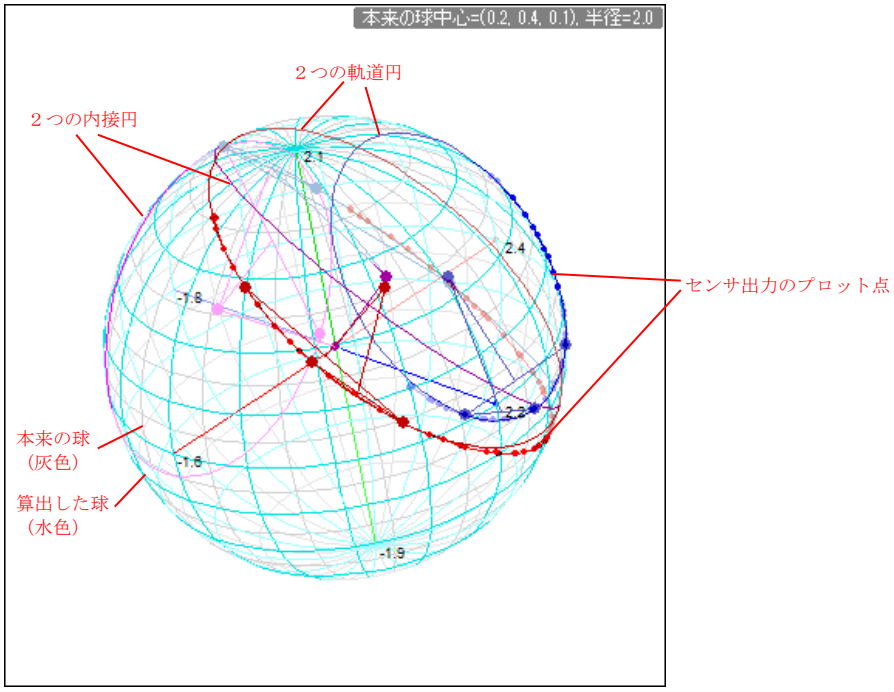
このサンプルプログラムでは、架空の２つ移動体に内蔵した３軸センサを想定して、このセンサの出力をプロット表示します。
３軸センサからの出力(x，y，z)は、球の表面を示す座標となります。
センサ出力は、本来の球(中心=(0.2，0.4，0.1)，半径=2.0)にノイズを加えて擬似的に生成します。
２つの移動体は、不規則に回転し、時々停止するものとします。(回転時 900 プロットと停止時 100 プロットを繰り返す)

このセンサ出力から、以下の情報を算出して描画します。

① 球の中心と半径を算出して球体を描画（４点から球を算出）
② 球の算出源となった情報を描画（２つの内接円）
③ プロット点の軌道を算出して、２つの軌道円を描画（３点から円を算出）
④ 軌道円の算出源となった情報を描画（三角形）

全て描画すると、以下のような図となります。

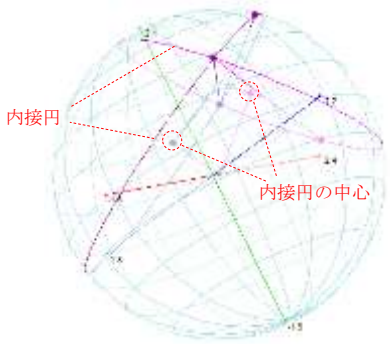




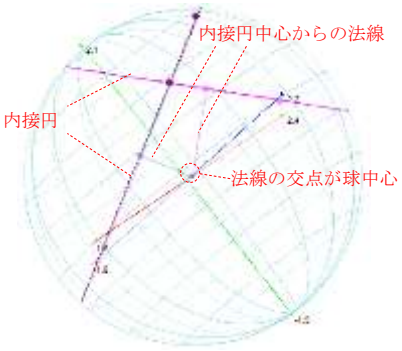
球の算出

「球算出情報描画」をチェックすると、選抜・収集した４点から、３点づつを使用して２つ三角形を構成し、内接円を算出します。
２つの内接円からの法線を引き、２つの法線の交点が球の中心となります。
以下は、余計な描画を消して、２つの内接円だけをクローズアップし、視点を変えた図です

内接円に関する情報だけをクローズアップ



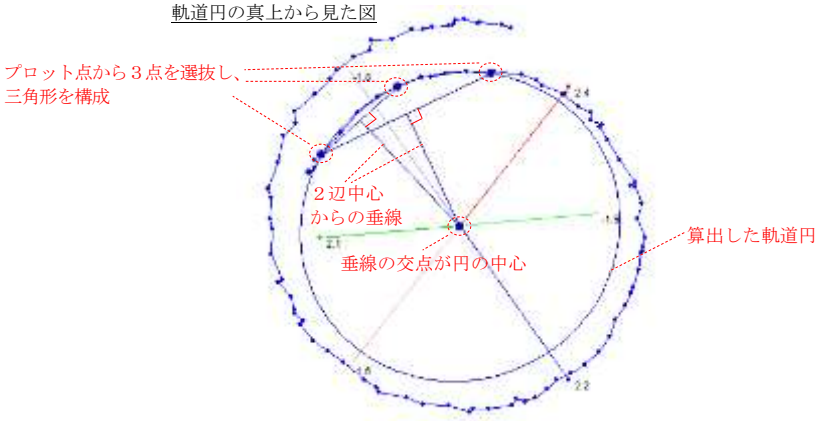
内接円の真横から見た図



軌道円の算出

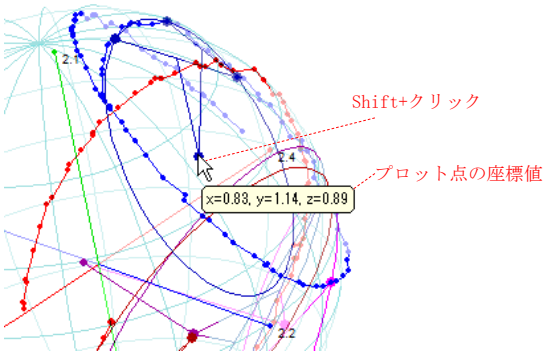
「起動円算出情報描画」をチェックすると、最新のプロット点から選抜・収集した３点から三角形を構成します。
三角形の２辺の中点から、同一平面上で垂線を引きます。
２つの垂線の交点が、軌道円の中心となります。
以下は、余計な描画を消して、（片方の）軌道円だけをクローズアップし、視点を変えた図です。

軌道円の真上から見た図



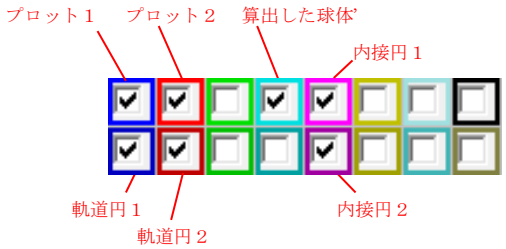
座標値の表示

Shift キーを押しながら、３Dグラフィックウインド上のプロット点を左クリックすると当該プロット点の座標値が表示されます。



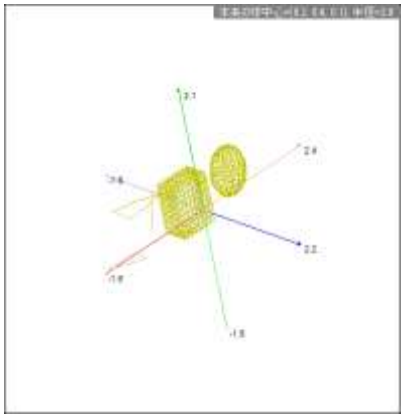
フィルタ

３Ｄグラフィックウインドの左上にカーソルを置くと、描画項目のフィルタ（チェックボックス）が表示されます。
各描画項目のチェックを外すと、その項目の描画を消すことができます。
「本来の球（グレー)」を消すには、「SPEHERE SCALE」のチェックを外します。

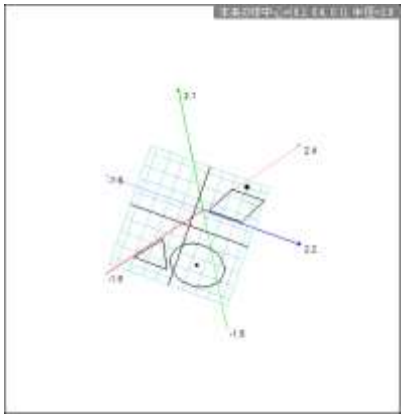


テスト１ボタン、テスト２ボタン、テスト３ボタンは、単に固定の図形を描画します。
テスト１ボタンは、３Ｄ空間上に図形を描画します。
テスト２ボタンは、３Ｄ空間上の平面に図形を描画します。
テスト３ボタンは、座標＝(1, 1, 1)の位置にテキストを描画します。

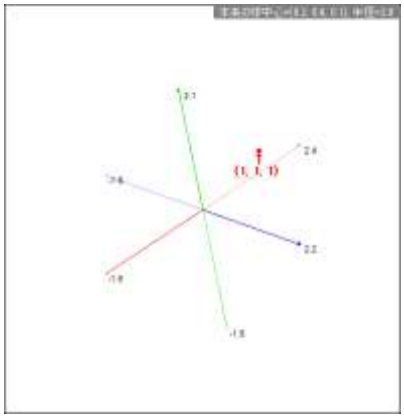
テスト１ボタンによる図形描画



テスト２ボタンによる図形描画



テスト３ボタンによる図形描画



```
 1 : //
 2 : //  SW_3DGraphic1.c
 3 : //
 4 :
 5 : #include    <AjrCstXX.h>
 6 : #include    <math.h>
 7 : #include    <tchar.h>
 8 : #include    "resource.h"
 9 :
10 : //-------------------------------------------------------------------------------------------------------------//
11 : //  タイマＩＤ                                                                                                 //
12 : //-------------------------------------------------------------------------------------------------------------//
13 : #define     TID_PLOT_PERIOD     1       //  プロットデータ生成周期
14 :
15 : //-------------------------------------------------------------------------------------------------------------//
16 : //  ワーク                                                                                                     //
17 : //-------------------------------------------------------------------------------------------------------------//
18 : static  HINSTANCE       hInst;                    //  ＤＬＬインスタンスハンドル
19 : static  HWND            hDlgMain;                 //  ダイアログボックスハンドル
20 : static  SIZE            szDlg;                    //  ダイアログの最小サイズ
21 : static  HWND            hWndG3d;                  //  ３Ｄグラフコントロールのウインドハンドル
22 : static  HWND            hWndVth;                  //  ログ表示コントロールのウインドハンドル
23 : static  BOOL            fPlot = FALSE;            //  プロット中を示すフラグ
24 : static  RECT            rcG3d, rcVth;             //  初期のウインド矩形
25 : static  BOOL            fTest1 = FALSE;
26 : static  BOOL            fTest2 = FALSE;
27 : static  BOOL            fTest3 = FALSE;
28 :
29 : //  球の中心と半径
30 : #define     CX      0.2
31 : #define     CY      0.4
32 : #define     CZ      0.1
```

```
#define     R       2.0

//  描画色ＩＤ
#define     ID_PLOT1        0       //  プロット点１
#define     ID_PLOT2        1       //  プロット点２
#define     ID_BALL         3       //  算出した球
#define     ID_INS1         4       //  球の内接円１
#define     ID_INS2         12      //  球の内接円２

#define     ID_TRACK1       8       //  軌道円１
#define     ID_TRACK2       9       //  軌道円２

#define     ID_2DGRID       6       //  ２Ｄ方眼
#define     ID_2DSHAPE      7       //  ２Ｄ図形
#define     ID_3DSHAPE      5       //  ２Ｄ図形
#define     ID_TXTPOINT     1       //  テキスト描画ポイント

//  プロット点演算パラメタ
static  AJCSPD_PARAM    prm[2] = {
//        vCent        radius  noise  xrot  yrot  pitch
        {{CX, CY, CZ}, R,      3.0,   0.5,  0.8,  9.0},
        {{CX, CY, CZ}, R,      3.0,   0.6,  0.7,  8.5},
};
//  プロット点演算インスタンスハンドル
HAJCSPD     hSpd[2] = {NULL};


//  プロットデータ収集バッファ情報
#define MAX_PLOT    4
typedef struct {
    UI          id1, id2, id3;                      //  描画色ＩＤ
    double      dist;                               //  プロット間の最低距離
    UI          num;                                //  プロット収集個数
    UI          cnt;                                //  プロットデータカウンタ
    AJC3DVEC    plt[MAX_PLOT];                      //  プロット点収集バッファ
    AJC3DVEC    cent;                               //  中心
    double      radius;                             //  半径
    AJC3DVEC    vec;                                //  法線
} PLOTINFO, *PPLOTINFO;

//  球算出用プロットバッファ
//                                  id1         id2         id3         dist        num
static  PLOTINFO    PltBall     =   { ID_INS1  , ID_INS2,   ID_BALL,   R * 0.6,    4   };

//  軌道円算出用プロットバッファ
//                                  id1         id2         id3         dist        num
static  PLOTINFO    PltCir[2]   =   {{ ID_TRACK1, 0      ,  0      ,  R * 0.4,    3   },
                                     { ID_TRACK2, 0      ,  0      ,  R * 0.4,    3   }};

//-----------------------------------------------------------------------------------------------------------------//
//  内部サブ関数                                                                                                   //
//-----------------------------------------------------------------------------------------------------------------//
AJC_DLGPROC_DEF(Main);

static  VO      SetControlSize(HWND hDlg);
static  C_UTP   CALLBACK cbGetTipText(HWND hCtrl, UTP pBuf, UI lBuf, UX cbp);
static  VO      BallCorrectAndShow(PCAJC3DVEC pVec, PPLOTINFO pBuf);
static  VO      CirCorrectAndShow (PCAJC3DVEC pVec, PPLOTINFO pBuf);
static  VO      DrawInscribedCircle(PCAJC3DCIRINFO pCif, int id);
static  VO      SetStyleToCheckBox(VO);
static  VO      SetCheckBoxToStyle(VO);

//=================================================================================================================//
//                                                                                                                 //
//  W i n M a i n                                                                                                  //
//                                                                                                                 //
//=================================================================================================================//
int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
{
    MSG     msg;
    RECT    rect;

    hInst = hInstance;

    //----- メイン・ダイアログオープン ------------------//
    hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_DLGMAIN), NULL, AJC_DLGPROC_NAME(Main));
    //----- ウインド最小サイズ退避 ----------------------//
    GetWindowRect(hDlgMain, &rect);
    szDlg.cx = rect.right - rect.left;
    szDlg.cy = rect.bottom - rect.top;
    //----- ダイアログ表示 ------------------------------//
    ShowWindow(hDlgMain, SW_SHOW);

    //----- メッセージループ ----------------------------//
    while (GetMessage(&msg, NULL, 0, 0)) {
        do {
            if (IsDialogMessage(hDlgMain, &msg)) break;
            TranslateMessage(&msg);
            DispatchMessage (&msg);
        } while (0);
```

```
123 :     }
124 :
125 :     return (int)msg.wParam ;
126 : }
127 : //=============================================================================================================//
128 : //                                                                                                             //
129 : //  ダイアログ・プロシージャ                                                                                   //
130 : //                                                                                                             //
131 : //=============================================================================================================//
132 : //----- ダイアログ初期化 ------------------------------------------------------------------------------------//
133 : AJC_DLGPROC(Main, WM_INITDIALOG     )
134 : {
135 :     hDlgMain = hDlg;
136 :     hWndG3d = GetDlgItem(hDlgMain, IDC_3DGRAPH);
137 :     hWndVth = GetDlgItem(hDlgMain, IDC_VTH_LOG);
138 :     //----- スタイルチェックボックスのツールチップ設定 -----------//
139 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_CHK_NODEPTHCTRL), TEXT("遠近表現（原点の手前側と向う側の色分け）禁止"));
140 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_CHK_NOFILTER   ), TEXT("データ表示／非表示用チェックボックス非表示"));
141 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_CHK_NOBORDER   ), TEXT("外枠非表示"));
142 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_CHK_NOANGLE    ), TEXT("ドラッグによる回転禁止"));
143 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_CHK_AXIS_X     ), TEXT("Ｘ軸表示"));
144 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_CHK_AXIS_Y     ), TEXT("Ｙ軸表示"));
145 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_CHK_AXIS_Z     ), TEXT("Ｚ軸表示"));
146 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_CHK_RECTSCALE  ), TEXT("方眼スケール表示（表示面は SCALE_XY/XZ/YZ で指定）"));
147 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_CHK_ELPSSCALE  ), TEXT("同心円スケール表示（表示面は SCALE_XY/XZ/YZ で指定）"));
148 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_CHK_SPHERESCALE), TEXT("球体スケール表示"));
149 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_CHK_SCALE_XY   ), TEXT("ＸＹ平面にスケール表示（スケールは RECTSCALE, ELPSSCALE で指定）"));
150 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_CHK_SCALE_XZ   ), TEXT("ＸＺ平面にスケール表示（スケールは RECTSCALE, ELPSSCALE で指定）"));
151 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_CHK_SCALE_YZ   ), TEXT("ＹＺ平面にスケール表示（スケールは RECTSCALE, ELPSSCALE で指定）"));
152 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_CHK_VAL_X      ), TEXT("Ｘ軸の値表示"));
153 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_CHK_VAL_Y      ), TEXT("Ｙ軸の値表示"));
154 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_CHK_VAL_Z      ), TEXT("Ｚ軸の値表示"));
155 :     //----- フィルタチェックボックスのツールチップ設定 -----------//
156 :     AjcG3dSetChkBoxTipText(hWndG3d, ID_PLOT1 , TEXT("プロット点１"));
157 :     AjcG3dSetChkBoxTipText(hWndG3d, ID_PLOT2 , TEXT("プロット点２"));
158 :     AjcG3dSetChkBoxTipText(hWndG3d, ID_BALL  , TEXT("算出した球" ));
159 :     AjcG3dSetChkBoxTipText(hWndG3d, ID_INS1  , TEXT("球の内接円１"));
160 :     AjcG3dSetChkBoxTipText(hWndG3d, ID_INS2  , TEXT("球の内接円２"));
161 :     AjcG3dSetChkBoxTipText(hWndG3d, ID_TRACK1, TEXT("軌道円１"   ));
162 :     AjcG3dSetChkBoxTipText(hWndG3d, ID_TRACK2, TEXT("軌道円２"   ));
163 :     //----- 初期のウインド矩形設定 ---------------------------//
164 :     GetWindowRect(hWndG3d, &rcG3d); MapWindowPoints(NULL, hDlg, (LPPOINT)&rcG3d, 2);
165 :     GetWindowRect(hWndVth, &rcVth); MapWindowPoints(NULL, hDlg, (LPPOINT)&rcVth, 2);
166 :     SetControlSize(hDlg);
167 :     //----- チップテキスト表示設定（状況依存で球情報表示）--------//
168 :     AjcTipTextAddEx(hWndG3d, TEXT(""), -1, -1, NULL, -1, -1, -1);
169 :     AjcTipTextSetCallBack(hWndG3d, 0, NULL, cbGetTipText);
170 :     //----- タイトル設定 -------------------------------------//
171 :     AjcG3dSetTitleText(hWndG3d, TEXT(" 本来の球中心=(0.2, 0.4, 0.1), 半径=2.0 "), -1, -1, NULL);
172 :     //----- ３Ｄグラフモード設定 -----------------------------//
173 :     AjcG3dInitV(hWndG3d, NULL, NULL, NULL, (AJC3DGS_AXIS | AJC3DGS_SCALEVALUE));
174 :     //----- プロットデータ演算初期化 -------------------------//
175 :     hSpd[0] = AjcSpdCreate(0);
176 :     hSpd[1] = AjcSpdCreate(0);
177 :     //----- ウインドスタイル値をチェックボックスに設定 -----------//
178 :     SetStyleToCheckBox();
179 :     //----- プロット数初期化 ---------------------------------//
180 :     AjcSetDlgItemUInt(hDlg, IDC_INP_PLOTNUM0, 100);
181 :     AjcSetDlgItemUInt(hDlg, IDC_INP_PLOTNUM1, 100);
182 :     //----- 設定値ロード -------------------------------------//
183 :     AjcDlgItemSetPermAtt(hDlg, IDC_TXT_3DGRAPH, AJCCTL_PSEL_EXCLUDE); // GetText は除外
184 :     AjcLoadAllControlSettings(hDlg, TEXT("Settings"), AJCOPT2(AJCCTL_SELACT_, CHKEXCLUDE, NTCINP));
185 :     //  チェックボックスからスタイルを設定
186 :     SetCheckBoxToStyle();
187 :     //  プロット線描画チェックボックス反映
188 :     SendMessage(hDlg, WM_COMMAND, MAKELONG(IDC_CHK_PLOTLINE, BN_CLICKED), (WPARAM)hDlg);
189 :     //----- グラフレンジ設定 ---------------------------------//
190 :     AjcG3dSetCenter(hWndG3d, CX, CY, CZ);
191 :     AjcG3dSetWidth (hWndG3d, R, R, R);
192 :     //----- 視点を３Ｄイメージのアングルに設定 ---------------//
193 :     AjcG3dSetAngle3D(hWndG3d);
194 :
195 :     return TRUE;
196 : }
197 : //----- ウインド破棄 ----------------------------------------------------------------------------------------//
198 : AJC_DLGPROC(Main, WM_DESTROY        )
199 : {
200 :     //----- 設定値セーブ -------------------------------------//
201 :     AjcSaveAllControlSettings(hDlg);
202 :     AjcDelAllCtrlPermAtt(hDlg); // リソース解放
203 :     //----- プロットデータ演算終了 ---------------------------//
204 :     AjcSpdDelete(hSpd[0]);
205 :     AjcSpdDelete(hSpd[1]);
206 :     //----- プログラム終了 -----------------------------------//
207 :     PostQuitMessage(0);
208 :     return TRUE;
209 : }
210 : //----- サイズ変更中 ----------------------------------------------------------------------------------------//
211 : AJC_DLGPROC(Main, WM_SIZING         )
212 : {
```

```
213 :     LPRECT  pRect = (LPRECT)lParam;
214 :     SIZE    sz;
215 :
216 :     sz.cx = pRect->right - pRect->left;
217 :     sz.cy = pRect->bottom - pRect->top;
218 :
219 :     if (sz.cx < szDlg.cx) pRect->right  = pRect->left + szDlg.cx;
220 :     if (sz.cy < szDlg.cy) pRect->bottom = pRect->top  + szDlg.cy;
221 :
222 :     return TRUE;
223 : }
224 : //----- サイズ変更 ----------------------------------------------------------------------------------------//
225 : AJC_DLGPROC(Main, WM_SIZE           )
226 : {
227 :     SetControlSize(hDlg);
228 :
229 :     return TRUE;
230 : }
231 : //- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - //
232 : static  VO      SetControlSize(HWND hDlg)
233 : {
234 :     RECT    r;
235 :     int     w, h;
236 :
237 :     GetClientRect(hDlg, &r);
238 :     w = r.right - r.left;
239 :     h = r.bottom - r.top;
240 :     MoveWindow(hWndG3d, rcG3d.left, 0        , w - rcG3d.left - 4      , h             - 4, FALSE);
241 :     MoveWindow(hWndVth, rcVth.left, rcVth.top, rcVth.right - rcVth.left, h - rcVth.top - 4, FALSE);
242 :     InvalidateRect(hDlg, NULL, TRUE);
243 : }
244 : //----- COMMAND -------------------------------------------------------------------------------------------//
245 : AJC_DLGPROC(Main, WM_COMMAND        )
246 : {
247 :     BOOL        rc = FALSE;
248 :     int         cmd = LOWORD(wParam);
249 :     UI          ix;
250 :     AJC3DGPROP  prop;
251 :
252 :     //----- スタイル設定チェックボックス操作 ------------------------------------------------//
253 :     if (cmd >= IDC_CHK_RECTSCALE && cmd <= IDC_CHK_AXIS_Z && HIWORD(wParam) == BN_CLICKED) {
254 :         //  チェックボックスの内容をスタイルに反映
255 :         SetCheckBoxToStyle();
256 :         rc = TRUE;
257 :     }
258 :     //----- プロット数設定操作 ----------------------------------------------------------------//
259 :     if (cmd >= IDC_INP_PLOTNUM0 && cmd <= IDC_INP_PLOTNUM1 && HIWORD(wParam) == AJCIVN_INTVALUE) {
260 :         ix = cmd - IDC_INP_PLOTNUM0;
261 :         AjcG3dGetProp(hWndG3d, &prop);
262 :         prop.Item[ix].MaxPlot = (UI)lParam;
263 :         AjcG3dSetProp(hWndG3d, &prop);
264 :         rc = TRUE;
265 :     }
266 :
267 :     return rc;
268 : }
269 : //----- WM_TIMER ------------------------------------------------------------------------------------------//
270 : AJC_DLGPROC(Main, WM_TIMER      )
271 : {
272 :     UI          id;
273 :     AJC3DVEC    v[2];
274 :
275 :     switch (wParam) {
276 :         case TID_PLOT_PERIOD:                       //  ●プロットデータ生成周期
277 :             for (id = 0; id < 2; id++) {
278 :                 //  ランダムなプロットデータ生成
279 :                 AjcSpdCalcV(hSpd[id], &v[id]);
280 :                 //  プロットデータ投与
281 :                 AjcG3dPutPlotDataV(hWndG3d, id, &v[id]);
282 :                 //  球データ収集と算出した球の表示
283 :                 BallCorrectAndShow(&v[id], &PltBall);
284 :                 //  軌道円データ収集と算出した軌道円の表示
285 :                 CirCorrectAndShow (&v[id], &PltCir[id]);
286 :             }
287 :             //  ログ表示
288 :             AjcVthTimeStamp(hWndVth);
289 :             AjcVthPrintF(hWndVth, TEXT(": x1=%6.2f, y1=%6.2f, z1=%6.2f | x2=%6.2f, y2=%6.2f, z2=%6.2f¥n"),
290 :                                                             v[0].x, v[0].y, v[0].z, v[1].x, v[1].y, v[1].z);
291 :             break;
292 :     }
293 :     return TRUE;
294 : }
295 : //----- プロットライン・チェックボックス ------------------------------------------------------------------//
296 : AJC_DLGPROC(Main, IDC_CHK_PLOTLINE)
297 : {
298 :     AJC3DGPROP  prop;
299 :
300 :     if (HIWORD(wParam) == BN_CLICKED) {
301 :         AjcG3dGetProp(hWndG3d, &prop);
302 :         prop.fPlotLine = AjcGetDlgItemChk(hDlg, IDC_CHK_PLOTLINE);
```

```
303 :         AjcG3dSetProp(hWndG3d, &prop);
304 :     }
305 :     return TRUE;
306 : }
307 : //----- サンプリング周期入力 ----------------------------------------------------------------------------------//
308 : AJC_DLGPROC(Main, IDC_INP_PERIOD)
309 : {
310 :     if (HIWORD(wParam) == AJCIVN_INTVALUE) {
311 :         if (fPlot) {
312 :             SetTimer(hDlgMain, TID_PLOT_PERIOD, AjcGetDlgItemUInt(hDlg, IDC_INP_PERIOD), NULL);
313 :         }
314 :     }
315 :     return TRUE;
316 : }
317 : //----- ノイズ値入力 (AjcLoadAllControlSettings()で、AJCCTL_SELACT_NTCINP 指定の為、初期化時にも実行される) ----//
318 : AJC_DLGPROC(Main, IDC_INP_NOISE )
319 : {
320 :     if (HIWORD(wParam) == AJCIVN_REALVALUE) {
321 :         prm[0].noise = prm[1].noise = *((double *)lParam);
322 :         AjcSpdSetParam(hSpd[0], &prm[0]);
323 :         AjcSpdSetParam(hSpd[1], &prm[1]);
324 :     }
325 :     return TRUE;
326 : }
327 : //----- プロット開始／停止ボタン ------------------------------------------------------------------------------//
328 : AJC_DLGPROC(Main, IDC_CMD_START_STOP)
329 : {
330 :     if (fPlot) {
331 :         fPlot = FALSE;
332 :         KillTimer(hDlg, TID_PLOT_PERIOD);
333 :         AjcSetDlgItemStr(hDlg, IDC_CMD_START_STOP, TEXT("プロット開始"));
334 :     }
335 :     else {
336 :         fPlot = TRUE;
337 :         SetTimer(hDlgMain, TID_PLOT_PERIOD, AjcGetDlgItemUInt(hDlg, IDC_INP_PERIOD), NULL);
338 :         AjcSetDlgItemStr(hDlg, IDC_CMD_START_STOP, TEXT("プロット停止"));
339 :     }
340 :     return TRUE;
341 : }
342 : //----- テスト１ボタン ----------------------------------------------------------------------------------------//
343 : AJC_DLGPROC(Main, IDC_CMD_TEST1     )
344 : {
345 :
346 :     if (!fTest1) {
347 :         AjcG3dCube     (hWndG3d, ID_3DSHAPE, +0.1, +0.2, +0.3,  0.2, 0.3, 0.4, 8);
348 :         AjcG3dSphere   (hWndG3d, ID_3DSHAPE, +0.7, +0.6, +0.8,  0.2, 0.3, 0.4, 12, 8);
349 :         AjcG3dMoveTo   (hWndG3d, ID_3DSHAPE, -0.6,  0.2,  0.3);
350 :         AjcG3dLineTo   (hWndG3d, ID_3DSHAPE, -0.3, -0.2, -0.1);
351 :         AjcG3dTriangle (hWndG3d, ID_3DSHAPE, -0.8, -0.7, -0.6,  -0.4, -0.3, -0.5,  -0.4, -0.5, -0.3);
352 :         AjcG3dSquare   (hWndG3d, ID_3DSHAPE,  0.0, -0.1,  0.6,  -0.2, -0.5,  0.6,  -0.7, -0.5,  0.3, -0.6,-0.3, 0.1);
353 :         fTest1 = TRUE;
354 :     }
355 :     else {
356 :         AjcG3dClearShape(hWndG3d, ID_3DSHAPE);
357 :         fTest1 = FALSE;
358 :     }
359 :     return TRUE;
360 : }
361 : //----- テスト２ボタン ----------------------------------------------------------------------------------------//
362 : AJC_DLGPROC(Main, IDC_CMD_TEST2     )
363 : {
364 :     UI      idAll = -1;
365 :     double  x, y;
366 :
367 :     if (!fTest2) {
368 :         AjcG3dDefPlane(hWndG3d, idAll, 0.2, 0.1, 0.0,   0.6738, -0.5510, 0.4924,  0.1, 0.0, 0.0);
369 :         AjcG2dRectangle(hWndG3d, ID_2DGRID, -0.9, -0.9,  +0.9, +0.9);
370 :         for (x = -0.8; x < -0.01; x += 0.2) AjcG2dLine(hWndG3d, ID_2DGRID, x, -0.9, x, +0.9);
371 :         for (x =  0.2; x <  0.89; x += 0.2) AjcG2dLine(hWndG3d, ID_2DGRID, x, -0.9, x, +0.9);
372 :         for (y = -0.8; y < -0.01; y += 0.2) AjcG2dLine(hWndG3d, ID_2DGRID, -0.9, y, +0.9, y);
373 :         for (y =  0.2; y <  0.89; y += 0.2) AjcG2dLine(hWndG3d, ID_2DGRID, -0.9, y, +0.9, y);
374 :         AjcG2dLine    (hWndG3d, ID_2DSHAPE, -0.9,  0.0, +0.9,   0.0);
375 :         AjcG2dLine    (hWndG3d, ID_2DSHAPE, 0.0, -0.9,   0.0, +0.9);
376 :         AjcG2dTriangle(hWndG3d, ID_2DSHAPE, -0.6, -0.7, -0.3, -0.3,  -0.1, -0.7);
377 :         AjcG2dSquare  (hWndG3d, ID_2DSHAPE, 0.2,  0.3,  0.4, 0.7,   0.9, 0.7,  0.7, 0.3);
378 :         AjcG2dPixel   (hWndG3d, ID_2DSHAPE, 0.3, -0.5,  2);
379 :         AjcG2dEllipse (hWndG3d, ID_2DSHAPE, 0.3, -0.5,  0.4, 0.3);
380 :         AjcG2dPixel   (hWndG3d, ID_2DSHAPE, 0.6, 0.8, 3);
381 :         fTest2 = TRUE;
382 :     }
383 :     else {
384 :         AjcG3dClearShape(hWndG3d, ID_2DGRID);
385 :         AjcG3dClearShape(hWndG3d, ID_2DSHAPE);
386 :         fTest2 = FALSE;
387 :     }
388 :     return TRUE;
389 : }
390 : //----- テスト３ボタン ----------------------------------------------------------------------------------------//
391 : AJC_DLGPROC(Main, IDC_CMD_TEST3     )
392 : {
```

```
393 :     AJC3DVEC    v = {1, 1, 1};
394 :
395 :     if (!fTest3) {
396 :         AjcG3dPixelV  (hWndG3d, ID_TXTPOINT, &v, 3);
397 :         AjcG3dTextOutV(hWndG3d, AJCG3DTXOMD_BELLOW_CENTER, &v, TEXT("¥x1B[1;31m    ↑    ¥n(1, 1, 1)"));
398 :         fTest3 = TRUE;
399 :     }
400 :     else {
401 :         AjcG3dClearShape(hWndG3d, ID_TXTPOINT);
402 :         AjcG3dClearAllText(hWndG3d);
403 :         fTest3 = FALSE;
404 :     }
405 :     return TRUE;
406 : }
407 : //----- クリアーボタン ------------------------------------------------------------------------------------------//
408 : AJC_DLGPROC(Main, IDC_CMD_CLEAR     )
409 : {
410 :     AjcG3dClear(hWndG3d);
411 :     fTest1 = fTest2 = fTest3 = FALSE;
412 :     return TRUE;
413 : }
414 : //----- GetText ボタン ------------------------------------------------------------------------------------------//
415 : AJC_DLGPROC(Main, IDC_CMD_GETTEXT   )
416 : {
417 :     UT      txt[1024];
418 :
419 :     GetWindowText(hWndG3d, txt, 1024);
420 :     AjcSetDlgItemStr(hDlg, IDC_TXT_3DGRAPH, txt);
421 :     return TRUE;
422 : }
423 : //----- SetText ボタン ------------------------------------------------------------------------------------------//
424 : AJC_DLGPROC(Main, IDC_CMD_SETTEXT   )
425 : {
426 :     UT      txt[1024];
427 :
428 :     AjcGetDlgItemStr(hDlg, IDC_TXT_3DGRAPH, txt, 1024);
429 :     SetWindowText(hWndG3d, txt);
430 :     return TRUE;
431 : }
432 : //----- Enable ボタン -------------------------------------------------------------------------------------------//
433 : AJC_DLGPROC(Main, IDC_CMD_ENABLE    )
434 : {
435 :     EnableWindow(hWndG3d, TRUE);
436 :     return TRUE;
437 : }
438 : //----- Disable ボタン ------------------------------------------------------------------------------------------//
439 : AJC_DLGPROC(Main, IDC_CMD_DISABLE   )
440 : {
441 :     EnableWindow(hWndG3d, FALSE);
442 :     return TRUE;
443 : }
444 : //----- ３Ｄグラフコントロール ----------------------------------------------------------------------------------//
445 : AJC_DLGPROC(Main, IDC_3DGRAPH       )
446 : {
447 :     POINT    pt;
448 :
449 :     switch (HIWORD(wParam)) {
450 :         case AJC3DGN_RCLICK:                        //  ●右クリック通知
451 :         {   PAJC3DGRCLK p = (PAJC3DGRCLK)lParam;
452 :             UT          txt[64] = {0};
453 :             AjcSnPrintF(txt, 64, TEXT("%s%s 右クリック発生(x = %d, y = %d)"), p->fShift ? TEXT("Shift+") : TEXT(""),
454 :                                                                               p->fCtrl  ? TEXT("Ctrl+" ) : TEXT(""),
455 :                                                                               p->x, p->y);
456 :             GetCursorPos(&pt);
457 :             AjcTipTextShow(pt.x, pt.y + 20, txt, 3000, NULL);
458 :             break;
459 :         }
460 :         case AJC3DGN_DROPDIR:                       //  ●ディレクトリドロップ
461 :         {   UT      path[MAX_PATH];
462 :             UT      txt[4096];
463 :             MAjcStrCpy(txt, AJCTSIZE(txt), TEXT("-- Dir dropped --¥n"));
464 :             while (AjcG3dGetDroppedDir(hWndG3d, path)) {
465 :                 MAjcStrCat(txt, AJCTSIZE(txt), path);
466 :                 MAjcStrCat(txt, AJCTSIZE(txt), TEXT("¥n"));
467 :             }
468 :             GetCursorPos(&pt);
469 :             AjcTipTextShow(pt.x, pt.y + 20, txt, 3000, NULL);
470 :             break;
471 :         }
472 :         case AJC3DGN_DROPFILE:                      //  ●ファイルドロップ
473 :         {   UT      path[MAX_PATH];
474 :             UT      txt[4096];
475 :             MAjcStrCpy(txt, AJCTSIZE(txt), TEXT("-- File dropped --¥n"));
476 :             while (AjcG3dGetDroppedFile(hWndG3d, path)) {
477 :                 MAjcStrCat(txt, AJCTSIZE(txt), path);
478 :                 MAjcStrCat(txt, AJCTSIZE(txt), TEXT("¥n"));
479 :             }
480 :             GetCursorPos(&pt);
481 :             AjcTipTextShow(pt.x, pt.y + 20, txt, 3000, NULL);
482 :             break;
```

```
483 :         }
484 :         case AJC3DGN_PLOTLIST:                          //  ●プロットリスト通知
485 :         {   PCAJC3DGPLOTLIST p = (PCAJC3DGPLOTLIST)lParam;
486 :             static UT txt[128];
487 :             AjcSnPrintF(txt, AJCTSIZE(txt), TEXT("x=%.2f, y=%.2f, z=%.2f"), p->lst[0].vec.x, p->lst[0].vec.y, p->lst[0].vec.z);
488 :             GetCursorPos(&pt);
489 :             AjcTipTextShow(pt.x, pt.y + 20, txt, 3000, NULL);
490 :             break;
491 :         }
492 :     }
493 :     return TRUE;
494 : }
495 : //----- 「Cancel」ボタン ----------------------------------------------------------------------------------------------//
496 : AJC_DLGPROC(Main, IDCANCEL          )
497 : {
498 :     DestroyWindow(hDlg);
499 :     return TRUE;
500 : }
501 : //-------------------------------------------------------------------------------------------------------------------//
502 : AJC_DLGMAP_DEF(Main)
503 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG     )
504 :     AJC_DLGMAP_MSG(Main, WM_SIZING         )
505 :     AJC_DLGMAP_MSG(Main, WM_SIZE           )
506 :     AJC_DLGMAP_MSG(Main, WM_DESTROY        )
507 :     AJC_DLGMAP_MSG(Main, WM_COMMAND        )
508 :     AJC_DLGMAP_MSG(Main, WM_TIMER          )
509 :
510 :     AJC_DLGMAP_CMD(Main, IDC_CHK_PLOTLINE  )
511 :     AJC_DLGMAP_CMD(Main, IDC_INP_PERIOD    )
512 :     AJC_DLGMAP_CMD(Main, IDC_INP_NOISE     )
513 :     AJC_DLGMAP_CMD(Main, IDC_CMD_START_STOP)
514 :     AJC_DLGMAP_CMD(Main, IDC_CMD_TEST1     )
515 :     AJC_DLGMAP_CMD(Main, IDC_CMD_TEST2     )
516 :     AJC_DLGMAP_CMD(Main, IDC_CMD_TEST3     )
517 :     AJC_DLGMAP_CMD(Main, IDC_CMD_CLEAR     )
518 :     AJC_DLGMAP_CMD(Main, IDC_CMD_GETTEXT   )
519 :     AJC_DLGMAP_CMD(Main, IDC_CMD_SETTEXT   )
520 :     AJC_DLGMAP_CMD(Main, IDC_CMD_ENABLE    )
521 :     AJC_DLGMAP_CMD(Main, IDC_CMD_DISABLE   )
522 :     AJC_DLGMAP_CMD(Main, IDC_3DGRAPH       )
523 :     AJC_DLGMAP_CMD(Main, IDCANCEL          )
524 : AJC_DLGMAP_END
525 :
526 : //-------------------------------------------------------------------------------------------------------------------//
527 : //  チップテキストコールバック                                                                                       //
528 : //-------------------------------------------------------------------------------------------------------------------//
529 : static  C_UTP   CALLBACK cbGetTipText(HWND hCtrl, UTP pBuf, UI lBuf, UX cbp)
530 : {
531 :     //  チップテキスト（球情報）設定
532 :     AjcSnPrintF(pBuf, lBuf,
533 :                     TEXT("球算出情報：¥n")
534 :                     TEXT("  中心 = (%6.2f, %6.2f, %6.2f )¥n")
535 :                     TEXT("  半径 =  %.2f"), PltBall.cent.x, PltBall.cent.y, PltBall.cent.z, PltBall.radius);
536 :     return pBuf;
537 : }
538 : //-------------------------------------------------------------------------------------------------------------------//
539 : //  球算出用プロット点の収集と描画                                                                                   //
540 : //-------------------------------------------------------------------------------------------------------------------//
541 : static  VO      BallCorrectAndShow(PCAJC3DVEC pVec, PPLOTINFO pBuf)
542 : {
543 :     UI      i;
544 :
545 :     //  プロット間の最低距離チェック
546 :     for (i = 0; i < pBuf->cnt; i++) {
547 :         if (AjcV3dDistP2P(pVec, &pBuf->plt[i]) < pBuf->dist) {
548 :             break;
549 :         }
550 :     }
551 :     //  プロット点の収集
552 :     if (i >= pBuf->cnt) {
553 :         memcpy(&pBuf->plt[i], pVec, sizeof(AJC3DVEC));
554 :         pBuf->cnt++;
555 :     }
556 :     //  球の算出と表示
557 :     if (pBuf->cnt >= pBuf->num) {
558 :         AJC3DSPHINFO    sph;
559 :         if ((pBuf->radius = AjcV3dCalcSphereVEx(&pBuf->plt[0], &pBuf->plt[1], &pBuf->plt[2], &pBuf->plt[3],
560 :                                                                                 &pBuf->cent, &sph)) != -1) {
561 :             //  ２つの平面円の角度が 30 度以上ならば球描画
562 :             double t = AjcV3dTheta(&sph.cif1.lvc.v, &sph.cif2.lvc.v);
563 :             if (t > 30 && t < 180 - 30) {
564 :                 //  前回の球表示をクリアー
565 :                 AjcG3dClearShape(hWndG3d, pBuf->id3);
566 :                 //  球表示
567 :                 AjcG3dSphereV(hWndG3d, pBuf->id3, &pBuf->cent, pBuf->radius, pBuf->radius, pBuf->radius, 8, 8);
568 :                 //  前回の内接円表示をクリアー
569 :                 AjcG3dClearShape(hWndG3d, pBuf->id1);
570 :                 AjcG3dClearShape(hWndG3d, pBuf->id2);
571 :                 //  ２つの内接円描画
572 :                 if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_BALLINFO)) {
```

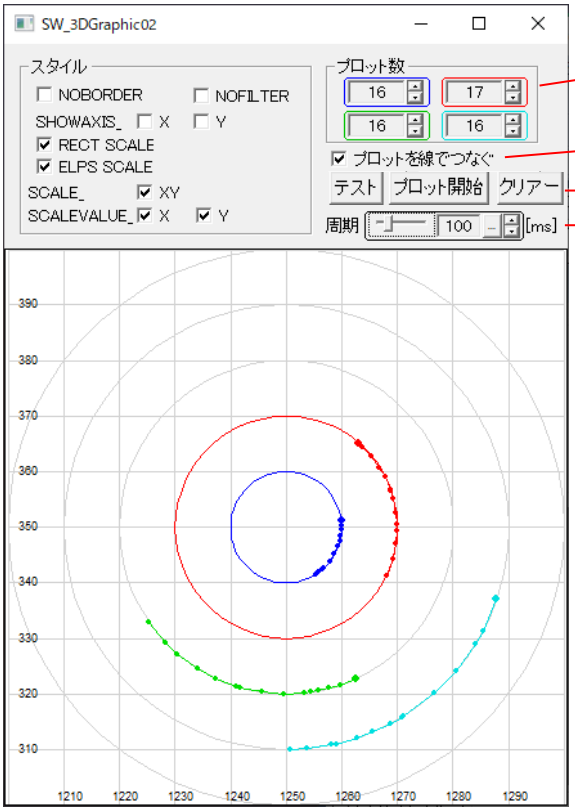
```
573 :                     //  球中心点描画
574 :                     AjcG3dPixelV(hWndG3d, pBuf->id1, &pBuf->cent, 3);
575 :                     AjcG3dPixelV(hWndG3d, pBuf->id2, &pBuf->cent, 3);
576 :                     //  球に内接する２つの円の情報を描画
577 :                     DrawInscribedCircle(&sph.cif1, pBuf->id1);
578 :                     DrawInscribedCircle(&sph.cif2, pBuf->id2);
579 :                     //  内接円中心から球中心への垂線
580 :                     AjcG3dLineV(hWndG3d, pBuf->id1, &sph.cif1.lvc.p, &pBuf->cent);
581 :                     AjcG3dLineV(hWndG3d, pBuf->id2, &sph.cif2.lvc.p, &pBuf->cent);
582 :                 }
583 :             }
584 :         }
585 :         //  球の情報クリアー
586 :         pBuf->cnt = 0;
587 :         memset(pBuf->plt, 0, sizeof pBuf->plt);
588 :     }
589 : }
590 : //-------------------------------------------------------------------------------------------------------------------//
591 : //  軌道円算出用プロット点の収集と描画                                                                               //
592 : //-------------------------------------------------------------------------------------------------------------------//
593 : static  VO      CirCorrectAndShow(PCAJC3DVEC pVec, PPLOTINFO pBuf)
594 : {
595 :     UI      i;
596 :
597 :     //  プロット間の最低距離チェック
598 :     for (i = 0; i < pBuf->cnt; i++) {
599 :         if (AjcV3dDistP2P(pVec, &pBuf->plt[i]) < pBuf->dist) {
600 :             break;
601 :         }
602 :     }
603 :     //  プロット点の収集
604 :     if (i >= pBuf->cnt) {
605 :         memcpy(&pBuf->plt[i], pVec, sizeof(AJC3DVEC));
606 :         pBuf->cnt++;
607 :     }
608 :
609 :     //  軌道円の算出と表示
610 :     if (pBuf->cnt >= pBuf->num) {
611 :         AJC3DCIRINFO    cif;
612 :         if ((pBuf->radius = AjcV3dCalcCircleVEx(&pBuf->plt[0], &pBuf->plt[1], &pBuf->plt[2],
613 :                                                                         &pBuf->cent, &pBuf->vec , &cif)) != -1) {
614 :             if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_CIRINFO)) {
615 :                 //  前の軌道円描画クリアー
616 :                 AjcG3dClearShape(hWndG3d, pBuf->id1);
617 :                 //  ３点描画
618 :                 AjcG3dPixelV(hWndG3d, pBuf->id1, &pBuf->plt[0], 4);
619 :                 AjcG3dPixelV(hWndG3d, pBuf->id1, &pBuf->plt[1], 4);
620 :                 AjcG3dPixelV(hWndG3d, pBuf->id1, &pBuf->plt[2], 4);
621 :                 //  内接円描画
622 :                 DrawInscribedCircle(&cif, pBuf->id1);
623 :             }
624 :         }
625 :         //  軌道円の情報クリアー
626 :         pBuf->cnt = 0;
627 :         memset(pBuf->plt, 0, sizeof pBuf->plt);
628 :     }
629 : }
630 : //-------------------------------------------------------------------------------------------------------------------//
631 : //  球算出の内接円／軌道円の描画                                                                                     //
632 : //-------------------------------------------------------------------------------------------------------------------//
633 : static  VO      DrawInscribedCircle(PCAJC3DCIRINFO pCif, int id)
634 : {
635 :     //  円に内接する２つの直線と端点
636 :     AjcG3dLineV (hWndG3d, id, &pCif->lt1.p1, &pCif->lt1.p2);
637 :     AjcG3dLineV (hWndG3d, id, &pCif->lt2.p1, &pCif->lt2.p2);
638 :     AjcG3dPixelV(hWndG3d, id, &pCif->lt1.p1, 4);
639 :     AjcG3dPixelV(hWndG3d, id, &pCif->lt1.p2, 4);
640 :     AjcG3dPixelV(hWndG3d, id, &pCif->lt2.p1, 4);
641 :     AjcG3dPixelV(hWndG3d, id, &pCif->lt2.p2, 4);
642 :     //  内接線中点からの垂線
643 :     AjcG3dLineV (hWndG3d, id, &pCif->lc1.p1, &pCif->lc1.p2);
644 :     AjcG3dLineV (hWndG3d, id, &pCif->lc2.p1, &pCif->lc2.p2);
645 :     //  内接円と内接円の中心点
646 :     AjcG3dDefPlaneV(hWndG3d, id, &pCif->lvc, NULL);
647 :     AjcG2dEllipse  (hWndG3d, id, 0, 0, pCif->cr, pCif->cr);
648 :     AjcG2dPixel    (hWndG3d, id, 0, 0, 4);
649 : }
650 :
651 : //-------------------------------------------------------------------------------------------------------------------//
652 : //  ウインドスタイル値をチェックボックスに設定                                                                       //
653 : //-------------------------------------------------------------------------------------------------------------------//
654 : static  VO  SetStyleToCheckBox(VO)
655 : {
656 :     UI      sty;
657 :
658 :     sty = (UI)MAjcGetWindowLong(hWndG3d, GWL_STYLE);
659 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_NODEPTHCTRL , (sty & AJC3DGS_NODEPTHCTRL  ) != 0);
660 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_NOBORDER    , (sty & AJC3DGS_NOBORDER     ) != 0);
661 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_NOFILTER    , (sty & AJC3DGS_NOFILTER     ) != 0);
662 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_NOANGLE     , (sty & AJC3DGS_NOANGLE      ) != 0);
```

```
663 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_AXIS_X      , (sty & AJC3DGS_SHOWAXIS_X   ) != 0);
664 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_AXIS_Y      , (sty & AJC3DGS_SHOWAXIS_Y   ) != 0);
665 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_AXIS_Z      , (sty & AJC3DGS_SHOWAXIS_Z   ) != 0);
666 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_RECTSCALE   , (sty & AJC3DGS_RECTSCALE    ) != 0);
667 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_ELPSSCALE   , (sty & AJC3DGS_ELPSSCALE    ) != 0);
668 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_SPHERESCALE , (sty & AJC3DGS_SPHERESCALE  ) != 0);
669 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_LINE_XY     , (sty & AJC3DGS_SCALE_XY     ) != 0);
670 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_LINE_XZ     , (sty & AJC3DGS_SCALE_XZ     ) != 0);
671 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_LINE_YZ     , (sty & AJC3DGS_SCALE_YZ     ) != 0);
672 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_VAL_X       , (sty & AJC3DGS_SCALEVALUE_X ) != 0);
673 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_VAL_Y       , (sty & AJC3DGS_SCALEVALUE_Y ) != 0);
674 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_VAL_Z       , (sty & AJC3DGS_SCALEVALUE_Z ) != 0);
675 : }
676 : //-------------------------------------------------------------------------------------------------------------//
677 : //  チェックボックスの設定をスタイルに反映                                                                     //
678 : //-------------------------------------------------------------------------------------------------------------//
679 : static  VO  SetCheckBoxToStyle(VO)
680 : {
681 :     UI      sty;
682 :
683 :     sty = (UI)MAjcGetWindowLong(hWndG3d, GWL_STYLE);
684 :     sty &= 0xFFFF0000;
685 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_NODEPTHCTRL )) sty |= AJC3DGS_NODEPTHCTRL;
686 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_NOBORDER    )) sty |= AJC3DGS_NOBORDER;
687 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_NOFILTER    )) sty |= AJC3DGS_NOFILTER;
688 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_NOANGLE     )) sty |= AJC3DGS_NOANGLE;
689 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_AXIS_X      )) sty |= AJC3DGS_SHOWAXIS_X;
690 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_AXIS_Y      )) sty |= AJC3DGS_SHOWAXIS_Y;
691 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_AXIS_Z      )) sty |= AJC3DGS_SHOWAXIS_Z;
692 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_RECTSCALE   )) sty |= AJC3DGS_RECTSCALE;
693 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_ELPSSCALE   )) sty |= AJC3DGS_ELPSSCALE;
694 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_SPHERESCALE )) sty |= AJC3DGS_SPHERESCALE;
695 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_SCALE_XY    )) sty |= AJC3DGS_SCALE_XY;
696 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_SCALE_XZ    )) sty |= AJC3DGS_SCALE_XZ;
697 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_SCALE_YZ    )) sty |= AJC3DGS_SCALE_YZ;
698 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_VAL_X       )) sty |= AJC3DGS_SCALEVALUE_X;
699 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_VAL_Y       )) sty |= AJC3DGS_SCALEVALUE_Y;
700 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_VAL_Z       )) sty |= AJC3DGS_SCALEVALUE_Z;
701 :     MAjcSetWindowLong(hWndG3d, GWL_STYLE, sty);
702 : }
```

6.13. SW_3DGraphic2（２Ｄ グラフィック）

このサンプルプログラムでは、２Ｄグラフィックモードで、データのプロッティングと図形の描画を行います。

４つのプロットデータを発生し、各プロットデータは円状に周回します。
また、プロットデータがＸ軸から０～９０度以内である場合は、当該プロットデータが周回する軌道を円で表示します。



プロットデータの保留数を設定します

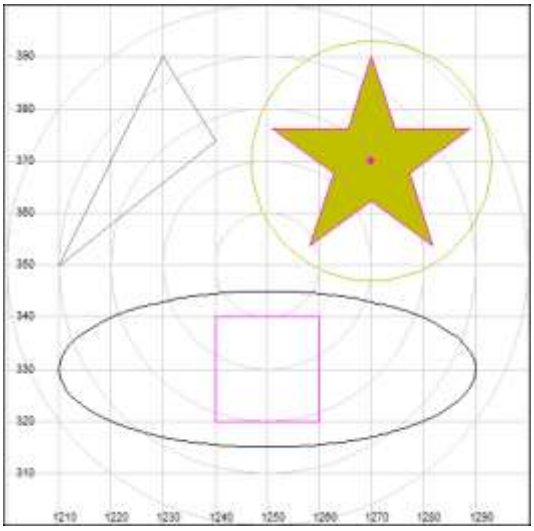
チェックするとプロットデータ間を結線します

図形描画／消去，プロットデータ生成の開始／停止，描画クリアー

プロットデータ生成周期

テストボタンを押すと、固定の図形を描画します。

テストボタンによる図形描画



```
 1 : //
 2 : //  SW_3DGraphic2.c
 3 : //
 4 :
 5 : #include    <AjrCstXX.h>
 6 : #include    <math.h>
 7 : #include    <tchar.h>
 8 : #include    "resource.h"
 9 :
10 : #define MYSECT  TEXT("Params")
11 :
12 : #define X1  1200.0
13 : #define X2  1300.0
14 : #define Y1  300.0
15 : #define Y2  400.0
16 : #define XC  ((X1 + X2) / 2.0)
17 : #define YC  ((Y1 + Y2) / 2.0)
18 :
19 : //-------------------------------------------------------------------------------------------------------------------//
20 : //  ワーク                                                                                                           //
21 : //-------------------------------------------------------------------------------------------------------------------//
22 : HINSTANCE      hInst;                      //  ＤＬＬインスタンスハンドル
23 : HWND           hDlgMain;                   //  ダイアログボックスハンドル
24 : HWND           hWndCtrl;                   //  ３Ｄグラフコントロールのウインドハンドル
25 : int            CtrlTop;                    //  ３Ｄグラフコントロールの上端位置
26 : BOOL           fPlot = FALSE;              //  プロット中を示すフラグ
27 : UI             period = 100;               //  プロット周期[ms]
28 : BOOL           fTest  = FALSE;
29 :
30 : //-------------------------------------------------------------------------------------------------------------------//
31 : //  内部サブ関数                                                                                                     //
32 : //-------------------------------------------------------------------------------------------------------------------//
33 : AJC_DLGPROC_DEF(Main);
34 : static  VO  SetStyleToCheckBox(VO);
35 : static  VO  SetCheckBoxToStyle(VO);
36 :
37 : //===================================================================================================================//
38 : //                                                                                                                   //
39 : //  Ｗｉｎ Ｍａｉｎ                                                                                                 //
40 : //                                                                                                                   //
41 : //===================================================================================================================//
42 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
43 : {
44 :     MSG     msg;
45 :     int     sty;
46 :     RECT    rect;
47 :
48 :     hInst = hInstance;
49 :
50 :     //----- メイン・ダイアログオープン -----------------//
51 :     hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_DLGMAIN), NULL, AJC_DLGPROC_NAME(Main));
52 :     sty = (int)MAjcGetWindowLong(hDlgMain, GWL_STYLE);
53 :     MAjcSetWindowLong(hDlgMain, GWL_STYLE, sty | WS_THICKFRAME);
54 :     //----- ３Ｄグラフコントロール位置初期化 -----------//
55 :     GetWindowRect(hWndCtrl, &rect);
56 :     MapWindowPoints(NULL, hDlgMain, (LPPOINT)&rect, 2);
57 :     CtrlTop = rect.top;
58 :     GetClientRect(hDlgMain, &rect);
59 :     MoveWindow(hWndCtrl, 0, CtrlTop, rect.right - rect.left, rect.bottom - rect.top - CtrlTop, TRUE);
60 :     //----- ダイアログ表示 -----------------------------//
61 :     ShowWindow(hDlgMain, SW_SHOW);
62 :
63 :     //----- メッセージループ ---------------------------//
64 :     while (GetMessage(&msg, NULL, 0, 0)) {
65 :         do {
66 :             if (IsDialogMessage(hDlgMain, &msg)) break;
67 :             TranslateMessage(&msg);
68 :             DispatchMessage (&msg);
69 :         } while (0);
70 :     }
71 :
72 :     return (int)msg.wParam ;
73 : }
74 : //===================================================================================================================//
75 : //                                                                                                                   //
76 : //  ダイアログ・プロシージャ                                                                                         //
77 : //                                                                                                                   //
78 : //===================================================================================================================//
79 : //----- ダイアログ初期化 -----------------------------------------------------------------------------------------//
80 : AJC_DLGPROC(Main, WM_INITDIALOG     )
81 : {
82 :     AJC3DGPROP  prop;
83 :
84 :     hDlgMain = hDlg;
85 :     hWndCtrl = GetDlgItem(hDlgMain, IDC_3DGRAPH);
86 :     //----- チップテキスト設定 -------------------------------//
87 :     AjcG3dSetTipText      (hWndCtrl,    TEXT("２Ｄグラフィック"));
88 :     AjcG3dSetChkBoxTipText(hWndCtrl, 0, TEXT("プロット０"));
89 :     AjcG3dSetChkBoxTipText(hWndCtrl, 1, TEXT("プロット１"));
```

```
 90 :     AjcG3dSetChkBoxTipText(hWndCtrl, 2, TEXT("プロット２"));
 91 :     AjcG3dSetChkBoxTipText(hWndCtrl, 2, TEXT("プロット３"));
 92 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_CHK_NOFILTER ), TEXT("データ表示／非表示用チェックボックス非表示"));
 93 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_CHK_NOBORDER ), TEXT("外枠非表示"));
 94 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_CHK_AXIS_X   ), TEXT("Ｘ軸表示"));
 95 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_CHK_AXIS_Y   ), TEXT("Ｙ軸表示"));
 96 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_CHK_RECTSCALE), TEXT("方眼スケール表示 (表示面は SCALE_XY で指定) "));
 97 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_CHK_ELPSSCALE), TEXT("同心円スケール表示 (表示面は SCALE_XY で指定) "));
 98 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_CHK_SCALE_XY ), TEXT("ＸＹ平面にスケール表示 (スケールは RECTSCALE, ELPSSCALE で指定) "));
 99 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_CHK_VAL_X    ), TEXT("Ｘ軸の値表示"));
100 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_CHK_VAL_Y    ), TEXT("Ｙ軸の値表示"));
101 :     //----- ２Ｄグラフモード設定 ------------------------------//
102 :     AjcG2dInit(hWndCtrl, X1, Y1, X2, Y2, AJC3DGS_2DMODE | AJC3DGS_ELPSSCALE);
103 :     //----- ウインドスタイル値をチェックボックスに設定 -----------//
104 :     SetStyleToCheckBox();
105 :     //----- 初期値設定 ----------------------------------------//
106 :     AjcSetDlgItemUInt(hDlg, IDC_INP_PLOTNUM0, 16);
107 :     AjcSetDlgItemUInt(hDlg, IDC_INP_PLOTNUM1, 16);
108 :     AjcSetDlgItemUInt(hDlg, IDC_INP_PLOTNUM2, 16);
109 :     AjcSetDlgItemUInt(hDlg, IDC_INP_PLOTNUM3, 16);
110 :     AjcSetDlgItemChk (hDlg, IDC_CHK_PLOTLINE, TRUE);
111 :     //----- 設定値ロード --------------------------------------//
112 :     AjcLoadAllControlSettings(hDlg, TEXT("Settings"), AJCCTL_SELACT_ALL);
113 :     //----- チェックボックスの内容をスタイルに反映 ---------------//
114 :     SetCheckBoxToStyle();
115 :     //----- グラフレンジ設定 ----------------------------------//
116 :     AjcG2dSetRange(hWndCtrl, X1, Y1, X2, Y2);
117 :     //----- ２Ｄグラフコントロールのプロパティ反映 ---------------//
118 :     AjcG2dGetProp(hWndCtrl, &prop);
119 :     prop.Item[0].MaxPlot = AjcGetDlgItemUInt(hDlg, IDC_INP_PLOTNUM0);
120 :     prop.Item[1].MaxPlot = AjcGetDlgItemUInt(hDlg, IDC_INP_PLOTNUM1);
121 :     prop.Item[2].MaxPlot = AjcGetDlgItemUInt(hDlg, IDC_INP_PLOTNUM2);
122 :     prop.Item[3].MaxPlot = AjcGetDlgItemUInt(hDlg, IDC_INP_PLOTNUM3);
123 :     prop.fPlotLine       = AjcGetDlgItemChk (hDlg, IDC_CHK_PLOTLINE);
124 :     AjcG2dSetProp(hWndCtrl, &prop);
125 :     return TRUE;
126 : }
127 : //----- ウインド破棄 ------------------------------------------------------------------------------------------//
128 : AJC_DLGPROC(Main, WM_DESTROY        )
129 : {
130 :     //----- 設定値セーブ --------------------------------------//
131 :     AjcSaveAllControlSettings(hDlg);
132 :     //----- プログラム終了 ------------------------------------//
133 :     PostQuitMessage(0);
134 :     return TRUE;
135 : }
136 : //----- サイズ変更 --------------------------------------------------------------------------------------------//
137 : AJC_DLGPROC(Main, WM_SIZE           )
138 : {
139 :     int     w = LOWORD(lParam);
140 :     int     h = HIWORD(lParam);
141 :
142 :     MoveWindow  (GetDlgItem(hDlg, IDC_3DGRAPH), 0, CtrlTop, w, h - CtrlTop, TRUE);
143 :
144 :     return TRUE;
145 : }
146 : //----- COMMAND -----------------------------------------------------------------------------------------------//
147 : AJC_DLGPROC(Main, WM_COMMAND        )
148 : {
149 :     BOOL        rc = FALSE;
150 :     int         cmd = LOWORD(wParam);
151 :     UI          ix;
152 :     AJC3DGPROP  prop;
153 :
154 :     //----- スタイル設定チェックボックス操作 ----------------------------------------//
155 :     if (cmd >= IDC_CHK_RECTSCALE && cmd <= IDC_CHK_AXIS_Y && HIWORD(wParam) == BN_CLICKED) {
156 :         //  チェックボックスの内容をスタイルに反映
157 :         SetCheckBoxToStyle();
158 :         rc = TRUE;
159 :     }
160 :     //----- プロット数設定操作 ------------------------------------------------------//
161 :     if (cmd >= IDC_INP_PLOTNUM0 && cmd <= IDC_INP_PLOTNUM3 && HIWORD(wParam) == AJCIVN_INTVALUE) {
162 :         ix = cmd - IDC_INP_PLOTNUM0;
163 :         AjcG2dGetProp(hWndCtrl, &prop);
164 :         prop.Item[ix].MaxPlot = (UI)lParam;
165 :         AjcG2dSetProp(hWndCtrl, &prop);
166 :         rc = TRUE;
167 :     }
168 :
169 :     return rc;
170 : }
171 : //----- WM_TIMER ----------------------------------------------------------------------------------------------//
172 : AJC_DLGPROC(Main, WM_TIMER      )
173 : {
174 :     UI      i;
175 :     double  x, y, r, xc, yc;
176 :     static  BOOL    fElps[4] = {0, 0, 0, 0};
177 :     static  double  theta[4] = {0, 0, 0, 0};
178 :
179 :     for (i = 0; i < 4; i++) {
```

```
180 :         xc = XC; yc = YC;
181 :         r  = 10 * (i + 1);
182 :         x  = r * cos(theta[i]);
183 :         y  = r * sin(theta[i]);
184 :         AjcG2dPutPlotData(hWndCtrl, i, XC + x, YC + y);
185 :         if (theta[i] >= 0.0 && theta[i] <= AJC_DEG2RAD(90.0)) {
186 :             if (!fElps[i]) {
187 :                 AjcG2dEllipse (hWndCtrl, i, xc, yc, r, r);
188 :                 fElps[i] = TRUE;
189 :             }
190 :         }
191 :         else {
192 :             if (fElps[i]) {
193 :                 AjcG2dClearShape(hWndCtrl, i);
194 :                 fElps[i] = FALSE;
195 :             }
196 :         }
197 :         theta[i] += (AJC_DEG2RAD(10.0) * (double)rand() / (double)RAND_MAX);
198 :         theta[i]  = fmod(theta[i], (AJC_PAI * 2.0));
199 :     }
200 :     return TRUE;
201 : }
202 : //----- プロットライン・チェックボックス ----------------------------------------------------------------------//
203 : AJC_DLGPROC(Main, IDC_CHK_PLOTLINE)
204 : {
205 :     AJC3DGPROP  prop;
206 :
207 :     if (HIWORD(wParam) == BN_CLICKED) {
208 :         AjcG2dGetProp(hWndCtrl, &prop);
209 :         prop.fPlotLine = AjcGetDlgItemChk(hDlg, IDC_CHK_PLOTLINE);
210 :         AjcG2dSetProp(hWndCtrl, &prop);
211 :     }
212 :     return TRUE;
213 : }
214 : //----- プロット開始／停止ボタン ------------------------------------------------------------------------------//
215 : AJC_DLGPROC(Main, IDC_CMD_START_STOP)
216 : {
217 :     if (fPlot) {
218 :         fPlot = FALSE;
219 :         KillTimer(hDlg, 1);
220 :         AjcSetDlgItemStr(hDlg, IDC_CMD_START_STOP, TEXT("プロット開始"));
221 :     }
222 :     else {
223 :         fPlot = TRUE;
224 :         SetTimer(hDlg, 1, period, NULL);
225 :         AjcSetDlgItemStr(hDlg, IDC_CMD_START_STOP, TEXT("プロット停止"));
226 :     }
227 :     return TRUE;
228 : }
229 : //----- プロット周期 ------------------------------------------------------------------------------------------//
230 : AJC_DLGPROC(Main, IDC_INP_PERIOD    )
231 : {
232 :     if (HIWORD(wParam) == AJCIVN_INTVALUE) {
233 :         period = AjcGetDlgItemUInt(hDlg, IDC_INP_PERIOD);
234 :         if (fPlot) {
235 :             SetTimer(hDlgMain, 1, period, NULL);
236 :         }
237 :     }
238 :     return TRUE;
239 : }
240 : //----- テストボタン ------------------------------------------------------------------------------------------//
241 : AJC_DLGPROC(Main, IDC_CMD_TEST      )
242 : {
243 :     if (!fTest) {
244 :         AjcG2dStar(hWndCtrl, 4, 1270, 370, 20);
245 :         AjcG2dFillB  (hWndCtrl, 5, 4, 1270, 370);
246 :         AjcG2dPixel(hWndCtrl, 4,  1270, 370, 3);
247 :
248 :         AjcG2dEllipse(hWndCtrl  , 5, 1270, 370,  23, 23);
249 :         AjcG2dTriangle(hWndCtrl , 6, 1210, 350,  1230, 390,  1240, 374);
250 :         AjcG2dEllipse(hWndCtrl  , 7, 1250, 330,  40, 15);
251 :         AjcG2dRectangle(hWndCtrl, 4, 1240, 340,  1260, 320);
252 :         fTest = TRUE;
253 :     }
254 :     else {
255 :         AjcG2dClearAllShape(hWndCtrl);
256 :         fTest = FALSE;
257 :     }
258 :     return TRUE;
259 : }
260 : //----- クリアーボタン ----------------------------------------------------------------------------------------//
261 : AJC_DLGPROC(Main, IDC_CMD_CLEAR     )
262 : {
263 :     fTest = FALSE;
264 :     AjcG2dClear(hWndCtrl);
265 :     return TRUE;
266 : }
267 : //----- 「Cancel」ボタン --------------------------------------------------------------------------------------//
268 : AJC_DLGPROC(Main, IDCANCEL          )
269 : {
```

```
270 :     DestroyWindow(hDlg);
271 :     return TRUE;
272 : }
273 : //-----------------------------------------------------------------------------------------------------------------//
274 : AJC_DLGMAP_DEF(Main)
275 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG     )
276 :     AJC_DLGMAP_MSG(Main, WM_SIZE           )
277 :     AJC_DLGMAP_MSG(Main, WM_DESTROY        )
278 :     AJC_DLGMAP_MSG(Main, WM_COMMAND        )
279 :     AJC_DLGMAP_MSG(Main, WM_TIMER          )
280 :
281 :     AJC_DLGMAP_CMD(Main, IDC_CHK_PLOTLINE  )
282 :     AJC_DLGMAP_CMD(Main, IDC_CMD_START_STOP)
283 :     AJC_DLGMAP_CMD(Main, IDC_INP_PERIOD    )
284 :     AJC_DLGMAP_CMD(Main, IDC_CMD_TEST      )
285 :     AJC_DLGMAP_CMD(Main, IDC_CMD_CLEAR     )
286 :     AJC_DLGMAP_CMD(Main, IDCANCEL          )
287 : AJC_DLGMAP_END
288 :
289 : //-----------------------------------------------------------------------------------------------------------------//
290 : //  ウインドスタイル値をチェックボックスに設定                                                                     //
291 : //-----------------------------------------------------------------------------------------------------------------//
292 : static  VO  SetStyleToCheckBox(VO)
293 : {
294 :     UI      sty;
295 :
296 :     sty = (UI)MAjcGetWindowLong(hWndCtrl, GWL_STYLE);
297 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_NOBORDER    , (sty & AJC3DGS_NOBORDER     ) != 0);
298 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_NOFILTER    , (sty & AJC3DGS_NOFILTER     ) != 0);
299 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_AXIS_X      , (sty & AJC3DGS_SHOWAXIS_X   ) != 0);
300 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_AXIS_Y      , (sty & AJC3DGS_SHOWAXIS_Y   ) != 0);
301 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_RECTSCALE   , (sty & AJC3DGS_RECTSCALE    ) != 0);
302 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_ELPSSCALE   , (sty & AJC3DGS_ELPSSCALE    ) != 0);
303 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_LINE_XY     , (sty & AJC3DGS_SCALE_XY     ) != 0);
304 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_VAL_X       , (sty & AJC3DGS_SCALEVALUE_X ) != 0);
305 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_VAL_Y       , (sty & AJC3DGS_SCALEVALUE_Y ) != 0);
306 : }
307 : //-----------------------------------------------------------------------------------------------------------------//
308 : //  チェックボックスの設定をスタイルに反映                                                                         //
309 : //-----------------------------------------------------------------------------------------------------------------//
310 : static  VO  SetCheckBoxToStyle(VO)
311 : {
312 :     UI      sty;
313 :
314 :     sty = (UI)MAjcGetWindowLong(hWndCtrl, GWL_STYLE);
315 :     sty &= ~(AJC3DGS_NOBORDER  | AJC3DGS_NOFILTER | AJC3DGS_SHOWAXIS_X    | AJC3DGS_SHOWAXIS_Y | AJC3DGS_RECTSCALE |
316 :              AJC3DGS_ELPSSCALE | AJC3DGS_SCALE_XY | AJC3DGS_SCALEVALUE_X | AJC3DGS_SCALEVALUE_Y);
317 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_NOBORDER    )) sty |= AJC3DGS_NOBORDER;
318 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_NOFILTER    )) sty |= AJC3DGS_NOFILTER;
319 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_AXIS_X      )) sty |= AJC3DGS_SHOWAXIS_X;
320 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_AXIS_Y      )) sty |= AJC3DGS_SHOWAXIS_Y;
321 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_RECTSCALE   )) sty |= AJC3DGS_RECTSCALE;
322 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_ELPSSCALE   )) sty |= AJC3DGS_ELPSSCALE;
323 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_SCALE_XY    )) sty |= AJC3DGS_SCALE_XY;
324 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_VAL_X       )) sty |= AJC3DGS_SCALEVALUE_X;
325 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_VAL_Y       )) sty |= AJC3DGS_SCALEVALUE_Y;
326 :     MAjcSetWindowLong(hWndCtrl, GWL_STYLE, sty);
327 : }
328 :
```

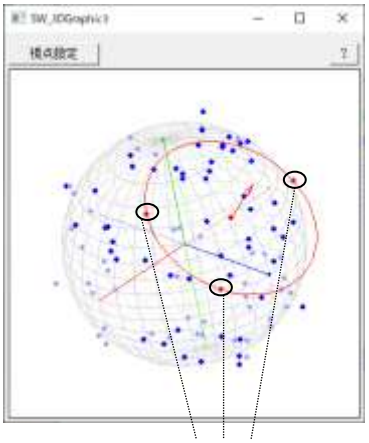
6.14. SW_3DGraphic3（視点設定）

このサンプルプログラムでは、最初にランダムな点を１００個表示します。
SHIFT キーを押しながら、３個の点をクリックしてください。
クリックした３点から平面円と平面の法線を算出し、表示します。
「視点設定」ボタンを押すと、平面の真上から見た図となるように、視点を設定します。

起動時

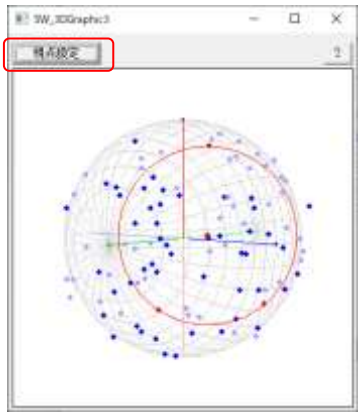


３点選択



SHIFT キーを押しながら、３点をクリック
（クリックした点は赤色に変わります）

視点設定



「視点設定」ボタンを押すと、
視点を平面円の真上に設定します。

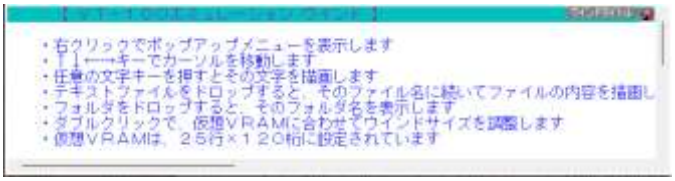
```
 1 : //
 2 : //  SW_3DGraphic3.c
 3 : //
 4 :
 5 : #include    <AjrCstXX.h>
 6 : #include    <math.h>
 7 : #include    <tchar.h>
 8 : #include    "resource.h"
 9 :
10 : //-------------------------------------------------------------------------------------------------------------//
11 : //  ワーク                                                                                                     //
12 : //-------------------------------------------------------------------------------------------------------------//
13 : static  HINSTANCE       hInst;                      //  ＤＬＬインスタンスハンドル
14 : static  HWND            hDlgMain;                   //  ダイアログボックスハンドル
15 : static  SIZE            szDlg;                      //  ダイアログの最小サイズ
16 : static  HWND            hWndG3d;                    //  ３Ｄグラフィックコントロール
17 : static  RECT            rcG3d;                      //  初期の３Ｄグラフィックコントロール矩形
18 : static  AJC3DVEC        cent = {1.0, 2.0, 3.0};     //  中心位置
19 :
20 : static  UI              Count = 0;                  //  選択ポイント数
21 : static  UI              Ix    = 0;                  //  ポイントバッファインデクス
22 : static  AJC3DVEC        Points[3];                  //  ポイントバッファ
23 : static  AJC3DVEC        vh;                         //  視点方向ベクトル（平面の法線）
24 :
25 : static const UT TipMsg[] =  TEXT("SHIFT+クリックで、任意の３点を選択してください。¥n")
26 :                             TEXT("３点が選択されたら、各点をとおる平面円が表示されます。¥n")
27 :                             TEXT("¥n")
28 :                             TEXT("視点設定ボタンを押すと、視点を平面の真上に設定します。¥n")
29 : ;
30 :
31 : //-------------------------------------------------------------------------------------------------------------//
32 : //  内部サブ関数                                                                                               //
33 : //-------------------------------------------------------------------------------------------------------------//
34 : AJC_DLGPROC_DEF(Main);
35 : static  VO      SetControlSize(HWND hDlg);
36 :
37 : //=============================================================================================================//
38 : //                                                                                                             //
39 : //  Ｗ ｉ ｎ Ｍ ａ ｉ ｎ                                                                                       //
40 : //                                                                                                             //
41 : //=============================================================================================================//
42 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
43 : {
44 :     MSG     msg;
45 :     RECT    rect;
46 :
47 :     hInst = hInstance;
48 :
49 :     //----- メイン・ダイアログオープン ----------------//
```

```
 50 :     hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_DLGMAIN), NULL, AJC_DLGPROC_NAME(Main));
 51 :     ShowWindow(hDlgMain, SW_SHOW);
 52 :     //----- ウインド最小サイズ退避 --------------------//
 53 :     GetWindowRect(hDlgMain, &rect);
 54 :     szDlg.cx = rect.right - rect.left;
 55 :     szDlg.cy = rect.bottom - rect.top;
 56 :
 57 :     //----- メッセージループ ---------------------------//
 58 :     while (GetMessage(&msg, NULL, 0, 0)) {
 59 :         do {
 60 :             if (IsDialogMessage(hDlgMain, &msg)) break;
 61 :             TranslateMessage(&msg);
 62 :             DispatchMessage (&msg);
 63 :         } while (0);
 64 :     }
 65 :
 66 :     return (int)msg.wParam ;
 67 : }
 68 : //=============================================================================================================//
 69 : //                                                                                                             //
 70 : //  ダイアログ・プロシージャ                                                                                   //
 71 : //                                                                                                             //
 72 : //=============================================================================================================//
 73 : //----- ダイアログ初期化 ---------------------------------------------------------------------------------------//
 74 : AJC_DLGPROC(Main, WM_INITDIALOG     )
 75 : {
 76 :     UI          i;
 77 :     AJC3DVEC    v;
 78 :     double      max;
 79 :
 80 :     hDlgMain = hDlg;
 81 :     hWndG3d = GetDlgItem(hDlgMain, IDC_3DGRAPH);
 82 :
 83 :     GetWindowRect(hWndG3d, &rcG3d); MapWindowPoints(NULL, hDlg, (LPPOINT)&rcG3d, 2);
 84 :     SetControlSize(hDlg);
 85 :
 86 :     AjcG3dSetCenter (hWndG3d, cent.x, cent.y, cent.z);
 87 :     AjcG3dSetAngle3D(hWndG3d);
 88 :
 89 :     //  球面付近にランダムな点を１００個表示
 90 :     for (i = 0; i < 100; i ++) {
 91 :         v.x = (double)rand() / (double)RAND_MAX;
 92 :         max = sqrt(1.0 - pow(v.x, 2.0));
 93 :         v.y = ((double)rand() / (double)RAND_MAX) * max;
 94 :         v.z = sqrt(1.0 - pow(v.x, 2.0) - pow(v.y, 2.0));
 95 :         if (rand() & 1) v.x += ((double)rand() / (double)RAND_MAX * 0.1); else v.x -= ((double)rand() / (double)RAND_MAX * 0.1);
 96 :         if (rand() & 1) v.y += ((double)rand() / (double)RAND_MAX * 0.1); else v.y -= ((double)rand() / (double)RAND_MAX * 0.1);
 97 :         if (rand() & 1) v.z += ((double)rand() / (double)RAND_MAX * 0.1); else v.z -= ((double)rand() / (double)RAND_MAX * 0.1);
 98 :         if (rand() & 1) v.x *= -1;
 99 :         if (rand() & 1) v.y *= -1;
100 :         if (rand() & 1) v.z *= -1;
101 :         AjcV3dAdd(&v, &cent, &v);
102 :         AjcG3dPixelV(hWndG3d, 0, &v, 3);
103 :     }
104 :
105 :     //  初期メッセージ表示
106 :     AjcTipTextShowCenter(hDlg, TipMsg, 10000, NULL);
107 :
108 :     return TRUE;
109 : }
110 : //----- ウインド破棄 -------------------------------------------------------------------------------------------//
111 : AJC_DLGPROC(Main, WM_DESTROY        )
112 : {
113 :     PostQuitMessage(0);
114 :     return TRUE;
115 : }
116 : //----- サイズ変更中 -------------------------------------------------------------------------------------------//
117 : AJC_DLGPROC(Main, WM_SIZING         )
118 : {
119 :     LPRECT  pRect = (LPRECT)lParam;
120 :     SIZE    sz;
121 :
122 :     sz.cx = pRect->right - pRect->left;
123 :     sz.cy = pRect->bottom - pRect->top;
124 :
125 :     if (sz.cx < szDlg.cx) pRect->right  = pRect->left + szDlg.cx;
126 :     if (sz.cy < szDlg.cy) pRect->bottom = pRect->top  + szDlg.cy;
127 :
128 :     return TRUE;
129 : }
130 : //----- サイズ変更 ---------------------------------------------------------------------------------------------//
131 : AJC_DLGPROC(Main, WM_SIZE           )
132 : {
133 :     SetControlSize(hDlg);
134 :     return TRUE;
135 : }
136 : //- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - //
137 : static  VO      SetControlSize(HWND hDlg)
138 : {
139 :     RECT    r;
```

```
140 :     int     w, h;
141 :
142 :     GetClientRect(hDlg, &r);
143 :     w = r.right - r.left;
144 :     h = r.bottom - r.top;
145 :     MoveWindow(hWndG3d, rcG3d.left, rcG3d.top, w - rcG3d.left - 4, h - rcG3d.top  - 4, FALSE);
146 :     InvalidateRect(hDlg, NULL, TRUE);
147 : }
148 : //----- 「視点設定」ボタン ----------------------------------------------------------------------------------------//
149 : AJC_DLGPROC(Main, IDC_CMD_VIEWPOINT )
150 : {
151 :     AjcG3dSetAngleV(hWndG3d, &vh);
152 :     return TRUE;
153 : }
154 : //----- 「？」ボタン ----------------------------------------------------------------------------------------------//
155 : AJC_DLGPROC(Main, IDC_CMD_HELP      )
156 : {
157 :     AjcTipTextShowCenter(hDlg, TipMsg, 10000, NULL);
158 :     return TRUE;
159 : }
160 : //----- ３Ｄグラフコントロール ------------------------------------------------------------------------------------//
161 : AJC_DLGPROC(Main, IDC_3DGRAPH       )
162 : {
163 :     switch (HIWORD(wParam)) {
164 :         case AJC3DGN_PLOTLIST:                          //  ●プロットリスト通知
165 :         {   PCAJC3DGPLOTLIST    p  = (PCAJC3DGPLOTLIST)lParam;
166 :             UI                  id = 1;
167 :             UI                  i;
168 :             AJC3DVEC            vc, v;
169 :             double              r;
170 :             AJC3DCIRINFO        cif;
171 :             //  プロット点をバッファへ格納
172 :             memcpy(&Points[Ix], &p->lst[0].vec, sizeof Points[Ix]);
173 :             Ix = (Ix + 1) % 3;
174 :             if (Count < 3) Count++;
175 :             //  プロット点表示
176 :             AjcG3dClearShape(hWndG3d, id);
177 :             for (i = 0; i < Count; i++) {
178 :                 AjcG3dPixelV(hWndG3d, id, &Points[i], 3);
179 :             }
180 :             //  ３点が揃ったら平面円と法線表示
181 :             if (Count >= 3) {
182 :                 AjcEnableDlgItem(hDlg, IDC_CMD_VIEWPOINT, TRUE);
183 :                 r = AjcV3dCalcCircleVEx(&Points[Ix], &Points[(Ix + 1) % 3], &Points[(Ix + 2) % 3], &vc, &vh, &cif);
184 :                 AjcG3dDefPlaneV(hWndG3d, id, &cif.lvc, NULL);
185 :                 AjcG2dEllipse  (hWndG3d, id, 0, 0, cif.cr, cif.cr);
186 :                 AjcG2dPixel    (hWndG3d, id, 0, 0, 3);
187 :                 AjcV3dNormal   (&vh, &v);
188 :                 AjcV3dMult     (&v, 0.5, &v);
189 :                 AjcV3dAdd      (&v, &vc, &v);
190 :                 AjcG3dArrowV   (hWndG3d, id, &vc, &v);
191 :             }
192 :             break;
193 :         }
194 :     }
195 :     return TRUE;
196 : }
197 : //----- 「Cancel」ボタン ------------------------------------------------------------------------------------------//
198 : AJC_DLGPROC(Main, IDCANCEL          )
199 : {
200 :     DestroyWindow(hDlg);
201 :     return TRUE;
202 : }
203 : //-----------------------------------------------------------------------------------------------------------------//
204 : AJC_DLGMAP_DEF(Main)
205 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG     )
206 :     AJC_DLGMAP_MSG(Main, WM_DESTROY        )
207 :     AJC_DLGMAP_MSG(Main, WM_SIZING         )
208 :     AJC_DLGMAP_MSG(Main, WM_SIZE           )
209 :
210 :     AJC_DLGMAP_CMD(Main, IDC_CMD_VIEWPOINT )
211 :     AJC_DLGMAP_CMD(Main, IDC_CMD_HELP      )
212 :     AJC_DLGMAP_CMD(Main, IDC_3DGRAPH       )
213 :     AJC_DLGMAP_CMD(Main, IDCANCEL          )
214 : AJC_DLGMAP_END
215 :
```

7. ＶＴ－１００エミュレーション・ウインド コントロール（AjcCtrlVT100 クラス）

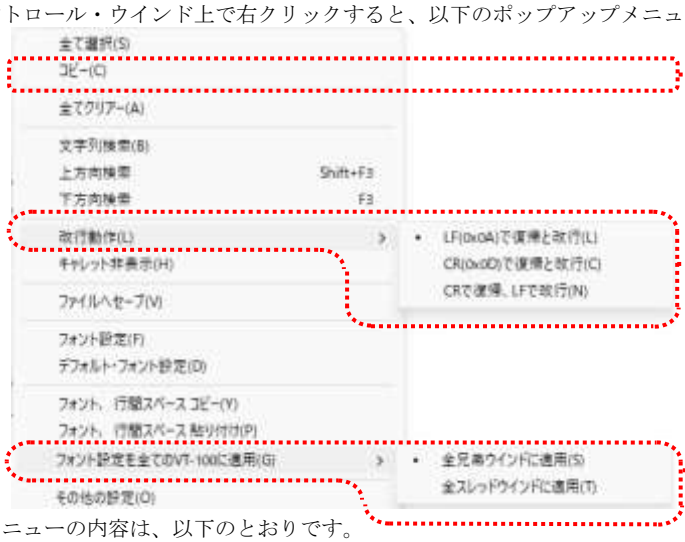
データのログ表示等、テキストの表示用ウインド・コントロールです。
ＡＮＳＩエスケープコードにより、描画スクリーン上の任意の（文字単位の）位置にグラフィカルに文字列を描画することもできます。
ＡＮＳＩエスケープコードでは、描画色の設定や、スクリーンの部分スクロール等ができます。（サポートするコードは、後述します）



7.1. 機能概要

7.1.1. ポップアップメニュー

コントロール・ウインド上で右クリックすると、以下のポップアップメニューが表示されます。



テキストが選択されていない場合は表示されません

AJCVTHS_FIXEDLF スタイルが設定されている場合は表示されません

AJCVTHS_FIXEDFONT スタイルが設定されている場合は表示されません

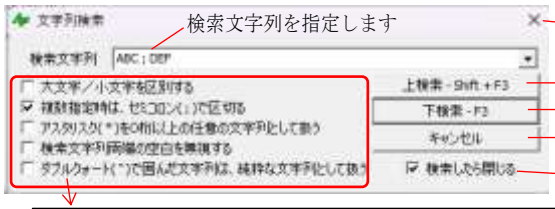
各メニューの内容は、以下のとおりです。

#	メニュー	内容
1	全て選択	全てのテキストを選択状態にします
2	コピー	選択されているテキストをクリップボードへコピーします （テキストが選択されていない場合、このメニューは表示されません）
3	全てクリアー	全てのテキストデータを破棄し、画面をクリアーします。
4	文字列検索	文字列の検索ダイアログボックスを表示します
5	上方向検索	文字列を上方向に検索します
6	下方向検索	文字列を下方向に検索します
7	改行動作	CR(0x0D)，LF(0x0A)による改行動作の選択
8	キャレット非表示	キャレット（点滅文字カーソル）を非表示にします。 次回は「キャレット表示」メニューに変わります
9	ファイルへセーブ	全てのテキスト／選択されているテキストを、テキストファイル／ＨＴＭＬファイルに書き込みます。
10	フォント設定	ダイアログにより、フォントの設定を行います
11	デフォルト・フォント設定	デフォルトフォント設定（ウインドキャプションで指定されたフォント／親ウインドのフォント）
12	フォント．行間スペース コピー	フォントと行間スペース情報をクリップボードにコピーします
13	フォント．行間スペース 貼り付け	クリップボードからフォントと行間スペース情報を読み出して設定します
14	フォント設定を全ての VT-100 に適用	本コントロールのフォントと行間スペースの設定を、AJCVTHS_FIXEDFONT スタイルが設定されていない全ての ＶＴ－１００コントロール（兄弟ウインド／スレッドウインド）へ適用します。 兄弟ウインド／スレッドウインド中に他の(AJCVTHS_FIXEDFONT スタイルが設定されていない)ＶＴ－１００ コントロールが無い場合は非表示
15	その他の設定	各種設定ダイアログを表示します

既定では、コントロールにフォーカスがある状態で、F1 キー押下でもポップアップメニューを表示します。
ポップアップメニュー表示キーを変更／無効にするには、AjcVthSetMenuKey()を実行します。

7.1.2. 文字列の検索

右クリックによるポップアップメニューから「文字列検索」を選択すると、以下のダイアログボックスが表示されます。



・**大文字／小文字を区別する**
　英字の大文字と小文字を区別して比較する（”AAA”≠”aaa”）
・**複数して時はセミコロンで区切る**
　複数の文字列を指定する場合は、セミコロン（;）で区切って指定できます。(ex. “ABC;XYZ”)
・**アスタリスク(*)を０桁以上の任意の文字列として扱う**
　検索文字列中「*」が存在する場合、当該文字を任意の（０桁以上の）文字列として扱います。
　つまり、「*」で分離された文字列群は、文字列の出現順を示すことになります。
　例えば、検索文字列=“BCD*OPQ*VWX”と指定した場合、“BCD”, “OPQ”, “VWX” が順に出現することを意味します。
　この時、文字列“ABCDEFGHIJKLMNOPQRSTUVWXYZ”が存在する場合、下線部分の文字列が「見つかった」と判断されます。
・**検索文字列両端の空白を無視する**
　検索文字列両端の空白を除去した文字列を検索します。(ex.“ ABC DEF “ -> “ABC DEF”)
　また、セミコロン(;)やアスタリスク(*)で分離された部分文字列も両端の空白を除去します。
　(ex.“ ABC ； DEF ； 123 * 456” -> “ABC;DEF;123*456”)
・**ダブルクォート（“）で囲んだ文字列は、純粋な文字列として扱う**
　ダブルクォートで囲んだ文字列(ex.“ ABC ; DEF ”等)は、指定した文字列そのものとして扱います。
　この文字列には、空白、セミコロン(;)やアスタリスク(*)を含めることができます。
　但し、「¥」はエスケープ文字として認識され、「¥」の次の文字を有効とします。(ex 「¥”」→「”」, 「¥¥」→「¥」, 「¥X」→「X」)

VT-100 エミュレーションコントロール／コンボボックスの検索文字列にフォーカスがある状態で、F3/Shift+F3 キーを押しても文字列の検索が実行されます。検索して見つかった文字列は、選択状態（反転表示）となります。

文字列検索ダイアログボックスで指定した情報を永続化するプロファイルへの記録セクションを指定するには、以下のＡＰＩを実行します。

AjcVthSetFindProfileSect() ： 文字列検索情報永続化プロファイルセクション名設定

このＡＰＩはコントロールを生成した直後に（WM_INITDIALOG 等で）実行してください。
このＡＰＩを実行していない場合、プロファイルセクションは「 _DefaultStrFindInfoSect_ 」固定となります。

上記ＡＰＩを実行する代わりに、キャプション文字列によるプロパティの設定で「FS=*SectionName*」を指定してもＯＫ。

文字列検索開始位置

文字列検索の開始位置は、以下のように設定されます。

操作状態		下方向検索 開始位置	上方向検索 開始位置
最初		ウインド上端の行頭	ウインド下端の次の行頭
文字列検索	見つかった	見つかった文字列先頭の次の文字	見つかった文字列先頭の前の文字
	見つからない	変化なし	変化なし
左ボタンクリック		クリックした位置	クリックした位置
スクロールバーを操作		ウインド上端の行頭	ウインド下端の次の行頭
マウスホィールを操作			
新たなテキストを表示			

文字列検索キー

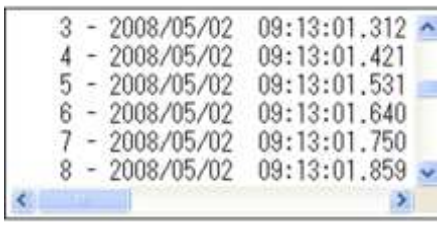
既定では、F3 キーで下検索、Shift+F3 キーで上検索を行います。
F3 キーを変更／無効にするには、AjcVthSetFindKey()を実行します。

7.1.3. テキストの選択とコピー

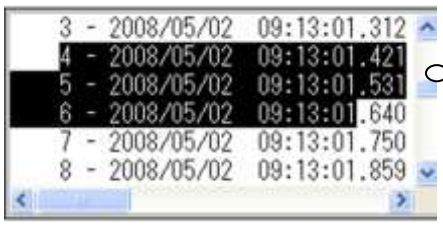
マウスの左ボタンでのドラッグ操作により、任意のテキストを選択できます。
CTRL キーを押下しながらドラッグした場合は、行単位のテキスト選択となります。
選択されたテキストは、ポップアップメニュー（右クリック）から「コピー」を選択、あるいはＣＴＲＬ＋Ｃキーを押下することにより、クリップボードへコピーできます。
また、ＳＨＩＦＴキーを押しながらドラッグ操作を行うことにより、スクロール後に前回のドラッグ操作を継続することができます。

マウスの左クリックにより、テキストの選択状態は解除されます。

テキストを選択している状態では、ウインドの外枠がグレー表示されます。



テキストを選択していない状態

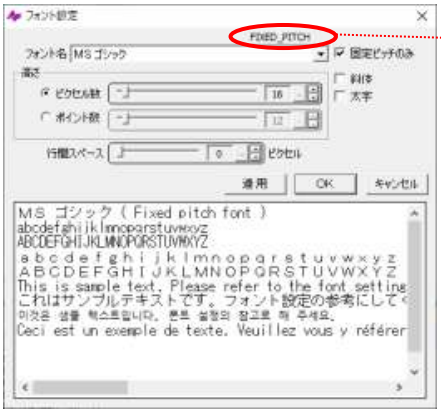


テキストを選択している状態

ＳＨＩＦＴキーを押しながら ↑，↓，←，or → キーにより選択範囲の変更は、ＶＴ１００コントロールがダイアログボックス下の項目となっている場合は使用できません。

7.1.4. フォントの設定

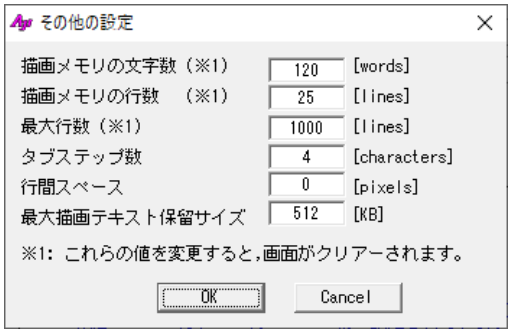
右クリックによるポップアップメニューから「フォント設定」を選択し、任意のフォントを設定することができます。



固定ピッチのフォントが選択されていることを示します。
可変ピッチのフォントが選択されている場合は、「VARIABLE_PITCH」と表示します。

7.1.5. その他の設定

右クリックによるポップアップメニューから「その他の設定」を選択し、以下のダイアログにより、各種設定を行います。



※「描画メモリの文字数」は、半角／全角の区別なく、１行に描画可能な文字数
※「タブステップ数」は、半角文字数で指定

＜注＞「描画メモリの文字数」「描画メモリの行数」「最大行数」を変更した場合は、全てのテキストがクリアーされます。

7.1.6. ファイルへセーブ

右クリックによるポップアップメニューから「ファイルへセーブ」を選択すると、以下のダイアログが表示されます。



「…」ボタンを押すと、ダイアログにより出力ファイルを設定できます。
ここで、以下の各設定を行い、ＯＫボタンを押すと、テキストをファイルに書き込みます。

グループ	項目	内容	備考
形式	テキスト	テキストファイルを出力	
	HTML(UTF-8)	ＨＴＭＬ形式のファイルを出力	UTF-8（BOM 付き）で出力
	フォントサイズ	ＨＴＭＬ形式時の、フォントサイズ（１～７）	
テキスト エンコード	S-JIS	シフトＪＩＳコードで出力	テキストファイル出力時の文字コード
	UTF-8	ＵＴＦ－８コードで出力	
	EUC-JP	日本語ＥＵＣコードで出力	
	UTF-16 LE	ＵＴＦ－１６（リトルエンディアン）	
	UTF-16 BE	ＵＴＦ－１６（ビッグエンディアン）	
	BOM	ファイルの先頭にＢＯＭを出力	UTF-8/UTF-16 選択時のみ有効
出力範囲	全テキスト	全てのテキストを出力	
	選択されている部分	選択されている部分のテキストを出力	

7.1.7. 表示の高速化

テキストを１字１句漏らさずに表示＆スクロールすると表示処理に時間がかかります。
そこで、AjcVthPause()により一定期間の表示を抑止することで表示時間を短縮することができます。

```
      ：
AjcVthPause(TRUE);          // 表示停止
AjcVthPrintF(・・・);       // テキスト描画
AjcVthPrintF(・・・);
      ：                                  } ①（この間描画したデータは表示されない）
      ：
AjcVthPause(FALSE);         // 表示再開（①の期間に描画したテキストを一気に表示）
      ：
```

AjcVthPause(FALSE) を実行すると、それまで表示を停止していたテキストを一気に表示します。(テキストを１字１句漏らさずに表示＆スクロールするわけではなく、最終描画状態だけを表示します)
AjcVthPause(TRUE)～AjcVthPause(FALSE)の間描画していたテキストはバッファに蓄えられていますので、通常どおりスクロールして見ることができます。

※AjcVthPause(TRUE)～AjcVthPause(FALSE)で表示を抑止している期間でも、ユーザ操作（ウインドのサイズを変えたり、最小化したタスクバーから戻す、等）によりウインドの再描画が必要な場合は、その瞬間の画面状態が表示されます。

7.1.8. 描画処理の高速化

AJCVTHS_FAST スタイルを設定すると、ウインドへの描画処理を高速に実行します。
前述の AjcVthPause()による高速化は、一定期間表示しないことで見かけ上高速に見せる方法でしたが、AJCVTHS_FAST スタイルを設定した場合は、ウインドへの描画処理自体を高速に実行します。
AJCVTHS_FAST スタイルは、極端に短い間隔（数ミリ秒オーダー）で連続してログ表示等を行う場合に有効です。

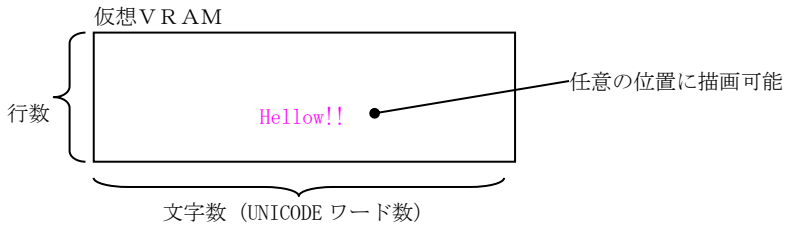
AJCVTHS_FAST スタイルを設定すると、ウインドイメージをＤＩＢで持ち（メモリ上にウインドのビットマップイメージを持ち)、スクロールをメモリ操作で行い、更新された行だけ描画するように制御します。

但し、AJCVTHS_FAST スタイルを設定した場合は、以下の制限があります。

・文字色は、全文字で一律の色となります。(文字単位で別々の色を設定することはできません)
・文字背景色と AJCVTHS_SEPARATE スタイルは無視されます。

7.1.9. ＡＮＳＩエスケープコード

ＡＮＳＩエスケープコードにより、描画メモリ（以降、仮想ＶＲＡＭという）の任意の桁位置（半角は１桁、全角は２桁とする）にカーソルを移動したり、描画色の設定等を行うことができます。(文字単位でグラフィカルな描画が可能です)
仮想ＶＲＡＭの文字数や、行数は任意に設定することができます。



仮想ＶＲＡＭの横サイズは、VramW プロパティ(キャプション文字列の場合は「VW=nnn」)により設定しますが、１行に表示できる文字数は条件により変動します。

・文字は全て UNICODE で格納します。(バイト文字モード時は、S-JIS → UNICODE 変換します)
・VramW プロパティは、横方向の最大格納ワード数を意味します。(１ワード＝２バイト(16 ビット))
・サロゲートペア文字以外の場合は、半角文字／全角文字は区別されずに、それぞれ１ワード分のメモリを消費します。
・全てサロゲートペア以外の文字の場合は、１行に格納できる文字数は VramW プロパティと等しくなります。
・サロゲートペア文字は、格納に２ワード分のメモリを消費します。
・仮に、全てサロゲートペア文字である場合は、１行に格納できる文字数は VramW プロパティの半分となります。

7.1.10. キー操作

以下のキーは、特別な操作を行います。

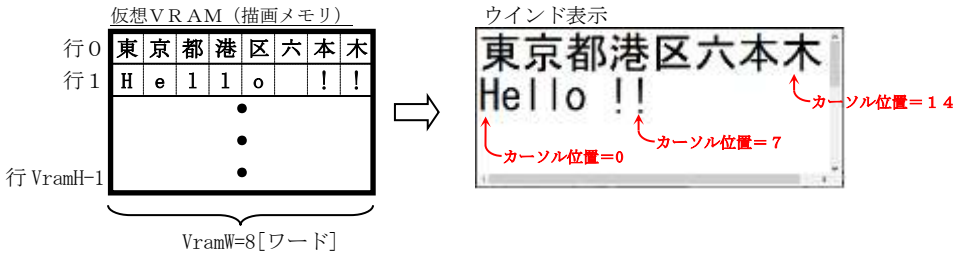
#	キー	操作内容	備考
1	CTRL + C	選択されたテキストのコピー	テキストが選択されている場合のみ
2	CTRL + X	選択状態の解除	
3	Shift+方向キー（↑↓←→)	選択範囲の変更	
4	F1	ポップアップメニュー表示	キーの変更／無効化が可能
5	F3	文字列の下方向検索(Shitft キーが押されている場合は上方向検索)	

7.1.11. 描画メモリとカーソル位置

仮想ＶＲＡＭ（描画メモリ）は、全角／半角にかかわらず、１文字を１ワード(16 ビット)の UNICODE で格納します。
仮想ＶＲＡＭへのテキスト描画は全てカーソル位置から描画します。
カーソル行位置は、仮想ＶＲＡＭ先頭行を０とし、０～VramH － 1 までとなります。(”ESC・[pl;pcH”で指定する pl は１～VRamH)
カーソル桁位置は、行頭を０として、**半角文字は１桁、全角文字は２桁**としてカウントします。
1 行に全て半角文字を描画した場合は、カーソル桁位置は０～VramW-1 の範囲となります。 (”ESC・[pl;pcH”で指定する pc は１～VRamW)
1 行に全て全角文字を描画した場合は、カーソル桁位置は０～(VramW-1) ×２の範囲となります。
カーソル桁位置を全角文字の中心に設定することはできません。

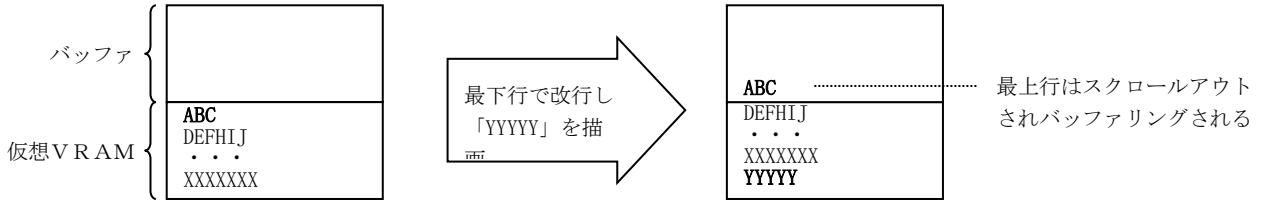
バッファに格納できる文字数は、VramW で制限される(VramW は半角／全角に関わらず１行に格納可能な文字数を意味する)ため、見かけ上、全角文字は半角文字よりも長く表示されます。

例えば、VramW=8 で「東京都港区六本木」（全角）と「Hello !!」（半角）を描画した場合、表示は以下のようになります。



7.1.12. スクロールアウトした行のバッファリング

AJCVTHS_NOSCRLOUT スタイルが設定されていない場合、仮想ＶＲＡＭをスクロールアウトした行（仮想ＶＲＡＭの最下行で改行を行うか最右端に文字を描画すると、最上行はスクロールアウトされる）は、バッファリングされ、スクロールバーでスクロールして見ることができます。
但し、このバッファリングされた行に描画することはできません。(描画は仮想ＶＲＡＭ上でのみ可能です)



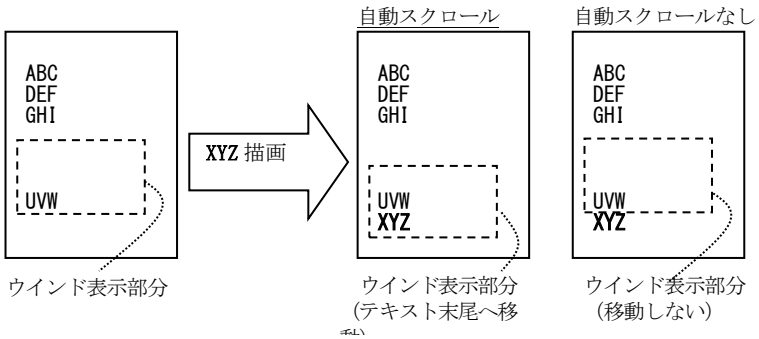
仮想ＶＲＡＭは、所定の行数×文字分の描画用メモリを確保していますが、スクロールアウトされたバッファでは、当該行の有効文字数だけのメモリを確保し、不要なメモリの消費を抑えています。
バッファの最大行数は、MaxLines プロパティで任意に設定可能です。(バッファと仮想ＶＲＡＭ合計の行数で設定します)

AJCVTHS_NOSCRLOUT スタイルが設定されている場合は、スクロールアウトせずに、仮想ＶＲＡＭ 最下行の次は、最上行にカーソルが移動します

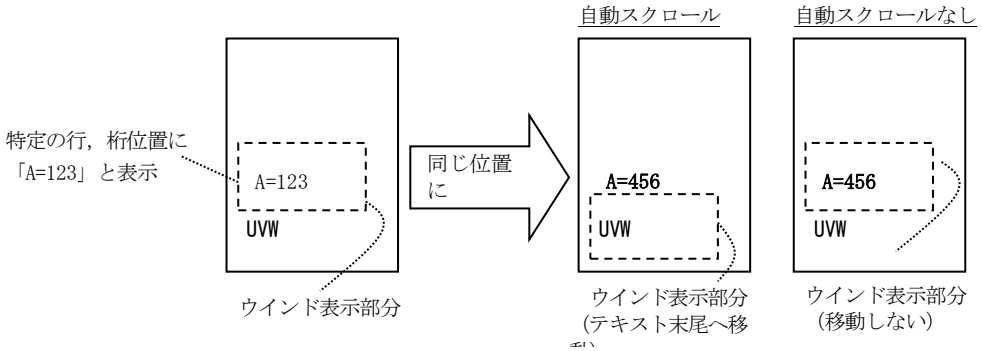


7.1.13. オートスクロール

AJCVTHS_NOSCRL スタイルが設定されていない場合、テキストを表示した際に、データ末尾位置まで自動的にスクロールします。
「オートスクロール」は、連続的なログ表示等で、常に最新の表示部分へ移動する場合に有効です。



AJCVTHS_NOSCRL スタイルが設定されている場合、手動でスクロールしない限り、スクロールを行いません。
「オートスクロールなし」は、例えばスクリーンの固定位置（固定の行，桁位置）に情報を表示する場合に有効です。



7.1.14. ＶＲＡＭとバッファを識別して表示

AJCVTHS_SEPARATE スタイルを指定した場合、バッファ部分の背景を若干暗く表示し、ＶＲＡＭ部分と識別できるようにします。
AJCVTHS_SEPARATE スタイルは、実行時にマウスの中ボタン（ホイールボタン）により設定／解除をトグルすることもできます。



7.1.15. 描画テキストの一時保留

スクロールバーの操作中や、テキストが選択されている場合、描画テキストデータは一時保留（バッファリング）されます。
この一時保留されたテキストデータは、テキストの選択状態が解除された時点、あるいは、スクロール操作を終了した時点で一気に描画されます。

但し、一時保留バッファが満杯になった場合は、強制的にテキストの選択状態が解除され、スクロール操作を終了します。

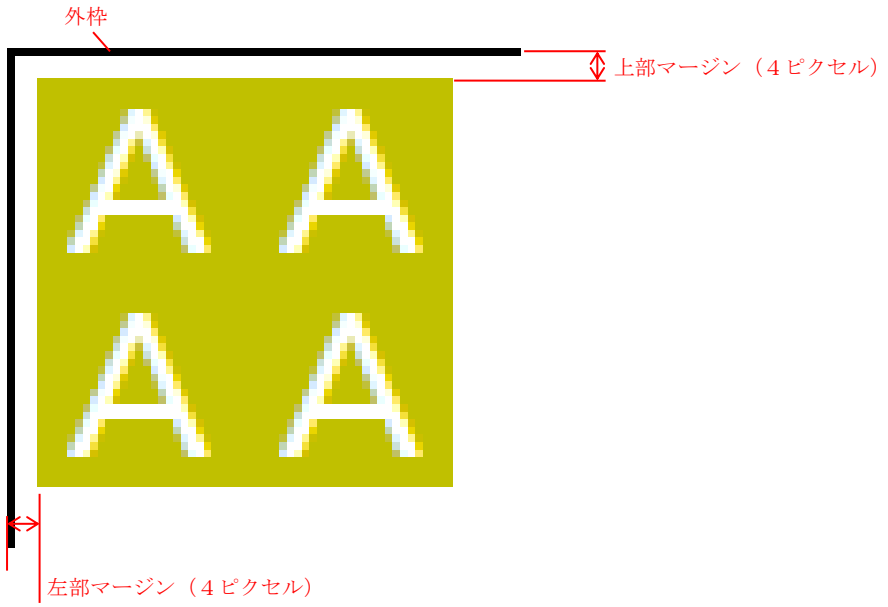
一時保留バッファの容量は、任意に設定できます。

※ 一時保留バッファは、常にバッファメモリを確保しているわけではなく、保留するテキストデータサイズに合わせて増減します。
一時保留状態となった時点でバッファメモリを確保し、保留テキストデータに合わせて４ＫＢ単位で増加します。
一時保留状態が解除された時点で、バッファメモリは開放されます。

7.1.16. マージン

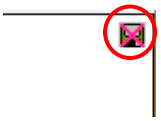
文字がウインドの外枠とくっつかないように、上下左右にマージン（４ピクセル固定）を設けています。
この４ピクセルのマージンは、外枠表示分の１ピクセルを含んでいます。
つまり、外枠を表示している場合、実際のマージンは３ピクセルとなります。

例えば、ＶＴ１００エミュレーションウインドの左上部分は、以下のように表示されます。



7.1.17. 表示内容をファイルへ出力

「AJCVTHS_LOGFILE」スタイルを指定することにより、画面への表示内容をファイルへも出力することができます。
「AJCVTHS_LOGFILE」スタイルを指定すると、ウインド右上に以下のマークが表示されます。



このマークをクリックすると、マークが点滅し、以降の表示内容がファイルに出力されます。
再びクリックすると、ファイルの出力を停止します。
ファイルを出力するフォルダを設定するには、このマークを右クリックします。
出力するファイル名は、「LGF_*yyyy-mm-dd_hh-mm-ss*.log」となります。

尚、エスケープシーケンスはファイル出力しません。

7.1.18. ファイルやディレクトリのドロップ

本コントロールにファイルやディレクトリをドロップした場合は、WM_COMMAND メッセージにて、親ウインドへ以下の通知を行います。

・AJCVTHN_DROPFILE ----- ファイルがドロップされたことを通知
・AJCVTHN_DROPDIR ----- フォルダがドロップされたことを通知

これらの通知では、ドロップされたファイルやディレクトリの個数を通知します。
ファイルやディレクトリのパス名は、本コントロールのＡＰＩ「AjcVthGetDroppedFile(),AjcVthGetDroppedDir[Ex]()」にて取得します。

尚、ＶＴ１００エミュレーションウインド・コントロールでファイルやディレクトリのドラッグ＆ドロップを有効とするには、拡張ウインドスタイルに「WS_EX_ACCEPTFILES」を指定する必要があります。

7.1.19. 固定ピッチ表示

固定ピッチフォントを設定しても、一部の文字が固定ピッチとはならない場合があります。
このような場合、「AJCVTHS_FIXEDPITCH」スタイルを設定することにより、強制的に文字を固定ピッチで表示することができます。
「AJCVTHS_FIXEDPITCH」スタイルを設定した場合、全角文字は全て同一ピッチで表示され、半角文字は全角文字の半分のピッチで表示されます。

以下の例は、固定ピッチフォント(MS ゴシック)を設定して、「AJCVTHS_FIXEDPITCH」スタイルによる表示の違いを示します。

「AJCVTHS_FIXEDPITCH」スタイル未設定



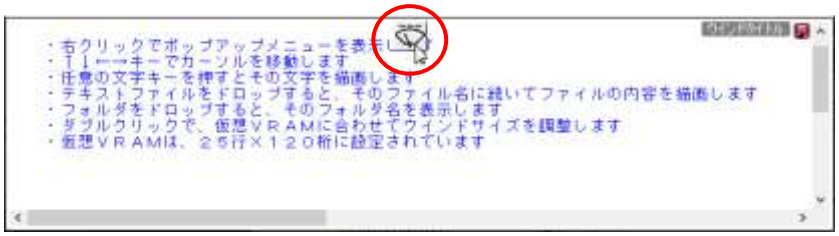
「AJCVTHS_FIXEDPITCH」スタイルを設定



この例では、固定ピッチフォント(MS ゴシック)を設定しても、ハングル文字が固定ピッチとならないことを示しています。
可変ピッチフォントの場合、「AJCVTHS_FIXEDPITCH」スタイルは無視されます。

7.1.20. 画面クリアーボタン

ウインド中央の上部にカーソルを置くと、画面クリアーボタン「  」が表示されます。
このボタンをクリックすると、画面がクリアーされます。



画面クリアーボタンは、「AJCVTHS_NOCLSBTN」スタイルが設定されている場合は表示されません。

7.2. 制御コードとＡＮＳＩエスケープコード

本コントロールでサポートする制御コードとＡＮＳＩエスケープコードは、以下のとおりです。

制御コード

#	制御コード	内 容
1	BS (0x08)	カーソルを左へ移動（カーソル位置が行頭の場合は移動しない）
2	TAB(0x09)	カーソルを右方向へタブステップ数の倍数の位置へ移動
3	LF (0x0A)	改行（最下行以外の場合はカーソルを下へ移動。最下行の場合は１行スクロールアップ）
4	CR (0x0D)	復帰（カーソルを行の先頭へ移動）

ＡＮＳＩエスケープコード

#	ESC シーケンス	内 容
1	ESC・[0J	カーソル位置～最終行の右端までクリアー
2	ESC・[1J	先頭行の左端～カーソル位置までクリアー
3	ESC・[2J ESC・*	画面をクリアーし、カーソルをホームへ移動
4	ESC・[0K ESC・[K	カーソル位置～同行右端までをクリアー
5	ESC・[1K	カーソル行の左端～カーソル位置までをクリアー
6	ESC・[2K	カーソル行をクリアー
7	ESC・[s	カーソル位置を退避
8	ESC・[u	カーソル位置を回復
9	ESC・[>5l	カーソル表示
10	ESC・[>5h	カーソル非表示
11	ESC・[pl;pcH	カーソル位置設定（pl は行位置（１～），pc は桁位置（１～））
12	ESC・[pnA	カーソルを上方向に移動（pn は移動行数であり，省略時は１を仮定）
13	ESC・[pnB	カーソルを下方向に移動（ 〃 〃 ）
14	ESC・[pnC	カーソルを右方向に移動（pn は移動桁数であり，省略時は１を仮定，行末の場合は、次の行頭へ移動）
15	ESC・[pnD	カーソルを左方向に移動（ 〃 〃 ，行頭の場合は、前の行末へ移動）
16	ESC・[pnM	カーソル行以降をスクロールアップ（pn はスクロール行数）
17	ESC・[pnL	カーソル行以降をスクロールダウン（ 〃 ）
18	ESC・[psm	描画属性設定（ps=属性値、0=デフォルト属性，7=反転，30～37(文字色) = 黒,赤,緑,黄,青,紫,水色,白，39:文字色リセット， 40～47(背景色) = 黒,赤,緑,黄,青,紫,水色,白，49:背景色リセット）
19	ESC・D	カーソルを１行下へ移動
20	ESC・E	カーソルを１行下の左端へ移動
21	ESC・M	カーソルを１行上へ移動

［注］「ESC・」はエスケープコード(0x1B)を意味します。

7.3. コントロールのスタイル

ＶＴ１００エミュレーションウインド・コントロールのスタイルは、以下のとおりです。

名称	ビット	値	内容
AJCVTHS_LOGFILE	15	0x8000	ログファイル出力コントロール表示
AJCVTHS_FIXEDPITCH	14	0x4000	固定ピッチで表示
AJCVTHS_FAST	13	0x2000	テキストの描画処理を高速に実行します。　（※１）
AJCVTHS_NOCLSBTN	12	0x1000	画面クリアーボタン（  ）を表示しない
AJCVTHS_FIXEDFONT	11	0x0800	フォント固定（ポップアップメニューにフォント関連の設定を表示しない）
AJCVTHS_FIXEDLF	10	0x0400	改行動作固定（ポップアップメニューに改行動作設定を表示しない）
AJCVTHS_NOVSCROLL	9	0x0200	縦スクロールバー非表示
AJCVTHS_NOHSCROLL	8	0x0100	横スクロールバー非表示
AJCVTHS_NOSCRL	7	0x0080	自動スクロール禁止（ＶＲＡＭへ描画時、最終行へ移動しない）
AJCVTHS_NOBORDER	6	0x0040	コントロールの外枠を表示しない
AJCVTHS_NOSCRLOUT	5	0x0020	スクロールアウト禁止
AJCVTHS_CRLFCTL	4	0x0010	制御コードの CR(0x0D)で復帰，LF(0x0A)で改行を行う　（※２）
AJCVTHS_CRCTRL	3	0x0008	制御コードの CR(0x0D)で復帰と改行を行う　（※２）
AJCVTHS_LFCTRL	2	0x0004	制御コードの LF(0x0A)で復帰と改行を行う　（※２）
AJCVTHS_SEPARATE	1	0x0002	ＶＲＡＭとその他の表示を分ける（ＶＲＡＭ以外の部分は背景色を若干暗くする） AJCVTHS_FAST スタイルが設定されている場合、AJCVTHS_SEPARATE は無視されます。

※１：カーソルの移動が無い場合、描画を高速に実行します。
数ミリ秒オーダーの速い周期で、テキストを連続的に表示する場合に有効です。
サンプルプログラム（SW_VT100_MesTime）で実際の処理時間を測定できます。

※２：これら３ビットがいずれも設定されていない場合は、制御コードの LF(0x0A)で復帰と改行を行います。

尚、ファイルやディレクトリのドラッグ＆ドロップを有効とするには、以下の拡張ウインドスタイルを指定する必要があります。

名称	ビット	値	内容
WS_EX_ACCEPTFILES	4	0x0010	ドラッグ＆ドロップでファイルやディレクトリを受け付けるようにする

7.4. コントロールのプロパティ構造体

ＶＴ－１００エミュレーションウインド・コントロールのプロパティは、以下の構造体で定義されます。

```
typedef struct {
    UI          VramW;                  //  ＶＲＡＭ文字数
    UI          VramH;                  //  ＶＲＡＭ行数
    UI          CaretH;                 //  キャレットの高さ
    UI          MaxLines;               //  最大行数
    UI          TabStep;                //  ＴＡＢステップ数
    UI          LSpace;                 //  行間スペース
    UI          PendSize;               //  最大表示保留サイズ
    COLORREF    rgb[8];                 //  ＲＧＢ色コードテーブル
    LOGFONT     LogFont;                //  フォント情報
} AJCVTHPROP, *PAJCVTHPROP;
typedef const AJCVTHPROP *PCAJCVTHPROP;
```

各メンバ変数の内容は、以下のとおりです。

メンバ	内容	規定値	備考
VramW	仮想ＶＲＡＭの文字数	256	最大 32,000
VramH	仮想ＶＲＡＭの行数	64	
CaretH	カレットの高さ	2	点滅文字カーソルの高さ（**現状は未使用**）　※１
MaxLines	最大行数	10000	バッファと仮想ＶＲＡＭの総行数
TabStep	タブコードによるスキップ幅	4	
LSpace	行間スペース	0	ピクセル数
PendSize	一時保留バッファのサイズ	524288	512KB
rgb[8]	各パレットの色	[0] : 0x000000 [1] : 0x0000FF [2] : 0x00FF00 [3] : 0x00FFFF [4] : 0xFF0000 [5] : 0xFF00FF [6] : 0xFFFF00 [7] : 0xFFFFFF	Bit23-16 : 青の成分 Bit15- 8 : 緑の成分 Bit 7- 0 : 赤の成分
LogFont	フォントデータ	-	FixedSys

※１：カレットのサイズは、幅＝２ピクセル，高さ＝文字の高さ固定で、カーソル位置の文字の左側に表示

7.5. キャプション文字列によるプロパティの設定

CreateWindow()/CreateWindowEx()の lpszWindowName 引数（ウインドキャプション）あるいは、ダイアログデザイン時におけるコントロールの Caption プロパティにより、３Ｄ／２Ｄグラフィック・コントロールのプロパティを設定することができます。

パラメタ（キャプション文字列）の形式は、以下のとおりです。([XXX]は、XXX を省略可能であることを意味します)

```
P:  [VW=nnn], [VH=nnn], [CH=nnn], [ML=nnn], [TS=nnn], [LS=nnn], [PS=nnn], [BC=外枠表示色], [FN=フォント名],
    [LF=[aaa]/[bbb]/[ccc]/[ddd]/[eee]/[fff]/[ggg]/[hhh]/[iii]/[jjj]/[kkk]/[lll]/[mmm]],
    [0=パレット０の色コード], ・・・・・, [7=パレット７の色コード], [FS=SectionName]
```

文字列の先頭は「P:」でなければなりません。(「P:」の直後には空白を置けます)
「nnn」は整数（１６進数の場合は先頭に'0x'を付加）で指定します。
各パラメタはカンマ（,）で区切ります。(カンマの前後には空白を置けます)
各パラメタの指定順序は任意です。

各パラメタの設定内容は、以下のとおりです。

キーワード	内　容
VW	仮想ＶＲＡＭ１行当たりの文字数
VH	仮想ＶＲＡＭの行数
CH	カレットの高さ
ML	最大行数（バッファと仮想ＶＲＡＭの総行数）
TS	タブコードによるスキップ幅
LS	行間スペース
PS	一時保留バッファのサイズ
FN	フォントフェース名（LOGFONT 構造体の lfFaceName の内容）
BC	コントロール外枠の表示色（0xbbggrr）
LF	フォントデータ（lfFaceName を除く LOGFONT 構造体の各メンバ変数の値）
0 ～ 7	各パレットの色コード(0x00bbggrr)
FS	検索情報永続化用セクション名を指定します（内部で「AjcVthSetFindProfileSect()」が実行されます）

表示色は、１６進数で「0xbbggrr」の形式で指定します（bb:青成分，gg:緑成分，rr:赤成分）

設定例

```
P: VW=256, VH=512
```

仮想ＶＲＡＭを２５６桁×５１２行とする

```
P: FN=ＭＳ ゴシック, LF=12
```

フォントを高さ１２ピクセルのＭＳゴシックとする

7.6. テキストの取得と設定

テキストを取得した場合は、プロパティ設定内容を表す文字列を返します。
デフォルトでの、取得テキストは、以下のとおりです。

```
P: VW=256, VH=64, CH=2, ML=10000, TS=4, LS=0, PS=524288, FN=FixedSys, BC=0x0, LF=0/0/0/0/0/0/0/0/1/0/0/0/1,
 0=0x0, 1=0xFF, 2=0xC000, 3=0xC0C0, 4=0xFF0000, 5=0xC000C0, 6=0xC0C000, 7=0xFFFFFF
```

テキストを設定する場合は、キャプション文字列によるプロパティの設定と同様に扱います。

7.7. WM_SETFONT について

WM_SETFONT メッセージにてフォントを設定する場合は、同じ WM_SETFONT を２回実行してください。

本コントロールは、WM_SETFONT メッセージによるフォントの設定が可能ですが、キャプション文字列でフォントが指定されている場合（「FN=XXX」や「LF=XXX」が指定されている場合）、最初の WM_SETFONT を無視します。
これは、本コントロールをダイアログ項目としている場合、ダイアログから WM_SETFONT によるフォント設定が発行されますが、キャプション文字列でもフォント情報が指定されている場合、キャプション文字列で指定されているフォントが優先されます。
キャプション文字列でフォント情報が指定されていない場合は、WM_SETFONT によるフォント設定が有効となります。
つまり・・・

- キャプション文字列でフォント情報を指定していない場合は、ダイアログと同じフォントが設定されます。
- キャプション文字列でフォント情報を指定している場合は、キャプション文字列で指定した内容が有効となります。
- WM_SETFONT メッセージでフォントを設定する場合は（少なくとも初回は）同じ WM_SETFONT を２回実行してください。

7.8. プロパティの永続化

設定したプロパティを、プロファイル（.ini ファイル／レジストリ）に記録し、次回起動時に記録されているプロファイルを読み出すことにより、プロパティを永続的に有効とすることができます。

プロパティをプロファイルから読み出すには、ダイアログやウインドの初期化時に、AjcVthLoadProp()を実行します。
尚、初回実行時はプロファイルにプロパティが記録されていない為、AjcVthLoadProp()でプロパティのデフォルト値を指定します。
AjcVthLoadProp()でプロパティのデフォルト値を指定しない場合は、現在設定されているプロパティがデフォルト値となります。

デフォルトプロパティを指定する場合

```
AJCVTHPROP DefProp;

DefProp.rgb[0] = RGB(128, 128, 128);
   ・・・
AjcVthLoadProp(hwnd, "SectName", &DefProp);
```

デフォルトプロパティを指定しない場合

```
AjcVthSetColor(hwnd, 0, RGB(128, 128, 128));
AjcVthLoadProp(hwnd, "SectName", NULL);
```

プロファイルにプロパティが記録されている場合は、AjcVthLoadProp()により読み出されたプロパティが設定される為、AjcVthSetColor()により設定された値は無効となります。

現在設定されているプロパティをプロファイルに記録するには、ダイアログやウインドの終了時に、AjcVthSaveProp()を実行します。

プロパティをプロファイルに記録

```
AjcVthSaveProp(hwnd, "SectName");
```

デフォルトでは、プロファイルの記録先は、レジストリになります。
プロファイルの記録先を初期化ファイル（自プログラムパス名の拡張子を「.ini」としたファイル）とする場合は、プログラムの最初（AjcVthLoadProp(), AjcVthSaveProp()を実行する前）に「AjcSetProfileIsRegistory(FALSE);」を実行してください。

プロファイルへのアクセスは、AjcGetProfile･･･() と AjcPutProfile･･･() により行います。
これらの関数仕様やプロファイルアクセスに関するその他の情報は、「プロファイル・アクセス」章を参照してください。

7.9. サポートＡＰＩ

ＶＴ－１００エミュレーションウインド・コントロールのサポートＡＰＩ一覧を以下に示します。

#	関数名	内容	備考
1	AjcVth{Set/Get}Prop	プロパティ設定／取得	
2	AjcVthResetProp	プロパティ・リセット	ウインド生成時の状態
3	AjcVthGetVramFitSize	ＶＲＡＭにフィットしたウインドサイズ取得	
4	AjcVthPutChar	１文字描画	
5	AjcVthPutText	テキスト描画	UNICODE は S-JIS に変換
6	AjcVthPutTextEUC	日本語ＥＵＣコードテキスト描画	S-JIS に変換して描画
7	AjcVthPutTextUTF8	ＵＴＦ－８コードテキスト描画	
8	AjcVthPutTextAuto	文字コードを自動判別してテキスト描画	
9	AjcVthPrintF	書式テキスト描画	
10	AjcVthTimeStamp	タイムスタンプ描画	
11	AjcVthHexDump	バイナリデータの１６進ダンプ描画	
12	AjcVthLocate AjcVthGetCursorPos	カーソル位置設定／取得	
13	AjcVthSetColor AjcVthGetColor	テキスト描画用パレット選択／選択パレット番号取得	
14	AjcVthSetBkColor AjcVthGetBkColor	文字背景描画用パレット選択／選択パレット番号取得	
15	AjcVthSetPalette AjcVthGetPalette	パレットの色コード設定／取得	
16	AjcVth{Set/Get}WndBkColor	ウインドのの背景色設定／取得	
17	AjcVthSelect	部分テキストを選択	
18	AjcVthSelectAll	全てのテキストを選択	
19	AjcVthCopyText	選択テキストをクリップボードへコピー	
20	AjcVthSetFont	ダイアログによるフォントの設定	
21	AjcVthShowCaret	キャレット表示／非表示	
22	AjcVthClear[AllText]	全テキストクリアー	
23	AjcVthGetDroppedFile	ドロップされたファイル名取得	
24	AjcVthGetDroppedDir[Ex]x	ドロップされたディレクトリ名取得	
25	AjcVthSetNtcRClk	右クリック通知設定	
26	AjcVthEnablePopupMenu	ポップアップメニューの許可／禁止	
27	AjcVth{Sel/Exc}MenuItems	ポップアップメニュー項目の有効化／無効化	
28	AjcGetText	テキストの取得	
29	AjcVthIsSelected	テキスト選択状態の取得	
30	AjcVthGetSelectedText	選択されているテキストの取得	
31	AjcVthGetDblClickedLine	ダブルクリックした行位置のテキスト取得	
32	AjcVthSetTipText AjcVthGetTipText	ツールチップの設定／取得	
33	AjcVthSetTipShowAlways AjcVthGetTipShowAlways	ツールチップ表示条件の設定／取得	
34	AjcVth{Set/Get}FontInfo AjcVthSetFontHandle	フォント，行間スペースの設定／取得	
35	AjcVthGetCharInfo	文字サイズ／行の高さ取得	
36	AjcVthGetLinesPerWindow	ウインドに表示可能な行数の取得	
37	AjcVthGetValidLines	バッファに格納されている有効な行数の取得	
38	AjcVthGetIxOfWndTopLine	ウインド先頭行の位置取得（０～）	
39	AjcVthSetTitleText	タイトル文字列の設定	
40	AjcVth{Load/Save}Prop	プロファイルからプロパティ値読出し／書き込み	

#	関数名	内容	備考
42	AjcVthSaveHtmlToFile	テキストをＨＴＭＬファイルへ書き込み	
43	AjcVthSaveAllTextWithEsc	ＥＳＣシーケンスを付加し全テキストをファイルへ書込み	
44	AjcVthGetLineCount	バッファに格納されている行数の取得	
45	AjcVthGetCursorPosInfo	マウスカーソル位置の行番号と文字位置取得	
46	AjvVthGetLineText	指定行位置の行テキスト取得	
47	AjcVth{Set/Get}VScrollPos	縦スクロール位置の設定／取得	
48	AjcVth{Set/Get}HScrollPos	横スクロール位置の設置／取得	
49	AjcVthGetWindowSize	表示ウインドサイズ（行数，文字数）取得	
50	AjcVthSearchBelow AjcVthSearchAbove	文字列の検索	
51	AjcVthPause	画面表示の停止／再開	
52	AjcVthLoadPermInfo[Ex] AjcVthSavePermInfo[Ex]	設定情報の永続化	
53	AjcVthSetFindProfileSect	文字列検索情報永続化プロファイルセクション名設定	検索情報の永続化
54	AjcVthSetFindKey	文字列検索キーの設定	
55	AjcVthSetMenuKey	ポップアップメニュー表示キーの設定	
56	AjcVthSetEvtCallBack	イベント通知用コールバック関数の設定	

7.9.1. プロパティ設定／取得(AjcVth{Set/Get}Prop)

形　式 ： BOOL　AjcVthSetProp(HWND hwnd, PCAJCVTHPROP pProp)；
BOOL　AjcVthGetProp(HWND hwnd, PAJCVTHPROP　pBuf)；

引　数 ： hwnd　- コントロールのウインドハンドル
pProp - 設定するプロパティデータのアドレス
pBuf　- 取得したプロパティデータを格納するバッファのアドレス

説　明 ： プロパティの設定／取得を行います。
プロパティデータについては「ＶＴ－１００エミュレーションウインド・コントロールのプロパティ構造体」を参照。

戻り値 ： TRUE　- 成功
FALSE - 失敗

7.9.2. プロパティ・リセット(AjcVthResetProp)

形　式 ： BOOL　AjcVthResetProp(HWND hwnd)；

引　数 ： hwnd　- コントロールのウインドハンドル

説　明 ： プロパティをウインド生成時の状態に戻します。
つまり、ウインド生成時にキャプションで指定したプロパティを再設定します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

7.9.3. ＶＲＡＭにフィットしたウインドサイズ取得(AjcVthGetVramFitSize)

形　式 ： BOOL　AjcVthGetVramFitSize(HWND hwnd, UIP pWidth, UIP pHeight);

引　数 ： hwnd　　- コントロールのウインドハンドル
pWidth　- ウインドの幅（ピクセル数）を格納するバッファのアドレス
pHeight - ウインドの高さ（ピクセル数）を格納するバッファのアドレス

説　明 ： 現在設定されている仮想ＶＲＡＭの桁数と行数にフィットした（仮想ＶＲＡＭ全体を表示可能な）コントロールのウインドサイズを取得します。
ウインドの幅は、全て半角文字を描画した場合のサイズとなります。
可変ピッチフォントの場合は、半角文字の平均的な文字幅を元に計算します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

7.9.4. １文字描画(AjcVthPutChar)

形　式 ： BOOL　AjcVthPutChar (HWND hwnd, UI c);

引　数 ： hwnd　- コントロールのウインドハンドル
c　　- 描画する文字（バイトモード時は８bit, UNICODE 時は１６bit）
lTxt　- 描画する文字列の文字数（－１の場合は自動算出）

説　明 ： バイトモード時は１バイトを、UNICODE 時は１文字を描画します。
バイトモード時、２バイト文字を描画するには、２回呼び出します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

7.9.5. テキスト描画 (AjcVthPutText)

形　式 ： BOOL　AjcVthPutText(HWND hwnd, C_UTP pTxt, UI lTxt);

引　数 ： hwnd　- コントロールのウインドハンドル
pTxt　- 描画する文字列のアドレス（バイトモードの場合はS-JIS文字列）
lTxt　- 描画する文字列のバイト数／文字数（－１の場合は自動算出）

説　明 ： pTxtとlTxtで指定された文字列を、ＶＴ－１００エミュレーションウインドへ描画します。
lTxt=-1とした場合は、文字列のバイト数／文字数を自動的に算出します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

7.9.6. 日本語ＥＵＣコードテキスト描画 (AjcVthPutTextEUC)

形　式 ： BOOL　AjcVthPutTextEUC(HWND hwnd, C_UBP pTxt, UI lTxt);

引　数 ： hwnd　- コントロールのウインドハンドル
pTxt　- 描画する日本語ＥＵＣコード文字列のアドレス
lTxt　- 描画する文字列のバイト数／文字数（－１の場合は自動算出）

説　明 ： pTxtとlTxtで指定されたＥＵＣコード文字列をＶＴ－１００エミュレーションウインドへ描画します。
文字列は、内部的にシフトＪＩＳに変換されます。
lTxt=-1とした場合は、文字列のバイト数／文字数を自動的に算出します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

7.9.7. ＵＴＦ－８コードテキスト描画 (AjcVthPutTextUTF8)

形　式 ： BOOL　AjcVthPutTextUTF8(HWND hwnd, C_UBP pTxt, UI lTxt);

引　数 ： hwnd　- コントロールのウインドハンドル
pTxt　- 描画するＵＴＦ－８コード文字列のアドレス
lTxt　- 描画する文字列のバイト数／文字数（－１の場合は自動算出）

説　明 ： pTxtとlTxtで指定されたＥＴＦ－８コード文字列をＶＴ－１００エミュレーションウインドへ描画します。
バイト文字モードの場合、文字列は内部的にシフトＪＩＳに変換されます。
UNICODEモードの場合、文字列は内部的にユニコードに変換されます。
lTxt=-1とした場合は、文字列のバイト数／文字数を自動的に算出します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

7.9.8. 文字コードを自動判別してテキスト描画 (AjcVthPutTextAuto)

形　式 ： BOOL　AjcVthPutTextAuto(HWND hwnd, C_UBP pTxt, UI lTxt);

引　数 ： hwnd　- コントロールのウインドハンドル
pTxt　- 描画する文字列のアドレス
lTxt　- 描画する文字列のバイト数／文字数（－１の場合は自動算出）

説　明 ： pTxtとlTxtで指定された文字列をＶＴ－１００エミュレーションウインドへ描画します。
文字コード（シフトＪＩＳ，日本語ＥＵＣ，ＵＴＦ－８）は自動判別して描画します。
lTxt=-1とした場合は、文字列のバイト数／文字数を自動的に算出します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

7.9.9. 書式テキスト描画(AjcVthPrintF)

形　式 ： BOOL　AjcVthPrintF(HWND hwnd, C_UTP pFmt, ...);

引　数 ： hwnd　- コントロールのウインドハンドル
pFmt　- 書式文字列のアドレス

説　明 ： pFmt で書式化された文字列を、ＶＴ－１００エミュレーションウインドへ描画します。
書式文字列の形式は、printf()と同じです。
但し、書式化後の文字列の文字数は最大２０４７バイト／文字であり、超えた部分はカットされます。

戻り値 ： TRUE　- 成功
FALSE - 失敗

7.9.10. タイムスタンプ描画(AjcVthTimeStamp)

形　式 ： BOOL　AjcVthTimeStamp(HWND hwnd);

引　数 ： hwnd　- コントロールのウインドハンドル

説　明 ： 現在時刻を、ＶＴ－１００エミュレーションウインドへ描画します。
時刻文字列の形式は、"hh:mm:ss.mmm" です。

戻り値 ： TRUE　- 成功
FALSE - 失敗

7.9.11. バイナリデータの１６進ダンプ描画(AjcVthHexDump)

形　式 ： BOOL　AjcVthHexDump(HWND hwnd, C_VOP pDat, UI lDat);

引　数 ： hwnd　- コントロールのウインドハンドル
pDat　- バイナリデータのアドレス
lDat　- バイナリデータのバイト数

説　明 ： バイナリデータをバイト単位の１６進表現でダンプ表示します。
各バイトの直前に１ヶの空白を挿入します。

```
例：　　UB　dat[] = {15, 16, 17};
　　AjcVthHexDump(hwnd, dat, 3);　// ---> " 0F 10 11"
```

戻り値 ： TRUE　- 成功
FALSE - 失敗

7.9.12. カーソル位置設定／取得(AjcVthLocate / AjcVthGetCursorPos)

形　式 ： BOOL　AjcVthLocate　　(HWND hwnd, UI　line,　UI　 col); -- カーソル位置設定
UI　　AjcVthGetCursorPos (HWND hwnd, UIP pLine, UIP pCol); -- カーソル位置取得

引　数 ： hwnd　- コントロールのウインドハンドル
line　- 設定するカーソルの行位置　（０～）
col　 - 設定するカーソルの文字位置（０～）
pLine - 取得したカーソル行位置（０～）を格納するバッファのアドレス（不要時はNULL）
pCol　- 取得したカーソル桁位置（０～）を格納するバッファのアドレス（不要時はNULL）

説　明 ： AjcVthLocate()は、文字カーソルを line と col で指定された桁位置に設定します。
全角文字は２桁としてカウントします。全角文字の中央にカーソル位置を設定することはできません。
line が仮想ＶＲＡＭの行数範囲を超える場合は、最終行位置に設定されます。
col が仮想ＶＲＡＭの桁数範囲を超える場合は、最終桁位置に設定されます。
AjcVthPutText()やAjcVthPrintF()で文字列を描画する場合は、このカーソル位置から描画されます。
AjcVthGetCursorPos()は、現在のカーソル位置を取得します。

戻り値 ： 設定時：TRUE　- 成功　　取得時：カーソルの行位置（pLine で指定されたバッファに格納した値と同じ）
FALSE - 失敗　　エラーの場合は、０を返します。

7.9.13. テキスト描画用パレット選択／選択パレット番号取得(AjcVth{Set/Get}Color)

形　式 ： BOOL　AjcVthSetColor(HWND hwnd, UI PaletteNo); -- テキスト描画用パレット選択
UI　　AjcVthGetColor(HWND hwnd); ---------- テキスト描画用パレット番号取得

引　数 ： hwnd　　- コントロールのウインドハンドル
PaletteNo - パレット番号（０～７）

説　明 ： テキストの描画色をパレット番号で指定／パレット番号を取得します。
AjcVthPutText()やAjcVthPrintF()で文字列を描画する場合は、当該パレット番号の描画色で描画されます。
高速描画モード(AJCVTHS_FAST スタイル)の場合、全文字で一律の色となります。(文字単位で別々の色を設定することはできません)

戻り値 ： 設定時：TRUE　- 成功　　取得時：現在設定されているテキストの描画パレット番号（０～７）
FALSE - 失敗　　エラーの場合は、０を返します。

7.9.14. 文字背景色用パレット選択／選択パレット番号取得(AjcVth{Set/Get}BkColor)

形　式 ： BOOL　AjcVthSetBkColor(HWND hwnd, UI PaletteNo);　- 背景描画用パレット選択
UI　AjcVthGetBkColor(HWND hwnd); ---------- 背景描画用パレット番号取得

引　数 ： hwnd　　- コントロールのウインドハンドル
PaletteNo - パレット番号（０～７）

説　明 ： テキストの背景描画色をパレット番号で設定／パレット番号を取得します。
AjcVthPutText()やAjcVthPrintF()で文字列を描画する場合は、当該パレット番号の描画色で文字の背景が描画されます。
高速描画モード(AJCVTHS_FAST スタイル)の場合、文字背景色は無視されます。

戻り値 ： 設定時：TRUE　- 成功　　取得時：現在設定されているテキスト背景の描画パレット番号（０～７）
FALSE - 失敗　　エラーの場合は、０を返します。

7.9.15. パレットの色コード設定／取得(AjcVth{Set/Get}Palette)

形　式 ： BOOL　　AjcVthSetPalette(HWND hwnd, UI PaletteNo, COLORREF rgb)； -- パレットの色コード設定
COLORREF AjcVthGetPalette(HWND hwnd, UI PaletteNo)； ---------- パレットの色コード取得

引　数 ： hwnd　　- コントロールのウインドハンドル
PaletteNo - パレット番号（０～７）
rgb　　- 設定するパレットの色コード

説　明 ： PaletteNo で指定されたパレット番号の色コードを設定/取得します。
パレットの色コードを変更した場合は、既に描画済のテキストも、設定された色コードで表示されます。

戻り値 ： 設定時：TRUE　- 成功　　取得時：指定されたパレット番号の色コード
FALSE - 失敗　　エラーの場合は、０を返します。

7.9.16. ウインドの背景色設定／取得(AjcVth{Set/Get}WndtBkColor)

形　式 ： BOOL　　AjcVthSetFastBkColor(HWND hwnd, UI PaletteNo)； -- ウインド背景色設定
UI　　AjcVthGetFastBkColor(HWND hwnd)； ---------- ウインド背景色取得

引　数 ： hwnd　　- コントロールのウインドハンドル
PaletteNo - 設定するウインド背景色のパレット番号

説　明 ： ウインドの背景色をパレット番号で設定／パレット番号を取得します。

戻り値 ： 設定時：TRUE　- 成功　　取得時：ウインド背景色のパレット番号
FALSE - 失敗　　エラーの場合は、０を返します。

7.9.17. 部分テキストを選択 (AjcVthSelect)

形　式 ： BOOL　AjcVthSelectAll(HWND hwnd , UI slp, UI scp, UI elp, UI ecp)；

引　数 ： hwnd　　- コントロールのウインドハンドル
slp, scp　- 選択開始位置（行位置（０～），桁位置（０～））
elp, ecp　- 選択終端位置（行位置（０～），桁位置（０～））

説　明 ： テキスト指定部分を選択状態にします。

戻り値 ： TRUE　- 成功
FALSE - 失敗

7.9.18. 全てのテキストを選択 (AjcVthSelectAll)

形　式 ： BOOL　AjcVthSelectAll(HWND hwnd)；

引　数 ： hwnd　　- コントロールのウインドハンドル

説　明 ： 描画済の全テキストを選択状態にします。

戻り値 ： TRUE　- 成功
FALSE - 失敗

7.9.19. 選択テキストをクリップボードへコピー (AjcVthCopyText)

形　式 ： BOOL AjcVthCopyText(HWND hwnd);

引　数 ： hwnd　　－ コントロールのウインドハンドル

説　明 ： 選択されているテキストをクリップボードへコピーします。
選択状態は解除されます。

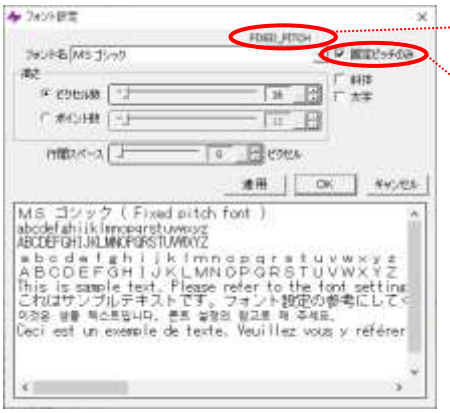
戻り値 ： TRUE　－ 成功
FALSE － 失敗

7.9.20. ダイアログによるフォント設定 (AjcVthSetFont)

形　式 ： BOOL AjcVthSetFont(HWND hwnd);

引　数 ： hwnd　　－ コントロールのウインドハンドル

説　明 ： 下図のダイアログにより、フォントの設定を行います。



固定ピッチのフォントが選択されていることを示します。
可変ピッチのフォントが選択されている場合は、「VARIABLE_PITCH」と表示します。
「固定ピッチのみ」をチェックした場合は、固定ピッチフォントだけが選択可能となります。

戻り値 ： TRUE　－ 成功（ＯＫボタン押下）
FALSE － キャンセルボタン押下

7.9.21. キャレット表示／非表示 (AjcVthShowCaret)

形　式 ： BOOL AjcVthShowCaret(HWND hwnd, BOOL fShow);

引　数 ： hwnd　　－ コントロールのウインドハンドル
fShow　　－ 表示／非表示フラグ（TRUE:表示, FALSE:非表示）

説　明 ： キャレット（点滅文字カーソル）を表示／非表示します。

戻り値 ： TRUE　－ 成功
FALSE － 失敗

7.9.22. 全テキストクリアー (AjcVthClear[AllText])

形　式 ： BOOL AjcVthClear　　(HWND hwnd);
BOOL AjcVthClearAllText (HWND hwnd);

引　数 ： hwnd　　－ コントロールのウインドハンドル

説　明 ： 描画済の全テキストをクリアーします。
スクロールアウトでバッファリングされたテキストも全てクリアーされます。
AjcVthClear()と、AjcVthClearAllText()は同じです。

戻り値 ： TRUE　－ 成功
FALSE － 失敗

7.9.23. ドロップされたファイル名取得(AjcVthGetDroppedFile)

形　式 ： BOOL AjcVthGetDroppedFile (HWND hwnd, UT buf[MAX_PATH]);

引　数 ： hwnd　　- コントロールのウインドハンドル
buf　　- ファイルのパス名を格納するバッファのアドレス

説　明 ： ドラッグ＆ドロップされたファイルのパス名を取得します。
本関数は、通知メッセージ（AJCVTHN_DROPFILE）が通知された場合、ドロップされたファイルのパス名を取得するために実行します。
本ＡＰＩにより全てのドロップされたファイル名を取得済の場合は、FALSE を返します。

戻り値 ： TRUE - ファイルのパス名をバッファに格納した
FALSE - ドロップされたファイル名なし（ドロップしたファイル数を超えて実行された）

7.9.24. ドロップされたディレクトリ名取得(AjcVthGetDroppedDir[Ex])

形　式 ： BOOL AjcVthGetDroppedDir (HWND hwnd, UT buf[MAX_PATH]);
BOOL AjcVthGetDroppedDirEx (HWND hwnd, UT buf[MAX_PATH], BOOL fTailIsDelimiter);

引　数 ： hwnd　　- コントロールのウインドハンドル
buf　　- ファイルのパス名を格納するバッファのアドレス
fTailIsDelimiter - TRUE ：ディレクトリパス名の末尾に「¥」を付加する
FALSE：ディレクトリパス名の末尾に「¥」を付加しない

説　明 ： ドラッグ＆ドロップされたディレクトリのパス名を取得します。
本関数は、通知メッセージ（AJCVTHN_DROPDIR）が通知された場合、ドロップされたディレクトリのパス名を取得するために実行します。
本ＡＰＩにより全てのドロップされたディレクトリ名を取得済の場合は、FALSE を返します。

戻り値 ： TRUE - ディレクトリのパス名をバッファに格納した
FALSE - ドロップされたディレクトリ名なし（ドロップしたディレクトリ数を超えて実行された）

7.9.25. 右クリック通知設定 (AjcVthSetNtcRClk)

形　式 ： BOOL AjcVthSetNtcRClk(HWND hwnd, BOOL fNtcRClk, UI MsgRBDown, UI MsgRBUp);

引　数 ： hwnd　　- コントロールのウインドハンドル
fNtcRClk　　- 右ボタンの DOWN/UP 通知フラグ（TRUE:通知する, FALSE:通知しない）
MsgRBDown　　- 右ボタン押下時の通知メッセージコード　（０の場合は非通知）
MsgRBUp　　- 右ボタン離され時の通知メッセージコード（０の場合は非通知）

説　明 ： コントロールを右クリックした場合に、当該操作を親ウインドへ通知するか否かを設定します。
MsgRBDown, MsgRBUp は、WM_USER+100 以降, WM_APP+500 以降か、RegisterWindowMessage()で取得したコードを指定します。
各引数と、右クリック通知動作は以下のとおりです。

引数			Shift/Ctrl	メッセージ	wParam	備考
fNtcRClk	MsgRBDown	MsgRBUp				
FALSE (default	－	－	未押下	WM_COMMAND	－（非通知）	ポップアップメニュー表示
			押下		ID + AJCVTHN_RCLICK	
TRUE	いずれかが０以外		－	MsgRBDown(押下時)	WM_RBUTTONDOWN の wParam	MsgRBDown ＝０の場合は非通知
				MsgRBUp（離し時）	WM_RBUTTONUP　の wParam	MsgRBUp　＝０の場合は非通知
	0	0	－		－（非通知）	－

右クリックの通知を禁止するには、fNtcRClk=TRUE, MsgRBDown=0, MsgRBUp=0 とします。

戻り値 ： TRUE - 成功
FALSE - 失敗

7.9.26. ポップアップメニューの許可／禁止 (AjcVthEnablePopupMenu)

形　式 ： BOOL AjcVthEnablePopupMenu (HWND hwnd, BOOL fEnable);

引　数 ： hwnd　　- コントロールのウインドハンドル
fEnable　　- ポップアップメニューの許可(TRUE)／禁止(FALSE)

説　明 ： 右クリックによるポップアップメニューを許可／禁止します。

戻り値 ： TRUE - 成功
FALSE - 失敗

備　考 ： 「AjcVthSetNtcRClk(hwnd, !fEnable, WM_RBUTTONDOWN, WM_RBUTTONUP);」を実行します。

7.9.27. ポップアップメニュー項目の有効化／無効化 (AjcVth{Sel/Exc}MenuItems)

形　式 ： BOOL AjcVthSelMenuItems(HWND hwnd, UI sel); - 有効化
BOOL AjcVthExcMenuItems(HWND hwnd, UI exc); - 無効化

引　数 ： hwnd　　- コントロールのウインドハンドル
sel　　- 有効とするメニュー項目群（０　　：右クリックによるポップアップメニュー禁止）
exc　　- 無効とするメニュー項目群(AJCVTHMM_ALL： 右クリックによるポップアップメニュー禁止)

説　明 ： 右クリックによるポップアップメニュー項目を有効化／無効化します。
sel/exc には以下のシンボルの合成値を指定します。

#	シンボル	メニュー項目
1	AJCVTHMM_SELALL	全て選択
2	AJCVTHMM_COPYTXT	コピー
3	AJCVTHMM_CLEAR	全てクリアー
4	AJCVTHMM_FIND	文字列検索、上検索、下検索
5	AJCVTHMM_LINEFEED	復帰改行の設定
6	AJCVTHMM_CARET	キャレット表示／非表示
7	AJCVTHMM_SAVE	ファイルへセーブ
8	AJCVTHMM_SETFONT	フォント設定
9	AJCVTHMM_SETDEFFONT	デフォルトフォント設定
10	AJCVTHMM_COPYFONT	フォント行間スペース コピー
11	AJCVTHMM_PASTEFONT	フォント行間スペース 貼り付け
12	AJCVTHMM_THROWFONT	フォントを全スレッドウインド／兄弟ウインドに適用
13	AJCVTHMM_SETOTHER	その他の設定
14	AJCVTHMM_ALL	全項目

ex. AjcVthExcMenuItems(hwnd, AJCOPT2(AJCVTHMM_, CARET, SETOTHER)); // キャレット表示／非表示、その他の設定をメニューから除外

戻り値 ： TRUE - 成功
FALSE - 失敗

7.9.28. テキストの取得(AjcVthGetText)

形　式 ： UI AjcVthGetText (HWND hwnd, UTP pBuf, UI lBuf);

引　数 ： hwnd　　- コントロールのウインドハンドル
pBuf　　- 取得したテキストを格納するバッファのアドレス
lBuf　　- 取得したテキストを格納するバッファの文字数

説　明 ： pBuf≠NULL かつ lBuf≠0 の場合は、現在格納されているテキストを取得します。
lBuf で示されるバッファサイズが、現在格納されているテキストのサイズより小さい場合は、格納テキストの先頭部分だけが格納されます。
pBuf＝NULL あるいは lBuf＝0 の場合は、格納テキストを取得するのに必要なバッファのサイズ（文字数）を取得します。

戻り値 ： pBuf≠NULL && lBuf≠0 の場合 - バッファに格納したテキストの文字数（文字列の終端(0x00)を含まない）
pBuf＝NULL || lBuf＝0 の場合 - テキストを取得するのに必要なバッファの文字数（文字列の終端(0x00)を含む）
エラーの場合は、０を返します。

7.9.29. テキスト選択の取得(AjcVthIsSelected)

形　式 ： BOOL AjcVthIsSelected (HWND hwnd;

引　数 ： hwnd　　- コントロールのウインドハンドル

説　明 ： テキストの選択状態を取得します。

戻り値 ： TRUE - テキストが選択されている
FALSE - テキストは選択されていない

7.9.30. 選択されているテキストの取得(AjcVthGetSelectedText)

形　式 ： UI AjcVthGetSelectedText (HWND hwnd, UTP pBuf, UI lBuf);

引　数 ： hwnd　　- コントロールのウインドハンドル
pBuf　　- 取得した選択テキストを格納するバッファのアドレス
lBuf　　- 取得した選択テキストを格納するバッファの文字数

説　明 ： pBuf≠NULL かつ lBuf≠0 の場合は、現在選択されているテキストを取得します。
lBuf で示されるバッファサイズが、現在選択されているテキストのサイズより小さい場合は、選択テキストの先頭部分だけが格納されます。
pBuf＝NULL あるいは lBuf＝0 の場合は、選択テキストを取得するのに必要なバッファのサイズ（文字数）を取得します。

戻り値 ： pBuf≠NULL && lBuf≠0 の場合 - バッファに格納したテキストの文字数（文字列の終端(0x00)を含まない）
pBuf＝NULL || lBuf＝0 の場合 - テキストを取得するのに必要なバッファの文字数（文字列の終端(0x00)を含む）
エラーの場合は、０を返します。

7.9.31. ダブルクリックした行位置のテキスト取得(AjcVthGetDblClickedLine[Ex])

形　式 ： UI AjcVthGetDblClickedLine (HWND hwnd, UTP pBuf, UI lBuf);
UI AjcVthGetDblClickedLineEx(HWND hwnd, UTP pBuf, UI lBuf, UIP pLine, UIP pCol);

引　数 ： hwnd　　- コントロールのウインドハンドル
pBuf　　- 取得した行テキストを格納するバッファのアドレス
lBuf　　- 取得した行テキストを格納するバッファの文字数
pLine　　- ダブルクリックしたバッファ上の行位置(0～)を格納するバッファのアドレス（不要時は NULL）
pCol　　- ダブルクリックした桁位置(0～)を格納するバッファのアドレス（不要時は NULL）

説　明 ： pBuf≠NULL かつ lBuf≠0 の場合は、ダブルクリックした行位置のテキストを取得します。
lBuf で示されるバッファサイズが、行テキストのサイズより小さい場合は、行テキストの先頭部分だけが格納されます。
pBuf＝NULL あるいは lBuf＝0 の場合は、行テキストを取得するのに必要なバッファのサイズ（文字数）を取得します。

戻り値 ： pBuf≠NULL && lBuf≠0 の場合 - バッファに格納したテキストの文字数（文字列の終端(0x00)を含まない）
pBuf＝NULL || lBuf＝0 の場合 - テキストを取得するのに必要なバッファの文字数（文字列の終端(0x00)を含む）
エラーの場合は、０を返します。

7.9.32. ツールチップの設定／取得 (AjcVth{Set/Get}TipText)

形　式 ： BOOL AjcVthSetTipText(HWND hwnd, C_UTP pTxt); --------- ツールチップ文字列の設定
BOOL AjcVthGetTipText(HWND hwnd, UTP pBuf, UI lBuf); -- ツールチップ文字列の取得

引　数 ： hwnd - コントロールのウインドハンドル
pTxt - ツールチップ（ツールヒント）文字列のアドレス（表示しない場合は NULL）
pBuf - ツールチップ（ツールヒント）文字列を格納するバッファのアドレス
lBuf - ツールチップ（ツールヒント）文字列を格納するバッファの文字数

説　明 ： コントロール上にカーソルを置いたときに表示するツールヒント文字列を設定／取得します。

戻り値 ： TRUE - 成功
FALSE - 失敗

7.9.33. ツールチップ表示条件の設定／取得(AjcVth{Set|Get} TipShowAlways)

形　式 ： BOOL AjcVthtTipShowAlways(HWND hwnd, BOOL fShowAlways); --- 設定
BOOL AjcVthtTipShowAlways(HWND hwnd); ---------- 取得

引　数 ： hwnd - コントロールのウインドハンドル
fShowAlways - ツールチップ表示条件（TRUE:非アクティブ時も表示, FALSE:非アクティブ時は非表示）

説　明 ： ツールチップの表示条件（自プログラムが非アクティブ時の表示／非表示）を設定／取得します。

戻り値 ： 設定時：TRUE - 成功　　取得時：ツールチップ表示条件
FALSE - 失敗

7.9.34. フォント，行間スペースの設定／取得(AjcVth{Set/Get}FontInfo)

形　式 ： BOOL AjcVthSetFontInfo(HWND hwnd, const LOGFONT *pLogFont, UI LSpace); ----- 設定（LOGFONT 指定）
BOOL AjcVthGetFontInfo(HWND hwnd, LPLOGFONT *pBuf, UIP pLSpace); ---------- 取得
BOOL AjcVthSetFontHandle(HWND hwnd, HFONT hFont, UI LSpace); ---------- 設定（フォントハンドル指定）

引　数 ： hwnd - コントロールのウインドハンドル
pLogFont - 設定するフォント情報へのポインタ（NULL：フォントを設定しない）
hFont - フォントハンドル
LSpace - 設定する行間スペース（-1 ： 行間スペースを設定しない）
pBuf - 取得したフォント情報を格納するバッファへのポインタ（不要時は NULL）
pLSpace - 取得した行間スペースを格納するバッファへのポインタ（不要時は NULL）

説　明 ： フォント情報，行間スペースを設定／取得します

戻り値 ： TRUE - 成功
FALSE - 失敗

7.9.35. 文字サイズ／行の高さ取得(AjcVthGetCharInfo)

形　式 ： BOOL AjcVthGetCharInfo(HWND hwnd, UIP pCx, UIP pCy, UIP pLy);

引　数 ： hwnd - コントロールのウインドハンドル
pCx - 文字の幅　（ピクセル数）を格納するバッファのアドレス（不要時は NULL）
pCy - 文字の高さ（ピクセル数）を格納するバッファのアドレス（不要時は NULL）
pLy - 行の高さ　（ピクセル数）を格納するバッファのアドレス（不要時は NULL）

説　明 ： 現在設定されているフォントの、文字サイズ（幅，高さ）と、行の高さを取得します。
行の高さは、文字の高さ＋行間スペースとなります。
設定される値は、すべてピクセル数です。

戻り値 ： TRUE - 成功
FALSE - 失敗

7.9.36. ウインドに表示可能な行数の取得(AjcVthGetLinesPerWindow)

形　式 ： UI AjcVthGetLinesPerWindow(HWND hwnd);

引　数 ： hwnd　　- コントロールのウインドハンドル

説　明 ： ウインドに表示可能なテキストの行数を取得します。

戻り値 ： ウインドに表示可能なテキストの行数
エラーの場合は、０を返します。

7.9.37. バッファに格納されている有効な行数の取得(AjcVthGetValidLines)

形　式 ： UI AjcVthGetValidLines(HWND hwnd);

引　数 ： hwnd　　- コントロールのウインドハンドル

説　明 ： バッファに格納されている有効な行数を取得します。

戻り値 ： バッファに格納されている有効な行数
エラーの場合は、０を返します。

7.9.38. ウインド先頭行の位置取得 (AjcVthGetIxOfWndTopLine)

形　式 ： UI AjcVthGetIxOfWndTopLine (HWND hwnd);

引　数 ： hwnd　　- コントロールのウインドハンドル

説　明 ： ウインド先頭行のバッファ先頭からの行位置（０～）を取得します。

戻り値 ： ウインド先頭行のバッファ先頭からの行位置（０～）
エラーの場合は、０を返します。

7.9.39. タイトル文字列の設定 (AjcVthSetTitleText)

形　式 ： BOOL AjcVthSetTitleText (HWND hwnd, C_BCP pTitleText, COLORREF TextColor, COLORREF BackColor, HFONT hFont);

引　数 ： hwnd　　- コントロールのウインドハンドル
pTitleText　- タイトルテキストへのポインタ（NULL 指定時は、タイトル非表示）
TextColor　- テキスト描画色（－１指定時は前回設定値，デフォルト色＝白）
BackColor　- テキスト背景色（－１指定時は前回設定値，デフォルト色＝グレー）
hFont　　- タイトルテキストのフォントハンドル（デフォルトのフォント）

説　明 ： コントロールウインドの右上にタイトル文字列を表示します。

ウインドタイトル表示例



戻り値 ： TRUE　- 成功
FALSE - 失敗

7.9.40. プロファイルからプロパティ値読出し／書き込み (AjcVth{Load/Save}Prop)

形　式 ： BOOL AjcVthLoadProp (HWND hwnd, C_UTP pProfileSect, PCAJCVTHPROP pDefProp)； --- プロパティ読み出し
BOOL AjcVthSaveProp (HWND hwnd, C_UTP pProfileSect)； ------------------------- プロパティ書き込み

引　数 ： hwnd - コントロールのウインドハンドル
pProfileSect - プロファイル・セクション名（文字列）へのポインタ
pDefProp - デフォルトプロパティ値へのポインタ（現在の設定値をデフォルトとする場合は NULL）

説　明 ： AjcVthLoadProp()は、プロファイルからプロパティ値や、ログファイル出力情報、フォント選択情報、テキストセーブ情報を読み出して設定します。
pDefProp は、プロファイルに当該プロパティ値が記録されていない場合の、デフォルト・プロパティ値を指定します。
pDefProp=NULL とした場合は、現在設定されているプロパティ値を、デフォルト・プロパティとして扱います。
AjcVthSaveProp()は、現在設定されているプロパティ値等を、プロファイルへ記録します。

プロファイルへセーブ／ロードする項目は以下のとおりです。

#	内容		キー名称	備考	
1	仮想ＶＲＡＭの幅，高さ		VramW, VramH	桁数，行数	プロパティ
2	表示最大行数		MaxLines	バッファ行数	
3	タブステップ		TabStep″	2/4/8/16	
4	行間スペース		LSpace	ピクセル数	
5	最大表示保留サイズ		PendSize	バイト数	
6	フォント情報		lfHeight lfStrikeOut lfWidth lfCharSet lfEscapement lfOutPrecision lfOrientation lfClipPrecision lfWeight lfQuality lfItalic lfPitchAndFamily lfUnderlin lfFaceName		
7	パレットの色コード		rgb0 ～ rgb7	0x00bbggrr	
8	ログファイル出力情報（ウインド右上の「フロッピマーク」に関する設定情報）	入力ファイルエンコード	LogFileInpTec	未使用	
9		出力ファイルエンコード	LogFileOutTec	0:SJIS/1:UTF8/2:EUC/3:UTF16(LE)/4:UTF16(BE)	
10		ＢＯＭ出力指定	LogFileBOM	0:未出力/1:BOM 出力	
11		出力フォルダパス	LogFileDir	－	
12	フォント選択情報（フォント選択ダイアログに関する設定情報）	固定ピッチのみ	fFSelFixed	0:全フォント/1:固定のみ	
13		固定ピッチフォント属性，名称	FSelFixedAttr, FSelFixedFace	CharSet, PitchAndFamily, FaceName	
14		可変ピッチフォント属性，名称	FSelPropoAttr, FSelPropoFace		
15	テキストセーブ情報（ポップアップメニューの「ファイルへセーブ」に関する設定情報）	セーブ形式	SvTxtFormat	0:Text/1:HTML	
16		ファイルエンコード	SvTxtEncode	0:SJIS/1:EUC/2:UTF8	
17		全テキスト／選択部分	SvTxtOutput	0:AllText/1:Selected	
18		フォントサイズ（ＨＴＭＬ時）	SvTxtFontSize	1 ～ 7	
19		ＢＯＭ出力	SvBom	0:未出力/1: BOM 出力	
20		出力フォルダパス	SvTxtPath	－	

戻り値 ： TRUE - 成功
FALSE - 失敗

7.9.41. テキストをファイルへ書き込み (AjcVthSaveTextToFile)

形　式 ： BOOL　AjcVthSaveTextToFile　(HWND hwnd, C_BCP pPath, BOOL fAllText, EAJCTEC tec, BOOL fBom);

引　数 ： hwnd　　- コントロールのウインドハンドル
pPath　- 書き込むテキストファイルのパス名
fAllText - TRUE:全テキスト, FALSE:選択テキスト
tec　　- テキスト文字コード種別（AJCTEC_{MBC / UTF_8 / EUC_J / UTF_16LE / UTF_16BE}
fBom　　- ＢＯＭ出力フラグ

説　明 ： バッファに格納されているテキストをテキストファイルへ書き込みます。
fAllText=TRUE の場合は全テキストを、fAllText=FALSE の場合は選択されているテキストを出力します。
tec= AJTEC_UTF8/AJCTEC_UTF_16LE/AJCTEC_UTF_16BE で fBom=TRUE を指定した場合、ファイルの先頭にＢＯＭを出力します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

7.9.42. テキストをＨＴＭＬファイルへの書き込み (AjcVthSaveHtmlToFile)

形　式 ： BOOL　AjcVthSaveTextToFile　(HHWND hwnd, C_BCP pPath, BOOL fAllText, UI FontSize);

引　数 ： hwnd　　- コントロールのウインドハンドル
pPath　- 書き込むＨＴＭＬファイルのパス名
fAllText - TRUE:全テキスト, FALSE:選択テキスト
FontSize - 文字の大きさ（１～７）

説　明 ： バッファに格納されているテキストをＨＴＭＬファイルへ書き込みます。
fAllText=TRUE の場合は全テキストを、fAllText=FALSE の場合は選択されているテキストを出力します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

7.9.43. ＥＳＣシーケンスを付加し全テキストをファイルへ書き込み (AjcVthSaveAllTextWithEsc)

形　式 ： BOOL　AjcVthSaveAllTextWithEsc　(HHWND hwnd, C_BCP pPath, EAJCTEC tec, BOOL fBom);

引　数 ： hwnd　　- コントロールのウインドハンドル
pPath　- 書き込むテキストファイルのパス名
tec　　- テキスト文字コード種別（AJCTEC_MBC / ..UTF_8 / ..EUC_J / ..UTF_16LE / ..UTF_16BE)
fBom　　- ＢＯＭ出力フラグ

説　明 ： バッファに格納されている全テキストをＥＳＣシーケンス（テキスト色, 背景色）を付加してファイルへ書き込みます。
tec= AJCTEC_UTF8/AJCTEC_UTF_16LE/AJCTEC_UTF_16BE で fBom=TRUE を指定した場合、ファイルの先頭にＢＯＭを出力します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

7.9.44. バッファに格納されている行数の取得 (AjcVthGetLineCount)

形　式 ： UI　AjcVthGetLineCount　(HHWND hwnd);

引　数 ： hwnd　　- コントロールのウインドハンドル

説　明 ： バッファに格納されている行数を取得します。

戻り値 ： バッファに格納されている行数（エラー時は０を返します）

7.9.45. マウスカーソル位置の行番号と文字位置取得 (AjcVthGetCursorPosInfo)

形　式 ： BOOL AjcVthGetCursorPosInfo (HHWND hwnd, UIP pLine, UIP pCol);

引　数 ： hwnd　　- コントロールのウインドハンドル
pLine　- 行位置（０～）を格納するバッファのアドレス（不要時は NULL）
pCol　　- 桁位置（０～）を格納するバッファのアドレス（不要時は NULL）

説　明 ： マウスカーソル位置が示す位置の行番号と文字位置を取得します。
行位置はバッファ先頭（スクロール最上端の行）を０とした行位置です。
マウスカーソル位置が当該行テキストの右端を超えている場合は、*pCol には当該行の最右端の文字位置が設定されます。
マウスカーソルがＶＴ１００エミュレーションウインドの外である場合は、何もせず FALSE を返します。

戻り値 ： TRUE　- 成功
FALSE - 失敗（カーソル位置がウインド範囲外）

7.9.46. 指定行位置の行テキスト取得 (AjcVthGetLineText)

形　式 ： UI AjcVthGetLineText (HHWND hwnd, UI pos, UTP pBuf, BCP lBuf);

引　数 ： hwnd　　- コントロールのウインドハンドル
pos　　- 行位置（０～）
pBuf　　- 取得した行テキストを格納するバッファのアドレス（不要時は NULL）
lBuf　　- 取得した行テキストを格納するバッファの文字数（１以上）

説　明 ： 指定行位置のテキストを取得し、バッファに格納したバイト数／文字数（文字列終端は含まない）を返します。
バッファの末尾には文字列終端（0x00）が付加されます。
pBuf＝NULL を指定した場合は、指定行位置のテキストの文字数（文字列終端を含む）の取得だけを行います。

戻り値 ： ≠０：指定行位置のテキストのバイト数／文字数（pBuf＝NULL の場合は、文字列終端も含めた長さ）
＝０：不正な行位置

7.9.47. 縦スクロール位置の設定／取得 (AjcVth{Set/Get}VScrollPos)

形　式 ： BOOL AjcVthSetVScrollPos (HHWND hwnd, UI pos); -- 縦スクロール位置の設定
UI AjcVthGetVScrollPos (HHWND hwnd); ---------- 縦スクロール位置の取得

引　数 ： hwnd　　- コントロールのウインドハンドル
pos　　- 設定するスクロール行位置（０～）

説　明 ： 設定時は、ウインドの上端が、指定した行位置となるように表示します。
指定した行位置をウインドの上端に設定できない場合は、指定行位置がウインドの途中になるように表示します。

取得時は、現在のスクロール行位置を取得します。
スクロール行位置とは、ウインド上端に表示されている行位置を意味します。

戻り値 ： 設定時：TRUE　- 成功　　取得時：スクロール行位置（０～，エラー時は０を返します）
FALSE - 失敗

7.9.48. 横スクロール位置の設定／取得 (AjcVth{Set/Get}HScrollPos)

形　式 ： BOOL AjcVthSetHScrollPos (HHWND hwnd, UI pos); -- 横スクロール位置の設定
UI AjcVthGetHScrollPos (HHWND hwnd); ---------- 横スクロール位置の取得

引　数 ： hwnd - コントロールのウインドハンドル
pos - 設定するスクロール桁位置（０～）

説　明 ： 設定時は、ウインドの左端が、指定した文字位置となるように表示します。
指定した文字位置をウインドの左端に設定できない場合は、指定文字位置がウインドの途中になるように表示します。

取得時は、現在のスクロール桁位置を取得します。
スクロール桁位置とは、ウインド左端に表示されている文字位置を意味します。

戻り値 ： TRUE - 成功
FALSE - 失敗

7.9.49. 表示ウインドサイズ（行数，文字数）取得 (AjcVthGetWindowSize)

形　式 ： BOOL AjcVthGetWindowSize (HWND hwnd, LPSIZE pSize);

引　数 ： hwnd - コントロールのウインドハンドル
pSize - 表示ウインドのサイズ（行数(cy)，文字数(cx)）を格納するバッファのアドレス

説　明 ： コントロールのウインドに表示可能な行数と文字数を取得します。
但し、プロポーショナルフォント（可変ピッチフォント）の場合は、正確な文字数は取得できません。
プロポーショナルフォントの場合は、平均的な文字ピッチでの文字数が設定されます。

戻り値 ： TRUE - 成功
FALSE - 失敗

7.9.50. 文字列の検索 (AjcVthSearchBelow)

形　式 ： UI AjcVthSearchBelow (HWND hwnd, C_UTP pStr, UT delimiter); --- 下方向検索
UI AjcVthSearchAbove (HWND hwnd, C_UTP pStr, UT delimiter); --- 上方向検索

引　数 ： hwnd - コントロールのウインドハンドル
pStr - 検索文字列（複数指定時は、区切り文字で区切る）
delimiter – 区切り文字（半角文字コード，０の場合は区切り無し）

説　明 ： 文字列を検索します。
下方向検索では、最初（前回の検索終了時からスクロール位置が変化している場合）は現表示ウインドの上端から検索を開始し、実行毎に次の下方向の文字列を検索します。
上方向検索では、最初（前回の検索終了時からスクロール位置が変化している場合）は現表示ウインドの下端から検索を開始し、実行毎に次の上方向の文字列を検索します。
文字列が見つかったら、見つかった文字列を選択状態にし、当該文字列の位置までスクロールします。
複数の文字列を検索する場合は、「delimiter」に区切り文字を指定します。　この場合、区切られた検索文字の前後の空白は除去されます。（**“ABC,XYZ”** と **” ABC , XYZ ”** は同じに扱います。）

ex. “ABC” or ”XYZ”を検索する → AjcVthSearchBelow(hwnd, “ ABC , XYZ ”, ‘,’);

「delimiter」に０を指定した場合は、検索文字列を区切らずに、全体を１つの文字列として扱います。

ex. “ABC, XYZ” を検索する → AjcVthSearchBelow(hwnd, “ABC, XYZ”, 0);

戻り値 ： ≠ -1 ： 行スクロール位置（ウインド上端の行位置）
＝ -1 ： エラー

注　意 ： **文字列検索後に 描画した内容を表示するには、ウインドをクリックし選択状態を解除してください。**
見つかった文字列は、選択状態となり反転表示となりますので（選択状態では）その後の描画内容が表示されません。

7.9.51. 画面表示の停止／再開 (AjcVthPause)

形　式 ： BOOL AjcVthPause (HWND hwnd, BOOL fPause);

引　数 ： hwnd　　- コントロールのウインドハンドル
fPause　- 表示停止／再開フラグ（TRUE:停止, FALSE:再開）

説　明 ： 表示を停止／再開します。
fPause=TRUE を指定すると表示が停止します。表示停止中でも、テキストの描画は有効です。
fPause=FALSE を指定すると、表示を再開します。この時、表示停止中に描画されたテキストは一気に表示されます。
（全てのテキストを表示＆スクロールするわけではなく、最終画面状態だけを表示します）

戻り値 ： TRUE　- 成功
FALSE - 失敗

7.9.52. 設定情報の永続化 (AjcVthLoadPermInfo[Ex] / AjcVthSavePermInfo)

形　式 ： BOOL AjcVthLoadPermInfo (HWND hwnd, C_UTP pProfileSect, C_UTP pKeyPrefix); ---------- 永続化情報の読み出し
BOOL AjcVthLoadPermInfoEx(HWND hwnd, C_UTP pProfileSect, C_UTP pKeyPrefix, UI PermItem); -　〃
BOOL AjcVthSavePermInfo (HWND hwnd); ---------- 永続化情報の書き込み
BOOL AjcVthSavePermInfoEx(HWND hwnd, C_UTP pProfileSect, C_UTP pKeyPrefix, UI PermItem); --　〃

引　数 ： hwnd　　- コントロールのウインドハンドル
pProfileSect - 永続化情報を記録するプロファイルセクション名のアドレス
pKeyPrefix　- 永続化情報を記録するプロファイルキー名先頭部分のアドレス
PermItem　- 永続化情報項目の選択（AJCVTH_PERM_*XXXXX*）

説　明 ： プロファイルから永続化情報の読み出し、あるいは、書き込みを行います。
AjcVthSavePermInfoEx()で PermItem=AJCVTH_PERM_BYLOAD を指定した場合は、AjcVthLoadPermInfoEx()で指定した PermItem が有効となります。
永続化する内容は、以下のとおりです

#	内容		キー末尾名称	備考	項目指定（AJCVTH_PERM_*XXX*）
1	ポップアップメニュー「フォント設定を全ての VT-100 に適用 」の設定内容		fSetF_T	0:全兄弟ウインド 1:全スレッドウインド	常時永続化　※1
2	ポップアップメニュー「改行動作 」の設定内容		Style		LFACT
3	フォント情報		lfHeight　lfStrikeOut lfWidth　lfCharSet lfEscapement　lfOutPrecision lfOrientation　lfClipPrecision lfWeight　lfQuality lfItalic　lfPitchAndFamily lfUnderlin　lfFaceName		FONT　※1
4	行間スペース		LSpace	ピクセル数	
5	フォント選択ダイアログの設定情報	固定ピッチのみフラグ	fFSelFixed	0:全フォント 1:固定のみ	
6		固定ピッチフォント属性, 名称	FSelFixedAttr FSelFixedFace	CharSet, PitchAndFamily, FaceName	
7		可変ピッチフォント属性, 名称	FSelPropoAttr FSelPropoFace		
8	仮想ＶＲＡＭの幅, 高さ		VramW, VramH	桁数, 行数	VRAM
9	表示最大行数		MaxLines	バッファ行数	OTHER
10	タブステップ		TabStep″	2/4/8/16	
11	最大表示保留サイズ		PendSize	バイト数	

※1 : AjcVth{Load/Save}PermInfo()や、Ajc{Load/Save}{All/Grp}ControlSettings()で永続化する項目

戻り値 ： TRUE　- 成功
FALSE - 失敗

7.9.53. 文字列検索情報永続化プロファイルセクション名設定 (AjcVthSetFindProfileSect)

形　式 ： BOOL AjcVthSetFindProfileSect (HWND hwnd, UI key);

引　数 ： hwnd　　- コントロールのウインドハンドル
key　　- 文字列検索キーコード（０：キーによる検索禁止，デフォルト＝VK_F3）

説　明 ： 文字列検索情報を永続化するためのプロファイルセクションを設定し、プロファイル情報を読み出します。
このＡＰＩを実行しない場合、プロファイルセクションは、「 _DefaultStrFindInfoSect_ 」固定となります。

戻り値 ： TRUE - 成功
FALSE - 失敗

7.9.54. 文字列検索キーの設定 (AjcVthSetFindKey)

形　式 ： UI AjcVthSetFindKey (HWND hwnd, UI key);

引　数 ： hwnd　　- コントロールのウインドハンドル
key　　- 文字列検索キーコード（０：キーによる検索禁止，デフォルト＝VK_F3）

説　明 ： 文字列検索用のキーコードを設定します。
指定したキーを押した場合、文字列を下方向検索します。
SHIFT キーを同時に押した場合は、上方向検索となります。

戻り値 ： ≠-1 - 前回設定値
＝-1 - 失敗

7.9.55. メニューキーの設定 (AjcVthSetMenuKey)

形　式 ： UI AjcVthSetMenuKey (HWND hwnd, UI key);

引　数 ： hwnd　　- コントロールのウインドハンドル
key　　- 文字列検索キーコード（０：キーによるメニュー表示禁止，デフォルト＝VK_F1）

説　明 ： ポップアップメニュー表示用のキーコードを設定します。
指定したキーを押した場合、ポップアップメニューを表示します。

戻り値 ： ≠-1 - 前回設定値
＝-1 - 失敗

7.9.56. ベント通知用コールバック関数の設定(AjcVthSetEvtCallBack)

形　式 ： BOOL AjcVthSetEvtCallBack (HWND hwnd, UX cbp, VO (CALLBACK *cbNtcEvt)(HWND hCtrl, UX wParam, UX lParam, UX cbp));

引　数 ： hwnd　　- コントロールのウインドハンドル
cbp　　- コールバックパラメタ
cbNtcEvt　- イベント通知用コールバック関数（NULL:コールバック関数を無効化）

説　明 ： イベント通知用のコールバック関数を設定します。
通常、イベントは WM_COMMAND メッセージで通知されますが、コールバック関数を設定することにより、コールバック関数の呼び出しにより通知されるようになります。

戻り値 ： TRUE　- 成功
FALSE - 失敗

コールバック： コールバック関数の仕様は、以下のとおりです。

cbNtcEvt（イベント通知）

形　式 ： VO CALLBACK *cbNtcEvt*(HWND hCtrl, UX wParam, UX lParam, UX cbp)；

引　数 ： hCtrl　- コントロールのウインドハンドル
wPatram - コントロール識別ＩＤ，通知コード（WM_COMMAND による通知と同じ）
lPatram - イベントパラメタ　　（WM_COMMAND による通知と同じ）
cbp　　- コールバックパラメタ

説　明 ： 各種イベントを通知します。（詳細は「コントロールの通知メッセージ」を参照）

戻り値 ： なし

7.10. 通知情報の取得ＡＰＩ

コントロールからの通知メッセージ（WM_COMMAND）に伴うパラメタを取得するＡＰＩ群です。
「AjcSetCmdWithHdl(TRUE);」を実行した場合、各通知メッセージ（WM_COMMAND）の lParam は通知内容に伴うパラメタは通知されず、
lParam＝コントロールのウインドハンドルとなります。
この場合、通知内容に伴うパラメタは以下のＡＰＩで取得します。

#	関 数 名	内容	対応する通知	
1	AjcVthGetNtcKey	入力キーコードの取得	AJCVTHN_KEYIN AJCVTHN_VKEYIN AJCVTHN_VKEYOUT	キー入力通知 拡張キー押下通知 拡張キー離し通知
2	AjcVthGetNtcKeyRep	キー押し続けによる繰り返し情報の取得	AJCVTHN_KEYIN AJCVTHN_VKEYIN	キー入力通知 拡張キー押下通知
3	AjcVthGetNtcFiles	ドロップしたファイル数の取得	AJCVTHN_DROPFILE	ファイルドロップ通知
4	AjcVthGetNtcDirs	ドロップしたディレクトリ数の取得	AJCVTHN_DROPDIR	ディレクトリドロップ通知
5	AjcVthGetNtcRClk	右クリック情報の取得	AJCVTHN_RCLICK	右クリック通知
6	AjcVthGetNtcDblClk	Shift, Ctrl キーの押下状態	AJCVTHN_DBLCLK	ダブルクリック通知
7	AjcVthGetNtcCyLine	行の高さ	AJCVTHN_CHARINFO	文字サイズ情報の変化通知
8	AjcVthGetNtcLeft	左端桁位置	AJCVTHN_HSCROLL	横スクロール通知
9	AjcVthGetNtcTop	上端行位置	AJCVTHN_VSCROLL	縦スクロール通知時

7.10.1. 入力キーコードの取得（AjcVthGetNtcKey）

形　式　：　UI　AjcVthGetNtcKey　(HHWND hwnd);

引　数　：　hwnd　　－ コントロールのウインドハンドル

説　明　：　キー入力通知通知（AJCVTHN_KEYIN）、拡張キー押下通知（AJCVTHN_VKEYIN）, 拡張キー離し通知（AJCVTHN_VKEYOUT）通知時のキーコード取得します。

戻り値　：　キーコード

7.10.2. キー押し続けによる繰り返し情報の取得（AjcVthGetNtcKeyRep）

形　式　：　UI　AjcVthGetNtcKeyRep(HHWND hwnd);

引　数　：　hwnd　　－ コントロールのウインドハンドル

説　明　：　キー入力通知通知（AJCVTHN_KEYIN）、拡張キー押下通知（AJCVTHN_VKEYIN）時のキーの押し続けによる連続キー入力か否かの情報を返します。
この情報が０の場合、単発のキー入力／キー押し続けた場合の初回キー入力通知を意味します。
この情報が１以上の場合は、キー押し続けたことによる、繰り返し回数を意味します。

戻り値　：　連続キー入力か否かの情報（0:初回, 1～:繰り返し回数）

7.10.3. ドロップしたファイル数の取得（AjcVthGetNtcFiles）

形　式 ： UI　AjcVthGetNtcFiles (HHWND hwnd);

引　数 ： hwnd　　- コントロールのウインドハンドル

説　明 ： ファイルドロップ通知（AJCVTHN_DROPFILE）時の、ドロップされたファイルの個数を取得します。

戻り値 ： ドロップされたファイルの個数

7.10.4. ドロップしたディレクトリ数の取得（AjcVthGetNtcDirs）

形　式 ： UI　AjcVthGetNtcDirs (HHWND hwnd);

引　数 ： hwnd　　- コントロールのウインドハンドル

説　明 ： ディレクトリドロップ通知（AJCVTHN_DROPDIR）時の、ドロップされたディレクトリの個数を取得します。

戻り値 ： ドロップされたディレクトリの個数

7.10.5. 右クリック情報の取得（AjcVthGetNtcRClk）

形　式 ： PAJCVTHRCLK　AjcVthGetNtcRClk (HHWND hwnd);

引　数 ： hwnd　　- コントロールのウインドハンドル

説　明 ： 右クリック通知（AJCVTHN_RCLICK）時の、右クリック情報を取得します。

戻り値 ： 右クリック情報（AJCVTHRCLK）へのポインタ

7.10.6. ダブルクリック情報の取得（AjcVthGetNtcDblClk）

形　式 ： UI　AjcVthGetNtcDblClk (HHWND hwnd);

引　数 ： hwnd　　- コントロールのウインドハンドル

説　明 ： ダブルクリック通知（AJCVTHN_DBLCLK）時の、Shift, Ctrl キーやマウスボタンの押下状態を取得します。

戻り値 ： 以下のシンボルの組み合わせ

シンボル	値	内容
MK_MBUTTON	0x10 (Bit4)	1:マウス中ボタン押下状態
MK_CONTROL	0x08 (Bit3)	1: Ctrl キー未押下
MK_SHIFT	0x04 (Bit2)	1: Shift キー未押下

シンボル	値	内容
MK_RBUTTON	0x02 (Bit1)	1:マウス右ボタン押下状態
MK_LBUTTON	0x01 (Bit0)	1:マウス左ボタン押下状態

7.10.7. 文字サイズ情報の変化通知時の行の高さ取得（AjcVthGetNtcCyLine）

形　式 ： UI AjcVthGetNtcCyLine (HHWND hwnd);

引　数 ： hwnd - コントロールのウインドハンドル

説　明 ： 文字サイズ情報の変化通知(AJCVTHN_CHARINFO)時の行の高さを取得します。

戻り値 ： 行の高さ（ピクセル数）

7.10.8. 横スクロール通知時の左端桁位置取得（AjcVthGetNtcLeft）

形　式 ： UI AjcVthGetNtcLeft (HHWND hwnd);

引　数 ： hwnd - コントロールのウインドハンドル

説　明 ： 横スクロール通知(AJCVTHN_HSCROLL)時の左端桁位置を取得します。

戻り値 ： 左端桁位置

7.10.9. 縦スクロール通知時の上端行位置取得（AjcVthGetNtcTop）

形　式 ： UI AjcVthGetNtcTop (HHWND hwnd);

引　数 ： hwnd - コントロールのウインドハンドル

説　明 ： 縦スクロール通知(AJCVTHN_VSCROLL)時の上端行位置を取得します。

戻り値 ： 上端行位置

7.11. 通知メッセージ

ＶＴ－１００エミュレーションウインド・コントロールからの通知メッセージは、WM_COMMAND メッセージにより通知されます。
WM_COMMAND メッセージの wParam には以下の情報が設定されます。

・LOWORD(wParam) - コントロールの識別ＩＤ
・HIWORD(wParam) - 通知メッセージコード

通知メッセージコードは、以下のとおりです。

#	通知メッセージコード	内容	備考
1	AJCVTHN_DBLCLK	ダブルクリック通知	
2	AJCVTHN_KEYIN	キー入力通知	本コントロールをダイアログ項目とする場合は、使用できません。
3	AJCVTHN_VKEYIN	拡張キー押下通知	
4	AJCVTHN_VKEYOUT	拡張キー離し通知	
5	AJCVTHN_DROPFILE	ファイルドロップ通知	
6	AJCVTHN_DROPDIR	ディレクトリロップ通知	
7	AJCVTHN_CHARINFO	文字サイズ情報の変化通知	
8	AJCVTHN_HSCROLL	横スクロール通知	
9	AJCVTHN_VSCROLL	縦スクロール通知	
10	AJCVTHN_RCLICK	右クリック通知	SHIFT/CTRL + 右クリック時
11	AJCVTHN_CLEAR	画面クリアー通知	ポップアップメニューによる画面クリアー

7.11.1. ダブルクリック通知（AJCVTHN_DBLCLK）

説　明　：　ダブルクリックされたことを通知します。
ダブルクリックされた位置の行テキストを取得するには、AjcGetDblClickedLine()を実行します。

パラメタ：　wParam - コントロールの識別ＩＤと通知メッセージコード（AJCVTHN_DBLCLK）
lParam - Shift, Ctrl キーやマウスボタンの押下状態　　（MFC 使用時はコントロールのウインドハンドル）
（以下のシンボルの組み合わせ）

シンボル	値	内容
MK_MBUTTON	0x10 (Bit4)	1:マウス中ボタン押下状態
MK_CONTROL	0x08 (Bit3)	1: Ctrl キー未押下
MK_SHIFT	0x04 (Bit2)	1: Shift キー未押下

シンボル	値	内容
MK_RBUTTON	0x02 (Bit1)	1:マウス右ボタン押下状態
MK_LBUTTON	0x01 (Bit0)	1:マウス左ボタン押下状態

戻り値　：　なし（ゼロを返してください）

7.11.2. キー入力通知（AJCVTHN_KEYIN）

説　明　：　テキストが選択されていない状態で、通常のキー入力があったことを通知します。
通常のキー入力とは、英数字，全角文字や特殊記号（@[*+-=・・・）あるいは、「Enter」「BackSpace」「Tab」やキーを意味します。その他のキー入力は、「AJCVTHN_VKEYIN」により通知されます。

パラメタ：　wParam - コントロールの識別ＩＤと通知メッセージコード（AJCVTHN_KEYIN）
lParam - キーコード（MFC 使用時はコントロールのウインドハンドル）
バイト文字モード時、全角文字は１バイトづつ２回に分けて通知します。
ワイド文字モード時、サロゲートペア文字は、１ワードづつ２回に分けて通知します。

戻り値　：　なし（ゼロを返してください）

備　考　：　テキストが未選択状態である場合、WM_CHAR メッセージによるキー入力を通知します。
ダイアログに対し、IsDialogMessage()を処理している場合は、通知されません。
AjcVthGetNtcKeyRep()により、キー押し続けによる繰り返し情報を取得できます。

7.11.3. 拡張キー押下通知（AJCVTHN_VKEYIN）

説　明　：　テキストが選択されていない状態で、非システムキーが押されたことを通知します。
非システムキーとは、Alt キーが押されていないときの全てのキーを意味し、「Delete」「Insert」「F1」「F2」・・・等のキーも含まれます。
主な、非システムキーのキーコードを以下に示します。

拡張キーコード

名称	コード	内容	名称	コード	内容
VK_SHIFT	0x10	Shift	VK_SELECT	0x29	SEL
VK_CONTROL	0x11	Ctrl	VK_PRINT	0x2A	Print
VK_CLEAR	0x0C	NumLock-OFF 時のテンキーの「5」	VK_EXECUTE	0x2B	Execute
VK_ESCAPE	0x1B	Esc	VK_SNAPSHOT	0x2C	Print Screen
VK_CONVERT	0x1C	変換	VK_INSERT	0x2D	Insert
VK_NOCONVERT	0x1D	無変換	VK_DELETE	0x2E	Delete
VK_ACCEPT	0x1E	ACCEPT	VK_HELP	0x2F	Help
VK_MODECHANGE	0x1F	MODECHANGE	VK_LWIN	0x5B	左 Windows
VK_PRIOR	0x21	PageUp	VK_RWIN	0x5C	右 Windows
VK_NEXT	0x22	PageDown	VK_APPS	0x5D	ｱﾌﾟﾘｹｰｼｮﾝ
VK_END	0x23	End	VK_F1	0x70	F1
VK_HOME	0x24	Home	VK_F2～VK_F23	0x71～	F2～F23
VK_LEFT	0x25	←	VK_F24	0x87	F24
VK_UP	0x26	↑	VK_NUMLOCK	0x90	Num Lock
VK_RIGHT	0x27	→	VK_SCROLL	0x91	Scroll Lock
VK_DOWN	0x28	↓			

パラメタ：　wParam - コントロールの識別ＩＤと通知メッセージコード（AJCVTHN_VKEYIN）
lParam - 拡張キーコード（MFC 使用時はコントロールのウインドハンドル）

戻り値　：　なし（ゼロを返してください）

備　考　：　WM_KEYDOWN メッセージによるキー押下を通知します。
既定では、VK_F1(メニュー表示)と VK_F3(文字列検索が有効時)は除外します。
本コントロールをダイアログボックスの項目としている場合、本通知メッセージは正常に機能しません。
AjcVthGetNtcKeyRep()により、キー押し続けによる繰り返し情報を取得できます。

7.11.4. 拡張キー離し通知（AJCVTHN_VKEYOUT）

説　明　：　非システムキーが離されたことを通知します。

パラメタ：　コントロールの識別ＩＤと通知メッセージコード（AJCVTHN_VKEYOUT）
lParam - 拡張キーコード(AJCVTHN_VKEYIN と同じ）（MFC 使用時はコントロールのウインドハンドル）

戻り値　：　なし（ゼロを返してください）

備　考　：　WM_KEYUP メッセージによるキー離しを通知します。
既定では、VK_F1(メニュー表示)と VK_F3(文字列検索が有効時)は除外します。
本コントロールをダイアログボックスの項目としている場合、本通知メッセージは正常に機能しません。

7.11.5. ファイルドロップ通知（AJCVTHN_DROPFILE）

説　明　：　ドラッグ＆ドロップ操作で、ファイルがドロップされたことを通知します。
本通知では、ドロップされたファイルの個数だけを通知します。
AjcVthGetDroppedFile() によりドロップされたファイルのパス名を取得できます。

パラメタ：　wParam - コントロールの識別ＩＤと通知メッセージコード（AJCVTHN_DROPFILE）
lParam - ドロップされたファイルの個数（MFC 使用時はコントロールのウインドハンドル）

戻り値　：　なし（ゼロを返してください）

7.11.6. ディレクトリドロップ通知（AJCVTHN_DROPDIR）

説　明 ： ドラッグ＆ドロップ操作で、ディレクトリがドロップされたことを通知します。
本通知では、ドロップされたディレクトリの個数だけを通知します。
AjcVthGetDroppedDir[Ex]()によりドロップされたディレクトリのパス名を取得できます。

パラメタ： wParam - コントロールの識別ＩＤと通知メッセージコード（AJCVTHN_DROPDIR）
lParam - ドロップされたディレクトリの個数（MFC 使用時はコントロールのウインドハンドル）

戻り値 ： なし（ゼロを返してください）

7.11.7. 文字サイズ情報の変化通知（AJCVTHN_CHARINFO）

説　明 ： フォントの設定等により、文字のサイズや行の高さが変化したことを通知します。
本通知では、lParam で行の高さを通知します。
その他の情報を取得するには、AjcVthGetCharInfo()や、AjcVthGetLinesPerWindow()を呼び出してください。

パラメタ： wParam - コントロールの識別ＩＤと通知メッセージコード（AJCVTHN_CHARINFO）
lParam - 行の高さ（文字の高さ＋行間スペース）（MFC 使用時はコントロールのウインドハンドル）

戻り値 ： なし（ゼロを返してください）

7.11.8. 横スクロール通知（AJCVTHN_HSCROLL）

説　明 ： 横スクロールバー操作による、横スクロールが発生したことを通知します。

パラメタ： wParam - コントロールの識別ＩＤと通知メッセージコード（AJCVTHN_HSCROLL）
lParam - ウインド左端の桁位置（０～）（MFC 使用時はコントロールのウインドハンドル）

戻り値 ： なし（ゼロを返してください）

7.11.9. 縦スクロール通知（AJCVTHN_VSCROLL）

説　明 ： 縦スクロールバー操作による、縦スクロールが発生したことを通知します。

パラメタ： wParam - コントロールの識別ＩＤと通知メッセージコード（AJCVTHN_VSCROLL）
lParam - ウインド上端の行位置（バッファ先頭からの行位置，０～）（MFC 使用時はコントロールのウインドハンドル）

戻り値 ： なし（ゼロを返してください）

7.11.10. 右クリック通知（AJCVTHN_RCLICK）

説　明　：　SHIFT か CTRL キーを押しながら右クリックしたことを通知します。
右クリック通知の可否については、AjcVthSetNtcRClk()を参照してください。
lParam で以下の右クリック情報を通知します。

```
typedef struct {
    int         x;              //  カーソルｘ座標
    int         y;              //  カーソルｙ座標
    BOOL        fShift;         //  SHIFT キー押下フラグ
    BOOL        fCtrl;          //  CTRL キー押下フラグ
} AJCVTHRCLK, *PAJCVTHRCLK;
```

パラメタ：　wParam – コントロールの識別ＩＤと通知メッセージコード（AJCVTHN_RCLICK）
lParam – 右クリック情報へのポインタ（PAJCVTHRCLK）（MFC 使用時はコントロールのウインドハンドル）

戻り値　：　なし（ゼロを返してください）

7.11.11. 画面クリアー通知（AJCVTHN_CLEAR）

説　明　：　右クリックによるポップアップメニューで「全てクリアー」が選択され、画面がクリアーされたことを通知します。

パラメタ：　wParam – コントロールの識別ＩＤと通知メッセージコード（AJCVTHN_CLEAR）
lParam – コントロールのウインドハンドル

戻り値　：　なし（ゼロを返してください）

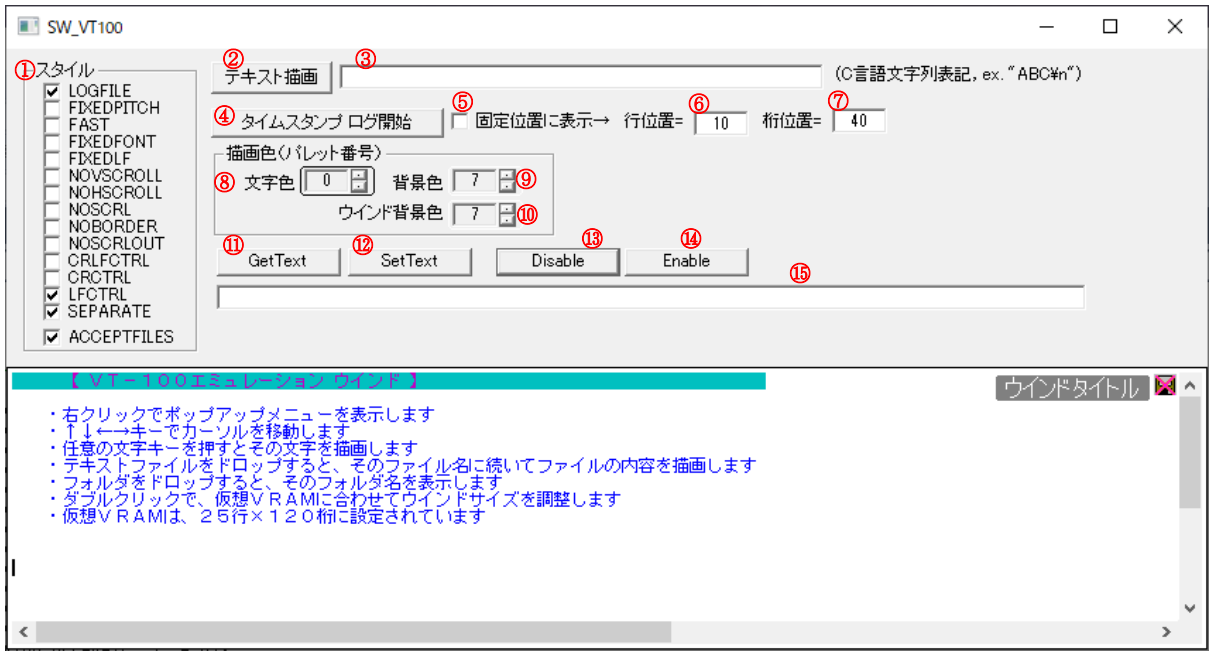
7.12. サンプルプログラム

7.13. SW_VT100（VT100 エミュレーションウインド）

このサンプルプログラムは、ＶＴ－１００エミュレーションウインド・コントロールへの描画等を行います。

ＶＴ－１００エミュレーションウインド・コントロールにフォーカスを与えた状態で、キー入力したデータは、そのままコントロールへ描画します。上下左右（↑↓←→）キーを入力した場合は、カーソルの移動を行います。

ファイルをドロップした場合は、そのファイル名と、ファイルの内容を表示します。 ⑮
ディレクトリをドロップした場合は、当該ディレクトリ名を表示します。



#	内　容	備　考
①	VT100 コントロールのスタイルを設定します	
②	③のテキストをコントロールへ描画します	
③	VT100 コントロールへの描画テキスト	C言語文字列の形式
④	タイムスタンプ（現在の日時テキスト）をコントロールへ描画します	100ms 周期で連続描画
⑤	チェックした場合、タイムスタンプを、⑥,⑦で指定された行／桁位置へ描画します	
⑥	タイムスタンプの描画行位置を設定します	
⑦	タイムスタンプの描画桁位置を設定します	
⑧	文字描画色（パレット番号）を指定します	0～7
⑨	背景描画色（パレット番号）を指定します	0～7
⑩	ウインド背景色（パレット番号）を指定します	0～7
⑪	VT100 コントロールからテキストを取得し、⑭に表示します	
⑫	⑭のテキストをコントロールに設定します	
⑬	VT100 コントロールが入力を受け付けないようにします	
⑭	VT100 コントロールが入力を受け付けるようにします	
⑮	VT100 コントロールの取得／設定テキスト	

```
 1 : //
 2 : //  SW_VT100.c
 3 : //
 4 : #include    <AjrCstXX.h>
 5 : #include    <math.h>
 6 : #include    <tchar.h>
 7 : #include    "resource.h"
 8 :
 9 : #define     IDC_VTH     5000
10 :
11 : //-------------------------------------------------------------------------------------------------------------//
12 : //  ワーク                                                                                                     //
13 : //-------------------------------------------------------------------------------------------------------------//
14 : HINSTANCE       hInst;                      //  ＤＬＬインスタンスハンドル
15 : HWND            hWndBack;                   //  バックウインドハンドル（ダイアログと VT-100 の親ウインド）
16 : HWND            hDlgMain;                   //  ダイアログボックスハンドル
17 : HWND            hWndVth;                    //  タイムチャート・グラフコントロールのウインドハンドル
18 :
19 : int             WndWidth, WndHeight;        //  ウインドの幅と高さ
20 : int             DlgWidth, DlgHeight;        //  ダイアログの幅と高さ
21 : BOOL            fTimeStampLog = FALSE;      //  タイムスタンプログ動作状態
22 : UI              TxtColor, BkColor, BkWnd;   //  描画色（パレット番号）
23 :
24 : //-------------------------------------------------------------------------------------------------------------//
25 : //  内部サブ関数                                                                                               //
26 : //-------------------------------------------------------------------------------------------------------------//
27 : AJC_WNDPROC_DEF(Back);
28 : AJC_DLGPROC_DEF(Main);
29 : static  VO  SetStyleToCheckBox(VO);
30 : static  VO  SetCheckBoxToStyle(VO);
31 :
32 : //=============================================================================================================//
33 : //                                                                                                             //
34 : //  Ｗｉｎ Ｍａｉｎ                                                                                           //
35 : //                                                                                                             //
36 : //=============================================================================================================//
37 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
38 : {
39 :     MSG         msg;
40 :     WNDCLASS    wndclass;
41 :     hInst = hInstance;
42 :
43 :     //----- バックウインド生成 -------------------------//
44 :     wndclass.style          = 0;
45 :     wndclass.lpfnWndProc    = AJC_WNDPROC_NAME(Back);
46 :     wndclass.cbClsExtra     = 0;
47 :     wndclass.cbWndExtra     = 0;
48 :     wndclass.hInstance      = hInst;
49 :     wndclass.hIcon          = NULL;
50 :     wndclass.hCursor        = LoadCursor(NULL, IDC_ARROW);
51 :     wndclass.hbrBackground  = (HBRUSH)GetStockObject(WHITE_BRUSH);
52 :     wndclass.lpszMenuName   = NULL;
53 :     wndclass.lpszClassName  = TEXT("SW_VT100");
54 :     RegisterClass(&wndclass);
55 :
56 :     hWndBack = CreateWindow(TEXT("SW_VT100"),           // window class name
57 :                             TEXT("SW_VT100"),           // window caption
58 :                             WS_OVERLAPPEDWINDOW,        // window style
59 :                             0,                          // initial x position
60 :                             0,                          // initial y position
61 :                             0,                          // initial x size
62 :                             0,                          // initial y size
63 :                             NULL,                       // parent window handle
64 :                             NULL,                       // window menu handle
65 :                             hInst,                      // program instance handle
66 :                             NULL);                      // creation parameters
67 :
68 :     //----- ウインド表示 -------------------------------//
69 :     ShowWindow(hWndBack, iCmdShow);
70 :     //----- ＶＴ－１００ウインドをフォーカス -----------//
71 :     SetFocus(hWndVth);
72 :
73 :     //----- メッセージループ ---------------------------//
74 :     while (GetMessage(&msg, NULL, 0, 0)) {
75 :         do {
76 :             if (IsDialogMessage(hDlgMain, &msg)) break;
77 :             TranslateMessage(&msg);
```

```
78 :             DispatchMessage (&msg);
79 :         } while (0);
80 :     }
81 :
82 :     return (int)msg.wParam ;
83 : }
84 : //=============================================================================================================//
85 : //                                                                                                             //
86 : //  バックウインド・プロシージャ                                                                               //
87 : //                                                                                                             //
88 : //=============================================================================================================//
89 : //----- WM_CREATE -----------------------------------------------------------------------------------------------//
90 : AJC_WNDPROC(Back, WM_CREATE         )
91 : {
92 :     RECT    r;
93 :     int     sty = (int)MAjcGetWindowLong(hwnd, GWL_STYLE);
94 :     int     exs = (int)MAjcGetWindowLong(hwnd, GWL_EXSTYLE);
95 :
96 :     hWndBack = hwnd;
97 :
98 :     //----- VT-100 エミュレーション・コントロール生成 -----------------//
99 :     hWndVth = CreateWindowEx(WS_EX_ACCEPTFILES,                                    // extend style
100 :                              TEXT("AjcCtrlVT100"),                                 // window class name
101 :                              TEXT("P: VW=120, VH=25, FN=MS Gothic"),               // window caption
102 :                              WS_CHILD | WS_VISIBLE | AJCVTHS_LOGFILE | AJCVTHS_LFCTRL | AJCVTHS_SEPARATE,
103 :                              0,                                                    // initial x position
104 :                              0,                                                    // initial y position
105 :                              0,                                                    // initial x size
106 :                              0,                                                    // initial y size
107 :                              hwnd,                                                 // parent window handle
108 :                              (HMENU)IDC_VTH,                                       // window menu handle
109 :                              hInst,                                                // program instance handle
110 :                              NULL);                                                // creation parameters
111 :     //----- VT100 設定値ロード ----------------------------------------//
112 :     AjcLoadAllControlSettings(hwnd, TEXT("Settings"), AJCCTL_SELACT_ALL);
113 :     //----- タイトル設定 ----------------------------------------------//
114 :     AjcVthSetTitleText(hWndVth, TEXT(" ウインドタイトル "), -1, -1, NULL);
115 :     //----- ツールチップ設定  ----------------------------------------//
116 :     AjcVthSetTipText(hWndVth, TEXT("ＶＴ１００エミュレーションウインド"));
117 :     //----- メイン・ダイアログ生成 ------------------------------------//
118 :     hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_DLGMAIN), hwnd, AJC_DLGPROC_NAME(Main));
119 :     //  ダイアログサイズ設定
120 :     GetWindowRect(hDlgMain, &r);
121 :     DlgWidth  = r.right - r.left;
122 :     DlgHeight = r.bottom - r.top;
123 :     //----- 初期ウインドサイズ設定 ------------------------------------//
124 :     sty = (int)MAjcGetWindowLong(hWndBack, GWL_STYLE);
125 :     exs = (int)MAjcGetWindowLong(hWndBack, GWL_EXSTYLE);
126 :     AdjustWindowRectEx(&r, sty, FALSE, exs);
127 :     WndWidth  = r.right - r.left;
128 :     WndHeight = r.bottom - r.top + 200;
129 :     SetWindowPos(hwnd, NULL, 0, 0, WndWidth, WndHeight, SWP_NOMOVE);
130 :     //----- ウインド位置, サイズロード --------------------------------//
131 :     AjcLoadWndRect(hwnd, TEXT("WndRect"));
132 :     //----- ウインド表示 ----------------------------------------------//
133 :     ShowWindow(hDlgMain, SW_SHOW);
134 :     ShowWindow(hWndVth , SW_SHOW);
135 :
136 :     return 0;
137 : }
138 : //----- WM_SIZING -----------------------------------------------------------------------------------------------//
139 : AJC_WNDPROC(Back, WM_SIZING         )
140 : {
141 :     LPRECT  p = (LPRECT)lParam;
142 :
143 :     if (p->right - p->left < WndWidth) {
144 :         p->right  = p->left + WndWidth;
145 :     }
146 :     if (p->bottom - p->top < WndHeight) {
147 :         p->bottom = p->top  + WndHeight;
148 :     }
149 :
150 :     return TRUE;
151 : }
152 : //----- WM_SIZE -------------------------------------------------------------------------------------------------//
153 : AJC_WNDPROC(Back, WM_SIZE           )
154 : {
155 :     UI  width  = LOWORD(lParam);
156 :     UI  height = HIWORD(lParam);
157 :
```

```
158 :     //----- ダイアログと VT-100 ウインド移動 ----------------------------//
159 :     MoveWindow(hDlgMain, 0, 0        , width, DlgHeight         , TRUE);
160 :     MoveWindow(hWndVth , 0, DlgHeight, width, height - DlgHeight, TRUE);
161 :
162 :     return 0;
163 : }
164 : //----- WM_DESTROY ------------------------------------------------------------------------------------//
165 : AJC_WNDPROC(Back, WM_DESTROY        )
166 : {
167 :     //----- ウインド位置，サイズセーブ ------------------------------//
168 :     AjcSaveWndRect(hwnd, TEXT("WndRect"));
169 :     //----- VT100 設定値セーブ ---------------------------------------//
170 :     AjcSaveAllControlSettings(hwnd);
171 :     //----- プログラム終了 -------------------------------------------//
172 :     PostQuitMessage(0);
173 :     return 0;
174 : }
175 : //----- ＶＴ１００ コントロールからの通知 ------------------------------------------------------------//
176 : AJC_WNDPROC(Back, IDC_VTH           )
177 : {
178 :     int     cw, ch, sw, sh, sty, exs;
179 :     RECT    r;
180 :     UT      path[MAX_PATH];
181 :
182 :     switch (HIWORD(wParam)) {
183 :         case AJCVTHN_KEYIN:         //  ●キー入力（入力した文字を描画）
184 :             AjcVthPutText(hWndVth, (UTP)&lParam, 1);
185 :             break;
186 :
187 :         case AJCVTHN_VKEYIN:        //  ●拡張キー押下（↑↓←→でカーソル移動）
188 :             switch (lParam) {
189 :                 case VK_UP:     AjcVthPutText(hWndVth, TEXT("¥x1B[A"), -1); break;
190 :                 case VK_DOWN:   AjcVthPutText(hWndVth, TEXT("¥x1B[B"), -1); break;
191 :                 case VK_RIGHT:  AjcVthPutText(hWndVth, TEXT("¥x1B[C"), -1); break;
192 :                 case VK_LEFT:   AjcVthPutText(hWndVth, TEXT("¥x1B[D"), -1); break;
193 :             }
194 :             break;
195 :
196 :         case AJCVTHN_DBLCLK:        //  ●ダブルクリック（ウインドをＶＲＡＭサイズに合わせる）
197 :             AjcVthGetVramFitSize(hWndVth, &cw, &ch);
198 :             r.top = r.left = 0;
199 :             r.right  = __max(DlgWidth, cw);
200 :             r.bottom = DlgHeight + ch;
201 :             sty = (int)MAjcGetWindowLong(hWndBack, GWL_STYLE);
202 :             exs = (int)MAjcGetWindowLong(hWndBack, GWL_EXSTYLE);
203 :             AdjustWindowRectEx(&r, sty, FALSE, exs);
204 :             sw = GetSystemMetrics(SM_CXSCREEN) * 90 / 100;
205 :             sh = GetSystemMetrics(SM_CYSCREEN) * 90 / 100;
206 :             cw = __min(sw, r.right - r.left);
207 :             ch = __min(sh, r.bottom - r.top);
208 :             SetWindowPos(hWndBack, NULL, 0, 0, cw, ch, SWP_NOMOVE);
209 :             break;
210 :
211 :         case AJCVTHN_DROPFILE:      //  ●ファイル ドロップ（ファイルの内容を描画）
212 :             while (AjcVthGetDroppedFile(hWndVth, path)) {
213 :                 HAJCFILE    hFile;
214 :                 WC          wbuf[256];
215 :                 AjcVthPrintF(hWndVth, TEXT("¥r¥n¥x1B[34m[[[ FILE = %s ]]]¥x1B[0m¥r¥n"), path);
216 :                 if (hFile = AjcFOpen(path, AJCTEC_AUTO)) {
217 :                     while (AjcFGetSW(hFile, wbuf, 256)) {
218 :                         AjcVthPutTextW(hWndVth, wbuf, -1);
219 :                     }
220 :                     AjcFClose(hFile);
221 :                 }
222 :             }
223 :             break;
224 :
225 :         case AJCVTHN_DROPDIR:       //  ●ディレクトリ ドロップ（ディレクトリ名表示）
226 :             while (AjcVthGetDroppedDir(hWndVth, path)) {
227 :                 AjcVthPrintF(hWndVth, TEXT("¥r¥n¥x1B[35m[[[ DIR = %s ]]]]¥x1B[0m¥r¥n"), path);
228 :             }
229 :             break;
230 :
231 :         case AJCVTHN_RCLICK:        //  ●右クリック
232 :         {   PAJCVTHRCLK p = (PAJCVTHRCLK)lParam;
233 :             UT          txt[64] = {0};
234 :             AjcSnPrintF(txt, 64, TEXT("%s%s 右クリック発生(x = %d, y = %d)"), p->fShift ? TEXT("Shift+") : TEXT(""),
235 :                                                                               p->fCtrl  ? TEXT("Ctrl+" ) : TEXT(""),
236 :                                                                               p->x, p->y);
237 :             MessageBox(hwnd, txt, TEXT("S_CtrlVT100_01"), MB_OK);
```

```
238 :             break;
239 :         }
240 :     }
241 :     return TRUE;
242 : }
243 : //-------------------------------------------------------------------------------------------------------------//
244 : AJC_WNDMAP_DEF(Back)
245 :     AJC_WNDMAP_MSG(Back, WM_CREATE     )
246 :     AJC_WNDMAP_MSG(Back, WM_SIZE       )
247 :     AJC_WNDMAP_MSG(Back, WM_SIZING     )
248 :     AJC_WNDMAP_MSG(Back, WM_DESTROY    )
249 :     AJC_WNDMAP_CMD(Back, IDC_VTH       )
250 : AJC_WNDMAP_END
251 : //=============================================================================================================//
252 : //                                                                                                             //
253 : //  ダイアログ・プロシージャ                                                                                   //
254 : //                                                                                                             //
255 : //=============================================================================================================//
256 : //----- ダイアログ初期化 ------------------------------------------------------------------------------------//
257 : AJC_DLGPROC(Main, WM_INITDIALOG     )
258 : {
259 :     UI          sty;
260 :
261 :     hDlgMain = hDlg;
262 :
263 :     //----- ダイアログ項目初期設定 ------------------------------------------------------------------//
264 :     sty = (UI)MAjcGetWindowLong(hWndVth, GWL_STYLE);
265 :     AjcSetDlgItemChk(hDlg, IDC_CHK_LOGFILE   , (sty & AJCVTHS_LOGFILE   ) != 0);
266 :     AjcSetDlgItemChk(hDlg, IDC_CHK_FIXEDPITCH, (sty & AJCVTHS_FIXEDPITCH) != 0);
267 :     AjcSetDlgItemChk(hDlg, IDC_CHK_FAST      , (sty & AJCVTHS_FAST      ) != 0);
268 :     AjcSetDlgItemChk(hDlg, IDC_CHK_FIXEDFONT , (sty & AJCVTHS_FIXEDFONT ) != 0);
269 :     AjcSetDlgItemChk(hDlg, IDC_CHK_FIXEDLF   , (sty & AJCVTHS_FIXEDFONT ) != 0);
270 :     AjcSetDlgItemChk(hDlg, IDC_CHK_NOVSCROLL , (sty & AJCVTHS_NOVSCROLL ) != 0);
271 :     AjcSetDlgItemChk(hDlg, IDC_CHK_NOHSCROLL , (sty & AJCVTHS_NOHSCROLL ) != 0);
272 :     AjcSetDlgItemChk(hDlg, IDC_CHK_NOSCRL    , (sty & AJCVTHS_NOSCRL    ) != 0);
273 :     AjcSetDlgItemChk(hDlg, IDC_CHK_NOBORDER  , (sty & AJCVTHS_NOBORDER  ) != 0);
274 :     AjcSetDlgItemChk(hDlg, IDC_CHK_NOSCRLOUT , (sty & AJCVTHS_NOSCRLOUT ) != 0);
275 :     AjcSetDlgItemChk(hDlg, IDC_CHK_CRCTRL    , (sty & AJCVTHS_CRCTRL    ) != 0);
276 :     AjcSetDlgItemChk(hDlg, IDC_CHK_LFCTRL    , (sty & AJCVTHS_LFCTRL    ) != 0);
277 :     AjcSetDlgItemChk(hDlg, IDC_CHK_SEPARATE  , (sty & AJCVTHS_SEPARATE  ) != 0);
278 :
279 :     sty = (UI)MAjcGetWindowLong(hWndVth, GWL_EXSTYLE);
280 :     AjcSetDlgItemChk(hDlg, IDC_CHK_ACCEPTFILES, (sty & WS_EX_ACCEPTFILES) != 0);
281 :
282 :     AjcSetDlgItemUInt(hDlg, IDC_TXT_LINE, 10);
283 :     AjcSetDlgItemUInt(hDlg, IDC_TXT_COL , 40);
284 :     AjcSetDlgItemUInt(hDlg, IDC_INP_TXTCOLOR, 0);
285 :     AjcSetDlgItemUInt(hDlg, IDC_INP_BKCOLOR , 7);
286 :     AjcSetDlgItemUInt(hDlg, IDC_INP_BKWND   , 7);
287 :
288 :     //----- 初期メッセージ表示 ----------------------------------------------------------------------//
289 :     AjcVthPrintF(hWndVth, TEXT("¥x1B[35;46m"));
290 :     AjcVthPrintF(hWndVth, TEXT("      【 ＶＴ－１００エミュレーション ウインド 】                                ¥n"));
291 :     AjcVthPrintF(hWndVth, TEXT("¥x1B[0m"));
292 :     AjcVthPrintF(hWndVth, TEXT("                                                                                ¥n"));
293 :     AjcVthPrintF(hWndVth, TEXT("¥x1B[34;47m"));
294 :     AjcVthPrintF(hWndVth, TEXT("    ・右クリックでポップアップメニューを表示します                              ¥n"));
295 :     AjcVthPrintF(hWndVth, TEXT("    ・↑↓←→キーでカーソルを移動します                                           ¥n"));
296 :     AjcVthPrintF(hWndVth, TEXT("    ・任意の文字キーを押すとその文字を描画します                                ¥n"));
297 :     AjcVthPrintF(hWndVth, TEXT("    ・テキストファイルをドロップすると、そのファイル名に続いてファイルの内容を描画します ¥n"));
298 :     AjcVthPrintF(hWndVth, TEXT("    ・フォルダをドロップすると、そのフォルダ名を表示します                      ¥n"));
299 :     AjcVthPrintF(hWndVth, TEXT("    ・ダブルクリックで、仮想ＶＲＡＭに合わせてウインドサイズを調整します      ¥n"));
300 :     AjcVthPrintF(hWndVth, TEXT("    ・仮想ＶＲＡＭは、２５行×１２０桁に設定されています                       ¥n"));
301 :     AjcVthPrintF(hWndVth, TEXT("                                                                                ¥n"));
302 :     AjcVthPrintF(hWndVth, TEXT("¥x1B[0m¥r¥n"));
303 :
304 :     //----- 設定値ロード ----------------------------------------------------------------------------//
305 :     //  ウインドスタイルをチェックボックスに反映
306 :     SetStyleToCheckBox();
307 :     //  設定値ロード
308 :     AjcDlgItemSetPermAtt(hDlg, IDC_TXT_GETTEXT, AJCCTL_PSEL_EXCLUDE); // GetText は除外
309 :     AjcLoadAllControlSettings(hDlg, TEXT("Settings"), AJCCTL_SELACT_CHKEXCLUDE);
310 :     //  チェックボックスをウインドスタイルに反映
311 :     SetCheckBoxToStyle();
312 :
313 :     return TRUE;
314 : }
315 : //----- ウインド破棄 ----------------------------------------------------------------------------------------//
316 : AJC_DLGPROC(Main, WM_DESTROY        )
317 : {
```

```
318 :     //----- 設定値セーブ ----------------------------------//
319 :     AjcSaveAllControlSettings(hDlg);
320 :
321 :     return TRUE;
322 : }
323 : //----- タイマ ----------------------------------------------------------------------------------------------//
324 : AJC_DLGPROC(Main, WM_TIMER          )
325 : {
326 :     UI          line, col;
327 :     SYSTEMTIME  lt;
328 :     static UI cnt = 1;
329 :
330 :     //----- カーソル移動 ----------------------------------//
331 :     if (AjcGetDlgItemChk(hDlg, IDC_CHK_FIXEDPOS)) {
332 :         line = AjcGetDlgItemUInt(hDlg, IDC_TXT_LINE);
333 :         col  = AjcGetDlgItemUInt(hDlg, IDC_TXT_COL );
334 :         AjcVthLocate(hWndVth, line, col);
335 :     }
336 :     //----- タイムスタンプログ表示 ------------------------//
337 :     GetLocalTime(&lt);
338 :     AjcVthPrintF(hWndVth, TEXT("%5u - %d/%02d/%02d  %02d:%02d:%02d.%03d"), cnt++, lt.wYear, lt.wMonth, lt.wDay,
339 :                                                         lt.wHour, lt.wMinute, lt.wSecond, lt.wMilliseconds);
340 :     //----- 改行と行クリアー ------------------------------//
341 :     if (!AjcGetDlgItemChk(hDlg, IDC_CHK_FIXEDPOS)) {
342 :         AjcVthPrintF(hWndVth, TEXT("¥n¥x1B[K"));
343 :     }
344 :
345 :     return TRUE;
346 : }
347 : //----- COMMAND ---------------------------------------------------------------------------------------------//
348 : AJC_DLGPROC(Main, WM_COMMAND        )
349 : {
350 :     BOOL        rc = FALSE;
351 :     int         cmd = LOWORD(wParam);
352 :     UI          sty;
353 :
354 :     //----- スタイル設定チェックボックス操作 --------------------------------------------//
355 :     if (cmd >= IDC_CHK_LOGFILE && cmd <= IDC_CHK_ACCEPTFILES && HIWORD(wParam) == BN_CLICKED) {
356 :         //----- ウインドスタイル -----//
357 :         SetCheckBoxToStyle();
358 :         //----- 拡張スタイル ---------//
359 :         sty = (UI)MAjcGetWindowLong(hWndVth, GWL_EXSTYLE);
360 :         sty &= ~WS_EX_ACCEPTFILES;
361 :         if (AjcGetDlgItemChk(hDlg, IDC_CHK_ACCEPTFILES)) sty |= WS_EX_ACCEPTFILES;
362 :         MAjcSetWindowLong(hWndVth, GWL_EXSTYLE, sty);
363 :         rc = TRUE;
364 :     }
365 :
366 :     return rc;
367 : }
368 : //----- 「テキスト描画」ボタン ------------------------------------------------------------------------------//
369 : AJC_DLGPROC(Main, IDC_CMD_PUTTEXT)
370 : {
371 :     UT      txt[1024];
372 :     UT      bin[1024];
373 :
374 :     AjcGetDlgItemStr(hDlg, IDC_TXT_PUTTEXT, txt, sizeof txt);
375 :     AjcCLangStrToBin(txt, bin, sizeof bin);
376 :     AjcVthPutText(hWndVth, bin , -1);
377 :
378 :     return TRUE;
379 : }
380 : //----- 「タイムスタンプ ログ開始／停止」ボタン -------------------------------------------------------------//
381 : AJC_DLGPROC(Main, IDC_CMD_LOG    )
382 : {
383 :     if (fTimeStampLog) {
384 :         fTimeStampLog = FALSE;
385 :         KillTimer(hDlg, 1);
386 :         AjcSetDlgItemStr(hDlg, IDC_CMD_LOG, TEXT("タイムスタンプ ログ開始"));
387 :     }
388 :     else {
389 :         fTimeStampLog = TRUE;
390 :         SetTimer(hDlg, 1, 100, NULL);
391 :         AjcSetDlgItemStr(hDlg, IDC_CMD_LOG, TEXT("タイムスタンプ ログ停止"));
392 :     }
393 :     return TRUE;
394 : }
395 : //----- 「文字色」入力 --------------------------------------------------------------------------------------//
396 : AJC_DLGPROC(Main, IDC_INP_TXTCOLOR)
397 : {
```

```
398 :     AJCVTHPROP  prop;
399 :     HWND        hCtrl = GetDlgItem(hDlg, IDC_INP_TXTCOLOR);
400 :
401 :     if (HIWORD(wParam) == AJCIVN_INTVALUE) {
402 :         TxtColor = (UI)lParam;
403 :         AjcVthGetProp(hWndVth, &prop);
404 :         SendMessage(hCtrl, AJCIVM_SETBORDERCOLOR, prop.rgb[TxtColor], 0);
405 :         AjcVthSetColor  (hWndVth, TxtColor);
406 :     }
407 :
408 :     return TRUE;
409 : }
410 : //----- 「背景色」入力 ----------------------------------------------------------------------------------------//
411 : AJC_DLGPROC(Main, IDC_INP_BKCOLOR)
412 : {
413 :     AJCVTHPROP  prop;
414 :     HWND        hCtrl = GetDlgItem(hDlg, IDC_INP_BKCOLOR);
415 :
416 :     if (HIWORD(wParam) == AJCIVN_INTVALUE) {
417 :         BkColor = (UI)lParam;
418 :         AjcVthGetProp(hWndVth, &prop);
419 :         SendMessage(hCtrl, AJCIVM_SETBORDERCOLOR, prop.rgb[BkColor], 0);
420 :         AjcVthSetBkColor(hWndVth, BkColor );
421 :     }
422 :
423 :     return TRUE;
424 : }
425 : //----- 「ウインド背景色」入力 --------------------------------------------------------------------------------//
426 : AJC_DLGPROC(Main, IDC_INP_BKWND)
427 : {
428 :     AJCVTHPROP  prop;
429 :     HWND        hCtrl = GetDlgItem(hDlg, IDC_INP_BKWND);
430 :
431 :     if (HIWORD(wParam) == AJCIVN_INTVALUE) {
432 :         BkWnd = (UI)lParam;
433 :         AjcVthGetProp(hWndVth, &prop);
434 :         SendMessage(hCtrl, AJCIVM_SETBORDERCOLOR, prop.rgb[BkWnd], 0);
435 :         AjcVthSetWndBkColor(hWndVth, BkWnd);
436 :     }
437 :
438 :     return TRUE;
439 : }
440 : //----- 「Enable」ボタン --------------------------------------------------------------------------------------//
441 : AJC_DLGPROC(Main, IDC_CMD_ENABLE    )
442 : {
443 :     EnableWindow(hWndVth, TRUE);
444 :     return TRUE;
445 : }
446 : //----- 「Disable」ボタン -------------------------------------------------------------------------------------//
447 : AJC_DLGPROC(Main, IDC_CMD_DISABLE   )
448 : {
449 :     EnableWindow(hWndVth, FALSE);
450 :     return TRUE;
451 : }
452 : //----- 「Get Text」ボタン ------------------------------------------------------------------------------------//
453 : AJC_DLGPROC(Main, IDC_CMD_GETTEXT   )
454 : {
455 :     UT      txt[1024];
456 :
457 :     GetWindowText(hWndVth, txt, 1024);
458 :     AjcSetDlgItemStr(hDlg, IDC_TXT_GETTEXT, txt);
459 :     return TRUE;
460 : }
461 : //----- 「Set Text」ボタン ------------------------------------------------------------------------------------//
462 : AJC_DLGPROC(Main, IDC_CMD_SETTEXT   )
463 : {
464 :     UT      txt[1024];
465 :
466 :     AjcGetDlgItemStr(hDlg, IDC_TXT_GETTEXT, txt, 1024);
467 :     SetWindowText(hWndVth, txt);
468 :     return TRUE;
469 : }
470 : //----- 「Cancel」ボタン --------------------------------------------------------------------------------------//
471 : AJC_DLGPROC(Main, IDCANCEL          )
472 : {
473 :     DestroyWindow(hDlg);
474 :     return TRUE;
475 : }
476 : //-----------------------------------------------------------------------------------------------------------//
477 : AJC_DLGMAP_DEF(Main)
```

```
478 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG   )
479 :     AJC_DLGMAP_MSG(Main, WM_DESTROY      )
480 :     AJC_DLGMAP_MSG(Main, WM_TIMER        )
481 :     AJC_DLGMAP_MSG(Main, WM_COMMAND      )
482 :     AJC_DLGMAP_CMD(Main, IDC_CMD_PUTTEXT )
483 :     AJC_DLGMAP_CMD(Main, IDC_CMD_LOG     )
484 :     AJC_DLGMAP_CMD(Main, IDC_INP_TXTCOLOR)
485 :     AJC_DLGMAP_CMD(Main, IDC_INP_BKCOLOR )
486 :     AJC_DLGMAP_CMD(Main, IDC_INP_BKWND   )
487 :     AJC_DLGMAP_CMD(Main, IDC_CMD_ENABLE  )
488 :     AJC_DLGMAP_CMD(Main, IDC_CMD_DISABLE )
489 :     AJC_DLGMAP_CMD(Main, IDC_CMD_GETTEXT )
490 :     AJC_DLGMAP_CMD(Main, IDC_CMD_SETTEXT )
491 :     AJC_DLGMAP_CMD(Main, IDCANCEL        )
492 : AJC_DLGMAP_END
493 :
494 : //-------------------------------------------------------------------------------------------------------------//
495 : //  ウインドスタイル値をチェックボックスに設定                                                                 //
496 : //-------------------------------------------------------------------------------------------------------------//
497 : static  VO  SetStyleToCheckBox(VO)
498 : {
499 :     UI      sty, exs;
500 :
501 :     sty = (UI)MAjcGetWindowLong(hWndVth, GWL_STYLE  );
502 :     exs = (UI)MAjcGetWindowLong(hWndVth, GWL_EXSTYLE);
503 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_LOGFILE      , (sty & AJCVTHS_LOGFILE    ) != 0);
504 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_FIXEDPITCH   , (sty & AJCVTHS_FIXEDPITCH ) != 0);
505 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_FAST         , (sty & AJCVTHS_FAST       ) != 0);
506 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_FIXEDFONT    , (sty & AJCVTHS_FIXEDFONT  ) != 0);
507 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_FIXEDLF      , (sty & AJCVTHS_FIXEDLF    ) != 0);
508 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_NOVSCROLL    , (sty & AJCVTHS_NOVSCROLL  ) != 0);
509 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_NOHSCROLL    , (sty & AJCVTHS_NOHSCROLL  ) != 0);
510 :
511 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_NOSCRL       , (sty & AJCVTHS_NOSCRL     ) != 0);
512 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_NOBORDER     , (sty & AJCVTHS_NOBORDER   ) != 0);
513 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_NOSCRLOUT    , (sty & AJCVTHS_NOSCRLOUT  ) != 0);
514 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_CRLFCTRL     , (sty & AJCVTHS_CRLFCTRL   ) != 0);
515 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_CRCTRL       , (sty & AJCVTHS_CRCTRL     ) != 0);
516 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_LFCTRL       , (sty & AJCVTHS_LFCTRL     ) != 0);
517 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_SEPARATE     , (sty & AJCVTHS_SEPARATE   ) != 0);
518 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_ACCEPTFILES  , (exs & WS_EX_ACCEPTFILES  ) != 0);
519 : }
520 : //-------------------------------------------------------------------------------------------------------------//
521 : //  チェックボックスの設定をスタイルに反映                                                                     //
522 : //-------------------------------------------------------------------------------------------------------------//
523 : static  VO  SetCheckBoxToStyle(VO)
524 : {
525 :     UI      sty, exs;
526 :
527 :     sty = (UI)MAjcGetWindowLong(hWndVth, GWL_STYLE);
528 :     sty &= ~(AJCVTHS_LOGFILE    | AJCVTHS_FIXEDPITCH | AJCVTHS_FAST    | AJCVTHS_FIXEDFONT | AJCVTHS_FIXEDLF    |
529 :              AJCVTHS_NOVSCROLL | AJCVTHS_NOHSCROLL  | AJCVTHS_NOSCRL | AJCVTHS_NOBORDER  | AJCVTHS_NOSCRLOUT |
530 :              AJCVTHS_CRLFCTRL  | AJCVTHS_CRCTRL     | AJCVTHS_LFCTRL | AJCVTHS_SEPARATE);
531 :     exs = (UI)MAjcGetWindowLong(hWndVth, GWL_EXSTYLE);
532 :     exs &= ~WS_EX_ACCEPTFILES;
533 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_LOGFILE      )) sty |= AJCVTHS_LOGFILE;
534 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_FIXEDPITCH   )) sty |= AJCVTHS_FIXEDPITCH;
535 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_FAST         )) sty |= AJCVTHS_FAST;
536 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_FIXEDFONT    )) sty |= AJCVTHS_FIXEDFONT;
537 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_FIXEDLF      )) sty |= AJCVTHS_FIXEDLF;
538 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_NOVSCROLL    )) sty |= AJCVTHS_NOVSCROLL;
539 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_NOHSCROLL    )) sty |= AJCVTHS_NOHSCROLL;
540 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_NOSCRL       )) sty |= AJCVTHS_NOSCRL;
541 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_NOBORDER     )) sty |= AJCVTHS_NOBORDER;
542 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_NOSCRLOUT    )) sty |= AJCVTHS_NOSCRLOUT;
543 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_CRLFCTRL     )) sty |= AJCVTHS_CRLFCTRL;
544 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_CRCTRL       )) sty |= AJCVTHS_CRCTRL;
545 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_LFCTRL       )) sty |= AJCVTHS_LFCTRL;
546 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_SEPARATE     )) sty |= AJCVTHS_SEPARATE;
547 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_ACCEPTFILES  )) exs |= WS_EX_ACCEPTFILES;
548 :     MAjcSetWindowLong(hWndVth, GWL_STYLE  , sty);
549 :     MAjcSetWindowLong(hWndVth, GWL_EXSTYLE, exs);
550 : }
551 :
552 :
```

7.14. SW_VT100__MesTime（VT100 描画とウインド表示時間の計測）

このサンプルプログラムは、ＶＴ－１００エミュレーションウインド・コントロールで、仮想ＶＲＡＭへのテキスト描画時間と、ウインドの表示時間（WM_PAINT メッセージの処理時間）を測定します。（測定単位はμ秒）

表示ファイルに適当なテキストファイルをドロップし、「START」ボタンを押すと、テキストファイルを表示しながら、仮想ＶＲＡＭへのテキスト描画時間と、ウインドの表示処理時間を測定します。テキストファイルの末端に達したら、再度先頭から表示します。
テキストは１行づつ、「表示周期」で設定された周期で表示します。
Windows のタイマは解像度が低いため、正確に「表示周期」で設定された間隔で表示されません。
「実際の表示タイマ周期」で実際に測定した周期を表示します。

「実際の表示タイマ周期」は、テキストの表示を行っていない間も常に測定値を表示します。
テキストの表示中の「実際の表示タイマ周期」が、停止中の「実際の表示タイマ周期」より長い場合は、テキストの描画＋ウインドの表示処理が重く、所定の周期内に処理できていないことを意味します。

「キャレット表示」「固定ピッチ」「高速表示」をチェックし、当該モードでの処理時間を測定できます。

尚、ＶＴ－１００エミュレーションウインド・コントロールでは、ウインドが以下のように構成されており、実際にテキストの表示を行っているウインドは、子ウインド（AjcVT100MainClass）です。



本サンプルプログラムでは、子ウインド（AjcVT100MainClass）をサブクラス化し、WM_PAINT メッセージをトラップします。



```
 1 : //
 2 : //  SW_VT100.c
 3 : //
 4 : #include    <AjrCstXX.h>
 5 : #include    <math.h>
 6 : #include    <tchar.h>
 7 : #include    "resource.h"
 8 :
 9 : #define     IDC_VTH     5000
10 :
11 : //-------------------------------------------------------------------------------------------------------------------//
12 : //  ワーク                                                                                                           //
13 : //-------------------------------------------------------------------------------------------------------------------//
14 : HINSTANCE       hInst;                      //  ＤＬＬインスタンスハンドル
15 : HWND            hWndBack;                   //  バックウインドハンドル（ダイアログと VT-100 の親ウインド）
16 : HWND            hDlgMain;                   //  ダイアログボックスハンドル
17 : HWND            hWndVth;                    //  タイムチャート・グラフコントロールのウインドハンドル
18 :
19 : int             WndWidth, WndHeight;        //  ウインドの幅と高さ
20 : int             DlgWidth, DlgHeight;        //  ダイアログの幅と高さ
21 : BOOL            fTimeStampLog = FALSE;      //  タイムスタンプログ動作状態
22 : UI              TxtColor, BkColor, BkWnd;   //  描画色（パレット番号）
23 :
24 : //-------------------------------------------------------------------------------------------------------------------//
25 : //  内部サブ関数                                                                                                     //
26 : //-------------------------------------------------------------------------------------------------------------------//
27 : AJC_WNDPROC_DEF(Back);
28 : AJC_DLGPROC_DEF(Main);
29 : static  VO  SetStyleToCheckBox(VO);
30 : static  VO  SetCheckBoxToStyle(VO);
31 :
32 : //===================================================================================================================//
33 : //                                                                                                                   //
34 : //  Ｗ ｉ ｎ Ｍ ａ ｉ ｎ                                                                                             //
35 : //                                                                                                                   //
36 : //===================================================================================================================//
37 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
38 : {
39 :     MSG         msg;
40 :     WNDCLASS    wndclass;
41 :     hInst = hInstance;
42 :
43 :     //----- バックウインド生成 -------------------------//
44 :     wndclass.style          = 0;
45 :     wndclass.lpfnWndProc    = AJC_WNDPROC_NAME(Back);
46 :     wndclass.cbClsExtra     = 0;
47 :     wndclass.cbWndExtra     = 0;
48 :     wndclass.hInstance      = hInst;
49 :     wndclass.hIcon          = NULL;
50 :     wndclass.hCursor        = LoadCursor(NULL, IDC_ARROW);
51 :     wndclass.hbrBackground  = (HBRUSH)GetStockObject(WHITE_BRUSH);
52 :     wndclass.lpszMenuName   = NULL;
53 :     wndclass.lpszClassName  = TEXT("SW_VT100");
54 :     RegisterClass(&wndclass);
55 :
56 :     hWndBack = CreateWindow(TEXT("SW_VT100"),           // window class name
57 :                             TEXT("SW_VT100"),           // window caption
58 :                             WS_OVERLAPPEDWINDOW,        // window style
59 :                             0,                          // initial x position
60 :                             0,                          // initial y position
61 :                             0,                          // initial x size
62 :                             0,                          // initial y size
63 :                             NULL,                       // parent window handle
64 :                             NULL,                       // window menu handle
65 :                             hInst,                      // program instance handle
66 :                             NULL);                      // creation parameters
67 :
68 :     //----- ウインド表示 -------------------------------//
69 :     ShowWindow(hWndBack, iCmdShow);
70 :     //----- ＶＴ－１００ウインドをフォーカス -----------//
71 :     SetFocus(hWndVth);
72 :
73 :     //----- メッセージループ ---------------------------//
74 :     while (GetMessage(&msg, NULL, 0, 0)) {
75 :         do {
76 :             if (IsDialogMessage(hDlgMain, &msg)) break;
77 :             TranslateMessage(&msg);
78 :             DispatchMessage (&msg);
79 :         } while (0);
```

```
    }

    return (int)msg.wParam ;
}
//=============================================================================================================//
//                                                                                                             //
//  バックウインド・プロシージャ                                                                               //
//                                                                                                             //
//=============================================================================================================//
//----- WM_CREATE -------------------------------------------------------------------------------------------//
AJC_WNDPROC(Back, WM_CREATE         )
{
    RECT    r;
    int     sty = (int)MAjcGetWindowLong(hwnd, GWL_STYLE);
    int     exs = (int)MAjcGetWindowLong(hwnd, GWL_EXSTYLE);

    hWndBack = hwnd;

    //----- VT-100 エミュレーション・コントロール生成 -----------------//
    hWndVth = CreateWindowEx(WS_EX_ACCEPTFILES,                                         // extend style
                             TEXT("AjcCtrlVT100"),                                      // window class name
                             TEXT("P: VW=120, VH=25, FN=MS Gothic"),                    // window caption
                             WS_CHILD | WS_VISIBLE | AJCVTHS_LOGFILE | AJCVTHS_LFCTRL | AJCVTHS_SEPARATE,
                             0,                                                         // initial x position
                             0,                                                         // initial y position
                             0,                                                         // initial x size
                             0,                                                         // initial y size
                             hwnd,                                                      // parent window handle
                             (HMENU)IDC_VTH,                                            // window menu handle
                             hInst,                                                     // program instance handle
                             NULL);                                                     // creation parameters
    //----- VT100 設定値ロード ----------------------------------------//
    AjcLoadAllControlSettings(hwnd, TEXT("Settings"), AJCCTL_SELACT_ALL);
    //----- 文字列検索情報セクションの設定と読み出し ------------------//
    AjcVthSetFindProfileSect(hWndVth, TEXT("FindSect"));
    //----- タイトル設定 ----------------------------------------------//
    AjcVthSetTitleText(hWndVth, TEXT(" ウインドタイトル "), -1, -1, NULL);
    //----- ツールチップ設定  ----------------------------------------//
    AjcVthSetTipText(hWndVth, TEXT("ＶＴ１００エミュレーションウインド"));
    //----- メイン・ダイアログ生成 ------------------------------------//
    hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_DLGMAIN), hwnd, AJC_DLGPROC_NAME(Main));
    //  ダイアログサイズ設定
    GetWindowRect(hDlgMain, &r);
    DlgWidth  = r.right - r.left;
    DlgHeight = r.bottom - r.top;
    //----- 初期ウインドサイズ設定 ------------------------------------//
    sty = (int)MAjcGetWindowLong(hWndBack, GWL_STYLE);
    exs = (int)MAjcGetWindowLong(hWndBack, GWL_EXSTYLE);
    AdjustWindowRectEx(&r, sty, FALSE, exs);
    WndWidth  = r.right - r.left;
    WndHeight = r.bottom - r.top + 200;
    SetWindowPos(hwnd, NULL, 0, 0, WndWidth, WndHeight, SWP_NOMOVE);
    //----- ウインド位置, サイズロード --------------------------------//
    AjcLoadWndRect(hwnd, TEXT("WndRect"));
    //----- ウインド表示 ----------------------------------------------//
    ShowWindow(hDlgMain, SW_SHOW);
    ShowWindow(hWndVth , SW_SHOW);

    return 0;
}
//----- WM_SIZING -------------------------------------------------------------------------------------------//
AJC_WNDPROC(Back, WM_SIZING         )
{
    LPRECT  p = (LPRECT)lParam;

    if (p->right - p->left < WndWidth) {
        p->right  = p->left + WndWidth;
    }
    if (p->bottom - p->top < WndHeight) {
        p->bottom = p->top  + WndHeight;
    }

    return TRUE;
}
//----- WM_SIZE ---------------------------------------------------------------------------------------------//
AJC_WNDPROC(Back, WM_SIZE           )
{
    UI  width  = LOWORD(lParam);
    UI  height = HIWORD(lParam);

```

```
160 :     //----- ダイアログと VT-100 ウインド移動 ----------------------------//
161 :     MoveWindow(hDlgMain, 0, 0        , width, DlgHeight         , TRUE);
162 :     MoveWindow(hWndVth , 0, DlgHeight, width, height - DlgHeight, TRUE);
163 :
164 :     return 0;
165 : }
166 : //----- WM_DESTROY ------------------------------------------------------------------------------------//
167 : AJC_WNDPROC(Back, WM_DESTROY        )
168 : {
169 :     //----- ウインド位置，サイズセーブ ------------------------------//
170 :     AjcSaveWndRect(hwnd, TEXT("WndRect"));
171 :     //----- VT100 設定値セーブ ---------------------------------------//
172 :     AjcSaveAllControlSettings(hwnd);
173 :     //----- プログラム終了 -------------------------------------------//
174 :     PostQuitMessage(0);
175 :     return 0;
176 : }
177 : //----- ＶＴ１００ コントロールからの通知 ------------------------------------------------------------//
178 : AJC_WNDPROC(Back, IDC_VTH           )
179 : {
180 :     int     cw, ch, sw, sh, sty, exs;
181 :     RECT    r;
182 :     UT      path[MAX_PATH];
183 :
184 :     switch (HIWORD(wParam)) {
185 :         case AJCVTHN_KEYIN:         //  ●キー入力（入力した文字を描画）
186 :             AjcVthPutText(hWndVth, (UTP)&lParam, 1);
187 :             break;
188 :
189 :         case AJCVTHN_VKEYIN:        //  ●拡張キー押下（↑↓←→でカーソル移動）
190 :             switch (lParam) {
191 :                 case VK_UP:     AjcVthPutText(hWndVth, TEXT("¥x1B[A"), -1); break;
192 :                 case VK_DOWN:   AjcVthPutText(hWndVth, TEXT("¥x1B[B"), -1); break;
193 :                 case VK_RIGHT:  AjcVthPutText(hWndVth, TEXT("¥x1B[C"), -1); break;
194 :                 case VK_LEFT:   AjcVthPutText(hWndVth, TEXT("¥x1B[D"), -1); break;
195 :             }
196 :             break;
197 :
198 :         case AJCVTHN_DBLCLK:        //  ●ダブルクリック（ウインドをＶＲＡＭサイズに合わせる）
199 :             AjcVthGetVramFitSize(hWndVth, &cw, &ch);
200 :             r.top = r.left = 0;
201 :             r.right  = __max(DlgWidth, cw);
202 :             r.bottom = DlgHeight + ch;
203 :             sty = (int)MAjcGetWindowLong(hWndBack, GWL_STYLE);
204 :             exs = (int)MAjcGetWindowLong(hWndBack, GWL_EXSTYLE);
205 :             AdjustWindowRectEx(&r, sty, FALSE, exs);
206 :             sw = GetSystemMetrics(SM_CXSCREEN) * 90 / 100;
207 :             sh = GetSystemMetrics(SM_CYSCREEN) * 90 / 100;
208 :             cw = __min(sw, r.right - r.left);
209 :             ch = __min(sh, r.bottom - r.top);
210 :             SetWindowPos(hWndBack, NULL, 0, 0, cw, ch, SWP_NOMOVE);
211 :             break;
212 :
213 :         case AJCVTHN_DROPFILE:      //  ●ファイル ドロップ（ファイルの内容を描画）
214 :             while (AjcVthGetDroppedFile(hWndVth, path)) {
215 :                 HAJCFILE    hFile;
216 :                 WC          wbuf[256];
217 :                 AjcVthPrintF(hWndVth, TEXT("¥r¥n¥x1B[34m[[[ FILE = %s ]]]¥x1B[0m¥r¥n"), path);
218 :                 if (hFile = AjcFOpen(path, AJCTEC_AUTO)) {
219 :                     while (AjcFGetSW(hFile, wbuf, 256)) {
220 :                         AjcVthPutTextW(hWndVth, wbuf, -1);
221 :                     }
222 :                     AjcFClose(hFile);
223 :                 }
224 :             }
225 :             break;
226 :
227 :         case AJCVTHN_DROPDIR:       //  ●ディレクトリ ドロップ（ディレクトリ名表示）
228 :             while (AjcVthGetDroppedDir(hWndVth, path)) {
229 :                 AjcVthPrintF(hWndVth, TEXT("¥r¥n¥x1B[35m[[[ DIR = %s ]]]]¥x1B[0m¥r¥n"), path);
230 :             }
231 :             break;
232 :
233 :         case AJCVTHN_RCLICK:        //  ●右クリック
234 :         {   PAJCVTHRCLK p = (PAJCVTHRCLK)lParam;
235 :             UT          txt[64] = {0};
236 :             AjcSnPrintF(txt, 64, TEXT("%s%s 右クリック発生(x = %d, y = %d)"), p->fShift ? TEXT("Shift+") : TEXT(""),
237 :                                                                               p->fCtrl  ? TEXT("Ctrl+" ) : TEXT(""),
238 :                                                                               p->x, p->y);
239 :             MessageBox(hwnd, txt, TEXT("S_CtrlVT100_01"), MB_OK);
```

```
240 :             break;
241 :         }
242 :     }
243 :     return TRUE;
244 : }
245 : //-------------------------------------------------------------------------------------------------------------//
246 : AJC_WNDMAP_DEF(Back)
247 :     AJC_WNDMAP_MSG(Back, WM_CREATE     )
248 :     AJC_WNDMAP_MSG(Back, WM_SIZE       )
249 :     AJC_WNDMAP_MSG(Back, WM_SIZING     )
250 :     AJC_WNDMAP_MSG(Back, WM_DESTROY    )
251 :     AJC_WNDMAP_CMD(Back, IDC_VTH       )
252 : AJC_WNDMAP_END
253 : //=============================================================================================================//
254 : //                                                                                                             //
255 : //  ダイアログ・プロシージャ                                                                                   //
256 : //                                                                                                             //
257 : //=============================================================================================================//
258 : //----- ダイアログ初期化 ------------------------------------------------------------------------------------//
259 : AJC_DLGPROC(Main, WM_INITDIALOG     )
260 : {
261 :     UI          sty;
262 :
263 :     hDlgMain = hDlg;
264 :
265 :     //----- ダイアログ項目初期設定 ------------------------------------------------------------------//
266 :     sty = (UI)MAjcGetWindowLong(hWndVth, GWL_STYLE);
267 :     AjcSetDlgItemChk(hDlg, IDC_CHK_LOGFILE   , (sty & AJCVTHS_LOGFILE   ) != 0);
268 :     AjcSetDlgItemChk(hDlg, IDC_CHK_FIXEDPITCH, (sty & AJCVTHS_FIXEDPITCH) != 0);
269 :     AjcSetDlgItemChk(hDlg, IDC_CHK_FAST      , (sty & AJCVTHS_FAST      ) != 0);
270 :     AjcSetDlgItemChk(hDlg, IDC_CHK_NOCLSBTN  , (sty & AJCVTHS_NOCLSBTN  ) != 0);
271 :     AjcSetDlgItemChk(hDlg, IDC_CHK_FIXEDFONT , (sty & AJCVTHS_FIXEDFONT ) != 0);
272 :     AjcSetDlgItemChk(hDlg, IDC_CHK_FIXEDLF   , (sty & AJCVTHS_FIXEDFONT ) != 0);
273 :     AjcSetDlgItemChk(hDlg, IDC_CHK_NOVSCROLL , (sty & AJCVTHS_NOVSCROLL ) != 0);
274 :     AjcSetDlgItemChk(hDlg, IDC_CHK_NOHSCROLL , (sty & AJCVTHS_NOHSCROLL ) != 0);
275 :     AjcSetDlgItemChk(hDlg, IDC_CHK_NOSCRL    , (sty & AJCVTHS_NOSCRL    ) != 0);
276 :     AjcSetDlgItemChk(hDlg, IDC_CHK_NOBORDER  , (sty & AJCVTHS_NOBORDER  ) != 0);
277 :     AjcSetDlgItemChk(hDlg, IDC_CHK_NOSCRLOUT , (sty & AJCVTHS_NOSCRLOUT ) != 0);
278 :     AjcSetDlgItemChk(hDlg, IDC_CHK_CRCTRL    , (sty & AJCVTHS_CRCTRL    ) != 0);
279 :     AjcSetDlgItemChk(hDlg, IDC_CHK_LFCTRL    , (sty & AJCVTHS_LFCTRL    ) != 0);
280 :     AjcSetDlgItemChk(hDlg, IDC_CHK_SEPARATE  , (sty & AJCVTHS_SEPARATE  ) != 0);
281 :
282 :     sty = (UI)MAjcGetWindowLong(hWndVth, GWL_EXSTYLE);
283 :     AjcSetDlgItemChk(hDlg, IDC_CHK_ACCEPTFILES, (sty & WS_EX_ACCEPTFILES) != 0);
284 :
285 :     AjcSetDlgItemUInt(hDlg, IDC_TXT_LINE, 10);
286 :     AjcSetDlgItemUInt(hDlg, IDC_TXT_COL , 40);
287 :     AjcSetDlgItemUInt(hDlg, IDC_INP_TXTCOLOR, 0);
288 :     AjcSetDlgItemUInt(hDlg, IDC_INP_BKCOLOR , 7);
289 :     AjcSetDlgItemUInt(hDlg, IDC_INP_BKWND   , 7);
290 :
291 :     //----- 初期メッセージ表示 ----------------------------------------------------------------//
292 :     AjcVthPrintF(hWndVth, TEXT("¥x1B[35;46m"));
293 :     AjcVthPrintF(hWndVth, TEXT("      【 ＶＴ－１００エミュレーション ウインド 】                                ¥n"));
294 :     AjcVthPrintF(hWndVth, TEXT("¥x1B[0m"));
295 :     AjcVthPrintF(hWndVth, TEXT("                                                                                ¥n"));
296 :     AjcVthPrintF(hWndVth, TEXT("¥x1B[34;47m"));
297 :     AjcVthPrintF(hWndVth, TEXT("    ・右クリックでポップアップメニューを表示します                              ¥n"));
298 :     AjcVthPrintF(hWndVth, TEXT("    ・↑↓←→キーでカーソルを移動します                                          ¥n"));
299 :     AjcVthPrintF(hWndVth, TEXT("    ・任意の文字キーを押すとその文字を描画します                                ¥n"));
300 :     AjcVthPrintF(hWndVth, TEXT("    ・テキストファイルをドロップすると、そのファイル名に続いてファイルの内容を描画します ¥n"));
301 :     AjcVthPrintF(hWndVth, TEXT("    ・フォルダをドロップすると、そのフォルダ名を表示します                      ¥n"));
302 :     AjcVthPrintF(hWndVth, TEXT("    ・ダブルクリックで、仮想ＶＲＡＭに合わせてウインドサイズを調整します      ¥n"));
303 :     AjcVthPrintF(hWndVth, TEXT("    ・仮想ＶＲＡＭは、２５行×１２０桁に設定されています                      ¥n"));
304 :     AjcVthPrintF(hWndVth, TEXT("                                                                                ¥n"));
305 :     AjcVthPrintF(hWndVth, TEXT("¥x1B[0m¥r¥n"));
306 :
307 :     //----- 設定値ロード ----------------------------------------------------------------------//
308 :     //  ウインドスタイルをチェックボックスに反映
309 :     SetStyleToCheckBox();
310 :     //  設定値ロード
311 :     AjcDlgItemSetPermAtt(hDlg, IDC_TXT_GETTEXT, AJCCTL_PSEL_EXCLUDE);                  // GetText は除外
312 :     AjcLoadAllControlSettings(hDlg, TEXT("Settings"), AJCCTL_SELACT_CHKEXCLUDE);       // ChkBox は除外
313 :     SendMessage(hDlg, WM_COMMAND, MAKELONG(IDC_CHK_PAUSE, BN_CLICKED), (LPARAM)hDlg);  // PAUSE チェックボタン
314 :     //  チェックボックスをウインドスタイルに反映
315 :     SetCheckBoxToStyle();
316 :
317 :     return TRUE;
318 : }
319 : //----- ウインド破棄 ----------------------------------------------------------------------------------------//
```

```
320 : AJC_DLGPROC(Main, WM_DESTROY        )
321 : {
322 :     //----- 設定値セーブ ----------------------------------//
323 :     AjcSaveAllControlSettings(hDlg);
324 :
325 :     return TRUE;
326 : }
327 : //----- タイマ ----------------------------------------------------------------------------------------------//
328 : AJC_DLGPROC(Main, WM_TIMER          )
329 : {
330 :     UI          line, col;
331 :     SYSTEMTIME  lt;
332 :     static UI cnt = 1;
333 :
334 :     //----- カーソル移動 ----------------------------------//
335 :     if (AjcGetDlgItemChk(hDlg, IDC_CHK_FIXEDPOS)) {
336 :         line = AjcGetDlgItemUInt(hDlg, IDC_TXT_LINE);
337 :         col  = AjcGetDlgItemUInt(hDlg, IDC_TXT_COL );
338 :         AjcVthLocate(hWndVth, line, col);
339 :     }
340 :     //----- タイムスタンプログ表示 ------------------------//
341 :     GetLocalTime(&lt);
342 :     AjcVthPrintF(hWndVth, TEXT("%5u - %d/%02d/%02d  %02d:%02d:%02d.%03d"), cnt++, lt.wYear, lt.wMonth, lt.wDay,
343 :                                                         lt.wHour, lt.wMinute, lt.wSecond, lt.wMilliseconds);
344 :     //----- 改行と行クリアー ------------------------------//
345 :     if (!AjcGetDlgItemChk(hDlg, IDC_CHK_FIXEDPOS)) {
346 :         AjcVthPrintF(hWndVth, TEXT("¥n¥x1B[K"));
347 :     }
348 :
349 :     return TRUE;
350 : }
351 : //----- COMMAND ---------------------------------------------------------------------------------------------//
352 : AJC_DLGPROC(Main, WM_COMMAND        )
353 : {
354 :     BOOL        rc = FALSE;
355 :     int         cmd = LOWORD(wParam);
356 :     UI          sty;
357 :
358 :     //----- スタイル設定チェックボックス操作 --------------------------------------------------//
359 :     if (cmd >= IDC_CHK_LOGFILE && cmd <= IDC_CHK_ACCEPTFILES && HIWORD(wParam) == BN_CLICKED) {
360 :         //----- ウインドスタイル -----//
361 :         SetCheckBoxToStyle();
362 :         //----- 拡張スタイル ---------//
363 :         sty = (UI)MAjcGetWindowLong(hWndVth, GWL_EXSTYLE);
364 :         sty &= ~WS_EX_ACCEPTFILES;
365 :         if (AjcGetDlgItemChk(hDlg, IDC_CHK_ACCEPTFILES)) sty |= WS_EX_ACCEPTFILES;
366 :         MAjcSetWindowLong(hWndVth, GWL_EXSTYLE, sty);
367 :         rc = TRUE;
368 :     }
369 :
370 :     return rc;
371 : }
372 : //----- 「テキスト描画」ボタン ------------------------------------------------------------------------------//
373 : AJC_DLGPROC(Main, IDC_CMD_PUTTEXT)
374 : {
375 :     UT      txt[1024];
376 :     UT      bin[1024];
377 :
378 :     AjcGetDlgItemStr(hDlg, IDC_TXT_PUTTEXT, txt, sizeof txt);
379 :     AjcCLangStrToBin(txt, bin, sizeof bin);
380 :     AjcVthPutText(hWndVth, bin , -1);
381 :
382 :     return TRUE;
383 : }
384 : //----- 「タイムスタンプ ログ開始／停止」ボタン ------------------------------------------------------------//
385 : AJC_DLGPROC(Main, IDC_CMD_LOG    )
386 : {
387 :     if (fTimeStampLog) {
388 :         fTimeStampLog = FALSE;
389 :         KillTimer(hDlg, 1);
390 :         AjcSetDlgItemStr(hDlg, IDC_CMD_LOG, TEXT("タイムスタンプ ログ開始"));
391 :     }
392 :     else {
393 :         fTimeStampLog = TRUE;
394 :         SetTimer(hDlg, 1, 100, NULL);
395 :         AjcSetDlgItemStr(hDlg, IDC_CMD_LOG, TEXT("タイムスタンプ ログ停止"));
396 :     }
397 :     return TRUE;
398 : }
399 : //----- 「文字色」入力 --------------------------------------------------------------------------------------//
```

```
400 : AJC_DLGPROC(Main, IDC_INP_TXTCOLOR)
401 : {
402 :     AJCVTHPROP  prop;
403 :     HWND        hCtrl = GetDlgItem(hDlg, IDC_INP_TXTCOLOR);
404 :
405 :     if (HIWORD(wParam) == AJCIVN_INTVALUE) {
406 :         TxtColor = (UI)lParam;
407 :         AjcVthGetProp(hWndVth, &prop);
408 :         SendMessage(hCtrl, AJCIVM_SETBORDERCOLOR, prop.rgb[TxtColor], 0);
409 :         AjcVthSetColor  (hWndVth, TxtColor);
410 :     }
411 :
412 :     return TRUE;
413 : }
414 : //-----「背景色」入力 ----------------------------------------------------------------------------------//
415 : AJC_DLGPROC(Main, IDC_INP_BKCOLOR)
416 : {
417 :     AJCVTHPROP  prop;
418 :     HWND        hCtrl = GetDlgItem(hDlg, IDC_INP_BKCOLOR);
419 :
420 :     if (HIWORD(wParam) == AJCIVN_INTVALUE) {
421 :         BkColor = (UI)lParam;
422 :         AjcVthGetProp(hWndVth, &prop);
423 :         SendMessage(hCtrl, AJCIVM_SETBORDERCOLOR, prop.rgb[BkColor], 0);
424 :         AjcVthSetBkColor(hWndVth, BkColor );
425 :     }
426 :
427 :     return TRUE;
428 : }
429 : //-----「ウインド背景色」入力 --------------------------------------------------------------------------//
430 : AJC_DLGPROC(Main, IDC_INP_BKWND)
431 : {
432 :     AJCVTHPROP  prop;
433 :     HWND        hCtrl = GetDlgItem(hDlg, IDC_INP_BKWND);
434 :
435 :     if (HIWORD(wParam) == AJCIVN_INTVALUE) {
436 :         BkWnd = (UI)lParam;
437 :         AjcVthGetProp(hWndVth, &prop);
438 :         SendMessage(hCtrl, AJCIVM_SETBORDERCOLOR, prop.rgb[BkWnd], 0);
439 :         AjcVthSetWndBkColor(hWndVth, BkWnd);
440 :     }
441 :
442 :     return TRUE;
443 : }
444 : //-----「Enable」ボタン --------------------------------------------------------------------------------//
445 : AJC_DLGPROC(Main, IDC_CMD_ENABLE    )
446 : {
447 :     EnableWindow(hWndVth, TRUE);
448 :     return TRUE;
449 : }
450 : //-----「Disable」ボタン -------------------------------------------------------------------------------//
451 : AJC_DLGPROC(Main, IDC_CMD_DISABLE   )
452 : {
453 :     EnableWindow(hWndVth, FALSE);
454 :     return TRUE;
455 : }
456 : //-----「Get Text」ボタン -------------------------------------------------------------------------------//
457 : AJC_DLGPROC(Main, IDC_CMD_GETTEXT   )
458 : {
459 :     UT      txt[1024];
460 :
461 :     GetWindowText(hWndVth, txt, 1024);
462 :     AjcSetDlgItemStr(hDlg, IDC_TXT_GETTEXT, txt);
463 :     return TRUE;
464 : }
465 : //-----「Set Text」ボタン -------------------------------------------------------------------------------//
466 : AJC_DLGPROC(Main, IDC_CMD_SETTEXT   )
467 : {
468 :     UT      txt[1024];
469 :
470 :     AjcGetDlgItemStr(hDlg, IDC_TXT_GETTEXT, txt, 1024);
471 :     SetWindowText(hWndVth, txt);
472 :     return TRUE;
473 : }
474 : //-----「PAUSE」チェックボタン -------------------------------------------------------------------------//
475 : AJC_DLGPROC(Main, IDC_CHK_PAUSE     )
476 : {
477 :     AjcVthPause(hWndVth, AjcGetDlgItemChk(hDlg, IDC_CHK_PAUSE));
478 :     return TRUE;
479 : }
```

```
480 : //----- 「Cancel」ボタン ----------------------------------------------------------------------------------------//
481 : AJC_DLGPROC(Main, IDCANCEL          )
482 : {
483 :     DestroyWindow(hDlg);
484 :     return TRUE;
485 : }
486 : //--------------------------------------------------------------------------------------------------------------//
487 : AJC_DLGMAP_DEF(Main)
488 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG   )
489 :     AJC_DLGMAP_MSG(Main, WM_DESTROY      )
490 :     AJC_DLGMAP_MSG(Main, WM_TIMER        )
491 :     AJC_DLGMAP_MSG(Main, WM_COMMAND      )
492 :     AJC_DLGMAP_CMD(Main, IDC_CMD_PUTTEXT )
493 :     AJC_DLGMAP_CMD(Main, IDC_CMD_LOG     )
494 :     AJC_DLGMAP_CMD(Main, IDC_INP_TXTCOLOR)
495 :     AJC_DLGMAP_CMD(Main, IDC_INP_BKCOLOR )
496 :     AJC_DLGMAP_CMD(Main, IDC_INP_BKWND   )
497 :     AJC_DLGMAP_CMD(Main, IDC_CMD_ENABLE  )
498 :     AJC_DLGMAP_CMD(Main, IDC_CMD_DISABLE )
499 :     AJC_DLGMAP_CMD(Main, IDC_CMD_GETTEXT )
500 :     AJC_DLGMAP_CMD(Main, IDC_CMD_SETTEXT )
501 :     AJC_DLGMAP_CMD(Main, IDC_CHK_PAUSE   )
502 :     AJC_DLGMAP_CMD(Main, IDCANCEL        )
503 : AJC_DLGMAP_END
504 :
505 : //--------------------------------------------------------------------------------------------------------------//
506 : //  ウインドスタイル値をチェックボックスに設定                                                                  //
507 : //--------------------------------------------------------------------------------------------------------------//
508 : static  VO  SetStyleToCheckBox(VO)
509 : {
510 :     UI      sty, exs;
511 :
512 :     sty = (UI)MAjcGetWindowLong(hWndVth, GWL_STYLE  );
513 :     exs = (UI)MAjcGetWindowLong(hWndVth, GWL_EXSTYLE);
514 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_LOGFILE      , (sty & AJCVTHS_LOGFILE    ) != 0);
515 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_FIXEDPITCH   , (sty & AJCVTHS_FIXEDPITCH ) != 0);
516 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_FAST         , (sty & AJCVTHS_FAST       ) != 0);
517 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_NOCLSBTN     , (sty & AJCVTHS_NOCLSBTN   ) != 0);
518 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_FIXEDFONT    , (sty & AJCVTHS_FIXEDFONT  ) != 0);
519 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_FIXEDLF      , (sty & AJCVTHS_FIXEDLF    ) != 0);
520 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_NOVSCROLL    , (sty & AJCVTHS_NOVSCROLL  ) != 0);
521 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_NOHSCROLL    , (sty & AJCVTHS_NOHSCROLL  ) != 0);
522 :
523 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_NOSCRL       , (sty & AJCVTHS_NOSCRL     ) != 0);
524 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_NOBORDER     , (sty & AJCVTHS_NOBORDER   ) != 0);
525 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_NOSCRLOUT    , (sty & AJCVTHS_NOSCRLOUT  ) != 0);
526 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_CRLFCTRL     , (sty & AJCVTHS_CRLFCTRL   ) != 0);
527 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_CRCTRL       , (sty & AJCVTHS_CRCTRL     ) != 0);
528 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_LFCTRL       , (sty & AJCVTHS_LFCTRL     ) != 0);
529 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_SEPARATE     , (sty & AJCVTHS_SEPARATE   ) != 0);
530 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_ACCEPTFILES  , (exs & WS_EX_ACCEPTFILES  ) != 0);
531 : }
532 : //--------------------------------------------------------------------------------------------------------------//
533 : //  チェックボックスの設定をスタイルに反映                                                                      //
534 : //--------------------------------------------------------------------------------------------------------------//
535 : static  VO  SetCheckBoxToStyle(VO)
536 : {
537 :     UI      sty, exs;
538 :
539 :     sty = (UI)MAjcGetWindowLong(hWndVth, GWL_STYLE);
540 :     sty &= ~(AJCVTHS_LOGFILE    | AJCVTHS_FIXEDPITCH | AJCVTHS_FAST      | AJCVTHS_NOCLSBTN | AJCVTHS_FIXEDFONT |
541 :              AJCVTHS_FIXEDLF    | AJCVTHS_NOVSCROLL  | AJCVTHS_NOHSCROLL | AJCVTHS_NOSCRL   | AJCVTHS_NOBORDER  |
542 :              AJCVTHS_NOSCRLOUT | AJCVTHS_CRLFCTRL   | AJCVTHS_CRCTRL    | AJCVTHS_LFCTRL   | AJCVTHS_SEPARATE);
543 :     exs = (UI)MAjcGetWindowLong(hWndVth, GWL_EXSTYLE);
544 :     exs &= ~WS_EX_ACCEPTFILES;
545 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_LOGFILE      )) sty |= AJCVTHS_LOGFILE;
546 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_FIXEDPITCH   )) sty |= AJCVTHS_FIXEDPITCH;
547 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_FAST         )) sty |= AJCVTHS_FAST;
548 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_NOCLSBTN     )) sty |= AJCVTHS_NOCLSBTN;
549 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_FIXEDFONT    )) sty |= AJCVTHS_FIXEDFONT;
550 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_FIXEDLF      )) sty |= AJCVTHS_FIXEDLF;
551 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_NOVSCROLL    )) sty |= AJCVTHS_NOVSCROLL;
552 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_NOHSCROLL    )) sty |= AJCVTHS_NOHSCROLL;
553 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_NOSCRL       )) sty |= AJCVTHS_NOSCRL;
554 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_NOBORDER     )) sty |= AJCVTHS_NOBORDER;
555 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_NOSCRLOUT    )) sty |= AJCVTHS_NOSCRLOUT;
556 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_CRLFCTRL     )) sty |= AJCVTHS_CRLFCTRL;
557 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_CRCTRL       )) sty |= AJCVTHS_CRCTRL;
558 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_LFCTRL       )) sty |= AJCVTHS_LFCTRL;
559 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_SEPARATE     )) sty |= AJCVTHS_SEPARATE;
```

```
560 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_ACCEPTFILES   )) exs |= WS_EX_ACCEPTFILES;
561 :     MAjcSetWindowLong(hWndVth, GWL_STYLE   , sty);
562 :     MAjcSetWindowLong(hWndVth, GWL_EXSTYLE, exs);
563 : }
564 :
565 :
```

8.　数値入力コントロール（AjcCtrlInpVal クラス）

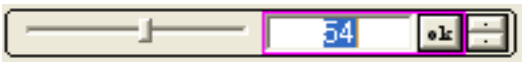
スライダやスピンボタンを使用し、１０進／１６進数値（符号付き整数(int)／実数(double)）を入力する為のコントロールです。
また、設定する数値を飛び値（ex. 0, 10, 20 ・・・ 90, 100）としたり、数値そのものを直接入力することもできます。

数値入力コントロールの外観を、以下に示します。



左右のスライダとスピンボタンで、値の増減を行います。
数値入力コントロールにフォーカスがある状態で「↑」や「↓」キーを押しても（スピンボタンと同様に）値を増減できます。
数値が変更されると、親ウインドに WM_COMMAND メッセージを送り、変更された数値を通知します。
数値の範囲や増減値は、コントロールのプロパティで設定します。

「...」ボタンを押すと、数値部分の表示が選択状態となり、数値を直接入力することができます。
１６進数を入力する場合は、先頭に"0x"を付加してください。



紫色の枠（この枠は１秒周期でブリンク表示されます）は、入力が確定していないことを示します。
この紫色の枠は、表示色を変更したり、非表示とすることもできます。
ここで数値を入力し、「ok」ボタンを押すか、空白キーを押すか、他のコントロールへフォーカスを移すと、入力した数値が確定します。
例えば、数値入力後、ダイアログの「ＯＫ」ボタン等でダイアログを終了すれば、入力も完了します。

尚、コントロールのプロパティで設定されている範囲外の数値を入力した場合は、範囲内の数値に調整されます。
例えば、数値の範囲が０～１００と設定されている場合、「１５０」を入力すると、「１００」に訂正されます。
入力途中で、数値を元に戻すには、「CTRL+Z」キーを押します。

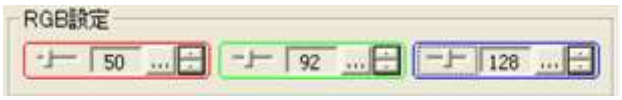
テキストボックス・クリックによる数値入力

「AJCIVS_AUTOEDIT」スタイルが設定されている場合、テキストボックスをクリックすると、(「...」ボタン押下と同様に）数値の直接入力が可能です。
この場合、ボタン非表示でも、テキストボックスをクリックすることにより数値入力ができます。

外枠表示

コントロールの外枠は、ウインド・スタイルを変更することにより「非表示」としたり、表示色を設定することができます。

外枠の表示色を設定した例



スライダ，ボタンやスピンボタンの非表示

スライダ，ボタンやスピンボタンは、ウインド・スタイルを変更することにより非表示とすることができます。



スライダを非表示



ボタンを非表示
（この場合、数値を直接入力するには数値部分をクリックし CTRL キーを押します）　※１



スライダとスピンボタンを非表示



全て非表示
（この場合、数値を直接入力するには数値部分をクリックし ENTER キーを押します）　※１

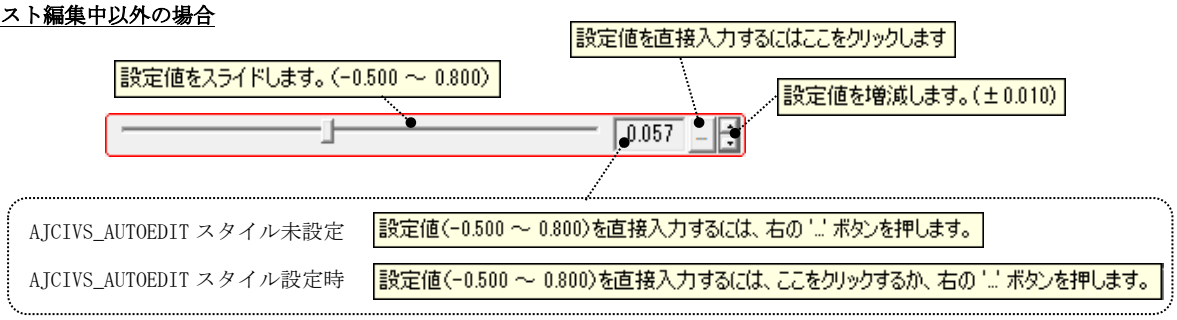
※１：「AJCIVS_AUTOEDIT」スタイルが設定されていない場合は、テキストボックスのクリックによるによる数値の直接入力はできません

8.1. ツールチップテキスト

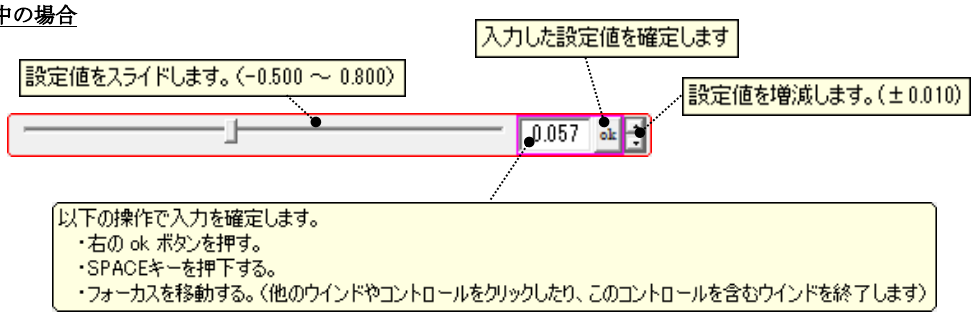
数値入力コントロールでは、デフォルトのツールチップ表示機能が用意されています。
コントロールのウインド生成時にAJCIVS_DEFTIP スタイルが設定されている場合、デフォルトツールチップが表示されます。
（AJCIVS_DEFTIP スタイルは、コントロールのウインド生成時のみ有効です。生成後はAjcIvEnaDefTipText()を実行し、デフォルトツールチップの表示／非表示を設定します）

デフォルトのツールチップは、以下のようなイメージです。
（各パーツ（スライダ，テキスト，ボタン，スピンボタン）ごとにツールチップが表示されます）

テキスト編集中以外の場合



テキスト編集中の場合



AjcIvEnaDefTipText()を実行し、デフォルトツールチップを禁止すると、デフォルトのツールチップは表示されなくなります。
（デフォルトでは、禁止状態となっています）

```
AjcIvEnaDefTipText(hWndInp, FALSE);
```
デフォルト・ツールチップの禁止

ユーザの自由なツールチップテキストを設定するには、AjcIvSetTipText()でツールチップテキストを設定します。

```
AjcIvSetTipText(WndInp, “ツールチップテキスト”);
```
※ デフォルトのツールチップを禁止した状態で実行します。

あるいは、AjcTipTextAdd()で、数値入力コントロールを指定してツールチップを設定します。(詳細は「チップテキストの表示」参照)

```
   AjcIvSetTipText (hWndInp, “ツールチップテキスト”);
[  AjcTipTextSetCallBack(hWndInp, 0, NULL, cbGetText);  ]
```
※ デフォルトのツールチップを禁止する必要はありません（自動的にデフォルトのツールチップは非表示となります）
※ AjcTipTextSetCallBack()により、コールバックを指定して、状況に依存したツールチップを表示することも可能で

8.1.1. コントロールのスタイル

数値入力コントロールのスタイルは、以下のとおりです。

名称	ビット	値	内容	備考
AJCIVS_DEFTIP	12	0x1000	デフォルトツールチップを有効にする	ウインド生成時のみ有効
AJCIVS_AUTOEDIT	10	0x0400	テキストボックス・クリックによる数値入力を許可	
AJCIVS_SHOWHEX	9	0x0200	数値を１６進数で表示する	（※１）
AJCIVS_NOBLINK	8	0x0100	数値の直接入力時にブリンク表示しない	
AJCIVS_NOBORDER	7	0x0080	コントロールの外枠を表示しない	
AJCIVS_NOSLD	6	0x0040	スライダ非表示	
AJCIVS_NOBTN	5	0x0020	ボタン非表示	
AJCIVS_NOSPN	4	0x0010	スピンボタン非表示	
AJCIVS_SEPARATE	3	0x0008	数値を特定桁数毎に区切る	（※２）
AJCIVS_REALMODE	2	0x0004	実数モード（このビットが０の場合は整数モード）	（※３）
AJCIVS_CENTER	1-0	0x0000	数値をテキストボックスの中央に表示する	
AJCIVS_LEFT		0x0001	数値をテキストボックスの左端に表示する	
AJCIVS_RIGHT		0x0002	数値をテキストボックスの右端に表示する	
AJCIVS_NOALL	6-4	0x0070	スライダ，ボタン，スピンボタンを全て非表示	テキストボックスを非表示にすることはできません
AJCIVS_SLDONLY		0x0030	スライダだけ表示	
AJCIVS_BTNONLY		0x0050	ボタンだけ表示	
AJCIVS_SPNONLY		0x0060	スピンボタンだけ表示	

※１：整数モードで使用してください。（実数モード（AJCIVS_REALMODE）の場合は、整数部分だけを１６進表示します）
※２：「AJCIVS_SEPARATE」を指定した場合は、数値の整数部を、ロケール情報に従って特定桁数毎に区切って表示します。
例えば、日本や米国では、３桁毎のカンマで区切ります。(ex. 1234567.3456 → 1,234,567.3456)
※３：「REALMODE」スタイルは、プロパティの設定より先に設定してください。

8.2. コントロールのプロパティ構造体

数値入力コントロールのプロパティは、以下の構造体で定義されます。

```
typedef struct {
    double  min;            //  レンジ最小値
    double  max;            //  レンジ最大値
    double  SldUnit;        //  値の最小単位
    double  SldPage;        //  スライダのページサイズ
    double  SpnStep;        //  スピンボタンのステップ数
    UI      TextLen;        //  テキストの桁数
    SW      Precision;      //  小数部の桁数（負数の場合は有効桁数）
    SW      HexLen;         //  １６進数表示桁数（負数の場合は無効）
} AJCIVPROP, *PAJCIVPROP;
typedef const AJCIVPROP *PCAJCIVPROP;
```

各メンバ変数の内容は、以下のとおりです。

メンバ	内容	規定値	備考
min	数値範囲の最小値（スライダのつまみが左端にある時の値）	0	min = max は不可
max	数値範囲の最大値（スライダのつまみが右端にある時の値）	100	min > max は可
SldUnit	数値の最小単位値	1	０より大きい正の数値
SldPage	スライダ・ページサイズ	10	SldUnit 以上の正の数値
SpnStep	スピンボタンによる増減値	1	SldUnit 以上の正の数値
TextLen	テキストボックス大きさ（幅）を、数値の最大桁数で指定	6	１以上
Precision	少数部の桁数（負数の場合は、数値の有効桁数）	0	実数モード時のみ有効
HexLen	１６進数表示桁数（負数の場合は、無効）	0	AJCIVS_SHOWHEX スタイル時のみ有効

・整数モードにおいて、「min」と「max」は、-2147483648～+2147483647 の範囲でなければなりません。
（範囲外である場合は、-2147483648 あるいは+2147483647 に訂正されます）
・「min」が「max」より大きい場合は、スライダを右にスライド（あるいはスピンボタン上部クリック）で値が減少し、
スライダを左にスライド（あるいはスピンボタン下部クリック）で値が増加します。
・「スライダ・ページサイズ」とは、スライダのつまみの左右のライン上をクリックした時の増減値を意味します。
・整数モードにおいて、「SldUnit」が１未満である場合は、１に訂正されます。
・「SldPage」と「SpnStep」は、「SldUnit」の整数倍に調整されます。
・本コントロールで設定される数値は、(「min」と「max」で指定された値を除いて）全て「SldUnit」の整数倍となります。
但し、スライダの両端値は「min」「max」で指定された値となります。
・Precision は printf 書式文字の桁数部分で、正の値の場合は「”%.*f”」，負数の場合は 「” %.*G”」 の「*」の部分の値となります。

8.3. キャプション文字列によるプロパティの設定

CreateWindow()/CreateWindowEx()の lpszWindowName 引数（ウインドキャプション）あるいは、ダイアログデザイン時におけるコントロールの Caption プロパティにより、数値入力コントロールのプロパティを設定することができます。

パラメタ（キャプション文字列）の形式は、以下のとおりです。([XXX]は、XXX を省略可能であることを意味します)

整数モードの設定とプロパティ等の設定

```
I: [L=fff], [R=fff], [G=fff], [S=fff], [U=nnn], [T=nnn], [H=nnn], [BC=nnn], [LC=nnn]
```

実数モードの設定とプロパティ等の設定

```
R: [L=fff], [R=fff], [G=fff], [S=fff], [U=fff], [T=nnn], [P=nnn], [BC=nnn], [LC=nnn]
```

文字列の先頭は「I:」あるいは「R:」でなければなりません。(「I:」あるいは「R:」の直後には空白を置けます)
「fff」は実数で、「nnn」は整数（１６進数の場合は先頭に'0x'を付加）で指定します。
各パラメタはカンマ（,）で区切ります。(カンマの前後には空白を置けます)
各パラメタの指定順序は任意です。

各パラメタの設定内容は、以下のとおりです。

キーワード	内　容
L	数値範囲の最小値（スライダのつまみが左端にある時の値)
R	数値範囲の最大値（スライダのつまみが右端にある時の値)
G	スライダ・ページサイズ　(数値の最小単位以上の値を指定してください)
U	数値の最小単位
S	スピンボタンによる増減値（数値の最小単位(U=fff)以上の値を指定してください)
T	テキストボックスに表示する数値の最大桁数
P	少数部の桁数（実数モード時のみ有効，負数を指定した場合は数値の有効桁数となります)
H	１６進数の表示桁数
BC	コントロール外枠の表示色
LC	ブリンク表示の表示色

表示色は、１６進数で「0xbbggrr」の形式で指定します（bb:青成分，gg:緑成分，rr:赤成分)

設定例

```
I: L=-1000, R=1000, G=100, S=1, T=8
```

整数モードで、数値範囲を-1000～+1000 とし、テキストボックスのサイズは8桁分とする。

```
R: L=0, R=0.1, G=0.01, S=0.001, U=0.001, T=6,
```

実数モードで、数値範囲を 0.0～0.100 とし、テキストボックスのサイズは6 桁分とする。
数値の最小単位は、0.001 とする。
スライダのページサイズは0.01 で、スピンボタンの増減値は0.001 とする。

```
I: BC=0x0000FF
```

整数モードとし、コントロールの外枠を赤色で表示する。

8.4. テキストの取得と設定

テキストを取得した場合、AJCIVS_SHOWHEX スタイルが設定されていない場合は、現在の値を現す１０進数表現の文字列を返します。
尚、AJCIVS_SEPARATE スタイルが設定されている場合は、整数部を特定桁数毎に区切ったテキストを返します。(ex. 123,456.78)
AJCIVS_SHOWHEX スタイルが設定されている場合は、先頭に"0x"を付加した１６進数表現の文字列を返します。

テキストを設定する場合は、当該テキストは１０進／１６進数表現のテキストでなければなりません。
１０進数の場合、設定するテキストの整数部は、特定桁数毎に区切られていてもＯＫです。(ex. "3,456,789.0012")
先頭が"0x"である場合は、１６進の整数として処理します。
テキストを設定する場合、設定しようとする数値が、プロパティで設定されている数値範囲を超えている場合は、所定の数値範囲内の値に訂正されます。(最小値か最大値に置き換えられます)

8.5. サポートＡＰＩ

数値入力・コントロールのサポートＡＰＩ一覧を以下に示します。

#	ＡＰＩ	内容	備考
1	AjcIvSetTextFormat	テキスト表示書式の設定	
2	AjcIv{Set/Get}Prop	プロパティ設定／取得	
3	AjcIv{Set/Get}BorderColor	コントロール外枠の表示色設定／取得	
4	AjcIv{Set/Get}BlinkColor	ブリンク表示色設定／取得	
5	AjcIvGetSilderHandle	スライダのウインドハンドル取得	
6	AjcIvGetTxtHandle	テキストボックスのウインドハンドル取得	
7	AjcIvGetBtnHandle	ボタンのウインドハンドル取得	
8	AjcIvGetSpnHandle	スピンボタンのウインドハンドル取得	
9	AjcIv{Set/Get}TxtLen	テキストボックスのサイズ（桁数）設定／取得	
10	AjcIvSetNtcRClk	右ボタン操作通知設定	DOWN/UP 操作の通知設定
11	AjcIv{Set/Get}Value	値の設定／取得	
12	AjcIvSetRange	数値範囲の設定	
13	AjcIvSetSldUnit	数値の最小単位値の設定	
14	AjcIvSetSldPage	スライダのページサイズ設定	
15	AjcIvSetSpnStep	スピンボタンのステップサイズ設定	
16	AjcIvSet Precision	数値の精度設定	
17	AjcIv{Set/Get}TipText	ツールチップの設定／取得	
18	AjcG3d{Set/Get}TipShowAlways	ツールチップ表示条件の設定／取得	
19	AjcIvEnaDefTipText	デフォルト ツールチップテキストの許可／禁止	
20	AjcIvGetEditState	数値編集状態の取得	
21	AjcVthLoadPermInfo AjcVthSavePermInfo	設定情報の永続化	
22	AjcIvSetEvtCallBack	イベント通知用コールバック関数の設定	

8.5.1. テキスト表示書式の設定（ AjcIvSetTextFormat ）

形　式 ： BOOL AjcIvSetTextFormat (HWND hwnd, int width, UI precision);

引　数： hwnd - 数値入力コントロールのハンドル
width - テキストボックスの大きさ（幅）を、文字の桁数で指定します。
precision - 正の数値の場合は、少数部の桁数を指定します。(書式指定「"%.*f"」 の「*」の部分の値)
負数の場合は、有効数字の桁数を指定します。 (書式指定「"%.*G"」 の「*」の部分の値)

説　明 ： テキストボックスのサイズ（幅）と、少数部の桁数を設定します。

戻り値 ： TRUE - 成功
FALSE - 失敗

8.5.2. プロパティの設定／取得取得（ AjcIv{Set/Get}Prop ）

形　式 ： BOOL AjcIvSetProp (HWND hwnd, PCAJCIVPROP pProp); --- 設定
BOOL AjcIvGetProp (HWND hwnd, PAJCIVPROP pBuf); ---- 取得

引　数： hwnd - 数値入力コントロールのハンドル
pBuf - 取得したプロパティを格納するバッファのアドレス

説　明 ： プロパティ情報を設定／取得します

戻り値 ： TRUE - 成功
FALSE - 失敗

8.5.3. コントロール外枠の表示色設定／取得（ AjcIv{Set/Get}BorderColor ）

形　式 ： BOOL AjcIvSetBorderColor (HWND hwnd, COLORREF color); --- 設定
COLORREF AjcIvGetBorderColor (HWND hwnd); ------------------- 取得

引　数： hwnd - 数値入力コントロールのハンドル

説　明 ： コントロール外枠の表示色を設定／取得します。

戻り値 ： 設定時：TRUE - 成功　　取得時：≠-1 - 成功（外枠の表示色(COLORREF 値)）
FALSE - 失敗　　=-1 – 失敗

8.5.4. 数値入力時のブリンク色設定／取得（AjcIv{Set/Get}BlinkColor ）

形　式 ： BOOL AjcIvSetBlinkColor (HWND hwnd, COLORREF color); --- 設定
COLORREF AjcIvGetBlinkColor (HWND hwnd); ------------------- 取得

引　数： hwnd - 数値入力コントロールのハンドル

説　明 ： 数値入力時のブリンク表示色を設定／取得します。

戻り値 ： 設定時：TRUE - 成功　　取得時：≠-1 - 成功（ブリンクの表示色(COLORREF 値)）
FALSE – 失敗　　=-1 - 失敗

8.5.5. スライダのウインドハンドル取得（ AjcIvGetSilderHandle ）

形　式 ： HWND AjcIvGetSilderHandle (HWND hwnd);

引　数： hwnd - 数値入力コントロールのハンドル

説　明 ： スライダのウインドハンドルを取得します

戻り値 ： ≠NULL - 成功（スライダのウインドハンドル）
=NULL - 失敗

8.5.6. テキストボックスのウインドハンドル取得（AjcIvGetTxtHandle ）

形　式 ： HWND AjcIvGetTxtHandle (HWND hwnd);

引　数： hwnd - 数値入力コントロールのハンドル

説　明 ： テキストボックスのウインドハンドルを取得します

戻り値 ： ≠NULL - 成功（テキストボックスのウインドハンドル）
＝NULL - 失敗

8.5.7. ボタンのウインドハンドル取得（ AjcIvGetBtnHandle）

形　式 ： HWND AjcIvGetBtnHandle (HWND hwnd);

引　数： hwnd - 数値入力コントロールのハンドル

説　明 ： ボタンのウインドハンドルを取得します

戻り値 ： ≠NULL - 成功（ボタンのウインドハンドル）
＝NULL - 失敗

8.5.8. スピンボタンのウインドハンドル取得（ AjcIvGetSpnHandle ）

形　式 ： HWND AjcIvGetSpnHandle (HWND hwnd);

引　数： hwnd - 数値入力コントロールのハンドル

説　明 ： スピンボタンのウインドハンドルを取得します

戻り値 ： ≠NULL - 成功（スピンボタンのウインドハンドル）
＝NULL - 失敗

8.5.9. テキストボックスのサイズ（桁数）設定／取得（ AjcIv{Set/Get}TxtLen ）

形　式 ： BOOL AjcIvSetTxtLen (HWND hwnd, UI len); --- 設定
UI AjcIvGetTxtLen (HWND hwnd); ---------- 取得

引　数： hwnd - 数値入力コントロールのハンドル

説　明 ： テキストボックスのサイズ（桁数，プロパティ項目「TextLen」）を設定／取得します。

戻り値 ： 設定時：TRUE - 成功　　取得時：≠0 - 現在設定されているテキストボックスのサイズ（桁数）
FALSE – 失敗　　＝0 - 失敗

8.5.10. 右ボタン操作通知設定（ AjcIvSetNtcRClk ）

形　式 ： BOOL AjcIvSetNtcRClk (HWND hwnd, fNtcRClk, UI msgRBtnDown, UI msgRBtnUp);

引　数： hwnd　　- 数値入力コントロールのハンドル
fNtcRClk　- TRUE ： 右ボタンの DOWN/UP を通知する，FALSE: 右ボタンの DOWN/UP を通知しない
msgRBtnDown　- 右ボタン押下時の通知メッセージコード　　（0 指定時は非通知）
msgRBtnUp　- 右ボタンを離した時の通知メッセージコード（0 指定時は非通知）

説　明 ： コントロールの右クリック操作を親ウインドへ通知するか否かを設定します。
各パラメタと、右クリック通知動作は以下のとおりです。

引数			右クリック 通知コード	備考
fNtcRClk	msgRBtnDown	msgRBtnUp		
FALSE (デフォルト)	ー	ー	AJCIVN_RCLICK	WM_COMMAND で通知
TRUE	いずれかが 0 以外		ボタン押下時：msgRBtnDown	msgRBtnDown ＝ 0 の場合は非通知
			ボタン離し時：msgRBtnUp	msgRBtnUp　＝ 0 の場合は非通知
	0	0	ー（非通知）	ー

戻り値 ： TRUE　- 成功
FALSE - 失敗

8.5.11. 値の設定／取得（ AjcIvGetValue ）

形　式 ： BOOL　AjcIvSetValue　(HWND hwnd, double value); ---------- 設定（実数）
BOOL　AjcIvSetValueEx(HWND hwnd, double value, BOOL fNtc); - 設定（通知イベント指定）

double　AjcIvGetValue　(HWND hwnd); ---------- 取得（実数）
int　AjcIvGetValueInt (HWND hwnd); ---------- 取得（整数）

引　数： hwnd　- 数値入力コントロールのハンドル
value　- 設定する数値
fNtc　- 通知イベント指定（FALSE：通知イベントを発生しない，TRUE：通知イベントを発生する）

説　明 ： 数値を設定／取得します。
設定する数値は、「SldUnit」プロパティの整数倍か、「min」「max」プロパティ値に調整されます。
取得する数値は、「SldUnit」プロパティ値を整数倍した値か、「min」「max」プロパティ値となります。
AjcIvSetValue()で値を変更した場合、整数モードの場合は AJCIVN_INTVALUE イベントを、実数モードの場合は AJCIVN_REALVALUE イベントを発生します。

fNtc=TRUE を指定した場合は、通知イベント（AJCIVN_INTVALUE / AJCIVN_REALVALUE）を発生します。

戻り値 ： 設定時：TRUE　- 成功　　取得時：設定されている数値
FALSE – 失敗

注　意 ： Ver1.6.0.3 までは、AjcIvSetValue()によりプログラムから値を設定した場合も、通知イベント（AJCIVN_INTVALUE / AJCIVN_REALVALUE）を発生していました。が、現在では、通知イベント（AJCIVN_INTVALUE / AJCIVN_REALVALUE）は、以下の場合に限り発生するように変更されています。

・スライダ、テキストボックスで数値入力、スピンボタンで値を設定した場合
・AjcIvSetValueEx()で、fNtc=TRUE を指定した場合

8.5.12. 数値範囲の設定（ AjcIvSetRange ）

形　式 ： BOOL AjcIvSetRange (HWND hwnd, double minValue, double maxValue);

引　数： hwnd　　- 数値入力コントロールのハンドル
minValue - 最小値
maxValue - 最大値

説　明 ： 数値範囲（プロパティ項目「min, max」）を設定します。
最小値 ＞ 最大値は許容されますが、最小値＝最大値は設定できません。

戻り値 ： TRUE - 成功
FALSE - 失敗

8.5.13. 数値最小単位値の設定（ AjcIvSetSldUnit ）

形　式 ： BOOL AjcIvSetSldUnit (HWND hwnd, double unit);

引　数： hwnd　　- 数値入力コントロールのハンドル
unit　　- 数値の最小単位値

説　明 ： 数値の最小単位値（プロパティ項目「SldUnit」）を設定します。
数値の最小単位値は、ゼロより大きい正の数値でなければなりません。
「SldPage」「SpnStep」プロパティは、指定された数値の最小単位値の整数倍に調整されます。

戻り値 ： TRUE - 成功
FALSE - 失敗

8.5.14. スライダ・ページサイズの設定（ AjcIvSetSldPage ）

形　式 ： BOOL AjcIvSetSldPage (HWND hwnd, double page);

引　数： hwnd　　- 数値入力コントロールのハンドル
page　　- スライダ・ページサイズ

説　明 ： スライダ・ページサイズ（つまみ左右のライン上をクリック時の増減値（プロパティ項目「SldPage」））を設定します。
この増減値は、「SldUnit」プロパティ値の整数倍に調整されます。

戻り値 ： TRUE - 成功
FALSE - 失敗

8.5.15. スピンボタンによる増減値設定（ AjcIvSetRealSpnStep ）

形　式 ： BOOL AjcIvSetSpnStep (HWND hwnd, double step);

引　数： hwnd　　- 数値入力コントロールのハンドル
step　　- スピンボタン操作時の増減値

説　明 ： スピンボタンによる増減値（プロパティ項目「SpnStep」）を設定します。
この増減値は、「SldUnit」プロパティ値の整数倍に調整されます。

戻り値 ： TRUE - 成功
FALSE - 失敗

8.5.16. 数値の精度設定（ AjcIvSetPrecision ）

形　式 ： BOOL AjcIvSetPrecision (HWND hwnd, int prec);

引　数： hwnd　- 数値入力コントロールのハンドル
prec　- 小数部の桁数（正数）、あるいは、数値の有効桁数（負数）
正の数値の場合は、少数部の桁数を指定します。(書式指定「"%.*f"」 の「*」の部分の値)
負数の場合は、有効数字の桁数を指定します。 (書式指定「"%.*G"」 の「*」の部分の値)

説　明 ： 実数モードでの、数値の小数部の桁数（プロパティ項目「Precision」）を指定します。
数値の有効桁数を指定する場合は、負数で指定します。

戻り値 ： TRUE - 成功
FALSE - 失敗

8.5.17. ツールチップの設定／取得（ AjcIvSetTipText ）

形　式 ： BOOL AjcIvSetTipText (HWND hwnd, C_UTP pTipTxt); ------- 設定
UI　AjcIvGetTipText (HWND hwnd, UTP pBuf, UI lBuf); ---- 取得

引　数： hwnd　- 数値入力コントロールのハンドル
pTipTxt - ツールチップ文字列のアドレス（NULL を指定した場合は、ツールチップを表示しません）
pBuf　- ツールチップ文字列を格納するバッファのアドレス（バッファサイズを取得する場合は NULL）
lBuf　- ツールチップ文字列を格納するバッファのバイト数／文字数

説　明 ： ツールチップ（ツールヒント文字列）を設定／取得します。

戻り値 ： 設定時：TRUE - 成功　　取得時：ツールチップ文字列のバイト数／文字数
FALSE - 失敗　　(pBuf=NULL の場合は、文字列終端('¥0')を含む)

8.5.18. ツールチップ表示条件の設定／取得(AjcIv{Set|Get}TipShowAlways)

形　式 ： BOOL AjcIvSetTipShowAlways(HWND hwnd, BOOL fShowAlways); --- 設定
BOOL AjcIvGetTipShowAlways(HWND hwnd); ---------- 取得

引　数 ： hwnd　- コントロールのウインドハンドル
fShowAlways - ツールチップ表示条件 (TRUE:非アクティブ時も表示, FALSE:非アクティブ時は非表示)

説　明 ： ツールチップの表示条件（自プログラムが非アクティブ時の表示／非表示）を設定／取得します。

戻り値 ： 設定時：TRUE - 成功　　取得時：ツールチップ表示条件
FALSE - 失敗

8.5.19. デフォルトツールチップテキストの許可／禁止（ AjcIvEnaDefTipText）

形　式 ： BOOL AjcIvEnaDefTipText (HWND hwnd, BOOL fEnable);

引　数： hwnd　- 数値入力コントロールのハンドル
fEnable - デフォルトツールチップテキストの許可(TRUE)／禁止(FALSE)

説　明 ： fEnable=TRUE を指定した場合、デフォルトのツールチップを表示するように設定します。
fEnable=FALSE とした場合は、デフォルトのツールチップを消去し、ツールチップ文字列の設定
(AjcIvSetTipText()) で設定されているツールチップを表示するように設定します。

戻り値 ： TRUE - 成功
FALSE - 失敗

8.5.20. 数値編集状態の取得（AjcIvGetEditState）

形　式 ： BOOL AjcIvGetEditState (HWND hwnd, BOOL fEnable);

引　数： hwnd　　- 数値入力コントロールのハンドル

説　明 ： テキストボックスによる数値の編集状態を返します。

戻り値 ： TRUE - 数値の編集状態である
FALSE - 数値の編集状態でない

8.5.21. 設定情報の永続化 (AjcIvLoadPermInfo / AjcIvSavePermInfo)

形　式 ： BOOL AjcIvLoadPermInfo (HWND hwnd, C_UTP pProfileSect, C_UTP pKey, BOOL fNtc); - 永続化情報の読み出し
BOOL AjcIvSavePermInfo (HWND hwnd); ---------- 永続化情報の書き込み
BOOL AjcIvSavePermInfoEx(HWND hwnd, C_UTP pProfileSect, C_UTP pKey); ---------- 〃

引　数 ： hwnd　　- コントロールのウインドハンドル
pProfileSect - 永続化情報を記録するプロファイルセクション名のアドレス
pKey　　- 永続化情報を記録するプロファイルキー名称のアドレス
fNtc　　- イベント通知フラグ（FALSE:イベント通知しない, TRUE:イベント通知する）

説　明 ： プロファイルから設定された数値の読み出し、あるいは、書き込みを行います。
fNtc=TRUE を指定した場合は、無条件に AJCIVM_NEEDNTC_INTVALUE / AJCIVM_NEEDNTC_REALVALUE イベントを発生します。
fNtc=FALSE を指定した場合は、永続情報読み出しが初回の値設定である場合や、設定値が変化した場合に、イベントを発生します。

戻り値 ： TRUE - 成功
FALSE - 失敗

8.5.22. ベント通知用コールバック関数の設定(AjcIvSetEvtCallBack)

形　式 ： BOOL AjcIvSetEvtCallBack (HWND hwnd, UX cbp, VO (CALLBACK *cbNtcEvt)(HWND hCtrl, UX wParam, UX lParam, UX cbp));

引　数 ： hwnd　　- コントロールのウインドハンドル
cbp　　- コールバックパラメタ
cbNtcEvt　- イベント通知用コールバック関数（NULL:コールバック関数を無効化）

説　明 ： イベント通知用のコールバック関数を設定します。
通常、イベントは WM_COMMAND メッセージで通知されますが、コールバック関数を設定することにより、コールバック関数の呼び出しにより通知されるようになります。

戻り値 ： TRUE - 成功
FALSE - 失敗

コールバック： コールバック関数の仕様は、以下のとおりです。

cbNtcEvt（イベント通知）

形　式 ： VO CALLBACK *cbNtcEvt*(HWND hCtrl, UX wParam, UX lParam, UX cbp);

引　数 ： hCtrl　- コントロールのウインドハンドル
wPatram - コントロール識別ＩＤ, 通知コード（WM_COMMAND による通知と同じ）
lPatram - イベントパラメタ　　（WM_COMMAND による通知と同じ）
cbp　　- コールバックパラメタ

説　明 ： 各種イベントを通知します。（詳細は「コントロールの通知メッセージ」を参照）

戻り値 ： なし

8.6. 通知情報の取得ＡＰＩ

コントロールからの通知メッセージ（WM_COMMAND）に伴うパラメタを取得するＡＰＩ群です。
「AjcSetCmdWithHdl(TRUE);」を実行した場合、各通知メッセージ（WM_COMMAND）の lParam は通知内容に伴うパラメタは通知されず、lParam＝コントロールのウインドハンドルとなります。
この場合、通知内容に伴うパラメタは以下のＡＰＩで取得します。

#	関 数 名	内容
1	AjcBarGetNtcRClk	右クリック情報の取得

8.6.1. 右クリック情報の取得（AjcIvGetNtcRClk）

形　式 ： PAJCIVRCLK AjcIvGetNtcRClk (HHWND hwnd);

引　数 ： hwnd　　- コントロールのウインドハンドル

説　明 ： 右クリック通知（AJCIVN_RCLICK）時の、右クリック情報を取得します。

戻り値 ： 右クリック情報（AJCIVRCLK）へのポインタ

8.7. 設定／取得メッセージ

数値入力コントロールへのアクセスは、以下のメッセージでも実行可能です。

#	メッセージ	内容	パラメタ		戻り値
			wParam	lParam	
1	AJCIVM_SETTEXTFORMAT	書式設定（表示桁数，小数桁数）	テキストボックスの桁数	正数：小数部桁数 負数：有効数字桁数	TRUE:成功，FALSE:失敗
2	AJCIVM_GETPROP	プロパティ取得	PAJCIVPROP	0	TRUE:成功，FALSE:失敗
3	AJCIVM_SETPROP	プロパティ設定	PAJCIVPROP	0	TRUE:成功，FALSE:失敗
4	AJCIVM_GETBORDERCOLOR	コントロール外枠の表示色取得	0	0	外枠表示色
5	AJCIVM_SETBORDERCOLOR	コントロール外枠の表示色設定	外枠の表示色	0	TRUE:成功，FALSE:失敗
6	AJCIVM_GETBLINKCOLOR	ブリンク表示色取得	0	0	ブリンク表示色
7	AJCIVM_SETBLINKCOLOR	ブリンク表示色設定	ブリンク表示色	0	TRUE:成功，FALSE:失敗
8	AJCIVM_GETSLDHANDLE	スライダのハンドル取得	0	0	スライダのハンドル
9	AJCIVM_GETTXTHANDLE	テキストボックスのハンドル取得	0	0	テキストボックスのハンドル
10	AJCIVM_GETBTNHANDLE	ボタンのハンドル取得	0	0	ボタンのハンドル
11	AJCIVM_GETSPNHANDLE	スピンボタン・のハンドル取得	0	0	スピンボタンのハンドル
12	AJCIVM_GETTXTLEN	テキストボックスの桁数取得	0	0	テキストボックスの桁数
13	AJCIVM_SETTXTLEN	テキストボックスの桁数設定	テキストボックスの桁数	0	TRUE:成功，FALSE:失敗
14	AJCIVM_SETNTCRCLK	右ボタン操作通知設定	TRUE :DOWN/UP 操作通知 FALSE: DOWN/UP 操作非通知	LO:DOWN 通知メッセージ HI: UP 通知メッセージ	TRUE:成功，FALSE:失敗
15	AJCIVM_I_GETVALUE	整数値の取得	0	0	整数値
16	AJCIVM_I_SETVALUE	整数値の設定	整数値	通知イベント発生フラグ	TRUE:成功，FALSE:失敗
17	AJCIVM_I_SETRANGE	レンジ設定	最小値	最大値	TRUE:成功，FALSE:失敗
18	AJCIVM_I_SETSLDUNIT	数値の最小単位値の設定	最小単位	0	TRUE:成功，FALSE:失敗
19	AJCIVM_I_SETSLDPAGE	スライダのページサイズ設定	スライダ・ページサイズ	0	TRUE:成功，FALSE:失敗
20	AJCIVM_I_SETSPNSTEP	スピンボタンの増減値設定	スピンボタンの増減値	0	TRUE:成功，FALSE:失敗
21	AJCIVM_R_GETVALUE	実数値の取得	実数値バッファ	0	TRUE:成功，FALSE:失敗
22	AJCIVM_R_SETVALUE	実数値の設定	実数値	通知イベント発生フラグ	TRUE:成功，FALSE:失敗
23	AJCIVM_R_SETRANGE	レンジ設定	最小値	最大値	TRUE:成功，FALSE:失敗
24	AJCIVM_R_SETSLDUNIT	数値の最小単位値の設定	最小単位	0	TRUE:成功，FALSE:失敗
25	AJCIVM_R_SETSLDPAGE	スライダのページサイズ設定	スライダ・ページサイズ	0	TRUE:成功，FALSE:失敗
26	AJCIVM_R_SETSPNSTEP	スピンボタンの増減値設定	スピンボタンの増減値	0	TRUE:成功，FALSE:失敗
27	AJCIVM_R_SETPREC	数値の精度設定	正数：小数部桁数 負数：有効数字桁数	0	TRUE:成功，FALSE:失敗
28	AJCIVM_SETTIPTEXT	ツールチップ文字列の設定	文字列のアドレス	0	TRUE:成功，FALSE:失敗
29	AJCIVM_GETTIPTEXT	ツールチップ文字列の取得	バッファアドレス	バッファバイト数／文字数	TRUE:成功，FALSE:失敗

※ グレーの塗りつぶし項目は、実数値を示す変数へのポインタ（double * 型）を意味します。

8.8. 通知メッセージ

コントロールからの通知メッセージは、WM_COMMAND メッセージにより通知されます。
WM_COMMAND メッセージの wParam には以下の情報が設定されます。

・LOWORD(wParam) - コントロールの識別ＩＤ
・HIWORD(wParam) - 通知メッセージコード

通知メッセージコードは、以下のとおりです。

#	通知メッセージコード	内容	備考
1	AJCIVN_INTVALUE	整数モードでの数値設定通知	
2	AJCIVN_REALVALUE	実数モードでの数値設定通知	
3	AJCIVN_RCLICK	右クリック通知	

8.8.1. 整数モードでの数値設定通知（ AJCIVN_INTVALUE ）

説　明 ： 整数モードで、スライダ／テキストボックス／スピンボタンにより数値が設定されたことを通知します。
または、AjcIvSetValueEx()で、fNtc=TRUE を指定された場合も発生します。
通知される数値は、「SldUnit」プロパティ値を整数倍した値か、「min」「max」プロパティ値となります。
この通知は、初回の値設定時、設定値が変化した場合や、AjcIvLoadPermInfo()で fNtc=TRUE とした場合に発生します。

パラメタ： wParam - コントロールの識別ＩＤと通知メッセージコード（AJCIVN_INTVALUE）
lParam - 変化後の数値（MFC 使用時はコントロールのウインドハンドル）

戻り値 ： なし

8.8.2. 実数モードでの数値設定通知（ AJCIVN_REALVALUE ）

説　明 ： 実数モードで、スライダ／テキストボックス／スピンボタンにより数値が設定されたことを通知します。
または、AjcIvSetValueEx()で、fNtc=TRUE を指定された場合も発生します。
通知される数値は、「SldUnit」プロパティ値を整数倍した値か、「min」「max」プロパティ値となります。
この通知は、初回の値設定時、設定値が変化した場合や、AjcIvLoadPermInfo()で fNtc=TRUE とした場合に発生します。

パラメタ： wParam - コントロールの識別ＩＤと通知メッセージコード（AJCIVN_REALVALUE）
lParam - 変化後の浮動小数点数値のアドレス（double *）（MFC 使用時はコントロールのウインドハンドル）

戻り値 ： なし

8.8.3. 右クリック通知（AJCIVN_RCLICK）

説　明 ： 右クリックしたことを通知します。
右クリック通知の可否については、AjcIvSetNtcRClk()を参照してください。
lParam で以下の情報を通知します。

```
typedef struct {
    int         x;              //  カーソルｘ座標
    int         y;              //  カーソルｙ座標
    BOOL        fShift;         //  SHIFT キー押下フラグ
    BOOL        fCtrl;          //  CTRL キー押下フラグ
} AJCIVRCLK, *PAJCIVRCLK;
```

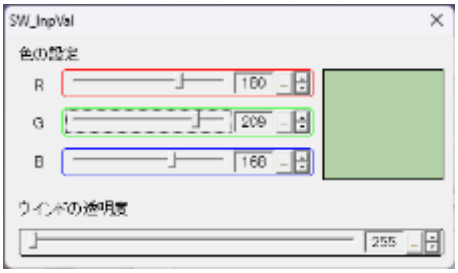
パラメタ： wParam - コントロールの識別ＩＤと通知メッセージコード（AJCIVN_RCLICK）
lParam - 右クリック情報へのポインタ（PAJCIVRCLK）（MFC 使用時はコントロールのウインドハンドル）

戻り値 ： なし（ゼロを返してください）

8.9. サンプルプログラム

8.10. SW_INPVAL（ウインドの透明度設定）

このサンプルプログラムは、数値入力コントロールで、色設定と自ウインドの透明度（255(不透明）～128(半透明)）を設定します。



※ 色の設定（R, G, B）は、0(min)～255(max)
※ ウインドの透明度は、255(min)～128(max)

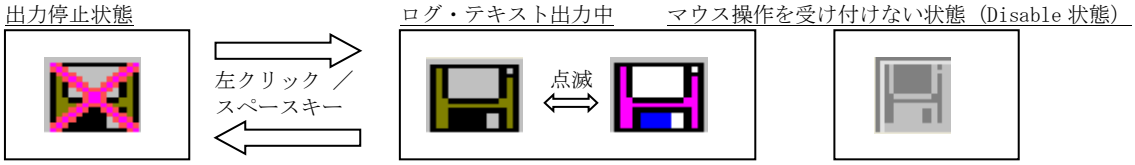
```
 1 : //
 2 : //  SW_InpVal.c
 3 : //
 4 :
 5 : #include    <AjrCstXX.h>
 6 : #include    <math.h>
 7 : #include    <tchar.h>
 8 : #include    "resource.h"
 9 :
10 : //-------------------------------------------------------------------------------------------------------------//
11 : //  ワーク                                                                                                     //
12 : //-------------------------------------------------------------------------------------------------------------//
13 : static  HINSTANCE       hInst;                      //  ＤＬＬインスタンスハンドル
14 : static  HWND            hDlgMain;                   //  ダイアログボックスハンドル
15 : static  UI              R=0, G=0, B=0;
16 : //-------------------------------------------------------------------------------------------------------------//
17 : //  内部サブ関数                                                                                               //
18 : //-------------------------------------------------------------------------------------------------------------//
19 : AJC_DLGPROC_DEF(Main);
20 : AJC_WNDPROC_DEF(Pic);
21 : //=============================================================================================================//
22 : //                                                                                                             //
23 : //  Ｗ ｉｎＭ ａ ｉ ｎ                                                                                         //
24 : //                                                                                                             //
25 : //=============================================================================================================//
26 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
27 : {
28 :     MSG     msg;
29 :
30 :     hInst = hInstance;
31 :
32 :     //----- メイン・ダイアログオープン ----------------//
33 :     hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_DLGMAIN), NULL, AJC_DLGPROC_NAME(Main));
34 :     ShowWindow(hDlgMain, SW_SHOW);
35 :
36 :     //----- メッセージループ --------------------------//
37 :     while (GetMessage(&msg, NULL, 0, 0)) {
38 :         do {
39 :             if (IsDialogMessage(hDlgMain, &msg)) break;
40 :             TranslateMessage(&msg);
41 :             DispatchMessage (&msg);
42 :         } while (0);
43 :     }
44 :
45 :     return (int)msg.wParam ;
46 : }
47 : //=============================================================================================================//
48 : //                                                                                                             //
49 : //  ダイアログ・プロシージャ                                                                                   //
50 : //                                                                                                             //
51 : //=============================================================================================================//
52 : //----- ダイアログ初期化 ----------------------------------------------------------------------------------------//
53 : AJC_DLGPROC(Main, WM_INITDIALOG     )
54 : {
55 :     hDlgMain = hDlg;
56 :     SetLayeredWindowAttributes(hDlg, 0, 255, LWA_ALPHA);
57 :     AjcSetDlgItemUInt(hDlg, IDC_INP, 255);
58 :     MAjcMmpSetSubclass(Pic, GetDlgItem(hDlg, IDC_PIC));
59 :     return TRUE;
60 : }
61 : //----- ウインド破棄 --------------------------------------------------------------------------------------------//
62 : AJC_DLGPROC(Main, WM_DESTROY        )
63 : {
64 :     //  プログラム終了
65 :     PostQuitMessage(0);
66 :     return TRUE;
67 : }
```

```
68 : //----- キャンセル ------------------------------------------------------------------------------------------------//
69 : AJC_DLGPROC(Main, IDCANCEL          )
70 : {
71 :     DestroyWindow(hDlg);
72 :     return TRUE;
73 : }
74 : //----- 赤成分通知 ------------------------------------------------------------------------------------------------//
75 : AJC_DLGPROC(Main, IDC_INP_R         )
76 : {
77 :     if (HIWORD(wParam) == AJCIVN_INTVALUE) {
78 :         R = (UI)lParam;
79 :         InvalidateRect(GetDlgItem(hDlg, IDC_PIC), NULL, TRUE);
80 :     }
81 :     return TRUE;
82 : }
83 : //----- 緑成分通知 ------------------------------------------------------------------------------------------------//
84 : AJC_DLGPROC(Main, IDC_INP_G         )
85 : {
86 :     if (HIWORD(wParam) == AJCIVN_INTVALUE) {
87 :         G = (UI)lParam;
88 :         InvalidateRect(GetDlgItem(hDlg, IDC_PIC), NULL, TRUE);
89 :     }
90 :     return TRUE;
91 : }
92 : //----- 青成分通知 ------------------------------------------------------------------------------------------------//
93 : AJC_DLGPROC(Main, IDC_INP_B         )
94 : {
95 :     if (HIWORD(wParam) == AJCIVN_INTVALUE) {
96 :         B = (UI)lParam;
97 :         InvalidateRect(GetDlgItem(hDlg, IDC_PIC), NULL, TRUE);
98 :     }
99 :     return TRUE;
100 : }
101 : //----- 透明度設定値通知 ------------------------------------------------------------------------------------------//
102 : AJC_DLGPROC(Main, IDC_INP           )
103 : {
104 :     if (HIWORD(wParam) == AJCIVN_INTVALUE) {
105 :         SetLayeredWindowAttributes(hDlg, 0, AjcGetDlgItemUInt(hDlg, IDC_INP), LWA_ALPHA);
106 :     }
107 :     return TRUE;
108 : }
109 : //-----------------------------------------------------------------------------------------------------------------//
110 : AJC_DLGMAP_DEF(Main)
111 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG     )
112 :     AJC_DLGMAP_MSG(Main, WM_DESTROY        )
113 :
114 :     AJC_DLGMAP_CMD(Main, IDCANCEL          )
115 :     AJC_DLGMAP_CMD(Main, IDC_INP_R         )
116 :     AJC_DLGMAP_CMD(Main, IDC_INP_G         )
117 :     AJC_DLGMAP_CMD(Main, IDC_INP_B         )
118 :     AJC_DLGMAP_CMD(Main, IDC_INP           )
119 : AJC_DLGMAP_END
120 : //-----------------------------------------------------------------------------------------------------------------//
121 : //  ピクチャのサブクラスプロシージャ                                                                              //
122 : //-----------------------------------------------------------------------------------------------------------------//
123 : AJC_WNDPROC(Pic, WM_PAINT       )
124 : {
125 :     PAINTSTRUCT ps;
126 :     HDC         hdc;
127 :     HBRUSH      hBrush;
128 :     RECT        r;
129 :     int         cx, cy;
130 :
131 :     hdc    = BeginPaint(hwnd, &ps);
132 :     hBrush = (HBRUSH)SelectObject(hdc, CreateSolidBrush(RGB(R, G, B)));
133 :     GetWindowRect(hwnd, &r);
134 :     cx = r.right - r.left;
135 :     cy = r.bottom - r.top;
136 :     Rectangle(hdc,  0, 0, cx, cy);
137 :     DeleteObject(SelectObject(hdc, hBrush));
138 :     EndPaint(hwnd, &ps);
139 :
140 :     return 0;
141 : }
142 : //-----------------------------------------------------------------------------------------------------------------//
143 : AJC_WNDMAP_DEF(Pic)
144 :     AJC_WNDMAP_MSG(Pic, WM_PAINT)
145 : AJC_WNDMAP_END
146 :
```

9. ログファイル出力コントロール（AjcCtrlLogFile クラス）

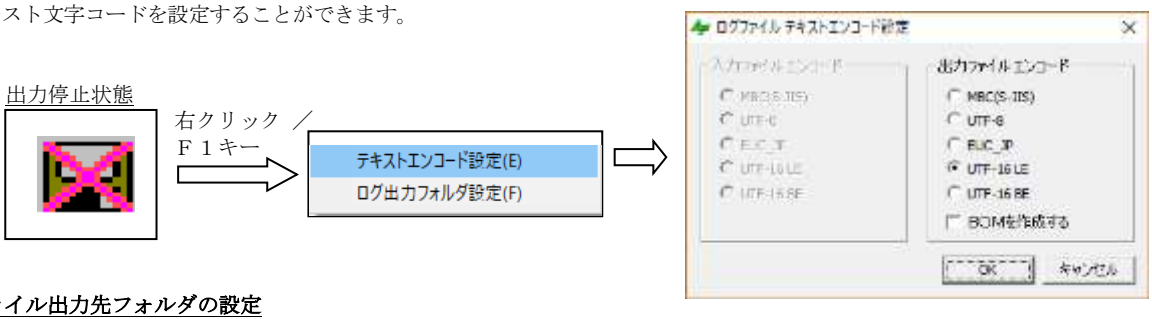
ログファイルへのテキスト出力を行うコントロールです。
ログファイルの出力先フォルダや、テキスト出力の開始／停止を行うことができます。

ログファイル出力コントロールの外観を、以下に示します。



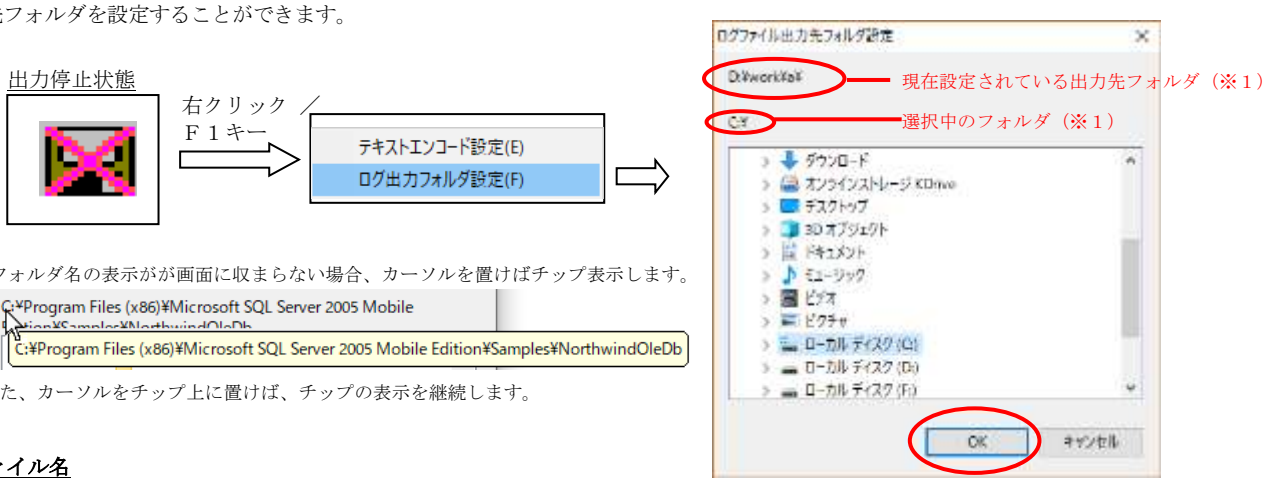
ログファイルのテキスト文字コード設定

コントロールを右クリックし、ポップアップメニューの「テキストエンコード設定」により、以下のダイアログが表示され、ログファイルのテキスト文字コードを設定することができます。



ログファイル出力先フォルダの設定

コントロールを右クリックし、ポップアップメニューの「ログ出力フォルダ設定」により、以下のダイアログが表示され、ログファイルの出力先フォルダを設定することができます。



※1：フォルダ名の表示がが画面に収まらない場合、カーソルを置けばチップ表示します。

また、カーソルをチップ上に置けば、チップの表示を継続します。

ログファイル名

ログファイルの名称は、現在時刻から「LGF_yyyy-mm-dd_hh-mm-ss.log」といったファイル名が自動生成されます。
接頭語の「LGF_」はデフォルトであり、AjcLgfSetLogFileNamePrefix()関数で変更できます。

ログファイルへのテキスト出力

ログファイルへテキストを出力するには、本コントロールが「ログ・テキスト出力中」の状態で、AjcLgfPutText()関数を実行します。
本コントロールが「ログ・テキスト出力中」の状態でない場合は、AjcLgfPutText()関数を実行してもテキストは出力されません。

```
AjcLgfPutText(hwnd, "[id]", TRUE, "This is output text¥n");
```

第１引数「hwnd」は、コントロールのウインドハンドルを指定します。
第２引数「"[id]"」は、テキストの先頭に付加する文字列を指定します。（必要ない場合は、空文字列（""）か NULL を指定）
第３引数「TRUE」は、タイムスタンプを付加することを指定します。（タイムスタンプを付加しない場合は「FALSE」を指定）
第４引数「"This is output text¥n"」は、ログテキスト本体を指定します。

上記例では、以下のようなテキストがログファイルに出力されます。（「¥n」は、「¥r¥n」に変換されて出力されます）

```
"[id] 19-31-23.548 This is output text¥n "
```

書式テキスト出力

書式テキストを出力するには、本コントロールが「ログ・テキスト出力中」の状態で、AjcLgfPrintF()関数を実行します。
本コントロールが「ログ・テキスト出力中」の状態でない場合は、AjcLgfPrintF()関数を実行してもテキストは出力されません。

```
AjcLgfPrintF(hwnd, "[id]", TRUE, "x=%d, y=%d, z=%d ¥n", x, y, z);
```

第１引数「hwnd」は、コントロールのウインドハンドルを指定します。
第２引数「"[id]"」は、テキストの先頭に付加する文字列を指定します。（必要ない場合は、空文字列（""）かNULL を指定）
第３引数「TRUE」は、タイムスタンプを付加することを指定します。（タイムスタンプを付加しない場合は「FALSE」を指定）
第４引数「"x=%d, y=%d, z=%d ¥n"」は、ログテキスト本体の書式を指定します。（書式の形式は「printf()」と同じ）
第５引数以降「x, y, z」は、書式に対応したパラメタを指定します。

上記例では、以下のようなテキストがログファイルに出力されます。（「¥n」は、「¥r¥n」に変換されて出力されます）

```
"[id] 19-31-23.548 x=123, y=456, z=789¥n "
```

タイムスタンプ形式の設定

前述の例では、タイムスタンプを「hh:mm:ss.nnn」のように、時分秒とミリ秒で出力していましたが、AjcLgfSetTimeStampFormat()関数により、タイムスタンプの形式を変更することができます。

```
AjcLgfSetTimeStampFormat(hwnd, <タイムスタンプ形式>);
```

第１引数「hwnd」は、コントロールのウインドハンドルを指定します。
第２引数は、タイムスタンプの形式です。（詳細は、AjcLgfSetTimeStampFormat()関数の説明を参照）

9.1. コントロールのスタイル

ログファイル出力コントロールのスタイルは、以下のとおりです。

名称	ビット	値	内容	備考
AJCLGF_DISCLICK	7	0x0080	マウス左クリック操作の禁止／許可	0：許可, 1：禁止
AJCLGF_NOBLINK	6	0x0040	ログファイル出力中のブリンク禁止	0：許可, 1：禁止

9.2. プロパティの永続化

設定したプロパティを、プロファイル（.ini ファイル／レジストリ）に記録し、次回起動時に記録されているプロファイルを読み出すことにより、プロパティを永続的に有効とすることができます。
プロパティとして永続化する項目は、「テキスト文字コード種別」と「ログファイルの出力先フォルダ」です。
「テキスト文字コード種別」のデフォルトは、S-JIS です。
「ログファイルの出力先フォルダ」のデフォルトは、規定のテンポラリフォルダです。

プロパティをプロファイルから読み出すには、ダイアログやウインドの初期化時に、AjcLgfLoadProp()を実行します。

```
AjcLgfLoadProp(hwnd, "SectName");
```

現在設定されているプロパティをプロファイルに記録するには、ダイアログやウインドの終了時に、AjcLgfSaveProp()を実行します。

```
AjcLgfSaveProp(hwnd, "SectName");
```

デフォルトでは、プロファイルの記録先は、レジストリになります。
プロファイルの記録先を初期化ファイル（自プログラムパス名の拡張子を「.ini」としたファイル）とする場合は、プログラムの最初（AjcLgfLoadProp(), AjcLgfSaveProp()を実行する前）に「AjcSetProfileIsRegistory(TRUE);」を実行してください。

プロファイルへのアクセスは、AjcGetProfile･･･() と AjcPutProfile･･･() により行います。
これらの関数仕様やプロファイルアクセスに関するその他の情報は、「プロファイル・アクセス」章を参照してください。

9.3. サポートＡＰＩ

ログファイル出力コントロールのサポートＡＰＩ一覧を以下に示します。

#	関数名	内容	備考
1	AjcLgfSetDir AjcLgfGetDir	ログファイル出力先ディレクトリ・パスの設定／取得	
2	AjcLgfSetTextEncode AjcLgfGetTextEncode	テキスト文字コード設定／取得	
3	AjcLgfStart	ログ出力開始	
4	AjcLgfStop	ログ出力停止	
5	AjcLgfIsActive	ログ出力可能状態取得	
6	AjcLgfPutText	ログファイルへテキスト出力	
7	AjcLgfPrintF	ログファイルへ書式テキスト出力	
8	AjcLgfSetTimeStampFormat AjcLgfGetTimeStampFormat	タイムスタンプ形式の設定／取得	
9	AjcLgfSetLogFileNamePrefix AjcLgfGetLogFileNamePrefix	ログファイル名・接頭語の設定／取得	
10	AjcLgfSetLogFileExtention AjcLgfGetLogFileExtention	ログファイル名・拡張子の設定／取得	
11	AjcLgfLoadProp	プロファイルからプロパティ値読み出し	
12	AjcLgfSaveProp	プロファイルへプロパティ値を記録	
13	AjcLgfSetEvtCallBack	イベント通知用コールバック関数の設定	

9.3.1. ログファイル出力先ディレクトリパスの設定／取得(AjcLgf{Set/Get}Dir)

形　式 ： VO AjcLgfSetDir(HWND hwnd, C_UTP pDirPath)； ---- ログファイル出力先ディレクトリパスの設定
VO AjcLgfGetDir(HWND hwnd, UTP pBuf, UI lBuf)； - ログファイル出力先ディレクトリパスの取得

引　数 ： hwnd　　- コントロールのウインドハンドル
pDirPath　- ログファイル出力先ディレクトリ・パスのアドレス
pBuf　　- ディレクトリ・パス名を格納するバッファのアドレス
lBuf　　- ディレクトリ・パス名を格納するバッファの文字数

説　明 ： ログファイルを出力するディレクトリを設定／取得します。

戻り値 ： なし

9.3.2. テキスト文字コード設定／取得(AjcLgf{Set/Get}TextEncode)

形　式 ： BOOL　AjcLgfSetTextEncode(HWND hwnd, EAJCTEC iTec, EAJCTEC oTec, BOOL fBOM)； ------- テキスト文字コード設定
BOOL　AjcLgfGetTextEncode(HWND hwnd, PEAJCTEC piTec, PEAJCTEC poTec, BOOL *pfBOM)； - テキスト文字コード取得

引　数 ： hwnd　　- コントロールのウインドハンドル
iTec　　- 入力テキスト文字コード（－１指定時は未設定，但し、現状は未使用）
oTec　　- 出力テキスト文字コード（－１指定時は未設定）
fBOM　　- BOM 出力フラグ　　（－１指定時は未設定）
piTec　　- 入力テキスト文字コードを格納するバッファのアドレス（不要時は NULL）
poTec　　- 出力テキスト文字コードを格納するバッファのアドレス（不要時は NULL）
pfBOM　　- BOM 出力フラグを格納するバッファのアドレス　　（不要時は NULL）

説　明 ： 本コントロールでテキストファイルをアクセスする際のテキスト文字コードを設定／取得します。
iTec，oTec には以下の値を指定します。
・AJCTEC_MBC　　- Ｓ－ＪＩＳ・・デフォルト値
・AJCTEC_UTF_8　- ＵＴＦ－８
・AJCTEC_EUC_J　- 日本語ＥＵＣ
・AJCTEC_UTF_16LE - ＵＴＦ－１６（リトルエンディアン）
・AJCTEC_UTF_16BE - ＵＴＦ－１６（ビッグエンディアン）

戻り値 ： TRUE　- 成功
FALSE - 失敗

9.3.3. ログ出力開始(AjcLgfStart)

形　式 ： VO AjcLgfStart(HWND hwnd)；

引　数 ： hwnd　　- コントロールのウインドハンドル

説　明 ： ログファイルを作成（オープン）し、ログテキストの出力が可能な状態にします。
但し、コントロールがディスエーブル状態の場合は、何もしません。

戻り値 ： なし

9.3.4. ログ出力停止(AjcLgfStop)

形　式 ： VO AjcLgfStop(HWND hwnd)；

引　数 ： hwnd　　- コントロールのウインドハンドル

説　明 ： ログファイルをクローズし、ログテキストの出力不能な状態にします。

戻り値 ： なし

9.3.5. ログ出力可能状態取得(AjcLgfIsActive)

形　式 ： BOOL AjcLgfIsActive(HWND hwnd)；

引　数 ： hwnd　　- コントロールのウインドハンドル

説　明 ： ログファイルへの出力が可能な常態か否かを取得します。

戻り値 ： TRUE　- ログファイルがオープンされており、ログ出力が可能な状態である
FALSE - ログファイルがクローズされており、ログ出力不能な状態である

9.3.6. ログファイルへのテキスト出力(AjcLgfPutText)

形　式 ： VO AjcLgfPutText(HWND hwnd, C_UTP pCmd, BOOL fTimeStamp, C_UTP pTxt)；

引　数 ： hwnd　　- コントロールのウインドハンドル
pCmd　　- 出力テキストの先頭に付加する文字列（不要時は NULL）
fTimeStamp - タイムスタンプ付加フラグ（FALSE:付加しない, TRUE:付加する）
pTxt　　- ログファイルへの出力テキスト本体

説　明 ： pTxt で指定したテキストをログファイルへ出力します。
pCmd は、出力するテキストの先頭に付加する文字列を指定します。
pCmd で指定した文字列の末尾には１ヶの空白が付加されます。（但し、pCmd=NULL とした場合は空白を付加しない）
fTimeStamp は、タイムスタンプを付加する(TRUE)か、否(FALSE)かを指定します。

（例）

AjcLgfPutText(hwnd, "[id]", TRUE, "This is output text¥n");	"[id] 19-31-23.548 This is output text¥n "
AjcLgfPutText(hwnd, NULL, TRUE, "This is output text¥n");	"19-31-23.548 This is output text¥n "
AjcLgfPutText(hwnd, NULL, FALSE, "This is output text¥n");	"This is output text¥n "

戻り値 ： なし

9.3.7. ログファイルへの書式テキスト出力(AjcLgfPrintF)

形　式 ： VO AjcLgfPrintF(HWND hwnd, C_UTP pCmd, BOOL fTimeStamp, C_UTP pFmt, ...)；

引　数 ： hwnd　　- コントロールのウインドハンドル
pCmd　　- 出力テキストの先頭に付加する文字列（不要時は NULL）
fTimeStamp - タイムスタンプ付加フラグ（FALSE:付加しない, TRUE:付加する）
pFmt　　- ログファイルへの出力する書式テキスト

説　明 ： pFmt で書式化されたテキストをログファイルへ出力します。
pCmd は、出力するテキストの先頭に付加する文字列を指定します。
pCmd で指定した文字列の末尾には１ヶの空白が付加されます。（但し、pCmd=NULL とした場合は空白を付加しない）
fTimeStamp は、タイムスタンプを付加する(TRUE)か、否(FALSE)かを指定します。

（例）

AjcLgfPrintF(hwnd, "[id]", TRUE, "x=%d, y=%d, z=%d ¥n", x, y, z);	"[id] 19-31-23.548 x=123, y=456, z=789¥n "
AjcLgfPrintF(hwnd, NULL, TRUE, "x=%d, y=%d, z=%d ¥n", x, y, z);	"19-31-23.548 x=123, y=456, z=789¥n "
AjcLgfPrintF(hwnd, NULL, FALSE, "x=%d, y=%d, z=%d ¥n", x, y, z);	"x=123, y=456, z=789¥n "

9.3.8. タイムスタンプ形式の設定／取得(AjcLgf{Set/Get}TimeStampFormat)

形　式 ： BOOL AjcLgfSetTimeStampFormat(HWND hwnd, AJCLGFTS TimeStampFormat)； - タイムスタンプ形式の設定
AJCLGFTS AjcLgfGetTimeStampFormat(HWND hwnd)； ---------- タイムスタンプ形式の取得

引　数 ： hwnd　　- コントロールのウインドハンドル
TimeStampFormat - タイムスタンプの形式

説　明 ： タイムスタンプの形式を指定します。
「TimeStampFormat」は、以下のいずれかを指定します。（デフォルトは「AJCLGFTS_HMSN」）

#	〈タイムスタンプ形式〉	出力形式	例	備考
1	AJCLGFTS_NO	タイムスタンプなし	""	空文字列
2	AJCLGFTS_HMSN	時分秒とミリ秒（hh:mm:ss.nnn）	"19-31-23.548 "	末尾に空白を付加し、１３桁
3	AJCLGFTS_HMS	時分秒（hh:mm:ss）	"19-31-23 "	〃　　９桁
4	AJCLGFTS_TICK	Windows 開始時からの経過時間[ms]	" 3,456,789 "	〃　　１１桁
5	AJCLGFTS_MS	ログファイル出力コントロール開始時からの経過時間[ms]	"　　2,995 "	〃　　１１桁
6	AJCLGFTS_US	ログファイル出力コントロール開始時からの経過時間[us]	"　　2,995,726 "	〃　　１５桁

戻り値 ： 設定時：TRUE　- 成功　　取得時：≠AJCLGFTS_ERR ： 現在設定されているタイムスタンプの形式（AJCLGFTS_*XXXXX*）
FALSE - 失敗　　＝AJCLGFTS_ERR ： エラー

9.3.9. ログファイル名の接頭語設定／取得 (AjcLgf{Set/Get}LogFileNamePrefix)

形　式 ： BOOL　AjcLgfSetLogFileNamePrefix(HWND hwnd, C_UTP pPrefix)； ------ ログファイル名の接頭語設定
BOOL　AjcLgfGetLogFileNamePrefix(HWND hwnd, UTP pBuf, UI lBuf)； -- ログファイル名の接頭語取得

引　数 ： hwnd　　- コントロールのウインドハンドル
pPrefix - 設定するログファイル名の接頭語（文字列）のアドレス

説　明 ： ログファイルの先頭に付加する接頭語を設定／取得します。
接頭語は、（半角換算で）１２７文字以内の文字列を指定してください。
接頭語のデフォルトは「LGF_」です。

戻り値 ： TRUE　- 成功
FALSE - 失敗

9.3.10. ログファイル名の拡張子設定／取得 (AjcLgf{Set/Get}LogFileExtention)

形　式 ： BOOL AjcLgfSetLogFileExtention(HWND hwnd, C_UTP pExt);
AjcLgfGetLogFileExtention(HWND hwnd, UTP pBuf, UI lBuf);

引　数 ： hwnd　- コントロールのウインドハンドル
pExt　- 設定するログファイル名の拡張子（文字列）のアドレス
pBuf　- ログファイル名の拡張子（文字列）を格納するバッファのアドレス
lBuf　- ログファイル名の拡張子（文字列）を格納するバッファの文字数

説　明 ： ログファイルの末尾に付加する拡張子を設定／取得します。
拡張子は、(半角換算で) 255(_MAX_EXT-1)文字以内の文字列を指定してください。
拡張子のデフォルトは「.log 」です。
拡張子の先頭がピリオド('.')でない場合は、拡張子の先頭にピリオド('.')を付加します。
pExt に NULL や空文字列("")を指定した場合は、拡張子無しとなります。

戻り値 ： TRUE　- 成功
FALSE - 失敗

9.3.11. プロファイルからプロパティ値読み出し (AjcLgfLoadProp)

形　式 ： BOOL AjcLgfLoadProp(HWND hwnd, C_UTP pProfileSect);

引　数 ： hwnd　　- コントロールのウインドハンドル
pProfileSect - プロファイル・セクション名（文字列）へのポインタ

説　明 ： プロファイル（.ini ファイル／レジストリ）からプロパティ値（出力フォルダパス）を読み出して設定します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

9.3.12. プロファイルへプロパティ値を記録 (AjcLgfSaveProp)

形　式 ： BOOL AjcLgfSaveProp(HWND hwnd, C_UTP pProfileSect);

引　数 ： hwnd　　- コントロールのウインドハンドル
pProfileSect - プロファイル・セクション名（文字列）へのポインタ

説　明 ： 現在設定されているプロパティ値（出力フォルダパス）を、プロファイル（.ini ファイル／レジストリ）へ記録します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

9.3.13. ベント通知用コールバック関数の設定(AjcLgfSetEvtCallBack)

形　式 ： BOOL AjcLgfSetEvtCallBack (HWND hwnd, UX cbp, VO (CALLBACK *cbNtcEvt)(HWND hCtrl, UX wParam, UX lParam, UX cbp));

引　数 ： hwnd　　- コントロールのウインドハンドル
cbp　　- コールバックパラメタ
cbNtcEvt　- イベント通知用コールバック関数（NULL:コールバック関数を無効化）

説　明 ： イベント通知用のコールバック関数を設定します。
通常、イベントは WM_COMMAND メッセージで通知されますが、コールバック関数を設定することにより、コールバック関数の呼び出しにより通知されるようになります。

戻り値 ： TRUE　- 成功
FALSE - 失敗

コールバック： コールバック関数の仕様は、以下のとおりです。

cbNtcEvt（イベント通知）

形　式 ： VO CALLBACK *cbNtcEvt*(HWND hCtrl, UX wParam, UX lParam, UX cbp)；

引　数 ： hCtrl　- コントロールのウインドハンドル
wPatram - コントロール識別ＩＤ，通知コード（WM_COMMAND による通知と同じ）
lPatram - イベントパラメタ　　（WM_COMMAND による通知と同じ）
cbp　　- コールバックパラメタ

説　明 ： 各種イベントを通知します。（詳細は「コントロールの通知メッセージ」を参照）

戻り値 ： なし

9.4. コントロールの通知メッセージ

コントロールからの通知メッセージは、WM_COMMAND メッセージにより通知されます。
WM_COMMAND メッセージの wParam には以下の情報が設定されます。

・LOWORD(wParam) - コントロールの識別ＩＤ
・HIWORD(wParam) - 通知メッセージ

9.4.1. ログ出力開始通知 (AJCLGFN_START)

説　明　：　ログファイルの出力が開始されたことを通知します。
lParam には、出力ログファイルのパス名へのポインタが設定されます。

パラメタ：　wParam - コントロール識別ＩＤと、通知メッセージコード（AJCLGFN_START）
lParam – 出力ファイルパス名のアドレス（MFC 使用時はコントロールのウインドハンドル）

戻り値　：　なし

9.4.2. ログ出力停止通知 (AJCLGFN_STOP)

説　明　：　ログファイルの出力が停止したことを通知します。

パラメタ：　wParam - コントロール識別ＩＤと、通知メッセージコード（AJCLGFN_STOP）
lParam – コントロールのウインドハンドル

戻り値　：　なし

9.4.3. テキスト文字コード設定変更通知 (AJCLGFN_ENCODE)

説　明　：　テキスト文字コードが変更されたことを通知します。

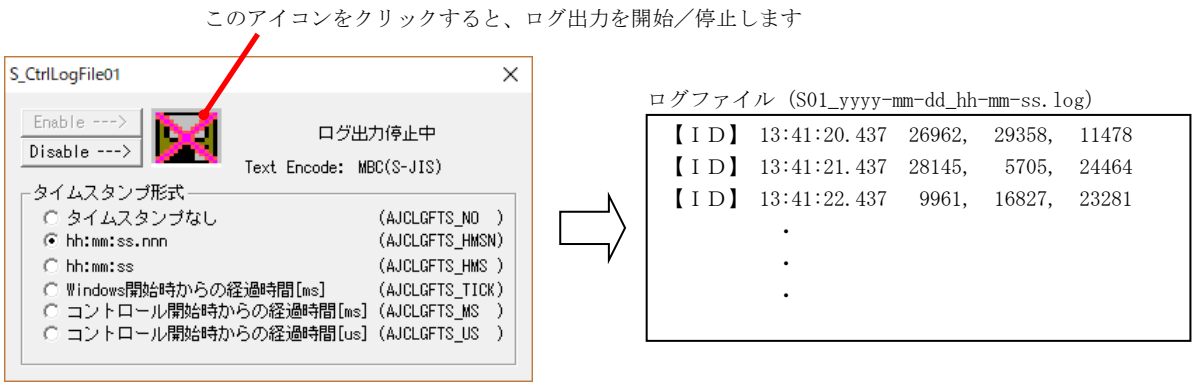
パラメタ：　wParam - コントロール識別ＩＤと、通知メッセージコード（AJCLGFN_STOP）
lParam - LOWORD: 出力文字コード, HIWORD: BOM 出力フラグ（MFC 使用時はコントロールのウインドハンドル）

戻り値　：　なし

9.5. サンプルプログラム

9.5.1. SW_LogFile（ログファイル出力）

このサンプルプログラムでは、ランダムな３つの整数を、ファイルへログ出力します。




```
 1 : //
 2 : //  SW_LogFile.c
 3 : //
 4 : #include    <AjrCstXX.h>
 5 : #include    "resource.h"
 6 :
 7 : //-------------------------------------------------------------------------------------------------------------//
 8 : //  ワーク                                                                                                     //
 9 : //-------------------------------------------------------------------------------------------------------------//
10 : static  HINSTANCE       hInst;
11 : static  HWND            hDlgMain;
12 : static  HWND            hWndLgf;
13 : static  HWND            hWndGrp;
14 :
15 : //-------------------------------------------------------------------------------------------------------------//
16 : //  内部サブ関数                                                                                               //
17 : //-------------------------------------------------------------------------------------------------------------//
18 : AJC_DLGPROC_DEF(Main);
19 : static  VO  ShowEncode(HWND hDlg, EAJCTEC OutTec, BOOL fBOM);
20 :
21 : //=============================================================================================================//
22 : //                                                                                                             //
23 : //  W i n M a i n                                                                                              //
24 : //                                                                                                             //
25 : //=============================================================================================================//
26 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
27 : {
28 :     MSG     msg;
29 :
30 :     hInst = hInstance;
31 :
32 :     //----- メイン・ダイアログオープン ----------------//
33 :     hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_DLGMAIN), NULL, AJC_DLGPROC_NAME(Main));
34 :     ShowWindow(hDlgMain, SW_SHOW);
35 :
36 :     //----- メッセージループ --------------------------//
37 :     while (GetMessage(&msg, NULL, 0, 0)) {
38 :         do {
39 :             if (IsDialogMessage(hDlgMain, &msg)) break;
40 :             TranslateMessage(&msg);
41 :             DispatchMessage (&msg);
42 :         } while (0);
43 :     }
44 :     return (int)msg.wParam ;
45 : }
46 : //=============================================================================================================//
47 : //                                                                                                             //
48 : //  ダイアログ・プロシージャ                                                                                   //
49 : //                                                                                                             //
50 : //=============================================================================================================//
51 : //----- ダイアログ初期化 ----------------------------------------------------------------------------------//
52 : AJC_DLGPROC(Main, WM_INITDIALOG )
53 : {
```

```
    EAJCTEC OutTec;
    BOOL    fBOM;

    hWndLgf = GetDlgItem(hDlg, IDC_LGF);
    hWndGrp = GetDlgItem(hDlg, IDC_GRP_TSFORM);

    AjcLgfSetLogFileNamePrefix(hWndLgf, TEXT("S01_"));
    AjcLgfSetTimeStampFormat(hWndLgf, AJCLGFTS_HMSN);
    AjcLgfLoadProp(hWndLgf, TEXT("LogFileSect"));

    AjcSbcRadioBtns(hWndGrp);
    AjcSbcSetRbt    (hWndGrp, AjcLgfGetTimeStampFormat(hWndLgf));
    SetTimer(hDlg, 1, 1000, NULL);

    AjcLgfGetTextEncode(hWndLgf, NULL, &OutTec, &fBOM);
    ShowEncode(hDlg, OutTec, fBOM);

    return TRUE;
}
//----- ウインド破棄 ----------------------------------------------------------------------------------------//
AJC_DLGPROC(Main, WM_DESTROY    )
{
    KillTimer(hDlg, 1);
    AjcLgfSaveProp(hWndLgf, TEXT("LogFileSect"));
    PostQuitMessage(0);
    return TRUE;
}
//----- タイマ ----------------------------------------------------------------------------------------------//
AJC_DLGPROC(Main, WM_TIMER      )
{
    AjcLgfPrintF(hWndLgf, TEXT("【ＩＤ】"), TRUE, TEXT("%6d, %6d, %6d¥n"), rand(), rand(), rand());
    return TRUE;
}
//----- キャンセルボタン ------------------------------------------------------------------------------------//
AJC_DLGPROC(Main, IDCANCEL      )
{
    DestroyWindow(hDlg);
    return TRUE;
}
//----- Enable ボタン ---------------------------------------------------------------------------------------//
AJC_DLGPROC(Main, IDC_CMD_ENABLE)
{
    EnableWindow(hWndLgf, TRUE);
    EnableWindow(GetDlgItem(hDlg, IDC_CMD_ENABLE ), FALSE);
    EnableWindow(GetDlgItem(hDlg, IDC_CMD_DISABLE), TRUE );
    SetFocus(GetDlgItem(hDlg, IDC_CMD_DISABLE));
    return TRUE;
}
//----- Disable ボタン --------------------------------------------------------------------------------------//
AJC_DLGPROC(Main, IDC_CMD_DISABLE)
{
    EnableWindow(hWndLgf, FALSE);
    EnableWindow(GetDlgItem(hDlg, IDC_CMD_ENABLE ), TRUE );
    EnableWindow(GetDlgItem(hDlg, IDC_CMD_DISABLE), FALSE);
    SetFocus(GetDlgItem(hDlg, IDC_CMD_ENABLE ));
    return TRUE;
}
//----- タイムスタンプ形式ラジオボタン ----------------------------------------------------------------------//
AJC_DLGPROC(Main, IDC_GRP_TSFORM)
{
    AjcLgfSetTimeStampFormat(hWndLgf, (AJCLGFTS)AjcSbcGetRbt(hWndGrp));
    return TRUE;
}
//----- ログファイル出力コントロールからの通知 --------------------------------------------------------------//
AJC_DLGPROC(Main, IDC_LGF       )
{
    switch (HIWORD(wParam)) {
        case AJCLGFN_START:
            AjcSetDlgItemStr(hDlg, IDC_LBL_MSG, TEXT("ログ出力中"));
            break;

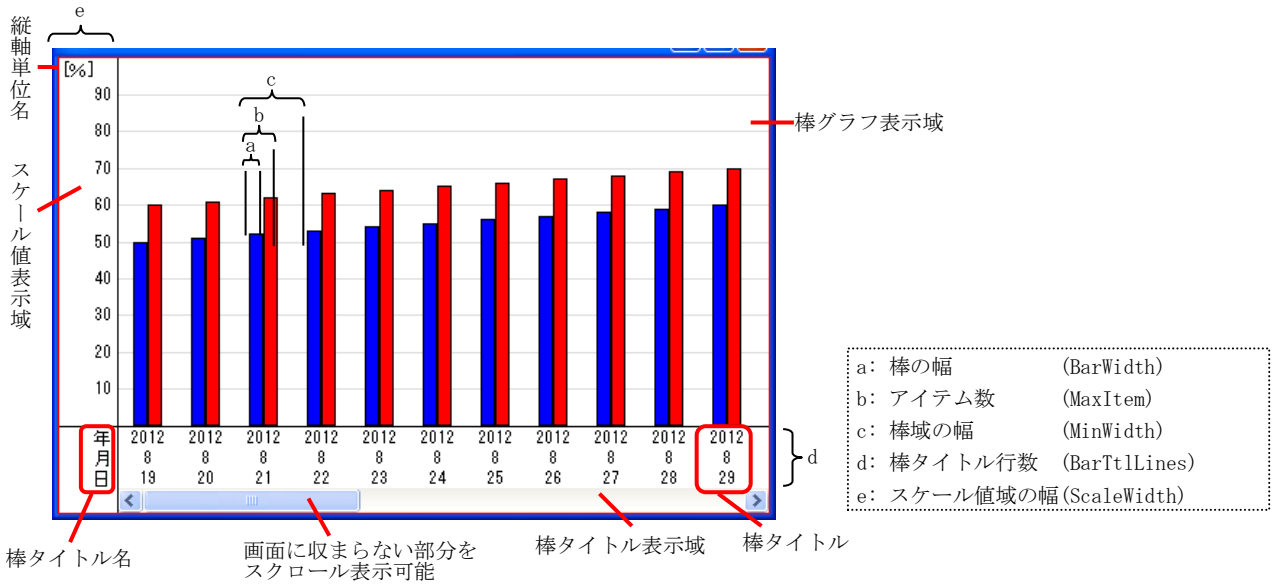
        case AJCLGFN_STOP:
            AjcSetDlgItemStr(hDlg, IDC_LBL_MSG, TEXT("ログ出力停止中"));
            break;

        case AJCLGFN_ENCODE:
            ShowEncode(hDlg, (EAJCTEC)LOWORD(lParam), (BOOL)HIWORD(lParam));
            break;
    }
    return TRUE;
```

```
134 : }
135 : //-----------------------------------------------------------------------------------------------------------------//
136 : AJC_DLGMAP_DEF(Main)
137 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG  )
138 :     AJC_DLGMAP_MSG(Main, WM_DESTROY     )
139 :     AJC_DLGMAP_MSG(Main, WM_TIMER       )
140 :
141 :     AJC_DLGMAP_CMD(Main, IDCANCEL       )
142 :     AJC_DLGMAP_CMD(Main, IDC_CMD_ENABLE )
143 :     AJC_DLGMAP_CMD(Main, IDC_CMD_DISABLE)
144 :     AJC_DLGMAP_CMD(Main, IDC_GRP_TSFORM )
145 :     AJC_DLGMAP_CMD(Main, IDC_LGF        )
146 : AJC_DLGMAP_END
147 :
148 : //-----------------------------------------------------------------------------------------------------------------//
149 : //  テキストエンコード表示                                                                                         //
150 : //-----------------------------------------------------------------------------------------------------------------//
151 : static  VO  ShowEncode(HWND hDlg, EAJCTEC OutTec, BOOL fBOM)
152 : {
153 :     UT      txt[64] = {0};
154 :     switch (OutTec) {
155 :         case AJCTEC_MBC:        MAjcStrCpy(txt, AJCTSIZE(txt), TEXT("MBC(S-JIS)")); break;  //  マルチバイト
156 :         case AJCTEC_UTF_8:      MAjcStrCpy(txt, AJCTSIZE(txt), TEXT("UTF-8"));      break;  //  ＵＴＦ－８
157 :         case AJCTEC_EUC_J:      MAjcStrCpy(txt, AJCTSIZE(txt), TEXT("EUC-J"));      break;  //  ＥＵＣ（日本語）
158 :         case AJCTEC_UTF_16LE:   MAjcStrCpy(txt, AJCTSIZE(txt), TEXT("UTF-16 LE")); break;  //  ＵＴＦ－１６ＬＥ
159 :         case AJCTEC_UTF_16BE:   MAjcStrCpy(txt, AJCTSIZE(txt), TEXT("UTF-16 BE")); break;  //  ＵＴＦ－１６ＢＥ
160 :     }
161 :     if (fBOM) {
162 :         MAjcStrCat(txt, AJCTSIZE(txt), TEXT(", BOM"));
163 :     }
164 :     AjcSetDlgItemStr(hDlg, IDC_LBL_ENC, txt);
165 : }
```

10. 棒グラフ／折れ線グラフ表示コントロール（AjcCtrlBarGraph クラス）

棒グラフをリアルタイムに表示するコントロールです。
棒グラフ表示コントロールの外観を以下に示します。

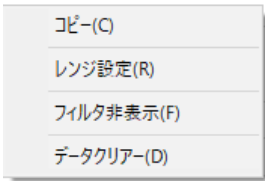


この例では２ヶのデータ項目を、色分けして表示しています。（最大８ヶのデータ項目を表示できます）
最大 10000 個（デフォルトは 50 個）のデータをバッファリングし、スクロールバーでスクロール表示することができます。
データ数がバッファの容量（個数）を超えた場合は、古いデータから順に破棄されます。

10.1. 機能概要

10.1.1. ポップアップメニュー

グラフ上で右クリックすると、以下のポップアップメニューが表示されます。

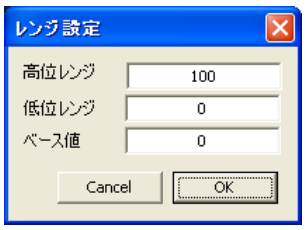


コピー	：	グラフ表示内容（ビットマップ）をクリップボードへコピーします
レンジ設定	：	棒グラフのレンジを設定します。
フィルタ非表示	：	コントロール左上のフィルタ（チェックボックス）を非表示にします。 次回は「フィルタ表示」メニューに変わります。
データクリアー	：	バッファリングされているデータを全て破棄し、画面をクリアーします。

コントロールにフォーカスがある状態で、F1 キーを押した場合も ポップアップメニューを表示します。

10.1.2. レンジ／ベース値設定

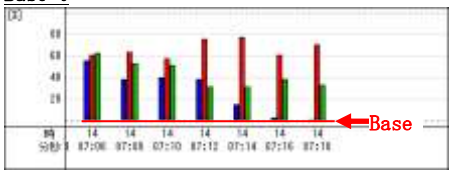
ポップアップメニューで「レンジ設定」を選択すると、以下のダイアログボックスが表示されます。



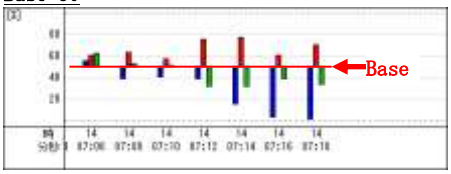
ここで、レンジ値／ベース値を入力し、「OK」ボタンを押すと、グラフのレンジ／ベース値が設定されます。
「Cancel」ボタンを押すと設定を中止します。

ベース値とは、棒グラフ描画の基点を意味します。

Base=0

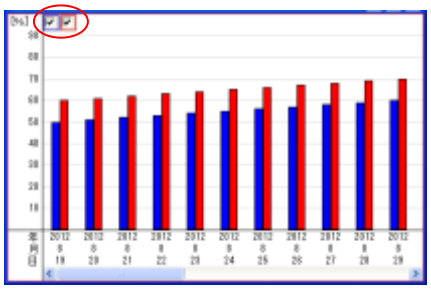


Base=50

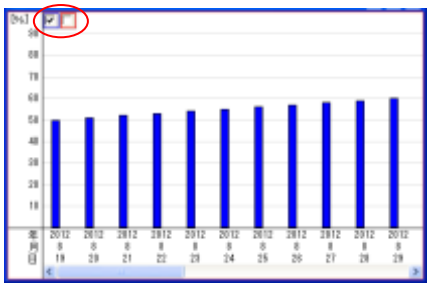


10.1.3. フィルタ機能

カーソルを、ウインドの左上隅に置くとチェクボックスが表示されます。
左上のチェックボックスは、データ項目の表示フィルタです。チェックを外すと、当該データ項目は非表示となります。



２ヶのデータ項目を表示

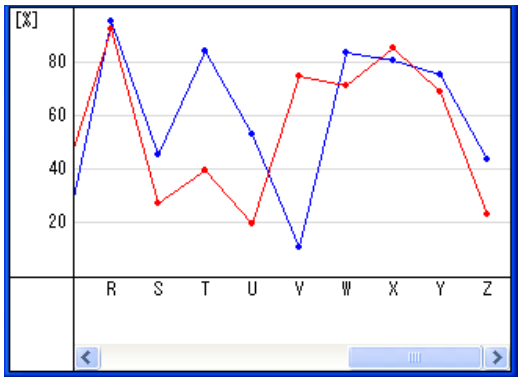


フィルタで、データ項目０だけを表示

10.1.4. 折れ線グラフ

「AJCBARS_LINEGRAPH」スタイルをセットすると、グラフを折れ線で表示します。
折れ線グラフのデータ表示間隔は「MinWidth」、プロパティで指定した値となります。

折れ線グラフの表示例



10.1.5. ファイルやディレクトリのドラッグ＆ドロップ

本コントロールにファイルやディレクトリをドラッグ＆ドロップした場合は、WM_COMMAND メッセージにて、親ウインドへ以下の通知を行います。

・AJCVTHN_DROPFILE ----- ファイルがドロップされたことを通知
・AJCVTHN_DROPDIR ----- フォルダがドロップされたことを通知

これらの通知では、ドロップされたファイルやディレクトリの個数を通知します。
ファイルやディレクトリのパス名は、本コントロールのＡＰＩ「 AjcBarGetDroppedFile(),AjcBarGetDroppedDir[Ex]()」 にて取得します。

尚、棒グラフ表示・コントロールでファイルやディレクトリのドラッグ＆ドロップを有効とするには、拡張ウインドスタイルに「WS_EX_ACCEPTFILES」を指定する必要があります。

10.2. コントロールのスタイル

棒グラフ表示コントロールのスタイルは、以下のとおりです。

名称	ビット	値	内容	備考
AJCBARS_NOBORDER	7	0x0080	コントロールの外枠を表示しない	
AJCBARS_LINEGRAPH	6	0x0040	折れ線グラフ表示	
AJCBARS_NOFILTER	4	0x0010	フィルタ（左上のチェックボックス）を表示しない	
AJCBARS_NOSCROLLBAR	3	0x0008	スクロールバー非表示	

尚、棒グラフ表示・コントロールでファイルやディレクトリのドラッグ＆ドロップを有効とするには、以下の拡張ウインドスタイルを指定する必要があります。

名称	ビット	値	内容
WS_EX_ACCEPTFILES	4	0x0010	ドラッグ＆ドロップでファイルを受け付けるようにする

10.3. プロパティ構造体

棒グラフ表示コントロールのプロパティは、以下の構造体で定義されます。

```
#define AJCBAR_MAXBUF   10000           //  最大データ数
#define AJCBAR_MAXITEM  8               //  最大データ項目数
#define AJCBAR_MAXSTR   32              //  最大棒タイトル文字数

typedef struct {
    double      RngL, RngH;             //  グラフレンジ（棒グラフ表示域の底部と上部の値）
    double      Base;                   //  棒グラフのベース値
    UI          MaxBuf;                 //  最大データ数
    UI          MaxItem;                //  最大アイテム数
    UI          ScaleWidth;             //  スケール値表示域の幅
    UI          BarWidth;               //  棒の幅
    UI          MinWidth;               //  棒域の最小幅
    UI          BarTtlLines;            //  棒タイトルの最大行数
    COLORREF    rgb[AJCBAR_MAXITEM];    //  棒の表示色
} AJCBARPROP, *PAJCBARPROP;
typedef const AJCBARPROP *PCAJCBARPROP;
```

各メンバ変数の内容は、以下のとおりです。

メンバ	内容	規定値	備考
RngL	グラフ低位のレンジ	0.0	
RngH	グラフ高位のレンジ	100.0	
Base	棒グラフのベース値	0.0	
MaxBuf	バッファ容量（２～１００００）	1024	※１
MaxItem	データ項目数（１～８）	2	※１
ScaleWidth	スケール値表示域の幅（３２～）	64	
BarWidth	棒の幅（４～）	10	
MinWidth	棒域の最小幅（１０～）	30	
BarTtlLines	棒タイトルの最大行数（１～）	3	
rgb	各データ項目の表示色	[0] : 0xFF0000 [1] : 0x0000FF [2] : 0x00C000 [3] : 0xE0E000 [4] : 0xFF00FF [5] : 0x00E0E0 [6] : 0x808080 [7] : 0x000000	Bit23-16 : 青の成分 Bit15- 8 : 緑の成分 Bit 7- 0 : 赤の成分

※１：バッファ容量／データ項目数を変更した場合、現在のデータは全て破棄されます。

10.4. キャプション文字列によるプロパティの設定

CreateWindow()/CreateWindowEx()の lpszWindowName 引数（ウインドキャプション）あるいは、ダイアログデザイン時におけるコントロールの Caption プロパティにより、タイムチャート・グラフコントロールのプロパティを設定することができます。

パラメタ（キャプション文字列）の形式は、以下のとおりです。([XXX]は、XXX を省略可能であることを意味します)

```
P: [L=fff], [H=fff], [Z=fff], [B=n], [I=n], [SW=n], [BW=n], [MW=n], [TL=n], [BC=n], [0=nnn], ・・・ [7=nnn]
```

文字列の先頭は「P:」でなければなりません。(「P:」の直後には空白を置けます)
「fff」は実数で、「nnn」は整数（１６進数の場合は先頭に'0x'を付加）で指定します。
各パラメタはカンマ（,）で区切ります。(カンマの前後には空白を置けます)
各パラメタの指定順序は任意です。

各パラメタの設定内容は、以下のとおりです。

キーワード	内　容
L	低位グラフレンジ
H	高位グラフレンジ
Z	棒グラフのベース値
B	バッファ容量（バッファに格納するデータ数）
I	有効なデータ項目数（１～８）
SW	スケール値表示域の幅
BW	棒の幅
MW	棒域の最小幅
TL	棒タイトルの最大行数
BC	コントロール外枠の表示色
0～7	各データ項目の表示色を指定

表示色は、１６進数で「0xbbggrr」の形式で指定します（bb:青成分，gg:緑成分，rr:赤成分）

設定例

```
P: L=0.0, H=1000.0
```
グラフレンジを、0～１０００とする

```
P: B=4096, I=1
```
バッファに格納するデータ数を４０９６個とし、データ項目数=1 とする。

```
P: BC=0x0000FF
```
コントロールの外枠を赤色で表示する。

10.5. テキストの取得と設定

テキストを取得した場合は、プロパティ設定内容を表す文字列を返します。
デフォルトでの、取得テキストは、以下のとおりです。

```
P: L=0, H=100, Z=0, B=50, I=2, SW=64, BW=10, MW=30, TL=3, BC=0xFF, 0=0xFF0000, 1=0xFF, 2=0xC000, 3=0xE0E000,
4=0xFF00FF, 5=0xE0E0, 6=0x808080, 7=0x0
```

テキストを設定する場合は、キャプション文字列によるプロパティの設定と同様に扱います。

10.6. プロパティの永続化

設定したプロパティを、プロファイル（.ini ファイル／レジストリ）に記録し、次回起動時に記録されているプロファイルを読み出すことにより、プロパティを永続的に有効とすることができます。

プロパティをプロファイルから読み出すには、ダイアログやウインドの初期化時に、AjcBarLoadProp()を実行します。
尚、初回実行時はプロファイルにプロパティが記録されていない為、AjcBarLoadProp()でプロパティのデフォルト値を指定します。
AjcBarLoadProp()でプロパティのデフォルト値を指定しない場合は、現在設定されているプロパティがデフォルト値となります。

デフォルトプロパティを指定する場合

```
AJCBARPROP DefProp;

DefProp.RngL = 0.0;
DefProp.RngH = 100.0;
   ・・・
AjcBarLoadProp(hwnd, "SectName", &DefProp);
```

デフォルトプロパティを指定しない場合

```
AjcBarSetRealRange(hwnd, 0.0, 100.0);
AjcBarLoadProp(hwnd, "SectName", NULL);
```

プロファイルにプロパティが記録されている場合は、AjcBarLoadProp()により読み出されたプロパティが設定される為、AjcBarSetRealRange()により設定された値は無効となります。(あらかじめ設定されているプロパティ値は、初回ロード時のデフォルト値となります)

現在設定されているプロパティをプロファイルに記録するには、ダイアログやウインドの終了時に、AjcBarSaveProp()を実行します。

プロパティをプロファイルに記録

```
AjcBarSaveProp(hwnd, "SectName");
```

デフォルトでは、プロファイルの記録先は、レジストリになります。
プロファイルの記録先を初期化ファイル（自プログラムパス名の拡張子を「.ini」としたファイル）とする場合は、プログラムの最初（AjcBarLoadProp(), AjcBarSaveProp()を実行する前）に「AjcSetProfileIsRegistory(FALSE);」を実行してください。

プロファイルへのアクセスは、AjcGetProfile･･･() と AjcPutProfile･･･() により行います。
これらの関数仕様やプロファイルアクセスに関するその他の情報は、「プロファイル・アクセス」章を参照してください。

10.7. サポートＡＰＩ

タイムチャート・グラフ表示コントロールのサポートＡＰＩ一覧を以下に示します。

#	関数名	内容	備考
1	AjcBarPut{Real/Int}Data	データ投与	
2	AjcBarSetBarTtl	棒タイトル名設定	
3	AjcBarSetVUnit	縦軸単位名設定	'%'や'Kg'等
4	AjcBarPurge	データクリアー	
5	AjcBarShowBorder	外枠の表示色と表示／非表示	
6	AjcBarShowFilter	フィルタ表示／非表示	
7	AjcBar{Set/Get}Prop	プロパティ設定／取得	
8	AjcBar{Set/Get}RealRange AjcBar{Set/Get}IntRange	レンジ設定／取得	
9	AjcBar{Set/Get}RealBase AjcBar{Set/Get}IntBase	ベース値設定／取得	
10	AjcBarSetBufSize	バッファサイズ設定	最大データ個数
11	AjcBarSetItemNumber	データ項目数設定	
12	AjcBarSetScaleWidth	スケール値表示域の幅設定	
13	AjcBarSetBarWidth	棒の幅設定	
14	AjcBarSetItemWidth	棒表示域の幅設定	
15	AjcBarSetTtlLines	棒タイトルの最大行数設定	
16	AjcBarGetBitmap	ビットマップデータ取得	
17	AjcBarLoadProp[Ex] AjcBarSaveProp{Ex}	プロファイルからプロパティ読出し／書き込み	
18	AjcBarSetNtcRClk	右クリック通知設定	
19	AjcBarEnablePopupMenu	ポップアップメニューの許可／禁止	
20	AjcBar{Set/Get}TipText	ツールチップの設定／取得	
21	AjcBar{Set/Get}TipShowAlways AjcBarSetTipShowAlwaysAll	ツールチップ表示条件の設定／取得	
22	AjcBar{Set/Get}ChkBoxTipText	フィィルタチェックボックス・ツールチップの設定／取得	
23	AjcBar{Set/Get}ChkBoxTipShowAlways	フィルタ・チェックボックス・ツールチップ表示条件の設定／取得	
24	AjcBar{Set/Get}ScrollPos	スクロール位置の設定／取得	
25	AjcBar{Set/Get}Filter	フィルタの設定／取得	
26	AjcBarSetHLineAtt	横線の属性設定	
27	AjcBarSetHLinePos	横線の描画位置設定	
28	AjcBarEnableHLine	横線描画の許可／禁止	
29	AjcBarGetCharSize	文字サイズの取得	半角文字のサイズ
30	AjcBarGetDroppedFile	ドロップされたファイル名取得	
31	AjcBarGetDroppedDir[Ex]	ドロップされたディレクトリ名取得	
32	AjcBarSetTitleText	タイトル文字列の設定	
33	AjcBarSetTextFont	テキスト描画フォント設定	
34	AjcBarTextOut	テキスト描画（ピクセル位置指定）	
35	AjcBarPrintF	書式テキスト描画	
36	AjcBarGetText	描画テキスト取得	
37	AjcBarClear[All]Text	描画テキスト消去	
38	AjcBarClear	全てのデータ（プロットデータ，描画テキスト）消去	
39	AjcBarSetEvtCallBack	イベント通知用コールバック関数の設定	

10.7.1. データ投与(AjcBarPut{Real/Int}Data)

形　式 ： BOOL　AjcBarPutRealData(HWND hwnd, double dat[], C_UTP pBarTtl); --- 実数データ投与
BOOL　AjcBarPutIntData(HWND hwnd, int　　dat[], C_UTP pBarTtl); --- 整数データ投与

引　数 ： hwnd　　- コントロールのウインドハンドル
dat　　- 投与するデータ値配列のアドレス
pBarTtl - 棒のタイトル文字列のアドレス

説　明 ： 棒グラフへデータを投与します。
棒タイトルが複数行である場合は、pBarTtl で指定する文字列を改行('¥n')で区切ります。

戻り値 ： TRUE　- 成功
FALSE - 失敗

10.7.2. 棒タイトル名設定(AjcBarSetBarTtl)

形　式 ： BOOL　AjcBarSetBarTtl(HWND hwnd, C_UTP pTtl);

引　数 ： hwnd　　- コントロールのウインドハンドル
pTtl　　- 棒タイトル名文字列のアドレス

説　明 ： 棒グラフの左下へ表示する 棒タイトル名を設定します。
棒タイトル名が複数行である場合は、pTtl で指定する文字列を改行('¥n')で区切ります。

戻り値 ： TRUE　- 成功
FALSE - 失敗

10.7.3. 縦軸単位名設定(AjcBarSetVUnit)

形　式 ： BOOL　AjcBarSetVUnit(HWND hwnd, C_UTP pUnit);

引　数 ： hwnd　　- コントロールのウインドハンドル
pUnit　　- 棒タイトル名文字列のアドレス

説　明 ： 棒グラフの左上へ表示する 棒グラフ値の単位名（"%" や "Kg" 等）を設定します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

10.7.4. データクリアー(AjcBarPurge)

形　式 ： BOOL　AjcBarPurge(HWND hwnd);

引　数 ： hwnd　　- コントロールのウインドハンドル

説　明 ： 棒グラフの全データを破棄します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

10.7.5. 外枠の表示色と表示／非表示(AjcBarShowBorder)

形　式 ： BOOL　AjcBarShowBorder(HWND hwnd, BOOL fShow, COLORREF rgb);

引　数 ： hwnd　- コントロールのウインドハンドル
fShow　- 外枠表示フラグ（TRUE:表示, FALSE:非表示）
rgb　- 外枠の表示色

説　明 ： 棒グラフの外枠の表示色と表示／非表示と、表示色を設定します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

10.7.6. フィルタ表示／非表示(AjcBarShowFilter)

形　式 ： BOOL　AjcBarShowFilter(HWND hwnd, BOOL fShow);

引　数 ： hwnd　- コントロールのウインドハンドル
fShow　- フィルタ表示フラグ（TRUE:表示, FALSE:非表示）

説　明 ： フィルタ（棒グラフ表示域・左上のチェックボックス）の表示／非表示を行います。

戻り値 ： TRUE　- 成功
FALSE - 失敗

10.7.7. プロパティ設定／取得(AjcBar{Set/Get}Prop)

形　式 ： BOOL　AjcBarSetProp(HWND hwnd, PCAJCBARPROP pProp); --- 設定
BOOL　AjcBarGetProp(HWND hwnd, PAJCBARPROP　pBuf); ---- 取得

引　数 ： hwnd　- コントロールのウインドハンドル
pProp　- 設定するプロパティ値のアドレス
pBuf　- プロパティを格納するバッファのアドレス

説　明 ： 棒グラフのプロパティを設定／取得します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

10.7.8. レンジ設定／取得(AjcBar{Set/Get}RealRange)

形　式 ： BOOL　AjcBarSetRealRange(HWND hwnd, double　low, double　high); --- 実数でレンジ設定
BOOL　AjcBarGetRealRange(HWND hwnd, double *pLow, double *pHigh); --- 実数でレンジ取得

BOOL　AjcBarSetIntRange(HWND hwnd, int　low, int　high); ---------- 整数でレンジ設定
BOOL　AjcBarGetIntRange(HWND hwnd, int *pLow, int *pHigh); ---------- 整数でレンジ取得

引　数 ： hwnd　- コントロールのウインドハンドル
low　- レンジ低位値
high　- レンジ高位値
pLow　- レンジ低位値を格納するバッファのアドレス（不要時はNULL）
pHigh　- レンジ高位値を格納するバッファのアドレス（不要時はNULL）

説　明 ： 棒グラフのレンジ値を設定／取得します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

10.7.9. ベース値設定／取得(AjcBar{Set/Get}{Real/Int}Base)

形　式 ： BOOL　AjcBarSetRealBase(HWND hwnd, double　base); --- 実数でベース値設定
BOOL　AjcBarGetRealBase(HWND hwnd, double *pBase); --- 実数でベース値取得

BOOL　AjcBarSetIntBase(HWND hwnd, int　base); ------- 整数でベース値設定
BOOL　AjcBarGetIntBase(HWND hwnd, int *pBase); ------- 整数でベース値取得

引　数 ： hwnd　- コントロールのウインドハンドル
base　- ベース値
pBase　- ベース値を格納するバッファのアドレス

説　明 ： 棒グラフのベース値を設定／取得します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

10.7.10. バッファサイズ設定(AjcBarSetBufSize)

形　式 ： BOOL　AjcBarSetBufSize(HWND hwnd, int n);

引　数 ： hwnd　- コントロールのウインドハンドル
n　- バッファ・サイズ（最大格納するデータ数）

説　明 ： 棒グラフ値を格納するバッファサイズを、最大格納データ数で指定します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

10.7.11. データ項目数設定(AjcBarSetItemNumber)

形　式 ： BOOL　AjcBarSetItemNumber (HWND hwnd, int n);

引　数 ： hwnd　- コントロールのウインドハンドル
n　- データ項目数（１～８）

説　明 ： 棒グラフの項目数を設定します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

10.7.12. スケール値表示域の幅設定(AjcBarSetScaleWidth)

形　式 ： BOOL　AjcBarSetScaleWidth (HWND hwnd, int width);

引　数 ： hwnd　- コントロールのウインドハンドル
width　- スケール値表示域の幅（ピクセル数）

説　明 ： スケール値表示域の幅を、ピクセル数で設定します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

10.7.13. 棒の幅設定(AjcBarSetBarWidth)

形　式 ： BOOL　AjcBarSetBarWidth (HWND hwnd, int width);

引　数 ： hwnd　　- コントロールのウインドハンドル
width　- 棒の幅（ピクセル数）

説　明 ： 棒の幅を、ピクセル数で設定します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

10.7.14. 棒表示域の幅設定(AjcBarSetItemWidth)

形　式 ： BOOL　AjcBarSetItemWidth　(HWND hwnd, int width);

引　数 ： hwnd　　- コントロールのウインドハンドル
width　- 棒域の幅（ピクセル数）

説　明 ： 棒域の幅を、ピクセル数で設定します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

10.7.15. 棒タイトルの最大行数設定(AjcBarSetTtlLines)

形　式 ： BOOL　AjcBarSetTtlLines(HWND hwnd, int lines);

引　数 ： hwnd　　- コントロールのウインドハンドル
lines　- 棒タイトルの最大行数（１～）

説　明 ： 棒タイトルの最大行数を設定します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

10.7.16. ビットマップデータ取得 (AjcTchGetBitmap)

形　式 ： HBITMAP　AjcBarGetBitmap (HWND hwnd);

引　数 ： hwnd　- コントロールのウインドハンドル

説　明 ： 現在表示している､グラフイメージのビットマップデータを取得します。

戻り値 ： ≠NULL - 成功（ビットマップハンドル）
＝NULL - 失敗

10.7.17. プロファイルからプロパティ値読み出し (AjcBarLoadProp / AjcBarLoadPropEx)

形　式 ： BOOL　AjcBarLoadProp　(HWND hwnd, C_UTP pProfileSect, PCAJCTCPROP pDefProp); --- 読み出し
BOOL　AjcBarLoadPropEx(HWND hwnd, C_UTP pProfileSect, PCAJCTCPROP pDefProp); ---　〃

BOOL　AjcBarSaveProp　(HWND hwnd, C_UTP pProfileSect); ------------------------- 書き込み
BOOL　AjcBarSavePropEx(HWND hwnd, C_UTP pProfileSect); -------------------------　〃

引　数 ： hwnd　　- コントロールのウインドハンドル
pProfileSect - プロファイル・セクション名（文字列）へのポインタ
pDefProp　　- デフォルトプロパティ値へのポインタ（現在の設定値をデフォルトとする場合は NULL）

説　明 ： AjcBarLoadProp は、プロファイル（.ini ファイル／レジストリ）からプロパティ値を読み出して設定します。
AjcBarLoadPropEx は、プロパティ値に加えて、フィルタ設定値とウインドスタイルも読み出して設定します。
pDefProp は、プロファイルに当該プロパティ値が記録されていない場合の、デフォルト・プロパティ値を指定します。
pDefProp=NULL とした場合は、現在設定されているプロパティ値を、デフォルト・プロパティとして扱います。

AjcBarSaveProp は、現在設定されているプロパティ値を、プロファイル（.ini ファイル／レジストリ）へ記録します。
AjcBarSavePropEx は、プロパティ値に加えて、フィルタ設定値とウインドスタイルも記録します。

プロファイルへセーブ／ロードする項目は以下のとおりです。

#	内容	キー名称	備考
1	棒グラフレンジ低位値，高位値	RngL，RngH	
2	棒グラフのベース値	Base	
3	バッファ容量	MaxBuf	蓄積可能なデータ数
4	データ項目数	MaxItem	１～８
5	スケール値表示域の幅	ScaleWidth	ピクセル数
6	棒の幅	BarWidth	ピクセル数
7	棒域の最小幅	MinWidth	ピクセル数
8	棒タイトルの最大行数	BarTtlLines	
9	棒の表示色	rgb0 ～ rgb7	

AjcBarLoadPropEx(),AjcBarSavePropEx()では、さらに以下の項目が追加されます。

#	内容	キー名称	備考
1	データ項目選択状態	FilterValue	チェックボックスの内容
2	スタイル値	WndStyle	

戻り値 ： TRUE　- 成功
FALSE - 失敗

10.7.18. 右クリック通知設定 (AjcBarSetNtcRClk)

形　式 ： BOOL　AjcBarSetNtcRClk(HWND hwnd, BOOL fNtcRClk, UI MsgRBDown, UI MsgRBUp);

引　数 ： hwnd　　- コントロールのウインドハンドル
fNtcRClk　- 右ボタンの DOWN/UP 通知フラグ（TRUE:通知する，FALSE:通知しない）
MsgRBDown　- 右ボタン押下時の通知メッセージコード　（０の場合は非通知）
MsgRBUp　- 右ボタン離され時の通知メッセージコード（０の場合は非通知）

説　明 ： コントロールを右クリックした場合に、当該操作を親ウインドへ通知するか否かを設定します。
MsgRBDown，MsgRBUp は、WM_USER+100 以降，WM_APP+500 以降か、RegisterWindowMessage()で取得したコードを指定します。
各引数と、右クリック通知動作は以下のとおりです。

引数			Shift/ Ctrl	メッセージ	wParam	備考
fNtcRClk	MsgRBDown	MsgRBUp				
FALSE (default	－	－	未押下	WM_COMMAND	－（非通知）	ポップアップメニュー表示
			押下		ID + AJCBARN_RCLICK	
TRUE	いずれかが 0 以外		－	MsgRBDown(押下時)	WM_RBUTTONDOWN の wParam	MsgRBDown ＝０の場合は非通知
				MsgRBUp（離し時）	WM_RBUTTONUP　の wParam	MsgRBUp　＝０の場合は非通知
	0	0	－		－（非通知）	－

右クリックの通知を禁止するには、fNtcRClk=TRUE，MsgRBDown=0，MsgRBUp=0 とします。

戻り値 ： TRUE　- 成功
FALSE - 失敗

10.7.19. ポップアップメニューの許可／禁止 (AjcTchEnablePopupMenu)

形　式 ： BOOL　AjcTchEnablePopupMenu (HWND hwnd, BOOL fEnable);

引　数 ： hwnd　　- コントロールのウインドハンドル
fEnable　- ポップアップメニューの許可(TRUE)／禁止(FALSE)

説　明 ： 右クリックによるポップアップメニューを許可／禁止します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

備　考 ： 「AjcTchSetNtcRClk(hwnd, !fEnable, WM_RBUTTONDOWN, WM_RBUTTONUP);」を実行します。

10.7.20. ツールチップの設定／取得 (AjcBar{Set|Get}TipText)

形　式 ： BOOL　AjcBarSetTipText (HWND hwnd, C_UTP pTxt);
UI　　AjcBarGetTipText (HWND hwnd, BCP pBuf, UI lBuf)

引　数 ： hwnd　　- コントロールのウインドハンドル
pTxt　　- ツールチップ（ツールヒント）文字列のアドレス（表示しない場合はNULL)
pBuf　　- 取得するツールチップ文字列を格納するバッファのアドレス
lBuf　　- 取得するツールチップ文字列を格納するバッファのバイト数／文字数

説　明 ： ツールヒント文字列を設定／取得します。

戻り値 ： 設定時：TRUE　- 成功　　取得時：チップテキストのバイト数／文字数
FALSE - 失敗

10.7.21. ツールチップ表示条件の設定／取得(AjcBar{Set|Get}TipShowAlways)

形　式 ： BOOL　AjcBarSetTipShowAlways(HWND hwnd, BOOL fShowAlways); --- 設定
BOOL　AjcBarGetTipShowAlways(HWND hwnd); ---------- 取得
BOOL　AjcBarGetTipShowAlwaysAll(HWND hwnd, BOOL fShowAlways); --- すべて設定

引　数 ： hwnd　　- コントロールのウインドハンドル
fShowAlways　- ツールチップ表示条件（TRUE:非アクティブ時も表示,　FALSE:非アクティブ時は非表示)

説　明 ： ツールチップの表示条件（自プログラムが非アクティブ時の表示／非表示）を設定／取得します。を設定／取得します。
AjcBarGetTipShowAlwaysAll()は、フィルタチェックボックスを含めたすべてのツールチップ表示条件を設定します。

戻り値 ： 設定時：TRUE　- 成功　　取得時：ツールチップ表示条件
FALSE - 失敗

10.7.22. フィルタ・チェックボックス・ツールチップの設定／取得 (AjcBar{Set/Get}ChkBoxTipText)

形　式 ： BOOL　AjcBarSetChkBoxTipText(HWND hwnd, UI n, C_UTP pTxt); --------- 設定
BOOL　AjcBarGetChkBoxTipText(HWND hwnd, UI n, UTP pBuf, UI lBuf); -- 取得

引　数 ： hwnd　　- コントロールのウインドハンドル
n　　- チェックボックスのインデクス
pTxt　　- ツールチップ（ツールヒント）文字列のアドレス（表示しない場合はNULL)
pBuf　　- ツールチップ（ツールヒント）文字列を格納するバッファのアドレス
lBuf　　- ツールチップ（ツールヒント）文字列を格納するバッファの文字数

説　明 ： フィルタ・チェックボックスのツールチップ（ツールヒント文字列）を設定／取得します。
ｎは、チェックボックスのインデクスで、左から順に０～７を指定します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

10.7.23. フィルタチェックボックスのツールチップ表示条件 設定／取得(AjcBar{Set|Get}ChkBoxTipShowAlways)

形　式 ： BOOL　AjcBarSetChkBoxTipShowAlways(HWND hwnd, UI n, BOOL fShowAlways)； --- 設定
BOOL　AjcBarGetChkBoxTipShowAlways(HWND hwnd, UI n)； ---------- 取得

引　数 ： hwnd　　- コントロールのウインドハンドル
n　　- チェックボックスのインデクス（0～15）
fShowAlways　- ツールチップ表示条件（TRUE:非アクティブ時も表示，　FALSE:非アクティブ時は非表示）

説　明 ： フィルタチェックボックスのツールチップの表示条件（自プログラムが非アクティブ時の表示／非表示）を設定／取得します。

戻り値 ： 設定時：TRUE　- 成功　　取得時：ツールチップ表示条件
FALSE - 失敗

10.7.24. スクロール位置の設定／取得 (AjcBar{Set/Get}ScrollPos)

形　式 ： BOOL　AjcBarSetScrollPos(HWND hwnd, int pos)； - スクロール位置の設定
int　AjcBarGetScrollPos(HWND hwnd)； ---------- スクロール位置の取得

引　数 ： hwnd　　- コントロールのウインドハンドル
pos　　- 設定するスクロール位置

説　明 ： pos で設定された位置までスクロールして棒グラフを表示します。
この関数でスクロール位置を設定する場合は、当該棒グラフのデータ更新を停止している状態で行ってください。

戻り値 ： 設定時：TRUE　- 成功　　取得時：スクロール位置（０～）
FALSE - 失敗

10.7.25. フィルタの設定／取得 (AjcBar{Set/Get}Filter)

形　式 ： BOOL　AjcBarSetFilter(HWND hwnd, UI n, BOOL state)；
BOOL　AjcBarGetFilter(HWND hwnd, UI n)；

引　数 ： hwnd　　- コントロールのウインドハンドル
n　　- チェックボックスのインデクス（０～７）
state　　- 設定するフィルタの状態(FALSE:非表示 / TRUE:表示)

説　明 ： フィルタ・チェックボックスの設定（当該項目の表示／非表示の設定）／取得を行います。
ｎは、チェックボックスのインデクスで、左から順に０～７を指定します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

10.7.26. 横線の属性設定 (AjcBarSetHLineAtt)

形　式 ： BOOL　AjcBarSetHLineAtt (HWND hwnd, UI id, COLORREF color, int width, int style);

引　数 ： hwnd　　- コントロールのウインドハンドル
id　　- 横線の識別（０～７）
color　　- 描画色
width　　- 線の太さ（１～）
style　　- 線種

説　明 ： 棒グラフに描画する横線の属性を設定します。
「style」は、線の種別であり、以下のいずれかを指定します。

・AJCBAR_SOLID　　- 実線
・AJCBAR_DASH　　- 破線
・AJCBAR_DOT　　- 点線
・AJCBAR_DASHDOT　　- １点鎖線
・AJCBAR_DASHDOTDOT　- ２点鎖線
・AJCBAR_NULL　　- ヌル線（描画しないのと同じ）

点線等、実線以外の線を描画する場合は、線の太さ＝１(width=1)を指定しなければなりません。

戻り値 ： TRUE　- 成功
FALSE - 失敗

10.7.27. 横線の描画位置設定 (AjcBarSetHLinePos)

形　式 ： BOOL　AjcBarSetHLinePos (HWND hwnd, UI id, double pos);

引　数 ： hwnd　　- コントロールのウインドハンドル
id　　- 横線の識別（０～７）
pos　　- 描画位置

説　明 ： 棒グラフに描画する横線の位置を設定します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

10.7.28. 横線描画の許可／禁止(AjcBarEnableHLine)

形　式 ： BOOL　AjcBarEnableHLine (HWND hwnd, UI id, BOOL fEnable);

引　数 ： hwnd　　- コントロールのウインドハンドル
id　　- 横線の識別（０～７）
fEnable　　- 描画指定（TRUE- 描画する，　FALSE- 描画しない）

説　明 ： 棒グラフ上に横線を描画するか否かを指定します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

10.7.29. 文字サイズの取得 (AjcBarGetCharSize)

形　式 ： BOOL　AjcBarGetCharSize (HWND hwnd, LPSIZE pSize);

引　数 ： hwnd　　- コントロールのウインドハンドル
pSize　　- 文字サイズを格納するバッファのアドレス

説　明 ： スケール値や棒タイトル等の(半角)文字のサイズを取得します。
pSize->cx に文字の幅（横ピクセル数）を，pSize->cy に文字の高さ（縦ピクセル数）を格納します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

10.7.30. ドロップされたファイル名取得(AjcBarGetDroppedFile)

形　式 ： BOOL　AjcBarGetDroppedFile (HWND hwnd, UT buf[MAX_PATH]);

引　数 ： hwnd　　- コントロールのウインドハンドル
buf　　- ファイルのパス名を格納するバッファのアドレス

説　明 ： ドラッグ＆ドロップされたファイルのパス名を取得します。
本関数は、通知メッセージ（AJCBARN_DROPFILE）が通知された場合、ドロップされたファイルのパス名を取得するために実行します。
本ＡＰＩにより全てのドロップされたファイル名を取得済の場合は、FALSE を返します。

戻り値 ： TRUE　- ファイルのパス名をバッファに格納した
FALSE - ドロップされたファイル名なし（ドロップしたファイル数を超えて実行された）

10.7.31. ドロップされたディレクトリ名取得(AjcBarGetDroppedDir[Ex])

形　式 ： BOOL　AjcBarGetDroppedDir　(HWND hwnd, UT buf[MAX_PATH]);
BOOL　AjcBarGetDroppedDirEx (HWND hwnd, UT buf[MAX_PATH] , BOOL fTailIsDelimiter);

引　数 ： hwnd　　- コントロールのウインドハンドル
buf　　- ファイルのパス名を格納するバッファのアドレス
fTailIsDelimiter - TRUE　：ディレクトリパス名の末尾に「¥」を付加する
FALSE：ディレクトリパス名の末尾に「¥」を付加しない

説　明 ： ドラッグ＆ドロップされたディレクトリのパス名を取得します。
本関数は、通知メッセージ（AJCBARN_DROPDIR）が通知された場合、ドロップされたディレクトリのパス名を取得するために実行します。
本ＡＰＩにより全てのドロップされたファイル名を取得済の場合は、FALSE を返します。

戻り値 ： TRUE　- ディレクトリのパス名をバッファに格納した
FALSE - ドロップされたディレクトリ名なし（ドロップしたディレクトリ数を超えて実行された）

10.7.32. タイトル文字列の設定 (AjcBarSetTitleText)

形　式 ： BOOL　AjcBarSetTitleText (HWND hwnd, C_BCP pTitleText, COLORREF TextColor, COLORREF BackColor, HFONT hFont);

引　数 ： hwnd　　- コントロールのウインドハンドル
pTitleText　- タイトルテキストへのポインタ（NULL 指定時は、タイトル非表示）
TextColor　- テキスト描画色（－１指定時は前回設定値，デフォルト色＝白）
BackColor　- テキスト背景色（－１指定時は前回設定値，デフォルト色＝グレー）
hFont　　- タイトルテキストのフォントハンドル（NULL 指定時はデフォルトのフォント）

説　明 ： コントロールウインドの右上にタイトル文字列を表示します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

10.7.33. テキスト描画フォント設定(AjcBarSetTextFont)

形　式 ： HFONT　AjcBarSetTextFont (HWND hwnd, HFONT hFont);

引　数 ： hwnd　　- コントロールのウインドハンドル

説　明 ： AjcBarTextOut ()や、AjcBarPrintF()で描画するテキストのフォントを設定します。

戻り値 ： ≠NULL - 前回のフォントハンドル
＝NULL - 失敗

10.7.34. テキスト描画(AjcBarTextOut)

形　式 ： UI　AjcBarTextOut (HWND hwnd, int x, int y, C_UTP pTxt);

引　数 ： hwnd　- コントロールのウインドハンドル
x　　- 描画ピクセルX位置（テキストを右隅／中央に表示する場合は、「AJCTXO_RIGHT ± n」or「AJCTXO_CENTER ± n」を指定）
y　　- 描画ピクセルY位置（テキストを下隅／中央に表示する場合は、「AJCTXO_BOTTOM ± n」or「AJCTXO_CENTER ± n」を指定）
pTxt　- 描画するテキストへのポインタ

説　明 ： 指定したピクセル描画位置へテキストを表示します。
pTxt で指定する描画テキストには、エスケープシーケンス（「テキスト描画」の章参照）を含めることができます。
但し、パレット色（０～７）は、データ項目の描画色と同じとなります。

戻り値 ： ≠0 - テキストキー
＝0 - 失敗

10.7.35. 書式テキスト描画(AjcBarPrintF)

形　式 ： UI　AjcBarPrintF　(HWND hwnd, int x, int y, C_UTP pFmt, ...);

引　数 ： hwnd　- コントロールのウインドハンドル
x　　- 描画ピクセルX位置（テキストを右隅／中央に表示する場合は、「AJCTXO_RIGHT　± n」or「AJCTXO_CENTER　± n」を指定）
y　　- 描画ピクセルY位置（テキストを下隅／中央に表示する場合は、「AJCTXO_BOTTOM ± n」or「AJCTXO_CENTER　± n」を指定）
pFmt　- 書式テキストへのポインタ（printf()と同じ）

説　明 ： 指定したピクセル位置へ書式化したテキストを表示します。
描画するテキストには、エスケープシーケンス（「テキスト描画」の章参照）を含めることができます。
但し、パレット色（０～７）は、データ項目の描画色と同じとなります。

戻り値 ： ≠0 - テキストキー
＝0 - 失敗

10.7.36. 描画テキスト取得(AjcBarGetText)

形　式 ： UI　AjcBarGetText (HWND hwnd, UI key, UTP pBuf, UI lBuf);

引　数 ： hwnd　　- コントロールのウインドハンドル
key　　- テキストキー（AjcTchTextOut[V]()や、AjcTchPrintF[V]()の戻り値）
pBuf　　- 取得したテキストを格納するバッファへのポインタ（不要時はNULL）
lBuf　　- 取得したテキストを格納するバッファのバイト数／文字数

説　明 ： AjcBarTextOut()や、AjcBarPrintF()で描画した文字列を取得します。

戻り値 ： ≠0 - 描画テキストのバイト数／文字数
＝0 - 失敗

10.7.37. 描画テキスト消去(AjcBarClear[All]Text)

形　式 ： BOOL　AjcBarClearText　(HWND hwnd, UI key);　// １つの描画テキスト消去
BOOL　AjcBarClearAllText (HWND hwnd);　　// すべての描画テキスト消去

引　数 ： hwnd　　- コントロールのウインドハンドル
key　　- テキストキー（AjcBarTextOut[V]()や、AjcBarPrintF[V]()の戻り値）

説　明 ： AjcBarTextOut()や、AjcBarPrintF()で描画した文字列を消去します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

10.7.38. 全てのデータ消去(AjcBarClear)

形　式 ： BOOL　AjcBarClear (HWND hwnd);

引　数 ： hwnd　　- コントロールのウインドハンドル

説　明 ： 全てのプロットデータ、描画テキストデータを消去します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

10.7.39. ベント通知用コールバック関数の設定(AjcBarSetEvtCallBack)

形　式　：　BOOL AjcBarSetEvtCallBack (HWND hwnd, UX cbp, VO (CALLBACK *cbNtcEvt)(HWND hCtrl, UX wParam, UX lParam, UX cbp));

引　数　：　hwnd　　- コントロールのウインドハンドル
cbp　　- コールバックパラメタ
cbNtcEvt　- イベント通知用コールバック関数（NULL:コールバック関数を無効化）

説　明　：　イベント通知用のコールバック関数を設定します。
通常、イベントは WM_COMMAND メッセージで通知されますが、コールバック関数を設定することにより、コールバック関数の呼び出しにより通知されるようになります。

戻り値　：　TRUE　- 成功
FALSE - 失敗

コールバック：　コールバック関数の仕様は、以下のとおりです。

cbNtcEvt（イベント通知）

形　式　：　VO CALLBACK *cbNtcEvt*(HWND hCtrl, UX wParam, UX lParam, UX cbp)；

引　数　：　hCtrl　- コントロールのウインドハンドル
wPatram - コントロール識別ＩＤ，通知コード（WM_COMMAND による通知と同じ）
lPatram - イベントパラメタ　　（WM_COMMAND による通知と同じ）
cbp　　- コールバックパラメタ

説　明　：各種イベントを通知します。（詳細は「コントロールの通知メッセージ」を参照）

戻り値　：なし

10.8. 通知情報の取得ＡＰＩ

コントロールからの通知メッセージ（WM_COMMAND）に伴うパラメタを取得するＡＰＩ群です。
「AjcSetCmdWithHdl(TRUE);」を実行した場合、各通知メッセージ（WM_COMMAND）の lParam は通知内容に伴うパラメタは通知されず、lParam＝コントロールのウインドハンドルとなります。
この場合、通知内容に伴うパラメタは以下のＡＰＩで取得します。

#	関 数 名	内容	対応する通知	
1	AjcBarGetNtcRng	グラフレンジ情報の取得	AJCBARN_RANGE	グラフレンジ通知
2	AjcBarGetNtcScrPos	スクロール位置の取得	AJCBARN_SCRPOS	スクロール位置通知
3	AjcBarGetNtcFiles	ドロップしたファイル数の取得	AJCBARN_DROPFILE	ファイルドロップ通知
4	AjcBarGetNtcDirs	ドロップしたディレクトリ数の取得	AJCBARN_DROPDIR	ディレクトリドロップ通知
5	AjcBarGetNtcRClk	右クリック情報の取得	AJCBARN_RCLICK	右クリック通知

10.8.1. グラフレンジ情報の取得（AjcBarGetNtcRng）

形　式　：　PAJCBAR_NTC_RANGE AjcBarGetNtcRng (HHWND hwnd);

引　数　：　hwnd　　- コントロールのウインドハンドル

説　明　：　グラフレンジ通知（AJCBARN_RANGE）時のグラフレンジ情報を取得します。

戻り値　：　グラフレンジ情報（AJCBAR_NTC_RANGE）へのポインタ

10.8.2. ドロップしたファイル数の取得（AjcBarGetNtcFiles）

形　式　：　UI　AjcBarGetNtcFiles (HHWND hwnd);

引　数　：　hwnd　　- コントロールのウインドハンドル

説　明　：　ファイルドロップ通知（AJCBARN_DROPFILE）時の、ドロップされたファイルの個数を取得します。

戻り値　：　ドロップされたファイルの個数

10.8.3. ドロップしたディレクトリ数の取得（AjcBarGetNtcDirs）

形　式　：　UI　AjcBarGetNtcDirs (HHWND hwnd);

引　数　：　hwnd　　- コントロールのウインドハンドル

説　明　：　ディレクトリドロップ通知（AJCBARN_DROPDIR）時の、ドロップされたディレクトリの個数を取得します。

戻り値　：　ドロップされたディレクトリの個数

10.8.4. 右クリック情報の取得（AjcBarGetNtcRClk）

形　式　：　PAJCBARRCLK AjcBarGetNtcRClk (HHWND hwnd);

引　数　：　hwnd　　- コントロールのウインドハンドル

説　明　：　右クリック通知（AJCBARN_RCLICK）時の、右クリック情報を取得します。

戻り値　：　右クリック情報（AJCBARRCLK）へのポインタ

10.9. コントロールの通知メッセージ

コントロールからの通知メッセージは、WM_COMMAND メッセージにより通知されます。
WM_COMMAND メッセージの wParam には以下の情報が設定されます。

・LOWORD(wParam) - コントロールの識別ＩＤ
・HIWORD(wParam) - 通知メッセージ

通知メッセージコードは、以下のとおりです。

#	通知メッセージコード	内容	備考
1	AJCBARN_RANGE	グラフレンジ通知	
2	AJCBARN_SCRPOS	スクロール位置通知	
3	AJCBARN_DBLCLK	ダブルクリック通知	
4	AJCBARN_DROPFILE	ファイルドロップ通知	
5	AJCBARN_DROPDIR	ディレクトリドロップ通知	
6	AJCBARN_RCLICK	右クリック通知	SHIFT/CTRL + 右クリック時

10.9.1. レンジ通知 (AJCBARN_RANGE)

説　明　：　グラフのレンジが変更されたことを通知します。
lParam には、変更後のレンジ情報のアドレスが設定されます。レンジ情報の形式は以下のとおりです。

```
typedef struct {
    double  RngL;       //  グラフレンジ低位値
    double  RngH;       //  グラフレンジ高位値
} AJCBAR_NTC_RANGE, *PAJCBAR_NTC_RANGE;
```

パラメタ：　wParam - コントロール識別ＩＤと、通知メッセージコード（AJCBARN_RANGE）
lParam - レンジ値情報のアドレス（PAJCBAR_NTC_RANGE）（MFC 使用時はコントロールのウインドハンドル）

戻り値　：　なし

10.9.2. スクロール位置通知 (AJCBARN_SCRPOS)

説　明　：　現在のスクロール位置（左端表示位置）を通知します

パラメタ：　wParam - コントロール識別ＩＤと、通知メッセージコード（AJCBARN_SCRPOS）
lParam - 現在のスクロール位置（MFC 使用時はコントロールのウインドハンドル）

戻り値　：　なし

10.9.3. ダブルクリック通知（AJCBARN_DBLCLK）

説　明　：　ダブルクリックされたことを通知します。

パラメタ：　wParam - コントロールの識別ＩＤと通知メッセージコード（AJCBARN_DBLCLK）
lParam – ダブルクリック位置（ｘ：下位ワード，ｙ：上位ワード）（MFC 使用時はコントロールのウインドハンドル）

戻り値　：　なし（ゼロを返してください）

10.9.4. ファイルドロップ通知（AJCBARN_DROPFILE）

説　明 ： ドラッグ＆ドロップ操作で、ファイルがドロップされたことを通知します。
本通知では、ドロップされたファイルの個数だけを通知します。
AjcBarGetDroppedFile() によりドロップされたファイルのパス名を取得できます。

パラメタ： wParam - コントロールの識別ＩＤと通知メッセージコード（AJCBARN_DROPFILE）
lParam - ドロップされたファイルの個数（MFC 使用時はコントロールのウインドハンドル）

戻り値 ： なし（ゼロを返してください）

10.9.5. ディレクトリドロップ通知（AJCBARN_DROPDIR）

説　明 ： ドラッグ＆ドロップ操作で、ディレクトリがドロップされたことを通知します。
本通知では、ドロップされたディレクトリの個数だけを通知します。
AjcBarGetDroppedDir[Ex]() によりドロップされたディレクトリのパス名を取得できます。

パラメタ： wParam - コントロールの識別ＩＤと通知メッセージコード（AJCBARN_DROPDIR）
lParam - ドロップされたディレクトリの個数（MFC 使用時はコントロールのウインドハンドル）

戻り値 ： なし（ゼロを返してください）

10.9.6. 右クリック通知（AJCBARN_RCLICK）

説　明 ： SHIFT か CTRL キーを押しながら右クリックしたことを通知します。
右クリック通知の可否については、AjcBarSetNtcRClk()を参照してください。
lParam で以下の右クリック情報を通知します。

```
typedef struct {
    int         x;              //  カーソルｘ座標
    int         y;              //  カーソルｙ座標
    BOOL        fShift;         //  SHIFT キー押下フラグ
    BOOL        fCtrl;          //  CTRL キー押下フラグ
} AJCBARRCLK, *PAJCBARRCLK;
```

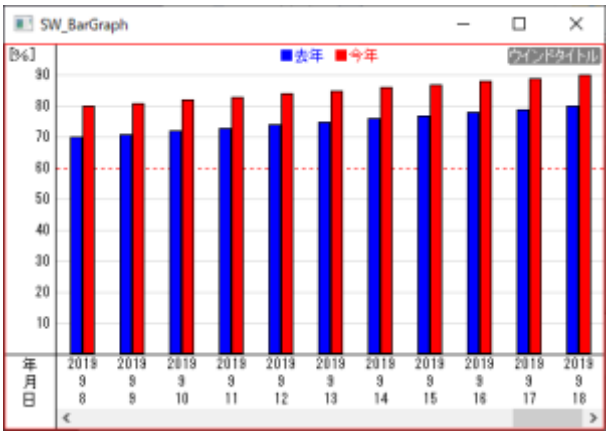
パラメタ： wParam - コントロールの識別ＩＤと通知メッセージコード（AJCBARN_RCLICK）
lParam - 右クリック情報へのポインタ（PAJCBARRCLK）（MFC 使用時はコントロールのウインドハンドル）

戻り値 ： なし（ゼロを返してください）

10.10. サンプルプログラム

10.10.1. SW_BarGraph（棒グラフの表示サンプル）

棒グラフ表示のサンプルプログラムを以下に示します。
このサンプルプログラムでは。単に、１～８０(赤)と１１～９０(青)までの２つの増分データをグラフに表示します。




```
 1 : //
 2 : //  SW_BarGraph.c
 3 : //
 4 : #include    <AjrCstXX.h>
 5 : #include    <math.h>
 6 : #include    <tchar.h>
 7 :
 8 : #define     IDC_BAR     5000
 9 :
10 : //-------------------------------------------------------------------------------------------------------------------//
11 : //  ワーク                                                                                                           //
12 : //-------------------------------------------------------------------------------------------------------------------//
13 : HINSTANCE       hInst;                      //  ＤＬＬインスタンスハンドル
14 : HWND            hWndBack;                   //  バックウインドハンドル（ダイアログと VT-100 の親ウインド）
15 : HWND            hWndBar;                    //  棒グラフコントロールのウインドハンドル
16 :
17 : //-------------------------------------------------------------------------------------------------------------------//
18 : //  内部サブ関数                                                                                                     //
19 : //-------------------------------------------------------------------------------------------------------------------//
20 : AJC_WNDPROC_DEF(Back);
21 :
22 : //===================================================================================================================//
23 : //                                                                                                                   //
24 : //  Ｗｉｎ Ｍａｉｎ                                                                                                  //
25 : //                                                                                                                   //
26 : //===================================================================================================================//
27 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
28 : {
29 :     MSG         msg;
30 :     WNDCLASS    wndclass;
31 :
32 :     hInst = hInstance;
33 :
34 :     //----- バックウインド生成 -------------------------//
35 :     wndclass.style          = 0;
36 :     wndclass.lpfnWndProc    = AJC_WNDPROC_NAME(Back);
37 :     wndclass.cbClsExtra     = 0;
38 :     wndclass.cbWndExtra     = 0;
39 :     wndclass.hInstance      = hInst;
40 :     wndclass.hIcon          = NULL;
41 :     wndclass.hCursor        = LoadCursor(NULL, IDC_ARROW);
42 :     wndclass.hbrBackground  = GetStockObject(WHITE_BRUSH);
43 :     wndclass.lpszMenuName   = NULL;
44 :     wndclass.lpszClassName  = TEXT("BarGraph01");
45 :     RegisterClass(&wndclass);
46 :
47 :     hWndBack = CreateWindow(TEXT("BarGraph01"),     // window class name
```

```
48 :                            TEXT("SW_BarGraph"),    // window caption
49 :                            WS_OVERLAPPEDWINDOW,    // window style
50 :                            0,                      // initial x position
51 :                            0,                      // initial y position
52 :                            500,                    // initial x size
53 :                            350,                    // initial y size
54 :                            NULL,                   // parent window handle
55 :                            NULL,                   // window menu handle
56 :                            hInst,                  // program instance handle
57 :                            NULL);                  // creation parameters
58 :
59 :    //----- ウインド表示 ------------------------------//
60 :    ShowWindow(hWndBack, iCmdShow);
61 :
62 :    //----- メッセージループ --------------------------//
63 :    while (GetMessage(&msg, NULL, 0, 0)) {
64 :        do {
65 :            TranslateMessage(&msg);
66 :            DispatchMessage (&msg);
67 :        } while (0);
68 :    }
69 :
70 :    return (int)msg.wParam ;
71 :}
72 ://=============================================================================================================//
73 ://                                                                                                             //
74 ://  バックウインド・プロシージャ                                                                               //
75 ://                                                                                                             //
76 ://=============================================================================================================//
77 ://----- WM_CREATE ---------------------------------------------------------------------------------------------//
78 :AJC_WNDPROC(Back, WM_CREATE         )
79 :{
80 :    //----- 棒グラフコントロール生成 ------------------------------//
81 :    hWndBar = CreateWindowEx(   WS_EX_ACCEPTFILES,
82 :                                TEXT("AjcCtrlBarGraph"),                    // window class name
83 :                                TEXT("P: I=2, BC=0x0000FF"),                // window caption
84 :                                WS_CHILD | WS_VISIBLE,                      // style
85 :                                0,                                          // initial x position
86 :                                0,                                          // initial y position
87 :                                0,                                          // initial x size
88 :                                0,                                          // initial y size
89 :                                hwnd,                                       // parent window handle
90 :                                (HMENU)IDC_BAR,                             // window menu handle
91 :                                hInst,                                      // program instance handle
92 :                                NULL);                                      // creation parameters
93 :    //----- タイトル設定 ------------------------------------------//
94 :    AjcBarSetTitleText(hWndBar, TEXT("ウインドタイトル"), -1, -1, NULL);
95 :    //----- チップテキスト設定 ------------------------------------//
96 :    AjcBarSetTipText      (hWndBar, TEXT("棒グラフ"));
97 :    AjcBarSetChkBoxTipText(hWndBar, 0, TEXT("データ０"));
98 :    AjcBarSetChkBoxTipText(hWndBar, 1, TEXT("データ１"));
99 :    //----- 横線設定 ----------------------------------------------//
100 :    AjcBarSetHLineAtt(hWndBar, 0, RGB(255, 0, 0), 1, AJCBAR_DOT);
101 :    AjcBarSetHLinePos(hWndBar, 0, 60.0);
102 :    AjcBarEnableHLine(hWndBar, 0, TRUE);
103 :    //----- プロファイルからプロパティ値読み出し ------------------//
104 :    AjcBarLoadProp(hWndBar, TEXT("BarGraph"), NULL);
105 :    //----- 最大データ数設定 --------------------------------------//
106 :    AjcBarSetBufSize(hWndBar, 100);
107 :    //----- スケール値表示域の幅設定 ------------------------------//
108 :    AjcBarSetScaleWidth(hWndBar, 40);
109 :    //----- 棒タイトル行数設定 ------------------------------------//
110 :    AjcBarSetTtlLines   (hWndBar, 3);
111 :    //----- 縦軸の単位名設定 --------------------------------------//
112 :    AjcBarSetVUnit(hWndBar, TEXT("[%]"));
113 :    //----- 棒タイトル名設定 --------------------------------------//
114 :    AjcBarSetBarTtl(hWndBar, TEXT("年¥n 月¥n 日"));
115 :    //----- 棒間の幅設定 ------------------------------------------//
116 :    AjcBarSetItemWidth(hWndBar, 40);
117 :    //----- テキスト描画 ------------------------------------------//
118 :    AjcBarTextOut(hWndBar, AJCTXO_CENTER, 3, TEXT("¥x1B[30m■去年  ¥x1B[31m■今年"));
119 :    //----- ウインド表示 ------------------------------------------//
```

```
120 :     ShowWindow(hWndBar , SW_SHOW);
121 :     //----- データ投与タイマ起動 ------------------------------------//
122 :     SetTimer(hwnd, 1, 200, NULL);
123 :
124 :     return 0;
125 : }
126 : //----- WM_DESTROY ---------------------------------------------------------------------------------------//
127 : AJC_WNDPROC(Back, WM_DESTROY        )
128 : {
129 :     //----- プロファイルへプロパティ値を記録 ------------------------//
130 :     AjcBarSaveProp(hWndBar, TEXT("BarGraph"));
131 :     //----- プログラム終了 ------------------------------------------//
132 :     PostQuitMessage(0);
133 :     return 0;
134 : }
135 : //----- WM_SIZE ------------------------------------------------------------------------------------------//
136 : AJC_WNDPROC(Back, WM_SIZE           )
137 : {
138 :     UI  width  = LOWORD(lParam);
139 :     UI  height = HIWORD(lParam);
140 :
141 :     //----- 棒グラフウインド移動 ----------------------------//
142 :     MoveWindow(hWndBar , 0, 0, width, height, FALSE);
143 :
144 :     return 0;
145 : }
146 : //----- WM_TIMER -----------------------------------------------------------------------------------------//
147 : AJC_WNDPROC(Back, WM_TIMER          )
148 : {
149 :     static  int yy = 2019, mm = 7, dd = 1;
150 :     static  UI  Count = 0;
151 :     int     data[2];
152 :     UT      txt[32];
153 :
154 :     if (++Count <= 80) {
155 :         data[0] = Count;
156 :         data[1] = Count + 10;
157 :         AjcSnPrintF(txt, sizeof txt, TEXT("%d¥n%d¥n%d"), yy, mm, dd);
158 :         AjcBarPutIntData(hWndBar, data, txt);
159 :         if (++dd > 31) {
160 :             dd = 1;
161 :             mm++;
162 :         }
163 :     }
164 :     else {
165 :         KillTimer(hwnd, 1);
166 :     }
167 :     return 0;
168 : }
169 : //----- 棒グラフコントロールからの通知 -------------------------------------------------------------------//
170 : AJC_WNDPROC(Back, IDC_BAR           )
171 : {
172 :     switch (HIWORD(wParam)) {
173 :         case AJCBARN_RANGE:         //  ●グラフレンジ通知      lParam : PAJCBAR_NTC_RANGE
174 :             break;
175 :
176 :         case AJCBARN_SCRPOS:        //  ●スクロール位置通知    lParam : クスロール位置
177 :             break;
178 :
179 :         case AJCBARN_RCLICK:        //  ●右クリック
180 :         {   PAJCBARRCLK p = (PAJCBARRCLK)lParam;
181 :             UT          txt[64] = {0};
182 :             AjcSnPrintF(txt, 64, TEXT("%s%s 右クリック発生(x = %d, y = %d)"), p->fShift ? TEXT("Shift+") : TEXT(""),
183 :                                                                               p->fCtrl  ? TEXT("Ctrl+" ) : TEXT(""),
184 :                                                                               p->x, p->y);
185 :             MessageBox(hwnd, txt, TEXT("SW_BarGraph"), MB_OK);
186 :             break;
187 :         }
188 :         case AJCBARN_DROPDIR:       //  ●ディレクトリドロップ
189 :         {   UT      path[MAX_PATH];
190 :             UT      txt[4096];
191 :             MAjcStrCpy(txt, AJCTSIZE(txt), TEXT("-- Dir dropped --¥n"));
```

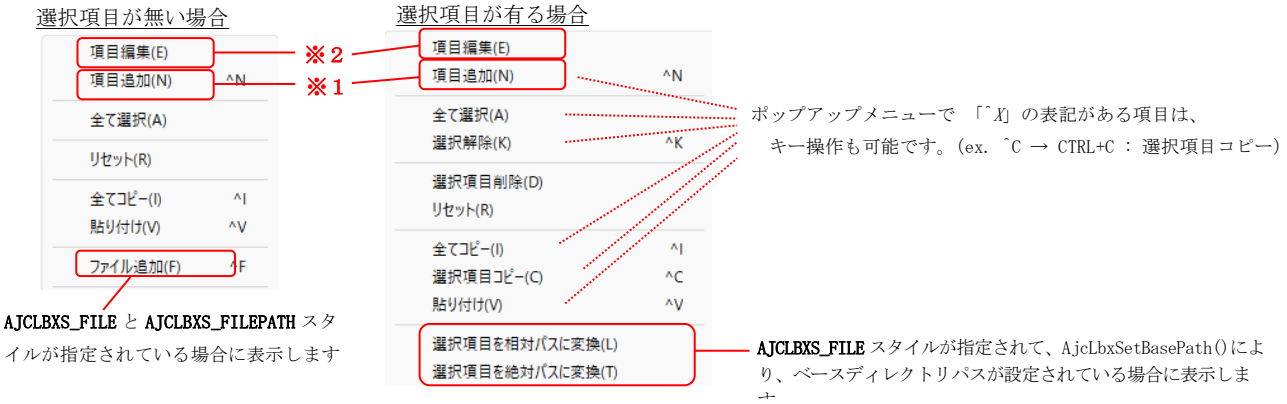
```
192 :             while (AjcBarGetDroppedDir(hWndBar, path)) {
193 :                 MAjcStrCat(txt, AJCTSIZE(txt), path);
194 :                 MAjcStrCat(txt, AJCTSIZE(txt), TEXT("¥n"));
195 :             }
196 :             MessageBox(hwnd, txt, TEXT("Dropped Dir"), MB_OK);
197 :             break;
198 :         }
199 :         case AJCBARN_DROPFILE:      //  ●ファイルドロップ
200 :         {   UT      path[MAX_PATH];
201 :             UT      txt[4096];
202 :             MAjcStrCpy(txt, AJCTSIZE(txt), TEXT("-- File dropped --¥n"));
203 :             while (AjcBarGetDroppedFile(hWndBar, path)) {
204 :                 MAjcStrCat(txt, AJCTSIZE(txt), path);
205 :                 MAjcStrCat(txt, AJCTSIZE(txt), TEXT("¥n"));
206 :             }
207 :             MessageBox(hwnd, txt, TEXT("Dropped File"), MB_OK);
208 :             break;
209 :         }
210 :     }
211 :     return TRUE;
212 : }
213 : //-----------------------------------------------------------------------------------------------------------------//
214 : AJC_WNDMAP_DEF(Back)
215 :     AJC_WNDMAP_MSG(Back, WM_CREATE     )
216 :     AJC_WNDMAP_MSG(Back, WM_DESTROY    )
217 :     AJC_WNDMAP_MSG(Back, WM_SIZE       )
218 :     AJC_WNDMAP_MSG(Back, WM_TIMER      )
219 :     AJC_WNDMAP_CMD(Back, IDC_BAR       )
220 : AJC_WNDMAP_END
```

11. 拡張リストボックス・コントロール（AjcCtrlListBox クラス）

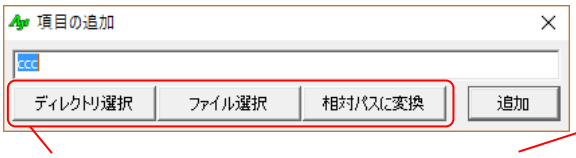
標準のリストボックスコントロール（ListBox クラス）をカスタム化したコントロールです。
標準のリストボックスコントロールは、拡張リストボックスの子ウインドとして配置されます。
標準のリストボックスコントロールに以下の機能を追加しました。

・リストボックス内に同一の項目（文字列）を追加できないようにしました。
・項目（文字列）の比較で、大文字と小文字を区別するか否かを選択できるようにしました。
・リストボックスの横幅より長い項目がある場合、横スクロールバーを表示するようにしました。
・右クリックで、以下のメニュー操作を追加しました。

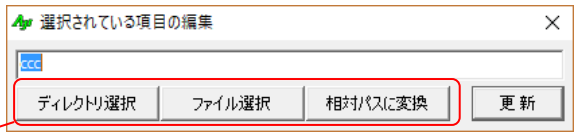
右クリックによるポップアップメニュー



※1（項目追加）



※2（右クリックしたマウスカーソル位置の項目編集）



AJCLBXS_FILE スタイルが指定されていない場合は非表示
AJCLBXS_FILE スタイルが指定されている場合・・
　ディレクトリ選択： AJCLBXS_DIRPATH スタイル未指定時はグレー表示
　ファイル選択　　： AJCLBXS_FILEPATH スタイル未指定時はグレー表示
　相対パスに変換　： AJCLBXS_DIRPATH, AJCLBXS_FILEPATH がいずれも指定されていないか
　　AjcLbxSetBasePath()によりベースディレクトリパスが設定されていない場合は、グレー表示

コントロールにフォーカスがある状態で、F1 キーを押した場合も ポップアップメニューを表示します。

最大文字列長（最大文字数）

リストボックス項目の最大文字数は１０２３です。(半角文字は１文字，全角文字は２文字で計算します)
文字列終端を含めた文字数を示す以下のシンボルが定義されています。

```
#define  AJCLBX_MAXSTL  1024
```

11.1. コントロールのスタイル

リストボックス・コントロールのスタイルは、以下のとおりです。

名称	ビット	値	内容	備考
AJCLBXS_RIGHT	15	0x8000	テキストを右揃えで表示します（未指定時は左揃え）	ウインド生成時 のみ有効**（※１）**
－	14	0x4000	－	
AJCLBXS_TRIM	13	0x2000	項目の追加／挿入時に、文字列の両端の空白を除去する	
AJCLBXS_FILE	12	0x1000	リスト項目としてファイル／フォルダのパス名を扱う場合に指定します。**（※2）**	
AJCLBXS_DIRTAIL	11	0x0800	ディレクトリ名の末尾に「¥」を付加する	ディレクトリ項目追加時
AJCLBXS_ADDITEMINDROP	10	0x0400	ディレクトリ／ファイルドロップ操作で、当該ディレクトリ／ファイルを リストボックス項目として追加する	AJCLBXS_ACCEPTDIRS／ AJCLBXS_ACCEPTFILES 指定要
AJCLBXS_ACCEPTDIRS	9	0x0200	ディレクトリのドラッグ＆ドロップを可能とします	AJCLBXN_DIRDROPPED イベント発生
AJCLBXS_ACCEPTFILES	8	0x0100	ファイルのドラッグ＆ドロップを可能とします	AJCLBXN_FILEDROPPED イベント発生
AJCLBXS_ACCEPTALL	9, 8	0x0300	(AJCLBXS_ACCEPTDIRS ｜ AJCLBXS_ACCEPTFILES)	
－	7	0x0080	－	
－	6	0x0040	－	
AJCLBXS_DIRPATH	5	0x0020	項目の追加／編集時にディレクトリパスを選択可能とします	右クリックメニューによる 項目追加 ／編集時
AJCLBXS_FILEPATH	4	0x0010	項目の追加／編集時にファイルパスを選択可能とします	
AJCLBXS_PATHALL	5, 4	0x0030	(AJCLBXS_DIRPATH ｜ AJCLBXS_FILEPATH)	
AJCLBXS_SORT_EXACT	3	0x0008	COMP_EXACT と SORT 指定時、英字大小を区別してソート **（※３）**	
AJCLBXS_COMP_EXACT	2	0x0004	英大文字と小文字を区別して文字列を比較します	
AJCLBXS_SINGLE	1	0x0002	単一選択モード（未指定時は、複数選択モード）	ウインド生成時 のみ有効**（※１）**
AJCLBXS_SORT	0	0x0001	アイテム追加時にソートします（未指定時はソートなし）	
AJCLBXS_STD_FILE AJCLBXS_STD_FILE_S	ファイルパスを扱うリストボックスの標準的なスタイル，以下のシンボルの合成値 ・AJCLBXS_FILE　・AJCLBXS_ADDITEMINDROP　・AJCLBXS_FILEPATH ・AJCLBXS_ACCEPTFILES　［ AJCLBXS_SORT ］			・左揃え ・複数選択 ・英字の大小を区別しない 末尾が「_S」の場合ソートあり
AJCLBXS_STD_DIR AJCLBXS_STD_DIR_S	ディレクトリパスを扱うリストボックスの標準的なスタイル，以下のシンボルの合成値 ・AJCLBXS_FILE　・AJCLBXS_ADDITEMINDROP　・AJCLBXS_DIRPATH ・AJCLBXS_DIRTAIL　・AJCLBXS_ACCEPTDIRS　［ AJCLBXS_SORT ］			
AJCLBXS_STD_FILEDIR AJCLBXS_STD_FILEDIR_S	ファイルとディレクトリを扱うリストボックスの標準的なスタイル，以下のシンボルの合成値 ・AJCLBXS_FILE　・AJCLBXS_ADDITEMINDROP　・AJCLBXS_PATHALL ・AJCLBXS_DIRTAIL　・AJCLBXS_ACCEPTALL　［ AJCLBXS_SORT ］			
AJCLBXS_STD_TEXT AJCLBXS_STD_TEXT_S	単純なテキストデータを扱うリストボックスの標準的なスタイル，以下のシンボルの合成値 ・AJCLBXS_COMP_EXACT　［AJCLBXS_SORT_EXACT ］　［ AJCLBXS_SORT ］			・左揃え ・複数選択 ・英字の大小を区別する 末尾が「_S」の場合ソートあり

※1：ダイアログボックス項目の場合は、リソースエディタの「Style」プロパティでの設定のみ有効となります。

※2：AJCLBXS_FILE スタイルは、以下の項目を有効にします。
- 項目追加ダイアログ中の「ディレクトリ選択」「ファイル選択」「相対パスに変換」ボタンを有効化
- AJCLBXS_FILEPATH 指定時に、ポップアップメニュー中の「ファイル追加」を有効化
- ベースパスが設定済で、選択項目がある場合、ポップアップメニュー中の「選択項目を相対パスに変換」「選択項目を絶対パスに変換」を有効化

AJCLBXS_FILE スタイルが指定されていない場合は、上記項目は全て非表示となります。

※3：AJCLBXS_COMP_EXACT と AJCLBXS_SOR_TEXACT 指定時は、単に文字列の大小でソートされる（ex. AAA < CCC < aaa < bbb）
AJCLBXS_COMP_EXACT だけ指定した場合は、アルファベット順にソートされる　（ex. AAA < aaa < bbb < CCC）

11.2. 機能別スタイル値

参考までに、機能選択に対するスタイルの値（リソースエディタの「Style」プロパティに設定する値）を以下に示します。

機能選択				スタイル・ビット（AJCLBXS_*XXXX*, Bit15 - 0）																スタイルの値（16 進）
				RIGHT	-	**TRIM**	FILE	**DIRTAIL**	ADDITEMINDROP	ACCEPTDIRS	ACCEPTFILES			DIRPATH	FILEPATH	SORT|EXACT	COMP|EXACT	SINGLE	SORT	
単一選択	ソートしない	FILE / DIR	FILE のみ	**0**	0	**0**	1	0	1	0	1	0	0	0	1	0	0	1	0	1512
			DIR のみ	**0**	0	**0**	1	1	1	1	0	0	0	1	0	0	0	1	0	1E22
			FILE & DIR	**0**	0	**0**	1	1	1	1	1	0	0	1	1	0	0	1	0	1F32
		通常のテキスト	英大小区別なし	**0**	0	**0**	0	0	0	0	0	0	0	0	1	0	0	1	0	0012
			英大小区別する	**0**	0	**0**	0	0	0	0	0	0	0	0	0	0	1	1	0	0006
	ソートする	FILE / DIR	FILE のみ	**0**	0	**0**	1	0	1	0	1	0	0	0	1	0	0	1	1	1513
			DIR のみ	**0**	0	**0**	1	1	1	1	0	0	0	1	0	0	0	1	1	1E23
			FILE & DIR	**0**	0	**0**	1	1	1	1	1	0	0	1	1	0	0	1	1	1F33
		通常のテキスト	英大小区別なし	**0**	0	**0**	0	0	0	0	0	0	0	0	1	0	0	1	1	0013
			英大小区別する	**0**	0	**0**	0	0	0	0	0	0	0	0	0	1	1	1	1	000F
複数選択	ソートしない	FILE / DIR	FILE のみ	**0**	0	**0**	1	0	1	0	1	0	0	0	1	0	0	0	0	1510
			DIR のみ	**0**	0	**0**	1	1	1	1	0	0	0	1	0	0	0	0	0	1E20
			FILE & DIR	**0**	0	**0**	1	1	1	1	1	0	0	1	1	0	0	0	0	1F30
		通常のテキスト	英大小区別なし	**0**	0	**0**	0	0	0	0	0	0	0	0	1	0	0	0	0	0010
			英大小区別する	**0**	0	**0**	0	0	0	0	0	0	0	0	0	0	1	0	0	0004
	ソートする	FILE / DIR	FILE のみ	**0**	0	**0**	1	0	1	0	1	0	0	0	1	0	0	0	1	1511
			DIR のみ	**0**	0	**0**	1	1	1	1	0	0	0	1	0	0	0	0	1	1E21
			FILE & DIR	**0**	0	**0**	1	1	1	1	1	0	0	1	1	0	0	0	1	1F31
		通常のテキスト	英大小区別なし	**0**	0	**0**	0	0	0	0	0	0	0	0	1	0	0	0	1	0011
			英大小区別する	**0**	0	**0**	0	0	0	0	0	0	0	0	0	1	1	0	1	000D
AJCLBXS_STD_FILE		標準FILE	ソートしない	**0**	0	**0**	1	0	1	0	1	0	0	0	1	0	0	0	0	1510
AJCLBXS_STD_DIR		標準DIR		**0**	0	**0**	1	1	1	1	0	0	0	1	0	0	0	0	0	1E20
AJCLBXS_STD_FILEDIR		標準FILE & DIR		**0**	0	**0**	1	1	1	1	1	0	0	1	1	0	0	0	0	1F30
AJCLBXS_STD_TEXT		標準テキスト		**0**	0	**0**	0	0	0	0	0	0	0	0	0	0	1	0	0	0004
AJCLBXS_STD_FILE_S		標準FILE	ソートする	**0**	0	**0**	1	0	1	0	1	0	0	0	1	0	0	0	1	1511
AJCLBXS_STD_DIR_S		標準DIR		**0**	0	**0**	1	1	1	1	0	0	0	1	0	0	0	0	1	1E21
AJCLBXS_STD_FILEDIR_S		標準FILE & DIR		**0**	0	**0**	1	1	1	1	1	0	0	1	1	0	0	0	1	1F31
AJCLBXS_STD_TEXT_S		標準テキスト		**0**	0	**0**	0	0	0	0	0	0	0	0	0	1	1	0	1	000D

この表では、AJCLBXS_RIGHT = **0**（左揃え）, AJCLBXS_TRIM = **0**（両端の空白は除去しない）固定としています。
また、FILE や DIR を扱う場合、ドロップ可能な設定としています。

11.3. リストボックスへ設定するスタイル

リストボックス（ListBox クラス）へ設定するスタイルは、以下のとおりです。

#	スタイル	備考
1	WS_CHILD	固定
2	WS_VISIBLE	
3	WS_HSCROLL	
4	LBS_STANDARD	
5	LBS_EXTENDEDSEL	
6	LBS_HASSTRINGS	

#	スタイル	備考
7	LBS_SORT	AJCLBXS_SORT 指定時
8	LBS_EXTENDEDSEL	AJCLBXS_SINGLE 未指定時
9	LBS_MULTIPLESEL	
10	WS_EX_ACCEPTFILES	AJCLBXS_ACCEPTDIRS / AJCLBXS_ACCEPTFILES 指定時
11	WS_EX_RIGHT	AJCLBXS_RIGHT 指定時

※リストボックスへ設定するスタイルを変更するには、AJCLBXN_QUERYSTYLE メッセージに応答し、lParam で指定されたスタイル情報を変更します。

11.4. リストボックス項目の永続化

リストボックスの全項目は、以下の関数により、全項目をプロファイルに記録することにより永続化することができます。

・AjcLbxLoadItems()　　・AjcLbxSaveItems()
・AjcLbxLoadPermInfo()　　・AjcLbxSavePermInfo[Ex]()

11.5. サポートＡＰＩ

リストボックス・コントロールのサポートＡＰＩ一覧を以下に示します。

#	関数名	内容	備考
1	AjcLbxGetAllItems	全リストボックス項目データの配列取得	
2	AjcLbxRelAllItems	取得した全リストボックス項目のメモリ解放	
3	AjcLbxGetSelectedItems	選択されている全リストボックス項目データの配列取得	
4	AjcLbxRelSelectedItems	取得した選択済リストボックス項目のメモリ解放	
5	AjcLbxLoadItems AjcLbxSaveItems	プロファイルから全リストボックス項目の読出し／書込み	
6	AjcLbxLoadPermInfo AjcLbxSavePermInfo[Ex]	プロファイルから全リストボックス項目の読出し／書込み	
7	AjcLbx{Set/Get}TipText	ツールチップの設定／取得	
8	AjcLbx{Set/Get}TipShowAlways	ツールチップ表示条件の設定／取得	
9	AjcLbxAddString	リストボックス項目の追加	ソートあり(AJCLBXS_SORT 指定時)
10	AjcLbxInsertString	リストボックス項目の挿入	ソートなし
11	AjcLbxDeleteString	リストボックス項目の削除	
12	AjcLbxFindString	リストボックス項目の検索	
13	AjcLbxGetCount	リストボックス項目の項目数取得	
14	AjcLbxGetCurSel	選択されているリストボックス項目のインデクス取得	
15	AjcLbxGetSel	リストボックス項目の選択状態取得	
16	AjcLbxGetSelCount	選択されているリストボックス項目の個数取得	複数選択モード時のみ
17	AjcLbxGetText	リストボックス項目の取得	
18	AjcLbxGetTextLen	リストボックス項目の文字列長取得	
19	AjcLbxResetContent	全リストボックス項目の消去	
20	AjcLbxSelectString	リストボックス項目の選択	文字列指定
21	AjcLbxSetCount	リストボックス項目数の設定	
22	AjcLbxSetCurSel	リストボックス項目の選択	インデクス指定
23	AjcLbxSetSel	リストボックス項目の選択／非選択状態設定	
24	AjcLbx{Set/Get}ItemData	リストボックス項目に関連付けられた数値の設定／取得	
25	AjcLbxSetBasePath	相対アドレス変換時のベースディレクトリパス設定	右クリックメニューで項目追加/編集時
26	AjcLbxSetNtcRClk	右クリック通知設定	
27	AjcLbxEnablePopupMenu	ポップアップメニューの許可／禁止	
28	AjcLbx{Sel/Exc}MenuItems	ポップアップメニュー項目の有効化／無効化	
29	AjcLbxSetFileFilter	ファイル選択時のフィルタ，デフォルト拡張子の設定	
30	AjcLbxGetDroppedFile	ドロップされたファイルのパス名を取得	
31	AjcLbxGetDroppedDir[Ex]	ドロップされたディレクトリのパス名を取得	
32	AjcLbxGetRemovedItem	削除された項目の文字列を取得	
33	AjcLbx{Set/Get}ItemHeight	リストボックス項目の高さ設定／取得	
34	AjcLbxGetLstHandle	リストボックスのハンドル取得	
35	AjcLbxGetLstID	リストボックスのコントロールＩＤ取得	
36	AjcLbxSetEvtCallBack	イベント通知用コールバック関数の設定	

11.5.1. 全リストボックス項目データの配列取得(AjcLbxGetAllItems)

形　式 ： PAJCLBXITEM　AjcLbxGetAllItems(HWND hwnd, UIP pCount);

引　数 ： hwnd　　- コントロールのウインドハンドル
pCount　- 全リストボックス項目数を格納するバッファのアドレス（不要時は NULL）

説　明 ： リストボックスの全項目データの配列を取得します。
取得した配列は、後で AjcLbxRelAllItems() により解放しなければなりません。

戻り値 ： ≠NULL - 成功（全リストボックス項目の配列の先頭アドレス）
＝NULL - 失敗

備　考 ： 戻り値は、「AJCLBXITEM」型配列へのポインタとなります。(型定義は下記参照)

```
typedef struct {
          UTP                  pStr;   // 項目（文字列）へのポインタ
          UX                   data;   // 項目に関連付けられた数値データ
} AJCLBXITEM, *PAJCLBXITEM;
typedef const AJCLBXITEM *PCAJCLBXITEM;
```

11.5.2. 取得した全リストボックス項目のメモリ解放(AjcLbxRelAllItems)

形　式 ： VO AjcLbxRelAllItems (VOP pAllItemBuf);

引　数 ： pAllItemBuf　- 解放する配列メモリのアドレス（AjcLbxGetAllItems()の戻り値）

説　明 ： AjcLbxGetAllItems()で生成した配列メモリを解放します。

戻り値 ： なし

11.5.3. 選択されているリストボックス項目データの配列取得(AjcLbxGetSelectedItems)

形　式 ： PAJCLBXITEM　AjcLbxGetSelectedItems (HWND hwnd, UIP pCount);

引　数 ： hwnd　　- コントロールのウインドハンドル
pCount　- 選択されているリストボックス項目数を格納するバッファのアドレス（不要時は NULL）

説　明 ： 選択されているリストボックス項目データの配列を取得します。
取得した配列は、後で AjcLbxRelSelectedItems () により解放しなければなりません。

戻り値 ： ≠NULL - 成功（選択されているリストボックス項目データの配列の先頭アドレス）
＝NULL - 失敗

備　考 ： 戻り値は、「AJCLBXITEM」型配列へのポインタとなります。(型定義は下記参照)

```
typedef struct {
          UTP                  pStr;   // 項目（文字列）へのポインタ
          UX                   data;   // 項目に関連付けられた数値データ
} AJCLBXITEM, *PAJCLBXITEM;
typedef const AJCLBXITEM *PCAJCLBXITEM;
```

11.5.4. 取得した選択済リストボックス項目群のメモリ解放(AjcLbxRelSelectedItems)

形　式 ： VO AjcLbxRelSelectedItems(VOP pSelectedItemBuf);

引　数 ： pSelectedItemBuf　- 解放する配列メモリのアドレス（AjcLbxGetSelectedItems ()の戻り値）

説　明 ： AjcLbxGetSelectedItems ()で生成した配列メモリを解放します。

戻り値 ： なし

11.5.5. プロファイルから全リストボックス項目の読出し／書込み(AjcLbx{Load/Save}Items)

形　式 ： BOOL AjcLbxLoadItems　(HWND hwnd, C_UTP pProfileSect, C_UTP pKeyPrefix); ---- 読み出し
BOOL AjcLbxSaveItems　(HWND hwnd, C_UTP pProfileSect, C_UTP pKeyPrefix); ---- 書き込み

引　数 ： hwnd　　- コントロールのウインドハンドル
pProfileSect - リスト項目をロードするプロファイルセクション名へのポインタ
pKeyPrefix　- リスト項目をロードするプロファイルキーの先頭部分の名称へのポインタ

説　明 ： AjcLbxLoadItems()は、プロファイルからリストボックス項目と各項目の選択状態を読み出してリストボックスへ設定します。
AjcLbxSaveItems()は、全リストボックス項目と各項目の選択状態をプロファイルへ書き出します。
pProfileSect には、プロファイルセクション名を指定します
プロファイルキーの名称は以下のとおりです。

・*KeyPrefix*_nnnnnn ： リスト項目　(nnnnnn はコントロールの ID)
・*KeyPrefix*_Cnt　　： リスト項目数

読み出し時は、最初にリストボックスをリセット（全項目削除）します。
リセット後に、プロファイルから全項目を読み出してリストボックスに設定します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

11.5.6. プロファイルから全リストボックス項目の読出し／書込み(AjcLbx{Load/Save}PermInfo[Ex])

形　式 ： BOOL AjcLbxLoadPermInfo　(HWND hwnd, C_UTP pProfileSect, C_UTP pKeyPrefix); --- 永続化情報の読み出し
BOOL AjcLbxSavePermInfo　(HWND hwnd); ---------- 永続化情報の書き込み
BOOL AjcLbxSavePermInfoEx(HWND hwnd, C_UTP pProfileSect, C_UTP pKeyPrefix); ---　〃

引　数 ： hwnd　　- コントロールのウインドハンドル
pSect　　- リスト項目をロードするプロファイルセクション名へのポインタ
pKeyPrefix - リスト項目をロードするプロファイルキーの先頭部分の名称へのポインタ

説　明 ： プロファイルからリストボックス項目と各項目の選択状態の読み出し、あるいは、書き込みを行います。

プロファイルキーの名称は以下のとおりです。

・*KeyPrefix_nnnnnn* ： リスト項目　(*nnnnnn* はコントロールの ID)
・*KeyPrefix*_Cnt　　： リスト項目数

AjcLbxSavePermInfo()での読み出しでは、最初にリストボックスをリセット（全項目削除）し、その後に、プロファイルから全項目を読み出してリストボックスに設定します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

11.5.7. ツールチップの設定／取得(AjcLbx{Set/Get}TipText)

形　式 ： BOOL AjcLbxSetTipText(HWND hwnd, C_UTP pTxt); -------- ツールチップ・テキスト設定
BOOL AjcLbxGetTipText(HWND hwnd, UTP pBuf, UI lBuf); - ツールチップ・テキスト取得

引　数 ： hwnd　　- コントロールのウインドハンドル
pTxt　　- ツールチップテキスト（文字列）のアドレス
pBuf　　- ツールチップテキスト（文字列）を格納するバッファのアドレス
lBuf　　- ツールチップテキスト（文字列）を格納するバッファの文字数

説　明 ： リストボックスにカーソルを置いた際に表示されるツールチップ文字列を設定／取得します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

11.5.8. ツールチップ表示条件の設定／取得(AjcLbx{Set|Get}TipShowAlways)

形　式 ： BOOL AjcLbxSetTipShowAlways(HWND hwnd, BOOL fShowAlways); --- 設定
BOOL AjcLbxGetTipShowAlways(HWND hwnd); ---------- 取得

引　数 ： hwnd - コントロールのウインドハンドル
fShowAlways - ツールチップ表示条件（TRUE:非アクティブ時も表示, FALSE:非アクティブ時は非表示）

説　明 ： ツールチップの表示条件（自プログラムが非アクティブ時の表示／非表示）を設定／取得します。を設定／取得します。

戻り値 ： 設定時：TRUE - 成功　　取得時：ツールチップ表示条件
FALSE - 失敗

11.5.9. リストボックス項目の追加(AjcLbxAddString)

形　式 ： int AjcLbxAddString(HWND hwnd, C_UTP pStr);

引　数 ： hwnd - コントロールのウインドハンドル
pStr - リストボックスに追加する項目（文字列）のアドレス

説　明 ： リストボックスに項目（文字列）を追加します。
「AJCLBXS_SORT」スタイルが設定されていない場合は、リストの末尾に挿入されます。
「AJCLBXS_SORT」スタイルが設定されている場合は、リストが並べ替えられます。（ソート方法につては「備考」を参照）
既に同一文字列の項目が存在する場合は、リストボックスに項目（文字列）は追加されません。
文字列の比較方法は「AJCLBXS_COMP_EXACT」スタイルで指定します。

戻り値 ： ≧０ - 追加したリストボックス項目のインデクス（０～）
＜０ - 失敗

備　考 ： 「AJCLBXS_SORT」と「AJCLBXS_COMP_EXACT」が指定されている場合、「AJCLBXS_SORT_EXACT」スタイルの指定により、以下のようにソート方法が異なります。

AJCLBXS_SORT_EXACT	ソート方法	例
指定無し	英大小文字を区別しないでソートしますが、同一文字列の場合は英大小文字を区別してソートします	bb BBB bbb
指定あり	英大小文字を区別し、単純に文字列の昇順にソートします。	BBB bb bbb

11.5.10. リストボックス項目の挿入(AjcLbxInsertString)

形　式 ： int AjcLbxInsertString(HWND hwnd, UI ix, C_UTP pStr);

引　数 ： hwnd - コントロールのウインドハンドル
ix - 項目を挿入する位置のインデクス（０～）
pStr - リストボックスに追挿入する項目（文字列）のアドレス

説　明 ： リストボックスに項目（文字列）を挿入します。
「AJCLBXS_SORT」スタイルの設定にかかわらず、所定の位置に挿入され、並べ替えは行われません。
既に同一文字列の項目が存在する場合は、リストボックスに項目（文字列）は追加されません。
文字列の比較方法は「AJCLBXS_COMP_EXACT」スタイルで指定します。

戻り値 ： ≧０ – 挿入したリストボックス項目のインデクス（０～）
＜０ - 失敗

11.5.11. リストボックス項目の削除(AjcLbxDeleteString)

形　式　：　BOOL AjcLbxDeleteString (HWND hwnd, UI ix);

引　数　：　hwnd　　- コントロールのウインドハンドル
ix　　- 削除する項目のインデクス（０～）

説　明　：　指定したインデクスのリストボックス項目を削除します。

戻り値　：　TRUE　- 成功
FALSE - 失敗

11.5.12. リストボックス項目の検索(AjcLbxFindString)

形　式　：　int AjcLbxFindString (HWND hwnd, UI ix, C_UTP pStr);

引　数　：　hwnd　　- コントロールのウインドハンドル
ix　　- 検索を開始するインデクス（-1 の場合は先頭から検索）
pStr　　- 検索する文字列のアドレス

説　明　：　リストボックスの項目（文字列）を検索します。
文字列の比較方法は「AJCLBXS_COMP_EXACT」スタイルで指定します。

戻り値　：　≧０：見つかったリストボックス項目のインデクス（０～）
＜０：指定文字列が見つからない／エラー

11.5.13. リストボックスの項目数取得(AjcLbxGetCount)

形　式　：　int AjcLbxGetCount (HWND hwnd);

引　数　：　hwnd　　- コントロールのウインドハンドル

説　明　：　リストボックスに設定されている項目の個数を取得します。

戻り値　：　≧０：リストボックスの項目数（０～）
＜０：エラー

11.5.14. 選択されているリストボックス項目のインデクス取得(AjcLbxGetCurSel)

形　式　：　int AjcLbxGetCurSel (HWND hwnd);

引　数　：　hwnd　　- コントロールのウインドハンドル

説　明　：　リストボックスの選択されている項目のインデクスを取得します。
複数項目選択可能なモードである場合は、最初の選択項目のインデクスを返します。

戻り値　：　≧０：選択されている項目のインデクス
＜０：エラー

11.5.15. リストボックス項目の選択状態取得(AjcLbxGetSel)

形　式　：　BOOL AjcLbxGetSel (HWND hwnd, UI ix);

引　数　：　hwnd　　- コントロールのウインドハンドル
ix　　- 選択状態を取得する項目のインデクス（０～）

説　明　：　ix で指定された項目の選択状態を返します。

戻り値　：　TRUE　：指定項目は選択状態である
FALSE：指定項目は選択状態でない

11.5.16. 選択されているリストボックス項目の個数取得(AjcLbxGetSelCount)

形　式 ： int AjcLbxGetSelCount (HWND hwnd);

引　数 ： hwnd　　- コントロールのウインドハンドル

説　明 ： リストボックス中で選択されている項目の個数を取得します。
「AJCLBXS_SINGLE」スタイルが設定されている場合は、戻り値＝０／１となります。

戻り値 ： ≧０：選択されているリストボックス項目の個数
＜０：単一選択モード

11.5.17. リストボックス項目の取得(AjcLbxGetText)

形　式 ： BOOL AjcLbxGetText (HWND hwnd, UI ix, UTP pBuf, UI lBuf);

引　数 ： hwnd　　- コントロールのウインドハンドル
ix　　- 取得する項目のインデクス（０～）
pBuf　　- 取得する項目（文字列）を格納するバッファのアドレス
lBuf　　- 取得する項目（文字列）を格納するバッファの文字数

説　明 ： 指定したインデクスのリストボックス項目（文字列）を取得します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

11.5.18. リストボックス項目の文字列長取得(AjcLbxGetTextLen)

形　式 ： int AjcLbxGetTextLen (HWND hwnd, UI ix);

引　数 ： hwnd　　- コントロールのウインドハンドル
ix　　- 文字数を取得する項目のインデクス（０～）

説　明 ： リストボックス項目（文字列）のバイト数／文字数を取得します。

戻り値 ： ≧０ - 成功（リストボックス項目のバイト数／文字数，文字列終端('¥0')は含まない）
＜０ - 失敗

11.5.19. 全リストボックス項目の消去(AjcLbxResetContent)

形　式 ： BOOL AjcLbxResetContent (HWND hwnd);

引　数 ： hwnd　　- コントロールのウインドハンドル

説　明 ： リストボックスの全項目を削除します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

11.5.20. リストボックス項目の選択（文字列指定）(AjcLbxSelectString)

形　式 ： int AjcLbxSelectString (HWND hwnd, UI ix, C_UTP pStr);

引　数 ： hwnd　　- コントロールのウインドハンドル
ix　　- 選択開始インデクス（－１の場合は先頭から検索）
pStr　　- 選択する文字列のアドレス

説　明 ： リストボックス中の項目（文字列）を検索し、一致する項目を選択状態にします。
文字列の比較方法は「AJCLBXS_COMP_EXACT」スタイルで指定します。

戻り値 ： ≧０：見つかったリストボックス項目のインデクス（０～）
＜０：指定文字列が見つからない

11.5.21. リストボックス項目数の設定(AjcLbxSetCount)

形　式 ： BOOL AjcLbxSetCount (HWND hwnd, UI count);

引　数 ： hwnd　　- コントロールのウインドハンドル
count　　- 設定するリストボックス項目数

説　明 ： リストボックスの新しい項目数を指定します。(LB_SETCOUNT メッセージ発行)

戻り値 ： TRUE　- 成功
FALSE - 失敗

11.5.22. リストボックス項目の選択（インデクス指定）(AjcLbxSetCurSel)

形　式 ： BOOL AjcLbxSetCurSel (HWND hwnd, UI ix);

引　数 ： hwnd　　- コントロールのウインドハンドル
ix　　- 設定するリストボックス項目のインデクス（０～）

説　明 ： リストボックスの項目を選択状態にします。

戻り値 ： TRUE　- 成功
FALSE - 失敗

11.5.23. リストボックス項目の選択／非選択状態設定(AjcLbxSetSel)

形　式 ： BOOL AjcLbxSetSel (HWND hwnd, UI ix, BOOL fSelect);

引　数 ： hwnd　　- コントロールのウインドハンドル
ix　　- 設定するリストボックス項目のインデクス（０～：単一項目　or　－１：全項目）
fSelect　- TRUE：選択状態にする，FALSE:非選択状態にする

説　明 ： リストボックス項目を選択状態／非選択状態にします。
単一選択モード（AJCLBXS_SINGLE）で、fSelect=FALSE の場合、ix の指定は無視されて全項目が非選択状態となります。
また、fSelect=TRUE, ix=-1（全項目を選択状態にする）は指定できません。
複数選択モードで、ix=-1 を指定した場合は、全項目の選択／非選択となります。

戻り値 ： TRUE　- 成功
FALSE - 失敗

11.5.24. リストボックス項目に関連付けた数値の設定／取得(AjcLbx{Set/Get}ItemData)

形　式 ： BOOL AjcLbxSetItemData (HWND hwnd, UI ix, UX data); - リストボックス項目に関連付けた数値の設定
UX AjcLbxGetItemData (HWND hwnd, UI ix); ---------- リストボックス項目に関連付けた数値の取得

引　数 ： hwnd　　- コントロールのウインドハンドル
ix　　- 設定するリストボックス項目のインデクス（０～）
data　　- 項目に関連付ける数値データ

説　明 ： リストボックス項目関連付けする数値を設定／取得ます。

戻り値 ： 設定時：TRUE　- 成功　　取得時：関連付けられた数値データ（エラー時は -1 を返す）
FALSE - 失敗

11.5.25. 相対アドレス変換時のベースディレクトリパス設定(AjcLbxSetBasePath)

形　式 ： BOOL AjcLbxSetBasePath (HWND hwnd, C_UTP pBasePath);

引　数 ： hwnd　　- コントロールのウインドハンドル
pBasePath - 相対パス変換時のベースディレクトリパス

説　明 ： 右クリック・メニューによる項目追加／変換時に「相対パスに変換」する場合の、ベースディレクトリパスを設定します。

戻り値 ： TRUE - 成功
FALSE - 失敗

11.5.26. 右クリック通知設定 (AjcLbxSetNtcRClk)

形　式 ： BOOL AjcLbxSetNtcRClk(HWND hwnd, BOOL fNtcRClk, UI MsgRBDown, UI MsgRBUp);

引　数 ： hwnd　　- コントロールのウインドハンドル
fNtcRClk　- 右ボタンの DOWN/UP 通知フラグ（TRUE:通知する, FALSE:通知しない）
MsgRBDown　- 右ボタン押下時の通知メッセージコード　（0 の場合は非通知）
MsgRBUp　- 右ボタン離され時の通知メッセージコード（0 の場合は非通知）

説　明 ： コントロールを右クリックした場合に、当該操作を親ウインドへ通知するか否かを設定します。
MsgRBDown, MsgRBUp は、WM_USER+100 以降, WM_APP+500 以降か、RegisterWindowMessage()で取得したコードを指定します。
各引数と、右クリック通知動作は以下のとおりです。

引数			Shift/ Ctrl	メッセージ	wParam	備考
fNtcRClk	MsgRBDown	MsgRBUp				
FALSE (default	－	－	未押下	WM_COMMAND	－（非通知）	ポップアップメニュー表示
			押下		ID + AJCLBXN_RCLICK	
TRUE	いずれかが 0 以外		－	MsgRBDown(押下時)	WM_RBUTTONDOWN の wParam	MsgRBDown ＝ 0 の場合は非通知
				MsgRBUp（離し時）	WM_RBUTTONUP　の wParam	MsgRBUp　＝ 0 の場合は非通知
	0	0	－		－（非通知）	－

右クリックの通知を禁止するには、fNtcRClk=TRUE, MsgRBDown=0, MsgRBUp=0 とします。

戻り値 ： TRUE - 成功
FALSE - 失敗

11.5.27. ポップアップメニューの許可／禁止 (AjcLbxEnablePopupMenu)

形　式 ： BOOL AjcLbxEnablePopupMenu (HWND hwnd, BOOL fEnable);

引　数 ： hwnd　　- コントロールのウインドハンドル
fEnable　- ポップアップメニューの許可(TRUE)／禁止(FALSE)

説　明 ： 右クリックによるポップアップメニューを許可／禁止します。

戻り値 ： TRUE - 成功
FALSE - 失敗

備　考 ： 「AjcLbxSetNtcRClk(hwnd, !fEnable, WM_RBUTTONDOWN, WM_RBUTTONUP);」を実行します。

11.5.28. ポップアップメニュー項目の有効化／無効化 (AjcVth{Sel/Exc}MenuItems)

形　式 ： BOOL　AjcLbxSelMenuItems(HWND hwnd, UI sel)； - 有効化
BOOL　AjcLbxExcMenuItems(HWND hwnd, UI exc)； - 無効化

引　数 ： hwnd　　- コントロールのウインドハンドル
sel　　- 有効とするメニュー項目群（0　　：右クリックによるポップアップメニュー禁止）
exc　　- 無効とするメニュー項目群(AJCLBXMM_ALL： 右クリックによるポップアップメニュー禁止)

説　明 ： 右クリックによるポップアップメニュー項目を有効化／無効化します。
sel/exc には以下のシンボルの合成値を指定します。

#	シンボル	メニュー項目
1	AJCLBXMM_EDTITEM	項目編集
2	AJCLBXMM_ADDITEM	項目追加
3	AJCLBXMM_SELALL	全て選択
4	AJCLBXMM_DESELALL	選択解除
5	AJCLBXMM_DELETE	選択項目削除
6	AJCLBXMM_CLEAR	リセット
7	AJCLBXMM_COPYALL	全てコピー
8	AJCLBXMM_COPYSEL	選択項目コピー
9	AJCLBXMM_PASTE	貼り付け
10	AJCLBXMM_ADDFILES	ファイル追加
11	AJCLBXMM_TO_RELATIVE	相対パスに変更
12	AJCLBXMM_TO_ABSOLUTE	絶対パスに変更
13	AJCLBXMM_ALL	全項目

ex.　AjcLbxExcMenuItems(hwnd, AJCOPT2(AJCVTHMM_, DELETE, CLEAR)); // 選択項目削除、リセットをメニューから除外

戻り値 ： TRUE　- 成功
FALSE - 失敗

11.5.29. ファイル選択時のフィルタ，デフォルト拡張子の設定 (AjcLbxSetFileFilter)

形　式 ： BOOL　AjcLbxSetFileFilter(HWND hwnd, C_UTP pFilter, C_UTP pDefExt);

引　数 ： hwnd　　- コントロールのウインドハンドル
pFilter　　- ファイル選択時のフィルタ文字列へのポインタ（不要時は NULL）
MsgRBDown　- ファイル選択時のデフォルト拡張子へのポインタ（不要時は NULL）

説　明 ： 右クリックにより、リストボックスへファイルを追加する場合のフィルタ、デフォルト拡張子を設定します。
フィルタは、「項目名／ワイルドカード」のペアで指定します。(ex. “TextFile(*.txt)/*.txt/AllFiles(*.*)/*.*”)
デフォルト・ファイル拡張子はピリオドを除いて指定します。(ex. “txt”)

戻り値 ： TRUE　- 成功
FALSE - 失敗

11.5.30. ドロップされたファイル名取得(AjcLbxGetDroppedFile)

形　式 ： BOOL　AjcLbxGetDroppedFile (HWND hwnd, UT buf[MAX_PATH]);

引　数 ： hwnd　　- コントロールのウインドハンドル
buf　　- ファイルのパス名を格納するバッファのアドレス

説　明 ： ドラッグ＆ドロップされたファイルのパス名を取得します。
本関数は、通知メッセージ（AJCLBXN_FILEDROPPED）が通知された場合、ドロップされたファイルのパス名を取得するために実行します。
本ＡＰＩにより全てのドロップされたファイル名を取得済の場合は、FALSE を返します。

戻り値 ： TRUE　- ファイルのパス名をバッファに格納した
FALSE - ドロップされたファイル名なし（ドロップしたファイル数を超えて実行された）

11.5.31. ドロップされたディレクトリ名取得(AjcLbxGetDroppedDir[Ex])

形　式 ： BOOL　AjcLbxGetDroppedDir　(HWND hwnd, UT buf[MAX_PATH]);
BOOL　AjcLbxGetDroppedDirEx (HWND hwnd, UT buf[MAX_PATH], BOOL fTailIsDelimiter);

引　数 ： hwnd　　- コントロールのウインドハンドル
buf　　- ファイルのパス名を格納するバッファのアドレス
fTailIsDelimiter - TRUE　：ディレクトリパス名の末尾に「¥」を付加する
FALSE ：ディレクトリパス名の末尾に「¥」を付加しない

説　明 ： ドラッグ＆ドロップされたディレクトリのパス名を取得します。
本関数は、通知メッセージ（AJCLBXN_DIRDROPPED）が通知された場合、ドロップされたディレクトリのパス名を取得するために実行します。
本ＡＰＩにより全てのドロップされたディレクトリ名を取得済の場合は、FALSE を返します。

戻り値 ： TRUE　- ディレクトリのパス名をバッファに格納した
FALSE - ドロップされたディレクトリ名なし（ドロップしたディレクトリ数を超えて実行された）

11.5.32. 削除された項目の文字列取得(AjcLbxGetRemovedItem)

形　式 ： BOOL　AjcLbxGetRemovedItem (HWND hwnd, UTP pBuf, UI lBuf);

引　数 ： hwnd　　- コントロールのウインドハンドル
pBuf　　- 削除された項目の文字列格納するバッファのアドレス
lBuf　　- 削除された項目の文字列格納するバッファの文字数

説　明 ： 削除されたリストボックス項目の文字列を取得します
本関数は、通知メッセージ（AJCLBXN_ REMOVED）が通知された場合、削除した項目の文字列を取得するために実行します。
複数の項目を削除した場合は、（AJCLBXN_ REMOVED の）lParam で通知された削除項目数だけ本関数を実行します。

戻り値 ： TRUE　- 削除した項目の文字列をバッファに格納した
FALSE - 削除した項目なし（削除した項目数を超えて実行された）

11.5.33. リストボックス項目の高さ設定／取得 (AjcLbx{Set/Get}Height)

形　式 ： BOOL AjcLbxSetItemHeight (HWND hwnd , UI ix, UI height); --- 設定
UI　　AjcLbxGetItemHeight(HWND hwnd, UI ix); ---------- 取得

引　数 ： hwnd　　- コントロールのウインドハンドル
ix　　- リスト項目のインデクス（０～）
height　　- 設定する項目の高さ（ピクセル数）

説　明 ： オーナー描画する際の、リストボックス項目の高さを設定／取得します。
height = 0 とした場合は、リストボックス・フォントの高さ＋２を設定します。

LBS_OWNERDRAWFIXED スタイルの場合、ix=0 とします。(全項目の高さが設定されます)
LBS_OWNERDRAWVARIABLE スタイルの場合、ix=設定する項目のインデクスとします。(各項目に個別の高さを設定する)

戻り値 ： AjcLbxSetItemHeight()　　AjcLbxGetItemHeight()
TRUE　- 成功　　項目の高さ (-1:エラー)
FALSE – 失敗

11.5.34. リストボックスのハンドル取得 (AjcLbxGetLstHandle)

形　式 ： HWND AjcLbxGetLstHandle (HWND hwnd);

引　数 ： hwnd　　- コントロールのウインドハンドル

説　明 ： 標準のリストボックスのウインドハンドルを取得します。

戻り値 ： ≠NULL　- 成功（標準のリストボックスのウインドハンドル）
＝NULL　– 失敗

備　考 ： 標準のリストボックスハンドルから、拡張リストボックスのハンドルを取得するには、以下のコードを実行します。

```
hLbx = GetParent(hListBox);          // hLbx:拡張リストボックス, hListBox:標準のリストボックス
```

11.5.35. リストボックスのコントロールＩＤ取得 (AjcLbxGetLstHandle)

形　式 ： UI AjcLbxGetLstID (HWND hwnd);

引　数 ： hwnd　　- コントロールのウインドハンドル

説　明 ： 標準のリストボックスのウインドハンドルを取得します。

戻り値 ： ≠０　- 成功（標準のリストボックスのコントロールＩＤ）
＝０　– 失敗

11.5.36. ベント通知用コールバック関数の設定(AjcLbxSetEvtCallBack)

形　式　：　BOOL AjcLbxSetEvtCallBack (HWND hwnd, UX cbp, VO (CALLBACK *cbNtcEvt)(HWND hCtrl, UX wParam, UX lParam, UX cbp));

引　数　：　hwnd　　- コントロールのウインドハンドル
cbp　　- コールバックパラメタ
cbNtcEvt　- イベント通知用コールバック関数（NULL:コールバック関数を無効化）

説　明　：　イベント通知用のコールバック関数を設定します。
通常、イベントは WM_COMMAND メッセージで通知されますが、コールバック関数を設定することにより、コールバック関数の呼び出しにより通知されるようになります。

戻り値　：　TRUE　- 成功
FALSE - 失敗

コールバック：　コールバック関数の仕様は、以下のとおりです。

cbNtcEvt（イベント通知）

形　式　：　VO CALLBACK *cbNtcEvt*(HWND hCtrl, UX wParam, UX lParam, UX cbp);

引　数　：　hCtrl　- コントロールのウインドハンドル
wPatram - コントロール識別ＩＤ，通知コード（WM_COMMAND による通知と同じ）
lPatram - イベントパラメタ　　（WM_COMMAND による通知と同じ）
cbp　　- コールバックパラメタ

説　明　：各種イベントを通知します。（詳細は「コントロールの通知メッセージ」を参照）

戻り値　：なし

11.6. 通知情報の取得ＡＰＩ

コントロールからの通知メッセージ（WM_COMMAND）に伴うパラメタを取得するＡＰＩ群です。
「AjcSetCmdWithHdl(TRUE);」を実行した場合、各通知メッセージ（WM_COMMAND）の lParam は通知内容に伴うパラメタは通知されず、lParam＝コントロールのウインドハンドルとなります。
この場合、通知内容に伴うパラメタは以下のＡＰＩで取得します。

#	関 数 名	内容	対応する通知	
1	AjcLbxGetListStyle	リストボックスへ設定するスタイル取得	AJCLBXN_QUERYSTYLE	リストボックス設定スタイル問い合わせ
2	AjcLbxGetNtcFiles	ドロップしたファイル数の取得	AJCLBXN_FILEDROPPED	ファイルドロップ通知
3	AjcLbxGetNtcDirs	ドロップしたディレクトリ数の取得	AJCLBXN_DIRDROPPED	ディレクトリドロップ通知
4	AjcLbxGetNtcDelItems	リストボックスから削除した項目数	AJCLBXN_REMOVED	項目削除通知
5	AjcLbxGetNtcRClk	右クリック情報の取得	AJCLBXN_RCLICK	右クリック通知

11.6.1. リストボックスへ設定するスタイル取得（AjcLbxGetListStyle）

形　式　：　PAJCLBXLSTSTY AjcLbxGetListStyle (HHWND hwnd);

引　数　：　hwnd　　- コントロールのウインドハンドル

説　明　：　リストボックスへ設定しようとしているスタイル情報を取得します。

戻り値　：　リストボックスへ設定しようとしているスタイル情報へのポインタ

11.6.2. ドロップしたファイル数の取得（AjcLbxGetNtcFiles）

形　式　：　UI AjcLbxGetNtcFiles (HHWND hwnd);

引　数　：　hwnd　　- コントロールのウインドハンドル

説　明　：　ファイルドロップ通知（AJCLBXN_FILEDROPPED）時のドロップし、追加されたファイルの個数を取得します。

戻り値　：　ドロップし、追加されたファイルの個数

11.6.3. ドロップしたディレクトリ数の取得（AjcLbxGetNtcDirs）

形　式　：　UI　AjcLbxGetNtcDirs (HHWND hwnd);

引　数　：　hwnd　　- コントロールのウインドハンドル

説　明　：　ディレクトリドロップ通知（AJCLBXN_DIRDROPPED）時の、ドロップし、追加されたディレクトリの個数を取得します。

戻り値　：　ドロップし、追加されたされたディレクトリの個数

11.6.4. 削除した項目数の取得（AjcLbxGetNtcRemovedItems）

形　式　：　UI　AjcLbxGetNtcRemovedItems (HHWND hwnd);

引　数　：　hwnd　　- コントロールのウインドハンドル

説　明　：　削除通知（AJCLBXN_REMOVED）時の、リストボックスから削除した項目の個数を取得します。

戻り値　：　リストボックスから削除した項目の個数

11.6.5. 右クリック情報の取得（AjcLbxGetNtcRClk）

形　式　：　PAJCLBXRCLK AjcLbxGetNtcRClk (HHWND hwnd);

引　数　：　hwnd　　- コントロールのウインドハンドル

説　明　：　右クリック通知（AJCLBXN_RCLICK）時の、右クリック情報を取得します。

戻り値　：　右クリック情報（AJCLBXRCLK）へのポインタ

11.7. 通知メッセージ

リストボックス・カスタムコントロールからの通知メッセージは、WM_COMMAND メッセージにより通知されます。
WM_COMMAND メッセージの wParam には以下の情報が設定されます。

・LOWORD(wParam) - コントロールの識別ＩＤ
・HIWORD(wParam) - 通知メッセージコード
・lParam　　- コントロールのウインドハンドル

通知メッセージコード（HIWORD(wParam)）の一覧を以下に示します。

#	メッセージコード	内容	備考
1	AJCLBXN_LISTSTYLE	リストボックスの設定スタイル問い合わせ	ウインド生成時
2	AJCLBXN_DBLCLK	ダブルクリック通知	
3	AJCLBXN_SELCANCEL	選択キャンセル通知	
4	AJCLBXN_SELCHANGE	選択変更通知	
5	AJCLBXN_SETFOCUS	フォーカス取得通知	
6	AJCLBXN_KILLFOCUS	フォーカス喪失通知	
7	AJCLBXN_RCLICK	右クリック通知	SHIFT/CTRL + 右クリック時
8	AJCLBXN_REMOVED	項目削除通知	
9	AJCLBXN_DIRDROPPED	ディレクトリドロップ通知	
10	AJCLBXN_FILEDROPPED	ファイルドロップ通知	
11	AJCLBXN_ERRSPACE	メモリ不足通知	

11.7.1. リストボックスの設定スタイル問い合わせ（AJCLBXN_LISTSTYLE）

説　明　：　リストボックス（ListBox クラス）への設定スタイルを通知します。
このメッセージは。リストボックス・ウインド生成時に１度だけ発行されます。
必要ならば lParam でポイントするスタイル情報を変更できます。
lParam でポイントするスタイル情報は、以下のとおり。

```
typedef struct {
    UI    sty;    // スタイル
    UI    exs;    // 拡張スタイル
} AJCLBXLSTSTY, *PAJCLBXLSTSTY;
typedef const AJCLBXLSTSTY *PCAJCLBXLSTSTY;
```

パラメタ：　wParam - コントロールの識別ＩＤと通知メッセージコード（AJCLBXN_QUERYSTYLE）
lParam - リストボックスへ設定するスタイル情報へのポインタ(PAJCLBXLSTSTY)

戻り値　：　なし（ゼロを返してください）

11.7.2. ダブルクリック通知（AJCLBXN_DBLCLK）

説　明　：　*ダブルクリックしたことを通知します。*

パラメタ：　*wParam - コントロールの識別ＩＤと通知メッセージコード（AJCLBXN_DBLCLK）*
lParam – コントロールのウインドハンドル

戻り値　：　*なし（ゼロを返してください）*

11.7.3. 選択キャンセル通知（AJCLBXN_SELCANCEL）

説　明　：　リストボックス中の選択をキャンセルしたことを通知します。

パラメタ：　wParam - コントロールの識別ＩＤと通知メッセージコード（AJCLBXN_SELCANCEL）
lParam – コントロールのウインドハンドル

戻り値　：　なし（ゼロを返してください）

11.7.4. 選択変更通知（AJCLBXN_SELCHANGE）

説　明　：　クリック操作やポップアップメニューでリストボックス中の選択が変更されたことを通知します。

パラメタ：　wParam – コントロールの識別ＩＤと通知メッセージコード（AJCLBXN_SELCHANGE）
lParam – コントロールのウインドハンドル

戻り値　：　なし（ゼロを返してください）

11.7.5. フォーカス取得通知（AJCLBXN_SETFOCUS）

説　明　：　リストボックスキーボードフォーカスを得たことを通知します。

パラメタ：　wParam – コントロールの識別ＩＤと通知メッセージコード（AJCLBXN_SETFOCUS）
lParam – コントロールのウインドハンドル

戻り値　：　なし（ゼロを返してください）

11.7.6. フォーカス喪失通知（AJCLBXN_KILLFOCUS）

説　明　：　リストボックスキーボードフォーカスを失ったことを通知します。

パラメタ：　wParam – コントロールの識別ＩＤと通知メッセージコード（AJCLBXN_KILLFOCUS）
lParam – コントロールのウインドハンドル

戻り値　：　なし（ゼロを返してください）

11.7.7. 右クリック通知（AJCLBXN_RCLICK）

説　明　：　SHIFT か CTRL キーを押しながら右クリックしたことを通知します。
右クリック通知の可否については、AjcLbxSetNtcRClk()を参照してください。
lParam で以下の右クリック情報を通知します。

```
typedef struct {
    int         x;              //  カーソル x 座標
    int         y;              //  カーソル y 座標
    BOOL        fShift;         //  SHIFT キー押下フラグ
    BOOL        fCtrl;          //  CTRL キー押下フラグ
} AJCLBXRCLK, *PAJCLBXRCLK;
```

パラメタ：　wParam – コントロールの識別ＩＤと通知メッセージコード（AJCLBXN_RCLICK）
lParam – 右クリック情報へのポインタ（PAJCLBXRCLK）（MFC 使用時はコントロールのウインドハンドル）

戻り値　：　なし（ゼロを返してください）

11.7.8. 項目削除通知（AJCLBXN_REMOVED）

説　明　：　右クリック操作によりリストボックスの項目が削除されたことを通知します。
AjcLbxGetRemovedItem()により、削除した項目の文字列を取得できます。

パラメタ：　wParam – コントロールの識別ＩＤと通知メッセージコード（AJCLBXN_REMOVED）
lParam – 削除された項目の数（MFC 使用時はコントロールのウインドハンドル）

戻り値　：　なし（ゼロを返してください）

11.7.9. ディレクトリドロップ通知（AJCLBXN_DROPDIR）

説　明　：　AJCLBXS_ACCEPTDIRS スタイルが指定されている場合、ディレクトリがドロップされたことを通知します。
AJCLBXS_ADDITEMINDROP スタイルが指定されている場合は、リストボックス項目に当該ディレクトリパスが追加されます。
AjcLbxGetDroppedDir[Ex]()により、ドロップされたディレクトリのパス名を取得できます。

パラメタ：　wParam – コントロールの識別ＩＤと通知メッセージコード（AJCLBXN_DROPDIR）
lParam – ドロップされたディレクトリ項目の数（MFC 使用時はコントロールのウインドハンドル）

戻り値　：　なし（ゼロを返してください）

11.7.10. ファイルドロップ通知（AJCLBXN_DROPFILE）

説　明　：　JCLBXS_ACCEPTFILESS スタイルが指定されている場合、ディレクトリがドロップされたことを通知します。
AJCLBXS_ADDITEMINDROP スタイルが指定されている場合は、リストボックス項目に当該ファイルパスが追加されます。
AjcLbxGetDroppedFile()により、ドロップされたファイルのパス名を取得できます。

パラメタ：　wParam – コントロールの識別ＩＤと通知メッセージコード（AJCLBXN_DROPFILE）
lParam – ドロップされたファイル項目の数（MFC 使用時はコントロールのウインドハンドル）

戻り値　：　なし（ゼロを返してください）

11.7.11. メモリ不足通知（AJCLBXN_ERRSPACE）

説　明　：　リストボックスの処理中にメモリが不足したことを通知します。

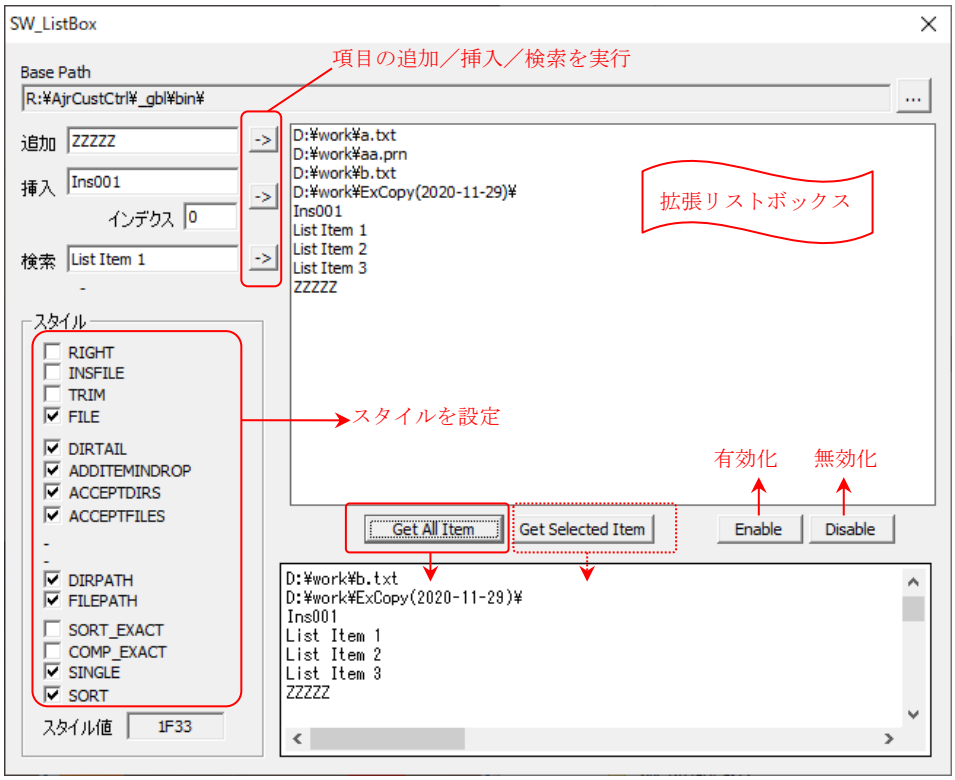
パラメタ：　wParam – コントロールの識別ＩＤと通知メッセージコード（AJCLBXN_ERRSPACE）
lParam – コントロールのウインドハンドル

戻り値　：　なし（ゼロを返してください）

11.8. サンプルプログラム

11.8.1. SW_ListBox（リストボックス操作サンプル）

リストボックス・コントロールのサンプルプログラムを以下に示します。




```
 1 : //
 2 : //  SW_ListBox.c
 3 : //
 4 : #include    <AjrCstXX.h>
 5 : #include    <math.h>
 6 : #include    <tchar.h>
 7 : #include    "resource.h"
 8 :
 9 : #define     TIPTEXT_ADD(ID, TEXT)   AjcTipTextAdd(GetDlgItem(hDlg, ID), TEXT)
10 :
11 : //-------------------------------------------------------------------------------------------------------------//
12 : //  ワーク                                                                                                     //
13 : //-------------------------------------------------------------------------------------------------------------//
14 : HINSTANCE       hInst;                  //  ＤＬＬインスタンスハンドル
15 : HWND            hDlgMain;               //  ダイアログボックスハンドル
16 : HWND            hLbx;                   //  リストボックスコントロールのウインドハンドル
17 :
18 : //-------------------------------------------------------------------------------------------------------------//
19 : //  内部サブ関数                                                                                               //
20 : //-------------------------------------------------------------------------------------------------------------//
21 : AJC_DLGPROC_DEF(Main);
22 : static  VO  SubCreateListBox(int sty);
23 :
24 : //=============================================================================================================//
25 : //                                                                                                             //
26 : //  ＷｉｎＭａｉｎ                                                                                            //
27 : //                                                                                                             //
28 : //=============================================================================================================//
29 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
30 : {
31 :     MSG     msg;
32 :
33 :     hInst = hInstance;
34 :
```

```
35 :     //----- メイン・ダイアログオープン ----------------//
36 :     hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_DLGMAIN), NULL, AJC_DLGPROC_NAME(Main));
37 :     //----- ダイアログ表示 ----------------------------//
38 :     ShowWindow(hDlgMain, SW_SHOW);
39 :
40 :     //----- メッセージループ --------------------------//
41 :     while (GetMessage(&msg, NULL, 0, 0)) {
42 :         do {
43 :             if (IsDialogMessage(hDlgMain, &msg)) break;
44 :             TranslateMessage(&msg);
45 :             DispatchMessage (&msg);
46 :         } while (0);
47 :     }
48 :
49 :     return (int)msg.wParam ;
50 : }
51 : //=============================================================================================================//
52 : //                                                                                                             //
53 : //  ダイアログ・プロシージャ                                                                                   //
54 : //                                                                                                             //
55 : //=============================================================================================================//
56 : //----- ダイアログ初期化 ----------------------------------------------------------------------------------//
57 : AJC_DLGPROC(Main, WM_INITDIALOG     )
58 : {
59 :     int     sty;
60 :     UT      path[MAX_PATH];
61 :     UT      defp[MAX_PATH];
62 :
63 :     hDlgMain = hDlg;
64 :     hLbx = GetDlgItem(hDlgMain, IDC_LISTBOX);
65 :
66 :     //----- リストボックスコントロール生成 ----------------------//
67 :     //
68 :     //  リストボックス生成時のみ有効なスタイルを設定可能とするため
69 :     //  リストボックスコントロールを CreateWindow()で生成します。
70 :     //
71 :     SubCreateListBox(AJCLBXS_STD_FILEDIR_S | AJCLBXS_SINGLE);
72 :     //----- 初期リスト項目設定 ----------------------------------//
73 :     AjcLbxAddString(hLbx, TEXT("List Item 1"));
74 :     AjcLbxAddString(hLbx, TEXT("List Item 2"));
75 :     AjcLbxAddString(hLbx, TEXT("List Item 3"));
76 :     //----- ベースパスのドロップ許可 ----------------------------//
77 :     AjcEnableDlgItemToDrop(hDlg, IDC_TXT_BASEPATH, AJCDROP_DIR);
78 :     //----- スタイルをチェックボックスに反映 --------------------//
79 :     sty = (int)MAjcGetWindowLong(hLbx, GWL_STYLE);
80 :     AjcSetDlgItemChk(hDlg, IDC_CHK_RIGHT        , (sty & AJCLBXS_RIGHT        ) != 0);
81 :     AjcSetDlgItemChk(hDlg, IDC_CHK_TRIM         , (sty & AJCLBXS_TRIM         ) != 0);
82 :     AjcSetDlgItemChk(hDlg, IDC_CHK_FILE         , (sty & AJCLBXS_FILE         ) != 0);
83 :     AjcSetDlgItemChk(hDlg, IDC_CHK_DIRTAIL      , (sty & AJCLBXS_DIRTAIL      ) != 0);
84 :     AjcSetDlgItemChk(hDlg, IDC_CHK_ADDITEMINDROP, (sty & AJCLBXS_ADDITEMINDROP) != 0);
85 :     AjcSetDlgItemChk(hDlg, IDC_CHK_ACCEPTDIRS   , (sty & AJCLBXS_ACCEPTDIRS   ) != 0);
86 :     AjcSetDlgItemChk(hDlg, IDC_CHK_ACCEPTFILES  , (sty & AJCLBXS_ACCEPTFILES  ) != 0);
87 :     AjcSetDlgItemChk(hDlg, IDC_CHK_DIRPATH      , (sty & AJCLBXS_DIRPATH      ) != 0);
88 :     AjcSetDlgItemChk(hDlg, IDC_CHK_FILEPATH     , (sty & AJCLBXS_FILEPATH     ) != 0);
89 :     AjcSetDlgItemChk(hDlg, IDC_CHK_SORT_EXACT   , (sty & AJCLBXS_SORT_EXACT   ) != 0);
90 :     AjcSetDlgItemChk(hDlg, IDC_CHK_COMP_EXACT   , (sty & AJCLBXS_COMP_EXACT   ) != 0);
91 :     AjcSetDlgItemChk(hDlg, IDC_CHK_SINGLE       , (sty & AJCLBXS_SINGLE       ) != 0);
92 :     AjcSetDlgItemChk(hDlg, IDC_CHK_SORT         , (sty & AJCLBXS_SORT         ) != 0);
93 :     //----- ツールチップ設定 ------------------------------------//
94 :     TIPTEXT_ADD(IDC_CHK_RIGHT       , TEXT("右詰め表示（ウインド生成時のみ有効）"));
95 :     TIPTEXT_ADD(IDC_CHK_INSFILE     , TEXT("選択項目無しで右クリック時「ファイル追加」メニュー表示"));
96 :     TIPTEXT_ADD(IDC_CHK_TRIM        , TEXT("項目の前後の空白を除去する"));
97 :     TIPTEXT_ADD(IDC_CHK_FILE        , TEXT("リスト項目としてファイル／フォルダパス名を扱う場合に指定します。¥n")
98 :         TEXT("FILE スタイルは、以下の項目を有効にします。¥n")
99 :         TEXT("  ・項目追加ダイアログ中の「ディレクトリ選択」「ファイル選択」「相対パスに変換」ボタンを有効化¥n")
100 :        TEXT("  ・FILEPATH スタイル指定時に、ポップアップメニュー中の「ファイル追加」を有効化¥n")
101 :        TEXT("  ・ベースパスが設定済で、選択項目がある場合、ポップアップメニュー中の「選択項目を相対パスに変換」")
102 :                                                                TEXT("「選択項目を絶対パスに変換」を有効化¥n")
103 :        TEXT("FILE スタイルが指定されていない場合は、上記項目は全て非表示となります。"));
104 :    TIPTEXT_ADD(IDC_CHK_DIRTAIL     , TEXT("ディレクトリ名の末尾に「¥¥」を付加する"));
105 :    TIPTEXT_ADD(IDC_CHK_ADDITEMINDROP,TEXT("ファイル／ディレクトリドロップでリストボックス項目を追加"));
106 :    TIPTEXT_ADD(IDC_CHK_ACCEPTDIRS  , TEXT("ディレクトリのドラッグ＆ドロップを可能にする"));
107 :    TIPTEXT_ADD(IDC_CHK_ACCEPTFILES , TEXT("ファイルのドラッグ＆ドロップを可能にする"));
108 :    TIPTEXT_ADD(IDC_CHK_DIRPATH     , TEXT("項目の追加／編集時にディレクトリパスを選択可能とする（AJCLBXS_FILE 有効時）"));
109 :    TIPTEXT_ADD(IDC_CHK_FILEPATH    , TEXT("項目の追加／編集時にファイルパスを選択可能とする     （AJCLBXS_FILE 有効時）"));
110 :    TIPTEXT_ADD(IDC_CHK_SORT_EXACT  , TEXT("COMP_EXACT と SORT_EXACT 指定時は、単に文字列の大小でソートされる")
111 :        TEXT("（ex. AAA < CCC < aaa < bbb）¥n")
112 :        TEXT("COMP_EXACT だけ指定した場合は、アルファベット順にソートされる（ex. AAA < aaa < bbb < CCC）"));
113 :    TIPTEXT_ADD(IDC_CHK_COMP_EXACT  , TEXT("英字の大小を区別して比較する"));
114 :    TIPTEXT_ADD(IDC_CHK_SINGLE      , TEXT("単一選択モード（ウインド生成時のみ有効）"));
```

```
115 :     TIPTEXT_ADD(IDC_CHK_SORT        , TEXT("アイテム追加時にソートする（ウインド生成時のみ有効）"));
116 :     //----- 設定値ロード ---------------------------------------//
117 :     AjcCtrlSetPermAtt (hLbx, AJCCTL_PSEL_EXCLUDE);
118 :     AjcLoadAllControlSettings(hDlg, TEXT("MySect"), AJCCTL_SELACT_ALL | AJCCTL_SELACT_NTCCHK);
119 :     AjcLbxLoadItems(hLbx, TEXT("MySect"), TEXT("Lbx"));
120 :     //----- プロファイルからベースパス読み出し -----------------//
121 :     AjcGetAppPath(defp, MAX_PATH);
122 :     AjcGetProfileStr(TEXT("MAIN"), TEXT("BasePath"), defp, path, MAX_PATH);
123 :     AjcSetDlgItemStr(hDlg, IDC_TXT_BASEPATH, path);
124 :     //----- 相対パスへ変換時のベースパス設定 -------------------//
125 :     AjcLbxSetBasePath(hLbx, path);
126 :     //----- 初期メッセージ -------------------------------------//
127 :     AjcVthPrintF(GetDlgItem(hDlg, IDC_VTH), TEXT("ここに「Get All Item」「Get Selected Item」ボタン¥n で取得した項目を表示します¥n"));
128 :     return TRUE;
129 : }
130 : //----- ウインド破棄 -------------------------------------------------------------------------------------//
131 : AJC_DLGPROC(Main, WM_DESTROY        )
132 : {
133 :     //----- 設定値セーブ ---------------------------------------//
134 :     AjcSaveAllControlSettings(hDlg);
135 :     AjcLbxSaveItems(hLbx, TEXT("MySect"), TEXT("Lbx"));
136 :     //----- リストボックス破棄 ---------------------------------//
137 :     DestroyWindow(hLbx);
138 :     //----- プログラム終了 -------------------------------------//
139 :     PostQuitMessage(0);
140 :     return TRUE;
141 : }
142 : //----- ベースパス設定ボタン -----------------------------------------------------------------------------//
143 : AJC_DLGPROC(Main, IDC_CMD_BASEPATH  )
144 : {
145 :     UT  path[MAX_PATH];
146 :
147 :     AjcGetDlgItemStr(hDlg, IDC_TXT_BASEPATH, path, MAX_PATH);
148 :     if (AjcGetFolderName(hDlg, TEXT("Set base path"), path, path, FALSE)) {
149 :         AjcSetDlgItemStr(hDlg, IDC_TXT_BASEPATH, path);
150 :         AjcLbxSetBasePath(hLbx, path);
151 :     }
152 :     return TRUE;
153 : }
154 : //----- GetAll Item ボタン -------------------------------------------------------------------------------//
155 : AJC_DLGPROC(Main, IDC_CMD_GETALL    )
156 : {
157 :     PAJCLBXITEM pTop, p;
158 :     UI          i, n;
159 :
160 :     AjcVthClearAllText(GetDlgItem(hDlg, IDC_VTH));
161 :     if (pTop = AjcLbxGetAllItems(hLbx, &n)) {
162 :         for (i = 0, p = pTop; i < n; i++, p++) {
163 :             AjcVthPrintF(GetDlgItem(hDlg, IDC_VTH), TEXT("%s¥n"), p->pStr);
164 :         }
165 :         AjcLbxRelAllItems(pTop);
166 :     }
167 :     return TRUE;
168 : }
169 : //----- Get Selected Item ボタン -------------------------------------------------------------------------//
170 : AJC_DLGPROC(Main, IDC_CMD_GETSEL    )
171 : {
172 :     PAJCLBXITEM pTop, p;
173 :     UI          i, n;
174 :
175 :     AjcVthClearAllText(GetDlgItem(hDlg, IDC_VTH));
176 :     if (pTop = AjcLbxGetSelectedItems(hLbx, &n)) {
177 :         for (i = 0, p = pTop; i < n; i++, p++) {
178 :             AjcVthPrintF(GetDlgItem(hDlg, IDC_VTH), TEXT("%s¥n"), p->pStr);
179 :         }
180 :         AjcLbxRelSelectedItems(pTop);
181 :     }
182 :     return TRUE;
183 : }
184 : //----- Enable ボタン ------------------------------------------------------------------------------------//
185 : AJC_DLGPROC(Main, IDC_CMD_ENABLE    )
186 : {
187 :     EnableWindow(hLbx, TRUE);
188 :     return TRUE;
189 : }
190 : //----- Disable ボタン -----------------------------------------------------------------------------------//
191 : AJC_DLGPROC(Main, IDC_CMD_DISABLE   )
192 : {
193 :     EnableWindow(hLbx, FALSE);
194 :     return TRUE;
```

```
195 : }
196 : //----- リストボックス ----------------------------------------------------------------------------------//
197 : AJC_DLGPROC(Main, IDC_LISTBOX       )
198 : {
199 :     switch (HIWORD(wParam)) {
200 :         case AJCLBXN_DBLCLK:                        //  ●ダブルクリック通知
201 :         case AJCLBXN_ERRSPACE:                      //  ●メモリ不足通知
202 :         case AJCLBXN_KILLFOCUS:                     //  ●フォーカス喪失通知
203 :         case AJCLBXN_SELCANCEL:                     //  ●選択キャンセル通知
204 :         case AJCLBXN_SELCHANGE:                     //  ●選択変更通知
205 :         case AJCLBXN_SETFOCUS:                      //  ●フォーカス取得通知
206 :             break;
207 :
208 :         case AJCLBXN_RCLICK:                        //  ●右クリック通知
209 :         {   PAJCLBXRCLK p = (PAJCLBXRCLK)lParam;
210 :             UT          txt[64] = {0};
211 :             AjcSnPrintF(txt, 64, TEXT("%s%s 右クリック発生(x = %d, y = %d)"), p->fShift ? TEXT("Shift+") : TEXT(""),
212 :                                                                               p->fCtrl  ? TEXT("Ctrl+" ) : TEXT(""),
213 :                                                                               p->x, p->y);
214 :             MessageBox(hDlg, txt, TEXT("S_CtrlVT100_01"), MB_OK);
215 :             break;
216 :         }
217 :
218 :         case AJCLBXN_DROPDIR:                       //  ●ディレクトリドロップ
219 :         {   UT      path[MAX_PATH];
220 :             UT      txt[4096];
221 :             MAjcStrCpy(txt, AJCTSIZE(txt), TEXT("-- Dir dropped --¥n"));
222 :             while (AjcLbxGetDroppedDir(hLbx, path)) {
223 :                 MAjcStrCat(txt, AJCTSIZE(txt), path);
224 :                 MAjcStrCat(txt, AJCTSIZE(txt), TEXT("¥n"));
225 :             }
226 :             MessageBox(hDlg, txt, TEXT("Dropped Dir"), MB_OK);
227 :             break;
228 :         }
229 :
230 :         case AJCLBXN_DROPFILE:                      //  ●ファイルドロップ
231 :         {   UT      path[MAX_PATH];
232 :             UT      txt[4096];
233 :             MAjcStrCpy(txt, AJCTSIZE(txt), TEXT("-- File dropped --¥n"));
234 :             while (AjcLbxGetDroppedFile(hLbx, path)) {
235 :                 MAjcStrCat(txt, AJCTSIZE(txt), path);
236 :                 MAjcStrCat(txt, AJCTSIZE(txt), TEXT("¥n"));
237 :             }
238 :             MessageBox(hDlg, txt, TEXT("Dropped File"), MB_OK);
239 :             break;
240 :         }
241 :
242 :         case AJCLBXN_REMOVED:                       //  ●項目削除
243 :         {   UT      str[AJCLBX_MAXSTL];
244 :             UT      txt[4096];
245 :             MAjcStrCpy(txt, AJCTSIZE(txt), TEXT("-- Removed --¥n"));
246 :             while (AjcLbxGetRemovedItem(hLbx, str, AJCTSIZE(str))) {
247 :                 if (MAjcStrLen(txt) + MAjcStrLen(str) < AJCTSIZE(txt) - 8) {
248 :                     MAjcStrCat(txt, AJCTSIZE(txt), str);
249 :                     MAjcStrCat(txt, AJCTSIZE(txt), TEXT("¥n"));
250 :                 }
251 :                 else {
252 :                     MAjcStrCat(txt, AJCTSIZE(txt), TEXT("...¥n"));
253 :                     break;
254 :                 }
255 :             }
256 :             MessageBox(hDlg, txt, TEXT("Removed Items"), MB_OK);
257 :             break;
258 :         }
259 :     }
260 :     return TRUE;
261 : }
262 : //----- スタイル・チェックボックス ----------------------------------------------------------------------//
263 : //  当該スタイルビットをセット／リセットする
264 : #define SET_STY(S)  if (HIWORD(wParam) == BN_CLICKED) {                                                          ¥
265 :                         UT      val[16];                                                                        ¥
266 :                         int     sty = (int)MAjcGetWindowLong(hLbx, GWL_STYLE);                                  ¥
267 :                         if (AjcGetDlgItemChk(hDlg, IDC_CHK_##S)) MAjcSetWindowLong(hLbx, GWL_STYLE, sty |=  AJCLBXS_##S);   ¥
268 :                         else                                     MAjcSetWindowLong(hLbx, GWL_STYLE, sty &= ~AJCLBXS_##S);   ¥
269 :                         AjcSnPrintF(val, AJCTSIZE(val), TEXT("%04X"), sty & 0x0000FFFF);                        ¥
270 :                         AjcSetDlgItemStr(hDlg, IDC_TXT_STYLE, val);                                             ¥
271 :                     }
272 : AJC_DLGPROC(Main, IDC_CHK_TRIM          )   {SET_STY(TRIM          ); return TRUE;}
273 : AJC_DLGPROC(Main, IDC_CHK_FILE          )   {SET_STY(FILE          ); return TRUE;}
274 : AJC_DLGPROC(Main, IDC_CHK_DIRTAIL       )   {SET_STY(DIRTAIL       ); return TRUE;}
```

```
275 : AJC_DLGPROC(Main, IDC_CHK_ADDITEMINDROP )   {SET_STY(ADDITEMINDROP  ); return TRUE;}
276 : AJC_DLGPROC(Main, IDC_CHK_ACCEPTDIRS    )   {SET_STY(ACCEPTDIRS     ); return TRUE;}
277 : AJC_DLGPROC(Main, IDC_CHK_ACCEPTFILES   )   {SET_STY(ACCEPTFILES    ); return TRUE;}
278 : AJC_DLGPROC(Main, IDC_CHK_DIRPATH       )   {SET_STY(DIRPATH        ); return TRUE;}
279 : AJC_DLGPROC(Main, IDC_CHK_FILEPATH      )   {SET_STY(FILEPATH       ); return TRUE;}
280 : AJC_DLGPROC(Main, IDC_CHK_SORT_EXACT    )   {SET_STY(SORT_EXACT     ); return TRUE;}
281 : AJC_DLGPROC(Main, IDC_CHK_COMP_EXACT    )   {SET_STY(COMP_EXACT     ); return TRUE;}
282 :
283 : //  AJCLBXS_RIGHT/SINGLE/SORTは、ウインド生成時にのみ指定可能なため、ウインドを生成し直してスタイルを設定する
284 : #define REMK_LBX(S) if (HIWORD(wParam) == BN_CLICKED) {                                                          ¥
285 :                         UT      val[16];                                                                        ¥
286 :                         int     sty = (int)MAjcGetWindowLong(hLbx, GWL_STYLE);          /*  スタイル更新値設定          */ ¥
287 :                         if (AjcGetDlgItemChk(hDlg, IDC_CHK_##S)) sty |=  AJCLBXS_##S;   /*  ・                          */ ¥
288 :                         else                                     sty &= ~AJCLBXS_##S;   /*  ・                          */ ¥
289 :                         AjcLbxSaveItems(hLbx, TEXT("LbxTemp"), TEXT("LBX"));            /*  リストボックス項目をセーブ  */ ¥
290 :                         DestroyWindow(hLbx);                                            /*  リストボックス破棄          */ ¥
291 :                         SubCreateListBox(sty & 0xFFFF);                                 /*  リストボックス再生成        */ ¥
292 :                         AjcLbxLoadItems  (hLbx, TEXT("LbxTemp"), TEXT("LBX"));          /*  リストボックス項目ロード    */ ¥
293 :                         AjcDelProfileSect(TEXT("LbxTemp"));                             /*  プロファイルセクション消去  */ ¥
294 :                         AjcSnPrintF(val, AJCTSIZE(val), TEXT("%04X"), sty & 0x0000FFFF);/*  スタイル値表示              */ ¥
295 :                         AjcSetDlgItemStr(hDlg, IDC_TXT_STYLE, val);                     /*  ・                          */ ¥
296 :                     }
297 :
298 : AJC_DLGPROC(Main, IDC_CHK_RIGHT         )   {REMK_LBX(RIGHT ); return TRUE;}
299 : AJC_DLGPROC(Main, IDC_CHK_SINGLE        )   {REMK_LBX(SINGLE); return TRUE;}
300 : AJC_DLGPROC(Main, IDC_CHK_SORT          )   {REMK_LBX(SORT  ); return TRUE;}
301 :
302 : //----- 項目追加 ----------------------------------------------------------------------------------------------//
303 : AJC_DLGPROC(Main, IDC_CMD_ADD       )
304 : {
305 :     if (HIWORD(wParam) == BN_CLICKED) {
306 :         UT  txt[AJCLBX_MAXSTL];
307 :         AjcGetDlgItemStr(hDlg, IDC_TXT_ADD, txt, AJCTSIZE(txt));
308 :         AjcLbxAddString(hLbx, txt);
309 :     }
310 :     return TRUE;
311 : }
312 : //----- 項目挿入 ----------------------------------------------------------------------------------------------//
313 : AJC_DLGPROC(Main, IDC_CMD_INS       )
314 : {
315 :     if (HIWORD(wParam) == BN_CLICKED) {
316 :         UT  txt[AJCLBX_MAXSTL];
317 :         AjcGetDlgItemStr(hDlg, IDC_TXT_INS, txt, AJCTSIZE(txt));
318 :         AjcLbxInsertString(hLbx,AjcGetDlgItemUInt(hDlg, IDC_TXT_INS_IX), txt);
319 :     }
320 :     return TRUE;
321 : }
322 : //----- 項目検索 ----------------------------------------------------------------------------------------------//
323 : AJC_DLGPROC(Main, IDC_CMD_SRH       )
324 : {
325 :     if (HIWORD(wParam) == BN_CLICKED) {
326 :         int ix;
327 :         UT  txt[AJCLBX_MAXSTL];
328 :         UT  rsu[256];
329 :         AjcGetDlgItemStr(hDlg, IDC_TXT_SRH, txt, AJCTSIZE(txt));
330 :         ix = AjcLbxFindString(hLbx, -1, txt);
331 :         if (ix >= 0) AjcSnPrintF(rsu, AJCTSIZE(rsu), TEXT("Find ix = %d"), ix);
332 :         else         AjcSnPrintF(rsu, AJCTSIZE(rsu), TEXT("Not found"   ));
333 :         AjcSetDlgItemStr(hDlg, IDC_LBL_SRH, rsu);
334 :
335 :         ix = AjcLbxGetTextLen(hLbx, 1);
336 :         txt[0] = 0;
337 :     }
338 :     return TRUE;
339 : }
340 : //----- 「Cancel」ボタン ----------------------------------------------------------------------------------------//
341 : AJC_DLGPROC(Main, IDCANCEL          )
342 : {
343 :     DestroyWindow(hDlg);
344 :     return TRUE;
345 : }
346 : //-------------------------------------------------------------------------------------------------------------//
347 : AJC_DLGMAP_DEF(Main)
348 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG          )
349 :     AJC_DLGMAP_MSG(Main, WM_DESTROY             )
350 :
351 :     AJC_DLGMAP_CMD(Main, IDC_CMD_BASEPATH       )
352 :     AJC_DLGMAP_CMD(Main, IDC_CMD_GETALL         )
353 :     AJC_DLGMAP_CMD(Main, IDC_CMD_GETSEL         )
354 :     AJC_DLGMAP_CMD(Main, IDC_CMD_ENABLE         )
```

```
355 :     AJC_DLGMAP_CMD(Main, IDC_CMD_DISABLE        )
356 :     AJC_DLGMAP_CMD(Main, IDC_LISTBOX            )
357 :
358 :     AJC_DLGMAP_CMD(Main, IDC_CMD_ADD            )
359 :     AJC_DLGMAP_CMD(Main, IDC_CMD_INS            )
360 :     AJC_DLGMAP_CMD(Main, IDC_CMD_SRH            )
361 :
362 :     AJC_DLGMAP_CMD(Main, IDC_CHK_TRIM           )
363 :     AJC_DLGMAP_CMD(Main, IDC_CHK_FILE           )
364 :     AJC_DLGMAP_CMD(Main, IDC_CHK_DIRTAIL        )
365 :     AJC_DLGMAP_CMD(Main, IDC_CHK_ADDITEMINDROP  )
366 :     AJC_DLGMAP_CMD(Main, IDC_CHK_ACCEPTDIRS     )
367 :     AJC_DLGMAP_CMD(Main, IDC_CHK_ACCEPTFILES    )
368 :     AJC_DLGMAP_CMD(Main, IDC_CHK_DIRPATH        )
369 :     AJC_DLGMAP_CMD(Main, IDC_CHK_FILEPATH       )
370 :     AJC_DLGMAP_CMD(Main, IDC_CHK_SORT_EXACT     )
371 :     AJC_DLGMAP_CMD(Main, IDC_CHK_COMP_EXACT     )
372 :
373 :     AJC_DLGMAP_CMD(Main, IDC_CHK_RIGHT          )
374 :     AJC_DLGMAP_CMD(Main, IDC_CHK_SINGLE         )
375 :     AJC_DLGMAP_CMD(Main, IDC_CHK_SORT           )
376 :
377 :     AJC_DLGMAP_CMD(Main, IDCANCEL              )
378 : AJC_DLGMAP_END
379 : //-------------------------------------------------------------------------------------------------------------------//
380 : //  リストボックス生成                                                                                               //
381 : //-------------------------------------------------------------------------------------------------------------------//
382 : static  VO  SubCreateListBox(int sty)
383 : {
384 :     RECT    r;
385 :
386 :     GetWindowRect(GetDlgItem(hDlgMain, IDC_LBL_LBX), &r);
387 :     MapWindowPoints(NULL, hDlgMain, (LPPOINT)&r, 2);
388 :
389 :     hLbx = CreateWindow(TEXT("AjcCtrlListBox"),         // window class name
390 :                         TEXT(""),                       // window caption
391 :                         WS_CHILD | WS_TABSTOP | sty,    // window style
392 :                         r.left,                         // initial x position
393 :                         r.top,                          // initial y position
394 :                         r.right - r.left,               // initial x size
395 :                         r.bottom - r.top,               // initial y size
396 :                         hDlgMain,                       // parent window handle
397 :                         (HMENU)IDC_LISTBOX,             // window menu handle
398 :                         hInst,                          // program instance handle
399 :                         NULL);                          // creation parameters
400 :     ShowWindow(hLbx, SW_SHOWNA);
401 :     //  タブ＃を「IDC_LBL_LBX」と同等にする
402 :     SetWindowPos(hLbx, GetDlgItem(hDlgMain, IDC_LBL_LBX), 0, 0, 0, 0, SWP_NOMOVE | SWP_NOSIZE);
403 :     AjcCtrlSetPermAtt (hLbx, AJCCTL_PSEL_EXCLUDE);
404 : }
```

12. 通信コントロールで共通な内容の説明

この章で説明する内容は、以下の通信モジュールで共通の内容となっています。

・シリアル通信　　（略称 SCP, ＣＯＭポート, メールスロット(UDP/IP), ソケット(TCP/IP クライアント））
・ソケットサーバ　　（略称 SSV, TCP/IP サーバ機能）
・ソケットクライアント（略称 SCT, TCP/IP クライアント機能）
・ソケットサーバ　　（略称 SSV, TCP/IP サーバ機能）

12.1. チャンクデータ

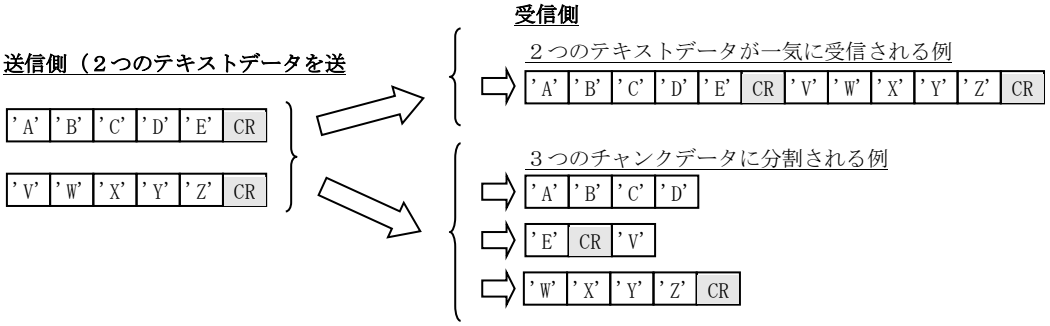
チャンクデータとは、１回の受信操作で読み出されたデータを意味します。
チャンクデータは、一連のバイトストリームの一部分のデータとなりますが、データのサイズも区切りも不定です。

12.2. テキストデータ

テキストデータの送受信は、マルチバイト文字（S-JIS / UTF-8 / EUC-J）で行います。
UNICODE モードでワイド文字を送信する場合は、内部でマルチバイト文字（S-JIS / UTF-8 / EUC-J）に変換します。
UNICODE モードで、文字列を通知する場合は、マルチバイト文字（S-JIS / UTF-8 / EUC-J）をワイド文字(UNICODE)に変換して通知します。

12.3. データ区切りの認識

通常、データの送受信動作は非同期に行われるため、受信側において意図しないチャンクデータとして受信される場合があります。
例えば、テキストデータの末尾に区切り記号として CR(0x0D)を付加してテキストを送信する場合、送信側で２つのテキストデータを送信したとしても、受信側では１つのチャンクデータとして受信されたり、また、３つのチャンクデータとして受信されたりします。



このような場合、受信側では、受信したデータを解析／編集し、２つのテキストデータに復元する必要がありますが、本ライブラリでは、このようなデータストリームの区切りを認識し、各テキストデータの単位で受信通知することができます。
上記の例では、"ABCDE" と "VWXYZ" の２つのテキストデータとして受信通知します。

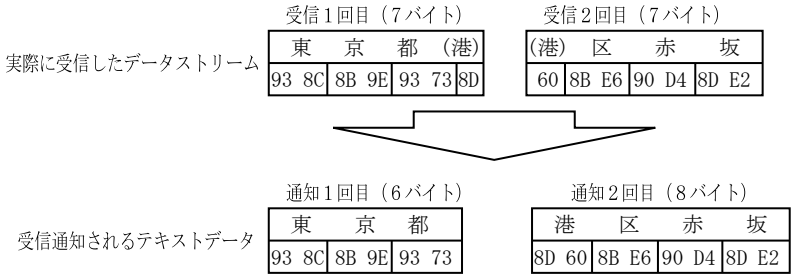
また、テキストデータのほかにも、以下のデータの区切りを認識し、当該データブロックの単位で受信通知することができます。

テキストデータ	テキストデータ	印字可能文字と TAB(0x09)
ＥＳＣシーケンス	ESC | 英字／制御コード以外 | 英字	ESC=0x1B
制御コード	CTRL	CTRL=0x00～0x1F or 0x7F（但し、TAB(0x09)は除く）
パケット・フレーム	DLE | STX | パケットデータ | DLE | ETX	デフォルトでは、STX=0x02, ETX=0x03, DLE=0x10
パケット外テキスト	テキストデータ	パケットフレーム以外の全データ
バイトペアデータ（14Bit データ）	1・・・ | 0・・・	MSB=1, MSB=0 の連続する２バイトで 14bit データを表す

12.4. テキストデータのマルチバイト文字分断抑止

リアルタイムにテキストデータを通知（テキストチャンクデータ，パケット外テキストデータを通知）する場合、受信チャンクデータ間で全角文字（複数バイト文字）が分断されないように通知します。

例えば、チャンクテキストの受信通知において、S-JIS による１４バイトのデータストリーム ”東京都港区赤坂”（16 進バイト列で、93 8C 8B 9E 93 73 8D 60 8B E6 90 D4 8D E2）を、７バイトづつ、２回に分けて受信した場合、１回目＝６バイト（UNICODE 時は３文字），２回目＝８バイト（UNICODE 時は４文字）で受信通知します。



バイナリチャンクデータを通知する場合は、マルチバイト文字の分断抑止は行われません。

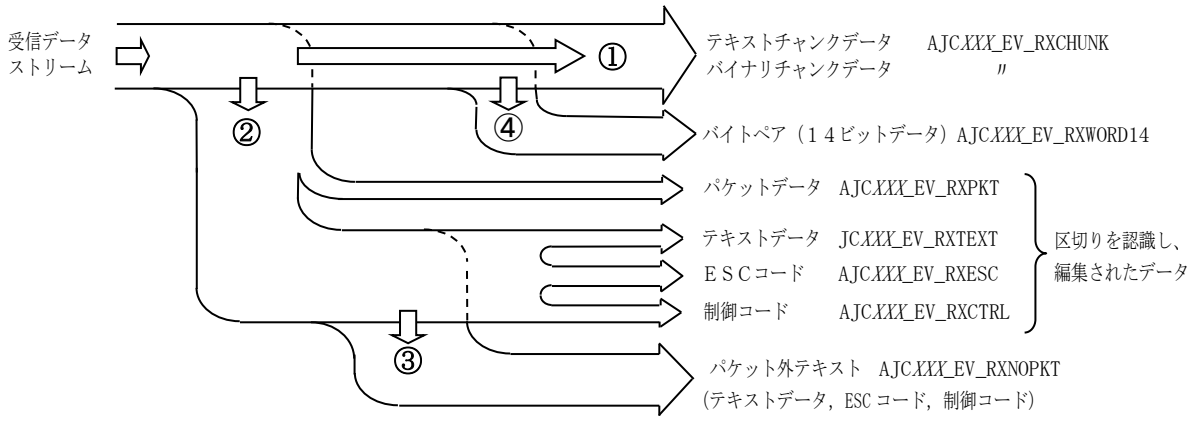
テキスト受信通知の場合は、１つまたは複数のチャンクデータに渡り、制御コード(TAB(0x09)を除く 0x00～0x1F，0x7F)の受信によりテキストストリームが完結するまで待つことから、マルチバイト文字の分断は発生しないのでマルチバイト文字分断抑止は行いません。

12.5. 受信通知イベント

本ライブラリの通信モジュールでは、以下のイベントで受信データを通知します。

#	イベント名	内容	備考
1	AJC*XXX*_EV_RXCHUNK	テキストチャンクデータ受信通知(param≠0)	リアルタイム
		バイナリチャンクデータ受信通知(param＝0)	リアルタイム
2	AJC*XXX*_EV_RXTEXT	テキスト受信通知	
3	AJC*XXX*_EV_RXESC	ＥＳＣシーケンス受信通知	
4	AJC*XXX*_EV_RXCTRL	制御コード受信通知	
5	AJC*XXX*_EV_RXPKT	パケットデータ受信通知	
6	AJC*XXX*_EV_RXNOPKT	パケット外テキスト受信通知	リアルタイム
7	AJC*XXX*_EV_RXWORD14	バイトペアによるワード（14Bit）データ受信	シリアル通信のみ

これらのイベントで通知されるデータは、以下のような流れになっています。



チャンクデータ（テキストチャンクデータ，バイナリチャンクデータ，不正チャンクテキスト）とパケット外テキストは、１回の受信操作で読み出したデータを元に通知されるデータで、受信と同時にリアルタイムに通知します。

その他のデータは、１回あるいは複数回に渡る受信操作で読み出したデータから、データストリームの切れ目を認識し、テキストデータについてはエンコードの変換や、マルチバイト文字の分断を抑止した上で、各種データブロックとして通知します。

12.5.1. テキスト チャンク・データ受信通知（AJC*XXX*_EV_RXCHUNK）

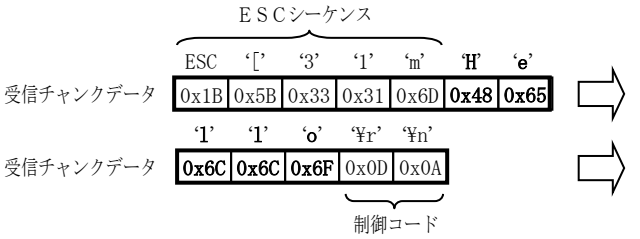
このイベントは、AjcScpSetMode()で、ChunkMode 引数に「AJCSCP_CM_TEXT」/「AJCSCP_CM_BOTH」を指定した場合に発生します。
１回の受信操作で読み出したチャンクデータを、テキストエンコードの変換とマルチバイト文字の分断を抑止した上で通知します。
チャンクデータ中に文字列終端(‘¥0’)が含まれる場合は、’¥0’以降の文字列は無視されます。

12.5.2. バイナリ チャンク・データ受信通知（AJC*XXX*_EV_RXCHUNK）

このイベントは、AjcScpSetMode()で、ChunkMode 引数に「AJCSCP_CM_BIN」/「AJCSCP_CM_BOTH」を指定した場合に発生します。
１回の受信操作で読み出したチャンクデータを、そのままバイナリデータとして通知します。

12.5.3. テキスト受信通知（AJC*XXX*_EV_RXTEXT）

テキストデータを受信した場合に通知されます。
テキストデータは、制御コード(TAB(0x09)を除く 0x00～0x1F, 0x7F)、パケットフレームやＥＳＣシーケンスによりサンドイッチされた部分のデータです。(TAB(0x09)はテキストデータとして扱います)
テキストデータは、複数のチャンクデータに渡って認識します。



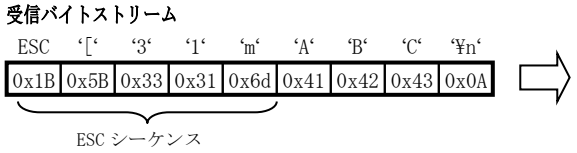
“¥x1B[31m”	ESC シーケンス受信通知

“Hello”	テキスト受信通知
“¥r”	制御コード受信通知
“¥n”	制御コード受信通知

12.5.4. ＥＳＣシーケンス受信通知（AJC*XXX*_EV_RXESC）

ＥＳＣシーケンスを受信した場合に通知されます。
ＥＳＣシーケンスは、ESC(0x1B)で始まり、英字(A～Z / a～z)で終端する文字列です。
ＥＳＣシーケンスも、テキストデータと同様に１つ、あるいは複数のチャンクデータに渡って認識します。

ＥＳＣシーケンスの受信通知が無効である場合は、ESC(0x1B)は制御コードとして認識します。



ESC シーケンス受信通知が有効の場合

“¥x1B[31m”	ESC シーケンス受信通知
“ABC”	テキスト受信通知
“¥n”	制御コード受信通知

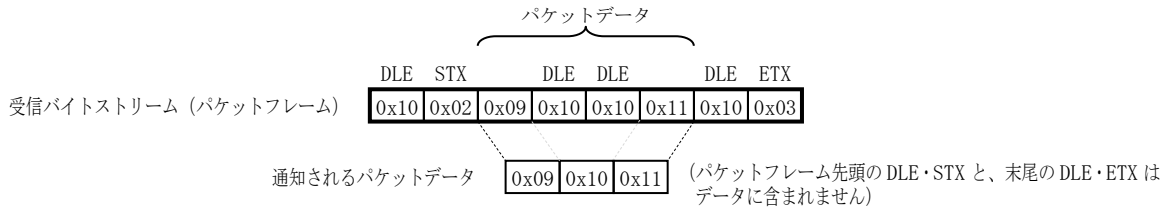
ESC シーケンス受信通知が無効の場合

“¥x1B”	制御コード受信通知
“[31mABC”	テキスト受信通知
“¥n”	制御コード受信通知

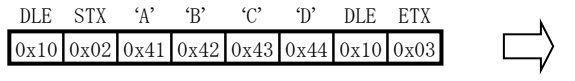
12.5.5. パケットデータ受信通知（AJC*XXX*_EV_RXPKT）

パケットフレームを受信した場合に通知されます。
パケットフレームは、DLE・STX で始まり、DLE・ETX で終端するバイナリデータです。（データ部分は２つの DLE で１バイトを表す）
STX,ETX や DLE の実際のコード値は、自由に設定可能です。（デフォルトでは、STX=0x02, ETX=0x03, DLE=0x10）
通知されるのは、先頭の DLE・ETX と、末尾の DLE・ETX でサンドイッチされた部分のデータです。
パケットフレームも、テキストデータと同様に１つ、あるいは複数のチャンクデータに渡って認識します。

パケットデータ中の、２つの連続する DLE は、１つの DLE に変換されます。
例えば、（DLE=0x10 として）実際の伝送路上のパケットデータが「0x09, 0x10, 0x10, 0x11」の４バイトである場合、重複する２つの DLE は、１つの DLE に変換され、通知されるパケットデータは「0x09, 0x10, 0x11」の３バイトとなります。
送信する場合は、この逆の操作を行います。つまり、パケットデータ中の DLE は、２つの DLE に変換して送信します。



パケットデータの通知イベントが指定されていない場合、STX, ETX, DLE は制御コードとして認識します。



パケットデータ受信通知が有効の場合

0x41, 0x42, 0x43, 0x44	パケットデータ受信通知

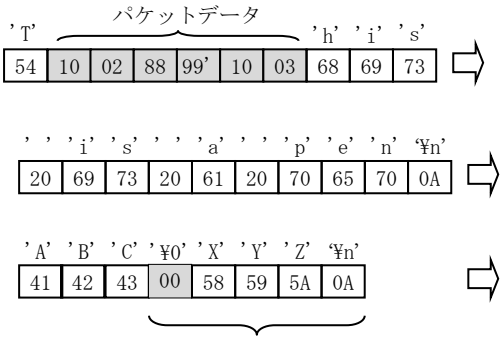
パケットデータ受信通知が無効の場合

“¥x10”	制御コード受信通知
“¥x02”	制御コード受信通知
“ABCD”	テキスト受信通知
“¥x10”	制御コード受信通知
“¥x03”	制御コード受信通知

12.5.6. パケット外テキスト受信通知（AJCXXX_EV_RXNOPKT）

パケット外テキストとは、受信したチャンクデータ中で、パケットフレーム部分を除いた部分のデータを意味します。
パケットデータの受信イベントが無効である場合は、受信チャンクデータ全体となります。
通知されるデータは、テキストデータ＋ＥＳＣシーケンス＋制御コードとほぼ同じになりますが、テキストデータの場合はテキスト終端として制御コード(TAB(0x09)を除く 0x00～0x1F, 0x7F)を認識するまで通知されないのに対し、パケット外テキストではテキストの終端を待たずにリアルタイムに通知されます。
但し、テキストの末尾でマルチバイト文字が分断される場合は、マルチバイト文字の分断を抑止します。
受信チャンクデータにヌル文字(0x00)が含まれる場合は、ヌル文字の直前までが有効となります。

受信チャンクデータ



ヌル文字('¥0')以降はパケット外テキストとして受信通知しない
（パケット外データは"ABC¥0XYZ¥n"だが、¥0 は文字列終端となる為
末尾の"¥0XYZ¥n はパケット外通知テキストから除外される"）

0x88, 0x99	パケットデータ受信通知
"This"	パケット外テキスト受信通知

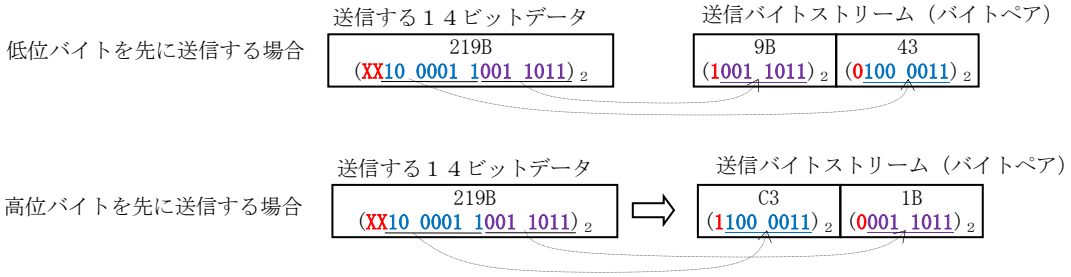
"This is a pen"	テキスト受信通知
" is a pen¥n"	パケット外テキスト受信通知
"¥x0A"	制御コード受信通知

"ABC"	テキスト受信通知
"ABC"	パケット外テキスト受信通知
"¥0"	制御コード受信通知
"XYZ"	テキスト受信通知
"¥n"	制御コード受信通知

12.5.7. バイトペアによるワード(14Bit)データ受信通知（AJC*XXX*_EV_RXWORD14）

バイトペアデータを認識した場合に通知します。
バイトペアデータは、MSB=1, MSB=0 の連続する２バイトで 14bit データを表します。
バイトペアデータも、テキストデータと同様に１つ、あるいは複数のチャンクデータに渡って認識します。

バイトペアデータは、MSB=1, MSB=0 の連続する２バイトで 14bit データを表します。
例えば、14 ビットのデータ「0x219B」を送信する場合、以下の２バイトに分割して送信します。



バイトペアの送受信は「シリアル通信」でのみ可能です。（ソケット通信ではバイトペアを認識しません）

12.5.8. パケットフレーム外での透過制御バイト（DLE)

パケットデータ受信通知が有効である場合、パケットデータ外でのＤＬＥは無視されます。
但し、２つ連続したＤＬＥは１ヶのＤＬＥとして認識します。



12.6. 送受信動作

送受信動作をメインスレッドとは非同期に行うために、内部的に送受信用のサブスレッドを生成します。
（TCP/IP サーバ機能では、接続クライアント毎にサブスレッドを生成します）
送信データは、バッファにスプールされ、サブスレッドにより順次取り出されて送出されるため、大量のデータを一気に送信する場合でも、ユーザプログラムでは送信の完了を待たずに処理を続行することができます。
受信データは、サブスレッドにより動的に確保されたメモリに格納され、ユーザ通知データとしてキューイングします。

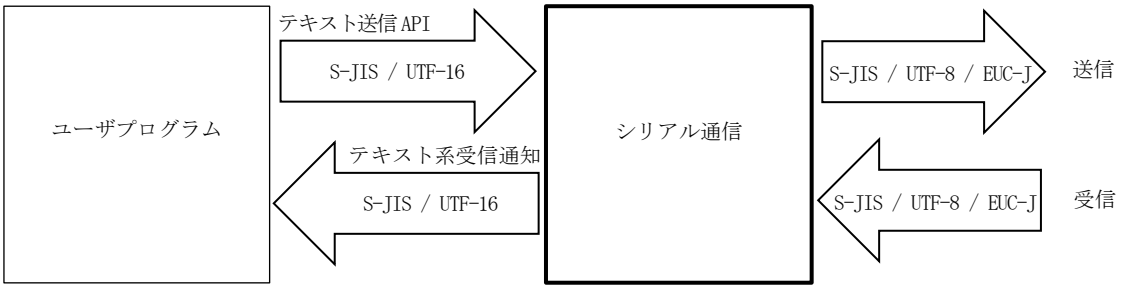
いずれの場合も、ユーザアプリケーションはシングルスレッドで動作可能であり、各受信データは（ボタンクリック等と同様の）イベントとして通知されます。

12.7. 送受信テキストデータの文字コード

テキストデータは、全てマルチバイト・テキストで送受信します。
サポートするマルチバイト・テキストの文字コードは以下の通りです。

・既定のマルチバイト・コードページ（日本語の場合は、CP932（S-JIS））
・ＵＴＦ－８　(UTF-8)
・日本語ＥＵＣ(EUC-J)

Windows では、テキストの文字コードを S-JIS/UTF-16 として扱います。
シリアル通信では送受信するテキストの文字コードを変換する為、ユーザは送受信テキストの文字コードを気にせずに処理することができます。



受信テキストの文字コードは、RxTextCode プロパティにより設定します。
受信するテキストの文字コードを指定することにより、受信したテキストコードを S-JIS/UTF-16 に変換して受信通知します。

・RxTextCode = SJIS - 受信するテキストの文字コードが SJIS であることを指定（デフォルト）
・RxTextCode = UTF8 - 受信するテキストの文字コードが UTF-8 であることを指定
・RxTextCode = EUC　- 受信するテキストの文字コードが日本語 EUC であることを指定
・RxTextCode = AUTO - 受信するテキストの文字コードを自動判別する

受信通知で通知されるテキストコードは、（プログラムで処理可能とするために）上記で指定された受信テキストの文字コードを S-JIS/UTF-16 に変換して通知します。

送信テキストの文字コードは、TxTextCode プロパティにより設定します。
SendText()や SendChar()メソッドでテキストデータを送信する場合、TxTextCode プロパティの指定に従いテキストの文字コードを変換して送信します。

・TxTextCode = SJIS - 送信テキストを S-JIS/UTF-16 → SJIS に変換して送信（デフォルト）
・TxTextCode = UTF8 - 送信テキストを S-JIS/UTF-16 → UTF-8 に変換して送信
・TxTextCode = EUC　- 送信テキストを S-JIS/UTF-16 → 日本語 EUC に変換して送信
・TxTextCode = AUTO - 送信テキストの文字コードを S-JIS/UTF-16 → 受信テキストの文字コードに変換して送信
　　(受信テキストの文字コードが AUTO である場合は、最後に受信したテキストの自動判定結果を反映します。)

テキストの文字コードを変換しないで、そのまま送信する場合は、SendBinary ()メソッドにて、バイナリデータとして送信します。

13. シリアル通信

ＣＯＭポート，メールスロット、あるいは、ソケット(TCP/IP，クライアント／サーバ)による通信制御モジュールです。
ソケット(TCP/IP，サーバ)では、１つのクライアントにだけ接続可能です。

シリアル通信では、定義名やＡＰ名がソケット(TCP/IP)サーバ機能や、クライアント機能と似通っています。
以下のように「#include <AjrCstXX.h>」の前に「#define」を定義すると、他の定義名やＡＰ名を抑止できます。
（誤って、ソケット(TCP/IP)サーバ機能やクライアント機能の定義名やＡＰ名を使用するとコンパイルエラーとなります）

```
#define AJCSOCKSERV_H_       // ソケット(TCP/IP)サーバ機能を無効化
#define AJCSOCKCLIENT_H_     // ソケット(TCP/IP)クライアント機能を無効化
#include <AjrCstXX.h>
```

13.1. イベントの通知方法

イベントの通知方法は、以下の２つから選択できます。

・ユーザが、その場でイベントの発生を待ち受ける
・ユーザのウインドへ、ウインドメッセージとして通知する

13.1.1. ユーザが その場でイベントの発生を待ち受ける場合

ユーザが、その場でイベントの発生を待ち受ける場合（定期的にイベント発生をチェックする場合）は、AjcScpSetMode()の hWndNtc 引数に NULL を、WndMsgNtc 引数に０を指定します。
イベントを待ち受けるには、AjcScpWaitEvent()を実行します。
イベントデータを取得するには、AjcScpWaitEvent()で取得した lParam を引数として、AjcScpGetEventData()を実行します。

イベントの発生を待ち受ける場合のプログラムコードスタイルは、およそ以下のようになります。（ＣＯＭポート(COM1)での通信例）

```
#define  COM1  1

AjcMain (int argc, UTP argv[])
{
    HAJCSCP     hScp;
    WPARAM      wParam;
    LPARAM      lParam;
    VOP         pDat;
    UI          len, param;

    hScp = AjcScpCreate();                                  //  シリアル通信インスタンス生成

    AjcScpSetMode(hScp, NULL, 0, AJCSCP_CM_BIN);            //  実行モード設定
    AjcScpSetEvtMask(hScp, AJCSCP_EV_RXCHUNK | ・・・・・);  //  使用するイベント設定

    AjcScpOpen(hScp, COM1, 115200, 8, 'N', 1 );             //  ＣＯＭポートオープン

    ・・・・・

    while (・・・)  {

        ・・・・・

        if (AjcScpWaitEvent(hScp, &wParam, &lParam, 200)) {     //  イベント待ち(200ms)，イベントあり？
            AjcScpGetEventData(hScp, lParam, &pDat, &len, &param);  //    イベントデータ取得
            switch (wParam) {
                case AJCSCP_EV_RXCHUNK:
                    ・・・ チャンクデータ受信時の処理 ・・・
                    break;
                ・・・その他のイベント処理・・・
            }
            AjcScpRelEventData(hScp, lParam);                   //    イベントデータ開放
        }

        ・・・・・

    }

    ・・・・・

    AjcScpClose (hScp);  // 通信ポートクローズ
    AjcScpDelete(hScp);  // シリアル通信インスタンス消去
}
```

13.1.2. ユーザのウインドへ、ウインドメッセージとして通知する場合

ユーザのウインドへ、ウインドメッセージとしてイベントを通知する場合は、AjcScpSetMode()の hWndNtc 引数でウインドハンドルを、WndMsgNtc 引数でウインド・メッセージコードを指定します。

ウインドメッセージの wParam にはイベントコードが、lParam にはイベント情報が設定されます。

イベントデータを取得するには、lParam を引数として、AjcScpGetEventData()を実行します。

ウインドメッセージとして通知する場合のプログラムコードスタイルは、およそ以下のようになります。

```
#define  WM_SCPEVENT  (WM_APP + 100)

HSCP  hScp;

//----- ダイアログ初期化 ----------------------------------------------------------------------------//
AJC_DLGPROC(Main, WM_INITDIALOG )
{
    hScp = AjcScpCreate();                                          // シリアル通信インスタンス生成
    AjcScpSetMode     (hScp, hDlg, WM_SCPEVENT, AJCSCP_CM_BIN);     // 実行モード設定
    AjcScpSetEvtMask (hScp, AJCSCP_EV_RXCHUNK |・・・・・);         // 使用するイベント指定
    AjcScpOpenDefault(hScp);                                        // 通信ポートオープン（通信リソースはダイアログで選択可）
    ・・・・
    return TRUE;
}
//----- シリアル通信イベント通知 --------------------------------------------------------------------//
AJC_DLGPROC(Main, WM_SCPEVENT )
{
    VOP  pDat;
    UI   len, param;

    AjcScpGetEventData(hScp, lParam, &pDat, &len, &param);        // イベントデータ取得
    switch (wParam) {
        case AJCSCP_EV_RXCHUNK:
            ・・・ チャンクデータ受信時の処理 ・・・
            break;
        ・・・・・  その他のイベント処理 ・・・・・
    }

    AjcScpRelEventData(hScp, lParam);                             // イベントデータ開放
    return TRUE;
}
//----- ウインド破棄 --------------------------------------------------------------------------------//
AJC_DLGPROC(Main, WM_DESTROY )
{
    AjcScpClose (hScp);  // 通信ポートクローズ
    AjcScpDelete(hScp);  // シリアル通信インスタンス消去
    PostQuitMessage(0);  // プログラム終了
    return TRUE;
}
//----- 通信パラメタ設定ボタン----------------------------------------------------------------------//
AJC_DLGPROC(Main, IDC_CMD_SETPARAM )
{
    if (HIWORD(wParam) == BN_CLICKED) {
        // ダイアログによるシリアル通信リソースの選択や，通信パラメタ設定
        AjcScpDlgParamEasy(hScp, hDlg);
    }
    return TRUE;
}
//----- キャンセル ----------------------------------------------------------------------------------//
AJC_DLGPROC(Main, IDCANCEL )
{
    // 通信ポートクローズ
    AjcScpClose(hScp);
    // シリアル通信インスタンス消去
    AjcScpDelete(hScp);
    return TRUE;
}

・・・・・その他のウインドメッセージ処理 ・・・・・

//-------------------------------------------------------------------------------------------------//
AJC_DLGMAP_DEF(Main)
    AJC_DLGMAP_MSG(Main, WM_INITDIALOG         )
    AJC_DLGMAP_MSG(Main, WM_SCPEVENT           )
    AJC_DLGMAP_MSG(Main, WM_DESTROY)
    AJC_DLGMAP_CMD(Main, IDC_CMD_SETPARAM      )
    AJC_DLGMAP_CMD(Main, IDCANCEL  )
    ・・・・・
AJC_DLGMAP_END
```

13.2. ＤＣＢとタイムアウト情報

本ライブラリでは、WindowsAPI を直接コールして、ＣＯＭポートのアクセスを行っていますが、この際に DCB とタイムアウト情報(COMMTIMEOUTS) で通信用のパラメタを設定します。

DCB と COMMTIMEOUTS 構造体の内容は、以下のとおりです。(「デフォルト」は、本ライブラリで設定するデフォルト値です)
(但し、これらの情報に関する明確な資料が無いため、以下の記述では、私の予想によるものも含まれます。正確な情報に関しては、ＭＳＤＮやＳＤＫを探して見てください)

ＤＣＢ

#	メンバ名	内容	デフォルト
1	DCBlength	ＤＣＢ構造体のサイズ (バイト数)	sizeof(DCB)
2	BaudRate	通信速度[bps]	9600
3	fBinary	ＥＯＦチェックしない (但し、現状は無効で、常に「1」を設定する)	1
4	fParity	パリティビットの有無 (0：なし, 1：あり)	0
5	fOutxCtsFlow	ＣＴＳ信号による送信フロー制御 (0：無効, 1：有効)	0
6	fOutxDsrFlow	ＤＳＲ信号による送信フロー制御 (0：無効, 1：有効)	0
7	fDtrControl	ＤＴＲ信号による受信フロー制御 (0：無効, 1：有効)	0
8	fDsrSensitivity	ＤＳＲ検知 (0：DSR-OFF 時でも送信を行う, 1：DSR-OFF 時は送信しない)	0
9	fTXContinueOnXoff	XOFF 送信後の動作 (0：データ送信を停止, 1：データ送信を継続)	0
10	fOutX	XON/XOFF によるソフトウェア送信フロー制御 (0：無効, 1：有効)	0
11	fInX	XON/XOFF によるソフトウェア受信フロー制御 (0：無効, 1：有効)	0
12	fErrorChar	受信エラー時の動作 (0：データ置換しない, 1：受信バイトを ErrorChar で置換)	0
13	fNull	NULL 文字(0x00)受信した場合の動作 (0：受信データとして採用, 1：破棄する)	0
14	fRtsControl	ＲＴＳ信号による受信フロー制御 (0：無効, 1：有効)	0
15	fAbortOnError	通信エラー発生時の動作 (0：送受信動作を継続, 1：送受信動作を停止)	0
16	XonLim	受信フロー制御における低位バッファ容量	512
17	XoffLim	受信フロー制御における高位バッファ容量	1536
18	ByteSize	データビット数 (4～8bit)	8
19	Parity	パリティビット・タイプ (0:None, 1:Odd, 2:Even, 3:Mark, 4:Space)	0
20	StopBits	ストップビット数 (0:1bit, 1:1.5bit, 2:2bit)	1
21	XonChar	ソフトウェア送信フロー制御における XON 制御コード	0x11
22	XoffChar	ソフトウェア送信フロー制御における XOFF 制御コード	0x13
23	ErrorChar	受信エラー時に置換するバイトデータ	0x3F
24	EofChar	ファイル終端文字コード (fBinary=1 の場合は無効)	0x1A
25	EvtChar	フラグバイト受信通知イベントを発生させるバイトデータ	0x03

ＣＯＭＭＴＩＭＥＯＵＴＳ

#	メンバ名	内容	デフォルト
1	ReadIntervalTimeout	受信時バイト間タイムアウト	0
2	ReadTotalTimeoutMultiplier	受信時タイムアウト時間 (可変部 (受信バイト数を乗算))	0
3	ReadTotalTimeoutConstant	〃 (定数部)	30
4	WriteTotalTimeoutMultiplier	送信時タイムアウト時間 (可変部 (送信バイト数を乗算))	0
5	WriteTotalTimeoutConstant	〃 (定数部)	0

各値とも、0はタイムアウトなしを意味する

13.3. イベント一覧

イベントの一覧を以下に示します。

#	イベントコード (wParam)	内容	イベントデータ (AjcScpGetEventData()で取得する情報)		
			pDat	lDat	param
1	AJCSCP_EV_PORTSTATE	ポート状態通知	ポート名テキストのアドレス (AJCSCP_PORTCHG :の場合は、変化前のポート名)	ポート名のバイト数/文字数	AJCSCP_CLOSED : クローズ／切断 状態 AJCSCP_OPENED : オープン／接続 状態 AJCSCP_OPENFAIL: オープン／接続 失敗 AJCSCP_PORTCHG : 通信リソース変化 AJCSCP_MYSLOTFAIL:自スロット生成失敗 AJCSCP_TXFAILURE :送信失敗 AJCSCP_SRVSTART : サーバ開始 AJCSCP_SRVSTOP : サーバ停止 AJCSCP_SRVSRTFAIL : サーバ開始失敗
2	AJCSCP_EV_RXCHUNK (※1)	チャンクデータ受信通知	受信データのアドレス	受信データのバイト数／文字数	0 : バイナリデータ(RXCHUNK 時のみ) 1 : バイト文字列 2 : ワイド文字列(UNICODE)
3	AJCSCP_EV_RXTEXT (※2)	テキスト受信通知			
4	AJCSCP_EV_RXESC	ＥＳＣコード受信通知			
5	AJCSCP_EV_RXCTRL	制御コード受信通知			
6	AJCSCP_EV_RXPKT	パケットデータ受信通知		パケットデータのバイト数	0 (バイナリデータ)
7	AJCSCP_EV_TXEMPTY	送信完了通知	NULL	－	0 (未使用)
8	AJCSCP_EV_RXNOPKT	パケット外テキスト通知	受信データのアドレス	受信データのバイト数／文字数	1 : バイト文字列 2 : ワイド文字列(UNICODE)
9	AJCSCP_EV_ERR	通信エラー発生通知	NULL	－	HIWORD:通信エラー要因 (※3)
10	AJCSCP_EV_RING	RING 変化通知	NULL	－	LOWORD:信号状態 (※4)
11	AJCSCP_EV_RLSD	RLSD 変化通知	NULL	－	
12	AJCSCP_EV_DSR	DSR 変化通知	NULL	－	
13	AJCSCP_EV_CTS	CTS 変化通知	NULL	－	
14	AJCSCP_EV_RXWORD14	バイトペアによるワード(14Bit)受信通知	受信ワードデータのアドレス	2	0: High byte first 1: Low byte first
15	AJCSCP_EV_SSEP	#3～#6 の合成	-	-	-
16	AJCSCP_EV_DEFAULT_EVT	#3～#6, #13,#14 の合成	-	-	-
17	AJCSCP_EV_DEFAULT_POST	#1, #3～#6, #13,#14 の合成	-	-	-

メールスロットやソケット通信の場合は、網掛け部分のイベントは発生しません。

※1 : テキストチャンクの場合、AjcScpSetRxTextCode()により設定された文字コードからシフト JIS/UNICODE に変換したテキストを通知します。
※2 : AjcScpSetRxTextCode()により設定された文字コードからシフト JIS/UNICODE に変換したテキストを通知します。

※3 : 通信エラー要因 (以下の合成値)

#	エラーコード	内容	
1	AJCSCP_CE_BREAK	ブレーク信号を検出した	ＣＯＭポート通信時のエラー
2	AJCSCP_CE_FRAME	フレーミングエラーを検出した	
3	AJCSCP_CE_OVERRUN	オーバーランエラーを検出した	
4	AJCSCP_CE_RXPARITY	パリティエラーを検出した	
5	AJCSCP_CE_IOE	デバイスアクセス中に I/O エラーを検出した	
6	AJCSCP_CE_RXOVER	受信キューオーバフロー	
7	AJCSCP_CE_TXFULL	送信キュー満杯	
8	AJCSCP_CE_RXERR	受信エラー (ReadFile() / recv()でエラー)	
9	AJCSCP_CE_TXERR	送信エラー (WriteFile() / send()でエラー	

※4 : 信号状態 (以下の合成値)

#	信号状態名	内容
1	AJCSCP_CTS	ＣＴＳ信号アクティブ
2	AJCSCP_DSR	ＤＳＲ信号アクティブ
3	AJCSCP_RING	ＲＩＮＧ信号アクティブ
4	AJCSCP_RLSD	ＲＬＳＤ信号アクティブ

13.4. COM ポート通信

ＣＯＭポートで通信するには、以下のＡＰＩによりインスタンス生成／消去，初期設定，ポートのオープン／クローズを行います。

#	ＡＰＩ	内容	備考
1	AjcScpCreate	インスタンス生成	
2	AjcScpSetMode	初期設定（モード設定）	
3	AjcScpSetEvtMask	初期設定（使用するイベント設定）	
4	**AjcScpOpen**	ＣＯＭポートオープン	
5	**AjcScmOpenEx**	ＣＯＭポートオープン（ＤＣＢ、タイムアウト指定）	
6	AjcScpClose	ＣＯＭポートクローズ	
7	AjcScpDelete	インスタンス消去	

サンプルコードは「基本サンプルコード（SW_ScpComC（COM ポート通信，　コンソールアプリ））」を参照

13.5. メールスロット通信

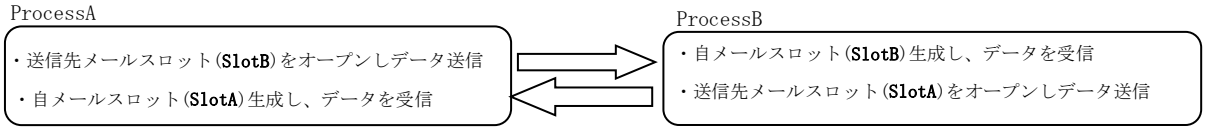
１つのコンピュータで２つのプログラムが通信したり、ＬＡＮに接続されている２つのコンピュータ間で通信します。
（※ Windows11 以降では、２つのコンピュータ間で通信することはできません）

メールスロットで通信するには、以下のＡＰＩによりインスタンス生成／消去，初期設定，ポートのオープン／クローズ等を行います。

#	ＡＰＩ	内容	備考
1	AjcScpCreate	インスタンス生成	
2	**AjcScpCreateSlot**	インスタンス生成（自メールスロット指定）	
3	AjcScpCreateSlotEx	インスタンス生成（自メールスロットやタイムアウト等指定）	
4	AjcScpSetMode	初期設定（モード設定）	
5	AjcScpSetEvtMask	初期設定（使用するイベント設定）	
7	**AjcScpOpenSlot**	メールスロットポートオープン	
8	AjcScpClose	メールスロットクローズ	
9	AjcScpDelete	インスタンス消去	
10	AjcScpCreateMySlot	自メールスロットの生成	
11	AjcScpDeleteMySlot	自メールスロットの破棄	
12	AjcScpMySlotIsCreated	自メールスロット生成状態の取得	

メールスロット通信は、同一コンピュータ内で２つのプロセス間でデータ転送を行うものです。（Windows10 以前は２つのコンピュータ間でも通信可能だったが、Windows11 以降は、同一コンピュータ内での２つのプロセス間での通信のみ可能）

各プロセスでは、受信用のメールスロット（自メールスロット）を生成し、互いの自メールスロットを指定し送信用メールスロットをオープンします。
メールスロットによる通信は、１方向のみの通信となりますが、互いに自メールスロットを開設（生成）することにより、２つのメールスロット回線により、お互いに送受信が可能となります。



メールスロット通信の場合、AjcScpCreateSlot()で、自メールスロットの生成も行うようにします。

```
HAJCSCP hScp = AjcScpCreateSlot(NULL, TRUE, TEXT("MySlotName"), NULL, NULL);
```

２つのプロセスで互いに 自メールスロットの生成を行った後に、AjcScpOpenSlot()で送信用メールスロットをオープンします。

```
HAJCSCP hScp = AjcScpCreateSlot(NULL, TRUE, TEXT("RemoteSlotName"), NULL, NULL);
```

サンプルコードは「基本サンプルコード（SW_ScpComC（SW_ScpMailSLotC（メールスロット通信，　コンソールアプリ））」を参照

13.6. ソケット（TCP/IP クライアント）通信

ソケット（TCP/IP）のクライアント側として動作します。
ソケット（クライアント）で通信するには、以下のＡＰＩによりインスタンス生成／消去，初期設定，ポートのオープン／クローズを行います。

#	ＡＰＩ	内容	備考
1	AjcScpCreate	インスタンス生成	
2	AjcScpSetMode	初期設定（モード設定）	
3	AjcScpSetEvtMask	初期設定（使用するイベント設定）	
4	**AjcScpOpenSock**	ソケット（クライアント）オープン	
5	AjcScpClose	ソケット（クライアント）クローズ	サーバとは、切断される
6	AjcScpDelete	インスタンス消去	

AjcScpOpenSock()は、サーバコンピュータ名とポート番号を指定してサーバとの接続を試みます。

AjcScpOpenSock()を実行しても、サーバとの接続が完了するまでは通信できません。
サーバとの接続完了は、以下のずれかの方法で検知することができます。

・サーバ接続イベント「AJCSCP_EV_PORTSTATE (param= AJCSCP_OPENED)」の発生を待つ
・適当な周期でAjcScpIsOpened() を実行し、戻り値=TRUEとなるのを待つ

サーバとの接続が失敗した場合は、AJCSCP_EV_PORTSTATE (param= AJCSCP_OPENFAIL) イベントが発生します。

サンプルコードは「基本サンプルコード (SW_ScpClientC（ソケット(クライアント)通信， コンソールアプリ))」を参照

13.7. ソケット（TCP/IP）サーバ通信

ソケット（TCP/IP）のサーバ側として動作します。
ソケット（サーバ）で通信するには、以下のＡＰＩによりインスタンス生成／消去，初期設定，ポートのオープン／クローズを行います。

#	ＡＰＩ	内容	備考
1	AjcScpCreate	インスタンス生成	
2	AjcScpSetMode	初期設定（モード設定）	
3	AjcScpSetEvtMask	初期設定（使用するイベント設定）	
4	**AjcScpOpenServ**	サーバを開始する	
5	AjcScpClose	サーバを停止する	クライアントは、切断される
6	AjcScpDelete	インスタンス消去	

AjcScpOpenServ()は、ポート番号を指定してサーバとして動作します。
接続できるクライアント数は「1」固定です。(複数のクライアントと接続する場合は、「ソケット(TCP/IP)サーバ機能」を参照)

AjcScpOpenServ()を実行しても、クライアントが接続するまでは通信できません。
クライアントとの接続完了は、以下のずれかの方法で検知することができます。

・クライアント接続イベント「AJCSCP_EV_PORTSTATE (param= AJCSCP_OPENED)」の発生を待つ
・適当な周期でAjcScpIsOpened() を実行し、戻り値=TRUEとなるのを待つ

クライアントが切断された場合は、AJCSCP_EV_PORTSTATE (param= AJCSCP_CLOSED) イベントが発生し、サーバを停止します。
再び通信を行うには、AjcScpOpenServ()により、サーバを開始します。

サンプルコードは「基本サンプルコード (SW_ScpSErverC（ソケット(サーバ)通信， コンソールアプリ))」を参照

13.8. 各通信リソースを切り替えて通信

ＣＯＭポート通信／メールスロット／ソケット通信（クライアント／サーバ）は、随時切り替えることができます。
AjcScpDlgParamEasy()により、ダイアログによるポートの設定を行い、AjcScpOpenDefault()/AjcScpOpenSelect()により、ダイアログで設定されたＣＯＭポート／メールスロット／ソケットをオープンすることができます。
AjcScpDlgParamEasy()は、以下のようなダイアログによる設定を行います。

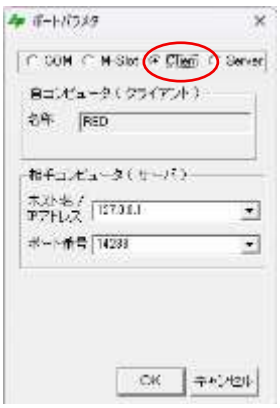
ＣＯＭポート通信



メールスロット通信



ソケット(TCP/IP クライアント)通信



ソケット(TCP/IP サーバ)通信



通信リソースを切り替えた場合や、設定内容を変更した場合は、設定前のリソースはクローズされます。
（ＣＯＭポートの場合、ポート番号以外の設定内容を変更した場合は、設定後の内容で再オープンします）

各通信リソースを切り替えて通信するには、以下のＡＰＩによりインスタンス生成／消去，初期設定，ポートのオープン／クローズを行います。

#	ＡＰＩ	内容	備考
1	AjcScpCreate	インスタンス生成	
2	AjcScpSetMode	初期設定（モード設定）	
3	AjcScpSetEvtMask	初期設定（使用するイベント設定）	
4	**AjcScpOpenDefault**	選択された通信リソースをオープン	
5	AjcScpClose	選択された通信リソースをクローズ	
6	AjcScpDelete	インスタンス消去	
7	AjcScpDlgParamEasy	通信リソースやパラメタの変更	ダイアログボックスによる設定
8	jcScpDlgParamDetail	ＣＯＭポートパラメタの詳細設定	

サンプルコードは「基本サンプルコード（SW_ScpAny（通信リソースを選択して通信））」を参照

13.9. サポートＡＰＩ

シリアル通信のサポートＡＰＩ一覧を以下に示します。

#	関数名	内容	スレッドセーフ
1	AjcScpCreate AjcScpCreatePf AjcScpCreateSlot[Ex]	インスタンス生成	×
2	AjcScpDelete	インスタンス消去	×
3	AjcScpSetMode	モード設定	○
4	AjcScpOpen	ＣＯＭポートオープン	○
5	AjcScpOpenEx	ポートオープン（ＣＯＭポート，ＤＣＢ指定）	○
6	AjcScpOpenSlot	ポートオープン（メールスロット）	○
7	AjcScpOpenSock	ポートオープン（ソケット通信，(TCP/IP クライアント)）	○
8	AjcScpOpenServ	ポートオープン（ソケット通信，(TCP/IP サーバ)）	○
9	AjcScpOpenDefault	ポートオープン（選択済通信リソース）	○
10	AjcScpOpenSelect	ポートオープン（通信リソース選択）	○
11	AjcScpClose	ポートクローズ	○
12	AjcScpServStop	サーバ停止	○
13	AjcScp{Set/Get}SelectedPort	通信ポート種別設定／取得	○
14	AjcScpIsOpened	ポートオープン状態取得	○
15	AjcScpSendChar	１文字送信	○
16	AjcScpSendWord14LF	バイトペアによるワード（14Bit）データ送信（Low byte first）	○
17	AjcScpSendWord14HF	バイトペアによるワード（14Bit）データ送信（High byte first）	○
18	AjcScpSetByteSeqRxWord14 AjcScpGetByteSeqRxWord14	14 ビットワードデータ（バイトペア）受信時のバイト順設定／取得	○
19	AjcScpSendText AjcScpSendTextF	テキストデータ送信	○
20	AjcScpSendBinData	バイナリデータ送信	○
21	AjcScpSendPacket	パケットデータ送信	○
22	AjcScpSendBreak	ブレーク信号送出／停止	○
23	AjcScpSetEvtMask AjcScpGetEvtMask	イベントマスク設定／取得	○
24	AjcScpSetDTR	ＤＴＲ信号設定	○
25	AjcScpSetRTS	ＲＴＳ信号設定	○
26	AjcScpGetSigState	信号状態取得	○
27	AjcScpGetTxBytes	送信待ちデータバイト数取得	○
28	AjcScpPurgeRecvData	全受信済データ破棄	○
29	AjcScpPurgeSendData	全送信待ちデータ破棄	○
30	AjcScpPurgeAllData	全送受信データ破棄	○
31	AjcScpWaitEvent	イベント発生待ち	○
32	AjcScpvGetIpAddrStr	クライアントのＩＰアドレス文字列取得	×
33	AjcScpGetEventData	イベントデータ取得	○
34	AjcScpRelEventData	イベントデータ開放	○
35	AjcScpSetPktCtrlCode AjcScpGetPktCtrlCode	パケットフレームを認識する為の制御コード設定／取得	○
36	AjcScpSetPktTimeout AjcScpGetPktTimeout	パケットフレーム受信タイムアウト値設定／取得	○

つづく

つづき

#	関数名	内容	スレッドセーフ
37	AjcScpDlgParamEasy	ダイアログによる通信パラメタ設定	○
38	AjcScpDlgParamDetail	ダイアログによるＣＯＭポート通信パラメタ詳細設定	○
39	AjcScp{Set/Get}GrayRbt	ダイアログのリ通信ソース選択を無効化／無効化状態の取得	○
40	AjcScp{Set/Get}GrayItem	ダイアログの各項目を無効化	○
41	AjcScp{Set/Get}Param	ＣＯＭポート情報（ＤＣＢとタイムアウト情報）設定／取得	○
42	AjcScp{Set/Get}SlotParam	メールスロット情報設定／取得	○
43	AjcScp{Set/Get}SockParam	ソケット情報設定／取得	○
44	AjcScp{Set/Get}ServParam	サーバ情報設定／取得	○
45	AjcScpEnumSerialPorts	ＣＯＭポートの列挙	×
46	AjcScpGetPortPathName	ポートパス名称取得	○
47	AjcScpGetPortName	ポート名称取得	○
48	AjcScpGetPortDevName	ＣＯＭポートのデバイス名取得	×
49	AjcScp{Set/Get}ChunkMode	チャンクデータの通知モード設定／取得	○
50	AjcScp{Set/Get}RxTextCode	受信テキストの文字コード種別設定／取得	○
51	AjcScp{Set/Get}TxTextCode	送信テキストの文字コード種別設定／取得	○
52	AjcScpGetActualRxTextCode AjcScpGetActualTxTextCode	実際の送受信文字コード種別取得	○
53	AjcScpCreateMySlot	自メールスロット生成	○
54	AjcScpDeleteMySlot	自メールスロット消去	○
55	AjcScpMySlotIsCreated	自メールスロット生成状態取得	○
56	AjcScpGetMySlotPathName	自メールスロットのパス名取得	○
57	AjcScpGetMyComputerName	自コンピュータ名取得	○
58	AjcScpSetMailSlotNames AjcScpGetMailSlotNames	メールスロット名情報設定／取得	○
59	AjcScpSetTxSpeedLimit AjcScpGetTxSpeedLimit	メールスロット送信制限速度の設定／取得	○
60	AjcScpSetRecognizeTxStopTime	送信停止の認識時間設定	○
61	AjcScpSetRecognizeRxStopTime	受信停止の認識時間設定	○

13.9.1. インスタンス生成（AjcScpCreate）

形　式 ： HAJCSCP　AjcScpCreate　(VO);
HAJCSCP　AjcScpCreatePf(C_UTP pSect);
// メールスロット通信向け
HAJCSCP　AjcScpCreateEx　(C_UTP pSect, BOOL fCreateMySlot, C_UTP pMySlot, C_UTP pRmtHost, C_UTP pRmtSlot); ※1
HAJCSCP　AjcScpCreateSlot　(C_UTP pSect, BOOL fCreateMySlot, C_UTP pMySlot, C_UTP pRmtHost, C_UTP pRmtSlot);
HAJCSCP　AjcScpCreateSlotEx(C_UTP pSect, BOOL fCreateMySlot, C_UTP pMySlot, C_UTP pRmtHost, C_UTP pRmtSlot,
UI nMaxMessageSize, UI lReadTimeout, LPSECURITY_ATTRIBUTES lpSecurityAttributes);

※１：非推奨（AjcScpCreateSlot()と同じ、互換性維持のため存在する）

引　数 ： pSect　- 設定値を格納するプロファイルセクション名のアドレス（NULL：プロファイルへ記録しない）
fCreateMySlot - 自メールスロットの生成フラグ(TRUE:生成する，FALSE:生成しない)
pMySlot　- 自メールスロット名へのポインタ　（NULL：既存の名称，空文字列：受信不可）
pRmtHost　- リモートコンピュータ名へのポインタ（NULL：既存の名称，空文字列：自コンピュータ）
pRmtSlot　- リモートスロット名へのポインタ　（NULL：既存の名称，空文字列：送信不可）
nMaxMessageSize　- メールスロット送信時の最大メッセージサイズ（０：任意）
lReadTimeout　- メールスロット読み出しタイムアウト[ms]（０：タイムアウトなし）
lpSecurityAttributes - メールスロットセキュリティ情報へのポインタ（NULL：セキュリティ指定なし）

説　明 ： シリアル通信機能のインスタンスを生成します。
この関数では、シリアル通信に関するイベントは、AjcScpWaitEvent()で受け取るように初期化されます。
また、受信チャンクデータ(AJCSCP_EV_CHUNK で通知されるデータ）はバイナリデータが通知されるように初期化されます。
シリアル通信に関するイベントをウインドへのメッセージで受け取る場合や、受信チャンクデータをテキストデータとして通知するようにするには、本ＡＰＩ実行直後に AjcScpSetMode()を実行してください。

pSect は、各パラメタの設定内容を記録するプロファイルセクションを指定します。
パラメタ（fCreateMySlot，nMaxMessageSize，lReadTimeout，lpSecurityAttributes を除く）の設定内容は、プロファイルセクションに記録され永続的に有効となります。
pSect=NULL とした場合は、パラメタをプロファイルセクションに記録しません。（永続化しません）
pSect 未指定のＡＰＩでは、pSect=" SerialComPort"を仮定します。

プログラムで複数のインスタンスを生成する場合は、pSect 引数で、各々異なるセクション名を指定してください。

fCreateMySlot は、自メールスロットの生成を行うか否かを指定します。
fCreateMySlot=TRUE とした場合は（pMySlot で有効なスロット名が指定されていれば）自メールスロットの生成を行い、
fCreateMySlot=FALSE とした場合は自メールスロットの生成を行いません。（後で、AjcScpCreateMySlot()で生成できます）
自メールスロットを生成しない場合は、メールスロットでの受信を行うことができません。

pMySlot，pRmtHost，pRmtSlot は、メールスロット通信を行う際の自メールスロット名，相手コンピュータ名，相手スロット名のデフォルト名称（pSect=NULL or pSect で指定したプロファイル無し時の名称）を指定します。
これらのデフォルト名称は、pMySlot=" MySlot"，pRmtHost=空文字列，pRmtSlot=" RmtSlot"です。

pRmtHost=空文字列("")を指定した場合は、送信先が自コンピュータとなります。
送信を行わない場合は、pRmtSlot=空文字列("")を指定します。

AjcScpCreateSlotEx()の nMaxMessageSize. lReadTimeout，lpSecurityAttributes は、内部で実行されるシステム API「CreateMailslot()」実行時の第２～４引数となります。（CreateMailslot()については MSDN 参照）
これらの引数が未指定の場合、nMaxMessageSize=0. lReadTimeout=0，lpSecurityAttributes=NULL を仮定します。

戻り値 ： ≠NULL - 成功（インスタンスハンドル）
＝NULL - 失敗

13.9.2. インスタンス消去（AjcScpDelete）

形　式 ： BOOL　AjcScpDelete(HAJCSCP hScp)

引　数 ： hScp　　－ インスタンスハンドル

説　明 ： 全てのリソースを開放し、シリアル通信機能のインスタンスを消去します。
オープン状態である場合は、クローズしてからインスタンスを消去します。
現在の設定内容は、プロファイルに記録されます。

戻り値 ： TRUE　－ 成功
FALSE － 失敗

13.9.3. 実行モード設定（AjcScpSetMode）

形　式 ： BOOL　AjcScpSetMode (HAJCSCP hScp, HWND hWndNtc, UI WndMsgNtc, AJCSCP_CHUNKMODE ChunkMode);

引　数 ： hScp　　－ インスタンスハンドル
hWndNtc　　－ イベントを通知するウインドのハンドル　　(AjcScpWaitEvent()を使用する場合はNULL)
WndMsgNtc　－ イベントを通知するウインドのメッセージ値（AjcScpWaitEvent()を使用する場合は０）
ChunkMode　－ 受信チャンクデータの扱い

説　明 ： シリアル通信機能の実行モードを設定します。
この関数は、AjcScpCreate()実行直後に１度だけ実行してください。

シリアル通信に関するイベントをウインドメッセージで受け取る場合は、hWndNtc に通知するウインドのハンドル, WndMsgNtc にウインドメッセージコード(WM_USER+100 以降, WM_APP+500 以降か、RegisterWindowMessage()で取得したコード)を指定します。
このウインドメッセージが通知された場合、wParam にイベントコードが,lParam にイベントデータ取得情報が設定されます。実際に、イベントデータを取得するには、lParam を指定して、AjcScpGetEventData()を実行します。

シリアル通信に関するイベントを AjcScpWaitEvent()で待つ場合は、hWndNtc=NULL, WndMsgNtc=0 とします。

ChunkMode は、受信チャンクデータ(AJCSCP_EV_CHUNK で通知されるデータ）をどのように通知するかを指定します。

ChunkMode	内容
AJCSCP_CM_NONE	受信チャンクデータを通知しません
AJCSCP_CM_BIN	受信チャンクデータをバイナリデータとして通知します（デフォルト）
AJCSCP_CM_TEXT	受信チャンクデータをテキストデータとして通知します
AJCSCP_CM_BOTH	受信チャンクデータをバイナリデータ＆テキストデータとして通知します (AJCSCP_EV_RXCHUNK イベントで、同じ受信チャンクデータが param=0(バイナリデータ)と、 param=1/2(テキストデータ)として２回通知されます)

イベントマスクは、以下のように初期化されます。(変更する場合は、AjcScpSetEvtMask()を実行してください)

イベント名	内容	hWndNtc ≠ NULL	hWndNtc ＝ NULL
AJCSCP_EV_PORTSTATE	ポート状態通知	○	×
AJCSCP_EV_RXTEXT	テキスト受信通知	○	○
AJCSCP_EV_RXESC	ＥＳＣコード受信通知	○	○
AJCSCP_EV_RXCTRL	制御コード受信通知	○	○
AJCSCP_EV_RXPKT	パケットデータ受信通知	○	○
AJCSCP_EV_DSR	DSR 変化通知	○	○
AJCSCP_EV_CTS	CTS 変化通知	○	○
その他	―	×	×

戻り値 ： TRUE　－ 成功
FALSE － 失敗

13.9.4. COM ポートオープン［ポートパラメタ指定］（AjcScpOpen）

形　式 ： BOOL　AjcScpOpen(HAJCSCP hScp, UI Port, UI Rate, UI DataBits, UI Parity, UI StopBit);

引　数 ： hScp　　- インスタンスハンドル
Port　　- ＣＯＭポート番号（１～２５５　　，０を指定した場合は、現在値を使用）
Rate　　- 通信速度[bps]　（１～　　，０を指定した場合は、現在値を使用）
DataBits　- データビット数　（７～８　　，０を指定した場合は、現在値を使用）
Parity　　- パリティ指定　　('N':パリティなし,'O':奇数パリティ,'E':偶数パリティ,０を指定した場合は、現在値を使用）
StopBit　　- ストップビット長（１～２　　，０を指定した場合は、現在値を使用）

説　明 ： 指定した通信パラメタでＣＯＭポートをオープン（ファイルハンドルを生成）します。
既にＣＯＭポートがオープン状態である場合は、クローズしてからオープンします。

ＣＯＭポートのオープンが成功すると AJCSCP_EV_PORTSTATE(param=AJCSCP_OPENED)イベントを発生します。
ＣＯＭポートのオープンを失敗した場合は、AJCSCP_EV_PORTSTATE(param=AJCSCP_OPENFAIL)イベントを発生します。

Port～StopBit で０を指定した項目については、現在値が有効となります。
現在値とは、AjcScpSetParam()/AjcScpDlgParamEasy()/AjcScpDlgParamDetail()で設定された値を意味します

戻り値 ： TRUE　- 成功
FALSE - 失敗

13.9.5. COM ポートオープン［DCB で詳細指定］（AjcScpOpenEx）

形　式 ： BOOL　AjcScpOpenEx (HAJCSCP hScp, UI Port, LPDCB pDcb, LPCOMMTIMEOUTS pTmo);

引　数 ： hScp　　- インスタンスハンドル
Port　　- ＣＯＭポート番号（１～２５５　　，０を指定した場合は、現在値を使用）
pDcb　　- ＤＣＢ情報のアドレス　　（ＮＵＬＬを指定した場合は、現在値を使用）
pTmo　　- タイムアウト情報のアドレス（ＮＵＬＬを指定した場合は、現在値を使用）

説　明 ： AjcScpOpen()と同様にＣＯＭポートをオープンします。
詳細な通信パラメタ（ＤＣＢ，タイムアウト情報）を指定できます。

Port=0, pDcb=NULL, pTmo=NULL とした場合は、各々、現在値が有効となります。

戻り値 ： TRUE　- 成功
FALSE - 失敗

13.9.6. メールスロット オープン（AjcScpOpenSlot）

形　式 ： BOOL　AjcScpOpenSlot (HAJCSCP hScp, C_UTP pRmtHost, C_UTP pRmtSlot);

引　数 ： hScp　　- インスタンスハンドル
pRmtHost　- リモートコンピュータ名　(NULL：既存の名称を使用，自コンピュータとする場合は空文字列)
pRmtSlot　- リモートスロット名　　(NULL：既存の名称を使用，送信を行わない場合は空文字列)

説　明 ： 送信先メールスロットをオープンし、送受信の準備を行います。
pRmtHost, pRmtSlot は、各々通信相手のコンピュータ名とスロット名を指定します。
既にオープン状態である場合は、クローズしてからオープンし直します。
pRmtHost=空文字列("")を指定した場合は、自コンピュータ内のスロットと通信します。
pRmtSlot=空文字列("")を指定した場合は、送信を行うことはできません。
（送信先メールスロットが存在しない場合、pRmtSlot=空文字列("")を指定し、受信専用としてオープン可能です）
尚、自メールスロットが未生成である場合は、(自メールスロットが空文字列でなければ)自メールスロットの生成も行います。

送信先スロットのオープンが成功すると AJCSCP_EV_PORTSTATE(param=AJCSCP_OPENED)イベントを発生します。
送信先スロットのオープンを失敗した場合は、AJCSCP_EV_PORTSTATE(param=AJCSCP_OPENFAIL)イベントを発生します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

13.9.7. ソケット（TCP/IP クライアント）通信 オープン（AjcScpOpenSock）

形　式 ： BOOL　AjcScpOpenSock(HAJCSCP hScp, C_UTP pServ, UI PortNo);

引　数 ： hScp　　- インスタンスハンドル
pServ　　- サーバコンピュータ名／IP アドレス（設定済名称を使用する場合は NULL）
PortNo　　- サーバのポート番号　　（設定済ポート番号を使用する場合は０）

説　明 ： ソケット（クライアント）通信用のオープン（ソケットの生成とサーバ接続準備）を行います。
pServ, PortNo は、各々通信相手のサーバコンピュータ名と TCP/IP ポート番号を指定します。
既にオープン状態である場合は、クローズしてからオープンし直します。
サーバと通信するには、サーバとの接続を待たなければなりません。

サーバとの接続が成功すると、AJCSCP_EV_PORTSTATE(param=AJCSCP_OPENED)イベントを発生します。
オープン失敗時は、AJCSCP_EV_PORTSTATE(param=AJCSCP_OPENFAIL)イベントを発生します。

サーバから切断された場合は、AJCSCP_EV_PORTSTATE(param=AJCSCP_CLOSED)イベントを発生します。

戻り値 ： TRUE　- 成功（オープン成功（ソケットの生成を成功しサーバとの接続待ち））
FALSE - 失敗（オープン失敗（ソケットの生成失敗、あるいは、サーバとの接続不可）

13.9.8. ソケット（TCP/IP サーバ）通信 オープン（AjcScpOpenServ）

形　式 ： BOOL　AjcScpOpenServ(HAJCSCP hScp, UI PortNo);

引　数 ： hScp　　- インスタンスハンドル
PortNo　　- サーバのポート番号（設定済ポート番号を使用する場合は０）

説　明 ： ソケット（サーバ）通信用のオープン（サーバ処理の起動とクライアントソケットの生成）を行います。
PortNo は、TCP/IP ポート番号を指定します。
既にオープン状態である場合は、クローズしてからオープンし直します。
クライアントと通信するには、クライアントからの接続を待たなければなりません。

オープンが成功した場合は、SCP_EV_PORTSTATE(param=AJCSCP_SRVSTART)イベントを発生します。
オープン失敗時は、AJCSCP_EV_PORTSTATE(param=AJCSCP_OPENFAIL)イベントを発生します。

クライアントが接続されると、AJCSCP_EV_PORTSTATE(param=AJCSCP_OPENED)イベントを発生します。

クライアントが切断された場合は、AJCSCP_EV_PORTSTATE(param=AJCSCP_CLOSED)イベントを発生します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

13.9.9. ポートオープン［選択済通信リソース］（AjcScpOpenDefault）

形　式 ： BOOL AjcScpOpenDefault (HAJCSCP hScp);

引　数 ： hScp　　- インスタンスハンドル

説　明 ： 現在設定されているポート番号と通信パラメタ／メールスロット／ソケット情報で、当該リソースをオープンします。
このＡＰＩは、各通信リソースにより、以下のＡＰＩが実行されます。

#	通信リソース	実行されるＡＰＩ	備考
1	ＣＯＭポート	AjcScpOpenEx(pW, pW->PortNo, NULL, NULL);	
2	メールスロット	AjcScpOpenSlot(pW, NULL, NULL);	
3	ソケット（クライアント）	AjcScpOpenSock(pW, NULL, 0);	
4	ソケット（サーバ）	AjcScpOpenServ(pW, 0);	

戻り値 ： TRUE - 成功
FALSE - 失敗

13.9.10. ポートートオープン［通信リソース選択］（AjcScpOpenSelect）

形　式 ： BOOL AjcScpOpenSelect (HAJCSCP hScp, AJCSCP_PORTSEL sel);

引　数 ： hScp　　- インスタンスハンドル
sel　　- 最初にオープンする通信デバイスの選択
・AJCSCP_SEL_COMPORT - ＣＯＭポート
・AJCSCP_SEL_MAILSLOT - メールスロット
・AJCSCP_SEL_SOCKET - ソケット（クライアント）

説　明 ： 通信リソースを指定して、通信回線をオープンします。
その後は、AjcScpOpenDefault()と同じです。

戻り値 ： TRUE - 成功
FALSE - 失敗

13.9.11. ポートクローズ（AjcScpClose）

形　式 ： BOOL AjcScpClose(HAJCSCP hScp);

引　数 ： hScp　　- インスタンスハンドル

説　明 ： 現在選択されている通信リソースクローズし、通信不能にします。
このＡＰＩは、各通信リソースにより、以下の操作が実行されます。

#	通信リソース	内容	備考
1	ＣＯＭポート	ＣＯＭポートハンドルをクローズ	
2	メールスロット	送受信メールスロットをクローズ	送信用スロットクローズ 他コンピュータとの通信時は、自スロットもクローズ(※１)
3	ソケット（クライアント）	ソケットハンドルをクローズ	サーバとの接続を切断
4	ソケット（サーバ）	サーバを停止	クライアントとの接続を切断

※1：同一ＰＣ間でのプロセス通信である場合は、自メールスロットを破棄しません。（再オープン時、相手との通信を継続可能とする為）
この場合、AjcScpClose()でポートをクローズした場合でも、相手がデータを送信すると、（内部的に）自プロセスでデータの受信が発生します。
そこで、AjcScpClose()でポートがクローズされている場合は、受信データを空読みし、破棄します。

戻り値 ： TRUE - 成功
FALSE - 失敗

13.9.12. サーバ停止（AjcScpServStopt）

形　式 ： BOOL　AjcScpServStop(HAJCSCP hScp)；

引　数 ： hScp　　- インスタンスハンドル

説　明 ： 通信リソースがソケット(TCP/IP サーバ)の場合、サーバを停止（サーバのソケットハンドルをクローズ）します。
実際のサーバの停止は、AJCSCP_SRVSTOP イベントで通知します。

戻り値 ： TRUE　- 成功
FALSE - 失敗／サーバ未起動

備　考 ： AjcScpOpenServ()でサーバが起動されたら、AjcScpClose()を実行してもサーバは起動状態のままです。
但し、通信リソースをサーバから、他のリソースに変更された場合は、サーバを停止します。

13.9.13. 通信ポート種別設定／取得（AjcScp{Set/Get}SelectedPort）

形　式 ： UI　AjcScpSetSelectedPort(HAJCSCP hScp, UI sel)；　--- 設定
UI　AjcScpGetSelectedPort(HAJCSCP hScp)；---------- 取得

引　数 ： hScp　　- インスタンスハンドル
sel　　- 設定する通信ポート種別（ AJCSCP_PS_COM / AJCSCP_PS_SLOT / AJCSCP_PS_CLIENT / AJCSCP_PS_SERVER)

説　明 ： ポート設定ダイアログで選択されている 通信ポート種別を設定／取得します。

戻り値 ： ≠0 - 設定されている通信ポート種別　1～255　　- ＣＯＭ１～２５５
AJCSCP_PS_SLOT(=256)　- メールスロット
AJCSCP_PS_CLIENT(=257) - ソケット（クライアント）
AJCSCP_PS_SERVER(=258) - ソケット（サー）
＝0 - エラー

13.9.14. ＣＯＭポートオープン状態取得（AjcScpIsOpened）

形　式 ： UI　　AjcScpIsOpened　(HAJCSCP hScp)；
BOOL　AjcScpIsTxOpened(HAJCSCP hScp)；
BOOL　AjcScpIsRxOpened(HAJCSCP hScp)；

引　数 ： hScp　　- インスタンスハンドル

説　明 ： 通信ポートのオープン状態を取得します。
AjcScpIsOpened()は、COM ポートがオープン状態である場合は、設定されているポート番号（１～２５５）を返します。
メールスロットがオープン状態である場合は、２５６を返します。
ソケット(TCP/IP クライアント）がオープン状態である場合は２５７を返します。
メールスロットの場合は、単に AjcScpOpen[*XXX*]()が実行されているか否かを示し、送受信が可能か否かを取得するには AjcScpIsTxOpened()や AjcScpIsRxOpened()を実行します。
メールスロット以外の場合は、送受信が可能か否かを返します。。
通信ポートがクローズ状態である場合は、０を返します。

AjcScpIsTxOpened()は、送信可能状態である場合は TRUE を、送信不能状態である場合は FALSE を返します。
AjcScpIsRxOpened()は、受信可能状態である場合は TRUE を、受信不能状態である場合は FALSE を返します。

戻り値 ： ≠０ - オープン状態（1～255:ＣＯＭポート番号, 256:メールスロットモード, 257：ソケットクライアント, 258:ソケットサーバ）
＝０ - クローズ状態（ソケット（サーバ）の場合はサーバと未接続状態を含む）　／　エラー

13.9.15. 1 文字送信（AjcScpSendChar）

形　式　：　BOOL　AjcScpSendChar(HAJCSCP hScp, UT code);

引　数　：　hScp　　- インスタンスハンドル
code　　- 送信するバイトデータ／文字コード（バイトモード時は 0x00～0xFF, UNICODE モード時は UTF-16）

説　明　：　指定されたバイトデータ／UTF-16 文字を送信します。
バイト文字モード(/DUNICODE オプション無し)の場合、単に 1 バイトのデータを送信します。
UNICODE モード(/DUNICODE オプション有り)の場合は、送信文字(UTF-16)を AjcScpSetTxTextCode()により指定された送信文字コードに変換したテキストを送信します。
この関数では、送信用のスプールバッファに送信データを格納するだけであり、すぐに制御を戻します。

戻り値　：　TRUE　- 成功
FALSE - 失敗

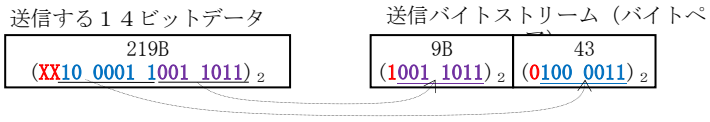
13.9.16. バイトペアによるワード（14Bit）データ送信 (Low byte first)（AjcScpSendWord14LF）

形　式　：　BOOL　AjcScpSendWord14LF(HAJCSCP hScp, UI data);

引　数　：　hScp　　- インスタンスハンドル
data　　- 送信する１４ビットデータ

説　明　：　指定されたデータの下位１４ビットをバイトペア（１バイト目は下位バイト、２バイト目は上位バイト）として送信します。
バイトペアとは、送信データの下位１４ビットを７ビットづつのバイトに分けて、１バイト目の MSB=1, ２バイト目の MSB=0 とした２バイトをを意味します。
１バイト目は下位７ビット、２バイト目は上位７ビットを送信します。

例えば、data=219B を送信した場合、バイトストリーム「9B 43」を送信します。



この関数では、送信用のスプールバッファに送信データを格納するだけであり、すぐに制御を戻します。

戻り値　：　TRUE　- 成功
FALSE - 失敗

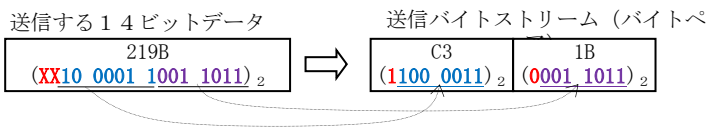
13.9.17. バイトペアによるワード（14Bit）データ送信 (High byte first)（AjcScpSendWord14HF）

形　式　：　BOOL　AjcScpSendWord14HF(HAJCSCP hScp, UI data);

引　数　：　hScp　　- インスタンスハンドル
data　　- 送信する１４ビットデータ

説　明　：　指定されたデータの下位１４ビットをバイトペア（１バイト目は上位バイト、２バイト目は下位バイト）として送信します。
バイトペアとは、送信データの下位１４ビットを７ビットづつのバイトに分けて、１バイト目の MSB=1, ２バイト目の MSB=0 とした２バイトを意味します。
１バイト目は上位７ビット、２バイト目は下位７ビットを送信します。

例えば、data=219B を送信した場合、バイトストリーム「C3　1B」を送信します。



この関数では、送信用のスプールバッファに送信データを格納するだけであり、すぐに制御を戻します。

戻り値　：　TRUE　- 成功
FALSE - 失敗

13.9.18. １４ビットデータのバイト順設定／取得（AjcScp{Set/Get}ByteSeqRxWord14）

形　式 ： BOOL　AjcScpSetByteSeqRxWord14 (HAJCSCP hScp, BOOL fLowByteFirst); - バイト順設定
BOOL　AjcScpGetByteSeqRxWord14 (HAJCSCP hScp); ---------- バイト順取得

引　数 ： hScp　　- インスタンスハンドル
fLowByteFirst - TRUE　：下位バイト，上位バイトの順で受信（リトルエンディアン，デフォルト）
FALSE ：上位バイト，下位バイトの順で受信（ビッグエンディアン）

説　明 ： １４ビットワードデータ（バイトペア）受信時のバイト順を設定／取得します。

戻り値 ： 設定時：TRUE　- 成功　　取得時：TRUE　- 下位バイト，上位バイトの順で受信（リトルエンディアン）
FALSE - 失敗　　FALSE - 上位バイト，下位バイトの順で受信（ビッグエンディアン）

13.9.19. テキストデータ送信（AjcScpSendText）

形　式 ： BOOL　AjcScpSendText (HAJCSCP hScp, C_UTP pTxt, UI lTxt);　--- テキスト送信
BOOL　AjcScpSendTextF(HAJCSCP hScp, C_UTP pFmt, ...); --------- 書式テキスト送信

引　数 ： hScp　　- インスタンスハンドル
pTxt　　- 送信するテキストデータのアドレス
lTxt　　- 送信するテキストデータの文字数（－１の場合は自動算出）
pFmt　　- 書式テキスト（printf()と同じ）

説　明 ： テキストデータの送信を行います。
AjcScpSetTxTextCode()により指定された送信文字コードに変換したテキストを送信します。

この関数では、送信用のスプールバッファに送信データを格納するだけであり、すぐに制御を戻します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

13.9.20. バイナリデータ送信（AjcScpSendBinData）

形　式 ： BOOL　AjcScpSendBinData(HAJCSCP hScp, C_VOP pDat, UI lDat);

引　数 ： hScp　　- インスタンスハンドル
pDat　　- 送信するバイナリデータのアドレス
lDat　　- 送信するバイナリデータのバイト数

説　明 ： バイナリデータの送信を行います。
pDat と lDat で指定したバイトストリームをそのまま送信します。

この関数では、送信用のスプールバッファに送信データを格納するだけであり、すぐに制御を戻します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

13.9.21. パケットデータ送信（AjcScpSendPacket）

形　式 ： UI　AjcScpSendPacket(HAJCSCP hScp, C_VOP pPkt, UI lPkt);

引　数 ： hScp　　- インスタンスハンドル
pPkt　　- 送信するパケットデータのアドレス（空パケット送信時はNULL）
lPkt　　- 送信するパケットデータのバイト数（空パケット送信時は０）

説　明 ： 指定されたパケットデータをパケットフレームに乗せて送信します。
つまり、パケットデータ中のDLEを２つのDLEに変換し、先頭にDLE・STXを、末尾にDLE・ETXを付加したデータを送信します。
pPkt=NULL, lPkt=0を指定した場合は空パケット（DLE,STX,DLE,ETXの４バイト）を送信します。
この関数では、送信用のスプールバッファに送信データを格納するだけであり、すぐに制御を戻します。

戻り値 ： 4～　- 成功（DLE・STX, DLE・ETXや、パケットデータ中の透過制御バイト(DLE)を含めた実際の送信バイト数）
0　　- 失敗

13.9.22. ブレーク信号送出／停止（AjcScpSendBreak）

形　式 ： BOOL　AjcScpSendBreak（HAJCSCP hScp, BOOL fBreak);

引　数 ： hScp　　- インスタンスハンドル
fBreak　　- TRUE:ブレーク送出開始, FALSE:ブレーク送出停止

説　明 ： ブレーク信号の送出を開始、あるいは、停止します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

13.9.23. イベントマスク設定／取得（AjcScp{Set/Get}EvtMask）

形　式 ： BOOL　AjcScpSetEvtMask（HAJCSCP hScp, UI Mask); - イベントマスク設定
UI　　AjcScpGetEvtMask（HAJCSCP hScp); ---------- イベントマスク取得

引　数 ： hScp　　- インスタンスハンドル
Mask　　- 設定するイベントマスク（イベントコードの合成値）

説　明 ： イベントマスク（使用するイベント）を設定／取得します。
イベントマスクのデフォルトは、AJCSCP_EV_DEFAULT_EVTです
この関数は、通信ポートがクローズ状態の時に実行してください。
イベントコードについては、本節の冒頭を参照してください。

戻り値 ： 設定時：TRUE　- 成功　　取得時：イベントマスク値（イベントコードの合成値),
FALSE - 失敗　　エラーの場合は０を返します。

13.9.24. ＤＴＲ信号設定（AjcScpSetDTR）

形　式 ： BOOL　AjcScpSetDTR（HAJCSCP hScp, BOOL fActive);

引　数 ： hScp　　- インスタンスハンドル
fActive　　- TRUE:DTRを有効状態に設定, FALSE:DTRを無効状態に設定

説　明 ： ＤＴＲ信号を有効／無効状態に設定します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

13.9.25. ＲＴＳ信号設定（AjcScpSetRTS）

形　式 ： BOOL　AjcScpSetRTS (HAJCSCP hScp, BOOL fActive);

引　数 ： hScp　　- インスタンスハンドル
fActive　　- TRUE:RTS を有効状態に設定, FALSE:RTS を無効状態に設定

説　明 ： ＲＴＳ信号を有効／無効状態に設定します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

13.9.26. 入力信号（DSR, CTS, RING, RLSD）状態取得（AjcScpGetSigState）

形　式 ： UI　AjcScpGetSigState (HAJCSCP hScp);

引　数 ： hScp　　- インスタンスハンドル

説　明 ： 入力信号（DSR, CTS, RING, RLSD）の状態を取得します。
各信号の記号名称については、冒頭の「イベントコード一覧」を参照してください。

戻り値 ： 入力信号状態（各信号とも、１＝有効状態, ０＝無効状態), エラーの場合は０を返します

13.9.27. 送信待ちデータバイト数取得（AjcScpGetTxBytes）

形　式 ： ULL　AjcScpGetTxBytes (HAJCSCP hScp);

引　数 ： hScp　　- インスタンスハンドル

説　明 ： 送信待ちデータの総バイト数を取得します。

戻り値 ： 送信待ちデータバイト数（エラーの場合は０を返します)

13.9.28. 全受信済データ破棄（AjcScpPurgeRecvData）

形　式 ： BOOL　AjcScpPurgeRecvData (HAJCSCP hScp);

引　数 ： hScp　　- インスタンスハンドル

説　明 ： 受信中のテキスト／ＥＳＣコード／制御コード／パケットデータ／パケット外データを全て破棄します。
受信チャンクデータについては、受信時にリアルタイムに通知される為、破棄対象外です。

戻り値 ： TRUE　- 成功
FALSE - 失敗

13.9.29. 全送信待ちデータ破棄（AjcScpPurgeSendData）

形　式 ： BOOL　AjcScpPurgeSendData (HAJCSCP hScp);

引　数 ： hScp　　- インスタンスハンドル

説　明 ： 送信待ちデータを全て破棄します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

13.9.30. 全受信済データと全送信待ちデータ破棄（AjcScpPurgeAllData）

形　式 ： BOOL　AjcScpPurgeAllData (HAJCSCP hScp);

引　数 ： hScp　　- インスタンスハンドル

説　明 ： 全受信済データと全送信データを破棄します。
AjcScpPurgeRecvData()と AjcScpPurgeSendData()を実行します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

13.9.31. イベント発生待ち（AjcScpWaitEvent）

形　式 ： BOOL　AjcScpWaitEvent (HAJCSCP hScp, WPARAM *pwParam, LPARAM *plParam, UI msTime);

引　数 ： hScp　　- インスタンスハンドル
pwParam　　- 発生したイベントコード格納するバッファのアドレス
plParam　　- イベントデータ取得情報を格納するバッファのアドレス
msTime　　- 待ち時間[ms]（－１を指定した場合は、永久にイベント待ちとなる）

説　明 ： シリアル通信に関するイベントの発生を待ちます。
本関数によりイベントの発生を待つには、AjcScpSetMode()で、hWndNtc 引数に NULL を指定していなければなりません。

pwParam で指定したバッファには、発生したイベントコード（合成値）が格納されます。
イベントコードについては、本節冒頭の「イベントコード一覧」を参照してください。

plParam で指定したバッファには、イベントデータ取得情報が設定されます。
この情報は、AjcScpGetEventData()の lParam 引数に指定します。

戻り値 ： TRUE　- イベントが発生した
FALSE - タイムアウト（イベント未発生）／エラー

13.9.32. クライアントのＩＰアドレス文字列取得（AjcScpvGetIpAddrStr）

形　式 ： C_UTP　AjcScpGetIpAddrStr (HAJCSCP hScp, UTP pBuf, UI lBuf);

引　数 ： hScp　　- インスタンスハンドル
pBuf　　- ＩＰアドレス文字列を格納するバッファのアドレス
lPkt　　- ＩＰアドレス文字列を格納するバッファの文字数

説　明 ： クライアントのＩＰアドレス文字列を作成します。

戻り値 ： ≠ NULL ： 作成したＩＰアドレス文字列へのポインタ（=pBuf）
＝ NULL ： 失敗

13.9.33. イベントデータ取得（AjcScpGetEventData）

形　式 ： BOOL　AjcScpGetEventData (HAJCSCP hScp, LPARAM lParam, VOP *ppDat, UIP plDat, UIP pParam);

引　数 ： hScp　　- インスタンスハンドル
lParam　　- イベントデータ取得情報
ppDat　　- 受信データへのポインタを格納するバッファのアドレス
plDat　　- 受信データのバイト数／文字数を格納するバッファのアドレス
pParam　　- パラメタ情報を格納するバッファのアドレス

説　明 ： 発生したイベントに関する付随情報を取得します。

lParam は、ウインドメッセージの lParam／AjcScpWaitEvent()で取得したイベントデータ取得情報を指定します。

ppDat～pParam で取得される情報については、本節冒頭の「イベントコード一覧」を参照してください。

本関数は、シリアルポートに関するイベントが発生した場合、必ず実行してください。
また、イベントで通知された情報を使用した後には、AjcScpRelEventData()を実行しなければなりません。

戻り値 ： TRUE　- 成功
FALSE - 失敗

13.9.34. イベントデータ開放（AjcScpRelEventData）

形　式 ： BOOL　AjcScpRelEventData (HAJCSCP hScp, LPARAM lParam);

引　数 ： hScp　　- インスタンスハンドル
lParam　　- イベントデータ取得情報

説　明 ： イベントデータを開放します。
lParam は、ウインドメッセージの lParam／AjcScpWaitEvent()で取得したイベントデータ取得情報を指定します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

13.9.35. パケットフレームを認識する為の制御コード設定／取得（AjcScp{Set/Get}PktCtrlCode）

形　式 ： BOOL　AjcScpSetPktCtrlCode (HAJCSCP hScp, UI　stx, UI　etx, UI　dle); - パケット制御コード設定
BOOL　AjcScpGetPktCtrlCode (HAJCSCP hScp, UIP pStx, UIP pEtx, UIP pDle); - パケット制御コード取得

引　数 ： hScp　　- インスタンスハンドル
stx　　- STX のコード値（設定しない場合は、0 を指定）
etx　　- ETX のコード値（設定しない場合は、0 を指定）
dle　　- DLE のコード値（設定しない場合は、0 を指定）
pStx　　- STX のコード値を格納するバッファのアドレス（不要時はＮＵＬＬ）
pEtx　　- ETX のコード値を格納するバッファのアドレス（不要時はＮＵＬＬ）
pDle　　- DLE のコード値を格納するバッファのアドレス（不要時はＮＵＬＬ）

説　明 ： パケットフレームを認識する為の制御コード（STX, ETX, DLE）の値を設定／取得します。
デフォルトでは、STX=0x02, ETX=0x03, DLE=0x10 となっています。

戻り値 ： TRUE　- 成功
FALSE - 失敗

13.9.36. パケットフレーム受信タイムアウト値設定／取得（AjcScp{Set/Get}PktTimeout）

形　式 ： BOOL　AjcScpSetPktTimeout (HAJCSCP hScp, UI　 msTime)；- パケット受信タイムアウト設定
BOOL　AjcScpGetPktTimeout (HAJCSCP hScp, UIP pMsTime)；- パケット受信タイムアウト取得

引　数 ： hScp　　- インスタンスハンドル
msTime　　- タイムアウト値[ms]（０：タイムアウトなし）
pMsTime　　- タイムアウト値[ms]を格納するバッファのアドレス（不要時はＮＵＬＬ）

説　明 ： パケットフレーム受信時タイムアウト値を設定します。
パケットフレーム受信時において、指定時間内にパケット受信が完了しない場合、当該パケットを破棄します。
デフォルトでは、３０００ｍｓに設定されています。

戻り値 ： TRUE　- 成功
FALSE - 失敗

13.9.37. ダイアログによる通信パラメタ設定（AjcScpDlgParamEasy）

形　式　：　UI　AjcScpDlgParamEasy　(HAJCSCP hScp, HWND hWndOwner);
UI　AjcScpDlgParamEasyEx (HAJCSCP hScp, HWND hWndOwner, int x, int y);

引　数　：　hScp　　－ インスタンスハンドル
hWndOwner　－ オーナーウインドハンドル（不要時は NULL）
x, y　　－ ダイアログ表示位置（スクリーン座標）

説　明　：　下図に示すダイアログボックスにより通信パラメタを設定します。

ＯＫボタンを押すと設定を反映します。キャンセルボタンを押すと設定を中止します。
詳細設定ボタンを押すと、ＣＯＭポートの詳細な設定となります。



戻り値　：　≠ 0 －　以下のビットの組み合わせ

- AJCSCPDGR_PORTCHANGED　－ COM ポート変更
- AJCSCPDGR_DCBCHANGED　－ ＤＣＢ変更
- AJCSCPDGR_TMOCHANGED　－ タイマ変更
- AJCSCPDGR_SELCHANGED　－ ＣＯＭポート／メールスロット／クライアント／サーバの選択変更
- AJCSCPDGR _OKBUTTON　－ ＯＫボタンによる終了
- AJCSCPDGR_SLOTCHANGED　－ メールスロット情報変更
- AJCSCPDGR_SERVCHANGED　－ サーバ情報変更

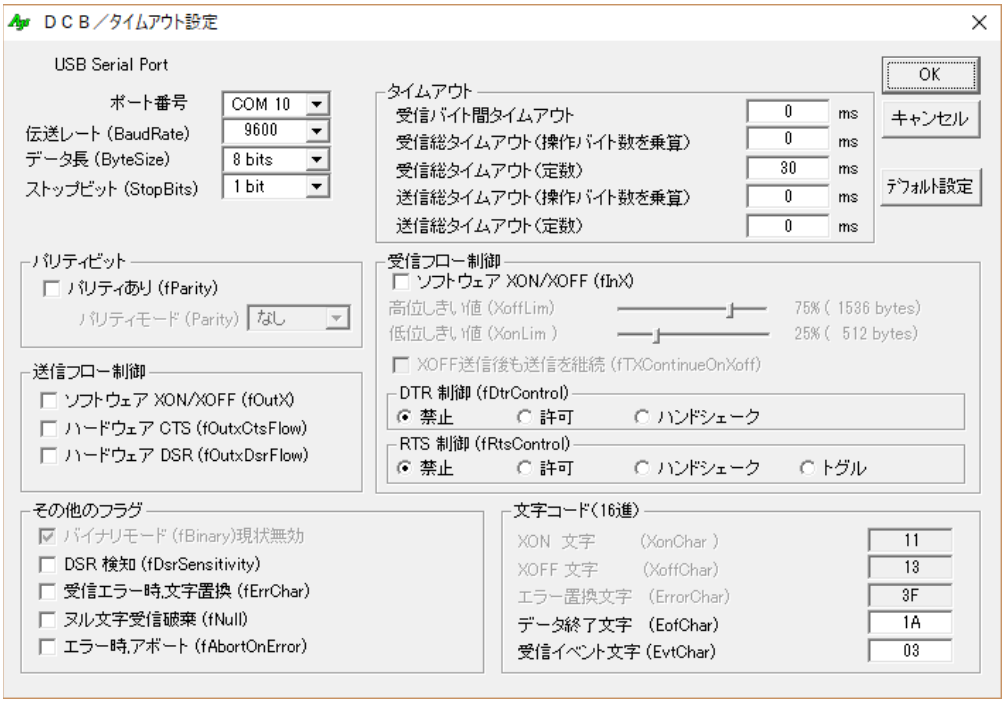
＝ 0 － キャンセルボタン押下／エラー

13.9.38. ダイアログによる通信パラメタ設定［詳細］（AjcScpDlgParamDetail）

形　式　：　UI　AjcScpDlgParamDetail　(HAJCSCP hScp, HWND hWndOwner);
UI　AjcScpDlgParamDetailEx(HAJCSCP hScp, HWND hWndOwner, int x, int y);

引　数　：　hScp　　－ インスタンスハンドル
hWndOwner　－ オーナーウインドハンドル（不要時は NULL）
x, y　　－ ダイアログ表示位置（スクリーン座標）

説　明　：　下図に示すダイアログボックスによりシリアルポート番号、通信パラメタと、タイムアウト情報を設定します。



上記表示例は、（本ライブラリでの）デフォルト値を表示したものです。
このダイアログでは、シリアル通信に関する全てのパラメタ，タイムアウト情報を設定できます。

「ＯＫ」ボタンを押すと設定を反映します。
「キャンセル」ボタンを押すと設定を中止します。
「デフォルト設定」を押すと、全てのパラメタをデフォルト値にします。

尚、オープン状態で、ポート番号を変更した場合は、新たなポート番号でポートのオープンを行います。

戻り値　：　≠ 0 － 以下のビットの組み合わせ
AJCSCPDGR_PORTCHANGED　－ ポート変更
AJCSCPDGR_DCBCHANGED　－ ＤＣＢ変更
AJCSCPDGR_TMOCHANGED　－ タイマ変更
AJCSCPDGR_OKBUTTON　－ ＯＫボタンによる終了
＝ 0 － キャンセルボタン押下／エラー

備　考　：　ポートがオープン状態で、通信パラメタが変更された場合は、ポートをクローズ後に、新たなパラメタで再オープンします。

13.9.39. ダイアログ・通信リソースの選択を無効化（AjcScp{Set/Get}GrayRbt）

形　式　：　BOOL　AjcScpSetGrayRbt (HAJCSCP hScp, UI mskRbt);　-- 無効化項目のマスク設定
UI　　AjcScpGetGrayRbt (HAJCSCP hScp); ---------- 無効化項目のマスク取得

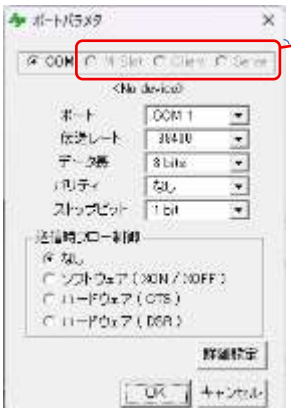
引　数　：　hScp　- インスタンスハンドル
mskRbt - 無効化項目のマスク

説　明　：　AjcScpDlgParamEasy()による通信パラメタの設定で、ＣＯＭポート／メールスロット／ソケット／サーバ選択のラジオボタンを無効化／無効化状態を取得します。
mskRbt には、以下のシンボルの合成値を指定します。

#	シンボル	内容
1	AJCSCPRBT_ComPort	ＣＯＭポートを選択不可にする
2	AJCSCPRBT_MailSlot	メールスロットを選択不可にする
3	AJCSCPRBT_Client	ソケット（クライアント）を選択不可にする
4	AJCSCPRBT_Server	ソケット（サーバ）を選択不可にする

例：　以下のコード例は、ＣＯＭポートだけを選択可能にします。(他の通信リソースを選択不能にします)

```
AjcScpSetGrayRbt(hScp, AJCOPT3(AJCSCPRBT_, MailSlot, Client, Server));
AjcScpDlgParamEasy(hScp, hDlg);
```



戻り値　：　AjcScpSetGrayRbt　　AjcScpGetGrayRbt
TRUE　– 成功　　各項目の無効化状態
FALSE – 失敗

13.9.40. ダイアログの各項目を無効化（AjcScp{Set/Get}GrayItem）

形　式 ： BOOL　AjcScpSetGrayItem (HAJCSCP hScp, ULL mskItem)； -- 無効化項目のマスク設定
ULL　AjcScpGetGrayItem (HAJCSCP hScp)； --------------- 無効化項目のマスク取得

引　数 ： hScp　　- インスタンスハンドル
mskItem - 無効化項目のマスク

説　明 ： AjcScpDlgParamEasy[Ex]()／AjcScpDlgParam Detail [Ex]()による通信パラメタの設定で、ダイアログ中の各項目を無効化します。
mskItemには、以下のシンボルの合成値を指定します。

●ＣＯＭポートの項目

#	シンボル	内容
1	AJCSCPGRY_Port	ポート番号
2	AJCSCPGRY_BaudRate	ボーレート（通信速度）
3	AJCSCPGRY_ByteSize	バイトサイズ（ビット数）
4	AJCSCPGRY_StopBits	ストップビット
5	AJCSCPGRY_fParity	パリティ有無
6	AJCSCPGRY_Parity	パリティ種別
7	AJCSCPGRY_Paritys	上記２つの合成値
8	AJCSCPGRY_fOutX	XON/XOFF 送信フロー
9	AJCSCPGRY_fOutxCtsFlow	CTS 送信フロー
10	AJCSCPGRY_fOutxDsrFlow	DSR 送信フロー
11	AJCSCPGRY_TxFlowControl	上記３つの合成値
12	AJCSCPGRY_fInX	XON/XOFF 受信フロー
13	AJCSCPGRY_XoffLim	受信フロー制御における低位バッファ容量
14	AJCSCPGRY_XonLim	受信フロー制御における高位バッファ容量
15	AJCSCPGRY_fTXContinueOnXoff	XOFF 送信後の動作（０：データ送信を停止，１：データ送信を継続）
16	AJCSCPGRY_RxFlowControl	上記４つの合成値
17	AJCSCPGRY_DtrControl	ＤＴＲ信号による受信フロー制御
18	AJCSCPGRY_RtsControl	ＲＴＳ信号による受信フロー制御
19	AJCSCPGRY_fBinary	ＥＯＦチェックしない（但し、現状は無効で、常に「１」を設定する）
20	AJCSCPGRY_fDsrSensitivity	ＤＳＲ検知（０：DSR-OFF 時でも送信を行う，１：DSR-OFF 時は送信しない）
21	AJCSCPGRY_fErrorChar	受信エラー時の動作（０：データ置換しない，１：受信バイトを ErrorChar で置換）
22	AJCSCPGRY_fNull	NULL 文字(0x00)受信した場合の動作（０：受信データとして採用，１：破棄する）
23	AJCSCPGRY_fAbortOnError	通信エラー発生時の動作（０：送受信動作を継続，１：送受信動作を停止）
24	AJCSCPGRY_Flags	上記５つの合成値
25	AJCSCPGRY_XonChar	ソフトウェア送信フロー制御における XON 制御コード
26	AJCSCPGRY_XoffChar	ソフトウェア送信フロー制御における XOFF 制御コード
27	AJCSCPGRY_ErrorChar	受信エラー時に置換するバイトデータ
28	AJCSCPGRY_EofChar	ファイル終端文字コード（fBinary=1 の場合は無効）
29	AJCSCPGRY_EvtChar	フラグバイト受信通知イベントを発生させるバイトデータ
30	AJCSCPGRY_Characters	上記５つの合成値
31	AJCSCPGRY_ReadIntervalTimeout	受信時バイト間タイムアウト
32	AJCSCPGRY_ReadTotalTimeoutMultiplier	受信時タイムアウト時間（可変部（受信バイト数を乗算））
33	AJCSCPGRY_ReadTotalTimeoutConstant	受信時タイムアウト時間（定数部）
34	AJCSCPGRY_WriteTotalTimeoutMultiplier	送信時タイムアウト時間（可変部（送信バイト数を乗算））
35	AJCSCPGRY_WriteTotalTimeoutConstant	送信時タイムアウト時間（定数部）
36	AJCSCPGRY_Timeouts	上記５つの合成値
37	AJCSCPGRY_SetDetail	詳細設定ボタン

●メールスロットの項目

#	シンボル	内容
1	AJCSCPGRY_SLT_MySlot	自スロット名
2	AJCSCPGRY_SLT_MySlotOpen	自コンピュータ・スロット オープン／クローズボタン
3	AJCSCPGRY_SLT_RmyHost	相手ホスト名
4	AJCSCPGRY_SLT_RmtSlot	相手スロット名
5	AJCSCPGRY_SLT_ChkTxDelay	送信速度制限チェックボックス
6	AJCSCPGRY_SLT_SpdTxDelay	送信速度制限 速度値
7	AJCSCPGRY_SLT_TxDelay	上記２つの合成値
8	AJCSCPGRY_SLT_All	上記全ての合成値

●ソケット（クライアント）の項目

#	シンボル	内容
1	AJCSCPGRY_CLI_ServName	サーバホスト名
2	AJCSCPGRY_CLI_PortNo	ポート番号
3	AJCSCPGRY_CLI_All	上記全ての合成値

●ソケット（サーバ）の項目

#	シンボル	内容
1	AJCSCPGRY_SRV_PortNo	ポート番号

戻り値 ： AjcScpSetGrayItem　　AjcScpGetGrayItem
TRUE – 成功　　各項目の無効化状態
FALSE – 失敗

13.9.41. ＣＯＭポート情報（ＤＣＢとタイムアウト情報）設定／取得（AjcScp{Set/Get}Param）

形　式 ： BOOL　AjcScpSetParam (HAJCSCP hScp, const DCB *pDcb, const COMMTIMEOUTS *pTmo); - DCB とタイムアウト情報設定
UI　　AjcScpGetParam (HAJCSCP hScp, LPDCB pDcb　　, LPCOMMTIMEOUTS　　pTmo); - DCB とタイムアウト情報取得

引　数 ： hScp　　- インスタンスハンドル
設定時
pDcb　　- 設定するポートパラメタのアドレス　（設定しない場合は NULL）
pTmo　　- 設定するタイムアウト情報のアドレス（設定しない場合は NULL）
取得時
pDcb　　- ポートパラメタを格納するバッファのアドレス　（不要時は NULL）
pTmo　　- タイムアウト情報を格納するバッファのアドレス（不要時は NULL）

説　明 ： ＣＯＭポートのパラメタ (DCB)と、タイムアウト情報を設定／取得します。
AjcScpDlgParamEasy[Ex]()/AjcScpDlgParamDetail[Ex]で表示するダイアログのＣＯＭポート初期値となります。

戻り値 ： 設定時：TRUE　- 成功　　取得時：　≠ 0 - 設定されている通信ポート種別情報
FALSE - 失敗　　1～255 – COM1～COM255, 256:M-Slot, 257:Socket, 258:Server
＝ 0 - エラー

13.9.42. メールスロット情報設定／取得（AjcScp{Set/Get}SlotParam）

形　式 ： BOOL　AjcScpSetSlotParam (HAJCSCP hScp, PCAJCSCP_SLOTINFO pSlotInfo); - 設定
UI　　AjcScpGetSlotParam (HAJCSCP hScp, PAJCSCP_SLOTINFO　pSlotInfo); - 取得

引　数 ： hScp　　- インスタンスハンドル
設定時
pSlotInfo　　- 設定するメールスロット情報のアドレス　（設定しない場合は NULL）
取得時
pSlotInfo　　- メールスロット情報を格納するバッファのアドレス　（不要時は NULL）

説　明 ： メールスロット情報を設定／取得します。
AjcScpDlgParamEasy[Ex]()で表示するダイアログのメールスロット初期値となります。
pSlotInfo の内容は以下のとおりです。

#	メンバ	タイプ	内容
1	MyHostName	UT[64]	ＣＯＭポートを選択不可にする
2	MySlotName	UT[64]	メールスロットを選択不可にする
3	RmtHostName	UT[64]	ソケット（クライアント）を選択不可にする
4	RmtSlotName	UT[64]	ソケット（サーバ）を選択不可にする
5	fTxDelay	BOOL	送信速度制限フラグ
6	TxDelayBPS	UI	送信制限速度[KBytes / sec]

戻り値 ： 設定時：TRUE　- 成功　　取得時：　≠ 0 - 設定されている通信ポート種別情報
FALSE - 失敗　　1～255 – COM1～COM255, 256:M-Slot, 257:Socket, 258:Server
＝ 0 - エラー

13.9.43. ソケット情報設定／取得（AjcScp{Set/Get}SockParam）

形　式 ： BOOL　AjcScpSetSockParam (HAJCSCP hScp, PCAJCSCP_SOCKINFO pSockInfo); - 設定
UI　　AjcScpGetSockParam (HAJCSCP hScp, PAJCSCP_SOCKINFO　pSockInfo); - 取得

引　数 ： hScp　　- インスタンスハンドル
設定時
pSockInfo　- 設定するソケット（クライアント）情報のアドレス　（設定しない場合は NULL）
取得時
pSockInfo　- ソケット（クライアント）情報を格納するバッファのアドレス　（不要時は NULL）

説　明 ： ソケット（クライアント）情報を設定／取得します。
AjcScpDlgParamEasy[Ex]()で表示するダイアログのソケット（クライアント）初期値となります。
pSockInfo の内容は以下のとおりです。

#	メンバ	タイプ	内容
1	ServName	UT[64]	サーバホスト名
2	PortNo	UI	ポート番号

戻り値 ： 設定時：TRUE　- 成功　　取得時：　≠ 0 - 設定されている通信ポート種別情報
FALSE - 失敗　　1～255 – COM1～COM255, 256:M-Slot, 257:Socket, 258:Server
＝ 0 - エラー

13.9.44. サーバ情報設定／取得（AjcScp{Set/Get}ServParam）

形　式 ： BOOL　AjcScpSetServParam (HAJCSCP hScp, PCAJCSCP_SERVINFO pServInfo); - 設定
UI　　AjcScpGetServParam (HAJCSCP hScp, PAJCSCP_SERVINFO　pServInfo); - 取得

引　数 ： hScp　　- インスタンスハンドル
設定時
pServInfo　- 設定するサーバ情報のアドレス　（設定しない場合は NULL）
取得時
pServInfo　- サーバ情報を格納するバッファのアドレス　（不要時は NULL）

説　明 ： サーバ情報を設定／取得します。
AjcScpDlgParamEasy[Ex]()で表示するダイアログのサーバ初期値となります。
pServInfo の内容は以下のとおりです。

#	メンバ	タイプ	内容
1	PortNo	UW	ポート番号

戻り値 ： 設定時：TRUE　- 成功　　取得時：　≠ 0 - 設定されている通信ポート種別情報
FALSE - 失敗　　1～255 – COM1～COM255, 256:M-Slot, 257:Socket, 258:Server
＝ 0 - エラー

13.9.45. 利用可能なシリアルポート番号の列挙（AjcScpEnumSerialPorts）

形　式　：　UI　AjcScpEnumSerialPorts(UBP pMap, UI lMap, UI CurrentPort);

引　数　：　pMap　　- 列挙したシリアルポート番号を格納するバッファのアドレス（バイト配列のアドレス）
lMap　　- 列挙したシリアルポート番号を格納するバッファのバイト数（＝最大列挙数）
CurrentPort - 無条件に列挙するポート番号（０を指定した場合は無効）

説　明　：　利用可能なシリアルポートの番号（１～２５５）を列挙します。
CurrentPort に０以外を指定した場合は、当該シリアルポート番号を無条件に列挙します。
pMap で示されるバッファには、CurrentPort で指定されたポート番号と、利用可能なシリアルポートの番号が、番号の若い順に設定されます。

戻り値　：　列挙したシリアルポート番号の個数

13.9.46. 通信ポートのパス名を取得（AjcScpGetPortPathName）

形　式　：　UTP　AjcScpGetPortPathName(HAJCSCP hScp);

引　数　：　hScp　　- インスタンスハンドル

説　明　：　現在設定されている通信リソースのパス名を取得します。
各通信リソースで、以下の文字列を返します

#	通信リソース	パス名文字列	備考
1	ＣＯＭポート(COM1 ～ COM9)	COM*n*	*n* = 1 ～ 9
	〃　　(COM10 以降)	¥¥.¥COM*n*	*n* = 10 ～ 255
2	メールスロット（自コンピュータ内）	¥¥.¥mailslot¥*name*	*name* ： スロット名
	〃　　（他コンピュータ）	¥¥*remotehost*¥mailslot¥*name*	*remotehos* ： コンピュータ名
3	ソケット(クライアント)	client¥*servname*:*n*	*servname*: サーバ名 / IP Address *n*　　: ポート番号
4	ソケット(サーバ)	server¥*n*	*n*　　: ポート番号

戻り値　：　≠NULL - 現在設定されているシリアルポートのパス名文字列へのポインタ
＝NULL - 失敗

注　意　：　戻り値が示す文字列は、一時的なバッファ領域です。他のＡＰＩを実行する前にこの文字列を引き取ってください。

13.9.47. ポート名称取得（AjcScpGetPortName）

形　式　：　UTP　AjcScpGetPortName(HAJCSCP hScp);

引　数　：　hScp　　- インスタンスハンドル

説　明　：　現在設定されている通信リソースの名称を取得します。
ＣＯＭポートが設定されている場合は、「COMn」 を返します。(n は１～２５５)
メールスロット通信の場合は、接続先メールスロットの名称(mailslot¥*name*)を返します。
ソケット(クライアント)の場合は、「*サーバ名*：ポート番号」を返します。
ソケット(サーバ)の場合は、「ポート番号」を返します。

戻り値　：　≠NULL - 現在設定されているシリアルポートのパス名文字列へのポインタ
＝NULL – 失敗

注　意　：　戻り値が示す文字列は、一時的なバッファ領域です。他のＡＰＩを実行する前にこの文字列を引き取ってください。

13.9.48. ＣＯＭポートのデバイス名称取得（AjcScpGetPortDevName）

形　式 ： int　AjcScpGetPortDevName(C_BCP pPortName, BCP pBuf, UI lBuf);

引　数 ： pPortName　－ ＣＯＭポート名（”COM1” ～ “COM255”）
pBuf　　－ デバイス名を格納するバッファのアドレス
lBuf　　－ デバイス名を格納するバッファの文字数

説　明 ： ＣＯＭポートのデバイス名を取得します。
デバイス名とは、デバイスマネージャ等で表示される名称を意味します。(Ex. “USB Serial Port”)
pBuf=NULL, lBuf=0 を指定した場合は、デバイス名を格納するのに必要なバッファの文字数を返します。

戻り値 ： pBuf != NULL && lBuf != 0 ： 格納したデバイス名の文字数（終端文字は含まない）
pBuf == NULL || lBuf == 0 ： デバイス名を格納するのに必要なバッファの文字数（終端文字を含む）
＝０　　：　当該ＣＯＭポート無し／エラー

13.9.49. チャンクデータの受信モード設定／取得（AjcScp{Set/Get}ChunkMode）

形　式 ： BOOL　AjcScpSetChunkMode (HAJCSCP hScp, AJCSCP_CHUNKMODE ChunkMode); - チャンクデータ通知モード設定
AJCSCP_CHUNKMODE AjcScpGetChunkMode (HAJCSCP hScp); ---------- チャンクデータ通知モード取得

引　数 ： hScp　　－ インスタンスハンドル
ChunkMode　－ チャンクデータの通知モード (AJCSCP_CM_NONE/BIN/TEXT/BOTH)

説　明 ： チャンクデータの通知モードを設定／取得します。

本ＡＰＩを実行した場合は、受信済のデータは、全て破棄されます。

戻り値 ： 設定時：TRUE　－ 成功　　取得時：チャンクデータの通知モード
FALSE － 失敗

13.9.50. 受信文字コード種別設定／取得（AjcScp{Set/Get}RxTextCode）

形　式 ： BOOL　　AjcScpSetRxTextCode　(HAJCSCP hScp, AJCSCP_TEXTCODE code); - 受信文字コード種別設定
AJCSCP_TEXTCODE　AjcScpGetRxTextCode　(HAJCSCP hScp); ---------- 受信文字コード種別取得

引　数 ： hScp　　－ インスタンスハンドル
code　　－ 受信テキストコード種別
・AJCSCP_TXT_SJIS　－ シフトＪＩＳ
・AJCSCP_TXT_EUC　－ 日本語ＥＵＣ
・AJCSCP_TXT_UTF8　－ ＵＴＦ－８
・AJCSCP_TXT_AUTO　－ 自動判別

説　明 ： 受信テキストの文字コード種別を設定／取得します。
AJCSCP_TXT_AUTO を設定した場合は、実際に受信したテキストから自動判別します。

戻り値 ： 設定時：TRUE　－ 成功　　取得時：文字コード種別 (AJCSCP_TXT_SJIS / EUC / UTF8 / AUTO)
FALSE － 失敗　　エラー時は０を返す

備　考 ： 受信したテキストは、指定文字コードから、バイト文字の場合はシフトＪＩＳに、UNICODE モードの場合は UTF-16 に変換して通知されます。
AJCSCP_TXT_AUTO（自動判別）を設定しても、受信中に文字コードが変化した場合は、変化前の文字コードと混在した状態となるため、しばらくは受信テキストの文字コードが正常に判断されない場合があります。

13.9.51. 送信文字コード種別設定／取得（AjcScp{Set/Get}TxTextCode）

形　式 ： BOOL　AjcScpSetTxTextCode（HAJCSCP hScp, AJCSCP_TEXTCODE code）; - 送信文字コード種別設定
AJCSCP_TEXTCODE　AjcScpGetTxTextCode（HAJCSCP hScp）; ---------- 送信文字コード取種別得

引　数 ： hScp　　- インスタンスハンドル
code　　- 受信テキストコード種別
・AJCSCP_TXT_SJIS　- シフトＪＩＳ
・AJCSCP_TXT_EUC　- 日本語ＥＵＣ
・AJCSCP_TXT_UTF8　- ＵＴＦ－８
・AJCSCP_TXT_AUTO　- 受信テキストコードと同じ

説　明 ： 送信テキストの文字コード種別を設定／取得します。
AJCSCP_TXT_AUTO を設定した場合は、受信文字コード種別と同じ設定となります。

戻り値 ： 設定時：TRUE　- 成功　　取得時：文字コード種別（AJCSCP_TXT_SJIS / EUC / UTF8 / AUTO）
FALSE - 失敗　　エラー時は０を返す

備　考 ： AjcScpSendText()やAjcScpSendChar()で送信するテキストは、指定文字コードに変換されて送信されます。

13.9.52. 実際の送受信文字コード種別取得（AjcScpGetActual{Rx/Tx}TextCode）

形　式 ： AJCSCP_TEXTCODE　AjcScpGetActualRxTextCode（HAJCSCP hScp）; --- 受信文字コード種別取得
AJCSCP_TEXTCODE　AjcScpGetActualTxTextCode（HAJCSCP hScp）; --- 送信文字コード種別取得

引　数 ： hScp　　- インスタンスハンドル

説　明 ： 実際の送受信文字コード種別を取得します。
AjcScpGetActualRxTextCode()では、AJCSCP_TXT_AUTO が設定されている場合は、実際に受信したテキストから判断された文字コード種別を返します。
AjcScpGetActualTxTextCode()では、AJCSCP_TXT_AUTO が設定されている場合は、受信文字コード種別を返します。

戻り値 ： 文字コード種別（AJCSCP_TXT_SJIS / EUC / UTF8）
エラー時は０を返す

13.9.53. 自メールスロット生成（AjcScpCreateMySlo）

形　式 ： BOOL　AjcScpCreateMySlot　(HAJCSCP hScp);
BOOL　AjcScpCreateMySlotEx(HAJCSCP hScp , UI nMaxMessageSize, UI lReadTimeout,
LPSECURITY_ATTRIBUTES lpSecurityAttributes);

引　数 ： hScp　　- インスタンスハンドル
nMaxMessageSize　　- メールスロット送信時の最大メッセージサイズ（未指定時は０（任意）を仮定）
lReadTimeout　　- メールスロット読み出しタイムアウト[ms]（未指定時は０を仮定）
lpSecurityAttributes – メールスロットセキュリティ情報へのポインタ（未指定時は NULL を仮定）

説　明 ： 自メールスロットの生成を行います。
既に自メールスロットが生成されている場合、既存のメールスロットは破棄されます。
AjcScpCreateMySlotEx()の nMaxMessageSize. lReadTimeout, lpSecurityAttributes は、内部で実行されるシステム API「CreateMailslot()」実行時の第２～４引数となります。(CreateMailslot()については MSDN 参照)
これらの引数が未指定の場合、nMaxMessageSize=0. lReadTimeout=0, lpSecurityAttributes=NULL を仮定します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

備　考 ： 自メールスロットの名称を設定するには、AjcScpSetMailSlotNames()を実行します。
自メールスロットの名称の設定と生成を行うには、以下のようにします。

```
AjcScpSetMailSlotNames(hScp, "NewMySlot", NULL, NULL);
AjcScpCreateMySlot     (hScp);
```

13.9.54. 自メールスロット消去（AjcScpDeleteMySlot）

形　式 ： BOOL AjcScpDeleteMySlot (HAJCSCP hScp);

引　数 ： hScp　　- インスタンスハンドル

説　明 ： 自メールスロットを消去します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

13.9.55. 自メールスロットの生成状態取得（AjcScpMySlotIsCreated）

形　式 ： BOOL　AjcScpMySlotIsCreated (HAJCSCP hScp);

引　数 ： hScp　　- インスタンスハンドル

説　明 ： 自メールスロットの生成状態を取得します。

戻り値 ： TRUE　- 自メールスロット生成済
FALSE - 自メールスロット未生成

13.9.56. 自メールスロットのパス名取得（AjcScpGetMySlotPathName）

形　式 ： UTP　AjcScpGetMySlotPathName(HAJCSCP hScp);

引　数 ： hScp　　- インスタンスハンドル

説　明 ： 現在設定されている自メールスロットのパス名（¥¥*自コンピュータ名*¥mailslot¥*自スロット名*）を取得します。

戻り値 ： 現在設定されている自メールスロットのパス名文字列へのポインタ

13.9.57. 自コンピュータ名取得（AjcScpGetMyComputerName）

形　式 ： BOOL　AjcScpGetMyComputerName (HAJCSCP hScp, UT pBuf[MAX_COMPUTERNAME_LENGTH + 1]);

引　数 ： hScp　　- インスタンスハンドル

説　明 ： 自コンピュータ名を取得します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

13.9.58. メールスロット名情報設定／取得（AjcScp{Set/Get}MailSlotNames）

形　式 ： BOOL AjcScpSetMailSlotNames (HAJCSCP hScp, C_UTP pMySlot, C_UTP pRmyHost, C_UTP pRmtSlot); - 設定
BOOL AjcScpGetMailSlotNames (HAJCSCP hScp, UT pMySlot[64], ---------- 取得
UT pRmyHost[64], UT pRmtSlot[64]);

引　数 ： hScp - インスタンスハンドル
設定時
pMySlot - 自スロット名 (未設定時は NULL)
pRmyHost - 送信先リモートコンピュータ名 (未設定時は NULL, 自コンピュータとする場合は、空文字列("")を指定)
pRmySlot - 送信先リモートスロット名 (未設定時は NULL, 送信を行わない場合は、空文字列("")を指定可)
取得時
pMySlot - 自スロット名を格納するバッファのアドレス (不要時は NULL)
pRmyHost - 送信先リモートコンピュータ名を格納するバッファのアドレス (不要時は NULL)
pRmySlot - 送信先リモートスロット名を格納するバッファのアドレス (不要時は NULL)

説　明 ： メールスロットに関する名称情報を設定／取得します。
AjcScpSetMailSlotNames()は、メールスロットの名称を設定するだけで、メールスロットの生成やオープンは行いません。
自メールスロット（受信用メールスロット）の生成を行うには、AjcScpCreateMySlot[Ex]()を実行します。
送信先メールスロットのオープンは、AjcScpOpenSlot()等で行われます。

戻り値 ： TRUE - 成功
FALSE - 失敗

13.9.59. メールスロット送信制限速度の設定／取得（AjcScp{Set/Get}TxSpeedLimit）

形　式 ： BOOL AjcScpSetTxSpeedLimit(HAJCSCP hScp, BOOL flag, UI bps); - メールスロット送信制限速度の設定
BOOL AjcScpGetTxSpeedLimit(HAJCSCP hScp, UIP pBps); ---------- メールスロット送信制限速度の取得

引　数 ： hScp - インスタンスハンドル
flag - 送信速度制限フラグ（TRUE:制限する, FALSE:制限しない）
bps - 送信制限速度[Kbytes / sec]
pBps - 現在設定されている送信制限速度[Kbytes / sec]を格納するバッファのアドレス（不要時は NULL）

説　明 ： メールスロットによる通信時の（擬似的な）送信速度の制限を設定します。
flag=TRUE とした場合、メールスロットの送信において、全般的な送信速度が「bps」で指定した速度（Kバイト／秒）程度となるように制御します。
flag=FALSE とした場合は、送信速度の制限は行いません。

戻り値 ： 設定時：TRUE - 成功　　取得時：送信速度制限フラグ（TRUE:制限する, FALSE:制限しない）
FALSE - 失敗　　エラー時は FALSE を返す

13.9.60. 送信停止の認識時間設定（AjcScpSetRecognizeTxStopTime）

形　式 ： ULL AjcScpSetRecognizeTxStopTime(HAJCSCP hScp, UI msTime);

引　数 ： hScp - インスタンスハンドル
usTime - 送信停止の認識時間[ms] （－１：未設定, デフォルトは 300ms）

説　明 ： 送信停止を認識する時間を設定します。
送信スレッドにおいて、送信要求が途絶えてから msTime で指定した時間が経過した場合、他のスレッドやアプリに CPU を明け渡すために最小時間のウェイトを行います。(Sleep(1)を実行します)

戻り値 ： 前回の設定値[ms]

13.9.61. 受信停止の認識時間設定（AjcScpSetRecognizeRxStopTime）

形　式 ： ULL　AjcScpSetRecognizeRxStopTime(HAJCSCP hScp, UI msTime);

引　数 ： hScp　　- インスタンスハンドル
msTime　　- 受信停止の認識時間[us]　（－１：未設定，デフォルトは 300ms）

説　明 ： 受信停止を認識する時間を設定します。
受信スレッドにおいて、データ受信が途絶えてから msTime で指定した時間が経過した場合、他のスレッドやアプリに CPU を明け渡すために最小時間のウェイトを行います。(Sleep(1)を実行します)

戻り値 ： 前回の設定値[ms]

13.10. 基本サンプルコード

13.10.1. SW_ScpComC（COM ポート通信， コンソールアプリ）

以下に、簡単なＣＯＭポート通信のコンソールアプリの例を示します。
このアプリは、ＣＯＭポートから受信したデータをそのまま返信します。

実行イメージ



プログラムコード

```
 1 : //
 2 : //  SW_ScpComC.c
 3 : //
 4 : #include    <AjrCstXX.h>
 5 : #include    <stdio.h>
 6 : #include    <conio.h>
 7 : #include    <time.h>
 8 : #include    <tchar.h>
 9 :
10 : #define COM4    4
11 :
12 : HAJCSCP hScp  = NULL;
13 : BOOL    fExit = FALSE;
14 :
15 : //=============================================================================================================//
16 : static  VO  SubExit(VO)
17 : {
18 :     AjcPrintF(TEXT("プログラムを終了します。¥n"));
19 :     Sleep(3000);
20 :     fExit = TRUE;
21 : }
22 : //=============================================================================================================//
23 : int  AjcMain(int argc, UTP argv[])
24 : {
25 :     WPARAM      wParam;
26 :     LPARAM      lParam;
27 :     UTP         pDat;
28 :     UI          lDat, param, Count = 0;
29 :
30 :     AjcSetStdMode(AJCTEC_UTF_8, FALSE); //  全角表示設定
31 :     //  シリアル通信インスタンス生成
32 :     hScp = AjcScpCreate();
33 :     //  モード設定 (TRUE : 受信チャンクデータをバイナリデータとして扱う)
34 :     AjcScpSetMode(hScp, NULL, 0, AJCSCP_CM_BIN);
35 :     //  使用するイベント設定
36 :     AjcScpSetEvtMask(hScp, AJCSCP_EV_RXCHUNK | AJCSCP_EV_PORTSTATE);
37 :
38 :     AjcPrintF(TEXT("ＣＯＭ%d ポート通信(115200, 8, N, 1)で、ループバックを行います。¥n")
39 :               TEXT("いずれかのキーを押すとプログラムを終了します。¥n¥n"), COM4);
40 :     //  ポートオープン
41 :     AjcScpOpen (hScp, COM4, 115200, 8, 'N', 1);                         //  ＣＯＭポート オープン
42 :     while (!fExit) {                                                        //      ポートクローズまでループ
43 :         if (AjcScpWaitEvent(hScp, &wParam, &lParam, 200)) {                 //          イベント待ち，イベントあり？
44 :             C_UTP   pPortName = AjcScpGetPortPathName(hScp);
45 :             AjcScpGetEventData(hScp, lParam, &pDat, &lDat, &param);         //              イベントデータ取得
46 :             switch (wParam) {
47 :                 case AJCSCP_EV_PORTSTATE:                                   //                  ●ポート状態通知
48 :                     switch (param) {
49 :                         case AJCSCP_OPENED:     // オープン通知
50 :                             AjcPrintF(TEXT("ＣＯＭ%d ポートをオープンしました。通信可能です。¥n"), COM4);
51 :                             break;
52 :                         case AJCSCP_CLOSED:     // クローズ通知
53 :                             AjcPrintF(TEXT("ＣＯＭ%d ポートをクローズしました。通信不能です。¥n"), COM4);
54 :                             SubExit();
55 :                             break;
56 :                         case AJCSCP_OPENFAIL:   // オープン失敗通知
57 :                             AjcPrintF(TEXT("ＣＯＭ%d ポートのオープンを失敗しました。¥n"), COM4);
58 :                             SubExit();
59 :                             break;
60 :                     }
61 :                     break;
62 :                 case AJCSCP_EV_RXCHUNK:                                     //                  ●バイナリチャンク受信通知
63 :                     Count += lDat;
64 :                     AjcPrintF(TEXT("%9d Bytes received¥r"), Count);         //                      受信バイト数表示
65 :                     AjcScpSendBinData(hScp, pDat, lDat);                    //                      エコーバック送信
66 :                     break;
67 :             }
68 :             AjcScpRelEventData(hScp, lParam);                               //              イベントデータ開放
69 :         }
70 :         if (_kbhit()) {                                                     //      キー押下？
71 :             _getch();
72 :             AjcScpClose(hScp);                                              //          ポートクローズ
73 :         }
74 :     }
75 :     AjcScpDelete(hScp);                                                     //  シリアル通信インスタンス消去
76 :     return 0;
77 : }
```

13.10.2. SW_ScpMailSLotC（メールスロット通信, コンソールアプリ）

以下に、簡単なメールスロット通信のコンソールアプリの例を示します。
最初は、自プログラムをもう１つ起動し、２つのプログラムで互いに通信し、受信したテキストを表示します。
「T」キーを押すと相手に現在時刻文字列を送信し、その他のキーを押すとプログラムを終了します。

実行イメージ



プログラムコード

```
 1 : //
 2 : //  SW_ScpMailSlotC.c
 3 : //
 4 : #include    <AjrCstXX.h>
 5 : #include    <stdio.h>
 6 : #include    <conio.h>
 7 : #include    <time.h>
 8 : #include    <tchar.h>
 9 :
10 : HANDLE  hMutex= NULL;
11 : HAJCSCP hScp  = NULL;
12 : BOOL    fExit = FALSE;
13 : UT      MySlot [16] = {0};
14 : UT      RmtSlot[16] = {0};
15 :
16 : //=============================================================================================================//
17 : static  VO  SendTime(VO)    //  現在時刻送信
18 : {
19 :     SYSTEMTIME  lt;
20 :     GetLocalTime(&lt);
21 :     AjcScpSendTextF(hScp, TEXT("%s¥n"), AjcTimeStr(&lt, TRUE));
22 : }
23 : //=============================================================================================================//
24 : static  VO  SubExit(VO)     //  プログラム終了
25 : {
26 :     AjcPrintF(TEXT("プログラムを終了します。¥n"));
27 :     Sleep(3000);
28 :     fExit = TRUE;
29 : }
30 : //=============================================================================================================//
31 : int  AjcMain(int argc, UTP argv[])
32 : {
33 :     WPARAM      wParam;
34 :     LPARAM      lParam;
35 :     UTP         pDat;
36 :     UI          lDat, param, Count = 0;
37 :
38 :     AjcSetStdMode(AJCTEC_UTF_8, FALSE); //  全角表示設定
39 :
40 :     //  初期メッセージ表示
41 :     AjcPrintF(TEXT("メールスロット通信を行い、受信したテキストを表示します。¥n")
42 :               TEXT("'T' キーを押すと、現在時刻文字列を送信します。¥n")
43 :               TEXT("'WSC' キーを押すとプログラムを終了します。¥n¥n"), MySlot, RmtSlot);
```

```
44 :     //  ミューテックス生成
45 :     hMutex = CreateMutex(NULL, TRUE, TEXT("SW_SerialComPort4"));
46 :     //  最初のプロセス起動時
47 :     if (GetLastError() != ERROR_ALREADY_EXISTS) {
48 :         UT      path[MAX_PATH];
49 :         //  スロット名設定
50 :         MAjcStrCpy(MySlot , 16, TEXT("SlotA"));
51 :         MAjcStrCpy(RmtSlot, 16, TEXT("SlotB"));
52 :         //  シリアル通信インスタンス生成と自メールスロット生成
53 :         hScp = AjcScpCreateSlot(NULL, TRUE, MySlot, NULL, NULL);
54 :         AjcPrintF(TEXT("ＳＣＰインスタンスを生成し、受信スロット(%s)を生成しました。¥n"),MySlot);
55 :         //  ２つ目のプロセス起動
56 :         AjcGetAppPath(path, MAX_PATH);
57 :         ShellExecute(NULL, NULL, TEXT("SW_ScpMailSlotC_64W.exe"), TEXT(""), path, SW_SHOWNORMAL);
58 :         Sleep(1000);    //  ２つ目のプロセス起動待ち
59 :     }
60 :     //  ２つ目のプロセス起動時・・・
61 :     else {
62 :         //  スロット名読設定
63 :         MAjcStrCpy(MySlot , 16, TEXT("SlotB"));
64 :         MAjcStrCpy(RmtSlot, 16, TEXT("SlotA"));
65 :         //  シリアル通信インスタンス生成と自メールスロット生成
66 :         hScp = AjcScpCreateSlot(NULL, TRUE, MySlot, NULL, NULL);
67 :         AjcPrintF(TEXT("ＳＣＰインスタンスを生成し、受信スロット(%s)を生成しました。¥n"),MySlot);
68 :     }
69 :     //  モード設定（受信チャンクデータをテキストデータとして扱う）
70 :     AjcScpSetMode(hScp, NULL, 0, AJCSCP_CM_TEXT);
71 :     //  使用するイベント設定
72 :     AjcScpSetEvtMask(hScp, AJCSCP_EV_RXCHUNK | AJCSCP_EV_PORTSTATE);
73 :     //  ポートオープン
74 :     AjcScpOpenSlot(hScp, NULL, RmtSlot);                    //  送信先スロット オープン
75 :     while (!fExit) {                                        //      ポートクローズまでループ
76 :         if (AjcScpWaitEvent(hScp, &wParam, &lParam, 200)) {//          イベント待ち，イベントあり？
77 :             C_UTP   pPortName = AjcScpGetPortPathName(hScp);
78 :             AjcScpGetEventData(hScp, lParam, &pDat, &lDat, &param);//              イベントデータ取得
79 :             switch (wParam) {
80 :                 case AJCSCP_EV_PORTSTATE:                   //                  ●ポート状態通知
81 :                     switch (param) {
82 :                         case AJCSCP_OPENED:     // 送信スロットオープン通知
83 :                             AjcPrintF(TEXT("送信スロット(%s)をオープンしました。通信可能です。¥n"), RmtSlot, MySlot);
84 :                             break;
85 :                         case AJCSCP_CLOSED:     // 送信スロットクローズ通知
86 :                             AjcPrintF(TEXT("送信スロット(%s)をクローズしました。¥n"), RmtSlot);
87 :                             SubExit();
88 :                             break;
89 :                         case AJCSCP_OPENFAIL:   // 送信スロットオープン失敗通知
90 :                             AjcPrintF(TEXT("送信スロット(%s)のオープンを失敗しました。¥n"), RmtSlot);
91 :                             SubExit();
92 :                             break;
93 :                         case AJCSCP_MYSLOTFAIL: // 受信スロットオープン失敗通知
94 :                             AjcPrintF(TEXT("受信スロット(%s)のオープンを失敗しました。¥n"), MySlot);
95 :                             SubExit();
96 :                             break;
97 :                     }
98 :                     break;
99 :                 case AJCSCP_EV_RXCHUNK:                     //                  ●テキストチャンク受信通知
100 :                    AjcPrintF(pDat);                        //                      受信テキスト表示
101 :                    break;
102 :            }
103 :            AjcScpRelEventData(hScp, lParam);               //              イベントデータ開放
104 :        }
105 :        if (_kbhit()) {                                     //      キー押下？
106 :            int c = toupper(_getch());
107 :            if      (c == 'T' ) SendTime();                 //      'T'キー押下ならば、現在時刻送信
108 :            else if (c == 0x1B) AjcScpClose(hScp);          //      'ESC'キー押下ならば、ポートクローズし終了
109 :        }
110 :    }
111 :    AjcScpDelete(hScp);                                     //  シリアル通信インスタンス消去
112 :    return 0;
113 : }
114 :
```

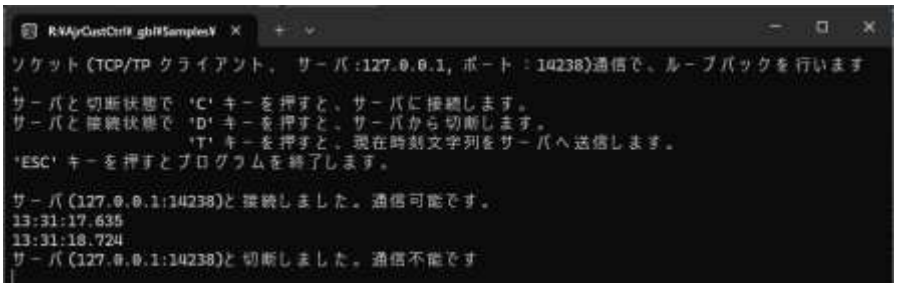
２つのプログラムが、AjcScpOpenSlot()により、送信スロットをオープンする前に、各々、受信スロット（自メールスロット）を生成しておく必要があります。

13.10.3. SW_ScpClientC（ソケット(クライアント)通信， コンソールアプリ）

以下に、簡単なソケット（TCP/IP クライアント）通信のコンソールアプリの例を示します。
以下のキー操作が可能です。

サーバ接続状態	キー	内容
切断	C	サーバへ接続
	ESC	終了
接続	D	サーバから切断
	T	サーバへ現在時刻文字列送信
	ESC	サーバから切断し終了

実行イメージ



プログラムコード

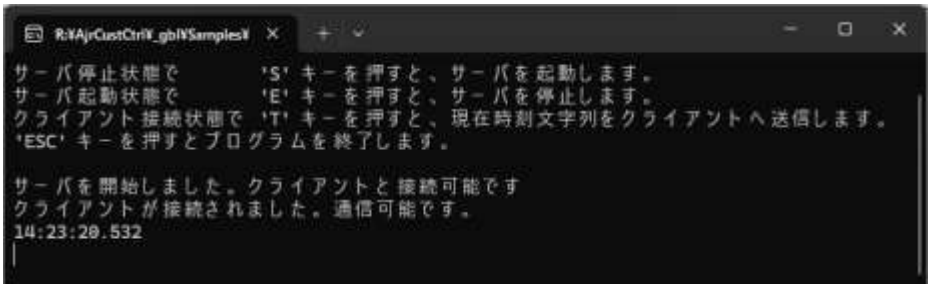
```
 1 : //
 2 : //  SW_ScpClientC.c
 3 : //
 4 : #include    <AjrCstXX.h>
 5 : #include    <stdio.h>
 6 : #include    <conio.h>
 7 : #include    <time.h>
 8 : #include    <tchar.h>
 9 :
10 : HAJCSCP hScp         = NULL;
11 : UT      ServName[32] = TEXT("127.0.0.1");
12 : UI      PortNo       = 14238;
13 : BOOL    fExit        = FALSE;
14 : BOOL    fEnd         = FALSE;
15 :
16 : //=============================================================================================================//
17 : static  VO  SendTime(VO)    //  現在時刻送信
18 : {
19 :     SYSTEMTIME  lt;
20 :     GetLocalTime(&lt);
21 :     AjcScpSendTextF(hScp, TEXT("%s¥n"), AjcTimeStr(&lt, TRUE));
22 : }
23 : //=============================================================================================================//
24 : static  VO  SubExit(VO)     //  プログラム終了
25 : {
26 :     AjcPrintF(TEXT("プログラムを終了します。¥n"));
27 :     Sleep(3000);
28 :     fExit = TRUE;
29 : }
```

```
30 : //=============================================================================================================//
31 : int  AjcMain(int argc, UTP argv[])
32 : {
33 :     WPARAM      wParam;
34 :     LPARAM      lParam;
35 :     UTP         pDat;
36 :     UI          lDat, param;
37 :
38 :     AjcSetStdMode(AJCTEC_UTF_8, FALSE); //  全角表示設定
39 :
40 :     //  初期メッセージ表示
41 :     AjcPrintF(TEXT("ソケット(TCP/TP クライアント,  サーバ:%s, ポート：%d)通信で、ループバックを行います。¥n")
42 :               TEXT("サーバと切断状態で 'C' キーを押すと、サーバに接続します。¥n")
43 :               TEXT("サーバと接続状態で 'D' キーを押すと、サーバから切断します。¥n")
44 :               TEXT("                   'T' キーを押すと、現在時刻文字列をサーバへ送信します。¥n")
45 :               TEXT("'ESC' キーを押すとプログラムを終了します。¥n¥n"), ServName, PortNo);
46 :     //  シリアル通信インスタンス生成
47 :     hScp = AjcScpCreate();
48 :     //  モード設定 (TRUE : 受信チャンクデータをテキストデータとして扱う)
49 :     AjcScpSetMode(hScp, NULL, 0, AJCSCP_CM_TEXT);
50 :     //  使用するイベント設定
51 :     AjcScpSetEvtMask(hScp, AJCSCP_EV_RXCHUNK | AJCSCP_EV_PORTSTATE);
52 :
53 :     while (!fExit) {                                                    //      ポートクローズまでループ
54 :         if (AjcScpWaitEvent(hScp, &wParam, &lParam, 200)) {             //          イベント待ち, イベントあり？
55 :             C_UTP   pPortName = AjcScpGetPortPathName(hScp);
56 :             AjcScpGetEventData(hScp, lParam, &pDat, &lDat, &param);     //              イベントデータ取得
57 :             switch (wParam) {
58 :                 case AJCSCP_EV_PORTSTATE:                               //                  ●ポート状態通知
59 :                     switch (param) {
60 :                         case AJCSCP_OPENED:     // オープン通知 (接続通知)
61 :                             AjcPrintF(TEXT("サーバ(%s:%d)と接続しました。通信可能です。¥n"), ServName, PortNo);
62 :                             break;
63 :                         case AJCSCP_CLOSED:     // クローズ通知 (切断通知)
64 :                             AjcPrintF(TEXT("サーバ(%s:%d)と切断しました。通信不能です¥n"), ServName, PortNo);
65 :                             if (fEnd) SubExit();
66 :                             break;
67 :                         case AJCSCP_OPENFAIL:   // オープン失敗通知
68 :                             AjcPrintF(TEXT("サーバ(%s:%d)との接続を失敗しました。¥n"), ServName, PortNo);
69 :                             break;
70 :                     }
71 :                     break;
72 :                 case AJCSCP_EV_RXCHUNK:                                 //                  ●テキストチャンク受信通知
73 :                     AjcPrintF(pDat);                                    //                      受信テキスト表示
74 :                     break;
75 :             }
76 :             AjcScpRelEventData(hScp, lParam);                           //              イベントデータ開放
77 :         }
78 :         if (_kbhit()) {                                                 //      キー押下？
79 :             int c = toupper(_getch());
80 :             if (AjcScpIsOpened(hScp)) {                                 //      サーバと接続状態？
81 :                 if (c == 'D' )  AjcScpClose(hScp);                      //          'D'キー押下ならば、サーバから切断
82 :                 if (c == 'T' )  SendTime();                             //          'T'キー押下ならば、サーバへ現在時刻送信
83 :                 if (c == 0x1B)  {fEnd = TRUE; AjcScpClose(hScp);}       //          ESC キー押下ならば、切断して終了
84 :             }
85 :             else {                                                      //      サーバと切断状態？
86 :                 if (c == 'C')  AjcScpOpenSock(hScp, ServName, PortNo);  //          'C'キー押下ならば、サーバへ接続
87 :                 if (c == 0x1B)SubExit();                                //          ESC キー押下ならば、終了
88 :             }
89 :         }
90 :     }
91 :     AjcScpDelete(hScp);                                                 //  シリアル通信インスタンス消去
92 :     return 0;
93 : }
```

13.10.4. SW_ScpServerC（ソケット(サーバ)通信， コンソールアプリ）

以下に、簡単なソケット（TCP/IP クライアント）通信のコンソールアプリの例を示します。
このアプリは、ソケットから受信したデータをそのまま返信（ループバック）します。

実行イメージ



ログラムコード

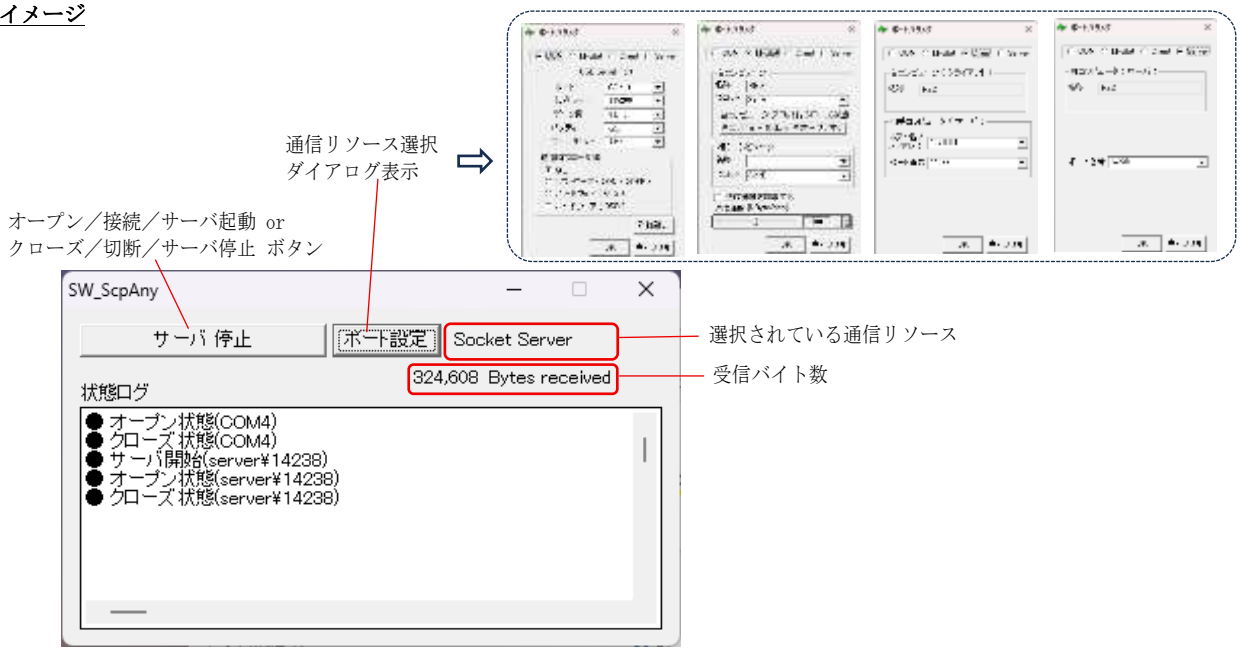
```
 1 : //
 2 : //  SW_ScpServerC.c
 3 : //
 4 : #include    <AjrCstXX.h>
 5 : #include    <stdio.h>
 6 : #include    <conio.h>
 7 : #include    <time.h>
 8 : #include    <tchar.h>
 9 :
10 : HAJCSCP hScp = NULL;
11 : UI      PortNo       = 14238;
12 : BOOL    fExit        = FALSE;
13 : BOOL    fEnd         = FALSE;
14 : //=============================================================================================================//
15 : static  VO  SendTime(VO)    //  現在時刻送信
16 : {
17 :     SYSTEMTIME  lt;
18 :     GetLocalTime(&lt);
19 :     AjcScpSendTextF(hScp, TEXT("%s¥n"), AjcTimeStr(&lt, TRUE));
20 : }
21 : //=============================================================================================================//
22 : static  VO  SubExit(VO)     //  プログラム終了
23 : {
24 :     AjcPrintF(TEXT("プログラムを終了します。¥n"));
25 :     Sleep(3000);
26 :     fExit = TRUE;
27 : }
```

```
28 : //=============================================================================================================//
29 : int  AjcMain(int argc, UTP argv[])
30 : {
31 :     WPARAM      wParam;
32 :     LPARAM      lParam;
33 :     UTP         pDat;
34 :     UI          lDat, param, Count = 0;
35 :
36 :     AjcSetStdMode(AJCTEC_UTF_8, FALSE); //  全角表示設定
37 :
38 :     //  初期メッセージ表示
39 :     AjcPrintF(TEXT("ソケット通信 (TCP/IP サーバ, ポート番号 = %d) 通信を行います。¥n")
40 :               TEXT("サーバ停止状態で       'S' キーを押すと、サーバを起動します。¥n")
41 :               TEXT("サーバ起動状態で       'E' キーを押すと、サーバを停止します。¥n")
42 :               TEXT("クライアント接続状態で 'T' キーを押すと、現在時刻文字列をクライアントへ送信します。¥n")
43 :               TEXT("'ESC' キーを押すとプログラムを終了します。¥n¥n"), PortNo);
44 :     //  シリアル通信インスタンス生成
45 :     hScp = AjcScpCreate();
46 :     //  モード設定 (TRUE : 受信チャンクデータをテキストデータとして扱う)
47 :     AjcScpSetMode(hScp, NULL, 0, AJCSCP_CM_TEXT);
48 :     //  使用するイベント設定
49 :     AjcScpSetEvtMask(hScp, AJCSCP_EV_RXCHUNK | AJCSCP_EV_PORTSTATE);
50 :     //  ポートオープン
51 :     while (!fExit) {                                                    //    キー押下までループ
52 :         if (AjcScpWaitEvent(hScp, &wParam, &lParam, 200)) {             //      イベント待ち, イベントあり？
53 :             C_UTP   pPortName = AjcScpGetPortPathName(hScp);
54 :             AjcScpGetEventData(hScp, lParam, &pDat, &lDat, &param);     //        イベントデータ取得
55 :             switch (wParam) {
56 :                 case AJCSCP_EV_PORTSTATE:                               //        ●ポート状態通知
57 :                     switch (param) {
58 :                         case AJCSCP_SRVSTART:   // サーバ開始
59 :                             AjcPrintF(TEXT("サーバを開始しました。クライアントと接続可能です¥n"));
60 :                             break;
61 :                         case AJCSCP_SRVSTOP:    // サーバ停止
62 :                             AjcPrintF(TEXT("サーバを停止しました。¥n"));
63 :                             if (fEnd) SubExit();
64 :                             break;
65 :                         case AJCSCP_SRVSRTFAIL:
66 :                             AjcPrintF(TEXT("サーバの開始を失敗しました。¥n"));
67 :                             break;
68 :                         case AJCSCP_OPENED:     // オープン状態 (クライアント接続)
69 :                             AjcPrintF(TEXT("クライアントが接続されました。通信可能です。¥n"));
70 :                             break;
71 :                         case AJCSCP_CLOSED:     // クローズ状態 (クライアント切断)
72 :                             AjcPrintF(TEXT("クライアントが切断されました。通信不能です。¥n"));
73 :                             break;
74 :                     }
75 :                     break;
76 :                 case AJCSCP_EV_RXCHUNK:                                 //        ●テキストチャンク受信通知
77 :                     AjcPrintF(pDat);                                    //          受信テキスト表示
78 :                     break;
79 :             }
80 :             AjcScpRelEventData(hScp, lParam);                           //        イベントデータ開放
81 :         }
82 :         //  キー操作
83 :         if (_kbhit()) {
84 :             int c = toupper(_getch());
85 :             if (AjcScpIsServerStarted(hScp)) {                          //      サーバ起動状態？
86 :                 if (c == 'E') AjcScpClose(hScp);                        //          'E'キー押下ならば、サーバ停止
87 :                 if (AjcScpIsOpened(hScp)) {                             //          クライアント接続状態？
88 :                     if (c == 'T')  SendTime();                          //              'T'キー押下ならば、クライアントへ現在時刻送信
89 :                 }
90 :                 if (c == 0x1B) {fEnd = TRUE; AjcScpClose(hScp);}        //          ESC キー押下ならば、サーバ停止して終了
91 :             }
92 :             else {                                                      //      サーバ停止状態？
93 :                 if (c == 'S') AjcScpOpenServ(hScp, PortNo);             //          'S'キー押下ならば、サーバ起動
94 :                 if (c == 0x1B) SubExit();                               //          ESC キー押下ならば、終了
95 :             }
96 :         }
97 :     }
98 :     AjcScpDelete(hScp);                                                 //  シリアル通信インスタンス消去
99 :     return 0;
100 : }
101 :
```

13.10.5. SW_ScpAny（通信リソースを選択して通信）

以下に、簡単な 通信リソースの選択が可能な Windows アプリの例を示します。
このアプリは、通信相手から受信したデータをそのまま返信（ループバック）し、受信したバイト数を表示しますます。

実行イメージ



プログラムコード

```
 1 :
 2 : //
 3 : //  SW_ScpAny.c  (通信リソースの選択が可能な Window アプリ)
 4 : //
 5 :
 6 : #include    <AjrCstXX.h>
 7 : #include    <math.h>
 8 : #include    <tchar.h>
 9 : #include    "resource.h"
10 :
11 : #define     WM_SCPEVENT     (WM_APP + 100)
12 :
13 : //-------------------------------------------------------------------------------------------------------------//
14 : //  ワーク                                                                                                     //
15 : //-------------------------------------------------------------------------------------------------------------//
16 : static  HINSTANCE       hInst;              //  ＤＬＬインスタンスハンドル
17 : static  HWND            hDlgMain;           //  ダイアログボックスハンドル
18 : static  HWND            hVthBin;            //  バイナリログウインド
19 : static  HWND            hVthSts;            //  状態ログウインド
20 : static  HAJCSCP         hScp;               //  シリアル通信インスタンスハンドル
21 : static  UI              Bytes = 0;          //  受信バイト数
22 :
23 : //-------------------------------------------------------------------------------------------------------------//
24 : //  内部サブ関数                                                                                               //
25 : //-------------------------------------------------------------------------------------------------------------//
26 : AJC_DLGPROC_DEF(Main);
27 : static  VO  ShowLabel(HWND hDlg);
28 :
29 : //=============================================================================================================//
30 : //                                                                                                             //
31 : //  W i n M a i n                                                                                              //
32 : //                                                                                                             //
33 : //=============================================================================================================//
34 : int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, PSTR szCmdLine, int iCmdShow)
35 : {
36 :     MSG     msg;
37 :
38 :     hInst   = hInstance;
39 :     //----- メイン・ダイアログオープン ------------------//
40 :     hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_DLGMAIN), NULL, AJC_DLGPROC_NAME(Main));
41 :     //----- ダイアログ表示 ------------------------------//
42 :     ShowWindow(hDlgMain, SW_SHOW);
43 :     //----- メッセージループ ----------------------------//
44 :     while (GetMessage(&msg, NULL, 0, 0)) {
45 :         do {
46 :             if (IsDialogMessage(hDlgMain, &msg)) break;
47 :             TranslateMessage(&msg);
```

```
48 :             DispatchMessage (&msg);
49 :         } while (0);
50 :     }
51 :
52 :     return (int)msg.wParam ;
53 : }
54 : //=============================================================================================================//
55 : //                                                                                                             //
56 : //  ダイアログ・プロシージャ                                                                                   //
57 : //                                                                                                             //
58 : //=============================================================================================================//
59 : //----- ダイアログ初期化 ------------------------------------------------------------------------------------//
60 : AJC_DLGPROC(Main, WM_INITDIALOG     )
61 : {
62 :     hDlgMain = hDlg;
63 :     hVthBin  = GetDlgItem(hDlg, IDC_VTH_BIN);
64 :     hVthSts  = GetDlgItem(hDlg, IDC_VTH_STS);
65 :     AjcLoadAllControlSettings(hDlg, TEXT("Settings"), AJCCTL_SELACT_ALL);   //  設定内容読み出し
66 :     hScp = AjcScpCreate();                                                  // シリアル通信インスタンス生成
67 :     AjcScpSetMode    (hScp, hDlg, WM_SCPEVENT, AJCSCP_CM_BIN);              // 実行モード設定（バイナリチャンク通知）
68 :     AjcScpSetEvtMask (hScp, AJCOPT2(AJCSCP_EV_, RXCHUNK, PORTSTATE));       // 使用するイベント指定
69 :     AjcScpOpenDefault(hScp);                                                // 通信ポートオープン（通信リソースはダイアログで選択可）
70 :     SetTimer(hDlg, 1, 10, NULL);
71 :     //  各ラベル表示
72 :     ShowLabel(hDlg);
73 :     return TRUE;
74 : }
75 : //----- シリアル通信イベント通知 ----------------------------------------------------------------------------//
76 : AJC_DLGPROC(Main, WM_SCPEVENT       )
77 : {
78 :     union {UBP pUB; UTP pUT; VOP pVO;} u;
79 :     UI      len, param, tick;
80 :
81 :     AjcScpGetEventData(hScp, lParam, &u.pVO, &len, &param); // イベントデータ取得
82 :     tick = GetTickCount();
83 :     switch (wParam) {
84 :         case AJCSCP_EV_PORTSTATE:                       //  ●ポート状態通知
85 :             switch (param) {
86 :                 case AJCSCP_OPENED:     AjcVthPrintF(hVthSts, TEXT("● 通信可能状態(%s)¥n"              ), u.pVO);
87 :                     Bytes = 0;
88 :                     break;
89 :                 case AJCSCP_CLOSED:     AjcVthPrintF(hVthSts, TEXT("● 通信不能状態(%s)¥n"              ), u.pVO);
90 :                     break;
91 :                 case AJCSCP_OPENFAIL:   AjcVthPrintF(hVthSts, TEXT("● オープン失敗(%s)¥n"              ), u.pVO);
92 :                     break;
93 :                 case AJCSCP_PORTCHG:    AjcVthPrintF(hVthSts, TEXT("● 通信リソース変化(%s→%s)¥n"      ), u.pVO, AjcScpGetPortPathName(hScp));
94 :                     break;
95 :                 case AJCSCP_MYSLOTFAIL: AjcVthPrintF(hVthSts, TEXT("● 自メールスロット生成失敗(%s)¥n" ), u.pVO);
96 :                     break;
97 :                 case AJCSCP_TXFAILURE:  AjcVthPrintF(hVthSts, TEXT("● 送信失敗(%s)¥n"                  ), u.pVO);
98 :                     break;
99 :                 case AJCSCP_SRVSTART:   AjcVthPrintF(hVthSts, TEXT("● サーバ開始(%s)¥n"                ), u.pVO);
100 :                    break;
101 :                case AJCSCP_SRVSTOP:    AjcVthPrintF(hVthSts, TEXT("● サーバ停止(%s)¥n"                ), u.pVO);
102 :                    break;
103 :                case AJCSCP_SRVSRTFAIL: AjcVthPrintF(hVthSts, TEXT("● サーバ開始失敗(%s)¥n"            ), u.pVO);
104 :                    break;
105 :            }
106 :            //  オープン・クローズボタン グレー解除
107 :            AjcEnableDlgItem(hDlg, IDC_CMD_OPEN, TRUE);
108 :            //  各ラベル表示
109 :            ShowLabel(hDlg);
110 :            break;
111 :        case AJCSCP_EV_RXCHUNK:                         //  ●チャンクデータ受信
112 :            Bytes += len;
113 :            AjcSepDlgItemUInt(hDlg, IDC_LBL_BYTES, Bytes);  //  受信バイト数表示
114 :            AjcScpSendBinData(hScp, u.pVO, len);            //  受信データ返送
115 :            break;
116 :    }
117 :    AjcScpRelEventData(hScp, lParam);                       // イベントデータ開放
118 :    return TRUE;
119 : }
120 : //----- ウインド破棄 ----------------------------------------------------------------------------------------//
121 : AJC_DLGPROC(Main, WM_DESTROY        )
122 : {
123 :     AjcSaveAllControlSettings(hDlg);    //  設定内容書き込み
124 :     AjcScpClose (hScp);                 //  通信ポートクローズ
125 :     AjcScpDelete(hScp);                 //  シリアル通信インスタンス消去
126 :     PostQuitMessage(0);                 //  プログラム終了
127 :     return TRUE;
128 : }
129 : //----- 通信パラメタ設定ボタン------------------------------------------------------------------------------//
130 : AJC_DLGPROC(Main, IDC_CMD_SETPARAM)
131 : {
132 :     if (HIWORD(wParam) == BN_CLICKED) {
133 :         // ダイアログによるシリアル通信リソースの選択や，通信パラメタ設定
134 :         if (AjcScpDlgParamEasy(hScp, hDlg)) {
135 :             ShowLabel(hDlg);    //  ボタンフェース表示
136 :         }
137 :     }
```

```
138 :     return TRUE;
139 : }
140 : //----- オープン／クローズボタン ----------------------------------------------------------------------------//
141 : AJC_DLGPROC(Main, IDC_CMD_OPEN  )
142 : {
143 :     if (AjcScpIsOpened(hScp) || AjcScpIsServerStarted(hScp)) AjcScpClose(hScp);         //  クローズ／サーバ停止
144 :     else                                                     AjcScpOpenDefault(hScp);   //  オープン／サーバ起動
145 :     //  オープン・クローズボタンをグレー表示
146 :     AjcEnableDlgItem(hDlg, IDC_CMD_OPEN, FALSE);
147 :
148 :     return TRUE;
149 : }
150 : //----- キャンセル ------------------------------------------------------------------------------------------//
151 : AJC_DLGPROC(Main, IDCANCEL      )
152 : {
153 :     AjcScpDelete(hScp);     //  シリアル通信インスタンス消去
154 :     DestroyWindow(hDlg);    //  ウインド破棄，終了
155 :     return TRUE;
156 : }
157 : //-----------------------------------------------------------------------------------------------------------//
158 : AJC_DLGMAP_DEF(Main)
159 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG      )
160 :     AJC_DLGMAP_MSG(Main, WM_SCPEVENT        )
161 :     AJC_DLGMAP_MSG(Main, WM_DESTROY         )
162 :     AJC_DLGMAP_CMD(Main, IDC_CMD_SETPARAM   )
163 :     AJC_DLGMAP_CMD(Main, IDC_CMD_OPEN       )
164 :     AJC_DLGMAP_CMD(Main, IDCANCEL           )
165 : AJC_DLGMAP_END
166 : //-----------------------------------------------------------------------------------------------------------//
167 : //  ボタンフェース表示                                                                                       //
168 : //-----------------------------------------------------------------------------------------------------------//
169 : static  VO      ShowLabel(HWND hDlg)
170 : {
171 :     UI  ps;
172 :
173 :     switch (ps = AjcScpGetSelectedPort(hScp)) {
174 :         default:                //  ●ＣＯＭポート
175 :             AjcSetDlgItemFStr(hDlg, IDC_LBL_SEL, TEXT("COM %d"), ps);
176 :             if (AjcScpIsOpened(hScp)) {                                         //  オープン状態？
177 :                 AjcSetDlgItemFStr(hDlg, IDC_CMD_OPEN, TEXT("COM %d 切断"), ps);
178 :             }
179 :             else {                                                              //  クローズ状態
180 :                 AjcSetDlgItemFStr(hDlg, IDC_CMD_OPEN, TEXT("COM %d 接続"), ps);
181 :             }
182 :             break;
183 :         case AJCSCP_PS_SLOT:    //  ●メールスロット
184 :             AjcSetDlgItemFStr(hDlg, IDC_LBL_SEL, TEXT("Mail Slot"), ps);
185 :             if (AjcScpIsOpened(hScp)) {                                         //  オープン状態？
186 :                 AjcSetDlgItemStr(hDlg, IDC_CMD_OPEN, TEXT("スロット 切断"));
187 :             }
188 :             else {                                                              //  クローズ状態
189 :                 AjcSetDlgItemFStr(hDlg, IDC_CMD_OPEN, TEXT("スロット 接続"));
190 :             }
191 :             break;
192 :         case AJCSCP_PS_CLIENT:  //  ●ソケット（クライアント）
193 :             AjcSetDlgItemFStr(hDlg, IDC_LBL_SEL, TEXT("Socket Client"), ps);
194 :             if (AjcScpIsOpened(hScp)) {                                         //  オープン状態？
195 :                 AjcSetDlgItemStr(hDlg, IDC_CMD_OPEN, TEXT("サーバから切断"));
196 :             }
197 :             else {                                                              //  クローズ状態
198 :                 AjcSetDlgItemFStr(hDlg, IDC_CMD_OPEN, TEXT("サーバへ接続"));
199 :             }
200 :             break;
201 :         case AJCSCP_PS_SERVER:  //  ●ソケット（サーバ）
202 :             AjcSetDlgItemFStr(hDlg, IDC_LBL_SEL, TEXT("Socket Server"), ps);
203 :             if (AjcScpIsServerStarted(hScp)) {                                  //  サーバ起動状態？
204 :                 AjcSetDlgItemStr(hDlg, IDC_CMD_OPEN, TEXT("サーバ 停止"));
205 :             }
206 :             else {                                                              //  サーバ停止状態
207 :                 AjcSetDlgItemStr(hDlg, IDC_CMD_OPEN, TEXT("サーバ 起動"));
208 :             }
209 :             break;
210 :     }
211 : }
```

13.11. サンプルプログラム

13.11.1. SW_SerialComPort1（シリアル通信による送受信）

このサンプルプログラムは、各種通信リソース（COM ポート，メールスロット，TCP/IP クライアント）における送受信ファンクションを実行します。




```
    1 : //
    2 : //  SW_SerialComPort1.c
    3 : //
    4 :
    5 : #include    <AjrCstXX.h>
    6 : #include    <tchar.h>
    7 : #include    "resource.h"
    8 :
    9 : #define WM_SCPEVENT     (WM_USER + 100)
   10 : #define MAX_TXTBOX_LEN  128
   11 :
   12 : //-------------------------------------------------------------------------------------------------------------//
   13 : //  ツールチップ                                                                                               //
   14 : //-------------------------------------------------------------------------------------------------------------//
   15 : typedef struct {
   16 :     int     id;
   17 :     C_UTP   pTxt;
   18 : } TIPTBL, *PTIPTBL;
   19 : typedef const TIPTBL *PCTIPTBL;
   20 :
   21 : static const TIPTBL TipTbl[] = {
   22 :     {IDC_TXT_SNDTEXT    , TEXT("Ｃ言語表記のテキストを設定してください。(ex. ABC¥¥n) ")                                  },
   23 :     {IDC_VTH_TXTCHUNK   , TEXT("リアルタイムに受信したデータをテキストデータとして表示します。¥n")
   24 :                           TEXT("複数バイト文字の場合は、文字が完結するまで次のリアルタイム受信データを待ちます。")          },
   25 :     {IDC_VTH_BINCHUNK   , TEXT("リアルタイムに受信したデータをバイナリデータとして表示します。")                        },
   26 :     {IDC_VTH_TEXT       , TEXT("受信ストリームから制御コード（TAB 以外）で区切られたテキストデータを抜き出して表示します。¥n")
   27 :                           TEXT("¥x1B[34m ここにファイルをドロップすると、ファイルの内容をバイナリデータとして送信します。") },
   28 :     {IDC_VTH_CTRL       , TEXT("受信ストリームから制御コード（TAB 以外）を抜き出して表示します。")                    },
   29 :     {IDC_VTH_ESC        , TEXT("受信したストリームから、ＥＳＣシーケンス（0x1B～英字）を抜きだして表示します。")          },
   30 :     {IDC_VTH_PKT        , TEXT("受信したパケットデータ（DLE・STX～DLE・ETX でサンドイッチされたデータ）をバイナリ表示します。
")},
   31 :     {IDC_VTH_NOPKT      , TEXT("リアルタイムに受信したデータ内のパケットデータ（DLE・STX～DLE・ETX）以外の部分をテキストとして
表示します。¥n")
   32 :                           TEXT("複数バイト文字の場合は、文字が完結するまで次のリアルタイム受信データを待ちます。")}
   33 : };
   34 :
   35 : #define MAX_TIPTBL  (sizeof TipTbl / sizeof TipTbl[0])
   36 :
```

```
37 : //-------------------------------------------------------------------------------------------------------------//
38 : //  ワーク                                                                                                     //
39 : //-------------------------------------------------------------------------------------------------------------//
40 : HINSTANCE       hInst;                          //  ＤＬＬインスタンスハンドル
41 : HWND            hDlgMain;                       //  ダイアログボックスハンドル
42 :
43 : HAJCSCP         hScp;
44 : HWND            hWndVthTxtChu;
45 : HWND            hWndVthBinChu;
46 : HWND            hWndVthText;
47 : HWND            hWndVthCtrl;
48 : HWND            hWndVthEsc;
49 : HWND            hWndVthPkt;
50 : HWND            hWndVthPkt;
51 : HWND            hWndVthNoPkt;
52 : HWND            hWndVthLog;
53 :
54 : //-------------------------------------------------------------------------------------------------------------//
55 : //  内部サブ関数                                                                                               //
56 : //-------------------------------------------------------------------------------------------------------------//
57 : AJC_DLGPROC_DEF(Main);
58 :
59 : static  VO      SubSendFile(HWND hDlg, C_UTP pFName);
60 : static  VO      ShowSignalState(UI sig);
61 : static  VO      ShowLabel(HWND hDlg);
62 :
63 : //=============================================================================================================//
64 : //                                                                                                             //
65 : //  Ｗｉｎ Ｍａｉｎ                                                                                           //
66 : //                                                                                                             //
67 : //=============================================================================================================//
68 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
69 : {
70 :     MSG     msg;
71 :
72 :     hInst = hInstance;
73 :
74 :     //----- メイン・ダイアログオープン ----------------//
75 :     hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_DLGMAIN), NULL, AJC_DLGPROC_NAME(Main));
76 :     ShowWindow(hDlgMain, SW_SHOW);
77 :
78 :     //----- メッセージループ --------------------------//
79 :     while (GetMessage(&msg, NULL, 0, 0)) {
80 :         do {
81 :             if (IsDialogMessage(hDlgMain, &msg)) break;
82 :             TranslateMessage(&msg);
83 :             DispatchMessage (&msg);
84 :         } while (0);
85 :     }
86 :
87 :     return (int)msg.wParam ;
88 : }
89 : //=============================================================================================================//
90 : //                                                                                                             //
91 : //  ダイアログ・プロシージャ                                                                                   //
92 : //                                                                                                             //
93 : //=============================================================================================================//
94 : //----- ダイアログ初期化 ----------------------------------------------------------------------------------//
95 : AJC_DLGPROC(Main, WM_INITDIALOG     )
96 : {
97 :     int     i;
98 :
99 :     hDlgMain        = hDlg;
100 :     hWndVthTxtChu   = GetDlgItem(hDlg, IDC_VTH_TXTCHUNK );
101 :     hWndVthBinChu   = GetDlgItem(hDlg, IDC_VTH_BINCHUNK );
102 :     hWndVthText     = GetDlgItem(hDlg, IDC_VTH_TEXT     );
103 :     hWndVthCtrl     = GetDlgItem(hDlg, IDC_VTH_CTRL     );
104 :     hWndVthEsc      = GetDlgItem(hDlg, IDC_VTH_ESC      );
105 :     hWndVthPkt      = GetDlgItem(hDlg, IDC_VTH_PKT      );
106 :     hWndVthNoPkt    = GetDlgItem(hDlg, IDC_VTH_NOPKT    );
107 :     hWndVthLog      = GetDlgItem(hDlg, IDC_VTH_EVTLOG   );
108 :
109 :     //----- ウインド位置ロード ----------------------//
110 :     AjcLoadWndPos(hDlg, NULL);
111 :     //----- ラジオボタンのグループ化 ----------------//
112 :     AjcSbcRadioBtns(GetDlgItem(hDlg, IDC_GRP_RXTEC));
113 :     AjcSbcRadioBtns(GetDlgItem(hDlg, IDC_GRP_TXTEC));
114 :     //----- ダイアログ項目の初期化 ------------------//
115 :     AjcSetDlgItemStr (hDlg, IDC_TXT_SNDTEXT  , TEXT("東京都港区赤坂 1-2-3¥¥n"));
116 :     AjcSetDlgItemStr (hDlg, IDC_TXT_SNDESC   , TEXT("[31m"               ));
```

```
117 :     AjcSetDlgItemStr (hDlg, IDC_TXT_SNDBIN   , TEXT("00 01 02"          ));
118 :     AjcSetDlgItemStr (hDlg, IDC_TXT_SNDPKT   , TEXT("10 11 12"          ));
119 :     AjcSetDlgItemStr (hDlg, IDC_TXT_SNDWORD14, TEXT("100"                   ));
120 :     AjcSetDlgItemChk (hDlg, IDC_CHK_RTS      , TRUE);
121 :     AjcSetDlgItemChk (hDlg, IDC_CHK_DTR      , TRUE);
122 :     AjcSetDlgItemUInt(hDlg, IDC_GRP_RXTEC    , 0    );
123 :     AjcSetDlgItemUInt(hDlg, IDC_GRP_TXTEC    , 0    );
124 :     //----- テキストボックス長設定 ------------------//
125 :     AjcSetDlgItemEdtLimit(hDlg, IDC_TXT_SNDTEXT, MAX_TXTBOX_LEN - 1);
126 :     AjcSetDlgItemEdtLimit(hDlg, IDC_TXT_SNDESC , MAX_TXTBOX_LEN - 1);
127 :     AjcSetDlgItemEdtLimit(hDlg, IDC_TXT_SNDBIN , MAX_TXTBOX_LEN - 1);
128 :     AjcSetDlgItemEdtLimit(hDlg, IDC_TXT_SNDPKT , MAX_TXTBOX_LEN - 1);
129 :     //----- ツールチップ設定 ------------------------//
130 :     for (i = 0; i < MAX_TIPTBL; i++) {
131 :         AjcTipTextAdd(GetDlgItem(hDlg, TipTbl[i].id), TipTbl[i].pTxt);
132 :     }
133 :     //----- ＳＣＰセットアップ ----------------------//
134 :     hScp = AjcScpCreateEx(TEXT("ComPort"), TRUE, NULL, NULL, NULL); //  ＳＣＰインスタンス生成
135 :     AjcScpSetMode(hScp, hDlg, WM_SCPEVENT, AJCSCP_CM_BOTH);         //  モード設定
136 :     AjcScpSetRxTextCode(hScp, AJCSCP_TXT_AUTO);                     //  受信テキストコード自動判別
137 :     AjcScpSetTxTextCode(hScp, AJCSCP_TXT_AUTO);                     //  送信テキストコード＝受信と同じ
138 :     AjcScpSetEvtMask(hScp,  0 |                                     //  イベントマスク設定
139 :                             AJCSCP_EV_PORTSTATE |                   //      ポート状態通知
140 :                             AJCSCP_EV_RXCHUNK   |                   //      チャンクデータ受信通知
141 :                             AJCSCP_EV_RXTEXT    |                   //      テキスト受信通知
142 :                             AJCSCP_EV_RXESC     |                   //      ＥＳＣコード受信通知
143 :                             AJCSCP_EV_RXCTRL    |                   //      制御コード受信通知
144 :                             AJCSCP_EV_RXPKT     |                   //      パケットデータ受信通知
145 :                             AJCSCP_EV_RXNOPKT   |                   //      パケット外データ受信通知
146 :                             AJCSCP_EV_TXEMPTY   |                   //      送信完了
147 :                             AJCSCP_EV_RING      |                   //      RING 変化通知
148 :                             AJCSCP_EV_ERR       |                   //      エラー通知
149 :                             AJCSCP_EV_RLSD      |                   //      RLSD 変化通知
150 :                             AJCSCP_EV_DSR       |                   //      DSR 変化通知
151 :                             AJCSCP_EV_CTS       |                   //      CTS 変化通知
152 :                             AJCSCP_EV_RXWORD14  |                   //      バイトペアによるワード (14Bit) データ受信
153 :     0);
154 :     //----- 初期グレー表示 --------------------------//
155 :     AjcEnableDlgGroup(hDlgMain, IDC_GRP_SETSIG, FALSE, FALSE);
156 :     AjcEnableDlgGroup(hDlgMain, IDC_GRP_SIGSTS, FALSE, FALSE);
157 :     AjcEnableDlgGroup(hDlgMain, IDC_GRP_SEND  , FALSE, FALSE);
158 :     AjcEnableDlgGroup(hDlgMain, IDC_GRP_RECV  , FALSE, FALSE);
159 :     AjcScpOpenDefault(hScp);
160 :     //----- ダイアログ項目のロード ------------------//
161 :     AjcLoadAllControlSettings (hDlg, TEXT("DlgSetting"), AJCOPT3(AJCCTL_SELACT_, ALL, NTCCHK, NTCRBT));
162 :     //----- 各種ラベル表示 --------------------------//
163 :     ShowLabel(hDlg);
164 :
165 :     return TRUE;
166 : }
167 : //----- ウインド破棄 ----------------------------------------------------------------------------------------//
168 : AJC_DLGPROC(Main, WM_DESTROY        )
169 : {
170 :     //----- ウインド位置セーブ ----------------------//
171 :     AjcSaveWndPos(hDlg, NULL);
172 :     //----- ダイアログ項目のセーブ ------------------//
173 :     AjcSaveAllControlSettings(hDlg);
174 :
175 :     AjcScpDelete(hScp);         //  ＳＣＰインスタンス消去
176 :     PostQuitMessage(0);
177 :     return TRUE;
178 : }
179 : //----- ＳＣＭイベント通知 ----------------------------------------------------------------------------------//
180 : AJC_DLGPROC(Main, WM_SCPEVENT       )
181 : {
182 :     UI      time = GetTickCount();
183 :     UI      len, param;
184 :     UT      txt[256];
185 :     union {UBP pBin; UWP pWord; UTP pTxt; VOP vp;} u;
186 :
187 :     AjcScpGetEventData(hScp, lParam, &u.vp, &len, &param);              //  イベントデータ取得
188 :     if(wParam & AJCSCP_EV_PORTSTATE) {                                  //  ●ポート状態通知
189 :         AjcVthPrintF(hWndVthLog, TEXT("%9d: PORTSTATE name=%s, len=%5d, param=%d¥n"), time, u.vp, len, param);
190 :         switch (param) {
191 :             case AJCSCP_OPENED:                                         //      ・オープン／接続状態
192 :                 if (AjcScpGetSelectedPort(hScp) <= AJCSCP_SEL_COMPORT) {
193 :                     //  出力信号設定
194 :                     AjcScpSetRTS(hScp, AjcGetDlgItemChk(hDlg, IDC_CHK_RTS));
195 :                     AjcScpSetDTR(hScp, AjcGetDlgItemChk(hDlg, IDC_CHK_DTR));
196 :                     //  入力信号表示
```

```
197 :                   AjcSetDlgItemChk(hDlg, IDC_CHK_CTS , (AjcScpGetSigState(hScp) & AJCSCP_CTS ) != 0);
198 :                   AjcSetDlgItemChk(hDlg, IDC_CHK_DSR , (AjcScpGetSigState(hScp) & AJCSCP_DSR ) != 0);
199 :                   AjcSetDlgItemChk(hDlg, IDC_CHK_RLSD, (AjcScpGetSigState(hScp) & AJCSCP_RLSD) != 0);
200 :                   AjcSetDlgItemChk(hDlg, IDC_CHK_RING, (AjcScpGetSigState(hScp) & AJCSCP_RING) != 0);
201 :               }
202 :               //  送受信表示有効化
203 :               AjcEnableDlgGroup(hDlgMain, IDC_GRP_SEND, TRUE, TRUE);
204 :               AjcEnableDlgGroup(hDlgMain, IDC_GRP_RECV, TRUE, TRUE);
205 :               break;
206 :           case AJCSCP_CLOSED:                                     //      ・クローズ／切断状態
207 :               //  送受信表示無効化
208 :               AjcEnableDlgGroup(hDlgMain, IDC_GRP_SEND, FALSE, FALSE);          //  送信表示無効化
209 :               AjcEnableDlgGroup(hDlgMain, IDC_GRP_RECV, FALSE, FALSE);          //  受信表示無効化
210 :               break;
211 :           case AJCSCP_SRVSTART:                                   //      ・サーバ開始
212 :               AjcTipTextShowCenter(hDlg, TEXT("¥n クライアントを接続してください ¥n "), 10000, NULL);
213 :               break;
214 :           case AJCSCP_OPENFAIL:                                   //      ・オープン失敗
215 :               AjcSnPrintF(txt, AJCTSIZE(txt), TEXT("%s オープン／接続失敗"), u.pTxt);
216 :               MessageBox(hDlg, txt, TEXT("SW_SerialComPort1"), MB_ICONERROR);
217 :               break;
218 :           case AJCSCP_SRVSRTFAIL:                                 //      ・サーバ開始失敗
219 :               MessageBox(hDlg, TEXT("サーバ開始失敗"), TEXT("SW_SerialComPort1"), MB_ICONERROR);
220 :               break;
221 :           case AJCSCP_MYSLOTFAIL:                                 //      ・自スロット生成失敗
222 :               AjcSnPrintF(txt, AJCTSIZE(txt), TEXT("自メールスロット (  %s  ) の生成失敗"), u.pTxt),
223 :               MessageBox(hDlg, txt, TEXT("SW_SerialComPort1"), MB_ICONERROR);
224 :               break;
225 :       }
226 :       ShowLabel(hDlg);                            //  各ラベル表示
227 :       AjcEnableDlgItem(hDlg, IDC_CMD_OPEN, TRUE); //  OPEN ボタン有効化
228 :   }
229 :   if(wParam & AJCSCP_EV_RXCHUNK ) {                               //  ●チャンクデータ受信通知
230 :       AjcVthPrintF(hWndVthLog, TEXT("%9d: RXCHUNK   len=%5d, param=0x%08X¥n"), time, len, param);
231 :       //  バイナリチャンク
232 :       if (param == 0) {
233 :           AjcVthHexDump(hWndVthBinChu, (C_VOP)u.pBin, len);
234 :           AjcVthPrintF (hWndVthBinChu, TEXT("¥n"));
235 :       }
236 :       //  テキストチャンク
237 :       else {
238 :           AjcVthPutText(hWndVthTxtChu, u.pTxt, -1);
239 :       }
240 :   }
241 :   if (wParam & AJCSCP_EV_RXTEXT ) {                               //  ●テキスト受信通知
242 :       AjcVthPrintF(hWndVthLog, TEXT("%9d: RXTEXT    len=%5d, param=0x%08X¥n"), time, len, param);
243 :       AjcVthPrintF(hWndVthText, TEXT("%s¥n"), u.pTxt);            //      テキストデータ表示
244 :   }
245 :   if (wParam & AJCSCP_EV_RXESC  ) {                               //  ●ＥＳＣコード受信通知
246 :       AjcVthPrintF(hWndVthLog, TEXT("%9d: RXESC     len=%5d, param=0x%08X¥n"), time, len, param);
247 :       AjcVthPrintF(hWndVthEsc, TEXT("¥¥x1B%s¥n"), u.pTxt + 1);        //      ＥＳＣデータ表示
248 :   }
249 :   if (wParam & AJCSCP_EV_RXCTRL ) {                               //  ●制御コード受信通知
250 :       AjcVthPrintF(hWndVthLog, TEXT("%9d: RXCTRL    len=%5d, param=0x%08X, data=%02X¥n"), time, len, param, *u.pBin);
251 :       AjcVthPrintF(hWndVthCtrl, TEXT("%02X¥n"), *u.pTxt);         //      制御コード表示
252 :   }
253 :   if (wParam & AJCSCP_EV_RXPKT  ) {                               //  ●パケットデータ受信通知
254 :       AjcVthPrintF(hWndVthLog, TEXT("%9d: RXPKT     len=%5d, param=0x%08X¥n"), time, len, param);
255 :       AjcVthHexDump(hWndVthPkt, (C_VOP)u.pBin, len);
256 :       AjcVthPrintF (hWndVthPkt, TEXT("¥n"));
257 :   }
258 :   if (wParam & AJCSCP_EV_TXEMPTY) {                               //  ●送信完了通知
259 :       AjcVthPrintF(hWndVthLog, TEXT("%9d: TXEMPTY   len=%5d, param=0x%08X¥n"), time, len, param);
260 :   }
261 :   if (wParam & AJCSCP_EV_RXNOPKT) {                               //  ●パケット外テキスト受信通知
262 :       AjcVthPrintF(hWndVthLog, TEXT("%9d: RXNOPKT   len=%5d, param=0x%08X¥n"), time, len, param);
263 :       AjcVthPutText(hWndVthNoPkt, u.pTxt, -1);                    //      パケット外テキスト表示
264 :   }
265 :   if (wParam & AJCSCP_EV_ERR    ) {                               //  ●エラー通知
266 :       AjcVthPrintF(hWndVthLog, TEXT("%9d: ERR       len=%5d, param=0x%08X¥n"), time, len, param);
267 :   }
268 :   if (wParam & AJCSCP_EV_RING   ) {                               //  ●RING 変化通知
269 :       AjcVthPrintF(hWndVthLog, TEXT("%9d: RING      len=%5d, param=0x%08X¥n"), time, len, param);
270 :       ShowSignalState(LOWORD(param));
271 :   }
272 :   if (wParam & AJCSCP_EV_RLSD   ) {                               //  ●RLSD 変化通知
273 :       AjcVthPrintF(hWndVthLog, TEXT("%9d: RLSD      len=%5d, param=0x%08X¥n"), time, len, param);
274 :       ShowSignalState(LOWORD(param));
275 :   }
276 :   if (wParam & AJCSCP_EV_DSR    ) {                               //  ●DSR 変化通知
```

```
277 :         AjcVthPrintF(hWndVthLog, TEXT("%9d: DSR       len=%5d, param=0x%08X¥n"), time, len, param);
278 :         ShowSignalState(LOWORD(param));
279 :     }
280 :     if (wParam & AJCSCP_EV_CTS    ) {                                   //  ●CTS 変化通知
281 :         AjcVthPrintF(hWndVthLog, TEXT("%9d: CTS       len=%5d, param=0x%08X¥n"), time, len, param);
282 :         ShowSignalState(LOWORD(param));
283 :     }
284 :     if (wParam & AJCSCP_EV_RXWORD14) {                                  //  ●バイトペアによるワード（14Bit）データ受信
285 :         RECT    r;
286 :         UT      txt[64];
287 :         AjcVthPrintF(hWndVthLog, TEXT("%9d: WORD14    len=%5d, param=0x%08X(%d)¥n"), time, len, *u.pWord, *u.pWord);
288 :         GetWindowRect(GetDlgItem(hDlg, IDC_LBL_SNDWORD14), &r);
289 :         AjcSnPrintF(txt, AJCTSIZE(txt), TEXT("１４ビットデータ（ %d (0x%04X) ）を受信しました"), *u.pWord, *u.pWord);
290 :         AjcTipTextShow(r.left, r.top, txt, -1, NULL);
291 :     }
292 :     AjcScpRelEventData(hScp, lParam);                                   //  イベントデータ開放
293 :
294 :     return TRUE;
295 : }
296 : //----- ウインドクローズ ------------------------------------------------------------------------------------------//
297 : AJC_DLGPROC(Main, IDCANCEL          )
298 : {
299 :     DestroyWindow(hDlg);
300 :     return TRUE;
301 : }
302 : //----- テキスト送信ボタン ----------------------------------------------------------------------------------------//
303 : AJC_DLGPROC(Main, IDC_CMD_SNDTEXT   )
304 : {
305 :     if (HIWORD(wParam) == BN_CLICKED) {
306 :         UT  txt[MAX_TXTBOX_LEN];
307 :         UT  snd[MAX_TXTBOX_LEN];
308 :         AjcGetDlgItemStr(hDlg, IDC_TXT_SNDTEXT, txt, AJCTSIZE(txt));
309 :         AjcCLangStrToBin(txt, snd, AJCTSIZE(snd));
310 :         AjcScpSendText(hScp, snd, -1);
311 :     }
312 :     return TRUE;
313 : }
314 : //----- ＥＳＣ送信ボタン ------------------------------------------------------------------------------------------//
315 : AJC_DLGPROC(Main, IDC_CMD_SNDESC    )
316 : {
317 :     if (HIWORD(wParam) == BN_CLICKED) {
318 :         UT  txt[MAX_TXTBOX_LEN];
319 :         AjcGetDlgItemStr(hDlg, IDC_TXT_SNDESC, txt, AJCTSIZE(txt));
320 :         AjcScpSendChar(hScp, 0x1B);
321 :         AjcScpSendText(hScp, txt, -1);
322 :     }
323 :     return TRUE;
324 : }
325 : //----- バイナリ送信ボタン ----------------------------------------------------------------------------------------//
326 : AJC_DLGPROC(Main, IDC_CMD_SNDBIN    )
327 : {
328 :     if (HIWORD(wParam) == BN_CLICKED) {
329 :         UI  ix;
330 :         UTP p;
331 :         UT  txt[MAX_TXTBOX_LEN];
332 :         UB  bin[MAX_TXTBOX_LEN];
333 :         ix = 0;
334 :         AjcGetDlgItemStr(hDlg, IDC_TXT_SNDBIN, txt, AJCTSIZE(txt));
335 :         if (p = MAjcStrTok(txt, TEXT(" "))) {
336 :             do {
337 :                 if (AjcHexStrToUB(p, 2, &bin[ix])) {
338 :                     ix++;
339 :                 }
340 :             } while (p = MAjcStrTok(NULL, TEXT(" ")));
341 :         }
342 :         if (ix != 0) {
343 :             AjcScpSendBinData(hScp, bin, ix);
344 :         }
345 :     }
346 :     return TRUE;
347 : }
348 : //----- パケット送信ボタン ----------------------------------------------------------------------------------------//
349 : AJC_DLGPROC(Main, IDC_CMD_SNDPKT    )
350 : {
351 :     if (HIWORD(wParam) == BN_CLICKED) {
352 :         UI  ix;
353 :         UTP p;
354 :         UT  txt[MAX_TXTBOX_LEN];
355 :         UB  bin[MAX_TXTBOX_LEN];
356 :         ix = 0;
```

```
357 :         AjcGetDlgItemStr(hDlg, IDC_TXT_SNDPKT, txt, AJCTSIZE(txt));
358 :         if (p = MAjcStrTok(txt, TEXT(" "))) {
359 :             do {
360 :                 if (AjcHexStrToUB(p, 2, &bin[ix])) {
361 :                     ix++;
362 :                 }
363 :             } while (p = MAjcStrTok(NULL, TEXT(" ")));
364 :         }
365 :         if (ix != 0) {
366 :             AjcScpSendPacket(hScp, bin, ix);
367 :         }
368 :     }
369 :     return TRUE;
370 : }
371 : //----- １４ビット値送信ボタン ----------------------------------------------------------------------------//
372 : AJC_DLGPROC(Main, IDC_CMD_SNDWORD14 )
373 : {
374 :     if (HIWORD(wParam) == BN_CLICKED) {
375 :         UI  val = AjcGetDlgItemUInt(hDlg, IDC_TXT_SNDWORD14);
376 :         AjcScpSendWord14LF(hScp, val);
377 :     }
378 :     return TRUE;
379 : }
380 : //----- ＤＴＲチェックボックス ----------------------------------------------------------------------------//
381 : AJC_DLGPROC(Main, IDC_CHK_DTR       )
382 : {
383 :     if (HIWORD(wParam) == BN_CLICKED) {
384 :         AjcScpSetDTR(hScp, AjcGetDlgItemChk(hDlg, IDC_CHK_DTR));
385 :     }
386 :     return TRUE;
387 : }
388 : //----- ＲＴＳチェックボックス ----------------------------------------------------------------------------//
389 : AJC_DLGPROC(Main, IDC_CHK_RTS       )
390 : {
391 :     if (HIWORD(wParam) == BN_CLICKED) {
392 :         AjcScpSetRTS(hScp, AjcGetDlgItemChk(hDlg, IDC_CHK_RTS));
393 :     }
394 :     return TRUE;
395 : }
396 : //----- オープン／クローズ ボタン -------------------------------------------------------------------------//
397 : AJC_DLGPROC(Main, IDC_CMD_OPEN      )
398 : {
399 :     if (HIWORD(wParam) == BN_CLICKED) {
400 :         if (AjcScpIsOpened(hScp) || AjcScpIsServerStarted(hScp)) {
401 :             AjcScpClose(hScp);
402 :         }
403 :         else {
404 :             AjcScpOpenDefault(hScp);
405 :         }
406 :         AjcEnableDlgItem(hDlg, IDC_CMD_OPEN, FALSE);    //  OPEN ボタン無効化
407 :     }
408 :     return TRUE;
409 : }
410 : //----- 簡易設定ボタン ------------------------------------------------------------------------------------//
411 : AJC_DLGPROC(Main, IDC_CMD_EASY      )
412 : {
413 :     if (HIWORD(wParam) == BN_CLICKED) {
414 :         AjcScpDlgParamEasy(hScp, hDlg);
415 :         ShowLabel(hDlg);
416 :     }
417 :     return TRUE;
418 : }
419 : //----- 詳細設定ボタン ------------------------------------------------------------------------------------//
420 : AJC_DLGPROC(Main, IDC_CMD_DETAIL    )
421 : {
422 :     if (HIWORD(wParam) == BN_CLICKED) {
423 :         AjcScpDlgParamDetail(hScp, hDlg);
424 :     }
425 :     return TRUE;
426 : }
427 : //----- 受信テキストエンコード ----------------------------------------------------------------------------//
428 : AJC_DLGPROC(Main, IDC_GRP_RXTEC     )
429 : {
430 :     if (HIWORD(wParam) == AJCRBTN_SELECT) {
431 :         //  受信テキストエンコード設定
432 :         switch (lParam) {
433 :             case 0: AjcScpSetRxTextCode(hScp, AJCSCP_TXT_SJIS); break;
434 :             case 1: AjcScpSetRxTextCode(hScp, AJCSCP_TXT_UTF8); break;
435 :             case 2: AjcScpSetRxTextCode(hScp, AJCSCP_TXT_EUC ); break;
436 :             case 3: AjcScpSetRxTextCode(hScp, AJCSCP_TXT_AUTO); break;
```

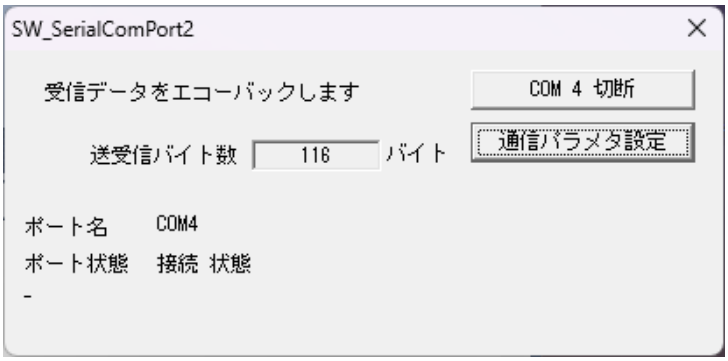
```
    437 :         }
    438 :         //  送信テキストエンコード設定（送信エンコードが AUTO の場合、受信エンコードと同一とするため）
    439 :         SendMessage(hDlg, WM_COMMAND, MAKELONG(IDC_GRP_TXTEC, AJCRBTN_SELECT), AjcGetDlgItemUInt(hDlg, IDC_GRP_TXTEC));
    440 :     }
    441 :     return TRUE;
    442 : }
    443 : //----- 送信テキストエンコード ----------------------------------------------------------------------------------//
    444 : AJC_DLGPROC(Main, IDC_GRP_TXTEC     )
    445 : {
    446 :     if (HIWORD(wParam) == AJCRBTN_SELECT) {
    447 :         //  送信テキストエンコード設定
    448 :         switch (lParam) {
    449 :             case 0: AjcScpSetTxTextCode(hScp, AJCSCP_TXT_SJIS); break;
    450 :             case 1: AjcScpSetTxTextCode(hScp, AJCSCP_TXT_UTF8); break;
    451 :             case 2: AjcScpSetTxTextCode(hScp, AJCSCP_TXT_EUC ); break;
    452 :             case 3: AjcScpSetTxTextCode(hScp, AJCSCP_TXT_AUTO); break;
    453 :         }
    454 :     }
    455 :     return TRUE;
    456 : }
    457 : //----- 全てクリアーボタン --------------------------------------------------------------------------------------//
    458 : AJC_DLGPROC(Main, IDC_CMD_CLRALL    )
    459 : {
    460 :     if (HIWORD(wParam) == BN_CLICKED) {
    461 :         AjcVthClear(hWndVthTxtChu   );
    462 :         AjcVthClear(hWndVthBinChu   );
    463 :         AjcVthClear(hWndVthText     );
    464 :         AjcVthClear(hWndVthCtrl     );
    465 :         AjcVthClear(hWndVthEsc      );
    466 :         AjcVthClear(hWndVthPkt      );
    467 :         AjcVthClear(hWndVthNoPkt    );
    468 :     }
    469 :     return TRUE;
    470 : }
    471 : //----- ＶＴＨ（テキスト）からの通知 ----------------------------------------------------------------------------//
    472 : AJC_DLGPROC(Main, IDC_VTH_TEXT  )
    473 : {
    474 :     if (HIWORD(wParam) == AJCVTHN_DROPFILE) {       //  ファイルドロップ
    475 :         UI      i, nFile = (UI)lParam;
    476 :         HANDLE  hFile;
    477 :         UL      bytes;
    478 :         UT      path[MAX_PATH];
    479 :         UB      buf[2014];
    480 :         for (i = 0; i < nFile; i++) {
    481 :             AjcVthGetDroppedFile(hWndVthText, path);
    482 :             if ((hFile = CreateFile(path, GENERIC_READ, FILE_SHARE_READ, NULL, OPEN_EXISTING, 0, NULL)) !=
INVALID_HANDLE_VALUE) {
    483 :                 while (ReadFile(hFile, buf, sizeof buf, &bytes, NULL) && bytes != 0) {
    484 :                     AjcScpSendBinData(hScp, buf, bytes);
    485 :                 }
    486 :                 CloseHandle(hFile);
    487 :             }
    488 :         }
    489 :     }
    490 :     return TRUE;
    491 : }
    492 :
    493 : //-------------------------------------------------------------------------------------------------------------//
    494 : AJC_DLGMAP_DEF(Main)
    495 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG      )
    496 :     AJC_DLGMAP_MSG(Main, WM_DESTROY         )
    497 :     AJC_DLGMAP_MSG(Main, WM_SCPEVENT        )
    498 :     AJC_DLGMAP_CMD(Main, IDCANCEL           )
    499 :     AJC_DLGMAP_CMD(Main, IDC_CMD_SNDTEXT    )
    500 :     AJC_DLGMAP_CMD(Main, IDC_CMD_SNDESC     )
    501 :     AJC_DLGMAP_CMD(Main, IDC_CMD_SNDBIN     )
    502 :     AJC_DLGMAP_CMD(Main, IDC_CMD_SNDPKT     )
    503 :     AJC_DLGMAP_CMD(Main, IDC_CMD_SNDWORD14  )
    504 :     AJC_DLGMAP_CMD(Main, IDC_CHK_DTR        )
    505 :     AJC_DLGMAP_CMD(Main, IDC_CHK_RTS        )
    506 :     AJC_DLGMAP_CMD(Main, IDC_CMD_OPEN       )
    507 :     AJC_DLGMAP_CMD(Main, IDC_CMD_EASY       )
    508 :     AJC_DLGMAP_CMD(Main, IDC_CMD_DETAIL     )
    509 :     AJC_DLGMAP_CMD(Main, IDC_GRP_RXTEC      )
    510 :     AJC_DLGMAP_CMD(Main, IDC_GRP_TXTEC      )
    511 :     AJC_DLGMAP_CMD(Main, IDC_CMD_CLRALL     )
    512 :     AJC_DLGMAP_CMD(Main, IDC_VTH_TEXT       )
    513 : AJC_DLGMAP_END
    514 :
    515 : //-------------------------------------------------------------------------------------------------------------//
```

```
516 : //  ファイル送信                                                                                                    //
517 : //--------------------------------------------------------------------------------------------------------------------//
518 : VO      SubSendFile(HWND hDlg, C_UTP pFName)
519 : {
520 :     HANDLE  fh;
521 :     UL      bytes;
522 :     UT      path[MAX_PATH];
523 :     UT      buf[1024];
524 :
525 :     AjcGetAppPath(path, MAX_PATH);
526 :     MAjcStrCat(path, MAX_PATH, pFName);
527 :     fh = CreateFile(path, GENERIC_READ, 0, NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
528 :     if (fh != INVALID_HANDLE_VALUE) {
529 :         while (ReadFile(fh, buf, AJCTSIZE(buf), &bytes, NULL) && bytes != 0) {
530 :             AjcScpSendBinData(hScp, buf, bytes);
531 :         }
532 :         CloseHandle(fh);
533 :     }
534 :     else {
535 :         UT      txt[512];
536 :         AjcSnPrintF(txt, AJCTSIZE(txt), TEXT("'%s open falure!"), path);
537 :         MessageBox(hDlgMain, txt, TEXT("Error"), MB_ICONERROR);
538 :     }
539 : }
540 :
541 : //--------------------------------------------------------------------------------------------------------------------//
542 : //  信号状態表示                                                                                                    //
543 : //--------------------------------------------------------------------------------------------------------------------//
544 : static  VO  ShowSignalState(UI sig)
545 : {
546 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_DSR , (sig & AJCSCP_DSR ) != 0);
547 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_CTS , (sig & AJCSCP_CTS ) != 0);
548 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_RING, (sig & AJCSCP_RING) != 0);
549 :     AjcSetDlgItemChk(hDlgMain, IDC_CHK_RLSD, (sig & AJCSCP_RLSD) != 0);
550 : }
551 :
552 : //--------------------------------------------------------------------------------------------------------------------//
553 : //  各ラベル表示                                                                                                    //
554 : //--------------------------------------------------------------------------------------------------------------------//
555 : static  VO      ShowLabel(HWND hDlg)
556 : {
557 :     UI  ps;
558 :     UT  DevName[128] = {0};
559 :
560 :     switch (ps = AjcScpGetSelectedPort(hScp)) {
561 :         default:                //  ●ＣＯＭポート
562 :             AjcEnableDlgGroup(hDlg, IDC_GRP_SETSIG, TRUE, TRUE);
563 :             AjcEnableDlgItem (hDlg, IDC_CMD_DETAIL, TRUE);
564 :             if (AjcScpIsOpened(hScp)) {                                         //  オープン状態？
565 :                 AjcSetDlgItemStr (hDlg, IDC_LBL_PORTSTATE, TEXT("接続 状態"));
566 :                 AjcSetDlgItemFStr(hDlg, IDC_CMD_OPEN, TEXT("COM %d 切断"), ps);
567 :             }
568 :             else {                                                              //  クローズ状態
569 :                 AjcSetDlgItemStr (hDlg, IDC_LBL_PORTSTATE, TEXT("切断 状態"));
570 :                 AjcSetDlgItemFStr(hDlg, IDC_CMD_OPEN, TEXT("COM %d 接続"), ps);
571 :             }
572 :             AjcScpGetPortDevName(AjcScpGetPortPathName(hScp), DevName, AJCTSIZE(DevName));
573 :             AjcSetDlgItemStr(hDlg, IDC_LBL_DEVNAME, DevName);
574 :             AjcSetDlgItemStr(hDlg, IDC_LBL_SERV, TEXT(""));
575 :             break;
576 :         case AJCSCP_PS_SLOT:    //  ●メールスロット
577 :             AjcEnableDlgGroup(hDlg, IDC_GRP_SETSIG, FALSE, FALSE);
578 :             AjcEnableDlgItem (hDlg, IDC_CMD_DETAIL, FALSE);
579 :             if (AjcScpIsOpened(hScp)) {                                         //  オープン状態？
580 :                 AjcSetDlgItemStr(hDlg, IDC_LBL_PORTSTATE, TEXT("接続 状態"));
581 :                 AjcSetDlgItemStr(hDlg, IDC_CMD_OPEN, TEXT("スロット 切断"));
582 :             }
583 :             else {                                                              //  クローズ状態
584 :                 AjcSetDlgItemStr (hDlg, IDC_LBL_PORTSTATE, TEXT("切断 状態"));
585 :                 AjcSetDlgItemFStr(hDlg, IDC_CMD_OPEN, TEXT("スロット 接続"));
586 :             }
587 :             AjcSetDlgItemStr(hDlg, IDC_LBL_DEVNAME, TEXT(""));
588 :             AjcSetDlgItemStr(hDlg, IDC_LBL_SERV, TEXT(""));
589 :             break;
590 :         case AJCSCP_PS_CLIENT:  //  ●ソケット（クライアント）
591 :             AjcEnableDlgGroup(hDlg, IDC_GRP_SETSIG, FALSE, FALSE);
592 :             AjcEnableDlgItem (hDlg, IDC_CMD_DETAIL, FALSE);
593 :             if (AjcScpIsOpened(hScp)) {                                         //  オープン状態？
594 :                 AjcSetDlgItemStr(hDlg, IDC_LBL_PORTSTATE, TEXT("接続 状態"));
595 :                 AjcSetDlgItemStr(hDlg, IDC_CMD_OPEN, TEXT("サーバから切断"));
```

```
596 :             }
597 :             else {                                                          //  クローズ状態
598 :                 AjcSetDlgItemStr (hDlg, IDC_LBL_PORTSTATE, TEXT("切断 状態"));
599 :                 AjcSetDlgItemFStr(hDlg, IDC_CMD_OPEN, TEXT("サーバへ接続"));
600 :             }
601 :             AjcSetDlgItemStr(hDlg, IDC_LBL_DEVNAME, TEXT(""));
602 :             AjcSetDlgItemStr(hDlg, IDC_LBL_SERV, TEXT(""));
603 :             break;
604 :         case AJCSCP_PS_SERVER:  //  ●ソケット (サーバ)
605 :             AjcEnableDlgGroup(hDlg, IDC_GRP_SETSIG, FALSE, FALSE);
606 :             AjcEnableDlgItem (hDlg, IDC_CMD_DETAIL, FALSE);
607 :             if (AjcScpIsServerStarted(hScp)) {                              //  サーバ起動状態？
608 :                 if (AjcScpIsOpened(hScp)) AjcSetDlgItemStr(hDlg, IDC_LBL_PORTSTATE, TEXT("接続 状態"));
609 :                 else                      AjcSetDlgItemStr(hDlg, IDC_LBL_PORTSTATE, TEXT("切断 状態"));
610 :                 AjcSetDlgItemStr(hDlg, IDC_CMD_OPEN, TEXT("サーバ 停止"));
611 :             }
612 :             else {                                                          //  サーバ停止状態
613 :                 AjcSetDlgItemStr(hDlg, IDC_LBL_PORTSTATE, TEXT("切断 状態"));
614 :                 AjcSetDlgItemStr(hDlg, IDC_CMD_OPEN, TEXT("サーバ 起動"));
615 :             }
616 :             AjcSetDlgItemStr(hDlg, IDC_LBL_DEVNAME, TEXT(""));
617 :             if (AjcScpIsServerStarted(hScp)) AjcSetDlgItemStr(hDlg, IDC_LBL_SERV, TEXT("サーバ動作中"));
618 :             else                             AjcSetDlgItemStr(hDlg, IDC_LBL_SERV, TEXT("サーバ停止中"));
619 :             break;
620 :     }
621 :     AjcSetDlgItemStr(hDlg, IDC_LBL_PORTNAME, AjcScpGetPortPathName(hScp));          //  ポート名表示
622 : }
623 :
```

13.11.2. SW_SerialComPort2（エコーバック）

このサンプルプログラムは、受信したデータをエコーバック（そのまま返信）します。



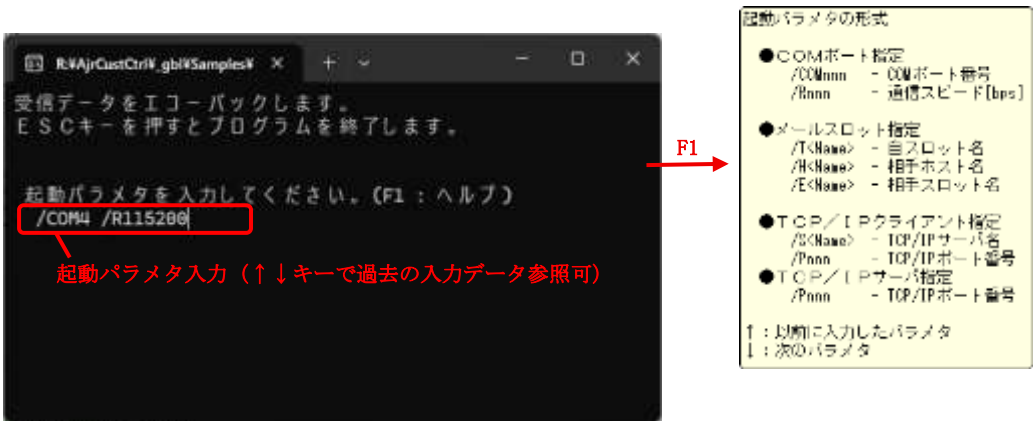

```
 1 : //
 2 : //  SW_SerialComPort2.c
 3 : //
 4 : #include    <AjrCstXX.h>
 5 : #include    <tchar.h>
 6 : #include    "resource.h"
 7 :
 8 : #define WM_SCPEVENT     (WM_USER + 100)
 9 :
10 : //-----------------------------------------------------------------------------------------------------------//
11 : //  ワーク                                                                                                   //
12 : //-----------------------------------------------------------------------------------------------------------//
13 : static  HINSTANCE       hInst;                          //  ＤＬＬインスタンスハンドル
14 : static  HWND            hDlgMain;                       //  ダイアログボックスハンドル
15 :
16 : static  HAJCSCP         hScp;
17 : static  UI              TotalBytes;
18 :
19 : //-----------------------------------------------------------------------------------------------------------//
20 : //  内部サブ関数                                                                                             //
21 : //-----------------------------------------------------------------------------------------------------------//
22 : AJC_DLGPROC_DEF(Main);
23 : static  VO      ShowLabel(HWND hDlg);
24 :
25 : //===========================================================================================================//
26 : //                                                                                                           //
27 : //  Ｗ ｉ ｎ Ｍ ａ ｉ ｎ                                                                                     //
28 : //                                                                                                           //
29 : //===========================================================================================================//
30 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
31 : {
32 :     MSG     msg;
33 :
34 :     hInst = hInstance;
35 :
36 :     //----- メイン・ダイアログオープン -----------------//
37 :     hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_DLGMAIN), NULL, AJC_DLGPROC_NAME(Main));
38 :     ShowWindow(hDlgMain, SW_SHOW);
39 :
40 :     //----- メッセージループ ---------------------------//
41 :     while (GetMessage(&msg, NULL, 0, 0)) {
42 :         do {
43 :             if (IsDialogMessage(hDlgMain, &msg)) break;
44 :             TranslateMessage(&msg);
45 :             DispatchMessage (&msg);
46 :         } while (0);
47 :     }
48 :
49 :     return (int)msg.wParam ;
50 : }
51 : //===========================================================================================================//
52 : //                                                                                                           //
53 : //  ダイアログ・プロシージャ                                                                                 //
54 : //                                                                                                           //
55 : //===========================================================================================================//
56 : //----- ダイアログ初期化 -----------------------------------------------------------------------------------//
57 : AJC_DLGPROC(Main, WM_INITDIALOG )
58 : {
59 :     hDlgMain   = hDlg;
```

```
60 :
61 :     AjcLoadWndPos(hDlg, NULL);                                          //  ウインド位置ロード
62 :
63 :     hScp = AjcScpCreateEx(TEXT("ComPortSect"), TRUE, NULL, NULL, NULL);     // インスタンス生成
64 :     AjcScpSetMode(hScp, hDlg, WM_SCPEVENT, AJCSCP_CM_BIN);              //  モード設定
65 :     AjcScpSetEvtMask(hScp, AJCSCP_EV_PORTSTATE |                        //  イベントマスク設定
66 :                            AJCSCP_EV_RXCHUNK);
67 :     ShowLabel(hDlg);
68 :     return TRUE;
69 : }
70 : //----- ウインド破棄 ----------------------------------------------------------------------------------------//
71 : AJC_DLGPROC(Main, WM_DESTROY    )
72 : {
73 :     AjcSaveWndPos(hDlg, NULL);                      //  ウインド位置セーブ
74 :     AjcScpDelete(hScp);                             //  インスタンス消去
75 :     PostQuitMessage(0);
76 :     return TRUE;
77 : }
78 : //----- ＳＣＭイベント通知 ----------------------------------------------------------------------------------//
79 : AJC_DLGPROC(Main, WM_SCPEVENT   )
80 : {
81 :     static  UI      TotalBytes = 0;
82 :     VOP     pDat;
83 :     UI      len, param;
84 :     UT      txt[512];
85 :
86 :     AjcScpGetEventData(hScp, lParam, &pDat, &len, &param);              //  イベントデータ取得
87 :     if (wParam == AJCSCP_EV_RXCHUNK ) {                                 //  ●チャンクデータ受信通知？
88 :         //  バイト数表示
89 :         TotalBytes += len;
90 :         AjcSepDlgItemUInt(hDlg, IDC_TXT_BYTES, TotalBytes);
91 :         //  データ返送
92 :         AjcScpSendBinData(hScp, pDat, len);
93 :     }
94 :     else if (wParam == AJCSCP_EV_PORTSTATE) {                           //  ●ポート状態通知？
95 :         switch (param) {
96 :             case AJCSCP_SRVSTART:                                       //      ・サーバ開始
97 :                 AjcTipTextShowCenter(hDlg, TEXT("¥n クライアントを接続してください ¥n "), 10000, NULL);
98 :                 break;
99 :             case AJCSCP_OPENFAIL:                                       //      ・オープン失敗
100 :                AjcSnPrintF(txt, AJCTSIZE(txt), TEXT("%s オープン／接続失敗"), pDat);
101 :                MessageBox(hDlg, txt, TEXT("SW_SerialComPort2"), MB_ICONERROR);
102 :                break;
103 :            case AJCSCP_SRVSRTFAIL:                                     //      ・サーバ開始失敗
104 :                MessageBox(hDlg, TEXT("サーバ開始失敗"), TEXT("SW_SerialComPort2"), MB_ICONERROR);
105 :                break;
106 :            case AJCSCP_MYSLOTFAIL:                                     //      ・自スロット生成失敗
107 :                AjcSnPrintF(txt, AJCTSIZE(txt), TEXT("自メールスロット (  %s ) の生成失敗"), pDat),
108 :                MessageBox(hDlg, txt, TEXT("SW_SerialComPort2"), MB_ICONERROR);
109 :                break;
110 :        }
111 :        ShowLabel(hDlg);
112 :    }
113 :    AjcScpRelEventData(hScp, lParam);                                   //  イベントデータ開放
114 :
115 :    return TRUE;
116 : }
117 : //----- ウインドクローズ ------------------------------------------------------------------------------------//
118 : AJC_DLGPROC(Main, IDCANCEL      )
119 : {
120 :    DestroyWindow(hDlg);
121 :    return TRUE;
122 : }
123 : //----- PORT OPEN/CLOSE ボタン ------------------------------------------------------------------------------//
124 : AJC_DLGPROC(Main, IDC_CMD_OPEN   )
125 : {
126 :    if (AjcScpIsOpened(hScp) || AjcScpIsServerStarted(hScp)) {
127 :        AjcScpClose(hScp);
128 :    }
129 :    else {
130 :        TotalBytes = 0;
131 :        AjcScpOpenDefault(hScp);
132 :        AjcSetDlgItemStr(hDlg, IDC_LBL_PORTSTATE, TEXT("Open in progress"));
133 :    }
134 :    return TRUE;
135 : }
136 : //----- ポート設定ボタン ------------------------------------------------------------------------------------//
137 : AJC_DLGPROC(Main, IDC_CMD_SETPORT)
138 : {
139 :    AjcScpDlgParamEasy(hScp, hDlg);
```

```
140 :     ShowLabel(hDlg);
141 :     return TRUE;
142 : }
143 : //-------------------------------------------------------------------------------------------------------------------//
144 : AJC_DLGMAP_DEF(Main)
145 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG  )
146 :     AJC_DLGMAP_MSG(Main, WM_DESTROY     )
147 :     AJC_DLGMAP_MSG(Main, WM_SCPEVENT    )
148 :     AJC_DLGMAP_CMD(Main, IDCANCEL       )
149 :     AJC_DLGMAP_CMD(Main, IDC_CMD_OPEN   )
150 :     AJC_DLGMAP_CMD(Main, IDC_CMD_SETPORT)
151 : AJC_DLGMAP_END
152 : //-------------------------------------------------------------------------------------------------------------------//
153 : //  各ラベル表示                                                                                                     //
154 : //-------------------------------------------------------------------------------------------------------------------//
155 : static  VO      ShowLabel(HWND hDlg)
156 : {
157 :     UI  ps;
158 :
159 :     switch (ps = AjcScpGetSelectedPort(hScp)) {
160 :         default:                //  ●ＣＯＭポート
161 :             if (AjcScpIsOpened(hScp)) {                                         //  オープン状態？
162 :                 AjcSetDlgItemStr (hDlg, IDC_LBL_PORTSTATE, TEXT("接続 状態"));
163 :                 AjcSetDlgItemFStr(hDlg, IDC_CMD_OPEN, TEXT("COM %d 切断"), ps);
164 :             }
165 :             else {                                                              //  クローズ状態
166 :                 AjcSetDlgItemStr (hDlg, IDC_LBL_PORTSTATE, TEXT("切断 状態"));
167 :                 AjcSetDlgItemFStr(hDlg, IDC_CMD_OPEN, TEXT("COM %d 接続"), ps);
168 :             }
169 :             AjcSetDlgItemStr(hDlg, IDC_LBL_SERV, TEXT(""));
170 :             break;
171 :         case AJCSCP_PS_SLOT:    //  ●メールスロット
172 :             if (AjcScpIsOpened(hScp)) {                                         //  オープン状態？
173 :                 AjcSetDlgItemStr(hDlg, IDC_LBL_PORTSTATE, TEXT("接続 状態"));
174 :                 AjcSetDlgItemStr(hDlg, IDC_CMD_OPEN, TEXT("スロット 切断"));
175 :             }
176 :             else {                                                              //  クローズ状態
177 :                 AjcSetDlgItemStr (hDlg, IDC_LBL_PORTSTATE, TEXT("切断 状態"));
178 :                 AjcSetDlgItemFStr(hDlg, IDC_CMD_OPEN, TEXT("スロット 接続"));
179 :             }
180 :             AjcSetDlgItemStr(hDlg, IDC_LBL_SERV, TEXT(""));
181 :             break;
182 :         case AJCSCP_PS_CLIENT:  //  ●ソケット（クライアント）
183 :             if (AjcScpIsOpened(hScp)) {                                         //  オープン状態？
184 :                 AjcSetDlgItemStr(hDlg, IDC_LBL_PORTSTATE, TEXT("接続 状態"));
185 :                 AjcSetDlgItemStr(hDlg, IDC_CMD_OPEN, TEXT("サーバから切断"));
186 :             }
187 :             else {                                                              //  クローズ状態
188 :                 AjcSetDlgItemStr (hDlg, IDC_LBL_PORTSTATE, TEXT("切断 状態"));
189 :                 AjcSetDlgItemFStr(hDlg, IDC_CMD_OPEN, TEXT("サーバへ接続"));
190 :             }
191 :             AjcSetDlgItemStr(hDlg, IDC_LBL_SERV, TEXT(""));
192 :             break;
193 :         case AJCSCP_PS_SERVER:  //  ●ソケット（サーバ）
194 :             if (AjcScpIsServerStarted(hScp)) {                                  //  サーバ起動状態？
195 :                 if (AjcScpIsOpened(hScp)) AjcSetDlgItemStr(hDlg, IDC_LBL_PORTSTATE, TEXT("接続 状態"));
196 :                 else                      AjcSetDlgItemStr(hDlg, IDC_LBL_PORTSTATE, TEXT("切断 状態"));
197 :                 AjcSetDlgItemStr(hDlg, IDC_CMD_OPEN, TEXT("サーバ 停止"));
198 :             }
199 :             else {                                                              //  サーバ停止状態
200 :                 AjcSetDlgItemStr(hDlg, IDC_LBL_PORTSTATE, TEXT("切断 状態"));
201 :                 AjcSetDlgItemStr(hDlg, IDC_CMD_OPEN, TEXT("サーバ 起動"));
202 :             }
203 :             if (AjcScpIsServerStarted(hScp)) AjcSetDlgItemStr(hDlg, IDC_LBL_SERV, TEXT("サーバ動作中"));
204 :             else                             AjcSetDlgItemStr(hDlg, IDC_LBL_SERV, TEXT("サーバ停止中"));
205 :             break;
206 :     }
207 :     AjcSetDlgItemStr(hDlg, IDC_LBL_PORTNAME, AjcScpGetPortPathName(hScp));      //  ポート名表示
208 : }
209 :
```

13.11.3. SW_SerialComPort2C（エコーバック – コンソールアプリ）

このサンプルプログラムは、SW_SerialComPort2 と同様に受信データをエコーバック（受信したデータをそのまま返信）を行うコンソールアプリケーションです。



起動パラメタを入力すると、当該通信ポートをオープン後、エコーバックを開始し、受信バイト数を表示します。
起動パラメタは、Enter キーで確定します。
起動パラメタの形式は、以下のとおりです。

ＣＯＭポートの場合、ＣＯＭポート番号と通信速度を指定します。（ポート番号と通信速度以外はデフォルト値）

```
/COM3  /R115200
```

メールスロットの場合、自スロット名と相手スロット名を指定します。

```
/TMySlotName  /ERemoteSlotName
```

TCP/IP クライアントの場合、接続先サーバ名とポート番号を指定します。

```
/SServerName  /P14238
```

TCP/IP サーバの場合、ポート番号を指定します。

```
/P14238
```

起動パラメタは、コマンドライン上でも指定可能です。

いずれかのキーを押すとプログラムを終了します。

```
 1 : //
 2 : //  SW_SerialComPort2C.c
 3 : //
 4 : #include    <AjrCstXX.h>
 5 : #include    <conio.h>
 6 :
 7 : //  ヘルプテキスト
 8 : static  UT          HelpText[] = TEXT("起動パラメタの形式¥n¥n"                  )
 9 :                                  TEXT("  ●ＣＯＭポート指定¥n"                 )
10 :                                  TEXT("      /COMnnn    - COM ポート番号¥n"     )
11 :                                  TEXT("      /Rnnn      - 通信スピード[bps]¥n¥n")
12 :                                  TEXT("  ●メールスロット指定¥n"               )
13 :                                  TEXT("      /T<Name>   - 自スロット名¥n"       )
14 :                                  TEXT("      /H<Name>   - 相手ホスト名¥n"       )
15 :                                  TEXT("      /E<Name>   - 相手スロット名¥n¥n"   )
16 :                                  TEXT("  ●ＴＣＰ／ＩＰクライアント指定¥n"    )
17 :                                  TEXT("      /S<Name>   - TCP/IP サーバ名¥n"    )
18 :                                  TEXT("      /Pnnn      - TCP/IP ポート番号¥n"  )
19 :                                  TEXT("  ●ＴＣＰ／ＩＰサーバ指定¥n"          )
20 :                                  TEXT("      /Pnnn      - TCP/IP ポート番号¥n"  )
21 :                                  TEXT("¥n"                                  )
22 :                                  TEXT("↑：以前に入力したパラメタ¥n"          )
23 :                                  TEXT("↓：次のパラメタ"                      );
24 : //  シリアル通信インスタンス
25 : static  HAJCSCP     hScp = NULL;
26 : //  通信デバイス
27 : typedef enum {COMPORT, MAILSLOT, CLIENT, SERV}  COMMDEV;
28 : //  通信パラメタ
29 : static COMMDEV       dev     = COMPORT;              //  通信デバイス
30 : static UI            ComPort = 1;                    //  ＣＯＭポート番号
31 : static UI            Speed   = 115200;               //  ＣＯＭポート速度
32 : static UT            MySlot [256] = TEXT("NoName");  //  自スロット名
33 : static UT            RmtSlot[256] = TEXT("SlotB");   //  相手スロット名
34 : static UT            RmtHost[256] = TEXT("");        //  相手ホスト名
35 : static UT            TcpServ[256] = TEXT("");        //  TCP/IP サーバ名
36 : static UI            TcpPort = 14238;                //  TCP/IP ポート番号
37 : //  バイトカウントタ
38 : static  UI          ByteCount = 0;
39 :
40 : //----- コンソール入力のコールバック (起動パラメタ解析) ----------------------------------------------------//
41 : static VO   CALLBACK cbNtcArgs(int argc, UT *argv[], C_UTP pTxt, UX cbp)
42 : {
43 :     int     i;
44 :
45 :     for (i = 0; i < argc; i++) {
46 :         if      (MAjcStrNICmp(argv[i], TEXT("/COM"), 4) == 0) {dev = COMPORT;  ComPort = AjcAscToInt(   argv[i] + 4);}
47 :         else if (MAjcStrNICmp(argv[i], TEXT("/R"  ), 2) == 0) {dev = COMPORT;  Speed   = AjcAscToInt(   argv[i] + 2);}
48 :         else if (MAjcStrNICmp(argv[i], TEXT("/T"  ), 2) == 0) {dev = MAILSLOT; MAjcStrCpy(MySlot , 256, argv[i] + 2);}
49 :         else if (MAjcStrNICmp(argv[i], TEXT("/H"  ), 2) == 0) {dev = MAILSLOT; MAjcStrCpy(RmtHost, 256, argv[i] + 2);}
50 :         else if (MAjcStrNICmp(argv[i], TEXT("/E"  ), 2) == 0) {dev = MAILSLOT; MAjcStrCpy(RmtSlot, 256, argv[i] + 2);}
51 :         else if (MAjcStrNICmp(argv[i], TEXT("/S"  ), 2) == 0) {dev = CLIENT;   MAjcStrCpy(TcpServ, 256, argv[i] + 2);}
52 :         else if (MAjcStrNICmp(argv[i], TEXT("/P"  ), 2) == 0) {dev = CLIENT;   TcpPort = AjcAscToInt(   argv[i] + 2);}
53 :         else {
54 :             AjcPrintF(TEXT("*** Invalid parameter %s ***¥n"), argv[i]);
55 :         }
56 :     }
57 :     if (dev == CLIENT && TcpServ[0] == 0) {
58 :         dev = SERV;
59 :     }
60 : }
61 :
62 : //----- コンソール強制終了ハンドラ ------------------------------------------------------------------------//
63 : static BOOL CALLBACK cbConApExit(DWORD CtrlType)
64 : {
65 :     //  シリアル通信インスタンス消去
66 :     if (hScp != NULL) {
67 :         AjcScpDelete(hScp);
68 :         hScp = NULL;
69 :     }
70 :     return FALSE;   //  FALSE : 次のハンドラをコール
71 : }
72 : //-------------------------------------------------------------------------------------------------------//
73 : //  主制御
74 : int  AjcMain(int argc, UTP argv[])
75 : {
76 :     WPARAM      wParam;
77 :     LPARAM      lParam;
78 :     UTP         pDat;
79 :     UI          lDat;
80 :     UI          param;
81 :     BOOL        rsu;
82 :     BOOL        fExit = FALSE;
83 :     UT          buf[256] = {0};
84 :
85 :     AjcSetStdoutMode();
86 :
87 :     do {
88 :         //  コンソール強制終了ハンドラ設定
89 :         SetConsoleCtrlHandler(cbConApExit, TRUE);
90 :         //  初期メッセージ
```

```
91 :         AjcPrintF(TEXT("受信データをエコーバックします。¥nＥＳＣキーを押すとプログラムを終了します。¥n¥n"));
92 :         //  コマンドパラメタ解析／入力
93 :         if (argc >= 2) {
94 :             cbNtcArgs(argc - 1, &argv[1], GetCommandLine(), 0);
95 :         }
96 :         else {
97 :             AjcPrintF(TEXT("¥n 起動パラメタを入力してください。(F1 : ヘルプ)¥n  "));
98 :             if (!AjcConInputEx(0, buf, AJCTSIZE(buf), HelpText, 0, cbNtcArgs)) {
99 :                 break;
100 :            }
101 :        }
102 :        AjcPrintF(TEXT("¥n"));
103 :        //  シリアル通信インスタンス生成
104 :        hScp = AjcScpCreateEx(TEXT("ScpParam"), TRUE, MySlot, RmtHost, RmtSlot);
105 :        //  モード設定 (TRUE : 受信チャンクデータをバイナリデータとして扱う)
106 :        AjcScpSetMode(hScp, NULL, 0, AJCSCP_CM_BIN);
107 :        //  使用するイベント設定
108 :        AjcScpSetEvtMask(hScp, AJCSCP_EV_RXCHUNK | AJCSCP_EV_PORTSTATE);
109 :
110 :        //  ポートオープン
111 :        switch (dev) {
112 :            case COMPORT:   rsu = AjcScpOpen (hScp, ComPort, Speed, 8, 'N', 1); break;
113 :            case MAILSLOT:  AjcScpSetMailSlotNames(hScp, MySlot, NULL, RmtSlot);
114 :                            rsu = AjcScpOpenSlot(hScp, NULL, RmtSlot);          break;
115 :            case CLIENT:    rsu = AjcScpOpenSock(hScp, TcpServ, TcpPort);       break;
116 :            case SERV:      rsu = AjcScpOpenServ(hScp, TcpPort);                break;
117 :        }
118 :        if (rsu) {                                                              //  ポート オープン成功？
119 :            while (!fExit) {                                                    //    キー押下までループ
120 :                if (AjcScpWaitEvent(hScp, &wParam, &lParam, 200)) {             //      イベント待ち，イベントあり？
121 :                    C_UTP   pPortName = AjcScpGetPortPathName(hScp);
122 :                    AjcScpGetEventData(hScp, lParam, &pDat, &lDat, &param);     //        イベントデータ取得
123 :                    switch (wParam) {
124 :                        case AJCSCP_EV_PORTSTATE:                               //        ●ポート状態通知
125 :                            switch (param) {
126 :                                case AJCSCP_CLOSED:     // クローズ状態
127 :                                    break;
128 :                                case AJCSCP_OPENED:     // オープン状態
129 :                                    AjcPrintF(TEXT("ポート %s をオープンしました。¥n"), pPortName);
130 :                                    AjcPrintF(TEXT("Received bytes count:¥n    ¥r"));
131 :                                    break;
132 :                                case AJCSCP_OPENFAIL:   // オープン失敗
133 :                                    AjcPrintF(TEXT("ポート %s のオープンを失敗しました。¥n"), pPortName);
134 :                                    fExit = TRUE;
135 :                                    break;
136 :                            }
137 :                            break;
138 :                        case AJCSCP_EV_RXCHUNK:                                 //        ●バイナリチャンク受信通知
139 :                            AjcScpSendBinData(hScp, pDat, lDat);                //          エコーバック送信
140 :                            ByteCount += lDat;
141 :                            AjcPrintF(TEXT("    %u¥r"), ByteCount);
142 :                            break;
143 :
144 :                    }
145 :                    AjcScpRelEventData(hScp, lParam);                           //        イベントデータ開放
146 :                }
147 :                //  キー入力チェック
148 :                if (_kbhit()) fExit = TRUE;
149 :            }
150 :            AjcScpClose(hScp);                                                  //    ＣＯＭポートクローズ
151 :        }
152 :        else {                                                                  //  ＣＯＭポート オープン失敗？
153 :            AjcPrintF(TEXT("%s open failure!¥n"), AjcScpGetPortPathName(hScp));  //      エラーメッセージ
154 :        }
155 :        AjcScpDelete(hScp);                                                     //  シリアル通信インスタンス消去
156 :    } while(0);
157 :
158 :    AjcPrintF(TEXT("¥nHit Enter Key!!"));
159 :    getchar();
160 :    return 0;
161 : }
```

13.11.4. SW_SerialComPort3（ループバックによるフォルダ構造コピー）

このサンプルプログラムは、エコーサーバを介して、入力フォルダ下のフォルダ構造と全ファイルを、出力フォルダ下に複製します。
入力フォルダ下の全ファイルをエコーサーバに送信し、返信されたデータで出力フォルダ下にフォルダ構造とファイルを複製します。
複製したファイルは、元のファイルとコンペアし、一致／不一致をログ表示します。
複製する前に、出力フォルダ下の全フォルダやファイルは消去されます。



※エコーサーバには、以下のサンプルプログラムを使用できます。
・SW_SeialComPort2 --- COM ポート／メールスロット接続時
・SW_SockServer2 ---- TCP/IP 接続時

```
  1 : //
  2 : //  SW_SerialComPort3.c
  3 : //
  4 :
  5 : #include    <AjrCstXX.h>
  6 : #include    <tchar.h>
  7 : #include    "resource.h"
  8 :
  9 : #define WM_SCPEVENT     (WM_USER + 100)
 10 :
 11 : //-------------------------------------------------------------------------------------------------------------//
 12 : //  ワーク                                                                                                     //
 13 : //-------------------------------------------------------------------------------------------------------------//
 14 : static  HINSTANCE       hInst;                                  //  ＤＬＬインスタンスハンドル
 15 : static  HWND            hDlgMain        = NULL;                 //  ダイアログボックスハンドル
 16 : static  HWND            hVthLog         = NULL;                 //  ログウインド
 17 :
 18 : static  HAJCSCP         hScp            = NULL;                 //  シリアル通信インスタンスハンドル
 19 : static  HBITMAP         hBmpProgImage   = NULL;                 //  ビットマップハンドル
 20 : static  HAJCVQUE        hVQue           = NULL;                 //  キューハンドル（入出力パス蓄積）
 21 : static  HANDLE          fhInp           = INVALID_HANDLE_VALUE; //  入力ファイルハンドル
 22 : static  HANDLE          fhOut           = INVALID_HANDLE_VALUE; //  出力ファイルハンドル
 23 : static  ULL             FileSize        = 0;                    //  入力ファイルサイズ
 24 : static  UI              Unmatch         = 0;                    //  比較不一致ファイル数
 25 : static  UI              ListCount       = 0;                    //  全ファイル数
 26 : static  UI              TxCount         = 0;                    //  送信バイトカウンタ
 27 : static  UI              RxCount         = 0;                    //  受信バイトカウンタ
 28 : static  BOOL            fSendBusy       = FALSE;                //  処理中フラグ
 29 : static  UT              InpDir[MAX_PATH];                       //  入力ディレクトリパス
 30 : static  UT              OutDir[MAX_PATH];                       //  出力ディレクトリパス
 31 : static  UT              InpFilePath[MAX_PATH];                  //  入力ファイルパス
 32 : static  UT              OutFilePath[MAX_PATH];                  //  出力ファイルパス
 33 :
 34 : //-------------------------------------------------------------------------------------------------------------//
 35 : //  内部サブ関数                                                                                               //
 36 : //-------------------------------------------------------------------------------------------------------------//
 37 : AJC_DLGPROC_DEF(Main);
 38 : static  VO      EnableButtons(HWND hDlg, BOOL fbtnParam, BOOL fbtnOpen, BOOL fbtnStart, BOOL fbtnStop);
 39 : static  BOOL    SendNextFile(VO);
 40 : static  BOOL    FileOpen(VO);
 41 : static  VO      FileSend(VO);
 42 : static  BOOL CALLBACK cbFind(UI nest, UTP pPath, UTP pName, UI attrib, UI wtime, UX cbp);
 43 : static  VO      ShowLabel(HWND hDlg);
 44 :
 45 : //=============================================================================================================//
 46 : //                                                                                                             //
 47 : //  Ｗｉｎ Ｍａｉｎ                                                                                           //
 48 : //                                                                                                             //
 49 : //=============================================================================================================//
```

```
int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
{
    MSG     msg;

    hInst = hInstance;

    //----- メイン・ダイアログオープン -----------------//
    hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_DLGMAIN), NULL, AJC_DLGPROC_NAME(Main));
    ShowWindow(hDlgMain, SW_SHOW);

    //----- メッセージループ ---------------------------//
    while (GetMessage(&msg, NULL, 0, 0)) {
        do {
            if (IsDialogMessage(hDlgMain, &msg)) break;
            TranslateMessage(&msg);
            DispatchMessage (&msg);
        } while (0);
    }

    return (int)msg.wParam ;
}
//=============================================================================================================//
//                                                                                                             //
//  ダイアログ・プロシージャ                                                                                   //
//                                                                                                             //
//=============================================================================================================//
//----- ダイアログ初期化 -----------------------------------------------------------------------------------//
AJC_DLGPROC(Main, WM_INITDIALOG )
{
    hDlgMain   = hDlg;
    hVthLog    = GetDlgItem(hDlg, IDC_VTH);

    //  ウインド位置ロード
    AjcLoadWndPos(hDlg, NULL);

    hScp = AjcScpCreateEx(TEXT("ComPortSect"), TRUE, NULL, NULL, NULL);     //  インスタンス生成
    AjcScpSetMode(hScp, hDlg, WM_SCPEVENT, AJCSCP_CM_BIN);                  //  モード設定
    AjcScpSetEvtMask(hScp, AJCSCP_EV_PORTSTATE |                            //  イベントマスク設定
                           AJCSCP_EV_RXCHUNK | AJCSCP_EV_RXPKT | AJCSCP_EV_TXEMPTY);
    AjcScpSetPktTimeout(hScp, 60000);   //  パケット受信タイムスト（低速通信用に長めに設定）
    //  ビットマップ表示
    AjcChangeBitmapColor((hBmpProgImage = LoadBitmap(hInst, MAKEINTRESOURCE(IDB_PROGIMAGE ))),
                                                  RGB(255, 255, 255), GetSysColor(COLOR_BTNFACE));
    SendDlgItemMessage(hDlg, IDC_PIC, STM_SETIMAGE, IMAGE_BITMAP, (LPARAM)hBmpProgImage);
    //  キュー生成
    hVQue = AjcVQueCreate(0, NULL);
    //  ダイアログ項目の関連付け
    MAjcSetDlgPropStr (hDlg, IDC_TXT_INPPATH  , TEXT("" ), NULL , 0);
    MAjcSetDlgPropStr (hDlg, IDC_TXT_OUTPATH  , TEXT("" ), NULL , 0);
    AjcLoadDlgProfiles(hDlg, TEXT("DlgSetting"));
    //  ポート名表示
    ShowLabel(hDlg);

    return TRUE;
}
//----- ウインド破棄 ---------------------------------------------------------------------------------------//
AJC_DLGPROC(Main, WM_DESTROY    )
{
    //  ウインド位置セーブ
    AjcSaveWndPos(hDlg, NULL);
    //  ダイアログ項目のセーブ
    AjcSaveDlgProfiles(hDlg, TEXT("DlgSetting"));
    AjcReleaseDlgProps(hDlg);
    //  リソース解放
    if (hBmpProgImage != NULL) DeleteObject(hBmpProgImage);
    if (hScp          != NULL) AjcScpDelete(hScp);
    if (hVQue         != NULL) AjcVQueDelete(hVQue);
    //  プログラム終了
    PostQuitMessage(0);
    return TRUE;
}
//----- ＳＣＭイベント通知 ---------------------------------------------------------------------------------//
AJC_DLGPROC(Main, WM_SCPEVENT   )
{
    static  UI      TotalBytes = 0;
    VOP     pDat;
    UI      len, param;
    UT      txt[512];

    AjcScpGetEventData(hScp, lParam, &pDat, &len, &param);              //  イベントデータ取得
    if(wParam & AJCSCP_EV_PORTSTATE) {                                  //  ●ポート状態通知
        AjcTipTextHide();
        //  ポート名表示
        AjcSetDlgItemStr(hDlg, IDC_LBL_PORTNAME, (UTP)pDat);
        switch (param) {
            case AJCSCP_OPENED:     // オープン状態
                EnableButtons(hDlg, FALSE, TRUE, TRUE, FALSE);
                break;
            case AJCSCP_CLOSED:     // クローズ状態
                if (fSendBusy) {
```

```
140 :                     SendMessage(hDlg, WM_COMMAND, MAKELONG(IDC_CMD_STOP, BN_CLICKED), (LPARAM)hDlg);
141 :                 }
142 :                 EnableButtons(hDlg, TRUE, TRUE, FALSE, FALSE);
143 :                 break;
144 :             case AJCSCP_SRVSTART:                                   //      ・サーバ開始
145 :                 AjcTipTextShowCenter(hDlg, TEXT("¥n エコー クライアントを接続してください ¥n "), 10000, NULL);
146 :                 break;
147 :             case AJCSCP_OPENFAIL:                                   //      ・オープン失敗
148 :                 AjcSnPrintF(txt, AJCTSIZE(txt), TEXT("%s オープン／接続失敗"), pDat);
149 :                 MessageBox(hDlg, txt, TEXT("SW_SerialComPort3"), MB_ICONERROR);
150 :                 break;
151 :             case AJCSCP_SRVSRTFAIL:                                 //      ・サーバ開始失敗
152 :                 MessageBox(hDlg, TEXT("サーバ開始失敗"), TEXT("SW_SerialComPort3"), MB_ICONERROR);
153 :                 break;
154 :             case AJCSCP_MYSLOTFAIL:                                 //      ・自スロット生成失敗
155 :                 AjcSnPrintF(txt, AJCTSIZE(txt), TEXT("自メールスロット (  %s ) の生成失敗"), pDat),
156 :                 MessageBox(hDlg, txt, TEXT("SW_SerialComPort3"), MB_ICONERROR);
157 :                 break;
158 :         }
159 :         ShowLabel(hDlg);
160 :     }
161 :     else if (wParam & AJCSCP_EV_RXCHUNK) {                          //  ●バイナリチャンク受信通知
162 :         if (param == 0) {
163 :             RxCount += len;
164 :             AjcSepDlgItemUInt(hDlg, IDC_LBL_BYTES_RX, RxCount);
165 :         }
166 :     }
167 :     else if (wParam & AJCSCP_EV_RXPKT) {                            //  ●パケット受信通知
168 :         //  空パケット（ファイル終端）以外ならば、受信データをファイルへ出力
169 :         if (len != 0) {
170 :             if (fhOut != INVALID_HANDLE_VALUE) {
171 :                 UL  bytes;
172 :                 WriteFile(fhOut, pDat, len, &bytes, NULL);
173 :             }
174 :         }
175 :         //  空パケット（ファイル終端）ならば、次のファイル送信へ・・
176 :         else {
177 :             //  出力ファイルクローズ
178 :             if (fhOut != INVALID_HANDLE_VALUE) {CloseHandle(fhOut); fhOut = INVALID_HANDLE_VALUE;}
179 :             //  ファイルコンペア
180 :             if (AjcFileCompare(InpFilePath, OutFilePath)) {
181 :                 AjcVthPrintF(hVthLog, TEXT("OK (Match)¥n"));
182 :             }
183 :             else {
184 :                 Unmatch++;
185 :                 AjcVthPrintF(hVthLog, TEXT("¥x1B[31mNG (Unmatch)¥x1B[0m¥n"));
186 :             }
187 :             //  次のファイル送信開始
188 :             SendNextFile();
189 :         }
190 :     }
191 :     else if (wParam & AJCSCP_EV_TXEMPTY) {                          //  ●送信完了通知
192 :         //  次のファイルデータ送信
193 :         if (fhInp != INVALID_HANDLE_VALUE) {
194 :             FileSend();
195 :         }
196 :     }
197 :
198 :     AjcScpRelEventData(hScp, lParam);                               //  イベントデータ開放
199 :
200 :     return TRUE;
201 : }
202 : //----- キャンセル ----------------------------------------------------------------------------------------//
203 : AJC_DLGPROC(Main, IDCANCEL      )
204 : {
205 :     DestroyWindow(hDlg);
206 :     return TRUE;
207 : }
208 : //----- ポート設定ボタン ----------------------------------------------------------------------------------//
209 : AJC_DLGPROC(Main, IDC_CMD_PARAM )
210 : {
211 :     if (HIWORD(wParam) == BN_CLICKED) {
212 :         AjcScpDlgParamEasy(hScp, hDlg);
213 :         ShowLabel(hDlg);
214 :     }
215 :     return TRUE;
216 : }
217 : //----- オープンボタン ------------------------------------------------------------------------------------//
218 : AJC_DLGPROC(Main, IDC_CMD_OPEN  )
219 : {
220 :     if (HIWORD(wParam) == BN_CLICKED) {
221 :         if (AjcScpIsOpened(hScp) || AjcScpIsServerStarted(hScp)) {
222 :             AjcScpClose(hScp);
223 :         }
224 :         else {
225 :             AjcScpOpenDefault(hScp);
226 :             EnableButtons(hDlg, FALSE, TRUE, FALSE, FALSE);
227 :             AjcSetDlgItemStr(hDlg, IDC_LBL_PORTSTATE, TEXT("Open in progress..."));
228 :         }
229 :     }
```

```
230 :     return TRUE;
231 : }
232 : //----- 開始ボタン ----------------------------------------------------------------------------------------------//
233 : AJC_DLGPROC(Main, IDC_CMD_START )
234 : {
235 :     if (HIWORD(wParam) == BN_CLICKED) {
236 :         //  入出力パス設定
237 :         AjcGetDlgItemStr(hDlg, IDC_TXT_INPPATH, InpDir, MAX_PATH);
238 :         AjcGetDlgItemStr(hDlg, IDC_TXT_OUTPATH, OutDir, MAX_PATH);
239 :         if (InpDir[0] != 0 && OutDir[0] != 0) {
240 :             if (MAjcStrICmp(InpDir, OutDir) != 0) {
241 :                 //  入出力パスリストクリアー
242 :                 AjcVQuePurge(hVQue);
243 :                 //  ファイル検索 (入出力パスリスト作成)
244 :                 ListCount = 0;
245 :                 Unmatch = 0;
246 :                 AjcSearchFilesEx(InpDir, TEXT("*.*"), TRUE, FALSE, 0, cbFind);
247 :                 //  フォルダ構造コピー
248 :                 if (ListCount != 0) {
249 :                     UT      txt[MAX_PATH * 64];
250 :                     //  出力フォルダ下の全ファイル削除
251 :                     AjcSnPrintF(txt, AJCTSIZE(txt), TEXT("%s 下の全ファイルを削除します。よろしいですか？"), OutDir);
252 :                     if (MessageBox(hDlg, txt, TEXT("SW_SerialComPort4"), MB_YESNO) == IDYES) {
253 :                         //  ファイル番号クリアー
254 :                         ListCount = 0;
255 :                         //  バイトカウンタクリアー
256 :                         AjcSepDlgItemUInt(hDlg, IDC_LBL_BYTES_TX, TxCount = 0);
257 :                         AjcSepDlgItemUInt(hDlg, IDC_LBL_BYTES_RX, RxCount = 0);
258 :                         //  ログクリアー
259 :                         AjcVthClear(hVthLog);
260 :                         //  ボタングレー化
261 :                         EnableButtons(hDlg, FALSE, FALSE, FALSE, TRUE);
262 :                         //  フォルダ下クリアー
263 :                         AjcCleanFolder(OutDir, 0, NULL);
264 :                         //  フォルダ構造コピー
265 :                         AjcCopyFolderStruct(InpDir, OutDir, AJCFSC_FORCETIME);
266 :                         //  ファイルをオープンし送信開始
267 :                         SendNextFile();
268 :                     }
269 :                 } else AjcVthPutText(hVthLog, TEXT("¥x1B[31m 入力フォルダ下にファイルがありません。¥x1B[0m"), -1);
270 :
271 :             } else AjcVthPutText(hVthLog,TEXT("¥x1B[31m 入出力フォルダが同じです。¥x1B[0m"), -1);
272 :
273 :         } else AjcVthPutText(hVthLog, TEXT("¥x1B[31m 入出力フォルダが設定されていません。¥x1B[0m"), -1);
274 :     }
275 :     return TRUE;
276 : }
277 : //----- 中止ボタン ----------------------------------------------------------------------------------------------//
278 : AJC_DLGPROC(Main, IDC_CMD_STOP  )
279 : {
280 :     if (HIWORD(wParam) == BN_CLICKED) {
281 :         //  送信中フラグクリアー
282 :         fSendBusy = FALSE;
283 :         //  ポートクローズ
284 :         AjcScpClose(hScp);
285 :         //  ファイルクローズ
286 :         if (fhInp != INVALID_HANDLE_VALUE) {CloseHandle(fhInp); fhInp = INVALID_HANDLE_VALUE;}
287 :         if (fhOut != INVALID_HANDLE_VALUE) {CloseHandle(fhOut); fhOut = INVALID_HANDLE_VALUE;}
288 :         //  ログメッセージ
289 :         AjcVthPrintF(hVthLog, TEXT("¥n¥n--- ループバックテストをキャンセルしました ---¥n"));
290 :     }
291 :     return TRUE;
292 : }
293 : //----- 入力フォルダ設定ボタン ----------------------------------------------------------------------------------//
294 : AJC_DLGPROC(Main, IDC_CMD_INPPATH)
295 : {
296 :     if (HIWORD(wParam) == BN_CLICKED) {
297 :         UT  path[MAX_PATH];
298 :         AjcGetDlgItemStr(hDlg, IDC_TXT_INPPATH, path, MAX_PATH);
299 :         if (AjcGetFolderName(hDlg, TEXT("入力フォルダ"), path, path, TRUE)) {
300 :             AjcSetDlgItemStr(hDlg, IDC_TXT_INPPATH, path);
301 :         }
302 :     }
303 :     return TRUE;
304 : }
305 : //----- 出力フォルダ設定ボタン ----------------------------------------------------------------------------------//
306 : AJC_DLGPROC(Main, IDC_CMD_OUTPATH)
307 : {
308 :     if (HIWORD(wParam) == BN_CLICKED) {
309 :         UT  path[MAX_PATH];
310 :         AjcGetDlgItemStr(hDlg, IDC_TXT_INPPATH, path, MAX_PATH);
311 :         if (AjcGetFolderName(hDlg, TEXT("出力フォルダ"), path, path, TRUE)) {
312 :             AjcSetDlgItemStr(hDlg, IDC_TXT_OUTPATH, path);
313 :         }
314 :     }
315 :     return TRUE;
316 : }
317 : //-------------------------------------------------------------------------------------------------------------//
318 : AJC_DLGMAP_DEF(Main)
319 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG  )
```

```
320 :     AJC_DLGMAP_MSG(Main, WM_DESTROY     )
321 :     AJC_DLGMAP_MSG(Main, WM_SCPEVENT    )
322 :     AJC_DLGMAP_CMD(Main, IDCANCEL       )
323 :     AJC_DLGMAP_CMD(Main, IDC_CMD_PARAM  )
324 :     AJC_DLGMAP_CMD(Main, IDC_CMD_OPEN   )
325 :     AJC_DLGMAP_CMD(Main, IDC_CMD_START  )
326 :     AJC_DLGMAP_CMD(Main, IDC_CMD_STOP   )
327 :     AJC_DLGMAP_CMD(Main, IDC_CMD_INPPATH)
328 :     AJC_DLGMAP_CMD(Main, IDC_CMD_OUTPATH)
329 : AJC_DLGMAP_END
330 :
331 : //-----------------------------------------------------------------------------------------------------------------------//
332 : //  ファイル検索通知                                                                                                     //
333 : //-----------------------------------------------------------------------------------------------------------------------//
334 : static  BOOL CALLBACK cbFind(UI nest, UTP pPath, UTP pName, UI attrib, UI wtime, UX cbp)
335 : {
336 :     UT      out  [MAX_PATH];
337 :     UT      inout[MAX_PATH * 2 + 16];
338 :
339 :     if (!(attrib & _A_SUBDIR)) {
340 :         ListCount++;
341 :         //  入出力ファイルパス設定 (セミコロン(;)で区切る)
342 :         MAjcStrCpy(out, MAX_PATH, OutDir);
343 :         AjcPathCat(out, pPath + MAjcStrLen(InpDir), MAX_PATH);
344 :         AjcSnPrintF(inout, AJCTSIZE(inout), TEXT("%s;%s"), pPath, out);
345 :         AjcVQueEnque(hVQue, (C_VOP)inout, ((UI)MAjcStrLen(inout) + 1) * sizeof(UT));
346 :     }
347 :     return TRUE;
348 : }
349 : //-----------------------------------------------------------------------------------------------------------------------//
350 : //  次のファイル送信                                                                                                     //
351 : //-----------------------------------------------------------------------------------------------------------------------//
352 : static  BOOL    SendNextFile(VO)
353 : {
354 :     BOOL    rc = FALSE;
355 :
356 :     if (FileOpen()) {
357 :         fSendBusy = TRUE;
358 :         FileSend();
359 :         rc = TRUE;
360 :     }
361 :     else {
362 :         fSendBusy = FALSE;
363 :         AjcScpClose(hScp);
364 :         AjcVthPrintF(hVthLog, TEXT("¥n--- ループバックテストを終了しました ---¥n"));
365 :         AjcVthPrintF(hVthLog, TEXT("    ループバック不一致数 : %d / %d[files]¥n"), Unmatch, ListCount);
366 :     }
367 :     return rc;
368 : }
369 : //-----------------------------------------------------------------------------------------------------------------------//
370 : //  ファイルオープン (TRUE:ゼロサイズ以外のファイルオープン, FALSE:ファイルなし)                                        //
371 : //-----------------------------------------------------------------------------------------------------------------------//
372 : static  BOOL    FileOpen(VO)
373 : {
374 :     BOOL    rc = FALSE;
375 :     UTP     pInp;
376 :     UTP     pOut;
377 :     UT      inout[MAX_PATH * 2 + 16];
378 :
379 :     while (AjcVQueDeque(hVQue, (VOP)inout, sizeof(inout))) {
380 :         ListCount++;
381 :         //  入出力パス名設定
382 :         pInp = MAjcStrTok(inout, TEXT(";")); if (pInp != NULL) MAjcStrCpy(InpFilePath, MAX_PATH, pInp);
383 :         pOut = MAjcStrTok(NULL , TEXT(";")); if (pOut != NULL) MAjcStrCpy(OutFilePath, MAX_PATH, pOut);
384 :         //  ログ表示
385 :         AjcVthPrintF(hVthLog, TEXT("¥n%3d : %s¥n"), ListCount, pInp);
386 :         AjcVthPrintF(hVthLog, TEXT("   -> %s - "), pOut);
387 :         //  入出力ファイルオープン
388 :         fhInp = CreateFile(pInp, GENERIC_READ , 0, NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
389 :         fhOut = CreateFile(pOut, GENERIC_WRITE, 0, NULL, CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
390 :         if (fhInp != INVALID_HANDLE_VALUE && fhOut != INVALID_HANDLE_VALUE) {
391 :             //  ファイルサイズ設定
392 :             FileSize = AjcGetFileSize(pInp);
393 :             //  ゼロサイズならば、ファイルクローズ
394 :             if (FileSize == 0) {
395 :                 AjcVthPrintF(hVthLog, TEXT("OK (Zero size)¥n"));
396 :                 CloseHandle(fhInp); fhInp = INVALID_HANDLE_VALUE;
397 :                 CloseHandle(fhOut); fhOut = INVALID_HANDLE_VALUE;
398 :             }
399 :             //  ゼロサイズ以外ならば、ファールオープンした旨を返す
400 :             else {
401 :                 rc = TRUE;
402 :                 break;
403 :             }
404 :         }
405 :         else {
406 :             Unmatch++;
407 :             AjcVthPrintF(hVthLog, TEXT("¥x1B[31mNG (Open/Creation failure)¥x1B[0m¥n"));
408 :             if (fhInp != INVALID_HANDLE_VALUE) {CloseHandle(fhInp); fhInp = INVALID_HANDLE_VALUE;}
409 :             if (fhOut != INVALID_HANDLE_VALUE) {CloseHandle(fhOut); fhOut = INVALID_HANDLE_VALUE;}
```

```
410 :         }
411 :     }
412 :     return rc;
413 : }
414 : //-----------------------------------------------------------------------------------------------------------------//
415 : //  ファイル送信                                                                                                   //
416 : //-----------------------------------------------------------------------------------------------------------------//
417 : static  VO  FileSend()
418 : {
419 :     UL      bytes = 0;
420 :     UB      buf[1024];
421 :
422 :     //  バッファサイズを超える残データ有りならば、データ送信し、バイト数減算
423 :     if (FileSize > sizeof buf) {
424 :         ReadFile(fhInp, buf, sizeof buf, &bytes, NULL);
425 :         TxCount  += AjcScpSendPacket(hScp, buf, bytes);
426 :         FileSize -= bytes;
427 :     }
428 :     //  バッファサイズ以下の残データ有りならば、ファイル末尾データ送信
429 :     else if (FileSize != 0) {
430 :         ReadFile(fhInp, buf, (UI)FileSize, &bytes, NULL);
431 :         TxCount  += AjcScpSendPacket(hScp, buf, bytes);
432 :         FileSize -= bytes;
433 :     }
434 :     //  残データ無しならば、ファイル終端を示す空パケットを送信
435 :     else {
436 :         //  空パケット送信
437 :         TxCount += AjcScpSendPacket(hScp, buf, 0);
438 :         //  入力ファイルクローズ
439 :         if (fhInp != INVALID_HANDLE_VALUE) {CloseHandle(fhInp); fhInp = INVALID_HANDLE_VALUE;}
440 :     }
441 :     //  送信バイトカウント表示
442 :     AjcSepDlgItemUInt(hDlgMain, IDC_LBL_BYTES_TX, TxCount);
443 : }
444 : //-----------------------------------------------------------------------------------------------------------------//
445 : //  ボタン群許可／禁止                                                                                             //
446 : //-----------------------------------------------------------------------------------------------------------------//
447 : static  VO  EnableButtons(HWND hDlg, BOOL fbtnParam, BOOL fbtnOpen, BOOL fbtnStart, BOOL fbtnStop)
448 : {
449 :     AjcEnableDlgItem(hDlg, IDC_CMD_PARAM, fbtnParam);
450 :     AjcEnableDlgItem(hDlg, IDC_CMD_OPEN , fbtnOpen );
451 :     AjcEnableDlgItem(hDlg, IDC_CMD_START, fbtnStart);
452 :     AjcEnableDlgItem(hDlg, IDC_CMD_STOP , fbtnStop );
453 : }
454 : //-----------------------------------------------------------------------------------------------------------------//
455 : //  各ラベル表示                                                                                                   //
456 : //-----------------------------------------------------------------------------------------------------------------//
457 : static  VO      ShowLabel(HWND hDlg)
458 : {
459 :     UI  ps;
460 :
461 :     switch (ps = AjcScpGetSelectedPort(hScp)) {
462 :         default:                //  ●ＣＯＭポート
463 :             if (AjcScpIsOpened(hScp)) {                                         //  オープン状態？
464 :                 AjcSetDlgItemStr (hDlg, IDC_LBL_PORTSTATE, TEXT("接続 状態"));
465 :                 AjcSetDlgItemFStr(hDlg, IDC_CMD_OPEN, TEXT("COM %d 切断"), ps);
466 :             }
467 :             else {                                                              //  クローズ状態
468 :                 AjcSetDlgItemStr (hDlg, IDC_LBL_PORTSTATE, TEXT("切断 状態"));
469 :                 AjcSetDlgItemFStr(hDlg, IDC_CMD_OPEN, TEXT("COM %d 接続"), ps);
470 :                 EnableButtons(hDlg, TRUE, TRUE, FALSE, FALSE);
471 :             }
472 :             AjcSetDlgItemStr(hDlg, IDC_LBL_SERV, TEXT(""));
473 :             break;
474 :         case AJCSCP_PS_SLOT:    //  ●メールスロット
475 :             if (AjcScpIsOpened(hScp)) {                                         //  オープン状態？
476 :                 AjcSetDlgItemStr(hDlg, IDC_LBL_PORTSTATE, TEXT("接続 状態"));
477 :                 AjcSetDlgItemStr(hDlg, IDC_CMD_OPEN, TEXT("スロット 切断"));
478 :             }
479 :             else {                                                              //  クローズ状態
480 :                 AjcSetDlgItemStr (hDlg, IDC_LBL_PORTSTATE, TEXT("切断 状態"));
481 :                 AjcSetDlgItemFStr(hDlg, IDC_CMD_OPEN, TEXT("スロット 接続"));
482 :                 EnableButtons(hDlg, TRUE, TRUE, FALSE, FALSE);
483 :             }
484 :             AjcSetDlgItemStr(hDlg, IDC_LBL_SERV, TEXT(""));
485 :             break;
486 :         case AJCSCP_PS_CLIENT:  //  ●ソケット（クライアント）
487 :             if (AjcScpIsOpened(hScp)) {                                         //  オープン状態？
488 :                 AjcSetDlgItemStr(hDlg, IDC_LBL_PORTSTATE, TEXT("接続 状態"));
489 :                 AjcSetDlgItemStr(hDlg, IDC_CMD_OPEN, TEXT("サーバから切断"));
490 :             }
491 :             else {                                                              //  クローズ状態
492 :                 AjcSetDlgItemStr (hDlg, IDC_LBL_PORTSTATE, TEXT("切断 状態"));
493 :                 AjcSetDlgItemFStr(hDlg, IDC_CMD_OPEN, TEXT("サーバへ接続"));
494 :                 EnableButtons(hDlg, TRUE, TRUE, FALSE, FALSE);
495 :             }
496 :             AjcSetDlgItemStr(hDlg, IDC_LBL_SERV, TEXT(""));
497 :             break;
498 :         case AJCSCP_PS_SERVER:  //  ●ソケット（サーバ）
499 :             if (AjcScpIsServerStarted(hScp)) {                                  //  サーバ起動状態？
```

```
500 :                 if (AjcScpIsOpened(hScp)) {
501 :                     AjcSetDlgItemStr(hDlg, IDC_LBL_PORTSTATE, TEXT("接続 状態"));
502 :                 }
503 :                 else {
504 :                     AjcSetDlgItemStr(hDlg, IDC_LBL_PORTSTATE, TEXT("切断 状態"));
505 :                 }
506 :                 AjcSetDlgItemStr(hDlg, IDC_CMD_OPEN, TEXT("サーバ 停止"));
507 :             }
508 :             else {                                                          //  サーバ停止状態
509 :                 AjcSetDlgItemStr(hDlg, IDC_LBL_PORTSTATE, TEXT("切断 状態"));
510 :                 AjcSetDlgItemStr(hDlg, IDC_CMD_OPEN, TEXT("サーバ 起動"));
511 :                 EnableButtons(hDlg, TRUE, TRUE, FALSE, FALSE);
512 :             }
513 :             if (AjcScpIsServerStarted(hScp)) AjcSetDlgItemStr(hDlg, IDC_LBL_SERV, TEXT("サーバ動作中"));
514 :             else                             AjcSetDlgItemStr(hDlg, IDC_LBL_SERV, TEXT("サーバ停止中"));
515 :             break;
516 :     }
517 :     AjcSetDlgItemStr(hDlg, IDC_LBL_PORTNAME, AjcScpGetPortPathName(hScp));          //  ポート名表示
518 : }
519 :
```

13.11.5. SW_SerialComPort4（メールスロット通信）

このサンプルプログラムは、メールスロットによる（自コンピュータ内の）プロセス間通信を行います。
プログラムを起動すると、通信相手として自プログラムを追加起動します。
２つのウインドの「オープン」ボタンを押すと通信可能となります。(ボタンフェースが「クローズ」に変わります)
「送信」ボタンを押すと、現在日時のテキストを送信します。
片方のプログラムを終了すると、もう一方のプログラムも終了します。




```
 1 : //
 2 : //  SW_SerialComPort4.c
 3 : //
 4 : #define AJCSOCKSERV_H_
 5 : #define AJCSOCKCLIENT_H_
 6 : #include    <AjrCstXX.h>
 7 : #include    <tchar.h>
 8 : #include    "resource.h"
 9 :
10 : #define WM_SCPEVENT     (WM_USER + 100)
11 :
12 : //-------------------------------------------------------------------------------------------------------------//
13 : //  ワーク                                                                                                     //
14 : //-------------------------------------------------------------------------------------------------------------//
15 : static  HINSTANCE   hInst       = NULL;
16 : static  HWND        hDlgMain    = NULL;
17 : static  HWND        hWndVth     = NULL;
18 : static  HAJCSCP     hScp        = NULL;
19 : static  HANDLE      hMutex      = NULL;
20 : static  UI          msgBC       = 0;
21 : static  UT          MySlot [16] = {0};
22 : static  UT          RmtSlot[16] = {0};
23 :
24 : //-------------------------------------------------------------------------------------------------------------//
25 : //  内部サブ関数                                                                                               //
26 : //-------------------------------------------------------------------------------------------------------------//
27 : AJC_DLGPROC_DEF(Main);
28 :
29 : //=============================================================================================================//
30 : //                                                                                                             //
31 : //  W i n M a i n                                                                                              //
32 : //                                                                                                             //
33 : //=============================================================================================================//
34 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
35 : {
36 :     MSG         msg;
37 :
38 :     hInst = hInstance;
39 :
40 :     //----- メイン・ダイアログオープン ----------------//
41 :     hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_DLGMAIN), NULL, AJC_DLGPROC_NAME(Main));
42 :     ShowWindow(hDlgMain, SW_SHOW);
43 :
```

```
44 :     //----- メッセージループ --------------------------//
45 :     while (GetMessage(&msg, NULL, 0, 0)) {
46 :         do {
47 :             if (IsDialogMessage(hDlgMain, &msg)) break;
48 :             TranslateMessage(&msg);
49 :             DispatchMessage (&msg);
50 :         } while (0);
51 :     }
52 :
53 :     return (int)msg.wParam ;
54 : }
55 : //=============================================================================================================//
56 : //                                                                                                             //
57 : //  ダイアログ・プロシージャ                                                                                   //
58 : //                                                                                                             //
59 : //=============================================================================================================//
60 : //----- ダイアログ初期化 ----------------------------------------------------------------------------------//
61 : AJC_DLGPROC(Main, WM_INITDIALOG )
62 : {
63 :     RECT    r;
64 :
65 :     hDlgMain = hDlg;
66 :     hWndVth  = GetDlgItem(hDlg, IDC_VTH);
67 :
68 :     //----- ツールチップ設定 --------------------------//
69 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_INP_MYSLOT ), TEXT("自身のスロット名を入力します。¥n 空白を入力した場合は受信不可"));
70 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_CMD_MYSLOT ), TEXT("自身のスロット名を設定します。"));
71 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_TXT_MYSLOT ), TEXT("自身のスロット名です。"));
72 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_INP_RMTSLOT), TEXT("送信先のスロット名を入力します。¥n 空白を入力した場合は送信不可"));
73 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_CMD_RMTSLOT), TEXT("送信先のスロット名を設定します。"));
74 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_TXT_RMTSLOT), TEXT("送信先のスロット名です。"));
75 :     //----- テキスト入力リミット設定 ------------------//
76 :     AjcSetDlgItemEdtLimit(hDlg, IDC_INP_MYSLOT , 15);
77 :     AjcSetDlgItemEdtLimit(hDlg, IDC_INP_RMTSLOT, 15);
78 :     //----- ブロードキャストメッセージ登録 -------------//
79 :     msgBC = RegisterWindowMessage(TEXT("SW_SerialComPort4_BC"));
80 :     //----- ミューテックス生成--------------------------//
81 :     hMutex = CreateMutex(NULL, TRUE, TEXT("SW_SerialComPort4"));
82 :     //----- 最初のプロセス起動時・・・ -----------------//
83 :     if (GetLastError() != ERROR_ALREADY_EXISTS) {
84 :         UT      path[MAX_PATH];
85 :         //  ウインド位置設定
86 :         SetWindowPos(hDlg, NULL, 0, 0, 0, 0, SWP_NOSIZE);
87 :         //  スロット名設定
88 :         MAjcStrCpy(MySlot , 16, TEXT("SlotA"));
89 :         MAjcStrCpy(RmtSlot, 16, TEXT("SlotB"));
90 :         //  ２つ目のプロセス起動
91 :         AjcGetAppPath(path, MAX_PATH);
92 :         ShellExecute(NULL, NULL, TEXT("SW_SerialComPort4_32A.exe"), TEXT(""), path, SW_SHOWNORMAL);
93 :     }
94 :     //----- ２つ目のプロセス起動時・・・ ---------------//
95 :     else {
96 :         //  ウインド位置設定
97 :         GetWindowRect(hDlg, &r);
98 :         SetWindowPos(hDlg, NULL, 0, r.bottom - r.top + 10, 0, 0, SWP_NOSIZE);
99 :         //  スロット名読設定
100 :        MAjcStrCpy(MySlot , 16, TEXT("SlotB"));
101 :        MAjcStrCpy(RmtSlot, 16, TEXT("SlotA"));
102 :    }
103 :    //  スロット名表示
104 :    AjcSetDlgItemStr(hDlg, IDC_TXT_MYSLOT , MySlot );
105 :    AjcSetDlgItemStr(hDlg, IDC_TXT_RMTSLOT, RmtSlot);
106 :    //  ＳＣＰインスタンス生成（自メールスロット生成）
107 :    hScp = AjcScpCreateEx(NULL, TRUE, MySlot, NULL, RmtSlot);
108 :    //  ＳＣＰ初期化
109 :    AjcScpSetMode(hScp, hDlg, WM_SCPEVENT, AJCSCP_CM_TEXT);
110 :    AjcScpSetEvtMask(hScp,  AJCSCP_EV_PORTSTATE | AJCSCP_EV_RXTEXT);
111 :
112 :    return TRUE;
113 : }
114 : //----- ウインド破棄 ------------------------------------------------------------------------------------//
115 : AJC_DLGPROC(Main, WM_DESTROY    )
116 : {
117 :    AjcScpDelete(hScp);
118 :    CloseHandle(hMutex);
119 :    PostMessage(HWND_BROADCAST, msgBC, 0, 0);
120 :    PostQuitMessage(0);
121 :    return TRUE;
122 : }
123 : //----- ＳＣＰイベント通知 ------------------------------------------------------------------------------//
```

```
124 : AJC_DLGPROC(Main, WM_SCPEVENT   )
125 : {
126 :     UI      time = GetTickCount();
127 :     UI      len, param;
128 :     union {UBP pBin; UTP pTxt; VOP vp;} u;
129 :     UT      txt[128];
130 :
131 :     AjcScpGetEventData(hScp, lParam, &u.vp, &len, &param);              //  イベントデータ取得
132 :     if(wParam & AJCSCP_EV_PORTSTATE) {                                  //  ●ポート状態通知
133 :         switch (param) {
134 :             case AJCSCP_CLOSED:
135 :                 AjcSetDlgItemStr(hDlg, IDC_CMD_OPEN, TEXT("スロット オープン"));
136 :                 EnableWindow(GetDlgItem(hDlg, IDC_CMD_SEND), FALSE);
137 :                 break;
138 :             case AJCSCP_OPENED:
139 :                 AjcSetDlgItemStr(hDlg, IDC_CMD_OPEN, TEXT("スロット クローズ"));
140 :                 EnableWindow(GetDlgItem(hDlg, IDC_CMD_SEND), TRUE);
141 :                 break;
142 :             case AJCSCP_OPENFAIL:
143 :                 AjcSnPrintF(txt, AJCTSIZE(txt), TEXT("%s オープン／接続失敗"), u.pTxt);
144 :                 MessageBox(hDlgMain, txt, TEXT("Error"), MB_ICONERROR);
145 :                 break;
146 :             case AJCSCP_MYSLOTFAIL:
147 :                 MessageBox(hDlgMain, TEXT("自メールスロット生成失敗"), MySlot, MB_ICONERROR);
148 :                 break;
149 :         }
150 :     }
151 :     if (wParam & AJCSCP_EV_RXTEXT ) {                                   //  ●テキスト受信通知
152 :         AjcVthPrintF(hWndVth, TEXT("%s¥n"), u.pTxt);                    //      テキストデータ表示
153 :     }
154 :     AjcScpRelEventData(hScp, lParam);                                   //  イベントデータ開放
155 :
156 :     return TRUE;
157 : }
158 : //----- ウインドクローズ ------------------------------------------------------------------------------//
159 : AJC_DLGPROC(Main, IDCANCEL      )
160 : {
161 :     DestroyWindow(hDlg);
162 :     return TRUE;
163 : }
164 : //----- オープン／クローズ ボタン ---------------------------------------------------------------------//
165 : AJC_DLGPROC(Main, IDC_CMD_OPEN  )
166 : {
167 :     if (AjcScpIsOpened(hScp)) AjcScpClose(hScp);
168 :     else AjcScpOpenSlot(hScp, NULL, RmtSlot);
169 :
170 :     return TRUE;
171 : }
172 : //----- 送信ボタン ------------------------------------------------------------------------------------//
173 : AJC_DLGPROC(Main, IDC_CMD_SEND  )
174 : {
175 :     SYSTEMTIME  lt;
176 :     UT          txt[128];
177 :
178 :     GetLocalTime(&lt);
179 :     AjcSnPrintF(txt, AJCTSIZE(txt), TEXT("%d/%02d/%02d  %02d:%02d:%02d.%03d¥n"), lt.wYear, lt.wMonth, lt.wDay,
180 :                                                          lt.wHour, lt.wMinute, lt.wSecond, lt.wMilliseconds);
181 :     AjcScpSendText(hScp, txt, (UI)MAjcStrLen(txt));
182 :     return TRUE;
183 : }
184 : //----- ブロードキャストメッセージ (プログラム終了) ---------------------------------------------------//
185 : AJC_DLGPROC(Main, WN_END_MYPROG )
186 : {
187 :     DestroyWindow(hDlg);
188 :     return TRUE;
189 : }
190 :
191 : //-----------------------------------------------------------------------------------------------------//
192 : AJC_DLGMAP_DEF(Main)
193 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG  )
194 :     AJC_DLGMAP_MSG(Main, WM_DESTROY     )
195 :     AJC_DLGMAP_MSG(Main, WM_SCPEVENT    )
196 :     AJC_DLGMAP_CMD(Main, IDCANCEL       )
197 :     AJC_DLGMAP_CMD(Main, IDC_CMD_OPEN   )
198 :     AJC_DLGMAP_CMD(Main, IDC_CMD_SEND   )
199 :     AJC_DLGMAP_RWM(Main, WN_END_MYPROG  , TEXT("SW_SerialComPort4_BC"))
200 : AJC_DLGMAP_END
```

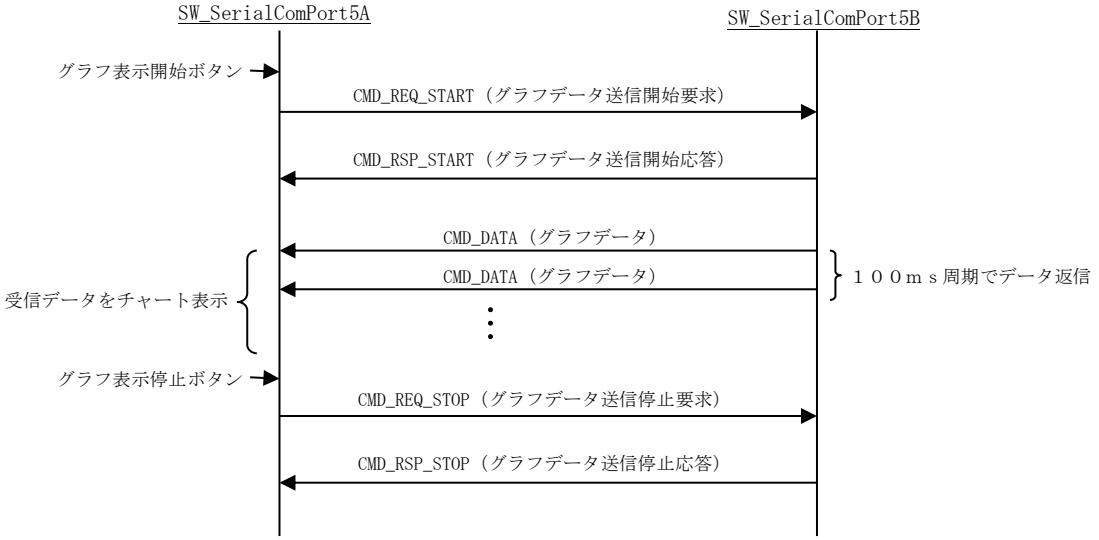
13.11.6. SW_SerialComPort5（テキストとバイナリパケットの多重通信）

このサンプルプログラムは、以下の２つのプログラムで構成します。

・SW_SerialComPort5A - SW_SerialComPort5B へコマンドを送信します。
・SW_SerialComPort5B - SW_SerialComPort5A からのコマンド要求に従い、データを送信します。

いずれかのプログラムを起動すると、通信相手として相手プログラムを自動的に起動します。

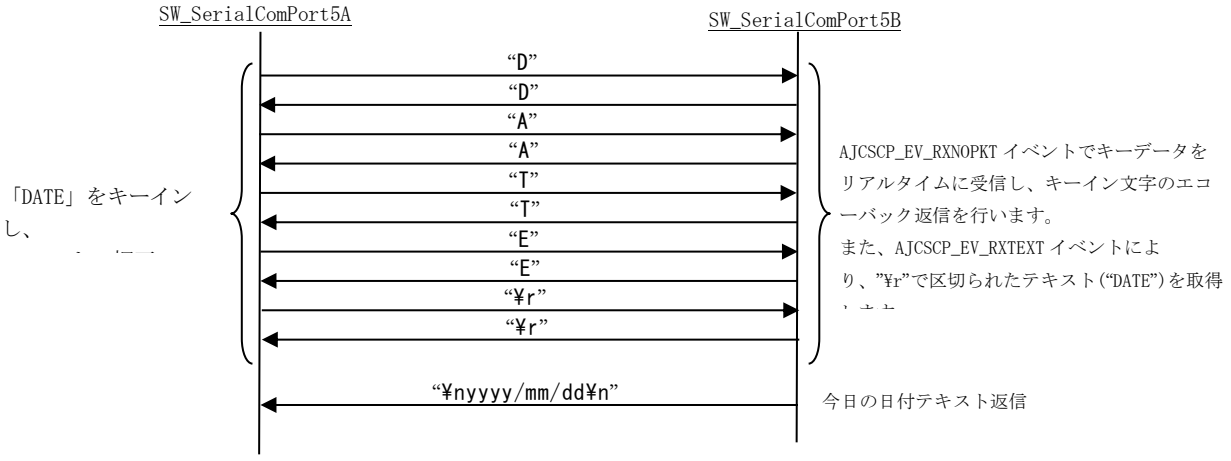
この２つのプログラムは、バイナリコマンド（バイナリパケット）で、以下の通信を行います。



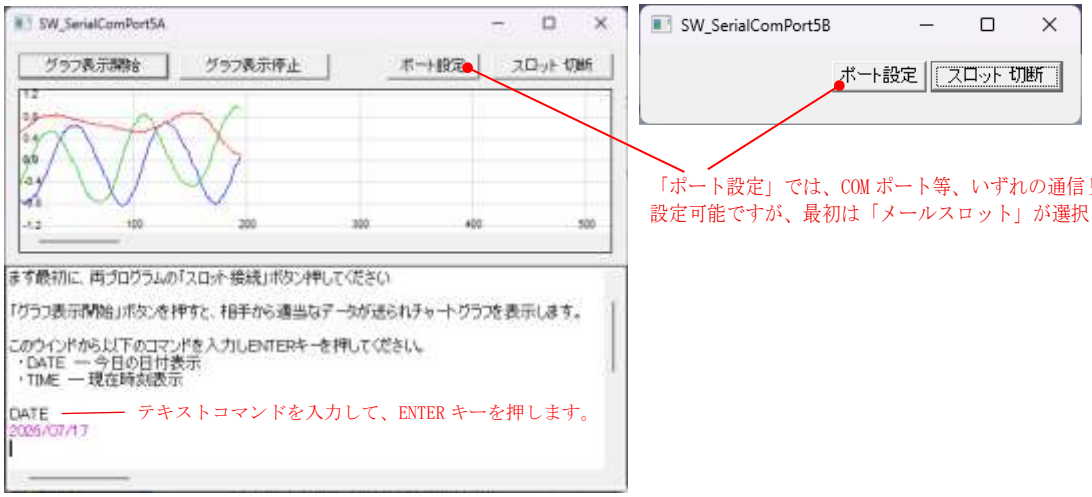
この２つのプログラムは、バイナリパケットでの通信と多重して、以下のテキスト・コマンド通信も行います。
テキスト・コマンドは、以下の２つです。

コマンド	内容
DATE	今日の日付テキスト（yyyy/mm/dd）を返信する
TIME	現在時刻のテキスト（hh:mm:ss）を返信する

例えば、DATE コマンドの場合、以下のように通信します。



プログラムイメージ



SW_SerialComPort5A

```
  1 : //
  2 : //  SW_SerialComPort5A.c
  3 : //
  4 :
  5 : #include    <AjrCstXX.h>
  6 : #include    <math.h>
  7 : #include    <tchar.h>
  8 : #include    "resource.h"
  9 :
 10 : #define WM_SCPEVENT         WM_APP
 11 : #define IDC_VTH             5001
 12 : #define VTH_HEIGHT          200
 13 :
 14 : //-------------------------------------------------------------------------------------------------------------//
 15 : //  通信パケットコマンド                                                                                       //
 16 : //-------------------------------------------------------------------------------------------------------------//
 17 : //  グラフデータ形式
 18 : #define CMD_REQ_START   0x01    //  グラフデータ送信開始要求
 19 : #define CMD_RSP_START   0x11    //          〃          応答
 20 : #define CMD_REQ_STOP    0x02    //  グラフデータ送信停止要求
 21 : #define CMD_RSP_STOP    0x12    //          〃          応答
 22 : #define CMD_DATA        0x40    //  グラフデータ
 23 :
 24 : typedef struct {
 25 :     UB          cmd;            //  コマンドコード
 26 :     UB          fil[3];         //  ー
 27 :     AJC3DVEC    vec;            //  グラフデータ
 28 : } CMDDATA, *PCMDDATA;
 29 :
 30 : //-------------------------------------------------------------------------------------------------------------//
 31 : //  ワーク                                                                                                     //
 32 : //-------------------------------------------------------------------------------------------------------------//
 33 : static  HINSTANCE       hInst    = NULL;             //  ＤＬＬインスタンスハンドル
 34 : static  HWND            hWndBack = NULL;             //  バックウインドハンドル（ダイアログと VT-100 の親ウインド）
 35 : static  HWND            hDlgMain = NULL;             //  ダイアログボックスハンドル
 36 : static  HWND            hWndVth  = NULL;             //  ＶＴＨウインドハンドル
 37 : static  HAJCSCP         hScp     = NULL;             //  ＳＣＰハンドル
 38 : static  HANDLE          hMut5A   = NULL;             //  自プログラム起動を示すミューテックス
 39 : static  HANDLE          hMut5B   = NULL;             //  相手プログラム起動チェック用ミューテックス
 40 : static  UT              MySlot[]  = TEXT("SlotA");   //  受信スロット名
 41 : static  UT              RmtSlot[] = TEXT("SlotB");   //  送信スロット名
 42 : //-------------------------------------------------------------------------------------------------------------//
 43 : //  内部サブ関数                                                                                               //
 44 : //-------------------------------------------------------------------------------------------------------------//
 45 : AJC_WNDPROC_DEF(Back);
 46 : AJC_DLGPROC_DEF(Main);
 47 : static  VO      ShowLabel(HWND hDlg);
 48 :
 49 : //=============================================================================================================//
 50 : //                                                                                                             //
 51 : //  Ｗ ｉ ｎ Ｍ ａ ｉ ｎ                                                                                       //
 52 : //                                                                                                             //
 53 : //=============================================================================================================//
 54 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
 55 : {
```

```
    MSG         msg;
    WNDCLASS    wc;

    hInst = hInstance;

    //  バックウインド生成
    wc.style        = 0;                        wc.hCursor          = LoadCursor(NULL, IDC_ARROW);
    wc.lpfnWndProc  = AJC_WNDPROC_NAME(Back);   wc.hbrBackground    = (HBRUSH)GetStockObject(WHITE_BRUSH);
    wc.cbClsExtra   = 0;                        wc.lpszMenuName     = NULL;
    wc.cbWndExtra   = 0;                        wc.lpszClassName    = TEXT("SW_SerialComPort5A");
    wc.hInstance    = hInst;
    wc.hIcon        = NULL;
    RegisterClass(&wc);

    hWndBack = CreateWindow(TEXT("SW_SerialComPort5A"), TEXT("SW_SerialComPort5A"),
                            WS_OVERLAPPEDWINDOW,
                            0, 0, 400, 300,
                            NULL, NULL, hInst, NULL);

    //  ウインド表示
    ShowWindow(hWndBack, iCmdShow);

    //  メッセージループ
    while (GetMessage(&msg, NULL, 0, 0)) {
        do {
            if (IsDialogMessage(hDlgMain, &msg)) break;
            TranslateMessage(&msg);
            DispatchMessage (&msg);
        } while (0);
    }

    return (int)msg.wParam ;
}
//=============================================================================================================================//
//                                                                                                                             //
//  バックウインド・プロシージャ                                                                                               //
//                                                                                                                             //
//=============================================================================================================================//
//----- WM_CREATE ---------------------------------------------------------------------------------------------------------------//
AJC_WNDPROC(Back, WM_CREATE         )
{
    RECT    r;

    //  自ミューテックス生成
    hMut5A = CreateMutex(NULL, TRUE, TEXT("SW_SerialComPort5A"));

    //  相手プログラム起動
    hMut5B = CreateMutex(NULL, TRUE, TEXT("SW_SerialComPort5B"));
    if (GetLastError() != ERROR_ALREADY_EXISTS) {
        UT  path[MAX_PATH];
        CloseHandle(hMut5B);
        AjcGetAppPath(path, MAX_PATH);
        ShellExecute(NULL, NULL, TEXT("SW_SerialComPort5B_64W.exe"), TEXT(""), path, SW_SHOWNORMAL);
    }
    //  メイン・ダイアログ生成
    hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_DLGMAIN), hwnd, AJC_DLGPROC_NAME(Main));
    GetWindowRect(hDlgMain, &r);
    AdjustWindowRect(&r, WS_OVERLAPPEDWINDOW, FALSE);
    SetWindowPos(hwnd, NULL, 0, 0, r.right - r.left, r.bottom - r.top + VTH_HEIGHT, SWP_NOMOVE);
    //  ダイアログ表示
    ShowWindow(hDlgMain, SW_SHOW);
    //  ＶＴＨウインド生成
    GetWindowRect(hDlgMain, &r);
    hWndVth = CreateWindowEx(WS_EX_ACCEPTFILES,                                     // extend style
                             TEXT("AjcCtrlVT100"),                                  // window class name
                             TEXT("P: VW=120, VH=25, FN=MS Gothic"),                // window caption
                             WS_CHILD | WS_VISIBLE | WS_DISABLED | AJCVTHS_LFCTRL,  // window style
                             0,                                                     // initial x position
                             r.bottom - r.top,                                      // initial y position
                             r.right - r.left,                                      // initial x size
                             VTH_HEIGHT,                                            // initial y size
                             hwnd,                                                  // parent window handle
                             (HMENU)IDC_VTH,                                        // window menu handle
                             hInst,                                                 // program instance handle
                             NULL);                                                 // creation parameters
    //  ツールチップ設定
    AjcTipTextAdd(hWndVth, TEXT("以下のコマンドを入力し ENTER キーを押してください。¥n")
                           TEXT("    ・DATE  -- 今日の日付表示¥n")
                           TEXT("    ・TIME  -- 現在時刻表示")
    );
```

```
136 :     //  初期メッセージ
137 :     AjcVthSetFontHandle(hWndVth, AjcGetStockFont(AJCFID_VAR14), 1);
138 :     AjcVthPrintF(hWndVth, TEXT("相手(SW_SerialComPort5B)と通信を行います。¥n¥n")
139 :                           TEXT("まず最初に、両プログラムの「スロット接続」ボタン押してください¥n¥n")
140 :                           TEXT("「グラフ表示開始」ボタンを押すと、相手から適当なデータが送られチャートグラフを表示します。¥n¥n")
141 :                           TEXT("このウインドから以下のコマンドを入力し ENTER キーを押してください。¥n")
142 :                           TEXT("   ・DATE  -- 今日の日付表示¥n")
143 :                           TEXT("   ・TIME  -- 現在時刻表示¥n"));
144 :
145 :     //  ＶＴＨウインドをフォーカス
146 :     SetFocus(hWndVth);
147 :
148 :     return 0;
149 : }
150 : //----- WM_DESTROY ----------------------------------------------------------------------------------------//
151 : AJC_WNDPROC(Back, WM_DESTROY        )
152 : {
153 :     if (hMut5A   != NULL) CloseHandle(hMut5A);
154 :     if (hScp     != NULL) AjcScpDelete(hScp);
155 :     if (hDlgMain != NULL) DestroyWindow(hDlgMain);
156 :     if (hWndVth  != NULL) DestroyWindow(hWndVth );
157 :     //  プログラム終了
158 :     PostQuitMessage(0);
159 :
160 :     return 0;
161 : }
162 : //----- ＶＴ１００ウインドからの通知 ----------------------------------------------------------------------//
163 : AJC_WNDPROC(Back, IDC_VTH           )
164 : {
165 :     if (HIWORD(wParam) == AJCVTHN_KEYIN) {
166 :         AjcScpSendText(hScp, (C_UTP)&lParam, 1);
167 :     }
168 :     return 0;
169 : }
170 : //--------------------------------------------------------------------------------------------------------//
171 : AJC_WNDMAP_DEF(Back)
172 :     AJC_WNDMAP_MSG(Back, WM_CREATE      )
173 :     AJC_WNDMAP_MSG(Back, WM_DESTROY     )
174 :     AJC_WNDMAP_CMD(Back, IDC_VTH        )
175 : AJC_WNDMAP_END
176 : //========================================================================================================//
177 : //                                                                                                        //
178 : //  ダイアログ・プロシージャ                                                                              //
179 : //                                                                                                        //
180 : //========================================================================================================//
181 : //----- ダイアログ初期化 ----------------------------------------------------------------------------------//
182 : AJC_DLGPROC(Main, WM_INITDIALOG     )
183 : {
184 :     hDlgMain = hDlg;
185 :     //  ＳＣＰセットアップ
186 :     hScp = AjcScpCreateEx(TEXT("Param"), TRUE, MySlot, NULL, RmtSlot);  //  ＳＣＰインスタンス生成
187 :
188 :     AjcScpSetMode(hScp, hDlg, WM_SCPEVENT, AJCSCP_CM_TEXT);             //  モード設定
189 :     AjcScpSetRxTextCode(hScp, AJCSCP_TXT_SJIS);                         //  受信テキストコード＝SJIS
190 :     AjcScpSetTxTextCode(hScp, AJCSCP_TXT_SJIS);                         //  送信テキストコード＝SJIS
191 :     AjcScpSetEvtMask(hScp,  AJCSCP_EV_PORTSTATE |                       //      ポート状態通知
192 :                             AJCSCP_EV_RXPKT     |                       //      パケットデータ受信通知
193 :                             AJCSCP_EV_RXNOPKT);                         //      パケット外データ受信通知
194 :     AjcScpSetSelectedPort(hScp, AJCSCP_PS_SLOT);                        //  メールスロット選択
195 :     ShowLabel(hDlg);
196 :     return TRUE;
197 : }
198 : //----- ウインド破棄 --------------------------------------------------------------------------------------//
199 : AJC_DLGPROC(Main, WM_DESTROY        )
200 : {
201 :     hDlgMain = NULL;
202 :     return TRUE;
203 : }
204 : //----- ＳＣＭイベント通知 --------------------------------------------------------------------------------//
205 : AJC_DLGPROC(Main, WM_SCPEVENT       )
206 : {
207 :     UI      time = GetTickCount();
208 :     UI      len, param;
209 :     UT      txt[256];
210 :     union {UBP pCmd; UTP pTxt; VOP vp; PCMDDATA pDat;} u;
211 :
212 :     AjcScpGetEventData(hScp, lParam, &u.vp, &len, &param);              //  イベントデータ取得
213 :     switch (wParam) {
214 :         case AJCSCP_EV_PORTSTATE:                                       //  ●ポート状態通知
215 :             AjcTipTextHide();
```

```
216 :             switch (param) {
217 :                 case AJCSCP_CLOSED:                                     //      ・クローズ状態
218 :                     AjcEnableDlgItem(hDlg, IDC_CMD_START, FALSE);
219 :                     AjcEnableDlgItem(hDlg, IDC_CMD_STOP , FALSE);
220 :                     EnableWindow(hWndVth, FALSE);
221 :                     SetFocus(GetDlgItem(hDlg, IDC_CMD_OPEN));
222 :                     break;
223 :                 case AJCSCP_OPENED:                                     //      ・オープン状態
224 :                     AjcEnableDlgItem(hDlg, IDC_CMD_START, TRUE);
225 :                     AjcEnableDlgItem(hDlg, IDC_CMD_STOP , TRUE);
226 :                     EnableWindow(hWndVth, TRUE);
227 :                     SetFocus(hWndVth);
228 :                     break;
229 :                 case AJCSCP_SRVSTART:                                   //      ・サーバ開始
230 :                     AjcTipTextShowCenter(hDlg, TEXT("¥n クライアントを接続してください ¥n "), 10000, NULL);
231 :                     break;
232 :                 case AJCSCP_OPENFAIL:                                   //      ・オープン失敗
233 :                     AjcSnPrintF(txt, AJCTSIZE(txt), TEXT("%s オープン／接続失敗"), u.pTxt);
234 :                     MessageBox(hDlg, txt, TEXT("SW_SerialComPort5A"), MB_ICONERROR);
235 :                     break;
236 :                 case AJCSCP_SRVSRTFAIL:                                 //      ・サーバ開始失敗
237 :                     MessageBox(hDlg, TEXT("サーバ開始失敗"), TEXT("SW_SerialComPort5A"), MB_ICONERROR);
238 :                     break;
239 :                 case AJCSCP_MYSLOTFAIL:                                 //      ・自スロット生成失敗
240 :                     AjcSnPrintF(txt, AJCTSIZE(txt), TEXT("自メールスロット (  %s ) の生成失敗"), u.pTxt),
241 :                     MessageBox(hDlg, txt, TEXT("SW_SerialComPort5A"), MB_ICONERROR);
242 :                     break;
243 :             }
244 :             ShowLabel(hDlg);
245 :             break;
246 :         case AJCSCP_EV_RXPKT:                                           //  ●パケットデータ受信通知
247 :             switch (*u.pCmd) {
248 :                 case CMD_RSP_START:         //  グラフデータ送信開始応答
249 :                     break;
250 :                 case CMD_RSP_STOP:          //  グラフデータ送信停止応答
251 :                     break;
252 :                 case CMD_DATA:              //  グラフデータ
253 :                     AjcTchPutRealData(GetDlgItem(hDlg, IDC_TCH), &u.pDat->vec.x);
254 :                     break;
255 :             }
256 :             break;
257 :         case AJCSCP_EV_RXNOPKT:                                         //  ●パケット外テキスト受信通知
258 :             AjcVthPutText(hWndVth, u.pTxt, len);
259 :             break;
260 :     }
261 :     AjcScpRelEventData(hScp, lParam);                                   //  イベントデータ開放
262 :
263 :     return TRUE;
264 : }
265 : //----- グラフ表示開始ボタン ----------------------------------------------------------------------------------//
266 : AJC_DLGPROC(Main, IDC_CMD_START     )
267 : {
268 :     UB  cmd;
269 :
270 :     if (HIWORD(wParam) == BN_CLICKED) {
271 :         cmd = CMD_REQ_START;
272 :         AjcScpSendPacket(hScp, (C_VOP)&cmd, 1);
273 :     }
274 :     SetFocus(hWndVth);
275 :     return 0;
276 : }
277 : //----- グラフ表示停止ボタン ----------------------------------------------------------------------------------//
278 : AJC_DLGPROC(Main, IDC_CMD_STOP      )
279 : {
280 :     UB  cmd;
281 :
282 :     if (HIWORD(wParam) == BN_CLICKED) {
283 :         cmd = CMD_REQ_STOP;
284 :         AjcScpSendPacket(hScp, (C_VOP)&cmd, 1);
285 :     }
286 :     SetFocus(hWndVth);
287 :     return 0;
288 : }
289 : //----- ポート設定ボタン --------------------------------------------------------------------------------------//
290 : AJC_DLGPROC(Main, IDC_CMD_SETPORT   )
291 : {
292 :     if (HIWORD(wParam) == BN_CLICKED) {
293 :         AjcScpDlgParamEasy(hScp, hDlg);
294 :         ShowLabel(hDlg);
295 :     }
```

```
296 :     SetFocus(hWndVth);
297 :     return 0;
298 : }
299 : //----- ＯＰＥＮボタン ----------------------------------------------------------------------------------------//
300 : AJC_DLGPROC(Main, IDC_CMD_OPEN      )
301 : {
302 :     if (HIWORD(wParam) == BN_CLICKED) {
303 :         if (AjcScpIsOpened(hScp) || AjcScpIsServerStarted(hScp)) {
304 :             AjcScpClose(hScp);
305 :         }
306 :         else {
307 :             AjcScpOpenDefault(hScp);
308 :         }
309 :     }
310 :     SetFocus(hWndVth);
311 :     return 0;
312 : }
313 : //-----------------------------------------------------------------------------------------------------------//
314 : AJC_DLGMAP_DEF(Main)
315 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG  )
316 :     AJC_DLGMAP_MSG(Main, WM_DESTROY     )
317 :     AJC_DLGMAP_MSG(Main, WM_SCPEVENT    )
318 :     AJC_DLGMAP_CMD(Main, IDC_CMD_START  )
319 :     AJC_DLGMAP_CMD(Main, IDC_CMD_STOP   )
320 :     AJC_DLGMAP_CMD(Main, IDC_CMD_SETPORT)
321 :     AJC_DLGMAP_CMD(Main, IDC_CMD_OPEN   )
322 : AJC_DLGMAP_END
323 : //-----------------------------------------------------------------------------------------------------------//
324 : //  各ラベル表示                                                                                            //
325 : //-----------------------------------------------------------------------------------------------------------//
326 : static  VO      ShowLabel(HWND hDlg)
327 : {
328 :     UI  ps;
329 :
330 :     switch (ps = AjcScpGetSelectedPort(hScp)) {
331 :         default:                // ●ＣＯＭポート
332 :             if (AjcScpIsOpened(hScp)) AjcSetDlgItemFStr(hDlg, IDC_CMD_OPEN, TEXT("COM %d 切断"), ps);
333 :             else                      AjcSetDlgItemFStr(hDlg, IDC_CMD_OPEN, TEXT("COM %d 接続"), ps);
334 :             break;
335 :         case AJCSCP_PS_SLOT:    // ●メールスロット
336 :             if (AjcScpIsOpened(hScp)) AjcSetDlgItemStr(hDlg, IDC_CMD_OPEN, TEXT("スロット 切断"));
337 :             else                      AjcSetDlgItemStr(hDlg, IDC_CMD_OPEN, TEXT("スロット 接続"));
338 :             break;
339 :         case AJCSCP_PS_CLIENT:  // ●ソケット（クライアント）
340 :             if (AjcScpIsOpened(hScp)) AjcSetDlgItemStr(hDlg, IDC_CMD_OPEN, TEXT("サーバから切断"));
341 :             else                      AjcSetDlgItemStr(hDlg, IDC_CMD_OPEN, TEXT("サーバへ接続"));
342 :             break;
343 :         case AJCSCP_PS_SERVER:  // ●ソケット（サーバ）
344 :             if (AjcScpIsServerStarted(hScp)) AjcSetDlgItemStr(hDlg, IDC_CMD_OPEN, TEXT("サーバ 停止"));
345 :             else                             AjcSetDlgItemStr(hDlg, IDC_CMD_OPEN, TEXT("サーバ 起動"));
346 :             break;
347 :     }
348 : }
```

SW_SerialComPort5B

```
//
//  SW_SerialComPort5B.c
//

#include    <AjrCstXX.h>
#include    <math.h>
#include    <tchar.h>
#include    "resource.h"

#define WM_SCPEVENT         WM_APP

//-------------------------------------------------------------------------------------------------------------------//
//  通信パケットコマンド                                                                                             //
//-------------------------------------------------------------------------------------------------------------------//
//  グラフデータ形式
#define CMD_REQ_START   0x01    //  グラフデータ送信開始要求
#define CMD_RSP_START   0x11    //          〃          応答
#define CMD_REQ_STOP    0x02    //  グラフデータ送信停止要求
#define CMD_RSP_STOP    0x12    //          〃          応答
#define CMD_DATA        0x40    //  グラフデータ

typedef struct {
    UB          cmd;            //  コマンドコード
    UB          fil[3];         //  ー
    AJC3DVEC    vec;            //  グラフデータ
} CMDDATA, *PCMDDATA;

//-------------------------------------------------------------------------------------------------------------------//
//  ワーク                                                                                                           //
//-------------------------------------------------------------------------------------------------------------------//
static  HINSTANCE       hInst    = NULL;            //  ＤＬＬインスタンスハンドル
static  HWND            hDlgMain = NULL;            //  ダイアログボックスハンドル
static  HAJCSCP         hScp     = NULL;            //  ＳＣＰハンドル
static  HAJCSPD         hSpd     = NULL;            //  テストデータ生成インスタンス
static  HANDLE          hMut5B   = NULL;            //  自プログラム起動を示すミューテックス
static  HANDLE          hMut5A   = NULL;            //  相手プログラム起動チェック用ミューテックス
//-------------------------------------------------------------------------------------------------------------------//
//  内部サブ関数                                                                                                     //
//-------------------------------------------------------------------------------------------------------------------//
AJC_DLGPROC_DEF(Main);
static  VO      ShowLabel(HWND hDlg);

//===================================================================================================================//
//                                                                                                                   //
//  ＷｉｎＭａｉｎ                                                                                                  //
//                                                                                                                   //
//===================================================================================================================//
int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
{
    MSG     msg;

    hInst = hInstance;

    //----- メイン・ダイアログオープン -----------------//
    hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_DLGMAIN), NULL, AJC_DLGPROC_NAME(Main));
    SetWindowPos(hDlgMain, NULL, 560, 0, 0, 0, SWP_NOSIZE);
    //----- ダイアログ表示 -----------------------------//
    ShowWindow(hDlgMain, SW_SHOW);

    //----- メッセージループ ---------------------------//
    while (GetMessage(&msg, NULL, 0, 0)) {
        do {
            if (IsDialogMessage(hDlgMain, &msg)) break;
            TranslateMessage(&msg);
            DispatchMessage (&msg);
        } while (0);
    }

    return (int)msg.wParam ;
}
//===================================================================================================================//
//                                                                                                                   //
//  ダイアログ・プロシージャ                                                                                         //
//                                                                                                                   //
//===================================================================================================================//
//----- ダイアログ初期化 -----------------------------------------------------------------------------------------//
AJC_DLGPROC(Main, WM_INITDIALOG     )
```

```
 78 : {
 79 :     hDlgMain = hDlg;
 80 :
 81 :     //  自ミューテックス生成
 82 :     hMut5B = CreateMutex(NULL, TRUE, TEXT("SW_SerialComPort5B"));
 83 :
 84 :     //  相手プログラム起動
 85 :     hMut5A = CreateMutex(NULL, TRUE, TEXT("SW_SerialComPort5A"));
 86 :     if (GetLastError() != ERROR_ALREADY_EXISTS) {
 87 :         UT  path[MAX_PATH];
 88 :         CloseHandle(hMut5A);
 89 :         AjcGetAppPath(path, MAX_PATH);
 90 :         ShellExecute(NULL, NULL, TEXT("SW_SerialComPort5A_64W.exe"), TEXT(""), path, SW_SHOWNORMAL);
 91 :     }
 92 :     //  終了監視タイマ起動
 93 :     SetTimer(hDlg, 2, 1000, NULL);
 94 :     //  プロットデータ演算初期化
 95 :     hSpd = AjcSpdCreate(0);
 96 :     //  ＳＣＰセットアップ
 97 :     hScp = AjcScpCreate();                                          //  ＳＣＰインスタンス生成
 98 :     AjcScpSetMode(hScp, hDlg, WM_SCPEVENT, AJCSCP_CM_TEXT);         //  モード設定
 99 :     AjcScpSetRxTextCode(hScp, AJCSCP_TXT_SJIS);                     //  受信テキストコード＝SJIS
100 :     AjcScpSetTxTextCode(hScp, AJCSCP_TXT_SJIS);                     //  送信テキストコード＝SJIS
101 :     AjcScpSetEvtMask(hScp,  AJCSCP_EV_PORTSTATE |                   //      ポート状態通知
102 :                             AJCSCP_EV_RXPKT     |                   //      パケットデータ受信通知
103 :                             AJCSCP_EV_RXTEXT    |                   //      テキスト受信通知
104 :                             AJCSCP_EV_RXNOPKT);                   //      パケット外テキスト受信通知
105 :     ShowLabel(hDlg);
106 :     return TRUE;
107 : }
108 : //----- ウインド破棄 ----------------------------------------------------------------------------------------//
109 : AJC_DLGPROC(Main, WM_DESTROY        )
110 : {
111 :     if (hMut5B   != NULL) CloseHandle(hMut5B);
112 :     if (hSpd != NULL) AjcSpdDelete(hSpd);
113 :     if (hScp != NULL) AjcScpDelete(hScp);
114 :     //  プログラム終了
115 :     PostQuitMessage(0);
116 :     return TRUE;
117 : }
118 : //----- タイマ ----------------------------------------------------------------------------------------------//
119 : AJC_DLGPROC(Main, WM_TIMER          )
120 : {
121 :     switch (wParam) {
122 :         case 1:             //  グラフデータ送信タイマ
123 :         {   CMDDATA     txd = {CMD_DATA};
124 :             AjcSpdCalcV(hSpd, &txd.vec);
125 :             AjcScpSendPacket(hScp, (C_VOP)&txd, sizeof txd);
126 :             break;
127 :         }
128 :         case 2:             //  終了監視タイマ
129 :             hMut5A = CreateMutex(NULL, TRUE, TEXT("SW_SerialComPort5A"));
130 :             if (GetLastError() != ERROR_ALREADY_EXISTS) {
131 :                 DestroyWindow(hDlg);
132 :             }
133 :             CloseHandle(hMut5A);
134 :     }
135 :     return TRUE;
136 : }
137 : //----- ＳＣＭイベント通知 ----------------------------------------------------------------------------------//
138 : AJC_DLGPROC(Main, WM_SCPEVENT       )
139 : {
140 :     UI      time = GetTickCount();
141 :     UI      len, param;
142 :     union {UBP pCmd; UTP pTxt; VOP vp;} u;
143 :     UB      cmd;
144 :
145 :     AjcScpGetEventData(hScp, lParam, &u.vp, &len, &param);          //  イベントデータ取得
146 :     switch (wParam) {
147 :         UT  txt[256];
148 :         case AJCSCP_EV_PORTSTATE:                                   //  ●ポート状態通知
149 :             AjcTipTextHide();
150 :             switch (param) {
151 :                 case AJCSCP_SRVSTART:   //  サーバ開始
152 :                     AjcTipTextShowCenter(hDlg, TEXT("¥n エコー クライアントを接続してください ¥n "), 10000, NULL);
153 :                     break;
154 :                 case AJCSCP_OPENFAIL:                               //      ・オープン失敗
155 :                     AjcSnPrintF(txt, AJCTSIZE(txt), TEXT("%s オープン／接続失敗"), u.pTxt);
156 :                     MessageBox(hDlg, txt, TEXT("SW_SerialComPort5B"), MB_ICONERROR);
157 :                     break;
```

```
158 :                 case AJCSCP_SRVSRTFAIL:                                       //       ・サーバ開始失敗
159 :                     MessageBox(hDlg, TEXT("サーバ開始失敗"), TEXT("SW_SerialComPort5B"), MB_ICONERROR);
160 :                     break;
161 :                 case AJCSCP_MYSLOTFAIL:                                       //       ・自スロット生成失敗
162 :                     AjcSnPrintF(txt, AJCTSIZE(txt), TEXT("自メールスロット (  %s ) の生成失敗"), u.pTxt),
163 :                     MessageBox(hDlg, txt, TEXT("SW_SerialComPort5B"), MB_ICONERROR);
164 :                     break;
165 :             }
166 :             ShowLabel(hDlg);
167 :             break;
168 :         case AJCSCP_EV_RXPKT:                                         //  ●パケットデータ受信通知
169 :             switch (*u.pCmd) {
170 :                 case CMD_REQ_START:         //  グラフデータ送信開始要求
171 :                     cmd = CMD_RSP_START;
172 :                     AjcScpSendPacket(hScp, (C_VOP)&cmd, 1);
173 :                     SetTimer(hDlg, 1, 100, NULL);
174 :                     break;
175 :                 case CMD_REQ_STOP:          //  グラフデータ送信停止要求
176 :                     cmd = CMD_RSP_STOP;
177 :                     AjcScpSendPacket(hScp, (C_VOP)&cmd, 1);
178 :                     KillTimer(hDlg, 1);
179 :                     break;
180 :             }
181 :             break;
182 :         case AJCSCP_EV_RXTEXT:                                        //  ●テキスト受信通知
183 :         {   SYSTEMTIME  lt;
184 :             UT          txt[64];
185 :             GetLocalTime(&lt);
186 :             AjcScpSendText(hScp, TEXT("¥n¥x1B[35m"), -1);
187 :             if      (_tcsicmp(u.pTxt, TEXT("DATE")) == 0) {
188 :                 AjcSnPrintF(txt, AJCTSIZE(txt), TEXT("%d/%02d/%02d¥n"), lt.wYear, lt.wMonth, lt.wDay);
189 :                 AjcScpSendText(hScp, txt, -1);
190 :             }
191 :             else if (_tcsicmp(u.pTxt, TEXT("TIME")) == 0) {
192 :                 AjcSnPrintF(txt, AJCTSIZE(txt), TEXT("%02d:%02d:%02d¥n"), lt.wHour, lt.wMinute, lt.wSecond);
193 :                 AjcScpSendText(hScp, txt, -1);
194 :             }
195 :             else {
196 :                 AjcScpSendTextF(hScp, TEXT("*** ¥"%s¥" は、不正なコマンドです ***¥n"), u.pTxt);
197 :             }
198 :             AjcScpSendText(hScp, TEXT("¥x1B[0m"), -1);
199 :             break;
200 :         }
201 :         case AJCSCP_EV_RXNOPKT:                                       //  ●パケット外テキスト受信通知
202 :             AjcScpSendBinData(hScp, u.vp, len);                       //       コマンドエコー返信
203 :             break;
204 :     }
205 :     AjcScpRelEventData(hScp, lParam);                                 //  イベントデータ開放
206 :
207 :     return TRUE;
208 : }
209 : //----- ポート設定ボタン ------------------------------------------------------------------------------------//
210 : AJC_DLGPROC(Main, IDC_CMD_SETPORT   )
211 : {
212 :     if (HIWORD(wParam) == BN_CLICKED) {
213 :         AjcScpDlgParamEasy(hScp, hDlg);
214 :         ShowLabel(hDlg);
215 :     }
216 :     return TRUE;
217 : }
218 : //----- ＯＰＥＮボタン --------------------------------------------------------------------------------------//
219 : AJC_DLGPROC(Main, IDC_CMD_OPEN      )
220 : {
221 :     if (HIWORD(wParam) == BN_CLICKED) {
222 :         if (AjcScpIsOpened(hScp) || AjcScpIsServerStarted(hScp)) {
223 :             AjcScpClose(hScp);
224 :         }
225 :         else {
226 :             AjcScpOpenDefault(hScp);
227 :         }
228 :     }
229 :     return TRUE;
230 : }
231 : //----- 「Cancel」ボタン ------------------------------------------------------------------------------------//
232 : AJC_DLGPROC(Main, IDCANCEL          )
233 : {
234 :     DestroyWindow(hDlg);
235 :     return TRUE;
236 : }
237 : //-----------------------------------------------------------------------------------------------------------//
```

```
238 : AJC_DLGMAP_DEF(Main)
239 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG     )
240 :     AJC_DLGMAP_MSG(Main, WM_DESTROY        )
241 :     AJC_DLGMAP_MSG(Main, WM_TIMER          )
242 :     AJC_DLGMAP_MSG(Main, WM_SCPEVENT       )
243 :
244 :     AJC_DLGMAP_CMD(Main, IDC_CMD_SETPORT   )
245 :     AJC_DLGMAP_CMD(Main, IDC_CMD_OPEN      )
246 :     AJC_DLGMAP_CMD(Main, IDCANCEL          )
247 : AJC_DLGMAP_END
248 : //-------------------------------------------------------------------------------------------------------------------//
249 : //  各ラベル表示                                                                                                     //
250 : //-------------------------------------------------------------------------------------------------------------------//
251 : static  VO      ShowLabel(HWND hDlg)
252 : {
253 :     UI  ps;
254 :
255 :     switch (ps = AjcScpGetSelectedPort(hScp)) {
256 :         default:                //  ●ＣＯＭポート
257 :             if (AjcScpIsOpened(hScp)) AjcSetDlgItemFStr(hDlg, IDC_CMD_OPEN, TEXT("COM %d 切断"), ps);
258 :             else                      AjcSetDlgItemFStr(hDlg, IDC_CMD_OPEN, TEXT("COM %d 接続"), ps);
259 :             break;
260 :         case AJCSCP_PS_SLOT:    //  ●メールスロット
261 :             if (AjcScpIsOpened(hScp)) AjcSetDlgItemStr(hDlg, IDC_CMD_OPEN, TEXT("スロット 切断"));
262 :             else                      AjcSetDlgItemStr(hDlg, IDC_CMD_OPEN, TEXT("スロット 接続"));
263 :             break;
264 :         case AJCSCP_PS_CLIENT:  //  ●ソケット（クライアント）
265 :             if (AjcScpIsOpened(hScp)) AjcSetDlgItemStr(hDlg, IDC_CMD_OPEN, TEXT("サーバから切断"));
266 :             else                      AjcSetDlgItemStr(hDlg, IDC_CMD_OPEN, TEXT("サーバへ接続"));
267 :             break;
268 :         case AJCSCP_PS_SERVER:  //  ●ソケット（サーバ）
269 :             if (AjcScpIsServerStarted(hScp)) AjcSetDlgItemStr(hDlg, IDC_CMD_OPEN, TEXT("サーバ 停止"));
270 :             else                             AjcSetDlgItemStr(hDlg, IDC_CMD_OPEN, TEXT("サーバ 起動"));
271 :             break;
272 :     }
273 : }
```

14. ソケット(TCP/IP)サーバ機能

ソケット(TCP/IP)のサーバ側通信制御モジュールです。
複数のクライアントをソケットで接続し通信することができます。

ソケット(TCP/IP)サーバでは、定義名やＡＰ名がシリアル通信やソケット(TCP/IP)クライアント機能と似通っています。
「#include 〈AjrCstXX.h〉」の前に、以下のように「#define」を定義すると、他のの定義名やＡＰ名を抑止できます。
（誤って、シリアル通信やソケット(TCP/IP)クライアント機能の定義名やＡＰ名を使用するとコンパイルエラーとなります）

```
#define AJCSERIALCOMPORT_H_          // シリアル通信を無効化
#define AJCSOCKCLIENT_H_             // ソケット(TCP/IP)クライアントを無効化
#include 〈AjrCstXX.h〉
```

14.1. イベントの通知方法

イベントの通知方法は、以下の２つから選択できます。

・ユーザのウインドへ、ウインドメッセージとして通知する
・ユーザが、その場でイベントの発生を待ち受ける

ユーザのウインドへ、ウインドメッセージとして通知する場合は、AjcSsvSetMode()の hWndNtc 引数でウインドハンドルを、WndMsgNtc 引数でウインド・メッセージコードを指定します。

ウインドプロシージャの wParam にはイベントコードが、lParam にはイベント情報が設定されます。
イベントデータを取得するには、lParam を引数として、AjcSsvGetEventData()を実行します。

ウインドメッセージとして通知する場合のプログラムコードスタイルは、およそ以下のようになります。

```
#define  WM_SSVEVENT  (WM_APP + 100)
HAJCSSV  hSsv;
//----- ダイアログ初期化 ----------------------------------------------------------------------------//
AJC_DLGPROC(Main, WM_INITDIALOG )
{
    // インスタンス生成
    hSsv = AjcSsvCreate();
    //  オプション設定
    AjcSsvSetOpt(hSsv, AJCSSV_SOP_SUP_SLEEP);
    // 送受信テキスト文字コード設定
    AjcSsvSetRxTextCode(hSsv, AJCSSV_TXT_AUTO);
    AjcSsvSetTxTextCode(hSsv, AJCSSV_TXT_AUTO);
    // バイナリチャンクモード設定 (AJCSSV_CM_TEXT:テキストチャンク, AJCSSV_CM_BOTH:両方)
    AjcSsvSetChunkMode(hSsv, AJCSSV_CM_BIN);
    // 使用するイベント指定 (デフォルトでは、全イベントが有効であるため、特に指定しなくてもよい)
    AjcSsvSetEvtMask(hSsv, AJCSSV_EV_RXCHUNK | AJCSSV_EV_STOP | AJCSSV_EV_ERR・・・・・);
    // サーバ開始       〈ポート番号〉   〈アドレスファミリ〉〈最大クライアント数〉〈ウインドハンドル〉〈通知メッセージ〉
    AjcSsvStart(hSsv, TEXT("14238"), AF_INET,        n,              hwnd          , WM_SSVEVENT);
    ・・・・
    return TRUE;
}
//----- サーバイベント通知 --------------------------------------------------------------------------//
AJC_DLGPROC(Main, WM_SSVEVENT )
{
    union {UBP  pByte; UWP pWord; UTP pTxt; VOP pVoid;} u;
    UI           len, param;
    HAJCSSVCLI   hClient = NULL;        // クライアントハンドル
    UT           szIpAddr[256] = {0};   // クライアントのＩＰアドレス
    // イベントデータ取得
    AjcSsvGetEventData(hSsv, lParam, &u.pVoid, &len, &param);
    // クライアント情報を取得し、クライアントのＩＰアドレスを設定
    //             (クライアント情報を取得できないイベントもあるので注意が必要 (後述のイベント一覧参照))
    if (AjcSsvGetClient (hSsv, lParam, &hClient)) {
        AjcSsvGetIpAddrStr(hClient, szIpAddr, AJCTSIZE(szIpAddr));
    }
```

つづく

つづき

```
    // イベント処理
    switch (wParam) {
        //●チャンクデータ受信通知
        case AJCSSV_EV_RXCHUNK:
            // 受信データをループバック送信（任意）
            AjcSsvSendText(hClient, u.pTxt, lTxt);
            break;

        // ●サーバ停止通知
        case AJCSSV_EV_STOP:
            //  ウインドクローズ（任意）
            DestroyWindow(hDlg);
            break;

        //●送信エラー通知
        case AJCSSV_EV_TXERR:
            // 送信エラーが発生したら、クライアント切断（任意）
            AjcSsvDisconnect(hClient);
            break;

        ・・・その他のイベント処理・・・

    }
    // イベントデータ開放
    AjcSsvRelEventData(hSsv, lParam);
    return TRUE;
}
```

ユーザが、その場でイベントの発生を待ち受ける場合は、AjcSsvSetMode()の hWndNtc 引数に NULL を、WndMsgNtc 引数に 0 を指定し、AjcSsvWaitEvent()によりイベントの発生を待ちます。

AjcSsvWaitEvent()を実行すると、イベントが発生している場合は、wParam にはイベントコードが、lParam にはイベント情報が設定され、TRUE を返します。イベントが発生していない場合は FALSE を返します。
イベントデータを取得するには、lParam を引数として、AjcSsvGetEventData()を実行します。

イベントの発生を待ち受ける場合のプログラムコードスタイルは、およそ以下のようになります。

```
int  AjcMain(int argc, UTP argv[])
{
    BOOL        fEnd = FALSE;
    HAJCSSV     hSsv;
    WPARAM      wParam;
    LPARAM      lParam;
    union {UBP  pByte; UWP pWord; UTP pTxt; VOP pVoid;} u;
    UI          len;
    UI          param;
    HAJCSSVCLI  hClient;

    // インスタンス生成
    hSsv = AjcSsvCreate();
    //  オプション設定
    AjcSsvSetOpt(hSsv, AJCSSV_SOP_SUP_SLEEP);
    // 送受信テキスト文字コード設定
    AjcSsvSetRxTextCode(hSsv, AJCSSV_TXT_AUTO);
    AjcSsvSetTxTextCode(hSsv, AJCSSV_TXT_AUTO);
    // バイナリチャンクモード設定（AJCSSV_CM_TEXT:テキストチャンク, AJCSSV_CM_BOTH:両方）
    AjcSsvSetChunkMode(hSsv, AJCSSV_CM_BIN);
    // 使用するイベント指定（デフォルトでは、全イベントが有効であるため、特に指定しなくてもよい）
    AjcSsvSetEvtMask(hSsv, AJCSSV_EV_RXCHUNK | AJCSSV_EV_SERVEND | AJCSSV_EV_ERR・・・・・);
    // サーバ開始       〈ポート番号〉    〈アドレスファミリ〉〈最大クライアント数〉 〈ウインドハンドル〉 〈通知メッセージ〉
    AjcSsvStart(hSsv, TEXT("14238"), AF_INET,        n,              NULL          , 0);
```

つづく

つづき

```
    while (!fEnd) {
        //  キー ファンクション（キー押下でサーバを停止する（任意））
        if (kbhit()) {
           // ボタン押下でサーバを停止する（この後に、接続中の全クライアントに対し「AJCSSV_EV_DISCONNECT」イベントが
           //  発生し、最後に「AJCSSV_EV_SERVEND」イベントが発生する）
           //  これらのイベントを辿る必要が無ければ、ここで「fEnd=TRUE」として終了してもよい。
            AjcSsvStop(hSsv, 10000);
        }
        //  イベント処理
        if (AjcSsvWaitEvent(hSsv, &wParam, &lParam, 200)) {
            HAJCSSVCLI      hClient= NULL;
            //  イベントデータ取得
            AjcSsvGetEventData(hSsv, lParam, &u.pVoid, &len, &param);
            //  クライアント情報を取得（取得できないイベントもあるので注意が必要（後述のイベント一覧参照））
            AjcSsvGetClient (hSsv, lParam, &hClient);
            //  イベント処理
            switch (wParam) {
                //  ●チャンクデータ受信通知
                case AJCSSV_EV_RXCHUNK:
                    //  受信データをループバック送信（任意）
                    AjcSsvSendText(hClient, pTxt, lTxt);
                    break;

                //  ●サーバ終了通知
                case AJCSSV_EV_SERVEND:
                    //  ループを抜けてプログラムを終了する
                    fEnd = TRUE;
                    break;

                //●送信エラー通知
                case AJCSSV_EV_TXERR:
                    // 送信エラーが発生したら、クライアント切断（任意）
                    AjcSsvDisconnect(hClient);
                    break;

                ・・・・・その他のイベント処理・・・・・

            }
            //  イベントデータ開放
            AjcSsvRelEventData(hSsv, lParam);
        }
    }
    //  ソケットサーバ インスタンス消去
    AjcSsvDelete(hSsv);

    return 0;
}
```

ユーザがイベントの発生を待ち受ける場合、相手局に何らかのデータを要求し、その場で応答データの着信を待つことも可能です。

```
AjcSsvSendText(hSsv, "データ要求コマンド", -1);                  //  相手局へデータ要求コマンド送信
if (AjcSsvWaitEvent(hSsv, &wParam, &lParam, 200)) {          //  イベント待ち(200ms)，イベントあり？
    AjcSsvGetEventData(hSsv, lParam, &pDat, &len, &param);   //    イベントデータ取得
    AjcSsvGetClient   (hSsv, lParam, &hClient);              //    クライアントハンドル取得
    ・・・・応答データ受信等のイベント処理・・・・
    AjcSsvRelEventData(hSsv, lParam);                        //    イベントデータ開放
}
```

14.2. イベント一覧

イベントの一覧を以下に示します。

AjcSsvGetClient ()による**クライアントハンドル**の取得
○：可能
×：不可

#	イベントコード (wParam)	内容	イベントデータ（AjcSsvGetEventData()で取得する情報）			クライアントの取得
			pDat	lDat	param	
1	AJCSSV_EV_RXNOPKT	パケット外テキスト通知	受信データのアドレス	受信データのバイト数／文字数	0：バイナリデータ 1：バイト文字列 2：ワイド文字列 (UNICODE)	○
2	AJCSSV_EV_RXTEXT (**※2**)	テキスト受信通知				○
3	AJCSSV_EV_RXESC	ＥＳＣコード受信通知				○
4	AJCSSV_EV_RXPKT	パケットデータ受信通知				○
5	AJCSSV_EV_RXCTRL	制御コード受信通知				○
6	AJCSSV_EV_RXCHUNK (**※1**)	チャンクデータ受信通知				○
7	AJCSSV_EV_TXEMPTY	送信完了通知	NULL	0	0	○
8	AJCSSV_EV_CONNECT	クライアント接続通知	NULL	0	0	○
9	AJCSSV_EV_DISCONNECT	クライアント切断通知	NULL	0	0	○
10	AJCSSV_EV_START	サーバ開始通知	NULL	0	0	×
11	AJCSSV_EV_STOP	サーバ停止通知	NULL	0	0	×
12	AJCSSV_EV_RXERR	受信エラー通知	NULL	0	0：WSARecv() 1：GetOverlappedResult()	○
13	AJCSSV_EV_TXERR	送信エラー通知	NULL	0	0：WSASend() 1：GetOverlappedResult()	○
14	AJCSSV_EV_ERR	エラー発生通知	NULL	0	エラー要因（**※3**）	×
15	AJCSSV_EV_SSEP	#1～#5　の合成値				
16	AJCSSV_EV_COMM	#6～#8　の合成値				
16	AJCSSV_EV_GENERAL	#9～#12　の合成値				
18	AJCSSV_EV_ERRS	#13～#15 の合成値				
19	AJCSSV_EV_CLIENT	#1～#11, #13, #14 の合成値（クライアントハンドルを取得可能なイベント）				
20	AJCSSV_EV_ALL	全イベント				

※1： テキストチャンクの場合、AjcSsvSetRxTextCode()により設定された文字コードからシフト JIS/UNICODE に変換したテキストを通知します。

※2： AjcSsvSetRxTextCode()により設定された文字コードからシフト JIS/UNICODE に変換したテキストを通知します。

※3：エラー要因（以下のいずれか）

#	通信エラーコード	内容
1	AJCSSV_ERR_CREEVT	イベントオブジェクト生成失敗
2	AJCSSV_ERR_SOCKET	接続要求待機用ソケット生成失敗(socket)
3	AJCSSV_ERR_BIND	バインド失敗（bindt）
4	AJCSSV_ERR_LISTEN	LISTEN 失敗(listen)
5	AJCSSV_ERR_ACCEPT	ACCEPT 失敗（accept）
6	AJCSSV_ERR_ADDRINFO	アドレス情報取得失敗（getaddrinfo）
7	AJCSSV_ERR_SOCKOPT	ソケットオプション設定失敗(setsockopt)
8	AJCSSV_ERR_THREADPOWCTRL	電源制御スレッド開始失敗
9	AJCSSV_ERR_THREADLISTEN	接続待機スレッド開始失敗
10	AJCSSV_ERR_THREADCLIENT	クライアント通信スレッド開始失敗
11	AJCSSV_ERR_CRESSEP	ストリーム分離インスタンス生成失敗
12	AJCSSV_ERR_CREMBXTXD	送信データ用メールボックス生成失敗
13	AJCSSV_ERR_TIMEOUT	サーバ終了タイムアウト
14	AJCSSV_ERR_CRETHREADWORK	クライアントスレッド用ワーク確保失敗
15	AJCSSV_ERR_CONNOVER	クライアント接続数オーバー
16	AJCSSV_ERR_CREIXMAP	インデクス割り当て用マップテーブル生成失敗

デフォルトでは（AjcSsvCreate()実行直後では）全てのイベントが許可状態となっています。
許可するイベントを選択する場合は、AjcSsvSetEvtMask()により、許可するイベントを指定してください。

14.3. サポートＡＰＩ

ソケット(TCP/IP)サーバのサポートＡＰＩ一覧を以下に示します。

サーバＡＰＩ

#	関数名	内容	スレッドセーフ
1	AjcSsvCreate	インスタンス生成	×
2	AjcSsvDelete	インスタンス消去	×
3	AjcSsvStart	サーバ起動	○
4	AjcSsvStop	サーバ停止	○
5	AjcSsvIsStarted	サーバの起動状態取得	○
6	AjcSsvSetOpt AjcSsvGetOpt	オプションの設定／取得	○
7	AjcSsvSetChunkMode AjcSsvGetChunkMode	チャンクデータの通知モード設定／取得	○
8	AjcSsvSetEvtMask AjcSsvGetEvtMask	イベントマスク設定／取得	○
9	AjcSsvSetRxTextCode AjcSsvGetRxTextCode	受信テキストの文字コード種別設定／取得	○
10	AjcSsvSetTxTextCode AjcSsvGetTxTextCode	送信テキストの文字コード種別設定／取得	○
11	AjcSsvWaitEvent	イベント発生待ち	○
12	AjcSsvGetEventData	イベントデータ取得	○
13	AjcSsvRelEventData	イベントデータ開放	○
14	AjcSsvWaitEmptyEvent	イベントが空になるまで待つ	－
15	AjcSsvGetClient	クライアント・ハンドル取得	○
16	AjcSsvSetPktCtrlCode AjcSsvGetPktCtrlCode	パケットフレームを認識する為の制御コード設定／取得	○
17	AjcSsvSetPktTimeout AjcSsvGetPktTimeout	パケットフレーム受信タイムアウト値設定／取得	○
18	AjcSsvGetClientCount	接続済のクライアント数取得	○
19	AjcSsvGetConnectCount	通算接続回数取得	○
20	AjcSsvIsClientConnected	クライアントの接続状態取得	○
21	AjcSsvEnumClients	接続済の全クライアント情報取得	○

クライアントＡＰＩ

#	関数名	内容	スレッドセーフ
22	AjcSsvDisconnect	クライアントを切断する	○
23	AjcSsvSendChar	クライアントへ１文字送信	○
24	AjcSsvSendText AjcSsvSendTextF	クライアントへテキストデータ送信	○
25	AjcSsvSendBinData	クライアントへバイナリデータ送信	○
26	AjcSsvSendPacket	クライアントへパケットデータ送信	○
27	AjcSsvPurgeRecvData	全受信済データ破棄	○
28	AjcSsvPurgeSendData	全送信待ちデータ破棄	○
29	AjcSsvPurgeAllData	全受信済データと送信待ちデータ破棄	○
30	AjcSsvSetClientData	クライアントにデータを関連付ける	○
31	AjcSsvGetClientData	クライアントに関連付けられたデータを取得する	○
32	AjcSsvGetSeqNo	クライアント接続順序番号取得	○
33	AjcSsvGetIndex	クライアントに割り当てられたインデクス取得	○
34	AjcSsvGetIpAddrStr	クライアントのＩＰアドレスを文字列取得	○
35	AjcSsvGetSeqNo	クライアント接続順序番号取得	○
36	AjcSsvGetThreadId	クライアントスレッドのスレッドＩＤ取得	○
37	AjcSsvGetSocket	クライアントのソケット記述子取得	○
38	AjcSsvGetActualRxTextCode AjcSsvGetActualTxTextCode	実際の送受信文字コード種別取得	○

※クライアントＡＰＩは、当該クライアント切断通知イベント（AJCSSV_EV_DISCONNECT）発生後は実行できません。

14.3.1. インスタンス生成（AjcSsvCreate）

形　式 ： HAJCSSV　AjcSsvCreate();

引　数 ： なし

説　明 ： ソケット(TCP/IP)サーバのインスタンスを生成し、サーバ処理を開始します。

戻り値 ： ≠NULL - 成功（インスタンスハンドル）
＝NULL - 失敗

14.3.2. インスタンス消去（AjcSsvDelete）

形　式 ： BOOL　AjcSsvDelete(HAJCSSV hSsv)

引　数 ： hSsv　　- インスタンスハンドル

説　明 ： 全てのリソースを開放し、ソケット(TCP/IP)サーバのインスタンスを消去します。
AjcSsvStop()によりサーバを停止していない場合は、サーバを停止しますが、「AJCSSV_EV_DISCONNECT」や「AJCSSV_EV_STOP」イベントは発生しません。

戻り値 ： TRUE　- 成功
FALSE - 失敗

14.3.3. サーバ起動（AjcSsvStart）

形　式 ： BOOL　AjcSsvStart (HHAJCSSV hSsv, C_UTP pPort, int AddressFamily, UI MaxClients, HWND hWndNtc, UI WndMsgNtc);

引　数 ：
hSsv	- インスタンスハンドル
pPort	- ポート名／ＩＰアドレス
AddressFamily	- アドレスファミリ(AF_INET / AF_INET6)
MaxClients	- 最大クライアント数
hWndNtc	- イベント通知用ウインドハンドル（AjcSsvWaitEvent()でイベントを待つ場合はNULL）
WndMsgNtc	- イベント通知用メッセージ　　（AjcSsvWaitEvent()でイベントを待つ場合は０）

説　明 ： サーバ機能を開始します。
イベントをウインドメッセージで受け取る場合は、hWndNtc に通知するウインドのハンドル, WndMsgNtc にウインドメッセージコード(WM_USER+100 以降, WM_APP+500 以降か、RegisterWindowMessage()で取得したコード）を指定します。
このウインドメッセージが通知された場合、wParam にイベントコードが,lParam にイベントデータ取得情報が設定されます。実際に、イベントデータを取得するには、lParam を指定して、AjcSsvGetEventData()を実行します。

イベントを AjcSsvWaitEvent()で待つ場合は、hWndNtc=NULL, WndMsgNtc=0 とします。

戻り値 ： TRUE　- 成功
FALSE - 失敗

14.3.4. サーバ停止（AjcSsvStop）

形　式 ： BOOL　AjcSsvStop (HAJCSSV hSsv, UI msTimeout)

引　数 ：
hSsv	- インスタンスハンドル
msTimeout	- サーバ停止待ち時間

説　明 ： サーバ機能を停止します。
接続中の全クライアントに対し「AJCSSV_EV_DISCONNECT」イベントが発生し、最後に「AJCSSV_EV_STOP」イベントが発生します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

14.3.5. サーバ起動状態取得（AjcSsvIsStarted）

形　式 ： BOOL　AjcSsvIsStarted (HAJCSSV hSsv)

引　数 ： hSsv　　- インスタンスハンドル
msTimeout　- サーバ停止待ち時間

説　明 ： サーバの起動状態を取得します。

戻り値 ： TRUE　- サーバ起動中
FALSE - サーバ停止中

14.3.6. オプションの設定／取得（AjcSsv{Set/Get}Opt）

形　式 ： BOOL　　AjcSsvSetOpt(HAJCSSV hSsv, AJCSSV_SERVOPT opt); --- 設定
AJCSSV_SERVOPT　AjcSsvGetOpt(HAJCSSV hSsv);　---------- 取得

引　数 ： hSsv　　- インスタンスハンドル
opt　　- オプション（AJCSSV_SOP_*XXXXX*）

説　明 ： サーバの動作オプションを設定／取得します。オプションには以下のシンボルを指定します。

オプション	内容	備考
AJCSSV_SOP_NOOPT	オプション無し	デフォルト値
AJCSSV_SOP_SUP_SLEEP	スリープ抑止	機種によっては、抑止できない場合があります。
AJCSSV_SOP_SUP_DISPOFF	ディスプレィ OFF 抑止	
AJCSSV_SOP_SUP_ALL	スリープ&ディスプレィ OFF 抑止	

戻り値 ： 設定時：TRUE　- 成功　　取得時：オプション（AJCSSV_SOP_*XXXXX*）
FALSE - 失敗

14.3.7. チャンクデータの通知モード設定／取得（AjcSsv{Set/Get}ChunkMode）

形　式 ： BOOL　AjcSsvSetChunkMode (HAJCSSV hSsv, AJCSSV_CHUNKMODE ChunkMode); -- チャンクデータ通知モードの設定
AJCSSV_CHUNKMODE　AjcSsvGetChunkMode (HAJCSSV hSsv);　---------- チャンクデータ通知モードの取得

引　数 ： hSsv　　- インスタンスハンドル
ChunkMode　- 設定するチャンクデータの通知モード

説　明 ： チャンクデータの通知モードを設定／取得します。ChunkMode には以下のいずれかを指定します。

ChunkMode	内容
AJCSSV_CM_NONE	受信チャンクデータを通知しません
AJCSSV_CM_BIN	受信チャンクデータをバイナリデータとして通知します（デフォルト）
AJCSSV_CM_TEXT	受信チャンクデータをテキストデータとして通知します
AJCSSV_CM_BOTH	受信チャンクデータをバイナリデータ&テキストデータとして通知します (AJCSSV_EV_RXCHUNK イベントで、同じ受信チャンクデータが param=0(バイナリデータ)と、 param=1/2(テキストデータ)として２回通知されます)

戻り値 ： 設定時：TRUE　- 成功　　取得時：チャンクデータの受信モード
FALSE - 失敗

14.3.8. イベントマスク設定／取得（AjcSsv{Set/Get}EvtMask）

形　式 ： BOOL　AjcSsvSetEvtMask (HAJCSSV hSsv, UI Mask); -- イベントマスク設定
UI　　AjcSsvGetEvtMask (HAJCSSV hSsv); ---------- イベントマスク取得

引　数 ： hSsv　　- インスタンスハンドル
Mask　　- 設定するイベントマスク（イベントコードの合成値）

説　明 ： イベントマスク（使用するイベント）を設定／取得します。
イベントマスクのデフォルトは、全イベントです
イベントコードについては、本節の冒頭を参照してください。

戻り値 ： 設定時：TRUE　- 成功　　取得時：イベントマスク値
FALSE - 失敗

14.3.9. 受信文字コード種別設定／取得（AjcSsv{Set/Get}RxTextCode）

形　式 ： BOOL AjcSsvSetRxTextCode (HAJCSSV hSsv, AJCSSV_TEXTCODE code); -- 受信文字コード種別設定
AJCSSV_TEXTCODE AjcSsvGetRxTextCode (HAJCSSV hSsv); ------------------------ 受信文字コード種別取得

引　数 ： hSsv - インスタンスハンドル
code - 受信テキストコード種別
・AJCSSV_TXT_SJIS - シフトＪＩＳ（デフォルト）
・AJCSSV_TXT_EUC - 日本語ＥＵＣ
・AJCSSV_TXT_UTF8 - ＵＴＦ－８
・AJCSSV_TXT_AUTO - 自動判別

説　明 ： 受信テキストの文字コード種別を指定します。
AJCSSV_TXT_AUTO を設定した場合は、実際に受信したテキストから自動判別します。

戻り値 ： 設定時：TRUE - 成功　　取得時：受信文字コード種別（AJCSSV_TXT_SJIS / EUC / UTF8 / AUTO）
FALSE - 失敗　　エラー時は 0 を返す

備　考 ： 受信したテキストは、指定文字コードから、バイト文字の場合はシフトＪＩＳに、UNICODE モードの場合は UTF-16 に変換して通知されます。
AJCSSV_TXT_AUTO（自動判別）を設定しても、受信中に文字コードが変化した場合は、変化前の文字コードと混在した状態となるため、しばらくは受信テキストの文字コードが正常に判断されない場合があります。

14.3.10. 送信文字コード種別設定／取得（AjcSsv{Set/Get}TxTextCode）

形　式 ： BOOL AjcSsvSetTxTextCode (HAJCSSV hSsv, AJCSSV_TEXTCODE code); -- 送信文字コード種別設定
AJCSSV_TEXTCODE AjcSsvGetTxTextCode (HAJCSSV hSsv); ------------------------ 送信文字コード種別取得

引　数 ： hSsv - インスタンスハンドル
code - 受信テキストコード種別
・AJCSSV_TXT_SJIS - シフトＪＩＳ（デフォルト）
・AJCSSV_TXT_EUC - 日本語ＥＵＣ
・AJCSSV_TXT_UTF8 - ＵＴＦ－８
・AJCSSV_TXT_AUTO - 受信文字コード種別と同じ

説　明 ： 送信テキストの文字コード種別を設定／取得します。
AJCSSV_TXT_AUTO を設定した場合は、受信文字コード種別と同じ設定となります。

戻り値 ： 設定時：TRUE - 成功　　取得時：送信文字コード種別（AJCSSV_TXT_SJIS / EUC / UTF8 / AUTO）
FALSE - 失敗　　エラー時は 0 を返す

備　考 ： AjcSsvSendText()やAjcSsvSendChar()で送信するテキストは、指定文字コードに変換されて送信されます。

14.3.11. イベント発生待ち（AjcSsvWaitEvent）

形　式 ： BOOL　AjcSsvWaitEvent (HAJCSSV hSsv, WPARAM *pwParam, LPARAM *plParam, UI msTime);

引　数 ： hSsv　　- インスタンスハンドル
pwParam　　- 発生したイベントコード格納するバッファのアドレス
plParam　　- イベントデータ取得情報を格納するバッファのアドレス
msTime　　- 待ち時間[ms]（－１を指定した場合は、永久にイベント待ちとなる）

説　明 ： イベントの発生を待ちます。
本関数によりイベントの発生を待つには、AjcSsvSetMode()で、hWndNtc 引数に NULL を指定していなければなりません。

pwParam で指定したバッファには、発生したイベントコードが格納されます。
イベントコードについては、本節冒頭の「イベントコード一覧」を参照してください。

plParam で指定したバッファには、イベントデータ取得情報が設定されます。
この情報は、AjcSsvGetEventData()の lParam 引数に指定します。

戻り値 ： TRUE　- イベントが発生した
FALSE - タイムアウト（イベント未発生）／エラー

14.3.12. イベントデータ取得（AjcSsvGetEventData）

形　式 ： BOOL　AjcSsvGetEventData (HAJCSSV hSsv, LPARAM lParam, VOP *ppDat, UIP plDat, UIP pParam);

引　数 ： hSsv　　- インスタンスハンドル
lParam　　- イベントデータ取得情報
ppDat　　- 受信データへのポインタを格納するバッファのアドレス
plDat　　- 受信データのバイト数／文字数を格納するバッファのアドレス
pParam　　- パラメタ情報を格納するバッファのアドレス

説　明 ： 発生したイベントに関する付随情報を取得します。
lParam は、ウインドメッセージの lParam／AjcSsvWaitEvent()で取得したイベントデータ取得情報を指定します。
ppDat～pParam で取得される情報については、本節冒頭の「イベントコード一覧」を参照してください。

本関数は、イベントが発生した場合、必ず実行してください。
また、イベントで通知された情報を使用した後には、AjcSsvRelEventData()を実行しなければなりません。

戻り値 ： TRUE　- 成功
FALSE - 失敗

14.3.13. イベントデータ開放（AjcSsvRelEventData）

形　式 ： BOOL　AjcSsvRelEventData (HAJCSSV hSsv, LPARAM lParam);

引　数 ： hSsv　　- インスタンスハンドル
lParam　　- イベントデータ取得情報

説　明 ： イベントデータを開放します。
lParam は、ウインドメッセージの lParam／AjcSsvWaitEvent()で取得したイベントデータ取得情報を指定します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

14.3.14. イベントが空になるまで待つ（AjcSsvWaitEmptyEvent）

形　式 ： BOOL　AjcSsvWaitEmptyEvent (HAJCSSV hSsv, UI msTime);

引　数 ： hSsv　　- インスタンスハンドル
msTime　　- 最大待ち時間[ms] (-1 : 永久)

説　明 ： イベントが空になるまで待ちます。(AjcSsvGetEventDat()で、非同期に蓄積されたイベントが全て取得されるまで待つ)
msTime で指定した時間が経過しても空にならない場合は、待つのをやめて FALSE を返します。

戻り値 ： TRUE　- イベントが空になった
FALSE - 失敗／指定時間内にイベントは空にならなかった

14.3.15. ライアント・ハンドル取得（AjcSsvGetEventData）

形　式 ： UI　AjcSsvGetClient (HAJCSSV hSsv, LPARAM lParam, HAJCSSVCLI *hClient);

引　数 ： hSsv　　- インスタンスハンドル
lParam　　- イベントデータ取得情報
hClient　　- クライアント・ハンドルを格納するバッファのアドレス

説　明 ： 発生したイベントに関するクライアント・ハンドルを取得します。
このＡＰＩは、AjcSsvGetEventData ()～AjcSsvRelEventData()の間で実行してください。
lParam は、ウインドメッセージの lParam／AjcSsvWaitEvent()で取得したイベントデータ取得情報を指定します。
クライアントハンドルは、イベントによって取得の可否が決まっています。(詳細は、本節冒頭の「イベントコード一覧」中の「クライアント取得」を参照)

戻り値 ： １～ - 成功（クライアント接続順序番号）
０　- 失敗

備　考 ： クライアント・ハンドルは、当該イベントの発生元となったクライアントを示します。
例えば、受信通知系のイベント(AJCSSV_EV_RXTEXT 等)が発生した場合、送信元のクライアントを示します。
送信元へデータを返信する場合は、クライアントハンドルを指定して「AjcSsvSendText(**hClient**, *pTxt*, *lTxt*)」のように実行します。

14.3.16. パケットフレーム認識・制御コード設定／取得（AjcSsv{Set/Get}PktCtrlCode）

形　式 ： BOOL　AjcSsvSetPktCtrlCode (HAJCSSV hSsv, UI　stx, UI　etx,　UI　dle);　-- 制御コード設定
BOOL　AjcSsvGetPktCtrlCode (HAJCSSV hSsv, UIP pStx, UIP pEtx, UIP pDle); -- 制御コード取得

引　数 ： hSsv　　- インスタンスハンドル
制御コード設定時
stx　　- STX のコード値（設定しない場合は、０を指定）
etx　　- ETX のコード値（設定しない場合は、０を指定）
dle　　- DLE のコード値（設定しない場合は、０を指定）
制御コード取得時
pStx　　- STX のコード値を格納するバッファのアドレス（不要時はＮＵＬＬ）
pEtx　　- ETX のコード値を格納するバッファのアドレス（不要時はＮＵＬＬ）
pDle　　- DLE のコード値を格納するバッファのアドレス（不要時はＮＵＬＬ）

説　明 ： パケットフレームを認識する為の制御コード（STX, ETX, DLE）の値を設定／取得します。
デフォルトでは、STX=0x02, ETX=0x03, DLE=0x10 となっています。

戻り値 ： TRUE　- 成功
FALSE - 失敗

14.3.17. パケット受信タイムアウト値 設定／取得（AjcSsv{Set/Get}PktTimeout）

形　式 ： BOOL　AjcSsvSetPktTimeout (HAJCSSV hSsv, UI　msTime)； -- タイムアウト値設定
BOOL　AjcSsvGetPktTimeout (HAJCSSV hSsv, UIP pMsTime)； -- タイムアウト値取得

引　数 ： hSsv　　- インスタンスハンドル
msTime　　- タイムアウト値[ms]（０：タイムアウトなし）

説　明 ： パケットフレーム受信時のタイムアウト値を設定／取得します。
パケットフレーム受信時において、指定時間内にパケット受信が完了しない場合、当該パケットを破棄します。
デフォルトでは、３０００ｍｓに設定されています。

戻り値 ： TRUE　- 成功
FALSE - 失敗

14.3.18. 接続中のクライアント数取得（AjcSsvGetClientCount）

形　式 ： UI　AjcSsvGetClientCount (HAJCSSV hSsv)；

引　数 ： hSsv　　- インスタンスハンドル

説　明 ： 接続中のクライアント数を取得します。

戻り値 ： 接続中のクライアント数（エラー時は０を返す）

14.3.19. クライアントの通算接続回数取得（AjcSsvGetConnectCount）

形　式 ： UI　AjcSsvGetConnectCount (HAJCSSV hSsv)；

引　数 ： hSsv　　- インスタンスハンドル

説　明 ： サーバ起動後、クライアントの通算接続回数を取得します。

戻り値 ： クライアントの通算接続回数（０は接続したクライアント無し／エラー）

14.3.20. クライアントの接続状態取得（AjcSsvIsClientConnected）

形　式 ： BOOL　AjcSsvIsClientConnected (HAJCSSV hSsv, HAJCSSVCLI hClient)；

引　数 ： hSsv　　- インスタンスハンドル
hClient　　- クライアントハンドル

説　明 ： クライアントの接続状態を取得します。

戻り値 ： TRUE　： 接続状態
FALSE ： 切断状態

14.3.21. 接続中の全クライアント取得（AjcSsvEnumClients）

形　式　：　UI AjcSsvEnumClients (HAJCSSV hSsv, UX cbp, BOOL (CALLBACK *cbNtcClients)(HAJCSSVCLI pC, PUX cbp));

引　数　：　hSsv　　- インスタンスハンドル
cbp　　- コールバックパラメタ
cbNtcClients - クライアント情報通知用コールバック関数（不要時はNULL）

説　明　：　コールバック関数(cbNtcClients)を呼び出して、接続中の全クライアントを通知します。
cbNtcClients＝NULLとした場合は、単に接続中のクライアント数を返します。

戻り値　：　接続中のクライアント数（エラー時は0を返す）クライアントハンドル

コールバック：　コールバック関数の仕様は、以下のとおりです。

cbNtcClients（クライアント通知）

形　式　：　BOOL　CALLBACK ***cbNtcClients*** (HAJCSSVCLI hClient, UX cbp);

引　数　：　hClient　- クライアント・ハンドル
cbp　　- コールバックパラメタ

説　明　：TRUEを返す間、現在の全てのノードについて、このコールバック関数が呼び出されます。
FALSEを返すと、クライアントの通知を中止します。

戻り値　：TRUE　- ノード読み出し継続
FALSE - ノード読み出し中止

14.3.22. クライアントを切断する（AjcSsvDisconnect）

形　式 ： BOOL AjcSsvDisconnect(HAJCSSVCLI hClient);

引　数 ： hClient　　- クライアントハンドル

説　明 ： hCli で指定した、接続中のクライアントを切断します。
クライアントが切断されたら「AJCSSV_EV_DISCONNECT」イベントが発生します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

14.3.23. クライアントへ１文字送信（AjcSsvSendChar）

形　式 ： BOOL　AjcSsvSendChar(HAJCSSVCLI hClient, UT code);

引　数 ： hClient　　- クライアントハンドル
code　　- 送信する文字のバイトコード／文字コード

説　明 ： 指定されたバイトコードの文字を送信します。
バイト文字モードの場合、マルチバイト文字を送信する場合は、この関数を２度コールしてください。
UNICODE モードの場合は、送信文字をマルチバイト文字に変換して送信します。
この関数では、送信用のスプールバッファに送信データを格納するだけであり、すぐに制御を戻します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

14.3.24. クライアントへテキストデータ送信（AjcSsvSendText）

形　式 ： BOOL　AjcSsvSendText (HAJCSSVCLI hClient, C_UTP pTxt, UI lTxt); ---- テキスト送信
BOOL　AjcSsvSendTextF(HAJCSSVCLI hClient, C_UTP pFmt, ...); --------- 書式テキスト送信

引　数 ： hClient　　- クライアントハンドル
pTxt　　- 送信するテキストデータのアドレス
lTxt　　- 送信するテキストデータのバイト数／文字数（－１の場合は自動算出）
pFmt　　- 書式テキスト（printf()と同じ）

説　明 ： テキストデータの送信を行います。
AjcSsvSetTxTextCode()により、送信文字コードが指定されている場合は、当該文字コードに変換したテキストを送信します。

この関数では、送信用のスプールバッファに送信データを格納するだけであり、すぐに制御を戻します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

14.3.25. クライアントへバイナリデータ送信（AjcSsvSendBinData）

形　式 ： BOOL　AjcSsvSendBinData(HAJCSSVCLI hClient, C_VOP pDat, UI lDat);

引　数 ： hClient　　- クライアントハンドル
pDat　　- 送信するバイナリデータのアドレス
lDat　　- 送信するバイナリデータのバイト数

説　明 ： バイナリデータの送信を行います。
pDat と lDat で指定したバイトストリームをそのまま送信します。

この関数では、送信用のスプールバッファに送信データを格納するだけであり、すぐに制御を戻します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

14.3.26. クライアントへパケットデータ送信（AjcSsvSendPacket）

形　式 ： UI　AjcSsvSendPacket(HAJCSSVCLI hClient, C_VOP pPkt, UI lPkt);

引　数 ： hClient　　- クライアントハンドル
pPkt　　- 送信するパケットデータのアドレス（空パケット送信時はNULL）
lPkt　　- 送信するパケットデータのバイト数（空パケット送信時は０）

説　明 ： 指定されたパケットデータをパケットフレームに乗せて送信します。
つまり、パケットデータ中のDLEを２つのDLEに変換し、先頭にDLE・STXを、末尾にDLE・ETXを付加したデータを送信します。
pPkt=NULL, lPkt=0を指定した場合は空パケット（DLE,STX,DLE,ETXの４バイト）を送信します。
この関数では、送信用のスプールバッファに送信データを格納するだけであり、すぐに制御を戻します。

戻り値 ： 4～　- 成功（DLE・STX, DLE・ETXや、パケットデータ中の透過制御バイト(DLE)を含めた実際の送信バイト数）
0　　- 失敗

14.3.27. 全受信済データ破棄（AjcSsvPurgeRecvData）

形　式 ： BOOL　AjcSsvPurgeRecvData (HAJCSSVCLI hClient);

引　数 ： hClient　　- クライアントハンドル

説　明 ： クライアントの受信中テキスト／ＥＳＣコード／制御コード／パケットデータ／パケット外データを全て破棄します。
受信チャンクデータについては、受信時にリアルタイムに通知される為、破棄対象外です。

戻り値 ： TRUE　- 成功
FALSE - 失敗

14.3.28. 全送信待ちデータ破棄（AjcSsvPurgeSendData）

形　式 ： BOOL　AjcSsvPurgeSendData (HAJCSSVCLI hClient);

引　数 ： hClient　　- クライアントハンドル

説　明 ： クライアントの送信待ちデータを全て破棄します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

14.3.29. 全受信済データと全送信待ちデータ破棄（AjcSsvPurgeAllData）

形　式 ： BOOL　AjcSsvPurgeAllData (HAJCSSVCLI hClient);

引　数 ： hClient　　- クライアントハンドル

説　明 ： クライアントの全受信済データと全送信待ちデータを破棄します。
AjcSsvPurgeRecvData()とAjcSsvPurgeSendData()を実行します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

14.3.30. クライアントにデータを関連付ける（AjcSsvSetClientData）

形　式 ： BOOL　AjcSsvSetClientData (HAJCSSVCLI hClient, UX data);

引　数 ： hClient　　- クライアントハンドル
data　　- クライアントに関連付けるデータ

説　明 ： クライアントにデータを関連付けます。

戻り値 ： TRUE　: 成功
FALSE : 失敗

14.3.31. クライアントに関連付けられたデータを取得する（Ajcssvgetclient Data）

形　式 ： BOOL　AjcSsvGetClientData(HAJCSSVCLI hClient, UXP pData);

引　数 ： hClient　　- クライアントハンドル
pData　　- クライアントに関連付けたデータを格納するバッファのアドレス

説　明 ： クライアントに関連付けられたデータを取得します。

戻り値 ： TRUE　: 成功
FALSE : 失敗

14.3.32. クライアント接続順序番号取得（AjcSsvGetSeqNo）

形　式 ： UI　AjcSsvGetSeqNo (HAJCSSVCLI hClient);

引　数 ： hClient　　- クライアントハンドル
pData　　- クライアントに関連付けたデータを格納するバッファのアドレス

説　明 ： クライアント接続順序番号（サーバ開始後、何回目の接続かを示す数値）を取得します。

戻り値 ： ≠ 0　: クライアント接続順序番号（１～）
＝ 0　: 失敗

14.3.33. クライアントに割り当てられたインデクス取得（AjcSsvGetIndex）

形　式 ： UI　AjcSsvGetIndex (HAJCSSVCLI hClient);

引　数 ： hClient　　- クライアントハンドル

説　明 ： クライアントに割り当てられたインデクスを取得します。
クライアントが接続した際には、当該クライアントに、接続中のクライアントで重複しないインデクス値が割り当てられます。

戻り値 ： ≠ -1　: クライアントに割り当てられたインデクス（０ ～ クライアント最大接続数－１）
＝ -1　: 失敗

備　考 ： このインデクスは、例えば、アプリケーションプログラムにおいて多数のクライアントを管理する場合に、クライアント管理テーブルのインデクスとして利用できます。

14.3.34. クライアントのＩＰアドレス文字列取得（AjcSsvGetIpAddrStr）

形　式 ： C_UTP　AjcSsvGetIpAddrStr (HAJCSSVCLI hClient, UTP pBuf, UI lBuf);

引　数 ： hClient　　- クライアントハンドル
pBuf　　- ＩＰアドレス文字列を格納するバッファのアドレス
lPkt　　- ＩＰアドレス文字列を格納するバッファの文字数

説　明 ： クライアントのＩＰアドレス文字列を作成します。

戻り値 ： ≠ NULL ： 作成したＩＰアドレス文字列へのポインタ（=pBuf）
＝ NULL ： 失敗

14.3.35. クライアント接続順序番号取得（AjcSsvGetSeqNo）

形　式 ： UI　　AjcSsvGetSeqNo (HAJCSSVCLI hClient);

引　数 ： hClient　　- クライアントハンドル

説　明 ： サーバ起動後、クライアント接続順序番号（何回目に接続したクライアントか）を返します。

戻り値 ： ≠ ０ ： クライアント接続順序番号
＝ ０ ： 失敗

14.3.36. クライアントスレッドのスレッドＩＤ取得（AjcSsvGetThreadId）

形　式 ： UI　　AjcSsvGetThreadId (HAJCSSVCLI hClient);

引　数 ： hClient　　- クライアントハンドル

説　明 ： クライアントスレッドのスレッドＩＤを取得します。

戻り値 ： ≠ ０ ： クライアントスレッドのスレッドＩＤ
＝ ０ ： 失敗

14.3.37. クライアントのソケット記述子取得（AjcSsvGetSocket）

形　式 ： SOCKET　　AjcSsvGetSocket (HAJCSSVCLI hClient);

引　数 ： hClient　　- クライアントハンドル

説　明 ： クライアントの（accept で設定された）ソケット記述しを取得します。

戻り値 ： ≠ ０ ： クライアントスレッドのスレッドＩＤ
＝ ０ ： 失敗

14.3.38. 実際の送受信文字コード種別取得（AjcSsvGetActual{Rx/Tx}TextCode）

形　式 ： AJCSSV_TEXTCODE AjcSsvGetActualRxTextCode(HAJCSSVCLI hClient);
AJCSSV_TEXTCODE AjcSsvGetActualTxTextCode(HAJCSSVCLI hClient);

引　数 ： hClient　　- クライアントハンドル

説　明 ： 実際の送受信文字コードを取得します。
AjcSsvGetActualRxTextCode()では、AJCSSV_TXT_AUTO が設定されている場合は、実際に受信したテキストから判断された文字コード種別を返します。(デフォルトは AJCSSV_TXT_SJIS)
AjcSsvGetActualTxTextCode()では、AJCSSV_TXT_AUTO が設定されている場合は、受信文字コード種別を返します。

戻り値 ： ≠ 0 ： 文字コード種別 (AJCSSV_TXT_SJIS / EUC / UTF8)
＝ 0 ： 失敗

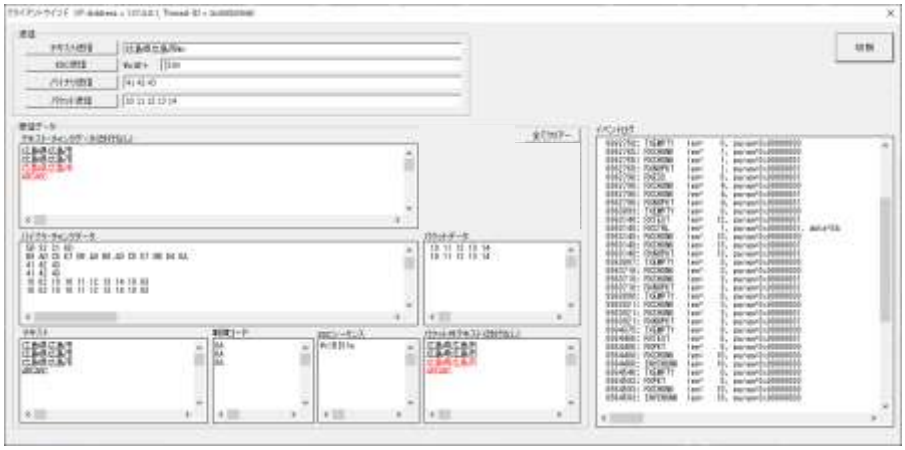
14.4. サンプルプログラム

14.4.1. SW_SockServer1（送受信テスト）

このサンプルプログラムは、ソケットサーバ機能における送受信ファンクションを実行します。
サンプルプログラム（Sil_SockClient1）と対向して通信できます。



接続毎にクライアントとの通信ウインドを開く



```
 1 : //
 2 : //  SW_SockServer1.c
 3 : //
 4 : #define AJCSERIALCOMPORT_H_
 5 : #include    <AjrCstXX.h>
 6 : #include    "resource.h"
 7 :
 8 : #define WM_SSVEVENT (WM_APP + 100)
 9 : #define MAX_TXTBOX_LEN  128
10 :
11 : //-------------------------------------------------------------------------------------------------------------------//
12 : //  ツールチップ                                                                                                     //
13 : //-------------------------------------------------------------------------------------------------------------------//
14 : typedef struct {
15 :     int     id;
16 :     C_UTP   pTxt;
17 : } TIPTBL, *PTIPTBL;
18 : typedef const TIPTBL *PCTIPTBL;
19 :
20 : static const TIPTBL TipTbl[] = {
21 :     {IDC_TXT_SNDTEXT    , TEXT("Ｃ言語表記のテキストを設定してください。(ex. ABC¥¥n) ")                                  },
22 :     {IDC_VTH_TXTCHUNK   , TEXT("リアルタイムに受信したデータをテキストデータとして表示します。¥n")
23 :                           TEXT("複数バイト文字の場合は、文字が完結するまで次のリアルタイム受信データを待ちます。")          },
24 :     {IDC_VTH_BINCHUNK   , TEXT("リアルタイムに受信したデータをバイナリデータとして表示します。")                        },
25 :     {IDC_VTH_TEXT       , TEXT("受信ストリームから制御コード（TAB 以外）で区切られたテキストデータを抜き出して表示します。¥n")
26 :                           TEXT("¥x1B[34m ここにファイルをドロップすると、ファイルの内容をバイナリデータとして送信します。") },
27 :     {IDC_VTH_CTRL       , TEXT("受信ストリームから制御コード（TAB 以外）を抜き出して表示します。")                      },
28 :     {IDC_VTH_ESC        , TEXT("受信したストリームから、ＥＳＣシーケンス（0x1B～英字）を抜きだして表示します。")          },
29 :     {IDC_VTH_PKT        , TEXT("受信したパケットデータ（DLE・STX～DLE・ETX でサンドイッチされたデータ）をバイナリ表示します。")},
30 :     {IDC_VTH_NOPKT      , TEXT("リアルタイムに受信したデータ内のパケットデータ（DLE・STX～DLE・ETX）以外の部分をテキストとして表示します。¥n")
31 :                           TEXT("複数バイト文字の場合は、文字が完結するまで次のリアルタイム受信データを待ちます。")}
32 : };
33 :
34 : #define MAX_TIPTBL  (sizeof TipTbl / sizeof TipTbl[0])
35 :
36 : //-------------------------------------------------------------------------------------------------------------------//
37 : //  ワーク                                                                                                           //
38 : //-------------------------------------------------------------------------------------------------------------------//
39 : static  HINSTANCE      hInst;                      //  ＤＬＬインスタンスハンドル
40 : static  HWND           hDlgMain;                   //  ダイアログボックスハンドル
41 : static  HWND           hLbxClients;
42 : static  HAJCSSV        hSsv;
43 :
44 : //-------------------------------------------------------------------------------------------------------------------//
45 : //  内部サブ関数                                                                                                     //
```

```
46 : //-------------------------------------------------------------------------------------------------------------//
47 : AJC_DLGPROC_DEF(Main);
48 : AJC_DLGPROC_DEF(Client);
49 : static  VO  SetRxTextCode(HWND hDlg);
50 : static  VO  SetTxTextCode(HWND hDlg);
51 :
52 : //=============================================================================================================//
53 : //                                                                                                             //
54 : //  W i n M a i n                                                                                              //
55 : //                                                                                                             //
56 : //=============================================================================================================//
57 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
58 : {
59 :     MSG     msg;
60 :
61 :     //----- メイン・ダイアログオープン -----------------//
62 :     hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_MAIN), NULL, AJC_DLGPROC_NAME(Main));
63 :     //----- ダイアログ表示 -----------------------------//
64 :     ShowWindow(hDlgMain, SW_SHOW);
65 :     //----- メッセージループ ---------------------------//
66 :     while (GetMessage(&msg, NULL, 0, 0)) {
67 :         do {
68 :             if (IsDialogMessage(hDlgMain, &msg)) break;
69 :             TranslateMessage(&msg);
70 :             DispatchMessage (&msg);
71 :         } while (0);
72 :     }
73 :     return (int)msg.wParam ;
74 : }
75 : //=============================================================================================================//
76 : //                                                                                                             //
77 : //  メイン ダイアログ・プロシージャ                                                                            //
78 : //                                                                                                             //
79 : //=============================================================================================================//
80 : //----- ダイアログ初期化 -----------------------------------------------------------------------------------//
81 : AJC_DLGPROC(Main, WM_INITDIALOG     )
82 : {
83 :     UL      lHost;
84 :     UT      szHost[128];
85 :     UT      txt[256];
86 :
87 :     hDlgMain        = hDlg;
88 :     hLbxClients     = GetDlgItem(hDlg, IDC_LBX_CLIENTS);
89 :
90 :     //  ウインド位置ロード
91 :     AjcLoadWndPos(hDlg, NULL);
92 :     //  ラジオボタンのグループ化
93 :     AjcSbcRadioBtns(GetDlgItem(hDlg, IDC_GRP_RXTEC));
94 :     AjcSbcRadioBtns(GetDlgItem(hDlg, IDC_GRP_TXTEC));
95 :     //  ウインドタイトル表示
96 :     lHost = AJCTSIZE(szHost);
97 :     GetComputerName(szHost, &lHost);
98 :     AjcSnPrintF(txt, AJCTSIZE(txt), TEXT("SW_SockServer1 ( %s : 14238 )"), szHost);
99 :     SetWindowText(hDlg, txt);
100 :    //  ホスト名表示
101 :    AjcSetDlgItemStr(hDlg, IDC_LBL_MYNAME, szHost);
102 :    //  ダイアログ項目の初期化
103 :    AjcSetDlgItemUInt(hDlg, IDC_TXT_PORTNO   , 14238);
104 :    AjcSetDlgItemUInt(hDlg, IDC_TXT_MAXCLIENT, 10   );
105 :    AjcSetDlgItemChk (hDlg, IDC_RBT_RXSJIS   , TRUE );
106 :    AjcSetDlgItemChk (hDlg, IDC_RBT_RXUTF8   , FALSE);
107 :    AjcSetDlgItemChk (hDlg, IDC_RBT_RXEUC    , FALSE);
108 :    AjcSetDlgItemChk (hDlg, IDC_RBT_RXAUTO   , FALSE);
109 :    AjcSetDlgItemChk (hDlg, IDC_RBT_TXSJIS   , TRUE );
110 :    AjcSetDlgItemChk (hDlg, IDC_RBT_TXUTF8   , FALSE);
111 :    AjcSetDlgItemChk (hDlg, IDC_RBT_TXEUC    , FALSE);
112 :    AjcSetDlgItemChk (hDlg, IDC_RBT_TXAUTO   , FALSE);
113 :    AjcSetDlgItemUInt(hDlg, IDC_GRP_RXTEC    , 0    );
114 :    AjcSetDlgItemUInt(hDlg, IDC_GRP_TXTEC    , 0    );
115 :    //  ソケットサーバ インスタンス生成
116 :    hSsv = AjcSsvCreate();
117 :    //  送受信テキスト文字コード設定
118 :    SetRxTextCode(hDlg);
119 :    SetTxTextCode(hDlg);
120 :    //  イベントマスク設定
121 :    AjcSsvSetEvtMask(hSsv,  0                   |
122 :                            AJCSSV_EV_RXTEXT    |   //  テキストデータ通知
123 :                            AJCSSV_EV_RXESC     |   //  ＥＳＣコードデータ通知
124 :                            AJCSSV_EV_RXPKT     |   //  パケットデータ通知
125 :                            AJCSSV_EV_RXCTRL    |   //  制御コード通知
```

```
126 :                           AJCSSV_EV_RXNOPKT   |   //  パケット外テキスト
127 :                           AJCSSV_EV_RXCHUNK   |   //  チャンクデータ受信通知
128 :                           AJCSSV_EV_TXEMPTY   |   //  送信完了通知
129 :                           AJCSSV_EV_CONNECT   |   //  接続通知
130 :                           AJCSSV_EV_DISCONNECT|   //  切断通知
131 :                           AJCSSV_EV_START     |   //  サーバ開始通知
132 :                           AJCSSV_EV_STOP      |   //  サーバ停止通知
133 :                           AJCSSV_EV_RXERR     |   //  受信エラー通知
134 :                           AJCSSV_EV_TXERR     |   //  送信エラー通知
135 :                           AJCSSV_EV_OVERR     |   //  クライアント接続数オーバー
136 :                           AJCSSV_EV_ERR       |   //  その他のエラー通知
137 :                           0);
138 :     //  チャンクモード設定
139 :     AjcSsvSetChunkMode(hSsv, AJCSSV_CM_BOTH);
140 :     //  ダイアログ項目のロード
141 :     AjcLoadAllControlSettings (hDlg, TEXT("DlgSetting"), AJCOPT3(AJCCTL_SELACT_, ALL, NTCCHK, NTCRBT));
142 :
143 :     return TRUE;
144 : }
145 : //----- ウインド破棄 ----------------------------------------------------------------------------------------//
146 : //  クライアント・ダイアログ クローズ
147 : static BOOL CALLBACK cbDisconnect(HAJCSSVCLI hCli, UX cbp)
148 : {
149 :     HWND    hDlgCli;
150 :     if (AjcSsvGetClientData(hCli, (UXP)&hDlgCli)){
151 :         DestroyWindow(hDlgCli);
152 :     }
153 :     return TRUE;
154 : }
155 : //- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - //
156 : AJC_DLGPROC(Main, WM_DESTROY        )
157 : {
158 :     //  ウインド位置セーブ
159 :     AjcSaveWndPos(hDlg, NULL);
160 :     //  全クライアントダイアログ クローズ
161 :     AjcSsvEnumClients(hSsv, 0, cbDisconnect);
162 :     //  ダイアログ項目のセーブ
163 :     AjcSaveAllControlSettings(hDlg);
164 :     //  ソケットサーバ インスタンス消去
165 :     AjcSsvDelete(hSsv);
166 :
167 :     PostQuitMessage(0);
168 :     return TRUE;
169 : }
170 : //----- WM_COMMAND ------------------------------------------------------------------------------------------//
171 : AJC_DLGPROC(Main, WM_COMMAND        )
172 : {
173 :     int     id = LOWORD(wParam);
174 :     if (HIWORD(wParam) == BN_CLICKED) {
175 :         if      (id >= IDC_RBT_RXSJIS && id <= IDC_RBT_RXAUTO) SetRxTextCode(hDlg);
176 :         else if (id >= IDC_RBT_TXSJIS && id <= IDC_RBT_TXAUTO) SetRxTextCode(hDlg);
177 :     }
178 :     return TRUE;
179 : }
180 : //----- ソケット(TCP/IP)イベント ----------------------------------------------------------------------------//
181 : AJC_DLGPROC(Main, WM_SSVEVENT       )
182 : {
183 :     UI      time = GetTickCount();
184 :     UI      len, param;
185 :     union {UTP pTxt; BCP pBcp; UBP pBin; VOP pVop;} u;
186 :     HAJCSSVCLI  hCli = NULL;
187 :     UI          ThreadId = 0;
188 :     UT          szIpAddr[256] = {0};
189 :     HWND        hDlgCli;
190 :     HWND        hLog, hVth;
191 :
192 :     //  イベントデータ取得
193 :     AjcSsvGetEventData(hSsv, lParam, &u.pVop, &len, &param);
194 :     //  クライアント情報を取得し、クライアントのＩＰアドレスを設定
195 :     if (AjcSsvGetClient   (hSsv, lParam, &hCli)) {
196 :         ThreadId = AjcSsvGetThreadId(hCli);
197 :         AjcSsvGetIpAddrStr(hCli, szIpAddr, AJCTSIZE(szIpAddr));
198 :     }
199 :
200 :     switch (wParam) {
201 :         case AJCSSV_EV_START:                                   //●サーバ開始通知
202 :             //  サーバ開始／終了ボタン有効化変更
203 :             AjcEnableDlgItem(hDlgMain, IDC_CMD_START, FALSE);
204 :             AjcEnableDlgItem(hDlgMain, IDC_CMD_STOP , TRUE );
205 :             //  コントロールの無効化
```

```
206 :             AjcEnableDlgItem(hDlg, IDC_TXT_PORTNO   , FALSE);
207 :             AjcEnableDlgItem(hDlg, IDC_TXT_MAXCLIENT, FALSE);
208 :             AjcEnableDlgGroup(hDlg, IDC_GRP_RXTEC   , FALSE, FALSE);
209 :             AjcEnableDlgGroup(hDlg, IDC_GRP_TXTEC   , FALSE, FALSE);
210 :             break;
211 :
212 :         case AJCSSV_EV_STOP:                                    //●サーバ停止通知
213 :             //  サーバ開始／終了ボタン有効化変更
214 :             AjcEnableDlgItem(hDlgMain, IDC_CMD_START, TRUE );
215 :             AjcEnableDlgItem(hDlgMain, IDC_CMD_STOP , FALSE);
216 :             //  コントロールの有効化
217 :             AjcEnableDlgItem(hDlg, IDC_TXT_PORTNO   , TRUE);
218 :             AjcEnableDlgItem(hDlg, IDC_TXT_MAXCLIENT, TRUE);
219 :             AjcEnableDlgGroup(hDlg, IDC_GRP_RXTEC   , TRUE, TRUE);
220 :             AjcEnableDlgGroup(hDlg, IDC_GRP_TXTEC   , TRUE, TRUE);
221 :             break;
222 :
223 :         case AJCSSV_EV_CONNECT:                                 //●接続通知
224 :         {   int     ix;
225 :             UT      txt[256];
226 :             //  クライアント用ダイアログ生成
227 :             hDlgCli = CreateDialogParam(hInst, MAKEINTRESOURCE(IDD_CLIENT), NULL, AJC_DLGPROC_NAME(Client), (LPARAM)hCli);
228 :             AjcSnPrintF(txt, AJCTSIZE(txt), TEXT("クライアントウインド (IP-Address = %s, Thread-ID = 0x%08X) "), szIpAddr, ThreadId);
229 :             SetWindowText(hDlgCli, txt);
230 :             ShowWindow(hDlgCli, SW_SHOW);
231 :             //  クライアントにダイアログを関連付ける
232 :             AjcSsvSetClientData(hCli, (UX)hDlgCli);
233 :             //  リストボックスにクライアント登録
234 :             ix = AjcLbxAddString(hLbxClients, szIpAddr);
235 :             AjcLbxSetItemData(hLbxClients, ix, (UX)hCli);
236 :             break;
237 :         }
238 :         case AJCSSV_EV_DISCONNECT:                              //●切断通知
239 :             if (AjcSsvGetClientData(hCli, (UXP)&hDlgCli)){
240 :                 int     ix;
241 :                 //  クライアントダイアログを閉じる
242 :                 DestroyWindow(hDlgCli);
243 :                 //  リストボックスからクライアント削除
244 :                 if ((ix = AjcLbxFindString(hLbxClients, -1, szIpAddr)) >= 0) {
245 :                     AjcLbxDeleteString(hLbxClients, ix);
246 :                 }
247 :             }
248 :             break;
249 :
250 :         case AJCSSV_EV_RXCHUNK:                                 //●チャンクデータ受信通知
251 :             if (AjcSsvGetClientData(hCli, (UXP)&hDlgCli)){
252 :                 hLog = GetDlgItem(hDlgCli, IDC_VTH_EVTLOG);
253 :                 AjcVthPrintF(hLog, TEXT("%9d: RXCHUNK   len=%5d, param=0x%08X¥n"), time, len, param);
254 :                 //  バイナリチャンク
255 :                 if (param == 0) {
256 :                     hVth = GetDlgItem(hDlgCli, IDC_VTH_BINCHUNK);
257 :                     AjcVthHexDump(hVth, (C_VOP)u.pBin, len);
258 :                     AjcVthPrintF (hVth, TEXT("¥n"));
259 :                 }
260 :                 //  テキストチャンク
261 :                 else {
262 :                     hVth = GetDlgItem(hDlgCli, IDC_VTH_TXTCHUNK);
263 :                     AjcVthPutText(hVth, u.pTxt, -1);
264 :                 }
265 :             }
266 :             break;
267 :
268 :         case AJCSSV_EV_RXTEXT:                                  //●テキストデータ通知
269 :             if (AjcSsvGetClientData(hCli, (UXP)&hDlgCli)){
270 :                 hLog = GetDlgItem(hDlgCli, IDC_VTH_EVTLOG);
271 :                 hVth = GetDlgItem(hDlgCli, IDC_VTH_TEXT);
272 :                 AjcVthPrintF(hLog, TEXT("%9d: RXTEXT    len=%5d, param=0x%08X¥n"), time, len, param);
273 :                 AjcVthPrintF(hVth, TEXT("%s¥n"), u.pTxt);
274 :             }
275 :             break;
276 :
277 :         case AJCSSV_EV_RXESC:                                   //●ＥＳＣデータ通知
278 :             if (AjcSsvGetClientData(hCli, (UXP)&hDlgCli)){
279 :                 hLog = GetDlgItem(hDlgCli, IDC_VTH_EVTLOG);
280 :                 hVth = GetDlgItem(hDlgCli, IDC_VTH_ESC);
281 :                 AjcVthPrintF(hLog, TEXT("%9d: RXESC     len=%5d, param=0x%08X¥n"), time, len, param);
282 :                 AjcVthPrintF(hVth, TEXT("¥¥x1B%s¥n"), u.pTxt + 1);
283 :             }
284 :             break;
285 :
```

```
286 :         case AJCSSV_EV_RXCTRL:                                  //●制御コード通知
287 :             if (AjcSsvGetClientData(hCli, (UXP)&hDlgCli)){
288 :                 hLog = GetDlgItem(hDlgCli, IDC_VTH_EVTLOG);
289 :                 hVth = GetDlgItem(hDlgCli, IDC_VTH_CTRL);
290 :                 AjcVthPrintF(hLog, TEXT("%9d: RXCTRL    len=%5d, param=0x%08X, data=%02X¥n"), time, len, param, *u.pBin);
291 :                 AjcVthPrintF(hVth, TEXT("%02X¥n"), *u.pTxt);
292 :             }
293 :             break;
294 :
295 :         case AJCSSV_EV_RXPKT:                                   //●パケットデータ通知
296 :             if (AjcSsvGetClientData(hCli, (UXP)&hDlgCli)){
297 :                 hLog = GetDlgItem(hDlgCli, IDC_VTH_EVTLOG);
298 :                 hVth = GetDlgItem(hDlgCli, IDC_VTH_PKT);
299 :                 AjcVthPrintF (hLog, TEXT("%9d: RXPKT     len=%5d, param=0x%08X¥n"), time, len, param);
300 :                 AjcVthHexDump(hVth, (C_VOP)u.pBin, len);
301 :                 AjcVthPrintF (hVth, TEXT("¥n"));
302 :             }
303 :             break;
304 :
305 :         case AJCSSV_EV_TXEMPTY:                                 //●送信完了通知
306 :             if (AjcSsvGetClientData(hCli, (UXP)&hDlgCli)){
307 :                 hLog = GetDlgItem(hDlgCli, IDC_VTH_EVTLOG);
308 :                 AjcVthPrintF(hLog, TEXT("%9d: TXEMPTY   len=%5d, param=0x%08X¥n"), time, len, param);
309 :             }
310 :             break;
311 :
312 :         case AJCSSV_EV_RXNOPKT:                                 //●パケット外テキストデータ
313 :             if (AjcSsvGetClientData(hCli, (UXP)&hDlgCli)){
314 :                 hLog = GetDlgItem(hDlgCli, IDC_VTH_EVTLOG);
315 :                 hVth = GetDlgItem(hDlgCli, IDC_VTH_NOPKT);
316 :                 AjcVthPrintF (hLog, TEXT("%9d: RXNOPKT   len=%5d, param=0x%08X¥n"), time, len, param);
317 :                 AjcVthPutText(hVth, u.pTxt, -1);
318 :             }
319 :             break;
320 :
321 :         case AJCSSV_EV_RXERR:                                   //●受信エラー通知
322 :             if (AjcSsvGetClientData(hCli, (UXP)&hDlgCli)){
323 :                 hLog = GetDlgItem(hDlgCli, IDC_VTH_EVTLOG);
324 :                 AjcVthPrintF(hLog, TEXT("%9d: RXERR     len=%5d, param=0x%08X¥n"), time, len, param);
325 :             }
326 :             break;
327 :
328 :         case AJCSSV_EV_TXERR:                                   //●送信エラー通知
329 :             if (AjcSsvGetClientData(hCli, (UXP)&hDlgCli)){
330 :                 hLog = GetDlgItem(hDlgCli, IDC_VTH_EVTLOG);
331 :                 AjcVthPrintF(hLog, TEXT("%9d: TXERR     len=%5d, param=0x%08X¥n"), time, len, param);
332 :             }
333 :             break;
334 :
335 :         case AJCSSV_EV_ERR:                                     //●その他のエラー通知
336 :             hLog = GetDlgItem(hDlgMain, IDC_VTH_ERROR);
337 :             AjcVthPrintF(hLog, TEXT("%9d: ERR       len=%5d, param=0x%08X¥n"), time, len, param);
338 :             break;
339 :
340 :     }
341 :
342 :     //  イベントデータ開放
343 :     AjcSsvRelEventData(hSsv, lParam);
344 :
345 :     return TRUE;
346 : }
347 : //----- サーバ開始ボタン ----------------------------------------------------------------------------------//
348 : AJC_DLGPROC(Main, IDC_CMD_START     )
349 : {
350 :     UI      MaxClient;
351 :     UT      szPort[64];
352 :
353 :     if (HIWORD(wParam) == BN_CLICKED) {
354 :         AjcGetDlgItemStr (hDlg, IDC_TXT_PORTNO, szPort, AJCTSIZE(szPort));
355 :         MaxClient = AjcGetDlgItemUInt(hDlg, IDC_TXT_MAXCLIENT);
356 :         AjcSsvStart(hSsv, szPort, AF_INET, MaxClient, hDlg, WM_SSVEVENT);
357 :     }
358 :     return TRUE;
359 : }
360 : //----- サーバ停止ボタン ----------------------------------------------------------------------------------//
361 : AJC_DLGPROC(Main, IDC_CMD_STOP      )
362 : {
363 :     if (HIWORD(wParam) == BN_CLICKED) {
364 :         AjcSsvStop(hSsv, 10000);
365 :     }
```

```
366 :     return TRUE;
367 : }
368 : //----- 受信テキストエンコード ----------------------------------------------------------------------------//
369 : AJC_DLGPROC(Main, IDC_GRP_RXTEC     )
370 : {
371 :     if (HIWORD(wParam) == AJCRBTN_SELECT) {
372 :         //  受信テキストエンコード設定
373 :         switch (lParam) {
374 :             case 0: AjcSsvSetRxTextCode(hSsv, AJCSSV_TXT_SJIS); break;
375 :             case 1: AjcSsvSetRxTextCode(hSsv, AJCSSV_TXT_UTF8); break;
376 :             case 2: AjcSsvSetRxTextCode(hSsv, AJCSSV_TXT_EUC ); break;
377 :             case 3: AjcSsvSetRxTextCode(hSsv, AJCSSV_TXT_AUTO); break;
378 :         }
379 :         //  送信テキストエンコード設定 (送信エンコードが AUTO の場合、受信エンコードと同一とするため)
380 :         SendMessage(hDlg, WM_COMMAND, MAKELONG(IDC_GRP_TXTEC, AJCRBTN_SELECT), AjcGetDlgItemUInt(hDlg, IDC_GRP_TXTEC));
381 :     }
382 :     return TRUE;
383 : }
384 : //----- 送信テキストエンコード ----------------------------------------------------------------------------//
385 : AJC_DLGPROC(Main, IDC_GRP_TXTEC     )
386 : {
387 :     if (HIWORD(wParam) == AJCRBTN_SELECT) {
388 :         //  送信テキストエンコード設定
389 :         switch (lParam) {
390 :             case 0: AjcSsvSetTxTextCode(hSsv, AJCSSV_TXT_SJIS); break;
391 :             case 1: AjcSsvSetTxTextCode(hSsv, AJCSSV_TXT_UTF8); break;
392 :             case 2: AjcSsvSetTxTextCode(hSsv, AJCSSV_TXT_EUC ); break;
393 :             case 3: AjcSsvSetTxTextCode(hSsv, AJCSSV_TXT_AUTO); break;
394 :         }
395 :     }
396 :     return TRUE;
397 : }
398 : //----- キャンセル ----------------------------------------------------------------------------------------//
399 : AJC_DLGPROC(Main, IDCANCEL          )
400 : {
401 :     DestroyWindow(hDlg);
402 :     return TRUE;
403 : }
404 : //--------------------------------------------------------------------------------------------------------//
405 : AJC_DLGMAP_DEF(Main)
406 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG     )
407 :     AJC_DLGMAP_MSG(Main, WM_DESTROY        )
408 :     AJC_DLGMAP_MSG(Main, WM_COMMAND        )
409 :     AJC_DLGMAP_MSG(Main, WM_SSVEVENT       )
410 :
411 :     AJC_DLGMAP_CMD(Main, IDC_CMD_START     )
412 :     AJC_DLGMAP_CMD(Main, IDC_CMD_STOP      )
413 :     AJC_DLGMAP_CMD(Main, IDC_GRP_RXTEC     )
414 :     AJC_DLGMAP_CMD(Main, IDC_GRP_TXTEC     )
415 :     AJC_DLGMAP_CMD(Main, IDCANCEL          )
416 : AJC_DLGMAP_END
417 : //========================================================================================================//
418 : //                                                                                                        //
419 : //  クライアント ダイアログ・プロシージャ                                                                 //
420 : //                                                                                                        //
421 : //========================================================================================================//
422 : //----- ダイアログ初期化 ----------------------------------------------------------------------------------//
423 : AJC_DLGPROC(Client, WM_INITDIALOG   )
424 : {
425 :     HAJCSSVCLI  hCli = (HAJCSSVCLI)lParam;
426 :     UI          i;
427 :     UT          szIpAddr[256];
428 :     UT          sect[256];
429 :     //  ダイアログにクライアントハンドルを関連付ける
430 :     SetProp(hDlg, TEXT("ClientHandle"), (HANDLE)lParam);
431 :     //  ＩＰアドレス文字列取得
432 :     AjcSsvGetIpAddrStr(hCli, szIpAddr, AJCTSIZE(szIpAddr));
433 :     //  ウインド位置ロード
434 :     AjcSnPrintF(sect, AJCTSIZE(sect), TEXT("WndPos_%s"), szIpAddr);
435 :     AjcLoadWndPos(hDlg, sect);
436 :     //   設定値ロード
437 :     AjcSnPrintF(sect, AJCTSIZE(sect), TEXT("Settings_%s"), szIpAddr);
438 :     AjcLoadAllControlSettings(hDlg, sect, AJCCTL_SELACT_ALL);
439 :     //  テキストボックス長設定
440 :     AjcSetDlgItemEdtLimit(hDlg, IDC_TXT_SNDTEXT, MAX_TXTBOX_LEN - 1);
441 :     AjcSetDlgItemEdtLimit(hDlg, IDC_TXT_SNDESC , MAX_TXTBOX_LEN - 1);
442 :     AjcSetDlgItemEdtLimit(hDlg, IDC_TXT_SNDBIN , MAX_TXTBOX_LEN - 1);
443 :     AjcSetDlgItemEdtLimit(hDlg, IDC_TXT_SNDPKT , MAX_TXTBOX_LEN - 1);
444 :     //----- ツールチップ設定 ----------------------//
445 :     for (i = 0; i < MAX_TIPTBL; i++) {
```

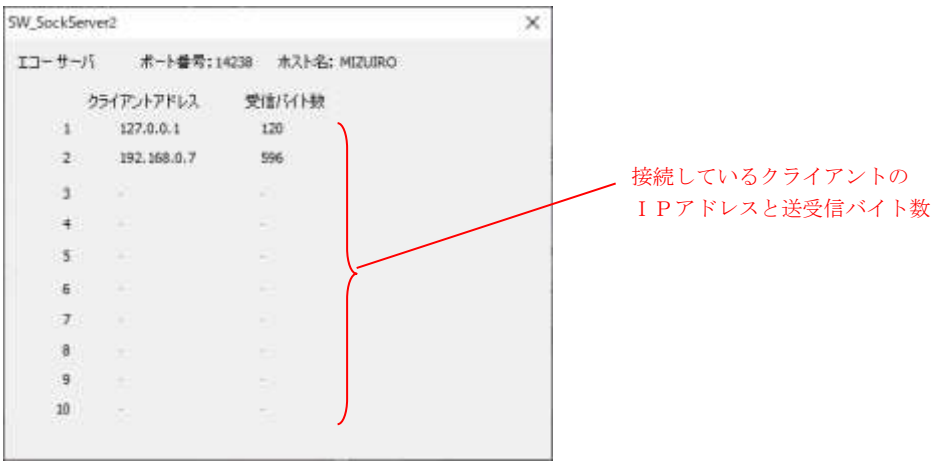
```
446 :         AjcTipTextAdd(GetDlgItem(hDlg, TipTbl[i].id), TipTbl[i].pTxt);
447 :     }
448 :     return TRUE;
449 : }
450 : //----- ウインド破棄 ----------------------------------------------------------------------------------//
451 : AJC_DLGPROC(Client, WM_DESTROY      )
452 : {
453 :     HAJCSSVCLI  hCli = (HAJCSSVCLI)GetProp(hDlg, TEXT("ClientHandle"));
454 :     UT          szIpAddr[256];
455 :     UT          sect[256];
456 :     //  ＩＰアドレス文字列取得
457 :     AjcSsvGetIpAddrStr(hCli, szIpAddr, AJCTSIZE(szIpAddr));
458 :     //  ウインド位置セーブ
459 :     AjcSnPrintF(sect, AJCTSIZE(sect), TEXT("WndPos_%s"), szIpAddr);
460 :     AjcSaveWndPos(hDlg, sect);
461 :     //   設定値セーブ
462 :     AjcSnPrintF(sect, AJCTSIZE(sect), TEXT("Settings_%s"), szIpAddr);
463 :     AjcSaveAllControlSettings(hDlg);
464 :     return TRUE;
465 : }
466 : //----- テキスト送信ボタン ----------------------------------------------------------------------------//
467 : AJC_DLGPROC(Client, IDC_CMD_SNDTEXT         )
468 : {
469 :     HAJCSSVCLI  hCli = (HAJCSSVCLI)GetProp(hDlg, TEXT("ClientHandle"));
470 :     UT          txt[MAX_TXTBOX_LEN];
471 :     UT          snd[MAX_TXTBOX_LEN];
472 :     AjcGetDlgItemStr(hDlg, IDC_TXT_SNDTEXT, txt, AJCTSIZE(txt));
473 :     AjcCLangStrToBin(txt, snd, AJCTSIZE(snd));
474 :     AjcSsvSendText(hCli, snd, -1);
475 :     return TRUE;
476 : }
477 : //----- ＥＳＣ送信ボタン ------------------------------------------------------------------------------//
478 : AJC_DLGPROC(Client, IDC_CMD_SNDESC          )
479 : {
480 :     HAJCSSVCLI  hCli = (HAJCSSVCLI)GetProp(hDlg, TEXT("ClientHandle"));
481 :     UT          txt[MAX_TXTBOX_LEN];
482 :     AjcGetDlgItemStr(hDlg, IDC_TXT_SNDESC, txt, AJCTSIZE(txt));
483 :     AjcSsvSendChar(hCli, 0x1B);
484 :     AjcSsvSendText(hCli, txt, -1);
485 :     return TRUE;
486 : }
487 : //----- バイナリ送信ボタン ----------------------------------------------------------------------------//
488 : AJC_DLGPROC(Client, IDC_CMD_SNDBIN          )
489 : {
490 :     HAJCSSVCLI  hCli = (HAJCSSVCLI)GetProp(hDlg, TEXT("ClientHandle"));
491 :     UI  ix;
492 :     UTP p;
493 :     UT  txt[MAX_TXTBOX_LEN];
494 :     UB  bin[MAX_TXTBOX_LEN];
495 :     ix = 0;
496 :     AjcGetDlgItemStr(hDlg, IDC_TXT_SNDBIN, txt, AJCTSIZE(txt));
497 :     if (p = MAjcStrTok(txt, TEXT(" "))) {
498 :         do {
499 :             if (AjcHexStrToUB(p, 2, &bin[ix])) {
500 :                 ix++;
501 :             }
502 :         } while (p = MAjcStrTok(NULL, TEXT(" ")));
503 :     }
504 :     if (ix != 0) {
505 :         AjcSsvSendBinData(hCli, bin, ix);
506 :     }
507 :     return TRUE;
508 : }
509 : //----- パケット送信ボタン ----------------------------------------------------------------------------//
510 : AJC_DLGPROC(Client, IDC_CMD_SNDPKT          )
511 : {
512 :     HAJCSSVCLI  hCli = (HAJCSSVCLI)GetProp(hDlg, TEXT("ClientHandle"));
513 :     UI  ix;
514 :     UTP p;
515 :     UT  txt[MAX_TXTBOX_LEN];
516 :     UB  bin[MAX_TXTBOX_LEN];
517 :     ix = 0;
518 :     AjcGetDlgItemStr(hDlg, IDC_TXT_SNDPKT, txt, AJCTSIZE(txt));
519 :     if (p = MAjcStrTok(txt, TEXT(" "))) {
520 :         do {
521 :             if (AjcHexStrToUB(p, 2, &bin[ix])) {
522 :                 ix++;
523 :             }
524 :         } while (p = MAjcStrTok(NULL, TEXT(" ")));
525 :     }
```

```
526 :     if (ix != 0) {
527 :         AjcSsvSendPacket(hCli, bin, ix);
528 :     }
529 :     return TRUE;
530 : }
531 : //----- 切断ボタン ----------------------------------------------------------------------------------------------//
532 : AJC_DLGPROC(Client, IDC_CMD_DISC            )
533 : {
534 :     HAJCSSVCLI  hCli = (HAJCSSVCLI)GetProp(hDlg, TEXT("ClientHandle"));
535 :     AjcSsvDisconnect(hCli);
536 :     return TRUE;
537 : }
538 : //----- キャンセル ----------------------------------------------------------------------------------------------//
539 : AJC_DLGPROC(Client, IDCANCEL            )
540 : {
541 :     HAJCSSVCLI  hCli = (HAJCSSVCLI)GetProp(hDlg, TEXT("ClientHandle"));
542 :     //  クライアント切断
543 :     AjcSsvDisconnect(hCli);
544 :     //  ダイアログをクローズ
545 :     DestroyWindow(hDlg);
546 :     return TRUE;
547 : }
548 : //----- 全てクリアーボタン --------------------------------------------------------------------------------------//
549 : AJC_DLGPROC(Client, IDC_CMD_CLRALL  )
550 : {
551 :     if (HIWORD(wParam) == BN_CLICKED) {
552 :         AjcVthClear(GetDlgItem(hDlg, IDC_VTH_TXTCHUNK));
553 :         AjcVthClear(GetDlgItem(hDlg, IDC_VTH_BINCHUNK));
554 :         AjcVthClear(GetDlgItem(hDlg, IDC_VTH_TEXT    ));
555 :         AjcVthClear(GetDlgItem(hDlg, IDC_VTH_CTRL    ));
556 :         AjcVthClear(GetDlgItem(hDlg, IDC_VTH_ESC     ));
557 :         AjcVthClear(GetDlgItem(hDlg, IDC_VTH_PKT     ));
558 :         AjcVthClear(GetDlgItem(hDlg, IDC_VTH_NOPKT   ));
559 :     }
560 :     return TRUE;
561 : }
562 : //----- ＶＴＨ（テキスト）からの通知 ----------------------------------------------------------------------------//
563 : AJC_DLGPROC(Client, IDC_VTH_TEXT    )
564 : {
565 :     HAJCSSVCLI  hCli = (HAJCSSVCLI)GetProp(hDlg, TEXT("ClientHandle"));
566 :
567 :     if (HIWORD(wParam) == AJCVTHN_DROPFILE) {       //  ファイルドロップ
568 :         UI      i, nFile = (UI)lParam;
569 :         HANDLE  hFile;
570 :         UL      bytes;
571 :         UT      path[MAX_PATH];
572 :         UB      buf[2014];
573 :         HWND    hVth = GetDlgItem(hDlg, IDC_VTH_TEXT);
574 :
575 :         for (i = 0; i < nFile; i++) {
576 :             AjcVthGetDroppedFile(hVth, path);
577 :             if ((hFile = CreateFile(path, GENERIC_READ, FILE_SHARE_READ, NULL, OPEN_EXISTING, 0, NULL)) != INVALID_HANDLE_VALUE) {
578 :                 while (ReadFile(hFile, buf, sizeof buf, &bytes, NULL) && bytes != 0) {
579 :                     AjcSsvSendBinData(hCli, buf, bytes);
580 :                 }
581 :                 CloseHandle(hFile);
582 :             }
583 :         }
584 :     }
585 :     return TRUE;
586 : }
587 : //----------------------------------------------------------------------------------------------------------------//
588 : AJC_DLGMAP_DEF(Client)
589 :     AJC_DLGMAP_MSG(Client, WM_INITDIALOG    )
590 :     AJC_DLGMAP_MSG(Client, WM_DESTROY       )
591 :
592 :     AJC_DLGMAP_CMD(Client, IDC_CMD_SNDTEXT  )
593 :     AJC_DLGMAP_CMD(Client, IDC_CMD_SNDESC   )
594 :     AJC_DLGMAP_CMD(Client, IDC_CMD_SNDBIN   )
595 :     AJC_DLGMAP_CMD(Client, IDC_CMD_SNDPKT   )
596 :     AJC_DLGMAP_CMD(Client, IDC_CMD_DISC     )
597 :     AJC_DLGMAP_CMD(Client, IDC_CMD_CLRALL   )
598 :     AJC_DLGMAP_CMD(Client, IDCANCEL         )
599 :     AJC_DLGMAP_CMD(Client, IDC_VTH_TEXT     )
600 : AJC_DLGMAP_END
601 : //----------------------------------------------------------------------------------------------------------------//
602 : //  受信テキストエンコード設定                                                                                    //
603 : //----------------------------------------------------------------------------------------------------------------//
604 : static  VO  SetRxTextCode(HWND hDlg)
605 : {
```

```
606 :     if      (AjcGetDlgItemChk(hDlg, IDC_RBT_RXSJIS)) AjcSsvSetRxTextCode(hSsv, AJCSSV_TXT_SJIS);
607 :     else if (AjcGetDlgItemChk(hDlg, IDC_RBT_RXUTF8)) AjcSsvSetRxTextCode(hSsv, AJCSSV_TXT_UTF8);
608 :     else if (AjcGetDlgItemChk(hDlg, IDC_RBT_RXEUC )) AjcSsvSetRxTextCode(hSsv, AJCSSV_TXT_EUC );
609 :     else                                             AjcSsvSetRxTextCode(hSsv, AJCSSV_TXT_AUTO);
610 : }
611 : //-----------------------------------------------------------------------------------------------------------------//
612 : //  送信テキストエンコード設定                                                                                     //
613 : //-----------------------------------------------------------------------------------------------------------------//
614 : static  VO  SetTxTextCode(HWND hDlg)
615 : {
616 :     if      (AjcGetDlgItemChk(hDlg, IDC_RBT_TXSJIS)) AjcSsvSetTxTextCode(hSsv, AJCSSV_TXT_SJIS);
617 :     else if (AjcGetDlgItemChk(hDlg, IDC_RBT_TXUTF8)) AjcSsvSetTxTextCode(hSsv, AJCSSV_TXT_UTF8);
618 :     else if (AjcGetDlgItemChk(hDlg, IDC_RBT_TXEUC )) AjcSsvSetTxTextCode(hSsv, AJCSSV_TXT_EUC );
619 :     else                                             AjcSsvSetTxTextCode(hSsv, AJCSSV_TXT_AUTO);
620 : }
```

14.4.2. SW_SockServer2（エコーサーバ）

このサンプルプログラムは、エコーサーバ（受信データをそのまま返信）処理を実行します。
サンプルプログラム（Sil_SockClient1/3 や Sil_SerialComPort1/3）と対向して通信できます。




```
 1 : //
 2 : //  SW_SockServer2.c
 3 : //
 4 : #define AJCSERIALCOMPORT_H_
 5 : #include    <AjrCstXX.h>
 6 : #include    "resource.h"
 7 :
 8 : #define MAX_CLIENT  10
 9 : #define WM_SSVEVENT (WM_APP + 100)
10 :
11 : //-------------------------------------------------------------------------------------------------------------------//
12 : //  ワーク                                                                                                           //
13 : //-------------------------------------------------------------------------------------------------------------------//
14 : static  HINSTANCE       hInst;                    //  ＤＬＬインスタンスハンドル
15 : static  HWND            hDlgMain;                 //  ダイアログボックスハンドル
16 : static  HAJCSSV         hSsv;
17 : static  ULL             total[MAX_CLIENT];
18 :
19 : //-------------------------------------------------------------------------------------------------------------------//
20 : //  内部サブ関数                                                                                                     //
21 : //-------------------------------------------------------------------------------------------------------------------//
22 : AJC_DLGPROC_DEF(Main);
23 : static UI GetFreeNumber(HWND hDlg);
24 :
25 : //===================================================================================================================//
26 : //                                                                                                                   //
27 : //  Ｗｉｎ Ｍａｉｎ                                                                                                  //
28 : //                                                                                                                   //
29 : //===================================================================================================================//
30 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
31 : {
32 :     MSG     msg;
33 :
34 :     //----- メイン・ダイアログオープン ----------------//
35 :     hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_MAIN), NULL, AJC_DLGPROC_NAME(Main));
36 :     AjcMoveWndIntoMonitor(hDlgMain);
37 :     //----- ダイアログ表示 ----------------------------//
38 :     ShowWindow(hDlgMain, SW_SHOW);
39 :     //----- メッセージループ --------------------------//
40 :     while (GetMessage(&msg, NULL, 0, 0)) {
41 :         do {
42 :             if (IsDialogMessage(hDlgMain, &msg)) break;
43 :             TranslateMessage(&msg);
44 :             DispatchMessage (&msg);
45 :         } while (0);
46 :     }
47 :     return (int)msg.wParam ;
48 : }
49 : //===================================================================================================================//
50 : //                                                                                                                   //
51 : //  メイン ダイアログ・プロシージャ                                                                                  //
```

```
52 : //                                                                                                                  //
53 : //==================================================================================================================//
54 : //----- ダイアログ初期化 -------------------------------------------------------------------------------------------//
55 : AJC_DLGPROC(Main, WM_INITDIALOG     )
56 : {
57 :     UL      lHost;
58 :     UT      szHost[128];
59 :
60 :     hDlgMain    = hDlg;
61 :
62 :     //  ホスト名表示
63 :     lHost = AJCTSIZE(szHost);
64 :     GetComputerName(szHost, &lHost);
65 :     AjcSetDlgItemStr(hDlg, IDC_LBL_MYNAME, szHost);
66 :     //  ソケットサーバ インスタンス生成
67 :     hSsv = AjcSsvCreate();
68 :     //  ウインド位置ロード
69 :     AjcLoadWndPos(hDlg, TEXT("WndPos"));
70 :     //  サーバ開始
71 :     AjcSsvStart(hSsv, TEXT("14238"), AF_INET, MAX_CLIENT, hDlg, WM_SSVEVENT);
72 :
73 :     return TRUE;
74 : }
75 : //----- ウインド破棄 -----------------------------------------------------------------------------------------------//
76 : AJC_DLGPROC(Main, WM_DESTROY        )
77 : {
78 :     //  ソケットサーバ インスタンス消去
79 :     AjcSsvDelete(hSsv);
80 :     //  ウインド位置セーブ
81 :     AjcSaveWndPos(hDlg, TEXT("WndPos"));
82 :     //  サーバ停止
83 :     AjcSsvStop(hSsv, 10000);
84 :
85 :     PostQuitMessage(0);
86 :     return TRUE;
87 : }
88 : //----- ソケット(TCP/IP)イベント -----------------------------------------------------------------------------------//
89 : AJC_DLGPROC(Main, WM_SSVEVENT       )
90 : {
91 :     union {UTP pTxt; BCP pBcp; UBP pBin; VOP pVop;} u;
92 :     UI              len;
93 :     UI              param;
94 :     HAJCSSVCLI      hCli = NULL;
95 :     UX              ux;
96 :     UI              i;
97 :     UT              szIpAddr[256];
98 :
99 :     //  イベントデータ取得
100 :    AjcSsvGetEventData(hSsv, lParam, &u.pVop, &len, &param);
101 :    //  クライアント情報を取得し、クライアントのＩＰアドレスを設定
102 :    if (AjcSsvGetClient(hSsv, lParam, &hCli)) {
103 :        AjcSsvGetIpAddrStr(hCli, szIpAddr, AJCTSIZE(szIpAddr));
104 :    }
105 :
106 :    switch (wParam) {
107 :        case AJCSSV_EV_CONNECT:                             //●接続通知
108 :            if ((i = GetFreeNumber(hDlg)) < MAX_CLIENT) {
109 :                //  バイト数カウンタクリアー
110 :                total[i] = 0;
111 :                //  クライアントに空き番号を関連付ける
112 :                AjcSsvSetClientData(hCli, i);
113 :                //  クライアント IP アドレス表示
114 :                AjcSsvGetIpAddrStr(hCli, szIpAddr, AJCTSIZE(szIpAddr));
115 :                AjcSetDlgItemStr (hDlg, IDC_LBL_ADDR0 + i, szIpAddr);
116 :                AjcSetDlgItemUInt(hDlg, IDC_LBL_CNT0  + i, 0);
117 :            }
118 :            break;
119 :
120 :        case AJCSSV_EV_DISCONNECT:                          //●切断通知
121 :            //  クライアントに関連付けられた番号取得
122 :            AjcSsvGetClientData(hCli, &ux); i = (UI)ux;
123 :            //  クライアント表示クリアー
124 :            AjcEnableDlgItem(hDlg, IDC_LBL_ADDR0 + i, FALSE);
125 :            AjcEnableDlgItem(hDlg, IDC_LBL_CNT0  + i, FALSE);
126 :            break;
127 :
128 :        case AJCSSV_EV_RXCHUNK:                             //●チャンクデータ受信通知
129 :            //  受信データを返送
130 :            AjcSsvSendBinData(hCli, u.pBin, len);
131 :            //  クライアントに関連付けられた番号取得
```

```
132 :             AjcSsvGetClientData(hCli, &ux); i = (UI)ux;
133 :             //  バイトカウンタ更新
134 :             total[i] += len;
135 :             AjcSepDlgItemUI64(hDlg, IDC_LBL_CNT0 + i, total[i]);
136 :             break;
137 :     }
138 :
139 :     //  イベントデータ開放
140 :     AjcSsvRelEventData(hSsv, lParam);
141 :
142 :     return TRUE;
143 : }
144 : //----- キャンセル ----------------------------------------------------------------------------------------------//
145 : AJC_DLGPROC(Main, IDCANCEL          )
146 : {
147 :     DestroyWindow(hDlg);
148 :     return TRUE;
149 : }
150 : //-------------------------------------------------------------------------------------------------------------//
151 : AJC_DLGMAP_DEF(Main)
152 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG     )
153 :     AJC_DLGMAP_MSG(Main, WM_DESTROY        )
154 :     AJC_DLGMAP_MSG(Main, WM_SSVEVENT       )
155 :     AJC_DLGMAP_CMD(Main, IDCANCEL          )
156 : AJC_DLGMAP_END
157 : //-------------------------------------------------------------------------------------------------------------//
158 : //  空番号取得                                                                                                 //
159 : //-------------------------------------------------------------------------------------------------------------//
160 : static UI GetFreeNumber(HWND hDlg)
161 : {
162 :     static UI ix = 0;
163 :     UI      rc = 0;
164 :     UI      i;
165 :
166 :     for (i = 0; i < MAX_CLIENT; i++, ix = ((ix + 1) % MAX_CLIENT)) {
167 :         if (MAjcGetWindowLong(GetDlgItem(hDlg, IDC_LBL_ADDR0 + ix), GWL_STYLE) & WS_DISABLED) {
168 :             AjcEnableDlgItem(hDlg, IDC_LBL_ADDR0 + ix, TRUE);
169 :             AjcEnableDlgItem(hDlg, IDC_LBL_CNT0  + ix, TRUE);
170 :             rc = ix;
171 :             ix = ((ix + 1) % MAX_CLIENT);
172 :             break;
173 :         }
174 :     }
175 :     return rc;
176 : }
```

14.4.3. SW_SockServer2C（エコーサーバ，コンソールアプリ）

このサンプルプログラムは、SW_SockServer2 と同様のエコーサーバ（受信データをそのまま返信）処理をコンソールアプリで実行します。




```
 1 : //
 2 : //  SW_SockServer2C.c
 3 : //
 4 : //  コマンド形式：  SW_SockServer2C [/P<ポート番号>]  [/C<最大クライアント数>] [/IPV6]
 5 : //
 6 : #define AJCSERIALCOMPORT_H_
 7 : #define AJCSOCKCLIENT_H_
 8 : #include    <AjrCstXX.h>
 9 : #include    <conio.h>
10 :
11 : //-------------------------------------------------------------------------------------------------------------------//
12 : //  クライアント情報
13 : #define MAX_CLIENT  10
14 : typedef struct {
15 :     UT      ipaddr[256];
16 :     UI      bytes;
17 : } ClientsInfo, *PClientsInfo;
18 :
19 : static  ClientsInfo CliTbl[MAX_CLIENT] = {0};
20 :
21 : //-------------------------------------------------------------------------------------------------------------------//
22 : //  ワーク
23 : static  HAJCSSV     hSsv;
24 :
25 : //-------------------------------------------------------------------------------------------------------------------//
26 : //  コンソール強制終了ハンドラ
27 : static BOOL CALLBACK cbConApExit(DWORD CtrlType)
28 : {
29 :     if (hSsv != NULL) {
30 :         AjcSsvDelete(hSsv);
31 :         hSsv = NULL;
32 :     }
33 :     return FALSE;   //  FALSE : 次のハンドラをコール
34 : }
35 : //-------------------------------------------------------------------------------------------------------------------//
36 : //  主制御
37 : int  AjcMain(int argc, UTP argv[])
38 : {
39 :     BOOL        fEnd = FALSE;
40 :     WPARAM      wParam;
41 :     LPARAM      lParam;
42 :     union {UTP pTxt; UBP pBin; VOP pVoid;} u;
43 :     UI          len;
44 :     UI          param;
45 :     UTP         pPort = TEXT("14238");
46 :     UI          MaxCli = SOMAXCONN;
47 :     UI          Family = AF_INET;
48 :     int         CurBaseX, CurBaseY;
49 :     UI          bytes;
50 :     UT          MyHostName[MAX_COMPUTERNAME_LENGTH + 1] = {0};
51 :
52 :     AjcSetStdoutMode();
```

```
 53 :
 54 :     //  コンソール強制終了ハンドラ設定
 55 :     SetConsoleCtrlHandler(cbConApExit, TRUE);
 56 :
 57 :     //  初期表示
 58 :     bytes = (AJCTSIZE(MyHostName));
 59 :     GetComputerName(MyHostName, &bytes);
 60 :     AjcPrintF(TEXT("¥n"));
 61 :     AjcPrintF(TEXT(" エコーサーバ ¥n"));
 62 :     AjcPrintF(TEXT("¥n"));
 63 :     AjcPrintF(TEXT("   ホスト名   :%s¥n"), MyHostName);
 64 :     AjcPrintF(TEXT("   ポート番号:14238¥n"));
 65 :     AjcPrintF(TEXT("¥n"));
 66 :     AjcPrintF(TEXT(" いずれかのキーを押すと、プログラムを終了します。¥n"));
 67 :     AjcPrintF(TEXT("¥n"));
 68 :     AjcPrintF(TEXT("      No. IP-Address      Bytes¥n"));
 69 :     AjcPrintF(TEXT("¥n"));
 70 :     AjcPrintF(TEXT("       1. -               -¥n"));
 71 :     AjcPrintF(TEXT("       2. -               -¥n"));
 72 :     AjcPrintF(TEXT("       3. -               -¥n"));
 73 :     AjcPrintF(TEXT("       4. -               -¥n"));
 74 :     AjcPrintF(TEXT("       5. -               -¥n"));
 75 :     AjcPrintF(TEXT("       6. -               -¥n"));
 76 :     AjcPrintF(TEXT("       7. -               -¥n"));
 77 :     AjcPrintF(TEXT("       8. -               -¥n"));
 78 :     AjcPrintF(TEXT("       9. -               -¥n"));
 79 :     AjcPrintF(TEXT("      10. -               -¥n"));
 80 :     //  カーソル位置取得
 81 :     AjcGetConsoleCursor(&CurBaseX, &CurBaseY);
 82 :     CurBaseY -= 10;
 83 :     //  ソケットサーバ インスタンス生成
 84 :     hSsv = AjcSsvCreate();
 85 :     //  送受信テキスト文字コード設定
 86 :     AjcSsvSetRxTextCode(hSsv, AJCSSV_TXT_AUTO);
 87 :     AjcSsvSetTxTextCode(hSsv, AJCSSV_TXT_AUTO);
 88 :     //  バイナリチャンクモード設定
 89 :     AjcSsvSetChunkMode(hSsv, AJCSSV_CM_BIN);
 90 :     //  対象とするイベントを設定
 91 :     AjcSsvSetEvtMask(hSsv, AJCSSV_EV_CONNECT | AJCSSV_EV_DISCONNECT | AJCSSV_EV_RXCHUNK | AJCSSV_EV_STOP);
 92 :     //  サーバスタート
 93 :     AjcSsvStart(hSsv, TEXT("14238"), AF_INET, 10, NULL, 0);
 94 :
 95 :     while (!fEnd) {
 96 :         //  いずれかのキー押下で終了
 97 :         if (_kbhit()) {
 98 :             _getch();
 99 :             AjcSsvStop(hSsv, 10000);
100 :             fEnd = TRUE;
101 :         }
102 :
103 :         if (AjcSsvWaitEvent(hSsv, &wParam, &lParam, 200)) { //  イベント待ち(200ms), イベントあり？
104 :             HAJCSSVCLI      hCli = NULL;
105 :             UT              szIpAddr[256] = {0};
106 :             //  イベントデータ取得
107 :             AjcSsvGetEventData(hSsv, lParam, &u.pVoid, &len, &param);
108 :             //  クライアント情報を取得し、クライアントのＩＰアドレスを設定
109 :             if (AjcSsvGetClient(hSsv, lParam, &hCli)) {
110 :                 AjcSsvGetIpAddrStr(hCli, szIpAddr, AJCTSIZE(szIpAddr));
111 :             }
112 :             //  各イベント処理
113 :             //  ●接続通知
114 :             if ((wParam & AJCSSV_EV_CONNECT) != 0) {
115 :                 //  クライアントインデクス取得
116 :                 UI      ix = AjcSsvGetIndex(hCli);
117 :                 //  テーブルエントリ設定
118 :                 MAjcStrCpy(CliTbl[ix].ipaddr, AJCTSIZE(CliTbl[ix].ipaddr), szIpAddr);
119 :                 CliTbl[ix].bytes  = 0;
120 :                 //  クライアント情報初期表示
121 :                 AjcSetConsoleCursor(10, CurBaseY + ix); AjcPrintF(CliTbl[ix].ipaddr);
122 :                 AjcSetConsoleCursor(26, CurBaseY + ix); AjcPrintF(TEXT("%d"), CliTbl[ix].bytes);
123 :             }
124 :             //  ●切断通知
125 :             if ((wParam & AJCSSV_EV_DISCONNECT) != 0) {
126 :                 //  クライアントインデクス取得
127 :                 UI  ix = AjcSsvGetIndex(hCli);;
128 :                 //  クライアント情報を消す
129 :                 AjcSetConsoleCursor(10, CurBaseY + (UI)ix); AjcPrintF(TEXT("-               "));
130 :                 AjcSetConsoleCursor(26, CurBaseY + (UI)ix); AjcPrintF(TEXT("-               "));
131 :             }
132 :             //  ●バイナリチャンク受信通知
```

```
133 :             if ((wParam & AJCSSV_EV_RXCHUNK) != 0) {
134 :                 //  クライアントインデクス取得
135 :                 UI  ix = AjcSsvGetIndex(hCli);;
136 :                 //  受信バイト数を表示
137 :                 CliTbl[ix].bytes += len;
138 :                 AjcSetConsoleCursor(26, (int)(CurBaseY + ix)); AjcPrintF(TEXT("%u"), CliTbl[ix].bytes);
139 :                 //  受信データを返信
140 :                 AjcSsvSendBinData(hCli, u.pBin, len);
141 :             }
142 :             //  ●サーバ終了通知
143 :             if ((wParam & AJCSSV_EV_STOP) != 0) {
144 :                 fEnd = TRUE;
145 :             }
146 :             //  イベントデータ開放
147 :             AjcSsvRelEventData(hSsv, lParam);
148 :         }
149 :     }
150 :     //  ソケットサーバ インスタンス消去
151 :     if (hSsv != NULL) {
152 :         AjcSsvDelete(hSsv);
153 :         hSsv = NULL;
154 :     }
155 :
156 :     AjcSetConsoleCursor(0, CurBaseY + 10);
157 :     AjcPrintF(TEXT("Hit enter key !"));
158 :     getchar();
159 :     return 0;
160 : }
```

15. ソケット(TCP/IP)クライアント機能

ソケット(TCP/IP)のサーバ側通信制御モジュールです。
サーバと接続して通信することができます。

ソケット(TCP/IP)サーバでは、定義名やＡＰ名がシリアル通信やソケット(TCP/IP)クライアント機能と似通っています。
「#include <AjrCstXX.h>」の前に、以下のように「#define」を定義すると、他のの定義名やＡＰ名を抑止できます。
（誤って、シリアル通信やソケット(TCP/IP)クライアント機能の定義名やＡＰ名を使用するとコンパイルエラーとなります）

```
#define AJCSERIALCOMPORT_H_          // シリアル通信を無効化
#define AJCSOCKCLIENT_H_             // ソケット(TCP/IP)クライアントを無効化
#include <AjrCstXX.h>
```

15.1. 機能概要

15.2. イベントの通知方法

イベントの通知方法は、以下の２つから選択できます。

・ユーザのウインドへ、ウインドメッセージとして通知する
・ユーザが、その場でイベントの発生を待ち受ける

ユーザのウインドへ、ウインドメッセージとして通知する場合は、AjcSctSetMode()の hWndNtc 引数でウインドハンドルを、WndMsgNtc 引数でウインド・メッセージコードを指定します。

ウインドプロシージャの wParam にはイベントコードが、lParam にはイベント情報が設定されます。
イベントデータを取得するには、lParam を引数として、AjcSctGetEventData()を実行します。

ウインドメッセージとして通知する場合のプログラムコードスタイルは、およそ以下のようになります。

```
#define  WM_SCTEVENT  (WM_APP + 100)

HAJCSCT  hSct;

//----- ダイアログ初期化 ----------------------------------------------------------------------//
AJC_DLGPROC(Main, WM_INITDIALOG )
{
    // インスタンス生成
    hSct = AjcSctCreate();
    // 送受信テキスト文字コード設定
    AjcSctSetRxTextCode(hSct, AJCSCT_TXT_AUTO);
    AjcSctSetTxTextCode(hSct, AJCSCT_TXT_AUTO);
    // バイナリチャンクモード設定 (AJCSCT_CM_TEXT:テキストチャンク, AJCSCT_CM_BOTH:両方)
    AjcSctSetChunkMode(hSct, AJCSCT_CM_BIN);
    // 使用するイベント指定 (デフォルトでは、全イベントが有効であるため、特に指定しなくてもよい)
    AjcSctSetEvtMask(hSct, AJCSCT_EV_RXCHUNK | AJCSCT_EV_DISCONNECT | AJCSCT_EV_TXERR・・・・・);
    // サーバと接続
    AjcSctConnect(hSct, TEXT("ServerName"), TEXT("14238"), AF_INET,  hDlg, WM_SCTEVENT);
    ・・・・
    return TRUE;
}
```

つづく

つづき

```
//-----イベント通知 ----------------------------------------------------------------------------//
AJC_DLGPROC(Main, WM_SCTEVENT )
{
    union {UBP  pByte; UWP pWord; UTP pTxt; VOP pVoid;} u;
    UI            len, param;

    // イベントデータ取得
    AjcSctGetEventData(hSct, lParam, &u.pVoid, &len, &param);
    // イベント処理
    switch (wParam) {
        //●チャンクデータ受信通知
        case AJCSCT_EV_RXCHUNK:
            // 受信データをループバック送信 (任意)
            AjcSctSendText(hClient, u.pTxt, lTxt);
            break;

        // ●回線切断通知
        case AJCSCT_EV_DISCONNECT:
             //  ウインドクローズ (任意)
            DestroyWindow(hDlg);
            break;

        //●送信エラー通知
        case AJCSCT_EV_TXERR:
            // 送信エラーが発生したら、回線切断 (任意)
            AjcSctDisconnect(hSct);
            break;

        ・・・その他のイベント処理・・・

    }
    // イベントデータ開放
    AjcSctRelEventData(hSct, lParam);
    return TRUE;
}
//----- ウインド破棄 ----------------------------------------------------------------------------//
AJC_DLGPROC(Main, WM_DESTROY )
{
    //  インスタンス消去
    AjcSctDelete(hSct);                                          // ソケットサーバ インスタンス消去
    return TRUE;
}
//----- キャンセル-------------------------------------------------------------------------------//
AJC_DLGPROC(Main, IDCANCEL )
{
   // キャンセルボタンでプログラムを終了する (任意)
    DestroyWindow(hDlg);
    return TRUE;
}

・・・・・その他のメッセージ処理 ・・・・・

//-----------------------------------------------------------------------------------------------//
AJC_DLGMAP_DEF(Main)
    AJC_DLGMAP_MSG(Main, WM_INITDIALOG    )
    AJC_DLGMAP_MSG(Main, WM_DESTROY       )
    AJC_DLGMAP_MSG(Main, WM_SCTEVENT      )
    AJC_DLGMAP_MSG(Main, WM_DESTROY       )
    AJC_DLGMAP_CMD(Main, IDCANCEL         )
    ・・・・・
AJC_DLGMAP_END
```

ユーザが、その場でイベントの発生を待ち受ける場合は、AjcSctSetMode()の hWndNtc 引数に NULL を、WndMsgNtc 引数に 0 を指定し、AjcSctWaitEvent()によりイベントの発生を待ちます。

AjcSctWaitEvent()を実行すると、イベントが発生している場合は、wParam にはイベントコードが、lParam にはイベント情報が設定され、TRUE を返します。イベントが発生していない場合は FALSE を返します。

イベントデータを取得するには、lParam を引数として、AjcSctGetEventData()を実行します。

イベントの発生を待ち受ける場合のプログラムコードスタイルは、およそ以下のようになります。

```
int  AjcMain(int argc, UTP argv[])
{
    BOOL        fEnd = FALSE;
    HAJCSCT     hSct;
    WPARAM      wParam;
    LPARAM      lParam;
    union {UBP  pByte; UWP pWord; UTP pTxt; VOP pVoid;} u;
    UI          len;
    UI          param;
    HAJCSCTCLI  hClient;

    // インスタンス生成
    hSct = AjcSctCreate();
    // 送受信テキスト文字コード設定
    AjcSctSetRxTextCode(hSct, AJCSCT_TXT_AUTO);
    AjcSctSetTxTextCode(hSct, AJCSCT_TXT_AUTO);
    // バイナリチャンクモード設定 (AJCSCT_CM_TEXT:テキストチャンク, AJCSCT_CM_BOTH:両方)
    AjcSctSetChunkMode(hSct, AJCSCT_CM_BIN);
    // 使用するイベント指定 (デフォルトでは、全イベントが有効であるため、特に指定しなくてもよい)
    AjcSctSetEvtMask(hSct, AJCSCT_EV_RXCHUNK | AJCSCT_EV_SERVEND | AJCSCT_EV_ERR・・・・・);
    // サーバと接続
    AjcSctConnect(hSct, TEXT("ServerName"), TEXT("14238"), AF_INET,  NULL, 0);
    while (!fEnd) {
        //  キー ファンクション (キー押下で回線を切断する (任意))
        if (kbhit()) {
           // キー押下で回線を切断する
            AjcSctDisconnect(hSct, 10000);
        }
        //  イベント処理
        if (AjcSctWaitEvent(hSct, &wParam, &lParam, 200)) {
            HAJCSCTCLI      hClient= NULL;
            //  イベントデータ取得
            AjcSctGetEventData(hSct, lParam, &u.pVoid, &len, &param);
            //  クライアント情報を取得 (取得できないイベントもあるので注意が必要 (後述のイベント一覧参照))
            AjcSctGetClient (hSct, lParam, &hClient);
            //  イベント処理
            switch (wParam) {
                //  ●チャンクデータ受信通知
                case AJCSCT_EV_RXCHUNK:
                    //  受信データをループバック送信 (任意)
                    AjcSctSendText(hClient, pTxt, lTxt);
                    break;

                //  ●回線切断通知
                case AJCSCT_EV_DISCONNECT:
                    //  ループを抜けてプログラムを終了する
                    fEnd = TRUE;
                    break;

                //●送信エラー通知
                case AJCSCT_EV_TXERR:
                    // 送信エラーが発生したら、クライアント切断 (任意)
                    AjcSctDisconnect(hClient);
                    break;

                ・・・・・その他のイベント処理・・・・・

            }
            //  イベントデータ開放
            AjcSctRelEventData(hSct, lParam);
        }
    }
    //  クライアント機能 インスタンス消去
    AjcSctDelete(hSct);

    return 0;
}
```

ユーザがイベントの発生を待ち受ける場合、相手局に何らかのデータを要求し、その場で応答データの着信を待つことも可能です。

```
AjcSctSendText(hSct, "データ要求コマンド", -1);                  //  相手局へデータ要求コマンド送信
if (AjcSctWaitEvent(hSct, &wParam, &lParam, 200)) {          //  イベント待ち(200ms), イベントあり？
    AjcSctGetEventData(hSct, lParam, &pDat, &len, &param);   //    イベントデータ取得
    ・・・・応答データ受信等のイベント処理・・・・
    AjcSctRelEventData(hSct, lParam);                        //    イベントデータ開放
}
```

15.3. イベント一覧

イベントの一覧を以下に示します。

#	イベントコード (wParam)	内容	イベントデータ（AjcSctGetEventData()で取得する情報）		
			pDat	lDat	param
1	AJCSCT_EV_RXNOPKT	パケット外テキスト通知	受信データのアドレス	受信データの バイト数／文字数	0：バイナリデータ 1：バイト文字列 2：ワイド文字列 (UNICODE)
2	AJCSCT_EV_RXTEXT（※2）	テキスト受信通知			
3	AJCSCT_EV_RXESC	ＥＳＣコード受信通知			
4	AJCSCT_EV_RXPKT	パケットデータ受信通知			
5	AJCSCT_EV_RXCTRL	制御コード受信通知			
6	AJCSCT_EV_RXCHUNK（※1）	チャンクデータ受信通知			
7	AJCSCT_EV_TXEMPTY	送信完了通知	NULL	0	0
8	AJCSCT_EV_CONNECT	クライアント接続通知	NULL	0	0
9	AJCSCT_EV_DISCONNECT	クライアント切断通知	NULL	0	0
10	AJCSCT_EV_CNFAIL	接続失敗通知	NULL	0	エラーコード GetLastError()の値
11	AJCSCT_EV_RXERR	受信エラー通知	NULL	0	0：WSARecv() 1：GetOverlappedResult()
12	AJCSCT_EV_TXERR	送信エラー通知	NULL	0	0：WSASend() 1：GetOverlappedResult()
13	AJCSCT_EV_ERR	エラー発生通知	NULL	0	エラー要因（※3）
14	AJCSCT_EV_SSEP	#1～#5　の合成値			
15	AJCSCT_EV_COMM	#6～#8　の合成値			
16	AJCSCT_EV_GENERAL	#9～#11　の合成値			
17	AJCSCT_EV_ERRS	#12～#14 の合成値			
18	AJCSCT_EV_ALL	全イベント			

※1： テキストチャンクの場合、AjcSctSetRxTextCode()により設定された文字コードからシフト JIS/UNICODE に変換したテキストを通知します。

※2： AjcSctSetRxTextCode()により設定された文字コードからシフト JIS/UNICODE に変換したテキストを通知します。

※3：エラー要因（以下のいずれか）

#	通信エラーコード	内容
1	AJCSCT_ERR_CREEVT	イベントオブジェクト生成失敗
2	AJCSCT_ERR_SOCKET	接続要求待機用ソケット生成失敗(socket)
3	AJCSCT_ERR_CONNECT	CONNECT 失敗(connect)
4	AJCSCT_ERR_ADDRINFO	アドレス情報取得失敗（getaddrinfo）
5	AJCSCT_ERR_SOCKOPT	ソケットオプション設定失敗(setsockopt)
6	AJCSCT_ERR_SUBTHREAD	接続待機スレッド開始失敗
7	AJCSCT_ERR_CRESSEP	ストリーム分離インスタンス生成失敗
8	AJCSCT_ERR_CREMBXTXD	送信データ用メールボックス生成失敗
9	AJCSCT_ERR_TIMEOUT	通信スレッド終了タイムアウト
10	AJCSCT_ERR_THREADBUSY	接続，通信スレッド実行中
11	AJCSCT_ERR_CREATEEVENT	WSACreateEvent 失敗
12	AJCSCT_ERR_EVENTSELECT	WSAEventSelect 失敗
13	AJCSCT_ERR_ENUMNETWORKEVENTS	WSAEnumNetworkEvents 失敗
14	AJCSCT_ERR_SOCKERRORCODE	Bit31=1 の場合、下位ビットがソケットエラーコード (MSDN 「Windows のソケットエラーコード」参照)

デフォルトでは（AjcSctCreate()実行直後では）全てのイベントが許可状態となっています。
許可するイベントを選択する場合は、AjcSCtSetEvtMask()により、許可するイベントを指定してください。

15.4. サポートＡＰＩ

ソケット(TCP/IP)クライアント機能のＡＰＩ一覧を以下に示します。

#	関数名	内容	スレッドセーフ
1	AjcSctCreate	インスタンス生成	×
2	AjcSctDelete	インスタンス消去	×
3	AjcSctConnect	回線接続	○
4	AjcSctDisconnect[Ex]	回線切断	○
5	AjcSctGetState	回線状態取得	○
6	AjcSct{Set/Get}ChunkMode	チャンクデータの受信モード（テキスト／バイナリ）設定／取得	○
7	AjcSct{Set/Get}EvtMask	イベントマスク設定／取得	○
8	AjcSct{Set/Get}RxTextCode	受信テキストの文字コード設定／取得	○
9	AjcSct{Set/Get}TxTextCode	送信テキストの文字コード設定／取得	○
10	AjcSctGetActual{RX/TX}TextCode	実際の送受信文字コード種別取得	○
11	AjcSctWaitEvent	イベント発生待ち	○
12	AjcSctGetEventData	イベントデータ取得	○
13	AjcSctRelEventData	イベントデータ開放	○
14	AjcSctWaitEmptyEvent	イベントが空になるまで待つ	
15	AjcSct{Set/Get}PktCtrlCode	パケットフレームを認識する為の制御コード設定/取得	○
16	AjcSct{Set/Get}PktTimeout	パケットフレーム受信タイムアウト値設定／取得	○
17	AjcSctSendChar	１文字送信	○
18	AjcSctSendText AjcSctSendTextF	テキストデータ送信	○
19	AjcSctSendBinData	バイナリデータ送信	○
20	AjcSctSendPacket	パケットデータ送信	○
21	AjcSctPurgeRecvData	全受信済データ破棄	○
22	AjcSctPurgeSendData	全送信待ちデータ破棄	○
23	AjcSctPurgeAllData	全送受信データ破棄	○

15.4.1. インスタンス生成（AjcSctCreate）

形　式 ： HAJCSCT　AjcSctCreate(VO);

引　数 ： なし

説　明 ： ソケット(TCP/IP)クライアント機能のインスタンスを生成します。

戻り値 ： ≠NULL - 成功（インスタンスハンドル）
＝NULL - 失敗

15.4.2. インスタンス消去（AjcSctDelete）

形　式 ： BOOL　AjcSctDelete(HAJCSCT hSct)

引　数 ： hSct　　- インスタンスハンドル

説　明 ： 全てのリソースを開放し、ソケット(TCP/IP)クライアント機能のインスタンスを消去します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

15.4.3. 回線接続（AjcSctConnect）

形　式 ： BOOL　AjcSctConnect (HAJCSCT hSct, C_UTP pServ, C_UTP pPort, int AddressFamily, HWND hWndNtc, UI WndMsgNtc);

引　数 ：

hSct	- インスタンスハンドル
pServ	- サーバ名／ＩＰアドレスへのポインタ
pPort	- ポート名／ポート番号へのポインタ
AddressFamily	- アドレスファミリ(AF_INET / AF_INET6)
hWndNtc	- イベント通知用ウインドハンドル（AjcSctWaitEvent()でイベントを待つ場合はNULL）
WndMsgNtc	- イベント通知用メッセージ　　（AjcSctWaitEvent()でイベントを待つ場合は０）

説　明 ： サーバとの回線を接続します。
イベントをウインドメッセージで受け取る場合は、hWndNtc に通知するウインドのハンドル，WndMsgNtc にウインドメッセージコード(WM_USER+100 以降，WM_APP+500 以降か、RegisterWindowMessage()で取得したコード）を指定します。
このウインドメッセージが通知された場合、wParam にイベントコードが,lParam にイベントデータ取得情報が設定されます。実際に、イベントデータを取得するには、lParam を指定して、AjcSctGetEventData()を実行します。

イベントを AjcSctWaitEvent()で待つ場合は、hWndNtc=NULL，WndMsgNtc=0 とします。

戻り値 ： TRUE　- 成功
FALSE - 失敗

15.4.4. 回線切断（AjcSctDisconnect）

形　式 ： BOOL　AjcSctDisconnec (HAJCSCT hSct);

引　数 ： hSct　　- インスタンスハンドル

説　明 ： サーバとの回線を切断します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

15.4.5. 回線状態取得（AjcSctGetState）

形　式 ： AJCCST_STATE AjcSctGetState (HAJCSCT hSct);

引　数 ： hSct　　- インスタンスハンドル

説　明 ： 回線の状態を取得します。

戻り値 ： 回線の状態　　AJCSCT_DISCONNECT　--- 切断状態
AJCSCT_CONNECTING　--- 接続中
AJCSCT_CONNECT　------ 接続状態
AJCSCT_DISCONNECTING - 切断中

15.4.6. チャンクデータの通知モード設定／取得（AjcSct{Set/Get}ChunkMode）

形　式 ： BOOL　AjcSctSetChunkMode (HAJCSCT hSct, AJCSCT_CHUNKMODE ChunkMode); -- チャンクデータ通知モードの設定
AJCSCT_CHUNKMODE　AjcSctGetChunkMode ((HAJCSCT hSct);　---------- チャンクデータ通知モードの取得

引　数 ： hSct　　- インスタンスハンドル
ChunkMode　- 設定するチャンクデータの通知モード

説　明 ： チャンクデータの通知モードを設定／取得します。ChunkMode には、以下のいずれかを指定します。

ChunkMode	内容
AJCSCT_CM_NONE	受信チャンクデータを通知しません
AJCSCT_CM_BIN	受信チャンクデータをバイナリデータとして通知します（デフォルト）
AJCSCT_CM_TEXT	受信チャンクデータをテキストデータとして通知します
AJCSCT_CM_BOTH	受信チャンクデータをバイナリデータ＆テキストデータとして通知します (AJCSCT_EV_RXCHUNK イベントで、同じ受信チャンクデータが param=0(バイナリデータ)と、 param=1/2(テキストデータ)として２回通知されます)

戻り値 ： 設定時：TRUE　- 成功　　取得時：チャンクデータの受信モード
FALSE - 失敗

15.4.7. イベントマスク設定／取得（AjcSct{Set/Get}EvtMask）

形　式 ： BOOL　AjcSctSetEvtMask (HAJCSCT hSct, UI Mask); -- イベントマスク設定
UI　　AjcSctGetEvtMask (HAJCSCT hSct); ---------- イベントマスク取得

引　数 ： hSct　　- インスタンスハンドル
Mask　　- 設定するイベントマスク（イベントコード(AJCSCT_EV_*XXXXX*)の合成値）

説　明 ： イベントマスク（使用するイベント）を設定／取得します。
イベントマスクのデフォルトは、全イベントです
イベントコードについては、本節の冒頭を参照してください。

戻り値 ： 設定時：TRUE　- 成功　　取得時：イベントマスク値
FALSE - 失敗

15.4.8. 受信文字コード種別設定／取得（AjcSct{Set/Get}RxTextCode）

形　式 ： BOOL AjcSctSetRxTextCode (HAJCSCT hSct, AJCSCT_TEXTCODE code); -- 受信文字コード種別設定
AJCSCT_TEXTCODE AjcSctGetRxTextCode (HAJCSCT hSct); ------------------------ 受信文字コード種別取得
AJCSCT_TEXTCODE AjcSctGetActualRxTextCode (HAJCSCT hSct); ------------------ 実際の受信文字コード種別取得

引　数 ： hSct - インスタンスハンドル
code - 受信テキストコード種別
・AJCSCT_TXT_SJIS - シフトＪＩＳ
・AJCSCT_TXT_EUC - 日本語ＥＵＣ
・AJCSCT_TXT_UTF8 - ＵＴＦ－８
・AJCSCT_TXT_AUTO - 自動判別

説　明 ： 受信テキストの文字コードを指定します。
AjcSctGetActualRxTextCode()は、AjcSctGetRxTextCode()と同様に設定されている受信テキストコードを返しますが、受信テキストコード種別がAJCSCP_TXT_AUTOに設定されている場合、受信データから判定した実際の受信テキストコード（AJCSCT_TXT_SJIS / EUC / UTF8）を返します。

戻り値 ： 設定時：TRUE - 成功　　取得時：受信文字コード種別（エラー時は0を返す）
FALSE - 失敗

備　考 ： 受信したテキストは、指定文字コードから、バイト文字の場合はシフトＪＩＳに、UNICODEモードの場合はUTF-16に変換して通知されます。
AJCSCT_TXT_AUTO（自動判別）を設定しても、受信中に文字コードが変化した場合は、変化前の文字コードと混在した状態となるため、しばらくは受信テキストの文字コードが正常に判断されない場合があります。

15.4.9. 送信文字コード種別設定／取得（AjcSct{Set/Get}TxTextCode）

形　式 ： BOOL AjcSctSetTxTextCode (HAJCSCT hSct, AJCSCT_TEXTCODE code); -- 送信文字コード種別設定
AJCSCT_TEXTCODE AjcSctGetTxTextCode (HAJCSCT hSct); ------------------------ 送信文字コード種別取得

引　数 ： hSct - インスタンスハンドル
code - 受信テキストコード種別
・AJCSCT_TXT_SJIS - シフトＪＩＳ
・AJCSCT_TXT_EUC - 日本語ＥＵＣ
・AJCSCT_TXT_UTF8 - ＵＴＦ－８
・AJCSCT_TXT_AUTO - 受信テキストコードと同じ

説　明 ： 送信テキストの文字コードを設定／取得します。

戻り値 ： 設定時：TRUE - 成功　　取得時：送信文字コード種別（エラー時は0を返す）
FALSE - 失敗

備　考 ： AjcSctSendText()やAjcSctSendChar()で送信するテキストは、指定文字コードに変換されて送信されます。

15.4.10. 実際の送受信文字コード種別取得（AjcSctGetActual{Rx/Tx}TextCode）

形　式 ： AJCSCT_TEXTCODE AjcSctGetActualRxTextCode (HAJCSCP hScp); --- 受信文字コード種別取得
AJCSCT_TEXTCODE AjcSctGetActualTxTextCode (HAJCSCP hScp); --- 送信文字コード種別取得

引　数 ： hSct - インスタンスハンドル

説　明 ： 実際の送受信文字コードを取得します。
AjcSctGetActualRxTextCode()では、AJCSCT_TXT_AUTOが設定されている場合は、実際に受信したテキストから判断された文字コード種別を返します。（デフォルトはAJCSCT_TXT_SJIS）
AjcSctGetActualTxTextCode()では、AJCSCT_TXT_AUTOが設定されている場合は、受信文字コード種別を返します。

戻り値 ： 文字コード種別（AJCSCP_TXT_SJIS / EUC / UTF8）　エラー時は０を返す

15.4.11. イベント発生待ち（AjcSctWaitEvent）

形　式 ： BOOL　AjcSctWaitEvent (HAJCSCT hSct, WPARAM *pwParam, LPARAM *plParam, UI msTime);

引　数 ： hSct　　- インスタンスハンドル
pwParam　　- 発生したイベントコード格納するバッファのアドレス
plParam　　- イベントデータ取得情報を格納するバッファのアドレス
msTime　　- 待ち時間[ms]（－１を指定した場合は、永久にイベント待ちとなる）

説　明 ： イベントの発生を待ちます。
本関数によりイベントの発生を待つには、AjcSctSetMode()で、hWndNtc 引数に NULL を指定していなければなりません。

pwParam で指定したバッファには、発生したイベントコードが格納されます。
イベントコードについては、本節冒頭の「イベントコード一覧」を参照してください。

plParam で指定したバッファには、イベントデータ取得情報が設定されます。
この情報は、AjcSctGetEventData()の lParam 引数に指定します。

戻り値 ： TRUE　- イベントが発生した
FALSE - タイムアウト（イベント未発生）／エラー

15.4.12. イベントデータ取得（AjcSctGetEventData）

形　式 ： BOOL　AjcSctGetEventData (HAJCSCT hSct, LPARAM lParam, VOP *ppDat, UIP plDat, UIP pParam);

引　数 ： hSct　　- インスタンスハンドル
lParam　　- イベントデータ取得情報
ppDat　　- 受信データへのポインタを格納するバッファのアドレス
plDat　　- 受信データのバイト数／文字数を格納するバッファのアドレス
pParam　　- パラメタ情報を格納するバッファのアドレス

説　明 ： 発生したイベントに関する付随情報を取得します。
lParam は、ウインドメッセージの lParam／AjcSctWaitEvent()で取得したイベントデータ取得情報を指定します。
ppDat～pParam で取得される情報については、本節冒頭の「イベントコード一覧」を参照してください。

本関数は、イベントが発生した場合、必ず実行してください。
また、イベントで通知された情報を使用した後には、AjcSctRelEventData()を実行しなければなりません。

戻り値 ： TRUE　- 成功
FALSE - 失敗

15.4.13. イベントデータ開放（AjcSctRelEventData）

形　式 ： BOOL　AjcSctRelEventData (HAJCSCT hSct, LPARAM lParam);

引　数 ： hSct　　- インスタンスハンドル
lParam　　- イベントデータ取得情報

説　明 ： イベントデータを開放します。
lParam は、ウインドメッセージの lParam／AjcSctWaitEvent()で取得したイベントデータ取得情報を指定します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

15.4.14. イベントが空になるまで待つ（AjcSctWaitEmptyEvent）

形　式 ： BOOL　AjcSctWaitEmptyEvent (HAJCSCT hSct, UI msTime);

引　数 ： hSct　　- インスタンスハンドル
msTime　　- 最大待ち時間[ms] (-1 : 永久)

説　明 ： イベントが空になるまで待ちます。(AjcSctGetEventDat()で、非同期に蓄積されたイベントが全て取得されるまで待つ)
msTime で指定した時間が経過しても空にならない場合は、待つのをやめて FALSE を返します。

戻り値 ： TRUE　- イベントが空になった
FALSE - 失敗／指定時間内にイベントは空にならなかった

15.4.15. パケットフレーム認識・制御コード設定／取得（AjcSct{Set/Get}PktCtrlCode）

形　式 ： BOOL　AjcSctSetPktCtrlCode (HAJCSCT hSct, UI　stx, UI　etx,　UI　dle);　-- 制御コード設定
BOOL　AjcSctGetPktCtrlCode (HAJCSCT hSct, UIP pStx, UIP pEtx, UIP pDle); -- 制御コード取得

引　数 ： hSct　　- インスタンスハンドル
制御コード設定時
stx　　- STX のコード値（設定しない場合は、０を指定）
etx　　- ETX のコード値（設定しない場合は、０を指定）
dle　　- DLE のコード値（設定しない場合は、０を指定）
制御コード取得時
pStx　　- STX のコード値を格納するバッファのアドレス（不要時はＮＵＬＬ）
pEtx　　- ETX のコード値を格納するバッファのアドレス（不要時はＮＵＬＬ）
pDle　　- DLE のコード値を格納するバッファのアドレス（不要時はＮＵＬＬ）

説　明 ： パケットフレームを認識する為の制御コード（STX, ETX, DLE）の値を設定／取得します。
デフォルトでは、STX=0x02, ETX=0x03, DLE=0x10 となっています。

戻り値 ： TRUE　- 成功
FALSE - 失敗

15.4.16. パケット受信タイムアウト値 設定／取得（AjcSct{Set/Get}PktTimeout）

形　式 ： BOOL　AjcSctSetPktTimeout (HAJCSCT hSct, UI　msTime);　-- タイムアウト値設定
BOOL　AjcSctGetPktTimeout (HAJCSCT hSct, UIP pMsTime); -- タイムアウト値取得

引　数 ： hSct　　- インスタンスハンドル
msTime　　- タイムアウト値[ms]（０：タイムアウトなし）

説　明 ： パケットフレーム受信時のタイムアウト値を設定／取得します。
パケットフレーム受信時において、指定時間内にパケット受信が完了しない場合、当該パケットを破棄します。
デフォルトでは、３０００ｍｓに設定されています。

戻り値 ： TRUE　- 成功
FALSE - 失敗

15.4.17. 1 文字送信（AjcSctSendChar）

形　式 ： BOOL　AjcSctSendChar(HAJCSCT hSct, UT code);

引　数 ： hSct　　- インスタンスハンドル
code　　- 送信する文字のバイトコード／文字コード

説　明 ： 指定されたバイトコードの文字を送信します。
バイト文字モードの場合、マルチバイト文字を送信する場合は、この関数を２度コールしてください。
UNICODE モードの場合は、送信文字をマルチバイト文字に変換して送信します。
この関数では、送信用のスプールバッファに送信データを格納するだけであり、すぐに制御を戻します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

15.4.18. テキストデータ送信（AjcSctSendText）

形　式 ： BOOL　AjcSctSendText(HAJCSCT hSct, C_UTP pTxt, UI lTxt); --- テキスト送信
BOOL　AjcSctSendText(HAJCSCT hSct, C_UTP pFmt, ...); -------- 書式テキスト送信

引　数 ： hSct　　- インスタンスハンドル
pTxt　　- 送信するテキストデータのアドレス
lTxt　　- 送信するテキストデータのバイト数／文字数（－１の場合は自動算出）
pFmt　　- 書式テキスト（printf()と同じ）

説　明 ： テキストデータの送信を行います。
AjcSctSetTxTextCode()により、送信文字コードが指定されている場合は、当該文字コードに変換したテキストを送信します。

この関数では、送信用のスプールバッファに送信データを格納するだけであり、すぐに制御を戻します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

15.4.19. バイナリデータ送信（AjcSctSendBinData）

形　式 ： BOOL　AjcSctSendBinData(HAJCSCT hSct, C_VOP pDat, UI lDat);

引　数 ： hSct　　- インスタンスハンドル
pDat　　- 送信するバイナリデータのアドレス
lDat　　- 送信するバイナリデータのバイト数

説　明 ： バイナリデータの送信を行います。
pDat と lDat で指定したバイトストリームをそのまま送信します。

この関数では、送信用のスプールバッファに送信データを格納するだけであり、すぐに制御を戻します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

15.4.20. パケットデータ送信（AjcSctSendPacket）

形　式 ： UI　AjcSctSendPacket(HAJCSCT hSct, C_VOP pPkt, UI lPkt);

引　数 ： hSct　　- インスタンスハンドル
pPkt　　- 送信するパケットデータのアドレス（空パケット送信時は NULL）
lPkt　　- 送信するパケットデータのバイト数（空パケット送信時は０）

説　明 ： 指定されたパケットデータをパケットフレームに乗せて送信します。
つまり、パケットデータ中の DLE を２つの DLE に変換し、先頭に DLE・STX を、末尾に DLE・ETX を付加したデータを送信します。
pPkt=NULL, lPkt=0 を指定した場合は空パケット（DLE,STX,DLE,ETX の４バイト）を送信します。
この関数では、送信用のスプールバッファに送信データを格納するだけであり、すぐに制御を戻します。

戻り値 ： 4～　- 成功（DLE・STX, DLE・ETX や、パケットデータ中の透過制御バイト(DLE)を含めた実際の送信バイト数）
0　　- 失敗

15.4.21. 全受信済データ破棄（AjcSctPurgeRecvData）

形　式 ： BOOL　AjcSctPurgeRecvData (HAJCSCT hSct);

引　数 ： hSct　　- インスタンスハンドル

説　明 ： 受信済のテキスト／ＥＳＣコード／制御コード／パケットデータ／パケット外データを全て破棄します。
受信チャンクデータについては、受信時にリアルタイムに通知される為、破棄対象外です。

戻り値 ： TRUE　- 成功
FALSE - 失敗

15.4.22. 全送信待ちデータ破棄（AjcSctPurgeSendData）

形　式 ： BOOL　AjcSctPurgeSendData (HAJCSCT hSct);

引　数 ： hSct　　- インスタンスハンドル

説　明 ： クライアントの送信待ちデータを全て破棄します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

15.4.23. 全受信済データと全送信待ちデータ破棄（AjcSctPurgeAllData）

形　式 ： BOOL　AjcSctPurgeAllData (HAJCSCT hSct);

引　数 ： hSct　　- インスタンスハンドル

説　明 ： クライアントの全受信済データと全送信待ちデータを破棄します。
AjcSctPurgeRecvData()と AjcSctPurgeSendData()を実行します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

15.5. サンプルプログラム

15.5.1. SW_SockClient1（送受信テスト）

このサンプルプログラムは、TCP/IP クライアント機能における送受信ファンクションを実行します。




```
 1 : //
 2 : //  S_SockClient1.c
 3 : //
 4 : #define AJCSERIALCOMPORT_H_
 5 : #define AJCSOCKSERV_H_
 6 : #include    <AjrCstXX.h>
 7 : #include    <tchar.h>
 8 : #include    "resource.h"
 9 :
10 : #define WM_SCTEVENT     (WM_USER + 100)
11 : #define MAX_TXTBOX_LEN  128
12 :
13 : //-------------------------------------------------------------------------------------------------------------//
14 : //  ツールチップ                                                                                               //
15 : //-------------------------------------------------------------------------------------------------------------//
16 : typedef struct {
17 :     int     id;
18 :     C_UTP   pTxt;
19 : } TIPTBL, *PTIPTBL;
20 : typedef const TIPTBL *PCTIPTBL;
21 :
22 : static const TIPTBL TipTbl[] = {
23 :     {IDC_TXT_SNDTEXT    , TEXT("Ｃ言語表記のテキストを設定してください。(ex. ABC¥¥n) ")                                    },
24 :     {IDC_VTH_TXTCHUNK   , TEXT("リアルタイムに受信したデータをテキストデータとして表示します。¥n")
25 :                           TEXT("複数バイト文字の場合は、文字が完結するまで次のリアルタイム受信データを待ちます。")          },
26 :     {IDC_VTH_BINCHUNK   , TEXT("リアルタイムに受信したデータをバイナリデータとして表示します。")                          },
27 :     {IDC_VTH_TEXT       , TEXT("受信ストリームから制御コード（TAB 以外）で区切られたテキストデータを抜き出して表示します。¥n")
28 :                           TEXT("¥x1B[34m ここにファイルをドロップすると、ファイルの内容をバイナリデータとして送信します。") },
29 :     {IDC_VTH_CTRL       , TEXT("受信ストリームから制御コード（TAB 以外）を抜き出して表示します。")                      },
30 :     {IDC_VTH_ESC        , TEXT("受信したストリームから、ＥＳＣシーケンス（0x1B～英字）を抜きだして表示します。")          },
31 :     {IDC_VTH_PKT        , TEXT("受信したパケットデータ（DLE・STX～DLE・ETX でサンドイッチされたデータ）をバイナリ表示します。")},
32 :     {IDC_VTH_NOPKT      , TEXT("リアルタイムに受信したデータ内のパケットデータ（DLE・STX～DLE・ETX）以外の部分をテキストとして表示します。¥n")
33 :                           TEXT("複数バイト文字の場合は、文字が完結するまで次のリアルタイム受信データを待ちます。")}
34 : };
35 :
36 : #define MAX_TIPTBL  (sizeof TipTbl / sizeof TipTbl[0])
37 :
38 : //-------------------------------------------------------------------------------------------------------------//
39 : //  ワーク                                                                                                     //
40 : //-------------------------------------------------------------------------------------------------------------//
41 : HINSTANCE      hInst;                      //  ＤＬＬインスタンスハンドル
```

```
HWND            hDlgMain;                       //  ダイアログボックスハンドル

HAJCSCT         hSct;
BOOL            fConnect = FALSE;
HWND            hWndVthTxtChu;
HWND            hWndVthBinChu;
HWND            hWndVthText;
HWND            hWndVthCtrl;
HWND            hWndVthEsc;
HWND            hWndVthPkt;
HWND            hWndVthPkt;
HWND            hWndVthNoPkt;
HWND            hWndVthLog;

//-------------------------------------------------------------------------------------------------------------//
//  内部サブ関数                                                                                               //
//-------------------------------------------------------------------------------------------------------------//
AJC_DLGPROC_DEF(Main);

static  VO      SubSendFile(HWND hDlg, C_UTP pFName);
static  VO      ShowOnConnect(VO);
static  VO      ShowOnDisconnect(VO);

//=============================================================================================================//
//                                                                                                             //
//  W i n M a i n                                                                                              //
//                                                                                                             //
//=============================================================================================================//
int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
{
    MSG     msg;

    hInst = hInstance;

    //----- メイン・ダイアログオープン ----------------//
    hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_DLGMAIN), NULL, AJC_DLGPROC_NAME(Main));
    ShowWindow(hDlgMain, SW_SHOW);

    //----- メッセージループ --------------------------//
    while (GetMessage(&msg, NULL, 0, 0)) {
        do {
            if (IsDialogMessage(hDlgMain, &msg)) break;
            TranslateMessage(&msg);
            DispatchMessage (&msg);
        } while (0);
    }

    return (int)msg.wParam ;
}
//=============================================================================================================//
//                                                                                                             //
//  ダイアログ・プロシージャ                                                                                   //
//                                                                                                             //
//=============================================================================================================//
//----- ダイアログ初期化 ------------------------------------------------------------------------------------//
AJC_DLGPROC(Main, WM_INITDIALOG     )
{
    int     i;

    hDlgMain        = hDlg;
    hWndVthTxtChu   = GetDlgItem(hDlg, IDC_VTH_TXTCHUNK );
    hWndVthBinChu   = GetDlgItem(hDlg, IDC_VTH_BINCHUNK );
    hWndVthText     = GetDlgItem(hDlg, IDC_VTH_TEXT     );
    hWndVthCtrl     = GetDlgItem(hDlg, IDC_VTH_CTRL     );
    hWndVthEsc      = GetDlgItem(hDlg, IDC_VTH_ESC      );
    hWndVthPkt      = GetDlgItem(hDlg, IDC_VTH_PKT      );
    hWndVthNoPkt    = GetDlgItem(hDlg, IDC_VTH_NOPKT    );
    hWndVthLog      = GetDlgItem(hDlg, IDC_VTH_EVTLOG   );

    //----- ラジオボタンのグループ化 ----------------//
    AjcSbcRadioBtns(GetDlgItem(hDlg, IDC_GRP_RXTEC));
    AjcSbcRadioBtns(GetDlgItem(hDlg, IDC_GRP_TXTEC));

    //----- ダイアログ項目の初期化 ------------------//
    AjcSetDlgItemStr (hDlg, IDC_TXT_SNDTEXT  , TEXT("東京都港区赤坂 1-2-3¥¥n"));
    AjcSetDlgItemStr (hDlg, IDC_TXT_SNDESC   , TEXT("[31m"                  ));
    AjcSetDlgItemStr (hDlg, IDC_TXT_SNDBIN   , TEXT("00 01 02"              ));
    AjcSetDlgItemStr (hDlg, IDC_TXT_SNDPKT   , TEXT("10 11 12"              ));
    AjcSetDlgItemStr (hDlg, IDC_TXT_SNDWORD14, TEXT("100"                   ));
    AjcSetDlgItemChk (hDlg, IDC_CHK_RTS      , TRUE);
```

```
122 :     AjcSetDlgItemChk (hDlg, IDC_CHK_DTR      , TRUE);
123 :     AjcSetDlgItemUInt(hDlg, IDC_GRP_RXTEC    , 0    );
124 :     AjcSetDlgItemUInt(hDlg, IDC_GRP_TXTEC    , 0    );
125 :     AjcSetDlgItemStr (hDlg, IDC_TXT_SERV     , TEXT("127.0.0.1"));
126 :     AjcSetDlgItemUInt(hDlg, IDC_TXT_PORTNO   , 14238);
127 :
128 :     //----- テキストボックス長設定 ----------------//
129 :     AjcSetDlgItemEdtLimit(hDlg, IDC_TXT_SNDTEXT, MAX_TXTBOX_LEN - 1);
130 :     AjcSetDlgItemEdtLimit(hDlg, IDC_TXT_SNDESC , MAX_TXTBOX_LEN - 1);
131 :     AjcSetDlgItemEdtLimit(hDlg, IDC_TXT_SNDBIN , MAX_TXTBOX_LEN - 1);
132 :     AjcSetDlgItemEdtLimit(hDlg, IDC_TXT_SNDPKT , MAX_TXTBOX_LEN - 1);
133 :
134 :     //----- ツールチップ設定 ----------------------//
135 :     for (i = 0; i < MAX_TIPTBL; i++) {
136 :         AjcTipTextAdd(GetDlgItem(hDlg, TipTbl[i].id), TipTbl[i].pTxt);
137 :     }
138 :
139 :     //----- ウインド位置ロード ---------------------//
140 :     AjcLoadWndPos(hDlg, NULL);
141 :
142 :     //----- ＳＣＴセットアップ ---------------------//
143 :     hSct = AjcSctCreate();                                      //  ＳＣＴインスタンス生成
144 :     AjcSctSetRxTextCode(hSct, AJCSCT_TXT_AUTO);                 //  受信テキストコード自動判別
145 :     AjcSctSetTxTextCode(hSct, AJCSCT_TXT_AUTO);                 //  送信テキストコード＝受信と同じ
146 :     AjcSctSetEvtMask(hSct,  0 |                                 //  イベントマスク設定
147 :                             AJCSCT_EV_RXTEXT        |           //      テキスト受信通知
148 :                             AJCSCT_EV_RXESC         |           //      ＥＳＣコード受信通知
149 :                             AJCSCT_EV_RXCTRL        |           //      制御コード受信通知
150 :                             AJCSCT_EV_RXPKT         |           //      パケットデータ受信通知
151 :                             AJCSCT_EV_RXNOPKT       |           //      パケット外データ受信通知
152 :                             AJCSCT_EV_RXCHUNK       |           //      チャンクデータ受信通知
153 :                             AJCSCT_EV_TXEMPTY       |           //      送信完了
154 :                             AJCSCT_EV_CONNECT       |           //      接続通知
155 :                             AJCSCT_EV_DISCONNECT    |           //      切断通知
156 :                             AJCSCT_EV_CNFAIL        |           //      接続失敗通知
157 :                             AJCSCT_EV_RXERR         |           //      受信エラー通知
158 :                             AJCSCT_EV_TXERR         |           //      送信エラー通知
159 :                             AJCSCT_EV_ERR           |           //      その他のエラー通知
160 :     0);
161 :     //  バイナリチャンク、テキストチャンクとも通知する
162 :     AjcSctSetChunkMode(hSct, AJCSCT_CM_BOTH);
163 :
164 :     //----- 初期グレー表示 -------------------------//
165 :     AjcEnableDlgGroup(hDlgMain, IDC_GRP_SETSIG, FALSE, FALSE);
166 :     AjcEnableDlgGroup(hDlgMain, IDC_GRP_SIGSTS, FALSE, FALSE);
167 :     AjcEnableDlgGroup(hDlgMain, IDC_GRP_SEND  , FALSE, FALSE);
168 :     AjcEnableDlgGroup(hDlgMain, IDC_GRP_RECV  , FALSE, FALSE);
169 :     //----- ダイアログ項目のロード -----------------//
170 :     AjcLoadAllControlSettings (hDlg, TEXT("DlgSetting"), AJCOPT2(AJCCTL_SELACT_, ALL, NTCRBT));
171 :
172 :     return TRUE;
173 : }
174 : //----- ウインド破棄 ---------------------------------------------------------------------------------------//
175 : AJC_DLGPROC(Main, WM_DESTROY        )
176 : {
177 :     //----- ウインド位置セーブ ---------------------//
178 :     AjcSaveWndPos(hDlg, NULL);
179 :     //----- ダイアログ項目のセーブ -----------------//
180 :     AjcSaveAllControlSettings(hDlg);
181 :
182 :     AjcSctDelete(hSct);         //  ＳＣＴインスタンス消去
183 :     PostQuitMessage(0);
184 :     return TRUE;
185 : }
186 : //----- ＳＣＴイベント通知 ---------------------------------------------------------------------------------//
187 : AJC_DLGPROC(Main, WM_SCTEVENT       )
188 : {
189 :     UI      time = GetTickCount();
190 :     UI      len, param;
191 :     union {UBP pBin; UWP pWord; UTP pTxt; VOP vp;} u;
192 :
193 :     AjcSctGetEventData(hSct, lParam, &u.vp, &len, &param);          //  イベントデータ取得
194 :     if(wParam & AJCSCT_EV_CONNECT) {                                //  ●接続通知
195 :         AjcVthPrintF(hWndVthLog, TEXT("%9d: CONNECT   len=%5d, param=0x%08X¥n"), time, len, param);
196 :         fConnect = TRUE;
197 :         ShowOnConnect();
198 :     }
199 :     if(wParam & AJCSCT_EV_DISCONNECT) {                             //  ●切断通知
200 :         AjcVthPrintF(hWndVthLog, TEXT("%9d: DISC.     len=%5d, param=0x%08X¥n"), time, len, param);
201 :         fConnect = FALSE;
```

```
202 :         ShowOnDisconnect();
203 :     }
204 :     if (wParam & AJCSCT_EV_CNFAIL) {                                    //  ●接続失敗通知
205 :         AjcVthPrintF(hWndVthLog, TEXT("%9d: CNERR     len=%5d, param=0x%08X¥n"), time, len, param);
206 :         MessageBox(hDlg, TEXT("接続を失敗しました"), TEXT("Error"), MB_ICONERROR);
207 :     }
208 :     if(wParam & AJCSCT_EV_RXCHUNK)      {                               //  ●チャンクデータ受信通知
209 :         AjcVthPrintF(hWndVthLog, TEXT("%9d: RXCHUNK   len=%5d, param=0x%08X¥n"), time, len, param);
210 :         //  バイナリチャンク
211 :         if (param == 0) {
212 :             AjcVthHexDump(hWndVthBinChu, (C_VOP)u.pBin, len);
213 :             AjcVthPrintF (hWndVthBinChu, TEXT("¥n"));
214 :         }
215 :         //  テキストチャンク
216 :         else {
217 :             AjcVthPutText(hWndVthTxtChu, u.pTxt, -1);
218 :         }
219 :     }
220 :     if (wParam & AJCSCT_EV_RXTEXT ) {                                   //  ●テキスト受信通知
221 :         AjcVthPrintF(hWndVthLog, TEXT("%9d: RXTEXT    len=%5d, param=0x%08X¥n"), time, len, param);
222 :         AjcVthPrintF(hWndVthText, TEXT("%s¥n"), u.pTxt);                //      テキストデータ表示
223 :     }
224 :     if (wParam & AJCSCT_EV_RXESC  ) {                                   //  ●ＥＳＣコード受信通知
225 :         AjcVthPrintF(hWndVthLog, TEXT("%9d: RXESC     len=%5d, param=0x%08X¥n"), time, len, param);
226 :         AjcVthPrintF(hWndVthEsc, TEXT("¥¥x1B%s¥n"), u.pTxt + 1);            //      ＥＳＣデータ表示
227 :     }
228 :     if (wParam & AJCSCT_EV_RXCTRL ) {                                   //  ●制御コード受信通知
229 :         AjcVthPrintF(hWndVthLog, TEXT("%9d: RXCTRL    len=%5d, param=0x%08X, data=%02X¥n"), time, len, param, *u.pBin);
230 :         AjcVthPrintF(hWndVthCtrl, TEXT("%02X¥n"), *u.pTxt);             //      制御コード表示
231 :     }
232 :     if (wParam & AJCSCT_EV_RXPKT  ) {                                   //  ●パケットデータ受信通知
233 :         AjcVthPrintF(hWndVthLog, TEXT("%9d: RXPKT     len=%5d, param=0x%08X¥n"), time, len, param);
234 :         AjcVthHexDump(hWndVthPkt, (C_VOP)u.pBin, len);
235 :         AjcVthPrintF (hWndVthPkt, TEXT("¥n"));
236 :     }
237 :     if (wParam & AJCSCT_EV_TXEMPTY) {                                   //  ●送信完了通知
238 :         AjcVthPrintF(hWndVthLog, TEXT("%9d: TXEMPTY   len=%5d, param=0x%08X¥n"), time, len, param);
239 :     }
240 :     if (wParam & AJCSCT_EV_RXNOPKT) {                                   //  ●パケット外テキスト受信通知
241 :         AjcVthPrintF(hWndVthLog, TEXT("%9d: RXNOPKT   len=%5d, param=0x%08X¥n"), time, len, param);
242 :         AjcVthPutText(hWndVthNoPkt, u.pTxt, -1);                        //      パケット外テキスト表示
243 :     }
244 :     if (wParam & AJCSCT_EV_RXERR      ) {                               //  ●受信エラー通知
245 :         AjcVthPrintF(hWndVthLog, TEXT("%9d: RXERR     len=%5d, param=0x%08X¥n"), time, len, param);
246 :     }
247 :     if (wParam & AJCSCT_EV_TXERR      ) {                               //  ●送信エラー通知
248 :         AjcVthPrintF(hWndVthLog, TEXT("%9d: TXERR     len=%5d, param=0x%08X¥n"), time, len, param);
249 :     }
250 :     if (wParam & AJCSCT_EV_ERR        ) {                               //  ●その他のエラー通知
251 :         AjcVthPrintF(hWndVthLog, TEXT("%9d: ERR       len=%5d, param=0x%08X¥n"), time, len, param);
252 :     }
253 :     AjcSctRelEventData(hSct, lParam);                                   //  イベントデータ開放
254 :
255 :     return TRUE;
256 : }
257 : //----- ウインドクローズ ----------------------------------------------------------------------------//
258 : AJC_DLGPROC(Main, IDCANCEL          )
259 : {
260 :     DestroyWindow(hDlg);
261 :     return TRUE;
262 : }
263 : //----- テキスト送信ボタン --------------------------------------------------------------------------//
264 : AJC_DLGPROC(Main, IDC_CMD_SNDTEXT   )
265 : {
266 :     if (HIWORD(wParam) == BN_CLICKED) {
267 :         UT  txt[MAX_TXTBOX_LEN];
268 :         UT  snd[MAX_TXTBOX_LEN];
269 :         AjcGetDlgItemStr(hDlg, IDC_TXT_SNDTEXT, txt, AJCTSIZE(txt));
270 :         AjcCLangStrToBin(txt, snd, AJCTSIZE(snd));
271 :         AjcSctSendText(hSct, snd, -1);
272 :     }
273 :     return TRUE;
274 : }
275 : //----- ＥＳＣ送信ボタン ----------------------------------------------------------------------------//
276 : AJC_DLGPROC(Main, IDC_CMD_SNDESC    )
277 : {
278 :     if (HIWORD(wParam) == BN_CLICKED) {
279 :         UT  txt[MAX_TXTBOX_LEN];
280 :         AjcGetDlgItemStr(hDlg, IDC_TXT_SNDESC, txt, AJCTSIZE(txt));
281 :         AjcSctSendChar(hSct, 0x1B);
```

```
282 :         AjcSctSendText(hSct, txt, -1);
283 :     }
284 :     return TRUE;
285 : }
286 : //----- バイナリ送信ボタン ----------------------------------------------------------------------------------//
287 : AJC_DLGPROC(Main, IDC_CMD_SNDBIN    )
288 : {
289 :     if (HIWORD(wParam) == BN_CLICKED) {
290 :         UI  ix;
291 :         UTP p;
292 :         UT  txt[MAX_TXTBOX_LEN];
293 :         UB  bin[MAX_TXTBOX_LEN];
294 :         ix = 0;
295 :         AjcGetDlgItemStr(hDlg, IDC_TXT_SNDBIN, txt, AJCTSIZE(txt));
296 :         if (p = MAjcStrTok(txt, TEXT(" "))) {
297 :             do {
298 :                 if (AjcHexStrToUB(p, 2, &bin[ix])) {
299 :                     ix++;
300 :                 }
301 :             } while (p = MAjcStrTok(NULL, TEXT(" ")));
302 :         }
303 :         if (ix != 0) {
304 :             AjcSctSendBinData(hSct, bin, ix);
305 :         }
306 :     }
307 :     return TRUE;
308 : }
309 : //----- パケット送信ボタン ----------------------------------------------------------------------------------//
310 : AJC_DLGPROC(Main, IDC_CMD_SNDPKT    )
311 : {
312 :     if (HIWORD(wParam) == BN_CLICKED) {
313 :         UI  ix;
314 :         UTP p;
315 :         UT  txt[MAX_TXTBOX_LEN];
316 :         UB  bin[MAX_TXTBOX_LEN];
317 :         ix = 0;
318 :         AjcGetDlgItemStr(hDlg, IDC_TXT_SNDPKT, txt, AJCTSIZE(txt));
319 :         if (p = MAjcStrTok(txt, TEXT(" "))) {
320 :             do {
321 :                 if (AjcHexStrToUB(p, 2, &bin[ix])) {
322 :                     ix++;
323 :                 }
324 :             } while (p = MAjcStrTok(NULL, TEXT(" ")));
325 :         }
326 :         if (ix != 0) {
327 :             AjcSctSendPacket(hSct, bin, ix);
328 :         }
329 :     }
330 :     return TRUE;
331 : }
332 :
333 : //----- 接続／切断 ボタン ----------------------------------------------------------------------------------//
334 : AJC_DLGPROC(Main, IDC_CMD_CONNECT   )
335 : {
336 :
337 :     if (HIWORD(wParam) == BN_CLICKED) {
338 :         if (fConnect) {
339 :             AjcSctDisconnect(hSct);
340 :         }
341 :         else {
342 :             UT      szServ[256];
343 :             UT      szPort[256];
344 :             AjcGetDlgItemStr(hDlg, IDC_TXT_SERV  , szServ, AJCTSIZE(szServ));
345 :             AjcGetDlgItemStr(hDlg, IDC_TXT_PORTNO, szPort, AJCTSIZE(szPort));
346 :             AjcSctConnect(hSct, szServ, szPort, AF_INET, hDlg, WM_SCTEVENT);
347 :         }
348 :     }
349 :     return TRUE;
350 : }
351 : //----- 受信テキストエンコード ------------------------------------------------------------------------------//
352 : AJC_DLGPROC(Main, IDC_GRP_RXTEC     )
353 : {
354 :     if (HIWORD(wParam) == AJCRBTN_SELECT) {
355 :         //  受信テキストエンコード設定
356 :         switch (lParam) {
357 :             case 0: AjcSctSetRxTextCode(hSct, AJCSCT_TXT_SJIS); break;
358 :             case 1: AjcSctSetRxTextCode(hSct, AJCSCT_TXT_UTF8); break;
359 :             case 2: AjcSctSetRxTextCode(hSct, AJCSCT_TXT_EUC ); break;
360 :             case 3: AjcSctSetRxTextCode(hSct, AJCSCT_TXT_AUTO); break;
361 :         }
```

```
362 :         //  送信テキストエンコード設定（送信エンコードが AUTO の場合、受信エンコードと同一とするため）
363 :         SendMessage(hDlg, WM_COMMAND, MAKELONG(IDC_GRP_TXTEC, AJCRBTN_SELECT), AjcGetDlgItemUInt(hDlg, IDC_GRP_TXTEC));
364 :     }
365 :     return TRUE;
366 : }
367 : //----- 送信テキストエンコード ----------------------------------------------------------------------------------//
368 : AJC_DLGPROC(Main, IDC_GRP_TXTEC     )
369 : {
370 :     if (HIWORD(wParam) == AJCRBTN_SELECT) {
371 :         //  送信テキストエンコード設定
372 :         switch (lParam) {
373 :             case 0: AjcSctSetTxTextCode(hSct, AJCSCT_TXT_SJIS); break;
374 :             case 1: AjcSctSetTxTextCode(hSct, AJCSCT_TXT_UTF8); break;
375 :             case 2: AjcSctSetTxTextCode(hSct, AJCSCT_TXT_EUC ); break;
376 :             case 3: AjcSctSetTxTextCode(hSct, AJCSCT_TXT_AUTO); break;
377 :         }
378 :     }
379 :     return TRUE;
380 : }
381 : //----- 全てクリアーボタン --------------------------------------------------------------------------------------//
382 : AJC_DLGPROC(Main, IDC_CMD_CLRALL    )
383 : {
384 :     if (HIWORD(wParam) == BN_CLICKED) {
385 :         AjcVthClear(hWndVthTxtChu   );
386 :         AjcVthClear(hWndVthBinChu   );
387 :         AjcVthClear(hWndVthText     );
388 :         AjcVthClear(hWndVthCtrl     );
389 :         AjcVthClear(hWndVthEsc      );
390 :         AjcVthClear(hWndVthPkt      );
391 :         AjcVthClear(hWndVthNoPkt    );
392 :     }
393 :     return TRUE;
394 : }
395 : //----- ＶＴＨ（テキスト）からの通知 ----------------------------------------------------------------------------//
396 : AJC_DLGPROC(Main, IDC_VTH_TEXT  )
397 : {
398 :     if (HIWORD(wParam) == AJCVTHN_DROPFILE) {       //  ファイルドロップ
399 :         UI      i, nFile = (UI)lParam;
400 :         HANDLE  hFile;
401 :         UL      bytes;
402 :         UT      path[MAX_PATH];
403 :         UB      buf[2014];
404 :         for (i = 0; i < nFile; i++) {
405 :             AjcVthGetDroppedFile(hWndVthText, path);
406 :             if ((hFile = CreateFile(path, GENERIC_READ, FILE_SHARE_READ, NULL, OPEN_EXISTING, 0, NULL)) != INVALID_HANDLE_VALUE) {
407 :                 while (ReadFile(hFile, buf, sizeof buf, &bytes, NULL) && bytes != 0) {
408 :                     AjcSctSendBinData(hSct, buf, bytes);
409 :                 }
410 :                 CloseHandle(hFile);
411 :             }
412 :         }
413 :     }
414 :     return TRUE;
415 : }
416 : //-------------------------------------------------------------------------------------------------------------//
417 : AJC_DLGMAP_DEF(Main)
418 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG      )
419 :     AJC_DLGMAP_MSG(Main, WM_DESTROY         )
420 :     AJC_DLGMAP_MSG(Main, WM_SCTEVENT        )
421 :     AJC_DLGMAP_CMD(Main, IDCANCEL           )
422 :     AJC_DLGMAP_CMD(Main, IDC_CMD_SNDTEXT    )
423 :     AJC_DLGMAP_CMD(Main, IDC_CMD_SNDESC     )
424 :     AJC_DLGMAP_CMD(Main, IDC_CMD_SNDBIN     )
425 :     AJC_DLGMAP_CMD(Main, IDC_CMD_SNDPKT     )
426 :     AJC_DLGMAP_CMD(Main, IDC_CMD_CONNECT    )
427 :     AJC_DLGMAP_CMD(Main, IDC_GRP_RXTEC      )
428 :     AJC_DLGMAP_CMD(Main, IDC_GRP_TXTEC      )
429 :     AJC_DLGMAP_CMD(Main, IDC_CMD_CLRALL     )
430 :     AJC_DLGMAP_CMD(Main, IDC_VTH_TEXT       )
431 : AJC_DLGMAP_END
432 :
433 : //-------------------------------------------------------------------------------------------------------------//
434 : //  ファイル送信                                                                                              //
435 : //-------------------------------------------------------------------------------------------------------------//
436 : VO      SubSendFile(HWND hDlg, C_UTP pFName)
437 : {
438 :     HANDLE  fh;
439 :     UL      bytes;
440 :     UT      path[MAX_PATH];
441 :     UT      buf[1024];
```

```
442 :
443 :     AjcGetAppPath(path, MAX_PATH);
444 :     MAjcStrCat(path, AJCTSIZE(path), pFName);
445 :     fh = CreateFile(path, GENERIC_READ, 0, NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
446 :     if (fh != INVALID_HANDLE_VALUE) {
447 :         while (ReadFile(fh, buf, AJCTSIZE(buf), &bytes, NULL) && bytes != 0) {
448 :             AjcSctSendBinData(hSct, buf, bytes);
449 :         }
450 :         CloseHandle(fh);
451 :     }
452 :     else {
453 :         UT      txt[512];
454 :         AjcSnPrintF(txt, AJCTSIZE(txt), TEXT("'%s' open falure!"), path);
455 :         MessageBox(hDlgMain, txt, TEXT("Error"), MB_ICONERROR);
456 :     }
457 : }
458 :
459 : //-------------------------------------------------------------------------------------------------------------//
460 : //  接続時の表示                                                                                               //
461 : //-------------------------------------------------------------------------------------------------------------//
462 : static  VO      ShowOnConnect(VO)
463 : {
464 :     AjcSetDlgItemStr (hDlgMain, IDC_CMD_CONNECT, TEXT("切 断"));        //  ボタンフェース
465 :     AjcEnableDlgGroup(hDlgMain, IDC_GRP_SEND, TRUE, TRUE);              //  送信表示有効化
466 :     AjcEnableDlgGroup(hDlgMain, IDC_GRP_RECV, TRUE, TRUE);              //  受信表示有効化
467 : }
468 : //-------------------------------------------------------------------------------------------------------------//
469 : //  切断時の表示                                                                                               //
470 : //-------------------------------------------------------------------------------------------------------------//
471 : static  VO      ShowOnDisconnect(VO)
472 : {
473 :     AjcSetDlgItemStr (hDlgMain, IDC_CMD_CONNECT, TEXT("接 続"));        //  ボタンフェース
474 :     AjcEnableDlgGroup(hDlgMain, IDC_GRP_SEND, FALSE, FALSE);            //  送信表示無効化
475 :     AjcEnableDlgGroup(hDlgMain, IDC_GRP_RECV, FALSE, FALSE);            //  受信表示無効化
476 : }
```

15.5.2. SW_SockClient2（エコーバック）

このサンプルプログラムは、受信データをエコーバック（受信したデータをそのままサーバへ返信）します。



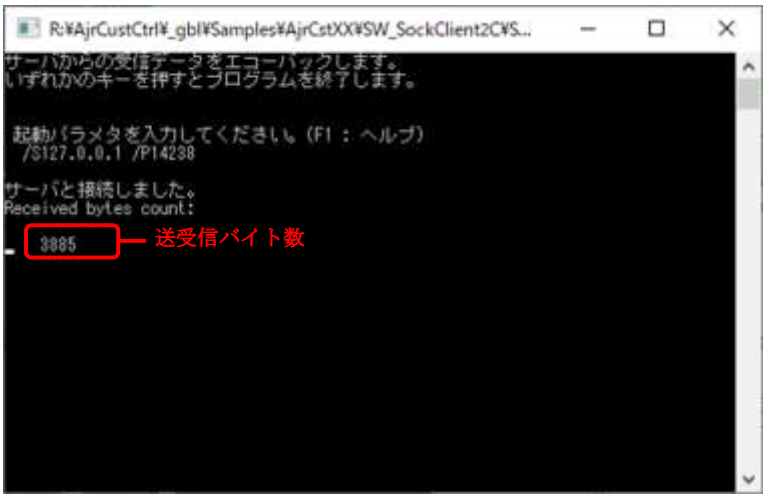

```
 1 : //
 2 : //  SW_SockClient2.c
 3 : //
 4 : #define AJCSOCKSERV_H_
 5 : #define AJCSERIALCOMPORT_H_
 6 : #include    <AjrCstXX.h>
 7 : #include    <tchar.h>
 8 : #include    "resource.h"
 9 :
10 : #define WM_SCPEVENT     (WM_USER + 100)
11 :
12 : //-----------------------------------------------------------------------------------------------------------------//
13 : //  ワーク                                                                                                         //
14 : //-----------------------------------------------------------------------------------------------------------------//
15 : static  HINSTANCE       hInst;                        //  ＤＬＬインスタンスハンドル
16 : static  HWND            hDlgMain;                     //  ダイアログボックスハンドル
17 :
18 : static  HAJCSCT         hSct;
19 : static  UI              TotalBytes;
20 : static  BOOL            fConnect = FALSE;
21 :
22 : //-----------------------------------------------------------------------------------------------------------------//
23 : //  内部サブ関数                                                                                                   //
24 : //-----------------------------------------------------------------------------------------------------------------//
25 : AJC_DLGPROC_DEF(Main);
26 :
27 : //=================================================================================================================//
28 : //                                                                                                                 //
29 : //  Ｗ ｉ ｎ Ｍ ａ ｉ ｎ                                                                                           //
30 : //                                                                                                                 //
31 : //=================================================================================================================//
32 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
33 : {
34 :     MSG     msg;
35 :
36 :     hInst = hInstance;
37 :
38 :     //----- メイン・ダイアログオープン -----------------//
39 :     hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_DLGMAIN), NULL, AJC_DLGPROC_NAME(Main));
40 :     ShowWindow(hDlgMain, SW_SHOW);
41 :
42 :     //----- メッセージループ ---------------------------//
43 :     while (GetMessage(&msg, NULL, 0, 0)) {
44 :         do {
45 :             if (IsDialogMessage(hDlgMain, &msg)) break;
46 :             TranslateMessage(&msg);
47 :             DispatchMessage (&msg);
48 :         } while (0);
49 :     }
50 :
51 :     return (int)msg.wParam ;
52 : }
53 : //=================================================================================================================//
54 : //                                                                                                                 //
55 : //  ダイアログ・プロシージャ                                                                                       //
56 : //                                                                                                                 //
57 : //=================================================================================================================//
58 : //----- ダイアログ初期化 -----------------------------------------------------------------------------------------//
59 : AJC_DLGPROC(Main, WM_INITDIALOG )
60 : {
```

```
 61 :     hDlgMain   = hDlg;
 62 :
 63 :     //  ウインド位置ロード
 64 :     AjcLoadWndPos(hDlg, NULL);
 65 :
 66 :     //  ダイアログ項目のロード
 67 :     MAjcSetDlgPropStr (hDlg, IDC_TXT_SERV, TEXT("127.0.0.1"), NULL , 0);
 68 :     MAjcSetDlgPropStr (hDlg, IDC_TXT_PORT, TEXT("14238"    ), NULL , 0);
 69 :     AjcLoadDlgProfiles(hDlg, TEXT("DlgSetting"));
 70 :
 71 :     hSct = AjcSctCreate();                                              // インスタンス生成
 72 :     AjcSctSetChunkMode(hSct, AJCSCT_CM_BIN);                            //  モード設定
 73 :     AjcSctSetEvtMask(hSct, AJCSCT_EV_CONNECT | AJCSCT_EV_DISCONNECT |   //  イベントマスク設定
 74 :                            AJCSCT_EV_CNFAIL  | AJCSCT_EV_RXCHUNK);
 75 :
 76 :     AjcSetDlgItemStr (hDlg, IDC_TXT_MSG, TEXT("接続ボタンを押して、サーバと接続してください。"));
 77 :     AjcSetDlgItemUInt(hDlg, IDC_TXT_BYTES   , 0);
 78 :
 79 :     return TRUE;
 80 : }
 81 : //----- ウインド破棄 ----------------------------------------------------------------------------------//
 82 : AJC_DLGPROC(Main, WM_DESTROY    )
 83 : {
 84 :     //  ウインド位置セーブ
 85 :     AjcSaveWndPos(hDlg, NULL);
 86 :     //  ダイアログ項目セーブ
 87 :     AjcSaveDlgProfiles(hDlg, TEXT("DlgSetting"));
 88 :     AjcReleaseDlgProps(hDlg);
 89 :     //  インスタンス消去
 90 :     AjcSctDelete(hSct);
 91 :
 92 :     PostQuitMessage(0);
 93 :     return TRUE;
 94 : }
 95 : //----- ＳＣＭイベント通知 ----------------------------------------------------------------------------//
 96 : AJC_DLGPROC(Main, WM_SCPEVENT   )
 97 : {
 98 :     static  UI      TotalBytes = 0;
 99 :     VOP     pDat;
100 :     UI      len, param;
101 :
102 :     AjcSctGetEventData(hSct, lParam, &pDat, &len, &param);              //  イベントデータ取得
103 :     if(wParam & AJCSCT_EV_RXCHUNK ) {                                   //  ●チャンクデータ受信通知？
104 :         //  バイト数表示
105 :         TotalBytes += len;
106 :         AjcSepDlgItemUInt(hDlg, IDC_TXT_BYTES, TotalBytes);
107 :         //  データ返送
108 :         AjcSctSendBinData(hSct, pDat, len);
109 :     }
110 :     if(wParam & AJCSCT_EV_CONNECT) {                                    //  ●接続通知？
111 :         fConnect = TRUE;
112 :         AjcSetDlgItemStr(hDlg, IDC_CMD_CONNECT, TEXT("切断"));
113 :         AjcSetDlgItemStr(hDlg, IDC_TXT_MSG , TEXT("¥n サーバと接続状態です。"));
114 :         //  バイト数リセット
115 :         AjcSetDlgItemUInt(hDlg, IDC_TXT_BYTES, TotalBytes = 0);
116 :     }
117 :     if(wParam & AJCSCT_EV_DISCONNECT) {                                 //  ●切断通知？
118 :         fConnect = FALSE;
119 :         AjcSetDlgItemStr(hDlg, IDC_CMD_CONNECT, TEXT("接続"));
120 :         AjcSetDlgItemStr(hDlg, IDC_TXT_MSG , TEXT("¥n 接続ボタンを押して、サーバと接続してください。"));
121 :     }
122 :     if(wParam & AJCSCT_EV_CNFAIL) {                                   //  ●接続失敗通知？
123 :         AjcSetDlgItemStr(hDlg, IDC_TXT_MSG, TEXT("接続を失敗しました。¥r¥n")
124 :                                             TEXT("接続ボタンを押して、サーバと接続してください。"));
125 :     }
126 :     AjcSctRelEventData(hSct, lParam);                                   //  イベントデータ開放
127 :
128 :     return TRUE;
129 : }
130 : //----- ウインドクローズ ------------------------------------------------------------------------------//
131 : AJC_DLGPROC(Main, IDCANCEL      )
132 : {
133 :     DestroyWindow(hDlg);
134 :     return TRUE;
135 : }
136 : //----- 接続／切断ボタン ------------------------------------------------------------------------------//
137 : AJC_DLGPROC(Main, IDC_CMD_CONNECT    )
138 : {
139 :     if (fConnect) {
140 :         AjcSctDisconnect(hSct);
```

```
141 :     }
142 :     else {
143 :         UT      serv[MAX_PATH];
144 :         UT      port[MAX_PATH];
145 :         TotalBytes = 0;
146 :         AjcGetDlgItemStr(hDlg, IDC_TXT_SERV, serv, AJCTSIZE(serv));
147 :         AjcGetDlgItemStr(hDlg, IDC_TXT_PORT, port, AJCTSIZE(port));
148 :         AjcSctConnect(hSct, serv, port, AF_INET, hDlg, WM_SCPEVENT);
149 :         AjcSetDlgItemStr(hDlg, IDC_LBL_PORTSTATE, TEXT("Open in progress"));
150 :     }
151 :     return TRUE;
152 : }
153 : //-----------------------------------------------------------------------------------------------------------------//
154 : AJC_DLGMAP_DEF(Main)
155 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG  )
156 :     AJC_DLGMAP_MSG(Main, WM_DESTROY     )
157 :     AJC_DLGMAP_MSG(Main, WM_SCPEVENT    )
158 :     AJC_DLGMAP_CMD(Main, IDCANCEL       )
159 :     AJC_DLGMAP_CMD(Main, IDC_CMD_CONNECT)
160 : AJC_DLGMAP_END
```

15.5.3. SW_SockClient2C（エコーバック，コンソールアプリ）

このサンプルプログラムは、SW_SockClient2 と同様に受信データをエコーバック（受信したデータをそのまま返信）を行うコンソールアプリケーションです。

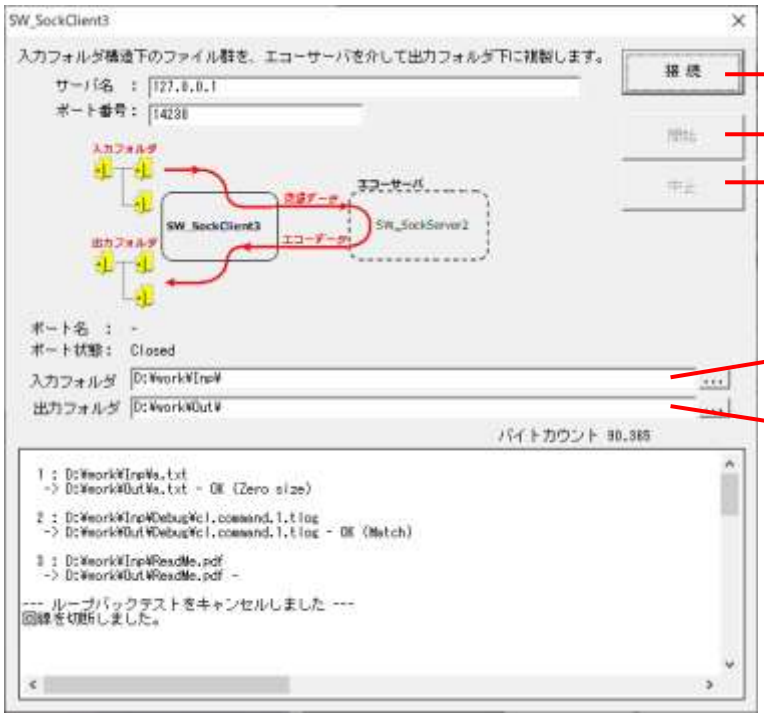



```
 1 : //
 2 : //  SW_SockClient2C.c
 3 : //
 4 : #define AJCSOCKSERV_H_
 5 : #define AJCSERIALCOMPORT_H_
 6 : #include    <AjrCstXX.h>
 7 : #include    <conio.h>
 8 : #include    <tchar.h>
 9 :
10 : //  ヘルプテキスト
11 : static  UT          HelpText[] = TEXT("起動パラメタの形式¥n¥n"                  )
12 :                                 TEXT("      /S<Name>  - TCP/IP サーバ名¥n"      )
13 :                                 TEXT("      /Pnnn     - TCP/IP ポート番号¥n"    )
14 :                                 TEXT("¥n"                                     );
15 : //  ソケット通信クライアントインスタンス
16 : static  HAJCSCT     hSct = NULL;
17 : //  通信パラメタ
18 : static UT           TcpServ[256] = TEXT("127.0.0.1");       //  TCP/IP サーバ名
19 : static UT           TcpPort[256] = TEXT("14238");           //  TCP/IP ポート番号
20 : //  バイトカウンタ
21 : static  UI          ByteCount = 0;
22 :
23 : //----- コンソール入力のコールバック (起動パラメタ解析) ----------------------------------------------------//
24 : static VO   CALLBACK cbNtcArgs(int argc, UT *argv[], C_UTP pTxt, UX cbp)
25 : {
26 :     int     i;
27 :
28 :     for (i = 0; i < argc; i++) {
29 :         if      (MAjcStrNICmp(argv[i], TEXT("/S"  ), 2) == 0) {MAjcStrCpy(TcpServ, AJCTSIZE(TcpServ), argv[i] + 2);}
30 :         else if (MAjcStrNICmp(argv[i], TEXT("/P"  ), 2) == 0) {MAjcStrCpy(TcpPort, AJCTSIZE(TcpPort), argv[i] + 2);}
31 :         else {
32 :             AjcPrintF(TEXT("*** Invalid parameter %s ***¥n"), argv[i]);
33 :         }
34 :     }
35 : }
36 :
37 : //----- コンソール強制終了ハンドラ ------------------------------------------------------------------------//
38 : static BOOL CALLBACK cbConApExit(DWORD CtrlType)
39 : {
40 :     //  シリアル通信インスタンス消去
41 :     if (hSct != NULL) {
42 :         AjcSctDelete(hSct);
43 :         hSct = NULL;
44 :     }
45 :     return FALSE;   //  FALSE : 次のハンドラをコール
46 : }
47 : //-------------------------------------------------------------------------------------------------------//
48 : //  主制御
49 : int AjcMain(int argc, UTP argv[])
50 : {
51 :     WPARAM      wParam;
```

```
52 :     LPARAM      lParam;
53 :     UTP         pDat;
54 :     UI          lDat;
55 :     UI          param;
56 :     BOOL        rsu;
57 :     BOOL        fExit = FALSE;
58 :     UT          buf[256] = {0};
59 :
60 :     AjcSetStdoutMode();
61 :
62 :     do {
63 :         //  コンソール強制終了ハンドラ設定
64 :         SetConsoleCtrlHandler(cbConApExit, TRUE);
65 :         //  初期メッセージ
66 :         AjcPrintF(TEXT("サーバからの受信データをエコーバックします。¥n いずれかのキーを押すとプログラムを終了します。¥n¥n"));
67 :         //  コマンドパラメタ解析／入力
68 :         if (argc >= 2) {
69 :             cbNtcArgs(argc - 1, &argv[1], GetCommandLine(), 0);
70 :         }
71 :         else {
72 :             AjcPrintF(TEXT("¥n 起動パラメタを入力してください。(F1 : ヘルプ)¥n  "));
73 :             if (!AjcConInputEx(0, buf, AJCTSIZE(buf), HelpText, 0, cbNtcArgs)) {
74 :                 break;
75 :             }
76 :         }
77 :         AjcPrintF(TEXT("¥n"));
78 :         //  ソケット通信インスタンス生成
79 :         hSct = AjcSctCreate();
80 :         //  モード設定 (バイナリチャンク受信)
81 :         AjcSctSetChunkMode(hSct, AJCSCT_CM_BIN);
82 :         //  使用するイベント設定
83 :         AjcSctSetEvtMask(hSct, AJCSCT_EV_CONNECT | AJCSCT_EV_DISCONNECT |
84 :                                AJCSCT_EV_CNFAIL  | AJCSCT_EV_RXCHUNK);
85 :
86 :         //  サーバと接続
87 :         rsu = AjcSctConnect(hSct, TcpServ, TcpPort, AF_INET, NULL, 0);
88 :         if (rsu) {                                                      //  ポート オープン成功？
89 :             while (!fExit) {                                            //    キー押下までループ
90 :                 if (AjcSctWaitEvent(hSct, &wParam, &lParam, 200)) {     //      イベント待ち，イベントあり？
91 :                     AjcSctGetEventData(hSct, lParam, &pDat, &lDat, &param);     //        イベントデータ取得
92 :                     if(wParam & AJCSCT_EV_RXCHUNK ) {                   //  ●チャンクデータ受信通知？
93 :                         AjcSctSendBinData(hSct, pDat, lDat);            //          エコーバック送信
94 :                         ByteCount += lDat;
95 :                         AjcPrintF(TEXT("    %u¥r"), ByteCount);
96 :                     }
97 :                     if(wParam & AJCSCT_EV_CONNECT) {                    //  ●接続通知？
98 :                         AjcPrintF(TEXT("サーバと接続しました。¥n"));
99 :                         AjcPrintF(TEXT("Received bytes count:¥n¥n    0¥r"));
100 :                    }
101 :                    if(wParam & AJCSCT_EV_DISCONNECT) {                 //  ●切断通知？
102 :                        AjcPrintF(TEXT("通信回線を切断しました。¥n"));
103 :                    }
104 :                    if(wParam & AJCSCT_EV_CNFAIL) {                     //  ●接続失敗通知？
105 :                        AjcPrintF(TEXT("サーバとの接続を失敗しました。¥n"));
106 :                        fExit = TRUE;
107 :                    }
108 :                    AjcSctRelEventData(hSct, lParam);                   //        イベントデータ開放
109 :                }
110 :                //  キー入力チェック
111 :                if (_kbhit()) fExit = TRUE;
112 :            }
113 :            AjcSctDisconnect(hSct);                                     //    回線切断
114 :        }
115 :        else {                                                          //  Ｃｏｎｅｃｔ失敗？
116 :            AjcPrintF(TEXT("Connect を失敗しました¥n"));
117 :        }
118 :        AjcSctDelete(hSct);                                             //  インスタンス消去
119 :    } while(0);
120 :
121 :    AjcPrintF(TEXT("Ｅｎｔｅｒキーを押してください"));
122 :    getchar();
123 :    return 0;
124 : }
```

15.5.4. SW_SockClient3（ループバックによるフォルダ構造コピー）

このサンプルプログラムは、エコーサーバを介して、入力フォルダ下のフォルダ構造と全ファイルを、出力フォルダ下に複製します。
入力フォルダ下の全ファイルをエコーサーバに送信し、返信されたデータで出力フォルダ下にフォルダ構造とファイルを複製します。
複製したファイルは、元のファイルとコンペアし、一致／不一致をログ表示します。
複製する前に、出力フォルダ下の全フォルダやファイルは消去されます。



回線を接続／切断します
エコーサーバへの送信を開始します
フォルダ構造の複製を中止します
複製するフォルダ構造の上位フォルダ
複製先のフォルダ

※エコーサーバには、以下のサンプルプログラムを使用できます。
・SW_SockServer2（エコーサーバ）

```
 1 : //
 2 : //  SW_SockClient3.c
 3 : //
 4 : #define AJCSOCKSERV_H_
 5 : #define AJCSERIALCOMPORT_H_
 6 : #include    <AjrCstXX.h>
 7 : #include    <tchar.h>
 8 : #include    "resource.h"
 9 :
10 : #define WM_SCTEVENT     (WM_USER + 100)
11 :
12 : //-------------------------------------------------------------------------------------------------------------//
13 : //  ワーク                                                                                                     //
14 : //-------------------------------------------------------------------------------------------------------------//
15 : static  HINSTANCE       hInst;                              //  ＤＬＬインスタンスハンドル
16 : static  HWND            hDlgMain        = NULL;             //  ダイアログボックスハンドル
17 : static  HWND            hVthLog         = NULL;             //  ログウインド
18 :
19 : static  HAJCSCT         hSct            = NULL;             //  TCP/IP クライアント通信インスタンスハンドル
20 : static  HBITMAP         hBmpProgImage   = NULL;             //  ビットマップハンドル
21 : static  HAJCVQUE        hVQue           = NULL;             //  キューハンドル（入出力パス蓄積）
22 : static  HANDLE          fhInp           = INVALID_HANDLE_VALUE; //  入力ファイルハンドル
23 : static  HANDLE          fhOut           = INVALID_HANDLE_VALUE; //  出力ファイルハンドル
24 : static  ULL             FileSize        = 0;                //  入力ファイルサイズ
25 : static  UI              Unmatch         = 0;                //  比較不一致ファイル数
26 : static  UI              ListCount       = 0;                //  全ファイル数
27 : static  UI              BytesCount      = 0;                //  送信バイトカウンタ
28 : static  BOOL            fSendBusy       = FALSE;            //  処理中フラグ
29 : static  BOOL            fConnect        = FALSE;            //  接続状態フラグ
30 : static  UT              InpDir[MAX_PATH];                   //  入力ディレクトリパス
31 : static  UT              OutDir[MAX_PATH];                   //  出力ディレクトリパス
32 : static  UT              InpFilePath[MAX_PATH];              //  入力ファイルパス
33 : static  UT              OutFilePath[MAX_PATH];              //  出力ファイルパス
34 :
35 : //-------------------------------------------------------------------------------------------------------------//
36 : //  内部サブ関数                                                                                               //
37 : //-------------------------------------------------------------------------------------------------------------//
```

```
AJC_DLGPROC_DEF(Main);
static  VO      EnableButtons(HWND hDlg, BOOL fbtnOpen, BOOL fbtnStart, BOOL fbtnStop);
static  BOOL    SendNextFile(VO);
static  BOOL    FileOpen(VO);
static  VO      FileSend(VO);
static  BOOL CALLBACK cbFind(UI nest, UTP pPath, UTP pName, UI attrib, UI wtime, UX cbp);

//=============================================================================================================//
//                                                                                                             //
//  W i n M a i n                                                                                              //
//                                                                                                             //
//=============================================================================================================//
int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
{
    MSG     msg;

    hInst = hInstance;

    //----- メイン・ダイアログオープン -----------------//
    hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_DLGMAIN), NULL, AJC_DLGPROC_NAME(Main));
    ShowWindow(hDlgMain, SW_SHOW);

    //----- メッセージループ ---------------------------//
    while (GetMessage(&msg, NULL, 0, 0)) {
        do {
            if (IsDialogMessage(hDlgMain, &msg)) break;
            TranslateMessage(&msg);
            DispatchMessage (&msg);
        } while (0);
    }

    return (int)msg.wParam ;
}
//=============================================================================================================//
//                                                                                                             //
//  ダイアログ・プロシージャ                                                                                   //
//                                                                                                             //
//=============================================================================================================//
//----- ダイアログ初期化 -----------------------------------------------------------------------------------//
AJC_DLGPROC(Main, WM_INITDIALOG )
{
    hDlgMain   = hDlg;
    hVthLog    = GetDlgItem(hDlg, IDC_VTH);

    //  ウインド位置ロード
    AjcLoadWndPos(hDlg, NULL);

    hSct = AjcSctCreate();                                          //  インスタンス生成
    AjcSctSetChunkMode(hSct, AJCSCT_CM_BIN);                        //  モード設定
    AjcSctSetEvtMask(hSct, AJCSCT_EV_CONNECT | AJCSCT_EV_DISCONNECT |       //  イベントマスク設定
                           AJCSCT_EV_CNFAIL  | AJCSCT_EV_RXPKT | AJCSCT_EV_TXEMPTY);
    //  ビットマップ表示
    AjcChangeBitmapColor((hBmpProgImage = LoadBitmap(hInst, MAKEINTRESOURCE(IDB_PROGIMAGE ))),
                                                RGB(255, 255, 255), GetSysColor(COLOR_BTNFACE));
    SendDlgItemMessage(hDlg, IDC_PIC, STM_SETIMAGE, IMAGE_BITMAP, (LPARAM)hBmpProgImage);
    //  キュー生成
    hVQue = AjcVQueCreate(0, NULL);
    //  ダイアログ項目の関連付け
    MAjcSetDlgPropStr (hDlg, IDC_TXT_SERV     , TEXT("127.0.0.1" ), NULL , 0);
    MAjcSetDlgPropStr (hDlg, IDC_TXT_PORT     , TEXT("14238"     ), NULL , 0);
    MAjcSetDlgPropStr (hDlg, IDC_TXT_INPPATH  , TEXT(""          ), NULL , 0);
    MAjcSetDlgPropStr (hDlg, IDC_TXT_OUTPATH  , TEXT(""          ), NULL , 0);
    AjcLoadDlgProfiles(hDlg, TEXT("DlgSetting"));

    return TRUE;
}
//----- ウインド破棄 ---------------------------------------------------------------------------------------//
AJC_DLGPROC(Main, WM_DESTROY    )
{
    //  ウインド位置セーブ
    AjcSaveWndPos(hDlg, NULL);
    //  ダイアログ項目のセーブ
    AjcSaveDlgProfiles(hDlg, TEXT("DlgSetting"));
    AjcReleaseDlgProps(hDlg);
    //  リソース解放
    if (hBmpProgImage != NULL) DeleteObject(hBmpProgImage);
    if (hSct          != NULL) AjcSctDelete(hSct);
    if (hVQue         != NULL) AjcVQueDelete(hVQue);
    //  プログラム終了
    PostQuitMessage(0);
```

```
118 :     return TRUE;
119 : }
120 : //----- ＳＣＭイベント通知 ----------------------------------------------------------------------------//
121 : AJC_DLGPROC(Main, WM_SCTEVENT   )
122 : {
123 :     static  UI      TotalBytes = 0;
124 :     VOP     pDat;
125 :     UI      len, param;
126 :
127 :     AjcSctGetEventData(hSct, lParam, &pDat, &len, &param);              //  イベントデータ取得
128 :     if(wParam & AJCSCT_EV_RXCHUNK ) {                                   //  チャンクデータ受信通知？
129 :
130 :     }
131 :     if (wParam & AJCSCT_EV_CONNECT) {                                   //  ●接続通知？
132 :         //  接続状態の旨、フラグ設定
133 :         fConnect = TRUE;
134 :         //  ボタンフェース変更
135 :         AjcSetDlgItemStr(hDlg, IDC_CMD_CONNECT, TEXT("切 断"));
136 :         AjcVthPrintF(hVthLog, TEXT("サーバと接続しました¥n"));
137 :         //  切断, 開始ボタン許可, ／中止ボタン禁止
138 :         EnableButtons(hDlg, TRUE, TRUE, FALSE);
139 :     }
140 :     if(wParam & AJCSCT_EV_DISCONNECT) {                                 //  ●切断通知？
141 :         fConnect = FALSE;
142 :         if (fSendBusy) {
143 :             SendMessage(hDlg, WM_COMMAND, MAKELONG(IDC_CMD_STOP, BN_CLICKED), (LPARAM)hDlg);
144 :         }
145 :         AjcSetDlgItemStr(hDlg, IDC_CMD_CONNECT, TEXT("接 続"));
146 :         AjcVthPrintF(hVthLog, TEXT("回線を切断しました。¥n"));
147 :         //  接続ボタン許可, 開始／中止ボタン禁止
148 :         EnableButtons(hDlg, TRUE, FALSE, FALSE);
149 :     }
150 :     if(wParam & AJCSCT_EV_CNFAIL) {                                     //  ●接続失敗通知？
151 :         AjcVthPrintF(hVthLog, TEXT("接続を失敗しました。¥n"));
152 :         //  接続ボタン許可, 開始／中止ボタン禁止
153 :         EnableButtons(hDlg, TRUE, FALSE, FALSE);
154 :     }
155 :     if(wParam & AJCSCT_EV_RXPKT ) {                                     //  ●パケット受信通知？
156 :         //  空パケット（ファイル終端）以外ならば、受信データをファイルへ出力
157 :         if (len != 0) {
158 :             if (fhOut != INVALID_HANDLE_VALUE) {
159 :                 UL  bytes;
160 :                 WriteFile(fhOut, pDat, len, &bytes, NULL);
161 :             }
162 :         }
163 :         //  空パケット（ファイル終端）ならば、次のファイル送信へ・・
164 :         else {
165 :             //  出力ファイルクローズ
166 :             if (fhOut != INVALID_HANDLE_VALUE) {CloseHandle(fhOut); fhOut = INVALID_HANDLE_VALUE;}
167 :             //  ファイルコンペア
168 :             if (AjcFileCompare(InpFilePath, OutFilePath)) {
169 :                 AjcVthPrintF(hVthLog, TEXT("OK (Match)¥n"));
170 :             }
171 :             else {
172 :                 Unmatch++;
173 :                 AjcVthPrintF(hVthLog, TEXT("¥x1B[31mNG (Unmatch)¥x1B[0m¥n"));
174 :             }
175 :             //  次のファイル送信開始
176 :             SendNextFile();
177 :         }
178 :     }
179 :     if (wParam & AJCSCT_EV_TXEMPTY) {                                   //  ●送信完了通知
180 :         //  次のファイルデータ送信
181 :         if (fhInp != INVALID_HANDLE_VALUE) {
182 :             FileSend();
183 :         }
184 :     }
185 :
186 :     AjcSctRelEventData(hSct, lParam);                                   //  イベントデータ開放
187 :
188 :     return TRUE;
189 : }
190 : //----- キャンセル ------------------------------------------------------------------------------------//
191 : AJC_DLGPROC(Main, IDCANCEL      )
192 : {
193 :     DestroyWindow(hDlg);
194 :     return TRUE;
195 : }
196 : //----- オープンボタン --------------------------------------------------------------------------------//
197 : AJC_DLGPROC(Main, IDC_CMD_CONNECT)
```

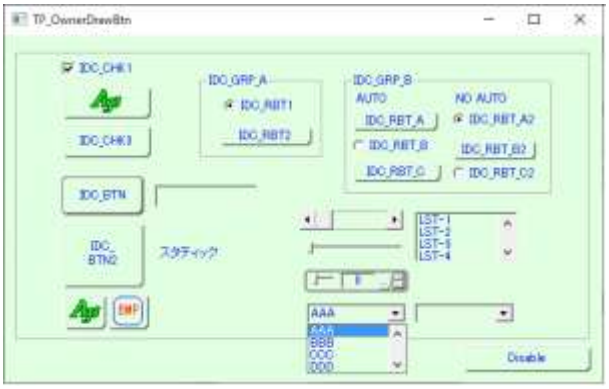
```
198 : {
199 :     if (HIWORD(wParam) == BN_CLICKED) {
200 :         if (fConnect) {
201 :             AjcSctDisconnect(hSct);
202 :         }
203 :         else {
204 :             UT  serv[256];
205 :             UT  port[256];
206 :             AjcGetDlgItemStr(hDlg, IDC_TXT_SERV, serv, AJCTSIZE(serv));
207 :             AjcGetDlgItemStr(hDlg, IDC_TXT_PORT, port, AJCTSIZE(serv));
208 :             AjcSctConnect(hSct, serv, port, AF_INET, hDlg, WM_SCTEVENT);
209 :             EnableButtons(hDlg, FALSE, FALSE, FALSE);
210 :         }
211 :     }
212 :     return TRUE;
213 : }
214 : //----- 開始ボタン ----------------------------------------------------------------------------------------//
215 : AJC_DLGPROC(Main, IDC_CMD_START )
216 : {
217 :     if (HIWORD(wParam) == BN_CLICKED) {
218 :         //  入出力パス設定
219 :         AjcGetDlgItemStr(hDlg, IDC_TXT_INPPATH, InpDir, MAX_PATH);
220 :         AjcGetDlgItemStr(hDlg, IDC_TXT_OUTPATH, OutDir, MAX_PATH);
221 :         if (InpDir[0] != 0 && OutDir[0] != 0) {
222 :             if (MAjcStrICmp(InpDir, OutDir) != 0) {
223 :                 //  入出力パスリストクリアー
224 :                 AjcVQuePurge(hVQue);
225 :                 //  ファイル検索 (入出力パスリスト作成)
226 :                 ListCount = 0;
227 :                 Unmatch = 0;
228 :                 AjcSearchFilesEx(InpDir, TEXT("*.*"), TRUE, FALSE, 0, cbFind);
229 :                 //  フォルダ構造コピー
230 :                 if (ListCount != 0) {
231 :                     UT      txt[MAX_PATH * 64];
232 :                     //  出力フォルダ下の全ファイル削除
233 :                     AjcSnPrintF(txt, AJCTSIZE(txt), TEXT("%s 下の全ファイルを削除します。よろしいですか？"), OutDir);
234 :                     if (MessageBox(hDlg, txt, TEXT("SW_SockClient4"), MB_YESNO) == IDYES) {
235 :                         //  ファイル番号クリアー
236 :                         ListCount = 0;
237 :                         //  バイトカウンタクリアー
238 :                         AjcSepDlgItemUInt(hDlg, IDC_LBL_BYTES, BytesCount = 0);
239 :                         //  ログクリアー
240 :                         AjcVthClear(hVthLog);
241 :                         //  ボタングレー化
242 :                         EnableButtons(hDlg, FALSE, FALSE, TRUE);
243 :                         //  フォルダ下クリアー
244 :                         AjcCleanFolder(OutDir, 0, NULL);
245 :                         //  フォルダ構造コピー
246 :                         AjcCopyFolderStruct(InpDir, OutDir, AJCFSC_FORCETIME);
247 :                         //  ファイルをオープンし送信開始
248 :                         SendNextFile();
249 :                     }
250 :                 } else AjcVthPutText(hVthLog, TEXT("¥x1B[31m 入力フォルダ下にファイルがありません。"), -1);
251 :
252 :             } else AjcVthPutText(hVthLog,TEXT("¥x1B[31m 入出力フォルダが同じです。"), -1);
253 :
254 :         } else AjcVthPutText(hVthLog, TEXT("¥x1B[31m 入出力フォルダが設定されていません。"), -1);
255 :     }
256 :     return TRUE;
257 : }
258 : //----- 中止ボタン ----------------------------------------------------------------------------------------//
259 : AJC_DLGPROC(Main, IDC_CMD_STOP  )
260 : {
261 :     if (HIWORD(wParam) == BN_CLICKED) {
262 :         //  送信中フラグクリアー
263 :         fSendBusy = FALSE;
264 :         //  ポートクローズ
265 :         AjcSctDisconnect(hSct);
266 :         //  ファイルクローズ
267 :         if (fhInp != INVALID_HANDLE_VALUE) {CloseHandle(fhInp); fhInp = INVALID_HANDLE_VALUE;}
268 :         if (fhOut != INVALID_HANDLE_VALUE) {CloseHandle(fhOut); fhOut = INVALID_HANDLE_VALUE;}
269 :         //  ログメッセージ
270 :         AjcVthPrintF(hVthLog, TEXT("¥n¥n--- ループバックテストをキャンセルしました ---¥n"));
271 :     }
272 :     return TRUE;
273 : }
274 : //----- 入力フォルダ設定ボタン ----------------------------------------------------------------------------//
275 : AJC_DLGPROC(Main, IDC_CMD_INPPATH)
276 : {
277 :     if (HIWORD(wParam) == BN_CLICKED) {
```

```
278 :         UT  path[MAX_PATH];
279 :         AjcGetDlgItemStr(hDlg, IDC_TXT_INPPATH, path, MAX_PATH);
280 :         if (AjcGetFolderName(hDlg, TEXT("入力フォルダ"), path, path, TRUE)) {
281 :             AjcSetDlgItemStr(hDlg, IDC_TXT_INPPATH, path);
282 :         }
283 :     }
284 :     return TRUE;
285 : }
286 : //----- 出力フォルダ設定ボタン ----------------------------------------------------------------------------------//
287 : AJC_DLGPROC(Main, IDC_CMD_OUTPATH)
288 : {
289 :     if (HIWORD(wParam) == BN_CLICKED) {
290 :         UT  path[MAX_PATH];
291 :         AjcGetDlgItemStr(hDlg, IDC_TXT_INPPATH, path, MAX_PATH);
292 :         if (AjcGetFolderName(hDlg, TEXT("出力フォルダ"), path, path, TRUE)) {
293 :             AjcSetDlgItemStr(hDlg, IDC_TXT_OUTPATH, path);
294 :         }
295 :     }
296 :     return TRUE;
297 : }
298 : //-------------------------------------------------------------------------------------------------------------//
299 : AJC_DLGMAP_DEF(Main)
300 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG  )
301 :     AJC_DLGMAP_MSG(Main, WM_DESTROY     )
302 :     AJC_DLGMAP_MSG(Main, WM_SCTEVENT    )
303 :     AJC_DLGMAP_CMD(Main, IDCANCEL       )
304 :     AJC_DLGMAP_CMD(Main, IDC_CMD_CONNECT)
305 :     AJC_DLGMAP_CMD(Main, IDC_CMD_START  )
306 :     AJC_DLGMAP_CMD(Main, IDC_CMD_STOP   )
307 :     AJC_DLGMAP_CMD(Main, IDC_CMD_INPPATH)
308 :     AJC_DLGMAP_CMD(Main, IDC_CMD_OUTPATH)
309 : AJC_DLGMAP_END
310 :
311 : //-------------------------------------------------------------------------------------------------------------//
312 : //  ファイル検索通知                                                                                           //
313 : //-------------------------------------------------------------------------------------------------------------//
314 : static  BOOL CALLBACK cbFind(UI nest, UTP pPath, UTP pName, UI attrib, UI wtime, UX cbp)
315 : {
316 :     UT      out  [MAX_PATH];
317 :     UT      inout[MAX_PATH * 2 + 16];
318 :
319 :     if (!(attrib & _A_SUBDIR)) {
320 :         ListCount++;
321 :         //  入出力ファイルパス設定 (セミコロン(;)で区切る)
322 :         MAjcStrCpy(out, MAX_PATH, OutDir);
323 :         AjcPathCat(out, pPath + MAjcStrLen(InpDir), MAX_PATH);
324 :         AjcSnPrintF(inout, AJCTSIZE(inout), TEXT("%s:%s"), pPath, out);
325 :         AjcVQueEnque(hVQue, (C_VOP)inout, ((UI)MAjcStrLen(inout) + 1) * sizeof(UT));
326 :     }
327 :     return TRUE;
328 : }
329 : //-------------------------------------------------------------------------------------------------------------//
330 : //  次のファイル送信                                                                                           //
331 : //-------------------------------------------------------------------------------------------------------------//
332 : static  BOOL    SendNextFile(VO)
333 : {
334 :     BOOL    rc = FALSE;
335 :
336 :     if (FileOpen()) {
337 :         fSendBusy = TRUE;
338 :         FileSend();
339 :         rc = TRUE;
340 :     }
341 :     else {
342 :         fSendBusy = FALSE;
343 :         AjcSctDisconnect(hSct);
344 :         AjcVthPrintF(hVthLog, TEXT("¥n--- ループバックテストを終了しました ---¥n"));
345 :         AjcVthPrintF(hVthLog, TEXT("    ループバック不一致数 : %d / %d[files]¥n"), Unmatch, ListCount);
346 :     }
347 :     return rc;
348 : }
349 : //-------------------------------------------------------------------------------------------------------------//
350 : //  ファイルオープン (TRUE:ゼロサイズ以外のファイルオープン, FALSE:ファイルなし)                               //
351 : //-------------------------------------------------------------------------------------------------------------//
352 : static  BOOL    FileOpen(VO)
353 : {
354 :     BOOL    rc = FALSE;
355 :     UTP     pInp;
356 :     UTP     pOut;
357 :     UT      inout[MAX_PATH * 2 + 16];
```

```
358 :
359 :     while (AjcVQueDeque(hVQue, (VOP)inout, sizeof(inout))) {
360 :         ListCount++;
361 :         //  入出力パス名設定
362 :         pInp = MAjcStrTok(inout, TEXT(";")); if (pInp != NULL) MAjcStrCpy(InpFilePath, MAX_PATH, pInp);
363 :         pOut = MAjcStrTok(NULL , TEXT(";")); if (pOut != NULL) MAjcStrCpy(OutFilePath, MAX_PATH, pOut);
364 :         //  ログ表示
365 :         AjcVthPrintF(hVthLog, TEXT("¥n%3d : %s¥n"), ListCount, pInp);
366 :         AjcVthPrintF(hVthLog, TEXT("    -> %s - "), pOut);
367 :         //  入出力ファイルオープン
368 :         fhInp = CreateFile(pInp, GENERIC_READ , 0, NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
369 :         fhOut = CreateFile(pOut, GENERIC_WRITE, 0, NULL, CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
370 :         if (fhInp != INVALID_HANDLE_VALUE && fhOut != INVALID_HANDLE_VALUE) {
371 :             //  ファイルサイズ設定
372 :             FileSize = AjcGetFileSize(pInp);
373 :             //  ゼロサイズならば、ファイルクローズ
374 :             if (FileSize == 0) {
375 :                 AjcVthPrintF(hVthLog, TEXT("OK (Zero size)¥n"));
376 :                 CloseHandle(fhInp); fhInp = INVALID_HANDLE_VALUE;
377 :                 CloseHandle(fhOut); fhOut = INVALID_HANDLE_VALUE;
378 :             }
379 :             //  ゼロサイズ以外ならば、ファールオープンした旨を返す
380 :             else {
381 :                 rc = TRUE;
382 :                 break;
383 :             }
384 :         }
385 :         else {
386 :             Unmatch++;
387 :             AjcVthPrintF(hVthLog, TEXT("¥x1B[31mNG (Open/Creation failure)¥x1B[0m¥n"));
388 :             if (fhInp != INVALID_HANDLE_VALUE) {CloseHandle(fhInp); fhInp = INVALID_HANDLE_VALUE;}
389 :             if (fhOut != INVALID_HANDLE_VALUE) {CloseHandle(fhOut); fhOut = INVALID_HANDLE_VALUE;}
390 :         }
391 :     }
392 :     return rc;
393 : }
394 : //-----------------------------------------------------------------------------------------------------------//
395 : //  ファイル送信                                                                                             //
396 : //-----------------------------------------------------------------------------------------------------------//
397 : static  VO  FileSend()
398 : {
399 :     UL      bytes;
400 :     UB      buf[1024];
401 :
402 :     //  バッファサイズを超える残データ有りならば、データ送信し、バイト数減算
403 :     if (FileSize > sizeof buf) {
404 :         ReadFile(fhInp, buf, sizeof buf, &bytes, NULL);
405 :         BytesCount += AjcSctSendPacket(hSct, buf, sizeof buf);
406 :         FileSize   -= sizeof buf;
407 :     }
408 :     //  バッファサイズ以下の残データ有りならば、ファイル末尾データ送信
409 :     else if (FileSize != 0) {
410 :         ReadFile(fhInp, buf, (UI)FileSize, &bytes, NULL);
411 :         BytesCount += AjcSctSendPacket(hSct, buf, (UI)FileSize);
412 :         FileSize = 0;
413 :     }
414 :     //  残データ無しならば、ファイル終端を示す空パケットを送信
415 :     else {
416 :         //  空パケット送信
417 :         BytesCount += AjcSctSendPacket(hSct, buf, 0);
418 :         //  入力ファイルクローズ
419 :         if (fhInp != INVALID_HANDLE_VALUE) {CloseHandle(fhInp); fhInp = INVALID_HANDLE_VALUE;}
420 :     }
421 :     //  送信バイトカウント表示
422 :     AjcSepDlgItemUInt(hDlgMain, IDC_LBL_BYTES, BytesCount);
423 : }
424 : //-----------------------------------------------------------------------------------------------------------//
425 : //  ボタン群許可／禁止                                                                                       //
426 : //-----------------------------------------------------------------------------------------------------------//
427 : static  VO  EnableButtons(HWND hDlg, BOOL fbtnOpen, BOOL fbtnStart, BOOL fbtnStop)
428 : {
429 :     AjcEnableDlgItem(hDlg, IDC_CMD_OPEN , fbtnOpen );
430 :     AjcEnableDlgItem(hDlg, IDC_CMD_START, fbtnStart);
431 :     AjcEnableDlgItem(hDlg, IDC_CMD_STOP , fbtnStop );
```

16. ダイアログボックス、コントロール群とメニューのカラー設定

ダイアログボックスの背景色や、ダイアロブボックス内コントロールの背景色と文字色を設定します。
デフォルトでは、背景＝薄いグリーン，文字色＝青で、以下のようなイメージで表示します。



上記のようなカラー設定されたダイアログを表示する場合は、プロセスの最初（WinMain()等）で以下のコードを実行します。

```
  AjcDgcSetup();                    // あるいは、AjcDgcSetupEx(・・・);
{ AjcDgcSetColors(・・・);         } // あるいは、AjcDgcSetColorsEx(・・・);
  AjcDgcSetColorInfo(・・・);     }
```

ダイアログプロシージャを「AJC_DLGMAP_DEF()」～「AJC_DLGMAP_END」マクロで記述している場合は、これだけで上記のようなカラー設定されたダイアログが表示されます。

AjcDgcSetColors[Ex]()；は、背景色やテキスト色を変更する場合に実行します。（実行しない場合は、既定の色となります）
より詳細な表示色を設定するには、AjcDgcSetColorInfo()を実行します。

ダイアログプロシージャを「AJC_**DLG**MAP_DEF()」～「AJC_**DLGM**AP_END」マクロで記述している場合、「AJC_**DLG**MAP_DEF()」マクロ内で「WM_INITDIALOG」の最後の処理として、「AjcDgcInitColorDisplayEx(hDlg, AjcDgcGetBmpTransColor());」が自動的に実行されます。
但し、ビットマップの透明化を行わないか、透明化の色を指定したい場合は、AjcDgcSetup()の後に、AjcDgcSetBmpTransColor()を実行してください。

```
ex.     int WINAPI AjcWinMain(・・・)
        {
            AjcDgcSetup();                          //  ダイアログカラー設定をセットアップ
            AjcDgcSetBmpTransColor(RGB(0, 0, 0));   //  ビットマップの黒色を透明化する（白色の場合は実行不要）
            ・・・
        }
        AJC_DLGPROC(Main, WM_INITDIALOG)
        {
            ・・・
            AjcDgcInitColorDisplayEx(hDlg, AjcDgcGetBmpTransColor());  // ウインドウコントロール群をモダン化し、背景、影等を強化
        }
        AJC_DLGMAP_DEF(Main)
            AJC_DLGMAP_MSG(Main, WM_INITDIALOG)
            ・・・
        AJC_DLGMAP_END
```

AJC_**DLG**MAP_DEF()／AJC_**WND**MAP_DEF()を使用した場合、暗黙的に実行される

ウインドプロシージャの場合も同様に、「AJC_**WND**MAP_DEF()」～「AJC_**WND**MAP_END」の「AJC_WNDMAP_DEF()」マクロ内で、「WM_CREATE」の最後の処理として「AjcDgcInitColorDisplayEx(hwnd, AjcDgcGetBmpTransColor());」が自動的に実行されます。

メッセージマップを「AJC_**WND**MAP_DEF()」～「AJC_**WND**MAP_END」マクロで記述している場合も、ウインドの表示色が設定されますが、この場合は、コントロール(子ウインド)の背景色や文字色は設定されますが、ウインド自体の背景色は自身で描画しなければなりません。

カラー設定するコントロール

ダイアログボックスと、ダイアログボックスやウインドに属する以下のコントロール（子ウインド）が カラー設定されます。

#	名称	クラス名
1	ボタン	Button
2	チェックボックス	
3	ラジオボタン	
4	グループボックス	
5	スタティック	Static
6	テキストボックス	Edit
7	スクロールバー	ScrollBar
8	リストボックス	ListBox
9	コンボボックス	ComboBox

#	名称	クラス名
10	ＩＰアドレス入力	SysIPAddress32
11	ステータスバー	msctls_statusbar32
12	プログレスバー	msctls_progress32
13	タブ ※1	SysTabControl32
14	日時選択 ※2	SysDateTimePick32
15	カレンダ	SysMonthCal32
16	リストビュー	SysListView32
17	ツリービュー	SysTreeView32
18	数値入力	AjcCtrlInpVal

※１：タブコントロールは「TCS_OWNERDRAWFIXED」スタイルを設定してください。但し、WM_DRAWITEM メッセージは内部で処理されます。
※２：コントロール自体（日時編集部分）は、背景色のみ設定可能（テキスト色はオリジナル（黒色）のままとなる）

個別コントロールの表示色設定

個別に、コントロールの色を設定するには、ダイアログ／ウインドで AJCM_CTLCOLOR_BTN / AJCM_CTLCOLOR_ANY メッセージを処理します。
AJCM_CTLCOLOR_BTN は、BS_PUSHBUTTON/BS_DEFPUSHBUTTON あるいは、BS_PUSHLIKE スタイルのチェックボックスやラジオボタンの場合のメッセージで、その他のコントロールの場合は、AJCM_CTLCOLOR_ANY メッセージを処理します。
これらのメッセージは、wParam に AJCDGCCTLCOLOR 構造体へのポインタを、lParam にダイアログ／コントロールのハンドルを通知します。
但し、コンボボックスの場合は、コンボボックス内のリストボックスのハンドルが通知されますので、lParam とコンボボックス内のリストボックスのハンドルを比較します（ex. if ((HWND)lParam == AjcGetDlgItemCboListHandle(hDlg, IDC_CBO_*XXXXX*)) {・・・} ）
（AJCM_CTLCOLOR_{BTN/ANY}メッセージは、特定の固定的なメッセージコードではなく、RegisterWindowMessage()で取得する動的な値です）

コントロールの背景色を変更する場合は AJCDGCCTLCOLOR 構造体の hBrush に背景色のブラシハンドルを、テキスト色を変更する場合は TextColor を、テキスト背景色を変更する場合は、BkColor を変更します。

AJCM_CTLCOLOR_BTN メッセージは、AJC_DLGMAP_**DGB**／AJC_WNDMAP_**DGB** マクロでメッセージマップを記述します。
AJCM_CTLCOLOR_ANY メッセージは、AJC_DLGMAP_**DGC**／AJC_WNDMAP_**DGC** マクロでメッセージマップを記述します。

AJC_DLGMAP_**DGB**／AJC_WNDMAP_**DGB** マクロは、RegisterWindowMessage(TEXT("_AJCM_CTLCOLOR_BTN_"))で取得した動的メッセージを受信します。
AJC_DLGMAP_**DGC**／AJC_WNDMAP_**DGC** マクロは、RegisterWindowMessage(TEXT("_AJCM_CTLCOLOR_ANY_"))で取得した動的メッセージを受信します。

例えば、ダイアログで IDC_BUTTON(プッシュボタン)と IDC_TXT(Edit コントロール)だけの背景をクリーム色（RGB(255, 255, 225)）、テキスト色を赤色（RGB(255, 0, 0)）にするには、以下のように記述します。

```
static hBruCtrl = NULL;

AJC_DLGPROC(Main, WM_INITDIALOG  )
{
    //  背景色（クリーム色）のブラシ生成
    hBruCtrl = CreateSolidBrush(RGB(255, 255, 225));
    return TRUE;
}

AJC_DLGPROC(Main, AJCM_CTLCOLOR_BTN )
{
    PAJCDGCCTLCOLOR pCC = (PAJCDGCCTLCOLOR)wParam;
    //  IDC_BUTTON の背景色，文字色設定
    if ((HWND)lParam == GetDlgItem(hDlg, IDC_BUTTON)) {
        pCC->hBrush    = hBruCtrl;
        pCC->TextColor = RGB(255, 0, 0); // 文字色＝赤
        pCC->BkColor   = -1;             // 文字背景＝透明
    }
    return TRUE;
}
```

```
AJC_DLGPROC(Main, AJCM_CTLCOLOR_ANY )
{
    PAJCDGCCTLCOLOR pCC = (PAJCDGCCTLCOLOR)wParam;
    //  IDC_TXT の背景色，文字色設定
    if ((HWND)lParam == GetDlgItem(hDlg, IDC_TXT)) {
        pCC->hBrush    = hBruCtrl;
        pCC->TextColor = RGB(255, 0, 0);      // 文字色＝赤
        pCC->BkColor   = RGB(255, 255, 225);  // 文字背景＝クリーム色
    }
    return TRUE;
}
//  メッセージマップ
AJC_DLGMAP_DEF(Main)
    AJC_DLGMAP_MSG(Main, WM_INITDIALOG     )
            :
            :
    AJC_DLGMAP_DGB(Main, AJCM_CTLCOLOR_BTN )
    AJC_DLGMAP_DGC(Main, AJCM_CTLCOLOR_ANY )
AJC_DLGMAP_END
```

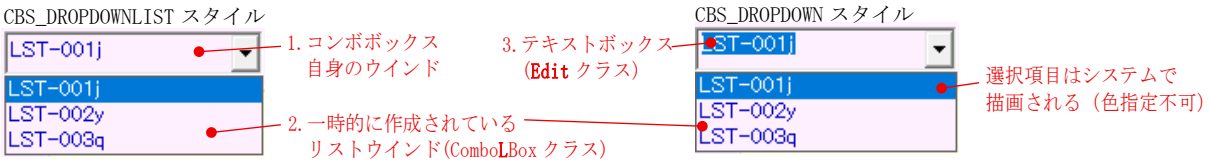
※#define 等で「AJCM_CTLCOLOR_BTN」「AJCM_CTLCOLOR_ANY」は定義不要です。

表示イメージ(IDC_BTN)：

表示イメージ(IDC_TXT)：

個別コントロールの表示色設定（コンボボックス）

コンボボックスの場合、AJCM_CTLCOLOR_ANY メッセージで、コンボボックス自身のウインドハンドルだけでなく、以下の３つのパーツのウインドハンドルに応答する必要があります。



下記例は、上記３つのウインドすべてに描画色を設定します。

```
if ((HWND)lParam == GetDlgItem (hDlg, IDC_CBO_XXXXX)                 ||  // コンボボックス自身のウインド
    (HWND)lParam == AjcGetDlgItemCboEditHandle(hDlg, IDC_CBO_XXXXX) ||  // コンボボックス内のテキストボックス
    (HWND)lParam == AjcGetDlgItemCboListHandle(hDlg, IDC_CBO_XXXXX)) {  // コンボボックス内のリストボックス
   pCC->hBrush    = hBruCtrl;       // 背景ブラシ
   pCC->TextColor = RGB(255, 0, 0); // 文字色＝赤
   pCC->BkColor   = -1;             // 文字背景＝透明
}
```

リストボックス、コンボボックス リスト項目の色設定／自由描画

リスト／コンボボックスのリスト項目の色設定／自由描画するには、当該コントロールにオーナー描画スタイルを設定します。
オーナー描画スタイルを設定した場合は、当該リストボックス／コンボボックスへの AJCM_CTLCOLOR_ANY メッセージは通知されません。

オーナー描画スタイル

リストボックス	コンボボックス	内容
LBS_OWNERDRAWFIXED	CBS_OWNERDRAWFIXED	リスト ボックスの所有者がその内容を描画し、リスト ボックス内の項目がすべて同じ高さであることを指定します。
LBS_OWNERDRAWVARIABLE	CBS_OWNERDRAWVARIABLE	リスト ボックスの所有者がその内容を描画し、リスト ボックス内の項目の高さが可変であることを指定します。

拡張リストボックス（AjcCtrlListBox クラス）の場合は、AJCLBXN_LISTSTYLE に応答し、内部のリストボックスへスタイルを設定する必要があります。

```
ex.   AJC_DLGPROC(Main, IDC_LBX_XXXXX     )
      {
          if (HIWORD(wParam) == AJCLBXN_LISTSTYLE) {
              PAJCLBXLSTSTY   p = (PAJCLBXLSTSTY)lParam;
              p->sty |= LBS_OWNERDRAWFIXED;
          }
          return TRUE;
      }
```

オーナー描画スタイルを設定したら、ダイアログ／ウインドで AJCM_CTL**COLOR**_LSI / AJCM_CTL**DRAW**_LSI メッセージを処理します。
（AJCM_CTLCOLOR_LSI / AJCM_CTLDRAW_LSI メッセージは、特定の固定的なメッセージコードではなく、RegisterWindowMessage()で取得する動的な値です）

AJCM_CTLDRAW_LSI メッセージは、リスト項目を自由に描画するためのメッセージで、wParam に AJCDGCCTLCOLOR 構造体へのポインタを、lParam に AJCDGCDRAWITEM 構造体へのポインタを通知します。
AJCM_CTLDRAW_LSI メッセージに応答し、リスト項目を自由に描画することができます。
AJCM_CTLDRAW_LSI メッセージに応答し、リスト項目を描画した場合は、AJCDGCCTLCOLOR 構造体のメンバ **fDrawn** に TRUE を設定します。

AJCM_CTLCOLOR_LSI メッセージは、リスト項目の背景色、テキストの文字色／文字背景色を設定するためのメッセージで、wParam に AJCDGCCTLCOLOR 構造体へのポインタを、lParam に AJCDGCDRAWITEM 構造体へのポインタを通知します。
AJCM_CTLCOLOR_LSI メッセージは、AJCM_CTLDRAW_LSI メッセージで、**fDrawn** に TRUE を設定した場合は、発行されません。
AJCDGCCTLCOLOR 構造体のメンバ変数を変更することにより、リスト項目の描画色を設定できます。

AJCM_CTLCOLOR_LSI メッセージは、AJC_DLGMAP_**CLI**／AJC_WNDMAP_**CLI** マクロでメッセージマップを記述します。（ **CLI** ： **C**olor **L**ist **I**tem）
AJCM_CTLDRAW_LSI メッセージは、AJC_DLGMAP_**DLI**／AJC_WNDMAP_**DLI** マクロでメッセージマップを記述します。（ **DLI** ： **D**raw **L**ist **I**tem ）

AJC_DLGMAP_**CLI**／AJC_WNDMAP_**CLI** マクロは、RegisterWindowMessage(TEXT("_AJCM_CTLCOLOR_LSI_"))で取得した動的メッセージを受信します。
AJC_DLGMAP_**DLI**／AJC_WNDMAP_**DLI** マクロは、RegisterWindowMessage(TEXT("_AJCM_CTLDRAW_LSI_"))で取得した動的メッセージを受信します。

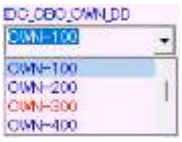
下記例は、ダイアログで コンボボックス(IDC_CBO_OWN_DD)の３番目(ix=2)の項目を赤色で表示し、リストボックス(IDC_LST_OWN)の３番目の項目をグラフィク描画（テキストの上に外枠と、たすき掛け線を描画）します。

```
AJC_DLGPROC(Main, AJCM_CTLCOLOR_LSI )
{
    PAJCDGCCTLCOLOR  pCC  = (PAJCDGCCTLCOLOR )wParam;
    PAJCDGCDRAWITEM  pDis = (PAJCDGCDRAWITEM)lParam

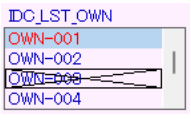
    //  ３番目のコンボボックス内リスト項目の文字色設定
    if (pDis->id == IDC_CBO_OWN_DD && pDis->ix == 2) {
        pCC->TextColor = RGB(255, 0, 0); // 文字色＝赤
    }
    return TRUE;
}
AJC_DLGPROC(Main, AJCM_CTLDRAW_LSI)
{
    PAJCDGCCTLCOLOR pCC  = (PAJCDGCCTLCOLOR)wParam;
    PAJCDGCDRAWITEM pDis = (PAJCDGCDRAWITEM)lParam
    UT  txt[256];

    //  ３番目のリスト項目描画
    if (pDis->id == IDC_LST_OWN  && pDis->ix == 2) {
        //  ＤＣオブジェクト設定
        HPEN    hPen = (HPEN  )SelectObject(pDis->hDC, GetStockObject(BLACK_PEN));
        HBRUSH  hBru = (HBRUSH)SelectObject(pDis->hDC, pCC->hBrush);
        //  項目塗りつぶしと外枠描画
        Rectangle(pDis->hDC, pDis->rect.left  , pDis->rect.top,
                             pDis->rect.right , pDis->rect.bottom);
        //  テキスト描画
        SendMessage(pDis->hWnd, LB_GETTEXT, 2,(LPARAM) txt);
        SetBkMode(pDis->hDC, TRANSPARENT);
        TextOut(pDis->hDC, pDis->rect.left + 3, pDis->rect.top + 1,
                                        txt, (UI)MAjcStrLen(txt));
        //  たすき掛け斜線描画
        MoveToEx(pDis->hDC ,pDis->rect.left ,pDis->rect.bottom - 1, NULL);
        LineTo  (pDis->hDC, pDis->rect.right,pDis->rect.top);
        MoveToEx(pDis->hDC ,pDis->rect.left ,pDis->rect.top, NULL);
        LineTo  (pDis->hDC, pDis->rect.right,pDis->rect.bottom - 1);
        //  ＤＣオブジェクトを元に戻す
        SelectObject(pDis->hDC, hBru);
        SelectObject(pDis->hDC, hPen);
        pDis->fDrawn = TRUE;    //  描画済フラグ設定
    }
    return TRUE;
}
//  メッセージマップ
AJC_DLGMAP_DEF(Main)
    AJC_DLGMAP_MSG(Main, WM_INITDIALOG     )
            .
            .
    AJC_DLGMAP_CLI(Main, AJCM_CTLCOLOR_LSI )  // #define 棟で、「AJCM_CTLCOLOR_LSI」は定義不要
    AJC_DLGMAP_DLI(Main, AJCM_CTLDRAW_LSI  )  // #define 棟で、「AJCM_CTLDRAW_LSI」は定義不要
AJC_DLGMAP_END
```

表示イメージ(IDC_CBOOWN_DD)：



表示イメージ(IDC_LST_OWN)：



※３番目以外の項目はデフォルト色で描画される（デフォルト色は AjcDgc{Get/Set}ColorInfo 参照）

メニューについて

ウインドのメニュー（メニューバー上のメニュー）は、「AjcDgcSetup()」を実行するだけで、自動的にカラー設定されます。
WindowsＡＰＩの「TrackPopupMenu()」や「TrackPopupMenuEx()」で表示するポップアップメニューは、**「AjcDgcTrackPopupMenu()」**や**「AjcDgcTrackPopupMenuEx()」**に変更する必要があります。(引数は、TrackPopupMenu()やTrackPopupMenuEx()と同じです)

メニュータイプの「MFT_STRING」と「MFT_BITMAP」は、いずれも**「MFT_OWNERDRAW」**を指定します。
メニュースタイルの「MFS_HILITE」(強調表示)と「MFS_DEFAULT」(太字表示)は無視されます。

メニューテキストには太字，斜字，下線，文字色や、文字背景色を指定するエスケープシーケンスを含むことができます。
エスケープシーケンスについては、「テキスト描画」の章を参照してください。

メニューテキスト（文字列は）は、AJCMT()マクロで記述してください。

ex. 例えば、InsertMenu()でメニュー項目を挿入する場合は、以下のようにします。

```
InsertMenu(hPopup, 0, MF_BYPOSITION | MFT_OWNERDRAW | MF_POPUP, (UX)hMTxt, AJCMT("Text Menu"));
```

※ バイト文字（ASCII）モードのアプリの場合、AJCMT()は、文字列の先頭に '¥b'(0x08)を付加します。

AJCMT("ABC") → "¥b" "ABC"

内部的にメニューテキストはUNICODEで処理しますが、内部ではバイト文字(ASCII)か、ワイド文字(UNICODE)か判断できません。
文字列の先頭が '¥b' である場合は、'¥b'をスキップし、後続の文字列をUNICODEへ返還して処理します。

ワイド文字(UNICODE)モードのアプリの場合は、AJCMT()は、文字列の先頭に「L」を付加します。

AJCMT("ABC") → L"ABC"

ワイド文字(UNICODE)モードのアプリの場合は、AJCMT()を使用せずに、「TEXT("ABC")」や「L"ABC"」で記述しても構いません。

表示色の詳細設定

以下のＡＰＩにより、ダイアログ／ウインドやメニューの描画色取得／設定を行ことができます。

#	ＡＰＩ	内容
1	AjcDgcSetColors[Ex]	ウインド，コントロール群とメニューの背景色とテキスト色設定
2	AjcDgcGetColorInfo	描画色の詳細情報取得
3	AjcDgcSetColorInfo	描画色の詳細情報設定

上記ＡＰＩで設定された表示色は、プログラム（プロセス）内で共通の設定となります。

デフォルト・プッシュボタン

AjcDgcInitColorDisplay[Ex]()を実行した場合、デフォルトプッシュボタン（ダイアログ既定のプッシュボタン）は、常にダイアログが生成された時点の BS_DEFPUSHBUTTON スタイルが設定されていたボタンとなります。

通常、ダイアログ上のボタンは、最後に押されたボタン（フォーカスを持つボタン）が BS_DEFPUSHBUTTON スタイルとなり、ボタン群がフォーカスを失った時点で、最初に設定されていた（おそらく、ダイアログが生成された時点の）BS_DEFPUSHBUTTON スタイルのボタンが既定のボタンとなります。
そして、ENTER キーを押すとフォーカスを持つボタンがクリックされます。

AjcDgcInitColorDisplay[Ex]()を実行した場合は、全てのボタンは BS_OWNERDRAW スタイルとなります。
BS_DEFPUSHBUTTON や BS_PUSHBUTTON スタイルのボタンは無くなり（つまり、フォーカスを持つボタンは無くなり）、ダイアログが生成された時点の BS_DEFPUSHBUTTON スタイルのボタンが既定のボタンとなります。
この場合、ENTER キーを押すと、常にダイアログが生成された時点の BS_DEFPUSHBUTTON スタイルのボタンがクリックされます。

尚、スペースキーを押した場合は、(AjcDgcInitColorDisplay[Ex]()を実行しない場合と同様に）フォーカスを持つボタンがクリックされま

ウインドスタイルの取得／設定

サブクラス化されたボタン(BS_PUSHBUTTON, BS_DEFPUSHBUTTON, BS_PUSHLIKE)のスタイルを取得／設定する場合は、以下のマクロで行うようにしてください。

```
MAjcGetWindowLong(hwnd, GWL_STYLE); ---------- ウインドスタイルの取得
MAjcSetWindowLong(hwnd, GWL_STYLE, style); --- ウインドスタイルの設定
```

これらのマクロでは、サブクラス化されたボタンのスタイル(下位 16Bit)を取得／設定する際は、内部的に保持するサブクラス化される前のスタイルを取得／設定します。システムのスタイル(上位 16Bit)はそのまま取得／設定します。
但し、設定できるボタンスタイル(下位 16Bit)は以下に限定されます。（下記以外のスタイルは変更できません）

・BS_TEXT	・BS_LEFT	・BS_TOP	・BS_MULTILINE
・BS_ICON	・BS_RIGHT	・BS_BOTTOM	・BS_NOTIFY
・BS_BITMAP	・BS_CENTER	・BS_VCENTER	・BS_FLAT

ボタン以外のコントロールでは、ウインドスタイル(32Bit)をそのまま取得／設定します。

マルチライン・テキスト

BS_MULTILINE スタイルのボタン(BS_PUSHBUTTON, BS_DEFPUSHBUTTON, BS_PUSHLIKE)は、テキストを複数行で表示できます。
テキストには太字，斜字，下線，文字色や、文字背景色を指定するエスケープシーケンスを含むこともできます。
エスケープシーケンスについては、「テキスト描画」の章を参照してください。

AJCDGCCTLCOLOR 構造体

描画色を指定するための構造体です。

AJCM_CTLCOLOR_BTN, AJCM_CTLCOLOR_ANY, AJCM_CTLCOLOR_LSI や AJCM_CTLDRAW_LSI メッセージの wParam で通知されます。

```
typedef struct {
    HBRUSH      hBrush;         //  背景のブラシハンドル
    COLORREF    TextColor;      //  テキスト色
    COLORREF    BkColor         //  文字背景色 (-1:透明)
} AJCDGCCTLCOLOR, *PAJCDGCCTLCOLOR;
typedef const AJCDGCCTLCOLOR *PCAJCDGCCTLCOLOR;
```

hBrush, TextColor, BkColor には、あらかじめ以下のデフォルト値が設定されています。

メンバ	デフォルト値		備考
	AJCM_CTLCOLOR_BTN メッセージの場合	AJCM_CTLCOLOR_ANY メッセージの場合	
hBrush	背景色のブラシハンドル		ソリッドブラシ
TextColor	文字色		コントロールが無効(Disable)状態の場合は、無視されます。
BkColor	-1 (透明)	文字背景色 ※1	

※1： 一時の Windows バージョンでは、BkColor = -1(透明)を設定しても一部のコントロール (コンボボックス(ドロップダウンリスト)やリストボックス) では、透明設定ができません。従って、これらのコントロールの場合 BkColor に背景ブラシと同じ色を設定してください。(但し、最新の Windows11 では解消の模様)

AJCDGCCTLDRAW 構造体

リストボックスやコンボボックスの特定のリスト項目を描画するための情報です。

AJCM_CTLCOLOR_LSI や AJCM_CTLDRAW_LSI メッセージの lParam で通知されます。

```
typedef struct {
    UI      kind;               //  コントロール種別(ODT_XXXXX)
    UI      id;                 //  コントロールのＩＤ(IDC_XXXXX)
    UI      ix;                 //  リストボックス／コンボボックス項目のインデクス（０～）
    UI      act;                //  必要な描画アクション(ODA_XXXXX)
    UI      state;              //  現在の描画アクションが実行された後のアイテムの表示状態(ODS_XXXXX)
    HWND    hWnd;               //  リストボックス／コンボボックスのウインドハンドル
    HDC     hDC;                //  デバイス コンテキストへのハンドル
    RECT    rect;               //  描画するコントロールの境界を定義する四角形
    UX      data;               //  リストボックス／コンボボックス項目に関連つけられたデータ
    HWND    hLbx;               //  拡張リストボックスコントロールのハンドル
    UI      idLbx;              //  拡張リストボックスコントロールのＩＤ
    BOOL    fDrawn;             //  描画済フラグ (AJCM_CTLDRAW_LSI メッセージ時のみ有効)
} AJCDGCDRAWITEM, *PAJCDGCDRAWITEM;
typedef const AJCDGCDRAWITEM *PCAJCDGCDRAWITEM;
```

hLbx は、リストボックスが、拡張リストボックスコントロールの子ウインドである場合に、拡張リストボックスコントロールのハンドルが設定されます。その他の場合は NULL が設定されます。(hWnd は子ウインドのリストボックスを示します)

この構造体は、ほとんど (hLbx, fDrawn 以外)、システムの DRAWITEMSTRUCT 構造体の内容と同じです。詳細は MSDN を参照してください。

コントロールの種別(kind) – 以下のいずれか

値		意味
ODT_LISTBOX	2	リストボックス
ODT_COMBOBOX	3	コンボボックス

必要な描画アクション(act) – 以下のいずれか

値		意味
ODA_DRAWENTIRE	0x0001	コントロール全体を描画する必要があります。
ODA_FOCUS	0x0004	コントロールが失われたり、キーボードフォーカスが取得されたりしました。 itemState メンバーをチェックして、コントロールにフォーカスがあるかどうかを確認する必要があります。
ODA_SELECT	0x0002	選択状態が変更されました。 itemState メンバーは、選択状態が新しくなったかどうかを判断する場合にチェックします。

現在の描画アクションが実行された後のアイテムの表示状態(state) - 以下の組み合わせ

値		意味
ODS_COMBOBOXEDIT	0x1000	コンボ ボックスの選択フィールド (編集コントロール)
ODS_DEFAULT	0x0020	項目は既定の項目です。
ODS_DISABLED	0x0004	アイテムは無効として描画されます。
ODS_FOCUS	0x0010	項目にキーボード フォーカスがあります。
ODS_HOTLIGHT	0x0040	項目がホット トラッキングされています。つまり、マウスが項目上にあると、項目が強調表示されます。
ODS_SELECTED	0x0001	メニュー項目の状態が選択されています。

描画色の動的な設定と永続化

以下のＡＰＩを使用して、背景色やテキスト色の動的な設定と、永続化ができます。

#	ＡＰＩ	内容
1	AjcDgcSetupEx	セットアップと、プロファイルパスの設定
2	AjcDgcSetColors[Ex]	プロファイルに記録された 背景色、テキスト色の設定
3	AjcDgcSetColorsDlg	ダイアログボックスにより指定された描画色，テキスト色をプロファイルに記録

描画色の動的な設定と永続化に関しては、サンプルプログラム(SW_DlgColor2)を参照してください。

サブクラス化

jcDgcSubclass[Ex]()を実行すると、ダイアログボックス、ウインドやコントロール(子ウインド)がサブクラス化されます。
サブクラス化するコントロールと、処理されるメッセージは以下のとおりです。

◎：インターセプト，○：トラップ，－：何もしない

#	コントロール / メッセージ	DIALOG BOX	WINDOW	ボタン (*3)	ボタン風 Chk, Radio (*4)	Chk, Rdio	IP ADDR	STATUS	PROGRESS	LISTVIEW	TREEVIEW	DATETIMEPICK	MONTHCAL	TAB
1	WM_CTLCOLORDLG	◎	－	－	－	－	－	－	－	－	－	－	－	－
2	WM_CTLCOLORBTN	－	－	－	－	－	－	－	－	－	－	－	－	－
3	WM_CTLCOLORSTATIC	◎	◎	－	－	－	－	－	－	－	－	－	－	－
4	WM_CTLCOLOREDIT	◎	◎	－	－	－	－	－	－	－	－	－	－	－
5	WM_CTLCOLORLISTBOX	◎	◎	－	－	－	－	－	－	－	－	－	－	－
6	WM_CTLCOLORSCROLLBAR	◎	◎	－	－	－	－	－	－	－	－	－	－	－
7	WM_INITMENU	○	○	－	－	－	－	－	－	－	－	－	－	－
8	WM_MEASUREITEM (*1)	◎	◎	－	－	－	－	－	－	－	－	－	－	－
9	WM_DRAWITEM (*2)	◎	◎	－	－	－	－	－	－	－	－	－	－	－
10	WM_SETTEXT	－	－	○	○	－	－	－	－	－	－	－	－	－
11	WM_ENABLE	－	－	○	○	－	－	○	○	○	○	○	－	－
12	WM_LBUTTONDOWN	－	－	◎	◎	－	－	－	－	－	－	－	○	－
13	WM_LBUTTONUP	－	－	◎	◎	○	－	－	－	－	－	－	－	－
14	WM_LBUTTONDBLCLK	－	－	◎	◎	－	－	－	－	－	－	－	－	－
15	WM_SETFOCUS	－	－	◎	◎	－	－	－	－	－	－	－	－	－
16	WM_KILLFOCUS	－	－	◎	◎	－	－	－	－	－	－	－	－	－
17	WM_ERASEBKGND	－	－	◎	◎	－	－	－	－	－	－	◎	－	◎
18	WM_PAINT	－	－	◎	◎	○	－	－	－	－	－	－	－	－
19	WM_CTLCOLOREDIT	－	－	－	－	－	◎	－	－	－	－	－	－	－
20	WM_DESTROY	○	○	○	○	－	－	－	－	－	－	－	－	－
21	BM_GETSTATE	－	－	◎	◎	－	－	－	－	－	－	－	－	－
22	BM_SETSTATE	－	－	◎	◎	－	－	－	－	－	－	－	－	－
23	BM_GETCHECK	－	－	－	◎	－	－	－	－	－	－	－	－	－
24	BM_SETCHECK	－	－	－	◎	－	－	－	－	－	－	－	－	－
25	BM_SETSTYLE	－	－	◎	◎	－	－	－	－	－	－	－	－	－

*1 ： ODT_MENU

*2 ： ODT_MENU, ODT_LISTBOX, ODT_COMBOBOX, ODT_TAB

*3 ： BS_DEFPUSHBUTTON, BS_PUSHBUTTON

*4 ： BS_PUSHLIKE(CheckBox, RadioButton)

16.1. サポートＡＰＩ

ダイアログボックス、コントロールとメニューのカラー設定のサポートＡＰＩ一覧を以下に示します。

#	関数名	内容	
1	AjcDgcSetup[Ex]	セットアップ	WinMain 内で実行
2	AjcDgcSetColors[Ex]	背景色とテキスト色の設定	
3	AjcDgc{Set|Get}BmpTransColor	ビットマップ描画時の透明色設定／取得	
4	AjcDgcDisableColorDisplay	カラー表示禁止	WM_INITDIALOG/WM_CREATE 内で実行
5	AjcDgcSetColorsDlg	ダイアログボックスによる背景色とテキスト色設定	
6	AjcDgcGetSbcButtonStyle	ボタンのスタイル取得	
7	AjcDgcSetSbcButtonStyle	ボタンのスタイル設定	
8	AjcDgcTrackPopupMenu[Ex]	ポップアップメニュー	
9	AjcDgc{Get/Set}ShowFocus	フォーカスの表示フラグ取得／設定	
10	AjcDgc{Get/Set}MenuFont	メニューフォント取得／設定	
11	AjcDgc{Get/Set}ColorInfo	描画色の詳細情報取得／設定	
12	AjcDgc{Set|Get}Opt	オプションの設定／取得	
13	AjcDgcInitColorDisplay[Ex]	ウインド・カラー表示の初期化	通常は実行する必要なし

16.1.1. セットアップ (AjcDgcSetup[Ex])

形　式 ： BOOL　AjcDgcSetup(VO);
BOOL　AjcDgcSetupEx(C_UTP pPath, C_UTP pSec);

引　数 ： pPath - プロファイルパスへのポインタ（規定値の場合は NULL）
pSec　- プロファイルセクション名へのポインタ

説　明 ： ダイアログボックス／ウインドとコントロールのカラー設定に関する初期化を行います。
AjcDgcSetupEx()の場合は、さらに、背景色やテキスト色を記録するプロファイル情報を指定します。
このプロファイルには、AjcDgcSetColorsDlg()により設定された背景色やテキスト色が記録されます。
pPath には、レジストリパス（ex. “Software¥AjrCstXX¥*AppName*”）や、INI ファイルパス（ex. “c:¥*AppName.ini*”）のいずれかを指定します。(pPath=NULL の場合は、現在設定されているレジストリのルートパス＋ミドルパスが仮定されます)
本ＡＰＩは、(WinMain()等で) プログラムの最初に実行します。
他のＡＰＩは、本ＡＰＩの後に実行してください。(本ＡＰＩ未実行の場合、他のＡＰＩは全て無視されます)

戻り値 ： TRUE　- 成功
FALSE - 失敗

16.1.2. 背景色とテキスト色の設定(AjcDgcSetColors)

形　式 ： BOOL　AjcDgcSetColors　(COLORREF BkColor, COLORREF TextColor);
BOOL　AjcDgcSetColorsEx(COLORREF BkColor, COLORREF TextColor, int ratio);

引　数 ： BkColor　　- 背景色　　(-1:設定しない)
TextColor　- テキスト色 (-1:設定しない)
rratio　　- 無効時背景色生成時の色要素の割合

説　明 ： ダイアログ, コントロール群とメニューの背景色と文字色を設定します。
詳細な変更項目は、描画色の詳細情報取得／設定(AjcDgc{Get/Set}ColorInfo)内で赤枠で示します。

ratio が１以上の場合、有効ウインド背景色から無効ウインド背景色を算出する三原色の割合(%)を指定します。
　　無効時背景色(R) = BkColor(R) × ratio / 100 // 但し、最大値=255
　　無効時背景色(G) = BkColor(G) × ratio / 100 // 但し、最大値=255
　　無効時背景色(B) = BkColor(B) × ratio / 100 // 但し、最大値=255
ratio = 0 を指定した場合は、デフォルトの割合(95[%])を仮定します。
ratio = -1 を指定した場合は、何もしません。

以下の条件を満たす場合、AjcDgcSetColorsDlg()で設定された背景色やテキスト色が反映されます。
・AjcDgcSetupEx()で、プロファイルパスが設定されている
・AjcDgcSetColorsDlg()で、ダイアログによる背景色、テキスト色が設定されている

本ＡＰＩは、(WinMain()等で) AjcDgcSetup()に続けてプログラムの最初に実行します。

16.1.3. ビットマップ透明色設定／取得 (AjcDgc{Get/Set}BmpTransColor)

形　式 ： COLORREF　AjcDgcGetBmpTransColor (VO); ---------- 取得
COLORREF　AjcDgcSetBmpTransColor(COLORREF ccBmpTransparent); ----- 設定

引　数 ： ccBmpTransparent - ビットマップを透明化する色 (-1 ／未指定 ： 透明化しない)

説　明 ： ビットマップを描画する際に透明化する色を設定／取得します。
この色は、デフォルトで白色 ((RGB(255,255, 255))に設定されています。
本ＡＰＩは、(WinMain()等で) AjcDgcSetup()に続けてプログラムの最初に実行します。

戻り値 ： ビットマップを描画する際に透明化する色（設定時は前の値）

備　考 ： ビットマップを透明化する色が白色である場合、この API を実行する必要はありません。
白色を変更、あるいは、透明化しない場合は通常、AjcDgcSetup()の直後に実行します。

16.1.4. カラー表示禁止 (AjcDgcDisablwColorDisplay)

形　式 ： BOOL　AjcDgcDisableColorDisplay (HWND hwnd);

引　数 ： hwnd – カラー表示を禁止するウインドハンドル

説　明 ： 背景色やテキスト色のカラー表示を禁止します。
ウインドやコントロールは、システム標準の色で表示されます。
本ＡＰＩは、WM_CREATE / WM_INITDIALOG 内で実行します

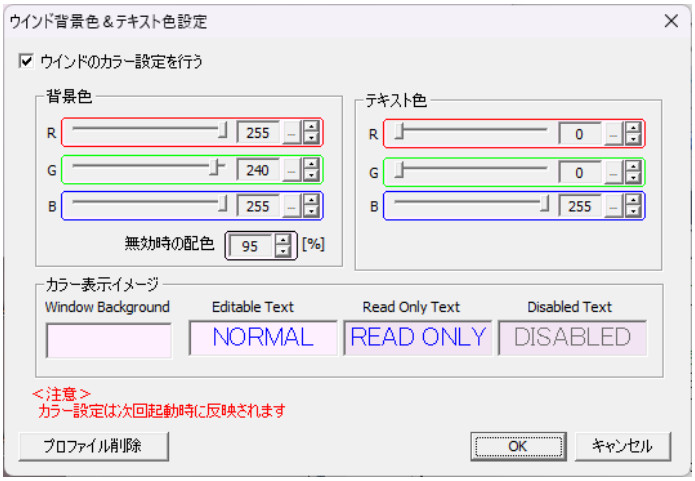
戻り値 ： TRUE　：成功
FALSE ：失敗

16.1.5. ダイアログボックスによる背景色とテキスト色設定 (AjcDgcSetColorsDlg)

形　式 ： BOOL　AjcDgcSetColorsDlg (HWND hOwner, COLORREF cNote, C_WCP pNote);

引　数 ： hOwner – オーナーウインド（不要時は NULL）
cNote　– ダイアログ中に表示する注記テキストの色
pNote　– ダイアログ中に表示する注記テキストへのポインタ

説　明 ： 以下のダイアログボックスにより、背景色やテキスト色を設定します。
設定した背景色やテキスト色は、次回起動時に、AjcDgcInitColorDisplay[Ex]()実行時に参照されます。



「ＯＫ」ボタンを押すと、設定を AjcDgcSetupEx()で指定したプロファイルに記録して終了ます。
「キャンセル」ボタンを押すと設定を無視して終了します。
「プロファイル削除」ボタンを押すと、プロファイルを削除して終了します。

戻り値 ： TRUE　：成功
FALSE ：失敗（AjcDgcSetupEx()で、プロファイルパスが設定されていない）

16.1.6. ボタンのスタイル取得(AjcDgcGetSbcButtonStyle)

形　式 ： BOOL　AjcDgcGetSbcButtonStyle(HWND hwnd, UIP pStyle);

引　数 ： hwnd　– ボタン(BS_PUSHBUTTON/BS_DEFPUSHBUTTON/BS_PUSHLIKE)のハンドル
pStyle – 取得したボタンスタイルを格納するバッファのアドレス（不要時は NULL）

説　明 ： ボタンのスタイルを取得します。
サブクラス化したボタンは、BS_OWNERDRAW スタイルが設定され、サブクラス化前のスタイルは退避されます。
本関数は、この退避されたスタイルを取得します。

戻り値 ： TRUE　：ボタンのスタイル
FALSE：失敗（サブクラス化されたボタン(BS_PUSHBUTTON/BS_DEFPUSHBUTTON/BS_PUSHLIKE)以外）

16.1.7. ボタンのスタイル設定(AjcDgcSetSbcButtonStyle)

形　式 ： BOOL AjcDgcSetSbcButtonStyle(HWND hwnd, UI style);

引　数 ： hwnd - ボタン(BS_PUSHBUTTON/BS_DEFPUSHBUTTON/BS_PUSHLIKE)のハンドル
style - ボタンのスタイル（ボタンコントロールのスタイル（下位 16Bit のみ））

説　明 ： ボタンのスタイルを設定します。
サブクラス化したボタンは、BS_OWNERDRAW スタイルが設定され、サブクラス化前のスタイルは退避されます。
但し、設定できるボタンスタイルは以下に限定されます。（下記以外のスタイルは変更できません）

・BS_TEXT	・BS_LEFT	・BS_TOP	・BS_MULTILINE
・BS_ICON	・BS_RIGHT	・BS_BOTTOM	・BS_NOTIFY
・BS_BITMAP	・BS_CENTER	・BS_VCENTER	・BS_FLAT

戻り値 ： TRUE ：設定前のボタン・スタイル
FALSE：失敗（サブクラス化されたボタン(BS_PUSHBUTTON/BS_DEFPUSHBUTTON/BS_PUSHLIKE)以外）

16.1.8. ポップアップメニュー(AjcDgcTrackPopupMenu[Ex])

形　式 ： BOOL AjcDgcTrackPopupMenu (HMENU hMenu, UI uFlags, int x, int y, int nReserved, HWND hWnd, const RECT *prcRect)：
BOOL AjcDgcTrackPopupMenuEx(HMENU hMenu, UI uFlags, int x, int y, HWND hWnd, LPTPMPARAMS lptpm);

引　数 ： hMenu - メニューハンドル
uFlag - オプション（TPM_*XXXXX*）
x, y - ポップアップメニュー表示位置
nReserved - 未使用（０を指定）
hwnd - オーナーウインド
prcRect - 未使用
lptpm - メニューが重ならないように画面の領域を指定

説　明 ： hwnd（オーナーウインド）が AjcDgcInitColorDisplay[Ex]()でサブクラス化されたウインドである場合は。
ポップアップメニューの背景色や文字色を指定した色で表示し、システムＡＰＩの TrackPopupMenu[Ex]()を実行します。
全てのメニュー項目は、オーナー描画タイプ(MFT_OWNERDRAW)が設定されます。

hwnd（オーナーウインド）がサブクラス化されたウインドでない場合は、単に、TrackPopupMenu[Ex]()を実行します。

使用方法，引数や戻り値は、システムＡＰＩ（TrackPopupMenu() / TrackPopupMenuEx()）と同じです。

戻り値 ： ≠０ – 成功（uFlag で TPM_RETURNCMD を指定した場合、戻り値はユーザーが選択した項目のメニュー項目識別子）
＝０ - 失敗

戻り値 ： TRUE - 成功
FALSE - 失敗

16.1.9. フォーカス表示フラグ取得／設定(AjcDgc{Get/Set}ShowFocus)

形　式 ： BOOL AjcDgcGetShowFocus(VO);
BOOL AjcDgcSetShowFocus(BOOL fShow);

引　数 ： fShow - 設定するフォーカス表示フラグ（TRUE:表示, FALSE:非表示）

説　明 ： ボタンのフォーカス表示フラグを取得／設定します。
フォーカス表示とは、フォーカスのあるボタンコントロールを点線で囲み表示することを意味します。

戻り値 ： AjcDgcGetShowFocus() - フォーカス表示フラグ（TRUE:表示, FALSE:非表示）
AjcDgcSetShowFocus() – TRUE:成功. FALSE：失敗

16.1.10. メニューフォント取得／設定(AjcDgc{Get/Set}MenuFont)

形　式 ： HFONT　AjcDgcGetMenuFont(VO);
BOOL　AjcDgcSetMenuFont(HFONT hFont);

引　数 ： hFont　- 設定するフォントハンドル（NULL ： システムのデフォルト・メニュー・フォント）

説　明 ： メニューテキストのフォントを取得／設定します

戻り値 ： AjcDgcGetMenuFont() - フォントハンドル
AjcDgcSetMenuFont() - TRUE:成功. FALSE：失敗

16.1.11. 描画色の詳細情報取得／設定(AjcDgc{Get/Set}ColorInfo)

形　式 ： HBRUSH　AjcDgcGetColorInfo(PAJCDLGCOLORS　pCtlColors, PAJCMENUCOLORS pMnuColors);　- 取得
BOOL　AjcDgcSetColorInfo(PCAJCDLGCOLORS pCtlColors, PCAJCMENUCOLORS pMnuColors); - 設定

引　数 ： pCtlColors - コントロール群の描画色情報格納バッファ／設定データ（不要時は NULL）
pMnuColors - メニューの描画色情報格納バッファ／設定データ　　（不要時は NULL）
BkRatio　- 無効ウインド背景色の割合[%]（０：デフォルトの割合，-1：指定値(ccBkDis)のままとする）

説　明 ： コントロール群とメニューの描画色の詳細情報を取得／設定します。

AJCDLGCOLORS 構造体(ダイアログ，コントロール群の描画色情報)の内容は以下のとおりです。

```
typedef struct {
    COLORREF    ccBkGnd;            //  ウインド背景色
    COLORREF    ccBkDis;            //  無効ウインド背景色
    COLORREF    ccTextColor;        //  テキスト色
    COLORREF    ccGrayText;         //  無効テキスト色
    COLORREF    ccLtGray;           //  薄い影
    COLORREF    ccGray;             //  通常影
    COLORREF    ccDark;             //  暗い影
    COLORREF    ccBright;           //  明るい色
    COLORREF    ccHatch;            //  ハッチ模様（BS_PUSHLIKE スタイル時のハッチ色）
    COLORREF    ccDefOutL;          //  デフォルトボタンの外枠
} AJCDLGCOLORS, *PAJCDLGCOLORS;
typedef const AJCDLGCOLORS *PCAJCDLGCOLORS;
```

赤枠は AjcDgcSetColors() で設定する項目

#	メンバ名	タイプ	内容	デフォルト		備考
1	ccBkGnd	COLORREF	ウインド背景色	RGB(220, 250, 220)		
2	ccBkDis	COLORREF	無効ウインド背景色	RGB(209, 237, 209)		
3	ccTextColor	COLORREF	テキスト色	RGB(0, 0, 180)		
4	ccGrayText	COLORREF	無効テキスト色	RGB(109, 109, 109)		
5	ccLtGray	COLORREF	薄い影	RGB(227, 227, 227)		ボタン描画用
6	ccGray	COLORREF	通常影	RGB(160, 160, 160)		
7	ccDark	COLORREF	暗い影	RGB(105, 105, 105)		
8	ccBright	COLORREF	明るい色	RGB(255, 255, 255)	白	
9	ccHatch	COLORREF	ハッチ模様（※１）	RGB(255, 255, 255)	白	
10	ccDefOutL	COLORREF	デフォルトボタン外枠	RGB(0, 100, 215)		
11	ccBtnFace	COLORREF	ボタンフェース色	RGB(240, 240, 240)		

※１：BS_PUSHLIKE スタイルのチェックボックスで、不定状態時のハッチ色（背景色に上書きするドットの色）

AJCMENUCOLORS 構造体(メニューの描画色情報)の内容は以下のとおりです。

```
typedef struct {
    COLORREF    std;    //  通常色
    COLORREF    sel;    //  選択時の色
    COLORREF    dis;    //  無効時の色
    COLORREF    both;   //  無効&選択時の色
} AJCSTACOLORS, *PAJCSTACOLORS;
typedef const AJCSTACOLORS *PCAJCSTACOLORS;

typedef struct {
    AJCSTACOLORS    ccBkList;       //  リスト項目の背景色
    AJCSTACOLORS    ccBkMenu;       //  メニュー項目の背景色
    AJCSTACOLORS    ccTextColor;    //  テキスト色
    AJCSTACOLORS    ccChkMark;      //  チェックマーク／ラジオボタン色
    AJCSTACOLORS    ccChkBack;      //  チェックマーク／ラジオボタンの背景色
    AJCSTACOLORS    ccBitmap;       //  ビットマップ色
    AJCSTACOLORS    ccTriangle;     //  サブメニューインジケータ（三角マーク）の色
    COLORREF        ccSepLine;      //  セパレータラインの色
    COLORREF        ccUnderLine;    //  ウインドメニューの下線色（-1:下線非表示）
    AJCSTACOLORS    ccTabSelBk;     //  タブコントロール・選択タブの背景色
    AJCSTACOLORS    ccTabSelTxt;    //  タブコントロール・選択タブのテキスト色
} AJCMENUCOLORS, *PAJCMENUCOLORS;
typedef const AJCMENUCOLORS *PCAJCMENUCOLORS;
```

赤枠は AjcDgcSetColors()で設定する項目

#	メンバ名	タイプ	デフォルト(RGB(rr, gg, bb)							
			std（非選択項目）		sel（選択項目）		dis（無効項目）		both（sel + dis）	
1	ccBkList	AJCSTACOLORS	220, 250, 220		183, 212, 240		209, 237, 209		200, 200, 200	
2	ccBkMwnu	AJCSTACOLORS	220, 250, 220		183, 212, 240		220, 250, 220		200, 200, 200	
3	ccTextColor	AJCSTACOLORS	0, 0, 240	X	0, 0, 240	X	109, 109, 109	X	255, 255, 255	X
4	ccChkMark	AJCSTACOLORS	0, 0, 240	↘	0, 0, 240	↘	160, 160, 160	↘	160, 160, 160	↘
5	ccChkBack	AJCSTACOLORS	145, 201, 247		86, 176, 250		228, 228, 228		216, 216, 216	
6	ccBitmap	AJCSTACOLORS	(BMP のまま)	BMP	(BMP のまま)	BMP	160, 160, 160	BMP	255, 255, 255	BMP
7	ccTriangle	AJCSTACOLORS	0, 0, 240	▸	0, 0, 240	▸	160, 160, 160	▸	160, 160, 160	▸
8	ccSepLine	COLORREF	160, 160, 160	—						
9	ccUnderLine	COLORREF	105, 105, 105	—						
10	ccTabSelBk	AJCSTACOLORS	220, 250, 220		183, 212, 240		220, 250, 220		200, 200, 200	
11	ccTabSelTxt	AJCSTACOLORS	0, 0, 240	X	0, 0, 240	X	160, 160, 160	X	255, 255, 255	

戻り値 ： AjcDgcGetColorInfo() －
≠NULL － 成功（ウインド有効時背景色のブラシハンドル）
＝NULL － 失敗

AjcDgcSetColorInfo() －
TRUE ： 成功
FALSE ： 失敗

備　考 ： 描画色の変更は、AjcDgcSetup()実行直後に、一度だけ行うようにしてください。
任意のタイミングで描画色を変更した場合は、各ウインドの再描画を自力で行う必要があります

16.1.12. オプションの設定／取得 (AjcDgc{Get/Set}Opt)

形　式 ： UI　AjcDgcGetOpt (VO)： -----　取得
UI　AjcDgcSetOpt(Opt)； -----　設定

引　数 ： opt – オプション（以下の名称の合成値，デフォルト＝０）

#	名称	内容
1	AJCDGCOPT_LV_DISABLE_KILLFOCUS	リストビューコントロールをサブクラス化し、WM_KILLFOCUS インターセプトし WM_KILLFOCUS を処理させないようにします。 この場合、WM_NOTIFY(code=NM_KILLFOCUS)は、サブクラス化した処理から発行します。 このオプションは、「リストビューコントロールでフォーカス喪失時に選択行の選択表示が解除され、グレー表示される」という不具合(?)を解消するものです。 **但し、リストビューコントロールから WM_KILLFOCUS をインターセプトすることによる弊害については不明です！**
2	AJCDGCOPT_TV_DISABLE_KILLFOCUS	ツリービューコントロールをサブクラス化し、WM_KILLFOCUS インターセプトし WM_KILLFOCUS を処理させないようにします。 この場合、WM_NOTIFY(code=NM_KILLFOCUS)は、サブクラス化した処理から発行します。 このオプションは、「ツリービューコントロールでフォーカス喪失時に選択行の選択表示が解除される」という不具合(?)を解消するものです。 **但し、ツリービューコントロールから WM_KILLFOCUS をインターセプトすることによる弊害については不明です！**
3	AJCDGCOPT_LVTV_DISABLE_KILLFOCUS	#1 と#2 の合成値

説　明 ： オプションを設定します

戻り値 ： AjcDgcGetOpt () -　　AjcDgcSetOpt() -
現在設定されているオプション値　　前回設定値

16.1.13. ウインドカラー表示の初期化 (AjcDgcInitColorDisplay [Ex])

形　式 ： BOOL　AjcDgcInitColorDisplay (HWND hDlg)：
BOOL　AjcDgcInitColorDisplayEx(HWND hDlg, COLORREF ccBmpTransparent)；

引　数 ： hDlg　- ウインドカラー表示の初期化を行うダイアログボックス／ウインドのハンドル
ccBmpTransparent - ビットマップを透明化する色 (-1 ／未指定 ： 透明化しない)

説　明 ： ダイアログボックス／ウインドと子コントロール群をサブクラス化し、カラー描画を有効にします。
ccBmpTransparent は、コントロールでビットマップを描画する際の透明化する色を指定します。(ビットマップ中の指定色の部分は背景色と同じになります)
本ＡＰＩは、WM_INITDIALOG / WM_CREATE の最後に実行します。
本ＡＰＩは、最初に実行されたものだけが有効となり、２回目以降は無視されます。
先に AjcDgcDisableColorDisplay()が実行されている場合、本ＡＰＩは何もしません。

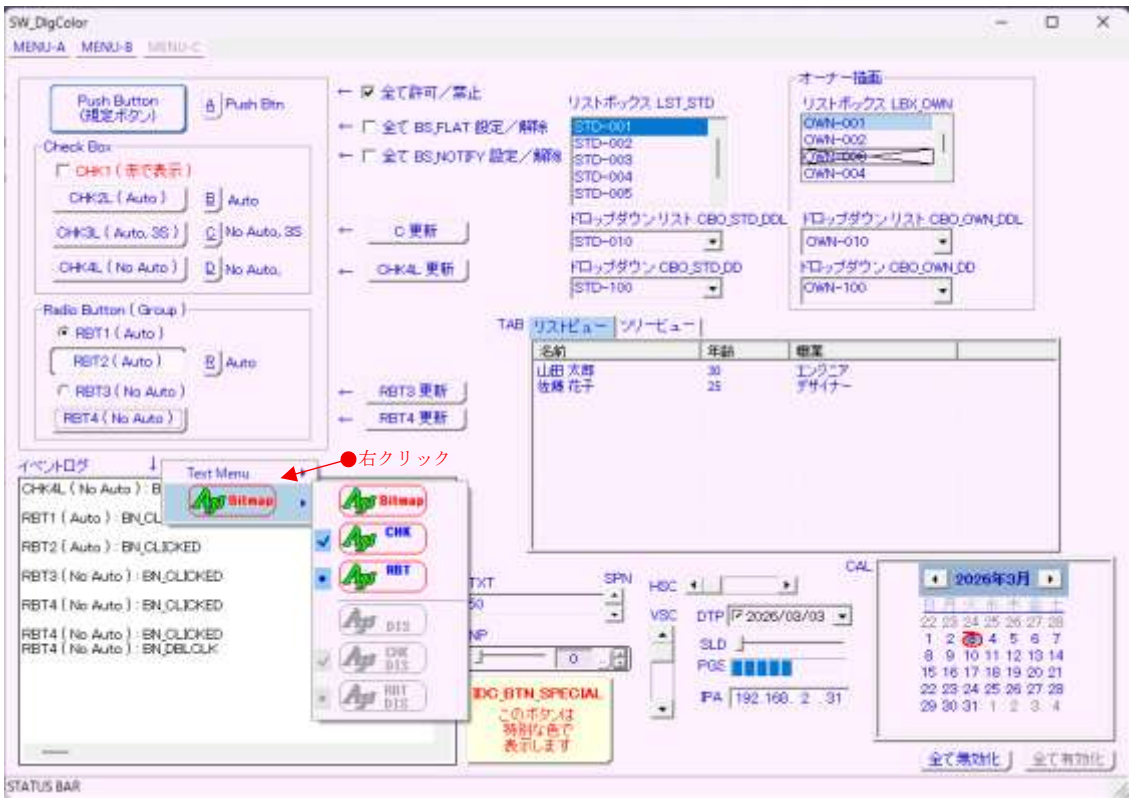
戻り値 ： TRUE　- 成功
FALSE - 失敗

備　考 ： AJC_DLGMAP_DEF()～AJC_DLGMAP_END マクロ あるいは、AJC_WNDMAP_DEF()～AJC_WNDMAP_END マクロでメッセージマップを記述している場合、本ＡＰＩを明示的に実行する必要はありません。
この場合、WM_INITDIALOG / WM_CREATE メッセージの最後に、暗黙的に「AjcDgcInitColorDisplayEx()」が実行され、ビットマップ透明色は白色（デフォルト）に設定されます。
ビットマップ透明色を変更するには、WM_INITDIALOG / WM_CREATE 内で、AjcDgcSetBmpTransColor()を実行します。

16.2. サンプルプログラム

16.2.1. SW_DlgColor（ダイアログ，コントロールやメニューの描画色を設定）

ダイアログ，コントロールやメニューの描画色を設定したサンプルプログラムです。
このサンプルは、単に表示イメージを表示するだけのサンプルプログラムです。
ダイアログを右クリックすると、ポップアップメニューが表示されます。




```
 1 : //
 2 : //  SW_DlgColor.c
 3 : //
 4 :
 5 : #include    <AjrCstXX.h>
 6 : #include    <math.h>
 7 : #include    <tchar.h>
 8 : #include    "resource.h"
 9 :
10 : #define IDC_STATUS  5000
11 :
12 : //-------------------------------------------------------------------------------------------------------------//
13 : //  タブコントロール情報                                                                                       //
14 : //-------------------------------------------------------------------------------------------------------------//
15 : //  タブ インデクス
16 : enum {
17 :     TABID_LISTVIEW      ,
18 :     TABID_TREEVIEW      ,
19 :     MAX_TABITEMS
20 : };
21 : //  タブ中のダイアログハンドル
22 : typedef union {
23 :     struct {
24 :         HWND        hDlgListView;
25 :         HWND        hDlgTreeView;
26 :     } s;
27 :     HWND    hArrDlg[MAX_TABITEMS];
28 : } TAB_ITEMS;
29 : static  TAB_ITEMS   TabItems;
30 :
31 :
32 : //-------------------------------------------------------------------------------------------------------------//
33 : //  ワーク                                                                                                     //
```

```
34 : //-------------------------------------------------------------------------------------------------------------//
35 : static  HINSTANCE       hInst;                          //  ＤＬＬインスタンスハンドル
36 : static  HMENU           hMenuBar;
37 : static  HWND            hDlgMain;                       //  ダイアログボックスハンドル
38 : static  HWND            hVth;
39 : static  HWND            hLbx;
40 : static  HWND            hSts;
41 : static  HWND            hTab;
42 : static  HBITMAP         hBmp1, hBmp2, hBmp3, hBmp4, hBmp5, hBmp6;
43 : static  HICON           hIcoXLine;
44 : static  HBRUSH          hBruCtrl;
45 :
46 : //-------------------------------------------------------------------------------------------------------------//
47 : //  内部サブ関数                                                                                               //
48 : //-------------------------------------------------------------------------------------------------------------//
49 : AJC_DLGPROC_DEF(Main);
50 : AJC_DLGPROC_DEF(ListView);
51 : AJC_DLGPROC_DEF(TreeView);
52 : static  BOOL CALLBACK   cbEnumChildWindows(HWND hwnd, LPARAM lParam);
53 : static  VO              EvtLog(HWND hDlg, WPARAM wParam, LPARAM lParam, HWND hVth);
54 :
55 : //=============================================================================================================//
56 : //                                                                                                             //
57 : //  Ｗ ｉ ｎ Ｍ ａ ｉ ｎ                                                                                       //
58 : //                                                                                                             //
59 : //=============================================================================================================//
60 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
61 : {
62 :     MSG             msg;
63 :
64 :     hInst = hInstance;
65 :
66 :     //----- 描画色設定 ---------------------------------//
67 :     AjcDgcSetup();
68 :     AjcDgcSetColors (RGB(255, 240, 255), RGB(0, 0, 255));
69 : //  AjcDgcSetOpt(AJCDGCOPT_LVTV_DISABLE_KILLFOCUS); // ListView/TreeView でフォーカス喪失時、選択表示が解除される不具合(?)対策オプション
70 :
71 :     //----- メイン・ダイアログオープン -----------------//
72 :     hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_DLGMAIN), NULL, AJC_DLGPROC_NAME(Main));
73 :     //----- ダイアログ表示 -----------------------------//
74 :     ShowWindow(hDlgMain, SW_SHOW);
75 :
76 :     //----- メッセージループ ---------------------------//
77 :     while (GetMessage(&msg, NULL, 0, 0)) {
78 :         do {
79 :             UI  i;
80 :             //  子ダイアログメッセージチェック
81 :             for (i = 0; i < MAX_TABITEMS; i++) {
82 :                 if (IsDialogMessage(TabItems.hArrDlg[i], &msg)) break;
83 :             }
84 :             if (i < MAX_TABITEMS) break;
85 :             //  親ダイアログメッセージチェック
86 :             if (IsDialogMessage(hDlgMain, &msg)) break;
87 :             TranslateMessage(&msg);
88 :             DispatchMessage (&msg);
89 :         } while (0);
90 :     }
91 :
92 :     return (int)msg.wParam ;
93 : }
94 : //=============================================================================================================//
95 : //                                                                                                             //
96 : //  ダイアログ・プロシージャ                                                                                   //
97 : //                                                                                                             //
98 : //=============================================================================================================//
99 : //----- ダイアログ初期化 -----------------------------------------------------------------------------------//
100 : AJC_DLGPROC(Main, WM_INITDIALOG     )
101 : {
102 :     UTP     pTxt;
103 :     TC_ITEM TcItem;
104 :     RECT    r;
105 :     int     x, y, cx, cy;
106 :
107 :     hDlgMain = hDlg;
108 :     hVth     = GetDlgItem(hDlg, IDC_VTH);
109 :     hLbx     = GetDlgItem(hDlg, IDC_LBX_OWN);
110 :     hTab     = GetDlgItem(hDlg, IDC_TAB);
111 :
112 :     hBruCtrl = CreateSolidBrush(RGB(255, 255, 225));
113 :     hBmp1 = LoadBitmap(hInst, MAKEINTRESOURCE(IDB_BITMAP1));
```

```
114 :     hBmp2 = LoadBitmap(hInst, MAKEINTRESOURCE(IDB_BITMAP2));
115 :     hBmp3 = LoadBitmap(hInst, MAKEINTRESOURCE(IDB_BITMAP3));
116 :     hBmp4 = LoadBitmap(hInst, MAKEINTRESOURCE(IDB_BITMAP4));
117 :     hBmp5 = LoadBitmap(hInst, MAKEINTRESOURCE(IDB_BITMAP5));
118 :     hBmp6 = LoadBitmap(hInst, MAKEINTRESOURCE(IDB_BITMAP6));
119 :     hIcoXLine = LoadImage(hInst, MAKEINTRESOURCE(IDI_XLINE), IMAGE_ICON, 16, 16, LR_DEFAULTCOLOR);
120 :
121 :
122 :     //  タブ追加
123 :     TcItem.mask = TCIF_TEXT;
124 :     TcItem.pszText = TEXT("リストビュー");
125 :     TabCtrl_InsertItem(hTab, TABID_LISTVIEW, &TcItem);
126 :
127 :     TcItem.mask = TCIF_TEXT;
128 :     TcItem.pszText = TEXT("ツリービュー");
129 :     TabCtrl_InsertItem(hTab, TABID_TREEVIEW, &TcItem);
130 :
131 :     TabItems.s.hDlgListView = CreateDialog(hInst, MAKEINTRESOURCE(IDD_LISTVIEW), hTab, (DLGPROC)AJC_DLGPROC_NAME(ListView));
132 :     TabItems.s.hDlgTreeView = CreateDialog(hInst, MAKEINTRESOURCE(IDD_TREEVIEW), hTab, (DLGPROC)AJC_DLGPROC_NAME(TreeView));
133 :
134 :     GetClientRect(hTab, &r);
135 :     TabCtrl_AdjustRect(hTab, FALSE, &r);
136 :     x  = r.left;
137 :     y  = r.top;
138 :     cx = r.right - r.left;
139 :     cy = r.bottom - r.top;
140 :
141 :     MoveWindow(TabItems.s.hDlgListView, x, y, cx, cy, FALSE);
142 :     MoveWindow(TabItems.s.hDlgTreeView, x, y, cx, cy, FALSE);
143 :     TabCtrl_SetCurSel(hTab, 0);
144 :     ShowWindow(TabItems.s.hDlgListView, SW_SHOW);
145 :     ShowWindow(TabItems.s.hDlgTreeView, SW_HIDE);
146 :
147 :     //  ステータスバー作成
148 :     hSts = CreateWindow(STATUSCLASSNAME, NULL, SBARS_SIZEGRIP | WS_CHILD | WS_VISIBLE, 0, 0, 0, 0,
149 :                 hDlg, (HMENU)IDC_STATUS, hInst, NULL);
150 :     SendMessage(hSts, SB_SETTEXT, 0, (LPARAM)TEXT("STATUS BAR"));
151 :
152 :     //  チップテキスト
153 :     pTxt = TEXT("AJCM_CTLCOLOR_ANY メッセージを処理し、文字色＝赤, 背景＝クリーム色に設定します");
154 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_CBO_STD_DD), pTxt);
155 :     AjcTipTextAdd(AjcGetDlgItemCboEditHandle(hDlg, IDC_CBO_STD_DD), pTxt);
156 :
157 :     pTxt = TEXT("¥x1B[0;73;16zAJCM_CTLDRAW_LSI メッセージを処理し、３番目のリスト項目上¥n")
158 :            TEXT("にクロスライン（     ）を描画");
159 :     AjcTipTextRegistIcon (hIcoXLine, 0, 0, 0, 0, 0);
160 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_LBX_OWN), pTxt);
161 :
162 :     pTxt = TEXT("AJCM_CTLCOLOR_LSI メッセージを処理し、２番目のリスト項目を赤で表示");
163 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_CBO_OWN_DDL), pTxt);
164 :
165 :     pTxt = TEXT("AJCM_CTLCOLOR_LSI メッセージを処理し、３番目のリスト項目を赤で表示");
166 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_CBO_OWN_DD), pTxt);
167 :     AjcTipTextAdd(AjcGetDlgItemCboEditHandle(hDlg, IDC_CBO_OWN_DD), pTxt);
168 :
169 :     //  各コントロール設定
170 :     AjcSetDlgItemChk(hDlg, IDC_CHK_ENA_DIS, TRUE);
171 :     AjcSetDlgItemChk(hDlg, IDC_RBT4       , TRUE);
172 :     AjcSetDlgItemStr(hDlg, IDC_TXT, TEXT("50"));
173 :     SendDlgItemMessage(hDlg, IDC_PGS, PBM_SETPOS    , 50, 35);                              //  プログレスバー
174 :     SendDlgItemMessage(hDlg, IDC_IPA, IPM_SETADDRESS,  0, MAKEIPADDRESS(192, 168, 2, 31)); //  ＩＰアドレス
175 :
176 :     //  リストボックス／コンボボックスに項目追加
177 :     AjcSetDlgItemLstList(hDlg, IDC_LST_STD     , TEXT("STD-001¥0STD-002¥0STD-003¥0STD-004¥0STD-005¥0STD-006¥0"));
178 :     AjcLbxSetList       (hLbx                  , TEXT("OWN-001¥0OWN-002¥0OWN-003¥0OWN-004¥0OWN-005¥0OWN-006¥0"));
179 :     AjcSetDlgItemCboList(hDlg, IDC_CBO_STD_DDL , TEXT("STD-010¥0STD-020¥0STD-030¥0STD-040¥0STD-050¥0STD-060¥0"));
180 :     AjcSetDlgItemCboList(hDlg, IDC_CBO_STD_DD  , TEXT("STD-100¥0STD-200¥0STD-300¥0STD-400¥0STD-500¥0STD-600¥0"));
181 :     AjcSetDlgItemCboList(hDlg, IDC_CBO_OWN_DDL , TEXT("OWN-010¥0OWN-020¥0OWN-030¥0OWN-040¥0OWN-050¥0OWN-060¥0"));
182 :     AjcSetDlgItemCboList(hDlg, IDC_CBO_OWN_DD  , TEXT("OWN-100¥0OWN-200¥0OWN-300¥0OWN-400¥0OWN-500¥0OWN-600¥0"));
183 :
184 :     //  リストボックス／コンボボックス リスト項目の高さ設定
185 :     AjcSetDlgItemLstHeight(hDlg, IDC_LST_STD, 0, 0);       AjcSetDlgItemLstIx(hDlg, IDC_LST_STD, 0);
186 :     AjcLbxSetItemHeight   (hLbx             , 0, 0);       AjcLbxSetCurSel(hLbx                , 0);
187 :     AjcSetDlgItemCboHeight(hDlg, IDC_CBO_STD_DDL, 0, 0);   AjcSetDlgItemCboIx(hDlg, IDC_CBO_STD_DDL, 0);
188 :     AjcSetDlgItemCboHeight(hDlg, IDC_CBO_STD_DD , 0, 0);   AjcSetDlgItemCboIx(hDlg, IDC_CBO_STD_DD , 0);
189 :     AjcSetDlgItemCboHeight(hDlg, IDC_CBO_OWN_DDL, 0, 0);   AjcSetDlgItemCboIx(hDlg, IDC_CBO_OWN_DDL, 0);
190 :     AjcSetDlgItemCboHeight(hDlg, IDC_CBO_OWN_DD , 0, 0);   AjcSetDlgItemCboIx(hDlg, IDC_CBO_OWN_DD , 0);
191 :
192 :     return TRUE;
193 : }
```

```
194 : //----- ウインド破棄 ----------------------------------------------------------------------------------------------//
195 : AJC_DLGPROC(Main, WM_DESTROY        )
196 : {
197 :     DeleteObject(hBruCtrl);
198 :     //----- プログラム終了 ------------------------------------//
199 :     PostQuitMessage(0);
200 :     return TRUE;
201 : }
202 : //----- WM_COMMAND ------------------------------------------------------------------------------------------------//
203 : AJC_DLGPROC(Main, WM_COMMAND        )
204 : {
205 :     HWND    hCtrl = GetDlgItem(hDlg, LOWORD(wParam));
206 :     UT      cname[256];
207 :
208 :     GetClassName(hCtrl, cname, 256);
209 :     if (_tcscmp(cname, TEXT("Button")) == 0) {
210 :         EvtLog(hDlg, wParam, lParam, GetDlgItem(hDlg, IDC_VTH));
211 :         SetTimer(hDlg, 1, 300, NULL);
212 :     }
213 :     return TRUE;
214 : }
215 : //----- WM_RBUTTONDOWN --------------------------------------------------------------------------------------------//
216 : AJC_DLGPROC(Main, WM_RBUTTONDOWN    )
217 : {
218 :     HMENU       hMenu    = NULL;
219 :     HMENU       hMTxt, hMBmp;
220 :     HMENU       hPopup   = NULL;
221 :     int         id;
222 :     POINT       pt;
223 :
224 :     hMenu  = LoadMenu(hInst, MAKEINTRESOURCE(IDR_MENU1));
225 :     hMTxt = CreatePopupMenu();
226 :     InsertMenu(hMTxt, 0, MF_BYPOSITION | MFT_OWNERDRAW                                                    , 5001, AJCMT("¥x1B[TNormal"));
227 :     InsertMenu(hMTxt, 1, MF_BYPOSITION | MFT_OWNERDRAW                    | MFS_CHECKED                 , 5002, AJCMT("CHK"));
228 :     InsertMenu(hMTxt, 2, MF_BYPOSITION | MFT_OWNERDRAW | MFT_RADIOCHECK | MFS_CHECKED                 , 5003, AJCMT("RBT"));
229 :     InsertMenu(hMTxt, 3, MF_BYPOSITION | MF_SEPARATOR                                                     ,    0, NULL       );
230 :     InsertMenu(hMTxt, 4, MF_BYPOSITION | MFT_OWNERDRAW                                    | MFS_DISABLED , 5004, AJCMT("DIS"));
231 :     InsertMenu(hMTxt, 5, MF_BYPOSITION | MFT_OWNERDRAW                | MFS_CHECKED | MFS_DISABLED , 5005, AJCMT("CHK, DIS"));
232 :     InsertMenu(hMTxt, 6, MF_BYPOSITION | MFT_OWNERDRAW | MFT_RADIOCHECK | MFS_CHECKED | MFS_DISABLED , 5006, AJCMT("RBT. DIS"));
233 :     hMBmp = CreatePopupMenu();
234 :     InsertMenu(hMBmp, 0, MF_BYPOSITION | MFT_OWNERDRAW                                                    , 5011, (UTP)hBmp1);
235 :     InsertMenu(hMBmp, 1, MF_BYPOSITION | MFT_OWNERDRAW                    | MFS_CHECKED                 , 5012, (UTP)hBmp2);
236 :     InsertMenu(hMBmp, 2, MF_BYPOSITION | MFT_OWNERDRAW | MFT_RADIOCHECK | MFS_CHECKED                 , 5013, (UTP)hBmp3);
237 :     InsertMenu(hMBmp, 3, MF_BYPOSITION | MF_SEPARATOR                                                     ,    0, NULL       );
238 :     InsertMenu(hMBmp, 4, MF_BYPOSITION | MFT_OWNERDRAW                                    | MFS_DISABLED , 5014, (UTP)hBmp4);
239 :     InsertMenu(hMBmp, 5, MF_BYPOSITION | MFT_OWNERDRAW                    | MFS_CHECKED | MFS_DISABLED , 5015, (UTP)hBmp5);
240 :     InsertMenu(hMBmp, 6, MF_BYPOSITION | MFT_OWNERDRAW | MFT_RADIOCHECK | MFS_CHECKED | MFS_DISABLED , 5016, (UTP)hBmp6);
241 :     hPopup = CreatePopupMenu();
242 :     InsertMenu(hPopup, 0, MF_BYPOSITION  |MFT_OWNERDRAW | MF_POPUP, (UX)hMTxt, AJCMT("Text Menu"));
243 :     InsertMenu(hPopup, 1, MF_BYPOSITION | MFT_OWNERDRAW | MF_POPUP , (UX)hMBmp, (UTP)hBmp1);
244 :     GetCursorPos(&pt);
245 :     id = AjcDgcTrackPopupMenuEx(hPopup, TPM_RETURNCMD | TPM_TOPALIGN, pt.x, pt.y, hDlg, NULL);
246 :     DestroyMenu(hPopup);
247 :     DestroyMenu(hMenu);
248 :     DestroyMenu(hMTxt);
249 :     DestroyMenu(hMBmp);
250 :
251 :     return TRUE;
252 : }
253 :
254 : //----- WM_TIMER --------------------------------------------------------------------------------------------------//
255 : AJC_DLGPROC(Main, WM_TIMER          )
256 : {
257 :     KillTimer(hDlg, 1);
258 :     AjcVthPrintF(GetDlgItem(hDlg, IDC_VTH), TEXT("¥n"));
259 :     return TRUE;
260 : }
261 : //----- WM_INITMENU -----------------------------------------------------------------------------------------------//
262 : AJC_DLGPROC(Main, WM_INITMENU       )
263 : {
264 :     HMENU   hMenu;
265 :     if (hMenu = GetMenu(hDlg)) {
266 :         EnableMenuItem(hMenu, ID_MENU_A80, MF_BYCOMMAND | MF_GRAYED );
267 :         EnableMenuItem(hMenu, ID_MENU_B10, MF_BYCOMMAND | MF_GRAYED );
268 :     }
269 :     return TRUE;
270 : }
271 : //-----  「Cancel」ボタン ------------------------------------------------------------------------------------------//
272 : AJC_DLGPROC(Main, IDCANCEL          )
273 : {
```

```
274 :     DestroyWindow(hDlg);
275 :     return TRUE;
276 : }
277 : //----- 全て許可／禁止 ----------------------------------------------------------------------------------------//
278 : AJC_DLGPROC(Main, IDC_CHK_ENA_DIS   )
279 : {
280 :     BOOL    f = AjcGetDlgItemChk(hDlg, IDC_CHK_ENA_DIS);
281 :     AjcEnableDlgGroup(hDlg, IDC_GRP_ALL, f, f);
282 :     return TRUE;
283 : }
284 : //----- BS_FLAT 設定 ------------------------------------------------------------------------------------------//
285 : #define SET_STY(ID, STY)                                                                                    ¥
286 :     if (AjcGetDlgItemChk(hDlg, LOWORD(wParam))) MAjcSetWindowLong(GetDlgItem(hDlg, ID), GWL_STYLE,          ¥
287 :                                                    MAjcGetWindowLong(GetDlgItem(hDlg, ID), GWL_STYLE) |  STY); ¥
288 :     else                                          MAjcSetWindowLong(GetDlgItem(hDlg, ID), GWL_STYLE,          ¥
289 :                                                    MAjcGetWindowLong(GetDlgItem(hDlg, ID), GWL_STYLE) & ~STY); ¥
290 :     InvalidateRect(GetDlgItem(hDlg, ID), NULL, TRUE)
291 :
292 : AJC_DLGPROC(Main, IDC_CHK_SET_FLAT  )
293 : {
294 :     SET_STY(IDC_BTN_PUSH_L, BS_FLAT);
295 :     SET_STY(IDC_BTN_PUSH_S, BS_FLAT);
296 :     SET_STY(IDC_CHK1      , BS_FLAT);
297 :     SET_STY(IDC_CHK2L     , BS_FLAT);
298 :     SET_STY(IDC_CHK2S     , BS_FLAT);
299 :     SET_STY(IDC_CHK3L     , BS_FLAT);
300 :     SET_STY(IDC_CHK3S     , BS_FLAT);
301 :     SET_STY(IDC_CHK4L     , BS_FLAT);
302 :     SET_STY(IDC_CHK4S     , BS_FLAT);
303 :     SET_STY(IDC_GRP_RBT   , BS_FLAT);
304 :     SET_STY(IDC_RBT1      , BS_FLAT);
305 :     SET_STY(IDC_RBT2L     , BS_FLAT);
306 :     SET_STY(IDC_RBT2S     , BS_FLAT);
307 :     SET_STY(IDC_RBT3      , BS_FLAT);
308 :     SET_STY(IDC_RBT4      , BS_FLAT);
309 :     return TRUE;
310 : }
311 : //----- BS_NOTIFY 設定 ----------------------------------------------------------------------------------------//
312 : AJC_DLGPROC(Main, IDC_CHK_SET_NOTIFY)
313 : {
314 :     SET_STY(IDC_BTN_PUSH_L, BS_NOTIFY);
315 :     SET_STY(IDC_BTN_PUSH_S, BS_NOTIFY);
316 :     SET_STY(IDC_CHK1      , BS_NOTIFY);
317 :     SET_STY(IDC_CHK2L     , BS_NOTIFY);
318 :     SET_STY(IDC_CHK2S     , BS_NOTIFY);
319 :     SET_STY(IDC_CHK3L     , BS_NOTIFY);
320 :     SET_STY(IDC_CHK3S     , BS_NOTIFY);
321 :     SET_STY(IDC_CHK4L     , BS_NOTIFY);
322 :     SET_STY(IDC_CHK4S     , BS_NOTIFY);
323 :     SET_STY(IDC_GRP_RBT   , BS_NOTIFY);
324 :     SET_STY(IDC_RBT1      , BS_NOTIFY);
325 :     SET_STY(IDC_RBT2L     , BS_NOTIFY);
326 :     SET_STY(IDC_RBT2S     , BS_NOTIFY);
327 :     SET_STY(IDC_RBT3      , BS_NOTIFY);
328 :     SET_STY(IDC_RBT4      , BS_NOTIFY);
329 :     return TRUE;
330 : }
331 : //----- C 更新 ------------------------------------------------------------------------------------------------//
332 : AJC_DLGPROC(Main, IDC_CMD_UPD_C     )
333 : {
334 :     AjcSetDlgItemChk(hDlg, IDC_CHK3S, (AjcGetDlgItemChk(hDlg, IDC_CHK3S) + 1) % 3);
335 :     return TRUE;
336 : }
337 : //----- CHK4L 更新 --------------------------------------------------------------------------------------------//
338 : AJC_DLGPROC(Main, IDC_CMD_UPD_CHK4L )
339 : {
340 :     AjcSetDlgItemChk(hDlg, IDC_CHK4L, (AjcGetDlgItemChk(hDlg, IDC_CHK4L) + 1) % 2);
341 :     return TRUE;
342 : }
343 : //----- RBT3 更新 ---------------------------------------------------------------------------------------------//
344 : AJC_DLGPROC(Main, IDC_CMD_UPD_RBT3  )
345 : {
346 :     AjcSetDlgItemChk(hDlg, IDC_RBT3, (AjcGetDlgItemChk(hDlg, IDC_RBT3) + 1) % 2);
347 :     return TRUE;
348 : }
349 : //----- RBT4 更新 ---------------------------------------------------------------------------------------------//
350 : AJC_DLGPROC(Main, IDC_CMD_UPD_RBT4  )
351 : {
352 :     AjcSetDlgItemChk(hDlg, IDC_RBT4, (AjcGetDlgItemChk(hDlg, IDC_RBT4) + 1) % 2);
353 :     return TRUE;
```

```
354 : }
355 : //----- IDC_BTN_SPECIAL ------------------------------------------------------------------------------------//
356 : AJC_DLGPROC(Main, IDC_BTN_SPECIAL   )
357 : {
358 :     return TRUE;
359 : }
360 : //----- IDC_LBX_OWN ----------------------------------------------------------------------------------------//
361 : AJC_DLGPROC(Main, IDC_LBX_OWN       )
362 : {
363 :     if (HIWORD(wParam) == AJCLBXN_LISTSTYLE) {
364 :         PAJCLBXLSTSTY   p = (PAJCLBXLSTSTY)lParam;
365 :         p->sty |= LBS_OWNERDRAWFIXED;
366 :     }
367 :     return TRUE;
368 : }
369 : //-----  AJCM_CTLCOLOR_BTN ---------------------------------------------------------------------------------//
370 : AJC_DLGPROC(Main, AJCM_CTLCOLOR_BTN)
371 : {
372 :     PAJCDGCCTLCOLOR pCC = (PAJCDGCCTLCOLOR)wParam;
373 :
374 :     if ((HWND)lParam == GetDlgItem(hDlg, IDC_BTN_SPECIAL))  {
375 :         pCC->hBrush    = hBruCtrl;
376 :         pCC->TextColor = RGB(255,0,0);
377 :     }
378 :     return TRUE;
379 : }
380 : //-----  AJCM_CTLCOLOR_ANY ---------------------------------------------------------------------------------//
381 : AJC_DLGPROC(Main, AJCM_CTLCOLOR_ANY)
382 : {
383 :     PAJCDGCCTLCOLOR pCC = (PAJCDGCCTLCOLOR)wParam;
384 :
385 :     //  コンボボックス (IDC_CBO_STD_DD) は、背景＝クリーム色, 文字＝赤で表示
386 :     if ((HWND)lParam == AjcGetDlgItemCboListHandle(hDlg, IDC_CBO_STD_DD)) {
387 :         pCC->hBrush    = hBruCtrl;
388 :         pCC->TextColor = RGB(255, 0, 0);
389 :         pCC->BkColor   = -1;
390 :     }
391 :     //  チェックボックス (IDC_CHK1) は、文字色＝赤
392 :     else if ((HWND)lParam == GetDlgItem(hDlg, IDC_CHK1)) {
393 :         pCC->TextColor = RGB(255, 0, 0);
394 :     }
395 :     return TRUE;
396 : }
397 : //-----  AJCM_CTLCOLOR_LSI ---------------------------------------------------------------------------------//
398 : AJC_DLGPROC(Main, AJCM_CTLCOLOR_LSI)
399 : {
400 :     PAJCDGCCTLCOLOR pCC  = (PAJCDGCCTLCOLOR)wParam;
401 :     PAJCDGCDRAWITEM pDis = (PAJCDGCDRAWITEM)lParam;
402 :
403 :     if      (pDis->id == IDC_CBO_OWN_DDL) {
404 :         if (pDis->ix == 1) pCC->TextColor = RGB(255,0,0);
405 :     }
406 :     else if (pDis->id == IDC_CBO_OWN_DD ) {
407 :         if (pDis->ix == 2) pCC->TextColor = RGB(255,0,0);
408 :     }
409 :     return TRUE;
410 : }
411 : //-----  AJCM_CTLDRAW_LSI ----------------------------------------------------------------------------------//
412 : AJC_DLGPROC(Main, AJCM_CTLDRAW_LSI)
413 : {
414 :     PAJCDGCCTLCOLOR pCC  = (PAJCDGCCTLCOLOR)wParam;
415 :     PAJCDGCDRAWITEM pDis = (PAJCDGCDRAWITEM)lParam;
416 :     UT      txt[256];
417 :
418 :     if (pDis->idLbx == IDC_LBX_OWN) {
419 :         //  ＤＣオブジェクト設定
420 :         HPEN    hPen = (HPEN  )SelectObject(pDis->hDC, (pDis->ix == 2) ? GetStockObject(BLACK_PEN) : GetStockObject(NULL_PEN));
421 :         HBRUSH  hBru = (HBRUSH)SelectObject(pDis->hDC, pCC->hBrush);
422 :         //  項目塗りつぶしと外枠描画
423 :         Rectangle(pDis->hDC, pDis->rect.left  , pDis->rect.top,
424 :                              pDis->rect.right , pDis->rect.bottom);
425 :         //  テキスト描画
426 :         SendMessage(pDis->hWnd, LB_GETTEXT, pDis->ix, (LPARAM) txt);    //  項目のテキスト取得
427 :         SetBkMode(pDis->hDC, TRANSPARENT);                              //  テキスト背景＝透明
428 :         TextOut(pDis->hDC, pDis->rect.left + 3, pDis->rect.top + 1, txt, (UI)MAjcStrLen(txt));
429 :         //  ３番目の項目は、文字の上にタスク掛け斜線描画
430 :         if (pDis->ix == 2) {
431 :             MoveToEx(pDis->hDC ,pDis->rect.left ,pDis->rect.bottom - 1, NULL);
432 :             LineTo  (pDis->hDC, pDis->rect.right,pDis->rect.top);
433 :             MoveToEx(pDis->hDC ,pDis->rect.left ,pDis->rect.top, NULL);
```

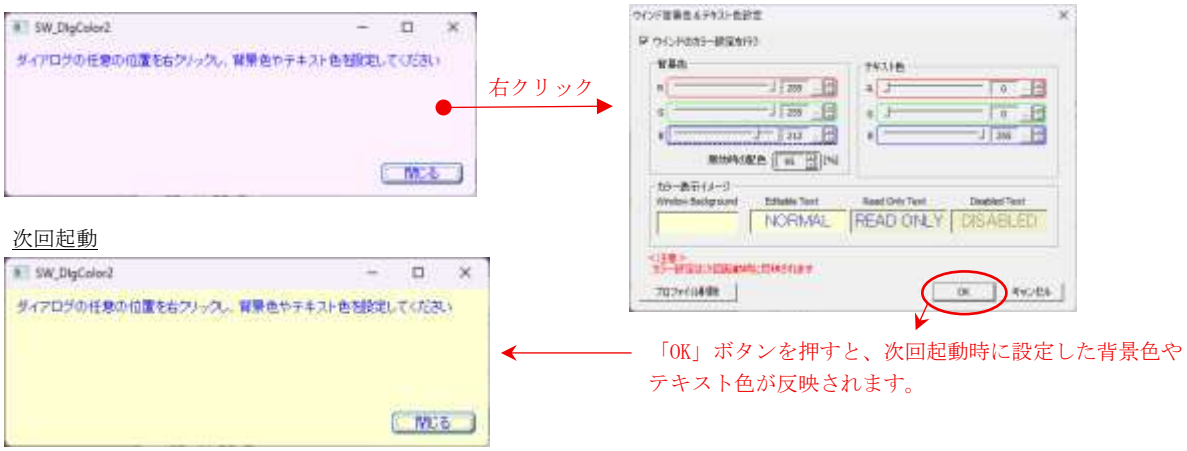
```
434 :             LineTo  (pDis->hDC, pDis->rect.right,pDis->rect.bottom - 1);
435 :         }
436 :         //  ＤＣオブジェクトを元に戻す
437 :         SelectObject(pDis->hDC, hBru);
438 :         SelectObject(pDis->hDC, hPen);
439 :         //  描画済フラグ設定
440 :         pDis->fDrawn = TRUE;
441 :     }
442 :     return TRUE;
443 : }
444 : //-----  全て禁止ボタン ----------------------------------------------------------------------------------------//
445 : AJC_DLGPROC(Main, IDC_CMD_DIS   )
446 : {
447 :     EnumChildWindows(hDlg, cbEnumChildWindows, FALSE);
448 :     EnableWindow(GetDlgItem(hDlg, IDC_CMD_ENA), TRUE);
449 :     return TRUE;
450 : }
451 : //-----  全て許可ボタン ----------------------------------------------------------------------------------------//
452 : AJC_DLGPROC(Main, IDC_CMD_ENA   )
453 : {
454 :     EnumChildWindows(hDlg, cbEnumChildWindows, TRUE);
455 :     EnableWindow(GetDlgItem(hDlg, IDC_CMD_ENA), FALSE);
456 :     return TRUE;
457 : }
458 : //----- すべて禁止／許可の列挙 --------------------------------------------------------------------------------//
459 : static  BOOL CALLBACK cbEnumChildWindows(HWND hwnd, LPARAM lParam)
460 : {
461 :     EnableWindow(hwnd, (UI)lParam);
462 :     return TRUE;
463 : }
464 : //----- タブコントロール --------------------------------------------------------------------------------------//
465 : AJC_DLGPROC(Main, IDC_TAB       )
466 : {
467 :     int     ix, i;
468 :
469 :     if (((LPNMHDR)lParam)->code == TCN_SELCHANGE) {
470 :         ix = TabCtrl_GetCurSel(hTab);
471 :         for (i = 0; i < MAX_TABITEMS; i++) {
472 :             if (i == ix) {
473 :                 ShowWindow(TabItems.hArrDlg[i], SW_SHOW);
474 :             }
475 :             else {
476 :                 ShowWindow(TabItems.hArrDlg[i], SW_HIDE);
477 :             }
478 :         }
479 :         SetFocus(TabItems.hArrDlg[ix]);
480 :     }
481 :     return TRUE;
482 : }
483 : //-----------------------------------------------------------------------------------------------------------//
484 : AJC_DLGMAP_DEF(Main)
485 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG      )
486 :     AJC_DLGMAP_MSG(Main, WM_DESTROY         )
487 :     AJC_DLGMAP_MSG(Main, WM_COMMAND         )
488 :     AJC_DLGMAP_MSG(Main, WM_RBUTTONDOWN     )
489 :     AJC_DLGMAP_MSG(Main, WM_TIMER           )
490 :     AJC_DLGMAP_MSG(Main, WM_INITMENU        )
491 :
492 :     AJC_DLGMAP_CMD(Main, IDCANCEL           )
493 :     AJC_DLGMAP_CMD(Main, IDC_CHK_ENA_DIS    )
494 :     AJC_DLGMAP_CMD(Main, IDC_CHK_SET_FLAT   )
495 :     AJC_DLGMAP_CMD(Main, IDC_CHK_SET_NOTIFY )
496 :     AJC_DLGMAP_CMD(Main, IDC_CMD_UPD_C      )
497 :     AJC_DLGMAP_CMD(Main, IDC_CMD_UPD_CHK4L  )
498 :     AJC_DLGMAP_CMD(Main, IDC_CMD_UPD_RBT3   )
499 :     AJC_DLGMAP_CMD(Main, IDC_CMD_UPD_RBT4   )
500 :     AJC_DLGMAP_CMD(Main, IDC_BTN_SPECIAL    )
501 :     AJC_DLGMAP_CMD(Main, IDC_LBX_OWN        )
502 :     AJC_DLGMAP_DGB(Main, AJCM_CTLCOLOR_BTN  )
503 :     AJC_DLGMAP_DGC(Main, AJCM_CTLCOLOR_ANY  )
504 :     AJC_DLGMAP_CLI(Main, AJCM_CTLCOLOR_LSI  )
505 :     AJC_DLGMAP_DLI(Main, AJCM_CTLDRAW_LSI   )
506 :     AJC_DLGMAP_CMD(Main, IDC_CMD_DIS        )
507 :     AJC_DLGMAP_CMD(Main, IDC_CMD_ENA        )
508 :     AJC_DLGMAP_NTC(Main, IDC_TAB            )
509 : AJC_DLGMAP_END
510 : //-----------------------------------------------------------------------------------------------------------//
511 : //  ダイアログプロシージャ (ListView)                                                                        //
512 : //----- ダイアログ初期化 --------------------------------------------------------------------------------//
513 : AJC_DLGPROC(ListView, WM_INITDIALOG        )
```

```
514 : {
515 :     HWND    hList = GetDlgItem(hDlg, IDC_LISTVIEW);
516 :     LVCOLUMN col;
517 :     LVITEM item;
518 :
519 :     // 拡張スタイル (フル行選択など)
520 :     ListView_SetExtendedListViewStyle(hList, LVS_EX_FULLROWSELECT | LVS_EX_GRIDLINES);
521 :
522 :     // 列追加
523 :     col.mask = LVCF_TEXT | LVCF_WIDTH | LVCF_SUBITEM;
524 :     col.cx = 150;
525 :
526 :     col.pszText = _T("名前");
527 :     col.iSubItem = 0;
528 :     ListView_InsertColumn(hList, 0, &col);
529 :
530 :     col.pszText = _T("年齢");
531 :     col.iSubItem = 1;
532 :     col.cx = 80;
533 :     ListView_InsertColumn(hList, 1, &col);
534 :
535 :     col.pszText = _T("職業");
536 :     col.iSubItem = 2;
537 :     col.cx = 150;
538 :     ListView_InsertColumn(hList, 2, &col);
539 :
540 :     // アイテム追加
541 :     item.mask = LVIF_TEXT;
542 :     item.iItem = 0;
543 :     item.iSubItem = 0;
544 :     item.pszText = _T("山田 太郎");
545 :     ListView_InsertItem(hList, &item);
546 :
547 :     ListView_SetItemText(hList, 0, 1, _T("30"));
548 :     ListView_SetItemText(hList, 0, 2, _T("エンジニア"));
549 :
550 :     item.iItem = 1;
551 :     item.pszText = _T("佐藤 花子");
552 :     ListView_InsertItem(hList, &item);
553 :
554 :     ListView_SetItemText(hList, 1, 1, _T("25"));
555 :     ListView_SetItemText(hList, 1, 2, _T("デザイナー"));
556 :
557 :     return TRUE;
558 : }
559 : //-------------------------------------------------------------------------------------------------------------//
560 : AJC_DLGMAP_DEF(ListView)
561 :     AJC_DLGMAP_MSG(ListView, WM_INITDIALOG      )
562 : AJC_DLGMAP_END
563 : //-------------------------------------------------------------------------------------------------------------//
564 : //  ダイアログプロシージャ (TreeView)                                                                          //
565 : //----- ダイアログ初期化 ------------------------------------------------------------------------------------//
566 : AJC_DLGPROC(TreeView, WM_INITDIALOG         )
567 : {
568 :     TVINSERTSTRUCT tvis = {0};
569 :     HWND        hTree = GetDlgItem(hDlg, IDC_TREE1);
570 :     HTREEITEM   hRoot;
571 :     HTREEITEM   hChild1;
572 :
573 :     // TreeView ノード追加
574 :     tvis.hParent = TVI_ROOT;
575 :     tvis.hInsertAfter = TVI_LAST;
576 :     tvis.item.mask = TVIF_TEXT;
577 :     tvis.item.pszText = TEXT("Tree Root");
578 :     hRoot = TreeView_InsertItem(hTree, &tvis);
579 :
580 :     tvis.hParent = hRoot;
581 :     tvis.item.pszText = TEXT("Child 1");
582 :     hChild1 = TreeView_InsertItem(hTree, &tvis);
583 :
584 :     tvis.hParent = hChild1;
585 :     tvis.item.pszText = TEXT("Child 1-1");
586 :     TreeView_InsertItem(hTree, &tvis);
587 :
588 :     tvis.item.pszText = TEXT("Child 1-2");
589 :     TreeView_InsertItem(hTree, &tvis);
590 :
591 :     tvis.hParent = hRoot;
592 :     tvis.item.pszText = TEXT("Child 2");
593 :     TreeView_InsertItem(hTree, &tvis);
```

```
594 :
595 :     TreeView_Expand(hTree, hRoot  , TVE_EXPAND);
596 :     TreeView_Expand(hTree, hChild1, TVE_EXPAND);
597 :
598 :     return TRUE;
599 : }
600 : //-----------------------------------------------------------------------------------------------------------------//
601 : AJC_DLGMAP_DEF(TreeView)
602 :     AJC_DLGMAP_MSG(TreeView, WM_INITDIALOG      )
603 : AJC_DLGMAP_END
604 : //-----------------------------------------------------------------------------------------------------------------//
605 : //  イベントログ表示                                                                                             //
606 : //-----------------------------------------------------------------------------------------------------------------//
607 : static  VO  EvtLog(HWND hDlg, WPARAM wParam, LPARAM lParam, HWND hVth)
608 : {
609 :     UT      txt[256];
610 :
611 :     AjcGetDlgItemStr(hDlg, LOWORD(wParam), txt, 256);
612 :     AjcVthPrintF(hVth, TEXT("%s : "), txt);
613 :
614 :     switch (HIWORD(wParam)) {
615 :         case BN_CLICKED:    AjcVthPrintF(hVth, TEXT("BN_CLICKED¥n"));   break;
616 :         case BN_DBLCLK:     AjcVthPrintF(hVth, TEXT("BN_DBLCLK¥n"));    break;
617 :         case BN_KILLFOCUS:  AjcVthPrintF(hVth, TEXT("BN_KILLFOCUS¥n")); break;
618 :         case BN_SETFOCUS:   AjcVthPrintF(hVth, TEXT("BN_SETFOCUS¥n"));  break;
619 :     }
620 : }
```

16.2.2. SW_DlgColor2（背景色やテキスト色の動的設定と永続化）

ダイアログ，コントロールやメニューの描画色を動的に設定し、永続絆するサンプルプログラムです。
ダイアログを右クリックすると、背景色やテキスト色の設定ダイアログが表示されます。




```
 1 : //
 2 : //  SW_DlgColor2.c
 3 : //
 4 :
 5 : #include    <AjrCstXX.h>
 6 : #include    <math.h>
 7 : #include    <tchar.h>
 8 : #include    "resource.h"
 9 :
10 : //-------------------------------------------------------------------------------------------------------------------//
11 : //  ワーク                                                                                                           //
12 : //-------------------------------------------------------------------------------------------------------------------//
13 : static  HINSTANCE       hInst;                      //  ＤＬＬインスタンスハンドル
14 : static  HWND            hDlgMain;                   //  ダイアログボックスハンドル
15 : //-------------------------------------------------------------------------------------------------------------------//
16 : //  内部サブ関数                                                                                                     //
17 : //-------------------------------------------------------------------------------------------------------------------//
18 : AJC_DLGPROC_DEF(Main);
19 : AJC_WNDPROC_DEF(SBC_DIALOG);
20 :
21 : //===================================================================================================================//
22 : //                                                                                                                   //
23 : //  Ｗ ｉ ｎ Ｍ ａ ｉ ｎ                                                                                             //
24 : //                                                                                                                   //
25 : //===================================================================================================================//
26 : int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, PSTR szCmdLine, int iCmdShow)
27 : {
28 :     MSG     msg;
29 :
30 :     hInst = hInstance;
31 :
32 :     //  描画色設定
33 : //  AjcDgcSetup();
34 :     AjcDgcSetupEx(NULL, TEXT("WndColorSect"));
35 :     AjcDgcSetColors(RGB(255, 240, 255), RGB(0, 0, 255));
36 :
37 :     //----- メイン・ダイアログオープン -----------------//
38 :     hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_DLGMAIN), NULL, AJC_DLGPROC_NAME(Main));
39 :     //----- ダイアログ表示 -----------------------------//
40 :     ShowWindow(hDlgMain, SW_SHOW);
41 :
42 :     //----- メッセージループ ---------------------------//
43 :     while (GetMessage(&msg, NULL, 0, 0)) {
44 :         do {
45 :             if (IsDialogMessage(hDlgMain, &msg)) break;
46 :             TranslateMessage(&msg);
47 :             DispatchMessage (&msg);
48 :         } while (0);
49 :     }
50 :
51 :     return (int)msg.wParam ;
52 : }
```

```
 53 : //=============================================================================================================//
 54 : //                                                                                                             //
 55 : //  ダイアログ・プロシージャ                                                                                   //
 56 : //                                                                                                             //
 57 : //=============================================================================================================//
 58 : //----- ダイアログ初期化 ------------------------------------------------------------------------------------//
 59 : AJC_DLGPROC(Main, WM_INITDIALOG     )
 60 : {
 61 :     hDlgMain = hDlg;
 62 :
 63 :     //----- ダイアログをサブクラス化 -------------------//
 64 :     MAjcMmpSetSubclass(SBC_DIALOG, hDlgMain);
 65 :
 66 :     return TRUE;
 67 : }
 68 : //----- ウインド破棄 ----------------------------------------------------------------------------------------//
 69 : AJC_DLGPROC(Main, WM_DESTROY        )
 70 : {
 71 :     //  プログラム終了
 72 :     PostQuitMessage(0);
 73 :     return TRUE;
 74 : }
 75 : //-----  「Cancel」ボタン ------------------------------------------------------------------------------------//
 76 : AJC_DLGPROC(Main, IDCANCEL          )
 77 : {
 78 :     DestroyWindow(hDlg);
 79 :     return TRUE;
 80 : }
 81 : //-------------------------------------------------------------------------------------------------------------//
 82 : AJC_DLGMAP_DEF(Main)
 83 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG     )
 84 :     AJC_DLGMAP_MSG(Main, WM_DESTROY        )
 85 :     AJC_DLGMAP_CMD(Main, IDCANCEL          )
 86 : AJC_DLGMAP_END
 87 : //-------------------------------------------------------------------------------------------------------------//
 88 : //                                                                                                             //
 89 : //  ダイアログのサブクラスプロシージャ                                                                         //
 90 : //                                                                                                             //
 91 : //-------------------------------------------------------------------------------------------------------------//
 92 : //----- WM_RBUTTONDOWN --------------------------------------------------------------------------------------//
 93 : AJC_WNDPROC(SBC_DIALOG, WM_RBUTTONDOWN  )
 94 : {
 95 :     AjcDgcSetColorsDlg(hwnd, RGB(255,0,0), TEXT("＜注意＞¥n  カラー設定は次回起動時に反映されます"));
 96 :     return 0;
 97 : }
 98 : //----- WM_DESTROY ------------------------------------------------------------------------------------------//
 99 : AJC_WNDPROC(SBC_DIALOG, WM_DESTROY      )
100 : {
101 : //  return 0;
102 :     return MAjcMmpCallOrgWndProc(SBC_DIALOG);
103 : }
104 : //-------------------------------------------------------------------------------------------------------------//
105 : AJC_WNDMAP_DEF(SBC_DIALOG)
106 :     AJC_WNDMAP_MSG(SBC_DIALOG, WM_RBUTTONDOWN)
107 :     AJC_WNDMAP_MSG(SBC_DIALOG, WM_DESTROY   )
108 : AJC_WNDMAP_END
109 :
```

17. ダイアログボックス項目／ウインド項目のアクセス

主に、ダイアログボックスやウインドのコントロール（テキストボックス等のチャイルド・コントロール）のアクセス関数群です。
主な内容は、以下のとおりです。

17.1. コントロールへ数値／文字列の設定や取得

WindowsAPI の SetDlgItemInt()や GetDlgItemInt()と同様に、テキストボックス等のコントロールへ数値（10/16 進整数(32bit/64bit)や実数(double)）を設定したり、取得することができます。
また、10 進整数／実数では、整数部を 3 桁毎のカンマ(,)で区切って表示することもできます。

ex. AjcSepDlgItemReal(hDlg, IDC_TXT, 1234567.890, 3); --> 1,234,567.890

コントロールから 10 進数値を取得する場合、数値がカンマ(,)で区切られていても正常に数値を取得できます。

数値以外にも、文字列，チェックボックス，ラジオボタン，コンボボックス，スライダへのアクセスもＡＰＩ化しています。

17.2. グループボックス内のコントロールを 一括アクセス

グループボックス内の全コントロールを一括して有効化／無効化したり、表示／非表示や移動することができます。

例えば、グループボックスとグループボックス内の全コントロールを無効化するには、以下のコードを実行します。

```
AjcEnableDlgGroup(hDlg, IDC_GROUP, FALSE, FALSE);
```

17.3. コントロールの設定内容を永続化

テキストボックス，チェックボックス，コンボボックス等の設定内容をプロファイルに記録することにより、各コントロールの設定内容を永続化することができます。(永続化するコントロール等の詳細は AjcLoadAllControlSettings()を参照)

例えば、ダイアログボックス中の全コントロールの設定内容を永続化するには以下のようにします

・プログラムの開始時（WM_INITDIALOG メッセージ等）に以下のコードを実行します。

```
AjcLoadAllControlSettings(hDlg, TEXT("CtrlSettings"), AJCCTL_SELACT_ALL);
```

・プログラムの終了時（WM_DESTROY メッセージ等）に以下のコードを実行します。

```
AjcSaveAllControlSettings(hDlg);
```

17.4. サポートＡＰＩ

ダイアログボックス／ウインド操作のサポートＡＰＩ一覧を以下に示します。

#	関数名	内容	
1	AjcGetDlgItem･･･ AjcGetCtrl･･･	ダイアログボックス／ウインド項目の取得関数群	
2	AjcSetDlgItem･･･ AjcSetCtrl･･･	ダイアログボックス／ウインド項目の設定関数郡	
3	AjcEnableDlgItemToDrop[Ex] AjcEnableCtrlToDrop[Ex]	EDIT コントロールにフォルダやファイルをドロップ可能にする	
4	AjcGetDlgItem{Pos/Size/Rect} AjcSetDlgItem{Pos/Size/Rect} AjcGetCtrl{Pos/Size/Rect} AjcSetCtrl{Pos/Size/Rect}	コントロールの位置，サイズや矩形取得／設定	
5	AjcEnableDlgItem AjcEnableCtrl	ダイアログ／ウインド項目の有効化／無効化	
6	AjcShowDlgItem AjcShowCtrl	ダイアログ／ウインド項目の表示／非表示	
7	AjcShowAndEnableDlgItem AjcShowAndEnableCtrl	コントロールの表示／非表示 と 許可／禁止	
8	AjcEnableDlgGroup AjcEnableGroup	グループボックス内の全コントロール有効化／無効化	
9	AjcEnableCtrlsInWnd	ウインド内の全コントロールの有効化／無効化	
10	AjcShowDlgGroup AjcShowGroup	グループボックスとグループボックス内の全コントロール表示／非表示	
11	AjcShowDlgGroupEx AjcShowGroupEx	グループボックスやグループボックス内の全コントロール表示／非表示 （グループボックスやグループボックス内の全コントロール群を表示／非表示）	
12	AjcMoveDlgGroupToLoc AjcMoveDlgGroupToCtl AjcMoveDlgGroupToOrg AjcMoveGroupToLoc AjcMoveGroupToCtl AjcMoveGroupToOrg	グループボックスとグループボックス内の全コントロールの移動	
13	AjcDlgItemEnumInGroup AjcCtrlEnumInGroup	グループボックス内のコントロール列挙	
14	AjcLoadAllControlSettings AjcSaveAllControlSettings AjcLoadGrpControlSettings AjcSaveGrpControlSettings	ウインド内／グループボックス内 全コントロールの設定内容を永続化	コントロール設定値の永続化機能
15	AjcCtrlSetPermAtt AjcDlgItemSetPermAtt	各コントロールの永続化属性設定	
16	AjcCtrlSetPermAttGrp AjcDlgItemSetPermAttGrp	グループボックス内全コントロールの永続化属性設定	
17	AjcDelAllCtrlPermAtt	ウインド内全コントロールの永続化属性解除	
18	Ajc{DlgItem/Ctrl}LoadTextBox[Ex] Ajc{DlgItem/Ctrl}SaveTextBox[Ex]	テキストボックスの永続化	
19	Ajc{DlgItem/Ctrl}LoadChkBox Ajc{DlgItem/Ctrl}SaveChkBox	チェックボックス／ラジオボタンの永続化	
20	Ajc{DlgItem/Ctrl}LoadComboBox Ajc{DlgItem/Ctrl}LoadComboBox	コンボボックスの永続化	
21	Ajc{DlgItem/Ctrl}LoadListBox Ajc{DlgItem/Ctrl}LoadListBox	リストボックスの永続化	

17.4.1. ダイアログボックス／ウインド項目の取得 (AjcGetDlgItem･･･/ AjcGetCtrl･･･)

形　式　：　**ダイアログボックス項目の取得**

```
UI     AjcGetDlgItemUInt    (HWND hDlg, int id);  ------------------------ 符号なし１０進整数
 SI    AjcGetDlgItemSInt    (HWND hDlg, int id);  ------------------------ 符号つき１０進整数
 UI    AjcGetDlgItemHex     (HWND hDlg, int id);  ------------------------ １６進整数
 double AjcGetDlgItemReal    (HWND hDlg, int id);  ------------------------ 実数
 ULL   AjcGetDlgItemUI64    (HWND hDlg, int id);   ----------------------- 符号なし１０進整数（６４ビット）
 SLL   AjcGetDlgItemSI64    (HWND hDlg, int id);   ----------------------- 符号つき１０進整数（６４ビット）
 ULL   AjcGetDlgItemH64     (HWND hDlg, int id);   ----------------------- １６進整数（６４ビット）
 UI    AjcGetDlgItemStr     (HWND hDlg, int id, UTP pBuf, UI lBuf);  ---- 文字列
 UI    AjcGetDlgItemStrLen  (HWND hDlg, int id);  ------------------------ 文字列長
 UI    AjcGetDlgItemChk     (HWND hDlg, int id);   ----------------------- チェックボックス／ラジオボタン状態

 UI    AjcGetDlgItemCboIx   (HWND hDlg, int id);   ----------------------- コンボボックスの選択項目インデクス
 UI    AjcGetDlgItemCboCount(HWND hDlg, int id);   ----------------------- コンボボックスに設定されている項目の個数
 UI    AjcGetDlgItemCboLen  (HWND hDlg, int id, UI ix);   ---------------- コンボボックス項目の文字長
 UI    AjcGetDlgItemCboItem (HWND hDlg, int id, UI ix, UTP pBuf, UI lBuf); - コンボボックス項目の文字列
 SX    AjcGetDlgItemCboData (HWND hDlg, int id, UI ix);   ---------------- コンボボックス項目に関連付けられた数値
 UI    AjcGetDlgItemCboFind (HWND hDlg, int id, UI ix, C_UTP pStr, UI flag); - コンボボックスの文字列検索
 UI    AjcGetDlgItemCboMaxLen(HWND hDlg, int id); ------------------------ コンボボックス項目群の最大文字列長取得
 UI    AjcGetDlgItemCboHeight(HWND hDlg, int id, UI ix); ----------------- コンボボックス項目の高さ取得
 HWND  AjcGetDlgItemCboEditHandle(HWND hDlg, int id);  --------- コンボボックス内テキストボックスのハンドル取得
 HWND  AjcGetDlgItemCboListHandle(HWND hDlg, int id);  --------- コンボボックス内リストボックスのハンドル取得

 UI    AjcGetDlgItemLstIx   (HWND hDlg, int id);   ----------------------- リストボックスの選択項目インデクス
                                                                            複数選択時は、最初の選択項目インデクス
 UI    AjcGetDlgItemLstCount(HWND hDlg, int id);   ----------------------- リストボックスに設定されている項目の個数
 UI    AjcGetDlgItemLstLen  (HWND hDlg, int id, UI ix);   ---------------- リストボックス項目の文字長
 UI    AjcGetDlgItemLstItem (HWND hDlg, int id, UI ix, UTP pBuf, UI lBuf); - リストボックス項目の文字列
 SX    AjcGetDlgItemLstData (HWND hDlg, int id, UI ix);   ---------------- リストボックス項目に関連付けられた数値
 UI    AjcGetDlgItemLstFind (HWND hDlg, int id, UI ix, C_UTP pStr, UI flag); - リストボックスの文字列検索
 UI    AjcGetDlgItemLstMaxLen(HWND hDlg, int id); ------------------------ リストボックス項目群の最大文字列長取得
 UI    AjcGetDlgItemLstHeight(HWND hDlg, int id, UI ix); ----------------- リストボックス項目の高さ取得

 int   AjcGetDlgItemSldPos  (HWND hDlg, int id);   ----------------------- スライダの現在位置
```

ウインド項目の取得

```
UI     AjcGetCtrlUInt    (HWND hwnd);  ------------------------------- 符号なし１０進整数
 SI    AjcGetCtrlSInt    (HWND hwnd);  ------------------------------- 符号つき１０進整数
 UI    AjcGetCtrlHex     (HWND hwnd);  ------------------------------- １６進整数
 double AjcGetCtrlReal    (HWND hwnd);  ------------------------------- 実数
 ULL   AjcGetCtrlUI64    (HWND hwnd);   ------------------------------ 符号なし１０進整数（６４ビット）
 SLL   AjcGetCtrlSI64    (HWND hwnd);   ------------------------------ 符号つき１０進整数（６４ビット）
 ULL   AjcGetCtrlH64     (HWND hwnd);   ------------------------------ １６進整数（６４ビット）
 UI    AjcGetCtrlStr     (HWND hwnd, UTP pBuf, UI lBuf);  ------------ 文字列
 UI    AjcGetCtrlStrLen  (HWND hwnd);  ------------------------------- 文字列長
 UI    AjcGetCtrlChk     (HWND hwnd);   ------------------------------ チェックボックス／ラジオボタン状態

 UI    AjcGetCtrlCboIx   (HWND hwnd);   ------------------------------ コンボボックスの選択項目インデクス
 UI    AjcGetCtrlCboCount(HWND hwnd);   ------------------------------ コンボボックスに設定されている項目の個数
 UI    AjcGetCtrlCboLen  (HWND hwnd, UI ix);   ----------------------- コンボボックス項目の文字長
 UI    AjcGetCtrlCboItem (HWND hwnd, UI ix, UTP pBuf, UI lBuf); ------ コンボボックス項目の文字列
 SX    AjcGetCtrlCboData (HWND hwnd, UI ix);   ----------------------- コンボボックス項目に関連付けられた数値
 UI    AjcGetCtrlCboFind (HWND hwnd, UI ix, C_UTP pStr, UI flag); ----- コンボボックスの文字列検索
 UI    AjcGetCtrlCboMaxLen(HWND hwnd); ------------------------------- コンボボックス項目群のの最大文字列長取得
 UI    AjcGetCtrlCboHeight(HWND hwnd, UI ix); ------------------------ コンボボックス項目の高さ取得
 HWND  AjcGetCtrlCboEditHandle(HWND hwnd);  -------------------- コンボボックス内テキストボックスのハンドル取得
 HWND  AjcGetCtrlCboListHandle(HWND hwnd);  -------------------- コンボボックス内リストボックスのハンドル取得

 UI    AjcGetCtrlLstIx   (HWND hwnd);   ------------------------------ リストボックスの選択項目インデクス
 UI    AjcGetCtrlLstCount(HWND hwnd);   ------------------------------ リストボックスに設定されている項目の個数
 UI    AjcGetCtrlLstLen  (HWND hwnd, UI ix);   ----------------------- リストボックス項目の文字長
 UI    AjcGetCtrlLstItem (HWND hwnd, UI ix, UTP pBuf, UI lBuf); ------ リストボックス項目の文字列
 SX    AjcGetCtrlLstData (HWND hwnd, UI ix);   ----------------------- リストボックス項目に関連付けられた数値
 UI    AjcGetCtrlLstFind (HWND hwnd, UI ix, C_UTP pStr, UI flag); ----- リストボックスの文字列検索
 UI    AjcGetCtrlLstMaxLen(HWND hwnd); ------------------------------- リストボックス項目群のの最大文字列長取得
 UI    AjcGetCtrlLstHeight(HWND hwnd, UI ix); ------------------------ リストボックス項目の高さ取得

 int   AjcGetCtrlSldPos  (HWND hwnd);   ------------------------------ スライダの現在位置
```

引　数　：　hDlg　－ ダイアログボックスのハンドル
id　　－ コントロールの識別番号
hwnd　－ コントロールのウインドハンドル
ix　　－ リストボックス／コンボボックスの項目インデクス
AjcGet･･･{Lst/Cbo}Find 以外では、現在選択されている項目を指定する場合は「－１」を指定します。
AjcGet･･･{Lst/Cbo}Find では、「－１」を指定した場合、先頭の項目から検索を開始します。
AjcGet･･･CboHeight では、「－１」を指定した場合、テキストボックスの高さを意味します。

pBuf　－ 文字列を格納するバッファのアドレス
lBuf　－ 文字列を格納するバッファの文字数
pStr　－ 検索する文字列のアドレス
flag　－ 検索(AjcGet･･･CboFind)における文字列の比較パラメタ。以下の値の合成値（あるいは０）を指定する。

指定値	指定時	未指定時
AJCCBF_EXACT_STR AJCCBF_EXACT	文字列全体を比較する	先頭部分の比較
AJCCBF_EXACT_WIDTH AJCCBF_RECOGNIZE_WIDTH AJCCBF_DISTINGUISH	英大小文字を区別する	英大小文字を区別しない
AJCCBF_SELECT	見つかった項目を選択状態にする	選択状態にしない

説　明　：　ダイアログボックス／ウインドのコントロールに設定されている内容を取得します。

AjcGet･･･UInt/SInt/Hex/UI64/SI64/H64 では、コントロールの文字列を整数変換した値を返します。
文字列の先頭が「0x」である場合は、１６新表現の文字列を整数に変換します。
その他の場合は、１０進表現の文字列を整数に変換します。
１０進文字列の場合、区切り記号（３桁毎のカンマ（,）等）を許容します。

AjcGet･･･Real では、実数表現の１０進数を実数に変換した値を返します。
AjcGet･･･Real では、整数部の区切り記号（３桁毎のカンマ（,）等）を許容します。

１０進数での区切り記号（３桁毎のカンマ（,）等）は、ロケール情報に依存します。

AjcGet･･･{Lst/Cbo}Find()では、ix で指定された次の項目からコンボボックスに設定されている全項目を検索し、最初に一致する項目のインデクスを返します。(見つからなかった場合は、－１を返します)
ix=-1 を指定した場合は、先頭の項目から検索を開始します。

戻り値　：

AjcGet･･･Str	：	取得した文字列の文字長　(終端(0x00)は含まない，エラー時は０を返す)
AjcGet･･･StrLen	：	コントロール・テキストの文字長　　(終端(0x00)は含まない，エラー時は０を返す)
AjcGet･･･Chk	：	チェックボックス／ラジオボタンの状態（０：チェックなし，１：チェックあり，２：不定)
AjcGet･･･{Lst/Cbo}Ix	：	現在選択されているコンボボックス項目のインデクス　　（０～，エラー時は-1 を返す)
AjcGet･･･{Lst/Cbo}CboCount	：	コンボボックスに設定されている項目の個数　　（０～，エラー時は-1 を返す)
AjcGet･･･{Lst/Cbo}CboLen	：	指定されたコンボボックス項目文字長（終端(0x00)は含まない，エラー時は０を返す)
AjcGet･･･{Lst/Cbo}Item	：	取得したコンボボックス項目の文字長（終端(0x00)は含まない，エラー時は０を返す)
AjcGet･･･{Lst/Cbo}Data	：	指定したコンボボックスに関連付けられている数値
AjcGet･･･{Lst/Cbo}Find	：	指定文字列と一致するコンボボックス項目のインデクス（０～，見つからない場合は-1 を返す)
AjcGet･･･{Lst/Cbo}MaxLen	：	コンボボックス項目群の最大文字列長
AjcGet･･･{Lst/Cbo}Height	：	コンボボックス項目の高さ（ピクセル数)
AjcGet･･･CboEditHandle	：	コンボボックス（CBS_DROPDOWN スタイル）内のテキストボックス(EDIT コントロール)のハンドル
AjcGet･･･SldPos	：	スライダの現在位置
その他	：	文字列表現の整数／実数値

17.4.2. ダイアログボックス項目の設定 (AjcSetDlgItem･･･／AjcSepDlgItem･･･／AjcSetCtrl･･･／AjcSepCtrl･･･)

形　式　：　**ダイアログボックス項目の設定**

```
// １０進数（区切り記号なし）／１６進数（桁揃えなし）
BOOL   AjcSetDlgItemUInt(HWND hDlg, int id, UI     value); ---------- 符号なし１０進整数
BOOL   AjcSetDlgItemSInt(HWND hDlg, int id, SI     value); ---------- 符号つき１０進整数
BOOL   AjcSetDlgItemHex (HWND hDlg, int id, UI     value); ---------- １６進整数
BOOL   AjcSetDlgItemReal(HWND hDlg, int id, double value, int prec); - 実数
BOOL   AjcSetDlgItemUI64(HWND hDlg, int id, ULL    value); ---------- 符号なし１０進整数（６４ビット）
BOOL   AjcSetDlgItemSI64(HWND hDlg, int id, SLL    value); ---------- 符号つき１０進整数（６４ビット）
BOOL   AjcSetDlgItemH64 (HWND hDlg, int id, ULL    value); ---------- １６進整数（６４ビット）
// １０進数（区切り記号あり）／１６進数（桁揃えあり）
BOOL   AjcSepDlgItemUInt(HWND hDlg, int id, UI     value); ---------- 符号なし１０進整数
BOOL   AjcSepDlgItemSInt(HWND hDlg, int id, SI     value); ---------- 符号つき１０進整数
BOOL   AjcSepDlgItemHex (HWND hDlg, int id, UI     value, int col ); - １６進整数
BOOL   AjcSepDlgItemReal(HWND hDlg, int id, double value, int prec); - 実数
BOOL   AjcSepDlgItemUI64(HWND hDlg, int id, ULL    value); ---------- 符号なし１０進整数（６４ビット）
BOOL   AjcSepDlgItemSI64(HWND hDlg, int id, SLL    value); ---------- 符号つき１０進整数（６４ビット）
BOOL   AjcSepDlgItemH64 (HWND hDlg, int id, ULL    value, int col ); - １６進整数（６４ビット）
// その他
BOOL   AjcSetDlgItemStr     (HWND hDlg, int id, C_UTP pStr); ---------- 文字列設定
BOOL   AjcSetDlgItemFStr    (HWND hDlg, int id, C_UTP pFmt, ...); -------- 書式文字列設定
BOOL   AjcSetDlgItemEdtLimit(HWND hDlg, int id, UI limit); ---------- エディットコントロールの入力可能文字数
BOOL   AjcSetDlgItemChk     (HWND hDlg, int id, UI state); ---------- チェックボックス／ラジオボタン状態
BOOL   AjcSetDlgItemRbt     (HWND hDlg, int id, int idFirst, int idLast); - ラジオボタンの選択

BOOL   AjcSetDlgItemCboIx   (HWND hDlg, int id, UI ix); ---------- コンボボックスの項目選択
UI     AjcSetDlgItemCboAdd  (HWND hDlg, int id, UI ix, C_UTP pItem, UI flag); - コンボボックスの項目を追加
UI     AjcSetDlgItemCboIns  (HWND hDlg, int id, UI ix, C_UTP pItem, UI flag); - コンボボックスの項目を挿入
BOOL   AjcSetDlgItemCboDel  (HWND hDlg, int id, UI ix); ---------- コンボボックスの項目を削除
BOOL   AjcSetDlgItemCboData (HWND hDlg, int id, UI ix, SX data); ---------- コンボボックス項目に関連付ける数値
BOOL   AjcSetDlgItemCboList (HWND hDlg, int id, C_UTP pList); ---------- コンボボックスの項目群を一括設定
BOOL   AjcSetDlgItemCboReset(HWND hDlg, int id); ---------- コンボボックスの全項目リセット
BOOL   AjcSetDlgItemCboHeight(HWND hDlg, int id, UI ix, UI height); ------- コンボボックス項目の高さ設定
BOOL   AjcSetDlgItemCboLimit(HWND hDlg, int id, UI limit); ---------- コンボボックスの入力可能文字数
BOOL   AjcSetDlgItemCboLimit(HWND hDlg, int id, UI limit); ---------- コンボボックスの入力可能文字数

BOOL   AjcSetDlgItemLstIx   (HWND hDlg, int id, UI ix); ---------- リストボックスの項目選択
UI     AjcSetDlgItemLstAdd  (HWND hDlg, int id, UI ix, C_UTP pItem, UI flag); - リストボックスの項目を追加
UI     AjcSetDlgItemLstIns  (HWND hDlg, int id, UI ix, C_UTP pItem, UI flag); - リストボックスの項目を挿入
BOOL   AjcSetDlgItemLstDel  (HWND hDlg, int id, UI ix); ---------- リストボックスの項目を削除
BOOL   AjcSetDlgItemLstData (HWND hDlg, int id, UI ix, SX data); ---------- リストボックス項目に関連付ける数値
BOOL   AjcSetDlgItemLstList (HWND hDlg, int id, C_UTP pList); ---------- リストボックスの項目群を一括設定
BOOL   AjcSetDlgItemLstReset(HWND hDlg, int id); ---------- リストボックスの全項目リセット
BOOL   AjcSetDlgItemLstHeight(HWND hDlg, int id, UI ix, UI height); ------- リストボックス項目の高さ設定

BOOL   AjcSetDlgItemPgsRange(HWND hDlg, int id, int low, int high); ---------- プログレスバーのレンジ
BOOL   AjcSetDlgItemPgsPos  (HWND hDlg, int id, int pos); ---------- プログレスバーの位置
BOOL   AjcSetDlgItemSldRange(HWND hDlg, int id, int low, int high, int page); --- スライダのレンジ
BOOL   AjcSetDlgItemSldPos  (HWND hDlg, int id, int pos); ---------- スライダの位置
BOOL   AjcSetDlgItemSpnInfo  (HWND hDlg, int id, int low, int high, int idBuddy); - スピンボタンの
                                                                          レンジとバディウインド
BOOL   AjcSetDlgItemSpnRange (HWND hDlg, int id, int low, int high); ---------- スピンボタンのレンジ
BOOL   AjcSetDlgItemSpnBuddy (HWND hDlg, int id, int idBuddy); ---------- スピンボタンのバディウインド
```

ウインド項目の設定

```
// １０進数（区切り記号なし）／１６進数（桁揃えなし）
BOOL   AjcSetCtrlUInt(HWND hwnd, UI     value); ---------- 符号なし１０進整数
BOOL   AjcSetCtrlSInt(HWND hwnd, SI     value); ---------- 符号つき１０進整数
BOOL   AjcSetCtrlHex (HWND hwnd, UI     value); ---------- １６進整数
BOOL   AjcSetCtrlReal(HWND hwnd, double value, int prec); - 実数
BOOL   AjcSetCtrlUI64(HWND hwnd, ULL    value); ---------- 符号なし１０進整数（６４ビット）
BOOL   AjcSetCtrlSI64(HWND hwnd, SLL    value); ---------- 符号つき１０進整数（６４ビット）
BOOL   AjcSetCtrlH64 (HWND hwnd, ULL    value); ---------- １６進整数（６４ビット）
// １０進数（区切り記号あり）／１６進数（桁揃えあり）
BOOL   AjcSepCtrlUInt(HWND hwnd, UI     value); ---------- 符号なし１０進整数
BOOL   AjcSepCtrlSInt(HWND hwnd, SI     value); ---------- 符号つき１０進整数
BOOL   AjcSepCtrlmHex (HWND hwnd, UI     value, int col ); - １６進整数
BOOL   AjcSepCtrlReal(HWND hwnd, double value, int prec); - 実数
BOOL   AjcSepCtrlUI64(HWND hwnd, ULL    value); ---------- 符号なし１０進整数（６４ビット）
BOOL   AjcSepCtrlSI64(HWND hwnd, SLL    value); ---------- 符号つき１０進整数（６４ビット）
BOOL   AjcSepCtrlH64 (HWND hwnd, ULL    value, int col ); - １６進整数（６４ビット）
// その他
BOOL   AjcSetCtrlStr     (HWND hwnd, C_UTP pStr); ---------------------- 文字列設定
BOOL   AjcSetCtrlFStr    (HWND hwnd, C_UTP pFmt, ...); ----------------- 書式文字列設定
BOOL   AjcSetCtrlEdtLimit(HWND hwnd, UI limit); ------------------------ エディットコントロールの入力可能文字数
BOOL   AjcSetCtrlChk     (HWND hwnd, UI state); ------------------------ チェックボックス／ラジオボタン状態

BOOL   AjcSetCtrlCboIx   (HWND hwnd, UI ix); ---------------------------- コンボボックスの項目選択
UI     AjcSetCtrlCboAdd  (HWND hwnd, UI ix, C_UTP pItem, UI flag); ---- コンボボックスの項目を追加
UI     AjcSetCtrlCboIns  (HWND hwnd, UI ix, C_UTP pItem, UI flag); ---- コンボボックスの項目を挿入
BOOL   AjcSetCtrlCboDel  (HWND hwnd, UI ix); ---------------------------- コンボボックスの項目を削除
BOOL   AjcSetCtrlCboData (HWND hwnd, UI ix, SX data); ------------------ コンボボックス項目に関連付ける数値
BOOL   AjcSetCtrlCboList (HWND hwnd, C_UTP pList); ---------------------- コンボボックスの項目群を一括設定
BOOL   AjcSetCtrlCboReset(HWND hwnd); ----------------------------------- コンボボックスの全項目リセット
BOOL   AjcSetCtrlCboHeight(HWND hwnd, UI ix, UI height); --------------- コンボボックス項目の高さ設定
BOOL   AjcSetCtrlCboLimit(HWND hwnd, UI limit); ------------------------ コンボボックスの入力可能文字数

BOOL   AjcSetCtrlLstIx   (HWND hwnd, UI ix); ---------------------------- リストボックスの項目選択
UI     AjcSetCtrlLstAdd  (HWND hwnd, UI ix, C_UTP pItem, UI flag); ---- リストボックスの項目を追加
UI     AjcSetCtrlLstIns  (HWND hwnd, UI ix, C_UTP pItem, UI flag); ---- リストボックスの項目を挿入
BOOL   AjcSetCtrlLstDel  (HWND hwnd, UI ix); ---------------------------- リストボックスの項目を削除
BOOL   AjcSetCtrlLstData (HWND hwnd, UI ix, SX data); ------------------ リストボックス項目に関連付ける数値
BOOL   AjcSetCtrlLstList (HWND hwnd, C_UTP pList); ---------------------- リストボックスの項目群を一括設定
BOOL   AjcSetCtrlLstReset(HWND hwnd); ----------------------------------- リストボックスの全項目リセット
BOOL   AjcSetCtrlLstHeight(HWND hwnd, UI ix, UI height); --------------- リストボックス項目の高さ設定

BOOL   AjcSetCtrlPgsRange(HWND hwnd, int low, int high); --------------- プログレスバーのレンジ
BOOL   AjcSetCtrlPgsPos  (HWND hwnd, int pos); -------------------------- プログレスバーの位置
BOOL   AjcSetCtrlSldRange(HWND hwnd, int low, int high, int page); ---- スライダのレンジ
BOOL   AjcSetCtrlSldPos  (HWND hwnd, int pos); -------------------------- スライダの位置
BOOL   AjcSetCtrlSpnInfo (HWND hwnd, int low, int high, int idBuddy); - スピンボタンのレンジとバディウインド
BOOL   AjcSetCtrlSpnRange(HWND hwnd, int low, int high); --------------- スピンボタンのレンジ
BOOL   AjcSetCtrlSpnBuddy(HWND hwnd, int idBuddy); ---------------------- スピンボタンのバディウインド
```

引　数 ： hDlg　- ダイアログボックスのハンドル
id　- コントロールの識別番号
hwnd　- コントロールのウインドハンドル
idFirst - グループ内の最初のラジオボタンの識別子
idLast　- グループ内の最後のラジオボタンの識別子
value　- 設定する整数／実数値
col　- １６進数の表示桁数（先頭の「0x」を除く）
prec　- 少数部の桁数（負数の場合は、数値の有効桁数）
pStr　- 設定する文字列
pFmt　- 書式文字列（形式は printf()と同じ）
state　- 設定するチェックボックスの状態（０：チェックなし，１：チェックあり，２：不定）
ix　- 対象とするリストボックス／コンボボックス項目のインデクス（０～）
　AjcSet・・・CboHeight では、「－１」を」指定した場合、コンボボックスの選択部を意味します。(CBS_DROPDOWN 時)
pItem　- リストボックス／コンボボックスに設定する項目（文字列）のアドレス
fExact　- 英文字の比較方法（TRUE:大小文字を区別する，FALSE:大小文字を区別しない）
data　- リストボックス／コンボボックス項目に関連付ける数値
pList　- リストボックス／コンボボックスに設定する項目群（多重文字列）のアドレス
height　- リストボックス／コンボボックスの高さ（ピクセル数，　＝０ ： フォントの高さ＋２を設定）
pos　- スライダ／プログレスバーの設定位置
low　- スライダ／プログレスバー／スピンボタンのレンジ（最小値）
high　-　〃　〃　（最大値）
page　- スライダのページサイズ（つまみの左右のライン上をクリックした時の増減値）
idBuddy - スピンボタンのバディウインド（テキストコントロール）の識別番号
flag　- 文字列の比較パラメタ。以下の値の合成値（あるいは０）を指定する。

指定値	指定時	未指定時	摘用		
			Find	Add	Ins
AJCCBF_SORT	CBS_SORT スタイル時： 　昇順に並べ替えられます。 CBS_SORT スタイル無し時： 　末尾に項目を追加します。 ix の指定は無視されます。	ix で指定された位置に項目を挿入します。ix=-1 の場合は、末尾に追加します。	－	●	－
AJCCBF_EXACT_STR AJCCBF_EXACT	文字列全体を比較	文字列の先頭部分だけを比較	●	－	－
AJCCBF_EXACT_WIDTH AJCCBF_RECOGNIZE_WIDTH AJCCBF_DISTINGUISH	文字列比較で英大小文字を区別する	文字列比較で英大小文字を区別しない	●	●	●
AJCCBF_EXCLUSION	同一項目存在時は追加／挿入しない。 AJCCBF_SAME_ERR 未指定ならば当該項目のインデクスを返す	同一項目存在時でも追加／挿入する	－	●	●
AJCCBF_SAME_ERR	同一項目存在時はエラーとする	－	－	●	●
AJCCBF_SELECT	追加／挿入した項目、あるいは既に存在する項目を選択状態にする	選択状態は変更しない	●	●	●
AJCCBF_ALL	AJCCBF_SAME_ERR を除く上記全項目の合成値	－	－	●	●
AJCCBF_ALL_NS	AJCCBF_SORT と AJCCBF_SAME_ERR を除く上記全項目の合成値	－	－	●	●

※「摘要」は、フラグの指定が有効な関数（AjcGet・・・CboFind, AjcSet・・・CboAdd or AjcSet・・・CboIns）を示します。
※ AJCCBF_SAME_ERR は、AJCCBF_EXCLUSION 指定時のみ有効です。(「AJCCBF_EXCLUSION| AJCCBF_SAME_ERR」で同一項目存在時エラーとします)

説　明 ： ダイアログボックス／ウインドの各種コントロールの内容を設定します。

AjcSet・・・UInt/SInt/Real/UI64/SI64 では、value を１０進数表現の文字列に変換したテキストをコントロールに設定します。(ex. AjcSetDlgItemUInt(1234567, id)； → "1234567")

AjcSep・・・UInt/SInt/Real/UI64/SI64 では、value を、整数部を特定桁数毎に区切り記号（カンマ（,）等）で区切った 10 進数表現の文字列に変換したテキストをコントロールに設定します。(ex. AjcSepDlgItemUInt(123456, id)； → "1,234,567")

AjcSet・・・Hex/H64 では、value を、先頭に「0x」を付加した１６進表現の文字列に変換したテキストをコントロールに設定します。(ex. AjcSetDlgItemHex(0x789ABC, id)； → "0x789ABC")

AjcSep・・・Hex/H64 では、value を、先頭に「0x」を付加した１６進表現の文字列に変換したテキストをコントロールに設定しますが、１６進数が col で指定された桁数に満たない場合は、先頭に「0」を付加し、桁揃えします。
(ex. AjcSepDlgItemHex(0x789ABC, id, 8)； → "0x00789ABC")

１０進数での区切り記号と区切り桁数（３桁毎のカンマ（,）等）は、ロケール情報に依存します。

AjcSet…{Lst/Cbo}List では、設定するリストボックス／コンボボックスの項目群を多重文字列で指定します。
AjcSet…{Lst/Cbo}List では、リストボックス／コンボボックスの全項目をリセットした後に、指定された項目群を設定します。

AjcSet…{Lst/Cbo}Add では、リストボックス／コンボボックスに項目を追加します。
項目追加の際のパラメタを「flag」引数で指定します。(AJCCBF_SORT / EXACT_WIDTH / EXCLUSION / SAME_IX / SELECT)
AJCCBF_SORT が指定された場合、LBS_SORT/CBS_SORT スタイルが設定されている場合は、昇順に追加され、LBS_SORT/CBS_SORT スタイルが設定されていない場合は、末尾に追加されます。
AJCCBF_SORT が指定されていない場合は、ix で指定された位置に挿入されます。

AjcSet…{Lst/Cbo}Ins では、リストボックス／コンボボックスに項目を挿入します。
項目追加の際のパラメタを「flag」引数で指定します。(AJCCBF_EXACT_WIDTH / EXCLUSION / SAME_IX / SELECT)
通常は ix で指定された位置に挿入しますが、ix に以下の定数を指定することもできます。

- AJCCBI_APPEND　－　末尾に挿入
- AJCCBI_ASCENDING　－　昇順となる項目位置に挿入（英大小文字区別あり）
- AJCCBI_DESCENDING　－　降順となる項目位置に挿入（英大小文字区別あり）

AjcSet…{Lst/Cbo}Add と AjcSet…{Lst/Cbo}List では、コンボボックスに設定する文字列と関連付ける数値を同時に指定することができます。
関連付ける数値を指定するには、設定する項目の文字列をタブ（¥t）で終端し、その後に数値(１０進数/１６進数)を指定します。数値の前後には空白を置くことができます。
尚、１６進数を指定する場合は先頭に「0x」を付けます。

```
例)　AjcSetDlgItemCboList(hDlg, id, ″ 東京都 ¥t  03 ¥0″
                                   ″ 横浜市 ¥t 045 ¥0″
                                   ″ 川崎市 ¥t 044 ¥0″);
```

この例では、コンボボックスに３つの項目「″ 東京都 ″」「″ 横浜市 ″」「″ 川崎市 ″」が設定され、AjcGetDlgItemCboData では、それぞれ「3」「45」「44」という数値が取得されます。

戻り値　：　AjcSet…{Lst/Cbo}Add　：追加した位置のインデクス（０～，エラー時は -1 を返す）
AjcSet…{Lst/Cbo}Ins　：挿入した位置のインデクス（０～，エラー時は -1 を返す）
その他　：TRUE　- 成功，　FALSE - 失敗

17.4.3. EDIT コントロールにフォルダやファイルをドロップ可能にする(AjcEnable{DlgItem/Ctrl}ToDrop)

形　式 ： BOOL AjcEnableDlgItemToDrop (HWND hDlg, int id, UI optDrop);
BOOL AjcEnableDlgItemToDropEx(HWND hDlg, int id, int idGrp, UI optDrop);
BOOL AjcEnableCtrlToDrop (HWND hwnd, UI optDrop);
BOOL AjcEnableCtrlToDropEx (HWND hwnd, HWND hGrp, UI optDrop);

引　数 ： hDlg　- ダイアログボックスのハンドル
id　- EDIT コントロールの識別番号
idGrp　- EDIT コントロールを覆うグループボックスの識別番号（無い場合はゼロ）
hwnd　- EDIT コントロールのウインドハンドル
hGrp　- EDIT コントロールを覆うグループボックスのウインドハンドル（無い場合は NULL）
optDrop - ドロップ動作の指定(AJCDROP_*XXXXX*)

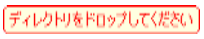
説　明 ： 指定された EDIT コントロールに（エクスプローラから）ディレクトリやファイルをドロップできるようにします。
ドロップした場合、EDIT コントロールにディレクトリやファイルのパス名が設定されます。
EDIT コントロールがグループボックス内に配置されている場合は、idGrp / hGrp でグループボックスを指定します。
EDIT コントロールを idGrp/hGrp で指定したグループボックスの前面に設定します。

optDrop には以下のシンボルの合成値を指定します。

シンボル／値	内容	備考
AJCDROP_DIR	ディレクトリをドロップ可能にする	いずれか片方／両方 を指定要
AJCDROP_FILE	ファイルをドロップ可能にする	
AJCDROP_BOTH AJCDROP_DIR_AND_FILE	AJCDROP_DIR と、AJCDROP_FILE の合成値	
AJCDROP_CONNECT	複数個ドロップを許可	全項目をセミコロン(;)で区切って連結
AJCDROP_DIRTAIL	ディレクトリの場合末尾に「¥」を付加する	
AJCDROP_NOERRTIP	目的のオブジェクト（ディレクトリ／ファイル）がドロップされない場合、エラーチップを表示しない	
AJCDROP_VOLEXP	元テキストの先頭が '<' の場合、ボリューム表現のパス名に変換	ボリューム表現のパス名については「ボリュームラベル操作」章参照
AJCDROP_RMVONLY	ボリューム表現のパス名に変換をリムーバブルディスクに限定	
AJCDROP_VOLEXPRO	上記２つの合成値	

AJCDROP_DIR / AJCDROP_FILE を片方だけ指定した場合で、指定したオブジェクト（ディレクトリ／ファイル）がドロップされない場合は、以下のエラーチップを表示します。(但し、AJCDROP_NOERRTIP を指定した場合は表示しません)

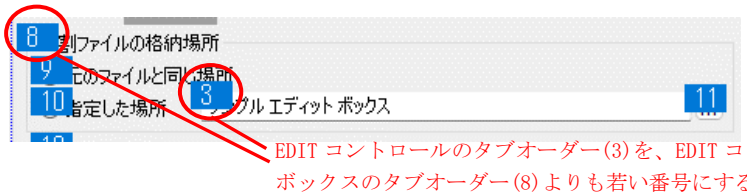
・AJCDROP_FILE を指定し、ディレクトリをドロップした場合　 

・AJCDROP_DIR　を指定し、ファイルをドロップした場合 ----　 

AJCDROP_CONNECT を指定した場合、複数のディレクトリやファイルをドロップした際に、これらのパス名をセミコロン(;)で連結した文字列を設定します。(ex. “d:¥wor¥aaa.txt;d:¥wor¥bbb.txt;d:¥wor¥ccc.txt”)
AJCDROP_CONNECT を指定しないで複数のディレクトリやファイルをドロップした場合、最初のディレクトリ／ファイルのパス名が設定されます。

戻り値 ： TRUE　- 成功
FALSE - 失敗

備　考 ： ドロップ可能とする EDIT コントロールは、EDIT コントロールを覆うグループボックスより Z オーダーが前面に設定されている必要があります。
つまり、EDIT コントロールのタブオーダーをグループボックスより若い番号に設定する必要があります。



AjcEnableDlgItemToDropEx()と、AjcEnableCtrlToDropEx()は、実行時に動的にタブオーダー（Z オーダー）を変更します。

17.4.4. コントロールの位置／サイズ／矩形取得 (Ajc{Get/Set}{DlgItem/Ctrl}{Pos/Size/Rect}

形　式 ： ［ダイアログボックス項目］

```
BOOL AjcGetDlgItemPos (HWND hDlg, int id, LPPOINT pPoint);          - 位置取得
BOOL AjcGetDlgItemSize(HWND hDlg, int id, LPSIZE  pSize);           - サイズ取得
BOOL AjcGetDlgItemRect(HWND hDlg, int id, LPRECT  pRect);           - 矩形取得

BOOL AjcSetDlgItemPos (HWND hDlg, int id, int x, int y);            - 位置設定
BOOL AjcSetDlgItemPosV(HWND hDlg, int id, const POINT *pPos);       -  〃
BOOL AjcSetDlgItemSize(HWND hDlg, int id, int cx, int cy);          - サイズ設定
BOOL AjcSetDlgItemSizeV(HWND hDlg, int id, const SIZE *pSize);      -  〃
BOOL AjcSetDlgItemRect(HWND hDlg, int id, int x, int y, int cx, int cy);  - 矩形設定
BOOL AjcSetDlgItemRectV(HWND hDlg, int id, const RECT *pRect);      -  〃
```

［コントロール］

```
BOOL AjcGetCtrlPos (HWND hwnd, LPPOINT pPoint);                     - 位置取得
BOOL AjcGetCtrlSize(HWND hwnd, LPSIZE  pSize);                      - サイズ取得
BOOL AjcGetCtrlRect(HWND hwnd, LPRECT  pRect);                      - 矩形取得

BOOL AjcSetCtrlPos  (HWND hwnd, int x, int y);                      - 位置設定
BOOL AjcSetCtrlPosV (HWND hwnd, const POINT *pPos);                 -  〃
BOOL AjcSetCtrlSize (HWND hwnd, int cx, int cy);                    - サイズ設定
BOOL AjcSetCtrlSizeV(HWND hwnd, const SIZE *pSize);                 -  〃
BOOL AjcSetCtrlRect (HWND hwnd, int x, int y, int cx, int cy);      - 矩形設定
BOOL AjcSetCtrlRectV(HWND hwnd, const RECT *pRect);                 -  〃
```

引　数 ： hDlg　- ダイアログボックスのハンドル
id　- コントロールの識別番号
hwnd　- コントロールのウインドハンドル
pPoint　- コントロールの位置
pSize　- コントロールのサイズ
pRect　- コントロールの矩形
x, y　- コントロールの位置
cx, cy　- コントロールのサイズ

説　明 ： 指定されたコントロールの親ウインド（クライアント領域）上での位置，サイズや矩形情報を取得／設定します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

17.4.5. ダイアログ／ウインド項目の有効化／無効化 (AjcEnableDlgItem／AjcEnableCtrl)

形　式 ： BOOL AjcEnableDlgItem(HWND hDlg, int id, BOOL fEnable);
BOOL AjcEnableCtrl (HWND hwnd, BOOL fEnable);

引　数 ： hDlg - ダイアログボックスのハンドル
id - 許可／禁止するコントロールの識別番号
hwnd - コントロールのウインドハンドル
fEnable - FALSE : コントロールをグレー表示し、入力を受け付けなくする
TRUE : コントロールを通常表示し、入力を受け付けるようにする

説　明 ： 指定されたコントロールを有効化／無効化します。
fEnable=FALSE とした場合は、当該ダイアログ項目をグレー表示し、入力を受け付けないようにします。
fEnable=TRUE とした場合は、通常表示（グレー表示を解除）し入力を受け付けるようにします。

戻り値 ： TRUE - 成功
FALSE - 失敗

17.4.6. ダイアログ／ウインド項目の表示／非表示 (AjcShowDlgItem／AjcShowCtrl)

形　式 ： BOOL AjcShowDlgItem (HWND hDlg, int id, BOOL fShow);
BOOL AjcShowCtrl (HWND hwnd, BOOL fShow);

引　数 ： hDlg - ダイアログボックスのハンドル
id - 表示／非表示するコントロールの識別番号
hwnd - 表示／非表示するコントロールのウインドハンドル
fShow - FALSE : 非表示
TRUE : 表示

説　明 ： 指定されたコントロールを表示／非表示します。

戻り値 ： TRUE - 成功
FALSE - 失敗

17.4.7. ダイアログ／ウインド項目の表示／非表示と有効化／無効化 (AjcShowAndEnableDlgItem／AjcShowAndEnableCtrl)

形　式 ： BOOL AjcShowAndEnableDlgItem (HWND hDlg, int id, BOOL fShow, BOOL fEnable);
BOOL AjcShowAndEnableCtrl (HWND hwnd, BOOL fShow, BOOL fEnable);

引　数 ： hDlg - ダイアログボックスのハンドル
id - 表示／非表示するコントロールの識別番号
hwnd - 表示／非表示するコントロールのウインドハンドル
fShow - FALSE : 非表示
TRUE : 表示
fEnable - FALSE : コントロールをグレー表示し、入力を受け付けなくする
TRUE : コントロールを通常表示し、入力を受け付けるようにする

説　明 ： 指定されたコントロールを表示／非表示および有効化／無効化します。

戻り値 ： TRUE - 成功
FALSE - 失敗

17.4.8. グループボックス内の全コントロール有効化／無効化 (AjcEnableDlgGroup／AjcEnableGroup)

形　式 ： BOOL AjcEnableDlgGroup (HWND hDlg, int idGrp, BOOL fEnaGrpBox, BOOL fEnaCtrls);
BOOL AjcEnableGroup (HWND hwnd, BOOL fEnaGrpBox, BOOL fEnaCtrls);

引　数 ： hDlg - ダイアログボックスのハンドル
idGrp - 許可／禁止するグループボックス・コントロールの識別番号
hwnd - グループボックス・コントロールのウインドハンドル
fEnaGrp - FALSE ： グループボックスをグレー表示し、入力を受け付けなくする
TRUE ： グループボックスを通常表示し、入力を受け付けるようにする
fEnaCtrls - FALSE ： グループボックス内の全コントロールをグレー表示し、入力を受け付けなくする
TRUE ： グループボックス内の全コントロールを通常表示し、入力を受け付けるようにする

説　明 ： グループボックス、および、グループボックス内の全コントロールを有効／無効化します。
fEnaGrp は、グループボックス自体の有効化／無効化を指定するフラグです。
fEnaCtrls は、グループボックス内の全コントロールを有効化／無効化するフラグです。

尚、idGrp は、グループボックス・コントロールに限らず、他のコントロールを囲んでいるコントロールであれば、どのようなコントロールでもＯＫです。

戻り値 ： TRUE - 成功
FALSE - 失敗

17.4.9. ウインド内の全コントロール有効化／無効化 (AjcEnableCtrlsInWnd)

形　式 ： AJCEXPORT BOOL WINAPI AjcEnableCtrlsInWnd(HWND hwnd, BOOL fEnaCtrls);

引　数 ： hwnd - ウインド・ハンドル
fEnaCtrls - FALSE ： ウインド内の全コントロールをグレー表示し、入力を受け付けなくする
TRUE ： ウインド内の全コントロールを通常表示し、入力を受け付けるようにする

説　明 ： ウインド内の全コントロールを有効／無効化します。
fEnaCtrls は、ウインド内の全コントロールを有効化（グレー解除）／無効化（グレー化）するフラグです。

戻り値 ： TRUE - 成功
FALSE - 失敗

17.4.10. グループボックス内の全コントロール表示／非表示［１］(AjcShowDlgGroup／AjcShowGroup)

形　式 ： BOOL AjcShowDlgGroup (HWND hDlg, int idGrp, BOOL fSHow);
BOOL AjcShowGroup (HWND hwnd, BOOL fShow);

引　数 ： hDlg - ダイアログボックスのハンドル
idGrp - 表示／非表示するグループボックス・コントロールの識別番号
hwnd - グループボックス・コントロールのウインドハンドル
fShow - FALSE ： グループボックスとグループボックス内の全コントロールを非表示にする
TRUE ： グループボックスとグループボックス内の全コントロールを表示（可視状態）にする

説　明 ： グループボックスとグループボックス内の全コントロールを表示／非表示します。
尚、idGrp は、グループボックス・コントロールに限らず、他のコントロールを囲んでいるコントロールであれば、どのようなコントロールでもＯＫです。

戻り値 ： TRUE - 成功
FALSE - 失敗

17.4.11. グループボックス内の全コントロール表示／非表示［２］(AjcShowDlgGroupEx／AjcShowGroupEx)

形　式 ： BOOL　AjcShowDlgGroupEx (HWND hDlg, int idGrp, BOOL fSHowGrpBox, BOOL fShowCtrls);
BOOL　AjcShowGroupEx　　(HWND hwnd, BOOL fSHowGrpBox, BOOL fShowCtrls);

引　数 ： hDlg　　- ダイアログボックスのハンドル
idGrp　　- 表示／非表示するグループボックス・コントロールの識別番号
hwnd　　- グループボックス・コントロールのウインドハンドル
fShowGrpBox - FALSE : グループボックスを非表示にする
TRUE　: グループボックスを表示（可視状態）にする
fShowCtrls　- FALSE : グループボックス内の全コントロールを非表示にする
TRUE　: グループボックス内の全コントロールを表示（可視状態）にする

説　明 ： グループボックスやグループボックス内の全コントロールを表示／非表示します。
尚、idGrp は、グループボックス・コントロールに限らず、他のコントロールを囲んでいるコントロールであれば、どのようなコントロールでもＯＫです。

戻り値 ： TRUE　- 成功
FALSE - 失敗

17.4.12. グループボックス内の全コントロールの移動(AjcMoveDlgGroupToLoc/Ctl/Org)

形　式 ： BOOL　AjcMoveDlgGroupToLoc(HWND hDlg, int idGrp, int x, int y);
BOOL　AjcMoveDlgGroupToCtl(HWND hDlg, int idGrp, int idCtl);
BOOL　AjcMoveDlgGroupToOrg(HWND hDlg, int idGrp);

BOOL　AjcMoveGroupToLoc(HWND hwnd, int x, int y);
BOOL　AjcMoveGroupToCtl(HWND hwnd, HWND hctl);
BOOL　AjcMoveGroupToOrg(HWND hwnd);

引　数 ： hDlg　　- ダイアログボックスのハンドル
idGrp　　- 表示／非表示するグループボックス・コントロールの識別番号
x、y　　- 移動先の位置
idCtl　　- 移動先コントロールの識別番号

hwnd　　- グループボックス・コントロールのウインドハンドル
hctl　　- 移動先コントロールのウインドハンドル

説　明 ： グループボックスとグループボックス内の全兄弟コントロールを指定された位置へ移動します。
「AjcMoveDlgGroupTo**Loc**／AjcMoveGroupTo**Loc**」は、x,y で指定された位置へ移動します。
「AjcMoveDlgGroupTo**Ctl**／AjcMoveGroupTo**Ctl**」は、idCtl/hctl で指定されたコントロールの位置へ移動します。
「AjcMoveDlgGroupTo**Org**／AjcMoveGroupTo**Org**」は、「・・・Loc()」や「・・・Ctl()」で移動する前の位置に戻ります。

尚、idGrp/hwnd は、グループボックス・コントロールに限らず、他のコントロールを囲んでいるコントロールであれば、どのようなコントロールでもＯＫです。

戻り値 ： TRUE　- 成功
FALSE - 失敗

17.4.13. グループボックス内の全コントロール列挙(AjcDlgItemEnumInGroup／AjcCtrlEnumInGroup)

形　式 ： UI AjcDlgItemEnumInGroup (HWND hDlg, int idGrp, UX cbp, BOOL (CALLBACK *cbEnum)(HWND hCtrl, UX cbp));
UI AjcCtrlEnumInGroup (HWND hwnd, UX cbp, BOOL (CALLBACK *cbEnum)(HWND hCtrl, UX cbp));

引　数 ： hDlg - ダイアログボックスのハンドル
idGrp - グループボックス・コントロールの識別番号
hwnd - グループボックス・コントロールのウインドハンドル
cbEnum - コントロール地一葉コールバック関数

説　明 ： グループボックス内の全コントロールを列挙します。

戻り値 ： 列挙したコントロールの個数

コールバック ：

cbEnum（コントロールの通知）

形　式 ： BOOL CALLBACK *cbEnum* (HWND hCtrl, UX cbp)

引　数 ： hCtrl - コントロールのウインドハンドル
cbp - コールバックパラメタ

説　明 ： コントロールのハンドルを通知します。

戻り値 ： TRUE ：列挙を継続
FALSE ：列挙を中止

17.4.14. ウインド／グループ内全コントロールの設定内容を永続化(Ajc{Load/Save}{All/Grp}ControlSettings)

形　式 ： BOOL AjcLoadAllControlSettings (HWND hwnd, C_UTP pSect, UI act); ── ウインド内全コントロール読み出し
BOOL AjcSaveAllControlSettings (HWND hwnd); ---------- ウインド内全コントロール書き込み
BOOL AjcSaveAllControlSettingsEx(HWND hwnd, C_UTP pSect, UI act); ── ウインド内全コントロール書き込み

BOOL AjcLoadGrpControlSettings (HWND hwnd, HWND hGrp, C_UTP pSect, UI act); - グループ内全コントロール読み出し
BOOL AjcSaveGrpControlSettings (HWND hwnd, HWND hGrp); ---------- グループ内全コントロール書き込み
BOOL AjcSaveGrpControlSettingsEx(HWND hwnd, HWND hGrp, C_UTP pSect, UI act); - グループ内全コントロール書き込み

BOOL AjcDlgLoadGrpControlSettings (HWND hDlg, int idGrp, C_UTP pSect, UI act);── グループ内全コントロール読み出し
BOOL AjcDlgSaveGrpControlSettings (HWND hDlg, int idGrp); ---------- グループ内全コントロール書き込み
BOOL AjcDlgSaveGrpControlSettingsEx(HWND hDlg, int idGrp, C_UTP pSect, UI act);- グループ内全コントロール書き込み

引　数 ： hwnd - 各コントロールの親ウインドのハンドル
hGrp - グループボックスのウインドハンドル（NULL 指定時は、ウインド内全コントロールを対象とする）
hDlg - ダイアログボックスハンドル
idGrp - グループボックスのコントロールＩＤ（ゼロ指定時は、ダイアログ内全コントロールを対象とする
pSect - プロファイルのセクション名文字列へのポインタ
act - 永続化動作指定（AJCCTL_SELACT_*XXXXXX*）

説　明 ： hwnd/hDlg で指定されたウインドの全ての子コントロール（hGrp/idGrp 指定時は、当該グループボックス内の全コントロール）について、各コントロールの設定値をプロファイルに書き込んだり、プロファイルから読み出すことにより、設定内容の永続化を行います。
通常は、開始時に Ajc[Dlg]Load～()を実行し、終了時に、（末尾に「Ex」の付かない）Ajc[Dlg]Save～()を実行します。
（Ajc[Dlg]Save～()の前に必ず、Ajc[Dlg]Load～()が実行されていなければなりません）

ex.
```
AjcLoadAllControlSettings (hDlg, "MySect", AJCCTL_SELACT_ ALL);                    // ダイアログ全体の永続化
        ・
        ・
AjcSaveAllControlSettings (hDlg);
```

```
AjcLoadGrpControlSettings (hDlg, IDC_GRP_XXXXX, "MySect", AJCCTL_SELACT_ ALL);   // グループの永続化
        ・
        ・
AjcSaveGrpControlSettings (hDlg, hGrp);
```

```
AjcDlgLoadGrpControlSettings (hDlg, IDC_GRP_XXXXX , "MySect", AJCCTL_SELACT_ ALL); // グループの永続化
        ・
        ・
AjcDlgSaveGrpControlSettings (hDlg);
```

セーブ時のプロファイル・セクション名，永続化動作指定は、ロード時の引数（pSect，act）と同じになります。
初回の Ajc[Dlg]Load～()では、（まだプロファイルは記録されていないので）各コントロールの設定値は変化しません。
Ajc[Dlg]Save～()実行後の Ajc[Dlg]Load～()では、プロファイルに記録された内容を読み出し、各コントロールに設定します。

Ajc[Dlg]Load～()を実行せずに、単独で各コントロールの設定値をプロファイルに記録するには、末尾に「Ex」の付いた Ajc[Dlg]Save～Ex()を実行します。

ex.
```
AjcSaveAllControlSettingsEx(hDlg, "MySect", AJCCTL_SELACT_ ALL);                    // ダイアログ全体の永続化
AjcSaveGrpControlSettingsEx(hDlg,  hGrp, "MySect", AJCCTL_SELACT_ ALL);             // グループの永続化
AjcDlgSaveGrpControlSettingsEx (hDlg, IDC_GRP_XXXXX, "MySect", AJCCTL_SELACT_ ALL); // グループの永続化
```

ここで、セクション名を NULL とし、永続化動作指定を「AJCCTL_SELACT_BYLLOAD」とした場合は、Ajc[Dlg]Load～()で指定したセクション名と永続化動作指定(act)を仮定します。
この場合、Ajc[Dlg]Save～Ex()の前に必ず、Ajc[Dlg]Load～()が実行されていなければなりません。

act は永続化に関する動作を指定します。

#	act	値	内容	AjcSetCtrlPermAtt()の指定／備考
1	AJCCTL_SELACT_ALL (※1)	0	全てのコントロールを対象とする（省略可）	AJCCTL_PSEL_INCLUDE/AJCCTL_PSEL_EXCLUDE は無視する
2	AJCCTL_SELACT_CHKEXCLUDE (※1)	0x0001	永続化除外としたコントロールは除外する	AJCCTL_PSEL_EXCLUDE で永続化対象外とするコントロールを指定する
3	AJCCTL_SELACT_CHKINCLUDE (※1)	0x0002	永続化対象としたコントロールのみを対象とする	AJCCTL_PSEL_INCLUDE で永続化対象とするコントロールを指定する
4	AJCCTL_SELACT_NTCCHK	0x0100	チェックボックスの読み出し時に BN_CLICKED イベントを発生する	(※3)
5	AJCCTL_SELACT_NTCRBT	0x0200	ラジオボタンの読み出し時に、チェックされているラジオボタンに BN_CLICKED イベントを発生する	
6	AJCCTL_SELACT_NTCCBO	0x0400	コンボボックスの読み出し時に CBN_SELENDOK イベントを発生する (※2)	
7	AJCCTL_SELACT_NTCINP	0x0800	数値入力の読み出し時に AJCIVN_INTVALUE / AJCIVN_REALVALUE イベントを発生する	
8	AJCCTL_SELACT_NTCALL	0x0300	上記３つの合成値	
9	AJCCTL_SELACT_BYLLOAD	0x8000	ロード時に指定された[act]を仮定	セーブ時のみ有効

※１： #1～3 は、いずれか１つを指定します。
※２：コンボボックスで、リスト項目が選択されていない場合は通知しません。
※３：プロファイルから読み出す際に、全てのチェックボックス，ラジオボタン、コンボボックスや数値入力コントロールに作用します。

個別のコントロールだけイベントを発生させたくない場合は、「AJCCTL_SELACT_NTC{CHK/RBT/CBO/INP}」を指定し、イベントを発生させたくないコントロールに対し AjcCtrlSetPermAtt()/ AjcDlgItemSetPermAtt()で「AJCCTL_PSEL_NONTC」を指定します。

個別のコントロールだけイベントを発生させたい場合は、AJCCTL_SELACT_NTC{CHK/RBT/CBO/INP}を未指定で、当該コントロールに対し Ajc{Ctrl/DlgItem}SetPermAtt()で「AJCCTL_PSEL_NTC」を指定します。

テキストボックス（Edit クラス）では、プロファイルを読み出した場合、無条件に BN_CHANGE イベントを発生します。
数値入力コントロール（AjcCtrlInpVal クラス）では、プロファイルを読み出した場合、無条件に AJCIVN_INTVALUE / AJCIVN_REALVALUE イベントを発生します。
その他のコントロールでは、特に通知イベントは発生しません。

永続化されるコントロールと その内容は以下のとおりです。(nnnnnn:コントロール ID,　ssssss:リスト項目のインデクス)

#	名称	クラス名	スタイル	永続化される内容（プロファイルキー名称：内容）	読出順
1	テキストボックス	Edit	－	Edt_nnnnnn ： テキスト (読み出し時に EN_CHANGE イベントが発生します)	1
2	チェックボックス	Button	BS_CHECKBOX BS_AUTOCHECKBOX BS_3STATE BS_AUTO3STATE	Chk_nnnnnn ： チェック状態 (読み出し時のイベントは AJCCTL_SELACT_NTCCHK 指定による)	
3	ラジオボタン	Button	BS_RADIOBUTTON BS_AUTORADIOBUTTON	Rbt_nnnnnn ： チェック状態 (読み出し時のイベントは AJCCTL_SELACT_NTCRBT 指定による)	
4	コンボボックス	Combobox	－	Cbo_nnnnnn_ssssss ： リスト項目（テキスト，データ） Cbo_nnnnnn_Cnt　 ： リスト項目数 Cbo_nnnnnn_Idx　 ： 選択項目のインデクス Cbo_nnnnnn_Txt　 ： 編集テキスト Cbo_nnnnnn_MaxLen ： 最大テキスト長（内部情報）	2
5	リストボックス	Listbox	－	Lst_nnnnnn_ssssss ： リスト項目（テキスト，データ，選択状態） Lst_nnnnnn_Cnt　 ： リスト項目数 Lst_nnnnnn_MaxLen ： 最大テキスト長（内部情報）	
6	拡張 リストボックス	AjcCtrlListBox	－	Lbx_nnnnnn_ssssss ： リスト項目（テキスト，データ，選択状態） Lbx_nnnnnn_Cnt　 ： リスト項目数	3
7	数値入力	AjcCtrlInpVal	－	Inp_nnnnnn ： 設定された数値 (初回読み出し時や値が変化する場合、読み出し時に（Load 時に） AJCIVN_{INT/REAL}VALUE イベントが発生します)	
8	VT-100 エミュレーション	AjcCtrlVT100	－	Vth_nnnnnn_fSetF_T ：他へのフォント適用種別 (0:兄弟ウインド ／ 1:スレッドウインド) Vth_nnnnnn_lf*XXXXXX*： フォント情報 **（※１）** Vth_nnnnnn_LSpace　 ： 行間スペース Vth_nnnnnn_*XXXXX*： フォント選択ダイアログ設定値 **（※２）**	
9	タイム チャート	AjcCtrlTmChart	－	Tch_nnnnnn_GauBarL　 ： 測定ゲージの位置（低位） Tch_nnnnnn_GauBarH　 ： 測定ゲージの位置（高位） Tch_nnnnnn_GauSetUTime: 測定の単位時間設定値 Tch_nnnnnn_fGauUseMes ： 単位時間種別（測定値／設定値）	

※１：LOGFONT 構造体の内容群です

※２：フォント選択ダイアログで使用する情報

#	キー名称	内容	備考／対応する LOGFONT 構造体メンバ
1	Vth_nnnnnn_id	ＩＤ（情報が有効であることを示す）	=0x298FC2B3（固定）
2	Vth_nnnnnn_fPixel	フォント高さの種別	０：ポイント数，１：ピクセル数
3	Vth_nnnnnn_fItalic	斜字指定	０：通常，１：斜字
4	Vth_nnnnnn_fBold	太字指定	０：通常，１：太字
5	Vth_nnnnnn_fFSelFixed	選択する対象フォント種別	０：可変ピッチ，１：固定ピッチ
6	Vth_nnnnnn_Pixels	フォント高さ（ピクセル数）	lfHeight（正数）
7	Vth_nnnnnn_Points	フォント高さ（ポイント数）	lfHeight（負数）
8	Vth_nnnnnn_FSelFixedAttr	固定ピッチ時の属性	Bit15-8 ： lfCharSet Bit 7-0 ： lfPitchAndFamily
9	Vth_nnnnnn_FSelPropoAttr	任意ピッチ時の属性	
10	Vth_nnnnnn_FSelFixedFace	固定ピッチ時のフォント名	lfFaceName
11	Vth_nnnnnn_FSelPropoFace	任意ピッチ時のフォント名	

「読出順」は、プロファイルから読み出す際の順番を意味します。同一読み出し順内では順不同です。
Ajc[Dlg]Load～()実行時に、(初回実行時等で) セーブされているプロファイルデータが無い場合は、各コントロールの内容は変化しません。

戻り値　：　TRUE　- 成功
FALSE - 失敗

使用例 ： コントロール設定値の永続化に関する使用例を示します。

1）すべてのコントロールを永続化する。(読み出し専用テキストやコンボボックス（ﾄﾞﾛｯﾌﾟﾀﾞｳﾝﾘｽﾄ）のリスト項目は含めない)

```
// プロファイルから全コントロールの設定値を読み出す
AjcLoadAllControlSettings(hDlg, TEXT("Settings"), AJCCTL_SELACT_ALL);
        ・・・
// 全コントロールの設定値をプロファイルへ書き込み
AjcSaveAllControlSettings(hDlg);
AjcDelAllCtrlPermAtt(hDlg);
```

2）すべてのコントロールを永続化する。(読み出し専用テキストやコンボボックス（ﾄﾞﾛｯﾌﾟﾀﾞｳﾝﾘｽﾄ）のリスト項目も含める)

```
// 読み出し専用テキストを含める
AjcDlgItemSetPermAtt(hDlg, IDC_TXT_RDONLY, AJCCTL_PSEL_INCRDONLY);
// コンボボックス(ドロップダウンリスト)でリスト項目群とリスト項目に関連付けられたデータを含める
AjcDlgItemSetPermAtt(hDlg, IDC_CBO_DDL   , AJCCTL_PSEL_INCLIST | AJCCTL_PSEL_INCDATA);
// プロファイルから全コントロールの設定値を読み出す
AjcLoadAllControlSettings(hDlg, TEXT("Settings"), AJCCTL_SELACT_ALL);
        ・・・
// 全コントロールの設定値をプロファイルへ書き込み
AjcSaveAllControlSettings(hDlg);
AjcDelAllCtrlPermAtt(hDlg);
```

3）すべてのコントロールを永続化する。(特定のチェックボックス，ラジオボタンだけ イベント(BN_CLICKED)を生成しない)

```
// BN_CLICKED を生成しないコントロールを設定
AjcDlgItemSetPermAtt(hDlg, IDC_CHK_XXXXX, AJCCTL_PSEL_NONTC);
AjcDlgItemSetPermAtt(hDlg, IDC_RBT_YYYYY, AJCCTL_PSEL_NONTC);
AjcDlgItemSetPermAtt(hDlg, IDC_RBT_ZZZZZ, AJCCTL_PSEL_NONTC);
        ・・・
// プロファイルから全コントロールの設定値を読み出す
AjcLoadAllControlSettings(hDlg, TEXT("Settings"), AJCCTL_SELACT_ALL | AJCCTL_SELACT_NTCALL);
        ・・・
// 全コントロールの設定値をプロファイルへ書き込み
AjcSaveAllControlSettings(hDlg);
AjcDelAllCtrlPermAtt(hDlg);
```

4）すべてのコントロールを永続化する。(特定のチェックボックス，ラジオボタンだけ イベント(BN_CLICKED)を生成する)

```
// BN_CLICKED を生成するコントロールを設定
AjcDlgItemSetPermAtt(hDlg, IDC_CHK_XXXXX, AJCCTL_PSEL_NTC);
AjcDlgItemSetPermAtt(hDlg, IDC_RBT_YYYYY, AJCCTL_PSEL_NTC);   } ラジオボタンは、選択されている項目のみ
AjcDlgItemSetPermAtt(hDlg, IDC_RBT_ZZZZZ, AJCCTL_PSEL_NTC);     イベントを発生する
        ・・・
// プロファイルから全コントロールの設定値を読み出す
AjcLoadAllControlSettings(hDlg, TEXT("Settings"), AJCCTL_SELACT_ALL);
        ・・・
// 全コントロールの設定値をプロファイルへ書き込み
AjcSaveAllControlSettings(hDlg);
AjcDelAllCtrlPermAtt(hDlg);
```

5）永続化するコントロールを指定する

```
// 永続化するコントロールを指定する
AjcDlgItemSetPermAtt (hDlg, IDC_TXT_EDIT  , AJCCTL_PSEL_INCLUDE);
AjcDlgItemSetPermAtt (hDlg, IDC_CBO_DD    , AJCCTL_PSEL_INCLUDE);
AjcDlgItemSetPermAtt (hDlg, IDC_LST_SINGLE, AJCCTL_PSEL_INCLUDE);
        ・・・
// プロファイルからAjcDlgItemSetPermAtt(～, AJCCTL_PSEL_INCLUDE)で指定したコントロール群の設定値を読み出す
AjcLoadAllControlSettings(hDlg, TEXT("Settings"), AJCCTL_SELACT_CHKINCLUDE);
        ・・・
// AjcDlgItemSetPermAtt()で指定したコントロールの設定値をプロファイルへ書き込み
AjcSaveAllControlSettings(hDlg);
AjcDelAllCtrlPermAtt(hDlg);
```

6）永続化しないコントロールを指定する

```
// 永続化しないコントロールを指定する
AjcDlgItemSetPermAtt (hDlg, IDC_TXT_EDIT  , AJCCTL_PSEL_EXCLUDE);
AjcDlgItemSetPermAtt (hDlg, IDC_CBO_DD    , AJCCTL_PSEL_EXCLUDE);
AjcDlgItemSetPermAtt (hDlg, IDC_LST_SINGLE, AJCCTL_PSEL_EXCLUDE);
        ・・・
// プロファイルからAjcDlgItemSetPermAtt(～, AJCCTL_PSEL_EXCLUDE)で指定したコントロール以外の設定値を読み出す
AjcLoadAllControlSettings(hDlg, TEXT("Settings"), AJCCTL_SELACT_CHKEXCLUDE);
        ・・・
// AjcDlgItemSetPermAtt()で指定したコントロール以外の設定値をプロファイルへ書き込み
AjcSaveAllControlSettings(hDlg);
AjcDelAllCtrlPermAtt(hDlg);
```

17.4.15. 各コントロールの永続化属性設定(Ajc{DlgItem/Ctrl}SetPermAtt)

形　式 ： BOOL AjcCtrlSetPermAtt (HWND hctrl, UI att); --- コントロールのウインドハンドル指定
BOOL AjcDlgItemSetPermAtt (HWND hDlg, UI id, UI att); --- ダイアログのコントロールＩＤ指定

引　数 ： hctrl - コントロールのウインドのハンドル
hDlg - ダイアログボックスのウインドハンドル
id - ダイアログ項目の番号
att - コントロールの永続化動作属性 (AJCCTL_PSEL_*XXXXX*)

説　明 ： Ajc[Dlg]Load{All/Grp}ControlSettings()で、全てのコントロールの永続化情報を読み出す際の各コントロール個別の動作属性を設定します。
att は以下のシンボルの合成値を指定します。

#	att	内容	備考
1	AJCCTL_PSEL_DELETE	属性解除 (※１)	単独で指定
2	AJCCTL_PSEL_INCLUDE	このコントロールを永続化対象に含める	いずれか片方を指定する
3	AJCCTL_PSEL_EXCLUDE	このコントロールを永続化対象から除外する	
4	AJCCTL_PSEL_INCRDONLY	読み出し専用テキストを永続化する	テキストボックスでのみ有効
5	AJCCTL_PSEL_INCDATA	リスト項目群に関連付けられたデータも永続化する	コンボボックス／リストボックスでのみ有効
6	AJCCTL_PSEL_INCLIST	リスト項目群を永続化する (※２)	コンボボックスでのみ有効
7	AJCCTL_PSEL_EXCLIST	リスト項目群を永続化しない (※３)	
8	AJCCTL_PSEL_EXCSEL	項目の選択情報を永続化しない	コンボボックス／リストボックスでのみ有効
9	AJCCTL_PSEL_EXCEDIT	エディットテキストの内容を永続化しない	コンボボックスでのみ有効
10	AJCCTL_PSEL_NTC	チェックボックス / ラジオボタン/コンボボックス読出し時にイベントを通知する (ラジオボタンの場合はチェック時のみ通知) (※４)	チェックボックス / ラジオボタン／コンボボックスでのみ有効 (AJCCTL_PSEL_NONTC 優先)
11	AJCCTL_PSEL_NONTC	チェックボックス / ラジオボタン/コンボボックス読出し時にイベントを通知しない (※５)	

※１：#2～でコントロールに設定した永続化属性を解除します。

※２：CBS_DROPDOWN/CBS_SIMPLE スタイルのコンボボックスの場合、「AJCCTL_PSEL_INCLIST」を指定しなくても、デフォルトでリスト項目群を永続化します。永続化しない場合は「AJCCTL_PSEL_EXCLIST」を指定します。

※３：CBS_DROPDOWNLIST スタイルのコンボボックスの場合、「AJCCTL_PSEL_EXCLIST」を指定しなくても、デフォルトでリスト項目群を永続化しません。永続化する場合は「AJCCTL_PSEL_INCLIST」を指定します。

※４：チェックボックスや選択されているラジオボタンの場合、BN_CLICKED イベントを生成します。
コンボボックスの場合、選択されている項目があれば、CBN_SELENDOK イベントを生成します。

※５：チェックボックス, 選択されているラジオボタンで BN_CLICKED イベントを、コンボボックスでで CBN_SELENDOK イベントを生成しません。

戻り値 ： TRUE - 成功
FALSE - 失敗

17.4.16. グループボックス内全コントロールの永続化属性設定(Ajc{DlgItem/Ctrl}SetPermGrp)

形　式 ： BOOL AjcCtrlSetPermAttGrp (HWND hctrl, UI att); --- グループボックスのウインドハンドル指定
BOOL AjcDlgItemSetPermAttGrp (HWND hDlg, UI idGrp, UI att); --- グループボックスのコントロールＩＤ指定

引　数 ： hctrl - グループボックスのコントロールのウインド・ハンドル
hDlg - ダイアログボックスのウインドハンドル
idGrp - グループボックスのダイアログ項目番号
att - コントロールの永続化動作属性 (AJCCTL_PSEL_*XXXXX*)

説　明 ： Ajc[Dlg]Load{All/Grp}ControlSettings()で、全てのコントロールの永続化情報を読み出す際のグループボックス内の全コントロールについて動作属性を設定します。
att はの内容は、AjcCtrlSetPermAtt()/AjcDlgItemSetPermAtt()と同じです。

戻り値 ： TRUE - 成功
FALSE - 失敗

17.4.17. 全コントロールの永続化属性解除(AjcDelAllCtrlPermAtt)

形　式 ： BOOL AjcDelAllCtrlPermAtt (HWND hwnd);

引　数 ： hwnd　　- コントロールの親ウインドのハンドル

説　明 ： hwnd で指定された全ての子ウインド（コントロール）について、AjcSet{DlgItem/Ctrl}PermAtt()で設定した永続化指定を解除します。　また、AjcSet{DlgItem/Ctrl}PermAtt()で確保されたリソースを解放します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

備　考 ： このＡＰＩを実行しなくても、ウインドのクローズ後に AjcSet{DlgItem/Ctrl}PermAtt()で確保されたリソースは自動的に解放されます。

17.4.18. テキストボックスの永続化(Ajc{DlgItem/Ctrl}{Load/Save}TextBox[Ex])

形　式 ：
```
BOOL  AjcDlgItemLoadTextBox  (HWND hDlg, UI id, C_UTP pSect); -------------------- プロファイルから読み出し
BOOL  AjcDlgItemLoadTextBoxEx(HWND hDlg, UI id, C_UTP pSect, BOOL fExcRdOnly); --     〃
BOOL  AjcDlgItemSaveTextBox  (HWND hDlg, UI id, C_UTP pSect); -------------------- プロファイルへ記録
BOOL  AjcDlgItemSaveTextBoxEx(HWND hDlg, UI id, C_UTP pSect, BOOL fExcRdOnly); --     〃

BOOL  AjcCtrlLoadTextBox     (HWND hwnd       , C_UTP pSect); --------------------- プロファイルから読み出し
BOOL  AjcCtrlLoadTextBoxEx   (HWND hwnd       , C_UTP pSect, BOOL fExcRdOnly); --     〃
BOOL  AjcCtrlSaveTextBox     (HWND hwnd       , C_UTP pSect); --------------------- プロファイルへ記録
BOOL  AjcCtrlSaveTextBoxEx   (HWND hwnd       , C_UTP pSect, BOOL fExcRdOnly); --     〃
```

引　数 ：
```
hwnd        - テキストボックス（Edit クラス）コントロールのウインドのハンドル
hDlg        - ダイアログボックスハンドル
id          - ダイアログボックス内のコントロールＩＤ
pSect       - セクション名へのポインタ
fExcRdOnly  - 読み出し専用テキストの永続化指定（TRUE:永続化しない（デフォルト）, FALSE:永続化する）
```

説　明 ： テキストボックス（Edit クラス）のテキストをプロファイルから読み出し、あるいは、プロファイルへ記録します。
fExcRdOnly は、読み出し専用テキスト（ES_READONLY スタイル）を永続化するか否かを指定します。
fExcRdOnly=TRUE を指定した場合は、読み出し専用テキストは永続化しません。
fExcRdOnly=FALSE を指定した場合は、読み出し専用テキストも永続化します。
プロファイルキーの名称は「Edt_nnnnnn」（nnnnnn は自コントロールＩＤ）となります。

戻り値 ： TRUE　- 成功
FALSE - 失敗

17.4.19. チェックボックス／ラジオボタンの永続化(Ajc{DlgItem/Ctrl}{Load/Save}ChkBox)

形　式 ：
```
BOOL  AjcDlgItemLoadChkBox(HWND hDlg, UI id, C_UTP pSect, BOOL fNtc); -- プロファイルから読み出し
BOOL  AjcDlgItemSaveChkBox(HWND hDlg, UI id, C_UTP pSect); ------------- プロファイルへ記録

BOOL  AjcCtrlLoadChkBox   (HWND hwnd       , C_UTP pSect, BOOL fNtc); -- プロファイルから読み出し
BOOL  AjcCtrlSaveChkBox   (HWND hwnd       , C_UTP pSect); ------------- プロファイルへ記録
```

引　数 ：
```
hwnd        - チェックボックス／ラジオボタン（Button クラス）コントロールのウインドのハンドル
hDlg        - ダイアログボックスハンドル
id          - ダイアログボックス内のコントロールＩＤ
pSect       - セクション名へのポインタ
fNtc        - イベント通知フラグ（FALSE：イベントを発生しない, TRUE:イベントを発生する）
```

説　明 ： チェックボックス／ラジオボタン(Button クラス)のチェック状態をプロファイルから読み出し、あるいは、プロファイルへ記録します。
プロファイルキーの名称は「Chk_nnnnnn」/「Rbt_nnnnnn」（nnnnnn は自コントロールＩＤ）となります。
fNtc=TRUE を指定した場合、（ラジオボタンの場合はチェックされている場合のみ）BN_CLICKED イベントを発生します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

17.4.20. コンボボックスの永続化(Ajc{DlgItem/Ctrl}{Load/Save}ComboBox)

形　式 ： BOOL AjcDlgItemLoadComboBox(HWND hDlg, UI id, C_UTP pSect, UI sel, BOOL fNtc); - プロファイルから読み出し
BOOL AjcDlgItemSaveComboBox(HWND hDlg, UI id, C_BCP pSect, UI sel); ---------- プロファイルへ書き込み

BOOL AjcCtrlLoadComboBox (HWND hwnd , C_UTP pSect, UI sel, BOOL fNtc); - プロファイルから読み出し
BOOL AjcCtrlSaveComboBox (HWND hwnd , C_BCP pSect, UI sel); ---------- プロファイルへ書き込み

引　数 ： hwnd - チェックボックス／ラジオボタン（Button クラス）コントロールのウインドのハンドル
hDlg - ダイアログボックスハンドル
id - ダイアログボックス内のコントロールＩＤ
pSect - セクション名へのポインタ
sel - 永続化項目の指定（AJCCTL_PSEL_*XXXXX* ／ 0）
fNtc - イベント通知フラグ（FALSE:イベントを発生しない, TRUE:イベントを発生する）

説　明 ： コンボボックス（Combobox クラス）の内容をプロファイルから読み出し、あるいは、プロファイルへ記録します。
sel は、永続化する項目を以下のシンボルの合成値で指定します。（何も指定しない場合は、０）

#	att	内容	備考
1	AJCCTL_PSEL_INCDATA	リスト項目群に関連付けられたデータも永続化する	読み出し時のみ有効 書き込み時は、常にプロファイルへ出力する
2	AJCCTL_PSEL_INCLIST	リスト項目群を永続化する (※1)	CBS_DROPDOWNLIST 時のみ有効
3	AJCCTL_PSEL_EXCSEL	項目の選択情報を永続化しない	読み出し時に指定した場合は、選択状態の設定を行いません。 書き込み時に指定した場合は、非選択状態として記録する。
4	AJCCTL_PSEL_EXCLIST	リスト項目群を永続化しない (※1)	CBS_DROPDOWN/CBS_SIMPLE 時のみ有効
5	AJCCTL_PSEL_EXCEDIT	エディットテキストの内容を永続化しない	

※１：CBS_DROPDOWN/CBS_SIMPLE スタイルでは、AJCCTL_PSEL_INCLIST を指定しなくてもリスト項目群を永続化します。
リスト項目群を永続化しない場合は、AJCCTL_PSEL_EXCLIST を指定します。

CBS_DROPDOWNLIST スタイルでは、AJCCTL_PSEL_EXCLIST を指定しなくてもリスト項目群を永続化しません。
リスト項目群を永続化する場合は、AJCCTL_PSEL_INCLIST を指定します。

永続化時の、プロファイルキーの名称は以下のとおりです。(nnnnnn は自コントロールＩＤ, ssssss はリスト項目のインデクス)
Cbo_nnnnnn_Cnt ： リスト項目数　　Cbo_nnnnnn_MaxLen ： 最大テキスト長（文字列終端は含まない）
Cbo_nnnnnn_Idx ：選択項目のインデクス　　Cbo_nnnnnn_ssssss ： リスト項目のテキストとデータ値
Cbo_nnnnnn_Txt ： エディットテキスト

fNtc=TRUE を指定した場合（選択されている項目があれば）コンボボックスの読み出し後に CBN_SELENDOK イベントを発生します。

戻り値 ： TRUE - 成功
FALSE - 失敗

17.4.21. リストボックスの永続化(Ajc{DlgItem/Ctrl}{Load/Save}ListBox)

形　式 ： BOOL　AjcDlgItemLoadListBox(HWND hDlg, UI id, C_UTP pSect, UI sel); -------- プロファイルから読み出し
BOOL　AjcDlgItemSaveListBox(HWND hDlg, UI id, C_BCP pSect, UI sel); -------- プロファイルへ記録

BOOL　AjcCtrlLoadListBox　(HWND hwnd　　, C_UTP pSect, UI sel); -------- プロファイルから読み出し
BOOL　AjcCtrlSaveListBox　(HWND hwnd　　, C_BCP pSect, UI sel); -------- プロファイルへ記録

引　数 ： hwnd　　- チェックボックス／ラジオボタン（Button クラス）コントロールのウインドのハンドル
hDlg　　- ダイアログボックスハンドル
id　　- ダイアログボックス内のコントロールＩＤ
pSect　　- セクション名へのポインタ
sel　　- 永続化項目の指定（AJCCTL_PSEL_INCDATA / 0）

説　明 ： リストボックス（Listbox クラス）の内容をプロファイルから読み出し、あるいは、プロファイルへ記録します。
sel は、永続化する項目を以下のシンボルで指定します。（何も指定しない場合は、０）

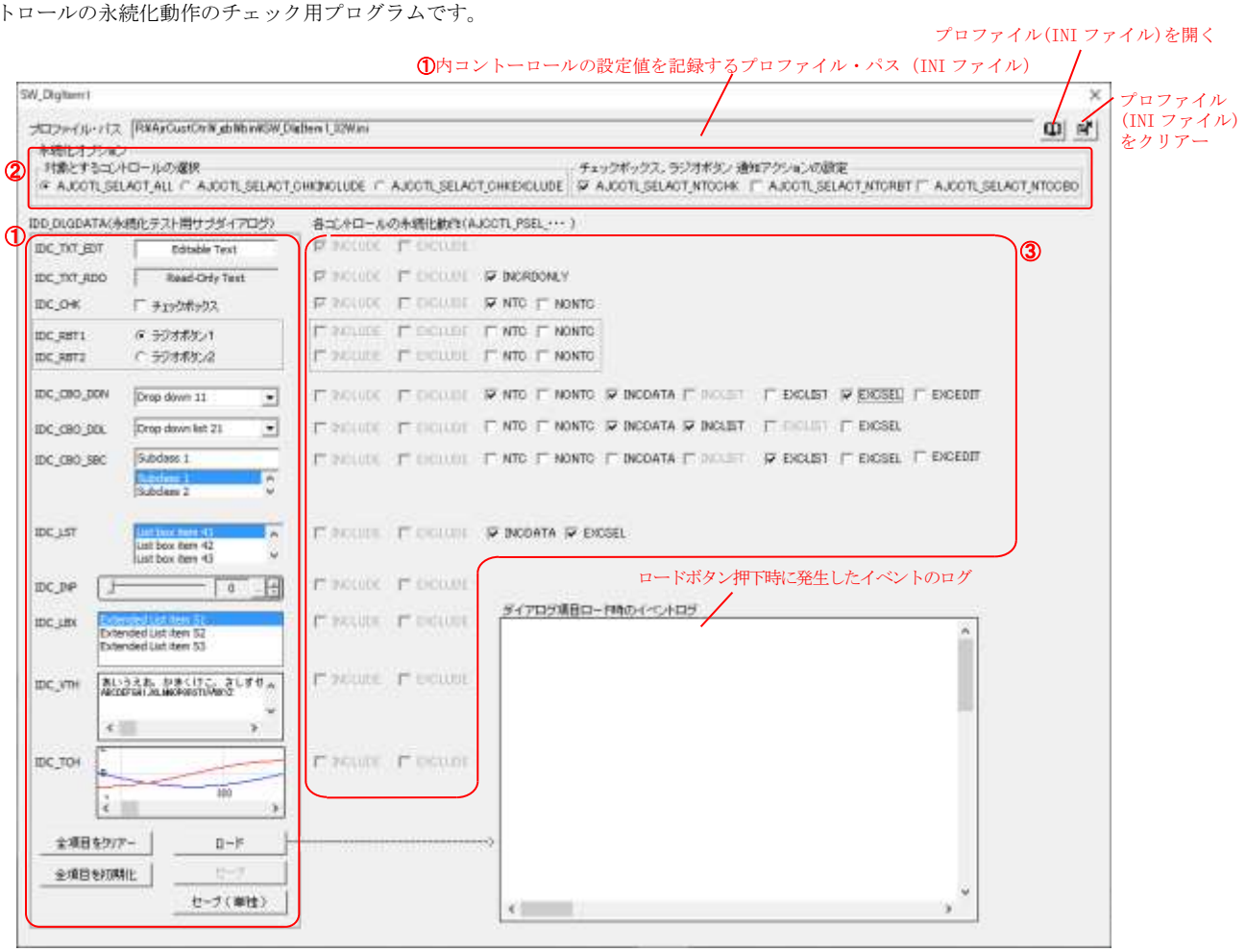
#	att	内容	備考
1	AJCCTL_PSEL_INCDATA	リスト項目群に関連付けられたデータも永続化する	読み出し時のみ有効 書き込み時は、常にプロファイルへ出力する
2	AJCCTL_PSEL_EXCSEL	項目の選択情報を永続化しない	読み出し時に指定した場合は、選択状態の設定を行いません。 書き込み時に指定した場合は、非選択状態として記録する。

プロファイルキーの名称は以下のとおりです。（nnnnnn は自コントロールＩＤ，ssssss はリスト項目のインデクス）
Lst_nnnnnn_Cnt　　：リスト項目数
Lst_nnnnnn_MaxLen：最大テキスト長（文字列終端は含まない）
Lst_nnnnnn_ssssss：リスト項目のテキスト，選択状態，データ値

戻り値 ： TRUE　- 成功
FALSE - 失敗

17.4.22. SW_DlgItem1（ダイアログボックス内の全コントロールの永続化）

このサンプルプログラムは、Ajc{Save/Load}AllControlSettings()と AjcDlgItemSetPermAtt()による、ダイアログボックス内全コントロールの永続化動作のチェック用プログラムです。



①：全コントロールの永続化を行うダイアログボックス（子ウインド）
②：Ajc{Save/Load}AllControlSettings()で指定する永続化動作（act 引数）
③：AjcDlgItemSetPermAtt()で設定する各コントロールの永続化属性（att 引数）

①のサブダイアログ内の各コントロールにカーソルを置くと、当該コントロールをプロファイルに記録する際のプロファイルキーをチップ表示します。
また、以下のコントロールでは、チップテキストに、以下のコントロールの設定内容も含めて表示します。

- IDC_DDN, IDC_DDL, IDC_SBC, IDC_LST, IDC_LBX --- 各リスト項目に関連付けられたデータ値
- IDC_VTH ---------- プロファイルに記録される情報（フォントや行間スペース等）
- IDC_TCH ---------- プロファイルに記録される情報（計測用ゲージ情報）

「ロード」ボタンを押すと、各コントロールに対して AjcDlgItemSetPermAtt()により　各コントロールへ③で指定された永続化属性を設定後、AjcLoadAllControlSettings()を実行し、プロファイルから各コントロールの設定値を読み出します。
AjcLoadAllControlSettings()の act 引数には、②で設定された内容を指定します。

「セーブ」ボタンを押すと、AjcSaveAllControlSettings()を実行し、各コントロールの内容をプロファイルに記録します。
この時、②と③で設定された内容された内容は、「ロード」ボタン押下時の内容となります。
「セーブ」ボタンは、一度「ロード」ボタンを押すまで禁止状態となっています。

「セーブ（単独）」ボタンを押すと、各コントロールに対して AjcDlgItemSetPermAtt()により　③で指定された永続化属性を設定後、AjcSavedAllControlSettingsEx()を実行します。
つまり、②と③で設定された内容を反映した上で、各コントロールの設定値をプロファイルに記録します。

```
//
//  SW_DlgItem1.c
//
#include    <AjrCstXX.h>
#include    <Shlwapi.h>
#include    <stdio.h>
#include    <tchar.h>
#include    "resource.h"

//------------------------------------------------------------------------------------------------------------//
//  ワーク                                                                                                    //
//------------------------------------------------------------------------------------------------------------//
static  HINSTANCE       hInst;                      //  ＤＬＬインスタンスハンドル
static  HWND            hDlgMain;                   //  ダイアログボックスハンドル
static  HWND            hDlgData;                   //  サブダイアログハンドル
static  HWND            hVthLog;                    //  ログウインド
static  HICON           hIcoOpen;
static  HICON           hIcoDel;
static  BOOL            fLoad    = FALSE;           //  ロード処理中を示すフラグ
static  UT              MySect[] = _T("Settings");  //  自プログラム設定値退避用セクション

//------------------------------------------------------------------------------------------------------------//
//  キープリフィックス                                                                                        //
//------------------------------------------------------------------------------------------------------------//
static  C_UTP   szPrefix[] = {TEXT("Edt"), TEXT("Chk"), TEXT("Rbt"), TEXT("Cbo"), TEXT("Lst"), TEXT("Inp"), TEXT("Lbx"), TEXT("Vth"), TEXT("Tch")};
enum                        {  PFX_EDT  ,   PFX_CHK  ,   PFX_RBT  ,   PFX_CBO  ,   PFX_LST  ,   PFX_INP  ,   PFX_LBX  ,   PFX_VTH  ,   PFX_TCH  };

//------------------------------------------------------------------------------------------------------------//
//  内部サブ関数                                                                                              //
//------------------------------------------------------------------------------------------------------------//
AJC_DLGPROC_DEF(Main);
AJC_DLGPROC_DEF(Data);
static  UI              SetPermOpt(VO);
static  C_UTP CALLBACK  cbGetTipText(HWND hCtrl, UTP pBuf, UI lBuf, UX cbp);

//============================================================================================================//
//                                                                                                            //
//  W i n M a i n                                                                                             //
//                                                                                                            //
//============================================================================================================//
int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
{
    MSG     msg;

    hInst = hInstance;

    //----- メイン・ダイアログオープン -----------------//
    hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_DLGMAIN), NULL, AJC_DLGPROC_NAME(Main));
    ShowWindow(hDlgMain, SW_SHOW);

    //----- メッセージループ ---------------------------//
    while (GetMessage(&msg, NULL, 0, 0)) {
        do {
            if (IsDialogMessage(hDlgMain, &msg)) break;
            TranslateMessage(&msg);
            DispatchMessage (&msg);
        } while (0);
    }

    return (int)msg.wParam ;
}
//============================================================================================================//
//                                                                                                            //
//  ダイアログ・プロシージャ                                                                                  //
//                                                                                                            //
//============================================================================================================//
#define     TTP(ID) AjcTipTextAdd       (GetDlgItem(hDlg, IDC_CHK_##ID), TEXT(""));         ¥
                    AjcTipTextSetCallBack(GetDlgItem(hDlg, IDC_CHK_##ID), 0, NULL, cbChkTool);

C_UTP CALLBACK cbChkTool(HWND hCtrl, UTP pBuf, UI lBuf, UX cbp)
{
    UT      txt[64];

    GetWindowText(hCtrl, txt, AJCTSIZE(txt));
    if      (MAjcStrCmp(txt, TEXT("INCLUDE" )) == 0) MAjcStrCpy(pBuf, lBuf, TEXT("このコントロールを永続化対象に含める"));
    else if (MAjcStrCmp(txt, TEXT("EXCLUDE" )) == 0) MAjcStrCpy(pBuf, lBuf, TEXT("このコントロールを永続化対象から除外する"));
    else if (MAjcStrCmp(txt, TEXT("INCRDONLY")) == 0) MAjcStrCpy(pBuf, lBuf, TEXT("読み出し専用テキストを永続化する"));
    else if (MAjcStrCmp(txt, TEXT("INCDATA" )) == 0) MAjcStrCpy(pBuf, lBuf, TEXT("リスト項目群に関連付けられたデータも永続化する"));
    else if (MAjcStrCmp(txt, TEXT("INCLIST" )) == 0) MAjcStrCpy(pBuf, lBuf, TEXT("リスト項目群を永続化する"));
    else if (MAjcStrCmp(txt, TEXT("EXCSEL"  )) == 0) MAjcStrCpy(pBuf, lBuf, TEXT("項目の選択情報を永続化しない"));
    else if (MAjcStrCmp(txt, TEXT("EXCLIST" )) == 0) MAjcStrCpy(pBuf, lBuf, TEXT("リスト項目群を永続化しない"));
    else if (MAjcStrCmp(txt, TEXT("EXCEDIT" )) == 0) MAjcStrCpy(pBuf, lBuf, TEXT("エディットテキストの内容を永続化しない"));
    else if (MAjcStrCmp(txt, TEXT("NTC"     )) == 0) MAjcStrCpy(pBuf, lBuf, TEXT("チェックボックス/ラジオボタン読出し時にイベント(BN_CLICKED)を通知する"));
    else if (MAjcStrCmp(txt, TEXT("NONTC"   )) == 0) MAjcStrCpy(pBuf, lBuf, TEXT("チェックボックス/ラジオボタン読出し時にイベント(BN_CLICKED)を通知しない"));
    return pBuf;
}
//----- ダイアログ初期化 ---------------------------------------------------------------------------------------//
AJC_DLGPROC(Main, WM_INITDIALOG     )
{
```

```
90 :     RECT    r;
91 :     UT      txt[AJCMAX_REGPATH];
92 :
93 :     hVthLog = GetDlgItem(hDlg, IDC_VTH_LOG);
94 :
95 :     //  アイコン生成
96 :     hIcoOpen  = (HICON)LoadImage(hInst, MAKEINTRESOURCE(IDI_ICO_OPEN), IMAGE_ICON, 16, 16, LR_DEFAULTCOLOR);
97 :     hIcoDel   = (HICON)LoadImage(hInst, MAKEINTRESOURCE(IDI_ICO_DEL ), IMAGE_ICON, 16, 16, LR_DEFAULTCOLOR);
98 :     //  ボタンへアイコン表示
99 :     SendDlgItemMessage(hDlg, IDC_CMD_OPEN, BM_SETIMAGE, IMAGE_ICON, (LPARAM)hIcoOpen);
100 :    SendDlgItemMessage(hDlg, IDC_CMD_DEL , BM_SETIMAGE, IMAGE_ICON, (LPARAM)hIcoDel );
101 :    //  ＩＮＩファイルパス表示
102 :    AjcGetIniFilePath(txt, AJCTSIZE(txt));
103 :    AjcSetDlgItemStr (hDlg, IDC_TXT_INIPATH, txt);
104 :    //  ダイアログ項目のデフォルト設定
105 :    AjcSetDlgItemChk(hDlg, IDC_RBT_SELACT_ALL, TRUE);
106 :    //  ダイアログ項目のロード
107 :    AjcLoadAllControlSettings(hDlg, TEXT("MainDialogItems"), AJCCTL_SELACT_ALL | AJCCTL_SELACT_NTCALL);
108 :    //  サブダイアログ生成
109 :    CreateDialog(hInst, MAKEINTRESOURCE(IDD_DLGDATA), hDlg, AJC_DLGPROC_NAME(Data));
110 :    GetWindowRect  (GetDlgItem(hDlg, IDC_LBL_SUBDLG), &r);
111 :    MapWindowPoints(NULL, hDlg, (LPPOINT)&r, 2);
112 :    SetWindowPos(hDlgData, NULL, r.left, r.top, 0, 0, SWP_NOSIZE);
113 :    ShowWindow(hDlgData, SW_SHOW);
114 :    //  ツールチップ設定
115 :    AjcTipTextAdd(GetDlgItem(hDlg, IDC_CMD_OPEN           ), TEXT("INI ファイルを開く"));
116 :    AjcTipTextAdd(GetDlgItem(hDlg, IDC_CMD_DEL            ), TEXT("INI ファイルをクリアーする"));
117 :    AjcTipTextAdd(GetDlgItem(hDlg, IDC_RBT_SELACT_ALL     ), TEXT("全てのコントロールを対象とする"));
118 :    AjcTipTextAdd(GetDlgItem(hDlg, IDC_RBT_SELACT_CHKINCLUDE), TEXT("永続化対象(INCLUDE)としたコントロールのみを対象とする"));
119 :    AjcTipTextAdd(GetDlgItem(hDlg, IDC_RBT_SELACT_CHKEXCLUDE), TEXT("永続化除外(EXCLUDE)としたコントロールは除外する"));
120 :    AjcTipTextAdd(GetDlgItem(hDlg, IDC_CHK_SELACT_NTCCHK   ), TEXT("チェックボックスの読み出し時にBN_CLICKED イベントを発生する"));
121 :    AjcTipTextAdd(GetDlgItem(hDlg, IDC_CHK_SELACT_NTCRBT   ), TEXT("ラジオボタンの読み出し時に、チェックされているラジオボタンにBN_CLICKED イベントを発生する"));
122 :    AjcTipTextAdd(GetDlgItem(hDlg, IDC_CHK_SELACT_NTCCBO   ), TEXT("コンボボックスの読み出し時に、CBN_SELENDOK イベントを発生する"));
123 :
124 :    TTP(EDT_INCLUDE);   TTP(EDT_EXCLUDE);
125 :    TTP(RDO_INCLUDE);   TTP(RDO_EXCLUDE);   TTP(RDO_INCRDONLY);
126 :    TTP(CHK_INCLUDE);   TTP(CHK_EXCLUDE);   TTP(CHK_NTC);       TTP(CHK_NONTC);
127 :    TTP(RBT1_INCLUDE);  TTP(RBT1_EXCLUDE);  TTP(RBT1_NTC);      TTP(RBT1_NONTC);
128 :    TTP(RBT2_INCLUDE);  TTP(RBT2_EXCLUDE);  TTP(RBT2_NTC);      TTP(RBT2_NONTC);
129 :    TTP(DDN_INCLUDE);   TTP(DDN_EXCLUDE);   TTP(DDN_INCDATA);   TTP(DDN_INCLIST);   TTP(DDN_EXCLIST);   TTP(DDN_EXCSEL);    TTP(DDN_EXCEDIT);
130 :    TTP(DDL_INCLUDE);   TTP(DDL_EXCLUDE);   TTP(DDL_INCDATA);   TTP(DDL_INCLIST);   TTP(DDL_EXCLIST);   TTP(DDL_EXCSEL);
131 :    TTP(SBC_INCLUDE);   TTP(SBC_EXCLUDE);   TTP(SBC_INCDATA);   TTP(SBC_INCLIST);   TTP(SBC_EXCLIST);   TTP(SBC_EXCSEL);    TTP(SBC_EXCEDIT);
132 :    TTP(LST_INCLUDE);   TTP(LST_EXCLUDE);   TTP(LST_INCDATA);   TTP(LST_EXCSEL);
133 :    TTP(INP_INCLUDE);   TTP(INP_EXCLUDE);
134 :    TTP(LBX_INCLUDE);   TTP(LBX_EXCLUDE);
135 :    TTP(VTH_INCLUDE);   TTP(VTH_EXCLUDE);
136 :    TTP(TCH_INCLUDE);   TTP(TCH_EXCLUDE);
137 :
138 :    return TRUE;
139 : }
140 : //----- ウインド破棄 ----------------------------------------------------------------------------------------//
141 : AJC_DLGPROC(Main, WM_DESTROY        )
142 : {
143 :
144 :    //  ダイアログ項目のセーブ
145 :    AjcSaveAllControlSettings(hDlg);
146 :    //  アイコン破棄
147 :    if (hIcoOpen  != NULL) {DeleteObject(hIcoOpen); hIcoOpen = NULL;}
148 :    if (hIcoDel   != NULL) {DeleteObject(hIcoDel ); hIcoDel  = NULL;}
149 :
150 :    PostQuitMessage(0);
151 :
152 :    return TRUE;
153 : }
154 : //----- INI ファイルを開くボタン ----------------------------------------------------------------------------//
155 : AJC_DLGPROC(Main, IDC_CMD_OPEN      )
156 : {
157 :    UT      path[MAX_PATH];
158 :    AjcGetDlgItemStr(hDlg, IDC_TXT_INIPATH, path, MAX_PATH);
159 :    ShellExecute(NULL, TEXT("open"), path, NULL, NULL, SW_SHOWNORMAL);
160 :    return TRUE;
161 : }
162 : //----- INI ファイルをクリアーボタン ------------------------------------------------------------------------//
163 : AJC_DLGPROC(Main, IDC_CMD_DEL       )
164 : {
165 :    HANDLE  hFile;
166 :    UT      path[MAX_PATH];
167 :    AjcGetIniFilePath(path, MAX_PATH);
168 :    DeleteFile(path);
169 :    if ((hFile = CreateFile(path, GENERIC_WRITE, FILE_SHARE_WRITE, NULL, CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL)) != INVALID_HANDLE_VALUE) {
170 :        CloseHandle(hFile);
171 :    }
172 :    return TRUE;
173 : }
174 : //----- ラジオボタン（全てのコントロールを対象とする） ------------------------------------------------------//
175 : AJC_DLGPROC(Main, IDC_RBT_SELACT_ALL        )
176 : {
177 :    AjcEnableDlgGroup(hDlg, IDC_GRP_EXCLUDE, FALSE, FALSE);
178 :    AjcEnableDlgGroup(hDlg, IDC_GRP_INCLUDE, FALSE, FALSE);
179 :    return TRUE;
```

```
180 : }
181 : //----- ラジオボタン（永続化対象としたコントロールのみを対象とする） ----------------------------------------//
182 : AJC_DLGPROC(Main, IDC_RBT_SELACT_CHKINCLUDE )
183 : {
184 :     AjcEnableDlgGroup(hDlg, IDC_GRP_EXCLUDE, FALSE, FALSE);
185 :     AjcEnableDlgGroup(hDlg, IDC_GRP_INCLUDE, TRUE , TRUE );
186 :     return TRUE;
187 : }
188 : //----- ラジオボタン（永続化除外としたコントロールは除外する） ----------------------------------------------//
189 : AJC_DLGPROC(Main, IDC_RBT_SELACT_CHKEXCLUDE )
190 : {
191 :     AjcEnableDlgGroup(hDlg, IDC_GRP_EXCLUDE, TRUE , TRUE );
192 :     AjcEnableDlgGroup(hDlg, IDC_GRP_INCLUDE, FALSE, FALSE);
193 :     return TRUE;
194 : }
195 : //----- ウインドクローズ ------------------------------------------------------------------------------------//
196 : AJC_DLGPROC(Main, IDCANCEL          )
197 : {
198 :     DestroyWindow(hDlg);
199 :     return TRUE;
200 : }
201 : //-----------------------------------------------------------------------------------------------------------//
202 : AJC_DLGMAP_DEF(Main)
203 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG              )
204 :     AJC_DLGMAP_MSG(Main, WM_DESTROY                 )
205 :
206 :     AJC_DLGMAP_CMD(Main, IDC_CMD_OPEN               )
207 :     AJC_DLGMAP_CMD(Main, IDC_CMD_DEL                )
208 :     AJC_DLGMAP_CMD(Main, IDC_RBT_SELACT_ALL         )
209 :     AJC_DLGMAP_CMD(Main, IDC_RBT_SELACT_CHKINCLUDE  )
210 :     AJC_DLGMAP_CMD(Main, IDC_RBT_SELACT_CHKEXCLUDE  )
211 :     AJC_DLGMAP_CMD(Main, IDCANCEL                   )
212 : AJC_DLGMAP_END
213 : //===========================================================================================================//
214 : //                                                                                                           //
215 : //  サブダイアログ・プロシージャ                                                                             //
216 : //                                                                                                           //
217 : //===========================================================================================================//
218 : #define TTP1(ID, PFX)                                                                                        ¥
219 :  AjcTipTextAddEx      (GetDlgItem(hDlg, ID), TEXT(""), 300, 5000, AjcGetStockFont(AJCFID_FIX12), -1, -1, -1);   ¥
220 :  AjcTipTextSetCallBack(GetDlgItem(hDlg, ID), MAKELONG(ID, PFX), NULL, cbGetTipText)
221 :
222 : #define TTP2(ID, PFX)                                                                                                                ¥
223 :  AjcTipTextAddEx      (AjcSbcGetEditCtrlInComboBox(GetDlgItem(hDlg, ID)), TEXT(""), 300, 5000, AjcGetStockFont(AJCFID_FIX12), -1, -1, -1);¥
224 :  AjcTipTextSetCallBack(AjcSbcGetEditCtrlInComboBox(GetDlgItem(hDlg, ID)), MAKELONG(ID, PFX), NULL, cbGetTipText)
225 :
226 : //----- ダイアログ初期化 ------------------------------------------------------------------------------------//
227 : AJC_DLGPROC(Data, WM_INITDIALOG     )
228 : {
229 :     hDlgData = hDlg;
230 :
231 :     //  IDC_CBO_SBC サブクラス化
232 :     AjcSbcComboBox     (GetDlgItem(hDlg, IDC_CBO_SBC), 10, 64); //  サブクラス化
233 :     AjcSbcSetCompExact (GetDlgItem(hDlg, IDC_CBO_SBC), TRUE);   //  英大小文字を区別
234 :     AjcSbcTipCtrl      (GetDlgItem(hDlg, IDC_CBO_SBC), TRUE);   //  選択中項目をチップ表示する
235 :
236 :     //  全項目初期化
237 :     SendMessage(hDlg, WM_COMMAND, MAKELONG(IDC_CMD_INIT, BN_CLICKED), (LPARAM)GetDlgItem(hDlg, IDC_CMD_CLEAR));
238 :
239 :     //  ツールチップテキスト設定
240 :     TTP1(IDC_TXT_EDT    , PFX_EDT);
241 :     TTP1(IDC_TXT_RDO    , PFX_EDT);
242 :     TTP1(IDC_CHK        , PFX_CHK);
243 :     TTP1(IDC_RBT1       , PFX_RBT);
244 :     TTP1(IDC_RBT2       , PFX_RBT);
245 :     TTP1(IDC_CBO_DDN    , PFX_CBO);
246 :     TTP2(IDC_CBO_DDN    , PFX_CBO);
247 :     TTP1(IDC_CBO_DDL    , PFX_CBO);
248 :     TTP1(IDC_CBO_SBC    , PFX_CBO);
249 :     TTP2(IDC_CBO_SBC    , PFX_CBO);
250 :     TTP1(IDC_LST        , PFX_LST);
251 :     TTP1(IDC_INP        , PFX_INP);
252 :     TTP1(IDC_LBX        , PFX_LBX);
253 :     TTP1(IDC_VTH        , PFX_VTH);
254 :     TTP1(IDC_TCH        , PFX_TCH);
255 :
256 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_CMD_CLEAR    ), TEXT("上記コントロールを全てクリアー"));
257 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_CMD_LOAD     ), TEXT("永続化情報読み出し¥n")
258 :                                                        TEXT("    ・オプション(AJCCTL_PSEL_・・) は、チェックボックスの設定内容を反映する。¥n")
259 :                                                        TEXT("    ・押下するとセーブボタンが有効となる。"));
260 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_CMD_SAVE     ), TEXT("永続化情報書き込み¥n")
261 :                                                        TEXT("    ・オプション(AJCCTL_PSEL_・・) は、ロード時の値が使用される。"));
262 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_CMD_SAVE_EX  ), TEXT("永続化情報書き込み¥n")
263 :                                                        TEXT("    ・オプション(AJCCTL_PSEL_・・) は、チェックボックスの設定内容を反映する。"));
264 :
265 :     return TRUE;
266 : }
267 : //----- ウインド破棄 ----------------------------------------------------------------------------------------//
268 : AJC_DLGPROC(Data, WM_DESTROY       )
269 : {
```

```
270 :     return TRUE;
271 : }
272 : //----- WM_TIMER (IDC_INP, AJCIVN_INTVALUE イベントは非同期で発生する為、ディレィ後に fLoad フラグをリセットする) -//
273 : AJC_DLGPROC(Data, WM_TIMER          )
274 : {
275 :     fLoad = FALSE;
276 :     KillTimer(hDlg, 1);
277 :     return TRUE;
278 : }
279 : //----- IDC_TXT_EDT ----------------------------------------------------------------------------------------//
280 : AJC_DLGPROC(Data, IDC_TXT_EDT       )
281 : {
282 :     if (HIWORD(wParam) == EN_CHANGE) {
283 :         if (fLoad) AjcVthPrintF(hVthLog, TEXT("Event  IDC_TXT_EDT  EN_CHANGE¥n"));
284 :     }
285 :     return TRUE;
286 : }
287 : //----- IDC_TXT_RDO ----------------------------------------------------------------------------------------//
288 : AJC_DLGPROC(Data, IDC_TXT_RDO       )
289 : {
290 :     if (HIWORD(wParam) == EN_CHANGE) {
291 :         if (fLoad) AjcVthPrintF(hVthLog, TEXT("Event  IDC_TXT_RDO  EN_CHANGE¥n"));
292 :     }
293 :     return TRUE;
294 : }
295 : //----- IDC_CHK --------------------------------------------------------------------------------------------//
296 : AJC_DLGPROC(Data, IDC_CHK           )
297 : {
298 :     if (HIWORD(wParam) == BN_CLICKED) {
299 :         if (fLoad) AjcVthPrintF(hVthLog, TEXT("Event  IDC_CHK      BN_CLICKED¥n"));
300 :     }
301 :     return TRUE;
302 : }
303 : //----- IDC_RBT1 -------------------------------------------------------------------------------------------//
304 : AJC_DLGPROC(Data, IDC_RBT1          )
305 : {
306 :     if (HIWORD(wParam) == BN_CLICKED) {
307 :         if (fLoad) AjcVthPrintF(hVthLog, TEXT("Event  IDC_RBT1     BN_CLICKED¥n"));
308 :     }
309 :     return TRUE;
310 : }
311 : //----- IDC_RBT2 -------------------------------------------------------------------------------------------//
312 : AJC_DLGPROC(Data, IDC_RBT2          )
313 : {
314 :     if (HIWORD(wParam) == BN_CLICKED) {
315 :         if (fLoad) AjcVthPrintF(hVthLog, TEXT("Event  IDC_RBT2     BN_CLICKED¥n"));
316 :     }
317 :     return TRUE;
318 : }
319 : //----- IDC_CBO_DDN ----------------------------------------------------------------------------------------//
320 : AJC_DLGPROC(Data, IDC_CBO_DDN       )
321 : {
322 :     if (HIWORD(wParam) == CBN_SELENDOK) {
323 :         if (fLoad) AjcVthPrintF(hVthLog, TEXT("Event  IDC_CBO_DDN  CBN_SELENDOK¥n"));
324 :     }
325 :     return TRUE;
326 : }
327 : //----- IDC_CBO_DDL ----------------------------------------------------------------------------------------//
328 : AJC_DLGPROC(Data, IDC_CBO_DDL       )
329 : {
330 :     if (HIWORD(wParam) == CBN_SELENDOK) {
331 :         if (fLoad) AjcVthPrintF(hVthLog, TEXT("Event  IDC_CBO_DDL  CBN_SELENDOK¥n"));
332 :     }
333 :     return TRUE;
334 : }
335 : //----- IDC_CBO_SBC ----------------------------------------------------------------------------------------//
336 : AJC_DLGPROC(Data, IDC_CBO_SBC       )
337 : {
338 :     if (HIWORD(wParam) == CBN_SELENDOK) {
339 :         if (fLoad) AjcVthPrintF(hVthLog, TEXT("Event  IDC_CBO_SBC  CBN_SELENDOK¥n"));
340 :     }
341 :     return TRUE;
342 : }
343 : //----- IDC_INP --------------------------------------------------------------------------------------------//
344 : AJC_DLGPROC(Data, IDC_INP           )
345 : {
346 :     if (HIWORD(wParam) == AJCIVN_INTVALUE) {
347 :         if (fLoad) AjcVthPrintF(hVthLog, TEXT("Event  IDC_INP      AJCIVN_INTVALUE¥n"));
348 :     }
349 :     return TRUE;
350 : }
351 : //----- クリアーボタン -------------------------------------------------------------------------------------//
352 : AJC_DLGPROC(Data, IDC_CMD_CLEAR     )
353 : {
354 :     AjcSetDlgItemStr     (hDlg, IDC_TXT_EDT, TEXT(""));
355 :     AjcSetDlgItemStr     (hDlg, IDC_TXT_RDO, TEXT(""));
356 :     AjcSetDlgItemChk     (hDlg, IDC_CHK    , FALSE);
357 :     AjcSetDlgItemChk     (hDlg, IDC_RBT1   , TRUE );
358 :     AjcSetDlgItemChk     (hDlg, IDC_RBT2   , FALSE);
359 :     AjcSetDlgItemCboReset(hDlg, IDC_CBO_DDN);
```

```
360 :     AjcSetDlgItemCboReset(hDlg, IDC_CBO_DDL);
361 :     AjcSetDlgItemCboReset(hDlg, IDC_CBO_SBC);
362 :     SendDlgItemMessage   (hDlg, IDC_LST    , LB_RESETCONTENT, 0, 0);
363 :     AjcSetDlgItemUInt    (hDlg, IDC_INP    , 0);
364 :     AjcLbxResetContent   (GetDlgItem(hDlg, IDC_LBX));
365 :     SendDlgItemMessage   (hDlg, IDC_VTH, WM_SETFONT, (WPARAM)AjcGetStockFont(AJCFID_FIX10), TRUE);
366 :     AjcTchSetRealRange   (GetDlgItem(hDlg, IDC_TCH), -2.0, 2.0);
367 :     return TRUE;
368 : }
369 : //----- 初期化ボタン ----------------------------------------------------------------------------------//
370 : AJC_DLGPROC(Data, IDC_CMD_INIT      )
371 : {
372 :     double      t;
373 :     double      dat[2];
374 :
375 :     //  全項目クリアー
376 :     SendMessage(hDlg, WM_COMMAND, MAKELONG(IDC_CMD_CLEAR, BN_CLICKED), (LPARAM)GetDlgItem(hDlg, IDC_CMD_CLEAR));
377 :
378 :     //  ダイアログ項目初期化
379 :     AjcSetDlgItemStr    (hDlg, IDC_TXT_EDT, TEXT("Editable Text"));
380 :     AjcSetDlgItemStr    (hDlg, IDC_TXT_RDO, TEXT("Read-Only Text"));
381 :
382 :     AjcSetDlgItemChk    (hDlg, IDC_CHK , FALSE);
383 :     AjcSetDlgItemChk    (hDlg, IDC_RBT1, TRUE );
384 :     AjcSetDlgItemChk    (hDlg, IDC_RBT2, FALSE);
385 :
386 :     AjcSetDlgItemCboIns (hDlg, IDC_CBO_DDN, -1, TEXT("Drop down 11"), AJCCBF_EXCLUSION);
387 :     AjcSetDlgItemCboIns (hDlg, IDC_CBO_DDN, -1, TEXT("Drop down 12"), AJCCBF_EXCLUSION);
388 :     AjcSetDlgItemCboIns (hDlg, IDC_CBO_DDN, -1, TEXT("Drop down 13"), AJCCBF_EXCLUSION);
389 :     AjcSetDlgItemCboData(hDlg, IDC_CBO_DDN, 0, 11); //  0xB
390 :     AjcSetDlgItemCboData(hDlg, IDC_CBO_DDN, 1, 12); //  0xC
391 :     AjcSetDlgItemCboData(hDlg, IDC_CBO_DDN, 2, 13); //  0xD
392 :     AjcSetDlgItemCboIx  (hDlg, IDC_CBO_DDN, 0);
393 :
394 :     AjcSetDlgItemCboIns (hDlg, IDC_CBO_DDL, -1, TEXT("Drop down list 21"), AJCCBF_EXCLUSION);
395 :     AjcSetDlgItemCboIns (hDlg, IDC_CBO_DDL, -1, TEXT("Drop down list 22"), AJCCBF_EXCLUSION);
396 :     AjcSetDlgItemCboIns (hDlg, IDC_CBO_DDL, -1, TEXT("Drop down list 23"), AJCCBF_EXCLUSION);
397 :     AjcSetDlgItemCboData(hDlg, IDC_CBO_DDL, 0, 21); //  0x15
398 :     AjcSetDlgItemCboData(hDlg, IDC_CBO_DDL, 1, 22); //  0x16
399 :     AjcSetDlgItemCboData(hDlg, IDC_CBO_DDL, 2, 23); //  0x17
400 :     AjcSetDlgItemCboIx  (hDlg, IDC_CBO_DDL, 0);
401 :
402 :     AjcSetDlgItemCboIns (hDlg, IDC_CBO_SBC, -1, TEXT("Subclass 1"                    ), AJCCBF_EXCLUSION);
403 :     AjcSetDlgItemCboIns (hDlg, IDC_CBO_SBC, -1, TEXT("Subclass 2"                    ), AJCCBF_EXCLUSION);
404 :     AjcSetDlgItemCboIns (hDlg, IDC_CBO_SBC, -1, TEXT("Subclass 3 +++++ LongText +++++"), AJCCBF_EXCLUSION);
405 :     AjcSetDlgItemCboIx  (hDlg, IDC_CBO_SBC, 0);
406 :
407 :     SendDlgItemMessage  (hDlg, IDC_LST, LB_INSERTSTRING, -1, (LPARAM)TEXT("List box item 41"));
408 :     SendDlgItemMessage  (hDlg, IDC_LST, LB_INSERTSTRING, -1, (LPARAM)TEXT("List box item 42"));
409 :     SendDlgItemMessage  (hDlg, IDC_LST, LB_INSERTSTRING, -1, (LPARAM)TEXT("List box item 43"));
410 :     SendDlgItemMessage  (hDlg, IDC_LST, LB_SETITEMDATA, 0, 41); //  0x29
411 :     SendDlgItemMessage  (hDlg, IDC_LST, LB_SETITEMDATA, 1, 42); //  0x2A
412 :     SendDlgItemMessage  (hDlg, IDC_LST, LB_SETITEMDATA, 2, 43); //  0x2B
413 :     SendDlgItemMessage  (hDlg, IDC_LST, LB_SETCURSEL   ,  0, 0);
414 :     AjcSetDlgItemSInt   (hDlg, IDC_INP, 0);
415 :
416 :     AjcLbxInsertString  (GetDlgItem(hDlg, IDC_LBX), -1, TEXT("Extended List item 51"));
417 :     AjcLbxInsertString  (GetDlgItem(hDlg, IDC_LBX), -1, TEXT("Extended List item 52"));
418 :     AjcLbxInsertString  (GetDlgItem(hDlg, IDC_LBX), -1, TEXT("Extended List item 53"));
419 :     AjcLbxSetItemData   (GetDlgItem(hDlg, IDC_LBX), 0, 51); //  0x33
420 :     AjcLbxSetItemData   (GetDlgItem(hDlg, IDC_LBX), 1, 52); //  0x34
421 :     AjcLbxSetItemData   (GetDlgItem(hDlg, IDC_LBX), 2, 53); //  0x35
422 :     AjcLbxSetSel        (GetDlgItem(hDlg, IDC_LBX), 0, TRUE);
423 :
424 :     //  VT100 にダミーデータ設定
425 :     AjcVthPutText       (GetDlgItem(hDlg, IDC_VTH), TEXT("あいうえお, かきくけこ, さしすせそ¥nABCDEFGHIJKLMNOPQRSTUVWXYZ"), -1);
426 :     //  タイムチャートにダミーデータ設定
427 :     for (t = 0.0; t < 360.0; t++) {
428 :         dat[0] = AjcSin(t);
429 :         dat[1] = AjcCos(t);
430 :         AjcTchPutRealData(GetDlgItem(hDlg, IDC_TCH), dat);
431 :     }
432 :     return TRUE;
433 : }
434 : //----- ロード ボタン ----------------------------------------------------------------------------------//
435 : AJC_DLGPROC(Data, IDC_CMD_LOAD      )
436 : {
437 :     UI      opt;
438 :
439 :     AjcVthPrintF(hVthLog, TEXT("¥n----- LOAD Button Pushed -----¥n"));
440 :     //  ロード中の旨、フラグ設定（フラグ解除はタイマで遅延する）
441 :     fLoad = TRUE;
442 :     SetTimer(hDlg, 1, 1, NULL);
443 :
444 :     //  プロファイル記憶先を .INI ファイルに設定
445 :     AjcSetProfileIsRegistory(FALSE);
446 :     //  現在の永続化オプション／各コントロールの永続化動作設定
447 :     opt = SetPermOpt();
448 :     //  プロファイルから各コントロールの設定値を読み出す
449 :     AjcLoadAllControlSettings(hDlgData, TEXT("DataDialogItems"), opt);
```

```
450 :     //  プロファイル記憶先を レジストリに戻す
451 :     AjcSetProfileIsRegistory(TRUE);
452 :     //  「ロードボタン押下時の永続化設定でセーブ」ボタン有効化
453 :     AjcEnableDlgItem(hDlg, IDC_CMD_SAVE, TRUE);
454 :
455 :     return TRUE;
456 : }
457 : //-----  「セーブ」 ボタン ----------------------------------------------------------------------------------//
458 : AJC_DLGPROC(Data, IDC_CMD_SAVE      )
459 : {
460 :     //  プロファイル記憶先を .INI ファイルに設定
461 :     AjcSetProfileIsRegistory(FALSE);
462 :     //  プロファイルセクションを削除 (クリーンアップ)
463 :     AjcDelProfileSect(TEXT("DataDialogItems"));
464 :     //  INI ファイルへ各コントロールの値をセーブ
465 :     AjcSaveAllControlSettings(hDlg);
466 :     //  プロファイル記憶先を レジストリに戻す
467 :     AjcSetProfileIsRegistory(TRUE);
468 :
469 :     return TRUE;
470 : }
471 : //-----  「単独セーブ」 ボタン ------------------------------------------------------------------------------//
472 : AJC_DLGPROC(Data, IDC_CMD_SAVE_EX   )
473 : {
474 :     UI      opt;
475 :
476 :     //  プロファイル記憶先を .INI ファイルに設定
477 :     AjcSetProfileIsRegistory(FALSE);
478 :     //  現在の永続化オプション／各コントロールの永続化動作設定
479 :     opt = SetPermOpt();
480 :     //  プロファイルセクションを削除 (クリーンアップ)
481 :     AjcDelProfileSect(TEXT("DataDialogItems"));
482 :     //  INI ファイルへ各コントロールの値をセーブ
483 :     AjcSaveAllControlSettingsEx(hDlg, TEXT("DataDialogItems"), opt);
484 :     //  プロファイル記憶先を レジストリに戻す
485 :     AjcSetProfileIsRegistory(TRUE);
486 :
487 :     return TRUE;
488 : }
489 : //-----------------------------------------------------------------------------------------------------------//
490 : AJC_DLGMAP_DEF(Data)
491 :     AJC_DLGMAP_MSG(Data, WM_INITDIALOG          )
492 :     AJC_DLGMAP_MSG(Data, WM_DESTROY             )
493 :     AJC_DLGMAP_MSG(Data, WM_TIMER               )
494 :
495 :     AJC_DLGMAP_CMD(Data, IDC_TXT_EDT            )
496 :     AJC_DLGMAP_CMD(Data, IDC_TXT_RDO            )
497 :     AJC_DLGMAP_CMD(Data, IDC_CHK                )
498 :     AJC_DLGMAP_CMD(Data, IDC_RBT1               )
499 :     AJC_DLGMAP_CMD(Data, IDC_RBT2               )
500 :     AJC_DLGMAP_CMD(Data, IDC_CBO_DDN            )
501 :     AJC_DLGMAP_CMD(Data, IDC_CBO_DDL            )
502 :     AJC_DLGMAP_CMD(Data, IDC_CBO_SBC            )
503 :
504 :     AJC_DLGMAP_CMD(Data, IDC_INP                )
505 :     AJC_DLGMAP_CMD(Data, IDC_CMD_CLEAR          )
506 :     AJC_DLGMAP_CMD(Data, IDC_CMD_INIT           )
507 :     AJC_DLGMAP_CMD(Data, IDC_CMD_LOAD           )
508 :     AJC_DLGMAP_CMD(Data, IDC_CMD_SAVE           )
509 :     AJC_DLGMAP_CMD(Data, IDC_CMD_SAVE_EX        )
510 : AJC_DLGMAP_END
511 :
512 : //-----------------------------------------------------------------------------------------------------------//
513 : //  永続化オプション, 各コントロールの永続化動作設定                                                          //
514 : //-----------------------------------------------------------------------------------------------------------//
515 : static  UI      SetPermOpt(VO)
516 : {
517 :     UI  opt = 0;
518 :     UI  att = 0;
519 :
520 :     //  永続化オプション設定
521 :     opt = 0;
522 :     if      (AjcGetDlgItemChk(hDlgMain, IDC_RBT_SELACT_CHKINCLUDE)) opt = AJCCTL_SELACT_CHKINCLUDE;
523 :     else if (AjcGetDlgItemChk(hDlgMain, IDC_RBT_SELACT_CHKEXCLUDE)) opt = AJCCTL_SELACT_CHKEXCLUDE;
524 :     else                                                            opt = AJCCTL_SELACT_ALL;
525 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_SELACT_NTCCHK)) opt |= AJCCTL_SELACT_NTCCHK;
526 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_SELACT_NTCRBT)) opt |= AJCCTL_SELACT_NTCRBT;
527 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_SELACT_NTCCBO)) opt |= AJCCTL_SELACT_NTCCBO;
528 :     //  永続化動作設定 (IDC_TXT_EDT)
529 :     att = 0;
530 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_EDT_INCLUDE) ? AJCCTL_PSEL_INCLUDE : 0);
531 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_EDT_EXCLUDE) ? AJCCTL_PSEL_EXCLUDE : 0);
532 :     AjcDlgItemSetPermAtt(hDlgData, IDC_TXT_EDT, att);
533 :     //  永続化動作設定 (IDC_TXT_RDO)
534 :     att = 0;
535 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_RDO_INCLUDE  ) ? AJCCTL_PSEL_INCLUDE   : 0);
536 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_RDO_EXCLUDE  ) ? AJCCTL_PSEL_EXCLUDE   : 0);
537 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_RDO_INCRDONLY) ? AJCCTL_PSEL_INCRDONLY : 0);
538 :     AjcDlgItemSetPermAtt(hDlgData, IDC_TXT_RDO, att);
539 :     //  永続化動作設定 (IDC_CHK)
```

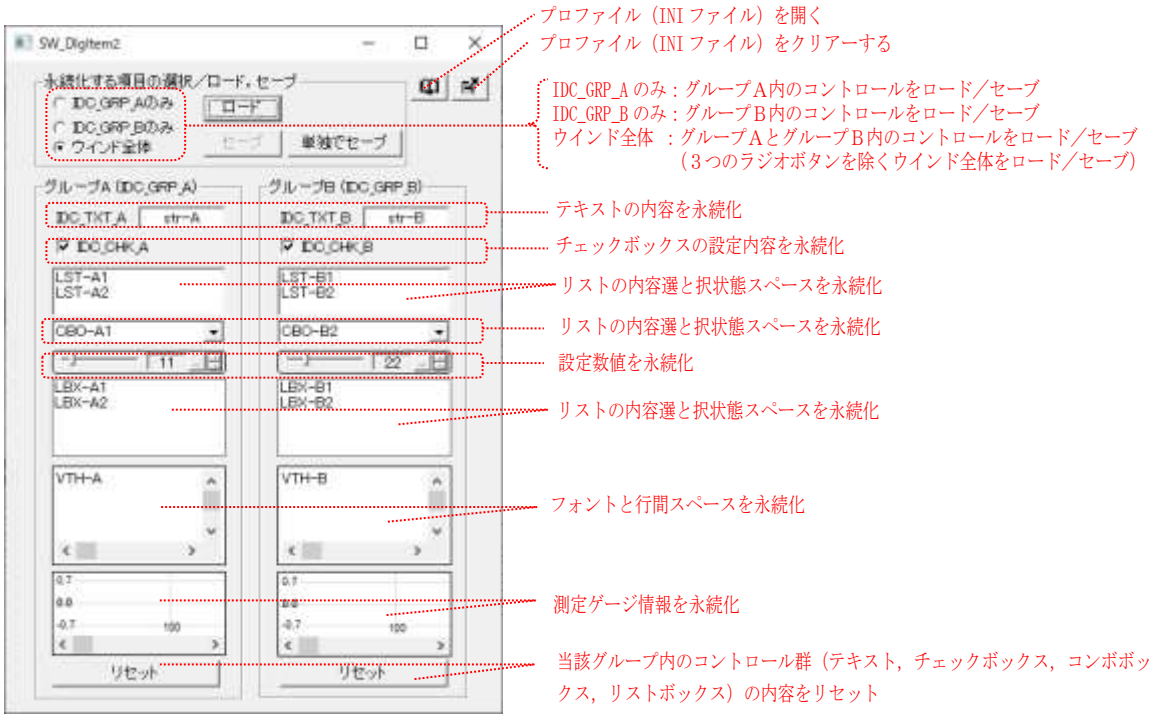
```
540 :     att = 0;
541 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_CHK_INCLUDE) ? AJCCTL_PSEL_INCLUDE : 0);
542 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_CHK_EXCLUDE) ? AJCCTL_PSEL_EXCLUDE : 0);
543 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_CHK_NTC    ) ? AJCCTL_PSEL_NTC     : 0);
544 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_CHK_NONTC  ) ? AJCCTL_PSEL_NONTC   : 0);
545 :     AjcDlgItemSetPermAtt(hDlgData, IDC_CHK, att);
546 :     //  永続化動作設定 (IDC_RBT1)
547 :     att = 0;
548 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_RBT1_INCLUDE) ? AJCCTL_PSEL_INCLUDE : 0);
549 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_RBT1_EXCLUDE) ? AJCCTL_PSEL_EXCLUDE : 0);
550 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_RBT1_NTC    ) ? AJCCTL_PSEL_NTC     : 0);
551 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_RBT1_NONTC  ) ? AJCCTL_PSEL_NONTC   : 0);
552 :     AjcDlgItemSetPermAtt(hDlgData, IDC_RBT1, att);
553 :     //  永続化動作設定 (IDC_RBT2)
554 :     att = 0;
555 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_RBT2_INCLUDE) ? AJCCTL_PSEL_INCLUDE : 0);
556 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_RBT2_EXCLUDE) ? AJCCTL_PSEL_EXCLUDE : 0);
557 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_RBT2_NTC    ) ? AJCCTL_PSEL_NTC     : 0);
558 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_RBT2_NONTC  ) ? AJCCTL_PSEL_NONTC   : 0);
559 :     AjcDlgItemSetPermAtt(hDlgData, IDC_RBT2, att);
560 :     //  永続化動作設定 (IDC_CBO_DDN)
561 :     att = 0;
562 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_DDN_INCLUDE) ? AJCCTL_PSEL_INCLUDE : 0);
563 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_DDN_EXCLUDE) ? AJCCTL_PSEL_EXCLUDE : 0);
564 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_DDN_NTC    ) ? AJCCTL_PSEL_NTC     : 0);
565 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_DDN_NONTC  ) ? AJCCTL_PSEL_NONTC   : 0);
566 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_DDN_INCDATA) ? AJCCTL_PSEL_INCDATA : 0);
567 :     //  ドロップダウンの場合、規定でリストは永続化されるため、以下のフラグは指定不要
568 : //  att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_DDN_INCLIST) ? AJCCTL_PSEL_INCLIST : 0);
569 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_DDN_EXCLIST) ? AJCCTL_PSEL_EXCLIST : 0);
570 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_DDN_EXCSEL ) ? AJCCTL_PSEL_EXCSEL  : 0);
571 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_DDN_EXCEDIT) ? AJCCTL_PSEL_EXCEDIT : 0);
572 :     AjcDlgItemSetPermAtt(hDlgData, IDC_CBO_DDN, att);
573 :     //  永続化動作設定 (IDC_CBO_DDL)
574 :     att = 0;
575 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_DDL_INCLUDE) ? AJCCTL_PSEL_INCLUDE : 0);
576 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_DDL_EXCLUDE) ? AJCCTL_PSEL_EXCLUDE : 0);
577 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_DDL_NTC    ) ? AJCCTL_PSEL_NTC     : 0);
578 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_DDL_NONTC  ) ? AJCCTL_PSEL_NONTC   : 0);
579 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_DDL_INCDATA) ? AJCCTL_PSEL_INCDATA : 0);
580 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_DDL_INCLIST) ? AJCCTL_PSEL_INCLIST : 0);
581 :     //  ドロップダウンリストの場合、規定でリストは永続化されないため、以下のフラグは指定不要
582 : //  att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_DDL_EXCLIST) ? AJCCTL_PSEL_EXCLIST : 0);
583 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_DDL_EXCSEL ) ? AJCCTL_PSEL_EXCSEL  : 0);
584 :     AjcDlgItemSetPermAtt(hDlgData, IDC_CBO_DDL, att);
585 :     //  永続化動作設定 (IDC_CBO_SBC)
586 :     att = 0;
587 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_SBC_INCLUDE) ? AJCCTL_PSEL_INCLUDE : 0);
588 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_SBC_EXCLUDE) ? AJCCTL_PSEL_EXCLUDE : 0);
589 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_SBC_NTC    ) ? AJCCTL_PSEL_NTC     : 0);
590 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_SBC_NONTC  ) ? AJCCTL_PSEL_NONTC   : 0);
591 :     //  AjcSbcComboBox()でサブクラス化したコンボボックスの場合、以下の２つのフラグは無条件に ON となる
592 : //  att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_SBC_INCDATA) ? AJCCTL_PSEL_INCDATA : 0);
593 : //  att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_SBC_INCLIST) ? AJCCTL_PSEL_INCLIST : 0);
594 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_SBC_EXCLIST) ? AJCCTL_PSEL_EXCLIST : 0);
595 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_SBC_EXCSEL ) ? AJCCTL_PSEL_EXCSEL  : 0);
596 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_SBC_EXCEDIT) ? AJCCTL_PSEL_EXCEDIT : 0);
597 :     AjcDlgItemSetPermAtt(hDlgData, IDC_CBO_SBC, att);
598 :     //  永続化動作設定 (IDC_LST)
599 :     att = 0;
600 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_LST_INCLUDE) ? AJCCTL_PSEL_INCLUDE : 0);
601 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_LST_EXCLUDE) ? AJCCTL_PSEL_EXCLUDE : 0);
602 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_LST_INCDATA) ? AJCCTL_PSEL_INCDATA : 0);
603 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_LST_EXCSEL ) ? AJCCTL_PSEL_EXCSEL  : 0);
604 :     AjcDlgItemSetPermAtt(hDlgData, IDC_LST, att);
605 :     //  永続化動作設定 (IDC_INP)
606 :     att = 0;
607 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_INP_INCLUDE) ? AJCCTL_PSEL_INCLUDE : 0);
608 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_INP_EXCLUDE) ? AJCCTL_PSEL_EXCLUDE : 0);
609 :     AjcDlgItemSetPermAtt(hDlgData, IDC_INP, att);
610 :     //  永続化動作設定 (IDC_LBX)
611 :     att = 0;
612 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_LBX_INCLUDE) ? AJCCTL_PSEL_INCLUDE : 0);
613 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_LBX_EXCLUDE) ? AJCCTL_PSEL_EXCLUDE : 0);
614 :     AjcDlgItemSetPermAtt(hDlgData, IDC_LBX, att);
615 :     //  永続化動作設定 (IDC_VTH)
616 :     att = 0;
617 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_VTH_INCLUDE) ? AJCCTL_PSEL_INCLUDE : 0);
618 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_VTH_EXCLUDE) ? AJCCTL_PSEL_EXCLUDE : 0);
619 :     AjcDlgItemSetPermAtt(hDlgData, IDC_VTH, att);
620 :     //  永続化動作設定 (IDC_TCH)
621 :     att = 0;
622 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_TCH_INCLUDE) ? AJCCTL_PSEL_INCLUDE : 0);
623 :     att |= (AjcGetDlgItemChk(hDlgMain, IDC_CHK_TCH_EXCLUDE) ? AJCCTL_PSEL_EXCLUDE : 0);
624 :     AjcDlgItemSetPermAtt(hDlgData, IDC_TCH, att);
625 :
626 :     return opt;
627 : }
628 :
629 : //-------------------------------------------------------------------------------------------------------------//
```

```
630 : //  各コントロールのチップテキスト設定                                                                                //
631 : //--------------------------------------------------------------------------------------------------------------------//
632 : static  C_UTP CALLBACK cbGetTipText(HWND hCtl, UTP pBuf, UI lBuf, UX cbp)
633 : {
634 :     UI      id      = LOWORD(cbp);
635 :     UI      pfx     = HIWORD(cbp);
636 :     C_UTP   pPrefix = szPrefix[pfx];
637 :
638 :     int     stl;
639 :     int     i, cnt;
640 :     UX      dat;
641 :
642 :     //  チップテキスト (プロファイルキー名称) 設定
643 :     AjcSnPrintF(pBuf, lBuf, TEXT("プロファイルキー : %s_%06d"), pPrefix, id);
644 :     stl = (int)MAjcStrLen(pBuf);
645 :
646 :     if      (MAjcStrCmp(pPrefix, TEXT("Edt")) == 0) {
647 :         //  None
648 :     }
649 :     else if (MAjcStrCmp(pPrefix, TEXT("Chk")) == 0) {
650 :         //  None
651 :     }
652 :     else if (MAjcStrCmp(pPrefix, TEXT("Rbt")) == 0) {
653 :         //  None
654 :     }
655 :     else if (MAjcStrCmp(pPrefix, TEXT("Cbo")) == 0) {
656 :         //  チップテキストにデータ値追加
657 :         if (cnt = AjcGetCtrlCboCount(GetDlgItem(hDlgData, id))) {
658 :             AjcSnPrintF(&pBuf[stl], lBuf - stl, TEXT("¥nDATA : ")); stl = (int)MAjcStrLen(pBuf);
659 :             for (i = 0; i < cnt; i++) {
660 :                 dat = (int)AjcGetCtrlCboData(GetDlgItem(hDlgData, id), i);
661 :                 AjcSnPrintF(&pBuf[stl], lBuf - stl, TEXT("  %d"), dat); stl = (int)MAjcStrLen(pBuf);
662 :             }
663 :         }
664 :     }
665 :     else if (MAjcStrCmp(pPrefix, TEXT("Lst")) == 0) {
666 :         //  チップテキストにデータ値追加
667 :         if (cnt = (int)SendMessage(hCtl, LB_GETCOUNT, 0, 0)) {
668 :             AjcSnPrintF(&pBuf[stl], lBuf - stl, TEXT("¥nDATA : ")); stl = (int)MAjcStrLen(pBuf);
669 :             for (i = 0; i < cnt; i++) {
670 :                 dat = (int)SendMessage(hCtl, LB_GETITEMDATA, i, 0);
671 :                 AjcSnPrintF(&pBuf[stl], lBuf - stl, TEXT("  %d"), dat); stl = (int)MAjcStrLen(pBuf);
672 :             }
673 :         }
674 :     }
675 :     else if (MAjcStrCmp(pPrefix, TEXT("Inp")) == 0) {
676 :         //  None
677 :     }
678 :     else if (MAjcStrCmp(pPrefix, TEXT("Lbx")) == 0) {
679 :         //  チップテキストにデータ値追加
680 :         if (cnt = AjcLbxGetCount(hCtl)) {
681 :             AjcSnPrintF(&pBuf[stl], lBuf - stl, TEXT("¥nDATA : ")); stl = (int)MAjcStrLen(pBuf);
682 :             for (i = 0; i < cnt; i++) {
683 :                 dat = (int)AjcLbxGetItemData(hCtl, i);
684 :                 AjcSnPrintF(&pBuf[stl], lBuf - stl, TEXT("  %d"), dat); stl = (int)MAjcStrLen(pBuf);
685 :             }
686 :         }
687 :     }
688 :     else if (MAjcStrCmp(pPrefix, TEXT("Vth")) == 0) {
689 :         AJCVTHPROP  prop;
690 :         //  チップテキストに設定情報追加
691 :         AjcVthGetProp(hCtl, &prop);
692 :         AjcSnPrintF(&pBuf[stl], lBuf - stl, TEXT("¥n")); stl = (int)MAjcStrLen(pBuf);
693 :         AjcSnPrintF(&pBuf[stl], lBuf - stl, TEXT("  lfHeight        (フォント高さ    ) = %d¥n")
694 :                                             TEXT("  lfWidth         (フォント幅      ) = %d¥n")
695 :                                             TEXT("  lfEscapement    (文字送り方向    ) = %d¥n")
696 :                                             TEXT("  lfOrientation   (文字の向き      ) = %d¥n")
697 :                                             TEXT("  lfWeight        (ボールド        ) = %d¥n")
698 :                                             TEXT("  lfItalic        (イタリック      ) = %d¥n")
699 :                                             TEXT("  lfUnderline     (アンダーライン  ) = %d¥n")
700 :                                             TEXT("  lfStrikeOut     (横線描画        ) = %d¥n")
701 :                                             TEXT("  lfCharSet       (文字セット      ) = %d¥n")
702 :                                             TEXT("  lfOutPrecision  (一致方法        ) = %d¥n")
703 :                                             TEXT("  lfClipPrecision (クリッピング    ) = %d¥n")
704 :                                             TEXT("  lfQuality       (照合方法        ) = %d¥n")
705 :                                             TEXT("  lfPitchAndFamily(ピッチ&ファミリ) = 0x%02X¥n")
706 :                                             TEXT("  lfFaceName      (フォント名      ) = %s¥n")
707 :                                             TEXT("  LSpace          (行間スペース    ) = %d¥n"),
708 :             prop.LogFont.lfHeight           ,       //  lfHeight
709 :             prop.LogFont.lfWidth            ,       //  lfWidth
710 :             prop.LogFont.lfEscapement       ,       //  lfEscapement
711 :             prop.LogFont.lfOrientation      ,       //  lfOrientation
712 :             prop.LogFont.lfWeight           ,       //  lfWeight
713 :             prop.LogFont.lfItalic           ,       //  lfItalic
714 :             prop.LogFont.lfUnderline        ,       //  lfUnderline
715 :             prop.LogFont.lfStrikeOut        ,       //  lfStrikeOut
716 :             prop.LogFont.lfCharSet          ,       //  lfCharSet
717 :             prop.LogFont.lfOutPrecision     ,       //  lfOutPrecision
718 :             prop.LogFont.lfClipPrecision    ,       //  lfClipPrecision
719 :             prop.LogFont.lfQuality          ,       //  lfQuality
```

```
720 :             prop.LogFont.lfPitchAndFamily   ,       //  lfPitchAndFamily
721 :             prop.LogFont.lfFaceName         ,       //  lfFaceName
722 :             prop.LSpace                             //  LSpace
723 :         );
724 :     }
725 :     else if (MAjcStrCmp(pPrefix, TEXT("Tch")) == 0) {
726 :         AJCTCGAUINFO    gau;
727 :         //  チップテキストに設定情報追加
728 :         AjcTchGetGauInfo(hCtl, &gau);
729 :         AjcSnPrintF(&pBuf[stl], lBuf - stl, TEXT("¥n")); stl = (int)MAjcStrLen(pBuf);
730 :         AjcSnPrintF(&pBuf[stl], lBuf - stl, TEXT("  GauBarL     (ゲージ低位置     ) = %d¥n")
731 :                                             TEXT("  GauBarH     (ゲージ高位置     ) = %d¥n")
732 :                                             TEXT("  fGauUseMes  (単位時間種別     ) = %d¥n")
733 :                                             TEXT("  GauSetUTime (単位時間(設定値)) = %d"),
734 :             gau.GauBarL                     ,       //  GauBarL
735 :             gau.GauBarH                     ,       //  GauBarH
736 :             gau.fGauUseMes                  ,       //  fGauUseMes
737 :             gau.GauSetUTime                         //  GauSetUTime
738 :         );
739 :     }
740 :
741 :     return pBuf;
742 : }
```

17.4.23. SW_DlgItem2（ウインド全体／グループボックス単位のコントロールの永続化）

このサンプルプログラムは、ウインド全体、あるいは、グループボックス単位での、コントロール永続化のチェック用プログラムです。
「ロード」ボタンを押すと、プロファイルに記録された情報を読み出して、グループＡ、グループＢ、あるいは、両グループ内の全コントロールへ設定します。
「セーブ」「単独でセーブ」ボタンを押すと、グループＡ、グループＢ、あるいは、両グループ内の全コントロールの内容をプロファイルへ記録します。
「セーブ」ボタンは、「ロード」ボタンを押すとアクティブになります。




```
 1 : //
 2 : // SW_DlgItem2.c
 3 : //
 4 :
 5 : #include   <AjrCstXX.h>
 6 : #include   <math.h>
 7 : #include   <tchar.h>
 8 : #include   "resource.h"
 9 :
10 : //──────────────────────────────────────────────────────────────────────────//
11 : // ワーク                                                                   //
12 : //──────────────────────────────────────────────────────────────────────────//
13 : static HINSTANCE     hInst;                  // ＤＬＬインスタンスハンドル
14 : static HWND          hDlgMain;               // ダイアログボックスハンドル
15 : static HICON          hIcoOpen;
16 : static HICON          hIcoDel;
17 : static BOOL           fLoadGrpA = FALSE;       // GRPA ロード済フラグ
18 : static BOOL           fLoadGrpB = FALSE;       // GRPB ロード済フラグ
19 : static BOOL           fLoadAll  = FALSE;       // 全体ロード済フラグ
20 : //──────────────────────────────────────────────────────────────────────────//
21 : // 内部サブ関数                                                             //
22 : //──────────────────────────────────────────────────────────────────────────//
23 : AJC_DLGPROC_DEF(Main);
24 : static UTP CALLBACK cbGetTipText(HWND hCtrl, UTP pBuf, UI lBuf, UX cbp);
25 :
26 : //══════════════════════════════════════════════════════════════════════════//
27 : //                                                                          //
28 : // ＷｉｎＭａｉｎ                                                          //
29 : //                                                                          //
30 : //══════════════════════════════════════════════════════════════════════════//
31 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
32 : {
33 :     MSG     msg;
34 :
35 :     hInst = hInstance;
36 :
37 :     //―― メイン・ダイアログオープン ――――――――――//
```

```
38 :    hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_DLGMAIN), NULL, AJC_DLGPROC_NAME(Main));
39 :    //―― ダイアログ表示 ――――――――――――――――――――――――//
40 :    ShowWindow(hDlgMain, SW_SHOW);
41 :
42 :    //―― メッセージループ ――――――――――――――――――――――――//
43 :    while (GetMessage(&msg, NULL, 0, 0)) {
44 :        do {
45 :            if (IsDialogMessage(hDlgMain, &msg)) break;
46 :            TranslateMessage(&msg);
47 :            DispatchMessage (&msg);
48 :        } while (0);
49 :    }
50 :
51 :    return (int)msg.wParam ;
52 : }
53 : //═══════════════════════════════════════════════════════════════════════════════//
54 : //                                                                               //
55 : //  ダイアログ・プロシージャ                                                     //
56 : //                                                                               //
57 : //═══════════════════════════════════════════════════════════════════════════════//
58 : //―― ダイアログ初期化 ―――――――――――――――――――――――――――――――――――――――//
59 : AJC_DLGPROC(Main, WM_INITDIALOG    )
60 : {
61 :    hDlgMain = hDlg;
62 :    // プロファイルを INI ファイルとする
63 :    AjcSetProfileIsRegistory(FALSE);
64 :    // アイコン生成
65 :    hIcoOpen  = (HICON)LoadImage(hInst, MAKEINTRESOURCE(IDI_OPEN), IMAGE_ICON, 16, 16, LR_DEFAULTCOLOR);
66 :    hIcoDel   = (HICON)LoadImage(hInst, MAKEINTRESOURCE(IDI_DEL ), IMAGE_ICON, 16, 16, LR_DEFAULTCOLOR);
67 :    // ボタンへアイコン表示
68 :    SendDlgItemMessage(hDlg, IDC_CMD_OPENINI, BM_SETIMAGE, IMAGE_ICON, (LPARAM)hIcoOpen);
69 :    SendDlgItemMessage(hDlg, IDC_CMD_CLEAR , BM_SETIMAGE, IMAGE_ICON, (LPARAM)hIcoDel );
70 :    // ラジオボタングループ化
71 :    AjcSbcRadioBtns(GetDlgItem(hDlg, IDC_GRP_LDSV));
72 :    // ツールチップ設定
73 :    AjcTipTextAdd(GetDlgItem(hDlg, IDC_CMD_OPENINI), TEXT("INI ファイルを開く"));
74 :    AjcTipTextAdd(GetDlgItem(hDlg, IDC_CMD_CLEAR ), TEXT("INI ファイルをクリアーする"));
75 :    // コントロールの初期値設定
76 :    AjcSetDlgItemChk  (hDlg, IDC_RBT_ALL, TRUE);
77 :    AjcSetDlgItemStr  (hDlg, IDC_TXT_A, TEXT("str-A"));   AjcSetDlgItemStr  (hDlg, IDC_TXT_B, TEXT("str-B"));
78 :    AjcSetDlgItemChk  (hDlg, IDC_CHK_A, TRUE);          AjcSetDlgItemChk  (hDlg, IDC_CHK_B, TRUE);
79 :    AjcSetDlgItemUInt (hDlg, IDC_INP_A, 11);            AjcSetDlgItemUInt (hDlg, IDC_INP_B, 22);
80 :    AjcSetDlgItemCboIns(hDlg, IDC_CBO_A, 0, TEXT("CBO-A1"), 0); AjcSetDlgItemCboIns(hDlg, IDC_CBO_B, 0, TEXT("CBO-B1"), 0);
81 :    AjcSetDlgItemCboIns(hDlg, IDC_CBO_A, 1, TEXT("CBO-A2"), 0); AjcSetDlgItemCboIns(hDlg, IDC_CBO_B, 1, TEXT("CBO-B2"), 0);
82 :    AjcSetDlgItemCboIx (hDlg, IDC_CBO_A, 0);            AjcSetDlgItemCboIx (hDlg, IDC_CBO_B, 1);
83 :    SendDlgItemMessage(hDlg, IDC_LST_A, LB_INSERTSTRING, -1, (LPARAM)TEXT("LST-A1"));
84 :    SendDlgItemMessage(hDlg, IDC_LST_A, LB_INSERTSTRING, -1, (LPARAM)TEXT("LST-A2"));
85 :    SendDlgItemMessage(hDlg, IDC_LST_B, LB_INSERTSTRING, -1, (LPARAM)TEXT("LST-B1"));
86 :    SendDlgItemMessage(hDlg, IDC_LST_B, LB_INSERTSTRING, -1, (LPARAM)TEXT("LST-B2"));
87 :    AjcLbxInsertString(GetDlgItem(hDlg, IDC_LBX_A), -1, TEXT("LBX-A1"));
88 :    AjcLbxInsertString(GetDlgItem(hDlg, IDC_LBX_A), -1, TEXT("LBX-A2"));
89 :    AjcLbxInsertString(GetDlgItem(hDlg, IDC_LBX_B), -1, TEXT("LBX-B1"));
90 :    AjcLbxInsertString(GetDlgItem(hDlg, IDC_LBX_B), -1, TEXT("LBX-B2"));
91 :    AjcVthPrintF(GetDlgItem(hDlg, IDC_VTH_A), TEXT("VTH-A"));
92 :    AjcVthPrintF(GetDlgItem(hDlg, IDC_VTH_B), TEXT("VTH-B"));
93 :    // ツールチップ関連付け
94 :    AjcTipTextAdd (GetDlgItem(hDlg, IDC_CMD_LOAD  ), TEXT("")); AjcTipTextSetCallBack(GetDlgItem(hDlg, IDC_CMD_LOAD  ), 0, NULL, cbGetTipText);
95 :    AjcTipTextAdd (GetDlgItem(hDlg, IDC_CMD_SAVE  ), TEXT("")); AjcTipTextSetCallBack(GetDlgItem(hDlg, IDC_CMD_SAVE  ), 0, NULL, cbGetTipText);
96 :    AjcTipTextAdd (GetDlgItem(hDlg, IDC_CMD_SAVE_EX), TEXT("")); AjcTipTextSetCallBack(GetDlgItem(hDlg, IDC_CMD_SAVE_EX), 0, NULL, cbGetTipText);
97 :    // ウインド全体セーブ／ロード時、選択情報（3つのラジオボタン）は除外する
98 :    AjcDlgItemSetPermAttGrp(hDlg, IDC_GRP_LDSV, AJCCTL_PSEL_EXCLUDE); // 選択情報(ラジオボタン)は除く
99 :    // 選択情報（3つのラジオボタン）ロード
100 :   AjcDlgLoadGrpControlSettings(hDlg, IDC_GRP_LDSV, TEXT("SecLDSV" ), AJCCTL_SELACT_ALL);
101 :
102 :   return TRUE;
103 : }
104 : //―― ウインド破棄 ―――――――――――――――――――――――――――――――――――――――//
105 : AJC_DLGPROC(Main, WM_DESTROY       )
106 : {
107 :   // 選択情報（3つのラジオボタン）セーブ
108 :   AjcDlgSaveGrpControlSettings(hDlg, IDC_GRP_LDSV);
109 :   // アイコン破棄
110 :   if (hIcoOpen  != NULL) {DeleteObject(hIcoOpen); hIcoOpen = NULL;}
111 :   //―― プログラム終了 ――――――――――――――――――――――――――――――――//
112 :   PostQuitMessage(0);
113 :   return TRUE;
114 : }
115 : //―― 3つのラジオボタン ―――――――――――――――――――――――――――――――――――――――//
116 : AJC_DLGPROC(Main, IDC_GRP_LDSV     )
117 : {
```

```
118 :     if (HIWORD(wParam) == AJCRBTN_SELECT) {
119 :         switch (lParam) {
120 :             case 0: AjcEnableDlgItem(hDlg, IDC_CMD_SAVE, fLoadGrpA); break; // GRP_A のみ
121 :             case 1: AjcEnableDlgItem(hDlg, IDC_CMD_SAVE, fLoadGrpB); break; // GRP_B のみ
122 :             case 2: AjcEnableDlgItem(hDlg, IDC_CMD_SAVE, fLoadAll); break; // ウインド全体
123 :         }
124 :     }
125 :     return TRUE;
126 : }
127 : //―― ロード ボタン ――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――//
128 : AJC_DLGPROC(Main, IDC_CMD_LOAD     )
129 : {
130 :     if     (AjcGetDlgItemChk(hDlg, IDC_RBT_A))
131 :         AjcEnableDlgItem(hDlg, IDC_CMD_SAVE, fLoadGrpA = AjcDlgLoadGrpControlSettings(hDlg, IDC_GRP_A, TEXT("SecA" ), AJCCTL_SELACT_ALL));
132 :     else if (AjcGetDlgItemChk(hDlg, IDC_RBT_B))
133 :         AjcEnableDlgItem(hDlg, IDC_CMD_SAVE, fLoadGrpB = AjcDlgLoadGrpControlSettings(hDlg, IDC_GRP_B, TEXT("SecB" ), AJCCTL_SELACT_ALL));
134 :     else
135 :         AjcEnableDlgItem(hDlg, IDC_CMD_SAVE, fLoadAll  = AjcLoadAllControlSettings   (hDlg,           TEXT("SecAll"), AJCCTL_SELACT_CHKEXCLUDE));
136 :     return TRUE;
137 : }
138 : //―― セーブ ボタン ――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――//
139 : AJC_DLGPROC(Main, IDC_CMD_SAVE     )
140 : {
141 :     if     (AjcGetDlgItemChk(hDlg, IDC_RBT_A)) AjcDlgSaveGrpControlSettings(hDlg, IDC_GRP_A);
142 :     else if (AjcGetDlgItemChk(hDlg, IDC_RBT_B)) AjcDlgSaveGrpControlSettings(hDlg, IDC_GRP_B);
143 :     else                                AjcSaveAllControlSettings   (hDlg);
144 :     return TRUE;
145 : }
146 : //―― 単独でセーブ ボタン ――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――//
147 : AJC_DLGPROC(Main, IDC_CMD_SAVE_EX  )
148 : {
149 :     if     (AjcGetDlgItemChk(hDlg, IDC_RBT_A)) AjcDlgSaveGrpControlSettingsEx(hDlg, IDC_GRP_A, TEXT("SecA" ), AJCCTL_SELACT_ALL);
150 :     else if (AjcGetDlgItemChk(hDlg, IDC_RBT_B)) AjcDlgSaveGrpControlSettingsEx(hDlg, IDC_GRP_B, TEXT("SecB" ), AJCCTL_SELACT_ALL);
151 :     else                                AjcSaveAllControlSettingsEx   (hDlg          , TEXT("SecAll"), AJCCTL_SELACT_CHKEXCLUDE);
152 :     return TRUE;
153 : }
154 : //―― プロファイル クリアー ボタン ――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――//
155 : AJC_DLGPROC(Main, IDC_CMD_CLEAR    )
156 : {
157 :     HANDLE  hFile;
158 :     UT      path[MAX_PATH];
159 :     AjcGetIniFilePath(path, MAX_PATH);
160 :     DeleteFile(path);
161 :     if ((hFile = CreateFile(path, GENERIC_WRITE, FILE_SHARE_WRITE, NULL, CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL)) != INVALID_HANDLE_VALUE) {
162 :         CloseHandle(hFile);
163 :     }
164 :     return TRUE;
165 : }
166 : //―― INI ファイルを開くボタン ――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――//
167 : AJC_DLGPROC(Main, IDC_CMD_OPENINI   )
168 : {
169 :     UT      path[MAX_PATH];
170 :     AjcGetIniFilePath(path, MAX_PATH);
171 :     ShellExecute(NULL, TEXT("open"), path, NULL, NULL, SW_SHOWNORMAL);
172 :     return TRUE;
173 : }
174 : //―― グループＡリセット ――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――//
175 : AJC_DLGPROC(Main, IDC_CMD_RESET_A  )
176 : {
177 :     AjcSetDlgItemStr  (hDlg, IDC_TXT_A, TEXT(""));
178 :     AjcSetDlgItemChk  (hDlg, IDC_CHK_A, FALSE);
179 :     AjcSetDlgItemUInt (hDlg, IDC_INP_A, 0);
180 :     SendDlgItemMessage(hDlg, IDC_CBO_A, CB_RESETCONTENT, 0, 0);
181 :     SendDlgItemMessage(hDlg, IDC_LST_A, LB_RESETCONTENT, 0, 0);
182 :     AjcLbxResetContent(GetDlgItem(hDlg, IDC_LBX_A));
183 :
184 :     return TRUE;
185 : }
186 : //―― グループＢリセット ――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――//
187 : AJC_DLGPROC(Main, IDC_CMD_RESET_B  )
188 : {
189 :     AjcSetDlgItemStr  (hDlg, IDC_TXT_B, TEXT(""));
190 :     AjcSetDlgItemChk  (hDlg, IDC_CHK_B, FALSE);
191 :     AjcSetDlgItemUInt (hDlg, IDC_INP_B, 0);
192 :     SendDlgItemMessage(hDlg, IDC_CBO_B, CB_RESETCONTENT, 0, 0);
193 :     SendDlgItemMessage(hDlg, IDC_LST_B, LB_RESETCONTENT, 0, 0);
194 :     AjcLbxResetContent(GetDlgItem(hDlg, IDC_LBX_B));
195 :
196 :     return TRUE;
197 : }
```

```
198 : //―― 「Cancel」ボタン ――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――//
199 : AJC_DLGPROC(Main, IDCANCEL        )
200 : {
201 :     DestroyWindow(hDlg);
202 :     return TRUE;
203 : }
204 : //――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――//
205 : AJC_DLGMAP_DEF(Main)
206 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG    )
207 :     AJC_DLGMAP_MSG(Main, WM_DESTROY       )
208 :     AJC_DLGMAP_CMD(Main, IDC_GRP_LDSV     )
209 :     AJC_DLGMAP_CMD(Main, IDC_CMD_LOAD     )
210 :     AJC_DLGMAP_CMD(Main, IDC_CMD_SAVE     )
211 :     AJC_DLGMAP_CMD(Main, IDC_CMD_SAVE_EX  )
212 :     AJC_DLGMAP_CMD(Main, IDC_CMD_CLEAR    )
213 :     AJC_DLGMAP_CMD(Main, IDC_CMD_OPENINI   )
214 :     AJC_DLGMAP_CMD(Main, IDC_CMD_RESET_A   )
215 :     AJC_DLGMAP_CMD(Main, IDC_CMD_RESET_B   )
216 :     AJC_DLGMAP_CMD(Main, IDCANCEL         )
217 : AJC_DLGMAP_END
218 :
219 : //――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――//
220 : //  チップテキスト取得のコールバック（ロード／セーブボタンで実行されるコードをチップ表示)                    //
221 : //――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――//
222 : static UTP CALLBACK cbGetTipText(HWND hCtrl, UTP pBuf, UI lBuf, UX cbp)
223 : {
224 :     switch (MAjcGetWindowLong(hCtrl, GWL_ID)) {
225 :       case IDC_CMD_LOAD:    MAjcStrCpy(pBuf, lBuf, TEXT("永続化情報読み出し（このボタン押下でセーブボタンを有効化）¥n"));  break;
226 :       case IDC_CMD_SAVE:    MAjcStrCpy(pBuf, lBuf, TEXT("永続化情報書き込み¥n"));                                  break;
227 :       case IDC_CMD_SAVE_EX: MAjcStrCpy(pBuf, lBuf, TEXT("永続化情報書き込み（ロードボタン押下不要）¥n"));              break;
228 :     }
229 :
230 :     switch (MAjcGetWindowLong(hCtrl, GWL_ID)) {
231 :       case IDC_CMD_LOAD:
232 :         switch (AjcGetDlgItemUInt(hDlgMain, IDC_GRP_LDSV)) {
233 :           case 0: MAjcStrCat(pBuf, lBuf, TEXT("AjcDlgLoadGrpControlSettings(hDlg, IDC_GRP_A, ¥"SecA¥", AJCCTL_SELACT_ALL);")); break;
234 :           case 1: MAjcStrCat(pBuf, lBuf, TEXT("AjcDlgLoadGrpControlSettings(hDlg, IDC_GRP_B, ¥"SecB¥", AJCCTL_SELACT_ALL);")); break;
235 :           case 2: MAjcStrCat(pBuf, lBuf, TEXT("AjcLoadAllControlSettings(hDlg, ¥"SecAll¥", AJCCTL_SELACT_ALL);")); break;
236 :         }
237 :         break;
238 :       case IDC_CMD_SAVE:
239 :         switch (AjcGetDlgItemUInt(hDlgMain, IDC_GRP_LDSV)) {
240 :           case 0: MAjcStrCat(pBuf, lBuf, TEXT("AjcDlgSaveGrpControlSettings(hDlg, IDC_GRP_A);")); break;
241 :           case 1: MAjcStrCat(pBuf, lBuf, TEXT("AjcDlgSaveGrpControlSettings(hDlg, IDC_GRP_B);")); break;
242 :           case 2: MAjcStrCat(pBuf, lBuf, TEXT("AjcSaveAllControlSettings(hDlg);")); break;
243 :         }
244 :         break;
245 :       case IDC_CMD_SAVE_EX:
246 :         switch (AjcGetDlgItemUInt(hDlgMain, IDC_GRP_LDSV)) {
247 :           case 0: MAjcStrCat(pBuf, lBuf, TEXT("AjcDlgSaveGrpControlSettingsEx(hDlg, IDC_GRP_A, ¥"IndepA¥"), AJCCTL_SELACT_ALL);")); break;
248 :           case 1: MAjcStrCat(pBuf, lBuf, TEXT("AjcDlgSaveGrpControlSettingsEx(hDlg, IDC_GRP_B, ¥"IndepB¥"), AJCCTL_SELACT_ALL);")); break;
249 :           case 2: MAjcStrCat(pBuf, lBuf, TEXT("AjcSaveAllControlSettingsEx(hDlg, ¥"IndepAll¥"), AJCCTL_SELACT_ALL);")); break;
250 :         }
251 :         break;
252 :     }
253 :     return pBuf;
254 : }
```

18. ウインドサポートＡＰＩ

ウインドのアクセスや操作をサポートするＡＰＩの集合です。

18.1. サポートＡＰＩ

ウインドサポートＡＰＩの一覧を以下に示します。

#	関数名	内容
1	AjcLoadWndPos AjcLoadWndPosEx	プロファイルからウインド位置を読み出す
2	AjcSaveWndPos AjcSaveWndPosEx	プロファイルへウインド位置を記録する
3	AjcLoadWndRect AjcLoadWndRectEx	プロファイルからウインド位置とサイズを読み出す
4	AjcSaveWndRect AjcSaveWndRectEx	プロファイルへウインド位置とサイズを記録する
5	AjcMoveWndIntoMonitor	ウインドの一部が隠れている場合 ウインドをモニタ内へ移動する
6	AjcMoveWndRectIntoMonitor	ウインド内矩の一部形が隠れている場合 ウインドをモニタ内へ移動する
7	AjcMoveOutWndToOrigin	ウインド全体が隠れている場合 ウインドを原点へ移動する
8	AjcIsSeeEntireInMonitors	全モニタで矩形が見えているかチェックする
9	AjcMoveWindowToCenter AjcMoveWindowToCenterOfScreen AjcMoveWindowToCenterOfMonitor AjcMoveWindowToCenterOfWindow	ウィンドを画面の中央へ移動する（親／オーナーウインドの中央） 〃　（デスクトップの中央） 〃　（自ウインドが属するモニタの中央） 〃　（他のウインドの中央）
10	AjcGetFocusWindow	現在フォーカスされているウインドを取得
11	AjcSetFocusWindow	指定ウインドをフォーカスする
12	AjcGetWindowNormalRect	通常表示時のウインド矩形情報取得
13	AjcSendMessageToThreadWindows	スレッドの全トップレベルウインドへメッセージ送信
14	AjcSendMessageToThreadWindowsEx	スレッドの全トップレベルウインドと、指定された２つのウインドへメッセージ送信
15	AjcEnableThreadWindows	スレッドの全トップレベルウインドを許可／禁止
16	AjcDoEvent	Windows イベント処理（ダイアログメッセージ処理なし）
17	AjcDoEventEx	Windows イベント処理（ダイアログメッセージ処理付き）
18	AjcGetExePathName	ウインドハンドルから実行プログラムファイルのパス名を取得する
19	AjcDefWindowProc	デフォルトウインドプロシージャの実行(UNICODE ウインド対応)
20	AjcCallWindowProc	ウインドプロシージャの呼び出し　(UNICODE ウインド対応)
21	AjcSetWindowLongPtr MAjcSetWindowLong	ウインドのＬＯＮＧ情報設定
22	AjcGetWindowLongPtr MAjcGetWindowLong	ウインドのＬＯＮＧ情報取得
23	AjcGetWndBkColor	ウインドの背景色取得
24	AjcGetWndBkBurush	ウインドの背景色のブラシハンドル取得

18.1.1. プロファイルからウインド位置を読み出す (AjcLoadWndPos)

形　式 ： BOOL　AjcLoadWndPos　(HWND hwnd, C_UTP pSec);
BOOL　AjcLoadWndPosEx (HWND hwnd, C_UTP pSec, C_UTP pKeyPrefix);

引　数 ： hwnd　　- ウインドハンドル
pSec　　- プロファイルのセクション名文字列のアドレス (NULL 指定時は “AjcWndPos”)
pKeyPrefix - プロファイルキーの接頭語文字列のアドレス

説　明 ： プロファイルから、ウインド位置を読み出し、ウインドを当該位置に移動します。
ウインドが画面に見えない位置になる場合は、ウインドを原点に移動します。
プロファイルのキー名称は、以下のとおりです。

[*KeyPrefix*_]AjcWndPX - ウインドのＸ位置
[*KeyPrefix*_]AjcWndPY - ウインドのＹ位置　　※「*KeyPrefix*」は、pKeyPrefix で指定した文字列

戻り値 ： TRUE　- 成功
FALSE - 失敗

18.1.2. プロファイルへウインド位置を記録する (AjcSaveWndPos)

形　式 ： BOOL　AjcSaveWndPos　(HWND hwnd, C_UTP pSec);
BOOL　AjcSaveWndPosEx (HWND hwnd, C_UTP pSec, C_UTP pKeyPrefix);

引　数 ： hwnd　　- ウインドハンドル
pSec　　- プロファイルのセクション名文字列のアドレス (NULL 指定時は “AjcWndPos”)
pKeyPrefix - プロファイルキーの接頭語文字列のアドレス

説　明 ： プロファイルへ、ウインド位置を記録します。
プロファイルのキー名称は、以下のとおりです。

[*KeyPrefix*_]AjcWndPX - ウインドのＸ位置
[*KeyPrefix*_]AjcWndPY - ウインドのＹ位置　　※「*KeyPrefix*」は、pKeyPrefix で指定した文字列

戻り値 ： TRUE　- 成功
FALSE - 失敗

18.1.3. プロファイルからウインド位置とサイズを読み出す (AjcLoadWndRect)

形　式 ： BOOL　AjcLoadWndRect　(HWND hwnd, C_UTP pSec);
BOOL　AjcLoadWndRectEx(HWND hwnd, C_UTP pSec, C_UTP pKeyPrefix);

引　数 ： hwnd　　- ウインドハンドル
pSec　　- プロファイルのセクション名文字列のアドレス (NULL 指定時は “AjcWndRect”)
pKeyPrefix - プロファイルキーの接頭語文字列のアドレス

説　明 ： プロファイルから、ウインド位置とサイズを読み出し、ウインドを当該位置とサイズに移動します。
ウインドが画面に見えない位置になる場合は、ウインドを原点に移動します。
プロファイルのキー名称は、以下のとおりです。

[*KeyPrefix*_]AjcWndPX - ウインドのＸ位置
[*KeyPrefix*_]AjcWndPY - ウインドのＹ位置
[*KeyPrefix*_]AjcWndCX - ウインドの幅
[*KeyPrefix*_]AjcWndCY - ウインドの高さ　　※「*KeyPrefix*」は、pKeyPrefix で指定した文字列

戻り値 ： TRUE　- 成功
FALSE - 失敗

18.1.4. プロファイルへウインド位置とサイズを記録する (AjcSaveWndRect)

形　式 ： BOOL AjcSaveWndRect (HWND hwnd, C_UTP pSec);
BOOL AjcSaveWndRectEx(HWND hwnd, C_UTP pSec, C_UTP pKeyPrefix);

引　数 ： hwnd - ウインドハンドル
pSec - プロファイルのセクション名文字列のアドレス (NULL 指定時は “AjcWndRect”)
pKeyPrefix - プロファイルキーの接頭語文字列のアドレス

説　明 ： プロファイルへ、ウインド位置とサイズを記録します。
プロファイルのキー名称は、以下の固定名称です。

[*KeyPrefix*_]AjcWndPX - ウインドのＸ位置
[*KeyPrefix*_]AjcWndPY - ウインドのＹ位置
[*KeyPrefix*_]AjcWndCX - ウインドの幅
[*KeyPrefix*_]AjcWndCY - ウインドの高さ　　※「*KeyPrefix*」は、pKeyPrefix で指定した文字列

戻り値 ： TRUE - 成功
FALSE - 失敗

18.1.5. ウインドの一部が隠れている場合 ウインドをモニタ内へ移動する(AjcMoveWndIntoMonitor)

形　式 ： BOOL AjcMoveWndIntoMonitor (HWND hwnd);

引　数 ： hwnd - ウインドハンドル

説　明 ： ウインドの一部が隠れている場合、あるいは、ウインドが複数のモニタにまたがっている場合、ウインド全体が見えるようにモニタ内へ移動します。　ウインド全体が隠れている場合は、ウインドを原点へ移動します。

戻り値 ： TRUE - ウインドを移動した
FALSE - ウインドは移動していない／エラー

18.1.6. ウインド内矩形の一部が隠れている場合 ウインドをモニタ内へ移動する(AjcMoveWndRectIntoMonitor)

形　式 ： BOOL AjcMoveWndRectIntoMonitor (HWND hwnd, LPRECT pRect);

引　数 ： hwnd - ウインドハンドル
pRect - ウインド内矩形情報（画面が座標）

説　明 ： ウインド内矩形の一部が隠れている場合、あるいは、矩形が複数のモニタにまたがっている場合、当該矩形が見えるようにウインドをモニタ内へ移動します。　矩形（あるいはウインド）全体が隠れている場合は、ウインドを原点へ移動します。

戻り値 ： TRUE - ウインドを移動した
FALSE - ウインドは移動していない／エラー

18.1.7. ウインド全体が隠れている場合 ウインドを原点へ移動する(AjcMoveWndRectIntoMonitor)

形　式 ： BOOL AjcMoveOutWndToOrigin (HWND hwnd);

引　数 ： hwnd - ウインドハンドル

説　明 ： ウインド全体が隠れている場合、(いずれのモニタにもウインドの一部すら表示されていない場合) ウインドを原点へ移動します。
いずれかのモニタにウインドの一部分が見えている場合は何もしません。

戻り値 ： TRUE - ウインドを移動した
FALSE - ウインドは移動していない／エラー

18.1.8. 全モニタで矩形が見えているかチェックする(AjcIsSeeEntireInMonitors)

形　式 ： BOOL　AjcIsSeeEntireInMonitors（LPRECT pRect, UIP pBounds, UIP pSqr);

引　数 ： hwnd　　- ウインドハンドル
pRect　　- 矩形情報（画面座標）
pBounds　- 見えているモニタ数を格納するバッファのアドレス（不要時は NULL）
pSqr　　- 見えている総面積（ピクセル数）を格納するバッファのアドレス（不要時は NULL）

説　明 ： pRect で指定した矩形が（モニタ群にまたがっている場合も含め）全て見えているかチェックします。
pBounds に格納される値は、矩形がいくつのモニタにまたがって見えているかの情報を示します。（モニタ数）
pSqr に格納される値は、矩形の見えている部分の総面積（ピクセル数）を示します。

ウインドの幅，高さがともに０の場合は以下の値が返されます。

・*pSqr = 0（面積＝０）となります。
・ウインド位置がいずれかのモニタ内であれば、*pBounds = 1 が設定されます。
・ウインド位置が全てのモニタ外の場合は、*pBounds = 0 となります。

戻り値 ： TRUE　- 矩形がすべて見えている（複数のモニタにまたがっている場合を含む）
FALSE – 矩形の一部／全部が見えない

18.1.9. ウィンドを画面の中央へ移動する (AjcMoveWindowToCenter・・・)

形　式 ： BOOL　AjcMoveWindowToCenter(HWND hwnd);
BOOL　AjcMoveWindowToCenterOfScreen (HWND hwnd);
BOOL　AjcMoveWindowToCenterOfMonitor(HWND hwnd);
BOOL　AjcMoveWindowToCenterOfWindow(HWND hwnd, hWndScr);

引　数 ： hwnd　　- 移動するウインドのハンドル
hWndScr - 他のウインドハンドル

説　明 ： AjcMoveWindowToCenter()は、ウインドを親ウインド／オーナーウインドの中央へ移動します。
親ウインド／オーナーウインドが無い場合は、AjcMoveWindowToCenterOfMonitor()と同じです。
AjcMoveWindowToCenterOfScreen()は、ウインドをデスクトップ全体の中央へ移動します。
AjcMoveWindowToCenterOfMonitor()は、自ウインドが属するモニタの中央へ移動します。
ウインドがいずれのモニタにも属さない場合は、プライマリウインドの中央へ移動します。
AjcMoveWindowToCenterOfWindow()は、ウインドを他のウインドの中央へ移動します。

いずれも、ウインドのサイズは変化しません。

戻り値 ： TRUE　- 成功
FALSE - 失敗

18.1.10. 現在フォーカスされているウインドを取得(AjcGetFocusWindow)

形　式 ： HWND　AjcGetFocusWindow(VO);

引　数 ： なし

説　明 ： 現在フォーカスされているウインドのハンドルを取得します。
他プロセスのフォーカスウインドでも取得可能です。

戻り値 ： ≠ NULL - 成功（フォーカスされているウインドのハンドル）
＝ NULL - 失敗

18.1.11. 指定ウインドをフォーカスする(AjcSetFocusWindow)

形　式 ： BOOL　AjcSetFocusWindow(HWND hwnd)；

引　数 ： hwnd -フォーカスするウインドのハンドル

説　明 ： hwnd で指定されたウインドをフォーカスします。
他プロセスのウインドでもフォーカス可能です。

戻り値 ： TRUE　- 成功
FALSE - 失敗

18.1.12. 通常表示時のウインド矩形情報取得(AjcGetWindowNormalRect)

形　式 ： int　AjcGetWindowNormalRect (HWND hwnd, LPRECT pRect)；

引　数 ： hwnd　- モジュールのインポート情報のアドレス（AjcCreateModImportInfo()の戻り値）
pRect - 矩形情報を格納するバッファのアドレス（不要時は NULL）

説　明 ： 指定したウインドの通常表示状態時の矩形情報を取得します。

戻り値 ： ≠-1：ウインドの状態(SW_HIDE/SW_MINIMIZE/SW_SHOW/SW_SHOWNORMAL・・・)
＝-1：エラー

18.1.13. スレッドの全トップレベルウインドへメッセージ送信 (AjcSendMessageToThreadWindows)

形　式 ： VO　AjcSendMessageToThreadWindows(UI msg, WPARAM wParam, LPARAM lParam)；

引　数 ： mag　　- メッセージコード
wParam, lParam　- メッセージパラメタ

説　明 ： 自スレッドの全トップレベルウインドへメッセージを送信(SendMessage)します。

戻り値 ： なし

18.1.14. スレッドの全トップレベルウインドと指定された２つのウインドへメッセージ送信 (AjcSendMessageToThreadWindowsEx)

形　式 ： VO　AjcSendMessageToThreadWindowsEx(UI msg, WPARAM wParam, LPARAM lParam, HWND hWndPre, HWND hWndPost)；

引　数 ： mag　　- メッセージコード
wParam, lParam　- メッセージパラメタ
hWndPre　　- 全スレッドウインドに先駆けてメセージを送信するウインドのハンドル（不要時は NULL）
hWndPost　　- 全スレッドウインドの後にメセージを送信するウインドのハンドル　　（不要時は NULL）

説　明 ： 自スレッドの全トップレベルウインドと、「hWndPre」「hWndPost」指定された２つのウインドへメッセージを送信(SendMessage)します。
メッセージの送信順序は、以下のとおりです。

１）hWndPre で指定されたウインドへメッセージを送信します。(hWndPre が NULL の場合は何もしない)
２）自スレッドの全トップレベルウインド（hWndPre, hWndPost で指定されたウインド以外）へメッセージを送信します。
３）hWndPost で指定されたウインドへメッセージを送信します。(hWndPost が NULL の場合は何もしない)

戻り値 ： なし

18.1.15. スレッドの全トップレベルウインドを許可／禁止(AjcEnableThreadWindows)

形　式　：　VO　AjcEnableThreadWindows(BOOL fEnable);

引　数　：　fEnable　　－ ウインドの許可／禁止フラグ（TRUE:許可, FALSE:禁止）

説　明　：　自スレッドの全トップレベルウインドを、許可（ユーザ入力を受け付ける）あるいは、禁止（ユーザ入力を受け付けない）ように設定します。

戻り値　：　なし

18.1.16. Windows イベント処理(AjcDoEvent)

形　式　：　UI　AjcDoEvent(VO);

引　数　：　なし

説　明　：　Windows のメッセージイベントを処理します。
ダイアログメッセージの処理は行いません。

戻り値　：　処理したメッセージの数

18.1.17. Windows イベント処理(AjcDoEvent)　・・　ダイアログメッセージ処理付き

形　式　：　UI　AjcDoEventEx(UI nDialog, ...);

引　数　：　nDialog － ダイアログハンドルの個数

説　明　：　Windows のメッセージイベントを処理します。
IsDialogMessage()により、ダイアログメッセージの処理も行います。
nDialog の後に、指定した個数のダイアログハンドルを指定します。

戻り値　：　処理したメッセージの数

18.1.18. ウインドハンドルから実行プログラムファイルのパス名を取得する (AjcGetExePathNam)

形　式　：　HINSTANCE　AjcGetExePathName (HWND hwnd, UTP pBuf, int lBuf);

引　数　：　hwnd － ウインドハンドル
pBuf － 実行プログラムのパス名を格納するバッファのアドレス（不要時は NULL）
lBuf － 実行プログラムのパス名を格納するバッファの文字数

説　明　：　ウインドハンドルから、当該ウインドの実行プログラム名を取得します。

戻り値　：　≠NULL －　成功（実行プログラムのインスタンスハンドル）
＝NULL －　失敗

18.1.19. デフォルトウインドプロシージャの実行 (AjcDefWindowProc)

形　式 ： LRESULT　AjcDefWindowProc (HWND hwnd, UI msg, WPARAM wParam, LPARAM lParam);

引　数 ： hwnd　　- ウインドハンドル
msg　　- メッセージコード
wParam, lParam – メッセージパラメタ

説　明 ： UNICODE ウインドの場合 DefWindowProcW()を、その他の場合 DefWindowProcA()を実行します。

戻り値 ： DefWindowProcW() / DefWindowProcA()の戻り値

18.1.20. ウインドプロシージャの呼び出し (AjcCallWindowProc)

形　式 ： LRESULT　AjcCallWindowProc (WNDPROC wndproc, HWND hwnd, UI msg, WPARAM wParam, LPARAM lParam);

引　数 ： wndproc　　- 呼び出すウインドプロシージャのアドレス
hwnd　　- ウインドハンドル
msg　　- メッセージコード
wParam, lParam – メッセージパラメタ

説　明 ： UNICODE ウインドの場合 CallWindowProcW()を、その他の場合 CallWindowProcA()を実行します。

戻り値 ： CallWindowProcW() / CallWindowProcA()の戻り値

18.1.21. ウインドのＬＯＮＧ情報設定 (AjcSetWindowLongPtr)

形　式 ： UX　AjcSetWindowLongPtr(HWND hwnd, UI ix, UX data);
UX　MAjcSetWindowLong　(HWND hwnd, UI ix, UX data); --- マクロ

引　数 ： hwnd　　- ウインドハンドル
ix　　- オフセット
data　　- 設定データ

説　明 ： UNICODE ウインドの場合 SetWindowLongPtrW()を、その他の場合 SetWindowLongPtrA()を実行します。
MAjcSetWindowLong()はマクロで、内部で AjcSetWindowLongPtr()を展開します。

戻り値 ： ウインド・ロングデータ値（「備考」に示す条件以外の場合は、SetWindowLongPtr()の戻り値）

備　考 ： ix = GWL_STYLE を指定しウインドのスタイルを取得する場合で、AjcSetDlgCtrlsColor()でダイアログとコントロールのカラー設定を行った場合、カラー設定したボタンコントロール（BS_PUSHBUTTON, BS_DEFPUSHBUTTON と、 BS_PUSHLIKE のチェックボックスやラジオボタン）のスタイル（下位 16Bit)は AjcSetSubclassedButtonStyle()により設定します。
実際のウインドスタイルは、上位 16 Bit のみ設定されます。
この場合、戻り値の下位 16Bit は、AjcSetSubclassedButtonStyle()の戻り値となります。

18.1.22. ウインドのＬＯＮＧ情報取得 (AjcGetWindowLongPtr)

形　式 ： UX　AjcGetWindowLongPtr(HWND hwnd, UI ix);
UX　MAjcGetWindowLong　(HWND hwnd, UI ix); --- マクロ

引　数 ： hwnd　　- ウインドハンドル
ix　　- オフセット

説　明 ： UNICODE ウインドの場合 GetWindowLongPtrW()を、その他の場合 GetWindowLongPtrA()を実行します。
MAjcGetWindowLong()はマクロで、内部で AjcGetWindowLongPtr()を展開します。

戻り値 ： ウインド・ロングデータの前回設定値（「備考」に示す条件以外の場合は、GetWindowLongPtr()の戻り値）

備　考 ： ix = GWL_STYLE を指定しウインドのスタイルを取得する場合で、AjcSetDlgCtrlsColor()でダイアログとコントロールのカラー設定を行った場合、カラー設定したボタンコントロール（BS_PUSHBUTTON, BS_DEFPUSHBUTTON と、 BS_PUSHLIKE のチェックボックスやラジオボタン）のスタイル(下位 16Bit)は AjcGetSubclassedButtonStyle()により取得します。

18.1.23. ウインドの背景色取得 (AjcGetWndBkColor)

形　式　：　COLORREF　AjcGetWndBkColor (HWND hwnd);

引　数　：　hwnd　　- ウインドハンドル

説　明　：　ウインドの背景色を取得します。
AjcDgcSetup()や、AjcDgcSetColors()でウインド／ダイアログボックスの背景色を設定している場合は、その背景色を返します。
その他の場合は、ウインドクラスから背景ブラシを取得し、そのブラシの色を返します。
ダイアログボックスの場合（ウインドクラスから背景ブラシを取得できない場合）は、背景色として(COLOR_BTNFACE)を返します。

戻り値　：　ウインドの背景色（取得できない場合は、GetSysColor(COLOR_BTNFACE)の値）

18.1.24. ウインドの背景色のブラシハンドル取得 (AjcGetWndBkBurush)

形　式　：　HBRUSH　AjcGetWndBkBurush (HWND hwnd);

引　数　：　hwnd　　- ウインドハンドル

説　明　：　ウインド背景色のブラシハンドルを取得します。
AjcDgcSetup()や、AjcDgcSetColors()でウインド／ダイアログボックスの背景色を設定している場合は、その背景色のブラシハンドルを返します。
その他の場合は、ウインドクラスから背景ブラシを取得し、そのブラシハンドルを返します。
ダイアログボックスの場合（ウインドクラスから背景ブラシを取得できない場合）は、COLOR_BTNFACE のブラシハンドル返します。

戻り値　：　ウインドの背景色のブラシハンドル（取得できない場合は、GetSysColor(COLOR_BTNFACE)のブラシハンドル）

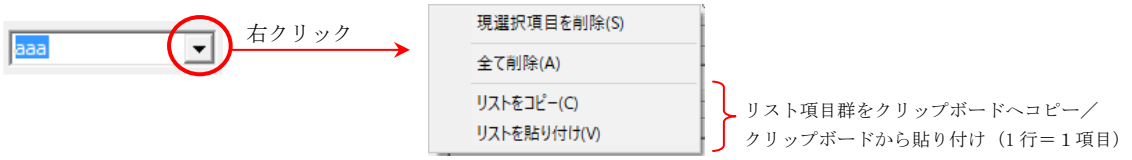
19. 拡張コンボボックス

システムのコンボボックス(Combobox クラス)をサブクラス化して機能を拡張します。
例えば文字列の検索で、検索した文字列をコンボボックスに追加登録して保存し、後で参照できるようにするような場合、個数の制限をしないと無限に項目が増えてしまいます。
拡張コンボボックスでは、このような検索文字列を登録するような場合、個数に制限を設けて、制限数を超える場合は最古の項目を削除し、最近登録した、いくつかの文字列だけを保持することができます。

コンボボックス内に既に存在する項目（文字列）と同じ項目を追加／挿入することはできません。
文字列の比較方法は、AjcSbcSetCompExact()で設定できます。
また、コンボボックスのリスト項目群を永続化（プロファイルへ記録）することができます。

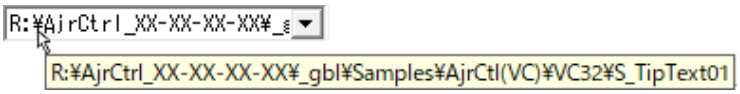
コンボボックス項目の削除

コンボボックスのドロップダウンリスト部分を右クリックするにより以下のポップアップメニューが表示されます。
下記ポップアップメニューにより 現選択項目の削除／全て削除 を行った場合はWM_COMMAND(HIWORD(wParam) = AJCCBN_REMOVED)により削除操作が行われたことを親ウインドへ通知します。



長テキストのツールチップ表示

選択中のテキストがコンボボックスに表示しきれていない場合、カーソルを置くと選択中のテキスト全体をツールチップで表示します。



この機能は、AjcSbcEnableLongTip()で許可／禁止を設定できます。

ファイル／フォルダのドロップ

コンボボックスが拡張スタイル「WS_EX_ACCEPTFILES」を持つ場合、エクスプローラからファイル／ディレクトリのドラッグ＆ドロップが可能となります。
ファイル／ディレクトリをドロップした場合は、コンボボックスへドロップしたファイルパス名／フォルダパス名を追加し、
WM_COMMAND(HIWORD(wParam) = AJCCBN_DROPPED)によりドロップ操作が行われたことを親ウインドへ通知します。

最大文字列長（最大文字数）

コンボボックス項目の最大文字数は５１１です。
文字列終端を含めた以下のシンボルが定義されています。

```
#define  AJCSBC_MAXSTL  512
```

19.1. サポートＡＰＩ

ウインドサポートＡＰＩ一覧を以下に示します。

#	関数名	内容
1	AjcSbcComboBox[Ex]	項目数の制限付きコンボボックス（コンボボックスのサブクラス化）
2	AjcSbcSetCompExact	コンボボックス項目のテキスト比較方法設定
3	AjcSbcLoadItems	プロファイルからコンボボックスのリスト項目群を読み出す
4	AjcSbcSaveItems	プロファイルへコンボボックスのリスト項目群を記録する
5	AjcSbcGetEditCtrlInComboBox	コンボボックス中のエディットコントロールのハンドルを取得する
6	AjcSbcSetMostNew	コンボボックス最新項目設定
7	AjcSbcTipCtrl	コンボボックスのチップテキスト表示制御
8	AjcSbcShowFirstLine	コンボボックスのチップテキストをファイルの１行目とする
9	AjcSbcEnableLongTip	長テキスト・ツールチップ表示の許可／禁止
10	AjcSbcGetRemovedItem	削除した項目の文字列取得
11	AjcSbcSetTextEncode	テキスト文字コード設定
12	AjcSbcGetTextEncode	テキスト文字コード取得

※ #2 以降のＡＰＩは、#1 の AjcSbcComboBox()でサブクラス化したコンボボックスでのみ使用可能です。

19.1.1. コンボボックスの拡張 (AjcSbcComboBox)

形　式 ： BOOL　AjcSbcComboBox　(HWND hwnd, UI MaxItem, UI LimitText);
BOOL　AjcSbcComboBoxEx(HWND hwnd, UI MaxItem, UI LimitText, UI flag);

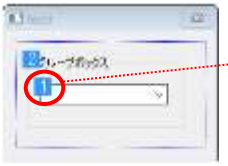
引　数 ： hwnd　　- コンボボックスのウインドハンドル
MaxItem　- コンボボックス項目の最大数
LimitText　- コンボボックスで入力可能な最大文字数（１～５１１（全角文字は２文字で計算））
flag　　- オプションフラグ（AJCSBCF_*XXXXX*）

説　明 ： コンボボックスをサブクラス化し、コンボボックス項目数の制限を行います。
CB_INSERTSTRING や CB_ADDSTRING メッセージでコンボボックスへ項目を追加する際、項目数が MaxItem で指定された個数を越える場合は、最古の項目が削除されます。
最古の項目とは、最も過去に CB_INSERTSTRING や CB_ADDSTRING メッセージを発行した項目を意味します。

flag は、以下のシンボルの合成値（無い場合は０）を指定します。

#	シンボル	内容	備考
1	AJCSBCF_IGNORWIDTH	英大小文字を区別しないで比較	デフォルトは、英大小文字を区別する
2	AJCSBCF_DROPFILE	ファイルのドロップを許可	ドロップしたファイル／フォルダをリストに登録し、当該項目を選択状態にする **（※２）**
3	AJCSBCF_DROPDIR	フォルダのドロップを許可	
4	AJCSBCF_DROPBOTH	ファイルとフォルダのドロップを許可	
5	AJCSBCF_DIRTAIL	ドロップしたフォルダの末尾に「¥」を付加	フォルダパス名の末尾が必ず「¥」となるようにする
6	AJCSBCF_TIPINSEL	リスト選択中のチップ制御	AjcSbcTipCtrl()参照
7	AJCSBCF_LONGTIP	長テキスト時のチップ表示	テキストが長い場合、テキスト全体をチップ表示する **（※１）**

※1：長テキストのチップ表示はデフォルトで有効ですが、AjcTipTextAdd[Ex]()でツールチップを設定した場合は無効となります。
※2：コンボボックスをグループボックスの中に配置する場合は、タブオーダーをグループボックスより若い番号に設定してください。



CB_INSERTSTRING や CB_ADDSTRING メッセージでコンボボックスへ項目を追加する際に、既に同一文字列の項目が存在する場合は、当該項目の追加は行われず、CB_ERR(=-1)を返します。
尚、各項目の文字列長は、LimitText で指定された長さに制限され、長い文字列を指定した場合は、末尾部分がカットされます。
本ＡＰＩでは、最初にコンボボックスの全項目をリセットします。
本ＡＰＩで、コンボボックスのサブクラス化を行った場合、コンボボックスに対する以下のメッセージは使用できません。

・CB_SETITEMDATA　（項目に関連付ける数値の設定）
・CB_DIR　　（コンボボックスにファイル名のリストを追加）

コンボボックスへ、これらのメッセージを発行した場合は、CB_ERR を返します。

コンボボックスのスタイルは、CBS_DROPDOWN か CBS_DROPDOWNLIST スタイルでなければなりません。
CBS_SIMPLE スタイルのコンボボックスでは使用できません。

戻り値 ： TRUE　- 成功
FALSE - 失敗

19.1.2. コンボボックス項目のテキスト比較方法設定 (AjcSbcSetCompExact)

形　式 ： BOOL AjcSbcSetCompExact(HWND hwnd, BOOL fExact);

引　数 ： hwnd　　- コンボボックスのウインドハンドル
fExact　　- テキスト比較方法 (TRUE:大文字と小文字を区別して比較 (デフォルト), FALSE:大文字／小文字を区別しないで比較)

説　明 ： コンボボックス項目のテキスト比較方法を設定します。
CB_ADDSTRING, CB_INSERTSTRING メッセージによる項目追加／挿入時の文字列の比較方法を設定します。

戻り値 ： TRUE - 成功
FALSE - 失敗

備　考 ： CB_ADDSTRING/ CB_INSERTSTRING メッセージで項目を追加する場合は、コンボボックス中に一致する文字列が無い場合のみ項目を追加します。(コンボボックス中に同一文字列がある場合は、CB_ERR を返します)
CB_FINDSTRING, CB_FINDSTRINGEXACT メッセージでも、テキスト比較方法の設定は有効です。

19.1.3. プロファイルからコンボボックスのリスト項目群を読み出す (AjcSbcLoadItems)

形　式 ： BOOL AjcSbcLoadItems(HWND hwnd, C_UTP pSect);

引　数 ： hwnd　　- コンボボックスのウインドハンドル
pSect　　- プロファイル・セクション名

説　明 ： AjcSbcComboBox()でサブクラス化したコンボボックスにおいて、プロファイルからリスト項目群を読み出して設定します。
pSect には、プロファイルセクション名を指定します。(セクション名中の「*」は、コントロールのＩＤ番号に変換します)

本関数を実行すると、最初にコンボボックスをリセット（全項目削除）します。
リセット後に、プロファイルから全項目を読み出してコンボボックスに設定します。

戻り値 ： TRUE - 成功
FALSE - 失敗

19.1.4. プロファイルへコンボボックスのリスト項目群を記録する (AjcSbcSaveItems)

形　式 ： BOOL AjcSbcSaveItems(HWND hwnd, C_UTP pSect);

引　数 ： hwnd　　- コンボボックスのウインドハンドル
pSect　　- プロファイル・セクション名

説　明 ： AjcSbcComboBox()でサブクラス化したコンボボックスにおいて、プロファイルへ、現在設定されている全リスト項目群を記録します。
pSect には、プロファイルセクション名を指定します。(セクション名中の「*」は、コントロールのＩＤ番号に変換します)
プロファイルキーの名称は以下のように設定されます。(nnnnnn は。コンボボックスのコントロールＩＤ)

- Cbo_nnnnnn_Cnt ---- コンボボックスのリスト項目数
- Cbo_nnnnnn_Idx ---- コンボボックス選択項目のインデクス
- Cbo_nnnnnn_xxxxxx - コンボボックスリスト項目（xxxxxx は、リスト項目の番号（０～））
- Cbo_nnnnnn_MaxLen - コンボボックスリスト項目の最大テキスト長（若干大きめの値）
- Cbo_nnnnnn_Txt ---- コンボボックス中のエディットテキスト（ドロップダウンリストの場合は無し）

戻り値 ： TRUE - 成功
FALSE - 失敗

19.1.5. コンボボックス下のエディットコントロールのハンドルを取得する (AjcSbcGetEditCtrlInComboBox)

形　式 ： HWND AjcSbcGetEditCtrlInComboBox (HWND hwnd);

引　数 ： hwnd　　- コンボボックスのウインドハンドル

説　明 ： コンボボックス中のエディット・コントロールのウインドハンドルを取得します。
コンボボックスの子ウインドからクラス名が「EDIT」であるウインドを検索し、当該ウインドのハンドルを返します。
「EDIT」クラスが見つからない場合は、NULL を返します。

戻り値 ： ≠NULL ： 成功（エディットコントロールのウインドハンドル）
＝NULL ： 失敗

備　考 ： 通常「CBS_DROPDOWN」スタイル等でテキストの編集ができる場合にコンボボックス下に「EDIT」クラスのウインドが存在し、「CBS_DROPDOWNLIST」等のテキストの編集ができない場合は「EDIT」クラスのウインドが存在しないようです。
このＡＰＩは、単に、子ウインドから最初に見つかった「EDIT」クラスのウインドハンドルを返します。

19.1.6. コンボボックス現表示項目を最新項目に設定(AjcSbcSetMostNew)

形　式 ： BOOL AjcSbcSetMostNew (HWND hwnd);

引　数 ： hwnd　　- コンボボックスのウインドハンドル

説　明 ： 現在表示されているテキストを、最新項目として設定します。
CBS_SORT スタイルが設定されていない場合は、当該項目をリストの先頭に移動します。
項目数がオーバーした場合、最古の項目を削除しますが、本ＡＰＩにより最新項目に設定することにより（現項目群の中では）最後に削除される項目となります。

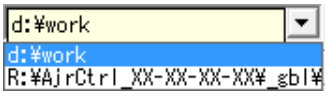
戻り値 ： TRUE　- 成功
FALSE - 失敗

19.1.7. コンボボックスのチップテキスト表示制御 (AjcSbcTipCtrl)

形　式 ： BOOL AjcSbcTipCtrl(HWND hwnd, BOOL fTipCtrl);

引　数 ： hwnd　　- コンボボックスのウインドハンドル
fTipCtrl　- ドロップダウンリストのツールチップ表示制御の指定
TRUE　： ドロップダウンリスト選択中に、選択項目をチップ表示する
FALSE ： ドロップダウンリスト選択中に、選択項目をチップ表示しない

説　明 ： fTipCtrl=TRUE とした場合、コンボボックスで選択中の項目を、コンボボックスのテキスト表示部分へチップテキストで表示します。
コンボボックスのテキスト表示部分が短い場合は、チップテキストが延長して表示されます。（下図参照）





戻り値 ： TRUE　- 成功
FALSE - 失敗

19.1.8. コンボボックスのチップテキストをファイルの１行目とする(AjcSbcShowFirstLine)

形　式 ： BOOL AjcSbcShowFirstLine (HWND hwnd, C_UTP pCharExclude);

引　数 ： hwnd　　- コンボボックスのウインドハンドル
pCharExclude - 除外する文字の配列（終端=0x00 の文字列で指定）

説　明 ： コンボボックスのチップテキストを表示する際、チップテキストとしてファイルの１行目を表示するようにします。
コンボボックスの設定項目をファイルパスとみなして、当該ファイルの１行目をチップテキストとして表示します。
「pCharExclude」は、チップテキストを作成する際に、ファイル１行目テキストの前後から除外する文字群を指定します。
(行末文字('¥n')は常に除外し、チップテキスト中のタブ(0x09)は、空白(0x20)に変換します)

ex. pCharExclude="/*　¥t"; ('/', '*', ' ', '¥t' の４文字と行末('¥n')を除外)
ファイルの１行目 "/*　This is Tip Text　*/¥n" ⇨ チップテキスト "This is Tip Text"

ファイルの１行目のテキスト（文字を除外したテキスト）が空行である場合は、２行目以降から同様にチップテキストを取得します。
但し、当該ファイルが見つからない場合や、ファイルが全て空行である場合は、コンボボックスの設定項目をチップテキストとします。

尚、「pCharExclude = NULL」とした場合は、ファイルの１行目をチップテキストとせずに、コンボボックスの設定項目をチップテキストとして表示ます

戻り値 ： TRUE - 成功
FALSE - 失敗

19.1.9. 長テキスト・ツールチップ表示の許可／禁止(AjcSbcEnableLongTip)

形　式 ： BOOL AjcSbcEnableLongTip (HWND hwnd, BOOL fEnable);

引　数 ： hwnd　　- コントロールのウインドハンドル
fEnable　- TRUE：許可（デフォルト), FALSE：禁止

説　明 ： テキストがコンボボックスのテキスト表示部に収まらない場合、テキスト全体をツールチップ表示する機能の許可／禁止を設定します。
ユーザが AjcTipTextAdd[Ex]()で独自のツールチップを設定した場合や、AjcTipTextRemove()でツールチップの関連付けを解除した場合、この機能は「禁止」に設定されます。

戻り値 ： TRUE - 成功
FALSE - 失敗

19.1.10. 削除された項目の文字列取得(AjcSbcGetRemovedItem)

形　式 ： BOOL AjcSbcGetRemovedItem (HWND hwnd, UTP pBuf, UI lBuf);

引　数 ： hwnd　　- コントロールのウインドハンドル
pBuf　　- 削除された項目の文字列格納するバッファのアドレス
lBuf　　- 削除された項目の文字列格納するバッファの文字数

説　明 ： 削除されたリストボックス項目の文字列を取得します
本関数は、通知メッセージ（AJCCBN_REMOVED）が通知された場合、削除した項目の文字列を取得するために実行します。
本ＡＰＩで全ての削除項目を取得済の場合は、FALSE を返します。

戻り値 ： TRUE - 削除した項目の文字列をバッファに格納した
FALSE - 削除した項目なし（削除した項目数を超えて実行された）

19.1.11. テキスト文字コード設定(AjcSbcSetTextEncode)

形　式 ： BOOL　AjcSbcSetTextEncode (HWND hwnd, EAJCTEC iTec, EAJCTEC oTec, BOOL fBOM);

引　数 ： hwnd　　- コントロールのウインドハンドル
iTec　　- 入力テキスト文字コード（－１指定時は未設定）
oTec　　- 出力テキスト文字コード（現状は未使用）
fBOM　　- BOM 出力フラグ　　（現状は未使用）

説　明 ： AjcSbcShowFirstLine()で、コンボボックスのチップテキストをファイルの１行目に設定した場合の、ファイルの１行目を読み出す際のテキスト文字コードを設定します。
iTec, oTec には以下の値を指定します。
・AJCTEC_MBC　　- マルチバイト(S-JIS)・・・デフォルト値
・AJCTEC_UTF_8　　- ＵＴＦ－８
・AJCTEC_EUC_J　　- 日本語ＥＵＣ
・AJCTEC_UTF_16LE　- ＵＴＦ－１６（リトルエンディアン）
・AJCTEC_UTF_16BE　- ＵＴＦ－１６（ビッグエンディアン）

戻り値 ： TRUE　- 成功
FALSE - 失敗

19.1.12. テキスト文字コード取得(AjcSbcGetTextEncode)

形　式 ： BOOL　AjcSbcGetTextEncode(HWND hwnd, PEAJCTEC piTec, PEAJCTEC poTec, BOOL *pfBOM)

引　数 ： hwnd　　- コントロールのウインドハンドル
iTec　　- 入力テキスト文字コードを格納するバッファのアドレス（不要時は NULL）
oTec　　- 出力テキスト文字コードを格納するバッファのアドレス（不要時は NULL）
fBOM　　- BOM 出力フラグを格納するバッファのアドレス　　（不要時は NULL）

説　明 ： AjcSbcShowFirstLine()で、コンボボックスのチップテキストをファイルの１行目に設定した場合の、ファイルの１行目を読み出す際のテキスト文字コードを取得します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

19.2. 通知メッセージ

Windows 標準のコンボボックスメッセージ（CBN_*XXXXX*）に加えて、以下のメッセージが通知されます。
コントロールからの通知メッセージは、WM_COMMAND メッセージにより通知されます。
WM_COMMAND メッセージの wParam には以下の情報が設定されます。

・LOWORD(wParam) - コントロールの識別ＩＤ
・HIWORD(wParam) - 通知メッセージコード

通知メッセージコード（HIWORD(wParam)）の一覧を以下に示します。

#	メッセージコード	内容	備考
1	AJCCBN_REMOVED	項目削除通知	
2	AJCCBN_DROPPED	ファイル／フォルダドロップ通知	

19.2.1. 項目削除通知（AJCCBN_REMOVED）

説　明 ： 右クリック操作により、コンボボックスの項目が削除されたことを通知します。
AjcSbcGetRemovedItem()により、削除された項目の文字列を取得できます。

パラメタ： wParam - コントロールの識別ＩＤと通知メッセージコード（AJCCBN_REMOVED）
lParam – 削除した項目数

戻り値 ： なし（ゼロを返してください）

19.2.2. ファイル／フォルダ ドロップ通知（AJCCBN_DROPPED）

説　明 ： ファイル／フォルダがドロップされたことを通知します。

パラメタ： wParam - コントロールの識別ＩＤと通知メッセージコード（AJCCBN_DROPPED）
lParam -ドロップしたファイル／フォルダの個数

戻り値 ： なし（ゼロを返してください）

19.2.3. サンプルプログラム

19.2.4. SW_ComboBox.c（拡張コンボボックスの操作サンプル）

次のプログラムは、コンボボックス項目の永続化と各ファンクションの実行を行います。
コンボボックスの最大項目数を１０個とし、各項目の最大文字数は６３とします。




```
 1 : //
 2 : //  SW_ComboBox.c
 3 : //
 4 : #include    <AjrCstXX.h>
 5 : #include    <tchar.h>
 6 : #include    "resource.h"
 7 :
 8 : #define     MYSECT      TEXT("SW_ComboBox")
 9 : #define     MAX_LEN     64
10 : //-------------------------------------------------------------------------------------------------------------//
11 : //  ワーク                                                                                                     //
12 : //-------------------------------------------------------------------------------------------------------------//
13 : HINSTANCE       hInst;                      //  ＤＬＬインスタンスハンドル
14 : HWND            hDlgMain;                   //  ダイアログボックスハンドル
15 : int             ixLast;                     //  コンボボックス直前の選択項目
16 :
17 : //-------------------------------------------------------------------------------------------------------------//
18 : //  内部サブ関数                                                                                               //
19 : //-------------------------------------------------------------------------------------------------------------//
20 : AJC_DLGPROC_DEF(Main);
21 :
22 : //=============================================================================================================//
23 : //                                                                                                             //
24 : //  Ｗｉｎ Ｍａｉｎ                                                                                            //
25 : //                                                                                                             //
26 : //=============================================================================================================//
27 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
28 : {
29 :     MSG     msg;
30 :
31 :     hInst = hInstance;
32 :
33 :     //----- プロファイルをレジストリとする場合は、以下の行のコメントを解除します -----//
34 :     // AjcSetProfileIsRegistory(TRUE);
35 :
36 :     //----- メイン・ダイアログオープン ---------------------------------------------//
37 :     hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_DLGMAIN), NULL, AJC_DLGPROC_NAME(Main));
38 :     ShowWindow(hDlgMain, SW_SHOW);
39 :
40 :     //----- メッセージループ -------------------------------------------------------//
```

```
41 :     while (GetMessage(&msg, NULL, 0, 0)) {
42 :         do {
43 :             if (IsDialogMessage(hDlgMain, &msg)) break;
44 :             TranslateMessage(&msg);
45 :             DispatchMessage (&msg);
46 :         } while (0);
47 :     }
48 :
49 :     return (int)msg.wParam ;
50 : }
51 : //=============================================================================================================//
52 : //                                                                                                             //
53 : //  ダイアログ・プロシージャ                                                                                   //
54 : //                                                                                                             //
55 : //=============================================================================================================//
56 : //----- ダイアログ初期化 ------------------------------------------------------------------------------------//
57 : AJC_DLGPROC(Main, WM_INITDIALOG     )
58 : {
59 :     //  コンボボックスをグループボックスの前面に移動（Ｄ＆Ｄ可能とするため、Ｚオーダー設定）
60 :     SetWindowPos( GetDlgItem(hDlg, IDC_GRP_CBO), GetDlgItem(hDlg, IDC_CBO_SORT  ), 0, 0, 0, 0, SWP_NOMOVE | SWP_NOSIZE);
61 :     SetWindowPos( GetDlgItem(hDlg, IDC_GRP_CBO), GetDlgItem(hDlg, IDC_CBO_NOSORT), 0, 0, 0, 0, SWP_NOMOVE | SWP_NOSIZE);
62 :     //  最大テキスト長設定
63 :     AjcSetDlgItemEdtLimit(hDlg, IDC_TXT_ADD   , MAX_LEN -1 );
64 :     AjcSetDlgItemEdtLimit(hDlg, IDC_TXT_INS   , MAX_LEN -1 );
65 :     AjcSetDlgItemEdtLimit(hDlg, IDC_TXT_FIND  , MAX_LEN -1 );
66 :     AjcSetDlgItemEdtLimit(hDlg, IDC_TXT_FINDEX, MAX_LEN -1 );
67 :     //  コンボボックスをサブクラス化
68 :     AjcSbcComboBox (GetDlgItem(hDlg, IDC_CBO_SORT  ), 10, MAX_LEN - 1);
69 :     AjcSbcTipCtrl  (GetDlgItem(hDlg, IDC_CBO_SORT  ), TRUE);
70 :     AjcSbcComboBox (GetDlgItem(hDlg, IDC_CBO_NOSORT), 10, MAX_LEN - 1);
71 :     AjcSbcTipCtrl  (GetDlgItem(hDlg, IDC_CBO_NOSORT), TRUE);
72 :     //  ダイアログ項目初期化
73 :     AjcSetDlgItemChk(hDlg, IDC_RBT_SORT       , TRUE);
74 :     AjcSetDlgItemChk(hDlg, IDC_RBT_DESTINGUISH, TRUE);
75 :     //  コンボボックスのリスト項目群を含めるように設定
76 :     AjcDlgItemSetPermAtt (hDlg, IDC_CBO_SORT  , AJCCTL_PSEL_INCLIST);
77 :     AjcDlgItemSetPermAtt (hDlg, IDC_CBO_NOSORT, AJCCTL_PSEL_INCLIST);
78 :     //  ダイアログ項目のロード
79 :     AjcLoadAllControlSettings (hDlg, MYSECT, AJCOPT2(AJCCTL_SELACT_, ALL, NTCRBT));
80 :
81 :     return TRUE;
82 : }
83 : //----- ウインド破棄 ----------------------------------------------------------------------------------------//
84 : AJC_DLGPROC(Main, WM_DESTROY        )
85 : {
86 :     //  ダイアログ項目のセーブ
87 :     AjcSaveAllControlSettings(hDlg);
88 :     //  プログラム終了
89 :     PostQuitMessage(0);
90 :
91 :     return TRUE;
92 : }
93 : //----- コンボボックスからの通知(SORT) ----------------------------------------------------------------------//
94 : AJC_DLGPROC(Main, IDC_CBO_SORT          )
95 : {
96 :     switch (HIWORD(wParam)) {
97 :         case CBN_SELENDOK:                                      //  ●項目選択
98 :             AjcSbcSetMostNew(GetDlgItem(hDlg, IDC_CBO_SORT));   //      選択された項目を最新項目とする
99 :             break;
100 :         case AJCCBN_REMOVED:                                    //  ●削除通知
101 :         {   UT      str[AJCSBC_MAXSTL];
102 :             UT      txt[16384];
103 :             MAjcStrCpy(txt, AJCTSIZE(txt), TEXT("-- Removed --¥n"));
104 :             while (AjcSbcGetRemovedItem(GetDlgItem(hDlg, IDC_CBO_SORT), str, AJCTSIZE(str))) {
105 :                 MAjcStrCat(txt, AJCTSIZE(txt), str);
106 :                 MAjcStrCat(txt, AJCTSIZE(txt), TEXT("¥n"));
107 :             }
108 :             MessageBox(hDlg, txt, TEXT("Removed Items in Sort"), MB_OK);
109 :             break;
110 :         }
111 :     }
112 :     return TRUE;
113 : }
114 : //----- コンボボックスからの通知(NOSORT) --------------------------------------------------------------------//
115 : AJC_DLGPROC(Main, IDC_CBO_NOSORT        )
116 : {
117 :     switch (HIWORD(wParam)) {
118 :         case CBN_SELENDOK:                                      //  ●項目選択
119 :             AjcSbcSetMostNew(GetDlgItem(hDlg, IDC_CBO_NOSORT)); //      選択された項目を最新項目とする
120 :             break;
121 :         case AJCCBN_REMOVED:                                    //  ●削除通知
122 :         {   UT      str[AJCSBC_MAXSTL];
123 :             UT      txt[16384];
124 :             MAjcStrCpy(txt, AJCTSIZE(txt), TEXT("-- Removed --¥n"));
125 :             while (AjcSbcGetRemovedItem(GetDlgItem(hDlg, IDC_CBO_NOSORT), str, sizeof str)) {
126 :                 MAjcStrCat(txt, AJCTSIZE(txt), str);
127 :                 MAjcStrCat(txt, AJCTSIZE(txt), TEXT("¥n"));
128 :             }
129 :             MessageBox(hDlg, txt, TEXT("Removed Items in NoSort"), MB_OK);
130 :             break;
```

```
    131 :         }
    132 :     }
    133 :     return TRUE;
    134 : }
    135 : //----- コンボボックス（ソートあり）を選択 ----------------------------------------------------------------------//
    136 : AJC_DLGPROC(Main, IDC_RBT_SORT      )
    137 : {
    138 :     if (HIWORD(wParam) == BN_CLICKED) {
    139 :         AjcEnableDlgItem(hDlg, IDC_CBO_SORT  , TRUE);
    140 :         AjcEnableDlgItem(hDlg, IDC_CBO_NOSORT, FALSE);
    141 :     }
    142 :     return TRUE;
    143 : }
    144 : //----- コンボボックス（ソートなし）を選択 ----------------------------------------------------------------------//
    145 : AJC_DLGPROC(Main, IDC_RBT_NOSORT    )
    146 : {
    147 :     if (HIWORD(wParam) == BN_CLICKED) {
    148 :         AjcEnableDlgItem(hDlg, IDC_CBO_SORT  , FALSE);
    149 :         AjcEnableDlgItem(hDlg, IDC_CBO_NOSORT, TRUE);
    150 :     }
    151 :     return TRUE;
    152 : }
    153 : //----- 項目追加 ------------------------------------------------------------------------------------------//
    154 : AJC_DLGPROC(Main, IDC_CMD_ADD       )
    155 : {
    156 :     UT      txt[MAX_LEN];
    157 :
    158 :     if (HIWORD(wParam) == BN_CLICKED) {
    159 :         AjcGetDlgItemStr(hDlg, IDC_TXT_ADD, txt, 256);
    160 :         if (AjcGetDlgItemChk(hDlg, IDC_RBT_SORT)) SendDlgItemMessage(hDlg, IDC_CBO_SORT  , CB_ADDSTRING, 0, (LPARAM)txt);
    161 :         else                                      SendDlgItemMessage(hDlg, IDC_CBO_NOSORT, CB_ADDSTRING, 0, (LPARAM)txt);
    162 :     }
    163 :     return TRUE;
    164 : }
    165 : //----- 項目挿入 ------------------------------------------------------------------------------------------//
    166 : AJC_DLGPROC(Main, IDC_CMD_INS       )
    167 : {
    168 :     UT      txt[MAX_LEN];
    169 :     int     ix;
    170 :
    171 :     if (HIWORD(wParam) == BN_CLICKED) {
    172 :         AjcGetDlgItemStr(hDlg, IDC_TXT_INS, txt, 256);
    173 :         ix = AjcGetDlgItemSInt(hDlg, IDC_TXT_INS_IX);
    174 :         if (AjcGetDlgItemChk(hDlg, IDC_RBT_SORT)) SendDlgItemMessage(hDlg, IDC_CBO_SORT  , CB_INSERTSTRING, ix, (LPARAM)txt);
    175 :         else                                      SendDlgItemMessage(hDlg, IDC_CBO_NOSORT, CB_INSERTSTRING, ix, (LPARAM)txt);
    176 :     }
    177 :     return TRUE;
    178 : }
    179 : //----- 項目削除 ------------------------------------------------------------------------------------------//
    180 : AJC_DLGPROC(Main, IDC_CMD_DEL       )
    181 : {
    182 :     int     ix;
    183 :
    184 :     if (HIWORD(wParam) == BN_CLICKED) {
    185 :         ix = AjcGetDlgItemSInt(hDlg, IDC_TXT_DEL_IX);
    186 :         if (AjcGetDlgItemChk(hDlg, IDC_RBT_SORT)) SendDlgItemMessage(hDlg, IDC_CBO_SORT  , CB_DELETESTRING, ix, 0);
    187 :         else                                      SendDlgItemMessage(hDlg, IDC_CBO_NOSORT, CB_DELETESTRING, ix, 0);
    188 :     }
    189 :     return TRUE;
    190 : }
    191 : //----- 全項目削除 ----------------------------------------------------------------------------------------//
    192 : AJC_DLGPROC(Main, IDC_CMD_RESET     )
    193 : {
    194 :     if (HIWORD(wParam) == BN_CLICKED) {
    195 :         if (AjcGetDlgItemChk(hDlg, IDC_RBT_SORT)) SendDlgItemMessage(hDlg, IDC_CBO_SORT  , CB_RESETCONTENT, 0, 0);
    196 :         else                                      SendDlgItemMessage(hDlg, IDC_CBO_NOSORT, CB_RESETCONTENT, 0, 0);
    197 :     }
    198 :     return TRUE;
    199 : }
    200 : //----- 前部検索 ------------------------------------------------------------------------------------------//
    201 : AJC_DLGPROC(Main, IDC_CMD_FIND      )
    202 : {
    203 :     UT      txt[MAX_LEN];
    204 :     int     ix;
    205 :
    206 :     if (HIWORD(wParam) == BN_CLICKED) {
    207 :         ix = AjcGetDlgItemSInt(hDlg, IDC_TXT_FIND_IX);
    208 :         AjcGetDlgItemStr(hDlg, IDC_TXT_FIND, txt, sizeof txt);
    209 :         if (AjcGetDlgItemChk(hDlg, IDC_RBT_SORT)) ix = (int)SendDlgItemMessage(hDlg, IDC_CBO_SORT  , CB_FINDSTRING, ix,
(LPARAM)txt);
    210 :         else                                      ix = (int)SendDlgItemMessage(hDlg, IDC_CBO_NOSORT, CB_FINDSTRING, ix,
(LPARAM)txt);
    211 :         AjcSetDlgItemSInt(hDlg, IDC_TXT_FIND_IX2, ix);
    212 :     }
    213 :     return TRUE;
    214 : }
    215 : //----- 全体検索 ------------------------------------------------------------------------------------------//
    216 : AJC_DLGPROC(Main, IDC_CMD_FINDEX    )
    217 : {
    218 :     UT      txt[MAX_LEN];
```

```
    int     ix;

    if (HIWORD(wParam) == BN_CLICKED) {
        ix = AjcGetDlgItemSInt(hDlg, IDC_TXT_FINDEX_IX);
        AjcGetDlgItemStr(hDlg, IDC_TXT_FINDEX, txt, sizeof txt);
        if (AjcGetDlgItemChk(hDlg, IDC_RBT_SORT)) ix = (int)SendDlgItemMessage(hDlg, IDC_CBO_SORT  , CB_FINDSTRINGEXACT, ix,
(LPARAM)txt);
        else                                      ix = (int)SendDlgItemMessage(hDlg, IDC_CBO_NOSORT, CB_FINDSTRINGEXACT, ix,
(LPARAM)txt);
        AjcSetDlgItemSInt(hDlg, IDC_TXT_FINDEX_IX2, ix);
    }
    return TRUE;
}
//----- 項目読み出し ----------------------------------------------------------------------------------------------//
AJC_DLGPROC(Main, IDC_CMD_GETTEXT   )
{
    UT      txt[MAX_LEN] = TEXT("NotRead");
    int     ix;

    if (HIWORD(wParam) == BN_CLICKED) {
        ix = AjcGetDlgItemSInt(hDlg, IDC_TXT_GETTEXT_IX);
        if (AjcGetDlgItemChk(hDlg, IDC_RBT_SORT)) SendDlgItemMessage(hDlg, IDC_CBO_SORT  , CB_GETLBTEXT, ix, (LPARAM)txt);
        else                                      SendDlgItemMessage(hDlg, IDC_CBO_NOSORT, CB_GETLBTEXT, ix, (LPARAM)txt);
        AjcSetDlgItemStr(hDlg, IDC_TXT_GETTEXT, txt);
    }
    return TRUE;
}
//----- 文字大小区別あり ------------------------------------------------------------------------------------------//
AJC_DLGPROC(Main, IDC_RBT_DESTINGUISH)
{
    if (HIWORD(wParam) == BN_CLICKED) {
        AjcSbcSetCompExact(GetDlgItem(hDlg, IDC_CBO_SORT  ), TRUE);
        AjcSbcSetCompExact(GetDlgItem(hDlg, IDC_CBO_NOSORT), TRUE);
    }
    return TRUE;
}
//----- 文字大小区別なし ------------------------------------------------------------------------------------------//
AJC_DLGPROC(Main, IDC_RBT_NODESTINGUISH)
{
    if (HIWORD(wParam) == BN_CLICKED) {
        AjcSbcSetCompExact(GetDlgItem(hDlg, IDC_CBO_SORT  ), FALSE);
        AjcSbcSetCompExact(GetDlgItem(hDlg, IDC_CBO_NOSORT), FALSE);
    }
    return TRUE;
}
//----- キャンセル ------------------------------------------------------------------------------------------------//
AJC_DLGPROC(Main, IDCANCEL         )
{
    DestroyWindow(hDlg);
    return TRUE;
}
//-----------------------------------------------------------------------------------------------------------------//
AJC_DLGMAP_DEF(Main)
    AJC_DLGMAP_MSG(Main, WM_INITDIALOG          )
    AJC_DLGMAP_MSG(Main, WM_DESTROY             )

    AJC_DLGMAP_CMD(Main, IDC_CBO_SORT           )
    AJC_DLGMAP_CMD(Main, IDC_CBO_NOSORT         )
    AJC_DLGMAP_CMD(Main, IDC_RBT_SORT           )
    AJC_DLGMAP_CMD(Main, IDC_RBT_NOSORT         )
    AJC_DLGMAP_CMD(Main, IDC_CMD_ADD            )
    AJC_DLGMAP_CMD(Main, IDC_CMD_INS            )
    AJC_DLGMAP_CMD(Main, IDC_CMD_DEL            )
    AJC_DLGMAP_CMD(Main, IDC_CMD_RESET          )
    AJC_DLGMAP_CMD(Main, IDC_CMD_FIND           )
    AJC_DLGMAP_CMD(Main, IDC_CMD_FINDEX         )
    AJC_DLGMAP_CMD(Main, IDC_CMD_GETTEXT        )
    AJC_DLGMAP_CMD(Main, IDC_RBT_DESTINGUISH    )
    AJC_DLGMAP_CMD(Main, IDC_RBT_NODESTINGUISH  )
    AJC_DLGMAP_CMD(Main, IDCANCEL               )
AJC_DLGMAP_END


```

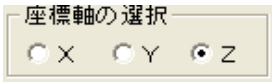
20. ラジオボタンのグループ操作

グループボックス内の全ラジオボタンを一括操作するＡＰＩの集合です。

ラジオボタンを含むグループボックスへアクセスすることにより、ラジオボタンを操作します。
（グループボックスへ数値を設定することによりラジオボタンが選択され、グループボックスから数値を読み出すことにより選択されているラジオボタンを取得します）

ex. AjcSbcSetRbt(GetDlgItem(hDlg, IDC_GRPBOX), 2);

⇩



グループボックス内の（0 オリジンで）2 番目のラジオボタンをチェック状態にする

※ラジオボタンのグルーピング挙動（１つのボタンを ON にすると他は全て OFF になるグループ）は、本ラジオボタンのグループ操作化とは関係なく、従来通りタブオーダーと WS_GROUP スタイルにより決まります。

20.1. サポートＡＰＩ

ラジオボタンのグループ操作のサポートＡＰＩ一覧を以下に示します。

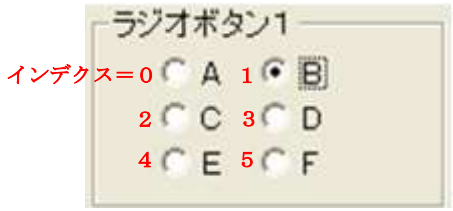
#	関数名	内容
1	AjcSbcRadioBtns	ラジオボタンのグループ操作化（グループボックスのサブクラス化）
2	AjcSbcGetRbt	グループ操作化したラジオボタンの状態取得
3	AjcSbcSetRbt	グループ操作化したラジオボタンの状態設定
4	AjcSbcGetRbtHandle	ラジオボタンのハンドル取得

20.1.1. ラジオボタンのグループ操作化 (AjcSbcRadioBtns)

形　式　：　UI　AjcSbcRadioBtns(HWND hGrpBox);

引　数　：　hGrpBox　　－ ラジオボタンを囲んでいるグループボックスのウインドハンドル

説　明　：　グループボックスのハンドルとラジオボタンのインデクス番号で、グループボックス内のラジオボタンを操作できるようにします。
　グループボックス内のラジオボタンに、左から右の順でインデクス番号を付与し、当該インデクス番号でラジオボタンの状態を取得したり、状態を設定することができます。
　グループボックス内にラジオボタンが複数行ある場合は、上から下の順でインデクス番号を付与します。
　例えば、以下のようなグループボックス内のラジオボタンでは、インデクス番号を０～５の順で付与します。



　グループボックス内のラジオボタンをクリックした場合、親ウインドに WM_COMMAND メッセージにより BN_CLICKED が通知されますが、この BN_CLICKED メッセージのパラメタは、以下のように変更されます。

パラメタ	内　容	備考（変更前の内容）
LOWORD(wParam)	ラジオボタンが属するグループボックスのＩＤ	ラジオボタンのＩＤ
HIWORD(wParam)	AJCRBTN_SELECT	BN_CLICKED
lParam	選択されたラジオボタンのインデクス番号（０～） ※ MFC 使用時はグループボックスのハンドル	ラジオボタンのハンドル

　１つのグループボックスには、最大１２８個のラジオボタンを配置できます。

戻り値　：　≠0 － 成功（グループ化したラジオボタンの個数）
　　＝0 － 失敗

備　考　：　グループボックス内・ラジオボタンの取得／設定は、以降に示す「AjcSbcGetRbt()」「AjcSbcGetRbt()」で行います。
または、以下のＡＰＩにおいても、グループボックス内・ラジオボタンの取得／設定を行うことができます。

ダイアログボックス項目／ウインド項目の取得	AjcGetDlgItemUInt() / AjcGetCtrlItemUInt() AjcGetDlgItemSInt() / AjcGetCtrlItemSInt() AjcGetDlgItemHex () / AjcGetCtrlItemHex () AjcGetDlgItemUI64() / AjcGetCtrlItemUI64() AjcGetDlgItemSI64() / AjcGetCtrlItemSI64() AjcGetDlgItemH64() / AjcGetCtrlItemH64()
ダイアログボックス項目／ウインド項目の設定	AjcSetDlgItemUInt() / AjcSetCtrlItemUInt() AjcSetDlgItemSInt() / AjcSetCtrlItemSInt() AjcSetDlgItemHex () / AjcSetCtrlItemHex () AjcSetDlgItemUI64() / AjcSetCtrlItemUI64() AjcSetDlgItemSI64() / AjcSetCtrlItemSI64() AjcSetDlgItemH64() / AjcSetCtrlItemH64()
ダイアログボックス項目／ウインド項目の 関連付け （digits, fComma 引数は、無視されます）	AjcSetDlgPropUInt() / AjcSetCtrlPropUInt() AjcSetDlgPropSInt() / AjcSetCtrlPropSInt() AjcSetDlgPropHex () / AjcSetCtrlPropHex () AjcSetDlgPropUI8() / AjcSetCtrlPropUI8() AjcSetDlgPropSI8() / AjcSetCtrlPropSI8() AjcSetDlgPropH8() / AjcSetCtrlPropH8() AjcSetDlgPropUI16() / AjcSetCtrlPropUI16() AjcSetDlgPropSI16() / AjcSetCtrlPropSI16() AjcSetDlgPropH16() / AjcSetCtrlPropH16() AjcSetDlgPropUI64() / AjcSetCtrlPropUI64() AjcSetDlgPropSI64() / AjcSetCtrlPropSI64() AjcSetDlgPropH64() / AjcSetCtrlPropH64()

20.1.2. グループ操作化したラジオボタンの状態取得 (AjcSbcGetRbt)

形　式　：　UI　AjcSbcGetRbt(HWND hGrpBox);

引　数　：　hGrpBox　　- ラジオボタンを囲んでいるグループボックスのウインドハンドル

説　明　：　AjcSbcRadioBtns()でグループ操作化したラジオボタンの状態を取得します。
グループボックス内のラジオボタンで、チェックされているラジオボタンのインデクス番号を返します。
複数チェックされている場合は、最も若いインデクス番号を返します。

戻り値　：　≠-1: 成功（チェックされているラジオボタンのインデクス番号（０～））
＝-1: 失敗

20.1.3. グループ操作化したラジオボタンの状態設定 (AjcSbcSetRbt)

形　式　：　BOOL　AjcSbcSetRbt(HWND hGrpBox, UI ix);

引　数　：　hGrpBox　　- ラジオボタンを囲んでいるグループボックスのウインドハンドル
ix　　- チェック状態に設定するラジオボタンのインデクス番号（０～）

説　明　：　AjcSbcRadioBtns()でグループ操作化したラジオボタンの状態を設定します。
グループボックス内のラジオボタンにおいて、ix で指定されたインデクス番号のラジオボタンをチェック状態に設定し、その他のラジオボタンを未チェック状態に設定します。

戻り値　：　TRUE　- 成功
FALSE - 失敗

20.1.4. ラジオボタンのハンドル取得 (AjcSbcGetRbtHandle)

形　式 ： HWND AjcSbcGetRbtHandle (HWND hGrpBox, UI ix);

引　数 ： hGrpBox　　- ラジオボタンを囲んでいるグループボックスのウインドハンドル
ix　　- チェック状態に設定するラジオボタンのインデクス番号（０～）

説　明 ： AjcSbcRadioBtns()でグループ操作化したラジオボタンのウインドハンドルを取得します。

戻り値 ： ≠ NULL – ラジオボタンのハンドル
＝ NULL - 失敗

20.2. サンプルプログラム

20.2.1. SW_RadioBbutton（ラジオボタングループ化の操作サンプル）

次のプログラムは、ラジオボタンをグループ操作化し、グループボックスのハンドルとラジオボタンに付与されたインデクス番号でラジオボタンの状態を取得／設定します。

ラジオボタンの状態は、プロファイルに記録し、永続化します。



チェック状態のラジオボタンのインデクス番号を表示します

```
 1 : //
 2 : //  SW_RadioBbutton.c
 3 : //
 4 : #include    <AjrCstXX.h>
 5 : #include    <tchar.h>
 6 : #include    "resource.h"
 7 :
 8 : //-------------------------------------------------------------------------------------------------------------------//
 9 : //  ワーク                                                                                                           //
10 : //-------------------------------------------------------------------------------------------------------------------//
11 : HINSTANCE       hInst;                      //  ＤＬＬインスタンスハンドル
12 : HWND            hDlgMain;                   //  ダイアログボックスハンドル
13 : HWND            hGrp1;
14 : HWND            hGrp2;
15 :
16 : //-------------------------------------------------------------------------------------------------------------------//
17 : //  内部サブ関数                                                                                                     //
18 : //-------------------------------------------------------------------------------------------------------------------//
19 : AJC_DLGPROC_DEF(Main);
20 :
21 : //===================================================================================================================//
22 : //                                                                                                                   //
23 : //  Ｗ ｉ ｎ Ｍ ａ ｉ ｎ                                                                                             //
24 : //                                                                                                                   //
25 : //===================================================================================================================//
26 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
27 : {
28 :     MSG     msg;
29 :
30 :     hInst = hInstance;
31 :
32 :     //----- プロファイルをレジストリとする場合は、以下の行のコメントを解除します -----//
33 :     // AjcSetProfileIsRegistory(TRUE);
34 :
35 :     //----- メイン・ダイアログオープン ---------------------------------------------//
36 :     hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_DLGMAIN), NULL, AJC_DLGPROC_NAME(Main));
37 :     ShowWindow(hDlgMain, SW_SHOW);
38 :
39 :     //----- メッセージループ -------------------------------------------------------//
40 :     while (GetMessage(&msg, NULL, 0, 0)) {
41 :         do {
42 :             if (IsDialogMessage(hDlgMain, &msg)) break;
43 :             TranslateMessage(&msg);
44 :             DispatchMessage (&msg);
45 :         } while (0);
46 :     }
47 :
48 :     return (int)msg.wParam ;
49 : }
50 : //===================================================================================================================//
51 : //                                                                                                                   //
```

```
52 : //  ダイアログ・プロシージャ                                                                                    //
53 : //                                                                                                            //
54 : //============================================================================================================//
55 : //----- ダイアログ初期化 -----------------------------------------------------------------------------------//
56 : AJC_DLGPROC(Main, WM_INITDIALOG     )
57 : {
58 :     //----- グループボックスのハンドル設定 ----------------------//
59 :     hGrp1 = GetDlgItem(hDlg, IDC_GRP_1);
60 :     hGrp2 = GetDlgItem(hDlg, IDC_GRP_2);
61 :     //----- グループボックス内のラジオボタンを統合化 ------------//
62 :     AjcSbcRadioBtns(hGrp1);
63 :     AjcSbcRadioBtns(hGrp2);
64 :     //----- プロファイルからラジオボタン状態を読み出して設定 ----//
65 :     AjcSbcSetRbt(hGrp1, AjcGetProfileUInt(TEXT("SW_RadioBbutton"), TEXT("GRP1"), 0));
66 :     AjcSbcSetRbt(hGrp2, AjcGetProfileUInt(TEXT("SW_RadioBbutton"), TEXT("GRP2"), 0));
67 :     //----- 初期のラジオボタン選択状態表示 ----------------------//
68 :     AjcSetDlgItemUInt(hDlg, IDC_TXT_1, AjcSbcGetRbt(hGrp1));
69 :     AjcSetDlgItemUInt(hDlg, IDC_TXT_2, AjcSbcGetRbt(hGrp2));
70 :
71 :     return TRUE;
72 : }
73 : //----- ウインド破棄 ---------------------------------------------------------------------------------------//
74 : AJC_DLGPROC(Main, WM_DESTROY        )
75 : {
76 :     //----- プロファイルへラジオボタン状態を記録する -------------//
77 :     AjcPutProfileUInt(TEXT("SW_RadioBbutton"), TEXT("GRP1"), AjcSbcGetRbt(hGrp1));
78 :     AjcPutProfileUInt(TEXT("SW_RadioBbutton"), TEXT("GRP2"), AjcSbcGetRbt(hGrp2));
79 :     //----- プログラム終了 ---------------------------------------//
80 :     PostQuitMessage(0);
81 :
82 :     return TRUE;
83 : }
84 : //----- キャンセル -----------------------------------------------------------------------------------------//
85 : AJC_DLGPROC(Main, IDCANCEL         )
86 : {
87 :     DestroyWindow(hDlg);
88 :
89 :     return TRUE;
90 : }
91 : //----- グループボックス１ ---------------------------------------------------------------------------------//
92 : AJC_DLGPROC(Main, IDC_GRP_1        )
93 : {
94 :     if (HIWORD(wParam) == AJCRBTN_SELECT) {
95 :         AjcSetDlgItemUInt(hDlg, IDC_TXT_1, (UI)lParam);
96 :     }
97 :     return TRUE;
98 : }
99 : //----- グループボックス２ ---------------------------------------------------------------------------------//
100 : AJC_DLGPROC(Main, IDC_GRP_2        )
101 : {
102 :     if (HIWORD(wParam) == AJCRBTN_SELECT) {
103 :         AjcSetDlgItemUInt(hDlg, IDC_TXT_2, (UI)lParam);
104 :     }
105 :     return TRUE;
106 : }
107 : //------------------------------------------------------------------------------------------------------------//
108 : AJC_DLGMAP_DEF(Main)
109 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG  )
110 :     AJC_DLGMAP_MSG(Main, WM_DESTROY     )
111 :     AJC_DLGMAP_CMD(Main, IDCANCEL       )
112 :     AJC_DLGMAP_CMD(Main, IDC_GRP_1      )
113 :     AJC_DLGMAP_CMD(Main, IDC_GRP_2      )
114 : AJC_DLGMAP_END
115 :
116 :
```

21. 他プロセス内のメモリ操作

他プロセス内にメモリを確保し、現プロセスから他プロセス内のメモリを読み書きします。
他プロセス内のメモリを使用して、他プロセスのウインドに（ポインタを伴った）メッセージを送ることができます。

※ 自プロセスと、他プロセスの種別（ビット数, x86(32 Bit) / x64(64 Bit)）は同じでなければなりません。

21.1. サポートＡＰＩ

ウインドサポートＡＰＩの一覧を以下に示します。

#	関数名	内容
1	AjcPmCreateByWnd AjcPmCreateByPid	他プロセスにメモリ確保（ウインド指定） 〃　　（プロセスＩＤ指定）
2	AjcPmDelete	他プロセスに確保したメモリを破棄
3	AjcPmWrite	他プロセスに確保したメモリへ書き込み
4	AjcPmRead	他プロセスに確保したメモリの読み出し
5	AjcPmGetAddr	他プロセスに確保したメモリ先頭からロケーション位置のアドレス取得
6	AjcPmSendMessage	他プロセスのウインドへメッセージ送信

21.1.1. 他プロセスにメモリ確保 (AjcPmCreateByWnd / AjcPmCreateByPid)

形　式 ： HAJCPM　AjcPmCreateByWnd(HWND hwnd, UX size);　　----- ウインド指定
HAJCPM　AjcPmCreateByPid(UL　　pid, UX size);　　----- プロセスＩＤ指定

引　数 ： hwnd　　- ウインドハンドル（メモリを確保する他プロセスのウインド）
pSec　　- プロセスＩＤ
size　　- メモリサイズ（バイト数）

説　明 ： 他のプロセス空間にメモリを確保します。

戻り値 ： ≠NULL - インスタンスハンドル
＝NULL - 失敗

21.1.2. 他プロセスにメモリ確保 したメモリを破棄(AjcPmDelete)

形　式 ： BOOL　AjcPmDelete (HAJCPM hPm);

引　数 ： hPm　　- インスタンスハンドル

説　明 ： 他のプロセス空間に確保したメモリを破棄します。

戻り値 ： TRUE　- 成功
FALSE – 失敗

21.1.3. 他プロセスに確保したメモリへ書き込み(AjcPmWrite)

形　式 ： BOOL　AjcPmWrite (HAJCPM hPm, UX loc, C_VOP pSrc, UX len);

引　数 ： hPm　　- インスタンスハンドル
loc　　- 他プロセスに確保したメモリの位置（バイトアドレス）
pSrc　　- 書き込むデータのアドレス
len　　- 書き込むデータのバイト数

説　明 ： 他のプロセス空間に確保したメモリにデータを書き込みます。

戻り値 ： TRUE　- 成功
FALSE – 失敗

21.1.4. 他プロセスに確保したメモリの読み出し(AjcPmRead)

形　式 ： BOOL　AjcPmRead (HAJCPM hPm, UX loc, VOP pBuf, UX len);

引　数 ： hPm　　- インスタンスハンドル
loc　　- 他プロセスに確保したメモリの位置（バイトアドレス）
pBuf　　- 読み出したデータを格納するバッファのアドレス
len　　- 読み出すデータのバイト数

説　明 ： 他のプロセス空間に確保したメモリからデータを読み出します。

戻り値 ： TRUE　- 成功
FALSE – 失敗

21.1.5. 他プロセスに確保したメモリ先頭からのロケーション位置のアドレス取得(AjcPmGetAddr)

形　式 ： UX　AjcPmGetAddr (HAJCPM hPm, UX loc);

引　数 ： hPm　　- インスタンスハンドル
loc　　- 他プロセスに確保したメモリの位置（バイトアドレス）

説　明 ： 他のプロセス空間に確保したメモリの先頭から「loc」で示されるロケーション一のアドレスを取得します。

戻り値 ： ≠０ - 他プロセス上のメモリアドレス
＝０ - 失敗

21.1.6. 他プロセスのウインドへメッセージ送信(AjcPmSendMessage)

形　式 ： LRESULT AjcPmSendMessage (HAJCPM hPm, HWND hwnd, UI msg, WPARAM wParam, LPARAM lParam);

引　数 ： hPm　　- インスタンスハンドル
hwnd　　- 送信先ウインドハンドル
msg　　- 送信メッセージコード
wParam　- メッセージパラメタ
lParam　- メッセージパラメタ

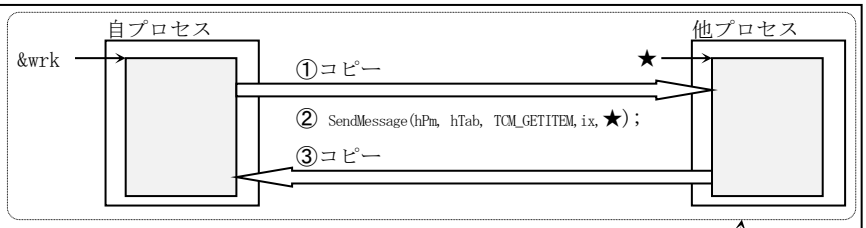
説　明 ： 他のプロセス内に確保したメモリへパラメタをセットアップし、他プロセスのウインドにメッセージを送信します。
本ＡＰＩを実行する前に、AjcPmCreate()で、他プロセス内に必要量のメモリを確保しておかなければなりません。
他の他のプロセス内に確保したメモリへパラメタをセットアップするには、wParam/lParam に以下のマクロを指定します。

マクロ名	内容	備考
AJCPM_WPARAM_W(HPM, PTR, LEN)	wParam で指定する文字列や構造体等のポインタを指定	HPM:インスタンスハンドル PTR:変数へのポインタ LEN:バイト数
AJCPM_WPARAM_R(HPM, PTR, LEN)	wParam で読み出しバッファのポインタを指定	
AJCPM_LPARAM_W(HPM, PTR, LEN)	lParam で指定する文字列や構造体等のポインタを指定	
AJCPM_LPARAM_R(HPM, PTR, LEN)	lParam で読み出しバッファのポインタを指定	

これらのマクロを指定した場合、「PTR」で指定したポインタから「LEN」で指定したバイト数を、他プロセスに確保したメモリの先頭にセットアップします。wParam/lParam の値は、他プロセス内でのメモリアドレスとなります。
また、末尾が「_R」のマクロを指定した場合は、SendMessage()でメッセージ送信後に、他プロセス内のメモリから「PTR」で指定したポインタへ「LEN」で指定したバイト数をコピーします。(つまり、SendMessage()の実行結果を自プロセスメモリへコピーします)

戻り値 ： メッセージの戻り値 (SendMessage()の戻り値)

備　考 ： 例えば、他プロセス内のタブコントロールからタブテキストを読み出す場合は、以下のように記述します。




```
typedef struct  {
    TC_ITEM tci;
    UT      txt[256];
} MYWORK, *PMYWORK;

    ・・・・・

    HAJCPM   hPm;
    MYWORK   wrk;
    int      ix = N;
    HWND     hTab = <他プロセス内タブ・コントロールのウインドハンドル>;
    hPm = AjcPmCreateByWnd(hWndProp, sizeof wrk);
    wrk.tci.mask       = TCIF_TEXT;
    wrk.tci.pszText    = (UTP)AjcPmGetAddr(hPm, sizeof wrk.tci);
    wrk.tci.cchTextMax = sizeof wrk.txt;
    // 他プロセス内・タブコントロールのタブテキストを wrk.txt に読み出す
    AjcPmSendMessage(hPm, hTab, TCM_GETITEM, ix, AJCPM_LPARAM_R(hPm, &wrk, sizeof wrk));
    AjcPmDelete(hPm);
```

また、他プロセス内のタブコントロールに新たなタブを追加する場合は、以下のように記述します。




```
typedef struct  {
    TC_ITEM tci;
    UT      txt[256];
} MYWORK, *PMYWORK;

    ・・・・・

    HAJCPM   hPm;
    MYWORK   wrk;
    int      ix = N;
    HWND     hTab = <他プロセス内タブ・コントロールのウインドハンドル>;
    hPm = AjcPmCreateByWnd(hWndProp, sizeof wrk);
    wrk.tci.mask       = TCIF_TEXT;
    wrk.tci.pszText    = (UTP)AjcPmGetAddr(hPm, sizeof wrk.tci);
    wrk.tci.cchTextMax = sizeof wrk.txt;
    MAjcStrCpy(wrk.txt, TEXT("NewTAB"));
    // 他プロセス内・タブコントロールへ新たなタブ項目「NewTAB」を追加する
    AjcPmSendMessage(hPm, hTab, TCM_INSERTITEM, ix, AJCPM_LPARAM_W(hPm, &wrk, sizeof wrk));
    AjcPmDelete(hPm);
```

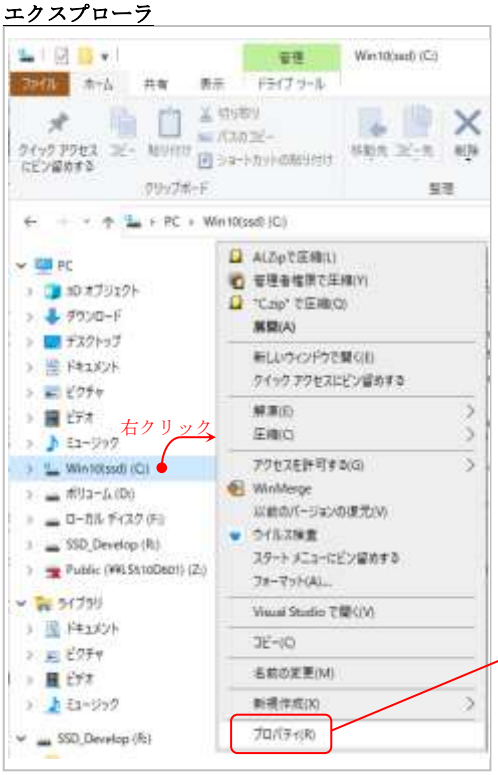
21.2. サンプルプログラム

21.2.1. SW_ProcessMem（エクスプローラプロセスからのメモリ読み出し）

このサンプルプログラムは、エクスプローラのドライブ・プロパティウインドから、タブテキストを読み出して表示します。
サンプルプログラムを実行する前に、エクスプローラを起動し、いずれかのドライブのプロパティウインドを表示しておいてください。

＜注＞　32 ビットシステムの場合は「SW_ProcessMem_32A」or「SW_ProcessMem_32W」を実行してください。
　　64 ビットシステムの場合は「SW_ProcessMem_64A」を実行してください。

エクスプローラ

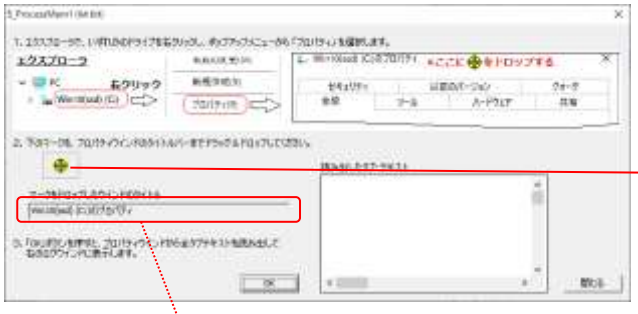


ドライブ・プロパティ ウインド



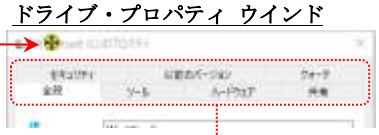
クリック

サンプルプログラムを実行すると以下のウインドが表示されます。
マーク「 」をドライブ・プロパティウインドのタイトルバーまでドラッグ＆ドロップします。

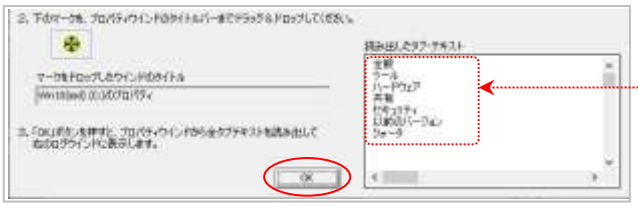


ドラッグ＆ドロップ

ドライブ・プロパティ ウインド



ＯＫボタンを押すと、ドライブ・プロパティウインドからタブテキストを読み出して、表示します。



```
 1 : //
 2 : //  SW_ProcessMem.c
 3 : //
 4 : #define _AJCSOCKSERV_H_
 5 : #include    <AjrCstXX.h>
 6 : #include    <tchar.h>
 7 : #include    "resource.h"
 8 :
 9 : #define WM_SCPEVENT     (WM_USER + 100)
10 : #define IDC_FINDER      5001
11 :
12 : typedef struct  {
13 :     TC_ITEM tci;
14 :     UT      txt[256];
15 : } MYWORK, *PMYWORK;
16 :
17 : //-----------------------------------------------------------------------------------------------------------------//
18 : //  ワーク                                                                                                         //
19 : //-----------------------------------------------------------------------------------------------------------------//
20 : HINSTANCE       hInst;                //  ＤＬＬインスタンスハンドル
21 : HWND            hDlgMain;             //  ダイアログボックスハンドル
22 : HWND            hWndDrag;             //  ドラッグ＆ドロップ ツールウインド
23 : HWND            hWndProp;             //  エクスプローラのドライブ・プロパティウインド
24 : HWND            hVth;                 //  ログ表示ウインド
25 : HBITMAP         hBmpFind;             //  ドラッグ＆ドロップツールのビットマップ
26 :
27 : BOOL            fCapture = FALSE;     //  マウスキャプチャ中フラグ
28 : POINT           ptSrt;                //  ドラッグ開始時のマウス位置
29 : RECT            rcSrt;                //  ドラッグ開始時のウインド位置
30 :
31 : //-----------------------------------------------------------------------------------------------------------------//
32 : //  内部サブ関数                                                                                                   //
33 : //-----------------------------------------------------------------------------------------------------------------//
34 : AJC_DLGPROC_DEF(Main    );
35 : AJC_WNDPROC_DEF(PicFind );
36 : AJC_WNDPROC_DEF(FindDrag);
37 : static  BOOL CALLBACK cbEnumChild(HWND hwnd, LPARAM lParam);
38 :
39 :
40 : //=================================================================================================================//
41 : //                                                                                                                 //
42 : //  W i n M a i n                                                                                                  //
43 : //                                                                                                                 //
44 : //=================================================================================================================//
45 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
46 : {
47 :     MSG         msg;
48 :     WNDCLASS    wndclass;
49 :
50 :     hInst = hInstance;
51 :
52 :     //----- 検索ウインドクラス生成 ---------------------//
53 :     wndclass.style          = 0;
54 :     wndclass.lpfnWndProc    = AJC_WNDPROC_NAME(FindDrag);
55 :     wndclass.cbClsExtra     = 0;
56 :     wndclass.cbWndExtra     = 0;
57 :     wndclass.hInstance      = hInst;
58 :     wndclass.hIcon          = NULL;
59 :     wndclass.hCursor        = LoadCursor(NULL, IDC_ARROW);
60 :     wndclass.hbrBackground  = GetStockObject(WHITE_BRUSH);
61 :     wndclass.lpszMenuName   = NULL;
62 :     wndclass.lpszClassName  = TEXT("S_ProcessMem1");
63 :     RegisterClass(&wndclass);
64 :
65 :     //----- メイン・ダイアログオープン -----------------//
66 :     hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_MAIN), NULL, AJC_DLGPROC_NAME(Main));
67 :     ShowWindow(hDlgMain, SW_SHOW);
68 :
69 :     //----- メッセージループ ---------------------------//
70 :     while (GetMessage(&msg, NULL, 0, 0)) {
71 :         do {
72 :             if (IsDialogMessage(hDlgMain, &msg)) break;
73 :             TranslateMessage(&msg);
74 :             DispatchMessage (&msg);
75 :         } while (0);
76 :     }
77 :
78 :     return (int)msg.wParam ;
79 : }
80 : //=================================================================================================================//
81 : //                                                                                                                 //
82 : //  ダイアログ・プロシージャ                                                                                       //
83 : //                                                                                                                 //
84 : //=================================================================================================================//
85 : //----- ダイアログ初期化 -----------------------------------------------------------------------------------------//
86 : AJC_DLGPROC(Main, WM_INITDIALOG )
87 : {
88 :     BOOL     w64 = FALSE;
89 :
90 :     hDlgMain = hDlg;
```

```
 91 :     hVth     = GetDlgItem(hDlg, IDC_VTH);
 92 :
 93 :     //  実行環境チェック（６４ビットＯＳ下の３２ビットプロセスの場合メッセージ表示）
 94 :     if  (IsWow64Process(GetCurrentProcess(), &w64) && w64) {
 95 :         AjcTipTextShowCenter(hDlg, TEXT("¥x1B[31m¥x1B[T")
 96 :                                    TEXT("このプログラムは、¥n")
 97 :                                    TEXT("64 ビット Windows では正常に実行されません。¥n")
 98 :                                    TEXT("SW_ProcessMem_64A を実行してください。")
 99 :                                    TEXT("¥x1B[0m"), 10000, NULL);
100 :     }
101 :
102 :     //  ウインドタイトル
103 :     #ifdef _WIN64
104 :         SetWindowText(hDlg, TEXT("S_ProcessMem1 (64 bit)"));
105 :     #else
106 :         SetWindowText(hDlg, TEXT("S_ProcessMem1 (32 bit)"));
107 :     #endif
108 :
109 :     //  ドラッグ＆ドロップ用ツールウインド生成（非表示）
110 :     hWndDrag = CreateWindowEx(WS_EX_TOPMOST | WS_EX_TOOLWINDOW,
111 :                               TEXT("S_ProcessMem1"),     // window class name
112 :                               TEXT(""),                  // window caption
113 :                               WS_POPUP,                  // window style
114 :                               0,                         // initial x position
115 :                               0,                         // initial y position
116 :                               19,                        // initial x size
117 :                               19,                        // initial y size
118 :                               NULL,                      // parent window handle
119 :                               NULL,                      // window menu handle
120 :                               hInst,                     // program instance handle
121 :                               NULL);                     // creation parameters
122 :     ShowWindow(hWndDrag, SW_HIDE);
123 :
124 :     //  ドラッグ＆ドロップ開始用ピクチャ・コントロールをサブクラス化
125 :     MAjcMmpSetSubclass(PicFind, GetDlgItem(hDlg, IDC_PIC_FIND));
126 :
127 :     return TRUE;
128 : }
129 : //----- ウインド破棄 ----------------------------------------------------------------------------------------//
130 : AJC_DLGPROC(Main, WM_DESTROY    )
131 : {
132 :     //  ドラッグ用ウインド破棄
133 :     DestroyWindow(hWndDrag);
134 :     //  プログラム終了
135 :     PostQuitMessage(0);
136 :
137 :     return TRUE;
138 : }
139 : //----- ＯＫボタン ------------------------------------------------------------------------------------------//
140 : AJC_DLGPROC(Main, IDOK)
141 : {
142 :     HWND    hTab;
143 :     UI      Count, i=0;
144 :     MYWORK  wrk;
145 :     HAJCPM  hPm;
146 :
147 :     if (IsWindow(hWndProp)) {
148 :         //  プロパティウインド内のタブコントロール検索
149 :         hTab = NULL;
150 :         EnumChildWindows(hWndProp, cbEnumChild, (LPARAM)&hTab);
151 :         //  タブコントロールあり・・・
152 :         if (IsWindow(hTab)) {
153 :             //  ログウインドクリアー
154 :             AjcVthClear(hVth);
155 :             //  エクスプローラ・プロセス内にメモリ確保
156 :             hPm = AjcPmCreateByWnd(hWndProp, sizeof wrk);
157 :             //  タブコントロールのタブ数取得
158 :             Count = (UI)SendMessage(hTab, TCM_GETITEMCOUNT, 0, 0);
159 :             //  タブコントロール内のタブテキストをログ表示
160 :             memset(&wrk, 0, sizeof wrk);
161 :             wrk.tci.mask = TCIF_TEXT;
162 :             wrk.tci.pszText = (UTP)AjcPmGetAddr(hPm, sizeof wrk.tci);
163 :             wrk.tci.cchTextMax = sizeof wrk.txt;
164 :             for (i = 0; i < Count; i++) {
165 :                 AjcPmSendMessage(hPm, hTab, TCM_GETITEM, i, AJCPM_LPARAM_R(hPm, &wrk, sizeof wrk));
166 :                 AjcVthPrintF(hVth, TEXT("  %s¥n"), wrk.txt);
167 :             }
168 :             //  エクスプローラ・プロセス内のメモリ解放
169 :             AjcPmDelete(hPm);
170 :         }
171 :         //  タブコントロールなし・・・
172 :         else {
173 :             AjcVthPrintF(hVth, TEXT("¥x1B[31m タブコントロールが見つかりません。¥x1B[0m¥n"), wrk.txt);
174 :         }
175 :     }
176 :     return TRUE;
177 : }
178 : //----- キャンセル ------------------------------------------------------------------------------------------//
179 : AJC_DLGPROC(Main, IDCANCEL      )
180 : {
```

```
181 :     //  メイン・ダイアログ破棄
182 :     DestroyWindow(hDlg);
183 :
184 :     return TRUE;
185 : }
186 : //-----------------------------------------------------------------------------------------------------------------//
187 : AJC_DLGMAP_DEF(Main)
188 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG  )
189 :     AJC_DLGMAP_MSG(Main, WM_DESTROY     )
190 :
191 :     AJC_DLGMAP_CMD(Main, IDOK           )
192 :     AJC_DLGMAP_CMD(Main, IDCANCEL       )
193 : AJC_DLGMAP_END
194 :
195 : //=================================================================================================================//
196 : //                                                                                                                 //
197 : //  ウインド検索用ピクチャ(IDC_PIC_FIND) サブクラス・ウインドプロシージャ                                          //
198 : //                                                                                                                 //
199 : //=================================================================================================================//
200 : //----- WM_LBUTTONDOWN ------------------------------------------------------------------------------------------//
201 : AJC_WNDPROC(PicFind, WM_LBUTTONDOWN     )
202 : {
203 :     RECT    r;
204 :     //  ドラッグ＆ドロップ開始用ピクチャ・コントロール非表示
205 :     ShowWindow(hwnd, SW_HIDE);
206 :     //  ドラッグ＆ドロップ用ツールウインド表示
207 :     GetWindowRect(GetDlgItem(hDlgMain, IDC_PIC_FIND), &r);
208 :     SetWindowPos(hWndDrag, NULL, r.left, r.top, 0, 0, SWP_NOSIZE);
209 :     ShowWindow(hWndDrag, SW_SHOW);
210 :     //  開始カーソル位置, ウインド位置退避
211 :     GetCursorPos(&ptSrt);
212 :     GetWindowRect(hWndDrag, &rcSrt);
213 :     //  マウスキャプチャ開始
214 :     SetCapture(hwnd);
215 :     fCapture = TRUE;
216 :
217 :     return 0;   //  インターセプト (オリジナルのウインドプロシージャは実行しない)
218 : }
219 : //----- WM_LBUTTONUP --------------------------------------------------------------------------------------------//
220 : AJC_WNDPROC(PicFind, WM_LBUTTONUP       )
221 : {
222 :     POINT   pt;
223 :     UT      txt[256];
224 :
225 :     //  マウスキャプチャ停止
226 :     ReleaseCapture();
227 :     fCapture = FALSE;
228 :     //  ドラッグ＆ドロップ開始用ピクチャ・コントロール表示
229 :     ShowWindow(hwnd, SW_SHOW);
230 :     //  ドラッグ＆ドロップ用ツールウインド非表示
231 :     ShowWindow(hWndDrag, SW_HIDE);
232 :     InvalidateRect(hDlgMain, NULL, TRUE);
233 :     //  ドラッグ位置のウインド・ハンドル取得
234 :     GetCursorPos(&pt);
235 :     hWndProp = WindowFromPoint(pt);
236 :     //  エクスプローラ・ドライブ・プロパティウインドのタイトル表示
237 :     if (IsWindow(hWndProp)) {
238 :         AjcVthClear(hVth);
239 :         GetWindowText(hWndProp, txt, AJCTSIZE(txt));
240 :         AjcSetDlgItemStr(hDlgMain, IDC_TXT_TITLE, txt);
241 :
242 :     }
243 :
244 :     return 0;   //  インターセプト (オリジナルのウインドプロシージャは実行しない)
245 : }
246 : //----- WM_MOUSEMOVE --------------------------------------------------------------------------------------------//
247 : AJC_WNDPROC(PicFind, WM_MOUSEMOVE       )
248 : {
249 :     POINT   pt;
250 :     int     x, y;
251 :
252 :     //  マウスキャプチャ中・・・
253 :     if (fCapture) {
254 :         //  ドラッグ＆ドロップ用ツールウインドを現カーソル位置に移動
255 :         GetCursorPos(&pt);
256 :         x = rcSrt.left + pt.x - ptSrt.x;
257 :         y = rcSrt.top  + pt.y - ptSrt.y;
258 :         SetWindowPos(hWndDrag, NULL, x, y, 0, 0, SWP_NOSIZE);
259 :     }
260 :     return 0;   //  インターセプト (オリジナルのウインドプロシージャは実行しない)
261 : }
262 : //-----------------------------------------------------------------------------------------------------------------//
263 : AJC_WNDMAP_DEF(PicFind)
264 :     AJC_WNDMAP_MSG(PicFind, WM_LBUTTONDOWN  )
265 :     AJC_WNDMAP_MSG(PicFind, WM_LBUTTONUP    )
266 :     AJC_WNDMAP_MSG(PicFind, WM_MOUSEMOVE    )
267 : AJC_WNDMAP_END
268 : //=================================================================================================================//
269 : //                                                                                                                 //
270 : //  ドラッグ＆ドロップツール ウインドプロシージャ                                                                  //
```

```
    271 : //                                                                                                                  //
    272 : //==================================================================================================================//
    273 : //----- WM_CREATE ----------------------------------------------------------------------------------------------------//
    274 : AJC_WNDPROC(FindDrag, WM_CREATE         )
    275 : {
    276 :     RECT    r;
    277 :
    278 :     AjcChangeBitmapColor(hBmpFind = LoadBitmap(hInst, MAKEINTRESOURCE(IDB_FIND)), RGB(255, 255, 255),
GetSysColor(COLOR_BTNFACE));
    279 :     GetWindowRect(GetDlgItem(hDlgMain, IDC_PIC_FIND), &r);
    280 :     MapWindowPoints(NULL, hDlgMain, (LPPOINT)&r, 2);
    281 :     SetWindowPos(hwnd, NULL, r.left, r.top, 0, 0, SWP_NOSIZE);
    282 :     return 0;
    283 : }
    284 : //----- WM_DESTROY ---------------------------------------------------------------------------------------------------//
    285 : AJC_WNDPROC(FindDrag, WM_DESTROY            )
    286 : {
    287 :     DeleteObject(hBmpFind);
    288 :     hWndDrag = NULL;
    289 :     return 0;
    290 : }
    291 : //----- WM_PAINT -----------------------------------------------------------------------------------------------------//
    292 : AJC_WNDPROC(FindDrag, WM_PAINT          )
    293 : {
    294 :     PAINTSTRUCT ps;
    295 :     HDC         hdc, mdc;
    296 :
    297 :     hdc  = BeginPaint(hwnd, &ps);
    298 :     mdc = CreateCompatibleDC(hdc);
    299 :     SelectObject(mdc, hBmpFind);
    300 :     BitBlt(hdc, 0, 0, 19, 19, mdc, 0, 0, SRCCOPY);
    301 :     DeleteDC(mdc);
    302 :     EndPaint(hwnd, &ps);
    303 :     return 0;
    304 : }
    305 : //------------------------------------------------------------------------------------------------------------------//
    306 : AJC_WNDMAP_DEF(FindDrag)
    307 :     AJC_WNDMAP_MSG(FindDrag, WM_CREATE      )
    308 :     AJC_WNDMAP_MSG(FindDrag, WM_DESTROY     )
    309 :     AJC_WNDMAP_MSG(FindDrag, WM_PAINT           )
    310 : AJC_WNDMAP_END
    311 :
    312 :
    313 : //------------------------------------------------------------------------------------------------------------------//
    314 : //  エクスプローラ・ドライブ・プロパティウインド内のタブコントロール（子孫ウインド）検索                            //
    315 : //------------------------------------------------------------------------------------------------------------------//
    316 : static  BOOL CALLBACK cbEnumChild(HWND hwnd, LPARAM lParam)
    317 : {
    318 :     BOOL    rc = TRUE;
    319 :     UT      name[256];
    320 :
    321 :     GetClassName(hwnd, name, 256);
    322 :
    323 :     if (MAjcStrICmp(name, TEXT("SysTabControl32")) == 0) {
    324 :         *((HWND*)lParam) = hwnd;
    325 :         rc = FALSE;
    326 :     }
    327 :     return rc;
    328 : }
    329 :
```

21.2.2. SW_ProcessMemC（エクスプローラプロセスからのメモリ読み出し）

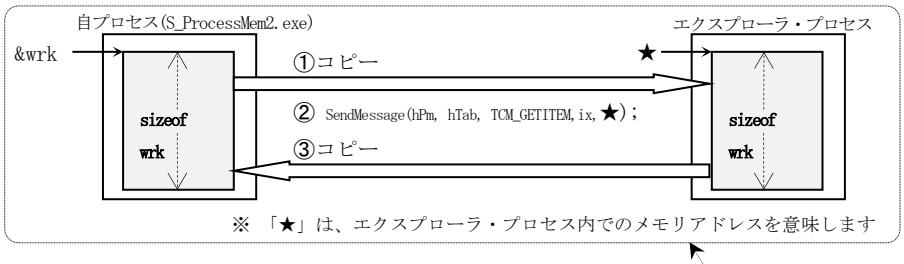
SW_ProcessMem と同様に、エクスプローラのドライブ・プロパティウインドから、タブテキストを読み出して表示します。
このサンプルプログラムでは、AjcPmSendMessage()は使用せずに、AjcPmWrite(), AjcPmGetAddr()と AjcPmRead()を使用しています。

タイトルテキストに文字列 「のプロパティ」 が含まれるウインドをドライブ・プロパティウインドとしています。

＜注＞ 32 ビットシステムの場合は「SW_ProcessMemC_32A」or「SW_ProcessMem_32W」を実行してください。
64 ビットシステムの場合は「SW_ProcessMemC_64A」を実行してください。



```
 1 : //
 2 : //  SW_ProcessMemC.C
 3 : //
 4 : #include    <AjrCstXX.h>
 5 : #include    <stdio.h>
 6 : #include    <conio.h>
 7 : #include    <tchar.h>
 8 :
 9 : static  BOOL CALLBACK cbEnumWindow(HWND hwnd, LPARAM lParam);
10 : static  BOOL CALLBACK cbEnumChild (HWND hwnd, LPARAM lParam);
11 :
12 : typedef struct  {
13 :     TC_ITEM tci;
14 :     UT      txt[256];
15 : } MYWORK, *PMYWORK;
16 :
17 : int  AjcMain(int argc, UTP argv[])
18 : {
19 :     HWND    hWndProp = NULL;
20 :     HWND    hTab;
21 :     UI      Count = 0;
22 :     UI      i;
23 :     MYWORK  wrk;
24 :     HAJCPM  hPm;
25 :     BOOL    w64;
26 :
27 :     AjcSetStdoutMode();
28 :
29 :     //  実行環境チェック （６４ビットＯＳ下の３２ビットプロセスの場合メッセージ表示）
30 :     if  (IsWow64Process(GetCurrentProcess(), &w64) && w64) {
31 :         AjcPrintF(TEXT("¥n64 ビット Windows では正常に実行されません。SW_ProcessMemC_64A を実行してください¥n"));
32 :     }
33 :
34 :     EnumWindows(cbEnumWindow, (LPARAM)&hWndProp);
35 :     if (IsWindow(hWndProp)) {
36 :         //  プロパティウインド内のタブコントロール検索
37 :         hTab = NULL;
38 :         EnumChildWindows(hWndProp, cbEnumChild, (LPARAM)&hTab);
39 :         //  タブコントロールあり・・・
40 :         if (IsWindow(hTab)) {
41 :             //  エクスプローラ・プロセス内にメモリ確保
42 :             hPm = AjcPmCreateByWnd(hWndProp, sizeof wrk);
43 :             //  タブコントロールのタブ数取得
44 :             Count = (UI)SendMessage(hTab, TCM_GETITEMCOUNT, 0, 0);
45 :             //  タブコントロール内のタブテキストをログ表示
46 :             memset(&wrk, 0, sizeof wrk);
47 :             wrk.tci.mask       = TCIF_TEXT;
48 :             wrk.tci.pszText    = (UTP)AjcPmGetAddr(hPm, sizeof wrk.tci);
49 :             wrk.tci.cchTextMax = sizeof wrk.txt;
50 :             for (i = 0; i < Count; i++) {
51 :                 //  タブテキスト取得，表示
52 :                 AjcPmWrite (hPm, 0, &wrk, sizeof wrk);
53 :                 SendMessage(hTab, TCM_GETITEM, i, AjcPmGetAddr(hPm, 0));  ①
54 :                 AjcPmRead  (hPm, 0, &wrk, sizeof wrk);                    ②
55 :                 AjcPrintF(TEXT("  %s¥n"), wrk.txt);                       ③
56 :             }
57 :             //  エクスプローラ・プロセス内のメモリ解放
58 :             AjcPmDelete(hPm);
```



```
59 :         }
60 :         //  タブコントロールなし・・・
61 :         else {
62 :             AjcPrintF(TEXT("タブコントロールが見つかりません。¥n"));
63 :         }
64 :     }
65 :     else {
66 :         AjcPrintF(TEXT("ドライブ・プロパティ・ウインドが見つかりません。¥n"));
67 :     }
68 :
69 :     AjcPrintF(TEXT("¥n Hit Enter Key !"));
70 :     getchar();
71 :
72 :     return 0;
73 : }
74 : //-------------------------------------------------------------------------------------------------------------//
75 : //  エクスプローラ・ドライブ・プロパティウインド検索（「のプロパティ」を含むウインド検索）                    //
76 : //-------------------------------------------------------------------------------------------------------------//
77 : static  BOOL CALLBACK cbEnumWindow(HWND hwnd, LPARAM lParam)
78 : {
79 :     BOOL    rc = TRUE;
80 :     UT      name[256];
81 :
82 :     GetWindowText(hwnd, name, 256);
83 :
84 :     if (MAjcStrStr(name, TEXT("のプロパティ"))) {
85 :         *((HWND*)lParam) = hwnd;
86 :         rc = FALSE;
87 :     }
88 :     return rc;
89 : }
90 : //-------------------------------------------------------------------------------------------------------------//
91 : //  エクスプローラ・ドライブ・プロパティウインド内のタブコントロール（子孫ウインド）検索                      //
92 : //-------------------------------------------------------------------------------------------------------------//
93 : static  BOOL CALLBACK cbEnumChild(HWND hwnd, LPARAM lParam)
94 : {
95 :     BOOL    rc = TRUE;
96 :     UT      name[256];
97 :
98 :     GetClassName(hwnd, name, 256);
99 :
100 :     if (MAjcStrICmp(name, TEXT("SysTabControl32")) == 0) {
101 :         *((HWND*)lParam) = hwnd;
102 :         rc = FALSE;
103 :     }
104 :     return rc;
105 : }
```

22. 拡張ツールチップ

ツールチップの表示を行います。
ここで表示するツールチップは、Windows 標準のツールチップウインドではありません。(独自のツールチップウインドです)
Windows 標準のチップテキストウインドと同等の機能に、以下の機能を追加しています。

表示拡張機能

・複数行のチップテキストも表示可能です。(複数行とするには、表示テキストの改行位置に '¥n' を含めます)
・ツールチップの表示を継続(延長) することができます (カーソルを素早くツールチップ上に移動する)
・チップテキストのフォント, 文字色, 背景色や外枠の色を指定できます。
・テキスト中にエスケープシーケンス ("¥x1B[・・") を含めることにより、部分テキストの文字色や文字背景色等を変更できます。

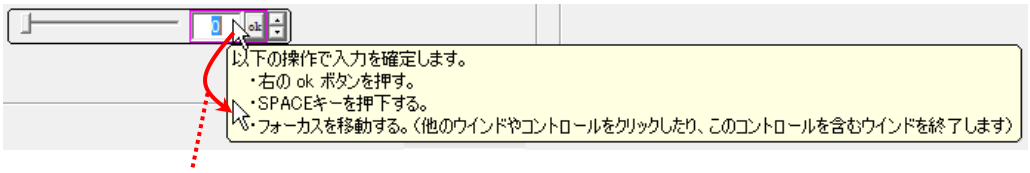
全体的な指定 (文字列中に１つだけ指定する。複数指定時は最後の指定が有効)

1	"¥x1B[*rrr*;*ggg*;*bbb*B" (※２)	ウインド背景色 (rrr;ggg;bbb は色の成分値(rrr:赤, ggg:緑, bbb:青))
	"¥x1B[**0**B" ～ "¥x1B[**7**B" (※２)	〃 (**0** ～ **7** はパレット番号)
2	"¥x1B[*rrr*;*ggg*;*bbb*F"	外枠表示色 (rrr;ggg;bbb は色の成分値(rrr:赤, ggg:緑, bbb:青))
	"¥x1B[**0**F" ～ "¥x1B[**7**F"	〃 (**0** ～ **7** はパレット番号

その他のエスケープシーケンス
その他のエスケープシーケンスについては「テキスト描画」の章を参照してください。

ツールチップの表示継続

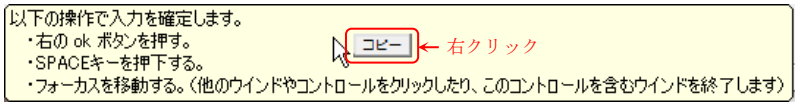
ツールチップ上にマウスカーソルを移動すると、ツールチップの表示を継続します。



マウスカーソルを素早くツールチップ上に移動するとツールチップの表示を継続します。
その後、マウスカーソルをツールチップ外に移動すると、ツールチップは消えます。

ツールチップのコピー

ツールチップ上を右クリックすると「コピー」ボタンが表示されます。
「コピー」ボタンを押すと、ツールチップ・テキストがクリップボードにコピーされます。
SHIFT キーを押しながら「コピー」ボタンを押した場合は、ツールチップのイメージがコピーされます。



ツールチップの非表示

ツールチップ上をクリックすると、ツールチップは消えます。
「コピー」ボタンが表示されている場合は、「コピー」ボタンが消えます。

イメージ (ビットマップ, アイコン) の描画

イメージ (ビットマップやアイコン) を描画するには、AjcTipTextRegist*XXX*()で、あらかじめイメージを登録しておき、文字列中にエスケープシーケンス("¥x1B[*id*・・・**z**")を指定します。

ex.

```
HICON   hIco = (HICON   )LoadImage(hInst, MAKEINTRESOURCE(IDI_BMP), IMAGE_ICON   , 16, 16, LR_DEFAULTCOLOR);
HBITMAP hBmp = (HBITMAP)LoadImage(hInst, MAKEINTRESOURCE(IDB_ICO), IMAGE_BITMAP, 48, 48, LR_DEFAULTCOLOR);
    ・・・
    AjcTipTextRegistIcon  (hTxo, hIco, 0, 0, 0, 0, 0);    // アイコン登録     (id = 0)
    AjcTipTextRegistBitmap(hTxo, hBmp, 1, 0, 0, 0, 0);    // ビットマップ登録(id = 1)
    ・・・
    AjcTipTextShow(hTxo,0, 0, TEXT("¥x1B[0;-20;2;0;0z")                 // (右端-20, 2) の位置にアイコン表示
                              TEXT("¥x1B[1;2;20;24;24z"), 5000, NULL); // (2, 20)の位置にビットマップ表示
    ・・・
```

22.1. サポートＡＰＩ

チップテキストＡＰＩの一覧を以下に示します。

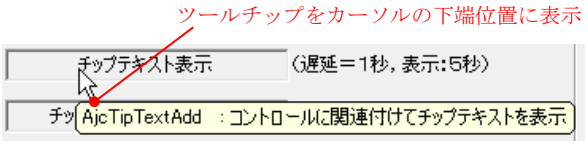
#	関数名	内容	
1	AjcTipTextSetTipPosMode	ツールチップ表示位置モードの設定	
2	AjcTipTextAdd	ツールチップの関連付け	コントロールに関連付けた チップテキストの表示／非表示
3	AjcTipTextAddEx	ツールチップの関連付け（拡張版）	
4	AjcTipText{Set/Get}ShowAlways	ツールチップ表示条件の設定／取得	
5	AjcTipText{Set/Get}ShowForActive	全ツールチップ表示条件の設定／取得	
6	AjcTipText{Enable/GetEnableState}	チップテキストの表示許可／禁止 設定／取得	
7	AjcTipText{EnableAll/GetEnableAllState}	全ツールチップの表示許可／禁止 設定／取得	
8	AjcTipTextRemove	ツールチップの関連付け解除	
9	AjcTipTextRemoveAll	全てのツールチップの関連付け解除	
10	AjcTipText{Set/Get}WindowTime	コントロール間移動猶予時間設定／取得	
11	AjcTipTextSetCallBack	コールバックの設定	
12	AjcTipTextSetCallBackNoBuf	コールバックの設定（作業バッファ未使用）	
13	AjcTipTextGetInfo	ツールチップの関連付け情報取得	
14	AjcTipTextShow AjcTipTextShowCentr	ツールチップの表示	表示位置を指定した チップテキストの表示
15	AjcTipTextShowEx AjcTipTextShowCenterEx	ツールチップの詳細表示	
16	AjcTipTextShowCloseMark	ツールチップウインドに閉じるマーク表示設定	
17	AjcTipTextMove	ツールチップウインドの移動	
18	AjcTipTextMoveCursor	マウスカーソルをツールチップ上へ移動	
19	AjcTipTextHide	ツールチップ非表示	
20	AjcTipTextGetRect	チップテキストウインド矩形情報取得	
21	AjcTipText{Set/Get}Palette	パレット色の設定／取得	
22	AjcTipTextGetSize	文字列によるツールチップウインドの表示 サイズ取得	
23	AjcTipText{Set/Get}DefTextColor AjcTipText{Set/Get}DefBorderColor AjcTipText{Set/Get}DefBkColor	デフォルトの表示色設定／取得	テキスト表示色 外枠表示色 ウインド背景色
24	AjcTipText{Set/Get}DefMsDelay	デフォルトの表示遅延時間の設定／取得	
25	AjcTipText{Set/Get}DefMsShow	デフォルトの表示時間の設定／取得	
26	AjcTipText{Set/Get}DefFont	デフォルトフォントの設定／取得	
27	AjcTipTextEnableMultiThread	マルチスレッドの許可／禁止	
28	AjcTipTextSetNtcSubclass	サブクラス化の通知情報設定	
29	AjcTipTextClrNtcSubclass	サブクラス化の通知情報消去	
30	AJCTIPCTRL_SETHWND	チップコントロール対象のウインドを設定	マクロ
31	AJCTIP_SETFRIEND	チップコントロール友達ウインド設定	
32	AjcTipTextRegistBitmap	描画イメージ（ビットマップ）の登録	
33	AjcTipTextRegistIcon	描画イメージ（アイコン）の登録	
34	AjcTipTextUnregistImage	描画イメージの登録解除	

22.1.1. ツールチップ表示位置モードの設定(AjcTipTextSetTipPosMode)

形　式 ： AJCTIP_POSMODE　AjcTipTextSetTipPosMode (AJCTIP_POSMODE PosMode);

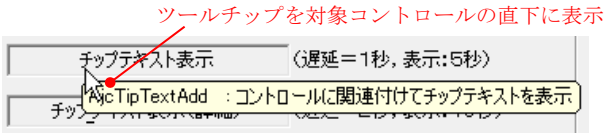
引　数 ： PosMode　- ツールチップ表示位置モード
AJCTIP_POS_UNDER_CONTROL ： ツールチップを対象コントロールの直下に表示（デフォルト）
AJCTIP_POS_UNDER_CURSOR　： ツールチップをマウスカーソルの下に表示
-1　： ツールチップ表示位置モードを設定しない

説　明 ： ツールチップを　どの位置に表示するかを設定します。
AJCTIP_POS_UNDER_CURSOR を指定した場合は、ツールチップをマウスカーソルの下端位置に表示します。



この場合、ツールチップの表示継続操作が多少困難となります。
（ツールチップの表示を継続するには、カーソルをコントロールからツールチップ上に素早く移動しなければならないが、コントロールとツールチップが離れている為、若干操作が難しくなる）

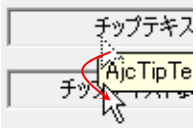
AJCTIP_POS_UNDER_CONTROL を指定した場合は、ツールチップを対象コントロールの直下に表示します。（デフォルト）



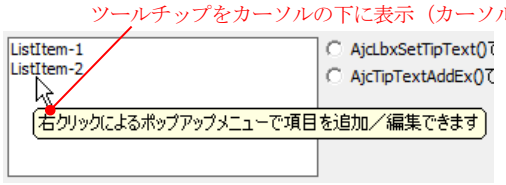
この場合、ツールチップとマウスカーソルが重なり、多少見えずらくなる場合があります。
これは、以下の操作で回避することができます。


あらかじめ、コントロールの上部にカーソルを置く


同一コントロール上でカーソルを少し移動する。(表示開始から１秒以内)


カーソルをツールチップ上に移動する。
この場合、カーソルをツールチップ外に移動するまでツールチップは表示され続けます。

AJCTIP_POS_UNDER_CONTROL を指定した場合でも、マウスカーソル下端位置が対象コントロール内である場合は、AJCTIP_POS_UNDER_CURSOR と同様にツールチップをマウスカーソルの下端位置に表示します。



PosMode = -1 とした場合は、ツールチップ表示位置モードは設定されず、前回設定値の取得だけとなります。

戻り値 ： 設定前のツールチップ表示位置モード

22.1.2. ツールチップの関連付け(AjcTipTextAdd[F])

形　式 ： BOOL AjcTipTextAdd (HWND hCtrl, C_UTP pTxt);
BOOL AjcTipTextAddF(HWND hCtrl, C_UTP pFmt, ...);

引　数 ： hCtrl - ツールチップを関連付けするコントロールのハンドル
pTtxt - 表示するツールチップテキスト（"@MyWndText@"を指定した場合は、自ウインドのテキストに置き換えます）
pFmt - 書式文字列(printf()と同じ)

説　明 ： hCtrl で指定したコントロールにツールチップを関連付けます。
当該コントロール上にマウスカーソルを置くと、ツールチップが表示されるようになります。
ツールチップは、マウスカーソルの下に表示されます。
ツールチップの表示が画面に収まらない場合は、画面に収まるように表示位置を変更します。
以下の項目は、固定値で設定されます。

・表示遅延時間	- 1000ms（1 秒）	・文字色	- 黒
・表示時間[ms]	- 5000ms（５秒）	・背景色	- クリーム色
・フォントハンドル	- "MS UI Gothic", 文字高さ＝12 ピクセル	・外枠色	- 黒

戻り値 ： TRUE - 成功
FALSE - 失敗

22.1.3. ツールチップの関連付け(AjcTipTextAdd[F]Ex)

形　式 ： BOOL AjcTipTextAddEx (HWND hCtrl, C_UTP pTxt , int msDelay, int msShow,
HFONT hFont, COLORREF TextColor, COLORREF BackGround, COLORREF BorderColor);

BOOL AjcTipTextAddFEx(HWND hCtrl, int msDelay, int msShow,
HFONT hFont, COLORREF TextColor, COLORREF BackGround, COLORREF BorderColor, C_UTP pFmt, ...);

引　数 ： hCtrl - ツールチップを関連付けするコントロールのハンドル
pTtxt - 表示するツールチップテキスト（"@MyWndText@"を指定した場合は、自ウインドのテキストに置き換えます）
msDelay - 表示遅延時間[ms]
msShow - 表示時間[ms]
hFont - フォントハンドル（NULL: "MS UI Gothic", 文字高さ＝１２ピクセル）
TextColor - 文字色（－１：黒）
BackGround - ウインド背景色（－１：クリーム色）
BorderColor - 外枠色（－１：黒, －２：外枠非表示）
pFmt - 書式文字列(printf()と同じ)

説　明 ： hCtrl で指定したコントロールにツールチップを関連付けます。
AjcTipTextShow()よりも詳細な設定ができます。
当該コントロール上にマウスカーソルを置くと、ツールチップが表示されるようになります。
ツールチップは、マウスカーソルの下に表示されます。
ツールチップの表示が画面に収まらない場合は、画面に収まるように表示位置を変更します。

戻り値 ： TRUE - 成功
FALSE - 失敗

22.1.4. ツールチップ表示条件の設定／取得(AjcTipText{Set/Get}ShowAlways)

形　式 ： BOOL AjcTipTextSetShowAlways (HWND hCtrl, BOOL fShowAlways);
BOOL AjcTipTextGetShowAlways(HWND hCtrl);

引　数 ： hCtrl - ツールチップが関連付けられているコントロールのウインドハンドル
fShowAlways - ツールチップの表示条件
TRUE ： 非アクティブな状態時もツールチップを表示する（デフォルト）
FALSE ： 非アクティブな状態時はツールチップを表示しない

説　明 ： ツールチップの表示条件を設定／取得します。
fShowAlways=TRUE を指定した場合は、非アクティブな状態時もツールチップを表示する
fShowAlways=FALSE を指定した場合は、非アクティブな状態時ではカーソルを置いてもツールチップを表示しません。

戻り値 ： TRUE - 成功
FALSE - 失敗

22.1.5. 全ツールチップ表示条件の設定／取得(AjcTipText{Set/Get}ShowForActive)

形　式 ： VO AjcTipTextSetShowForActive(BOOL fShowForActive); --- 設定
BOOL AjcTipTextGetShowForActive(VO); ---------- 取得

引　数 ： BOOL fShowForActive - TRUE ： 全て、アクティブ状態時のみツールチップを表示する
FALSE ： 個々の AjcTipTextSetShowAlways()による設定に従い表示する

説　明 ： すべてのツールチップにおける、表示条件を設定／取得します。
fShowForActive=TRUE に設定した場合、すべてのコントロールにおいて、自アプリがアクティブ状態時のみツールチップを表示します。
fShowForActive=FALSE に設定した場合は、個々の AjcTipTextSetShowAlways()による設定に従いツールチップを表示します。

戻り値 ： [AjcTipTextSetShowForActive] [AjcTipTextGetShowForActive]
なし fShowForActive の設定状態

22.1.6. ツールチップの表示／非表示の 設定／取得(AjcTipText{Enable/GetEnableState})

形　式 ： BOOL AjcTipTextEnable (HWND hCtrl, BOOL fEnable); -- 設定
BOOL AjcTipTextGetEnableState(HWND hCtrl); ---------- 取得

引　数 ： hCtrl - ツールチップが関連付けられているコントロールのウインドハンドル
fEnale - ツールチップの表示許可／禁止設定(TRUE:許可. FALSE:禁止)

説　明 ： 指定したコントロールで、ツールチップの表示許可／禁止を設定します。
fEnable=FALSE を指定すると、hCtrl で指定したコントロールで、カーソルを置いてもツールチップが表示されなくなります。

戻り値 ： [AjcTipTextShow] [AjcTipTextGetShowState]
TRUE ： 成功 設定状態(TRUE:表示許可. FALSE:表示禁止)
FALSE: 失敗

22.1.7. 全ツールチップの表示許可／禁止の 設定／取得(AjcTipText{EnableAll/GetEnableAllState})

形　式 ： VO AjcTipTextEnableAll (BOOL fEnable);
BOOL AjcTipTextGetEnableAllStae(VO);

引　数 ： fEnable - ツールチップの表示設定(TRUE:表示許可. FALSE:表示禁止)

説　明 ： 全てのコントロールでコントロールで、ツールチップの表示許可／禁止を設定します。
fEnable=FALSE を指定すると、全てのコントロールで、カーソルを置いてもツールチップが表示されなくなります。

戻り値 ： [AjcTipTextEnableAll] [AjcTipTextGetEnableAllStae]
なし 設定状態(TRUE:表示許可. FALSE:表示禁止)

22.1.8. ツールチップの関連付け解除(AjcTipTextRemove)

形　式 ： BOOL AjcTipTextRemove (HWND hCtrl);

引　数 ： hCtrl　　- ツールチップが関連付けられているコントロールのウインドハンドル

説　明 ： 指定したコントロールで、ツールチップの関連付けを解除します。
hCtrl で指定したコントロールにおいて、ツールチップが表示されなくなります。
指定したコントロールでチップテキストが関連付けられていない場合は、何もせずに FALSE を返します。

戻り値 ： TRUE - 成功
FALSE - 失敗

22.1.9. 全てのツールチップの関連付け解除(AjcTipTextRemoveAll)

形　式 ： BOOL AjcTipTextRemoveAll(VO);

引　数 ： なし

説　明 ： 全てのコントロールで、ツールチップの関連付けを解除します。

戻り値 ： TRUE - 成功
FALSE - 失敗

22.1.10. コントロール間移動猶予時間設定／取得(AjcTipText{Set|Get}DefementTime)

形　式 ： VO AjcTipTextSetWindowTime (int msTime); --- 設定
int AjcTipTextGetWindowTime (VO); ---------- 取得

引　数 ： msTime　- コントロール間移動猶予時間[ms]

説　明 ： コントロール間移動猶予時間（遅延時間を無視する時間）を設定します。
通常ははマウスカーソルがコントロール上に置かれ、遅延時間だけ停止した場合にツールチップが表示されますが、
ある一定の時間内にコントロールからコントロールへ移動した際（あるいは、一旦コントロールを離た後に、また戻った場合）は、遅延なしにツールチップを表示します。
この「ある一定の時間」がコントロール間移動猶予時間です。
コントロール間移動猶予時間のデフォルトは 500ms（0.5 秒）です。

戻り値 ： コントロール間移動猶予時間（取得時）

22.1.11. コールバックの設定(AjcTipTextSetCallBack)

形　式 ： BOOL AjcTipTextSetCallBack (HWND hCtrl, UX cbp,
BOOL (CALLBACK *cbChkCursor)(HWND hCtrl, LPPOINT ptClient, UX cbp),
C_UTP (CALLBACK *cbGetText)(HWND hCtrl, UTP pBuf, UI lBuf, UX cbp));

引　数 ： hCtrl - ツールチップが関連付けられているコントロールのウインドハンドル
cbp - コールバックパラメタ
cbChkCursor - マウス位置判定用コールバック （不要時は NULL）
cbGetText - チップテキスト取得用コールバック （不要時は NULL）

説　明 ： hCtrl で指定したコントロールにツールチップを表示する際のコールバック関数を設定します。

cbChkCursor は、マウスカーソル位置の判定用のコールバック関数を指定します。
カーソルがコントロール上にある場合、cbChkCursor が呼び出されます。
cbChkCursor はカーソル位置をチェックし、目的の位置にある場合は TRUE を返し、ツールチップの表示を開始させます。
cbChkCursor は、コントロール内の特定の領域がチップテキストの表示対象となる場合に使用します。
cbChkCursor=NULL とした場合は、コントロール全体がツールチップ表示の対象域となります。

チップテキストを表示する際には、cbGetText が呼び出されます。
cbGetText が、ツールチップテキストを返すことにより、状況に依存した（可変の）ツールチップの表示が可能となります。

戻り値 ： TRUE - 成功
FALSE - 失敗

コールバック：

cbChkCursor（マウス位置判定用コールバック）

形　式 ： BOOL CALLBACK *cbChkCursor* (HWND hCtrl, LPPOINT ptClient, UX cbp)

引　数 ： hCtrl - ツールチップを表示するコントロールのウインドハンドル
ptClient - マウスカーソルの位置（コントロールのクライアント領域の位置）
cbp - コールバックパラメタ

説　明 ： マウスカーソルの位置をチェックし、対象となる領域上であれば TRUE を返し、対象外の位置であれば FALSE を返します。
TRUE を返した場合、ツールチップの表示が開始されます。
FALSE を返した場合は、ツールチップを表示しません。

戻り値 ： TRUE ：マウスカーソルが対象となる領域上にある
FALSE ：マウスカーソルは、対象外の位置にある

cbGetText（チップテキスト文字列取得用コールバック）

形　式 ： C_UTP CALLBACK *cbGetText* (HWND hCtrl, UTP pBuf, UI lBuf, UX cbp)

引　数 ： hCtrl - ツールチップを表示するコントロールのウインドハンドル）
pBuf - チップテキスト設定用バッファ
lBuf - チップテキスト設定用バッファのバイト数／文字数（４０９６固定）
cbp - コールバックパラメタ

説　明 ： 表示するツールチップテキストを返します。
NULL を返した場合は、ツールチップを表示しません。
pBuf はチップテキストを設定用の作業バッファです。pBuf にチップテキストを設定して pBuf を戻り値とすることができます。但し、チップテキストがバッファサイズ(lBuf)を超える場合は使用できません。

戻り値 ： ≠NULL：表示するチップテキスト文字列へのポインタ
＝NULL：チップテキストを表示しない

22.1.12. コールバックの設定(AjcTipTextSetCallBackNoBuf) ・・作業用バッファ未使用

形　式 ： BOOL AjcTipTextSetCallBackNoBuf (HWND hCtrl, UX cbp,
BOOL (CALLBACK *cbChkCursor)(HWND hCtrl, LPPOINT ptClient, UX cbp),
C_UTP (CALLBACK *cbGetText)(HWND hCtrl, UX cbp));

引　数 ： hCtrl　　- ツールチップが関連付けられているコントロールのウインドハンドル
cbp　　- コールバックパラメタ
cbChkCursor - マウス位置判定用コールバック　　(不要時は NULL)
cbGetText　- チップテキスト取得用コールバック (不要時は NULL)

説　明 ： hCtrl で指定したコントロールにツールチップを表示する際のコールバック関数を設定します。
このＡＰＩは、チップテキスト取得のコールバック時に、チップテキスト設定用の作業バッファ(pBuf)を使用しない点を除いて AjcTipTextSetCallBack()と同じです。

戻り値 ： TRUE　- 成功
FALSE - 失敗

コールバック：

cbChkCursor（マウス位置判定用コールバック）

形　式 ： BOOL CALLBACK *cbChkCursor* (HWND hCtrl, LPPOINT ptClient, UX cbp)

引　数 ： hCtrl　　- ツールチップを表示するコントロールのウインドハンドル
ptClient - マウスカーソルの位置（コントロールのクライアント領域の位置）
cbp　　- コールバックパラメタ

説　明 ： マウスカーソルの位置をチェックし、対象となる領域上であれば TRUE を返し、対象外の位置であれば FALSE を返します。
TRUE を返した場合、ツールチップの表示が開始されます。
FALSE を返した場合は、ツールチップを表示しません。

戻り値 ： TRUE　：マウスカーソルが対象となる領域上にある
FALSE ：マウスカーソルは、対象外の位置にある

cbGetText（チップテキスト文字列取得用コールバック）

形　式 ： C_UTP CALLBACK *cbGetText* (HWND hCtrl, UTP pBuf, UI lBuf, UX cbp)

引　数 ： hCtrl　　- ツールチップを表示するコントロールのウインドハンドル)
cbp　　- コールバックパラメタ

説　明 ： 表示するツールチップテキストを返します。
NULL を返した場合は、ツールチップを表示しません。

戻り値 ： ≠NULL：表示するチップテキスト文字列へのポインタ
＝NULL：チップテキストを表示しない

22.1.13. ツールチップの関連付け情報取得(AjcTipTextGetInfo)

形　式 ： UI　AjcTipTextGetInfo (HWND hCtrl, PAJCTIPINFO pInfoBuf, UTP pTxtBuf, UI lTxtBuf);

引　数 ： hCtrl　　- ツールチップが関連付けられているコントロールのウインドハンドル
pInfoBuf　- ツールチップの関連付け情報を格納するバッファのアドレス（不要時は NULL）
pTxtBuf　- ツールチップテキストを格納するバッファのアドレス(必要なバイト数/文字数を取得する場合は NULL)
lTxtBuf　- ツールチップテキストを格納するバッファのバイト数／文字数

説　明 ： ツールチップの関連付け情報を取得します。
pInfoBuf で示されるバッファには、以下の情報が格納されます。

```
typedef struct {
    HWND        hCtrl;              //  コントロールのウインドハンドル
    int         msDelay;            //  表示遅延時間[ms]
    int         msShow;             //  表示時間    [ms]
    HFONT       hFont;              //  フォントハンドル     (NULL : デフォルトフォント)
    COLORREF    TextColor;          //  テキスト描画色       (-1 : デフォルト色)
    COLORREF    BackGround;         //  背景色               (-1 : デフォルト色)
    COLORREF    BorderColor;        //  外枠描画色           (-1 : デフォルト色, -2 : 外枠非表示)
    UI          WindowTime;         //  コントロール間移動猶予時間[ms]
    HFONT       hDefFont;           //  デフォルトのフォントハンドル
    COLORREF    DefTextColor;       //      〃     のテキスト描画色
    COLORREF    DefBackGround;      //      〃     の背景色
    COLORREF    DefBorderColor;     //      〃     の外枠描画色
    int         DefMsDelay;         //      〃     の表示遅延時間
    int         DefMsShow;          //      〃     の表示時間
} AJCTIPINFO, *PAJCTIPINFO;
typedef const AJCTIPINFO *PCAJCTIPINFO;
```

pTxtBuf で示されるバッファにはツールチップテキスト（文字列）が格納されます。
pTxtBuf=NULL あるいは、lTxtBuf で指定したバッファサイズが小さい場合は文字列を格納せずに、文字列を格納するのに必要なバッファサイズ（バイト数／文字数）を返します。

戻り値 ： ≠-1 ： ツールチップテキストのを格納するのに必要なバッファのバイト数／文字数
＝-1 ： 失敗

22.1.14. ツールチップの表示(AjcTipTextShow)

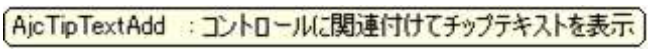
形　式 ： BOOL　AjcTipTextShow　　(int x, int y, C_UTP pTxt, int msTime, HFONT hFont);
BOOL　AjcTipTextShowCenter (HWND hwnd　, C_UTP pTxt, int msTime, HFONT hFont);

引　数 ： x, y　　- ツールチップ・ウインドの表示位置
hwnd　　- ウインドハンドル（NULL ：カーソルが位置するモニタ）
pTtxt　- 表示するテキスト
msTime　- 表示する時間[ms]（-1 の場合はデフォルト値）
hFont　- 表示フォント（NULL の場合はデフォルトのフォントを使用）

説　明 ： ツールチップを画面の最前面に表示します。
AjcTipTextShowCenter()は、hwnd で指定したウインドの中央にツールチップを表示します。
hwnd=NULL の場合は、カーソル位置のモニタの中央にツールチップを表示します。
カーソルの動きに依存しないで、単純にツールチップウインドの表示を行います。
ツールチップの表示が画面に収まらない場合は、画面に収まるように表示位置を変更します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

備　考 ： 本ＡＰＩでは、AjcTipTextEnableAll()で表示禁止状態でも、ツールチップを表示します。
ツールチップの表示例を以下に示します。(デフォルトのフォント)



22.1.15. ツールチップの詳細表示(AjcTipTextShowEx)

形　式 ： BOOL　AjcTipTextShowEx　　(int x, int y, int width, int height, C_UTP pTxt, int msTime, HFONT hFont,
COLORREF TextColor, COLORREF BackGround, COLORREF BorderColor);

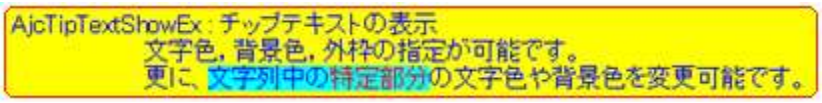
BOOL　AjcTipTextShowCenterEx(HWND hwnd　, int width, int height, C_UTP pTxt, int msTime, HFONT hFont,
COLORREF TextColor, COLORREF BackGround, COLORREF BorderColor);

引　数 ： x, y　　- ツールチップ・ウインドの表示位置
hwnd　　- ウインドハンドル（NULL : カーソルが位置するモニタ）
width　　- ツールチップ・ウインドの幅　（０指定時は自動算出）
height　　- ツールチップ・ウインドの高さ（０指定時は自動算出）
pTtxt　　- 表示するテキスト
msTime　　- 表示する時間[ms]　(-1 の場合はデフォルト値)
hFont　　- 表示フォント（NULL の場合はデフォルトのフォントを使用）
TextColor　- 文字色（－１：デフォルト色（黒））
BackGround　- ウインド背景色（－１：デフォルト色（AjcTipTextShow()の「備考」で示した表示例参照））
BorderColor - 外枠色（－１：外枠非表示）

説　明 ： ツールチップを画面の最前面に表示します。
hwnd=NULL の場合は、カーソル位置のモニタの中央にツールチップを表示します。
AjcTipTextShowCenterEx()は、hwnd で指定したウインドの中央にツールチップを表示します。
カーソルの動きに依存しないで、単純にツールチップの表示を行います。
ツールチップの表示が画面に収まらない場合は、画面に収まるように表示位置を変更します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

備　考 ： 本ＡＰＩでは、AjcTipTextEnableAll()で表示禁止状態でも、ツールチップを表示します。
ツールチップの表示例を以下に示します。(文字色＝青，背景色＝黄，外枠＝赤を指定，３行目はESC シーケンスで色を指定)



22.1.16. ツールチップウインドに閉じるマーク(X)の表示設定(AjcTipTextShowCloseMark)

形　式 ： VO　AjcTipTextShowCloseMark(BOOL fMark);

引　数 ： fMark -クローズマーク表示フラグ（FALSE:表示しない，TRUE:表示する）

説　明 ： 以下のＡＰＩでツールチップウインドをを表示した際に、閉じるマーク(X)の表示設定を行います。

・AjcTipTextShow()　　・AjcTipTextShowCenter()
・AjcTipTextShowEx()　　・AjcTipTextShowCenterEx()

fMark=TRUE を指定した場合、ツールチップの右上に 閉じるマーク(X)を表示します。

fMark = TRUE　　閉じるマーク



fMark = FALSE



戻り値 ： なし

備　考 ： コントロールに関連付けずに、単独でツールチップを表示した場合、カーソルがツールチップから離れても、所定の時間が経過するまでツールチップは消えません。
そこで、「クリックすればツールチップが閉じる」ことを明示する意味で、閉じるマークを表示することを意図しています。

22.1.17. ツールチップウインドの移動(AjcTipTextMove)

形　式 ： VO　AjcTipTextMove (int x, int y);

引　数 ： x　　- 移動先のＸピクセル位置
y　　- 移動先のＹピクセル位置

説　明 ： ツールチップウインドを移動します。

戻り値 ： なし

22.1.18. カーソルをツールチップ上へ移動(AjcTipTextMoveCursor)

形　式 ： VO　AjcTipTextMoveCursor (AJCTIP_CURMOVE cm);

引　数 ： cm　　- 移動位置(AJCTIP_CM_CENT/LU/RU/LD/RD)

説　明 ： マウスカーソルをツールチップ上の cm で指定された位置に移動します。
マウスカーソルがツールチップ上にある間は、(表示時間が経過しても) ツールチップを表示し続けます。
cm は、以下のいずれかのシンボルを指定します。

cm	内容
AJCTIP_CM_CENT	カーソルをチップテキストの中心へ移動
AJCTIP_CM_LU	〃　左上隅へ移動
AJCTIP_CM_RU	〃　右上隅へ移動
AJCTIP_CM_LD	〃　左下隅へ移動
AJCTIP_CM_RD	〃　右下隅へ移動

戻り値 ： なし

備　考 ： 手動でマウスカーソルをツールチップ上に移動しても、ツールチップを表示し続けますが、素早く移動する必要があります。

22.1.19. ツールチップ非表示(AjcTipTextHide)

形　式 ： VO　AjcTipTextHide (VO);

引　数 ： なし

説　明 ： ツールチップを非表示にします。

戻り値 ： なし

22.1.20. チップテキストウインド矩形情報取得(AjcTipTextGetRect)

形　式 ： BOOL　AjcTipTextGetRect (LPRECT pRect);

引　数 ： pRect – 矩形所法を格納するバッファのアドレス（不要時は NULL）

説　明 ： ツールチップウインドの矩形情報を取得します。

戻り値 ： TRUE　- 成功
FALSE - 失敗（チップテキスト非表示中）

22.1.21. パレット色の設定／取得(AjcTipText{Set|Get}Palette)

形　式 ： VO　　AjcTipTextSetPalette (int ix, COLORREF color); -- 設定
COLORREF AjcTipTextGetPalette(int ix) -------------------- 取得

引　数 ： ix　- パレット番号（０～７）
color - パレット色

説　明 ： 指定パレット番号の色を設定／取得します。

戻り値 ： パレットの色（取得時）

22.1.22. 文字列によるツールチップウインドの表示サイズ取得(AjcTipTextGetSize)

形　式 ： BOOL　AjcTipTextGetSize(C_UTP pTxt, HFONT hFont, BOOL fBorder, LPSIZE pBuf);

引　数 ： pTtxt　　- 表示するテキスト
hFont　　- 表示フォント（NULL の場合はデフォルトのフォントを使用）
fBorder　- 外枠の有無（TRUE:外枠あり，FALSE:外枠なし）
pBuf　　- ツールチップの表示サイズを格納するバッファのアドレス

説　明 ： pTxt で示す文字列を表示する際のツールチップウインドの表示サイズを取得します。
”¥x1B[*id*・・・**z**” によるイメージ描画も計算に含まれます。
pBuf->cx に横ピクセル数，pBuf->cy に縦ピクセル数が格納されます。
pTxt=NULL を指定した場合は、pBuf->cx，pBuf->cy ともにゼロが設定されます。

戻り値 ： TRUE　- 成功
FALSE - 失敗

22.1.23. デフォルトの表示色設定／取得(AjcTipText{Set|Get}Def{Text/Border/Bk}Color)

形　式 ： VO AjcTipTextSetDefTextColor　(COLORREF color);　--　デフォルトテキスト表示色設定
VO AjcTipTextSetDefBorderColor(COLORREF color);　--　デフォルト外枠表示色設定
VO AjcTipTextSetDefBkColor　　(COLORREF color);　--　デフォルトのウインド背景色設定

COLORREF AjcTipTextGetDefTextColor　(VO);　--------　デフォルトテキスト表示色取得
COLORREF AjcTipTextGetDefBorderColor(VO);　---------　デフォルト外枠表示色取得
COLORREF AjcTipTextGetDefBkColor　　(VO);　--------　デフォルトのウインド背景色取得

引　数 ： color　- デフォルト表示色

説　明 ： 各種デフォルト表示色を設定／取得します。
表示色指定の無いＡＰＩや、各種表示色に -1 を指定した場合の表示色となります。

戻り値 ： 各種表示色（取得時）

22.1.24. デフォルト表示遅延時間の設定／取得(AjcTipText{Set|Get}DefMsDelay)

形　式 ： VO　AjcTipTextSetDefMsDelay(int msDelay);　-- 設定
int AjcTipTextGetDefMsDelay(VO);　---------- 取得

引　数 ： msDelay　- デフォルト表示遅延時間[ms]　（マウスカーソルが停止してから、ツールチップが表示されるまでの時間）

説　明 ： デフォルトの表示遅延時間を設定／取得します。

戻り値 ： デフォルト表示遅延時間（取得時）

22.1.25. デフォルト表示時間の設定／取得(AjcTipText{Set|Get}DefMsShow)

形　式 ： VO AjcTipTextSetDefMsShow (int msShow); -- 設定
int AjcTipTextGetDefMsShow (VO); ---------- 取得

引　数 ： msShow　- デフォルト表示時間[ms]　(ツールチップを表示する時間)

説　明 ： デフォルトの表示時間を設定／取得します。

戻り値 ： デフォルト表示時間（取得時）

22.1.26. デフォルトフォントの設定／取得(AjcTipText{Set|Get}DefFont)

形　式 ： VO　AjcTipTextSetDefFont(HFONT hFont);
HFONT AjcTipTextGetDefFont(VO);

引　数 ： hFont　　- フォントハンドル

説　明 ： デフォルトのフォントを設定／取得します。
フォント指定のないＡＰＩや、フォントハンドルに NULL を指定した場合のフォントとなります。

戻り値 ： フォントハンドル（取得時）

22.1.27. マルチスレッドの許可／禁止(AjcTipTextEnableMultiThread)

形　式 ： BOOL AjcTipTextEnableMultiThread (BOOL fEnable);

引　数 ： fEnable　- TRUE　: 複数のスレッドからのアクセスを許可
FALSE : 複数のスレッドからのアクセスを禁止（デフォルト）

説　明 ： fEnable=TRUE とした場合、複数のスレッドによるチップテキストの追加／削除等を可能にします。
この場合、各ＡＰＩの入り口と出口で、クリティカルセクションによるスレッド間の排他制御を行います。
但し、次のＡＰＩは、(fEnable=TRUE としても）マルチスレッドでの排他制御を行いません。

・AjcTipTextCreate（ツールチップ表示ウインドの生成）
・AjcTipTextDelete（ツールチップ表示ウインドの破棄）
・AjcAvlEnableMultiThread（本ＡＰＩ）

fEnable=FALSE（デフォルト）とした場合は、マルチスレッドでの排他制御を行いません。

戻り値 ： 前回の禁止／許可状態

22.1.28. サブクラス化の通知情報設定(AjcTipTextSetNtcSubclass)

形　式 ： BOOL AjcTipTextSetNtcSubclass (HWND hCtrl, HWND hNtc, UI msg, UX lParam);

引　数 ： hCtrl　- ツールチップを関連付けるコントロールのハンドル
hNtc　- 通知するウインドのハンドル
msg　- 通知メッセージコード
lParam　– 通知パラメタ

説　明 ： ツールチップの関連付けを行ったコントロールはサブクラス化されます。
コントロールをサブクラス化した／サブクラス化を解除した際に、その旨のメッセージを受け取ることができます。

メッセージは、以下の形式で送信されます。

サブクラス化時　：　SendMessage(hNtc, msg, **TRUE** , lParam);
サブクラス解除時：　SendMessage(hNtc, msg, **FALSE**, lParam);

このＡＰＩは、まだツールチップを関連付けしていないコントロールに対しても設定可能です。

戻り値 ： TRUE　- 成功
FALSE　- 失敗

22.1.29. サブクラス化の通知情報消去(AjcTipTextClrNtcSubclass)

形　式 ： BOOL AjcTipTextClrNtcSubclass(HWND hCtrl);

引　数 ： hCtrl　- ツールチップを関連付けるコントロールのハンドル

説　明 ： AjcTipTextSetNtcSubclass()で設定された、サブクラス化の通知情報を消去します。
以降、サブクラス化／サブクラス化解除の通知は行われなくなります。

戻り値 ： TRUE　- 成功
FALSE　- 失敗

22.1.30. チップコントロール対象のウインドを設定(AJCTIPCTRL_SETHWND)

形　式 ： BOOL AJCTIPCTRL_SETHWND (HCTRL, HWNDTIP);

引　数 ： HCTRL　- ツールチップの関連付け(AjcTipTextAdd[Ex])で指定するウインドハンドル
HTIPWND - 実際にツールチップ制御を行うウインドのハンドル

説　明 ： AjcTipTextAdd()/AjcTipTextAddEx()によりツールチップの関連付けを行うウインド(HCTRL)と、実際にツールチップ制御を行うウインドが異なる場合に使用します。
AjcTipTextAdd()/AjcTipTextAddEx()の呼び出し時に、指定したウインド(HCTRL)を HWNDTIP で指定したウインドに置き換えが行われます。
このマクロは、AjcTipTextAdd()/AjcTipTextAddEx()より前に実行しなければなりません。

戻り値 ： ≠NULL：成功
＝NULL：失敗

22.1.31. チップコントロールの友達ウインドの設定(AJCTIP_SETFRIEND)

形　式 ： BOOL AJCTIP_SETFRIEND (HFRIEND);

引　数 ： HFRIEND　- 友達ウインドとするウインドのハンドル

説　明 ： ツールチップ制御を行う際に、HFRIEND で指定したウインドを実際にツールチップの関連付けを行うウインドと同様にツールチップ表示を行うことを指示します。
例えば、AjcTipTextAdd()/AjcTipTextAddEx()によりツールチップの関連付けを行うウインドに子ウインドがある場合に、子ウインドでも親ウインドと同様にツールチップの表示を行いたい場合に使用します。

HFRIEND で指定したウインドでは、実際にツールチップの関連付けを行っているウインドへ以下のメッセージをパスしなければなりません。

- WM_MOUSEMOVE
- WM_LBUTTONDOWN
- WM_MBUTTONDOWN
- WM_RBUTTONDOWN

戻り値 ： ≠NULL：成功
＝NULL：失敗

22.1.32. 描画イメージ（ビットマップ）の登録 (AjcTipTextRegistBitmap)

形　式 ： BOOL AjcTipTextRegistBitmap (HBITMAP hBmp , UI id, int x, int y, int cx, int cy);

引　数 ： hBmp　- ビットマップハンドル
id　- 登録ＩＤ（０～）
x, y　- デフォルト描画位置
cx, cy - デフォルト描画サイズ

説　明 ： エスケープシーケンス（"¥x1B[*id*・・・z"）で描画するビットマップを登録します。
x, y はデフォルトの描画位置で、エスケープシーケンスで x, y が指定されていない場合の値です。
cx, cy はデフォルトの描画サイズで、エスケープシーケンスで cx, cy が指定されていない場合の値です。
id で指定したイメージが既に登録済みの場合は、旧イメージの登録は解除されます。

戻り値 ： TRUE　- 成功
FALSE - 失敗

22.1.33. 描画イメージ（ビットマップ）の登録 (AjcTxoRegistBitmap)

形　式 ： BOOL AjcTipTextRegistIcon (HICON hIcon , UI id, int x, int y, int cx, int cy);

引　数 ： hIcon　- アイコンハンドル
id　- 登録ＩＤ（０～）
x, y　- デフォルト描画位置
cx, cy - デフォルト描画サイズ

説　明 ： エスケープシーケンス（"¥x1B[*id*・・・z"）で描画するアイコンを登録します。
x, y はデフォルトの描画位置で、エスケープシーケンスで x, y が指定されていない場合の値です。
cx, cy はデフォルトの描画サイズで、エスケープシーケンスで cx, cy が指定されていない場合の値です。
Id で指定したイメージが既に登録済みの場合は、旧イメージの登録は解除されます。

戻り値 ： TRUE　- 成功
FALSE - 失敗

22.1.34. 描画イメージの登録解除 (AjcTxoUnregistImage)

形　式 ： BOOL　AjcTipTextUnregistImage (HAJCTXO hTxo, UI id);

引　数 ： hTxo　 - インスタンス・ハンドル
id　　- 登録ＩＤ（０～、 -1：全て登録解除）

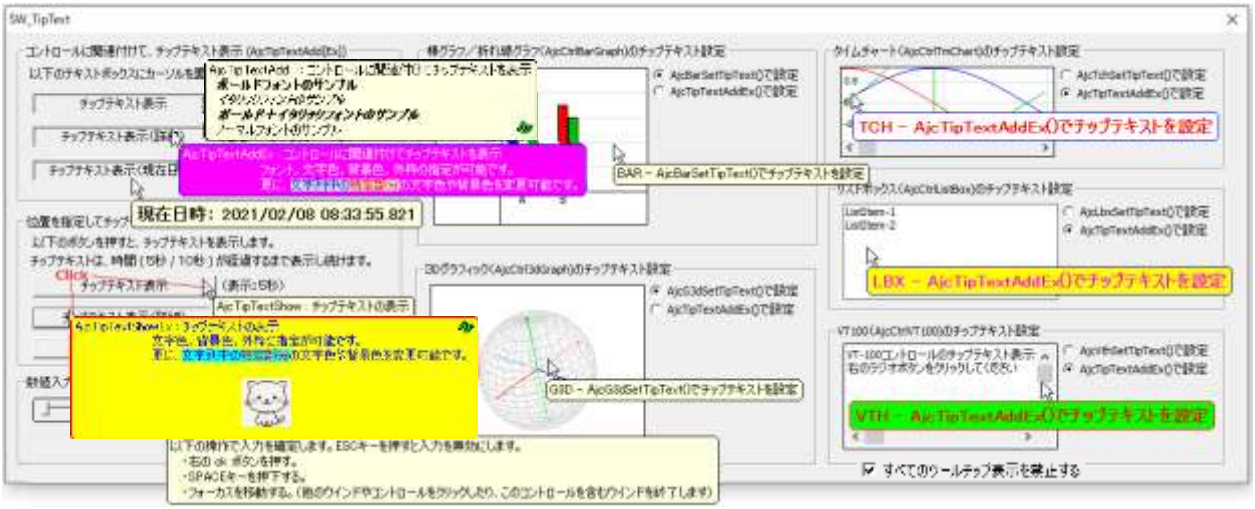
説　明 ： エスケープシーケンス（”¥x1B[*id*・・・z”）で描画するイメージの登録を解除します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

22.1.35. SW_TipText（チップテキストの表示サンプル）

22.2. ツールチップテキストの表示サンプルプログラムです。サンプルプログラム

プログラムの外観は、以下の通りです。(下図では複数のツールチップが表示されていますが、実際にはいずれか１つのツールチップが表示されます)




```
 1 : //
 2 : //  SW_TipText.c
 3 : //
 4 : #include    <AjrCstXX.h>
 5 : #include    <tchar.h>
 6 : #include    "resource.h"
 7 :
 8 : #define BLACK   RGB(   0    0,    0)      //  0: 黒
 9 : #define RED     RGB( 255,    0,    0)      //  1: 赤
10 : #define GREEN   RGB(   0,  255,    0)      //  2: 緑
11 : #define YELLOW  RGB( 255,  255,    0)      //  3: 黄
12 : #define BLUE    RGB(   0,    0,  255)      //  4: 青
13 : #define MAGENTA RGB( 255,    0,  255)      //  5: 紫
14 : #define SKYBLUE RGB(   0,  255,  255)      //  6: 水色
15 : #define WHITE   RGB( 255,  255,  255)      //  7: 白
16 :
17 : //-------------------------------------------------------------------------------------------------------------//
18 : //  ワーク                                                                                                     //
19 : //-------------------------------------------------------------------------------------------------------------//
20 : HINSTANCE       hInst;                          //  ＤＬＬインスタンスハンドル
21 : HWND            hDlgMain;                       //  ダイアログボックスハンドル
22 : HFONT           hMyFont;
23 : HICON           hIco;
24 : HBITMAP         hBmp;
25 :
26 : //-------------------------------------------------------------------------------------------------------------//
27 : //  内部サブ関数                                                                                               //
28 : //-------------------------------------------------------------------------------------------------------------//
29 : AJC_DLGPROC_DEF(Main);
30 :
31 : //=============================================================================================================//
32 : //                                                                                                             //
33 : //  ＷｉｎＭａｉｎ                                                                                            //
34 : //                                                                                                             //
35 : //=============================================================================================================//
36 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
37 : {
38 :     MSG     msg;
39 :
40 :     hInst = hInstance;
41 :
42 :     //----- メイン・ダイアログオープン ----------------//
43 :     hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_DLGMAIN), NULL, AJC_DLGPROC_NAME(Main));
44 :     ShowWindow(hDlgMain, SW_SHOW);
45 :
46 :     //----- メッセージループ --------------------------//
```

```
47 :     while (GetMessage(&msg, NULL, 0, 0)) {
48 :         do {
49 :             if (IsDialogMessage(hDlgMain, &msg)) break;
50 :             TranslateMessage(&msg);
51 :             DispatchMessage (&msg);
52 :         } while (0);
53 :     }
54 :
55 :     return (int)msg.wParam ;
56 : }
57 : //=============================================================================================================//
58 : //                                                                                                             //
59 : //  ダイアログ・プロシージャ                                                                                   //
60 : //                                                                                                             //
61 : //=============================================================================================================//
62 : //  カーソル位置チェック用コールバック (カーソルがテキストボックスの右半分の位置にある場合に TRUE を返す)
63 : static  BOOL CALLBACK cbChkCursor(HWND hCtrl, LPPOINT pPt, UX cbp)
64 : {
65 :     BOOL    rc = FALSE;
66 :     RECT    rect;
67 :
68 :     GetClientRect(hCtrl, &rect);
69 :     rect.left = (rect.left + rect.right) / 2;
70 :     if (pPt->x >= rect.left && pPt->x < rect.right && pPt->y >= rect.top && pPt->y < rect.bottom) {
71 :         rc = TRUE;
72 :     }
73 :     return rc;
74 : }
75 :
76 : //  チップテキスト取得用コールバック
77 : static  C_UTP CALLBACK cbGetText(HWND hCtrl, UTP pBuf, UI lBuf, UX cbp)
78 : {
79 :     SYSTEMTIME      lt;
80 :
81 :     GetLocalTime(&lt);
82 :     AjcSnPrintF(pBuf, lBuf, TEXT("現在日時： %4d/%02d/%02d %02d:%02d:%02d.%03d"),
83 :                             lt.wYear, lt.wMonth, lt.wDay, lt.wHour, lt.wMinute, lt.wSecond, lt.wMilliseconds);
84 :     return pBuf;
85 : }
86 :
87 : //----- ダイアログ初期化 ----------------------------------------------------------------------------------------//
88 : AJC_DLGPROC(Main, WM_INITDIALOG     )
89 : {
90 :     double  t;
91 :     double  rdat[3];
92 :     LOGFONT lf;
93 :     UTP     pTxt1 = TEXT("AjcTipTextAdd   : コントロールに関連付けてチップテキストを表示¥n")
94 :                     TEXT("  ¥x1B[T ボールドフォントのサンプル¥x1B[t¥n")
95 :                     TEXT("  ¥x1B[I イタリックフォントのサンプル¥x1B[i¥n")
96 :                     TEXT("  ¥x1B[T¥x1B[I ボールド＋イタリックフォントのサンプル¥x1B[N¥n")
97 :                     TEXT("  ノーマルフォントのサンプル")
98 :                     TEXT("¥x1B[0;-20;-20;16;16z");          //  右下にアイコン表示
99 :
100 :    UTP     pTxt2 = TEXT("AjcTipTextAddEx : コントロールに関連付けてチップテキストを表示¥x1B[50L¥n")
101 :                    TEXT("                    フォント，文字色，背景色，外枠の指定が可能です。¥n")
102 :                    TEXT("               更に、¥x1B[46m¥x1B[34m 文字列中の¥x1B[31m 特定部分¥x1B[0m の文字色や背景色を変更可能です。"));
103 :
104 :    //  アイコン，ビットマップ読み出し
105 :    hIco = (HICON  )LoadImage(hInst, MAKEINTRESOURCE(IDI_ICO), IMAGE_ICON  , 16, 16, LR_DEFAULTCOLOR);
106 :    hBmp = (HBITMAP)LoadImage(hInst, MAKEINTRESOURCE(IDB_BMP), IMAGE_BITMAP, 48, 48, LR_DEFAULTCOLOR);
107 :    //  アイコン，ビットマップ登録
108 :    AjcTipTextRegistIcon  (hIco, 0, 0, 0, 0, 0);
109 :    AjcTipTextRegistBitmap(hBmp, 1, 0, 0, 0, 0);
110 :
111 :    //  ツールチップ表示位置モード設定
112 : //  AjcTipTextSetTipPosMode(AJCTIP_POS_UNDER_CURSOR);
113 :
114 :    //  フォント生成
115 :    lf.lfHeight=16;         lf.lfItalic=0;          lf.lfClipPrecision=2;
116 :    lf.lfWidth=0;           lf.lfUnderline=0;       lf.lfQuality=1;
117 :    lf.lfEscapement=0;      lf.lfStrikeOut=0;       lf.lfPitchAndFamily=VARIABLE_PITCH;
118 :    lf.lfOrientation=0;     lf.lfCharSet=128;       MAjcStrCpy(lf.lfFaceName, AJCTSIZE(lf.lfFaceName), TEXT("MS UI Gothic"));
119 :    lf.lfWeight=FW_BOLD;    lf.lfOutPrecision=3;
120 :    hMyFont = CreateFontIndirect(&lf);
121 :
122 :    //  ラジオボタン初期化
123 :    AjcSetDlgItemChk(hDlg, IDC_RBT_INP0, TRUE);
124 :    AjcSetDlgItemChk(hDlg, IDC_RBT_BAR1, TRUE);
125 :    AjcSetDlgItemChk(hDlg, IDC_RBT_G3D1, TRUE);
126 :    AjcSetDlgItemChk(hDlg, IDC_RBT_TCH1, TRUE);
```

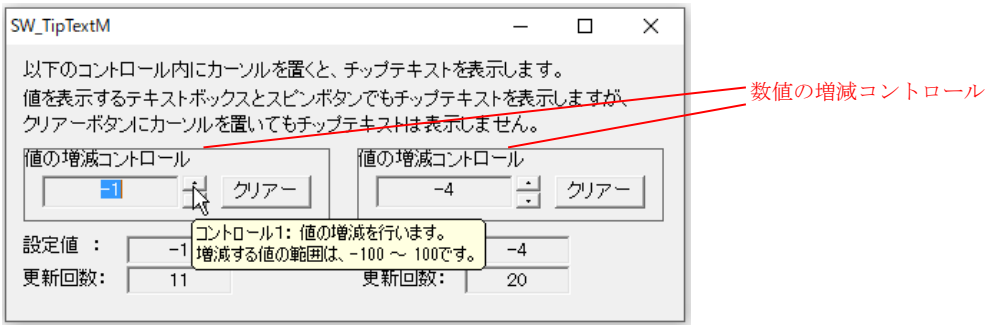
```
127 :     AjcSetDlgItemChk(hDlg, IDC_RBT_LBX1, TRUE);
128 :     AjcSetDlgItemChk(hDlg, IDC_RBT_VTH1, TRUE);
129 :
130 :     //  設定値ロード
131 :     AjcLoadAllControlSettings(hDlg, TEXT("Settings"), AJCCTL_SELACT_ALL | AJCCTL_SELACT_NTCALL);
132 :
133 :     //  ダミーデータ表示
134 :     for (t = 0; t < 200; t++) {
135 :         rdat[0] = AjcSin(t); rdat[1] = AjcCos(t); rdat[2] = AjcTan(t);
136 :         AjcTchPutRealData(GetDlgItem(hDlg, IDC_TCH), rdat);
137 :     }
138 :     rdat[0] = 50; rdat[1] = 20; rdat[2] = 80;
139 :     AjcBarPutRealData(GetDlgItem(hDlg, IDC_BAR), rdat, TEXT("A"));
140 :     rdat[0] = 30; rdat[1] = 70; rdat[2] = 60;
141 :     AjcBarPutRealData(GetDlgItem(hDlg, IDC_BAR), rdat, TEXT("B"));
142 :     AjcLbxInsertString(GetDlgItem(hDlg, IDC_LBX), -1, TEXT("ListItem-1"));
143 :     AjcLbxInsertString(GetDlgItem(hDlg, IDC_LBX), -1, TEXT("ListItem-2"));
144 :     AjcVthPutText(GetDlgItem(hDlg, IDC_VTH), TEXT("VT-100 コントロールのチップテキスト表示テスト¥n"), -1);
145 :     AjcVthPutText(GetDlgItem(hDlg, IDC_VTH), TEXT("右のラジオボタンをクリックしてください¥n"), -1);
146 :
147 :     //  パレット６をライトグレーに設定
148 :     AjcTipTextSetPalette(6, RGB(224, 224, 224));
149 :
150 :     //  テキストボックスのカーソルを通常の矢印とする
151 :     MAjcSetClassLong(GetDlgItem(hDlg, IDC_TXT_TIP1), GCL_HCURSOR, (UX)LoadCursor(NULL, IDC_ARROW));
152 :
153 :     //  テキストボックスタイトル
154 :     AjcSetDlgItemStr(hDlg, IDC_TXT_TIP1, TEXT("チップテキスト表示"));
155 :     AjcSetDlgItemStr(hDlg, IDC_TXT_TIP2, TEXT("チップテキスト表示 (詳細) "));
156 :     AjcSetDlgItemStr(hDlg, IDC_TXT_TIP3, TEXT("チップテキスト表示 (現在日時) "));
157 :
158 :     //  コントロールにチップテキストを関連付ける
159 :     AjcTipTextAdd          (GetDlgItem(hDlg, IDC_TXT_TIP1), pTxt1);
160 :     AjcTipTextAddEx        (GetDlgItem(hDlg, IDC_TXT_TIP2), pTxt2, 2000, 10000, NULL, WHITE, MAGENTA, BLUE);
161 :     AjcTipTextSetShowAlways(GetDlgItem(hDlg, IDC_TXT_TIP2), FALSE); //  非アクティブ時は表示しない
162 :
163 :     //  コールバックによる状況依存のツールチップ設定 (現在日時を独自フォントで表示)
164 :     AjcTipTextAddEx(GetDlgItem(hDlg, IDC_TXT_TIP3), NULL ,    0,  3000, hMyFont, -1, -1, -1);
165 :     AjcTipTextSetCallBack(GetDlgItem(hDlg, IDC_TXT_TIP3), (UX)0, cbChkCursor, cbGetText);
166 :
167 :     return TRUE;
168 : }
169 : //----- ウインド破棄 ----------------------------------------------------------------------------------------//
170 : AJC_DLGPROC(Main, WM_DESTROY        )
171 : {
172 :     //  アイコン, ビットマップ登録解除
173 :     AjcTipTextUnregistImage(-1);
174 :     //  アイコン破棄
175 :     DeleteObject(hIco);
176 :     DeleteObject(hBmp);
177 :     //  設定値セーブ
178 :     AjcSaveAllControlSettings(hDlg);
179 :
180 :     DeleteObject(hMyFont);
181 :     PostQuitMessage(0);
182 :     return TRUE;
183 : }
184 : //----- チップテキスト表示 ----------------------------------------------------------------------------------//
185 : AJC_DLGPROC(Main, IDC_CMD_SHOW1     )
186 : {
187 :     POINT   pt;
188 :     UTP     pTxt = TEXT("AjcTipTextShow : チップテキストの表示");
189 :
190 :     GetCursorPos(&pt);
191 :     AjcTipTextShow(pt.x + 16,           //  チップテキストの表示位置
192 :                    pt.y + 16,           //  ・
193 :                    pTxt,                //  表示テキスト
194 :                    5000,                //  表示時間 (ms)
195 :                    NULL);               //  フォントハンドル (NULL:デフォルトフォント)
196 :
197 :     return TRUE;
198 : }
199 : //----- チップテキスト表示 (詳細) ---------------------------------------------------------------------------//
200 : AJC_DLGPROC(Main, IDC_CMD_SHOW2     )
201 : {
202 :     POINT   pt;
203 :     UTP     pTxt =  TEXT("AjcTipTextShowEx : チップテキストの表示¥n")
204 :                     TEXT("                   文字色, 背景色, 外枠の指定が可能です。¥n"
205 :                     TEXT("                   更に、¥x1B[46m 文字列中の¥x1B[31m 特定部分¥x1B[0m の文字色や背景色を変更可能です。
        ")
```

```
206 :                     TEXT("¥x1B[0;-20;2;16;16z")     //  (右端-20, 2) の位置にアイコン表示
207 :                     TEXT("¥x1B[1;168;60;48;48z")); //  (168, 60)の位置にビットマップ表示
208 :
209 :     GetCursorPos(&pt);
210 :     AjcTipTextShowEx(pt.x + 16,             //  チップテキストの表示位置
211 :                      pt.y + 16,             //      ・
212 :                      400,                   //      テキストの幅   (400 ピクセル)
213 :                      120,                   //      テキストの高さ (120 ピクセル)
214 :                      pTxt,                  //      表示テキスト
215 :                      10000,                 //      表示時間 (ms)
216 :                      NULL,                  //      フォントハンドル (NULL:デフォルトフォント)
217 :                      BLUE,                  //      文字色 (青)
218 :                      YELLOW,                //      背景色 (黄)
219 :                      RED);                  //      外枠   (赤)
220 :     AjcTipTextMoveCursor(AJCTIP_CM_LD);     //  カーソルをチップテキストの左下隅へ移動
221 :
222 :     return TRUE;
223 : }
224 : //----- チップテキスト非表示 ----------------------------------------------------------------------------------//
225 : AJC_DLGPROC(Main, IDC_CMD_HIDE      )
226 : {
227 :     AjcTipTextHide();
228 :     return TRUE;
229 : }
230 : //----- AjcCtrlTmChart - AjcTchSetTipText()で設定 -------------------------------------------------------------//
231 : AJC_DLGPROC(Main, IDC_RBT_TCH1      )
232 : {
233 :     AjcTchSetTipText(GetDlgItem(hDlg, IDC_TCH), TEXT("TCH - AjcTchSetTipText()でチップテキストを設定"));
234 :     return TRUE;
235 : }
236 : //----- AjcCtrlTmChart - AjcTipTextAddEx()で設定 --------------------------------------------------------------//
237 : AJC_DLGPROC(Main, IDC_RBT_TCH2      )
238 : {
239 :     AjcTipTextAddEx(GetDlgItem(hDlg, IDC_TCH), TEXT("TCH - AjcTipTextAddEx()でチップテキストを設定"), -1, -1, hMyFont, RED, WHITE, BLUE);
240 :     return TRUE;
241 : }
242 : //----- AjcCtrlInpVal - デフォルトのツールチップ --------------------------------------------------------------//
243 : AJC_DLGPROC(Main, IDC_RBT_INP0      )
244 : {
245 :     AjcIvEnaDefTipText(GetDlgItem(hDlg, IDC_INP), TRUE);
246 :     return TRUE;
247 : }
248 : //----- AjcCtrlInpVal - AjcIvSetTipText()で設定 ---------------------------------------------------------------//
249 : AJC_DLGPROC(Main, IDC_RBT_INP1      )
250 : {
251 :     AjcIvEnaDefTipText(GetDlgItem(hDlg, IDC_INP), FALSE);
252 :     AjcIvSetTipText(GetDlgItem(hDlg, IDC_INP), TEXT("INP - AjcIvSetTipText()でチップテキストを設定"));
253 :     return TRUE;
254 : }
255 : //----- AjcCtrlInpVal - AjcTchSetTipText()で設定 --------------------------------------------------------------//
256 : AJC_DLGPROC(Main, IDC_RBT_INP2      )
257 : {
258 :     AjcIvEnaDefTipText(GetDlgItem(hDlg, IDC_INP), FALSE);
259 :     AjcTipTextAddEx(GetDlgItem(hDlg, IDC_INP), TEXT("INP - AjcTipTextAddEx()でチップテキストを設定"), -1, -1, hMyFont, MAGENTA, -1, BLUE);
260 :     return TRUE;
261 : }
262 : //----- AjcCtrlBarGraph - AjcTchSetTipText()で設定 ------------------------------------------------------------//
263 : AJC_DLGPROC(Main, IDC_RBT_BAR1      )
264 : {
265 :     AjcBarSetTipText(GetDlgItem(hDlg, IDC_BAR), TEXT("BAR - AjcBarSetTipText()でチップテキストを設定"));
266 :     return TRUE;
267 : }
268 : //----- AjcCtrlBarGraph - AjcTipTextAddEx()で設定 -------------------------------------------------------------//
269 : AJC_DLGPROC(Main, IDC_RBT_BAR2      )
270 : {
271 :     AjcTipTextAddEx(GetDlgItem(hDlg, IDC_BAR), TEXT("BAR - AjcTipTextAddEx()でチップテキストを設定"), -1, -1, hMyFont, YELLOW, MAGENTA, BLUE);
272 :     return TRUE;
273 : }
274 : //----- AjcCtrl3dGraph - AjcTchSetTipText()で設定 -------------------------------------------------------------//
275 : AJC_DLGPROC(Main, IDC_RBT_G3D1      )
276 : {
277 :     AjcG3dSetTipText(GetDlgItem(hDlg, IDC_G3D), TEXT("G3D - AjcG3dSetTipText()でチップテキストを設定"));
278 :     return TRUE;
279 : }
280 : //----- AjcCtrl3dGraph - AjcTipTextAddEx()で設定 --------------------------------------------------------------//
281 : AJC_DLGPROC(Main, IDC_RBT_G3D2      )
282 : {
283 :     AjcTipTextAddEx(GetDlgItem(hDlg, IDC_G3D), TEXT("G3D - AjcTipTextAddEx()でチップテキストを設定"), -1, -1, hMyFont, MAGENTA, YELLOW, BLUE);
284 :     return TRUE;
285 : }
```

```
286 : //----- AjcCtrlListBox - AjcLbxSetTipText()で設定 ----------------------------------------------------------//
287 : AJC_DLGPROC(Main, IDC_RBT_LBX1      )
288 : {
289 :     AjcLbxSetTipText(GetDlgItem(hDlg, IDC_LBX), TEXT("LBX - AjcLbxSetTipText()でチップテキストを設定"));
290 :     return TRUE;
291 : }
292 : //----- AjcCtrlListBox - AjcTipTextAddEx()で設定 ----------------------------------------------------------//
293 : AJC_DLGPROC(Main, IDC_RBT_LBX2      )
294 : {
295 :     AjcTipTextAddEx(GetDlgItem(hDlg, IDC_LBX), TEXT("LBX - AjcTipTextAddEx()でチップテキストを設定"), -1, -1, hMyFont, MAGENTA, YELLOW, BLUE);
296 :     return TRUE;
297 : }
298 : //----- AjcCtrlVT100 - AjcVthSetTipText()で設定 -----------------------------------------------------------//
299 : AJC_DLGPROC(Main, IDC_RBT_VTH1      )
300 : {
301 :     AjcVthSetTipText(GetDlgItem(hDlg, IDC_VTH), TEXT("VTH - AjcVthSetTipText()でチップテキストを設定"));
302 :     return TRUE;
303 : }
304 : //----- AjcCtrlVT100 - AjcTipTextAddEx()で設定 -----------------------------------------------------------//
305 : AJC_DLGPROC(Main, IDC_RBT_VTH2      )
306 : {
307 :     AjcTipTextAddEx(GetDlgItem(hDlg, IDC_VTH), TEXT("VTH - AjcTipTextAddEx()でチップテキストを設定"), -1, -1, hMyFont, RED, GREEN, RED);
308 :     return TRUE;
309 : }
310 : //----- 全てのツールチップ表示禁止 ------------------------------------------------------------------------//
311 : AJC_DLGPROC(Main, IDC_CHK_DISABLE   )
312 : {
313 :     AjcTipTextEnableAll(!AjcGetDlgItemChk(hDlg, IDC_CHK_DISABLE));
314 :     return TRUE;
315 : }
316 : //----- キャンセル ----------------------------------------------------------------------------------------//
317 : AJC_DLGPROC(Main, IDCANCEL          )
318 : {
319 :     DestroyWindow(hDlg);
320 :     return TRUE;
321 : }
322 : //----------------------------------------------------------------------------------------------------------//
323 : AJC_DLGMAP_DEF(Main)
324 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG     )
325 :     AJC_DLGMAP_MSG(Main, WM_DESTROY        )
326 :
327 :     AJC_DLGMAP_CMD(Main, IDC_CMD_SHOW1     )
328 :     AJC_DLGMAP_CMD(Main, IDC_CMD_SHOW2     )
329 :     AJC_DLGMAP_CMD(Main, IDC_CMD_HIDE      )
330 :
331 :     AJC_DLGMAP_CMD(Main, IDC_RBT_TCH1      )
332 :     AJC_DLGMAP_CMD(Main, IDC_RBT_TCH2      )
333 :
334 :     AJC_DLGMAP_CMD(Main, IDC_RBT_INP0      )
335 :     AJC_DLGMAP_CMD(Main, IDC_RBT_INP1      )
336 :     AJC_DLGMAP_CMD(Main, IDC_RBT_INP2      )
337 :
338 :     AJC_DLGMAP_CMD(Main, IDC_RBT_BAR1      )
339 :     AJC_DLGMAP_CMD(Main, IDC_RBT_BAR2      )
340 :
341 :     AJC_DLGMAP_CMD(Main, IDC_RBT_G3D1      )
342 :     AJC_DLGMAP_CMD(Main, IDC_RBT_G3D2      )
343 :
344 :     AJC_DLGMAP_CMD(Main, IDC_RBT_LBX1      )
345 :     AJC_DLGMAP_CMD(Main, IDC_RBT_LBX2      )
346 :
347 :     AJC_DLGMAP_CMD(Main, IDC_RBT_VTH1      )
348 :     AJC_DLGMAP_CMD(Main, IDC_RBT_VTH2      )
349 :
350 :     AJC_DLGMAP_CMD(Main, IDC_CHK_DISABLE   )
351 :
352 :     AJC_DLGMAP_CMD(Main, IDCANCEL          )
353 : AJC_DLGMAP_END
354 :
```

22.2.1. SW_TipTextM（子孫構造を持つコントロールのツールチップ制御）

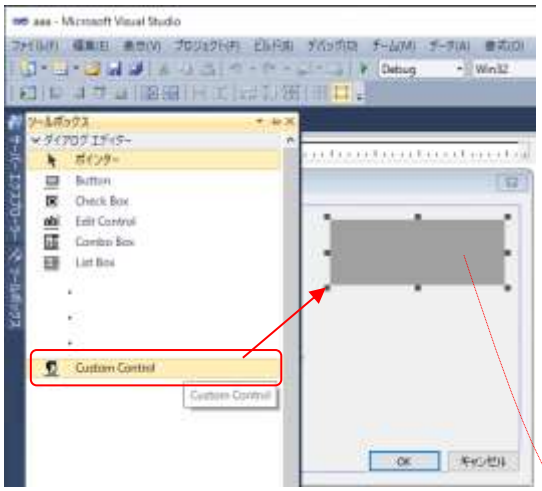
以下のサンプルプログラムでは、数値の増減とクリアーを行うコントロールを例に、コントロールの実装とツールチップの表示例を示します。
数値の増減(スピンボタン)やクリアーを行うと、コントロールの下に設定値と更新回数を表示します。

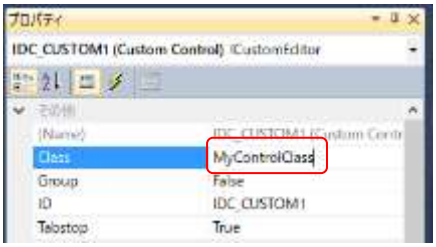


このサンプルプログラムは、値の増減を行うコントロールの実装部分と、このコントロールを利用する部分に分かれます。
数値の増減コントロールは、以下のクラス名、ＡＰＩとイベントを公開しているものとします。

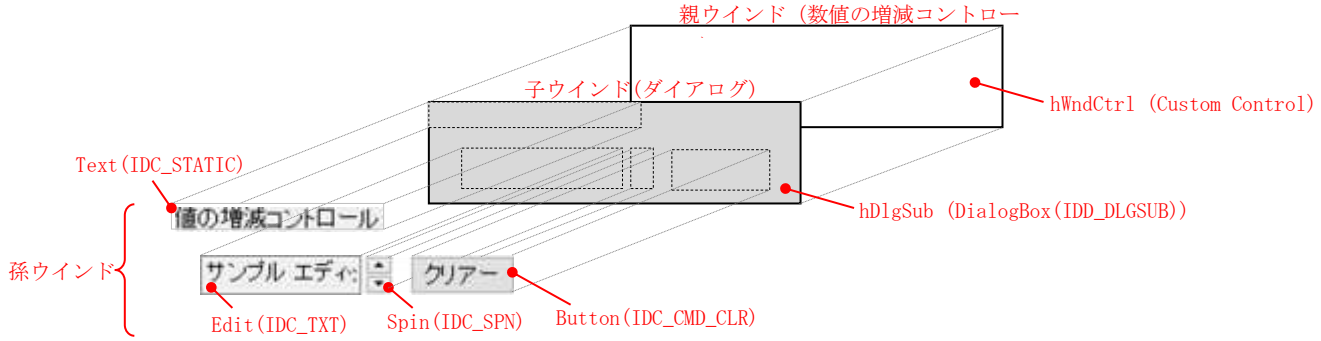
クラス名	ＡＰＩ	イベント
・MyControlClass	・MyControlInit() ---- 初期設定	・NTC_UPDATED --- 数値の更新を WM_COMMAND で通知
	・MyControlEnd() ---- 後処理	LOWORD(wParam) ： コントロールＩＤ
	・MyControlGetValue() --- 設定値の取得	HIWORD(wParam) ： NTC_UPDATED
	・MyControlGetCount() --- 更新回数の取得	lParam ： コントロールのハンドル

このコントロールを利用するには、リソースエディタで、ツールボックスから「Custom Control」をユーザのダイアログに配置し、Class プロパティに「MyControlClass」を設定します。





数値の増減コントロールは、以下のような子孫ウインド構造を持ちます。



このコントロールを使用するユーザは、ツールチップを設定する場合、hWndCtrl (Custom Control)を指定します。
しかし、hWndCtrl (Custom Control)は全体を hDlgSub (DialogBox(IDD_DLGSUB))で覆われていることから (WM_MOUSEMOVE 等の) マウスイベントが発生しない為、ツールチップが表示されません。

そこで、コントロールでは、AJCTIPCTRL_SETHWND マクロでツールチップの表示コントロールを hDlgSub (DialogBox(IDD_DLGSUB))上で行うように指示しておきます。
これで hDlgSub (DialogBox(IDD_DLGSUB))上にカーソルを置くとツールチップが表示されるようになります。

但し、これでも、hDlgSub (DialogBox(IDD_DLGSUB))上の子ウインドである Edit(IDC_TXT)や Spin (IDC_SPN)上にカーソルが移動するとツールチップは消えてしまいます。

最後に、AJCTIP_SETFRIEND マクロで Edit(IDC_TXT)と Spin (IDC_SPN)を指定することにより、これらのコントロール上でもツールチップが表示されるようになります。
Button(IDC_CMD_CLR)は AJCTIP_SETFRIEND マクロで指定していない為、クリアーボタン上にカーソルを移動するとツールチップは消えます。(AJCTIP_SETFRIEND マクロの効果を確認する意味で、あえて指定していません)

尚、Edit(IDC_TXT)と Spin (IDC_SPN)では、自身のマウスイベント(WM_MOUSEMOVE, WM_LBUTTONDOWN, WM_RBUTTONDOWN, WM_MBUTTONDOWN)を hDlgSub (DialogBox(IDD_DLGSUB))へパスする必要があります。
このサンプルプログラムでは、これら２つのコントロールをサブクラス化してマウスイベントを hDlgSub (DialogBox(IDD_DLGSUB))へパスしています。

```
 1 : //
 2 : //  SW_TipTextM.c
 3 : //
 4 :
 5 : #include    <AjrCstXX.h>
 6 : #include    <math.h>
 7 : #include    <tchar.h>
 8 : #include    "resource.h"
 9 :
10 :
11 : //-------------------------------------------------------------------------------------------------------------//
12 : //  コントロールの公開情報                                                                                     //
13 : //-------------------------------------------------------------------------------------------------------------//
14 : //  クラス名
15 : #define     CMYCTRL     TEXT("MyControlClass")
16 : //  ＡＰＩ
17 : extern  VO      MyControlInit(VO);
18 : extern  VO      MyControlEnd(VO);
19 : extern  int     MyControlGetValue(HWND hCtrl);
20 : extern  int     MyControlGetCount(HWND hCtrl);
21 : //  通知コード (WM_COMMAND の HIWORD(wParam))
22 : #define NTC_UPDATED     0x1234
23 :
24 : //=============================================================================================================//
25 : //                                                                                                             //
26 : //  メインプログラム                                                                                           //
27 : //                                                                                                             //
28 : //=============================================================================================================//
29 : //  ワーク
30 : static  HINSTANCE       hInst;
31 : static  HWND            hDlgMain;
32 :
33 : //  メインダイアログプロシージャ プロトタイプ
34 : AJC_DLGPROC_DEF(Main);
35 :
36 : //-------------------------------------------------------------------------------------------------------------//
37 : //                                                                                                             //
38 : //  メイン・プログラム                                                                                         //
39 : //                                                                                                             //
40 : //-------------------------------------------------------------------------------------------------------------//
41 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
42 : {
43 :     MSG     msg;
44 :
45 :     hInst = hInstance;
46 :
47 :     //  コントロール初期化
48 :     MyControlInit();
49 :
50 :     //  メイン・ダイアログ生成
51 :     hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_DLGMAIN), NULL, AJC_DLGPROC_NAME(Main));
52 :     //  ダイアログ表示
```

```
 53 :     ShowWindow(hDlgMain, SW_SHOW);
 54 :
 55 :     //----- メッセージループ --------------------------//
 56 :     while (GetMessage(&msg, NULL, 0, 0)) {
 57 :         do {
 58 :             if (IsDialogMessage(hDlgMain, &msg)) break;
 59 :             TranslateMessage(&msg);
 60 :             DispatchMessage (&msg);
 61 :         } while (0);
 62 :     }
 63 :
 64 :     //  コントロール後処理
 65 :     MyControlEnd();
 66 :
 67 :     return (int)msg.wParam ;
 68 : }
 69 : //-----------------------------------------------------------------------------------------------------------------//
 70 : //                                                                                                                 //
 71 : //  メイン・ダイアログ・プロシージャ                                                                               //
 72 : //                                                                                                                 //
 73 : //-----------------------------------------------------------------------------------------------------------------//
 74 : //----- ダイアログ初期化 --------------------------------------------------------------------------------------//
 75 : AJC_DLGPROC(Main, WM_INITDIALOG     )
 76 : {
 77 :     hDlgMain = hDlg;
 78 :
 79 :     //  コントロールのチップテキスト設定
 80 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_MYCTRL1), TEXT("コントロール１ : 値の増減を行います。¥n"
 81 :                                                  TEXT("増減する値の範囲は、-100 ～ 100 です。")));
 82 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_MYCTRL2), TEXT("コントロール２ : 値の増減を行います。¥n"
 83 :                                                  TEXT("増減する値の範囲は、-100 ～ 100 です。")));
 84 :
 85 :     return TRUE;
 86 : }
 87 : //----- ウインド破棄 ------------------------------------------------------------------------------------------//
 88 : AJC_DLGPROC(Main, WM_DESTROY        )
 89 : {
 90 :     //  プログラム終了
 91 :     PostQuitMessage(0);
 92 :     return TRUE;
 93 : }
 94 : //----- 値の増減コントロールからの通知（１）------------------------------------------------------------------//
 95 : AJC_DLGPROC(Main, IDC_MYCTRL1       )
 96 : {
 97 :     if (HIWORD(wParam) == NTC_UPDATED) {
 98 :         AjcSetDlgItemSInt(hDlg, IDC_TXT_VAL1, MyControlGetValue(GetDlgItem(hDlg, IDC_MYCTRL1)));
 99 :         AjcSetDlgItemSInt(hDlg, IDC_TXT_CNT1, MyControlGetCount(GetDlgItem(hDlg, IDC_MYCTRL1)));
100 :     }
101 :     return TRUE;
102 : }
103 : //----- 値の増減コントロールからの通知（２）------------------------------------------------------------------//
104 : AJC_DLGPROC(Main, IDC_MYCTRL2       )
105 : {
106 :     if (HIWORD(wParam) == NTC_UPDATED) {
107 :         AjcSetDlgItemSInt(hDlg, IDC_TXT_VAL2, MyControlGetValue(GetDlgItem(hDlg, IDC_MYCTRL2)));
108 :         AjcSetDlgItemSInt(hDlg, IDC_TXT_CNT2, MyControlGetCount(GetDlgItem(hDlg, IDC_MYCTRL2)));
109 :     }
110 :     return TRUE;
111 : }
112 : //-----  「Cancel」ボタン ------------------------------------------------------------------------------------//
113 : AJC_DLGPROC(Main, IDCANCEL          )
114 : {
115 :     DestroyWindow(hDlg);
116 :     return TRUE;
117 : }
118 : //-----------------------------------------------------------------------------------------------------------------//
119 : AJC_DLGMAP_DEF(Main)
120 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG     )
121 :     AJC_DLGMAP_MSG(Main, WM_DESTROY        )
122 :
123 :     AJC_DLGMAP_CMD(Main, IDC_MYCTRL1       )
124 :     AJC_DLGMAP_CMD(Main, IDC_MYCTRL2       )
125 :     AJC_DLGMAP_CMD(Main, IDCANCEL          )
126 : AJC_DLGMAP_END
127 :
128 : //=================================================================================================================//
129 : //                                                                                                                 //
130 : //  数値の増減コントロール                                                                                         //
131 : //                                                                                                                 //
132 : //=================================================================================================================//
133 : //  コントロールのクラスアトム
134 : static  ATOM    AtmMyCtrl = 0;
135 :
136 : //  コントロールのワーク
137 : typedef struct {
138 :     int     val;        //  現在値
139 :     int     cnt;        //  更新回数
140 :     HWND    hWndCtrl;   //  コントロールのメインウインド
141 :     HWND    hDlgSub;    //  サブダイアログハンドル
142 : } WRKMYCTRL, *PWRKMYCTRL;
```

```
143 : typedef const WRKMYCTRL *PCWRKMYCTRL;
144 :
145 : //  ウインドプロシージャ・プロトタイプ
146 : AJC_WNDPROC_DEF(MyControl);
147 : AJC_DLGPROC_DEF(SubDialog);
148 : AJC_WNDPROC_DEF(SubClass );
149 :
150 : //-----------------------------------------------------------------------------------------------------------------//
151 : //  コントロール初期化                                                                                             //
152 : //-----------------------------------------------------------------------------------------------------------------//
153 : VO      MyControlInit(VO)
154 : {
155 :     WNDCLASS    wndclass;
156 :
157 :     //  コントロールクラス生成
158 :     wndclass.style          = CS_GLOBALCLASS;
159 :     wndclass.lpfnWndProc    = AJC_WNDPROC_NAME(MyControl);
160 :     wndclass.cbClsExtra     = 0;
161 :     wndclass.cbWndExtra     = sizeof(PWRKMYCTRL);
162 :     wndclass.hInstance      = hInst;
163 :     wndclass.hIcon          = NULL;
164 :     wndclass.hCursor        = LoadCursor(NULL, IDC_ARROW);
165 :     wndclass.hbrBackground  = (HBRUSH)GetStockObject(WHITE_BRUSH);
166 :     wndclass.lpszMenuName   = NULL;
167 :     wndclass.lpszClassName  = CMYCTRL;
168 :     AtmMyCtrl = RegisterClass(&wndclass);
169 : }
170 : //-----------------------------------------------------------------------------------------------------------------//
171 : //  コントロール後処理                                                                                             //
172 : //-----------------------------------------------------------------------------------------------------------------//
173 : VO      MyControlEnd(VO)
174 : {
175 :     //  コントロールクラス破棄
176 :     UnregisterClass((UTP)AtmMyCtrl, hInst);
177 : }
178 :
179 : //-----------------------------------------------------------------------------------------------------------------//
180 : //  コントロールから数値取得                                                                                       //
181 : //-----------------------------------------------------------------------------------------------------------------//
182 : int     MyControlGetValue(HWND hCtrl)
183 : {
184 :     PWRKMYCTRL      pW = (PWRKMYCTRL)MAjcGetWindowLong(hCtrl, 0);
185 :     return pW->val;
186 : }
187 :
188 : //-----------------------------------------------------------------------------------------------------------------//
189 : //  コントロールから更新回数取得                                                                                   //
190 : //-----------------------------------------------------------------------------------------------------------------//
191 : int     MyControlGetCount(HWND hCtrl)
192 : {
193 :     PWRKMYCTRL      pW = (PWRKMYCTRL)MAjcGetWindowLong(hCtrl, 0);
194 :     return pW->cnt;
195 : }
196 :
197 : //-----------------------------------------------------------------------------------------------------------------//
198 : //                                                                                                                 //
199 : //  コントロール・ウインドプロシージャ                                                                             //
200 : //                                                                                                                 //
201 : //-----------------------------------------------------------------------------------------------------------------//
202 : //----- WM_CREATE -------------------------------------------------------------------------------------------------//
203 : AJC_WNDPROC(MyControl, WM_CREATE        )
204 : {
205 :     PWRKMYCTRL      pW = NULL;
206 :     RECT            r;
207 :
208 :     //  ワーク生成
209 :     pW = (PWRKMYCTRL)malloc(sizeof(WRKMYCTRL));
210 :     //  ウインド・エクストラ領域へワークポインタ設定
211 :     MAjcSetWindowLong(hwnd, 0, (UX)pW);
212 :     //  ワーク初期化
213 :     memset(pW, 0, sizeof(WRKMYCTRL));
214 :     pW->hWndCtrl = hwnd;
215 :     //  サブ・ダイアログ生成
216 :     pW->hDlgSub  = CreateDialogParam(hInst, MAKEINTRESOURCE(IDD_DLGSUB), hwnd, AJC_DLGPROC_NAME(SubDialog), (LPARAM)pW);
217 :     ShowWindow(pW->hDlgSub, SW_SHOW);
218 :     //  コントロールのサイズをダイアログに合わせる
219 :     GetWindowRect(pW->hDlgSub, &r);
220 :     SetWindowPos(hwnd, NULL, 0, 0, r.right - r.left, r.bottom - r.top, SWP_NOMOVE);
221 :     //  ツールチップの制御対象をサブダイアログとするように指示
222 :     AJCTIPCTRL_SETHWND(hwnd, pW->hDlgSub);
223 :     //  以下のテキストボックスとスピンボタンでもチップテキストを表示するように設定
224 :     AJCTIP_SETFRIEND(GetDlgItem(pW->hDlgSub, IDC_TXT));
225 :     AJCTIP_SETFRIEND(GetDlgItem(pW->hDlgSub, IDC_SPN));
226 :
227 :     return 0;
228 : }
229 : //----- WM_DESTROY ------------------------------------------------------------------------------------------------//
230 : AJC_WNDPROC(MyControl, WM_DESTROY   )
231 : {
232 :     PWRKMYCTRL      pW = (PWRKMYCTRL)MAjcGetWindowLong(hwnd, 0);
```

```
233 :
234 :     //  サブ・ダイアログ破棄
235 :     DestroyWindow(pW->hDlgSub);
236 :     //  ワーク破棄
237 :     free(pW);
238 :
239 :     return 0;
240 : }
241 : //-----------------------------------------------------------------------------------------------------------------//
242 : AJC_WNDMAP_DEF(MyControl)
243 :     AJC_WNDMAP_MSG(MyControl, WM_CREATE     )
244 :     AJC_WNDMAP_MSG(MyControl, WM_DESTROY    )
245 : AJC_WNDMAP_END
246 : //-----------------------------------------------------------------------------------------------------------------//
247 : //                                                                                                                 //
248 : //  コントロール・サブダイアログプロシージャ                                                                       //
249 : //                                                                                                                 //
250 : //-----------------------------------------------------------------------------------------------------------------//
251 : //----- ダイアログ初期化 ------------------------------------------------------------------------------------------//
252 : AJC_DLGPROC(SubDialog, WM_INITDIALOG    )
253 : {
254 :     PWRKMYCTRL      pW = (PWRKMYCTRL)lParam;
255 :
256 :     //  ダイアログにワークポインタを関連付ける
257 :     SetProp(hDlg, TEXT("SubDlgProp"), (HANDLE)lParam);
258 :     //  スピンボタン初期設定
259 :     AjcSetCtrlSpnRange(GetDlgItem(hDlg, IDC_SPN), -100,  100);
260 :     //  テキストボックスとスピンボタンをサブクラス化し、マウスイベントをトラップする
261 :     MAjcMmpSetSubclass(SubClass, GetDlgItem(hDlg, IDC_TXT));
262 :     MAjcMmpSetSubclass(SubClass, GetDlgItem(hDlg, IDC_SPN));
263 :     return TRUE;
264 : }
265 : //----- ダイアログ破棄 --------------------------------------------------------------------------------------------//
266 : AJC_DLGPROC(SubDialog, WM_DESTROY       )
267 : {
268 :     //  ダイアログに関連付けたワークポインタ解放
269 :     RemoveProp(hDlg, TEXT("SubDlgProp"));
270 :
271 :     return TRUE;
272 : }
273 : //----- IDC_TXT ---------------------------------------------------------------------------------------------------//
274 : AJC_DLGPROC(SubDialog, IDC_TXT          )
275 : {
276 :     PWRKMYCTRL      pW = (PWRKMYCTRL)GetProp(hDlg, TEXT("SubDlgProp"));
277 :
278 :     if (HIWORD(wParam) == EN_CHANGE) {
279 :         //  スピンボタンに「AutoBuddy」が設定されている場合「WM_INITDIALOG」より先にテキストボックスが設定される場合がある為、
280 :         //  「WM_INITDIALOG」が実行されていない場合は、処理を行わないようにする必要があります。
281 :         if (pW != NULL) {
282 :             pW->val = AjcGetDlgItemSInt(hDlg, IDC_TXT);
283 :             pW->cnt++;
284 :             SendMessage(GetParent(pW->hWndCtrl), WM_COMMAND,
285 :                                 MAKELONG(MAjcGetWindowLong(pW->hWndCtrl, GWL_ID), NTC_UPDATED), (LPARAM)pW->hWndCtrl);
286 :         }
287 :     }
288 :     return TRUE;
289 : }
290 : //----- IDC_CMD_CLR -----------------------------------------------------------------------------------------------//
291 : AJC_DLGPROC(SubDialog, IDC_CMD_CLR      )
292 : {
293 :     PWRKMYCTRL      pW = (PWRKMYCTRL)GetProp(hDlg, TEXT("SubDlgProp"));
294 :
295 :     if (HIWORD(wParam) == BN_CLICKED) {
296 :         AjcSetDlgItemSInt(hDlg, IDC_TXT, pW->val = 0);
297 :     }
298 :     return TRUE;
299 : }
300 : //-----------------------------------------------------------------------------------------------------------------//
301 : AJC_DLGMAP_DEF(SubDialog)
302 :     AJC_DLGMAP_MSG(SubDialog, WM_INITDIALOG    )
303 :     AJC_DLGMAP_MSG(SubDialog, WM_DESTROY       )
304 :     AJC_DLGMAP_CMD(SubDialog, IDC_TXT          )
305 :     AJC_DLGMAP_CMD(SubDialog, IDC_CMD_CLR      )
306 : AJC_DLGMAP_END
307 : //-----------------------------------------------------------------------------------------------------------------//
308 : //                                                                                                                 //
309 : //  サブクラス化したウインドプロシージャ（テキストボックス，スピンボタン）                                         //
310 : //                                                                                                                 //
311 : //-----------------------------------------------------------------------------------------------------------------//
312 : AJC_WNDPROC(SubClass, WM_MOUSEMOVE         )
313 : {
314 :     SendMessage(GetParent(hwnd), msg, wParam, lParam);
315 :     return MAjcMmpCallOrgWndProc(SubClass);;
316 : }
317 : //-----------------------------------------------------------------------------------------------------------------//
318 : AJC_WNDPROC(SubClass, WM_LBUTTONDOWN       )
319 : {
320 :     SendMessage(GetParent(hwnd), msg, wParam, lParam);
321 :     return MAjcMmpCallOrgWndProc(SubClass);;
322 : }
```

```
323 : //-------------------------------------------------------------------------------------------------------------//
324 : AJC_WNDPROC(SubClass, WM_MBUTTONDOWN       )
325 : {
326 :     SendMessage(GetParent(hwnd), msg, wParam, lParam);
327 :     return MAjcMmpCallOrgWndProc(SubClass);;
328 : }
329 : //-------------------------------------------------------------------------------------------------------------//
330 : AJC_WNDPROC(SubClass, WM_RBUTTONDOWN       )
331 : {
332 :     SendMessage(GetParent(hwnd), msg, wParam, lParam);
333 :     return MAjcMmpCallOrgWndProc(SubClass);;
334 : }
335 : //-------------------------------------------------------------------------------------------------------------//
336 : AJC_WNDMAP_DEF(SubClass)
337 :     AJC_WNDMAP_MSG(SubClass, WM_MOUSEMOVE  )
338 :     AJC_WNDMAP_MSG(SubClass, WM_LBUTTONDOWN)
339 :     AJC_WNDMAP_MSG(SubClass, WM_MBUTTONDOWN)
340 :     AJC_WNDMAP_MSG(SubClass, WM_RBUTTONDOWN)
341 : AJC_WNDMAP_END
342 :
```

23. コンソール入出力

コンソールアプリ用のＡＰＩ群です。

23.1. サポートＡＰＩ

コンソール入出力ＡＰＩ一覧を以下に示します。

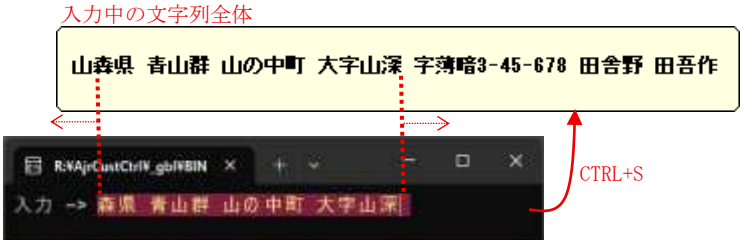
#	関 数 名	内容
1	AjcConInput AjcConInputByNtc AjcConInputEx AjcConInpSetColor	コンソールから１行入力／カラー設定
2	AjcSetStdMode	標準出力／標準エラー のモード設定
3	Ajc[Err]PutC Ajc[Err]PutS Ajc[Err]PrintF	１文字／文字列／書式文字列をコンソールへ出力
4	AjcGetConsoleScreenBufferInfo	コンソール情報取得
5	AjcSetConsoleCursor	コンソールカーソル位置設定
6	AjcGetConsoleCursor	コンソールカーソル位置取得
7	AjcSystem	コマンド実行し、出力をファイルへリダイレクト

23.1.1. コンソールから１行入力（AjcConInput[Ex] / AjcConInputByNtc）

形　式 ： BOOL　AjcConInput　(UTP pBuf, UI lBuf);
BOOL　AjcConInputByNtc(int BufSize, UX cbp, VO (CALLBACK *cbNtcArgs)(int argc, UT *argv[], C_UTP pTxt, UX cbp));
BOOL　AjcConInputEx(int FieldLength, UTP pBuf, int lBuffer, C_UTP pHelp,
UX cbp, VO (CALLBACK *cbNtcArgs)(int argc, UT *argv[], C_UTP pTxt, UX cbp));
VO　AjcConInpSetColor(COLORREF rgbText, COLORREF rgbBack);

引　数 ： FieldLength　- 入力域の長さ（バイト数(６以上)、0：現カーソル位置からウインド右端(90%)までの桁数)
pBuf　- 入力バッファのアドレス（初期文字列 / 空文字列("")）
lBuf　- 入力バッファ長（６以上,　半角文字冠山の文字数、文字列終端を含む）
BufSize　- 入力文字格納バッファのバイト数／文字数（６以上,　内部でバッファ領域確保）
pHelp　- Ｆ１キー押下時に表示するヘルプテキスト（不要時は NULL）
TextColor　- 文字色　（-1 ： 文字色を設定しない）
BackColor　- 文字背景色（-1 ： 背景色を設定しない）
cbp　- コールバックパラメタ
cbNtcArgs　- 入力テキストの通知用コールバック（未使用時は NULL）
rgbText, rgbBack - 文字色と背景色（-1:設定しない）

説　明 ： コンソールから１行入力します。(コンソールからのキー入力のみで、標準入力(STDIN) のインダイレクトは不可)
入力できる文字は、ASCII 文字/マルチバイト文字（日本の場合は、シフト JIS）です。
入力文字数が入力域の長さを超える場合は、横スクロールしながら入力します。CTRL+S キーを押すと入力中の文字列全体が画面右上にチップ表示されます。



以下のキー操作による、入力の編集が可能です。

#	キー	内容
1	←	カーソル左移動
2	→	カーソル右移動
3	↑	過去の入力文字列へ１つ戻す　(※1)
4	↓	過去の入力文字列を１つ進める　(※1)
5	DEL	カーソル位置の文字を１文字削除
6	BS	１文字後退
7	ENTER	テキストの入力を確定し、終了する
8	ESC	テキストの入力を中止する

#	キー	内容
9	CTRL+C	文字列全体をコピー　(※2)
10	CTRL+H	キー操作ヘルプをチップ表示　(※4)
11	CTRL+P	貼り付け(カーソル位置に挿入)
12	CTRL+S	入力中文字列全体をチップ表示　(※4)
13	CTRL+V	CCTRL+P と同じ　(※3)
14	CTRL+Y	やり直しのやり直し
15	CTRL+Z	やり直し
16	F1	ユーザヘルプをチップ表示する　(※4)

※1 : ENTER キーで入力を確定した文字列を最大３２個まで記憶する
※２：ウインド中に選択文字列がある場合、CTRL+C はシステムにインタセプトされます。
※３：CTRL+V はシステムにインタセプトされます。代わりに CTRL+P を使用するか、システム設定で CTRL+V を無効化する必要あり。
※４：F1, CTRL+H や CTRL+S キーによるチップ表示は、画面右上（マルチモニタ時は、マウスカーソルのあるモニタ右上）に表示します。
チップテキスト上にマウスカーソルを置くと表示が延長され、クリックするか F1, CTRL+H や CTRL+S キーで表示は消えます。

pHelp で、ヘルプテキストを指定した場合、F1 キー押下で、指定したヘルプテキストを表示します。
ヘルプテキストには、エスケープシーケンスが使用可能です。(エスケースシーケンスに関しては「テキスト描画」の章参照)

コールバック関数(cbNtcArgs)が指定されている場合、EBTER キーで入力が確定した際に、空白で分離されたテキストが通知されます。

AjcConInpSetColor()は、入力操作中のテキストの文字色と背景色を設定します。(Windows11 のみ)
Windows10 以下で AjcConInpSetColor()で色設定すると、ウインド縮小時に表示色が乱れる可能性があります。

戻り値 ： TRUE － 成功（ENTER キーで入力を確定した）
FALSE － 失敗（ESC キーで入力を中止した／エラー）

コールバック： コールバック関数の仕様は、以下のとおりです。
cbNtcArgs（入力テキスト通知）

形　式 ： VO CALLBACK ***cbNtcArgs*** (int argc, UT *argv[], C_UTP pTxt, UX cbp) ;

引　数 ： argc － パラメタの個数
argv － パラメタ文字列へのポインタテーブル
pTxt － 入力した文字列
cbp － コールバックパラメタ

説　明 ： Enter キーを押して入力が確定した際に呼び出されます。
入力したテキストを空白で分離した部分文字列群を、文字列のポインタテーブルで通知します。

ex. 入力テキスト　　通知されるパラメタ(argc = 3)
argv[0] = “AAA”, argv[1]=”BBB”, argv[2] = “X=1 2 3”

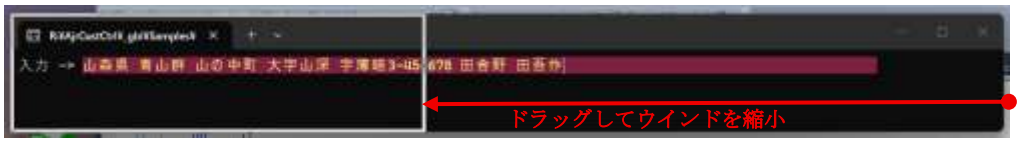
AAA BBB X=”1 2 3”

ダブルクォート(”)で囲まれた部分は、(空白を含んでも) 一連の文字列として認識し、
通知時にはダブルクォートが取り除かれた文字列となります。

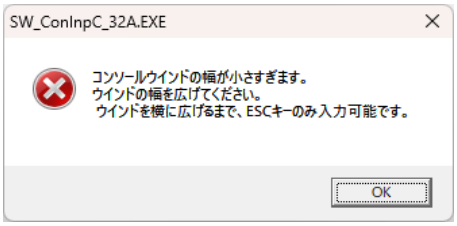
戻り値 ： なし

注　意 ： ・入力テキスト（pBuf 中の初期文字列や、CTRL+P/CTRL+V による貼り付け文字列）に制御文字(0x00～0x1F, 0x7F)が存在する場合、当該文字は、ピリオド(.)に変更されます。

・テキスト入力中に、コンソールウインドの幅を（文字列が切れる位置まで）狭くしないでください。



コンソールウインドの幅を狭くしすぎた場合は、以下のメッセージを表示し、一時的にキー入力を無効にします。



コンソールウインドの横幅を広げると、キー入力が再開されます。

23.1.2. 標準出力／標準エラー のモード設定 (AjcSetStdMode)

形 式 ： BOOL AjcSetStdoutMode (VO);
BOOL AjcSetStdoutModeEx (EAJCTEC tec, BOOL fBom);
BOOL AjcSetStdMode (EAJCTEC tec, BOOL fBom);
} 同一機能

引 数 ： tec - ディスクファイルへインダイレクト出力する際のテキストエンコード
fBom - ディスクファイルへインダイレクト出力する際のＢＯＭ出力指定（TRUE:出力する，FALSE:出力しない）

説 明 ： 標準出力(AjcPrintF 等)と標準エラー出力(AjcErrPrintF 等)における以下の設定を行います。
・コンソールウインドへ出力する際に、_setmode()によりファイル変換モードを設定するように制御します。
バイト文字モード時は「_O_TEXT」を設定、UNICODE モード時は「_O_U8TEXT」を設定し全角文字の表示を可能にします。
・AjcSetStdoutModeEx()では、AjcPrintF()/AjcErrPrintF()等においてディスクファイルへインダイレクトする場合の出力ファイルのテキストコードを「tec」で指定します。

tec	値	内容	備考
AJCTEC_MBC	0	マルチバイト	日本語の場合は S-JIS
AJCTEC_UTF_8	1	ＵＴＦ－８	
AJCTEC_EUC_J	2	ＥＵＣ（日本語）	
AJCTEC_UTF_16LE	3	ＵＴＦ－１６（リトルエンディアン）	
AJCTEC_UTF_16BE	4	ＵＴＦ－１６（ビッグエンディアン）	

「fBom」は、出力するファイルへＢＯＭを書き込むか否かを指定します。
fBom=TRUE を指定した場合は、最初の AjcPrintF()や AjcErrPrintF()等の実行時にＢＯＭが出力されます。
但し、インダイレクトで既存のファイルへ追記する場合はＢＯＭを出力しません。(ex. a.exe >>x.txt)
AjcSetStdoutMode()の場合は、「AJCTEC_MBC」が選択されます。

戻り値 ： TRUE ： 成功
FALSE ： 失敗

23.1.3. 1 文字／文字列／書式文字列をコンソールへ出力（Ajc[Err]{PutC/PutS/PrintF}）

形 式 ： BOOL AjcPutC (UT c); ---------- 標準出力(stdout)へ１バイト／１文字出力
BOOL AjcPutS (C_UTP pStr); ---------- 標準出力(stdout)へ文字列
BOOL AjcPrintF (C_UTP pFmt, ...); ----- 標準出力(stdout)へ書式文字列出力

BOOL AjcErrPutC(UT c); ---------- 標準エラー(stderr)へ１バイト／１文字出力
BOOL AjcErrPutS (C_UTP pStr); -------- 標準エラー(stderr)へ文字列
BOOL AjcErrPrintF (C_UTP pFmt, ...); -- 標準エラー(stderr)へ書式文字列出力

引 数 ： c - 出力するバイトデータ／１文字
pStr - 出力する文字列のアドレス
pFmt - 書式文字列のアドレス（書式の規則は printf()と同じ）

説 明 ： 標準出力(stdout)／標準エラー(stderr)にテキストを出力します。
AjcSetStdoutMode[Ex]()を実行していない場合は、単にシステムＡＰＩ（printf()や wprintf()等）を実行します。
AjcSetStdoutMode[Ex]()を実行している場合で、出力をディスクファイルにインダイレクトしていない場合は、システムＡＰＩを実行する前に_setmode()でファイルの変換モードを設定します。
バイト文字モード時は「_setmode(_O_TEXT)」を、UNICODE モード時は「_setmode(_O_U8TEXT)」を実行します。
AjcSetStdoutMode[Ex]()を実行している場合で、出力をインダイレクトしている場合は、システムＡＰＩを実行せずに、直接ファイルに書き込みを行います。
出力をインダイレクトしているか否かは、AjcSetStdoutMode[Ex]()実行時に判断します。
AjcSetStdoutMode[Ex]()実行後に、標準出力や標準エラーのハンドルを変更してはなりません。

戻り値 ： TRUE ： 成功
FALSE ： 失敗

備 考 ： 出力をインダイレクトする場合は、以下のようにコマンドパラメタで指定するようにしてください。

```
ex.  C:MyProg >StdOut.txt 2>StdErr.txt
```

23.1.4. コンソール情報取得 (AjcGetConsoleScreenBufferInfo)

形　式 ： BOOL AjcGetConsoleScreenBufferInfo (LPSIZE pBufSize, LPRECT pRcWnd, LPSIZE pMaxWndSize);

引　数 ： pBufSize　　- バッファサイズを格納するバッファのアドレス（不要時は NULL）
pRcWnd　　- ウインド矩形を格納するバッファのアドレス　（不要時は NULL）
pMaxWndSize - 最大ウインドサイズを格納するバッファのアドレス（不要時は NULL）

説　明 ： コンソールウインドの各種情報を取得します。
サイズは半角文字単位のサイズで、矩形情報は文字単位(０～)です。

戻り値 ： TRUE　: 成功
FALSE : 失敗

23.1.5. コンソールカーソル位置設定(AjcSetConsoleCursor

形　式 ： BOOL AjcSetConsoleCursor(int x, int y) ;

引　数 ： x - 桁位置（０～）
y - 行位置（０～）

説　明 ： コンソールのカーソル位置(半角文字単位)を設定します。

戻り値 ： TRUE　: 成功
FALSE : 失敗

23.1.6. コンソールカーソル位置取得(AjcGetConsoleCursor)

形　式 ： BOOL AjcGetConsoleCursor(int *pX, int *pY) ;

引　数 ： pX - 桁位置（０～）を格納するバッファのアドレス（不要時は NULL）
pY - 行位置（０～）を格納するバッファのアドレス（不要時は NULL）

説　明 ： コンソールのカーソル位置(半角文字単位)を取得します。

戻り値 ： TRUE　: 成功
FALSE : 失敗

23.1.7. コマンドを実行し、出力をファイルへリダイレクト(AjcSystem)

形　式 ： BOOL AjcSystem(C_UTP pCmdLine, C_UTP pOutFile, UI msTime) ;

引　数 ： pCmdLine - コマンドライン文字列
pOutFile - リダイレクトするファイルのパス名（不要時は NULL）
msTime　 - コマンド終了待ち時間[ms] （INFINIT : 永久）

説　明 ： コンソールコマンドを実行し、出力を pOutFile で指定したファイルへリダイレクトします。
pOutFile=NULL の場合は、stdout へ出力します

戻り値 ： TRUE　: 成功
FALSE : 失敗

23.2. サンプルプログラム

23.2.1. SW_ConInpC（コンソール入力）

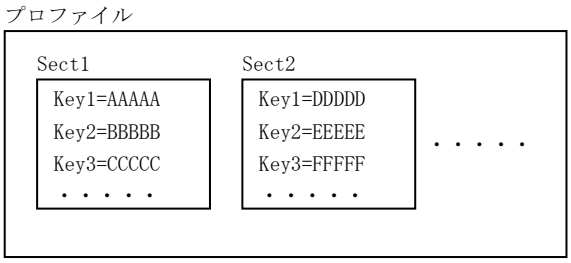
このサンプルプログラムは、コンソールからテキストを入力します。
ENTER キーを押すと、入力されたテキストと、テキストを空白で分離したテキスト群を表示します。




```
 1 : //
 2 : //  SW_ConInpC.c
 3 : //
 4 : #include    <AjrCstXX.h>
 5 : #include    <tchar.h>
 6 : #include    <conio.h>
 7 : #include <dwmapi.h>
 8 : #pragma comment(lib, "dwmapi.lib")
 9 : #define MAX_BUFLEN  (64 * sizeof(UT))
10 :
11 : //----- ユーザヘルプ ---------------------------------------------------------------------------------------//
12 : static  UT      szHelp[] = TEXT("¥n")
13 :                          TEXT(" ¥x1B[34mF1 キーでユーザ指定のヘルプを表示します。¥x1B[0m¥n¥n")
14 :                          TEXT(" 文字列中にエスケープシーケンス¥n")
15 :                          TEXT(" を指定できます¥n")
16 :                          TEXT("¥n");
17 :
18 : //----- コンソール入力のコールバック -----------------------------------------------------------------------//
19 : static VO   CALLBACK cbNtcArgs(int argc, UT *argv[], C_UTP pTxt, UX cbp)
20 : {
21 :     int     i;
22 :
23 :     AjcPrintF(TEXT("¥n コールバックに渡された情報 --¥n"));
24 :     AjcPrintF(TEXT("  pTxt = ¥"%s¥"¥n"), pTxt);
25 :     AjcPrintF(TEXT("  空白で分離したテキスト¥n"), pTxt);
26 :     for (i = 0; i < argc; i++) {
27 :         AjcPrintF(TEXT("¥tArg %02d = %s¥n"), i + 1, argv[i]);
28 :     }
29 : }
30 : //=============================================================================================================//
31 : int  AjcMain(int argc, UTP argv[])
32 : {
33 :     UT      buf[256] = TEXT("山森県 青山群 山の中町 大字山深 字薄暗 3-45-678 田舎野 田吾作");
34 :
35 :     AjcSetStdoutMode();             //  標準出力モード設定
36 :
37 :     AjcPrintFA("入力 -> ");
38 :     AjcConInpSetColor(RGB(255,255,128), RGB(128, 32, 64));
39 :     if (AjcConInputEx(30, buf, 256, szHelp, 0, cbNtcArgs)) {
40 :         AjcPrintF(TEXT("¥n 入力された文字列: ¥"%s¥"¥n"), buf);
41 :     }
42 :     else {
43 :         AjcPrintF(TEXT("ＥＳＣキーで入力をキャンセルしました¥n"));
44 :     }
45 :     AjcPrintF(TEXT("¥nHit ENTER key!!"));
46 :
47 :     getchar();
48 :
49 :
50 :     return 0;
51 : }
52 :
```

24. プロファイル・アクセス

プロファイルとは、キーと値（数値や文字列等）を関連付けて保存する媒体を意味します。
また、一連のキーをまとめたものをセクションと言います。
プロファイルへアクセスする場合は、このセクションとキー名称を指定します。



プロファイルの実際の保存先は、初期化ファイル（ＩＮＩファイル（通常のテキストファイル））か、あるいは、レジストリです。
ＩＮＩファイルへアクセスするか、レジストリへアクセスするかは、AjcSetProfileIsRegistory()での設定によります。
（デフォルトでは、レジストリとなっています）

ＩＮＩファイルへアクセスする場合は、「fRegistory」引数を「FALSE」として、AjcSetProfileIsRegistory()を実行します。

```
AjcSetProfileIsRegistory(FALSE);        //  プロファイル記録先＝INI ファイル
```

レジストリへアクセスする場合は、「fRegistory」引数を「TRUE」として、AjcSetProfileIsRegistory()を実行します。

```
AjcSetProfileIsRegistory(TRUE);         //  プロファイル記録先＝レジストリ
```

INI ファイルパスの初期値は、自プログラムパスの拡張子を「.ini」に変更したパス名となります。

「d:¥SubDir¥Mypath¥Myprog.**exe**」→「d:¥SubDir¥Mypath¥Myprog.**ini**」

レジストリのアクセスパス（レジストリキー）は、以下のように構成します。

＜トップキー＞¥＜ルートパス名＞¥＜ミドルパス名＞¥＜セクション名＞

<トップキー>は、デフォルトで「HKEY_CURRENT_USER」となっていますが、AjcSetRegTopKey()により変更できます。
<ルートパス名>は、デフォルトで「Software¥AjrCstXX」となっていますが、AjcSetRegRootPath()により変更できます。
<ミドルパス名>は、デフォルトでは、自プログラムのファイル名となっていますが、AjcSetRegMidPath()により変更できます。
<セクション名>は、各プロファイル・アクセス関数の「pSecName」引数で指定します。

ＩＮＩファイルのパスとレジストリのパスは独立して設定されます。
従って、ＩＮＩファイルのパスを変更しても、レジストリのパスはそのまま維持されます。
また、レジストリのパスを変更しても、ＩＮＩファイルのパスはそのまま維持されます。

24.1. プロファイルのアクセス手順

プロファイルのアクセス手順は、以下のとおりです。([XXXXX] は、省略可能であることを意味します)

プロファイル先をＩＮＩファイルとする場合のアクセス手順

```
  AjcSetProfileIsRegistory(FALSE);               // プロファイル先をＩＮＩファイルとする
[ AjcSetProfilePath("c:¥¥MyDir¥¥MyProg.ini"); ] // ＩＮＩファイル名設定

  AjcGetProfile･･･(･･･);  // プロファイル情報読み出し
      ・
      ・

   (自プログラムのメイン処理)

  AjcPutProfile･･･(･･･);  // プロファイル情報書き込み
      ・
      ・
```

プロファイル先をレジストリとする場合のアクセス手順

```
[ AjcSetProfileIsRegistory(TRUE);              ] // プロファイル先をレジストリとする (デフォルト)
[ AjcSetRegOptionVolatile(TRUE);               ] // レジストリへの記録を一時的なものとする (省略時は 永続記録)
[ AjcSetRegTopKey(HKEY_LOCAL_MACHINE);         ] // トップキー設定 (省略時は「HKEY_CURRENT_USER」)
[ AjcSetRegRootPath("Software¥¥SpecialName"); ] // ルートパス設定 (省略時は "Software¥¥AjrCst32")
[ AjcSetProfilePath("SpecialMidPath");         ] // ミドルパス設定 (省略時は、自プログラムのファイル名)

  AjcGetProfile･･･(･･･);                         // プロファイル情報読み出し
      ・
      ・
[ AjcCloseProfile();                           ] // 念のため、レジストリキーをクロースする

   (自プログラムのメイン処理)

  AjcPutProfile･･･(･･･);                         // プロファイル情報書き込み
      ・
      ・
[ AjcCloseProfile();                           ] // 念のため、レジストリキーをクロースする
```

基本的には、プロファイル先をレジストリとする場合だけ、最初に「AjcSetProfileIsRegistory(**TRUE**)」を実行し、あとは「AjcGetProfile･･･(･･･)」や「AjcPutProfile･･･(･･･)」によりプロファイルの読み書きを行うだけでＯＫです。

※WindowsVISTA 以降では通常、UAC(ユーザーアカウント制御)が有効な場合「HKEY_CURRENT_USER」以外のレジストリに書き込みを行うには、管理者権限でプログラムを実行する必要があります。

24.2. INI ファイルの UNICODE 化

バイト文字バージョンで INI ファイルに文字列を記録した場合、INI ファイルはバイト文字ファイルとして作成されます。
その後、UNICODE バージョンで多国語の文字列等を INI ファイルに記録する場合、INI ファイルを UNICODE 化する必要があります。
この場合、AjcIniFileToUnicode()を実行し、INI ファイルを UNICODE 化することができます。(「INI ファイル・アクセス」参照)
最初から UNICODE バージョンで INI ファイルに記録している場合、INI ファイルは UNICODE ファイルとして作成されるので、UNICODE 化する必要はありません。

24.3. サポートＡＰＩ

プロファイル・アクセスのサポートＡＰＩ一覧を以下に示します。

#	関数名	内容	備考
1	AjcSetProfileIsRegistory AjcGetProfileIsRegistory	プロファイルアクセス先設定／取得	.ini ファイル／レジストリの選択
2	AjcSetRegOptionVolatile AjcGetRegOptionVolatile	レジストリの記録方法設定／取得	
3	AjcSetIniFilePath AjcGetIniFilePath	INI ファイルパス設定／取得	
4	AjcSetRegRootPath AjcGetRegRootPath	レジストリ・ルートパスの設定／取得	
5	AjcSetRegMidPath AjcGetRegMidPath	レジストリ・ミドルパス設定／取得	
6	AjcSetProfilePath	プロファイル・アクセス・パス設定	
7	AjcGetProfilePath	プロファイル・アクセス先とパス名の取得	
8	AjcSetBinDataDir	バイナリデータ格納フォルダパス設定／取得	
9	AjcPushProfileStack	プロファイル記録先情報の退避	
10	AjcPopProfileStack	プロファイル記録先情報の回復	
11	AjcResetProfileStack	プロファイル記録先情報のリセット	
12	AjcGetProfile・・・	プロファイルからの読み出し関数群	整数(16/32Bit)，実数，文字列配列，バイナリデータ
13	AjcPutProfile・・・	プロファイルへの書き込み関数群	
14	AjcDelProfileSect	プロファイル・セクションの削除	
15	AjcDelProfileKey	プロファイル・キーの削除	
16	AjcRemoveProfileSect AjcCleanupProfileSect	プロファイル・セクションの消去／クリーンアップ	
17	AjcCloseProfile	プロファイル・クローズ	
18	AjcEnumProfileSect	プロファイル・セクション名収集	
19	AjcEnumProfileKey	プロファイル・キー名収集	

24.3.1. プロファイル・アクセス先設定／取得 (AjcSetProfileIsRegistory)

形　式 ： BOOL　AjcSetProfileIsRegistory (BOOL fRegistory);
BOOL　AjcGetProfileIsRegistory(VO);

引　数 ： fRegistory - 保存先種別 (FALSE:ＩＮＩファイル, TRUE:レジストリ(デフォルト))

説　明 ： 「AjcGetProfile…」や「AjcPutProfile…」でのプロファイル・アクセス先を設定します。
fRegistory=FALSE とした場合は、ＩＮＩファイルにアクセスします。
fRegistory=TRUE とした場合は、レジストリにアクセスします。

戻り値 ： AjcSetProfileIsRegistory()は前回設定値を、AjcGetProfileIsRegistory ()は現在の設定値を返します。
TRUE　- レジストリ
FALSE -ＩＮＩファイル

24.3.2. レジストリの記録方法設定／取得 (Ajc{Set/Get}RegOptionVolatile)

形　式 ： BOOL　AjcSetRegOptionVolatile (BOOL fVolatile);
BOOL　AjcGetRegOptionVolatile (VO);

引　数 ： fVolatile　- レジストリの記録方法 (FALSE:恒久記録 (デフォルト), TRUE:一時記録)

説　明 ： 新たなレジストリキー (新たなプロファイルセクションやキー) を作成する場合の作成方法を設定／取得します。
AjcSetRegOptionVolatile()は、新たなレジストリキーの作成を恒久的とするか、一時的とするかを設定します。
fVolatile=FALSE とした場合は、作成したレジストリキーは恒久的となります。この場合、システムをシャットダウンしても作成したレジストリキーが消滅することはありません。
fVolatile=TRUE とした場合は、レジストリキーの作成が一時的なものとなります。この場合、システムをシャットダウンすると作成したレジストリキーは消滅します。
尚,デフォルトは、fVolatile=FALSE (恒久記録) となっています。

AjcGetRegOptionVolatile ()は、現在のレジストリの記録方法を取得します。

戻り値 ： AjcSetRegOptionVolatile()は前回設定値を、AjcGetRegOptionVolatile ()は現在の設定値を返します。
TRUE　- 一時記録
FALSE - 恒久記録

注　意 ： fVolatile=TRUE としてもレジストリキーの作成が一時的とならない場合があります。
詳しい内容については (Windows の内部処理なので) 不明ですが、少なくとも、HKEY_CURRENT_USER 下で、デフォルトのレジストリ・ルートパスとミドルパスへ新たなキー (新たなプロファイルセクション, キー) を作成する場合は、一時的なキー作成が可能なようです。

24.3.3. INI ファイルパス設定／取得 (Ajc{Set/Get}IniFilePath)

形　式 ： int　AjcSetIniFilePath (UTP pPath);
int　AjcGetIniFilePath (UTP pBuf, int lBuf);

引　数 ： pPath　- INI ファイルパス名文字列のアドレス
pBuf　- INI ファイルパスを格納するバッファのアドレス
lBuf　- INI ファイルパスを格納するバッファの文字数

説　明 ： プロファイルとして INI ファイルにアクセスする際の INI ファイルパスを設定／取得します。
AjcSetProfileIsRegistory()によるプロファイルアクセス先の設定に関わらず、INI ファイルパスを設定します。
AjcSetIniFilePath()は、INI ファイルのパスを設定します。(レジストリ・パスは変更されません)
pPath=NULL とした場合は、デフォルトの INI ファイルパス (自プログラムパスの拡張子を「.ini」とした名称) が設定されます。
AjcGetIniFilePath()は、INI ファイルパスを取得します。

戻り値 ： パス名の文字数 (INI ファイルパス名の文字数)

24.3.4. レジストリ・ルートパスの設定／取得 (Ajc{Set/Get}RegRootPath)

形　式 ： int　AjcSetRegRootPath (C_UTP pRootPath);
int　AjcGetRegRootPath (UTP pBuf, UI lBuf);

引　数 ： pRootPath - 設定するルートパス名文字列のアドレス
pBuf　- ルートパスを格納するバッファのアドレス
lBuf　- ルートパスを格納するバッファの文字数

説　明 ： プロファイルとしてレジストリにアクセスする際のルートパス名を設定／取得します。
pRootPath/pBuf=NULL とした場合は、現ルートパス名の文字数を返します。

戻り値 ： パス名の文字数（ルートパス名の文字数）

24.3.5. レジストリ・ミドルパス設定／取得 (Ajc{Set/Get}RegMidPath)

形　式 ： int　AjcSetRegMidPath (UTP pPath);
int　AjcGetRegMidPath (UTP pBuf, int lBuf);

引　数 ： pPath　- ミドルパス名文字列のアドレス
pBuf　- ミドルパスを格納するバッファのアドレス
lBuf　- ミドルパスを格納するバッファの文字数

説　明 ： プロファイルとしてレジストリにアクセスする際のミドルパスを設定／取得します。
AjcSetProfileIsRegistory()によるプロファイルアクセス先の設定に関わらず、レジストリ・ミドルパスを設定します。
AjcSetRegMidPath()は、レジストリアクセスする際のミドルパスを設定します。(INI ファイルパスは変更されません)
pPath=NULL とした場合は、デフォルトのミドルパス（自プログラムのファイル名）が設定されます。
AjcGetRegMidPath()は、レジストリアクセスする際のミドルパスを取得します。
pBuf=NULL とした場合は、現ミドルパスの文字数を返します。

戻り値 ： パス名の文字数（ミドルパス名の文字数）

24.3.6. プロファイル・アクセス・パス設定 (AjcSetProfilePath)

形　式 ： int　AjcSetProfilePath (C_UTP pPath);

引　数 ： pPath　- パス名文字列のアドレス

説　明 ： プロファイル・アクセス・パスを設定します。
アクセス先がＩＮＩファイルである場合は、ＩＮＩファイルのパス名を設定します。
アクセス先がレジストリである場合は、**レジストリのミドルパス名**を設定します。

pPath=NULL とした場合は、デフォルトのパス名（アクセス先が INI ファイルの場合は自プログラムパスの拡張子を「.ini」とした名称、レジストリの場合は自プログラム名）を設定します。

戻り値 ： 生成されたパス名の文字数（INI ファイルパス名／**レジストリのミドルパス名**の文字数）

24.3.7. プロファイル・アクセス先とパス名の取得 (AjcGetProfilePath)

形　式　：　BOOL　AjcGetProfilePath (UTP pBuf, int lBuf);

引　数　：　pPath - プロファイル保存先のパス名を格納するバッファのアドレス
lBuf　- プロファイル保存先のパス名を格納するバッファの文字数

説　明　：　プロファイルのアクセス先とパス名を取得します。
アクセス先がＩＮＩファイルに設定されている場合は、pBuf で示されるバッファに INI ファイルのパスを格納し、FALSE を返します。
アクセス先がレジストリに設定されている場合は、pBuf で示されるバッファにパス名（**ルートパス＋ミドルパス**）を格納し、TRUE を返します。

戻り値　：　FALSE ： アクセス先はＩＮＩファイル
TRUE　： アクセス先はレジストリ

24.3.8. バイナリデータ格納フォルダパス設定／取得 (Ajc{Set/Get}BinDataDir)

形　式　：　VO　　AjcSetBinDataDir(C_UTP pBinDir)
VO　　AjcGetBinDataDir(UTP pBuf, UI lBuf);

引　数　：　pBinDir　- バイナリデータ格納フォルダパス名のアドレス（NULL の場合は、自プログラムのフォルダパスを設定）
pBuf　- バイナリデータ格納フォルダのパス名を格納するバッファのアドレス
lBuf　- バイナリデータ格納フォルダのパス名を格納するバッファの文字数

説　明　：　以下のＡＰＩで作成するバイナリデータファイルの格納先（フォルダパス）を設定／取得します。

・AjcGetProfileBin　　・AjcGetIniFileBin　　・AjcGetRegFileBin
・AjcPutProfileBin　　・AjcPutIniFileBin　　・AjcPutRegFileBin

AjcSetBinDataDir()は、バイナリデータファイルの格納先（フォルダパス）を設定します。
pBinDir に NULL 以外を指定した場合は、当該フォルダをバイナリデータの格納先として設定します。
pPath=NULL あるいは、pPath=""（空文字列）とした場合は、バイナリデータの格納先として自プログラムのフォルダが設定されます。
AjcSetBinDataDir()は、レジストリ用と、INI ファイル用の両方のバイナリデータファイルの格納先（フォルダパス）を設定します。
レジストリ用と、INI ファイル用のバイナリデータファイルの格納先（フォルダパス）を各々別に設定する場合は、以下のＡＰＩを使用してください。

・AjcSetIniBinDir() --- INI ファイルアクセス時のバイナリデータファイルの格納先（フォルダパス）を設定
・AjcSetRegBinDir() --- レジストリアクセス時のバイナリデータファイルの格納先（フォルダパス）を設定

AjcGetBinDataDir()は、現在設定されているバイナリデータファイルの格納先（フォルダパス）を取得します。
AjcSetProfileIsRegistory()で設定されている、アクセス先用のフォルダパスを取得します。

戻り値　：　なし

24.3.9. プロファイル記録先情報の退避 (AjcPushProfileStack)

形　式　： UI　　AjcPushProfileStack (VO);

引　数　： なし

説　明　： 現在設定されている以下のプロファイル記録先情報を退避します。
- プロファイル記録先（INI ファイル／レジストリ）
- レジストリの一時記録フラグ（Volatile 指定）
- INI ファイルパス
- レジストリパス
- レジストリルートパス
- レジストリミドルパス
- バイナリデータ格納ディレクトリ

戻り値　： プロファイル記録先の退避数

24.3.10. プロファイル記録先情報の回復 (AjcPopProfileStack)

形　式　： UI　　AjcPopProfileStack (VO);

引　数　： なし

説　明　： AjcPushProfileStack()で退避したプロファイル記録先情報を回復します。

戻り値　： プロファイル記録先の退避数

24.3.11. プロファイル記録先情報リセット (AjcResetProfileStack)

形　式　： UI　　AjcPopProfileStack (VO);

引　数　： なし

説　明　： 退避されているプロファイル記録先情報を全て破棄します。

戻り値　： プロファイル記録先の退避数（＝０）

24.3.12. プロファイル・読み出し (AjcGetProfile･･･)

形　式 ：
```
UI     AjcGetProfileUInt (C_UTP pSecName, C_UTP pKeyName, UI     defValue); ---------- 符号なし32ビット整数
SI     AjcGetProfileSInt (C_UTP pSecName, C_UTP pKeyName, SI     defValue); ---------- 符号付き32ビット整数
UI     AjcGetProfileHex  (C_UTP pSecName, C_UTP pKeyName, UI     defValue); ---------- 32ビット１６進数
double AjcGetProfileReal (C_UTP pSecName, C_UTP pKeyName, double defValue); ---------- 実数
ULL    AjcGetProfileUI64 (C_BCP pSecName, C_BCP pKeyName, ULL defValue); ------------- 符号なし64ビット整数
SLL    AjcGetProfileSI64 (C_BCP pSecName, C_BCP pKeyName, SLL defValue); ------------- 符号付き64ビット整数
ULL    AjcGetProfileH64  (C_BCP pSecName, C_BCP pKeyName, ULL defValue); ------------- 64ビット１６進数
UI     AjcGetProfileStr  (C_UTP pSecName, C_UTP pKeyName, C_UTP pDefStr, UTP pBuf, UI lBuf); - 文字列
UI     AjcGetProfileArr  (C_UTP pSecName, C_UTP pKeyName, VOP pArr, UI lArr, UI nArr); ------- 配列
UI     AjcGetProfileBin  (C_UTP pSecName, C_UTP pKeyName, VOP pBuf, UI lBuf); ---------------- バイナリデータ
BOOL   AjcGetProfileLogFont(C_UTP pSecName, C_UTP pKeyPrefix, LPLOGFONT pLogFont);  -- フォント情報
HFONT  AjcGetProfileFontObj(C_UTP pSecName, C_UTP pKeyPrefix); ---------------------- フォントオブジェクト
```

引　数 ：
```
pSecName   - セクション名文字列のアドレス
pKeyName   - 項目キーの名称文字列のアドレス
pKeyPrefix - 項目キー先頭部分の名称文字列のアドレス
defValue   - 当該キーが存在しない場合のデフォルト数値
pDefStr    - 当該キーが存在しない場合のデフォルト文字列のアドレス（NULL：空文字列を仮定）
pBuf       - 読み出した文字列を格納するバッファのアドレス／ 読み出したバイナリデータを格納するバッファのアドレス
lBuf       - 読み出した文字列を格納するバッファの文字数 ／ 読み出したバイナリデータを格納するバッファのバイト数
pArr       - 読み出した配列を格納するバッファのアドレス
lArr       - 配列の要素サイズ（１，２，４ or ８）
nArr       - 配列の要素数
pLogFont   - 読み出したフォント情報を格納するバッファのアドレス
```

説　明 ：
プロファイルから、数値、文字列、配列、あるいはバイナリデータを読み出します。

AjcGetProfileBin()の場合、プロファイルに記録されているバイナリファイルのパス名を元に、当該ファイルを読み出します。

AjcGetProfileLogFont()は、AjcPutProfileLogFont() / AjcPutProfileFontObj()でセーブしたプロファイルからフォント情報を読み出します。

AjcGetProfileFontObj()は、AjcPutProfileLogFont() / AjcPutProfileFontObj()でセーブしたプロファイルからフォント情報を読み出して、そのフォントオブジェクト（フォントハンドル）を返します。

このフォントオブジェクトは、後で、ユーザ側で解放（DeleteObject(hFont)）する必要があります。

戻り値 ：
```
AjcGetProfileStr - pBuf≠NULL & lBuf≠0の場合は、読み出した文字列の文字列長（最大lBuf - 1）
                   pBuf=NULL or lBuf=0の場合は、レジストリ・データの文字列長（但し、INIファイルの場合は最大1023文字）
                   見つからない場合は、設定したデフォルト文字列の文字列長（最大lBuf - 1）
AjcGetProfileArr      - 読み出した配列の要素数              （無い場合は、０を返す）
AjcGetProfileBin      - 読み出したバイナリデータのバイト数  （無い場合は、０を返す）
AjcGetProfileLogFont - TRUE:成功, FALSE:失敗
AjcGetProfileFontObj - ≠NULL:フォントオブジェクト（このフォントオブジェクトはユーザが解放要）
                       =NULL：失敗
その他               - 読み出した数値の値                  （無い場合は、「defValue」で指定された値を返す）
```

24.3.13. プロファイル・書き込み (AjcPutProfile･･･)

形　式 ： BOOL　AjcPutProfileUInt (C_UTP pSecName, C_UTP pKeyName, UI　　value); -------- 符号なし 32 ビット整数
BOOL　AjcPutProfileSInt (C_UTP pSecName, C_UTP pKeyName, SI　　value); -------- 符号付き 32 ビット整数
BOOL　AjcPutProfileHex　(C_UTP pSecName, C_UTP pKeyName, UI　　value); -------- 32 ビット１６進数
BOOL　AjcPutProfileReal (C_UTP pSecName, C_UTP pKeyName, double value); -------- 実数
BOOL　AjcPutProfileUI64 (C_BCP pSecName, C_BCP pKeyName, ULL　　Value); -------- 符号なし 64 ビット整数
BOOL　AjcPutProfileSI64 (C_BCP pSecName, C_BCP pKeyName, SLL　　Value); -------- 符号付き 64 ビット整数
BOOL　AjcPutProfileH64　(C_BCP pSecName, C_BCP pKeyName, ULL　　Value); -------- 64 ビット１６進数
BOOL　AjcPutProfileStr　(C_UTP pSecName, C_UTP pKeyName, C_UTP　pStr); ---------- 文字列
BOOL　AjcPutProfileArr　(C_UTP pSecName, C_UTP pKeyName, VOP pArr, UI lArr, UI nArr); ----- 配列
BOOL　AjcPutProfileBin　(C_UTP pSecName, C_UTP pKeyName, VOP pDat, UI lDat); ---------- バイナリデータ
BOOL　AjcPutProfileLogFont(C_UTP pSecName, C_UTP pKeyPrefix, const LOGFONT *pLogFont); ---------- フォント情報
BOOL　AjcPutProfileFontObj(C_UTP pSecName, C_UTP pKeyPrefix, HFONT hFont); ---------- フォントハンドル

引　数 ： pSecName　- セクション名文字列のアドレス
pKeyName　- 項目キーの名称文字列のアドレス
pKeyPrefix - 項目キー先頭部分の名称文字列のアドレス
value　　- プロファイルへ書き込む数値
pStr　　- プロファイルへ書き込む文字列のアドレス
pArr　　- プロファイルへ書き込む配列の先頭アドレス
lArr　　- プロファイルへ書き込む配列の要素サイズ（１，２，４ or ８）
nArr　　- プロファイルへ書き込む配列の要素数
pDat　　- プロファイルへ書き込むバイナリデータのアドレス
lDat　　- プロファイルへ書き込むバイナリデータのバイト数
pLogFont　- プロファイルへ書き込むフォント情報のアドレス
hFont　　- プロファイルへ書き込むフォント情報のハンドル

説　明 ： プロファイルへ、数値、文字列、配列あるいは、バイナリデータを書き込みます。

AjcPutProfileArr()で、配列をＩＮＩファイルへ書き込む場合は、各配列要素を１６進文字列に変換し、カンマ (,) で区切ったテキストを、1 行で出力します。
AjcPutProfileArr()で、配列をレジストリへ書き込む場合は、単に、配列全体をバイナリデータとして書き込みます。
いずれの場合でも、あまり大きなサイズの配列を書き込むことは好ましくありません。(エラーになる場合もあります)
大きなサイズのデータを書き込む場合は、AjcPutProfileBin()を使用してください。

AjcPutProfileBin()では、プロファイルへはバイナリファイルのパス名（下表の＜*プログラム名*＞以降）を文字列として記録し、バイナリデータは当該パス名のファイルとして書き込みます。
バイナリデータファイルのパス名は、本関数内で以下の形式で自動的に生成します。

アクセス先	記録方法	バイナリデータファイルのパス名
ＩＮＩファイル	恒久記録	*<バイナリデータ格納フォルダ>*¥＜*プログラム名*＞**_BIN**¥Ajr*<XXX>*.tmp
レジストリ	恒久記録	同上
	一時記録(VOLATILE)	*<バイナリデータ格納フォルダ>*¥＜*プログラム名*＞**_TMP**¥Ajr*<XXX>*.tmp

*<バイナリデータ格納フォルダ>*は、AjcSetBinDataDir()ＡＰＩで設定されたバイナリデータ格納フォルダパス、あるいは、自プログラムのフォルダパスを意味します。
*<プログラム名>*は、自プログラムパス名のファイル名部分を意味します。
例えば、自プログラムのパス名が「C:¥MyDir¥MyPath¥ProgA.exe」である場合、自プログラムのフォルダパスは「C:¥MyDir¥MyPath」、*<プログラム名>*は「ProgA」となります。
*<XXX>*は、ディレクトリ内で重複しない適当な名称（１６進数の文字列）が割り当てられます。

AjcPutProfileLogFont()は、フォント情報(LOGFONT)の内容をプロファイルへ書き込みます。
AjcPutProfileFontObj()は、フォントハンドルからフォント情報(LOGFONT)を取得し、AjcPutProfileLogFont()を実行します。

レジストリを恒久記録とするか、一時記録とするかは、AjrSetRegOptionVolatile()での設定によります。

戻り値 ： TRUE　- 成功
FALSE - 失敗

24.3.14. プロファイル・セクションの削除 (AjcDelProfileSect)

形　式 ： VO　　AjcDelProfileSect (C_UTP pSecName);

引　数 ： pSecName　- 削除するセクション名文字列のアドレス

説　明 ： プロファイルから指定されたセクションを削除します。
当該セクション内のキーは全て消去されます。
また、当該セクションに記録されている、AjcPutProfileBin()で作成されたバイナリファイルもすべて削除されます。

戻り値 ： TRUE　- 成功
FALSE - 失敗

注　意 ： レジストリアクセス時において、セクション名自体が階層を持っている場合（つまりセクション名に「¥」が含まれている場合）末尾の階層キーは削除されますが、セクション名自体の上位階層は削除されません。
例えば、pSect=「TopSect¥MidSect¥MySect」である場合、末尾の「MaySect」だけが削除され、先頭部分のレジストリキー「TopSect¥MidSect」は削除されずに残ってしまいます。
また、指定したセクションの下にサブキーが存在する場合は、セクションを削除できません。

24.3.15. プロファイル・キーの削除 (AjcDelProfileKey)

形　式 ： VO　　AjcDelProfileKey (C_UTP pSecName, C_UTP pKeyName);

引　数 ： pSecName　- 削除するキーが存在するセクション名文字列のアドレス
pKeyName　- 削除するキー名文字列のアドレス

説　明 ： プロファイルの指定されたセクション内から、指定キーを削除します。
AjcPutProfileBin()で作成されたキーを削除した場合は、当該バイナリファイルも削除されます。

戻り値 ： TRUE　- 成功
FALSE - 失敗

24.3.16. プロファイル・セクションの消去／クリーンアップ (Ajc{Remove/Cleanup}ProfileSect)

形　式 ： VO　　AjcRemoveProfileSect (C_UTP pSecName);
BOOL　AjcCleanupProfileSect(C_UTP pSecName)

引　数 ： pSecName　- 消去／クリーンアップするセクション名文字列のアドレス

説　明 ： AjcCleanupProfileSect()は、レジストリの指定されたセクション内のサブセクションやキーを全て消去します。
AjcRemoveProfileSect()は、指定されたセクション自体も消去します。
また、当該セクションに記録されている、AjcPutProfileBin()で作成されたバイナリファイルもすべて削除されます。

戻り値 ： TRUE　- 成功
FALSE - 失敗

24.3.17. プロファイル・クローズ (AjcCloseProfile)

形　式 ： VO　　AjcCloseProfile (VO);

引　数 ： なし

説　明 ： この関数は、プロファイルのアクセス先をレジストリに設定している場合のみ意味を持ちます。
一連のプロファイル（レジストリ）アクセスの終了時に、本関数を実行してください。
尚、プロファイルのアクセス先をＩＮＩファイルに設定している場合は、何も行いません。

本ライブラリでは、レジストリをアクセスする際に、レジストリキーのハンドルを１ヶだけ持ち、初回アクセス時にオープンし、アクセス先のパスが変化した場合に、(前のハンドルをクローズし）ハンドルをオープンし直しています。
つまり、レジストリへアクセスした場合、常に１ヶのレジストリキーハンドルがオープン状態となっています。
本関数を実行することにより、当該レジストリキーハンドルを明示的にクローズすることができます。
本関数実行後の、プロファイル（レジストリ）アクセス時には、初回アクセスと同様に、当該パスのレジストリキーハンドルをオープンします。

尚、本関数を実行しなくても、プログラムの終了時には、暗黙的に当該レジストリキーハンドルはクローズされます。

戻り値 ： なし

24.3.18. プロファイル・セクション名収集 (AjcEnumProfileSect)

形　式 ： int　　AjcEnumProfileSect　　(　　UX cbp, BOOL (CALLBACK *cbEnumSect)(C_UTP pSect, UX cbp));
int　　AjcEnumProfileRegSect (UTP pSecName, UX cbp, BOOL (CALLBACK *cbEnumSect)(C_UTP pSect, UX cbp));

引　数 ： cbp　　- コールバックパラメタ
cbEnumSect - プロファイル・セクション名を通知するコールバック関数（不要時は NULL）
pSecName　- 列挙するセクションの上位セクション名へのポインタ（不要時は NULL）

説　明 ： 「cbEnumSect」で指定されたコールバック関数をコールすることにより、プロファイル内のセクション名群を通知します。
コールバック関数に通知される「pSect」は、セクション名称文字列へのポインタです。
AjcEnumProfileRegSect()は、プロファイルをレジストリに設定した場合のみ有効で、上位のセクション名を指定できます。
指定された上位セクション名(ex. “*TopSect¥SubSect*”)の下位のセクションを列挙します。

戻り値 ： 通知したセクション名の個数

コールバック：

cbEnumSect（プロファイル・セクション名通知用コールバック）

形　式 ： BOOL CALLBACK *cbEnumSect* (C_UTP pSect, UX cbp)

引　数 ： pSect　　- セクション名へのポインタ
cbp　　- コールバックパラメタ

説　明 ： 順次、全てのセクション名を通知します。

戻り値 ： TRUE　: セクション名の通知を継続する
FALSE : セクション名の通知を中止する

24.3.19. プロファイル・キー名収集 (AjcEnumProfileKey)

形　式 ： int　AjcEnumProfileKey (C_UTP pSecName, UX cbp, BOOL (CALLBACK *cbEnumKey)(C_UTP pKey, UX cbp));

引　数 ： pSecName　- キー名を収集するセクションの名称
cbp　　- コールバックパラメタ
cbEnumKey - プロファイルセクション内のキー名を通知するコールバック関数（不要時は NULL）

説　明 ： 「cbEnumKey」で指定されたコールバック関数をコールすることにより、プロファイルセクション内のキー名群を通知します。
コールバック関数に通知される「pKey」は、キー名称文字列へのポインタです。

戻り値 ： 通知したキー名の個数

コールバック：

cbEnumKey（プロファイルセクション・キー名通知用コールバック）

形　式 ： BOOL CALLBACK *cbEnumKey*(C_UTP pKey, UX cbp)

引　数 ： pKey　　- キー名へのポインタ
cbp　　- コールバックパラメタ

説　明 ： 順次、セクション内の全てのキー名を通知します。

戻り値 ： TRUE　: キー名の通知を継続する
FALSE : キー名の通知を中止する

24.3.20. プロファイル・アクセス・マクロ

プロファイル・アクセス用として以下のマクロが用意されています。

機能	マクロ形式	展開形（ワイド文字バージン時は、「#V」→「L#V」）
符号なし整数読み出し	AJCGETPF_UINT(S, V, D)	V = AjcGetProfileUInt(S, #V, D)
符号付整数読み出し	AJCGETPF_SINT(S, V, D)	V = AjcGetProfileSInt(S, #V, D)
１６進数値読み出し	AJCGETPF_HEX (S, V, D)	V = AjcGetProfileHex (S, #V, D)
実数値読み出し	AJCGETPF_REAL(S, V, D)	V = AjcGetProfileReal(S, #V, D)
文字列読み出し	AJCGETPF_STR (S, V, D)	AjcGetProfileStr(S, #V, D, (VOP)V, AJCTSIZE(V))
配列読み出し	AJCGETPF_ARR (S, V)	AjcGetProfileArr (S, #V, (VOP)&V, sizeof V[0], (sizeof V / sizeof V[0]))
バイナリデータ読み出し	AJCGETPF_BIN (S, V)	AjcGetProfileBin (S, #V, (VOP)&V, sizeof V)
符号なし整数書き込み	AJCPUTPF_UINT(S, V)	AjcPutProfileUInt(S, #V, V)
符号付整数書き込み	AJCPUTPF_SINT(S, V)	AjcPutProfileSInt(S, #V, V)
１６進数値書き込み	AJCPUTPF_HEX (S, V)	AjcPutProfileHex (S, #V, V)
実数値書き込み	AJCPUTPF_REAL(S, V)	AjcPutProfileReal(S, #V, V)
文字列書き込み	AJCPUTPF_STR (S, V)	AjcPutProfileStr (S, #V, V)
配列書き込み	AJCGETPF_ARR (S, V)	AjcPutProfileArr (S, #V, (VOP)&V, sizeof V[0], (sizeof V / sizeof V[0]))
バイナリデータ書き込み	AJCPUTPF_BIN (S, V)	AjcPutProfileBin (S, #V, (VOP)&V, sizeof V)

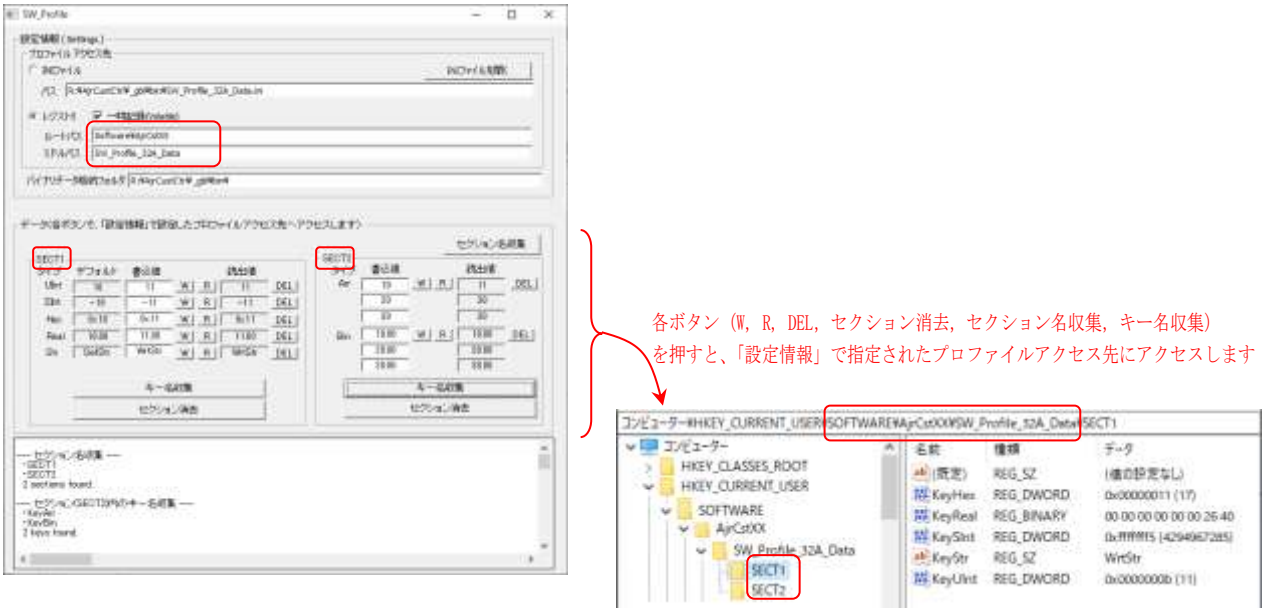
Ｓ：セクション名，　Ｖ：変数名，　Ｄ：デフォルト値

これらのマクロでは、変数名が、そのままキーの名称となります。

24.4. サンプルプログラム

24.4.1. SW_Profile（プロファイルのアクセスサンプル）

プロファイル・アクセスのテストプログラムです。



「SECT1」と「SECT2」は、プロファイルのセクション名です。
「デフォルト」は、プロファイルの読み出し時に指定するデフォルト値です。
各々、「W」ボタンを押すと、「書込値」に入力した値がプロファイルに書き込まれます。
「R」ボタンを押すと、プロファイルから読み出した値が「読出値」に表示されます。
「DEL」ボタンを押すと、プロファイルの当該キーが削除されます。
プロファイルのキー名称は、各々「タイプ」で示す名称の先頭に「Key」を付加した名称としています。

「タイプ」の意味は以下のとおりです。

タイプ	データの内容	対応するプロファイル・ファンクション
UInt	符号なし３２ビット整数	AjcGetProfileUInt, AjcPutProfileUInt
SInt	符号付き３２ビット整数	AjcGetProfileSInt, AjcPutProfileSInt
Hex	符号なし３２ビット整数	AjcGetProfileHex , AjcPutProfileHex
Real	倍精度浮動小数点	AjcGetProfileReal, AjcPutProfileReal
Str	文字列（最大２５５文字）	AjcGetProfileStr , AjcPutProfileStr
Arr	符号なし整数の配列（要素数＝３）	AjcGetProfileArr , AjcPutProfileArr
Bin	倍精度浮動小数点の配列（要素数＝３）	AjcGetProfileBin , AjcPutProfileBin

「セクション消去」ボタンは、各々、プロファイルから「SECT1」あるいは「SECT2」セクションを削除します。
「セクション名収集」と「キー名収集」ボタンは、収集したセクション名／キー名を下部のログウインドに表示します。

ダイアログ自体の設定値（各パス名、ラジオボタンや、書き込み値等）は以下の特定のレジストリに記録し永続化します。

- 設定情報　　－　SOFTWARE¥AjrCstXX¥SW_Profile¥Conf¥Settings
- 書き込み値等　－　SOFTWARE¥AjrCstXX¥SW_Profile¥Conf¥Data

```
 1 : //
 2 : //  SW_Profile.c
 3 : //
 4 : #include    <AjrCstXX.h>
 5 : #include    <Shlwapi.h>
 6 : #include    <stdio.h>
 7 : #include    <tchar.h>
 8 : #include    "resource.h"
 9 :
10 : //-------------------------------------------------------------------------------------------------------------//
11 : //  ワーク                                                                                                     //
12 : //-------------------------------------------------------------------------------------------------------------//
13 : static  HINSTANCE       hInst;                      //  ＤＬＬインスタンスハンドル
14 : static  HWND            hWndBack;                   //  バックウインドハンドル
15 : static  HWND            hDlgConf;                   //  設定用ダイアロ・ハンドル
16 : static  HWND            hDlgMain;                   //  メインダイアロ・ハンドル
17 : static  HWND            hVthLog;                    //  ログウインドハンドル
18 : static  UT              MyPath[MAX_PATH];           //  自プログラム設定値退避用レジストリパス
19 : static  UT              MySect[] = _T("Settings");  //  自プログラム設定値退避用セクション
20 : static  BOOL            fVolatile  = TRUE;          //  一時記録フラグ
21 :
22 : //----- プロファイルＲ／Ｗテスト  デフォルト値／書き込み値 ---------------------------------------------------//
23 : static  UI              defUInt =   10, wdUInt =  11;
24 : static  SI              defSInt =  -10, wdSInt = -11;
25 : static  UI              defHex  = 0x10, wdHex  = 0x11;
26 : static  double          defReal = 10.0, wdReal = 11.0;
27 : static  UT              defStr[256] = TEXT("DefStr"),   wdStr[256] = TEXT("WrtStr");
28 :
29 : static  UI              wdArr[3] = {10, 20, 30};
30 : static  double          wdBin[3] = {10.0, 20.0, 30.0};
31 :
32 : //----- プロファイルＲ／Ｗテスト  読み出し値 -----------------------------------------------------------------//
33 : static  UI              rdUInt;
34 : static  SI              rdSInt;
35 : static  UI              rdHex;
36 : static  double          rdReal;
37 : static  UT              rdStr[256];
38 :
39 : static  UI              rdArr[3];
40 : static  double          rdBin[3];
41 :
42 : #define MAX_ARR ((sizeof rdArr) / (sizeof rdArr[0]))
43 :
44 : //-------------------------------------------------------------------------------------------------------------//
45 : //  内部サブ関数                                                                                               //
46 : //-------------------------------------------------------------------------------------------------------------//
47 : AJC_WNDPROC_DEF(Back);
48 : AJC_DLGPROC_DEF(Conf);
49 : AJC_DLGPROC_DEF(Main);
50 :
51 : static  VO      SetProfileInfo(HWND hDlg);
52 :
53 : static  VO      ReadSect1(VO);
54 : static  VO      ReadSect2(VO);
55 : static  VO      SaveRegInfo(VO);
56 : static  VO      RecvRegInfo(VO);
57 : static  VO      LoadAllDlgItems(HWND hDlg, C_UTP pSect);
58 : static  VO      SaveAllDlgItems(HWND hDlg);
59 :
60 : //=============================================================================================================//
61 : //                                                                                                             //
62 : //  Ｗ ｉ ｎ Ｍ ａ ｉ ｎ                                                                                       //
63 : //                                                                                                             //
64 : //=============================================================================================================//
65 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
66 : {
67 :     MSG         msg;
68 :     WNDCLASS    wndclass;
69 :     UT          buf[MAX_PATH];
70 :     UT          drv[_MAX_DRIVE], dir[_MAX_DIR], fname[_MAX_FNAME], ext[_MAX_EXT];
71 :
72 :     hInst = hInstance;
73 :
74 :     //----- 自プログラム用の INI ファイルパス設定 --------//
75 :     GetModuleFileName(NULL, buf, MAX_PATH);
76 :     MAjcSplitPath(buf   , drv, dir, fname, ext);
77 :     MAjcMakePath (MyPath, drv, dir, fname, TEXT(".ini"));
78 :     PathCombine  (MyPath, MyPath, NULL);
79 :
80 :     //----- バックウインド生成 -------------------------//
```

```
 81 :     wndclass.style          = 0;
 82 :     wndclass.lpfnWndProc    = AJC_WNDPROC_NAME(Back);
 83 :     wndclass.cbClsExtra     = 0;
 84 :     wndclass.cbWndExtra     = 0;
 85 :     wndclass.hInstance      = hInst;
 86 :     wndclass.hIcon          = NULL;
 87 :     wndclass.hCursor        = LoadCursor(NULL, IDC_ARROW);
 88 :     wndclass.hbrBackground  = (HBRUSH)GetStockObject(WHITE_BRUSH);
 89 :     wndclass.lpszMenuName   = NULL;
 90 :     wndclass.lpszClassName  = TEXT("SW_Profile");
 91 :     RegisterClass(&wndclass);
 92 :
 93 :     hWndBack = CreateWindow(TEXT("SW_Profile"),         // window class name
 94 :                             TEXT("SW_Profile"),         // window caption
 95 :                             WS_OVERLAPPEDWINDOW & ~WS_THICKFRAME,       // window style
 96 :                             0,                          // initial x position
 97 :                             0,                          // initial y position
 98 :                             0,                          // initial x size
 99 :                             0,                          // initial y size
100 :                             NULL,                       // parent window handle
101 :                             NULL,                       // window menu handle
102 :                             hInst,                      // program instance handle
103 :                             NULL);                      // creation parameters
104 :
105 :     //----- ウインド表示 ----------------------------//
106 :     ShowWindow(hWndBack, iCmdShow);
107 :     //----- メッセージループ ------------------------//
108 :     while (GetMessage(&msg, NULL, 0, 0)) {
109 :         do {
110 :             if (IsDialogMessage(hDlgConf, &msg)) break;
111 :             if (IsDialogMessage(hDlgMain, &msg)) break;
112 :             TranslateMessage(&msg);
113 :             DispatchMessage (&msg);
114 :         } while (0);
115 :     }
116 :     return (int)msg.wParam ;
117 : }
118 : //=============================================================================================================//
119 : //                                                                                                             //
120 : //  バックウインド・プロシージャ                                                                               //
121 : //                                                                                                             //
122 : //=============================================================================================================//
123 : //----- WM_CREATE -------------------------------------------------------------------------------------------//
124 : AJC_WNDPROC(Back, WM_CREATE         )
125 : {
126 :     RECT    rm, r1, r2;
127 :     int     w1, h1, w2, h2;
128 :     int     sty = (int)MAjcGetWindowLong(hwnd, GWL_STYLE);
129 :     int     exs = (int)MAjcGetWindowLong(hwnd, GWL_EXSTYLE);
130 :
131 :     hWndBack = hwnd;
132 :     //----- ダイアログ生成 --------------------------------------//
133 :     hDlgConf = CreateDialog(hInst, MAKEINTRESOURCE(IDD_DLGCONF), hWndBack, AJC_DLGPROC_NAME(Conf));
134 :     hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_DLGMAIN), hWndBack, AJC_DLGPROC_NAME(Main));
135 :     //----- ダイアログサイズ設定 --------------------------------//
136 :     GetWindowRect(hDlgConf, &r1);   GetWindowRect(hDlgMain, &r2);
137 :     w1 = r1.right - r1.left;        w2 = r2.right - r2.left;
138 :     h1 = r1.bottom - r1.top;        h2 = r2.bottom - r2.top;
139 :     //----- ウインドサイズ設定 ----------------------------------//
140 :     sty = (int)MAjcGetWindowLong(hWndBack, GWL_STYLE);
141 :     exs = (int)MAjcGetWindowLong(hWndBack, GWL_EXSTYLE);
142 :     rm.left = rm.top = 0;
143 :     rm.right  = __max(w1, w2);
144 :     rm.bottom = h1 + h2;
145 :     AdjustWindowRectEx(&rm, sty, FALSE, exs);
146 :     SetWindowPos(hwnd, NULL, 0, 0, rm.right - rm.left, rm.bottom - rm.top, SWP_NOMOVE);
147 :     //----- ダイアログ移動 --------------------------------------//
148 :     SetWindowPos(hDlgConf, NULL, 0, 0 , 0, 0, SWP_NOSIZE);
149 :     SetWindowPos(hDlgMain, NULL, 0, h1, 0, 0, SWP_NOSIZE);
150 :     //----- ウインド表示 ----------------------------------------//
151 :     ShowWindow(hDlgMain, SW_SHOW);
152 :     ShowWindow(hDlgConf, SW_SHOW);
153 :
154 :     return 0;
155 : }
156 : //----- WM_DESTROY ------------------------------------------------------------------------------------------//
157 : AJC_WNDPROC(Back, WM_DESTROY        )
158 : {
159 :     //----- ダイアログ破棄 --------------------------------------//
160 :     DestroyWindow(hDlgConf);
```

```
161 :     DestroyWindow(hDlgMain);
162 :     //----- プログラム終了 ----------------------------------------//
163 :     PostQuitMessage(0);
164 :     return 0;
165 : }
166 : //-------------------------------------------------------------------------------------------------------------//
167 : AJC_WNDMAP_DEF(Back)
168 :     AJC_WNDMAP_MSG(Back, WM_CREATE     )
169 :     AJC_WNDMAP_MSG(Back, WM_DESTROY    )
170 : AJC_WNDMAP_END
171 : //=============================================================================================================//
172 : //                                                                                                             //
173 : //  設定情報・ダイアログ・プロシージャ                                                                         //
174 : //                                                                                                             //
175 : //=============================================================================================================//
176 : static  BOOL            fEndOfInit = FALSE;            //  初期化終了フラグ
177 : //----- ダイアログ初期化 ----------------------------------------------------------------------------------//
178 : AJC_DLGPROC(Conf, WM_INITDIALOG     )
179 : {
180 :     UT      txt[MAX_PATH];
181 :     UT      drv[_MAX_DRIVE], dir[_MAX_DIR], fname[_MAX_FNAME], fext[_MAX_EXT];
182 :
183 :     hDlgConf = hDlg;
184 :     //  ダイアログ項目のデフォルト設定
185 :     AjcSetDlgItemChk(hDlg, IDC_RBT_REG, TRUE);          //  アクセス先＝レジストリ
186 :     AjcSetDlgItemChk(hDlg, IDC_CHK_VOLATILE, TRUE);     //  一時記録
187 :     AjcGetIniFilePath(txt, MAX_PATH);                   //  INI ファイルパス
188 :     MAjcSplitPath(txt, drv, dir,fname, fext);
189 :     MAjcStrCat(fname, _MAX_FNAME, TEXT("_Data"));
190 :     MAjcMakePath (txt, drv, dir, fname, fext);
191 :     AjcSetDlgItemStr(hDlg, IDC_TXT_INIPATH, txt);
192 :     AjcGetRegRootPath(txt, MAX_PATH);                   //  レジストリルートパス
193 :     AjcSetDlgItemStr(hDlg, IDC_TXT_ROOTPATH, txt);
194 :     AjcGetRegMidPath(txt, MAX_PATH);                    //  レジストリミドルパス
195 :     MAjcStrCat(txt, MAX_PATH, TEXT("_Data"));
196 :     AjcSetRegMidPath(txt);
197 :     AjcSetDlgItemStr(hDlg, IDC_TXT_MIDPATH, txt);
198 :     AjcGetProfilePath(txt, MAX_PATH);                   //  レジストリパス
199 :     AjcSetDlgItemStr(hDlg, IDC_TXT_REGPATH, txt);
200 :     AjcGetBinDataDir(txt, MAX_PATH);                    //  バイナリデータ格納フォルダパス
201 :     AjcSetDlgItemStr(hDlg, IDC_TXT_BINPATH, txt);
202 :     //  ダイアログ設定値ロード
203 :     LoadAllDlgItems(hDlg, TEXT("Settings"));
204 :     //  初期化終了の旨設定
205 :     fEndOfInit =TRUE;
206 :     //  プロファイルアクセス先設定
207 :     SetProfileInfo(hDlg);
208 :     //  プロファイル読み出し
209 :     ReadSect1();
210 :     ReadSect2();
211 :
212 :     return TRUE;
213 : }
214 : //----- ウインド破棄 --------------------------------------------------------------------------------------//
215 : AJC_DLGPROC(Conf, WM_DESTROY        )
216 : {
217 :     //  ダイアログ設定値セーブ
218 :     SaveAllDlgItems(hDlg);
219 :
220 :     return TRUE;
221 : }
222 : //----- WM_COMMAND ----------------------------------------------------------------------------------------//
223 : AJC_DLGPROC(Conf, WM_COMMAND        )
224 : {
225 :     UI      cmd = LOWORD(wParam);
226 :     UI      act = HIWORD(wParam);
227 :
228 :     //  初期化終了後
229 :     if (fEndOfInit) {
230 :         //  設定変更ならばプロファイルアクセス先設定し全データ読み出し
231 :         if (((cmd == IDC_RBT_INI || cmd == IDC_RBT_REG || cmd == IDC_CHK_VOLATILE) && act == BN_CLICKED) || act == EN_CHANGE) {
232 :             SetProfileInfo(hDlg);
233 :             ReadSect1();
234 :             ReadSect2();
235 :         }
236 :     }
237 :     return TRUE;
238 : }
239 : //----- INI ファイルパスを開くボタン ----------------------------------------------------------------------//
240 : AJC_DLGPROC(Conf, IDC_CMD_OPENINI   )
```

```
241 : {
242 :     UT      path[MAX_PATH];
243 :     AjcGetDlgItemStr(hDlg, IDC_TXT_INIPATH, path, MAX_PATH);
244 :     ShellExecute(NULL, TEXT("open"), path, NULL, NULL, SW_SHOWNORMAL);
245 :     return TRUE;
246 : }
247 : //----- キャンセル ----------------------------------------------------------------------------------------------------//
248 : AJC_DLGPROC(Conf, IDCANCEL          )
249 : {
250 :     DestroyWindow(hDlg);
251 :     return TRUE;
252 : }
253 : //---------------------------------------------------------------------------------------------------------------------//
254 : AJC_DLGMAP_DEF(Conf)
255 :     AJC_DLGMAP_MSG(Conf, WM_INITDIALOG     )
256 :     AJC_DLGMAP_MSG(Conf, WM_DESTROY        )
257 :     AJC_DLGMAP_MSG(Conf, WM_COMMAND        )
258 :
259 :     AJC_DLGMAP_CMD(Conf, IDC_CMD_OPENINI   )
260 :     AJC_DLGMAP_CMD(Conf, IDCANCEL          )
261 : AJC_DLGMAP_END
262 : //---------------------------------------------------------------------------------------------------------------------//
263 : //  プロファイルアクセス先設定                                                                                       //
264 : //---------------------------------------------------------------------------------------------------------------------//
265 : static  VO      SetProfileInfo(HWND hDlg)
266 : {
267 :     UT      txt[MAX_PATH];
268 :
269 :     //  データ用プロファイル記録先設定
270 :     if (AjcGetDlgItemChk(hDlg, IDC_RBT_INI)) {
271 :         AjcSetProfileIsRegistory(FALSE);                                    //  アクセス先＝INI ファイル
272 :         AjcGetDlgItemStr(hDlg, IDC_TXT_INIPATH, txt, MAX_PATH);             //  INI ファイルパス設定
273 :         AjcSetIniFilePath(txt);
274 :     }
275 :     else {
276 :         AjcSetProfileIsRegistory(TRUE);                                     //  アクセス先＝レジストリ
277 :         AjcGetDlgItemStr(hDlg, IDC_TXT_ROOTPATH, txt, MAX_PATH);            //  レジストリルートパス設定
278 :         AjcSetRegRootPath(txt);
279 :         AjcGetDlgItemStr(hDlg, IDC_TXT_MIDPATH, txt, MAX_PATH);             //  レジストリミドルパス設定
280 :         AjcSetRegMidPath(txt);
281 :         AjcSetRegOptionVolatile(AjcGetDlgItemChk(hDlg, IDC_CHK_VOLATILE));  //  恒久記録／一時記録設定
282 :     }
283 :     AjcGetDlgItemStr(hDlg, IDC_TXT_BINPATH, txt, MAX_PATH);                 //  バイナリデータ格納パス設定
284 :     AjcSetBinDataDir(txt);
285 :
286 : }
287 : //=====================================================================================================================//
288 : //                                                                                                                     //
289 : //  データ・ダイアログ・プロシージャ                                                                                 //
290 : //                                                                                                                     //
291 : //=====================================================================================================================//
292 : //----- ダイアログ初期化 ----------------------------------------------------------------------------------------------//
293 : AJC_DLGPROC(Main, WM_INITDIALOG     )
294 : {
295 :     hDlgMain = hDlg;
296 :     hVthLog  = GetDlgItem(hDlg, IDC_VTH_LOG);
297 :
298 :     //  プロファイルテストのデフォルト値設定
299 :     AjcSetDlgItemUInt(hDlg, IDC_TXT_UINTDEF, defUInt);
300 :     AjcSetDlgItemSInt(hDlg, IDC_TXT_SINTDEF, defSInt);
301 :     AjcSetDlgItemHex (hDlg, IDC_TXT_HEXDEF , defHex );
302 :     AjcSetDlgItemReal(hDlg, IDC_TXT_REALDEF, defReal, 2);
303 :     AjcSetDlgItemStr (hDlg, IDC_TXT_STRDEF , defStr);
304 :
305 :     AjcSetDlgItemUInt(hDlg, IDC_TXT_UINTWD , wdUInt);
306 :     AjcSetDlgItemSInt(hDlg, IDC_TXT_SINTWD , wdSInt);
307 :     AjcSetDlgItemHex (hDlg, IDC_TXT_HEXWD  , wdHex );
308 :     AjcSetDlgItemReal(hDlg, IDC_TXT_REALWD , wdReal , 2);
309 :     AjcSetDlgItemStr (hDlg, IDC_TXT_STRWD  , wdStr );
310 :
311 :     AjcSetDlgItemSInt(hDlg, IDC_TXT_ARRWD0 , wdArr[0]);
312 :     AjcSetDlgItemSInt(hDlg, IDC_TXT_ARRWD1 , wdArr[1]);
313 :     AjcSetDlgItemSInt(hDlg, IDC_TXT_ARRWD2 , wdArr[2]);
314 :
315 :     AjcSetDlgItemReal(hDlg, IDC_TXT_BINWD0 , wdBin[0], 2);
316 :     AjcSetDlgItemReal(hDlg, IDC_TXT_BINWD1 , wdBin[1], 2);
317 :     AjcSetDlgItemReal(hDlg, IDC_TXT_BINWD2 , wdBin[2], 2);
318 :     //  ダイアログ設定値ロード
319 :     LoadAllDlgItems(hDlg, TEXT("Data"));
320 :     //  プロファイル読み出し
```

```
321 :     ReadSect1();
322 :     ReadSect2();
323 :
324 :     return TRUE;
325 : }
326 : //----- ウインド破棄 ----------------------------------------------------------------------------------------//
327 : AJC_DLGPROC(Main, WM_DESTROY        )
328 : {
329 :     //  ダイアログ設定値セーブ
330 :     SaveAllDlgItems(hDlg);
331 :
332 :     return TRUE;
333 : }
334 : //----- UInt Wボタン ----------------------------------------------------------------------------------------//
335 : AJC_DLGPROC(Main, IDC_CMD_UINTW     )
336 : {
337 :     wdUInt = AjcGetDlgItemUInt(hDlg, IDC_TXT_UINTWD);
338 :     AjcSetDlgItemUInt(hDlg, IDC_TXT_UINTWD, wdUInt);
339 :     AjcPutProfileUInt(TEXT("SECT1"), TEXT("KeyUInt"), wdUInt);
340 :
341 :     return TRUE;
342 : }
343 : //----- UInt Rボタン ----------------------------------------------------------------------------------------//
344 : AJC_DLGPROC(Main, IDC_CMD_UINTR     )
345 : {
346 :     rdUInt = AjcGetProfileUInt(TEXT("SECT1"), TEXT("KeyUInt"), defUInt);
347 :     AjcSetDlgItemUInt(hDlg, IDC_TXT_UINTRD, rdUInt);
348 :
349 :     return TRUE;
350 : }
351 : //----- UInt ＤＥＬボタン -----------------------------------------------------------------------------------//
352 : AJC_DLGPROC(Main, IDC_CMD_UINTDEL   )
353 : {
354 :     AjcDelProfileKey(TEXT("SECT1"), TEXT("KeyUInt"));
355 :     SendMessage(hDlg, WM_COMMAND, MAKELONG(IDC_CMD_UINTR, BN_CLICKED), (LPARAM)GetDlgItem(hDlg, IDC_CMD_UINTR));
356 :
357 :     return TRUE;
358 : }
359 : //----- SInt Wボタン ----------------------------------------------------------------------------------------//
360 : AJC_DLGPROC(Main, IDC_CMD_SINTW     )
361 : {
362 :     wdSInt = AjcGetDlgItemSInt(hDlg, IDC_TXT_SINTWD);
363 :     AjcSetDlgItemSInt(hDlg, IDC_TXT_SINTWD, wdSInt);
364 :     AjcPutProfileSInt(TEXT("SECT1"), TEXT("KeySInt"), wdSInt);
365 :
366 :     return TRUE;
367 : }
368 : //----- SInt Rボタン ----------------------------------------------------------------------------------------//
369 : AJC_DLGPROC(Main, IDC_CMD_SINTR     )
370 : {
371 :     rdSInt = AjcGetProfileSInt(TEXT("SECT1"), TEXT("KeySInt"), defSInt);
372 :     AjcSetDlgItemSInt(hDlg, IDC_TXT_SINTRD, rdSInt);
373 :
374 :     return TRUE;
375 : }
376 : //----- SInt ＤＥＬボタン -----------------------------------------------------------------------------------//
377 : AJC_DLGPROC(Main, IDC_CMD_SINTDEL   )
378 : {
379 :     AjcDelProfileKey(TEXT("SECT1"), TEXT("KeySInt"));
380 :     SendMessage(hDlg, WM_COMMAND, MAKELONG(IDC_CMD_SINTR, BN_CLICKED), (LPARAM)GetDlgItem(hDlg, IDC_CMD_SINTR));
381 :
382 :     return TRUE;
383 : }
384 : //----- Hex Wボタン -----------------------------------------------------------------------------------------//
385 : AJC_DLGPROC(Main, IDC_CMD_HEXW      )
386 : {
387 :     wdHex = AjcGetDlgItemHex(hDlg, IDC_TXT_HEXWD);
388 :     AjcSetDlgItemHex(hDlg, IDC_TXT_HEXWD, wdHex);
389 :     AjcPutProfileHex(TEXT("SECT1"), TEXT("KeyHex"), wdHex);
390 :
391 :     return TRUE;
392 : }
393 : //----- Hex Rボタン -----------------------------------------------------------------------------------------//
394 : AJC_DLGPROC(Main, IDC_CMD_HEXR      )
395 : {
396 :     rdHex = AjcGetProfileHex(TEXT("SECT1"), TEXT("KeyHex"), defHex);
397 :     AjcSetDlgItemHex(hDlg, IDC_TXT_HEXRD, rdHex);
398 :
399 :     return TRUE;
400 : }
```

```
401 : //----- Hex ＤＥＬボタン ----------------------------------------------------------------------------------------//
402 : AJC_DLGPROC(Main, IDC_CMD_HEXDEL    )
403 : {
404 :     AjcDelProfileKey(TEXT("SECT1"), TEXT("KeyHex"));
405 :     SendMessage(hDlg, WM_COMMAND, MAKELONG(IDC_CMD_HEXR , BN_CLICKED), (LPARAM)GetDlgItem(hDlg, IDC_CMD_HEXR ));
406 :
407 :     return TRUE;
408 : }
409 : //----- Real Wボタン --------------------------------------------------------------------------------------------//
410 : AJC_DLGPROC(Main, IDC_CMD_REALW     )
411 : {
412 :     wdReal = AjcGetDlgItemReal(hDlg, IDC_TXT_REALWD);
413 :     AjcSetDlgItemReal(hDlg, IDC_TXT_REALWD, wdReal, 2);
414 :     AjcPutProfileReal(TEXT("SECT1"), TEXT("KeyReal"), wdReal);
415 :
416 :     return TRUE;
417 : }
418 : //----- Real Rボタン --------------------------------------------------------------------------------------------//
419 : AJC_DLGPROC(Main, IDC_CMD_REALR     )
420 : {
421 :     rdReal = AjcGetProfileReal(TEXT("SECT1"), TEXT("KeyReal"), defReal);
422 :     AjcSetDlgItemReal(hDlg, IDC_TXT_REALRD, rdReal, 2);
423 :
424 :     return TRUE;
425 : }
426 : //----- Real ＤＥＬボタン ---------------------------------------------------------------------------------------//
427 : AJC_DLGPROC(Main, IDC_CMD_REALDEL   )
428 : {
429 :     AjcDelProfileKey(TEXT("SECT1"), TEXT("KeyReal"));
430 :     SendMessage(hDlg, WM_COMMAND, MAKELONG(IDC_CMD_REALR, BN_CLICKED), (LPARAM)GetDlgItem(hDlg, IDC_CMD_REALR));
431 :
432 :     return TRUE;
433 : }
434 : //----- Str Wボタン ---------------------------------------------------------------------------------------------//
435 : AJC_DLGPROC(Main, IDC_CMD_STRW      )
436 : {
437 :     AjcGetDlgItemStr(hDlg, IDC_TXT_STRWD, wdStr, AJCTSIZE(wdStr));
438 :     AjcSetDlgItemStr(hDlg, IDC_TXT_STRWD, wdStr);
439 :     AjcPutProfileStr(TEXT("SECT1"), TEXT("KeyStr"), wdStr);
440 :
441 :     return TRUE;
442 : }
443 : //----- Str Rボタン ---------------------------------------------------------------------------------------------//
444 : AJC_DLGPROC(Main, IDC_CMD_STRR      )
445 : {
446 :     AjcGetProfileStr(TEXT("SECT1"), TEXT("KeyStr"), defStr, rdStr, AJCTSIZE(rdStr));
447 :     AjcSetDlgItemStr(hDlg, IDC_TXT_STRRD, rdStr);
448 :
449 :     return TRUE;
450 : }
451 : //----- Str ＤＥＬボタン ----------------------------------------------------------------------------------------//
452 : AJC_DLGPROC(Main, IDC_CMD_STRDEL    )
453 : {
454 :     AjcDelProfileKey(TEXT("SECT1"), TEXT("KeyStr"));
455 :     SendMessage(hDlg, WM_COMMAND, MAKELONG(IDC_CMD_STRR , BN_CLICKED), (LPARAM)GetDlgItem(hDlg, IDC_CMD_STRR ));
456 :
457 :     return TRUE;
458 : }
459 : //----- Arr Wボタン ---------------------------------------------------------------------------------------------//
460 : AJC_DLGPROC(Main, IDC_CMD_ARRW      )
461 : {
462 :     wdArr[0] = AjcGetDlgItemUInt(hDlg, IDC_TXT_ARRWD0);
463 :     wdArr[1] = AjcGetDlgItemUInt(hDlg, IDC_TXT_ARRWD1);
464 :     wdArr[2] = AjcGetDlgItemUInt(hDlg, IDC_TXT_ARRWD2);
465 :     AjcSetDlgItemUInt(hDlg, IDC_TXT_ARRWD0, wdArr[0]);
466 :     AjcSetDlgItemUInt(hDlg, IDC_TXT_ARRWD1, wdArr[1]);
467 :     AjcSetDlgItemUInt(hDlg, IDC_TXT_ARRWD2, wdArr[2]);
468 :     AjcPutProfileArr(TEXT("SECT2"), TEXT("KeyArr"), wdArr, sizeof wdArr[0], sizeof wdArr / sizeof wdArr[0]);
469 :
470 :     return TRUE;
471 : }
472 : //----- Arr Rボタン ---------------------------------------------------------------------------------------------//
473 : AJC_DLGPROC(Main, IDC_CMD_ARRR      )
474 : {
475 :     if (AjcGetProfileArr(TEXT("SECT2"), TEXT("KeyArr"), rdArr, sizeof rdArr[0], sizeof rdArr / sizeof rdArr[0]) == MAX_ARR) {
476 :         AjcSetDlgItemUInt(hDlg, IDC_TXT_ARRRD0, rdArr[0]);
477 :         AjcSetDlgItemUInt(hDlg, IDC_TXT_ARRRD1, rdArr[1]);
478 :         AjcSetDlgItemUInt(hDlg, IDC_TXT_ARRRD2, rdArr[2]);
479 :     }
480 :     else {
```

```
481 :         AjcSetDlgItemStr(hDlg, IDC_TXT_ARRRD0, TEXT("-"));
482 :         AjcSetDlgItemStr(hDlg, IDC_TXT_ARRRD1, TEXT("-"));
483 :         AjcSetDlgItemStr(hDlg, IDC_TXT_ARRRD2, TEXT("-"));
484 :     }
485 :
486 :     return TRUE;
487 : }
488 : //----- Arr ＤＥＬボタン ----------------------------------------------------------------------------------//
489 : AJC_DLGPROC(Main, IDC_CMD_ARRDEL    )
490 : {
491 :     AjcDelProfileKey(TEXT("SECT2"), TEXT("KeyArr"));
492 :     SendMessage(hDlg, WM_COMMAND, MAKELONG(IDC_CMD_ARRR, BN_CLICKED), (LPARAM)GetDlgItem(hDlg, IDC_CMD_ARRR));
493 :
494 :     return TRUE;
495 : }
496 : //----- Bin Ｗボタン --------------------------------------------------------------------------------------//
497 : AJC_DLGPROC(Main, IDC_CMD_BINW      )
498 : {
499 :     wdBin[0] = AjcGetDlgItemReal(hDlg, IDC_TXT_BINWD0);
500 :     wdBin[1] = AjcGetDlgItemReal(hDlg, IDC_TXT_BINWD1);
501 :     wdBin[2] = AjcGetDlgItemReal(hDlg, IDC_TXT_BINWD2);
502 :     AjcSetDlgItemReal(hDlg, IDC_TXT_BINWD0, wdBin[0], 2);
503 :     AjcSetDlgItemReal(hDlg, IDC_TXT_BINWD1, wdBin[1], 2);
504 :     AjcSetDlgItemReal(hDlg, IDC_TXT_BINWD2, wdBin[2], 2);
505 :     AjcPutProfileBin(TEXT("SECT2"), TEXT("KeyBin"), wdBin, sizeof wdBin);
506 :
507 :     return TRUE;
508 : }
509 : //----- Bin Ｒボタン --------------------------------------------------------------------------------------//
510 : AJC_DLGPROC(Main, IDC_CMD_BINR      )
511 : {
512 :     if (AjcGetProfileBin(TEXT("SECT2"), TEXT("KeyBin"), rdBin, sizeof rdBin) == sizeof rdBin) {
513 :         AjcSetDlgItemReal(hDlg, IDC_TXT_BINRD0, rdBin[0], 2);
514 :         AjcSetDlgItemReal(hDlg, IDC_TXT_BINRD1, rdBin[1], 2);
515 :         AjcSetDlgItemReal(hDlg, IDC_TXT_BINRD2, rdBin[2], 2);
516 :     }
517 :     else {
518 :         AjcSetDlgItemStr(hDlg, IDC_TXT_BINRD0, TEXT("-"));
519 :         AjcSetDlgItemStr(hDlg, IDC_TXT_BINRD1, TEXT("-"));
520 :         AjcSetDlgItemStr(hDlg, IDC_TXT_BINRD2, TEXT("-"));
521 :     }
522 :     return TRUE;
523 : }
524 : //----- Bin ＤＥＬボタン ----------------------------------------------------------------------------------//
525 : AJC_DLGPROC(Main, IDC_CMD_BINDEL    )
526 : {
527 :     AjcDelProfileKey(TEXT("SECT2"), TEXT("KeyBin"));
528 :     SendMessage(hDlg, WM_COMMAND, MAKELONG(IDC_CMD_BINR, BN_CLICKED), (LPARAM)GetDlgItem(hDlg, IDC_CMD_BINR));
529 :
530 :     return TRUE;
531 : }
532 : //----- SECT1 セクション消去ボタン ------------------------------------------------------------------------//
533 : AJC_DLGPROC(Main, IDC_CMD_SECT1DEL  )
534 : {
535 :     AjcDelProfileSect(TEXT("SECT1"));
536 :     ReadSect1();
537 :
538 :     return TRUE;
539 : }
540 : //----- SECT2 セクション消去ボタン ------------------------------------------------------------------------//
541 : AJC_DLGPROC(Main, IDC_CMD_SECT2DEL  )
542 : {
543 :     AjcDelProfileSect(TEXT("SECT2"));
544 :     ReadSect2();
545 :
546 :     return TRUE;
547 : }
548 : //----- セクション名収集 ----------------------------------------------------------------------------------//
549 : static BOOL CALLBACK cbEnumName (C_UTP pName, UX cbp)
550 : {
551 :     AjcVthPrintF(hVthLog, TEXT("  ・%s¥n"), pName);
552 :     return TRUE;
553 : }
554 : //- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - //
555 : AJC_DLGPROC(Main, IDC_CMD_DATASECT  )
556 : {
557 :     int     n;
558 :     AjcVthPrintF(hVthLog, TEXT("¥n--- セクション名収集 ---¥n"));
559 :     n = AjcEnumProfileSect(0, cbEnumName);
560 :     AjcVthPrintF(hVthLog, TEXT("  %d sections found.¥n"), n);
```

```
561 :     return TRUE;
562 : }
563 : //----- キー名収集 (SECT1) ----------------------------------------------------------------------------------//
564 : AJC_DLGPROC(Main, IDC_CMD_DATAKEY1  )
565 : {
566 :     int     n;
567 :     AjcVthPrintF(hVthLog, TEXT("¥n--- セクション(SECT1)内のキー名収集 ---¥n"));
568 :     n = AjcEnumProfileKey(TEXT("SECT1"), 0, cbEnumName);
569 :     AjcVthPrintF(hVthLog, TEXT("  %d keys found.¥n"), n);
570 :     return TRUE;
571 : }
572 : //----- キー名収集 (SECT2) ----------------------------------------------------------------------------------//
573 : AJC_DLGPROC(Main, IDC_CMD_DATAKEY2  )
574 : {
575 :     int     n;
576 :     AjcVthPrintF(hVthLog, TEXT("¥n--- セクション(SECT2)内のキー名収集 ---¥n"));
577 :     n = AjcEnumProfileKey(TEXT("SECT2"), 0, cbEnumName);
578 :     AjcVthPrintF(hVthLog, TEXT("  %d keys found.¥n"), n);
579 :     return TRUE;
580 : }
581 : //----- ウインドクローズ ------------------------------------------------------------------------------------//
582 : AJC_DLGPROC(Main, IDCANCEL          )
583 : {
584 :     DestroyWindow(hDlg);
585 :     return TRUE;
586 : }
587 : //-----------------------------------------------------------------------------------------------------------//
588 : AJC_DLGMAP_DEF(Main)
589 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG     )
590 :     AJC_DLGMAP_MSG(Main, WM_DESTROY        )
591 :
592 :     AJC_DLGMAP_CMD(Main, IDC_CMD_UINTW     )
593 :     AJC_DLGMAP_CMD(Main, IDC_CMD_UINTR     )
594 :     AJC_DLGMAP_CMD(Main, IDC_CMD_UINTDEL   )
595 :     AJC_DLGMAP_CMD(Main, IDC_CMD_SINTW     )
596 :     AJC_DLGMAP_CMD(Main, IDC_CMD_SINTR     )
597 :     AJC_DLGMAP_CMD(Main, IDC_CMD_SINTDEL   )
598 :     AJC_DLGMAP_CMD(Main, IDC_CMD_HEXW      )
599 :     AJC_DLGMAP_CMD(Main, IDC_CMD_HEXR      )
600 :     AJC_DLGMAP_CMD(Main, IDC_CMD_HEXDEL    )
601 :     AJC_DLGMAP_CMD(Main, IDC_CMD_REALW     )
602 :     AJC_DLGMAP_CMD(Main, IDC_CMD_REALR     )
603 :     AJC_DLGMAP_CMD(Main, IDC_CMD_REALDEL   )
604 :     AJC_DLGMAP_CMD(Main, IDC_CMD_STRW      )
605 :     AJC_DLGMAP_CMD(Main, IDC_CMD_STRR      )
606 :     AJC_DLGMAP_CMD(Main, IDC_CMD_STRDEL    )
607 :     AJC_DLGMAP_CMD(Main, IDC_CMD_ARRW      )
608 :     AJC_DLGMAP_CMD(Main, IDC_CMD_ARRR      )
609 :     AJC_DLGMAP_CMD(Main, IDC_CMD_ARRDEL    )
610 :     AJC_DLGMAP_CMD(Main, IDC_CMD_BINW      )
611 :     AJC_DLGMAP_CMD(Main, IDC_CMD_BINR      )
612 :     AJC_DLGMAP_CMD(Main, IDC_CMD_BINDEL    )
613 :     AJC_DLGMAP_CMD(Main, IDC_CMD_SECT1DEL  )
614 :     AJC_DLGMAP_CMD(Main, IDC_CMD_SECT2DEL  )
615 :     AJC_DLGMAP_CMD(Main, IDC_CMD_DATASECT  )
616 :     AJC_DLGMAP_CMD(Main, IDC_CMD_DATAKEY1  )
617 :     AJC_DLGMAP_CMD(Main, IDC_CMD_DATAKEY2  )
618 :     AJC_DLGMAP_CMD(Main, IDCANCEL          )
619 : AJC_DLGMAP_END
620 : //-----------------------------------------------------------------------------------------------------------//
621 : //  SECT1 セクションの全データ読み出し                                                                       //
622 : //-----------------------------------------------------------------------------------------------------------//
623 : static  VO      ReadSect1(VO)
624 : {
625 :     SendMessage(hDlgMain, WM_COMMAND, MAKELONG(IDC_CMD_UINTR, BN_CLICKED), (LPARAM)GetDlgItem(hDlgMain, IDC_CMD_UINTR));
626 :     SendMessage(hDlgMain, WM_COMMAND, MAKELONG(IDC_CMD_SINTR, BN_CLICKED), (LPARAM)GetDlgItem(hDlgMain, IDC_CMD_SINTR));
627 :     SendMessage(hDlgMain, WM_COMMAND, MAKELONG(IDC_CMD_HEXR , BN_CLICKED), (LPARAM)GetDlgItem(hDlgMain, IDC_CMD_HEXR ));
628 :     SendMessage(hDlgMain, WM_COMMAND, MAKELONG(IDC_CMD_REALR, BN_CLICKED), (LPARAM)GetDlgItem(hDlgMain, IDC_CMD_REALR));
629 :     SendMessage(hDlgMain, WM_COMMAND, MAKELONG(IDC_CMD_STRR , BN_CLICKED), (LPARAM)GetDlgItem(hDlgMain, IDC_CMD_STRR ));
630 : }
631 : //-----------------------------------------------------------------------------------------------------------//
632 : //  SECT2 セクションの全データ読み出し                                                                       //
633 : //-----------------------------------------------------------------------------------------------------------//
634 : static  VO      ReadSect2(VO)
635 : {
636 :     SendMessage(hDlgMain, WM_COMMAND, MAKELONG(IDC_CMD_ARRR, BN_CLICKED), (LPARAM)GetDlgItem(hDlgMain, IDC_CMD_ARRR));
637 :     SendMessage(hDlgMain, WM_COMMAND, MAKELONG(IDC_CMD_BINR, BN_CLICKED), (LPARAM)GetDlgItem(hDlgMain, IDC_CMD_BINR));
638 : }
639 : //-----------------------------------------------------------------------------------------------------------//
640 : //  レジストリ記録先情報を退避し、特定の記録先設定                                                           //
```

```
641 : //-----------------------------------------------------------------------------------------------------------------//
642 : static  VO  SaveRegInfo(VO)
643 : {
644 :     //  記録先退避
645 :     AjcPushProfileStack();
646 :     //  特定の記録先設定
647 :     AjcSetProfileIsRegistory(TRUE);
648 :     AjcSetRegOptionVolatile(FALSE);
649 :     AjcSetRegRootPath(TEXT("Software¥¥AjrCstXX"));
650 :     AjcSetRegMidPath (TEXT("SW_Profile¥¥Conf"));
651 : }
652 : //-----------------------------------------------------------------------------------------------------------------//
653 : //  レジストリ記録先情 回復                                                                                        //
654 : //-----------------------------------------------------------------------------------------------------------------//
655 : static  VO  RecvRegInfo(VO)
656 : {
657 :     AjcPopProfileStack();
658 : }
659 : //-----------------------------------------------------------------------------------------------------------------//
660 : //  特定のレジストリからダイアログの全項目をロード                                                                 //
661 : //-----------------------------------------------------------------------------------------------------------------//
662 : static  VO  LoadAllDlgItems(HWND hDlg, C_UTP pSect)
663 : {
664 :     SaveRegInfo();                                          //  レジストリ記録先情報を退避し、特定の記録先設定
665 :     AjcLoadAllControlSettings(hDlg, pSect, AJCCTL_SELACT_ALL);  //  ダイアログの全項目をロード
666 :     RecvRegInfo();                                          //  レジストリ記録先情 回復
667 :
668 : }
669 : //-----------------------------------------------------------------------------------------------------------------//
670 : //  特定のレジストリにダイアログの全項目をセーブ                                                                   //
671 : //-----------------------------------------------------------------------------------------------------------------//
672 : static  VO  SaveAllDlgItems(HWND hDlg)
673 : {
674 :     SaveRegInfo();                                          //  レジストリ記録先情報を退避し、特定の記録先設定
675 :     AjcSaveAllControlSettings(hDlg);                        //  ダイアログの全項目をセーブ
676 :     RecvRegInfo();                                          //  レジストリ記録先情 回復
677 : }
```

25. ＩＮＩファイル・アクセス

ＩＮＩファイルのアクセス関数群です。

25.1. サポートＡＰＩ

ＩＮＩファイル・アクセスのサポートＡＰＩ一覧を以下に示します。

#	関数名	内容	備考
1	AjcGetIniFile･･･	.ini ファイルからのデータ読み出し関数群	
2	AjcPutIniFile･･･	.ini ファイルへのデータ書き込み関数群	
3	AjcDelIniSect	.ini ファイル・セクションの削除	
4	AjcDelIniKey	.ini ファイル・キーの削除	
5	AjcRemoveIniSect AjcCleanupIniSect	.ini ファイル・セクションの消去／クリーンアップ	
6	AjcIniFileToUnicode	.ini ファイルを UNICODE 化する	
7	AjcEnumIniSect	.ini ファイル・セクション名収集	
8	AjcEnumIniKey	.ini ファイル・キー収集	
9	Ajc{Set/Get}IniBinDir	バイナリデータ格納フォルダパス設定／取得	

25.1.1. ＩＮＩファイル・読み出し (AjcGetIniFile･･･)

形　式 ：
```
UI     AjcGetIniFileUInt (C_UTP pIniFile, C_UTP pSecName, C_UTP pKeyName, UI     defValue);  -- 符号なし整数
SI     AjcGetIniFileSInt (C_UTP pIniFile, C_UTP pSecName, C_UTP pKeyName, SI     defValue);  -- 符号付整数
UI     AjcGetIniFileHex  (C_UTP pIniFile, C_UTP pSecName, C_UTP pKeyName, UI     defValue);  -- １６進数
double AjcGetIniFileReal (C_UTP pIniFile, C_UTP pSecName, C_UTP pKeyName, double defValue);  -- 実数
UI     AjcGetIniFileStr  (C_UTP pIniFile, C_UTP pSecName, C_UTP pKeyName, C_UTP pDefStr, UTP pBuf, UI lBuf); - 文字列
UI     AjcGetIniFileArr  (C_UTP pIniFile, C_UTP pSecName, C_UTP pKeyName, VOP pArray, UI lArray, UI nArray); - 配列
UI     AjcGetIniFileBin  (C_UTP pIniFile, C_UTP pSecName, C_UTP pKeyName, VOP pBuf, UI lBuf);  ---- バイナリデータ
```

引　数 ：
```
pIniFile   - ＩＮＩファイルのパス名文字列のアドレス
pSecName   - ＩＮＩファイルセクション名文字列のアドレス
pKeyName   - ＩＮＩファイル項目キーの名称文字列のアドレス
defValue   - 当該キーが存在しない場合のデフォルト数値
pDefStr    - 当該キーが存在しない場合のデフォルト文字列のアドレス（NULL：空文字列を仮定）
pBuf       - 読み出した文字列を格納するバッファのアドレス／　読み出したバイナリデータを格納するバッファのアドレス
lBuf       - 読み出した文字列を格納するバッファの文字数　／　読み出したバイナリデータを格納するバッファのバイト数
pArr       - 読み出した配列を格納するバッファのアドレス
lArr       - 配列の要素サイズ（1，2，4 or 8）
nArr       - 配列の要素数
```

説　明 ：　ＩＮＩファイルから、数値、文字列、配列あるいは、バイナリデータを読み出します。
AjcGetIniFileBin()の場合、プロファイルに記録されているバイナリファイルのパス名を元に、当該ファイルを読み出します。

戻り値 ：
```
AjcGetIniFileStr - pBuf≠NULL ＆ lBuf≠0 の場合、読み出した文字列／デフォルト文字列の文字長（最大 lBuf － 1）
                   pBuf＝NULL or lBuf=0 の場合は、INI ファイル上の文字列長（但し、最大１０２３文字）
                   見つからない場合は、設定したデフォルト文字列の文字列長（最大 lBuf － 1）
AjcGetIniFileArr - 読み出した配列の要素数                    （無い場合は、０を返す）
AjcGetIniFileBin - 読み出したバイナリデータのバイト数        （無い場合は、０を返す）
その他           - 読み出した数値の値                        （無い場合は、「defValue」で指定された値を返す）
```

25.1.2. ＩＮＩファイル・書き込み (AjcPutIniFile･･･)

形　式 ： BOOL　AjcPutIniFileUInt (C_UTP pIniFile, C_UTP pSecName, C_UTP pKeyName, UI　　Value);　-- 符号なし整数
BOOL　AjcPutIniFileSInt (C_UTP pIniFile, C_UTP pSecName, C_UTP pKeyName, SI　　Value);　-- 符号付整数
BOOL　AjcPutIniFileHex　(C_UTP pIniFile, C_UTP pSecName, C_UTP pKeyName, UI　　Value);　-- １６進数
BOOL　AjcPutIniFileReal (C_UTP pIniFile, C_UTP pSecName, C_UTP pKeyName, double Value);　-- 実数
BOOL　AjcPutIniFileStr　(C_UTP pIniFile, C_UTP pSecName, C_UTP pKeyName, C_UTP　pStr);　　-- 文字列
BOOL　AjcPutIniFileArr　(C_UTP pIniFile, C_UTP pSecName, C_UTP pKeyName, VOP pArray, UI lArray, UI nArray); - 配列
BOOL　AjcPutIniFileBin (C_UTP pIniFile, C_UTP pSecName, C_UTP pKeyName, VOP pDat, UI lDat); --　バイナリデータ

引　数 ： pIniFile　- ＩＮＩファイル・パス名文字列のアドレス
pSecName　- ＩＮＩファイルセクション名文字列のアドレス
pKeyName　- ＩＮＩファイル項目キーの名文字列のアドレス
Value　　- 書き込む数値
pStr　　- 書き込む文字列のアドレス
pArray　　- 書き込む配列の先頭アドレス
lArray　　- 書き込む配列の要素サイズ（１，２，４ or ８）
nArray　　- 書き込む配列の要素数
pDat　　- 書き込むバイナリデータのアドレス
lDat　　- 書き込むバイナリデータのバイト数

説　明 ： ＩＮＩファイルへ、数値、文字列、配列あるいはバイナリデータを書き込みます。
尚、AjcPutIniFileHex()では、数値を、先頭に「0x」を付加した１６進文字列として出力します。

AjcPutIniFileArr()で、配列を書き込む場合は、各配列要素を１６進文字列に変換し、カンマ（,）で区切ったテキストを、1行で出力します。この場合、あまり大きなサイズの配列を書き込むことは好ましくありません。大きなサイズのデータを書き込む場合は、AjcPutIniFileBin()を使用してください。

AjcPutIniFileBin()では、プロファイルへはバイナリファイルの（＜*プログラム名*＞以降の）パス名を文字列として記録し、バイナリデータは当該パス名のファイルとして書き込みます。
バイナリデータファイルのパス名は、本関数内で以下の形式で自動的に生成します。

＜*バイナリデータ格納フォルダ*＞¥＜*プログラム名*＞_BIN¥Ajr＜*ＸＸＸ*＞.tmp

＜*バイナリデータ格納フォルダ*＞は、AjcSetBinDataDir()ＡＰＩで設定されたバイナリデータ格納フォルダパス、あるいは、自プログラムのフォルダパスを意味します。
＜*プログラム名*＞は、自プログラムパス名のファイル名部分を意味します。
例えば、自プログラムのパス名が「C:¥MyDir¥MyPath¥ProgA.exe」である場合、自プログラムのフォルダパスは「C:¥MyDir¥MyPath」、＜*プログラム名*＞は「ProgA」となります。
＜*XXX*＞は、ディレクトリ内で重複しない適当な名称（１６進数の文字列）が割り当てられます。

戻り値 ： TRUE　- 成功
FALSE - 失敗

25.1.3. ＩＮＩファイル・セクションの削除 (AjcDelIniSect)

形　式 ： BOOL　AjcDelIniSect (C_UTP pIniFile, C_UTP pSecName);

引　数 ： pIniFile　－ ＩＮＩファイル・パス名文字列のアドレス
pSecName　－ 削除するセクション名文字列のアドレス

説　明 ： ＩＮＩファイルから指定されたセクションを削除します。
当該セクション内のキーは全て消去されます。
また、当該セクションに記録されている、AjcPutIniFileBin()で作成されたバイナリファイルもすべて削除されます。

戻り値 ： TRUE　－ 成功
FALSE － 失敗

25.1.4. ＩＮＩファイル・キーの削除 (AjcDelIniKey)

形　式 ： BOOL　AjcDelIniKey (C_UTP pIniFile, C_UTP pSecName, C_UTP pKeyName);

引　数 ： pIniFile　－ ＩＮＩファイル・パス名文字列のアドレス
pSecName　－ 削除するキーが存在するセクション名文字列のアドレス
pKeyName　－ 削除するキー名文字列のアドレス

説　明 ： ＩＮＩファイルの指定されたセクション内から、指定キーを削除します。
AjcPutIniFileBin()で作成されたキーを削除した場合は、当該バイナリファイルも削除されます。

戻り値 ： TRUE　－ 成功
FALSE － 失敗

25.1.5. ＩＮＩファイルセクションの消去／クリーンアップ (Ajc{Remove/Cleanup}IniFileSect)

形　式 ： BOOL　AjcRemoveIniSect (C_UTP pIniFile, C_UTP pSecName);
BOOL　AjcCleanupIniSect(C_UTP pIniFile, C_UTP pSecName);

引　数 ： pIniFile　－ ＩＮＩファイル・パス名文字列のアドレス
pSecName　－ 消去／クリーンアップするセクション名文字列のアドレス

説　明 ： AjcCleanupIniSect()は、指定されたセクション内のキーを全て消去します。
AjcRemoveIniFSect()は、指定されたセクション自体も消去します。
また、当該セクションに記録されている、AjcPutIniFileBin()で作成されたバイナリファイルもすべて削除されます。

戻り値 ： TRUE　－ 成功
FALSE － 失敗

25.1.6. ＩＮＩファイルを UNICODE 化 (AjcIniFileToUnicode)

形　式 ： BOOL　AjcIniFileToUnicode (C_UTP pIniFile, BOOL fMkBkup);

引　数 ： pIniFile　－ ＩＮＩファイル・パス名文字列のアドレス
fMkBkup　－ オリジナルファイルをバックアップする旨のフラグ（TRUE:バックアップを作成する，FALSE：作成しない）

説　明 ： ＩＮＩファイルが UNICODE ファイルでない場合、内容はそのままで UNICODE ファイルに変更します。
fMkBkup=TRUE の場合、ＩＮＩファイルと同じフォルダ下に「bkup」フォルダを作成し、UNICODE 化前のオリジナルファイルを退避します。

戻り値 ： TRUE　－ 成功（INI ファイル化に成功／既に INI ファイルである／INI ファイルなし(まだ作成されていない)）
FALSE － 失敗（INI ファイル化を失敗）

備　考 ： バイト文字バージョンで INI ファイルに文字列を記録した場合、INI ファイルはバイト文字ファイルとして作成されます。
この場合、UNICODE バージョンで多国語の文字列を INI ファイルに記録する場合、INI ファイルを UNICODE 化する必要があります。

25.1.7. ＩＮＩファイル・セクション名収集 (AjcEnumIniSect)

形　式 ： int　AjcEnumIniSect(C_UTP pIniFile, UX cbp, BOOL (CALLBACK *cbEnumSect)(C_UTP pSect, UX cbp));

引　数 ： pIniFile　－ ＩＮＩファイル・パス名文字列のアドレス
cbp　－ コールバックパラメタ
cb　－ ＩＮＩファイル・セクション名を通知するコールバック関数（不要時はNULL）

説　明 ： 「cbEnumSect」で指定されたコールバック関数をコールすることにより、ＩＮＩファイル内のセクション名群を通知します。
コールバック関数に通知される「pSect」は、セクション名称文字列へのポインタです。

戻り値 ： 通知したセクション名の個数

コールバック：

cbEnumSect（プロファイル・セクション名通知用コールバック）

形　式 ： BOOL CALLBACK *cbEnumSect* (C_UTP pSect, UX cbp)

引　数 ： pSect　－ セクション名へのポインタ
cbp　－ コールバックパラメタ

説　明 ： 順次、全てのセクション名を通知します。

戻り値 ： TRUE ：セクション名の通知を継続する
FALSE ：セクション名の通知を中止する

備　考 ： セクション名の合計が３２Ｋ文字を超える部分のセクション名は読み出されません。

25.1.8. ＩＮＩファイル・キー収集 (AjcEnumIniKey)

形　式 ： int AjcEnumIniKey (C_UTP pIniFile, C_UTP pSecName, UX cbp, BOOL (CALLBACK *cbEnumKey)(C_UTP pKey, C_UTP pVal, UX cbp));

引　数 ： pIniFile　－ ＩＮＩファイル・パス名文字列のアドレス
pSecName　－ キーを収集するセクション名文字列のアドレス
cbp　－ コールバックパラメタ
cbEnumKey　－ ＩＮＩファイル・キーを通知するコールバック関数（不要時はNULL）

説　明 ： 「cbEnumKey」で指定されたコールバック関数をコールすることにより、ＩＮＩファイル内・セクション内のキー群を通知します。

戻り値 ： 通知したキーの個数

コールバック：

cbEnumKey（キー名称通知用コールバック）

形　式 ： BOOL CALLBACK *cbEnumKey*(C_UTP pKey, C_UTP pVal, UX cbp)

引　数 ： pSect　－ キー名称へのポインタ
pVal　－ キーの値（文字列）へのポインタ
cbp　－ コールバックパラメタ

説　明 ： 順次、全てのキー名称を通知します。

戻り値 ： TRUE ：キー名称の通知を継続する
FALSE ：キー名称の通知を中止する

備　考 ： ６４Ｋ文字を超えるセクションのキー名称は正常に読み出されません。

25.1.9. バイナリデータ格納フォルダパス設定／取得 (Ajc{Set/Get}IniBinDir)

形　式 ： VO　　AjcSetIniBinDir (C_UTP pBinDir)
VO　　AjcGetIniBinDir (UTP pBuf, UI lBuf);

引　数 ： pBinDir　－ バイナリデータ格納フォルダパス名のアドレス (NULL の場合は、自プログラムのフォルダパスを設定)
pBuf　　－ バイナリデータ格納フォルダのパス名を格納するバッファのアドレス
lBuf　　－ バイナリデータ格納フォルダのパス名を格納するバッファの文字数

説　明 ： 以下のＡＰＩで作成するバイナリデータファイルの格納先（フォルダパス）を設定／取得します。

・AjcGetIniFileBin　　・AjcGetProfileBin (INI ファイルアクセス時)
・AjcPutIniFileBin　　・AjcPutProfileBin (INI ファイルアクセス時)

AjcSetIniBinDir()は、バイナリデータファイルの格納先（フォルダパス）を設定します。
pBinDir に NULL 以外を指定した場合は、当該フォルダをバイナリデータの格納先として設定します。
pBinDir =NULL あるいは、pBinDir =""（空文字列）とした場合は、バイナリデータの格納先として自プログラムのフォルダが設定されます。

AjcGetIniBinDir()は、現在設定されているバイナリデータファイルの格納先（フォルダパス）を取得します。

戻り値 ： なし

26. レジストリ・アクセス

レジストリへのアクセスを行う関数群です。

レジストリのアクセスパス（レジストリキー）は、「**＜トップキー＞￥＜パス名＞￥＜セクション名＞**」となります。
トップキーは、デフォルトで「HKEY_CURRENT_USER」となっていますが、AjcSetRegTopKey()により変更できます。
パス名は、各レジストリ・アクセス関数の「pPath」引数で指定します。
セクション名は、各レジストリ・アクセス関数の「pSecName」引数で指定します。

※WindowsVISTA 以降では通常、UAC(ユーザーアカウント制御)が有効な場合「HKEY_CURRENT_USER」以外のレジストリに書き込みを行うには、管理者権限でプログラムを実行する必要があります。

26.1. レジストリタイプ

レジストリタイプは、以下の列挙型で定義されています。

```
typedef enum {                                                  // 値   内容
    AJCREG_NONE                         REG_NONE                //  0 : 特定の型を持たない値
    AJCREG_SZ                           REG_SZ                  //  1 : 文字列値
    AJCREG_EXPAND_SZ                    REG_EXPAND_SZ           //  2 : 展開可能な文字列値
    AJCREG_BINARY                       REG_BINARY              //  3 : バイナリ値
    AJCREG_DWORD                        REG_DWORD               //  4 : ３２ビット値（リトルエンディアン）
    AJCREG_DWORD_BIG_ENDIAN             REG_DWORD_BIG_ENDIAN    //  5 : ３２ビット値（ビッグエンディアン）
    AJCREG_LINK                         REG_LINK                //  6 : シンボリックリンク値
    AJCREG_MULTI_SZ                     REG_MULTI_SZ            //  7 : 複数の文字列値
    AJCREG_RESOURCE_LIST                REG_RESOURCE_LIST       //  8 : リソースマップ値
    AJCREG_FULL_RESOURCE_DESCRIPTOR     REG_FULL_RESOURCE_DESCRIPTOR    //  9 : ハードウェアリソースリスト値
    AJCREG_RESOURCE_REQUIREMENTS_LIST   REG_RESOURCE_REQUIREMENTS_LIST  // 10 : ネストされた一連の配列値
    AJCREG_QWORD                        REG_QWORD               // 11 : ６４ビット値（リトルエンディアン）
} AJCREGTYPE, *PAJCREGTYPE;

#define     AJCREG_DWORD_LITTLE_ENDIAN  REG_DWORD_LITTLE_ENDIAN     //  4 : ３２ビット値(REG_DWORD と同じ)
#define     AJCREG_QWORD_LITTLE_ENDIAN  REG_QWORD_LITTLE_ENDIAN     // 11 : ６４ビット値(REG_QWORD と同じ)
```

26.2. サポートＡＰＩ

#	関数名	内容	備考
1	AjcGetRegFile ･･･	レジストリからのデータ読み出し関数群	
2	AjcSetRegFile ･･･	レジストリへのデータ書き込み関数群	
3	AjcDelRegSect	レジストリ・セクションの削除	
4	AjcDelRegKey	レジストリ・キーの削除	
5	AjcRemoveRegSec AjcCleanupRegSect	レジストリ・セクションの消去／クリーンアップ	
6	AjcSetRegTopKey AjcGetRegTopKey	レジストリ・トップキーの設定／取得	
7	AjcCloseRegFile	レジストリ・キーハンドル クローズ	
8	AjcEnumRegSect	レジストリセクション名収集	
9	AjcEnumRegKey	レジストリキー収集	
10	AjcRegGet	レジストリ項目の読み出し	
11	AjcRegPut	レジストリ項目の書き込み	
12	AjcRegDelSubKey	レジストリ・サブキー削除	
13	AjcRegDelValue	レジストリ項目の削除	
14	AjcRegEnvGet AjcRegHkCuEnvGet AjcRegHkLmEnvGet	レジストリに記録されている環境変数の読み出し	
15	AjcRegEnvPut AjcRegHkCuEnvPut AjcRegHkLmEnvPut	レジストリへ環境変数と、その内容を書き込む	
16	AjcRegHkCuEnvDel AjcRegHkLmEnvDel	レジストリ中の環境変数を削除	
17	AjcRegHkCuEnvSrhItem AjcRegHkLmEnvSrhItem	レジストリの環境変数中の指定項目存在チェック	区切り文字で区切られた 部分文字列の検索
18	AjcRegHkCuEnvAddItem AjcRegHkLmEnvAddItem	レジストリの環境変数へ項目を追加	区切り文字で区切られた 部分文字列の追加
19	AjcRegHkCuEnvDelItem AjcRegHkLmEnvDelItem	レジストリの環境変数から項目を削除	区切り文字で区切られた 部分文字列の削除
20	AjcRegHkCuEnvSrhPathItem AjcRegHkLmEnvSrhPathItem	レジストリの PATH 環境変数中の指定パス存在チェック	セミコロン(;)で区切られた 部分文字列の検索
21	AjcRegHkCuEnvAddPathItem AjcRegHkLmEnvAddPathItem	レジストリの PATH 環境変数へパス項目を追加	セミコロン(;)で区切られた パス文字列の追加
22	AjcRegHkCuEnvDelPathItem AjcRegHkLmEnvDelPathItem	レジストリの PATH 環境変数からパス項目を削除	セミコロン(;)で区切られた パス文字列の削除
23	AjcRegEnableEnvironment	環境変数の有効化	
24	AjcRegIsPathExist	レジストリパスの存在チェック	
25	AjcRegIsKeyExist	レジストリキーの存在チェック	値項目の存在チェック
26	Ajc{Set/Get}RegBinDir	バイナリデータ格納フォルダパス設定／取得	

26.2.1. レジストリ・読み出し (AjcGetRegFile･･･)

形　式 ：
```
UI     AjcGetRegFileUInt (C_UTP pPath, C_UTP pSecName, C_UTP pKeyName, UI     defValue);   -- 符号なし整数(32Bit)
SI     AjcGetRegFileSInt (C_UTP pPath, C_UTP pSecName, C_UTP pKeyName, SI     defValue);   -- 符号付整数   (32Bit)
UI     AjcGetRegFileHex  (C_UTP pPath, C_UTP pSecName, C_UTP pKeyName, UI     defValue);   -- １６進数     (32Bit)
double AjcGetRegFileReal (C_UTP pPath, C_UTP pSecName, C_UTP pKeyName, double defValue);   -- 実数
ULL    AjcGetRegFileUI64 (C_UTP pPath, C_UTP pSecName, C_UTP pKeyName, ULL    defValue);   -- 符号なし整数(64Bit)
SLL    AjcGetRegFileSI64 (C_UTP pPath, C_UTP pSecName, C_UTP pKeyName, SLL    defValue);   -- 符号付整数   (64Bit)
ULL    AjcGetRegFileH64  (C_UTP pPath, C_UTP pSecName, C_UTP pKeyName, ULL    defValue);   -- １６進数     (64Bit)
UI     AjcGetRegFileStr  (C_UTP pPath, C_UTP pSecName, C_UTP pKeyName, C_UTP pDefStr, UTP pBuf, UI lBuf); - 文字列
UI     AjcGetRegFileExp  (C_UTP pPath, C_UTP pSecName, C_UTP pKeyName, C_UTP pDefStr, UTP pBuf, UI lBuf); - 展開文字列
UI     AjcGetRegFileArr  (C_UTP pPath, C_UTP pSecName, C_UTP pKeyName, VOP pArr, UI lArr, UI nArr); - 配列
UI     AjcGetRegFileBin  (C_UTP pPath, C_UTP pSecName, C_UTP pKeyName, VOP pDat, UI lDat); -- バイナリデータ
```

引　数 ：
```
pPath      - レジストリ・パス名文字列のアドレス
pSecName   - レジストリ・セクション名文字列のアドレス
pKeyName   - レジストリ・データ名文字列のアドレス（「規定」項目をアクセスする場合はNULL）
defValue   - 当該キーが存在しない場合のデフォルト数値
pDefStr    - 当該キーが存在しない場合のデフォルト文字列のアドレス（NULL：空文字列を仮定）
pBuf       - 読み出した文字列を格納するバッファのアドレス／　読み出したバイナリデータを格納するバッファのアドレス
lBuf       - 読み出した文字列を格納するバッファの文字数　　／　読み出したバイナリデータを格納するバッファのバイト数
pArr       - 読み出した配列を格納するバッファのアドレス
lArr       - 配列の要素サイズ（１，２，４ or ８）
nArr       - 配列の要素数
```

説　明 ：　レジストリから、数値、文字列、配列あるいはバイナリデータを読み出します。
　レジストリ・パス名（レジストリキー）は、pPath と pSecName を合成した名称となります。例えば、pPath=「MyRegPath¥Sub1」とし、pSecName=「Sect1」とした場合、実際のレジストリ・パス名は、「MyRegPath¥Sub1¥Sect1」となります。

　AjcGetRegFileBin()の場合、レジストリに記録されているバイナリファイルのパス名を元に、当該ファイルを読み出します。

　尚、AjcGetRegFileUInt()と AjcGetRegFileHex()は、機能上の差異はありません。

戻り値 ：
```
AjcGetRegFileStr - pBuf≠NULL ＆ lBuf≠0 の場合は、読み出した文字列の文字列長（最大 lBuf － 1）
                   pBuf＝NULL or lBuf＝0 の場合は、レジストリ・データの文字列長
                   見つからない場合は、設定したデフォルト文字列の文字列長（最大 lBuf － 1）
AjcGetRegFileExp - 同上
AjcGetRegFileArr - 読み出した配列の要素数                （無い場合は、０を返す）
AjcGetRegFileBin - 読み出したバイナリデータのバイト数    （無い場合は、０を返す）
その他           - 読み出した数値の値                    （無い場合は、「defValue」で指定された値を返す）
```

26.2.2. レジストリ・書き込み (AjcPutRegFile･･･)

形　式 ： BOOL AjcPutRegFileUInt (C_UTP pPath, C_UTP pSecName, C_UTP pKeyName, BOOL fVolatile, UI Value); -- 符号なし整数(32Bit)
BOOL AjcPutRegFileSInt (C_UTP pPath, C_UTP pSecName, C_UTP pKeyName, BOOL fVolatile, SI Value); -- 符号付整数(32Bit)
BOOL AjcPutRegFileHex (C_UTP pPath, C_UTP pSecName, C_UTP pKeyName, BOOL fVolatile, UI Value); -- １６進数(32Bit)
BOOL AjcPutRegFileReal (C_UTP pPath, C_UTP pSecName, C_UTP pKeyName, BOOL fVolatile, double Value); -- 実数
BOOL AjcPutRegFileUI64 (C_UTP pPath, C_UTP pSecName, C_UTP pKeyName, BOOL fVolatile, ULL Value); -- 符号なし整数(64Bit)
BOOL AjcPutRegFileSI64 (C_UTP pPath, C_UTP pSecName, C_UTP pKeyName, BOOL fVolatile, SLL Value); -- 符号付整数 (64Bit)
BOOL AjcPutRegFileH64 (C_UTP pPath, C_UTP pSecName, C_UTP pKeyName, BOOL fVolatile, ULL Value); -- １６進数 (64Bit)
BOOL AjcPutRegFileStr (C_UTP pPath, C_UTP pSecName, C_UTP pKeyName, BOOL fVolatile, C_UTP pStr); -- 文字列
BOOL AjcPutRegFileExp (C_UTP pPath, C_UTP pSecName, C_UTP pKeyName, BOOL fVolatile, C_UTP pStr); -- 展開文字列
BOOL AjcPutRegFileArr (C_UTP pPath, C_UTP pSecName, C_UTP pKeyName, BOOL fVolatile, VOP pArr, UI lArr, UI nArr); - 配列
BOOL AjcPutRegFileBin (C_UTP pPath, C_UTP pSecName, C_UTP pKeyName, BOOL fVolatile, VOP pDat, UI lDat); - バイナリデータ

引　数 ： pPath - レジストリパス名・文字列のアドレス
pSecName - レジストリセクション名・文字列のアドレス
pKeyName - レジストリ・データ名文字列のアドレス（「規定」項目をアクセスする場合はNULL）
fVolatile - レジストリへ恒久的な記録を行う(FALSE)か、一時的な記録を行う(TRUE)かの選択
Value - 書き込む数値
pStr - 書き込む文字列のアドレス
pArr - 書き込む配列の先頭アドレス
lArr - 書き込む配列の要素サイズ（１，２，４ or ８）
nArr - 書き込む配列の要素数
pDat - 書き込むバイナリデータのアドレス
lDat - 書き込むバイナリデータのバイト数

説　明 ： レジストリへ、数値、文字列、配列あるいは、バイナリデータを書き込みます。
レジストリ・パス名（レジストリキー）は、pPath と pSecName を合成した名称となります。例えば、pPath=「MyRegPath¥Sub1」とし、pSecName=「Sect1」とした場合、実際のレジストリ・パスは、「MyRegPath¥Sub1¥Sect1」となります。

fVolatile=FALSE とした場合は、レジストリへ恒久的な記録を行います。この場合、システムをシャットダウンしてもレジストリへ記録したデータが消滅することはありません。
fVolatile=TRUE とした場合は、新たに作成するレジストリキーを一時的なものとします。
この場合、システムをシャットダウンすると、新たに作成したレジストリキーは消滅します。
但し、既に存在するレジストリキーの場合は、永続記録となります。

＜注意＞ fVolatile=TRUE としても

AjcPutRegFileArr()で、配列をレジストリへ書き込む場合は、単に配列全体をバイナリデータとして書き込みます。
この場合、あまり大きなサイズの配列を書き込むことは好ましくありません。(エラーになる場合もあります)
大きなサイズのデータを書き込む場合は、AjcPutRegFileBin()を使用してください。

AjcPutRegFileBin()では、レジストリへは（*＜プログラム名＞*以降の）バイナリファイルのパス名を文字列として記録し、バイナリデータは当該パス名のファイルとして書き込みます。
バイナリデータファイルのパス名は、本関数内で以下の形式で自動的に生成します。

fVolatile=FALSE の場合： *〈バイナリデータ格納フォルダ〉*¥*＜プログラム名＞*_BIN¥Ajr*〈XXX〉*.tmp
fVolatile=TRUE の場合 ： *〈バイナリデータ格納フォルダ〉*¥*＜プログラム名＞*_TMP¥Ajr*〈XXX〉*.tmp

*〈バイナリデータ格納フォルダ〉*は、AjcSetBinDataDir()ＡＰＩで設定されたバイナリデータ格納フォルダパス、あるいは、自プログラムのフォルダパスを意味します。
*〈プログラム名〉*は、自プログラムパス名のファイル名部分を意味します。
例えば、自プログラムのパス名が「C:¥MyDir¥MyPath¥ProgA.exe」である場合、自プログラムのフォルダパスは「C:¥MyDir¥MyPath」、*〈プログラム名〉*は「ProgA」となります。
*〈XXX〉*は、ディレクトリ内で重複しない適当な名称（１６進数の文字列）が割り当てられます。

尚、AjcPutRegFileUInt()と AjcPutRegFileHex()は、機能上の差異はありません。

戻り値 ： TRUE - 成功
FALSE - 失敗

注　意 ： fVolatile=TRUE としてもレジストリキーの作成が一時的とならない場合があります。
詳しい内容については（Windows の内部処理なので）不明ですが、少なくとも、HKEY_CURRENT_USER の Software 下へ新たなレジストリキー（新たなセクション，キー）を作成する場合は、一時的なキー作成が可能なようです。

26.2.3. レジストリ・セクションの削除 (AjcDelRegSect)

形　式 ： BOOL　AjcDelRegSect (C_UTP pPath, C_UTP pSecName);

引　数 ： pPath　　- レジストリ・パス名文字列のアドレス
pSecName　- 削除するレジストリ・セクション名文字列のアドレス

説　明 ： レジストリから指定されたセクションを削除します。
当該セクション内のデータは全て消去されます。
また、当該セクションに記録されている、AjcPutRegFileBin()で作成されたバイナリファイルもすべて削除されます。

戻り値 ： TRUE　- 成功
FALSE - 失敗

注　意 ： セクション名自体が階層を持っている場合（つまりセクション名に「¥」が含まれている場合）末尾階層のレジストリキーは削除されますが、セクション名自体の上位階層は削除されません。
例えば、pSect=「TopSect¥MidSect¥MySect」である場合、末尾の「MaySect」だけが削除され、先頭部分のレジストリキー「TopSect¥MidSect」は削除されずに残ります。
また、指定したセクションの下にサブキーが存在する場合は、セクションを削除できません。

26.2.4. レジストリ・データの削除 (AjcDelRegKey)

形　式 ： BOOL　AjcDelRegKey (C_UTP pPath, C_UTP pSecName, C_UTP pKeyName);

引　数 ： pPath　　- レジストリ・パス名文字列のアドレス
pSecName　- 削除するデータが存在するレジストリ・セクション名文字列のアドレス
pKeyName　- 削除するデータ名文字列のアドレス

説　明 ： レジストリの指定されたセクション内から、pKeyName で指定されたデータを削除します。
AjcRegRegFileBin()で作成されたデータを削除した場合は、当該バイナリファイルも削除されます。

戻り値 ： TRUE　- 成功
FALSE - 失敗

26.2.5. レジストリ・セクションの消去／クリーンアップ(Ajc{Remove/Cleanup}RegSect)

形　式 ： BOOL　AjcRemoveRegSect (C_UTP pPath, C_UTP pSecName);
BOOL　AjcCleanupRegSect(C_UTP pPath, C_UTP pSecName)

引　数 ： pPath　　- レジストリ・パス名文字列のアドレス
pSecName　- 消去／クリーンアップするレジストリ・セクション名文字列のアドレス

説　明 ： AjcCleanupRegSect()は、レジストリの指定されたセクション内のサブセクションやキーを全て消去します。
AjcRemoveRegSect ()は、指定されたセクション自体も消去します。
また、当該セクションに記録されている、AjcPutRegFileBin()で作成されたバイナリファイルもすべて削除されます。

戻り値 ： TRUE　- 成功
FALSE - 失敗

26.2.6. レジストリ・トップキーの設定／取得 (Ajc{Set/Get}RegTopKey)

形　式 ： HKEY　AjcSetRegTopKey (HKEY hKey);
HKEY　AjcGetRegTopKey (VO);

引　数 ： hKey　- レジストリのトップキー（以下のいずれか）を指定します。

・HKEY_CLASS_ROOT　　(HKEY_LOCAL_MACHINE の一部で、文書タイプやファイル関連付け情報)
・HKEY_CURRENT_USER　(現在のユーザに対応する、HKEY_USERS のリンク)
・HKEY_USERS　　(ユーザの作業設定環境)
・HKEY_LOCAL_MACHINE　(ハードウェア構成，ネットワークプロトコル等)

説　明 ： レジストリ・トップキーを設定／取得します。
各レジストリアクセス関数では、指定されたトップキー下のレジストリキーをアクセスすることになります。

戻り値 ： 設定時：前回設定されていたレジストリ・トップキー　　取得時：現在設定されているレジストリ・トップキー

26.2.7. レジストリ・キーハンドル・クローズ (AjcCloseRegFile)

形　式 ： VO　AjcCloseRegFile (VO);

引　数 ： なし

説　明 ： 内部的なレジストリ・キーハンドルをクローズします。
一連のレジストリアクセスの終了時に、本関数を実行してください。

本ライブラリでは、レジストリアクセスする際に、レジストリキーのハンドルを１ヶだけ持ち、初回アクセス時にオープンし、アクセス先のパスが変化した場合に、(前のハンドルをクローズし）ハンドルをオープンし直しています。
つまり、レジストリへアクセスした場合、常に１ヶのレジストリキーハンドルがオープン状態となっています。
本関数を実行することにより、当該レジストリキーハンドルを明示的にクローズすることができます。
本関数実行後の、レジストリアクセス時には、初回アクセスと同様に、アクセス先（パス）の変化をチェックせずに、当該パスのレジストリキーハンドルをオープンします。

尚、本関数を実行しなくても、プログラムの終了時には、暗黙的に当該レジストリキーハンドルはクローズされます。

戻り値 ： TRUE　- 成功
FALSE - 失敗

26.2.8. レジストリ・セクション名収集 (AjcEnumRegSect)

形　式 ： int　AjcEnumRegSect (C_UTP pPath, UX cbp, BOOL (CALLBACK *cbEnumSect)(C_UTP pSect, UX cbp));

引　数 ： pPath　　- レジストリ・パス名文字列のアドレス
cbp　　- コールバックパラメタ
cbEnumSect - レジストリ・セクション名を通知するコールバック関数（不要時は NULL）

説　明 ： 「cbEnumSect」で指定されたコールバック関数をコールすることにより、pPath で指定したパス下のセクション名群を通知します。（つまり、pPath で指定したレジストリ・パス下のレジストリ・サブキー群を通知します）

戻り値 ： 通知したセクション名の個数

コールバック：

cbEnumSect（セクション名通知用コールバック）

形　式 ： BOOL CALLBACK *cbEnumSect* (C_UTP pSect, UX cbp)

引　数 ： pSect　- セクション名へのポインタ
cbp　　- コールバックパラメタ

説　明 ： 順次、全てのセクション名を通知します。

戻り値 ： TRUE　: セクション名の通知を継続する
FALSE : セクション名の通知を中止する

26.2.9. レジストリ・キー収集 (AjcEnumRegKey)

形　式 ： int AjcEnumRegKey (C_UTP pPath, C_UTP pPath UX cbp, BOOL (CALLBACK *cb)(C_UTP pKey, AJCREGTYPE type, UX cbp));

引　数 ： pPath - レジストリ・パス名文字列のアドレス
cbp　- コールバックパラメタ
cb　- レジストリ・セクション名を通知するコールバック関数（不要時は NULL）

説　明 ： 「cb」で指定されたコールバック関数をコールすることにより、pPath で指定したパス下の pSecName で指定されたセクション内のキー群を通知します。（つまり、pPath と pSecName で指定したレジストリ・パス下の値群を通知します）
コールバック関数に通知される「pKey」は、キー名称称文字列へのポインタです。

コールバック関数が TRUE を返した場合は、キーの検索を続行します。
コールバック関数が FALSE を返した場合は、キーの検索を中止します。

戻り値 ： 通知したキーの個数

コールバック：

cbEnumKey（キー名称通知用コールバック）

形　式 ： BOOL CALLBACK *cbEnumKey*(C_UTP pKey, AJCREGTYPE type, UX cbp)

引　数 ： pSect　- キー名称へのポインタ
type　- データタイプ
cbp　- コールバックパラメタ

説　明 ： 順次、全てのキー名称を通知します。

戻り値 ： TRUE　: キー名称の通知を継続する
FALSE : キー名称の通知を中止する

26.2.10. レジストリ項目の読み出し (AjcRegGet)

形　式 ： UI　　AjcRegGet (HKEY hTop, C_UTP pPath, C_UTP pKey, PAJCREGTYPE pType, VOP pBuf, UI lBuf);

引　数 ： hTop　- トップキー(HKEY_CURRENT_USER, HKEY_LOCAL_MACHINE ・・)
pPath - レジストリパス名のアドレス (ex. "Software¥¥MyApp")
pKey　- 値項目名称のアドレス　　(ex. "TitleText", 「規定」項目をアクセスする場合はNULL))
pType - 読み出した項目のタイプを格納するバッファのアドレス (不要時はNULL)
pBuf　- 読み出した項目の内容を格納するバッファのアドレス(NULL時は必要なバッファサイズ取得)
lBuf　- 読み出した項目の内容を格納するバッファのバイト数(文字列の場合は文字列終端('¥0')を含める
UNICODEの場合は文字数の２倍の値を指定する)

説　明 ： レジストリから、項目の内容を読み出します。
pBuf≠NULLの場合は、項目を読み出して、読み出したバイト数を返します。
pBuf＝NULLとした場合は、項目の内容は読み出さずに、項目を読み出すのに必要なバッファのバイト数を返します。

戻り値 ： ≠０：読み出したデータのバイト数／データの読み出しに必要なバイト数
＝０：エラー (バッファのバイト数(lBuf)が不足する場合もエラーとなる)

26.2.11. レジストリ項目の書き込み (AjcRegPut)

形　式 ： BOOL　AjcRegPut (HKEY hTop, C_UTP pPath, C_UTP pKey, AJCREGTYPE type, C_VOP pDat, UI lDat);

引　数 ： hTop　- トップキー(HKEY_CURRENT_USER, HKEY_LOCAL_MACHINE ・・)
pPath - レジストリパス名のアドレス (ex. "Software¥¥MyApp")
pKey　- データ項目名称のアドレス　(ex. "TitleText", 「規定」項目をアクセスする場合はNULL))
type　- 書き込むデータのタイプ (AJCREG_SZ, AJCREG_EXPAND_SZ, AJCREG_DWORD, AJCREG_BINARY ・・)
pDat　- 書き込むデータのアドレス
lBuf　- 書き込むデータのバイト数(文字列の場合は文字列終端('¥0')を含める
UNICODEの場合は文字数の２倍の値を指定する)

説　明 ： レジストリへ、データ項目を書き込みます。

戻り値 ： TRUE　- 成功
FALSE - 失敗

26.2.12. レジストリ・サブキー削除(AjcRegDelSubKey)

形　式 ： BOOL　AjcRegDelSubKey (HKEY hTop, C_BCP pPath, C_BCP pSubKey);

引　数 ： hTop　　- トップキー(HKEY_CURRENT_USER, HKEY_LOCAL_MACHINE ・・)
pPath　- レジストリパス名のアドレス (ex. "Software¥¥MyApp")
pSubKey - サブキー名のアドレス　　(ex. "Property")

説　明 ： pPathで指定されたレジストリパス下の pSubKeyで指定されたサブキーを削除します。
指定したサブキー下の値項目は全て削除されます。
指定したサブキーの下にサブキーがある場合は、指定サブキーを削除できません。

戻り値 ： TRUE　- 成功
FALSE - 失敗

26.2.13. レジストリ項目の削除(AjcRegDelValue)

形　式 ： BOOL　AjcRegDelValue (HKEY hTop, C_UTP pPath, C_UTP pKey);

引　数 ： hTop　- トップキー(HKEY_CURRENT_USER, HKEY_LOCAL_MACHINE ・・)
pPath - レジストリパス名のアドレス (ex. “Software¥¥MyApp”)
pKey　- 値項目名称のアドレス　　(ex. “TitleText”)

説　明 ： レジストリのデータ項目を削除します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

26.2.14. レジストリに記録されている環境変数の読み出し (AjcReg[HkCu|HkLm]EnvGet)

形　式 ： UI　　AjcRegEnvGet (C_UTP pName, UTP pBuf, UI lBuf);
UI　　AjcRegHkCuEnvGet (C_UTP pName, PAJCREGTYPE pType, UTP pBuf, UI lBuf);
UI　　AjcRegHkLmEnvGet (C_UTP pName, PAJCREGTYPE pType, UTP pBuf, UI lBuf);

引　数 ： pName – 読み出す環境変数名のアドレス
pType - 読み出した環境変数のタイプを格納するバッファのアドレス (不要時は NULL)
pBuf　- 読み出した環境変数の内容を格納するバッファのアドレス (NULL 時は必要なバッファサイズ取得)
lBuf　- 読み出した環境変数の内容を格納するバッファの文字数 (文字列終端(0x00)を含む)

説　明 ： レジストリに記録されている、環境変数を読み出します。
「AjcRegEnvGet」は、HKEY_CURRENT_USER からユーザ環境変数の内容を読み出します。
「AjcRegHkCuEnvGet」は、HKEY_CURRENT_USER からユーザ環境変数のタイプと内容を読み出します。
「AjcRegHkLmEnvGet」は、HKEY_LOCAL_MACHINE からシステム環境変数のタイプと内容を読み出します。

戻り値 ： ≠0　- 読み出したデータのバイト数／データの読み出しに必要なバイト数
＝0　- 失敗

26.2.15. レジストリへ環境変数とその内容を書き込む (AjcReg[HkCu|HkLm]EnvPut)

形　式 ： BOOL　AjcRegEnvPut (C_UTP pName, C_UTP pTxt);
BOOL　AjcRegHkCuEnvPut (C_UTP pName, AJCREGTYPE type, UTP pTxt);
BOOL　AjcRegHkLmEnvPut (C_UTP pName, AJCREGTYPE type, UTP pTxt);

引　数 ： pName – 書き込む環境変数名のアドレス
type　- 書き込む環境変数のタイプ(AJCREG_SZ, AJCREG_EXPAND_SZ or AJCREG_MULTI_SZ)
pTxt　- 書き込む環境変数の内容 (文字列) のアドレス

説　明 ： レジストリに環境変数を作成し、その内容を書き込みます。
「AjcRegEnvPut」は、HKEY_CURRENT_USER へユーザ環境変数を書き込みます。(REG_SZ タイプ固定)
「AjcRegHkCuEnvPut」は、HKEY_CURRENT_USER へ指定されたタイプでユーザ環境変数を書き込みます。
「AjcRegHkLmEnvPut」は、HKEY_LOCAL_MACHINE へ指定されたタイプでシステム環境変数を書き込みます。

書き込みが成功した場合は、当該環境変数を有効化します。(再起動は必要ありません)

戻り値 ： TRUE　- 成功
FALSE - 失敗

26.2.16. レジストリ中の環境変数を削除 (AjcReg{HkCu|HkLm}EnvDel)

形　式 ： BOOL　AjcRegHkCuEnvDel(C_UTP pName);
BOOL　AjcRegHkLmEnvDel(C_UTP pName);

引　数 ： pName – 削除する環境変数名のアドレス

説　明 ： レジストリ中の環境変数を削除します。
「AjcRegHkCuEnvDel」は、HKEY_CURRENT_USER 中のユーザ環境変数を削除します。
「AjcRegHkLmEnvPut」は、HKEY_LOCAL_MACHINE 中のシステム環境変数を削除します。

削除が成功した場合は、環境変数の削除を有効化します。(再起動は必要ありません)
当該環境変数が無い場合は何もせずに TRUE を返します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

26.2.17. レジストリの環境変数中の指定項目存在チェック (AjcReg{HkCu|HkLm}EnvSrhItem)

形　式 ： BOOL　AjcRegHkCuEnvSrhItem(C_UTP pName, C_UTP pItem, UT Dlm);
BOOL　AjcRegHkLmEnvSrhItem(C_UTP pName, C_UTP pItem, UT Dlm);

引　数 ： pName – 存在をチェックする文字列のアドレス

説　明 ： レジストリ中の環境変数中に、指定した項目が存在するかチェックします。(区切り文字で区切られた部分文字列の検索)
「AjcRegHkCuEnvSrhItem」は、HKEY_CURRENT_USER 中のユーザ環境変数を検索します。
「AjcRegHkLmEnvSrhItem」は、HKEY_LOCAL_MACHINE 中のシステム環境変数を検索します。

戻り値 ： TRUE　- 当該項目あり
FALSE - 当該環境変数無し／当該項目なし／失敗

26.2.18. レジストリの環境変数へ項目を追加 (AjcReg{HkCu|HkLn}EnvAddItem)

形　式 ： BOOL　AjcRegHkCuEnvAddItem (C_UTP pName, C_UTP pItem, UT Dlm, BOOL fFront, AJCREGTYPE type);
BOOL　AjcRegHkLmEnvAddItem (C_UTP pName, C_UTP pItem, UT Dlm, BOOL fFront, AJCREGTYPE type);

引　数 ： pName　- 環境変数名文字列のアドレス
pItem　- 追加する項目文字列のアドレス (ex. “c:¥¥Program files¥¥MyProgram”)
Dlm　- 項目の区切り文字
fFront　- 追加する位置 (TRUE:先頭へ追加, FALSE:末尾へ追加)
type　- 新規に環境変数を作成する場合のレジストリタイプ

説　明 ： レジストリの環境変数へ pItem で指定した項目を追加します。
この環境変数は、各項目が、Dlm で指定した文字で区切られているものとします。
(ex. “AAA;BBB;CCC” → セミコロン(;)で区切られた末尾へ”ZZZ”を追加 → “AAA;BBB;CCC;ZZZ”)
但し、ダブルクォート(“)で囲まれた文字列中の区切り文字は認識しません。
「AjcRegHkLmEnvAddItem」は、ユーザ環境変数(HKEY_CURRENT_USER 中)へ項目を追加します。
「AjcRegHkLmEnvAddItem」は、システム環境変数 (HKEY_LOCAL_MACHINE 中) へ項目を追加します。
環境変数が存在しない場合は、新規に type で指定されたレジストリタイプで環境変数を作成し、項目を設定します。
既に環境変数中に項目文字列がある場合は、何もせずに TRUE を返します。

書き込みが成功した場合は、当該環境変数を有効化します。(再起動は必要ありません)

戻り値 ： TRUE　- 成功
FALSE - 失敗

26.2.19. レジストリの環境変数から項目を削除 (AjcReg{HkCu|HkLm}EnvDelItem)

形　式 ： BOOL　AjcRegHkCuEnvDelItem (C_UTP pName, C_UTP pItem, UT Dlm, BOOL fErase);
BOOL　AjcRegHkLmEnvDelItem (C_UTP pName, C_UTP pItem, UT Dlm, BOOL fErase);

引　数 ： pName　- 環境変数名文字列のアドレス
pItem　- 追加する項目文字列のアドレス (ex. “c:¥¥Program files¥¥MyProgram¥¥”)
Dlm　- 項目の区切り文字
fErase - 項目が無くなった場合の環境変数削除指定

説　明 ： レジストリの環境変数から、指定した項目を削除します。
この環境変数は、各項目が、Dlm で指定した文字で区切られているものとします。
(ex. “AAA;BBB;CCC” → セミコロン(;)で区切られた”BBB”を削除→ “AAA;CCC”)
但し、ダブルクォート(“)で囲まれた文字列中の区切り文字は認識しません。
「AjcRegHkCuEnvDelItem」は、ユーザ環境変数(HKEY_CURRENT_USER 中)から項目を削除します。
「AjcRegHkLmEnvDelItem」は、システム環境変数 (HKEY_LOCAL_MACHINE 中)から項目を削除します。
fErase=TRUE を指定した場合、項目を削除した結果 項目が１つも無くなった時は環境変数自体を削除します。
fErase=FALSE を指定した場合、項目が１つも無くなった時は空文字列となります。

書き込みが成功した場合は、当該環境変数を有効化します。(再起動は必要ありません)
環境変数中に、指定した項目が存在しない場合は何もせずに TRUE を返します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

26.2.20. レジストリの PATH 環境変数中の指定パス存在チェック (AjcReg{HkCu|HkLm}EnvSrhPathItem)

形　式 ： BOOL　AjcRegHkCuEnvSrhPathItem (C_UTP pPath);
BOOL　AjcRegHkLmEnvSrhPathItem (C_UTP pPath);

引　数 ： pName – 存在をチェックするパス文字列のアドレス

説　明 ： レジストリ中の PATH 環境変数中に、指定したパスが存在するかチェックします。(セミコロン(;)で区切られた部分文字列の検索)
「AjcRegHkCuEnvSrhPathItem」は、HKEY_CURRENT_USER 中のユーザ PATH 環境変数を検索します。
「AjcRegHkLmEnvSrhPathItem」は、HKEY_LOCAL_MACHINE 中のシステム PATH 環境変数を検索します。

戻り値 ： TRUE　- 当該項目あり
FALSE - 当該環境変数無し／当該項目なし／失敗

26.2.21. レジストリの PATH 環境変数へパス項目を追加 (AjcReg{HkCu|HkLm}AddPathItem)

形　式 ： BOOL　AjcRegHkCuEnvAddPathItem (C_UTP pPath, BOOL fFront, AJCREGTYPE type);
BOOL　AjcRegHkLmEnvAddPathItem (C_UTP pPath, BOOL fFront, AJCREGTYPE type);

引　数 ： pPath　- 追加するパス文字列のアドレス (ex. “c:¥¥Program files¥¥MyProgram¥¥”)
fFront - 追加する位置 (TRUE:先頭へ追加, FALSE:末尾へ追加)
type　- 新規に環境変数を作成する場合のレジストリタイプ

説　明 ： レジストリの PATH 環境変数に (セミコロン (;)で区切って) パス項目を追加します。
(ex. “C:¥MyProg¥Sample¥” → 末尾へ”E:¥Product¥”を追加 → “C:¥MyProg¥Sample¥;E:¥Product¥”)
「AjcRegHkCuEnvAddPathItem」は、HKEY_CURRENT_USER へユーザ環境パスを追記します。
「AjcRegHkLmEnvAddPathItem」は、HKEY_LOCAL_MACHINE へシステム環境パスを追記します。

書き込みが成功した場合は、当該環境変数を有効化します。(再起動は必要ありません)
既に PATH 環境変数中にパス文字列がある場合は、何もせずに TRUE を返します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

26.2.22. レジストリの PATH 環境変数からパス項目を削除 (AjcReg{HkCu{HkLm}DelPathItem)

形　式 ： BOOL　AjcRegHkCuEnvDelPathItem (C_UTP pPath, BOOL fErase);
BOOL　AjcRegHkLmEnvDelPathItem (C_UTP pPath, BOOL fErase);

引　数 ： pPath　- 削除するパス文字列のアドレス (ex. “c:¥¥Program files¥¥MyProgram¥¥”)
fErase - 項目が１つも無くなった場合の環境変数削除指定 (TRUE の場合環境変数自体を削除)

説　明 ： レジストリの PATH 環境変数から、指定したパス項目を追加します。
(ex. “C:¥MyProg¥Sample¥;E:¥Product¥” → ”E:¥Product¥”を削除 → “C:¥MyProg¥Sample¥”)
「AjcRegHkCuEnvDelPathItem」は、HKEY_CURRENT_USER のユーザ環境パスからパス項目を削除します。
「AjcRegHkLmEnvDelPathItem」は、HKEY_LOCAL_MACHINE のシステム環境パスからパス項目を削除します。
fErase=TRUE を指定した場合、項目を削除した結果 項目が１つも無くなった時は環境変数自体を削除します。
fErase=FALSE を指定した場合、項目が１つも無くなった時は空文字列となります。

書き込みが成功した場合は、当該環境変数を有効化します。(再起動は必要ありません)
PATH 環境変数中に、指定したパスが存在しない場合は何もせずに TRUE を返します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

26.2.23. 環境変数の有効化 (AjcRegEnableEnvironment)

形　式 ： VO　　AjcRegEnableEnvironment (VO);

引　数 ： なし

説　明 ： レジストリに記録された環境変数を有効化します。(実際の環境変数に反映します)

戻り値 ： なし

26.2.24. レジストリパスの存在チェック (AjcRegIsPathExist)

形　式 ： BOOL　AjcRegIsPathExist (HKEY hTop, C_UTP pPath, UIP pErr);

引　数 ： hTop　- トップキー(HKEY_CURRENT_USER, HKEY_LOCAL_MACHINE ・・)
pPath　- レジストリパス名のアドレス (ex. “Software¥¥MyApp”)
pErr　- エラーコードを格納するバッファのアドレス (不要時は NULL)

説　明 ： レジストリに pPath で指定されたパスが存在するかチェックします。
パスが存在する場合は TRUE を返します。
パスが存在しないか、あるいは、レジストリアクセスが不能な場合は FALSE を返します。
pErr で示すバッファには、以下に示すシステムエラーコードが格納されます。

- ・ERROR_SUCCESS　　(=0) ： 正常
- ・ERROR_FILE_NOT_FOUND(=2) ： パスが見つからない
- ・その他　　： アクセスエラー等

戻り値 ： TRUE　- パスが存在する
FALSE - パスは存在しない／アクセスエラー

26.2.25. レジストリキーの存在チェック (AjcRegIsKeyExist)

形　式 ： BOOL　AjcRegIsKeyExist (HKEY hTop, C_BCP pPath, C_BCP pKey, UIP pErr);

引　数 ： hTop　- トップキー(HKEY_CURRENT_USER, HKEY_LOCAL_MACHINE ・・)
pPath - レジストリパス名のアドレス（ex. “Software¥¥MyApp”）
pKey　- レジストリキー（値項目）名のアドレス（ex. “Count”）
pErr　- エラーコードを格納するバッファのアドレス（不要時は NULL）

説　明 ： レジストリの pPath で指定されたパス中に、pKey で指定された項目が存在するかチェックします。
項目が存在する場合は TRUE を返します。
項目（あるいはパス）が存在しないか、あるいは、レジストリアクセスが不能な場合は FALSE を返します。
pErr で示すバッファには、以下に示すシステムエラーコードが格納されます。

・ERROR_SUCCESS　　(=0) ： 正常
・ERROR_FILE_NOT_FOUND(=2) ： 項目／パスが見つからない
・その他　　： アクセスエラー等

戻り値 ： TRUE　- 項目が存在する
FALSE – 項目は存在しない／アクセスエラー

26.2.26. バイナリデータ格納フォルダパス設定／取得 (Ajc{Set/Get}RegBinDir)

形　式 ： VO　　AjcSetRegBinDir (C_UTP pBinDir)
VO　　AjcGetRegBinDir (UTP pBuf, UI lBuf);

引　数 ： pBinDir　- バイナリデータ格納フォルダパス名のアドレス（NULL の場合は、自プログラムのフォルダパスを設定）
pBuf　　- バイナリデータ格納フォルダのパス名を格納するバッファのアドレス
lBuf　　- バイナリデータ格納フォルダのパス名を格納するバッファの文字数

説　明 ： 以下のＡＰＩで作成するバイナリデータファイルの格納先（フォルダパス）を設定／取得します。

・AjcGetRegFileBin　　・AjcGetProfileBin（レジストリアクセス時）
・AjcPutRegFileBin　　・AjcPutProfileBin（レジストリアクセス時）

AjcSetRegBinDir()は、バイナリデータファイルの格納先（フォルダパス）を設定します。
pBinDir に NULL 以外を指定した場合は、当該フォルダをバイナリデータの格納先として設定します。
pBinDir =NULL あるいは、pBinDir =”” (空文字列) とした場合は、バイナリデータの格納先として自プログラムのフォルダが設定されます。

AjcGetRegBinDir()は、現在設定されているバイナリデータファイルの格納先（フォルダパス）を取得します。

戻り値 ： なし

27. ファイル名／フォルダ名の取得

主に、Windows コモンコントロール等を使用した、ダイアログによるファイル名やフォルダ名の取得関数群です。

27.1. サポートＡＰＩ

ファイル名／フォルダ名の取得のサポートＡＰＩ一覧を以下に示します。

#	関数名	内容	備考
1	AjcGetSaveFile	保存ファイル名の取得	ダイアログ操作による取得
2	AjcGetOpenFile	オープンファイル名の取得	
3	AjcGetOpenFiles	複数のオープンファイル名取得	
4	AjcReleaseOpenFilesArray	AjcGetOpenFiles()で生成されたリソースの開放	
5	AjcGetFolderName	フォルダ名の取得	
6	AjcGetAppPath AjcGetAppName	起動したプログラムのフォルダパス名／プログラム名取得	
7	AjcGetAppPathByHWnd (AjcGetAppPathEx)	ウインドを所有するプロセスの実行ファイル・パス名取得	

27.1.1. 保存ファイル名取得 (AjcGetSaveFile)

形　式 ： BOOL　AjcGetSaveFile(HWND hWnd, C_UTP pTitle, C_UTP pFilter, C_UTP pDefExt, UTP pBuf, UI lBuf)

引　数 ： hWnd　　－ オーナーウインドゥハンドル （不要時はＮＵＬＬ）
pTitle　－ ダイアログボックスのタイトル（ＮＵＬＬを指定した場合は「名前を付けて保存」を仮定）
pFilter － フィルタ（不要時はＮＵＬＬ）
pDefExt － 既定の拡張子（ピリオド (.) は指定不要，不要時はＮＵＬＬ）
pBuf　　－ 取得したファイル名を格納するバッファのアドレス
lBuf　　－ 取得したファイル名を格納するバッファのバイト数／文字数

説　明 ： ダイアログボックス操作により、保存ファイル名を取得します。
pfilter は、表示するファイル名を制限するためのワイルドカードを指定するものであり、「項目名／ワイルドカード」のペアで、複数指定可能です。(ex. ”AllFiles/*.*/TextFiles/*.txt”)
pFilter の先頭に'@'を付加した場合は、選択されたファイルが存在する場合に上書きを確認するメッセージボックスを表示します。(ex. ”@AllFiles/*.*/TextFiles/*.txt”)
pDefExt は、デフォルトの拡張子をピリオドを含めないで指定します。(ex. “txt”)
デフォルトの拡張子は、ユーザが拡張子を指定しないでファイル名を入力した場合に、付加されます。
初期のフォルダやファイルを指定する場合は、pBuf で示すバッファに当該パス名を格納しておきます。それ以外の場合は、pBuf で示されるバッファは空文字列(“”)でなければなりません。

戻り値 ： TRUE　：ＯＫボタンによりダイアログボックスを閉じた
FALSE ：キャンセル

27.1.2. オープンファイル名取得 (AjcGetOpenFile)

形　式 ： BOOL　AjcGetOpenFile(HWND hWnd, C_UTP pTitle, C_UTP pFilter, C_UTP pDefExt, UTP pBuf, UI lBuf);

引　数 ： hWnd　　－ オーナーウインドゥハンドル （不要時はＮＵＬＬ）
pTitle　－ ダイアログボックスのタイトル（ＮＵＬＬを指定した場合は「ファイルを開く」を仮定）
pFilter － フィルタ（不要時はＮＵＬＬ）
pDefExt － 既定の拡張子（ピリオド (.) は指定不要，不要時はＮＵＬＬ）
pBuf　　－ 取得したファイル名を格納するバッファのアドレス
lBuf　　－ 取得したファイル名を格納するバッファのバイト数／文字数

説　明 ： ダイアログボックス操作により、オープン・ファイル名を取得します。
pfilter は、表示するファイル名を制限するためのワイルドカードを指定するものであり、「項目名／ワイルドカード」のペアで、複数指定可能です。(”AllFiles/*.*/TextFiles/*.txt”)
pDefExt は、デフォルトの拡張子をピリオドを含めないで指定します。(ex. “txt”)
デフォルトの拡張子は、ユーザが拡張子を指定しないでファイル名を入力した場合に、付加されます。
初期のフォルダやファイルを指定する場合は、pBuf で示すバッファに当該パス名を格納しておきます。それ以外の場合は、pBuf で示されるバッファは空文字列(“”)でなければなりません。

戻り値 ： TRUE　－ ＯＫボタンによりダイアログボックスを閉じた
FALSE － キャンセル

27.1.3. 複数のオープンファイル名取得 (AjcGetOpenFile)

形　式 ： UTP * AjcGetOpenFiles(HWND hWnd, C_UTP pInitialPath, C_UTP pTitle, C_UTP pFilter, C_UTP pDefExt,
BOOL fMulti, UIP pNum);

引　数 ： hWnd – オーナーウインドゥハンドル （不要時はＮＵＬＬ）
pInitialPath – 初期パス名（不要時はＮＵＬＬ）
pTitle – ダイアログボックスのタイトル（ＮＵＬＬを指定した場合は「ファイルを開く」を仮定）
pFilter – フィルタ（不要時はＮＵＬＬ）
pDefExt – 既定の拡張子（ピリオド (.) は指定不要, 不要時はＮＵＬＬ）
fMulti – TRUE:複数ファイル選択可能とする， FALSE:１ヶのファイルのみ選択可能
pNum – 取得したファイルの個数を格納するバッファのアドレス

説　明 ： ダイアログボックス操作により、オープンファイル名を取得します。
fMulti=TRUE とした場合は、複数のファイルを選択することができます。
pfilter は、表示するファイル名を制限するためのワイルドカードを指定するものであり、「項目名／ワイルドカード」のペアで、複数指定可能です。(”AllFiles/*.*/TextFiles/*.txt”)
pDefExt は、デフォルトの拡張子をピリオドを含めないで指定します。(ex. “txt”)
デフォルトの拡張子は、ユーザが拡張子を指定しないでファイル名を入力した場合に、付加されます。
初期のフォルダやファイルを指定する場合は、pInitialPath で当該パス名を指定します。

この関数は、ＯＫボタンにより正常にファイル名群を取得した場合、内部的に確保したバッファにファイル名群へのポインタテーブルを作成し、このポインタテーブルの先頭アドレスを返します。
この内部的に確保したバッファは、使用後に、AjcReleaseOpenedFilesArray 関数により開放しなければなりません。

以下に、取得したすべてのファイル名を表示するサンプルコードを示します。

```
UTP *pArr;
UI  i, n;
if (pArr = AjcGetOpenFiles(hWnd, TEXT(“c:¥¥temp”, “title”, “All/*.*”), NULL, TRUE, &n)) {
    for (i=0; i<n; i++) {
        AjcPrintF(TEXT(“%s¥n”), pArr[i]);
    }
    AjcReleaseOpenedFilesArray(pArr);
}
```

戻り値 ： ≠NULL – 正常（ファイル名群へのポインタテーブルの先頭アドレス）
＝NULL – キャンセル

27.1.4. AjcGetOpenFiles で取得したファイル名群バッファの開放 (AjcReleaseOpenedFilesArray)

形　式 ： VO AjcReleaseOpenedFilesArray(VOP pArr);

引　数 ： pArr – gsGetOpenFiles()の戻り値

説　明 ： GsGetOpenFiles で取得した、ファイル名群バッファを開放します。

戻り値 ： なし

27.1.5. フォルダ名取得 (AjcGetFolderName)

形　式 ： BOOL　AjcGetFolderName(HWND hWnd, C_UTP pWndTtl, C_UTP pBoxTtl, UT pBuf[MAX_PATH], BOOL fTailIsDelimiter);

引　数 ： hWnd　　- オーナーウインドゥハンドル　　(不要時はＮＵＬＬ)
pWndTtl　　- ウィンドタイトルバーに表示するテキストのアドレス　　(不要時はＮＵＬＬ)
pBoxTtl　　- ダイアログボックス内に表示するタイトルテキストのアドレス (不要時はＮＵＬＬ)
pBuf　　- 取得したフォルダ名を格納するバッファのアドレス
fTailIsDelimiter – TRUE を指定した場合、フォルダ名の末尾が常に「¥」となります。

説　明 ： ダイアログボックス操作により、フォルダ名を取得します。

戻り値 ： TRUE　- ＯＫボタンにより終了した
FALSE – キャンセル／エラー

27.1.6. 起動したプログラムのフォルダパス名／プログラム名取得 (AjcGetApp{Path | Name})

形　式 ： BOOL　AjcGetAppPath(UTP pBuf, UI bfl)
BOOL　AjcGetAppName(UTP pBuf, UI bfl)

引　数 ： pBuf - 取得したフォルダ名／プログラム名を格納するバッファのアドレス
bfl　- 取得したフォルダ名／プログラム名を格納するバッファのバイト数／文字数

説　明 ： pBuf,bfl で示されるバッファに、起動したプログラムのフォルダパス名／プログラム名を格納します。
AjcGetAppPath()は、起動したプログラムのフォルダ・パス名を取得します。(ex. “d:¥Sample¥Prog¥”)
取得したフォルダ名の末尾には、常に「¥」が付加されます。
AjcGetAppName()は、起動したプログラムのプログラム名を取得します。(ex. “Sample1.exe”)

戻り値 ： TRUE　- 成功
FALSE - 失敗

27.1.7. ウインドを所有するプロセスの実行ファイル・パス名取得 (AjcGetAppPathByHWnd)

形　式 ： BOOL　AjcGetAppPathByHWnd (HWND hWnd, UTP pBuf, UI bfl)
BOOL　AjcGetAppPathEx　　(HWND hWnd, UTP pBuf, UI bfl)

引　数 ： hWnd – ウインドハンドル
pBuf - 取得したパス名を格納するバッファのアドレス
bfl　- 取得したパス名を格納するバッファのバイト数／文字数

説　明 ： hWnd で指定したプロセスの実行ファイル・パス名(ex. “d:¥Sample¥Prog¥Sample1.exe”)を、pBuf, bfl で示されるバッファに格納します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

28. ボリュームラベル操作

ボリュームラベル名の取得に関するＡＰＩです。

ボリュームラベル表現のパス名

ボリュームラベル表現のパス名とは、ドライブ（'a' ～ 'z'）をボリューム名を '<' と '>' で囲み表現したパス名です。
例えば、Ｇドライブのボリュームラベルが "USB-32G" である場合、以下のパス名は同じ意味を持ちます。

"G:¥work"
"<USB-32G>:¥work"

この２つのパスは、AjcVolExpPathToNormalPath()と AjcNormalPathToVolExpPath()により相互変換できます。

但し、ボリュームラベル表現のパス名は、Windows API や、本ライブラリで指定するパス名として使用できませんので、アプリケーションで相互変換して使用する必要があります。

28.1. サポートＡＰＩ

ボリュームラベル操作のサポートＡＰＩ一覧を以下に示します。

#	関数名	内容	備考
1	AjcGetVolumeLabel	ドライブのボリュームラベル名取得	
2	AjcGetDriveByVolumeLabel	ボリュームラベル名からドライブ種別取得	
3	AjcIsVolExpPath	ボリューム表現のパス名チェック	
4	AjcVolExpPathToNormalPath	ボリューム表現のパス名を通常のパス名に変更	
5	AjcNormalPathToVolExpPath	通常のパス名をボリューム表現のパス名に変更	

28.1.1. ドライブのボリュームラベル名取得 (AjcGetVolumeLabel)

形　式 ： BOOL　AjcGetVolumeLabel (UI drive, UTP pBuf, UI lBuf)

引　数 ： drive - ドライブレター('a' ～ 'z' or 'A' ～ 'Z')
pBuf　- 取得したボリュームラベル名を格納するバッファのアドレス
lBuf　- 取得したボリュームラベル名を格納するバッファのバイト数／文字数

説　明 ： drive で指定したドライブのボリュームラベル名を、pBuf, bfl で示されるバッファに格納します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

28.1.2. ボリュームラベル名からドライブ種別取得(AjcGetDriveByVolumeLabel)

形　式 ： UI　　AjcGetDriveByVolumeLabel(C_UTP pVol)

引　数 ： pVol – ボリュームラベル名へのポインタ

説　明 ： 指定したボリュームラベル名のドライブ種別（'a' ～ 'z'）を取得します。

戻り値 ： 'a' ～ 'z' - 成功（ドライブ種別）
0　　- 失敗

28.1.3. ボリューム表現のパス名チェック(AjcIsVolExpPath)

形　式 ： BOOL　AjcIsVolExpPath (C_UTP pPath)

引　数 ： pVol – パス名のポインタ

説　明 ： 指定したパスが、ボリューム表現のパス名かチェックします。
指定したパス名の先頭が「<・・・>:」であるかをチェックします。

戻り値 ： TRUE　- ボリューム表現のパス名　　（先頭が「<・・・>:」である）
FALSE - ボリューム表現のパス名以外（先頭が「<・・・>:」 以外）

28.1.4. ボリューム表現のパス名を通常のパス名に変更(AjcVolExpPathToNormalPath)

形　式 ： BOOL　AjcVolExpPathToNormalPath (C_UTP pVolExp, UT path[MAX_PATH])

引　数 ： pVolExp - ボリューム表現のパス名のポインタ
path　　- 変換した通常のパス名を格納するバッファのアドレス（バッファのバイト数／文字数は MAX_PATH）

説　明 ： 指定したボリューム表現のパス名を、通常のパス名に変換します。
pVolExp がボリューム表現のパス名以外である場合は、pVolExp の内容を path にコピーし、TRUE を返します。
pVolExp で指定されたボリュームラベル名が見つからない場合は、pVolExp の内容を path にコピーし、FALSE を返します。
pVolExp と path は重複可能です。

戻り値 ： TRUE　- 成功
FALSE - 失敗

28.1.5. 通常のパス名をボリューム表現のパス名に変更(AjcNormalPathToVolExpPath)

形　式 ： BOOL　AjcVolExpPathToNormalPath (C_UTP pPath, UT VolExp[MAX_PATH])

引　数 ： pVolExp – 通常のパス名へのポインタ
path　　- 変換したボリューム表現のパス名を格納するバッファのアドレス（バッファのバイト数／文字数は MAX_PATH）

説　明 ： 指定した通常のパス名を、ボリューム表現のパス名に変換します。
pPath の先頭がドライブ指定（”a:” ～ ”z:” or ”A:” ～ ”Z:”）以外である場合は、pPath の内容を VolExp にコピーし、TRUE を返します。
pPath で指定されたドライブが見つからない場合は、pVolExp の内容を path にコピーし、FALSE を返します。
pPath と VolExp は重複可能です。

戻り値 ： TRUE　- 成功
FALSE - 失敗

29. ファイル検索

サブフォルダを含む、全フォルダからファイルを検索します。

29.1. サポートＡＰＩ

ファイル検索ＡＰＩ一覧を以下に示します。

#	関 数 名	内容
1	AjcSearchFilesEx	サブディレクトリ下を含めたファイル／ディレクトリ検索
2	AjcSearchMyComputerEx	マイコンピュータ内のファイル／ディレクトリ検索

29.1.1. サブディレクトリ下を含めたファイル／ディレクトリ検索 (AjcSearchFiles)

形　式 ： int AjcSearchFilesEx(C_UTP pDir, C_UTP pWild, BOOL fSubDir, BOOL fNtcDir, UX cbp
BOOL (CALLBACK *cbFind)(UI nest, UTP pPath, UTP pName, UI attrib, UI wtime, UX cbp));

引　数 ： pDir　　- ファイル／ディレクトリを検索するディレクトリ・パス名のアドレス
pWild　- ワイルドカード群のアドレス（複数指定時は、';' or '/' で区切る。NULL 指定時は全ファイルを検索）
fSubDir　- サブディレクトリ検索フラグ（TRUE:検索する, FALSE:検索しない）
fNtcDir　- 検索中のフォルダ通知フラグ（TRUE:通知する, FALSE：通知しない）
cbp　　- コールバックパラメタ
cbFind　- 検索したファイル／ディレクトリの情報通知用コールバック関数のアドレス

説　明 ： pDir で指定されたディレクトリ下の、ファイル／ディレクトリを検索し、当該ファイル／ディレクトリ群の情報をコールバック関数にて通知します。
pWild は、検索するファイルをワイルドカードで指定します。複数指定する場合は、各ワイルドカードを';' あるいは '/' で区切ります。(ex. "*.txt/*.c/*.h" or "*.txt;*.c;*.h")
fSubDir は、サブディレクトリも検索するか否かを指定します。
fSubDir=FALSE とした場合はサブディレクトリを検索しません。(pDir で指定されたディレクトリ直下のみ検索します)
fSubDir=TRUE とした場合は、サブディレクトリ（さらにサブディレクトリがある場合は、そのサブディレクトリも）検索します。
コールバック関数の戻り値が TRUE の場合は、ファイル／ディレクトリ検索を継続し、FALSE の場合は検索を中止します。

戻り値 ： 検索されたファイルの個数

備　考 ： 「pWild」で指定するワイルドカードには、フォルダパス（末尾のフォルダパス）を含むことができます。
例えば、pWild="¥¥Sub1¥¥Sub2¥¥*.txt" と指定した場合、「・・・¥Sub1¥Sub2¥」フォルダ直下の「*.txt」で示されるフォルダ／ファイルが検索されます。

コールバック：

cbFind（検索中のフォルダ通知）

形　式　：　BOOL CALLBACK *cbFind*(UI nest, UTP pPath, UTP pName, UI attrib, UI wtime, UX cbp)

引　数　：　**nest　　－ －１：検索中のフォルダ通知を意味します**
pPath　　－ 検索中のフォルダパス名（絶対パス名）
pName　　－ 空文字列(“”)
attrib　　－ 不定
wtime　　－ 不定
cbp　　－ コールバックパラメタ

説　明　：　AjcSearchFilesEx()／AjcSearchFiles()による検索中のフォルダを通知します。
pPath は、検索中フォルダの絶対パス名を示します。

戻り値　：　TRUE ： ファイル検索を継続する
FALSE ： ファイル検索を中止する

cbFind（検索したファイル通知）

形　式　：　BOOL CALLBACK *cbFind*(UI nest, UTP pPath, UTP pName, UI attrib, UI wtime, UX cbp)

引　数　：　**nest　　－ ０以上：サブディレクトリの深さ**
pPath　　－ 検索されたファイルのパス名（絶対パス名）
pName　　－ 検索されたファイルの名称（ドライブとディレクトリを除いたファイル名部分）
attrib　　－ ファイル／ディレクトリの属性
wtime　　－ ファイル／ディレクトリの最終書き込み時刻（1970/1/1 00:00:00 からの通算秒数(UTC 時刻)）
cbp　　－ コールバックパラメタ（AjcSearchFiles で指定した cbp）

説　明　：　AjcSearchFiles()により検索されたファイル／ディレクトリの情報をを通知します。
nest は、検索されたファイル／ディレクトリの深さ（サブディレクトリの段数）を示します。
指定されたディレクトリ直下である場合は、nest=0 となります。
pPath は、検索されたファイル／ディレクトリの絶対パス名を示します。
pName は、検索されたファイル／ディレクトリの名称を示します。

attribu は、検索されたファイル／ディレクトリの属性を、以下の名称の合成値で示します。
・_A_ARCH　(=0x20) ： アーカイブファイル（更新されたファイル）
・_A_SUBDIR(=0x10) ： サブディレクトリ
・_A_SYSTEM(=0x04) ： システムファイル
・_A_HIDDEN(=0x02) ： 隠しファイル
・_A_RDONLY(=0x01) ： 読み出し専用ファイル

何の属性も無い、通常のファイルは、attrib=0 となります。

wtime は、ファイルの最終更新時刻を、1970/1/1 00:00:00 からの通算秒数で示します。

戻り値　：　TRUE ： ファイル検索を継続する
FALSE ： ファイル検索を中止する

サンプルコード１

以下のサンプルコードは、「c:¥MyDir」下の、拡張子が「.c」「.h」である全ファイルを検索し、表示します。
但し、サブディレクトリ，隠しファイルとシステムファイルは表示対象外とします。

```
#include    <AjrCstXX.h>

static  UI  Count = 0;

BOOL CALLBACK cbFind(UI nest, UTP pPath, UTP pName, UI attrib, UI wtime, UX cbp)
{
    SYSTEMTIME  st, lt;

    if (!(attrib & (_A_SUBDIR | _A_HIDDEN | _A_SYSTEM))) {
        Count++;
        AjcTime1970ToSysTime(wtime, &st);
        AjcSysTimeToLocalTime(&st, &lt);
        AjcPrintF(TEXT("%2d   %02X   %s %s %s¥n"), nest, attrib, AjcDateStr(&lt), AjcTimeStr(&lt, FALSE),
pName);
        AjcPrintF(TEXT("          %s¥n"), pPath);
    }
    return TRUE;
}

main()
{
    int     n;

    AjcPrintF(TEXT("Nest Att  Time                FileName¥n"));
    n = AjcSearchFilesEx(TEXT("c:¥¥MyDir"), TEXT("*.c/*.h"), TRUE, FALSE, 0, cbFind);
    AjcPrintF(TEXT("== Find %d/%d files ==¥n"), Count, n);
    getchar();
}
```

サンプルコード２

以下のサンプルコードは、「c:¥¥Program files (x86)」下において、「¥Common7¥ide¥」フォルダ直下の「devenv.*」を検索し、表示します。また、検索中のフォルダを「Seartching・・」と表示します。

```
#pragma warning(disable:4996)
#include    <AjrCstXX.h>
#include    <stdio.h>
#include    <tchar.h>

int     Count = 0;

static BOOL CALLBACK cbFind(UI nest, UTP pPath, UTP pName, UI att, UI wtime, UX cbp)
{
    int     stl = AjcTextLen(pPath);

    if (nest == -1) {           //  検索中フォルダの通知
        _tprintf(TEXT("Searching %s'¥r"), pPath);
    }
    else {                      //  見つかったフォルダ／ファイルの通知
        _tprintf(TEXT("%5d : %-70s¥n"), ++Count, pPath);
    }
    return TRUE;
}

main()
{
    AjcSearchFilesEx(TEXT("c:¥¥Program files (x86)"), TEXT("¥¥Common7¥¥ide¥¥devenv.*"), TRUE, TRUE, 0, cbFind);
    _tprintf(TEXT("%-79s¥n"), TEXT("--- Hit Enter key !! ---"));
    getchar();
}
```

サンプルコード３

以下のサンプルコードは、「C:¥Program Files (x86)¥Microsoft Visual Studio 8」下、「include」フォルダを検索し、さらに、「include」フォルダ内からファイル名の先頭が「S」であるヘッダファイル(.h)を検索し表示します。

```
#pragma warning(disable:4996)
#include    <AjrCstXX.h>
#include    <stdio.h>
#include    <tchar.h>

BOOL CALLBACK cbFindH  (UI nest, UTP pPath, UTP pName, UI attrib, UI wtime, UX cbp)
{
    _tprintf(TEXT("¥t%s¥n"), pName);
    return TRUE;
}

BOOL CALLBACK cbFindInc(UI nest, UTP pPath, UTP pName, UI attrib, UI wtime, UX cbp)
{
    if (attrib & _A_SUBDIR) {
        _tprintf(TEXT("<<< %s >>>¥n"), pPath);
        AjcSearchFilesEx(pPath, TEXT("s*.h"), FALSE, FALSE, 0, cbFindH);
    }
    return TRUE;
}

main()
{
    AjcSearchFilesEx(TEXT("C:¥¥Program Files (x86)¥¥Microsoft Visual Studio 8"), TEXT("include"), TRUE, FALSE, 0,
cbFindInc);
    _tprintf(TEXT("¥nHit Enter Key !"));
    getchar();
}
```

29.1.2. マイコンピュータ内のファイル／ディレクトリ検索 (AjcSearchMyComputer)

形　式 ： int AjcSearchMyComputerEx(C_UTP pPath, C_UTP pWild, BOOL fSubDir, BOOL fNtcDir, UX cbp,
BOOL (CALLBACK *cbFind)(UI nest, UTP pPath, UTP pName, UI attrib, UI wtime, UX cbp));

引　数 ： pPath　- ファイル／ディレクトリを検索するディレクトリ・パス名のアドレス（NULL 指定時は全ディレクトリ検索）
pWild　- ワイルドカード群のアドレス（複数指定時は、';' or '/' で区切る。NULL 指定時は全ファイルを検索）
fRemote　- ネットワークドライブ検索フラグ（TRUE:検索する，FALSE:検索しない）
fNtcDir　- 検索中のフォルダ通知フラグ（TRUE:通知する，FALSE：通知しない）
cbp　- コールバックパラメタ
cbFind　- 検索したファイル／ディレクトリの情報通知用コールバック関数のアドレス

説　明 ： マイコンピュータ内の全固定ドライブとネットワークドライブから、pPath で指定されたディレクトリ下の、ファイル／ディレクトリを検索し、当該ファイル／ディレクトリ群の情報をコールバック関数にて通知します。
pPath では、ルート直下からのディレクトリを指定してください。(つまり、'¥'から始まるディレクトリを指定)
pPath=NULL を指定した場合は、マイコンピュータ内の全ディレクトリを検索します。
pWild は、検索するファイルをワイルドカードで指定します。複数指定する場合は、各ワイルドカードを';' あるいは '/' で区切ります。(ex. "*.txt/*.c/*.h" or "*.txt;*.c;*.h")
fRemote は、ネットワークドライブも検索するか否かを指定します。
fRemote=FALSE とした場合はネットワークドライブを検索しません。
fRemote=TRUE とした場合は、ネットワークドライブも検索します。
コールバック関数の戻り値が TRUE の場合は、ファイル／ディレクトリ検索を継続し、FALSE の場合は検索を中止します。

戻り値 ： 検索されたファイルの個数

備　考 ： 「pWild」で指定するワイルドカードには、フォルダパスを含むことができます。
例えば、pWild="¥¥Sub1¥¥Sub2¥¥*.txt" と指定した場合、「・・・¥Sub1¥Sub2¥」フォルダ直下の「*.txt」で示されるフォルダ／ファイルが検索されます。

コールバック：

cbFind (検索中のフォルダ通知)

形　式 ： BOOL CALLBACK *cbFind*(UI nest, UTP pPath, UTP pName, UI attrib, UI wtime, UX cbp)

引　数 ： **nest　- -1 ： 検索中のフォルダ通知**
pPath　- 検索中フォルダのパス名（絶対パス名）
pName　- 空文字列("")
attrib　- 不定
wtime　- 不定
cbp　- コールバックパラメタ

説　明 ： AjcSearchMyComputer()による検索中のフォルダを通知します。
pPath は、検索中フォルダの絶対パス名を示します。

戻り値 ： TRUE ： ファイル検索を継続する
FALSE ： ファイル検索を中止する

cbFind（検索したファイル通知）

形　式 ： BOOL CALLBACK *cbFind*(UI nest, UTP pPath, UTP pName, UI attrib, UI wtime, UX cbp)

引　数 ： **nest　　０以上 ： サブディレクトリの深さ**
pPath　- 検索されたファイルのパス名（絶対パス名）
pName　- 検索されたファイルの名称（ドライブとディレクトリを除いたファイル名部分）
attrib　- ファイル／ディレクトリの属性
wtime　- ファイル／ディレクトリの最終書き込み時刻（1970/1/1 00:00:00 からの通算秒数(UTC 時刻)）
cbp　- コールバックパラメタ（AjcSearchFiles で指定した cbp）

説　明 ： AjcSearchMyComputerEx()／AjcSearchMyComputer()により検索されたファイル／ディレクトリ情報をを通知します。
nest は、検索されたファイル／ディレクトリの深さ（サブディレクトリの段数）を示します。
AjcSearchMyComputer パス直下である場合は、nest=0 となります。
pPath は、検索されたファイル／ディレクトリの絶対パス名を示します。
pName は、検索されたファイル／ディレクトリの名称を示します。

attribu は、検索されたファイル／ディレクトリの属性を、以下の名称の合成値で示します。
・_A_ARCH　(=0x20) ： アーカイブファイル（更新されたファイル）
・_A_SUBDIR(=0x10) ： サブディレクトリ
・_A_SYSTEM(=0x04) ： システムファイル
・_A_HIDDEN(=0x02) ： 隠しファイル
・_A_RDONLY(=0x01) ： 読み出し専用ファイル

何の属性も無い、通常のファイルは、attrib=0 となります。

wtime は、ファイルの最終更新時刻を、1970/1/1 00:00:00 からの通算秒数で示します。

戻り値 ： TRUE ： ファイル検索を継続する
FALSE ： ファイル検索を中止する

サンプルコード

以下のサンプルコードは、マイコンピュータ内の全ての固定ディスクドライブとネットワークドライブの「¥work」ディレクトリ下の「x.txt」ファイルを検索し、表示します、

```
#include    <AjrCstXX.h>
#include    <stdio.h>
#include    <tchar.h>

static BOOL CALLBACK cb(UI nest, UTP pPath, UTP pName, UI att, UI wtime, UX cbp)
{
    _tprintf(TEXT("%s¥n"), pPath);
    return TRUE;
}

main()
{
    AjcSearchMyComputerEx(TEXT("¥¥work"), TEXT("x.txt"), TRUE, FALSE, 0, cb);
    getchar();
}
```

30. ファイル／フォルダ操作

ファイルやフォルダを操作するＡＰＩ群です。

30.1. サポートＡＰＩ

#	関 数 名	内容
1	AjcCopyFolderStruct AjcCopyFolderStructEx	フォルダ構造のコピー
2	AjcCopyFiles	フォルダ下のファイル群をコピー
3	AjcCopyFile[Ex]	ファイルのコピー（１ファイルコピー，コピー中止機能つき）
4	AjcRemoveFolder AjcCleanFolder	フォルダ削除／フォルダクリーンアップ
5	AjcEnumFilesMatchingList	２つのフォルダ下のファイル群・突合せリスト生成
6	AjcGetParentDirectory	親ディレクトリのパス取得
7	AjcCreateDirectory[Ex]	ディレクトリの作成
8	AjcCreateDirectoryStruct[Ex]	多階層ディレクトリ構造の一括作成
9	AjcPathMatchSpec	パスがワイルドカード[群]の指定に一致するかチェックする
10	AjcPathMatchStr	パスが指定文字列[群]を含むかチェックする
11	AjcIsPathBackslash[Ex]	パスの末尾がバックスラッシュ「¥」かチェックする
12	AjcPathCat	２つのパスの間に必ず１つの「¥」を挿入しパスを結合する
13	AjcPathCmp	文字列末尾の「¥」を除去して比較
14	AjcChangeFNameToCorrect	ファイル名に使用できない文字を特定の文字に変換する
15	AjcPathExists	パスが存在するかチェックする
16	AjcPathIsDirectory	パスがディレクトリかチェックする
17	AjcPathIsEmptyDirectory	パスが空のディレクトリかチェックする
18	AjcPathIsFile	パスがファイルかチェックする
19	AjcGetFileSize	ファイルサイズ取得
20	AjcGetFileTime1970	ファイルタイム取得（1970/01/01 00:00:00 からの通算秒数）
21	AjcFileCompare	ファイル比較
22	AjcGetFileTime	ファイル／ディレクトリの日時取得
23	AjcSetFileTime AjcSetFileTimeBySysTime	ファイル／ディレクトリの日時設定
24	AjcIsSamePath	２つのパスが同一パスかチェックする
25	AjcIsUnderPath	２つのパスで、片方がサブパスであるかをチェックする

30.1.1. フォルダ構造のコピー(AjcCopyFolderStruct

形　式 ： BOOL AjcCopyFolderStruct (C_UTP pPathFrom, C_UTP pPathTo, AJCFSC_OPT opt);
BOOL AjcCopyFolderStructEx(C_UTP pPathFrom, C_UTP pPathTo, AJCFSC_OPT opt, UX cbp,
BOOL (CALLBACK *cbNotify)(C_UTP pPathFrom, C_UTP pPathTo, EAJCCFS ntc, UX cbp),
UTP (CALLBACK *cbQuery)(C_UTP pPathFront, UTP pNewDir, UX cbp));

引　数 ： pPathFrom - コピー元の先頭フォルダのパス名
pPathTo - コピー先の先頭フォルダのパス名
opt - オプション
cbp - コールバックパラメタ
cbNotify - ディレクトリのコピー結果を通知するためのコールバック関数（不要時は NULL）
cbQuery - ディレクトリのコピー／名称を変更問い合わせ用のコールバック関数（不要時は NULL）

説　明 ： 「pPathTo」で指定したフォルダの下に、「pPathFrom」下のフォルダ構造を作成します。
「pPathTo」下にフォルダを作成するだけで、ファイルのコピーは行いません。
オプションの内容は以下のとおりです。

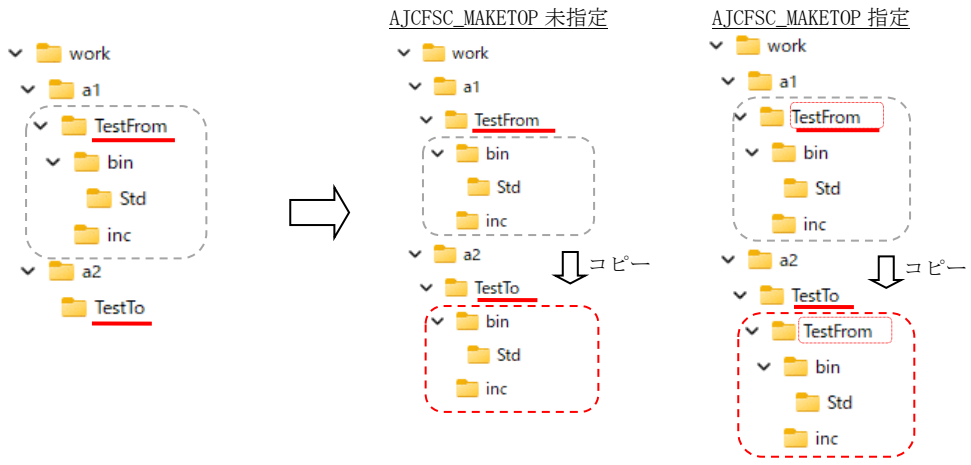
●コピー先フォルダのタイムスタンプ（ファイルの作成日時と更新日時）は、「opt」の指定に従います。

コピー先フォルダ / opt	同一フォルダ無し（新規作成）	同一フォルダあり
AJCFSC_NONE	実際にフォルダを作成した現在日時	既存フォルダの日時のまま
AJCFSC_SAMETIME	コピー元フォルダの日時と同じ	既存フォルダの日時のまま
AJCFSC_FORCETIME	コピー元フォルダの日時と同じ	コピー元フォルダの日時と同じ

●その他のオプション

opt	内容
AJCFSC_MAKETOP	転送元末尾フォルダ名を転送先の先頭フォルダとして作成する（下記、例参照）
AJCFSC_ALLOWEXIST	同一フォルダが存在してもエラーとしないで、処理を続行する

（例）AjcCopyFolderStruct(TEXT("f:¥¥work¥¥**a1**¥¥TestFrom"), TEXT("f:¥¥work¥¥**a2**¥¥TestTo"), ・・・);



AjcCopyFolderStructEx()では、コールバック関数を介してコピー先のフォルダ名を変更することもできます。
既に、同一名称のフォルダが存在する場合は、コールバックにより「AJCCFS_EXIST」を通知します。
既に、同一名称のフォルダが存在する場合でも、(AJCFSC_ALLOWEXIST 指定では)処理を続行します。
既に、フォルダと同一名称の「ファイル」が存在する場合は、エラーとなります。

戻り値 ： TRUE - 成功
FALSE - 失敗／中止

コールバック関数　：

cbNotify（フォルダコピー結果の通知）

形　式　：　BOOL CALLBACK cbNotify(C_UTP pPathFrom, C_UTP pPathTo, EAJCCFS ntc, UX cbp);

引　数　：　pPathFrom - コピー元フォルダのパス名
pPathTo　- コピー先フォルダのパス名
ntc　　- 通知コード　・AJCCFS_SUCCESS　- ディレクトリ作成 成功
・AJCCFS_FAILURE　- ディレクトリ作成 失敗
・AJCCFS_EXIST　　- 既に同一パスが存在する
cbp　　- コールバックパラメタ

説　明　：　コピー元とコピー先フォルダのパス名を通知します。

戻り値　：　TRUE　：コピー操作を続行する
FALSE：コピーを中止する

cbQuery（フォルダコピー／フォルダ名変更の問い合わせ）

形　式　：　UTP CALLBACK cbQuery(C_UTP pPathFront, UTP pNewDir, UX cbp);

引　数　：pPathFront - 新たなフォルダを作成する位置（１つ上のフォルダパス）
pNewDir　- 新たに作成するフォルダ名
cbp　　- コールバックパラメタ

説　明　：　新たに作成するフォルダ名を通知します。
pPathFront で示すフォルダのサブフォルダとして、pNewDir で示すフォルダを作成することを通知します。

作成するフォルダ名を変更しない場合は、pNewDir を返します。
作成するフォルダ名を変更する場合は、pNewDir で示すバッファに変更するフォルダ名を設定し、pNewDir を返します。
pNewDir で示すバッファの容量は、MAX_PATH バイト／MAX_PATH 文字です。
このフォルダを作成しない場合は、NULL を返します。

戻り値　：　≠NULL：フォルダの作成を行う（作成するフォルダ名称へのポインタ）
＝NULL：このフォルダは作成しない

備　考　：　pPathFront はパス名を、pNewDir は新たに作成するフォルダ名を示します。
例えば新たに作成するフォルダが「e:¥work¥sub」である場合、pPathFront=”e:¥work”, pNewDir=”sub”となります。

尚、pPathFront は多重文字列でコピー先のパス名の後に、コピー元のパス名が格納されています。
コピー先の新たに作成しようとしているパスが「e:¥work¥sub」で、コピー元のパス名が「d:¥source¥sub」である場合、pPathFront と pNewDr は、以下のように設定されます。

pPathFront→	e:¥work	¥0	d:¥source¥sub	¥0	¥0
pNewDir　→	sub	¥0			

30.1.2. ファイル群のコピー(AjcCopyFiles)

形　式 ： BOOL AjcCopyFiles (C_UTP pPathFrom, C_UTP pPathTo, C_UTP pWildCard, UI opt);

```
BOOL AjcCopyFilesEx(C_UTP pPathFrom, C_UTP pPathTo, C_UTP pWildCard, UI opt , UX cbp,
                    BOOL (CALLBACK *cbNotify  )(C_UTP pFileFrom, C_UTP pFileTo  , EAJCCFS ntc, UX cbp),
                    UTP  (CALLBACK *cbQuery   )(                 UTP pFileName, UIP pAtt   , UX cbp),
                    BOOL (CALLBACK *cbProgress)(ULL FileSize, ULL Copied, UX cbp));
```

引　数 ： pPathFrom　　- コピー元フォルダのパス名
pPathTo　　- コピー先フォルダのパス名
pWildCard　　- コピー／除外するファイルのワイルドカード（NULL 指定時は、全ファイルをコピーします）
（複数指定時は、セミコロン(;)か スラッシュ(/)で区切る。各ワイルドカードの両端の空白は無視します）
opt　　- オプション
cbNotify　　- ファイルのコピー結果を通知するためのコールバック関数　　（不要時は NULL）
cbQuery　　- ファイルのコピー／名称の変更問い合わせ用コールバック関数（不要時は NULL）
cbNtcProgress – ファイルコピーの進行状況通知用コールバック関数　　（不要時は NULL）

説　明 ： 「pPathFrom」で指定したフォルダ直下の（ワイルドカードで指定された）すべてのファイルを「pPathTo」で指定したフォルダ下へコピーします。
「pPathFrom」で指定されたコピー元ファイル群は「FILE_SHARE_READ」モードでファイルをオープンします。
「pPathFrom」で指定したフォルダ直下のファイルだけがコピーされ、サブフォルダ下のファイルはコピーされません。

opt(オプション)は、以下の合成値を指定します。

名称	値	内容
AJCCPF_NONE	0x0000	オプション無し
AJCCPF_CREATEALWAYS	0x0001	既にファイルが存在する場合、上書きする
AJCCPF_WILDEXC	0x0002	ワイルドカードに一致するファイルを除外する
AJCCPF_INP_SHARE_READ	0x0010	入力ファイルで読み出し共有
AJCCPF_INP_SHARE_WRITE	0x0020	入力ファイルで書き込み共有
AJCCPF_INP_SHARE_BOTH	0x0030	入力ファイルで読み書き共有
AJCCPF_OUT_SHARE_READ	0x0100	出力ファイルで読み出し共有
AJCCPF_OUT_SHARE_WRITE	0x0200	出力ファイルで書き込み共有
AJCCPF_OUT_SHARE_BOTH	0x0300	出力ファイルで読み書き共有
AJCCPF_ALL_SHARE_READ	0x0110	入出力ファイルで読み出し共有
AJCCPF_ALL_SHARE_WRITE	0x0220	入出力ファイルで書き込み共有
AJCCPF_ALL_SHARE	0x0330	入出力ファイルで読み書き共有

AjcCopyFiles()では、opt に AJCCPF_CREATEALWAYS を指定した場合、コピー先に同名ファイルが存在しても上書きコピーします。(「読み出し専用」「隠しファイル」や「システム」属性を持ったファイルも上書きコピーします。)
opt に AJCCPF_CREATEALWAYS を指定しない場合は、コピー先の同名ファイルを上書きしません。(処理は続行します)

AjcCopyFilesEx()では、コールバック関数を介して、ファイル名やファイル属性の変更、ファイルコピーの可否、および、ファイルコピーの中止を指定できます。

戻り値 ： TRUE　- 成功
FALSE - 失敗／中止（１つでもコピーを失敗した場合は、FALSE を返す）

コールバック関数　：

cbNotify（ファイルコピー結果の通知）

形　式 ： BOOL CALLBACK cbNotify(C_UTP pFileFrom, C_UTP pFileTo, EAJCCFS ntc, UX cbp);

引　数 ： pFileFrom - コピー元ファイル・パス名
pFileTo　- コピー先ファイル・パス名
ntc　　- 通知コード　・AJCCFS_SUCCESS　- ファイルコピー 成功
・AJCCFS_FAILURE　- ファイルコピー 失敗
・AJCCFS_EXIST　　- 既に同一ファイルが存在する
cbp　　- コールバックパラメタ

説　明 ： コピー元とコピー先フォルダのパス名を通知します。

戻り値 ： TRUE　：コピー操作を続行する
FALSE ：コピーを中止する

cbQuery（ファイルコピー／ファイル名変更の問い合わせ）

形　式　：　UTP CALLBACK cbQuery(C_UTP pFileFrom, C_UTP pFileTo, UTP pFileName, UIP pAtt, UX cbp);

引　数　：pFileFrom　- コピー元ファイル・パス名
pFileTo　- コピー先ファイル・パス名（ファイル名の部分は、pPathFrom と同じ）
pFileName　- コピー元ファイル名（パス名を含まないファイル名の部分のみ(ex. “Sample.txt”)）
pAtt　- ファイルの属性（FILE_ATTRIBUTE_{ARCHIVE/HIDDEN/NORMAL/READONLY/SYSTEM}等）へのポインタ
cbp　- コールバックパラメタ

説　明　：　新たに作成するコピー先のファイル名を通知します。
作成するファイル名を変更しない場合は、pFileName を返します。
作成するファイル名を変更する場合は、pFileName で示すバッファに変更するファイル名を設定します。
pFileName で示すバッファの容量は、MAX_PATH バイト／MAX_PATH 文字です。
コピー先のファイル属性を変更するには、pAtt で示すバッファのファイル属性を変更してください。
このファイルを作成する（コピーする）場合は、戻り値＝pFileName とします。
このファイルを作成しない（コピーしない）場合は、NULL を返します。

戻り値　：　≠NULL：ファイルの作成を行う（作成するファイル名へのポインタ）
＝NULL：このファイルは作成しない

cbProgress（ファイルコピー状況通知）

AjcCopyFile()のコールバック関数(cbProgress)を参照してください。

30.1.3.　ファイルコピー(AjcCopyFile[Ex])

形　式　：
```
BOOL  AjcCopyFile(C_UTP pFrom, C_UTP pTo, UI opt, UX cbp,
                  BOOL (CALLBACK *cbProgress)(ULL FileSize, ULL Copied, UX cbp));
BOOL  AjcCopyFileEx(C_UTP pFrom, C_UTP pTo, UI opt, BOOL *pfCancel, UX cbp,
                  BOOL (CALLBACK *cbProgress)(ULL FileSize, ULL Copied, UX cbp));
```

引　数　：
pFrom　- コピー元ファイルパス名
pTo　- コピー先ファイルパス名／コピー先フォルダパス名
opt　- オプション（AjcCopyFiles[Ex]()を参照）
pfCancel　- キャンセルを示すフラグを格納するバッファ（不要時は NULL）
cbp　- コールバックパラメタ
cbProgress　- ファイルコピー状況通知用コールバック関数（不要時は NULL）

説　明　：　「pFrom」で指定したファイルを、「pTo」で指定したファイル／フォルダにコピーします。
pFrom で指定されたコピー元ファイルは「FILE_SHARE_READ」モードでファイルをオープンします。
pTo でフォルダを指定した場合は、コピー元ファイル名と同名のファイルを作成します。
コールバック関数を介して、ファイルコピーを中止することもできます。
AjcCopyFileEx()ではコールバックからキャンセルが返された場合は、pfCancel で示すバッファに TRUE が設定され、FALSE を返します。

戻り値　：　TRUE　- 成功
FALSE – 失敗／中止

コールバック関数　：

cbProgress（ファイルコピー状況通知）

形　式　：　BOOL CALLBACK *cbProgress*(ULL FileSize, ULL Copied, UX cbp);

引　数　：　FileSize　- コピー元ファイルのバイト数
Copied　- コピー済みバイト数
cbp　- コールバックパラメタ

説　明　：　ファイルのコピー状況を 4096 バイトコピー毎に通知します。

戻り値　：　TRUE　：ファイルコピーを続行する
FALSE：ファイルコピーを中止する

30.1.4. フォルダ削除／クリーンアップ(Ajc{Remove/Clean}Folder)

形　式 ： BOOL AjcRemoveFolder(C_UTP pPath, UX cbp, BOOL (CALLBACK *cbNotify)(C_UTP pPathRemoved, UI ntc, UX cbp));
BOOL AjcCleanFolder (C_UTP pPath, UX cbp, BOOL (CALLBACK *cbNotify)(C_UTP pPathRemoved, UI ntc, UX cbp));

引　数 ： pPath　　- 削除するフォルダのパス名
cbp　　- コールバックパラメタ
cbNotify　- 削除したファイル／ディレクトリのパス名を通知するためのコールバック関数（不要時はNULL）

説　明 ： 「pPath」で指定したフォルダとその下の全てのディレクトリとファイルを削除します。
AjcRemoveFolder()の場合、「pPath」で指定したフォルダ自身も削除します。
AjcCleanFolder ()の場合、「pPath」で指定したフォルダは、空のフォルダとなります。
フォルダ中に「読み取り専用」「隠しファイル」や「システム」属性を持っているファイル／フォルダも全て削除します。

フォルダ／ファイルの削除を失敗しても、以降のフォルダ／ファイルの削除は続行します。
但し、コールバック関数（cbNotify()）でFALSEを返した場合は、以降のフォルダ／ファイルの削除を中止します。

戻り値 ： TRUE　- 成功
FALSE - 失敗／中止

コールバック関数　：

cbNotify（削除したフォルダ／ファイルの通知）

形　式 ： BOOL CALLBACK cbNotify(C_UTP pPathRemoved, UI ntc, UX cbp);

引　数 ： pPathRemove　- 削除したフォルダ／ファイルのパス名
ntc　　- 通知情報
cbp　　- コールバックパラメタ

説　明 ： 削除したフォルダ／ファイルのパス名を通知します。
ntc（通知情報）の内容は以下のとおりです。

名称	値	内容	備考
AJCRMV_FILE	0x00	ファイルの削除	フォルダの場合、Bit0(0x01)がセットされる
AJCRMV_DIR	0x01	フォルダの削除	
AJCRMV_ERR	0x80	エラーフラグ	エラー発生時は Bit7(0x80)がセットされる

戻り値 ： TRUE　：フォルダ／ファイルの削除操作を続行する
FALSE：フォルダ／ファイルの削除操作を中止する

30.1.5. ２つのフォルダ下のファイル群・突き合せリスト列挙 (AjcEnumFilesMatchingList)

形　式 ： int AjcEnumFilesMatchingList(C_UTP pPath1, C_UTP pPath2, UI opt, UX cbp,
BOOL (CALLBACK *cbQuery)(UI id, C_UTP pPath, UI att, BOOL *pfDiscard, UX cbp),
BOOL (CALLBACK *cbEnum)(AJCFMLITEM item[2], UX cbp));

引　数 ： pPath1, pPath2 - 突き合せリストを作成する２つのフォルダ（絶対パス／カレントディレクトリからの相対パス）
opt - オプション(AJCFML_*XXXXX*)
cbp - コールバックパラメタ
cbQuery - ファイル通知用コールバック関数（不要時は NULL）
cbEnum - 突き合せリスト通知用コールバック関数（不要時は NULL）

説　明 ： 「pPath1」と「pPath2」で指定したフォルダ下におけるファイル／ディレクトリ群の突き合せリストを作成します。
「pPath1」と「pPath2」下に存在する全ファイル／ディレクトリをリストアップします。
各リスト項目は、２つのフォルダに後続するパス名が同じファイル／フォルダのペアで作成されます。

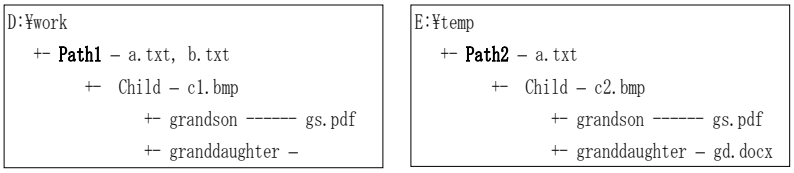
例えば、「D:¥Path1」と「E:¥Path2」が指定された場合、「D:¥Path1¥Child¥a.txt」と「E:¥Path2¥Child¥a.txt」は、同じ後続パス名「Child¥a.txt」のペアとしてリストアップします。

「pPath1」か「pPath2」が NULL の場合は、片方のみリストアップします。

opt（オプション）は、以下の以下の値のいずれかの値を指定します。

シンボル	値	内容
AJCFML_BOTH	0x03	ファイルとディレクトリの突合せリストを列挙
AJCFML_FILE	0x02	ファイルの突合せリストを列挙
AJCFML_DIR	0x01	ディレクトリの突合せリストを列挙
AJCFML_NOFILE	0x01	突き合せリストにファイルを含めない
AJCFML_NODIR	0x02	突き合せリストにディレクトリを含めない

例えば、２つのフォルダの構成が以下の内容で、「D:¥Path1」と「E:¥Path2」を指定した場合、下表のようにリストアップされます。(AJCFML_NODIR 指定時)



リストアップ内容

#	リスト項目内容	備考
1	D:¥work¥Path1¥a.txt E:¥temp¥Path2¥a.txt	.¥a.txt は両方に存在する
2	D:¥work¥Path1¥b.txt -	.¥b.txt は Path1 側にだけ存在する
3	D:¥work¥Path1¥Child¥c1.bmp -	.¥Child¥c1.bmp は Path1 側にだけ存在する
4	- E:¥temp¥Path2¥Child¥c2.bmp	.¥Child¥c2.bmp は Path2 側にだけ存在する
5	- E:¥temp¥Path2¥Cild¥ granddaughter¥gd.docx	.¥Child¥granddaughter¥gd.docx は Path2 側にだけ存在する
6	D:¥work¥Path1¥Child¥grandson¥gs.pdf E:¥temp¥Path2¥Child¥grandson¥gs.pdf	.¥Child¥grandson¥gs.pdf は両方に存在する

リスト項目の内容には、ファイルのパス名以外に各ファイルの「タイムスタンプ」「ファイル属性」「ファイルサイズ」が含まれます。
リストアップされた項目は、cbEnum()をコールバックすることにより、後続パス名の順に通知されます。

戻り値 ： ≧０ － 成功（リストアップした項目数）
＜０ － 中止／失敗

AJCFMLERR_NULL (-1) : pPath1,pPath2 が共に NULL
AJCFMLERR_PATH1(-2) : pPath1 で指定したフォルダが存在しない
AJCFMLERR_PATH2(-3) : pPath2 で指定したフォルダが存在しない
AJCFMLERR_SYSAPI(-4) ：システムＡＰＩエラー
AJCFMLERR_MEM(-5) ：メモリエラー
AJCFMLERR_STOP_BY_USER(-9) ：コールバックから中止指示

コールバック関数 ：

cbQuery（検出したファイルの通知）

形 式 ： BOOL CALLBACK *cbQuery*(UI id, C_UTP pPath, UI att, BOOL *pfDiscard, UX cbp);

引 数 ： id ― ２つのパスの識別（0:パス１側, 1：パス２側）
pPath ― ファイルパス名
att ― ファイル属性(FILT_ATTRIBUTE_*XXXXX*)
pfDiscard ― 除去フラグへのポインタ（デフォルト＝FALSE）
cbp ― コールバックパラメタ

説 明 ： 突合せリスト作成中に検出したファイルのパス名を順次通知します。
通知されたファイル／ディレクトリを突合せリストから除外する場合は、*pfDiscard=TRUE を設定します。

戻り値 ： TRUE ： 突合せリストの作成を続行する
FALSE ： 突合せリストの作成を中止する

cEnum（突き合せリスト項目の通知）

形 式 ： BOOL CALLBACK *cbEnum*(AJCFMLITEM item[2], UX cbp);

引 数 ： item ― ２つのファイルの情報（[0]:パス１側, [1]：パス２側）
cbp ― コールバックパラメタ

説 明 ： 突き合せリストリスト項目を順次通知します。

戻り値 ： TRUE ： 突き合せリストの列挙を続行する
FALSE ： 突き合せリストの列挙を中止する

突き合せリスト項目の内容：

```
// ヘッダ情報
typedef struct {
    UI      utc;                    // ファイル・タイムスタンプ（ＵＴＣ, 1970/1/1 00:00:00 からの通算秒数）
    UI      att;                    // ファイル属性（FILE_ATTRIBUTE_{READONLY, HIDDEN, SYSTEM, NORMAL, DIRECTORY 等}    ※1
    ULL     size;                   // ファイルサイズ（バイト数）
} AJCFMLHD, *PAJCFMLHD;

// 列挙情報
typedef struct {
    BOOL        fValid;                 // 有効フラグ（TRUE:有効(ファイル/ディレクトリは存在する), FALSE:無効（存在しない））
    AJCFMLHD    hd;                     // ヘッダ情報
    UT          path[MAX_PATH];         // ファイル／ディレクトリパス名      (ex. “D:¥path1¥sub1¥sub2¥Sample.txt”)
    UT          name[MAX_PATH];         // ファイル名と拡張子                (ex. “Sample.txt”)              ※2
    UT          tail[MAX_PATH];         // パス名の後部                      (ex. “sub1¥sub2¥Sample.txt”)    ※3
} AJCFMLITEMA, *PAJCFMLITEMA;
typedef const AJCFMLITEMA *PCAJCFMLITEMA;
```

※1：ディレクトリの場合、att には、Bit4(FILE_ATTRIBUTE_DIRECTORY = 0x10)がセットされます
※2：ディレクトリの場合、name には、末尾のパス名が設定されます(ex. 「d:¥work¥sub¥child」の場合、name には「child」が設定される)
※3：pPath1,pPath2 で指定したパスに続くパスの後部部分

30.1.6. 親ディレクトリのパス取得(AjcGetParentDirectory)

形　式 ： BOOL AjcGetParentDirectory(C_UTP pPath, BCP pBuf, UI lBuf, BOOL *pfRoot)

引　数 ： pPath　　- ディレクトリパス
pBuf　　- 親ディレクトリのパスを格納するバッファ（不要時は NULL）
lBuf　　- ディレクトリのパスを格納するバッファのバイト数／文字数
pfRoot　- 親ディレクトリがルートディレクトリか否かを格納するバッファ（不要時は NULL）

説　明 ： 「pPath」で指定したディレクトリの親ディレクトリのパスを取得します。
pfRoot で示すバッファには、親ディレクトリがルートディレクトリか否かを格納します。
*pfRoot==TRUE ならば、親ディレクトリがルートディレクトリであることを意味します。

戻り値 ： TRUE　- 成功
FALSE – 失敗

30.1.7. ディレクトリ作成(AjcCreateDirectory[Ex])

形　式 ： BOOL AjcCreateDirectory(C_BCP lpNewDirectory, BOOL fSaveParentDirTime)
BOOL AjcCreateDirectoryEx(C_BCP lpTemplateDirectory, C_BCP lpNewDirectory,
LPSECURITY_ATTRIBUTES lpSecurityAttributes, BOOL fSaveParentDirTime)

引　数 ： lpTemplateDirectory　- テンプレートディレクトリパス（不要時は NULL）
lpNewDirectory　　- 作成するディレクトリのパス
lpSecurityAttributes - SECURITY_ATTRIBUTES 構造体へのポインター（不要時は NULL）
fSaveParentDirTime　- 親ディレクトリのタイムスタンプを維持する(TRUE)か、維持しない(FALSE)かのフラグ

説　明 ： lpNewDirectory で指定したディレクトリを作成します
lpNewDirectory＝NULL の場合は、CreateDirectory()でディレクトリを作成します。
lpNewDirectory≠NULL の場合は、CreateDirectoryEx()でディレクトリを作成します
fSaveParentDirTime＝TRUE を指定した場合は、親ディレクトリのタイムスタンプが変更されないようにします。
fSaveParentDirTime＝FALSE を指定した場合は、親ディレクトリのタイムスタンプ現在時刻に変更されます。

戻り値 ： TRUE　- 成功
FALSE – 失敗

30.1.8. 多階層ディレクトリ構造の一括作成(AjcCreateDirectoryStruct[Ex])

形　式 ： BOOL AjcCreateDirectoryStruct　(C_UTP pPath);
BOOL AjcCreateDirectoryStructEx(C_UTP pPath, LPSECURITY_ATTRIBUTES lpSecurityAttributes);

引　数 ： pPath　　- ディレクトリ構造を表すパス（ex. “d:¥work¥sub¥child”）
lpSecurityAttributes - SECURITY_ATTRIBUTES 構造体へのポインター

説　明 ： 「pPath」で指定したディレクトリパスのディレクトリ構造を一括作成します。
ディレクトリは、CreateDirectory()で作成します。
lpSecurityAttributes は、CreateDirectory()の第２引数となります。
AjcCreateDirectoryStruct()の場合は、lpSecurityAttributes=NULL を仮定します。

戻り値 ： TRUE　- 成功
FALSE – 失敗

30.1.9. パスがワイルドカード[群]の指定に一致するかチェックする(AjcPathMatchSpec)

形　式 ： BOOL AjcPathMatchSpec(C_UTP pPath, C_UTP pWild);

引　数 ： pPath　- チェックするパス名
pWild　- ワイルドカード（複数指定時はセミコロン(;)か スラッシュ(/)で区切る）

説　明 ： 「pPath」で指定したパスが「pWild」で指定したワイルドカードに一致するかチェックします。
ワイルドカードを複数指定する場合は、セミコロン(;)か スラッシュ(/)で区切って指定します。
（ex. ” *.txt ； *.bmp ”）
各ワイルドカードの前後の空白は無視します。
ワイルドカードにファイル名そのものを指定可能です。(ex. pPath=”d:¥¥work¥¥abc.txt”, pWild=”abc.txt” → 一致)
pPath=NULL / pWild=NULL / pWild = 空文字列の場合は、FALSE を返します。

戻り値 ： TRUE - 一致
FALSE - 不一致

30.1.10. パスが指定文字列[群]を含むかチェックする(AjcPathMatchStr)

形　式 ： BOOL AjcPathMatchStr (C_UTP pPath, C_UTP pStr);

引　数 ： pPath　- チェックするパス名
pStr　- チェックする文字列（複数指定時はセミコロン(;)か スラッシュ(/)で区切る）

説　明 ： 「pPath」で指定したパスが「pStr」で指定した文字列を含むかチェックします。
「pPath」で指定したパスがディレクトリである場合は、末尾に ‘¥’ を補完します。
「pStr」でチェックする文字列を複数指定する場合は、セミコロン(;)か スラッシュ(/)で区切って指定します。
文字列を複数指定した場合、いずれかの文字列が一致すれば TRUE を返します。
文字列中のアスタリスク(*)は、任意の０桁以上の文字列を意味します。
pStr=空文字列（すべて空白の場合を含む）である場合は、TRUE を返します。
pPath=NULL / pStr=NULL の場合は、FALSE を返します。

セミコロン(;)、スラッシュ(/)やアスタリスク(*)で区切られた比較文字列（下記・下線部）の前後の空白は除去されます。
ex. “ c:¥Program Files¥ * MyProg¥ ; ¥Debug¥ ” -> “c:¥Program Files¥*MyProg¥;¥Debug¥”
（“c:¥Program Files¥” の後に ”MyProg¥” があるか、あるいは、”¥Debug¥” がある）

戻り値 ： TRUE - 一致
FALSE - 不一致

30.1.11. パスの末尾が「¥」(or「/」)かチェックする(AjcIsPathBackslash[Ex])

形　式 ： BOOL AjcIsPathBackslash (C_UTP pPath);
BOOL AjcIsPathBackslashEx(C_UTP pPath);

引　数 ： pPath　- チェックするパス名

説　明 ： AjcIsPathBackSlash()は、「pPath」で指定したパス名の末尾が「¥」かチェックします。
AjcIsPathBackSlashEx()は、「pPath」で指定したパス名の末尾が「¥」or「/」かチェックします。

戻り値 ： TRUE - パス名の末尾が「¥」（ or「/」 ）である
FALSE - パス名の末尾は「¥」（ or「/」 ）以外

30.1.12. ２つのパスの間に必ず１つの「￥」を挿入しパスを結合する (AjcPathCat)

形　式 ： BOOL　AjcPathCat (UTP pTop, C_UTP pTail, int lBuf);

引　数 ： pTop　- 先頭部分の文字列のアドレス
pTail - 連結する文字列のアドレス
lBuf　- 連結される文字列バッファの文字数（pTop で指定される文字列バッファの最大文字数）

説　明 ： pTop で指定された文字列の後ろに、pTail で指定された文字列を連結します。
この時、２つの文字列の間に、必ず１ヶの「¥」が含まれるようにします。
つまり、pTop で示される文字列の末尾と pTail で示される文字列の先頭文字がともに「¥」でない場合は、２つの文字列の間に「¥」を挿入し、いずれかが「¥」である場合は、単純に文字列を連結します。
pTop で示される文字列の末尾と pTail で示される文字列の先頭文字がともに「¥」である場合は、pTail の先頭文字である「¥」を除去します。

```
ex.   「ABC」  +「XYZ」 --->「ABC¥XYZ」
      「ABC¥」 +「XYZ」 --->「ABC¥XYZ」
      「ABC」  +「¥XYZ」--->「ABC¥XYZ」
      「ABC¥」 +「¥XYZ」--->「ABC¥XYZ」
```

戻り値 ： TRUE　- 成功
FALSE - 失敗

30.1.13. パス末尾の「￥」を除去して比較（AjcPathCmp）

形　式 ： int　AjcPathCmp(C_UTP pPath1, C_UTP pPath2);

引　数 ： pPath1, pPath2　- 比較する２つの文字列

説　明 ： ２つの文字列の末尾の「¥」を除去し、英字の大文字／小文字を区別しないで比較します。

戻り値 ： ＝０ - 等しい
＞０ - path1 ＜ path2
＜０ - path1 ＞ path2

30.1.14. ファイル名に使用できない文字を特定の文字に変換する（AjcChangeFNameToCorrect）

形　式 ： int　AjcChangeFNameToCorrect (UTP pPath, UT c);

引　数 ： pPath　- 変換するファイル名文字列
c　　- 変換する文字「

説　明 ： pFName で指定されたファイル名中のファイル名として使用できない文字を c で指定された文字に変換する。

戻り値 ： ≧０ - 変換した文字数
-1　- エラー

30.1.15. パスが存在するかチェック(AjcPathExists)

形　式 ： BOOL　AjcPathExists (C_UTP pPath);

引　数 ： pPath　　- チェックするパス名

説　明 ： 「pPath」で指定したパスが存在するかチェックします。

戻り値 ： TRUE　- パスは存在する
FALSE - パスは存在しない

30.1.16. パスがディレクトリかチェック(AjcPathIsDirectory)

形　式　：　BOOL　AjcPathIsDirectory (C_UTP pPath);

引　数　：　pPath　　- チェックするパス名

説　明　：　「pPath」で指定したパスがディレクトリとして存在するかチェックします。

戻り値　：　TRUE　- パスはディレクトリである
FALSE - パスはディレクトリ以外

30.1.17. パスが空のディレクトリかチェック(AjcPathIsEmptyDirectory)

形　式　：　BOOL AjcPathIsEmptyDirectory(C_UTP pPath, UX cbp, BOOL (CALLBACK *cbQuery)(C_UTP pPath, UI att, BOOL *pfIgnore, UX cbp));

引　数　：　pPath　　- チェックするディレクトリパス名
cbp　　- コールバックパラメタ
cbQuery　- コールバック関数（不要時は NULL）

説　明　：　「pPath」で指定したパスがディレクトリとして存在し、内容が空（サブディレクトリ群を含めてファイルが１つも無い）かチェックします。
pPath で指定したディレクトリとそのサブディレクトリ群を検索し、ファイルが１つも存在しない場合 TRUE を返します。

コールバック関数により、特定のファイルやディレクトリの存在を無視する（特定のファイルやディレクトリが存在しても空ディレクトリとする）ことができます。

戻り値　：　TRUE　- パスは空のディレクトリである
FALSE - パスは空のディレクトリ以外

コールバック関数　：

cbQuery（検出したファイルの通知）

形　式　：　BOOL CALLBACK *cbQuery*(C_WCP pPath, UI att, BOOL *pfIgnore, UX cbp);

引　数　：　pPath　　- ファイル／ディレクトリパス名
att　　- ファイル／ディレクトリ属性(FILT_ATTRIBUTE_*XXXXX*)
pfIgnore　- 当該ファイル／ディレクトリの無視フラグバッファポインタ（デフォルト＝FALSE）
cbp　　- コールバックパラメタ

説　明　：　検出したファイル／ディレクトリのパス名を順次通知します。
*pfIgnore = TRUE を設定すると、ファイルの場合は、当該ファイルの存在を無視します。
ディレクトリの場合は、当該ディレクトリのサブディレクトリの検索を行いません。

戻り値　：　TRUE　：検索を続行する
FALSE ：検索を中止する

30.1.18. パスがファイルかチェック(AjcPathIsFile)

形　式 ： BOOL　AjcPathIsFile (C_UTP pPath);

引　数 ： pPath　　- チェックするパス名

説　明 ： 「pPath」で指定したパスがファイルとして存在するかチェックします。

戻り値 ： TRUE　- パスはファイルである
FALSE - パスはファイル以外（あるいは、パスが存在しない）

30.1.19. ファイルサイズ取得(AjcGetFileSize)

形　式 ： ULL　AjcGetFileSize(C_UTP pPath);

引　数 ： pPath　　- サイズを取得するファイルのパス名

説　明 ： 「pPath」で指定したファイルのサイズ（バイト数）を取得します。

戻り値 ： ≠-1 - ファイルサイズ
＝-1 - エラー

30.1.20. ファイルタイム取得(AjcGetFileTime1970)

形　式 ： UI　AjcGetFileTime1970 (C_UTP pPath);

引　数 ： pPath　　- タイムスタンプを取得するファイルのパス名

説　明 ： 「pPath」で指定したファイルの最終更新タイム(UTC 時刻)を 1970/1/1 00:00:00 からの通算秒数で取得します。

戻り値 ： ≠-1 - ファイルタイム（1970/01/01 00:00:00 からの通算秒数，　最大 0xFFFFFFFE=2106/02/07 06:28:14）
＝-1 - エラー

30.1.21. ファイル比較(AjcFileCompare)

形　式 ： BOOL AjcFileCompare (C_UTP FilePath1, C_UTP FilePath2);

引　数 ： FilePath1, FilePath2　- 比較する２つのファイルパス名へのポインタ

説　明 ： ２つのファイルをバイナリ比較します。

戻り値 ： TRUE　- ２つのファイルは一致する
FALSE – ２つのファイルは異なる／エラー

30.1.22. ファイル／ディレクトリ日時取得(AjcGetFileTime)

形　式 ： BOOL　AjcGetFileTime (C_UTP pPath, PAJCFTIMES pFTimes, PAJCSTIMES pSTimes, BOOL fLocalTime);

引　数 ： pPath　　- 時刻を取得するファイル／ディレクトリのパス名（ディレクトリの場合でも末尾に「¥」を付加しないこと）
pFTimes　- ファイル／ディレクトリ時刻(1601 年からの 100ns 単位カウンタ)を格納するバッファ（不要時は NULL）
pSTimes　- ファイル／ディレクトリ日時を格納する SYSTEMTIME 構造体バッファ　　（不要時は NULL）
fLocalTime - FALSE ： ＵＴＣ時刻を取得
TRUE　： ローカルタイムを取得

説　明 ： 「pPath」で指定したファイルの「作成日時」「アクセス日時」「更新日時」を取得します。
pFTimes で示されるバッファには、1601 年からの 100ns 単位カウンタで示すファイル／ディレクトリ時刻が設定されます。

```
typedef union {
    FILETIME    ft;
    ULL         v;
} AJCUFT, *PAJCUFT;
typedef const AJCUFT *PCAJCUFT;

typedef struct {
    AJCUFT  mk;             //  作成日時
    AJCUFT  ac;             //  アクセス日時
    AJCUFT  up;             //  更新日時
} AJCFTIMES, *PAJCFTIMES;
typedef const AJCFTIMES *PCAJCFTIMES;
```

pSTimes で示されるバッファには、以下の構造体の形式で、ファイル／ディレクトリ日時が設定されます。

```
typedef struct {
    SYSTEMTIME  mk;         //  作成日時
    SYSTEMTIME  ac;         //  アクセス日時
    SYSTEMTIME  up;         //  更新日時
} AJCSTIMES, *PAJCSTIMES;
typedef const AJCSTIMES *PCAJCSTIMES;
```

戻り値 ： TRUE　- 成功
FALSE – 失敗

30.1.23. ファイル／ディレクトリ日時設定(AjcSetFileTime / AjcSetFileTimeBySysTime)

形　式 ： BOOL　AjcSetFileTime　　(C_UTP pPath, PCAJCFTIMES　pFTimes, BOOL fLocalTime);
BOOL　AjcSetFileTimeBySysTime(C_UTP pPath, LPSYSTEMTIME pSTimes, BOOL fLocalTime);

引　数 ： pPath　　- 時刻を設定するファイル／ディレクトリのパス名（ディレクトリの場合でも末尾に「¥」を付加しないこと）
pFTimes　- 設定するファイル／ディレクトリ時刻(1601 年からの 100ns 単位カウンタ)
pSTimes　- 設定するファイル／ディレクトリ時刻(SYSTEMTIME 構造体)
fLocalTime - FALSE ： ＵＴＣ時刻を指定
TRUE　： ローカルタイムを指定

説　明 ： 「pPath」で指定したファイル／ディレクトリの「作成日時」「アクセス日時」「更新日時」を設定します。
AjcSetFileTime()は、1601/01/01 00:00:00 からの 100ns 単位の経過カウンタで指定されたファイル時刻を設定します。

AjcSetFileTimeBySysTime()は、SYSTEMTIME 構造体で指定した日時を設定しますが、0xFFFF を指定した項目は元ファイル／ディレクトリの日時のままとなり、変更されません。
例えば、以下のコードはファイル／ディレクトリ日時の「年」だけを 2019 年に変更します。

```
SYSTEMTIME  st;

memset(&st, 0xFF, sizeof st);
st.wYear = 2019;
AjcSetFileTimeBySysTime("D:¥¥work¥¥b¥¥Debug", &st, TRUE);
```

設定する日時の矛盾は訂正されます。
(閏年以外の 2 月 29 日は 28 日に訂正，更新日時が作成日時より古い場合は更新日時を作成日時と同じに訂正・・等)

pFTimes, pSTimes で指定するデータの形式については、AjcGetFileTime()を参照してください。

戻り値 ： TRUE　- 成功
FALSE – 失敗

30.1.24. ２つのパスが同一パスかチェックする

形　式 ： C_UTP　AjcIsSamePath(C_WCP pBase, C_UTP pPath1, C_UTP pPath2);

引　数 ： pBase　　- ベースパスを指定する（NULL の場合カレントディレクトリを仮定）
pPath1, pPath2　- 比較する２つのパス

説　明 ： pPath1 と pPath2 で指定した２つのパスが同一パスかチェックします。
pPath1 と pPath2 は任意の相対パスか絶対パスを指定できます。

２つのパスが一致する場合、相対パスを指定した場合は、pBase で指定したパスと連結したパス名を返します。
フォルダパスの場合、パス末尾に「¥」を付加するか否かは、pPath1 の指定に合わせます。

戻り値 ： ≠NULL - 一致（実際のパス名へのポインタ）
＝NULL - 不一致

30.1.25. ２つのパスで、片方がサブパスであるかをチェックする

形　式 ： C_UTP　AjcIsUnderPath(C_WCP pBase, C_UTP pUpperPath, pUnderPath);

引　数 ： pBase　　- ベースパスを指定する（NUUL ／ 絶対パス）
pUpperPath　- 上位パス名へのポインタ
pUnderPath　- 下位パス名へのポインタ

説　明 ： 下位パスが、上位パスのサブパスかチェックします。
下位パスが、上位パスと同じか、上位パス下のファイルやサブフォルダである場合は TRUE を返します。
下位パスと上位パスが同じである場合も TRUE を返します。
それ以外は、FALSE を返します。

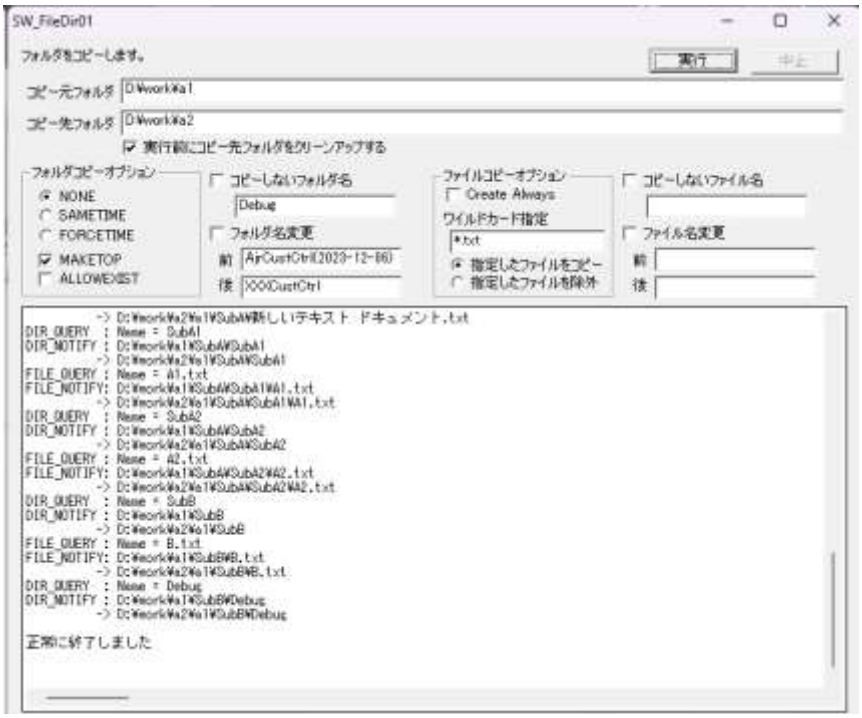
ベースパスは、上位パスと下位パスの先行部分を指定しますが、NULL を指定した場合は、上位パス／下位パスだけで判断します。
上位パス／下位パス自身が絶対パスである場合、ベースパスは無視されます。

戻り値 ： TRUE　- 下位パスは上位パスのサブパスである（同一パスを含む）
FALSE - 下位パスは上位パスのサブフパスでない
-1　　- エラー

30.2. サンプルプログラム

30.2.1. SW_FileDir01（フォルダコピー）

以下のサンプルプログラムは、フォルダ下のファイルやサブフォルダを全てコピーします。
コピーしないフォルダやファイルを指定したり、フォルダやファイル名称の変更等ができます。




```
 1 : //
 2 : //  SW_FileDir01.c
 3 : //
 4 :
 5 : #include    <AjrCstXX.h>
 6 : #include    <math.h>
 7 : #include    <tchar.h>
 8 : #include    "resource.h"
 9 :
10 : //-------------------------------------------------------------------------------------------------------------//
11 : //  ワーク                                                                                                     //
12 : //-------------------------------------------------------------------------------------------------------------//
13 : static  HINSTANCE       hInst;                          //  ＤＬＬインスタンスハンドル
14 : static  HWND            hDlgMain;                       //  ダイアログボックスハンドル
15 : static  HWND            hVth;
16 : static  BOOL    fBusy = FALSE;
17 : static  BOOL    fCan  = FALSE;
18 : static  BOOL    fEnd  = FALSE;
19 : //-------------------------------------------------------------------------------------------------------------//
20 : //  内部サブ関数                                                                                               //
21 : //-------------------------------------------------------------------------------------------------------------//
22 : AJC_DLGPROC_DEF(Main);
23 :
24 : static  UTP     CALLBACK cbDirQuery    (C_UTP pPathFront, UTP pNewDir, UX cbp);
25 : static  BOOL    CALLBACK cbDirNotify   (C_UTP pPathFrom, C_UTP pPathTo, EAJCCFS ntc, UX cbp);
26 : static  UTP     CALLBACK cbFileQuery   (C_UTP pFileFrom, C_UTP pFileTo, UTP pFileName, UIP pAtt, UX cbp);
27 : static  BOOL    CALLBACK cbFileNotify  (C_UTP pFileFrom, C_UTP pFileTo , EAJCCFS ntc, UX cbp);
28 : static  BOOL    CALLBACK cbCopyProgress(ULL FileSize, ULL Copied, UX cbp);
29 :
30 : //=============================================================================================================//
31 : //                                                                                                             //
32 : //  ＷｉｎＭａｉｎ                                                                                            //
33 : //                                                                                                             //
34 : //=============================================================================================================//
35 : int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, PSTR szCmdLine, int iCmdShow)
36 : {
37 :     MSG     msg;
38 :
```

```
    hInst = hInstance;

    //----- メイン・ダイアログオープン -----------------//
    hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_DLGMAIN), NULL, AJC_DLGPROC_NAME(Main));
    //----- ダイアログ表示 -----------------------------//
    ShowWindow(hDlgMain, SW_SHOW);

    //----- メッセージループ ---------------------------//
    while (GetMessage(&msg, NULL, 0, 0)) {
        do {
            if (IsDialogMessage(hDlgMain, &msg)) break;
            TranslateMessage(&msg);
            DispatchMessage (&msg);
        } while (0);
    }

    return (int)msg.wParam ;
}
//=============================================================================================================//
//                                                                                                             //
//  コールバック                                                                                               //
//                                                                                                             //
//=============================================================================================================//
//  フォルダ問い合わせ
static  UTP CALLBACK cbDirQuery(C_UTP pPathFront, UTP pNewDir, UX cbp)
{
    UTP     rc = pNewDir;
    HWND    hDlg = (HWND)cbp;
    UT      path[MAX_PATH];

    AjcDoEvent();
    if (!(fCan || fEnd)) {
        AjcVthPrintF(hVth, TEXT("DIR_QUERY  : Name = %s"), pNewDir);
        //  コピーしないフォルダ名
        if      (AjcGetDlgItemChk(hDlg, IDC_CHK_NOCPYDIR)) {
            UT  ncp[MAX_PATH];
            AjcGetDlgItemStr(hDlg, IDC_TXT_NOCPYDIR, ncp, MAX_PATH);
            if (MAjcStrICmp(pNewDir, ncp) == 0) {
                AjcVthPrintF(hVth, TEXT(" (コピーしないファイル) "), pPathFront, pNewDir);
                rc = NULL;
            }
        }
        //  リネーム
        if (AjcGetDlgItemChk(hDlg, IDC_CHK_CHGDIR)) {
            UT  bfr[MAX_PATH];
            UT  aft[MAX_PATH];
            AjcGetDlgItemStr(hDlg, IDC_TXT_DIRBFR, bfr, MAX_PATH);
            AjcGetDlgItemStr(hDlg, IDC_TXT_DIRAFT, aft, MAX_PATH);
            if (MAjcStrICmp(pNewDir, bfr) == 0) {
                AjcVthPrintF(hVth, TEXT("    ReName to %s¥n"), aft);
                MAjcStrCpy(pNewDir, MAX_PATH, aft);
            }
        }
        //  同一フォルダありを表示
        MAjcStrCpy(path, MAX_PATH, pPathFront);
        AjcPathCat(path, pNewDir, MAX_PATH);
        if (AjcPathExists(path)) {
            AjcVthPrintF(hVth, TEXT(" (既に同一パスが存在します。) "));
        }
        AjcVthPrintF(hVth, TEXT("¥n"));
    }
    return rc;
}
//  フォルダ作成通知
static  BOOL CALLBACK cbDirNotify(C_UTP pPathFrom, C_UTP pPathTo, EAJCCFS ntc, UX cbp)
{
    BOOL    rc   = TRUE;
    HWND    hDlg = (HWND)cbp;
    UI      opt  = AJCCPF_NONE;
    UT      wild[MAX_PATH];

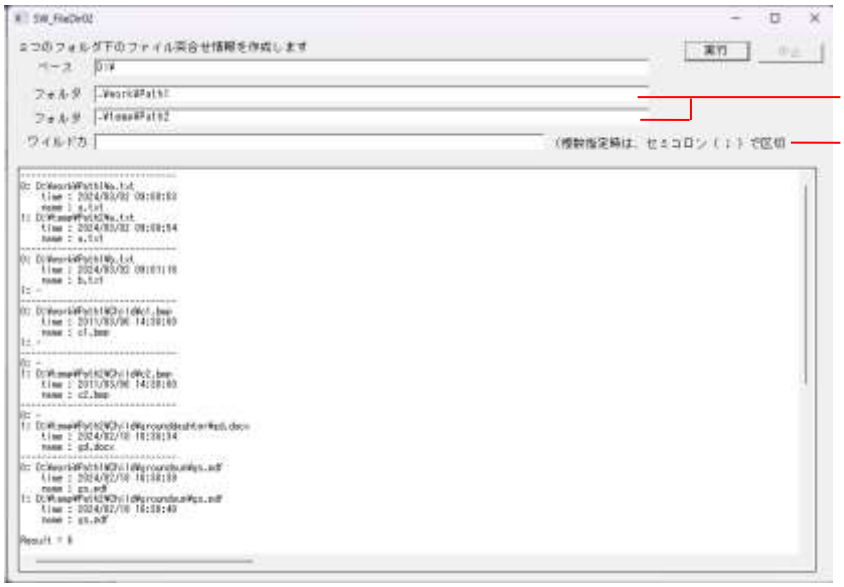
    AjcDoEvent();
    if (!(fCan || fEnd)) {
        AjcVthPrintF(hVth, TEXT("DIR_NOTIFY : %s¥n")
                           TEXT("          -> %s¥n"),  pPathFrom, pPathTo);
        if (ntc == AJCCFS_SUCCESS || ntc == AJCCFS_EXIST) {
            UTP     pWild = NULL;
            //  フォルダ内ファイルコピー
            if (AjcGetDlgItemChk(hDlg, IDC_CHK_CREATEALWAYS)) opt |= AJCCPF_CREATEALWAYS;
            if (AjcGetDlgItemChk(hDlg, IDC_RBT_WILDEXC     )) opt |= AJCCPF_WILDEXC;
```

```
119 :             AjcGetDlgItemStr(hDlg, IDC_TXT_WILD , wild, MAX_PATH);
120 :             if (wild[0] != 0) pWild = wild;
121 :             if (!AjcCopyFilesEx(pPathFrom, pPathTo, pWild, opt, cbp, cbFileNotify, cbFileQuery, cbCopyProgress)) {
122 :                 rc = FALSE;
123 :             }
124 :         }
125 :     }
126 :     else rc = FALSE;
127 :
128 :     return rc;
129 : }
130 : //  ファイル問い合わせ
131 : static  UTP CALLBACK cbFileQuery(C_UTP pFileFrom, C_UTP pFileTo, UTP pFileName, UIP pAtt, UX cbp)
132 : {
133 :     UTP     rc = pFileName;
134 :     HWND    hDlg = (HWND)cbp;
135 :
136 :     AjcDoEvent();
137 :     if (!(fCan || fEnd)) {
138 :         AjcVthPrintF(hVth, TEXT("FILE_QUERY : Name = %s¥n"), pFileName);
139 :         //  コピーしないフォルダ名
140 :         if      (AjcGetDlgItemChk(hDlg, IDC_CHK_NOCPYFILE)) {
141 :             UT  ncp[MAX_PATH];
142 :             AjcGetDlgItemStr(hDlg, IDC_TXT_NOCPYFILE, ncp, MAX_PATH);
143 :             if (MAjcStrICmp(pFileName, ncp) == 0) {
144 :                 AjcVthPrintF(hVth, TEXT("    Not Copy¥n"));
145 :                 rc = NULL;
146 :             }
147 :         }
148 :         //  リネーム
149 :         if (AjcGetDlgItemChk(hDlg, IDC_CHK_CHGFILE)) {
150 :             UT  bfr[MAX_PATH];
151 :             UT  aft[MAX_PATH];
152 :             AjcGetDlgItemStr(hDlg, IDC_TXT_FILEBFR, bfr, MAX_PATH);
153 :             AjcGetDlgItemStr(hDlg, IDC_TXT_FILEAFT, aft, MAX_PATH);
154 :             if (MAjcStrICmp(pFileName, bfr) == 0) {
155 :                 AjcVthPrintF(hVth, TEXT("    ReName to %s¥n"), aft);
156 :                 MAjcStrCpy(pFileName, MAX_PATH, aft);
157 :             }
158 :         }
159 :     }
160 :     return rc;
161 : }
162 : //  ファイル作成通知
163 : static  BOOL CALLBACK cbFileNotify(C_UTP pFileFrom, C_UTP pFileTo, EAJCCFS ntc, UX cbp)
164 : {
165 :     BOOL    rc   = TRUE;
166 :     HWND    hDlg = (HWND)cbp;
167 :     AjcDoEvent();
168 :     if (!(fCan || fEnd)) {
169 :         AjcVthPrintF(hVth, TEXT("FILE_NOTIFY: %s¥n")
170 :                            TEXT("          -> %s¥n"), pFileFrom, pFileTo);
171 :     }
172 :     else rc = FALSE;
173 :
174 :     return rc;
175 : }
176 : //  ファイルコピー状況通知
177 : static  BOOL CALLBACK cbCopyProgress(ULL FileSize, ULL Copied, UX cbp)
178 : {
179 :     BOOL    rc   = TRUE;
180 :     HWND    hDlg = (HWND)cbp;
181 :     AjcDoEvent();
182 :     if (!(fCan || fEnd)) {
183 :     }
184 :     else rc = FALSE;
185 :
186 :     return rc;
187 : }
188 : //=============================================================================================================//
189 : //                                                                                                             //
190 : //  ダイアログ・プロシージャ                                                                                   //
191 : //                                                                                                             //
192 : //=============================================================================================================//
193 : //----- ダイアログ初期化 ------------------------------------------------------------------------------------//
194 : AJC_DLGPROC(Main, WM_INITDIALOG     )
195 : {
196 :     hDlgMain = hDlg;
197 :     hVth     = GetDlgItem(hDlg, IDC_VTH);
198 :
```

```
199 :     hVth     = GetDlgItem(hDlg, IDC_VTH);
200 :     AjcEnableDlgItemToDrop(hDlg, IDC_TXT_PATH1, AJCDROP_DIR);
201 :     AjcEnableDlgItemToDrop(hDlg, IDC_TXT_PATH2, AJCDROP_DIR);
202 :     AjcSetDlgItemChk(hDlg, IDC_RBT_SAMETIME, TRUE);
203 :     AjcSetDlgItemChk(hDlg, IDC_RBT_WILDINC , TRUE);
204 :     AjcLoadAllControlSettings(hDlg, TEXT("Main"), AJCCTL_SELACT_ALL);
205 :
206 :     return TRUE;
207 : }
208 : //----- ウインド破棄 ----------------------------------------------------------------------------------------//
209 : AJC_DLGPROC(Main, WM_DESTROY        )
210 : {
211 :     AjcSaveAllControlSettings(hDlg);
212 :     PostQuitMessage(0);
213 :     return TRUE;
214 : }
215 : //----- 「実行」ボタン --------------------------------------------------------------------------------------//
216 : AJC_DLGPROC(Main, IDC_CMD_RUN       )
217 : {
218 :     int     rsu;
219 :     UI      opt = 0;
220 :     UT      src [MAX_PATH];
221 :     UT      dst [MAX_PATH];
222 :
223 :     AjcEnableDlgItem(hDlg, IDC_CMD_RUN , FALSE);
224 :     AjcEnableDlgItem(hDlg, IDC_CMD_STOP, TRUE);
225 :     fCan = fEnd = FALSE;
226 :     fBusy = TRUE;
227 :     //  コピーするパス名設定
228 :     AjcGetDlgItemStr(hDlg, IDC_TXT_PATH1, src , MAX_PATH);
229 :     AjcGetDlgItemStr(hDlg, IDC_TXT_PATH2, dst , MAX_PATH);
230 :     //  コピー先クリーンアップ
231 :     if (AjcGetDlgItemChk(hDlg, IDC_CHK_CLEAN)) {
232 :         AjcCleanFolder(dst, 0, NULL);
233 :     }
234 :     //  オプション設定
235 :     if (AjcGetDlgItemChk(hDlg, IDC_RBT_SAMETIME  )) opt |= AJCFSC_SAMETIME;
236 :     if (AjcGetDlgItemChk(hDlg, IDC_RBT_FORCETIME )) opt |= AJCFSC_FORCETIME;
237 :     if (AjcGetDlgItemChk(hDlg, IDC_CHK_MAKETOP   )) opt |= AJCFSC_MAKETOP;
238 :     if (AjcGetDlgItemChk(hDlg, IDC_CHK_ALLOWEXIST)) opt |= AJCFSC_ALLOWEXIST;
239 :     //  フォルダ構造コピー
240 :     rsu = AjcCopyFolderStructEx(src, dst, opt, (UX)hDlg, cbDirNotify, cbDirQuery);
241 :     //  結果表示
242 :     if (rsu) AjcVthPrintF(hVth, TEXT("¥n 正常に終了しました¥n¥n"));
243 :     else     AjcVthPrintF(hVth, TEXT("¥n エラー／中止しました¥n¥n"));
244 :     AjcEnableDlgItem(hDlg, IDC_CMD_RUN , TRUE);
245 :     AjcEnableDlgItem(hDlg, IDC_CMD_STOP, FALSE);
246 :     fBusy = FALSE;
247 :     if (fEnd) DestroyWindow(hDlg);
248 :     return TRUE;
249 : }
250 : //----- 「中止」ボタン --------------------------------------------------------------------------------------//
251 : AJC_DLGPROC(Main, IDC_CMD_STOP      )
252 : {
253 :     if (fBusy) fCan = TRUE;
254 :     return TRUE;
255 : }
256 : //----- 「Cancel」 ------------------------------------------------------------------------------------------//
257 : AJC_DLGPROC(Main, IDCANCEL          )
258 : {
259 :     if (fBusy) fEnd = TRUE;
260 :     else DestroyWindow(hDlg);
261 :     return TRUE;
262 : }
263 : //-----------------------------------------------------------------------------------------------------------//
264 : AJC_DLGMAP_DEF(Main)
265 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG     )
266 :     AJC_DLGMAP_MSG(Main, WM_DESTROY        )
267 :
268 :     AJC_DLGMAP_CMD(Main, IDC_CMD_RUN       )
269 :     AJC_DLGMAP_CMD(Main, IDC_CMD_STOP      )
270 :     AJC_DLGMAP_CMD(Main, IDCANCEL          )
271 : AJC_DLGMAP_END
272 :
```

30.2.2. SW_FileDir02（ファイル突き合せリストの作成）

以下のサンプルプログラムは、２つのフォルダ下のファイル突き合せリストを作成し、列挙します。




```
 1 : //
 2 : //  SW_FileDir02.c
 3 : //
 4 :
 5 : #include    <AjrCstXX.h>
 6 : #include    <math.h>
 7 : #include    <tchar.h>
 8 : #include    "resource.h"
 9 :
10 : //-------------------------------------------------------------------------------------------------------------//
11 : //  ワーク                                                                                                     //
12 : //-------------------------------------------------------------------------------------------------------------//
13 : static  HINSTANCE       hInst;                      //  ＤＬＬインスタンスハンドル
14 : static  HWND            hDlgMain;                   //  ダイアログボックスハンドル
15 : static  HWND            hVth;
16 : //-------------------------------------------------------------------------------------------------------------//
17 : //  内部サブ関数                                                                                               //
18 : //-------------------------------------------------------------------------------------------------------------//
19 : AJC_DLGPROC_DEF(Main);
20 : static  BOOL    fBusy = FALSE;
21 : static  BOOL    fCan  = FALSE;
22 : static  BOOL    fEnd  = FALSE;
23 :
24 : //=============================================================================================================//
25 : //                                                                                                             //
26 : //  Ｗ ｉ ｎ Ｍ ａ ｉ ｎ                                                                                       //
27 : //                                                                                                             //
28 : //=============================================================================================================//
29 : int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, PSTR szCmdLine, int iCmdShow)
30 : {
31 :     MSG     msg;
32 :
33 :     hInst = hInstance;
34 :
35 :     //----- メイン・ダイアログオープン ------------------//
36 :     hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_DLGMAIN), NULL, AJC_DLGPROC_NAME(Main));
37 :     //----- ダイアログ表示 ------------------------------//
38 :     ShowWindow(hDlgMain, SW_SHOW);
39 :
40 :     //----- メッセージループ ----------------------------//
41 :     while (GetMessage(&msg, NULL, 0, 0)) {
42 :         do {
43 :             if (IsDialogMessage(hDlgMain, &msg)) break;
44 :             TranslateMessage(&msg);
```

```
45 :             DispatchMessage (&msg);
46 :         } while (0);
47 :     }
48 :
49 :     return (int)msg.wParam ;
50 : }
51 : //=============================================================================================================//
52 : //                                                                                                             //
53 : //  ダイアログ・プロシージャ                                                                                   //
54 : //                                                                                                             //
55 : //=============================================================================================================//
56 : static  BOOL CALLBACK cbQuery(UI id, C_UTP pPath, UI att, BOOL *pfDiscard, UX cbp)
57 : {
58 :     BOOL    rc = TRUE;
59 :     HWND    hDlg = (HWND)cbp;
60 :     UT      wild[MAX_PATH];
61 :
62 :     AjcDoEvent();
63 :
64 :     if (!(fCan || fEnd)) {
65 :         //  ワイルドカードチェック
66 :         AjcGetDlgItemStr(hDlg, IDC_TXT_WILD, wild, MAX_PATH);
67 :         if (wild[0] != 0) {
68 :             if (!AjcPathMatchSpec(pPath, wild)) {
69 :                 *pfDiscard = TRUE;
70 :             }
71 :         }
72 :     }
73 :     else {
74 :         rc = FALSE;
75 :     }
76 :     return rc;
77 : }
78 : //-------------------------------------------------------------------------------------------------------------//
79 : static  BOOL CALLBACK cbEnum(AJCFMLITEM item[2], UX cbp)
80 : {
81 :     BOOL        rc = TRUE;
82 :     UI          ix;
83 :     SYSTEMTIME  st, lt;
84 :
85 :     AjcDoEvent();
86 :
87 :     if (!(fCan || fEnd)) {
88 :         AjcVthPrintF(hVth, TEXT("----------------------------¥n"));
89 :         for (ix = 0; ix < 2; ix++) {
90 :             if (item[ix].fValid) {
91 :                 AjcVthPrintF(hVth, TEXT("%d: %s¥n"), ix, item[ix].path);
92 :                 AjcTime1970ToSysTime(item[ix].hd.utc, &st);
93 :                 AjcSysTimeToLocalTime(&st, &lt);
94 :                 AjcVthPrintF(hVth, TEXT("    time : %04d/%02d/%02d %02d:%02d:%02d¥n"), lt.wYear, lt.wMonth, lt.wDay,
95 :                                                                                lt.wHour, lt.wMinute, lt.wSecond);
96 :                 AjcVthPrintF(hVth, TEXT("    name : %s¥n"), item[ix].name);
97 :             }
98 :             else {
99 :                 AjcVthPrintF(hVth, TEXT("%d: -¥n"), ix);
100 :            }
101 :        }
102 :    }
103 :    else {
104 :        rc = FALSE;
105 :    }
106 :    return rc;
107 : }
108 : //----- ダイアログ初期化 ----------------------------------------------------------------------------------//
109 : AJC_DLGPROC(Main, WM_INITDIALOG     )
110 : {
111 :     UT  base[MAX_PATH];
112 :
113 :     hDlgMain = hDlg;
114 :     hVth     = GetDlgItem(hDlg, IDC_VTH);
115 :     AjcEnableDlgItemToDrop(hDlg, IDC_TXT_BASE , AJCDROP_DIR);
116 :     AjcEnableDlgItemToDrop(hDlg, IDC_TXT_PATH1, AJCDROP_DIR);
```

```
117 :     AjcEnableDlgItemToDrop(hDlg, IDC_TXT_PATH2, AJCDROP_DIR);
118 :     AjcLoadAllControlSettings(hDlg, TEXT("Main"), AJCCTL_SELACT_ALL);
119 :
120 :     AjcGetDlgItemStr(hDlg, IDC_TXT_BASE, base, MAX_PATH);
121 :     if (base[0] != 0) {
122 :         SetCurrentDirectory(base);
123 :     }
124 :
125 :     return TRUE;
126 : }
127 : //----- ウインド破棄 ----------------------------------------------------------------------------------------------//
128 : AJC_DLGPROC(Main, WM_DESTROY        )
129 : {
130 :     AjcSaveAllControlSettings(hDlg);
131 :     PostQuitMessage(0);
132 :     return TRUE;
133 : }
134 : //----- 「実行」ボタン --------------------------------------------------------------------------------------------//
135 : AJC_DLGPROC(Main, IDC_CMD_RUN       )
136 : {
137 :     int     rsu;
138 :     UT      path1[MAX_PATH];
139 :     UT      path2[MAX_PATH];
140 :
141 :     AjcEnableDlgItem(hDlg, IDC_CMD_RUN , FALSE);
142 :     AjcEnableDlgItem(hDlg, IDC_CMD_STOP, TRUE);
143 :     fCan = fEnd = FALSE;
144 :     fBusy = TRUE;
145 :     AjcGetDlgItemStr(hDlg, IDC_TXT_PATH1, path1, MAX_PATH);
146 :     AjcGetDlgItemStr(hDlg, IDC_TXT_PATH2, path2, MAX_PATH);
147 :     rsu = AjcEnumFilesMatchingList(path1[0] != 0 ? path1 : NULL,
148 :                                    path2[0] != 0 ? path2 : NULL, AJCFML_NODIR, (UX)hDlg, cbQuery, cbEnum);
149 :     AjcVthPrintF(hVth, TEXT("¥nResult = %d¥n¥n"), rsu);
150 :     AjcEnableDlgItem(hDlg, IDC_CMD_RUN , TRUE);
151 :     AjcEnableDlgItem(hDlg, IDC_CMD_STOP, FALSE);
152 :     fBusy = FALSE;
153 :     if (fEnd) DestroyWindow(hDlg);
154 :     return TRUE;
155 : }
156 : //----- 「中止」ボタン --------------------------------------------------------------------------------------------//
157 : AJC_DLGPROC(Main, IDC_CMD_STOP      )
158 : {
159 :     if (fBusy) fCan = TRUE;
160 :     return TRUE;
161 : }
162 : //----- 「Cancel」 ------------------------------------------------------------------------------------------------//
163 : AJC_DLGPROC(Main, IDCANCEL          )
164 : {
165 :     if (fBusy) fEnd = TRUE;
166 :     else DestroyWindow(hDlg);
167 :     return TRUE;
168 : }
169 : //-----------------------------------------------------------------------------------------------------------------//
170 : AJC_DLGMAP_DEF(Main)
171 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG     )
172 :     AJC_DLGMAP_MSG(Main, WM_DESTROY        )
173 :
174 :     AJC_DLGMAP_CMD(Main, IDC_CMD_RUN       )
175 :     AJC_DLGMAP_CMD(Main, IDC_CMD_STOP      )
176 :     AJC_DLGMAP_CMD(Main, IDCANCEL          )
177 : AJC_DLGMAP_END
178 :
```

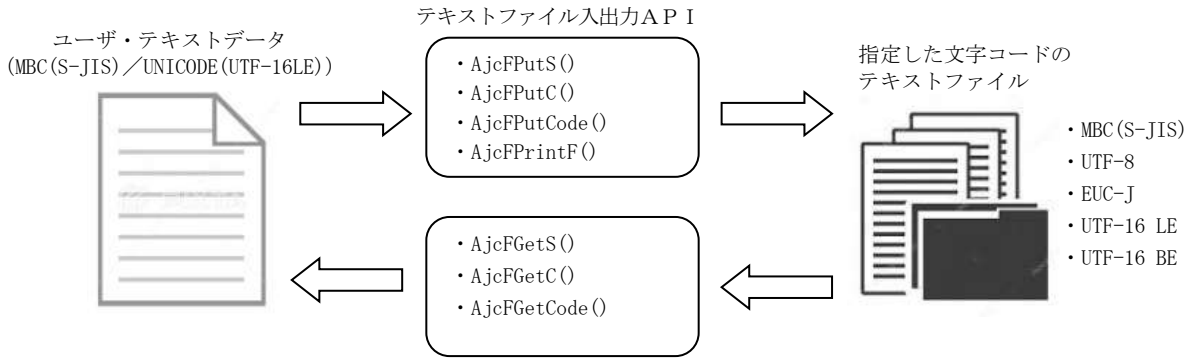
31. テキストファイル・アクセス

テキストファイルの文字コードを指定し、テキストファイルの入出力を行います。
指定可能な文字コードは、以下の通りです。

・マルチバイト（S-JIS）
・ＵＴＦ－８
・ＥＵＣ－ＪＰ（日本語ＥＵＣ）
・ＵＴＦ－１６（リトルエンディアン）
・ＵＴＦ－１６（ビッグエンディアン）

「マルチバイト(S-JIS)」は、規定のマルチバイト文字を意味しますが、特に、日本語環境ではシフトＪＩＳ（コードページ９３２）であることを示します。

各テキストファイル入出力ＡＰＩでは、バイトモードではマルチバイト(S-JIS)を、UNICODE モードでは UTF-16LE のテキストを扱います。



改行コードの変換

AjcFSetLfConv()により、さまざまな改行コードの変換モードを指定することができます。(Linux や Mac のテキストファイルにも対応)
ファイル読み出し時のデフォルトでは、CR(0x0D),LF(0x0A)の２文字を、LF(0x0A)に変換します。
ファイル書き込み時のデフォルトでは、LF(0x0A)を、CR(0x0D),LF(0x0A)の２文字に変換します。

マルチバイト文字／サロゲートペア

入力テキストに複数バイト文字やサロゲートペア文字が含まれる場合、先頭バイト／先頭ワードで複数バイト／ワード文字を判断し、後続のバイトワードに関しては妥当性をチェックしません。(変換不能な場合は、「?」に変換されます)
ファイル読み出し時、ファイルの末尾が半端な複数バイト文字／上位サロゲートペアである場合、この末尾の文字は無視されて、読み出されません。(ex. S-JIS ファイルの末尾が 93 8C 8B 9E 93 (“東京” + ”都”の１バイト目) の場合 93 8C 8B 9E(“東京”)のみ読み出す)
ファイル書き込みに指定されたテキストの末尾が、半端な複数バイト文字／上位サロゲートペアである場合、半端な複数バイト文字／上位サロゲートペアは退避され、次回の書き込みテキスト（複数バイト文字後半／下位サロゲートペア）と連結します。

ＢＯＭ (Byte Order Mark)

ファイルの先頭に作成される、ファイルのテキスト文字コードを表す符号（２～３バイト)。
ＢＯＭの実際のコードは、以下のとおりです。

テキスト文字コード	ＢＯＭデータ	備考
UTF-8	0xEF 0xBB 0xBF	３バイト
UTF-16LE	0xFF 0xFE	２バイト
UTF-16BE	0xFE 0xFF	２バイト

31.1. サポートＡＰＩ

テキストファイル・アクセスのサポートＡＰＩ一覧を以下に示します。

#	機能	関数名	備考
1	入力テキストファイルオープン	AjcFOpen / AjcFOpenShare	
2	文字列入力	AjcFGetS[Ex]	
3	文字コード入力	AjcFGetCode	
4	１バイト／１ワード入力	AjcFGetC	
5	ファイル読み出しポイント退避	AjcFSavePoint	
6	ファイル読み出しポイント回復	AjcFRecvPoint	
7	ファイル読み出しポイント取得	AjcFGetPoint	
8	ファイル読み出しポイント設定	AjcFSetPoint	
9	ファイル読み出しバイトポイント取得	AjcFGetBytePoint	
10	入力ファイルのＥＯＦチェック	AjcFEof	
11	出力テキストファイル生成	AjcFCreate / AjcFCreateShare	
12	ファイルを追記モードでオープン／生成	AjcFAppend / AjcFAppendShare	
13	文字列出力	AjcFPutS	
14	文字コード出力	AjcFPutCode	
15	１バイト／１ワード出力	AjcFPutC	
16	書式文字列出力	AjcFPrintF AjcFVPrintF	可変個パラメタ指定
17	テキストファイル出力データフラッシュ	AjcFFlush	
18	テキストファイルクローズ	AjcFClose	
19	改行コード変換モード設定／取得	AjcF{Set/Get}InpLfMode	
20	テキスト文字コード種別取得	AjcFGetTec	
21	入力ファイルのＢＯＭ有無の取得	AjcFGetExistBOM	
22	ファイル入出力種別取得	AjcFIsModeInput	
23	書式文字列バッファサイズ設定	AjcFSetFmtLen	
24	システムのファイルハンドル取得	AjcFGetFileHandle	
25	文字コード種別設定ダイアログ	AjcFTecDialog	
26	システムのファイルハンドルでファイル入力用インスタンス生成	AjcFAttachOpened	
27	システムのファイルハンドルでファイル出力用インスタンス生成	AjcFAttachCreated	
28	ファイルをクローズしないでインスタンス解放	AjcFDetach	

31.1.1. 入力テキストファイル オープン（AjcFOpen / AjcFOpenShare）

形　式 ： HAJCFILE AjcFOpen (C_UTP pPath, EAJCTEC tec);
HAJCFILE AjcFOpenShare (C_UTP pPath, EAJCTEC tec, UI share);

引　数 ： pPath - 入力テキストファイルのパス名
tec - 入力テキストファイルの文字コード
share - ファイル共有モード（0（共有禁止）, FILE_SHARE_READ, FILE_SHARE_WRITE}）

説　明 ： 指定したテキストファイルを読み出し用にオープンします・
指定したテキストファイルは、既に存在していなければなりません。
「tec」は、以下のいずれかで、入力テキストファイルの文字コードを指定します。

シンボル	値	内容	備考
AJCTEC_MBC	0	マルチバイト	日本語の場合は S-JIS
AJCTEC_UTF_8	1	ＵＴＦ－８	
AJCTEC_EUC_J	2	ＥＵＣ（日本語）	
AJCTEC_UTF_16LE	3	ＵＴＦ－１６（リトルエンディアン）	
AJCTEC_UTF_16BE	4	ＵＴＦ－１６（ビッグエンディアン）	
AJCTEC_AUTO	9	ＡＵＴＯ（自動検出）	値が範囲外の場合は AJCTEC_AUTO を仮定

但し、テキストファイルの先頭が UTF-8/UTF-16 のＢＯＭである場合は、「tec」の指定は無視されて、ＢＯＭで示される文字コードとみなします。
テキストファイルの先頭がＢＯＭ以外で、AJCTEC_AUTO を指定した場合は、テキストファイルの先頭（最大 16382 バイト）から自動的に文字コード種別を判断します。
AjcFGetS(), AjcFGetC()等でファイルの読み出しを行う際は、「tec」で指定された文字コードから、マルチバイト（S-JIS）／UNICODE に変換されたテキストが取得されます。

戻り値 ： ≠NULL - 成功 （ファイルハンドルを返します）
＝NULL - 失敗

31.1.2. 文字列入力（AjcFGetS）

形　式 ： UTP AjcFGetS (HAJCFILE hFile, UTP pBuf, UI lBuf);
UTP AjcFGetSEx(HAJCFILE hFile, UTP pBuf, UI lBuf, UX cbp, BOOL (CALLBACK *cbNtcRPos)(ULL pos, UX cbp));

引　数 ： hFile - 入力テキストファイルのファイルハンドル
pBuf - 読み出した文字列を格納するバッファのアドレス
lBuf - 読み出した文字列を格納するバッファのサイズ（バイト数／文字数, ３以上を指定要）
cbp - コールバックパラメタ
cbNtcRPos - 入力ファイル位置通知用コールバック関数（不要時は NULL）

説　明 ： テキストファイルからテキストを読み出します。
読み出したテキストは AjcFOpen()で指定されたテキストコードから S-JIS/UNICODE に変換してバッファに格納します。
読み出しは、改行／ＥＯＦ検出, 制御文字検出,、あるいは、バッファ満杯まで行われます。
改行コード／ＥＯＦまでのテキストをバッファに格納できない場合は、いずれかの制御コードまでのテキストをバッファへ格納します。
いずれかの制御コードまでのテキストをバッファへ格納できない場合は、バッファが満杯となるまでのテキストを格納します。
バッファに格納したテキストの末尾には必ず、終端文字(0x00 / 0x0000)が付加されます。
従って、最大で、「lBuf」で指定するバッファサイズより１つ少ないバイト数／文字数のテキストが読み出されます。
「lBuf」は、３以上を指定してください。

AjcFGetSEx()は、ファイルの入力バイト／文字位置を通知するためのコールバック関数を指定できます。

戻り値 ： ≠NULL - 成功 （pBuf を返します）
＝NULL - ＥＯＦ

注　意　：　改行コードや、いずれかの制御文字までのテキストをバッファに格納できない場合は、入力ファイルから１文字づつ解析しながら、バッファに格納するため、多大な処理時間を要します。
　この１文字づつ解析するテキストが長すぎる場合、処理が長時間ブロックされる（AjcFGetS()から戻らない）ことになります。
　この状態を回避するには、コールバック関数（cbNtcRPos()）で、キャンセル・ボタン等に反応し、FALSE を返すことにより、ファイルの読み出しを停止することが必要になります。

コールバック：　コールバック関数の仕様は、以下のとおりです。

cbNtcRPos（ファイル読み出し位置の通知）

形　式　：　BOOL　CALLBACK ***cbNtcRPos***(ULL pos, UX cbp) ;

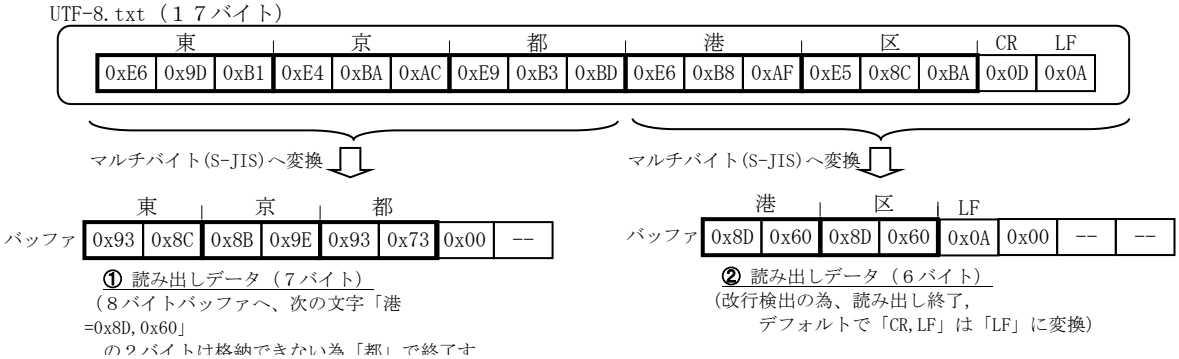
引　数　：　pos　　- 現在のファイル入力バイト／文字位置
　　cbp　　- コールバックパラメタ

説　明　：　現在のファイル読み出しバイト位置／文字位置を通知します。
　　バイト文字（SJIS/UTF-8/EUC-J）ファイルの場合は、バイト位置を通知します。
　　UNICODE ファイルの場合は、文字位置（１６ビット単位のワード位置）を通知します。

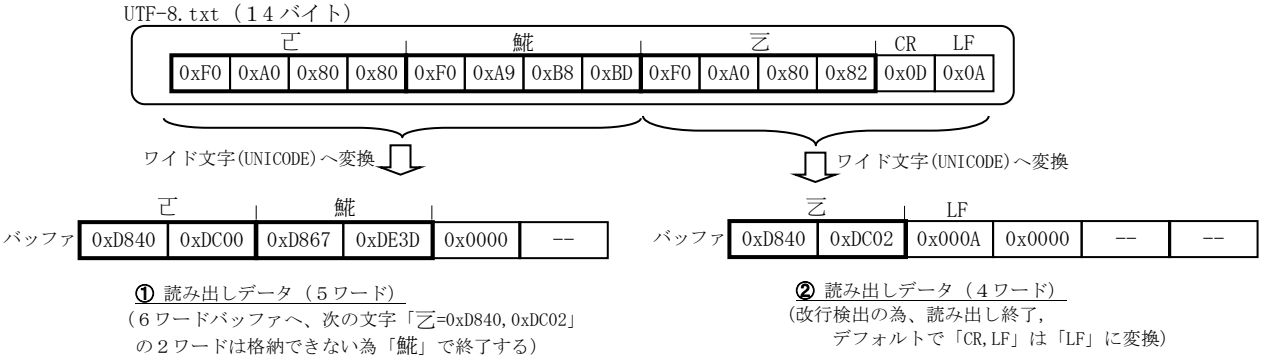
戻り値　：　TRUE　：ファイルの読み出しを継続する
　　FALSE ：ファイルの読み出しを中止する（AjcFGetSEx()は、NULL(EOF)を返す）

備　考　：　テキストファイルの入力時、途切れた文字がバッファに格納されることはありません。
　マルチバイト文字や、UNICODE 文字(サロゲートペア)が途中で途切れた場合は、途切れた文字はバッファに格納しないで、次回のアクセスで取得されます。

　例えば、バイト文字モードで、文字列 ”東京都港区¥r¥n” が格納されているＵＴＦ－８テキストファイルを８バイトバッファで読み出した場合は 以下のように読み出されます。



　また、ワイド文字モードで、サロゲートペア文字列 ”𠮟鮎乙¥r¥n” が格納されているＵＴＦ－８テキストファイルを６ワードバッファで読み出した場合は 以下のように読み出されます。

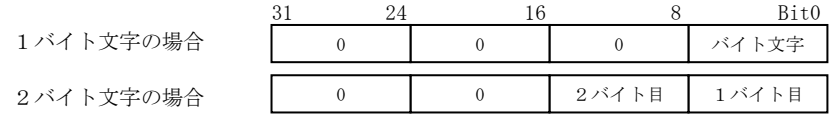


31.1.3. 文字コード入力（AjcFGetCode）

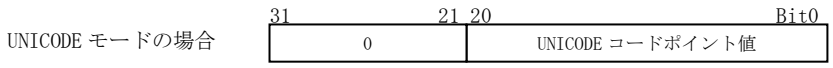
形　式 ： UI　AjcFGetCode（HAJCFILE　hFile);

引　数 ： hFile　　- 入力テキストファイルのファイルハンドル

説　明 ： テキストファイルから、AjcFOpen()で指定されたテキストコードから S-JIS/UNICODE に変換した１文字を読み出します。
バイトモード時は、読み出したバイトの文字コードを返します。
全角文字（２バイト文字）の場合は、２バイトを読み出して全角文字コードを返します。



UNICODE モード時は、読み出したデータを UNICODE のコードポイントに変換した値を返します。
サロゲートペアの場合は、２ワードを読み出して UNICODE コードポイントに変換します。



戻り値 ： ≠-1 - 読み出した文字コード
＝-1 - ＥＯＦ

31.1.4. １バイト／１ワード入力（AjcFGetC）

形　式 ： UT　AjcFGetC（HAJCFILE　hFile);

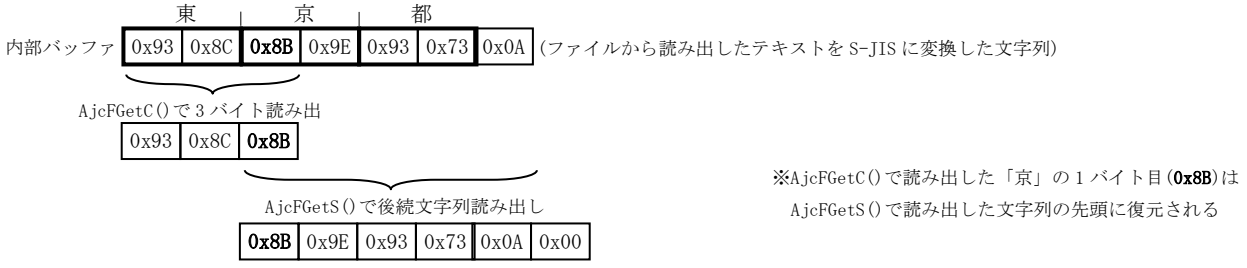
引　数 ： hFile　　- 入力テキストファイルのファイルハンドル

説　明 ： テキストファイルから、AjcFOpen()で指定されたテキストコードから S-JIS/UNICODE に変換された１バイト(8Bit)／１ワード(16Bit)を読み出します。
バイトモードでマルチバイト文字（２バイト文字）を読み出す際や、UNICODE モードでサロゲートペア文字を読み出す際は、本ＡＰＩを２回実行する必要があります。

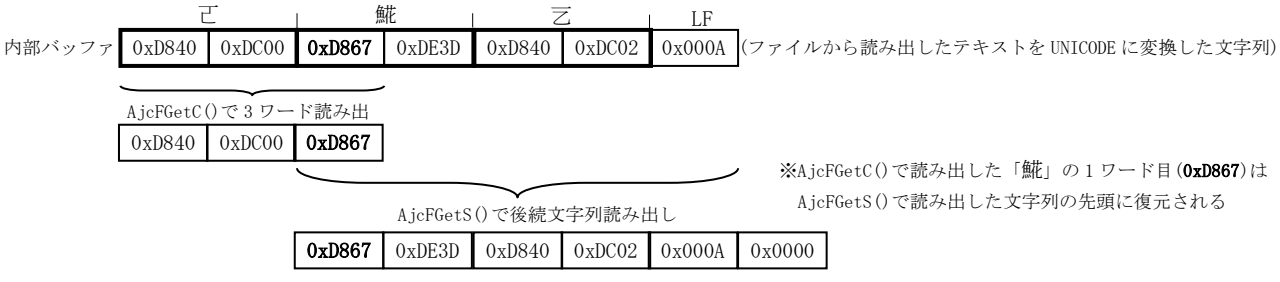
戻り値 ： ≠（UT)-1 - バイトモード時　：１バイトデータ
UNICODE モード時：１ワードデータ
＝（UT)-1 - ＥＯＦ

注　意 ：基本的に、AjcFGetS[Ex]()や、AjcFGetCode()と、AjcFGetC()を混同して使用しないでください。
AjcFGetC()でマルチバイト文字の１バイト目や、サロゲートペア文字の１ワード目を読み出した状態で、次に
AjcFGetS()
により後続文字列を読み出す（あるいは、AjcFGetCode()により１文字読み出す）場合、最後に AjcFGetC()で読み出したマルチバイト１バイト目／サロゲート１ワード目からの文字列（あるいは、文字）が取得されます。

バイト文字モードの例



ワイド文字モードの例



31.1.5. ファイル読み出しポイント退避（AjcFSavePoint）

形　式 ： ULL　AjcFSavePoint (HAJCFILE　hFile);

引　数 ： hFile　　- 入力テキストファイルのファイルハンドル

説　明 ： 現在のファイル読み出し位置を退避します。
現在のファイル読み出し位置がマルチバイト文字の２バイト目／サロゲートペアの２ワード目である場合は、マルチバイト文字１バイト目／サロゲートペアの１ワード目の位置に訂正されます。
AjcFRecvPoint()で読み出し位置を回復し、再度、現在の読み出し位置から読み出すことができます。

戻り値 ： ≠-1 – 退避したファイル読み出し位置（バイト位置／文字位置）
＝-1 - 失敗

31.1.6. ファイル読み出しポイント回復（AjcFRecvPoint）

形　式 ： BOOL　AjcFRecvPoint (HAJCFILE　hFile);

引　数 ： hFile　　- 入力テキストファイルのファイルハンドル

説　明 ： AjcFSavePoint()で退避したファイル読み出し位置を回復します。
再度、AjcFSavePoint()実行時の読み出し位置から読み出すことができます。

戻り値 ： TRUE　- 成功
FALSE – 失敗

31.1.7. ファイル読み出しポイント取得（AjcFGetPoint）

形　式 ： ULL　AjcFGetPoint (HAJCFILE　hFile);

引　数 ： hFile　　- 入力テキストファイルのファイルハンドル

説　明 ： 現在のファイル読み出し位置を取得します。
読み出しファイルがバイト文字ファイル（SJIS / UTF-8 / EUC）の場合はバイト位置を返します。
読み出しファイルがワイド文字ファイル（UTF-16(LE) / UTF-16(BE)）の場合は文字位置を返します。
現在のファイル読み出し位置がマルチバイト文字の２バイト目／サロゲートペアの２ワード目である場合は、マルチバイト文字１バイト目／サロゲートペアの１ワード目の位置に訂正されます。
先頭がＢＯＭであるファイルの場合でも、ファイルの先頭が０となります。

戻り値 ： ≠-1 - 現在のファイル読み出し位置（バイト位置／文字位置）
＝-1 - 失敗

31.1.8. ファイル読み出しポイント設定（AjcFSetPoint）

形　式 ： BOOL　AjcFSetPoint (HAJCFILE　hFile, ULL point);

引　数 ： hFile　　- 入力テキストファイルのファイルハンドル
point　　- 設定するファイル読み出し位置（バイト位置／文字位置）

説　明 ： ファイル読み出し位置を設定します。
読み出しファイルがバイト文字ファイル（SJIS / UTF-8 / EUC）の場合はバイト位置を設定します。
読み出しファイルがワイド文字ファイル（UTF-16(LE) / UTF-16(BE)）の場合は文字位置を設定します。
point がファイルのバイト数／文字数を超える場合はファイル末尾に設定します。
先頭がＢＯＭであるファイルの場合は、ＢＯＭの位置が０となります。

戻り値 ： TRUE　- 成功
FALSE – 失敗

注　意 ： point には、AjcFGetPoint()で取得した読み出し位置を設定するようにしてください。
読み出し位置をマルチバイト文字の２バイト目／サロゲートペアの２ワード目に設定しないでください。
(設定自体は可能ですが、以降、意図しない文字が入力される場合があります)

31.1.9. ファイル読み出しバイトポイント取得（AjcFGetBytePoint）

形　式 ： ULL　AjcFGetBytePoint（HAJCFILE　hFile);

引　数 ： hFile　　- 入力テキストファイルのファイルハンドル

説　明 ： 現在のファイル読み出しバイト位置を取得します。
　読み出しファイルがバイト文字ファイル（SJIS / UTF-8 / EUC）ワイド文字ファイル（UTF-16(LE) / UTF-16(BE)）に関わらず、バイト単位の読み出し位置を返します。
　現在のファイル読み出し位置がマルチバイト文字の２バイト目／サロゲートペアの２ワード目である場合は、マルチバイト文字１バイト目／サロゲートペアの１ワード目の位置に訂正されます。
　先頭がＢＯＭであるファイルの場合でも、ファイルの先頭が０となります。

戻り値 ： ≠-1 - 現在のファイル読み出し位置（バイト位置／文字位置）
　　＝-1 - 失敗

31.1.10. 入力ファイルのＥＯＦチェック（AjcFEof）

形　式 ： BOOL　AjcFEof（HAJCFILE　hFile);

引　数 ： hFile　　- 入力テキストファイルのファイルハンドル

説　明 ： 入力ファイルがＥＯＦ（ファイルの終端）に達したかをチェックします。

戻り値 ： TRUE　- ＥＯＦ
　　FALSE – ＥＯＦ以外

31.1.11. 出力テキストファイル 生成（AjcFCreate / AjcFCreateShare）

形　式 ： HAJCFILE　AjcFCreate　　(C_UTP pPath, EAJCTEC tec, BOOL fBOM);
　　HAJCFILE　AjcFCreateShare (C_UTP pPath, EAJCTEC tec, BOOL fBOM, UI share);

引　数 ： pPath　　- 出力テキストファイルのパス名
　　tec　　- 出力テキストファイルの文字コード
　　fBOM　　- UTF-8/UTF-16 の BOM 出力フラグ（FALSE:出力しない, TRUE:出力する）
　　share　　- ファイル共有モード（0（共有禁止）, FILE_SHARE_READ, FILE_SHARE_WRITE}）

説　明 ： 指定したテキストファイルを書き込み用に作成します。
　　指定したテキストファイルが既に存在している場合は、既存のテキストファイルのデータは消去されます。
　　「tec」は、以下のいずれかで、出力テキストファイルの文字コードを指定します。

シンボル	値	内容	備考
AJCTEC_MBC	0	マルチバイト	日本語の場合は S-JIS
AJCTEC_UTF_8	1	ＵＴＦ－８	
AJCTEC_EUC_J	2	ＥＵＣ（日本語）	
AJCTEC_UTF_16LE	3	ＵＴＦ－１６（リトルエンディアン）	
AJCTEC_UTF_16BE	4	ＵＴＦ－１６（ビッグエンディアン）	
AJCTEC_AUTO	9	同一スレッドで最後にオープンしたファイルのテキストエンコード（未オープン時はAJCTEC_MBC）	値が範囲外の場合はAJCTEC_AUTO を仮定

　　AjcFPutS()、AjcFPrintF()や、AjcFPutC()等で、ファイルへ出力する場合は、マルチバイト（S-JIS)／UNICODE から「tec」で指定された文字コードに変換しファイルへ出力されます。

　　「fBOM」は、「tec」に「AJCTEC_UTF_8, AJCTEC_UTF_16LE or AJCTEC_UTF_16BE」を指定した場合だけ有効なフラグで、「TRUE」を指定すると、最初のファイル出力時に（ファイルの先頭に）UTF-8/UTF-16 の BOM を出力します。

戻り値 ： ≠NULL - 成功（ファイルハンドルを返します）
　　＝NULL - 失敗

31.1.12. ファイルを追記モードでオープン／生成（AjcFAppend / AjcFAppendShare）

形　式 ： HAJCFILE　AjcFAppend　　(C_UTP pPath, EAJCTEC tec, BOOL fBOM);
HAJCFILE　AjcFAppendShare(C_UTP pPath, EAJCTEC tec, BOOL fBOM, UI share);

引　数 ： pPath　　- 出力テキストファイルのパス名
tec　　- 入力テキストファイルの文字コード
fBom　　- ＢＯＭ出力フラグ
share　　- ファイル共有モード（0（共有禁止）, FILE_SHARE_READ, FILE_SHARE_WRITE}）

説　明 ： ファイルが存在する場合は。既存のファイルを追記モードで生成します。
このファイルは GENERIC_WRITE 属性で作成され、以降、出力したテキストはファイルの末尾に追記されます。

tec = AJCTEC_AUTO を指定した場合は、ファイルエンコードを以下のように決定します。

・ファイルサイズ≠０の場合、ファイルの内容から決定する
・ファイルサイズ＝０の場合、同一スレッドで最後にオープンしたファイルのテキストエンコード（未オープン時は AJCTEC_MBC）

ファイルサイズ≠０の場合 fBom は無視されます。
ファイルが存在しない場合は、新たにファイルを生成します。（AjcFCreate / AjcFCreateShare と同じ）

戻り値 ： ≠NULL - 成功（ファイルハンドルを返します）
＝NULL - 失敗

31.1.13. 文字列出力（AjcFPutS）

形　式 ： BOOL　AjcFPutS（HAJCFILE　hFile, C_BCP pText, UI lText);

引　数 ： hFile　　- 出力テキストファイルのファイルハンドル
pText　　- 出力テキストデータのアドレス
lText　　- 出力テキストデータのバイト数／文字数（－１の場合は自動算出）

説　明 ： 指定したテキストデータを AjcFCreate()で指定した文字コードに変換して、ファイルへ書き込みます。
pText で指定するテキストデータは、バイトモード時はマルチバイト(S-JIS)で指定し、UNICODE モード時は UNICODE で指定します。
出力テキストデータ中にヌル文字(¥0)が含まれている場合は、ヌル文字(¥0)と、それ以降は出力されません。

戻り値 ： TRUE　- 成功
FALSE - 失敗

31.1.14. 文字コード出力（AjcFPutCode）

形　式 ： BOOL　AjcFPutCode（HAJCFILE　hFile, UI c);

引　数 ： hFile　　- 出力テキストファイルのファイルハンドル
c　　- 出力する文字コード（マルチバイト文字コード／UNICODE コードポイント）（０指定時は文字を出力しない）

説　明 ： 指定したテキスト１文字を AjcFCreate()で指定した文字コードに変換して、ファイルへ書き込みます。
バイトモード時、半角文字は８ビット値で指定し、全角文字は１６ビット値（Bit0-8:１バイト目, Bit8-15:２バイト目）で指定します。
UNICODE モード時は、UNICODE コードポイントを指定します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

31.1.15. １バイト／１ワード出力（AjcFPutC）

形　式 ： BOOL AjcFPutC (HAJCFILE hFile, UT c);

引　数 ： hFile　- 出力テキストファイルのファイルハンドル
c　- 出力する１バイト(8Bit)／１ワード(16Bit)　（０指定時は文字を出力しない）

説　明 ： 指定した S-JIS バイトデータ／UNICODE ワードデータを AjcFCreate()で指定した文字コードに変換してファイルへ書き込みます。
バイトモード時で全角文字を出力する場合や、UNICODE モード時でサロゲートペアを出力する場合は、本ＡＰＩを２回実行してください。

戻り値 ： TRUE - 成功
FALSE - 失敗

31.1.16. 書式文字列出力（AjcFPrintF）

形　式 ： BOOL AjcFPrintF (HAJCFILE hFile, C_UTP pFmt, ...);
BOOL AjcFVPrintF (HAJCFILE hFile, C_UTP pFmt, va_list vls);

引　数 ： hFile　- 出力テキストファイルのファイルハンドル
pFmt　- 書式文字列（printf()と同じ）
vls　- 引数リストへのポインタ

説　明 ： 書式化した文字列を AjcFCreate()で指定した文字コードに変換して、テキストファイルに出力します。
書式化の仕様は、printf()と同じです。
書式化文字列作成用のバッファは、デフォルトで 2048 文字(バイトモード時は 2048 バイト, UNICODE 文字時は 2048 文字)ですが、AjcFSetFmtLen()で変更できます。
このバッファは、AjcFPrintF(),AjcFVPrintF()の初回実行時か、 AjcFSetFmtLen()実行時に確保され、AjcFClose()実行時に解放されます。

戻り値 ： TRUE - 成功
FALSE - 失敗

31.1.17. テキストファイル出力データフラッシュ（AjcFFlush）

形　式 ： BOOL AjcFFlush (HAJCFILE hFile);

引　数 ： hFile　- 出力テキストファイルのファイルハンドル

説　明 ： バッファリングされているテキストデータをファイルへ書き出します。

戻り値 ： TRUE - 成功
FALSE - 失敗

31.1.18. テキストファイルクローズ（AjcFClose）

形　式 ： BOOL AjcFClose (HAJCFILE hFile);

引　数 ： hFile　- 入出力テキストファイルのファイルハンドル

説　明 ： AjcFOpen() / AjcFCreate() でオープン／作成したテキストファイルをクローズします。
以降、hFile で指定したファイルハンドルでテキストファイルをアクセスすることはできません。

戻り値 ： TRUE - 成功
FALSE - 失敗

31.1.19. 改行コード変換モード設定／取得（AjcF{Set/Get}LfConv）

形　式 ： BOOL　　AjcFSetLfConv (HAJCFILE　hFile, AJCFLFCONV mode);
AJCFLFCONV　AjcFGetLfConv (HAJCFILE　hFile);

引　数 ： hFile　　- 入出力テキストファイルのファイルハンドル
mode　　- 改行コード変換モード

説　明 ： 改行コードの変換モードを指定します。
指定できる改行コードの変換モードは、以下のとおりです。

#	mode	内容	AjcFGetS()で行末と判断するコード	備　考
1	AJCFLF_NONE	変換無し	－	改行を認識しないで、バッファ満杯まで読み出す
2	AJCFLF_LF_TO_CRLF	ＬＦ → ＣＲ＋ＬＦ	ＬＦ	ファイル書き込み時のデフォルト
3	AJCFLF_LF_TO_CR	ＬＦ → ＣＲ	ＬＦ	
4	AJCFLF_CR_TO_CRLF	ＣＲ → ＣＲ＋ＬＦ	ＣＲ	
5	AJCFLF_CR_TO_LF	ＣＲ → ＬＦ	ＣＲ	
6	AJCFLF_CRLF_TO_LF	ＣＲ＋ＬＦ → ＬＦ	ＣＲ＋ＬＦ	ファイル読み出し時のデフォルト
7	AJCFLF_CRLF_TO_CR	ＣＲ＋ＬＦ → ＣＲ	ＣＲ＋ＬＦ	
8	AJCFLF_LF	変換無し	ＬＦ	
9	AJCFLF_CR	変換無し	ＣＲ	
10	AJCFLF_LF_CRSKIP	ＣＲ読み飛ばし	ＬＦ	ＣＲは除去する
11	AJCFLF_CR_LFSKIP	ＬＦ読み飛ばし	ＣＲ	ＬＦは除去する

ＣＲ＝0x0D, ＬＦ＝0x0A

戻り値 ： TRUE　- 成功
FALSE - 失敗

31.1.20. テキスト文字コード種別取得（AjcFGetTec）

形　式 ： EAJCTEC AjcFGetTec (HAJCFILE　hFile);

引　数 ： hFile　　- 入出力テキストファイルのファイルハンドル

説　明 ： AjcFOpen() / AjcFCreate() でオープン／生成したテキストファイルの文字コードを返します。
基本的に、AjcFOpen() / AjcFCreate()の「tec」引数で指定した値を返しますが、AjcFOpen()でオープンしたテキストファイルの先頭にUTF-8/UTF-16 のBOMがある場合は、当該BOMが示すコード（AJCTEC_UTF_8, AJCTEC_UTF_16LE or AJCTEC_UTF_16BE）を返します。

戻り値 ： テキストファイルの文字コード
・AJCTEC_MBC　　：マルチバイト(S-JIS)
・AJCTEC_UTF_8　　：ＵＴＦ－８
・AJCTEC_EUC_J　　：日本語ＥＵＣ
・AJCTEC_UTF_16LE：ＵＴＦ－１６（リトルエンディアン）
・AJCTEC_UTF_16BE：ＵＴＦ－１６（ビッグエンディアン）
・AJCTEC_ERROR　　：エラー（パラメタエラー）

31.1.21. 入力ファイルのＢＯＭ有無の取得（AjcFGetBOM）

形　式 ： BOOL AjcFGetBOM (HAJCFILE　hFile);

引　数 ： hFile　　- 入出力テキストファイルのファイルハンドル

説　明 ： 入力ファイルの先頭にＢＯＭが有ったか否かを取得します。
このＡＰＩはAjcFOpen()の後に実行してください。

戻り値 ： ファイルの入出力モード
TRUE　　- 入力モード（AjcFOpen()でオープンしたファイル）
FALSE　- 出力モード（AjcFCreate()で作成されたファイル）

31.1.22. ファイル入出力種別取得（AjcFIsModeInput）

形　式 ： BOOL AjcFIsModeInput (HAJCFILE hFile);

引　数 ： hFile　 - 入出力テキストファイルのファイルハンドル

説　明 ： ファイルが入力モード（AjcFOpen()でオープンした）か、出力モード（AjcFCreate()で作成されたか）の識別を取得します。

戻り値 ： ファイルの入出力モード
TRUE　 - 入力モード（AjcFOpen()でオープンしたファイル）
FALSE　 - 出力モード（AjcFCreate()で作成されたファイル）

31.1.23. 書式文字列バッファサイズ設定（AjcFSetFmtLen）

形　式 ： BOOL AjcFSetFmtLen (HAJCFILE hFile , UI len);

引　数 ： hFile　 - 出力テキストファイルのファイルハンドル
len　　 - 書式文字列バッファ長（文字列の終端(0x00)を含むバイト数／文字数，１６以上）

説　明 ： AjcFPrintF()で使用する、書式化した文字列作成用のバッファサイズ（バイト数／文字数，１６以上）を指定します。
書式化文字列作成用のバッファは、バイトモード時はバイト数で，UNICODE モード時は文字数で指定します。
バッファサイズのデフォルトは、2048 バイト／文字です。

戻り値 ： TRUE　- 成功
FALSE - 失敗

31.1.24. システムのファイルハンドル取得（AjcFGetFileHandle）

形　式 ： HANDLE AjcFGetFileHandle (HAJCFILE hFile);

引　数 ： hFile　 - 出力テキストファイルのファイルハンドル

説　明 ： システムのファイルハンドルを取得します。
AjcFOpen()やAjcFCreate()では、CreateFile() (Windows API) によりファイルをオープン／作成しています。
このＡＰＩでは、CreateFile() (Windows API) により取得したハンドルを返します。
本ＡＰＩは、AjcFOpen()あるいはAjcFCreate()の後に実行してください。

戻り値 ： ≠NULL - 成功（システムのファイルハンドル）
＝NULL - 失敗

31.1.25. 文字コード種別設定ダイアログ（AjcFTecDialog）

形　式　：　BOOL AjcFTecDialog (HWND hOwner, C_UTP pTitle, HICON hIcon, PEAJCTEC pInpTec, PEAJCTEC pOutTec, BOOL *pOutBom);

引　数　：　hWoner　- 出力テキストファイルのファイルハンドル
pTitle　- タイトル文字列（NULL 指定時は、”テキスト・文字コード設定”）
hIcon　- アイコンハンドル（不要時は NULL）
pInpTec　– 入力テキストファイルの文字コードバッファのアドレス（不要時は NULL）
pOutTec　- 出力テキストファイルの文字コードバッファのアドレス（不要時は NULL）
pOutBom　- 出力テキストファイルへのＢＯＭ出力フラグバッファのアドレス（不要時は NULL）

説　明　：　文字コード種別設定用のダイアログを表示します。
pTitle は、タイトルバーに表示する文字列を指定します。
pTitle=NULL とした場合は ”テキスト・文字コード設定” を仮定します。
hIcon はダイアログのタイトルバーに表示するアイコンのハンドルを指定します
pInpTec, pOutTec, pOutBo で示されるバッファの内容は、ダイアログでの初期値表示用に使用され、ダイアログ終了時にはダイアログで設定された値が格納されます。
pInpTec, pOutTec は、以下に示す、入出力テキストファイルの文字コードを指定／格納します。

・AJCTEC_MBC　：マルチバイト(S-JIS)
・AJCTEC_UTF_8　：ＵＴＦ－８
・AJCTEC_EUC_J　：日本語ＥＵＣ
・AJCTEC_UTF_16LE　：ＵＴＦ－１６（リトルエンディアン）
・AJCTEC_UTF_16BE　：ＵＴＦ－１６（ビッグエンディアン）
・AJCTEC_AUTO　：自動設定

pOutBom は、UTF-8/UTF-16 で出力する際にＢＯＭを付加する(TRUE)か、付加しない(FALSE)を指定／格納します。
pInpTec, pOutTec や pOutBom に NULL を指定した場合、当該項目は無効表示（グレー表示）されます。

このＡＰＩは、ユーザがダイアログで設定値を取得するために用意されているだけです。
他のテキストファイルアクセスＡＰＩ(AjcFOpen(), AjcFCreate()や AjcFGetS()等)とは関連しません。

このダイアログの表示イメージは、以下のとおりです。



戻り値　：　TRUE　：ＯＫ
FALSE：キャンセル

31.1.26. システムのファイルハンドルでファイル入力用インスタンス生成（AjcFAttachOpened）

形　式　：　HAJCFILE　AjcFAttachOpened　(HANDLE hFile, EAJCTEC tec);

引　数　：　hFile　　- オープン済みファイルのハンドル
　　tec　　- 入力テキストファイルの文字コード

説　明　：　CreateFile()でオープン済みのファイルハンドルを指定して、ファイル入力用のインスタンスを生成します。
　　このオープン済みファイルは GENERIC_READ 属性を持っていなければなりません。
　　その他は、AjcFOpen()と同じです。

戻り値　：　≠NULL - 成功（ファイルハンドルを返します）
　　＝NULL - 失敗

備　考　：　AjcFAttachOpened(),AjcFAttachCreated()と、AjcFDetach()は、ファイルのオープンとクローズを自前で行う場合に使用します。(以下の例は、オープンとクローズを自前で行い、ファイルの入力(自動判別)と出力(UTF-8)を行います)
　　ファイルをクローズする際は、AjcFDetach()で、インスタンスを解放する必要があります。

```
#include    <AjrCstXX.h>
int main()
{
    HANDLE      hInp = NULL, hOut = NULL;
    HAJCFILE    fInp = NULL, fOut = NULL;
    UT          path[MAX_PATH] = TEXT("d:¥¥work¥¥a1¥¥xxx.txt");
    UT          buf[1024];

    //  出力ファイル生成
    hOut = CreateFile(path, GENERIC_WRITE, FILE_SHARE_READ , NULL, CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
    fOut = AjcFAttachCreated(hOut, AJCTEC_UTF_8, TRUE); //  UTF-8, BOM 出力
    //
    AjcFPutS(fOut, TEXT("AAA¥n"), -1);
    AjcFPutS(fOut, TEXT("BBB¥n"), -1);
    AjcFPutS(fOut, TEXT("CCC¥n"), -1);
    AjcFFlush(fOut);
    //  入力ファイルオープン
    hInp = CreateFile(path, GENERIC_READ , FILE_SHARE_WRITE, NULL, OPEN_EXISTING, 0, NULL);
    fInp = AjcFAttachOpened(hInp, AJCTEC_AUTO);
    //  ファイル読み出し
    while (AjcFGetS(fInp, buf, AJCTSIZE(buf))) {
        AjcPrintF(buf);
    }
    //  インスタンス解放
    AjcFDetach(fInp);
    AjcFDetach(fOut);
    //  入出力ファイルクローズ
    CloseHandle(hInp);
    CloseHandle(hOut);

    return 0;
}
```

31.1.27. システムのファイルハンドルでファイル出力用インスタンス生成（AjcFAttachCreated）

形　式 ： HAJCFILE　AjcFAttachCreated(HANDLE hFile, EAJCTEC tec, BOOL fBOM);

引　数 ： hFile　　- 生成済みファイルのハンドル
tec　　- 入力テキストファイルの文字コード
fBom　　- ＢＯＭ出力フラグ

説　明 ： CreateFile()で生成済みのファイルハンドルを指定して、ファイル出力用のインスタンスを生成します。
この生成済みファイルは GENERIC_WRITE 属性を持っていなければなりません。
ファイルサイズが０以外の場合 fBom は無視されます。
その他は、AjcFCreate()と同じです。

戻り値 ： ≠NULL - 成功（ファイルハンドルを返します）
＝NULL - 失敗

31.1.28. ファイルをクローズしないでインスタンス解放（AjcFDetach）

形　式 ： BOOL　AjcFDetach(HAJCFILE　hFile);

引　数 ： hFile　　- 入出力テキストファイルのファイルハンドル

説　明 ： システムのファイルハンドルをクローズしないで、テキストファイルアクセス用のインスタンスだけ解放します。
以降、hFile で指定したファイルハンドルでテキストファイルをアクセスすることはできません。
システムのファイルハンドルは、後で、自分でクローズ(CloseHandle())する必要があります。

戻り値 ： TRUE　- 成功
FALSE - 失敗

31.2. サンプルプログラム

31.2.1. SW_TextFile（テキストファイルのアクセスサンプル）

このサンプルプログラムは、テキストファイルのエンコード変換を行います。
「入力パス」は、エンコード変換を行うディレクトリを指定します。
「ワイルドカード」は、エンコード変換を行うファイル群のワイルドカードを指定します。(複数指定時はセミコロン(;)で区切る)
「入力ファイルの退避パス」は、エンコード変換を行う前に、対象ファイルを退避するディレクトリパスを指定します。
このディレクトリ下に、日時を付加したディレクトリが作成され、対象ソースファイルがコピーされます。

このサンプルプログラムでは、(実用性を持たせるために) エラーチェックも行っています。




```
 1 :  //
 2 :  //  SW_TextFile.c
 3 :  //
 4 : #include    <AjrCstXX.h>
 5 : #include    <shlwapi.h>
 6 : #include    "resource.h"
 7 :
 8 : #define MAX_CBOSTR  MAX_PATH
 9 : #define EXTRA_FILE_ATT (FILE_ATTRIBUTE_ARCHIVE | FILE_ATTRIBUTE_NORMAL)
10 : //-------------------------------------------------------------------------------------------------------------//
11 : //  ワーク                                                                                                     //
12 : //-------------------------------------------------------------------------------------------------------------//
13 : static  UT          AppName[] = TEXT("エンコード変換");
14 : static  HINSTANCE   hInst       = NULL;         //  自インスタンスハンドル
15 : static  HWND        hDlgMain    = NULL;         //  ダイアログハンドル
16 : static  HWND        hCboPath    = NULL;         //  入力パス・コンボボックスハンドル
17 : static  HWND        hTxtPath    = NULL;         //  入力パス・テキストボックスハンドル
18 : static  HWND        hCboWild    = NULL;         //  ワイルドカード・コンボボックスハンドル
19 : static  HWND        hTxtWild    = NULL;         //  ワイルドカード・テキストボックスハンドル
20 : static  HWND        hVth        = NULL;         //  VTH コントロール
21 :
22 : static  SIZE        WndMinSize;                 //  ダイアログ最小サイズ
23 : static  int         yVth;                       //  ＶＴＨ表示位置
24 :
25 : static  HAJCFILE    hLogFile    = NULL;         //  ログファイルハンドル
26 : static  BOOL        fLogFirst   = TRUE;         //  初回ログ出力フラグ
27 : static  EAJCTEC     TecTo       = AJCTEC_UTF_8; //  変換するファイルエンコード
28 : static  BOOL        fBusy       = FALSE;        //  変換処理中フラグ
29 : static  BOOL        fStop       = FALSE;        //  中止フラグ
30 : static  BOOL        fCan        = FALSE;        //  キャンセルフラグ
31 : static  UI          Count       = 0;            //  ファイルカウンタ
32 : static  UI          nOK         = 0;            //  成功数カウンタ
33 : static  UI          nNG         = 0;            //  失敗数カウンタ
34 : static  UI          nNC         = 0;            //  未変換カウンタ
35 : static  UT          szWild[256];                //  対象ファイルのワイルドカード
36 : static  UT          szSrcPath[MAX_PATH];        //  ソースパス
37 : static  UT          szSavPath[MAX_PATH];        //  ソース退避パス
38 : static  UT          szSavTime[MAX_PATH];        //  ソース退避サブパス (末尾に日時付加)
39 : static  UT          szLogFile[MAX_PATH];        //  ログファイルパス
40 :
41 : //-------------------------------------------------------------------------------------------------------------//
```

```
42 : //  内部サブ関数                                                                                                    //
43 : //------------------------------------------------------------------------------------------------------------------//
44 : AJC_DLGPROC_DEF(Main    );
45 : static  BOOL        SaveSource(HWND hDlg, C_UTP pWild);
46 : static  BOOL        ChangeTec (HWND hDlg, C_UTP pWild);
47 : static  BOOL        ChkInpTec(HWND hDlg, EAJCTEC tec, BOOL bom);
48 : static  C_UTP       TecName(EAJCTEC tec, BOOL bom);
49 : static  VO          LogPrintF(C_UTP pFmt, ...);
50 :
51 : //==================================================================================================================//
52 : //                                                                                                                  //
53 : //  W i n M a i n                                                                                                   //
54 : //                                                                                                                  //
55 : //==================================================================================================================//
56 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
57 : {
58 :     MSG         msg;
59 :
60 :
61 :     hInst = hInstance;
62 :
63 :     //----- メイン・ダイアログオープン ----------------//
64 :     hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_MAIN), NULL, AJC_DLGPROC_NAME(Main));
65 :     ShowWindow(hDlgMain, SW_SHOW);
66 :
67 :     //----- メッセージループ --------------------------//
68 :     while (GetMessage(&msg, NULL, 0, 0)) {
69 :         do {
70 :             if (IsDialogMessage(hDlgMain, &msg)) break;
71 :             TranslateMessage(&msg);
72 :             DispatchMessage (&msg);
73 :         } while (0);
74 :     }
75 :     return (int)msg.wParam ;
76 : }
77 : //==================================================================================================================//
78 : //                                                                                                                  //
79 : //  ダイアログ・プロシージャ                                                                                        //
80 : //                                                                                                                  //
81 : //==================================================================================================================//
82 : //----- ダイアログ初期化 ---------------------------------------------------------------------------------------//
83 : AJC_DLGPROC(Main, WM_INITDIALOG )
84 : {
85 :     RECT    rect;
86 :     UT      path[MAX_PATH];
87 :
88 :     hDlgMain = hDlg;
89 :     hCboPath = GetDlgItem(hDlg, IDC_CBO_PATH);  hTxtPath = AjcGetCtrlCboEditHandle(hCboPath);
90 :     hCboWild = GetDlgItem(hDlg, IDC_CBO_WILD);  hTxtWild = AjcGetCtrlCboEditHandle(hCboWild);
91 :     hVth     = GetDlgItem(hDlg, IDC_VTH);
92 :     //  ダイアログ最小サイズ設定
93 :     GetWindowRect(hDlg, &rect);
94 :     WndMinSize.cx = rect.right - rect.left;
95 :     WndMinSize.cy = rect.bottom - rect.top;
96 :     //  VTH 先頭位置設定
97 :     GetWindowRect(hVth, &rect);
98 :     MapWindowPoints(NULL, hDlg, (LPPOINT)&rect, 2);
99 :     yVth = rect.top;
100 :    //  入力退避パスデフォルト設定
101 :    GetTempPath(MAX_PATH, path);
102 :    AjcPathCat(path, TEXT("ChangeTextEncode"), MAX_PATH);
103 :    CreateDirectory(path, NULL);
104 :    AjcSetDlgItemStr(hDlg, IDC_TXT_SVPATH, path);
105 :    //  コンボボックスサブクラス化
106 :    AjcSbcComboBoxEx(hCboPath, 40, MAX_CBOSTR - 1, AJCSBCF_IGNORWIDTH);
107 :    AjcSbcTipCtrl   (hCboPath, TRUE);
108 :    AjcSbcComboBoxEx(hCboWild, 40, MAX_CBOSTR - 1, AJCSBCF_IGNORWIDTH);
109 :    AjcSbcTipCtrl   (hCboWild, TRUE);
110 :
111 :    //  ラジオボタングループ化
112 :    AjcSbcRadioBtns(GetDlgItem(hDlg, IDC_GRP_TECAFT));
113 :    //  チェックボックス／ラジオボタン初期化
114 :    AjcSetDlgItemChk (hDlg, IDC_CHK_SJIS  , TRUE);
115 :    AjcSetDlgItemUInt(hDlg, IDC_GRP_TECAFT, AJCTEC_UTF_8);
116 :    //  フォルダドロップ許可
117 :    AjcEnableCtrlToDrop   (hTxtPath, AJCDROP_DIR);
118 :    AjcEnableCtrlToDrop   (hTxtWild, AJCDROP_DIR);
119 :    AjcEnableDlgItemToDrop(hDlg, IDC_TXT_SVPATH, AJCDROP_DIR);
120 :    //  全設定値読み出し
121 :    AjcLoadAllControlSettings (hDlg, TEXT("MySect"), AJCCTL_SELACT_ALL);
122 :    //  VTH 配置
123 :    GetClientRect(hDlg, &rect);
124 :    SendMessage(hDlg, WM_SIZE, SIZE_RESTORED, MAKELONG(rect.right - rect.left, rect.bottom - rect.top));
125 :
126 :    //  初期メッセージ
127 :    AjcVthPutText(hVth, TEXT(" 指定されたファイルのエンコード (Shift-JIS / UTF-8 / EUC-J / UTF-16) を変更します。¥n"), -1);
128 :    AjcVthPutText(hVth, TEXT("¥n"), -1);
129 :    AjcVthPutText(hVth, TEXT(" 「入力パス」下の「ワイルドカード」で指定したファイルエンコード変更対象となります。¥n"), -1);
130 :    AjcVthPutText(hVth, TEXT("¥n"), -1);
```

```
131 :     AjcVthPutText(hVth, TEXT(" 変更対象のファイルは「入力ファイル退避パス」下に、日時を付加したフォルダを作成し退避されます。
¥n"), -1);
132 :     AjcVthPutText(hVth, TEXT("¥n"), -1);
133 :     AjcVthPutText(hVth, TEXT(" 「ワイルドカード」はセミコロン( ; )か、スラッシュ( / )で区切って複数指定できます。¥n"), -1);
134 :     AjcVthPutText(hVth, TEXT("¥n"), -1);
135 :     AjcVthPutText(hVth, TEXT(" ログの内容は、「入力ファイル退避パス」下にも出力されます。( .logv ファイル ) ¥n"), -1);
136 :
137 :     return TRUE;
138 : }
139 : //----- WM_DESTROY --------------------------------------------------------------------------------------//
140 : AJC_DLGPROC(Main, WM_DESTROY     )
141 : {
142 :     //  ログファイルクローズ
143 :     if (hLogFile != NULL) {
144 :         AjcFClose(hLogFile);
145 :         hLogFile = NULL;
146 :     }
147 :     //  全設定値セーブ
148 :     AjcSaveAllControlSettings (hDlg);
149 :
150 :     PostQuitMessage(0);
151 :     return TRUE;
152 : }
153 : //----- WM_SIZING ---------------------------------------------------------------------------------------//
154 : AJC_DLGPROC(Main, WM_SIZING  )
155 : {
156 :     LPRECT      p = (LPRECT)lParam;
157 :     int         w = p->right - p->left;
158 :     int         h = p->bottom - p->top;
159 :
160 :     if (w < WndMinSize.cx) p->right = p->left + WndMinSize.cx;
161 :     if (h < WndMinSize.cy) p->bottom = p->top + WndMinSize.cy;
162 :
163 :     return TRUE;
164 : }
165 : //----- WM_SIZE -----------------------------------------------------------------------------------------//
166 : AJC_DLGPROC(Main, WM_SIZE    )
167 : {
168 :     RECT    rect;
169 :     int     w, h;
170 :     GetClientRect(hDlg, &rect);
171 :     w = rect.right - rect.left;
172 :     h = rect.bottom - rect.top;
173 :     SetWindowPos(hVth, NULL, 0, yVth, w - GetSystemMetrics(SM_CXVSCROLL), h - yVth - 5, SWP_NOMOVE);
174 :
175 :     return TRUE;
176 : }
177 : //----- IDC_GRP_TEC -------------------------------------------------------------------------------------//
178 : AJC_DLGPROC(Main, IDC_GRP_TECAFT    )
179 : {
180 :     if (HIWORD(wParam) == AJCRBTN_SELECT) {
181 :         if (lParam == AJCTEC_UTF_8 || lParam == AJCTEC_UTF_16LE || lParam == AJCTEC_UTF_16BE) {
182 :             AjcEnableDlgItem(hDlg, IDC_CHK_BOM, TRUE);
183 :         }
184 :         else {
185 :             AjcEnableDlgItem(hDlg, IDC_CHK_BOM, FALSE);
186 :         }
187 :     }
188 :     return TRUE;
189 : }
190 : //----- IDC_CMD_STOP ------------------------------------------------------------------------------------//
191 : AJC_DLGPROC(Main, IDC_CMD_STOP  )
192 : {
193 :     if (fBusy) {
194 :         fStop = TRUE;
195 :     }
196 :     return TRUE;
197 : }
198 : //----- IDOK (開始)  ------------------------------------------------------------------------------------//
199 : AJC_DLGPROC(Main, IDOK )
200 : {
201 :     SYSTEMTIME  lt;
202 :     UT  last [MAX_PATH];
203 :     UT  tstr [128];
204 :     UT  fname[MAX_PATH];
205 :     UT  fext [MAX_PATH];
206 :
207 :     //  ソースパス設定
208 :     AjcGetCtrlStr(hTxtPath, szSrcPath, MAX_PATH);
209 :     PathRemoveBackslash(szSrcPath);
210 :
211 :     //  ソース退避パス設定
212 :     AjcGetDlgItemStr(hDlg, IDC_TXT_SVPATH, szSavPath, MAX_PATH);
213 :     PathRemoveBackslash(szSavPath);
214 :
215 :     //  対象ワイルドカード設定
216 :     AjcGetCtrlStr(hTxtWild, szWild, AJCTSIZE(szWild));
217 :
218 :     //  テキストエンコード設定
219 :     TecTo = (EAJCTEC)AjcGetDlgItemUInt(hDlg, IDC_GRP_TECAFT);   //  変更先エンコード
```

```
220 :
221 :     //  ボタングレー化
222 :     AjcEnableDlgItem(hDlg, IDOK        , FALSE);
223 :     AjcEnableDlgItem(hDlg, IDCANCEL    , FALSE);
224 :     AjcEnableDlgItem(hDlg, IDC_CMD_STOP, TRUE);
225 :
226 :     //  フラグ初期化
227 :     fBusy = TRUE;       //  エンコード変更処理中
228 :     fStop = FALSE;      //  中止フラグ
229 :     fCan  = FALSE;      //  キャンセルフラグ
230 :     Count = 0;          //  ファイルカウンタ
231 :
232 :     do {
233 :         //  ソースパスチェック
234 :         if (szSrcPath[0] == 0 || !AjcPathIsDirectory(szSrcPath)) {
235 :             UT  txt[MAX_PATH + 128];
236 :             AjcSnPrintF(txt, AJCTSIZE(txt), TEXT("ソースフォルダが見つかりません(%s)¥n"), szSrcPath);
237 :             MessageBox(hDlg, txt, AppName, MB_ICONERROR);
238 :             goto idok_end;
239 :         }
240 :         //  ソース退避パスチェック
241 :         if (szSavPath[0] == 0 || !AjcPathIsDirectory(szSavPath)) {
242 :             UT  txt[MAX_PATH + 128];
243 :             AjcSnPrintF(txt, AJCTSIZE(txt), TEXT("ソース退避フォルダが見つかりません(%s)¥n"), szSavPath);
244 :             MessageBox(hDlg, txt, AppName, MB_ICONERROR);
245 :             goto idok_end;
246 :         }
247 :         //  コンボボックスの現表示パスを最新項目に設定
248 :         AjcSbcSetMostNew(hCboPath);
249 :         AjcSbcSetMostNew(hCboWild);
250 :
251 :         //  時刻文字列作成
252 :         GetLocalTime(&lt);
253 :         AjcSnPrintF(tstr, AJCTSIZE(tstr), TEXT("(%04d-%02d-%02d_%02d-%02d-%02d-%03d)"),
254 :                                   lt.wYear, lt.wMonth, lt.wDay, lt.wHour, lt.wMinute, lt.wSecond, lt.wMilliseconds);
255 :
256 :         //  ソースフォルダの最終パス名抜出
257 :         MAjcSplitPath(szSrcPath, NULL, NULL, fname, fext);
258 :         MAjcMakePath (last     , NULL, NULL, fname, fext);
259 :
260 :         //  ログファイルパス作成
261 :         AjcStrCpy(szLogFile, MAX_PATH, szSavPath);
262 :         AjcPathCat(szLogFile, last, MAX_PATH);
263 :         MAjcStrCat(szLogFile, MAX_PATH, tstr);
264 :         MAjcStrCat(szLogFile, MAX_PATH, TEXT(".logv"));
265 :
266 :         //  ログファイル生成 (エラー発生しても続行する)
267 :         if (!(hLogFile = AjcFCreate(szLogFile, AJCTEC_UTF_8, TRUE))) {
268 :             LogPrintF(TEXT("¥x1B[31m"));
269 :             LogPrintF(TEXT("¥t ログファイルの作成を失敗しました(%s)¥n"), szLogFile);
270 :             LogPrintF(TEXT("¥x1B[0m"));
271 :         }
272 :
273 :         //  開始ログ
274 :         LogPrintF(TEXT("¥x1B[37;40m"));
275 :         LogPrintF(TEXT("処理を開始します - %s%s¥n"), last, tstr);
276 :         LogPrintF(TEXT("¥x1B[0m"));
277 :
278 :         //  ソース退避ディレクトリ作成
279 :         {
280 :             //  ソース退避ディレクトリ名作成
281 :             MAjcStrCpy(szSavTime, MAX_PATH, szSavPath);
282 :             AjcPathCat(szSavTime, last, MAX_PATH);
283 :             MAjcStrCat(szSavTime, MAX_PATH, tstr);
284 :             //  ディレクトリ作成
285 :             if (!CreateDirectory(szSavTime, NULL)) {
286 :                 LogPrintF(TEXT("¥x1B[31m"));
287 :                 LogPrintF(TEXT("¥t ソース退避ディレクトリの作成を失敗しました。(%s)。¥n"), szSavTime);
288 :                 LogPrintF(TEXT("¥x1B[0m"));
289 :                 break;
290 :             }
291 :         }
292 :         //  ソース退避処理
293 :         {   UT  wild[MAX_PATH];
294 :             LogPrintF(TEXT("●ソースファイルを退避します¥n"));
295 :             MAjcStrCpy(wild, MAX_PATH, szSrcPath);
296 :             AjcPathCat(wild, TEXT("*.*"), MAX_PATH);
297 :             Count = 0;
298 :             SaveSource(hDlg, wild);
299 :         }
300 :
301 :         //  エンコード変更処理
302 :         {   UT  wild[MAX_PATH];
303 :             LogPrintF(TEXT("●エンコード変換を実行します¥n"));
304 :             MAjcStrCpy(wild, MAX_PATH, szSrcPath);
305 :             AjcPathCat(wild, TEXT("*.*"), MAX_PATH);
306 :             Count = nOK = nNG = nNC = 0;
307 :             ChangeTec(hDlg, wild);
308 :         }
309 :
```

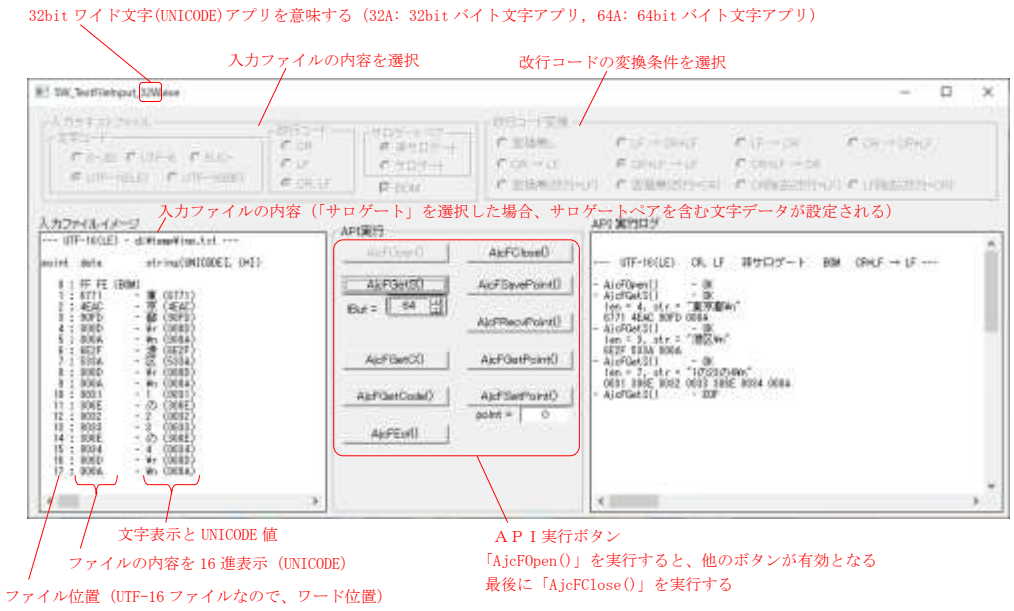
```
310 :     } while(0);
311 :
312 :     //  キャンセル押下ならば、終了
313 :     if (fCan) {
314 :         LogPrintF(TEXT("¥x1B[31m"));
315 :         LogPrintF(TEXT("¥t キャンセルされました。処理を終了します。¥n"));
316 :         LogPrintF(TEXT("¥x1B[0m"));
317 :         DestroyWindow(hDlg);
318 :     }
319 :
320 :     //  中止ボタン押下
321 :     if (fStop) {
322 :         LogPrintF(TEXT("¥x1B[31m"));
323 :         LogPrintF(TEXT("¥t 処理を中止しました。¥n"));
324 :         LogPrintF(TEXT("¥x1B[0m"));
325 :     }
326 :     else {
327 :         //  終了ログ
328 :         LogPrintF(TEXT("¥x1B[34m"));
329 :         LogPrintF(TEXT("処理を終了しました - %s%s ( OK = %d (No Change = %d), NG = %d )¥n"), last, tstr, nOK, nNC, nNG);
330 :         LogPrintF(TEXT("¥x1B[0m"));
331 :     }
332 : idok_end:;
333 :
334 :     //  フラグ解除
335 :     fBusy = FALSE;
336 :     fStop = FALSE;
337 :     fCan  = FALSE;
338 :
339 :     //  ボタングレー化解除
340 :     AjcEnableDlgItem(hDlg, IDOK        , TRUE);
341 :     AjcEnableDlgItem(hDlg, IDCANCEL    , TRUE);
342 :     AjcEnableDlgItem(hDlg, IDC_CMD_STOP, FALSE);
343 :
344 :     //  ログファイルクローズ
345 :     if (hLogFile != NULL) {
346 :         AjcFClose(hLogFile);
347 :         hLogFile = NULL;
348 :     }
349 :     return TRUE;
350 : }
351 : //----- IDCANCEL ------------------------------------------------------------------------------------------------//
352 : AJC_DLGPROC(Main, IDCANCEL)
353 : {
354 :     if (fBusy) {
355 :         fStop = TRUE;
356 :         fCan  = TRUE;
357 :     }
358 :     else {
359 :         DestroyWindow(hDlg);
360 :     }
361 :     return TRUE;
362 : }
363 : //---------------------------------------------------------------------------------------------------------------//
364 : AJC_DLGMAP_DEF(Main)
365 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG  )
366 :     AJC_DLGMAP_MSG(Main, WM_DESTROY     )
367 :     AJC_DLGMAP_MSG(Main, WM_SIZING      )
368 :     AJC_DLGMAP_MSG(Main, WM_SIZE        )
369 :     AJC_DLGMAP_CMD(Main, IDC_GRP_TECAFT )
370 :     AJC_DLGMAP_CMD(Main, IDC_CMD_STOP   )
371 :     AJC_DLGMAP_CMD(Main, IDOK           )
372 :     AJC_DLGMAP_CMD(Main, IDCANCEL       )
373 : AJC_DLGMAP_END
374 : //---------------------------------------------------------------------------------------------------------------//
375 : //  ソースフォルダセーブ処理                                                                                     //
376 : //---------------------------------------------------------------------------------------------------------------//
377 : static  BOOL    SaveSource(HWND hDlg, C_UTP pWild)
378 : {
379 :     HANDLE              hFind = NULL;
380 :     WIN32_FIND_DATA     fd;
381 :     HAJCFILE            hInp = NULL;
382 :     HAJCFILE            hOut = NULL;
383 :     UT                  drv[MAX_PATH];
384 :     UT                  dir[MAX_PATH];
385 :
386 :     AjcDoEvent();
387 :
388 :     if (!fStop && !fCan) {
389 :         if ((hFind = FindFirstFile(pWild, &fd)) != INVALID_HANDLE_VALUE) {
390 :             do {
391 :                 AjcDoEvent();
392 :                 //  ファイル
393 :                 if (!(fd.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)) {
394 :                     UT  path_inp[MAX_PATH];
395 :                     UT  path_out[MAX_PATH];
396 :                     UT  path_dir[MAX_PATH];
397 :                     //  ファイルパス名作成
398 :                     MAjcSplitPath(pWild   , drv, dir, NULL, NULL);
399 :                     MAjcMakePath (path_inp, drv, dir, NULL, NULL);
```

```
400 :                     AjcPathCat   (path_inp, fd.cFileName, MAX_PATH);
401 :                     //  ワイルドカード一致
402 :                     if (AjcPathMatchSpec(path_inp, szWild)) {
403 :                         UI  stl;
404 :                         //  転送先ファイルパス設定
405 :                         stl = (UI)MAjcStrLen(szSrcPath);
406 :                         MAjcStrCpy(path_out, MAX_PATH, szSavTime);
407 :                         AjcPathCat(path_out, &path_inp[stl], MAX_PATH);
408 :                         //  転送先ディレクトリ作成
409 :                         MAjcSplitPath(path_out, drv, dir, NULL, NULL);
410 :                         MAjcMakePath (path_dir, drv, dir, NULL, NULL);
411 :                         AjcCreateDirectoryStruct(path_dir);
412 :                         //  ファイルコピー
413 :                         if (CopyFile(path_inp, path_out, FALSE)) {
414 :                             LogPrintF(TEXT("¥t%5d : Save OK - %s¥n"), ++Count, path_out);
415 :                         }
416 :                         else {
417 :                             LogPrintF(TEXT("¥x1B[31m"));
418 :                             LogPrintF(TEXT("¥t%5d : Save NG - %s¥n"), ++Count, path_out);
419 :                             LogPrintF(TEXT("¥x1B[0m"));
420 :                         }
421 :                     }
422 :                 }
423 :                 //  ディレクトリならば、サブディレクトリ検索（再帰呼び出し）
424 :                 else {
425 :                     if (AjcGetDlgItemChk(hDlg, IDC_CHK_SUBDIR)) {
426 :                         UT  wild[MAX_PATH];
427 :                         if (MAjcStrCmp(fd.cFileName, TEXT(".")) != 0 && MAjcStrCmp(fd.cFileName, TEXT("..")) != 0) {
428 :                             MAjcSplitPath(pWild, drv, dir, NULL, NULL);
429 :                             MAjcMakePath (wild , drv, dir, NULL, NULL);
430 :                             AjcPathCat(wild, fd.cFileName, MAX_PATH);
431 :                             AjcPathCat(wild, TEXT("*.*"), MAX_PATH);
432 :                             //  サブディレクトリ検索（再帰呼び出し）
433 :                             SaveSource(hDlg, wild);
434 :                         }
435 :                     }
436 :                 }
437 :             } while (FindNextFile(hFind, &fd));
438 :             //  ファイル検索クローズ
439 :             FindClose(hFind);
440 :         }
441 :     }
442 :     return (!fStop && !fCan);   //  TRUE:完了
443 : }
444 : //-----------------------------------------------------------------------------------------------------------------//
445 : //  エンコード変更処理                                                                                             //
446 : //-----------------------------------------------------------------------------------------------------------------//
447 : #define     UTC_TIME    FALSE
448 : static  BOOL    ChangeTec(HWND hDlg, C_UTP pWild)
449 : {
450 :     HANDLE              hFind = NULL;   //  検索ハンドル
451 :     WIN32_FIND_DATA     fd;             //  検索したファイル情報
452 :     EAJCTEC             tec;            //  入力ファイルのエンコード種別
453 :     BOOL                bom;            //  入力ファイルのＢＯＭ有無
454 :     HAJCFILE            hInp = NULL;    //  入力ファイルハンドル
455 :     HAJCFILE            hOut = NULL;    //  出力ファイルハンドル
456 :     UI                  att;            //  入力ファイル属性
457 :     AJCFTIMES           ftm;            //  入力ファイルタイムスタンプ
458 :     UT                  buf[4096];      //  ファイル入力バッファ
459 :     UT                  drv[MAX_PATH];  //  パス編集ワーク
460 :     UT                  dir[MAX_PATH];  //  ・
461 :
462 :     AjcDoEvent();
463 :
464 :     if (!fStop && !fCan) {
465 :         if ((hFind = FindFirstFile(pWild, &fd)) != INVALID_HANDLE_VALUE) {
466 :             do {
467 :                 AjcDoEvent();
468 :                 //  ファイルならば、エンコード変換処理
469 :                 if (!(fd.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)) {
470 :                     UT  path_inp[MAX_PATH];
471 :                     UT  path_out[MAX_PATH];
472 :                     //  ファイルパス名作成
473 :                     MAjcSplitPath(pWild   , drv, dir, NULL, NULL);
474 :                     MAjcMakePath (path_inp, drv, dir, NULL, NULL);
475 :                     AjcPathCat(path_inp, fd.cFileName, MAX_PATH);
476 :                     //  ワイルドカード一致（エンコード変換対象ファイル）
477 :                     if (AjcPathMatchSpec(path_inp, szWild)) {
478 :                         BOOL    fErr = FALSE;   //  エラー発生フラグ
479 :                         BOOL    fOut = FALSE;   //  出力ファイル生成済フラグ
480 :                         //  入力ファイルオープン
481 :                         if (hInp = AjcFOpen(path_inp, AJCTEC_AUTO)) {
482 :                             //  入力ファイルエンコード取得
483 :                             tec = AjcFGetTec(hInp);
484 :                             bom = AjcFGetBOM(hInp);
485 :                             //  入力対象エンコード
486 :                             if (ChkInpTec(hDlg, tec, bom)) {
487 :                                 BOOL    bom_out = AjcGetDlgItemChk(hDlg, IDC_CHK_BOM);
488 :                                 //  出力ファイルパス作成
489 :                                 GetTempPath(MAX_PATH, path_out);
```

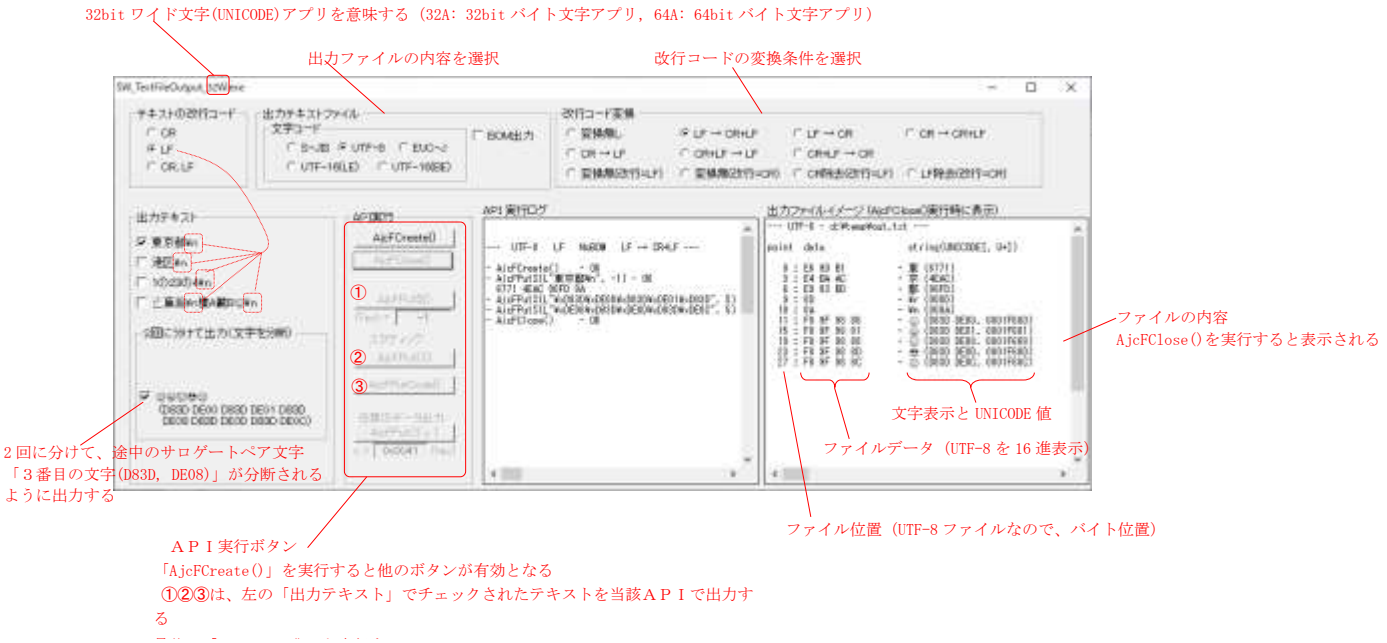
```
490 :                               AjcPathCat(path_out, TEXT("Temp.tmp"), MAX_PATH);
491 :                               //  出力ファイル生成
492 :                               if (hOut = AjcFCreate(path_out, TecTo, bom_out)) {
493 :                                   fOut = TRUE;    //  出力ファイル生成済
494 :                                   LogPrintF(TEXT("¥t%5d : %s -> %s - %s"), ++Count, TecName(tec, bom),
495 :                                                                                     TecName(TecTo, bom_out), path_inp);
496 :                                   //  ファイル入出力ループ
497 :                                   while (AjcFGetS(hInp, buf, AJCTSIZE(buf))) {
498 :                                       if (!AjcFPutS(hOut, buf, -1)) {
499 :                                           fErr = TRUE;
500 :                                           break;
501 :                                       }
502 :                                   }
503 :                                   //  出力ファイルクローズ
504 :                                   AjcFClose(hOut);
505 :                                   hOut = NULL;
506 :                               }
507 :                               else {
508 :                                   LogPrintF(TEXT("¥x1B[31m"));
509 :                                   AjcVthPrintF(hVth, TEXT("¥t%5d : File Creation Error       - %s"), ++Count, path_inp);
510 :                                   LogPrintF(TEXT("¥x1B[0m"));
511 :                                   fErr = TRUE;
512 :                               }
513 :                           }
514 :                           //  入力対象外のエンコード
515 :                           else {
516 :                               LogPrintF(TEXT("¥t%5d : %s -> NoChange      - %s"), ++Count, TecName(tec, bom), path_inp);
517 :                               nNC++;
518 :                           }
519 :                           //  入力ファイルクローズ
520 :                           if (hInp != NULL) {
521 :                               AjcFClose(hInp);
522 :                               hInp = NULL;
523 :                           }
524 :                           //  出力ファイル成作済みならば、入力側へコピー（上書き）
525 :                           if (fOut) {         //  出力ファイル生成済み？
526 :                               if (!fErr) {    //  エラーなし？
527 :                                   //  タイムスタンプコピー
528 :                                   if (AjcGetFileTime(path_inp, &ftm, NULL, UTC_TIME) &&
529 :                                       AjcSetFileTime(path_out, &ftm, UTC_TIME)) {
530 :                                       //  出力ファイルの属性設定
531 :                                       if ((att = GetFileAttributes(path_inp) != INVALID_FILE_ATTRIBUTES) &&
532 :                                           (SetFileAttributes(path_out, (att & ~EXTRA_FILE_ATT)))) {
533 :                                           //  出力ファイルを入力側へコピー
534 :                                           SetFileAttributes(path_inp, FILE_ATTRIBUTE_NORMAL);
535 :                                           if (!CopyFile(path_out, path_inp, FALSE)) {
536 :                                               fErr = TRUE;
537 :                                           }
538 :                                       }
539 :                                       else fErr = TRUE;
540 :                                   }
541 :                                   else fErr = TRUE;
542 :                               }
543 :                               //  出力ファイル削除
544 :                               SetFileAttributes(path_out, FILE_ATTRIBUTE_NORMAL);
545 :                               DeleteFile(path_out);
546 :                           }
547 :                       }
548 :                       //  入力ファイルオープン失敗
549 :                       else {  //  !(hInp = AjcFOpen(path_inp, AJCTEC_AUTO)
550 :                           LogPrintF(TEXT("¥x1B[31m"));
551 :                           AjcVthPrintF(hVth, TEXT("¥t%5d : File Open Error          - %s"), ++Count, path_inp);
552 :                           LogPrintF(TEXT("¥x1B[0m"));
553 :                           fErr = TRUE;
554 :                       }
555 :                       //  結果ログ
556 :                       if (!fErr) {
557 :                           //  正常ログ
558 :                           LogPrintF(TEXT(" - OK¥n"));
559 :                           nOK++;
560 :                       }
561 :                       //  出力ファイル作成失敗ならば、エラーログ
562 :                       else {
563 :                           //  エラーログ
564 :                           LogPrintF(TEXT("¥x1B[31m"));
565 :                           LogPrintF(TEXT(" - NG¥n"));
566 :                           LogPrintF(TEXT("¥x1B[0m"));
567 :                           nNG++;
568 :                       }
569 :                   }
570 :               }
571 :               //  ディレクトリならば、サブディレクトリ検索
572 :               else {
573 :                   if (AjcGetDlgItemChk(hDlg, IDC_CHK_SUBDIR)) {
574 :                       UT  wild[MAX_PATH];
575 :                       if (MAjcStrCmp(fd.cFileName, TEXT(".")) != 0 && MAjcStrCmp(fd.cFileName, TEXT("..")) != 0) {
576 :                           MAjcSplitPath(pWild, drv, dir, NULL, NULL);
577 :                           MAjcMakePath (wild , drv, dir, NULL, NULL);
578 :                           AjcPathCat(wild, fd.cFileName, MAX_PATH);
579 :                           AjcPathCat(wild, TEXT("*.*"), MAX_PATH);
```

```
580 :                             //  サブディレクトリ検索（再帰呼び出し）
581 :                             ChangeTec(hDlg, wild);
582 :                         }
583 :                     }
584 :                 }
585 :             } while (FindNextFile(hFind, &fd));
586 :             //  ファイル検索クローズ
587 :             FindClose(hFind);
588 :         }
589 :     }
590 :     return (!fStop && !fCan);   //  TRUE:完了
591 : }
592 : //-------------------------------------------------------------------------------------------------------------------//
593 : //  入力ファイルエンコードチェック                                                                                   //
594 : //-------------------------------------------------------------------------------------------------------------------//
595 : static  BOOL    ChkInpTec(HWND hDlg, EAJCTEC tec, BOOL bom)
596 : {
597 :     BOOL    rc = FALSE;
598 :     BOOL    fChkSJis    = AjcGetDlgItemChk(hDlg, IDC_CHK_SJIS);
599 :     BOOL    fChkUtf8    = AjcGetDlgItemChk(hDlg, IDC_CHK_UTF8);
600 :     BOOL    fChkEucJ    = AjcGetDlgItemChk(hDlg, IDC_CHK_EUCJ);
601 :     BOOL    fChkUtf16LE = AjcGetDlgItemChk(hDlg, IDC_CHK_UTF16LE);
602 :     BOOL    fChkUtf16BE = AjcGetDlgItemChk(hDlg, IDC_CHK_UTF16BE);
603 :     EAJCTEC OutTec      = (EAJCTEC)AjcGetDlgItemUInt(hDlg, IDC_GRP_TECOUT);
604 :     BOOL    OutBom      = AjcGetDlgItemChk(hDlg, IDC_CHK_BOM);
605 :
606 :     switch (tec) {
607 :         case AJCTEC_MBC:        rc = (fChkSJis    && (OutTec != AJCTEC_MBC));                       break;
608 :         case AJCTEC_UTF_8:      rc = (fChkUtf8    && (OutTec != AJCTEC_UTF_8    || OutBom != bom)); break;
609 :         case AJCTEC_EUC_J:      rc = (fChkEucJ    && (OutTec != AJCTEC_EUC_J));                     break;
610 :         case AJCTEC_UTF_16LE:   rc = (fChkUtf16LE && (OutTec != AJCTEC_UTF_16LE || OutBom != bom)); break;
611 :         case AJCTEC_UTF_16BE:   rc = (fChkUtf16BE && (OutTec != AJCTEC_UTF_16BE || OutBom != bom)); break;
612 :     }
613 :     return rc;
614 : }
615 : //-------------------------------------------------------------------------------------------------------------------//
616 : //  ファイルエンコード名取得                                                                                         //
617 : //-------------------------------------------------------------------------------------------------------------------//
618 : static  C_UTP   TecName(EAJCTEC tec, BOOL bom)
619 : {
620 :     C_UTP   rc;
621 :
622 :     switch (tec) {
623 :         case AJCTEC_MBC:        rc =        TEXT("Shift JIS    "); break;      //  マルチバイト
624 :         case AJCTEC_UTF_8:      rc = bom ?  TEXT("UTF-8(BOM)   "):             //  ＵＴＦ－８
625 :                                             TEXT("UTF-8        "); break;
626 :         case AJCTEC_EUC_J:      rc =        TEXT("EUC-J        "); break;      //  ＥＵＣ（日本語）
627 :         case AJCTEC_UTF_16LE:   rc = bom ?  TEXT("UTF-16LE(BOM)"):             //  ＵＴＦ－１６ＬＥ
628 :                                             TEXT("UTF-16LE     "); break;
629 :         case AJCTEC_UTF_16BE:   rc = bom ?  TEXT("UTF-16BE(BOM)"):             //  ＵＴＦ－１６ＢＥ
630 :                                             TEXT("UTF-16BE     "); break;
631 :         default:                rc =        TEXT("UnKnown      "); break;      //  エラー
632 :     }
633 :     return rc;
634 : }
635 : //-------------------------------------------------------------------------------------------------------------------//
636 : //  ログ出力                                                                                                         //
637 : //-------------------------------------------------------------------------------------------------------------------//
638 : static  VO      LogPrintF(C_UTP pFmt, ...)
639 : {
640 :     va_list     vls;
641 :     UT          buf[1024];
642 :
643 :     va_start(vls, pFmt);
644 :     AjcVSnPrintF(buf, 1024, pFmt, vls);
645 :     buf[1023] = 0;
646 :     va_end  (vls);
647 :     //  初回ログならば、画面クリアー
648 :     if (fLogFirst) {
649 :         AjcVthClear(hVth);
650 :         fLogFirst = FALSE;
651 :     }
652 :     //  ログファイル出力
653 :     if (hLogFile != NULL) {
654 :         AjcFPutS(hLogFile, buf, -1);
655 :     }
656 :     //  ログ画面出力
657 :     AjcVthPutText(hVth, buf, -1);
658 : }
659 :
```

31.2.2. SW_TextFileInput （テキストファイルの入力テスト）

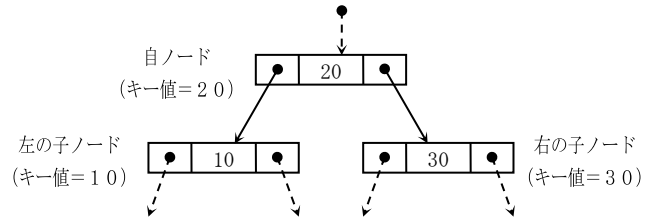


31.2.3. SW_TextFileOutput（テキストファイルの出力テスト）

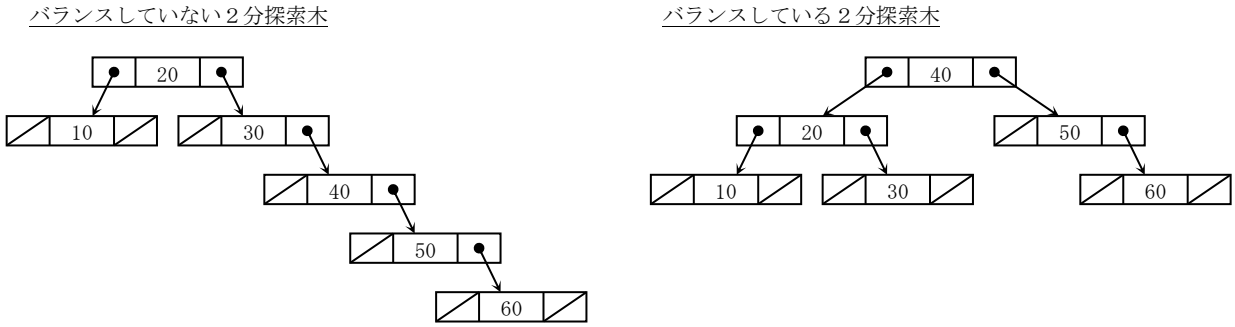


32. 平衡２分木（ＡＶＬ木）

２分探索木の各ノードは、２つの子ノード（左の子と右の子）への位置情報（ポインタ）を持ち、左の子は自ノードより低い値のキーを持ち、右の子は自ノードより高い値のキーを持ちます。



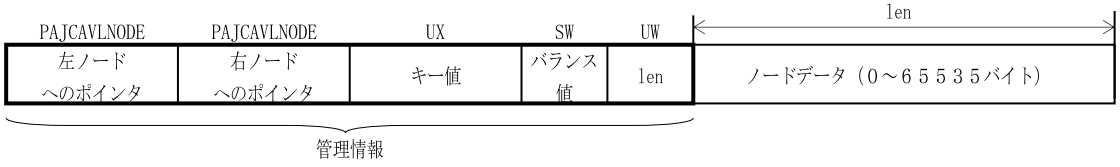
平衡２分木とは、さらに、木の高さがバランスした２分探索木構造を意味します。



平衡した２分探索木構造では、特定のノードを検索するのに必要な探索回数は、木の高さと同じであり、ノードの個数をｎとした場合、ｌｏｇ$_2$（ｎ）＋１回の探索で見つけることができます。たとえば、ノード数が最大６５５３５ヶである場合、１６回以内の探索で見つけ出すことができます。

本ライブラリでは、ランダムに与えられるキー値から、平衡した２分探索木（ＡＶＬ木）を構築し、ノードの検索や、シーケンシャルなノード読み出しを行うとができます。

各ノードの先頭には、以下のような管理情報が付加されます。



※「バランス値」は、自ノードを頂点とした、右側ノードの高さから左側ノードの高さを減算した値（ -1 ～ +1 ）を意味します。

キー値は、ＵＸタイプ（ポインタを格納可能なビット数の符号なし整数）として比較しますが、キー値の比較用コールバック関数を指定することにより、キー値の比較方法をカスタマイズすることもできます。例えば、キー値を文字列へのポインタとして、コールバックにより文字列比較を行う、といったようなことができます。

文字列ノード

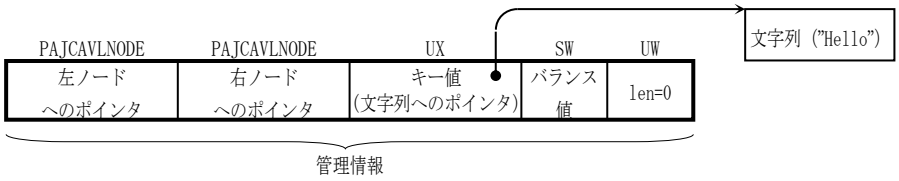
ノードデータを文字列として扱うことが可能です。
文字列のポインタをキー値とすることにより、見かけ上文字列をキーとした（文字列の昇順／降順に構成された）２分木を作成することができます。　この場合、文字列比較用のコールバック（下記例の「cbComp」）を指定する必要があります。

```
//  文字列比較用コールバック
static int  CALLBACK cbComp(UX key1, UX key2, UX cbp)
{
    return strcmp((C_BCP)key1, (C_BCP)key2);
}
        ・・・・・

    HAJCAVL  hAvl = NULL;
    BCP      pStr = NULL;

    //  ２分木インスタンス生成
    hAvl = AjcAvlCreate(0, cbComp, NULL);
        ・・・・・
    //  文字列ノード挿入（文字列へのポインタをキーとする）
    pStr =(BCP) malloc(6);
    strcpy(pStr, "Hello");
    AjcAvlInsNode(hAvl, (UX)pStr, NULL, 0);
```

挿入したノードは、以下のような構成となります。(AvlInsNode()でノードデータのアドレス＝NULL としているので、ノードデータ無し)



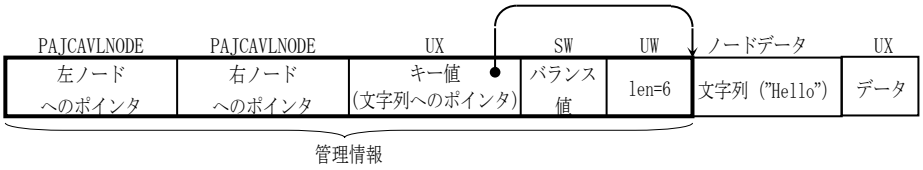
あるいは、文字列ノード作成用のＡＰＩ（関数名に「StrNode」を含むＡＰＩ）を使用し、ノードデータ部分に文字列を格納したノードを作成することができます。この場合も、文字列比較用のコールバック（下記例の「cbComp」）を指定する必要があります。

```
//  文字列比較用コールバック
static int  CALLBACK cbComp(UX key1, UX key2, UX cbp)
{
    return strcmp((C_BCP)key1, (C_BCP)key2);
}
        ・・・・・

    HAJCAVL  hAvl = NULL;

    //  ２分木インスタンス生成
    hAvl = AjcAvlCreate(0, cbComp, NULL);
        ・・・・・
    //  文字列ノード挿入（文字列へのポインタをキーとし、文字列をノードデータとする）
    AjcAvlInsStrNode(hAvl, "Hello");
```

文字列ノードを作成した場合、ノードは以下のような構成となります。（ノードデータとして文字列が格納される）



32.1. サポートＡＰＩ

平衡２分木のサポートＡＰＩ一覧を以下に示します。

#	機能	関数名	備考
1	インスタンス生成	AjcAvlCreate	
2	インスタンス消去	AjcAvlDelete	
3	コールバックパラメタの設定	AjcAvlSetCbp	
4	ノード挿入	AjcAvlInsNode	
5	データノード挿入	AjcAvlInsDataNode	ノードデータのアドレスをキーとする
6	ノード取得	AjcAvlGetNodeByKey	
7	ノードアドレス取得	AjcAvlGetNodePtr	
8	ノードのキー値取得	AjcAvlGetNodeKey	
9	先頭ノード取得	AjcAvlGetTopNode	
10	最終ノード取得	AjcAvlGetLastNode	
11	ノード置換	AjcAvlRepNode	
12	ノード挿入／置換	AjcAvlInsOrRepNode	
13	ノード削除	AjcAvlDelNode	
14	全ノードを削除	AjcAvlDelAllNodes	
15	文字列ノード挿入	AjcAvlInsStrNode	文字列をノードデータとし、文字列のアドレスをキーとする
16	文字列ノード取得	AjcAvlGetStrNode	
17	文字列ノード挿入／取得	AjcAvlInsOrGetStrNode	
18	文字列ノード削除	AjcAvlDelStrNode	
19	文字列ノードにデータを関連付ける	AjcAvlSetStrNodeData	
20	文字列ノードに関連付けたデータを取得	AjcAvlGetStrNodeData	
21	ノード数取得	AjcAvlGetCount	
22	全ノードの列挙	AjcAvlEnumNodes AjcAvlEnumNodesEx	- インスタンス生成時のコールバックパラメタを使用 - コールバックパラメタを独自に指定
23	全ノードへのポインタ配列生成	AjcAvlCreatePtrArr	
24	全ノードへのポインタ配列解放	AjcAvlReleasePtrArr	
25	マルチスレッドの許可／禁止	AjcAvlEnableMultiThread	

32.1.1. インスタンス生成（AjcAvlCreate）

形　式 ： HAJCAVL　AjcAvlCreate (UX cbp,
　　int (CALLBACK *cbComp　)(UX key1, UX key2, UX cbp),
　　VO　(CALLBACK *cbRemove)(UX key, VOP pNodeData, UI len, UI nest, UX cbp));

引　数 ： cbp　　- コールバックパラメタ
　　cbComp　- キー比較のカスタマイズ用コールバック関数（不要時はＮＵＬＬ）
　　cbRemove - 削除されるノード通知用コールバック関数　（不要時はＮＵＬＬ）

説　明 ： 平衡２分木機能のインスタンスを生成し、初期化します。
　　平衡２分木機能（ノードの挿入や削除等）を実行する場合は、本関数により返されたインスタンスハンドルを指定します。

　　cbComp は、キー値の比較方法をカスタマイズする場合の、キー値比較用コールバック関数を指定します。
　　cbComp に、NULL を指定した場合は、キー値をＵＸタイプとして比較します。

　　cbRemove は、削除されるノード通知用コールバック関数を指定します。
　　このコールバック関数は、AjcAvlDelNode, AjcAvlDelAllNodes、あるいは AjcAvlDelete 関数の実行中に、ノードを削除（開放）する直前にコールバックされます。
　　削除されるノードを通知する必要が無い場合は、NULL を指定します。

戻り値 ： ≠NULL - 成功　（インスタンスハンドルを返します）
　　＝NULL - 失敗

コールバック： コールバック関数の仕様は、以下のとおりです。

cbComp（キー値比較）

形　式 ： int CALLBACK ***cbComp***(UX key1, UX key2, UX cbp)；

引　数 ： key1 － 比較するキー値１
key2 － 比較するキー値２
cbp － コールバックパラメタ

説　明 ： key1 と key2 を比較します。
key1 と key2 は、ノード作成時に（AjcAvlInsNode 関数の key 引数で）ユーザが指定した値であり、比較の方法は任意です。

戻り値 ： ＞０ ： key1 ＞ key2
＝０ ： key1 と key2 は等しい
＜０ ： key1 ＜ key2

cbRemove（削除ノード通知）

形　式 ： VO CALLBACK ***cbRemove***(UX key, VOP pNodeData, UI len, UI nest, UX cbp)；

引　数 ： key － キー値
pNodeData － 削除するノードデータのアドレス（ノードに付与されたデータの先頭アドレス）
len － 削除するノードデータのバイト数（ノードに付与されたデータのバイト数）
nest － 削除するノードデータのネスト位置
（ルートノードからの自ノードまでの木の高さ，ルートノードの場合は１）
cbp － コールバックパラメタ

説　明 ： このコールバック関数の実行直後に、当該ノードが削除されることを通知します。
AjcAvlRepNode()や AjcAvlInsOrRepNode()でノードが置換される場合も通知します。
必要ならば、key と pNodeData で示される、キー値，ノードデータをもとに、ローカルなリソースの解放等を行います。

戻り値 ： なし

32.1.2. インスタンス消去（AjcAvlDelete）

形　式 ： BOOL AjcAvlDelete (HAJCAVL hAvl)；

引　数 ： hAvl － インスタンスハンドル（AjcAvlCreate の戻り値）

説　明 ： 全ノードの削除（開放）を行い、AjcAvlCreate で動的に確保したリソースを解放します。

戻り値 ： TRUE － 成功
FALSE － 失敗

32.1.3. コールバックパラメタの設定（AjcAvlSetCbp）

形　式 ： VO AjcAvlSetCbp (HAJCAVL hAvl、UX cbp)；

引　数 ： hAvl － インスタンスハンドル（AjcAvlCreate の戻り値）
cbp － 設定するコールバックパラメタ

説　明 ： コールバックパラメタを設定します。(AjcAvlCreate()で設定したコールバックパラメタを変更します)

戻り値 ： TRUE － 成功
FALSE － 失敗

32.1.4. ノード挿入（AjcAvlInsNode）

形　式 ： BOOL　AjcAvlInsNode(HAJCAVL hAvl, UX key, C_VOP pNodeData, UI len);

引　数 ： hAvl　　- インスタンスハンドル
key　　- ノードに付与するキー値
pNodeData - ノードに付与するデータの先頭アドレス（ノードデータ無しの場合は NULL）
len　　- ノードに付与するデータのバイト数　　（ノードデータ無しの場合は０）

説　明 ： 平衡２分木にノードを追加し、指定されたキー値とデータのペアを付与します。
ノードを追加した後も、２分木の平衡は保たれます。
key で指定されたキー値と同じキー値のノードが既に存在する場合は、ノードを追加することはできません。

戻り値 ： TRUE　- 成功
FALSE - 失敗

32.1.5. データノード挿入（AjcAvlInsDataNode）

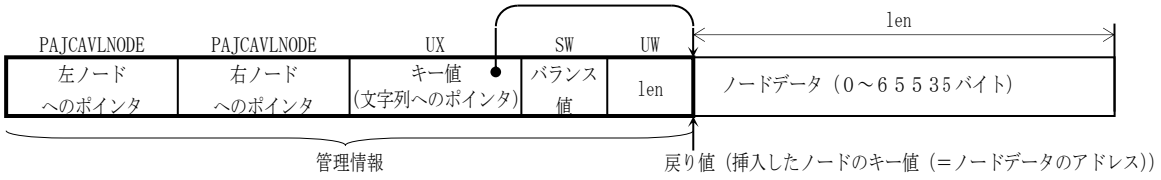
形　式 ： UX　AjcAvlInsDataNode(HAJCAVL hAvl, C_VOP pNodeData, UI len);

引　数 ： hAvl　　- インスタンスハンドル
pNodeData - ノードに付与するデータの先頭アドレス
len　　- ノードに付与するデータのバイト数（0～65535）

説　明 ： 平衡２分木にノードを追加します。
ノードのキー値は、作成されたノードデータのアドレス値となります。
ノードを追加した後も、２分木の平衡は保たれます。
ノードデータの最大長は、65535 バイトです。
ノードデータの最大長を超えた場合は、登録は行わずに０を返します。

戻り値 ： ≠０：成功（挿入したノードのキー値（＝ノードデータのアドレス））
＝０：失敗

備　考 ： 本ＡＰＩでは、以下のようなノードを作成します。



32.1.6. ノード取得（AjcAvlGetNodeByKey）

形　式 ： int　AjcAvlGetNode　　(HAJCAVL hAvl, UX key, VOP pBuf, UI lBuf);
int　AjcAvlGetNodeByKey(HAJCAVL hAvl, UX key, VOP pBuf, UI lBuf);

引　数 ： hAvl　　- インスタンスハンドル
key　　- 検索するノードのキー値
pBuf　　- 見つかったノードの付与データを格納するバッファのアドレス（不要時はＮＵＬＬ）
lBuf　　- 見つかったノードの付与データを格納するバッファのバイト数（pBuf=NULL の場合は不要）

説　明 ： 平衡２分木から、key で指定されたキー値のノードを検索し、見つかった場合は、pBuf で示されるバッファにノードデータをコピーします。

戻り値 ： ≠-1 - 指定キー値のノードが見つかった（当該ノードデータのバイト数）
＝-1 - 失敗（当該ノードなし）

32.1.7. ノードアドレス取得（AjcAvlGetNodePtr）

形　式 ： VOP　AjcAvlGetNodePtr(HAJCAVL hAvl, UX key, UIP pLen);

引　数 ： hAvl　　- インスタンスハンドル
key　　- 検索するノードのキー値
pLen　　- 見つかったノードデータのバイト数を格納するバッファのアドレス（不要時はＮＵＬＬ）

説　明 ： 平衡２分木から、key で指定されたキー値のノードを検索し、見つかった場合は、pLen で示されるバッファにノードデータのバイト数を格納し、ノードデータのアドレスを返します。

戻り値 ： ≠NULL - 成功（当該ノード・データへのポインタ）
＝NULL - 失敗（当該ノードなし）

32.1.8. ノードのキー値取得（AjcAvlGetNodeKey）

形　式 ： UX　AjcAvlGetNodeKey(HAJCAVL hAvl, C_VOP pNode);

引　数 ： hAvl　　- インスタンスハンドル
pNode　　- ノードデータのアドレス

説　明 ： ノードデータへのポインタから、ノードのキー値を取得します。

戻り値 ： ノードのキー値（エラーの場合は -1 を返しますが、-1 をキー値として使用している場合は区別できません）

32.1.9. 先頭ノード取得（AjcAvlGetTopNode）

形　式 ： int　AjcAvlGetTopNode(HAJCAVL hAvl, UXP pKey, VOP pBuf, UI lBuf);

引　数 ： hAvl　　- インスタンスハンドル
pKey　　- 先頭ノードのキー値を格納するバッファのアドレス（不要時はＮＵＬＬ）
pBuf　　- 見つかったノードデータを格納するバッファのアドレス（不要時はＮＵＬＬ）
lBuf　　- 見つかったノードデータを格納するバッファのバイト数（pBuf=NULL の場合は不要）

説　明 ： 平衡２分木から、pBuf で示されるバッファに、キー値が最小の先頭ノードデータをコピーします。

戻り値 ： ≠-1 - 先頭ノードが見つかった（先頭ノードデータのバイト数）
＝-1 - 失敗

32.1.10. 最終ノード取得（AjcAvlGetLastNode）

形　式 ： int　AjcAvlGetLastNode(HAJCAVL hAvl, UXP pKey, VOP pBuf, UI lBuf);

引　数 ： hAvl　　- インスタンスハンドル
pKey　　- 最終ノードのキー値を格納するバッファのアドレス（不要時はＮＵＬＬ）
pBuf　　- 見つかったノードデータを格納するバッファのアドレス（不要時はＮＵＬＬ）
lBuf　　- 見つかったノードデータを格納するバッファのバイト数（pBuf=NULL の場合は不要）

説　明 ： 平衡２分木から、pBuf で示されるバッファに、キー値が最大のノードデータをコピーします。

戻り値 ： ≠-1 - 最終ノードが見つかった（最終ノードデータのバイト数）
＝-1 - 失敗

32.1.11. ノード置換（AjcAvlRepNode）

形　式 ： BOOL　AjcAvlRepNode(HAJCAVL hAvl, UX key, C_VOP pNodeData, UI len);

引　数 ： hAvl　　- インスタンスハンドル
key　　- 置換するノードのキー値
pNodeData - 置換する新ノードデータのアドレス（ノードデータ無しの場合はNULL）
len　　- 置換する新ノードデータのバイト数（ノードデータ無しの場合は０）

説　明 ： key で指定されたノードのデータを置換します。
key で指定されたキー値と同じキー値のノードが見つかった場合、当該ノードデータを置換し TRUE を返します。
現存のノードとデータ長が同じ場合は、ノードデータを上書きします。
現存のノードとデータ長が異なる場合は、一旦ノードを解放し、新たにノード作成します。
key で指定されたキー値と同じキー値のノードが見つからない場合は、FALSE を返します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

32.1.12. ノード挿入／置換（AjcAvlInsOrRepNode）

形　式 ： BOOL　AjcAvlInsOrRepNode(HAJCAVL hAvl, UX key, C_VOP pNodeData, UI len);

引　数 ： hAvl　　- インスタンスハンドル
key　　- キー値
pNodeData - 置換するノードデータのアドレス（ノードデータ無しの場合はNULL）
len　　- 置換するノードデータのバイト数（ノードデータ無しの場合は０）

説　明 ： 平衡２分木に同一キーのノードがある場合は、「AjcAvlRepNode()」と同様に指定されたノードデータを書き換えます。
同一文字列のノードデータが無い場合は、「AjcAvlInsNode()」と同様に平衡２分木にノードを追加します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

32.1.13. ノード削除（AjcAvlDelNode）

形　式 ： BOOL　AjcAvlDelNode(HAVP hAvl, UX key);

引　数 ： hAvl　　- インスタンスハンドル
key　　- 削除するノードのキー値

説　明 ： 平衡２分木から key で指定されたキー値のノードを削除します。
ノードを削除した後も、２分木の平衡は保たれます。

戻り値 ： TRUE　- 成功
FALSE - 失敗

32.1.14. 全ノード削除（AjcAvlDelAllNodes）

形　式 ： BOOL　AjcAvlDelAllNodes(HAJCAVL hAvl);

引　数 ： hAvl - インスタンスハンドル

説　明 ： 平衡２分木から全てのノードを削除します。
この関数を実行すると、ノードが１つも無い状態になります。

戻り値 ： TRUE　- 成功
FALSE - 失敗

32.1.15. 文字列ノード挿入（AjcAvlInsStrNode）

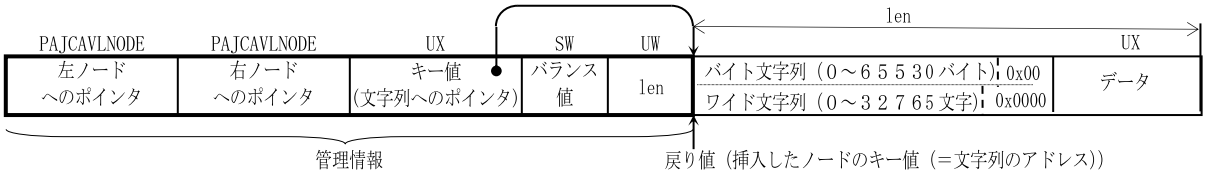
形　式　：　UX　AjcAvlInsStrNode(HAJCAVL hAvl, C_UTP pStr);

引　数　：　hAvl　　－ インスタンスハンドル
pStr　　－ ノードに付与するデータ（文字列）の先頭アドレス

説　明　：　平衡２分木にノードを追加し、指定された文字列をノードデータとして作成します。
ノードのキー値は、作成されたノードデータ（文字列）のアドレス値となります。
ノードを追加した後も、２分木の平衡は保たれます。
文字列の最大長は、バイトモード時で 65534 バイト，UNICODE 時で 32766 文字です。
文字列の最大長を超えた場合は、登録は行わずに０を返します。

戻り値　：　≠０：成功（挿入したノードのキー値（＝ノードデータ（文字列）のアドレス））
＝０：失敗

備　考　：　本ＡＰＩでは、以下のようなノードを作成します。



32.1.16. 文字列ノード取得（AjcAvlInsNode）

形　式　：　UX　AjcAvlInsNode (HAJCAVL hAvl, C_UTP pStr);

引　数　：　hAvl　　－ インスタンスハンドル
pStr　　－ キーデータ（文字列）の先頭アドレス

説　明　：　平衡２分木に同一文字列のノードデータがある場合は、当該ノードのキー値（文字列のアドレス）を返します。
同一文字列のノードデータが無い場合は、０を返します。

戻り値　：　≠０ － 成功（取得したノードのキー値（ノードデータ（文字列）のアドレス））
＝０ － 失敗

32.1.17. 文字列ノード挿入／取得（AjcAvlGetStrNode）

形　式　：　UX　AjcAvlGetStrNode(HAJCAVL hAvl, C_UTP pStr);

引　数　：　hAvl　　－ インスタンスハンドル
pStr　　－ キーデータ（文字列）の先頭アドレス

説　明　：　平衡２分木に同一文字列のノードデータがある場合は、当該ノードのキー値（文字列のアドレス）を返します。
同一文字列のノードデータが無い場合は、ノードを挿入し、当該ノードのキー値（文字列のアドレス）を返します。

戻り値　：　≠０ － 成功（ノードのキー値（ノードデータ（文字列）のアドレス））
＝０ － 失敗

32.1.18. 文字列ノード削除（AjcAvlDelStrNode）

形　式 ： BOOL　AjcAvlDelStrNode(HAJCAVL hAvl, C_UTP pStr);

引　数 ： hAvl　　- インスタンスハンドル
pStr　　- キーデータ（文字列）の先頭アドレス

説　明 ： 平衡２分木に同一文字列のノードデータがある場合は、ノードを消去し TRUE を返します。
同一文字列のノードデータが無い場合は、FALSE を返します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

32.1.19. 文字列ノードにデータを関連付ける（AjcAvlSetStrNodeData）

形　式 ： BOOL　AjcAvlSetStrNodeData(HAJCAVL hAvl, C_UTP pStr, UX data);

引　数 ： hAvl　　- インスタンスハンドル
pStr　　- キーデータ（文字列）の先頭アドレス
data　　- 関連付けるデータ

説　明 ： 文字列のノードデータにデータを関連付けます。

戻り値 ： TRUE　- 成功
FALSE - 失敗

32.1.20. 文字列ノードに関連付けたデータを取得（AjcAvlSetStrNodeData）

形　式 ： BOOL　AjcAvlGetStrNodeData (HAJCAVL hAvl, C_UTP pStr, UXP pData);

引　数 ： hAvl　　- インスタンスハンドル
pStr　　- キーデータ（文字列）の先頭アドレス
pData　　- 関連付けられているデータを格納するバッファ

説　明 ： 文字列のノードに関連付けられたデータを取得します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

32.1.21. ノード数取得（AjcAvlGetCount）

形　式 ： UI　AjcAvlGetCount (HAJCAVL hAvl);

引　数 ： hAvl - インスタンスハンドル

説　明 ： ノードの個数を取得します。

戻り値 ： ノードの個数（パラメタエラー時は、０を返す）

32.1.22. 全ノードの列挙（AjcAvlEnumNodes[Ex]）

形　式 ： UI AjcAvlEnumNodes(HAJCAVL hAvl,
BOOL (CALLBACK *cbNtcNode)(UX key, VOP pNodeData, UI len, UI nest, UX cbp),
BOOL fDownSeq);

UI AjcAvlEnumNodesEx(HAJCAVL hAvl, **UX cbp**,
BOOL (CALLBACK *cbNtcNode)(UX key, VOP pNodeData, UI len, UI nest, UX cbp),
BOOL fDownSeq);

引　数 ： hAvl　　- インスタンスハンドル
cbp　　- コールバックパラメタ
cbNtcNode - 読み出したノードの情報を通知するコールバック関数（不要時はＮＵＬＬ）
fDownSeq - ノードの読み出し順の指定（FALSE ： 昇順， TRUE ： 降順）

説　明 ： 平衡２分木から、キー値の昇順、あるいは、降順に、順次ノードを読み出します。
読み出したノードの情報は、cbNtcNode で示されるコールバック関数を呼び出すことにより通知します。
CbNtcNode に NULL を指定した場合は、単に、現在の総ノード数を返します。

戻り値 ： 現在の総ノード数（エラー時は０を返す）

コールバック： コールバック関数の仕様は、以下のとおりです。

cbNtcNode（ノード通知）

形　式 ： BOOL CALLBACK ***cbNtcNode*** (UX key, VOP pNodeData, UI len, UI nest, UX cbp)；

引　数 ： key　　- キー値
pNodeData - 読み出したノードデータのアドレス（ノードに付与されたデータの先頭アドレス）
len　　- 読み出したノードデータのバイト数（ノードに付与されたデータのバイト数）
nest　　- 読み出したノードデータのネスト位置（ルートノードからの自ノードまでの木の高さ，
ルートノードの場合は１）
cbp　　- コールバックパラメタ
AjcAvlEnumNodes()は、AjcAvlCreate()/AjcAvlSetCbp()で指定したコールバックパラメタ
AjcAvsEnumNodesEx()の場合は当該関数で指定したコールバックパラメタ

説　明 ： 順次、全てのノードを通知します。
FALSE を返すと、ノードの読み出しを中止します。

戻り値 ： TRUE　- ノード読み出し継続

32.1.23. 全ノードへのポインタ配列生成（AjcAvlCreatePtrArr）

形　式 ： PCAJCAVLPTR　AjcAvlCreatePtrArr (HAJCAVL hAvl, UIP pNum, BOOL fDownSeq)

引　数 ： hAvl　　- インスタンスハンドル
pNum　　- 配列要素数を格納するバッファのアドレス（不要時はＮＵＬＬ）
fDownSeq　- ポインタ配列順の指定（FALSE ： 昇順,　TRUE ： 降順）

説　明 ： 全ノードへのポインタ配列を生成します。
ポインタ配列は、ノード数＋１の要素が作成され、最終要素には NULL が格納されます。
生成した配列を使用した後は、AjcAvlReleasePtrArr ()により配列を解放してください。

戻り値 ： ≠NULL - 成功（全ノードへのポインタ配列の先頭アドレス）
＝NULL - 失敗

備　考 ： 本 API は、以下の構造体の配列の先頭アドレスを返します。

```
typedef struct {
    UX      key;                    /*  キー                */
    VOP     pNode;                  /*  ノードへのポインタ  */
    UI      len;                    /*  ノードのバイト数    */
} AJCAVLPTR, *PAJCAVLPTR;
typedef const AJCAVLPTR *PCAJCAVLPTR;
```

また、「MAJCAVLPTR」マクロにより、構造体名とポインタタイプを変更した、同型の構造体を定義できます。

```
  MAJCAVLPTR(MYSTRUCT, PTRTYPE)
+ typedef struct {
+    UX        key;                 /*  キー                */
+    PTRTYPE   pNode;               /*  ノードへのポインタ  */
+    UI        len;                 /*  ノードのバイト数    */
+ } MYSTRUCT, *PMYSTRUCT;
+ typedef const MYSTRUCT  *PCMYSTRUCT;
```

「MAJCAVLPTR」マクロを定義し、本ＡＰＩを呼び出す際に、生成した構造体でキャストできます。

```
  PCMYSTRUCT  p:
  p = (PCMYSTRUCT)AjcAvlCreatePtrArr(・・・);
```

32.1.24. 全ノードへのポインタ配列解放（AjcAvlReleasePtrArr）

形　式 ： BOOL AjcAvlReleasePtrArr (HAJCAVL hAvl, PCAJCAVLPTR pArr)

引　数 ： hAvl　　- インスタンスハンドル
pArr　　- 全ノードへのポインタ配列の先頭アドレス（AjcAvlCreatePtrArr()の戻り値）

説　明 ： AjcAvlCreatePtrArr()で生成した、全ノードへのポインタ配列を解放します。

戻り値 ： TRUE　- 成功
FALSE　- 失敗

32.1.25. マルチスレッドの許可／禁止（AjcAvlEnableMultiThread）

形　式　：　BOOL　AjcAvlEnableMultiThread(HAJCAVL hAvl, BOOL fEnable);

引　数　：　hAvl　　－ インスタンスハンドル
fEnable　－ TRUE　：複数のスレッドからのアクセスを許可
FALSE：複数のスレッドからのアクセスを禁止

説　明　：　fEnable=TRUE とした場合、複数のスレッドによる、平衡２分木へのノード追加／削除や、検索を可能にします。
この場合、各関数の入り口と出口で、クリティカルセクションによるスレッド間の排他制御を行います。
但し、次の関数は、（fEnable=TRUE としても）マルチスレッドでの排他制御を行いません。

・AjcAvlCreate（インスタンス生成）
・AjcAvlDelete（インスタンス消去）
・AjcAvlEnableMultiThread（本関数）

fEnable=FALSE（デフォルト）とした場合は、マルチスレッドでの排他制御を行いません。

戻り値　：　前回の許可／禁止状態

32.2. サンプルプログラム

32.2.1. SW_AvlTreeC（ＡＶＬツリー操作サンプル）

以下のサンプルプログラムは、コマンド入力により、２文木にノード挿入や削除等を行います。
ノードデータは、時刻文字列へのポインタです。
時刻文字列は動的なメモリに格納し、ノードの削除時に解放します。
起動時は初期ノードとしてキー＝50, 60, 70, 80, 90, 100 の６つのノードを挿入します。
コマンドの形式は以下の通りです。 ESC キーを押すとプログラムを終了します。

#	コマンド形式	内容
1	I nnn	ノード挿入, key=nnn
2	R nnn	ノード置換, key=nnn
3	X nnn	ノードの挿入／置換, key=nnn
4	D nnn	ノード削除, key=nnn
5	A	全ノードを破棄
6	P	全ノード表示




```
 1 : //
 2 : //  SW_AvlTreeC.c
 3 : //
 4 : #include    <AjrCstXX.h>
 5 : #include    <stdio.h>
 6 : #include    <conio.h>
 7 : #include    <time.h>
 8 : #include    <tchar.h>
 9 :
10 : //----- ワーク ----------------------------------------------------------------------------------------------//
11 : static  HAJCAVL     hAvl = NULL;
12 :
13 : //----- ノードデータ形式 ------------------------------------------------------------------------------------//
14 : typedef struct {
15 :     UTP     pTimStr;            //  ヒープ上の時刻文字列へのポインタ
16 : } NODEDATA, *PNODEDATA;
17 :
18 : //----- ノード削除時のコールバック関数 ----------------------------------------------------------------------//
19 : static  VO   CALLBACK cbRemove(UX key, VOP pData, UI len, UI nest, UX cbp)
20 : {
21 :     AjcPrintF(TEXT("Deleted   - key=%4u, len=%d, nest=%d, data='%s'¥n"), (UI)key, len, nest, ((PNODEDATA)pData)->pTimStr);
22 :     AjcTFree(((PNODEDATA)pData)->pTimStr);
23 : }
24 : //----- 全ノード取得（全ノードプリント）用コールバック関数 --------------------------------------------------//
25 : static BOOL CALLBACK cbPrintNode(UX key, VOP pData, UI len, UI nest, UX cbp)
26 : {
27 :     PAJCAVLNODE pNode = ((PAJCAVLNODE)pData) - 1;
28 :
29 :     AjcPrintF(TEXT("%-26s:%*u, %d¥n"), ((PNODEDATA)pData)->pTimStr, nest * 6, (UI)key, pNode->bal);
30 :     return TRUE;
31 : }
32 : //----- 現在時刻文字列生成（現在時刻文字列を格納した、動的メモリへのポインタを返す）----------------------//
33 : static  UTP     GetTimeStr(VO)
34 : {
35 :     UTP         rc;
36 :     SYSTEMTIME  st;
37 :     UI          len;
38 :     UT          szTim[64];
39 :
40 :     GetLocalTime(&st);
```

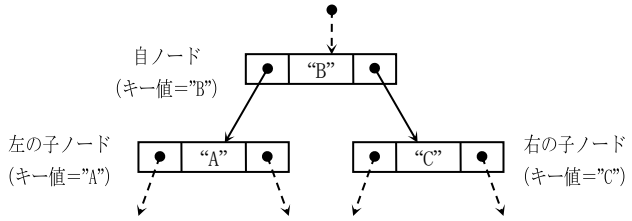
```
41 :     AjcSnPrintF(szTim, AJCTSIZE(szTim), TEXT("%02d:%02d:%02d.%03d"), st.wHour, st.wMinute, st.wSecond, st.wMilliseconds);
42 :     len = (UI)_tcslen(szTim) + 1;
43 :     MAjcStrCpy(rc = AjcTAlloc(len), len, szTim);
44 :     return rc;
45 : }
46 : //----- 文字列入力通知 ----------------------------------------------------------------------------------------//
47 : static VO CALLBACK cbNtcArgs(int argc, UT *argv[], C_UTP pTxt, UX cbp)
48 : {
49 :     NODEDATA    Node;
50 :
51 :     //  コマンド実行
52 :     if (argc >= 1) {
53 :         //  ノード挿入
54 :         if      (_tcsicmp(argv[0], TEXT("I")) == 0) {
55 :             if (argc == 2) {
56 :                 Node.pTimStr = GetTimeStr();
57 :                 if (!AjcAvlInsNode(hAvl, _ttoi(argv[1]), &Node, sizeof Node)) {
58 :                     if (Node.pTimStr != NULL) AjcTFree(Node.pTimStr); Node.pTimStr = NULL;
59 :                     AjcPrintF(TEXT("ノードの挿入を失敗しました。¥n"));
60 :                 }
61 :             }
62 :             else AjcPrintF(TEXT("*** Invalid parameter.¥n"));
63 :         }
64 :         //  ノード置換
65 :         else if (_tcsicmp(argv[0], TEXT("R")) == 0) {
66 :             if (argc == 2) {
67 :                 Node.pTimStr = GetTimeStr();
68 :                 if (!AjcAvlRepNode(hAvl, _ttoi(argv[1]), &Node, sizeof Node)) {
69 :                     if (Node.pTimStr != NULL) AjcTFree(Node.pTimStr); Node.pTimStr = NULL;
70 :                     AjcPrintF(TEXT("ノードの置換を失敗しました。¥n"));
71 :                 }
72 :             }
73 :             else AjcPrintF(TEXT("*** Invalid parameter.¥n"));
74 :         }
75 :         //  ノードの挿入／置換
76 :         else if (_tcsicmp(argv[0], TEXT("X")) == 0) {
77 :             if (argc == 2) {
78 :                 Node.pTimStr = GetTimeStr();
79 :                 if (!AjcAvlInsOrRepNode(hAvl, _ttoi(argv[1]), &Node, sizeof Node)) {
80 :                     if (Node.pTimStr != NULL) AjcTFree(Node.pTimStr); Node.pTimStr = NULL;
81 :                     AjcPrintF(TEXT("ノードの挿入／置換を失敗しました。¥n"));
82 :                 }
83 :             }
84 :             else AjcPrintF(TEXT("*** Invalid parameter.¥n"));
85 :         }
86 :         //  ノード削除
87 :         else if (_tcsicmp(argv[0], TEXT("D"))  == 0) {
88 :             if (argc == 2) {
89 :                 if (!AjcAvlDelNode(hAvl, _ttoi(argv[1]))) {
90 :                     AjcPrintF(TEXT("ノードの削除を失敗しました。¥n"));
91 :                 }
92 :             }
93 :             else AjcPrintF(TEXT("*** Invalid parameter.¥n"));
94 :         }
95 :         //  全ノード破棄
96 :         else if (_tcsicmp(argv[0], TEXT("A"))  == 0) {
97 :             if (!AjcAvlDelAllNodes(hAvl)) {
98 :                 AjcPrintF(TEXT("全ノードの削除を失敗しました。¥n"));
99 :             }
100 :        }
101 :        //  全ノード表示
102 :        else if (_tcsicmp(argv[0], TEXT("P"))  == 0) {
103 :            AjcAvlEnumNodes(hAvl, cbPrintNode, TRUE);
104 :        }
105 :        //  その他
106 :        else {
107 :            AjcPrintF(TEXT("*** Invalid command.¥n"));
108 :        }
109 :    }
110 : }
111 : //----- コンソール強制終了ハンドラ ----------------------------------------------------------------------------//
112 : static BOOL CALLBACK cbConApExit(DWORD CtrlType)
113 : {
114 :     //  ＡＶＬインスタンス消去
115 :     if (hAvl != NULL) {
116 :         AjcAvlDelete(hAvl);
117 :         hAvl = NULL;
118 :     }
119 :     return FALSE;   //  FALSE : 次のハンドラをコール
120 : }
121 : //=============================================================================================================//
122 : int  AjcMain(int argc, UTP argv[])
123 : {
124 :     NODEDATA    Node;
125 :
126 :     AjcSetStdoutMode();
127 :
128 :     //----- コンソール強制終了ハンドラ設定 ------------------------------------------------------------------//
129 :     SetConsoleCtrlHandler(cbConApExit, TRUE);
130 :
```

```
131 :     //----- インスタンス生成 ----------------------------------------------------------------------------//
132 :     hAvl = AjcAvlCreate(0, NULL, cbRemove);
133 :
134 :     //  コマンドメニュー
135 :     AjcPrintF(TEXT("¥n 以下のコマンドを入力してください。¥n¥n"));
136 :     AjcPrintF(TEXT("  I   nnn     : ノード挿入, key=nnn¥n"));
137 :     AjcPrintF(TEXT("  R   nnn     : ノード置換, key=nnn¥n"));
138 :     AjcPrintF(TEXT("  X   nnn     : ノードの挿入／置換, key=nnn¥n"));
139 :     AjcPrintF(TEXT("  D   nnn     : ノード削除, key=nnn¥n"));
140 :     AjcPrintF(TEXT("  A           : 全ノードを破棄¥n"));
141 :     AjcPrintF(TEXT("  P           : 全ノード表示¥n"));
142 :     AjcPrintF(TEXT("  <ESC>キー   : 終了¥n"));
143 :     AjcPrintF(TEXT("¥n"));
144 :
145 :     //----- 初期ノード登録／表示 ------------------------------------------------------------------------//
146 :     Node.pTimStr = AjcTAlloc(16); MAjcStrCpy(Node.pTimStr, 16, TEXT("Pre-Inited  50")); AjcAvlInsNode(hAvl,  50, &Node, sizeof Node);
147 :     Node.pTimStr = AjcTAlloc(16); MAjcStrCpy(Node.pTimStr, 16, TEXT("Pre-Inited  60")); AjcAvlInsNode(hAvl,  60, &Node, sizeof Node);
148 :     Node.pTimStr = AjcTAlloc(16); MAjcStrCpy(Node.pTimStr, 16, TEXT("Pre-Inited  70")); AjcAvlInsNode(hAvl,  70, &Node, sizeof Node);
149 :     Node.pTimStr = AjcTAlloc(16); MAjcStrCpy(Node.pTimStr, 16, TEXT("Pre-Inited  80")); AjcAvlInsNode(hAvl,  80, &Node, sizeof Node);
150 :     Node.pTimStr = AjcTAlloc(16); MAjcStrCpy(Node.pTimStr, 16, TEXT("Pre-Inited  90")); AjcAvlInsNode(hAvl,  90, &Node, sizeof Node);
151 :     Node.pTimStr = AjcTAlloc(16); MAjcStrCpy(Node.pTimStr, 16, TEXT("Pre-Inited 100")); AjcAvlInsNode(hAvl, 100, &Node, sizeof Node);
152 :     AjcAvlEnumNodes(hAvl, cbPrintNode, TRUE);
153 :
154 :     //----- コマンド処理 --------------------------------------------------------------------------------//
155 :     AjcPrintF(TEXT("Input(%2d) : "), AjcAvlGetCount(hAvl));
156 :     while (AjcConInputByNtc(32, 0, cbNtcArgs)) {
157 :         AjcPrintF(TEXT("Input(%2d) : "), AjcAvlGetCount(hAvl));
158 :     }
159 :     if (hAvl != NULL) {
160 :         AjcAvlDelete(hAvl);
161 :         hAvl = NULL;
162 :     }
163 :
164 :     return 0;
165 : }
```

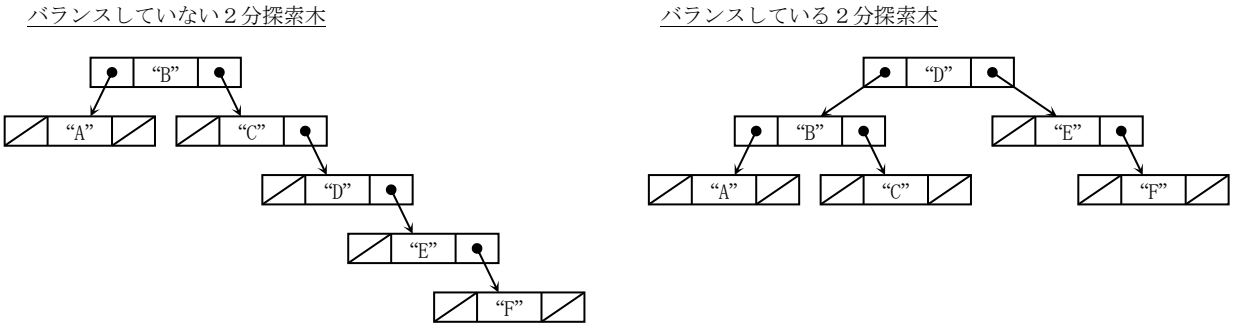
33. 平衡２分木（文字列キー）

文字列をキーとした平衡２分木を構成します。

２分探索木の各ノードは、２つの子ノード（左の子と右の子）への位置情報（ポインタ）を持ち、左の子は自ノードより低い値の文字列キーを持ち、右の子は自ノードより高い値の文字列キーを持ちます。



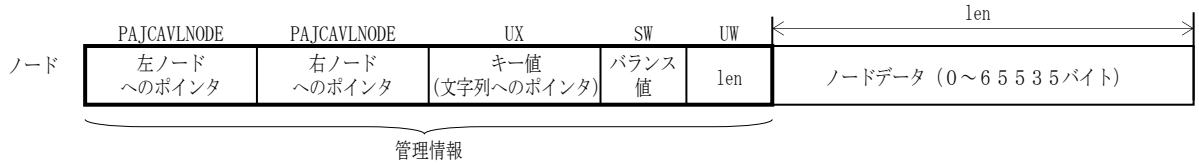
平衡２分木とは、さらに、木の高さがバランスした２分探索木構造を意味します。



平衡した２分探索木構造では、特定のノードを検索するのに必要な探索回数は、木の高さと同じであり、ノードの個数をｎとした場合、ｌｏｇ２（ｎ）＋１回の探索で見つけることができます。たとえば、ノード数が最大６５５３５ヶである場合、１６回以内の探索で見つけ出すことができます。

本ライブラリでは、ランダムに与えられる文字列キーから、平衡した２分探索木（ＡＶＬ木）を構築し、ノードの検索や、シーケンシャルなノード読み出しを行うとができます。

各ノードの先頭には、以下のような管理情報が付加されます。



※「バランス値」は、自ノードを頂点とした、右側ノードの高さから左側ノードの高さを減算した値（ -1 ～ +1 ）を意味します。

キー値は文字列であり、文字列の比較でキーの大小を判定します。

33.1. サポートＡＰＩ

平衡２分木（文字列キー）のサポートＡＰＩ一覧を以下に示します。

#	機能	関数名	備考
1	インスタンス生成	AjcAvsCreate	
2	インスタンス消去	AjcAvsDelete	
3	コールバックパラメタの設定	AjcAvsSetCbp	
4	文字列比較モードの設定	AjcAvsSetCompMode	
5	ノード挿入	AjcAvsInsNode	
6	ノード取得	AjcAvsGetNodeByKey	
7	ノードアドレス取得	AjcAvsGetNodePtr	
8	ノードのキー値取得	AjcAvsGetNodeKey	
9	先頭ノード取得	AjcAvsGetTopNode	
10	最終ノード取得	AjcAvsGetLastNode	
11	ノード置換	AjcAvsRepNode	
12	ノード挿入／置換	AjcAvsInsOrRepNode	
13	ノード削除	AjcAvsDelNode	
14	全ノードを削除	AjcAvsDelAllNodes	
15	ノード数取得	AjcAvsGetCount	
16	全ノードのシーケンシャルな読み出し	AjcAvsEnumNodes AjcAvsEnumNodesEx	インスタンス生成時のコールバックパラメタを使用 コールバックパラメタを独自に指定
17	全ノードへのポインタ配列生成	AjcAvsCreatePtrArr	
18	全ノードへのポインタ配列解放	AjcAvsReleasePtrArr	
19	マルチスレッドの許可／禁止	AjcAvsEnableMultiThread	

33.1.1. インスタンス生成（AjcAvsCreate）

形　式　：　HAJCAVS　AjcAvsCreate (EAJCCMPMODE CmpMode, UX cbp,
VO　(CALLBACK *cbRemove)(C_UTP pKey, VOP pNodeData, UI len, UI nest, UX cbp));

引　数　：　CmpMode －　文字列の比較モード
・AJCCMP_EXACT_WIDTH　：英大小文字を区別する
・AJCCMP_IGNORE_WIDTH　：英大小文字を区別しない
・AJCCMP_MIX　：同一の場合のみ英大小文字を区別する
cbp　－ コールバックパラメタ
cbRemove　－ 削除されるノード通知用コールバック関数　（不要時はＮＵＬＬ）

説　明　：　平衡２分木（文字列キー）のインスタンスを生成し、初期化します。
CmpMode = AJCCMP_EXACT_WIDTH とした場合は、英大小文字を区別して文字列の比較を行います。(”ABC”≠”abc”)
CmpMode = AJCCMP_IGNORE_WIDTH とした場合は、英大小文字を区別せずに文字列の比較を行います。(”ABC”＝”abc”)
CmpMode = AJCCMP_MIX とした場合は、英大小文字を区別せずに文字列の比較を行いますが、同一文字列の場合は、英大小文字を区別して文字列の比較を行います。(アルファベット順に並ぶように比較する)

cbRemove は、削除されるノード通知用コールバック関数を指定します。
このコールバック関数は、AjcAvsDelNode, AjcAvsDelAllNodes、あるいは AjcAvsDelete 関数の実行中に、ノードを削除（開放）する直前にコールバックされます。
削除されるノードを通知する必要が無い場合は、NULL を指定します。

戻り値　：　≠NULL － 成功　（インスタンスハンドルを返します）
＝NULL － 失敗

コールバック： コールバック関数の仕様は、以下のとおりです。

cbRemove（削除ノード通知）

形　式 ： VO　CALLBACK ***cbRemove***(C_UTP pKey, VOP pNodeData, UI len, UI nest, UX cbp)；

引　数 ： pKey　　- 文字列キー（文字列へのポインタ）
pNodeData - 削除するノードデータのアドレス（ノードに付与されたデータの先頭アドレス）
len　　- 削除するノードデータのバイト数（ノードに付与されたデータのバイト数）
nest　　- 削除するノードデータのネスト位置（ルートノードからの自ノードまでの木の高さ，ルートノードは1）
cbp　　- コールバックパラメタ

説　明 ： このコールバック関数の実行直後に、当該ノードが削除されることを通知します。
必要ならば、key と pNodeData で示される、キー値，ノードデータをもとに、ローカルな資源の解放等を行います。

33.1.2. インスタンス消去（AjcAvsDelete）

形　式 ： BOOL　AjcAvsDelete (HAJCAVS hAvs)；

引　数 ： hAvs - インスタンスハンドル

説　明 ： 全ノードの削除（開放）を行い、AjcAvsCreate で動的に確保したリソースを解放します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

33.1.3. コールバックパラメタの設定（AjcAvsSetCbp）

形　式 ： BOOL AjcAvsSetCbp (HAJCAVS hAvs、UX cbp)；

引　数 ： hAvs - インスタンスハンドル
cbp　- 設定するコールバックパラメタ

説　明 ： コールバックパラメタを設定します。(AjcAvsCreate()で設定したコールバックパラメタを変更します)

戻り値 ： TRUE　- 成功
FALSE - 失敗

33.1.4. 文字列比較モードの設定（AjcAvsSetCompMode）

形　式 ： VO　AjcAvsSetCompMode (HAJCAVS hAvs、EAJCCMPMODE CmpMode)；

引　数 ： hAvs　　- インスタンスハンドル
CmpMode　- 文字列比較モード
・AJCCMP_EXACT_WIDTH　　：英大小文字を区別する
・AJCCMP_IGNORE_WIDTH　：英大小文字を区別しない
・AJCCMP_MIX　　：同一の場合のみ英大小文字を区別する

説　明 ： 文字列キーの比較モードを設定します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

33.1.5. ノード挿入（AjcAvsInsNode）

形　式 ： BOOL　AjcAvsInsNode(HAJCAVS hAvs, C_UTP pKey, C_VOP pNodeData, UI len);

引　数 ： hAvs　　- インスタンスハンドル
pKey　　- ノードに付与する文字列キー（文字列へのポインタ）
pNodeData – 挿入するノードデータの先頭アドレス（ノードデータ無しの場合はNULL）
len　　- 挿入するノードデータのバイト数　　（ノードデータ無しの場合は０）

説　明 ： 平衡２分木にノードを追加し、指定された文字列キーとデータのペアを作成します。
ノードを追加した後も、２分木の平衡は保たれます。
pKeyで指定された文字列キーと同じ文字列キーのノードが既に存在する場合は、ノードを追加することはできません。
pKeyで指定した文字列キーは、ノード挿入後も保持しておく必要はありません。(内部的に動的なメモリに退避します)

戻り値 ： TRUE　- 成功
FALSE - 失敗

33.1.6. ノード取得（AjcAvsGetNode）

形　式 ： int　AjcAvsGetNode　　(HAJCAVS hAvs, C_UTP pKey, VOP pBuf, UI lBuf);

引　数 ： hAvs　　- インスタンスハンドル
pKey　　- 検索するノードの文字列キー（文字列へのポインタ）
pBuf　　- 見つかったノードのデータを格納するバッファのアドレス（不要時はＮＵＬＬ）
lBuf　　- 見つかったノードのデータを格納するバッファのバイト数（pBuf=NULLの場合は不要）

説　明 ： 平衡２分木から、pKeyで指定された文字列キーのノードを検索し、見つかった場合は、pBufで示されるバッファにノードデータ部分をコピーします。

戻り値 ： ≠-1 - 指定キー値のノードが見つかった（当該ノードデータのバイト数）
＝-1 - 失敗

33.1.7. ノードアドレス取得（AjcAvsGetNodePtr）

形　式 ： VOP　AjcAvsGetNodePtr(HAJCAVS hAvs, C_UTP pKey, UIP pLen);

引　数 ： hAvs　　- インスタンスハンドル
pKey　　- 検索するノードの文字列キー（文字列へのポインタ）
pLen　　- 見つかったノードデータのバイト数を格納するバッファのアドレス（不要時はＮＵＬＬ）

説　明 ： 平衡２分木から、pKeyで指定された文字列キーのノードを検索し、見つかった場合は、pLenで示されるバッファにノードデータのバイト数を格納し、ノードデータのアドレスを返します。

戻り値 ： ≠NULL - 成功（当該ノード・データへのポインタ）
＝NULL - 失敗

33.1.8. ノードのキー値取得（AjcAvsGetNodeKey）

形　式 ： C_VOP　AjcAvsGetNodeKey(HAJCAVS hAvs, C_VOP pNode);

引　数 ： hAvs　　- インスタンスハンドル
pNode　　- ノードデータへのポインタ

説　明 ： ノードデータへのポインタから、ノードのキー値（文字列へのポインタ）を取得します。

戻り値 ： ≠NULL - ノードのキー値（文字列へのポインタ）
＝NULL - 失敗

33.1.9. 先頭ノード取得（AjcAvsGetTopNode）

形　式 ： int AjcAvsGetTopNode(HAJCAVS hAvs, C_UTP *ppKey, VOP pBuf, UI lBuf);

引　数 ： hAvl　　- インスタンスハンドル
ppKey　- 先頭ノードの文字列キー（文字列へのポインタ）を格納するバッファのアドレス（不要時はＮＵＬＬ）
pBuf　　- 見つかったノードデータを格納するバッファのアドレス（不要時はＮＵＬＬ）
lBuf　　- 見つかったノードデータを格納するバッファのバイト数（pBuf=NULL の場合は不要）

説　明 ： 平衡２分木から、pBuf で示されるバッファに、先頭ノード（キー値が最小のノード）のデータ部分をコピーします。

戻り値 ： ≠-1 - 先頭ノードが見つかった（先頭ノードデータのバイト数）
＝-1 - 失敗

33.1.10. 最終ノード取得（AjcAvsGetLastNode）

形　式 ： int AjcAvsGetLastNode(HAJCAVS hAvs, C_UTP *ppKey, VOP pBuf, UI lBuf);

引　数 ： hAvl　　- インスタンスハンドル
ppKey　- 最終ノードの文字列キー（文字列へのポインタ）を格納するバッファのアドレス（不要時はＮＵＬＬ）
pBuf　　- 見つかったノードの付与データを格納するバッファのアドレス（不要時はＮＵＬＬ）
lBuf　　- 見つかったノードの付与データを格納するバッファのバイト数（pBuf=NULL の場合は不要）

説　明 ： 平衡２分木から、pBuf で示されるバッファに、最終ノード（キー値が最大のノード）のデータ部分をコピーします。

戻り値 ： ≠-1 - 最終ノードが見つかった（最終ノードデータのバイト数）
＝-1 - 失敗

33.1.11. ノード置換（AjcAvsRepNode）

形　式 ： BOOL AjcAvsRepNode(HAJCAVS hAvs, C_UTP pKey, C_VOP pNodeData, UI len);

引　数 ： hAvs　　- インスタンスハンドル
pKey　　- 検索するノードの文字列キー（文字列へのポインタ）
pNodeData - 置換する新ノードデータのアドレス（ノードデータ無しの場合は NULL）
len　　- 置換する新ノードデータのバイト数（ノードデータ無しの場合は０）

説　明 ： pKey で指定されたノードのデータを置換します。
pKey で指定された文字列キーと同じ文字列キーのノードが見つかった場合、当該ノードのデータを置換し TRUE を返します。
pKey で指定された文字列キーと同じキー値のノードが見つからない場合は、FALSE を返します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

33.1.12. ノード挿入／置換（AjcAvsInsOrRepNode）

形　式 ： BOOL AjcAvsInsOrRepNode(HAJCAVS hAvs, C_UTP pKey, C_VOP pNodeData, UI len);

引　数 ： hAvs　　- インスタンスハンドル
pKey　　- 検索するノードの文字列キー（文字列へのポインタ）
pNodeData - 置換するノードデータのアドレス（ノードデータ無しの場合は NULL）
len　　- 置換するノードデータのバイト数（ノードデータ無しの場合は０）

説　明 ： 平衡２分木に同一文字列キーのノードがある場合は、指定されたノードのデータを書き換えます。
同一文字列キーのノードが無い場合は、「AjcAvsInsNode()」と同様に平衡２分木にノードを追加します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

33.1.13. ノード削除（AjcAvsDelNode）

形　式 ： BOOL　AjcAvsDelNode(HAJCAVS hAvs, C_UTP pKey);

引　数 ： hAvl　　- インスタンスハンドル
pKey　　- 検索するノードの文字列キー（文字列へのポインタ）

説　明 ： 平衡２分木から pKey で指定された文字列キーのノードを削除します。
ノードを削除した後も、２分木の平衡は保たれます。

戻り値 ： TRUE　- 成功
FALSE - 失敗

33.1.14. 全ノード削除（AjcAvsDelAllNodes）

形　式 ： BOOL　AjcAvsDelAllNodes(HAJCAVS hAvs);

引　数 ： hAvs - インスタンスハンドル

説　明 ： 平衡２分木から全てのノードを削除します。
この関数を実行すると、ノードが１つも無い状態になります。

戻り値 ： TRUE　- 成功
FALSE - 失敗

33.1.15. ノード数取得（AjcAvsGetCount）

形　式 ： UI　AjcAvsGetCount (HAJCAVS hAvs);

引　数 ： hAvs - インスタンスハンドル

説　明 ： ノードの個数を取得します。

戻り値 ： ノードの個数（パラメタエラー時は、０を返す）

33.1.16. 全ノードのシーケンシャルな読み出し（AjcAvsEnumNodes[Ex]）

形　式 ： UI AjcAvsEnumNodes (HAJCAVS hAvs,
BOOL (CALLBACK *cbNtcNode)(C_UTP pKey, VOP pNodeData, UI len, UI nest, UX cbp),
BOOL fDownSeq);

UI AjcAvsEnumNodesEx(HAJCAVS hAvs, **UX cbp,**
BOOL (CALLBACK *cbNtcNode)(C_UTP pKey, VOP pNodeData, UI len, UI nest, UX cbp),
BOOL fDownSeq);

引　数 ： hAvs　　- インスタンスハンドル
cbp　　- コールバックパラメタ
cbNtcNode - 読み出したノードの情報を通知するコールバック関数（不要時はＮＵＬＬ）
fDownSeq - ノードの読み出し順の指定（FALSE ： 昇順,　TRUE ： 降順）

説　明 ： 平衡２分木から、文字列キー値の昇順、あるいは、降順に、順次ノードを読み出します。
読み出したノードの情報は、cbNtcNode で示されるコールバック関数を呼び出すことにより通知します。
CbNtcNode に NULL を指定した場合は、単に、現在の総ノード数を返します。

戻り値 ： 現在の総ノード数（エラー時は０を返す）

コールバック： コールバック関数の仕様は、以下のとおりです。

cbNtcNode（ノード通知）

形　式 ： BOOL CALLBACK ***cbNtcNode*** (C_UTP pKey, VOP pNodeData, UI len, UI nest, UX cbp);

引　数 ： pKey　　- キー文字列へのポインタ
pNodeData - 読み出したノードデータのアドレス（ノードに付与されたデータの先頭アドレス）
len　　- 読み出したノードデータのバイト数（ノードに付与されたデータのバイト数）
nest　　- 読み出したノードデータのネスト位置（ルートノードからの自ノードまでの木の高さ，
ルートノードの場合は１）
cbp　　- コールバックパラメタ
AjcAvsEnumNodes()は、AjcAvsCreate()/AjcAvsSetCbp()で指定したコールバックパラメタ
AjcAvsEnumNodesEx()の場合は当該関数で指定したコールバックパラメタ

説　明 ： 順次、全てのノードを通知します。
FALSE を返すと、ノードの読み出しを中止します。

戻り値 ： TRUE　- ノード読み出し継続
FALSE　- ノード読み出し中止

33.1.17. 全ノードへのポインタ配列生成（AjcAvlCreatePtrArr）

形　式 ： PCAJCAVSPTR AjcAvsCreatePtrArr (HAJCAVS hAvs, UIP pNum, BOOL fDownSeq)

引　数 ： hAvs　　- インスタンスハンドル
pNum　　- 配列要素数を格納するバッファのアドレス（不要時はＮＵＬＬ）
fDownSeq　- ポインタ配列順の指定（FALSE ： 昇順,　TRUE ： 降順）

説　明 ： 全ノードへのポインタ配列を生成します。
ポインタ配列は、ノード数＋１の要素が作成され、最終要素には NULL が格納されます。
生成した配列を使用した後は、AjcAvlReleasePtrArr ()により配列を解放してください。

戻り値 ： ≠NULL - 成功（全ノードへのポインタ配列の先頭アドレス）
＝NULL - 失敗

備　考 ： 本 API は、以下の構造体の配列の先頭アドレスを返します。

```
typedef struct {
    C_UTP   pKey;                   /*  文字列キーへのポインタ  */
    VOP     pNode;                  /*  ノードへのポインタ      */
    UI      len;                    /*  ノードのバイト数        */
} AJCAVSPTR, *PAJCAVSPTR;
typedef const AJCAVSPTR *PCAJCAVSPTR;
```

また、「MAJCAVSPTR」マクロにより、構造体名とポインタタイプを変更した、同型の構造体を定義できます。

```
  MAJCAVSPTR(MYSTRUCT, PTRTYPE)
+ typedef struct {
+    C_UTP     pKey;                /*  文字列キーへのポインタ  */
+    PTRTYPE   pNode;               /*  ノードへのポインタ      */
+    UI        len;                 /*  ノードのバイト数        */
+ } MYSTRUCT, *PMYSTRUCT;
+ typedef const MYSTRUCT *PCMYSTRUCT;
```

「MAJCAVSPTR」マクロを定義し、本ＡＰＩを呼び出す際に、生成した構造体でキャストできます。

```
  PCMYSTRUCT  p;
  p = (PCMYSTRUCT)AjcAvsCreatePtrArr(・・・);
```

33.1.18. 全ノードへのポインタ配列解放（AjcAvsReleasePtrArr）

形　式 ： BOOL AjcAvsReleasePtrArr (HAJCAVS hAvs, PCAJCAVSPTR pArr)

引　数 ： hAvs　　- インスタンスハンドル
pArr　　- 全ノードへのポインタ配列の先頭アドレス（AjcAvsCreatePtrArr()の戻り値）

説　明 ： AjcAvsCreatePtrArr()で生成した、全ノードへのポインタ配列を解放します。

戻り値 ： TRUE　- 成功
FALSE　- 失敗

33.1.19. マルチスレッドの許可／禁止（AjcAvsEnableMultiThread）

形　式　：　BOOL　AjcAvsEnableMultiThread(HAJCAVS hAvs, BOOL fEnable);

引　数　：　hAvl　　－ インスタンスハンドル
fEnable　－ TRUE　：複数のスレッドからのアクセスを許可
FALSE：複数のスレッドからのアクセスを禁止

説　明　：　fEnable=TRUE とした場合、複数のスレッドによる、平衡２分木へのノード追加／削除や、検索を可能にします。
この場合、各関数の入り口と出口で、クリティカルセクションによるスレッド間の排他制御を行います。
但し、次の関数は、（fEnable=TRUE としても）マルチスレッドでの排他制御を行いません。

・AjcAvsCreate（インスタンス生成）
・AjcAvsDelete（インスタンス消去）
・AjcAvsEnableMultiThread（本関数）

fEnable=FALSE（デフォルト）とした場合は、マルチスレッドでの排他制御を行いません。

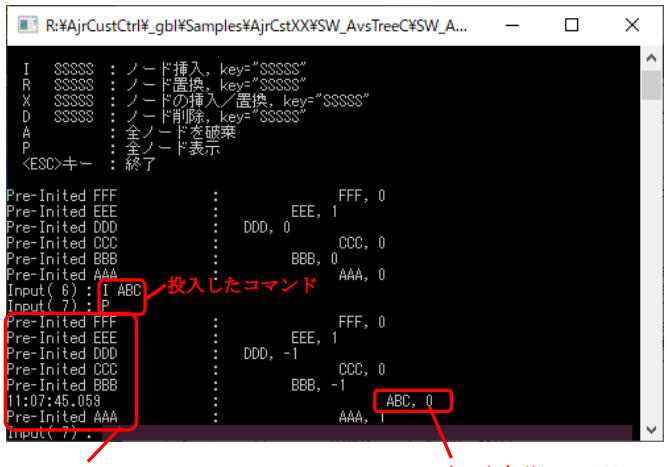
戻り値　：　前回の許可／禁止状態

33.2. サンプルプログラム

33.2.1. SW_AvsTreeC（ＡＶＳツリー操作サンプル）

以下のサンプルプログラムは、コマンド入力により、２文木にノード挿入や削除等を行います。
ノードデータは、時刻文字列へのポインタです。
時刻文字列は動的なメモリに格納し、ノードの削除時に解放します。
起動時は初期ノードとしてキー＝”AAA”, “BBB”, “CCC”, “DDD”, “EEE”, “FFF” の６つのノードを挿入します。
コマンドの形式は以下の通りです。 ESC キーを押すとプログラムを終了します。

#	コマンド形式	内容
1	I　<キー文字列>	ノード挿入, key=<キー文字列>
2	R　<キー文字列>	ノード置換, key=<キー文字列>
3	X　<キー文字列>	ノードの挿入／置換, key=<キー文字列>
4	D　<キー文字列>	ノード削除, key=<キー文字列>
5	A	全ノードを破棄
6	P	全ノード表示




```
 1 : //
 2 : //  SW_AvsTreeC.c
 3 : //
 4 : #include    <AjrCstXX.h>
 5 : #include    <stdio.h>
 6 : #include    <conio.h>
 7 : #include    <time.h>
 8 : #include    <tchar.h>
 9 :
10 : //----- ワーク ----------------------------------------------------------------------------------------//
11 : static  HAJCAVS     hAvs = NULL;
12 :
13 : //----- ノードデータ形式 ------------------------------------------------------------------------------//
14 : typedef struct {
15 :     UTP     pTimStr;            //  ヒープ上の時刻文字列へのポインタ
16 : } NODEDATA, *PNODEDATA;
17 :
18 : //----- ノード削除時のコールバック関数 ----------------------------------------------------------------//
19 : static  VO   CALLBACK cbRemove(C_UTP pKey, VOP pData, UI len, UI nest, UX cbp)
20 : {
21 :     AjcPrintF(TEXT("Deleted   - key='%s', len=%d, nest=%d, data='%s'¥n"), pKey, len, nest, ((PNODEDATA)pData)->pTimStr);
22 :     AjcTFree(((PNODEDATA)pData)->pTimStr);
23 : }
24 : //----- 全ノード取得（全ノードプリント）用コールバック関数 --------------------------------------------//
25 : static BOOL CALLBACK cbPrintNode(C_UTP pKey, VOP pData, UI len, UI nest, UX cbp)
26 : {
27 :     PAJCAVLNODE pNode = ((PAJCAVLNODE)pData) - 1;
28 :
29 :     AjcPrintF(TEXT("%-26s:%*s, %d¥n"), ((PNODEDATA)pData)->pTimStr, nest * 6, pKey, pNode->bal);
30 :     return TRUE;
31 : }
32 : //----- 現在時刻文字列生成（現在時刻文字列を格納した、動的メモリへのポインタを返す）----------------//
33 : static  UTP     GetTimeStr(VO)
```

```
{
    UTP         rc;
    SYSTEMTIME  st;
    UI          len;
    UT          szTim[64];

    GetLocalTime(&st);
    AjcSnPrintF(szTim, AJCTSIZE(szTim), TEXT("%02d:%02d:%02d.%03d"), st.wHour, st.wMinute, st.wSecond, st.wMilliseconds);
    len = (UI)_tcslen(szTim) + 1;
    MAjcStrCpy(rc = AjcTAlloc(len), len, szTim);
    return rc;
}
//----- 文字列入力通知 ----------------------------------------------------------------------------------------//
static VO CALLBACK cbNtcArgs(int argc, UT *argv[], C_UTP pTxt, UX cbp)
{
    NODEDATA    Node;

    //  コマンド実行
    if (argc >= 1) {
        //  ノード挿入
        if      (_tcsicmp(argv[0], TEXT("I")) == 0) {
            if (argc == 2) {
                Node.pTimStr = GetTimeStr();
                if (!AjcAvsInsNode(hAvs, argv[1], &Node, sizeof Node)) {
                    if (Node.pTimStr != NULL) AjcTFree(Node.pTimStr); Node.pTimStr = NULL;
                    AjcPrintF(TEXT("ノードの挿入を失敗しました。¥n"));
                }
            }
            else AjcPrintF(TEXT("*** Invalid parameter.¥n"));
        }
        //  ノード置換
        else if (_tcsicmp(argv[0], TEXT("R")) == 0) {
            if (argc == 2) {
                Node.pTimStr = GetTimeStr();
                if (!AjcAvsRepNode(hAvs, argv[1], &Node, sizeof Node)) {
                    if (Node.pTimStr != NULL) AjcTFree(Node.pTimStr); Node.pTimStr = NULL;
                    AjcPrintF(TEXT("ノードの置換を失敗しました。¥n"));
                }
            }
            else AjcPrintF(TEXT("*** Invalid parameter.¥n"));
        }
        //  ノードの挿入／置換
        else if (_tcsicmp(argv[0], TEXT("X")) == 0) {
            if (argc == 2) {
                Node.pTimStr = GetTimeStr();
                if (!AjcAvsInsOrRepNode(hAvs, argv[1], &Node, sizeof Node)) {
                    if (Node.pTimStr != NULL) AjcTFree(Node.pTimStr); Node.pTimStr = NULL;
                    AjcPrintF(TEXT("ノードの挿入／置換を失敗しました。¥n"));
                }
            }
            else AjcPrintF(TEXT("*** Invalid parameter.¥n"));
        }
        //  ノード削除
        else if (_tcsicmp(argv[0], TEXT("D"))  == 0) {
            if (argc == 2) {
                if (!AjcAvsDelNode(hAvs, argv[1])) {
                    AjcPrintF(TEXT("ノードの削除を失敗しました。¥n"));
                }
            }
            else AjcPrintF(TEXT("*** Invalid parameter.¥n"));
        }
        //  全ノード破棄
        else if (_tcsicmp(argv[0], TEXT("A"))  == 0) {
            if (!AjcAvsDelAllNodes(hAvs)) {
                AjcPrintF(TEXT("全ノードの削除を失敗しました。¥n"));
            }
        }
        //  全ノード表示
        else if (_tcsicmp(argv[0], TEXT("P"))  == 0) {
            AjcAvsEnumNodes(hAvs, cbPrintNode, TRUE);
        }
        //  その他
```

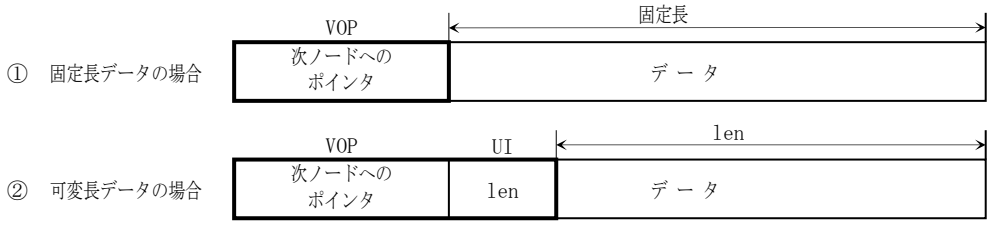
```
106 :         else {
107 :             AjcPrintF(TEXT("*** Invalid command.¥n"));
108 :         }
109 :     }
110 : }
111 : //----- コンソール強制終了ハンドラ ----------------------------------------------------------------------------//
112 : static BOOL CALLBACK cbConApExit(DWORD CtrlType)
113 : {
114 :     //  ＡＶＬインスタンス消去
115 :     if (hAvs != NULL) {
116 :         AjcAvsDelete(hAvs);
117 :         hAvs = NULL;
118 :     }
119 :     return FALSE;   //  FALSE : 次のハンドラをコール
120 : }
121 : //=============================================================================================================//
122 : int  AjcMain(int argc, UTP argv[])
123 : {
124 :     NODEDATA    Node;
125 :
126 :     AjcSetStdoutMode();
127 :
128 :     //----- コンソール強制終了ハンドラ設定 ------------------------------------------------------------------//
129 :     SetConsoleCtrlHandler(cbConApExit, TRUE);
130 :
131 :     //----- インスタンス生成 --------------------------------------------------------------------------------//
132 :     hAvs = AjcAvsCreate(AJCCMP_ALPHABETIC, 0, cbRemove);
133 :
134 :     //  コマンドメニュー
135 :     AjcPrintF(TEXT("¥n 以下のコマンドを入力してください。¥n¥n"));
136 :     AjcPrintF(TEXT("  I   SSSSS  : ノード挿入, key=¥"SSSSS¥"¥n"));
137 :     AjcPrintF(TEXT("  R   SSSSS  : ノード置換, key=¥"SSSSS¥"¥n"));
138 :     AjcPrintF(TEXT("  X   SSSSS  : ノードの挿入／置換, key=¥"SSSSS¥"¥n"));
139 :     AjcPrintF(TEXT("  D   SSSSS  : ノード削除, key=¥"SSSSS¥"¥n"));
140 :     AjcPrintF(TEXT("  A          : 全ノードを破棄¥n"));
141 :     AjcPrintF(TEXT("  P          : 全ノード表示¥n"));
142 :     AjcPrintF(TEXT("  <ESC>キー  : 終了¥n"));
143 :     AjcPrintF(TEXT("¥n"));
144 :
145 :     //----- 初期ノード登録／表示 ----------------------------------------------------------------------------//
146 :     Node.pTimStr = AjcTAlloc(16); MAjcStrCpy(Node.pTimStr, 16, TEXT("Pre-Inited AAA")); AjcAvsInsNode(hAvs, TEXT("AAA"), &Node, sizeof Node);
147 :     Node.pTimStr = AjcTAlloc(16); MAjcStrCpy(Node.pTimStr, 16, TEXT("Pre-Inited BBB")); AjcAvsInsNode(hAvs, TEXT("BBB"), &Node, sizeof Node);
148 :     Node.pTimStr = AjcTAlloc(16); MAjcStrCpy(Node.pTimStr, 16, TEXT("Pre-Inited CCC")); AjcAvsInsNode(hAvs, TEXT("CCC"), &Node, sizeof Node);
149 :     Node.pTimStr = AjcTAlloc(16); MAjcStrCpy(Node.pTimStr, 16, TEXT("Pre-Inited DDD")); AjcAvsInsNode(hAvs, TEXT("DDD"), &Node, sizeof Node);
150 :     Node.pTimStr = AjcTAlloc(16); MAjcStrCpy(Node.pTimStr, 16, TEXT("Pre-Inited EEE")); AjcAvsInsNode(hAvs, TEXT("EEE"), &Node, sizeof Node);
151 :     Node.pTimStr = AjcTAlloc(16); MAjcStrCpy(Node.pTimStr, 16, TEXT("Pre-Inited FFF")); AjcAvsInsNode(hAvs, TEXT("FFF"), &Node, sizeof Node);
152 :     AjcAvsEnumNodes(hAvs, cbPrintNode, TRUE);
153 :
154 :     //----- コマンド処理 ------------------------------------------------------------------------------------//
155 :     AjcPrintF(TEXT("Input(%2d) : "), AjcAvsGetCount(hAvs));
156 :     while (AjcConInputByNtc(32, 0, cbNtcArgs)) {
157 :         AjcPrintF(TEXT("Input(%2d) : "), AjcAvsGetCount(hAvs));
158 :     }
159 :     if (hAvs != NULL) {
160 :         AjcAvsDelete(hAvs);
161 :         hAvs = NULL;
162 :     }
163 :
164 :     return 0;
165 : }
166 :
167 :
```

34. 線形リスト制御

線形リストとは、各ノード（メモリブロック）が、次のノードを示す情報（ポインタや）を持ち、次々と鎖状に連なっているノードの集まりを意味します。最後のノードはＮＵＬＬポインタを持ちます。



各ノードの先頭には、以下の情報が付加されます。



34.1. サポートＡＰＩ

線形リスト制御のサポートＡＰＩ一覧を以下に示します。

#	内容	関 数 名		備 考
		固定長データ	可変長データ	
1	インスタンス生成	AjcFQueCreate	AjcVQueCreate	
2	インスタンス消去	AjcFQueDelete	AjcVQueDelete	
3	ノードを末尾へ追加	AjcFQueEnque	AjcVQueEnque	
4	ノードを先頭へ挿入	AjcFQueEnqTop	AjcVQueEnqTop	
5	先頭ノード取り出し	AjcFQueDeque	AjcVQueDeque	
6	先頭ノード取り出し	－	AjcVQueDequeEx	AjcVQueDeque()の改良版
7	末尾ノード取り出し	AjcFQueDeqLast	AjcVQueDeqLast	
8	全ノード消去	AjcFQuePurge	AjcVQuePurge	
9	個別ノード消去	AjcFQueDelAny	AjcVQueDelAny	
10	ノードの個数取得	AjcFQueGetCount	AjcVQueGetCount	
11	２つのリストをマージ	AjcFQueMerge	AjcVQueMerge	
12	先頭ノードアドレス取得	AjcFQueTopNode	AjcVQueTopNode	
13	次のノードアドレス取得	AjcFQueNextNode	AjcVQueNextNode	
14	全ノードへのポインタ配列生成	AjcFQueCreatePtrArr	AjcVQueCreatePtrArr	
15	全ノードへのポインタ配列解放	AjcFQueReleasePtrArr	AjcVQueReleasePtrArr	
16	マルチスレッド許可／禁止	AjcFQueEnableMultiThread	AjcVQueEnableMultiThread	

34.1.1. インスタンス生成（Ajc?QueCreate）

形　式 ： HAJCFQUE　AjcFQueCreate(UI DatSize, UX cbp, VO (CALLBACK *cbRemove)(VOP pDat, UX cbp)); ----　固定長ノード用
HAJCVQUE　AjcVQueCreate(UX cbp, VO (CALLBACK *cbRemove)(VOP pDat, UI len, UX cbp)); --------- 可変長ノード用

引　数 ： DatSize　- ノード内のデータ部分のバイト数
cbp　　- コールバックパラメタ
cbRemove - ノード消去通知用コールバック関数のアドレス（不要時は NULL）

説　明 ： リスト機能のインスタンスを生成／初期化します。
「cbRemove」は、Ajc?QuePurge/ Ajc?QueDelAny や Ajc?QueDelete により、消去されるノードを通知するためのコールバック関数を指定します。
ノード消去時に、何も操作する必要が無い場合は、cbRemove=NULL を指定します。

戻り値 ： ≠NULL - 成功　（インスタンスハンドルを返します）
＝NULL - 失敗

コールバック： コールバック関数の仕様は、以下のとおりです。

cbRemove（消去されるノードデータ通知）

形　式 ： VO CALLBACK ***cbRemove***(VOP pDat, UX cbp); ---------- 固定長ノード用
VO CALLBACK ***cbRemove***(VOP pDat, UI len, UX cbp); ----- 可変長ノード用

引　数 ： pDat - ノードデータのアドレス（ノードのデータ部分を示します）
len　- ノードデータのバイト数
cbp　- コールバックパラメタ

説　明 ： 当該ノードデータに関連した後処理（リソースの開放等）を行います。

戻り値 ： なし

34.1.2. インスタンス消去（Ajc?QueDelete）

形　式 ： BOOL AjcFQueDelete (HAJCFQUE hFQue); - 固定長ノード用
BOOL AjcVQueDelete (HAJCVQUE hVQue); - 可変長ノード用

引　数 ： hFQue/hVQue - インスタンスハンドル

説　明 ： 全ノードを削除（開放）し、AjcFQueCreate/ AjcVQueCreate で動的に確保した、リソースを解放します。

AjcFQueCreate/AjcVQueCreate で cbRemove が指定されている場合は、全てのノードについて、ノードが破棄される直前に、当該ノードデータを通知します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

34.1.3. リストの末尾へノードを追加（Ajc?QueEnque[Ex]）

形　式 ： BOOL　AjcFQueEnque (HAJCFQUE hFQue, C_VOP pDat);　---------- 固定長ノード用
BOOL　AjcVQueEnque (HAJCVQUE hVQue, C_VOP pDat, UI lDat);　----- 可変長ノード用

VOP　AjcFQueEnqueEx (HAJCFQUE hFQue, C_VOP pDat);　---------- 固定長ノード用
VOP　AjcVQueEnqueEx (HAJCVQUE hVQue, C_VOP pDat, UI lDat);　--- 可変長ノード用

引　数 ： hFQue/hVQue - インスタンスハンドル
pDat　　- ノードに格納するデータのアドレス
lDat　　- ノードに格納するデータのバイト数

説　明 ： 新たなノードを作成し、リストの末尾に追加します。
pDat で示されるデータは、新たに作成したノード内にコピーされます。従って、本関数実行後、pDat で示されるデータを保持している必要はありません。

戻り値 ： TRUE /　≠NULL - 成功（追加したノードデータのアドレス）
FALSE / ＝NULL - 失敗

34.1.4. リストの先頭へノードを追加（Ajc?QueEnqTop）

形　式 ： BOOL　AjcFQueEnqTop　(HAJCFQUE hFQue, C_VOP pDat);　---------- 固定長ノード用
BOOL　AjcVQueEnqTop　(HAJCVQUE hVQue, C_VOP pDat, UI lDat);　----- 可変長ノード用

VOP　AjcFQueEnqTopEx (HAJCFQUE hFQue, C_VOP pDat);　---------- 固定長ノード用
VOP　AjcVQueEnqTopEx (HAJCVQUE hVQue, C_VOP pDat, UI lDat);　---- 可変長ノード用

引　数 ： hFQue/hVQue - インスタンスハンドル
pDat　　- ノードに格納するデータのアドレス
lDat　　- ノードに格納するデータのバイト数

説　明 ： 新たなノードを作成し、リストの先頭に追加します。
pDat で示されるデータは、新たに作成したノード内にコピーされます。従って、本関数実行後、pDat で示されるデータを保持している必要はありません。

戻り値 ： TRUE　/ ≠NULL - 成功（追加したノードデータのアドレス）
FALSE / ＝NULL - 失敗

34.1.5. リストの先頭ノード取り出し（Ajc?QueDeque）

形　式 ： UI AjcFQueDeque (HAJCFQUE hFQue, VOP pBuf); ------------- 固定長ノード用
UI AjcVQueDeque (HAJCVQUE hVQue, VOP pBuf, UI lBuf); ----- 可変長ノード用

引　数 ： hFQue/hVQue - インスタンスハンドル
pBuf　　- 取り出したノードのデータを格納（コピー）するバッファのアドレス（不要時はNULL）
lBuf　　- バッファのバイト数

説　明 ： リストの先頭ノードを取り出して、データをpBufで示されるバッファに格納（コピー）します。
取り出したノードのメモリブロックは解放されます。

固定長ノードの場合は、pBufで示されるバッファのバイト数は、インスタンスの初期化時に指定されたノードデータサイズ(DatSize)でなければなりません。

可変長ノードの場合は、実際のノードデータサイズが、lBufで示されるバイト数よりも大きい場合は、lBufで示されるバイト数のデータだけをバッファに格納します。

pBuf=NULLを指定した場合は、データを読み捨てます。

戻り値 ： ≠０：当該ノードデータのバイト数
＝０：ノードなし

34.1.6. リストの先頭ノード取り出し（Ajc?QueDequeEx）

形　式 ： UI AjcVQueDequeEx (HAJCVQUE hVQue, VOP pBuf, UI lBuf); ----- 可変長ノード用

引　数 ： hVQue　　- インスタンスハンドル
pBuf　　- 取り出したノードのデータを格納（コピー）するバッファのアドレス（不要時はNULL）
lBuf　　- バッファのバイト数

説　明 ： リストの先頭ノードを取り出して、データをpBufで示されるバッファに格納（コピー）します。
取り出したノードのメモリブロックは解放されます。

実際のノードデータサイズが、lBufで示されるバイト数よりも大きい場合は、lBufで示されるバイト数のデータだけをバッファに格納します。

pBuf=NULLを指定した場合は、データを読み捨てます。

戻り値 ： ≠０：当該ノードデータのバイト数
＝-1：ノードなし

備　考 ： このＡＰＩは、基本的にAjcVQueDeque()と同じですが、ノード無しの場合は－１を返します。
AjcVQueDeque()では、０を返すため、０バイトデータとノード無しが混同します。
０バイトデータを扱う場合は、AjcVQueDequeEx()を使用してください。

34.1.7. リストの末尾ノード取り出し（Ajc?QueDeqLast）

形　式 ： UI AjcFQueDeqLast (HAJCFQUE hFQue, VOP pBuf); -------------- 固定長ノード用
UI AjcVQueDeqLast (HAJCVQUE hVQue, VOP pBuf, UI lBuf); ------ 可変長ノード用

引　数 ： hFQue/hVQue - インスタンスハンドル

説　明 ： リストの先頭ノードを取り出して、データをpBufで示されるバッファに格納（コピー）します。

取り出したノードのメモリブロックは解放されます。

実際のノードデータサイズが、lBufで示されるバイト数よりも大きい場合は、lBufで示されるバイト数のデータだけをバッファに格納します。

pBuf=NULLを指定した場合は、データを読み捨てます。

戻り値 ： ≠０：当該ノードデータのバイト数
＝-1：ノードなし

34.1.8. リスト中の全ノード破棄（Ajc?QuePurge）

形　式 ： BOOL AjcFQuePurge (HAJCFQUE hFQue); ---- 固定長ノード用
BOOL AjcVQuePurge (HAJCVQUE hVQue); ------ 可変長ノード用

引　数 ： hFQue/hVQue - インスタンスハンドル

説　明 ： 全てのノードを破棄します。

AjcFQueCreate/AjcVQueCreate で cbRemove が指定されている場合は、全てのノードについて、ノードが破棄される直前に、当該ノードデータを通知します。

戻り値 ： TRUE - 成功
FALSE - 失敗

34.1.9. リスト中の個別ノード破棄（Ajc?QueDelAny）

形　式 ： BOOL AjcFQueDelAny (HAJCFQUE hFQue, UX cbp, BOOL (CALLBACK *cbQuery)(VOP pDat, UX cbp)); -------- 固定長ノード用
BOOL AjcVQueDelAny (HAJCVQUE hVQue, UX cbp, BOOL (CALLBACK *cbQuery)(VOP pDat, UI len, UX cbp)); ── 可変長ノード用

引　数 ： hFQue/hVQue - インスタンスハンドル
cbp - コールバックパラメタ
cbQuery - コールバック関数（破棄問い合わせ）

説　明 ： 個別にノードを破棄します。
破棄するか否かは、コールバック関数で問い合わせます。

戻り値 ： TRUE - 成功
FALSE - 失敗

コールバック： コールバック関数の仕様は、以下のとおりです。

cbQuery（ノードデータ破棄の問い合わせ）

形　式 ： BOOL CALLBACK ***cbQuery***(VOP pDat, UX cbp); ---------- 固定長ノード用
BOOL CALLBACK ***cbQuery***(VOP pDat, UI len, UX cbp); ----- 可変長ノード用

引　数 ： pDat - ノードデータのアドレス（ノードのデータ部分を示します）
len - ノードデータのバイト数
cbp - コールバックパラメタ

説　明 ： 通知されたノードを削除する場合は TURE を、削除しない場合は FALSE を返します。

戻り値 ： TRUE - ノードを削除する
FALSE – ノードを削除しない

34.1.10. ノード数取得（Ajc?QueGetCount）

形　式 ： UI AjcFQueGetCount (HAJCFQUE hFQue); ---------- 固定長ノード用
UI AjcVQueGetCount (HAJCVQUE hVQue); ---------- 可変長ノード用

引　数 ： hFQue/hVQue - インスタンスハンドル

説　明 ： 現在の総ノード数を返します。

戻り値 ： 総ノード数（失敗時は０）

34.1.11. ２つのリストを連結する（Ajc?QueMerge）

形　式 ： BOOL　AjcFQueMerge (HAJCFQUE hFQue, HAJCFQUE hSrc);　----- 固定長ノード用
BOOL　AjcVQueMerge (HAJCVQUE hVQue, HAJCVQUE hSrc);　----- 可変長ノード用

引　数 ： hFQue/hVQue - 自インスタンスハンドル
hSrc　　- 連結元リストのインスタンスハンドル

説　明 ： hSrc で示されるリストを、hFQue/hVQue で示される、自リストの後部へ連結します。
hSrc で示されるリストは、空になります。

戻り値 ： TRUE　- 成功
FALSE - 失敗

34.1.12. 先頭ノードデータアドレス取得（Ajc?QueTopNode）

形　式 ： VOP　AjcFQueTopNode (HAJCFQUE hFQue);　---------- 固定長ノード用
VOP　AjcVQueTopNode (HAJCVQUE hVQue, UIP pBytes);　----- 可変長ノード用

引　数 ： hFQue/hVQue - インスタンスハンドル
pBytes　　- ノードデータのバイト数を格納するバッファのアドレス（不要時は、ＮＵＬＬ）

説　明 ： 先頭ノードの情報を取得します。
リスト上にノードがある場合は、先頭ノードのデータ部分のアドレスを返します。
また、可変長ノードの場合は、pBytes で示されるバッファに当該ノードデータのバイト数が設定されます。

リスト上にノードが１つも無い場合は、ＮＵＬＬポインタを返します。

戻り値 ： ≠NULL - 当該ノードデータのアドレス(ノード内のデータ部分を示します)
＝NULL - ノードなし

34.1.13. 次のノードデータアドレス取得（Ajc?QueNextNode）

形　式 ： VOP　AjcFQueNextNode(HAJCFQUE hFQue, C_VOP pCurrentNode);　---------- 固定長ノード用
VOP　AjcVQueNextNode(HAJCVQUE hVQue, C_VOP pCurrentNode, UIP pBytes);　--------- 可変長ノード用

引　数 ： hFQue/hVQue　- インスタンスハンドル
pCurrentNode - 現在のノードデータのアドレス
pBytes　　- ノードデータのバイト数を格納するバッファのアドレス（不要時は、ＮＵＬＬ）

説　明 ： pCurrentNode で指定したノードの次のノードアドレスを取得します。
リスト上に、次ノードがある場合は、当該ノードのデータ部分のアドレスを返します。
また、可変長ノードの場合は、pBytes で示されるバッファに当該ノードデータのバイト数が設定されます。

リスト上に次ノードが無い場合は、ＮＵＬＬポインタを返します。

戻り値 ： ≠NULL - 当該ノードデータのアドレス
＝NULL - ノードなし（終端）／エラー

34.1.14. 全ノードへのポインタ配列生成（Ajc?QueCreatePtrArr）

形　式 ： PCAJCFQUEPTR　AjcFQueCreatePtrArr(HAJCFQUE hFQue, UIP pNum)； ---------- 固定長ノード用
PCAJCVQUEPTR　AjcVQueCreatePtrArr(HAJCVQUE hVQue, UIP pNum)； ---------- 可変長ノード用

引　数 ： hFQue/hVQue　- インスタンスハンドル
pNum　　- 配列要素数を格納するバッファのアドレス（不要時はＮＵＬＬ）

説　明 ： 全ノードへのポインタ配列を生成します。
ポインタ配列は、ノード数＋１の要素が作成され、最終要素には NULL が格納されます。
生成した配列を使用した後は、AjcFQueReleasePtrArr()/AjcVQueReleasePtrArr()により配列を解放してください。

戻り値 ： ≠NULL ： 全ノードへのポインタ配列の先頭アドレス
＝NULL ： エラー

備　考 ： 本 API は、以下の構造体の配列の先頭アドレスを返します。

固定長ノード用
```
typedef struct {
    VOP     pNode; //  ノードデータへのポインタ
} AJCFQUEPTR, *PAJCFQUEPTR;
typedef const AJCFQUEPTR *PCAJCFQUEPTR;
```

可変長ノード用
```
typedef struct {
    VOP     pNode; //  ノードデータへのポインタ
    UI      len;   //  ノードのバイト数
} AJCVQUEPTR, *PAJCVQUEPTR;
typedef const AJCVQUEPTR *PCAJCVQUEPTR;
```

また、「MAJCFQUEPTR」「MAJCVQUEPTR」マクロにより、構造体名とポインタタイプを変更した、同型の構造体を定義できます。

固定長ノード用
```
  MAJCFQUEPTR(MYSTRUCT, PTRTYPE)
+ typedef struct {
+     PTRTYPE  pNode; // ノードデータへのポインタ
+ } MYSTRUCT, *PMYSTRUCT;
+ typedef const MYSTRUCT *PCMYSTRUCT;
```

可変長ノード用
```
  MAJCVQUEPTR(MYSTRUCT, PTRTYPE)
+ typedef struct {
+     PTRTYPE  pNode; // ノードデータへのポインタ
+     UI       len;   //  ノードのバイト数
+ } MYSTRUCT, *PMYSTRUCT;
+ typedef const MYSTRUCT *PCMYSTRUCT;
```

「MAJCFQUEPTR」「MAJCVQUEPTR」マクロを定義し、本 API を呼び出す際に、生成した構造体でキャストできます。

固定長ノード用
```
  PCMYSTRUCT  p;
  p = (PCMYSTRUCT)AjcFQueCreatePtrArr(・・・);
```

可変長ノード用
```
  PCMYSTRUCT  p;
  p = (PCMYSTRUCT)AjcVQueCreatePtrArr(・・・);
```

34.1.15. 全ノードへのポインタ配列解放（Ajc?QueReleasePtrArr）

形　式 ： VO　AjcFQueReleasePtrArr (HAJCFQUE hFQue, PCAJCFQUEPTR pArr)； ---------- 固定長ノード用
VO　AjcVQueReleasePtrArr (HAJCVQUE hVQue, PCAJCVQUEPTR pArr)； ---------- 可変長ノード用

引　数 ： hFQue/hVQue　- インスタンスハンドル
pArr　　- 全ノードへのポインタ配列の先頭アドレス (Ajc?QueCreatePtrArr ()の戻り値)

説　明 ： AjcFQueCreatePtrArr()/AjcVQueCreatePtrArr()で生成した、全ノードへのポインタ配列を解放します。

戻り値 ： なし

34.1.16. マルチスレッドの許可／禁止（Ajc?QueEnableMultiThread）

形　式 ： BOOL　AjcFQueEnableMultiThread(HAJCFQUE hFQue, BOOL fEnable);　---------- 固定長ノード用
BOOL　AjcVQueEnableMultiThread(HAJCVQUE hVQue, BOOL fEnable);　---------- 可変長ノード用

引　数 ： hFQue/hVQue - インスタンスハンドル
fEnable　　- TRUE　: 複数のスレッドからのアクセスを許可
FALSE : 複数のスレッドからのアクセスを禁止

説　明 ： fEnable=TRUE とした場合、複数のスレッドによる、アクセスを可能にします。
この場合、各関数の入り口と出口で、クリティカルセクションによるスレッド間の排他制御を行います。
但し、次の関数は、(fEnable=TRUE としても）マルチスレッドでの排他制御を行いません。

・AjcFQueCreate / AjcVQueCreate（インスタンス生成）
・AjcFQueDelete / AjcVQueDelete（インスタンス消去）
・AjcFQueEnableMultiThread / AjcVQueEnableMultiThread（本関数）

fEnable=FALSE（デフォルト）とした場合は、マルチスレッドでの排他制御を行いません。

戻り値 ： 前回の許可／禁止状態

34.2. サンプルプログラム

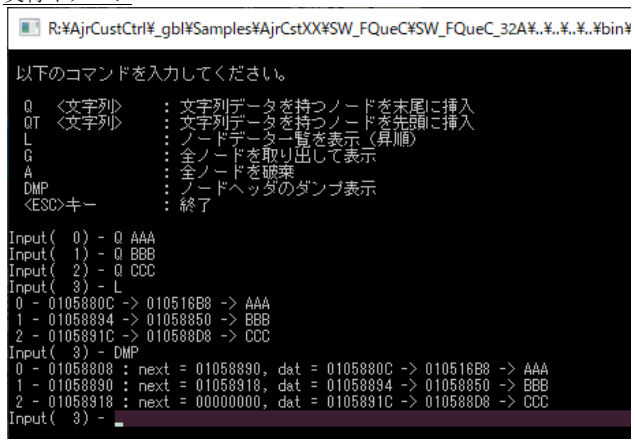
34.2.1. SW_FQue（線形リスト - 固定長データ）

以下のサンプルプログラムは、コマンド入力により、双方向リストのノード挿入や削除等を行います。
ノードデータは、入力した文字列へのポインタです。文字列は動的に確保したメモリに格納します。
コマンドの形式は以下の通りです。 ESC キーを押すとプログラムを終了します。

#	コマンド形式	内容
1	Q 〈文字列〉	文字列データを持つノードを末尾に挿入
2	QT 〈文字列〉	文字列データを持つノードを先頭に挿入
3	L	ノードデータ一覧を表示（昇順）
4	G	全ノードを取り出して表示
5	A	全ノードを破棄
6	S 〈文字列〉	指定した文字列を含むノードを破棄します
7	DMP	ノードヘッダのダンプ表示

※＜文字列＞に空白を含む場合はダブルクォート(")で囲みます（ex. "A B C"）

実行イメージ




```
 1 : //
 2 : //  SW_FQueC.c
 3 : //
 4 : #include    <AjrCstXX.h>
 5 : #include    <stdio.h>
 6 : #include    <string.h>
 7 : #include    <tchar.h>
 8 :
 9 : static  VOP     GetNodeAddr(UI ix);
10 :
11 : //-------------------------------------------------------------------------------------------//
12 : //  作業領域                                                                                 //
13 : //-------------------------------------------------------------------------------------------//
14 : static  HAJCFQUE    hQue;
15 :
16 : //-------------------------------------------------------------------------------------------//
17 : //  ノード削除通知                                                                           //
18 : //-------------------------------------------------------------------------------------------//
19 : static VO CALLBACK cbRemove(VOP pDat, UX cbp)
20 : {
21 :     AjcPrintF(TEXT("ノード削除 %p : %s¥n"), pDat, *((UTP*)pDat));
22 :     AJCFREE(*((UTP*)pDat));
23 : }
24 : //-------------------------------------------------------------------------------------------//
25 : //  ノード削除の問い合わせ                                                                   //
26 : //-------------------------------------------------------------------------------------------//
27 : static BOOL CALLBACK cbDelAny(VOP pDat, UX cbp)
28 : {
29 :     return ((MAjcStrStr(*((UTP*)pDat), (UTP)cbp)) != NULL);
30 : }
31 : //-------------------------------------------------------------------------------------------//
32 : //  テキスト入力通知                                                                         //
33 : //-------------------------------------------------------------------------------------------//
34 : static  VO CALLBACK cbNtcInput(int argc, UT *argv[], C_UTP pTxt, UX cbp)
35 : {
36 :     UI      seq;
37 :
38 :     if (argc >= 1) {
39 :         //  文字列ポインタを持つノードを末尾に挿入
40 :         if      (_tcsicmp(argv[0], TEXT("Q")) == 0) {
41 :             if (argc == 2) {
42 :                 UI      stl = ((UI)_tcslen(argv[1]) + 1);
```

```
 43 :                 UI      len = stl * (sizeof(UT));
 44 :                 UTP     p   = (UTP)AJCMEM(len);
 45 :                 _tcscpy_s(p, stl, argv[1]);
 46 :                 AjcFQueEnque(hQue, &p);
 47 :             }
 48 :             else AjcPrintF(TEXT("*** Invalid parameter.¥n"));
 49 :         }
 50 :         //  文字列ポインタを持つノードを先頭に挿入
 51 :         else if (_tcsicmp(argv[0], TEXT("QT")) == 0) {
 52 :             if (argc == 2) {
 53 :                 UI      stl = ((UI)_tcslen(argv[1]) + 1);
 54 :                 UI      len = stl * (sizeof(UT));
 55 :                 UTP     p   = (UTP)AJCMEM(len);
 56 :                 _tcscpy_s(p, stl, argv[1]);
 57 :                 AjcFQueEnqTop(hQue, &p);
 58 :             }
 59 :             else AjcPrintF(TEXT("*** Invalid parameter.¥n"));
 60 :         }
 61 :         //  ノードデータ一覧を表示 (昇順)
 62 :         else if (_tcsicmp(argv[0], TEXT("L")) == 0) {
 63 :             UTP     *p;
 64 :             seq = 0;
 65 :             if (p = (UTP*)AjcFQueTopNode(hQue)) {
 66 :                 do {
 67 :                     AjcPrintF(TEXT("%2d - %p -> %p -> %s¥n"), seq++, p, *p, *p);
 68 :                 } while(p = (UTP*)AjcFQueNextNode(hQue, p));
 69 :             }
 70 :         }
 71 :         //  全ノードを取り出して表示
 72 :         else if (_tcsicmp(argv[0], TEXT("G")) == 0) {
 73 :             UTP     p;
 74 :             seq = 0;
 75 :             while ((AjcFQueDeque(hQue, &p)) != 0) {
 76 :                 AjcPrintF(TEXT("%2d - %p -> %s¥n"), seq++, p, p);
 77 :                 AJCFREE(p);
 78 :             }
 79 :         }
 80 :         //  最古ノードを取り出して表示
 81 :         else if (_tcsicmp(argv[0], TEXT("T"))  == 0) {
 82 :             UTP     p;
 83 :             if ((AjcFQueDeqLast(hQue, &p)) != -1) {
 84 :                 AjcPrintF(TEXT(" Deque last node : %s¥n"), p);
 85 :             }
 86 :         }
 87 :         //  全ノードを破棄
 88 :         else if (_tcsicmp(argv[0], TEXT("A"))  == 0) {
 89 :             AjcFQuePurge(hQue);
 90 :         }
 91 :         //  文字列を含むノードを破棄
 92 :         else if (_tcsicmp(argv[0], TEXT("S"))  == 0) {
 93 :             if (argc == 2) {
 94 :                 UI      stl = ((UI)_tcslen(argv[1]) + 1);
 95 :                 UI      len = stl * (sizeof(UT));
 96 :                 UTP     p   = (UTP)AJCMEM(len);
 97 :                 _tcscpy_s(p, stl, argv[1]);
 98 :                 AjcFQueDelAny(hQue, (UX)p, cbDelAny);
 99 :             }
100 :             else AjcPrintF(TEXT("*** Invalid parameter.¥n"));
101 :         }
102 :         //  ノードヘッダのダンプ表示
103 :         else if (_tcsicmp(argv[0], TEXT("DMP"))  == 0) {
104 :             UTP     *p;
105 :             seq = 0;
106 :             if (p = (UTP*)AjcFQueTopNode(hQue)) {
107 :                 do {
108 :                     PAJCFQNODE pH = ((PAJCFQNODE)p) - 1;
109 :                     AjcPrintF(TEXT("%2d - %p : next = %p, dat = %p -> %p -> %s¥n"), seq++, pH, pH->pNxt, p, *p, *p);
110 :                 } while(p = (UTP*)AjcFQueNextNode(hQue, p));
111 :             }
112 :         }
113 :         //  その他
114 :         else {
115 :             AjcPrintF(TEXT("*** Invalid command.¥n"));
116 :         }
117 :     }
118 : }
119 : //-----------------------------------------------------------------------------------------//
120 : //  m a i n                                                                                //
121 : //-----------------------------------------------------------------------------------------//
122 : int  AjcMain(int argc, UTP argv[])
123 : {
124 :     AjcSetStdoutMode();
125 :
126 :     AjcSetConsoleBufSize(128, 64);
127 :     AjcSetConsoleWndRect(0, 0, 127, 30);
128 :
129 :     AjcPrintF(TEXT("¥n 以下のコマンドを入力してください。¥n¥n"));
130 :     AjcPrintF(TEXT("  Q   〈文字列〉     : 文字列データを持つノードを末尾に挿入¥n"));
131 :     AjcPrintF(TEXT("  QT  〈文字列〉     : 文字列データを持つノードを先頭に挿入¥n"));
132 :     AjcPrintF(TEXT("  L                 : ノードデータ一覧を表示 (昇順) ¥n"));
```

```
133 :     AjcPrintF(TEXT("  G               : 全ノードを取り出して表示¥n"));
134 :     AjcPrintF(TEXT("  T               : 末尾ノードを取り出して表示¥n"));
135 :     AjcPrintF(TEXT("  A               : 全ノードを破棄¥n"));
136 :     AjcPrintF(TEXT("  S   <文字列>    : 文字列を含むノードを削除¥n"));
137 :     AjcPrintF(TEXT("  DMP             : ノードヘッダのダンプ表示¥n"));
138 :     AjcPrintF(TEXT("  <ESC>キー       : 終了¥n"));
139 :     AjcPrintF(TEXT("¥n"));
140 :
141 :     //  線形リストインスタンス生成
142 :     hQue = AjcFQueCreate(sizeof(VOP), 0, cbRemove);
143 :
144 :     AjcPrintF(TEXT("Input(%3u) - "), AjcFQueGetCount(hQue));
145 :     while (AjcConInputByNtc(256, 0, cbNtcInput)) {
146 :         AjcPrintF(TEXT("Input(%3u) - "), AjcFQueGetCount(hQue));
147 :     }
148 :     //  線形リストインスタンス消去
149 :     AjcFQueDelete(hQue);
150 :
151 :     return 0;
152 : }
```

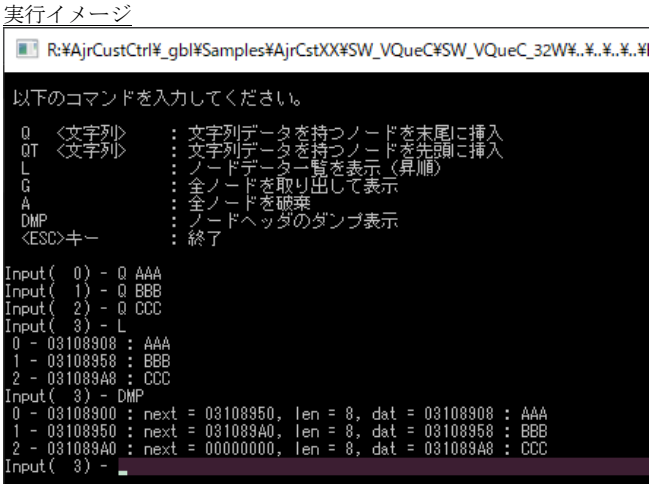
34.2.2. SW_VQueC（線形リスト – 可変長データ）

以下のサンプルプログラムは、コマンド入力により、双方向リストのノード挿入や削除等を行います。
ノードデータは、入力した文字列です。
コマンドの形式は以下の通りです。 ESC キーを押すとプログラムを終了します。

#	コマンド形式	内容
1	Q 〈文字列〉	文字列データを持つノードを末尾に挿入
2	QT 〈文字列〉	文字列データを持つノードを先頭に挿入
3	L	ノードデーター覧を表示（昇順）
4	G	全ノードを取り出して表示
5	A	全ノードを破棄
6	S 〈文字列〉	指定した文字列を含むノードを破棄します
7	DMP	ノードヘッダのダンプ表示

※＜文字列＞に空白を含む場合はダブルクォート(“)で囲みます（ex. “A B C”）

実行イメージ




```
 1 : //
 2 : //  SW_VQueC.c
 3 : //
 4 : #include    <AjrCstXX.h>
 5 : #include    <stdio.h>
 6 : #include    <string.h>
 7 : #include    <tchar.h>
 8 :
 9 : //-------------------------------------------------------------------------------------------//
10 : //  作業領域                                                                                 //
11 : //-------------------------------------------------------------------------------------------//
12 : static  HAJCVQUE    hQue;
13 :
14 : //-------------------------------------------------------------------------------------------//
15 : //  ノード削除通知                                                                           //
16 : //-------------------------------------------------------------------------------------------//
17 : static VO CALLBACK cbRemove(VOP pDat, UI len, UX cbp)
18 : {
19 :     AjcPrintF(TEXT("ノード削除 %p : %s, len = %d¥n"), pDat, pDat, len);
20 : }
21 : //-------------------------------------------------------------------------------------------//
22 : //  ノード削除の問い合わせ                                                                   //
23 : //-------------------------------------------------------------------------------------------//
24 : static BOOL CALLBACK cbDelAny(VOP pDat, UI len, UX cbp)
25 : {
26 :     return ((MAjcStrStr(pDat, (UTP)cbp)) != NULL);
27 : }
28 : //-------------------------------------------------------------------------------------------//
29 : //  テキスト入力通知                                                                         //
30 : //-------------------------------------------------------------------------------------------//
31 : static  VO CALLBACK cbNtcInput(int argc, UT *argv[], C_UTP pTxt, UX cbp)
32 : {
33 :     VOP pDat;
34 :     UI  bytes, seq;
35 :
36 :     if (argc >= 1) {
37 :         //  文字列データを持つノードを末尾に挿入
38 :         if      (_tcsicmp(argv[0], TEXT("Q")) == 0) {
39 :             if (argc == 2) AjcVQueEnque(hQue, argv[1], ((UI)_tcslen(argv[1]) + 1) * (sizeof(UT)));
40 :             else           AjcPrintF(TEXT("*** Invalid parameter.¥n"));
41 :         }
```

```
        //  文字列データを持つノードを先頭に挿入
        else if (_tcsicmp(argv[0], TEXT("QT")) == 0) {
            if (argc == 2) AjcVQueEnqTop(hQue, argv[1], ((UI)_tcslen(argv[1]) + 1) * (sizeof(UT)));
            else           AjcPrintF(TEXT("*** Invalid parameter.¥n"));
        }
        //  ノードデータ一覧を表
        else if (_tcsicmp(argv[0], TEXT("L")) == 0) {
            seq = 0;
            if (pDat = (UTP)AjcVQueTopNode(hQue, &bytes)) {
                do {
                    AjcPrintF(TEXT("%2d - %p : %s¥n"), seq++, pDat, pDat);
                } while(pDat = AjcVQueNextNode(hQue, pDat, &bytes));
            }
        }
        //  全ノードを取り出して表示
        else if (_tcsicmp(argv[0], TEXT("G"))  == 0) {
            UT  buf[256];
            seq = 0;
            while ((AjcVQueDequeEx(hQue, &buf, sizeof buf)) != -1) {
                AjcPrintF(TEXT("%2d - %s¥n"), seq++, buf);
            }
        }
        //  最古ノードを取り出して表示
        else if (_tcsicmp(argv[0], TEXT("T"))  == 0) {
            UT  buf[256];
            if ((AjcVQueDeqLast(hQue, &buf, sizeof buf)) != -1) {
                AjcPrintF(TEXT(" Deque last node : %s¥n"), buf);
            }
        }
        //  文字列を含むノードを破棄
        else if (_tcsicmp(argv[0], TEXT("S"))  == 0) {
            if (argc == 2) {
                AjcVQueDelAny(hQue, (UX)argv[1], cbDelAny);
            }
            else AjcPrintF(TEXT("*** Invalid parameter.¥n"));
        }
        //  全ノードを破棄
        else if (_tcsicmp(argv[0], TEXT("A"))  == 0) {
            AjcVQuePurge(hQue);
        }
        //  ノードヘッダのダンプ表示
        else if (_tcsicmp(argv[0], TEXT("DMP"))  == 0) {
            seq = 0;
            if (pDat = (UTP)AjcVQueTopNode(hQue, &bytes)) {
                do {
                    PAJCVQNODE pH = ((PAJCVQNODE)pDat) - 1;
                    AjcPrintF(TEXT("%2d - %p : next = %p, len = %d, dat = %p : %s¥n"), seq++, pH, pH->pNxt, pH->len, pDat,
pDat);
                } while(pDat = AjcVQueNextNode(hQue, pDat, &bytes));
            }
        }
        //  その他
        else {
            AjcPrintF(TEXT("*** Invalid command.¥n"));
        }
    }
}
//-----------------------------------------------------------------------------------------//
//  m a i n                                                                                //
//-----------------------------------------------------------------------------------------//
int  AjcMain(int argc, UTP argv[])
{
    AjcSetStdoutMode();

    AjcSetConsoleBufSize(128, 64);
    AjcSetConsoleWndRect(0, 0, 127, 30);

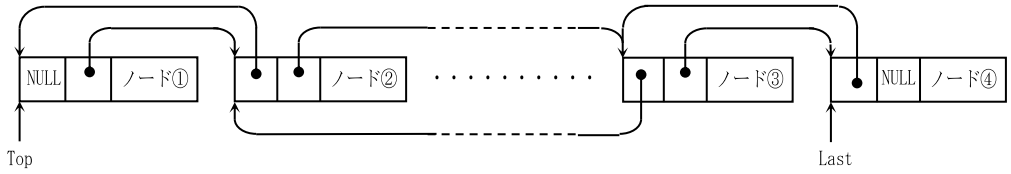
    AjcPrintF(TEXT("¥n 以下のコマンドを入力してください。¥n¥n"));
    AjcPrintF(TEXT("  Q   <文字列>     : 文字列データを持つノードを末尾に挿入¥n"));
    AjcPrintF(TEXT("  QT  <文字列>     : 文字列データを持つノードを先頭に挿入¥n"));
    AjcPrintF(TEXT("  L                : ノードデータ一覧を表示 (昇順) ¥n"));
    AjcPrintF(TEXT("  G                : 全ノードを取り出して表示¥n"));
    AjcPrintF(TEXT("  T                : 最古ノードを取り出して表示¥n"));
    AjcPrintF(TEXT("  A                : 全ノードを破棄¥n"));
    AjcPrintF(TEXT("  S   <文字列>     : 文字列を含むノードを削除¥n"));
    AjcPrintF(TEXT("  DMP              : ノードヘッダのダンプ表示¥n"));
    AjcPrintF(TEXT("  <ESC>キー        : 終了¥n"));
    AjcPrintF(TEXT("¥n"));

    //  線形リストインスタンス生成
```

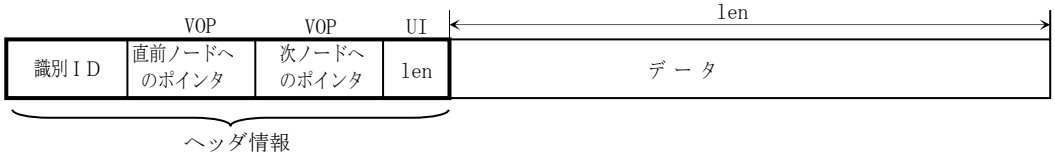
```
121 :     hQue = AjcVQueCreate(0, cbRemove);
122 :
123 :     AjcPrintF(TEXT("Input(%3u) - "), AjcVQueGetCount(hQue));
124 :
125 :     while (AjcConInputByNtc(32, 0, cbNtcInput)) {
126 :         AjcPrintF(TEXT("Input(%3u) - "), AjcVQueGetCount(hQue));
127 :     }
128 :     //  線形リストインスタンス消去
129 :     AjcVQueDelete(hQue);
130 :
131 :     return 0;
132 : }
```

35. 双方向リスト制御

双方向リストとは、各ノード（メモリブロック）が、次のノードを示すポインタと直前のノードを示すポインタを持ち、次々と鎖状に連なっているノードの集まりを意味します。先頭ノードの直前ノードへのポインタはＮＵＬＬで、最後のノードの次のノードへのポインタはＮＵＬＬです。



各ノードの先頭には、以下のヘッダ情報が付加されます。



35.1. サポートＡＰＩ

双方向リスト制御のサポートＡＰＩ一覧を以下に示します。

#	内容	関数名	備　考
1	インスタンス生成	AjcXQueCreate	
2	インスタンス消去	AjcXQueDelete	
3	ノードを末尾へ追加	AjcXQueEnque	
4	ノードを先頭へ挿入	AjcXQueEnqTop	
5	ノードを指定ノードの直前に挿入	AjcXQueInsert	
6	指定ノード削除	AjcXQueRemove	
7	全ノード破棄	AjcXQuePurge	
8	個別ノード破棄	AjcXQueDelAny	
9	先頭ノード取り出し	AjcXQueDeque	
10	ノード数取得	AjcXQueGetCount	
11	２つのリストを連結	AjcXQueMerge	
12	先頭ノードアドレス取得	AjcXQueTopNode	
13	末尾ノードアドレス取得	AjcXQueLastNode	
14	次のノードアドレス取得	AjcXQueNextNode	
15	直前のノードアドレス取得	AjcXQuePrevNode	
17	全ノードへのポインタ配列生成	AjcXQueCreatePtrArr	
18	全ノードへのポインタ配列解放	AjcXQueReleasePtrArr	
19	マルチスレッド許可／禁止	AjcXQueEnableMultiThread	

35.1.1. インスタンス生成（AjcXQueCreate）

形　式 ： HAJCXQUE　AjcXQueCreate(UX cbp, VO (CALLBACK *cbRemove)(VOP pDat, UI len, UX cbp));

引　数 ： cbp　　- コールバックパラメタ
cbRemove - ノード消去通知用コールバック関数のアドレス（不要時は NULL）

説　明 ： リスト機能のインスタンスを生成／初期化します。
「cbRemove」は、AjcXQuePurge/ AjcXQueDelAny や AjcXQueDelete により、消去されるノードを通知するためのコールバック関数を指定します。
ノード消去時に、何も操作する必要が無い場合は、cbRemove=NULL を指定します。

戻り値 ： ≠NULL - 成功　（インスタンスハンドルを返します）
＝NULL - 失敗

コールバック： コールバック関数の仕様は、以下のとおりです。

cbRemove（消去されるノードデータ通知）

形　式 ： VO CALLBACK ***cbRemove***(VOP pDat, UI len, UX cbp);　----- 可変長ノード用

引　数 ： pDat - ノードデータのアドレス（ノードのデータ部分を示します）
len　- ノードデータのバイト数
cbp　- コールバックパラメタ

説　明 ： 当該ノードデータに関連した後処理（リソースの開放等）を行います。

戻り値 ： なし

35.1.2. インスタンス消去（AjcXQueDelete）

形　式 ： BOOL AjcXQueDelete (HAJCXQUE hXQue);

引　数 ： hXQue - インスタンスハンドル

説　明 ： 全ノードを削除（開放）し、AjcXQueCreate で動的に確保した、リソースを解放します。

AjcXQueCreate で cbRemove が指定されている場合は、全てのノードについて、ノードが破棄される直前に、当該ノードデータを通知します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

35.1.3. リストの末尾へノードを追加（AjcXQueEnque[Ex]）

形　式 ： VOP　AjcXQueEnque (HAJCFXQU hXQue, C_VOP pDat, UI len);

引　数 ： hXQue - インスタンスハンドル
pDat　- ノードに格納するデータのアドレス（NULL の場合は、0x00 で埋めた空のノードを作成）
lDat　- ノードに格納するデータのバイト数

説　明 ： 新たなノードを作成し、リストの末尾に追加します。
pDat で示されるデータは、新たに作成したノード内にコピーされます。従って、本関数実行後、pDat で示されるデータを保持している必要はありません。

戻り値 ： ≠NULL - 成功（追加したノードデータのアドレス）
＝NULL - 失敗

35.1.4. リストの先頭へノードを追加（AjcXQueEnqTop）

形　式 ： VOP　AjcXQueEnqTop (HAJCXQUE hXQue, C_VOP pDat, UI lDat);

引　数 ： hXQue - インスタンスハンドル
pDat　- ノードに格納するデータのアドレス（NULL の場合は、0x00 で埋めた空のノードを作成）
lDat　- ノードに格納するデータのバイト数

説　明 ： 新たなノードを作成し、リストの先頭に追加します。
pDat で示されるデータは、新たに作成したノード内にコピーされます。従って、本関数実行後、pDat で示されるデータを保持している必要はありません。

戻り値 ： TRUE　/ ≠NULL - 成功（追加したノードのアドレス）
FALSE / ＝NULL - 失敗

35.1.5. ノードを指定ノードの直前に挿入（AjcXQueInsert）

形　式 ： UI　AjcXQueInsert (HAJCXQUE hXQue, C_VOP pDat, UI lDat, C_VOP pIns);

引　数 ： hXQue - インスタンスハンドル
pDat　- 挿入するノードデータのアドレス（NULL の場合は、0x00 で埋めた空のノードを作成）
lDat　- 挿入するノードデータのバイト数
pIns　- ノード挿入位置（挿入位置を示すノードのデータアドレス，NULL の場合は末尾に挿入）

説　明 ： 新たなノードを作成し、pIns で示すノードの直前にノードを挿入します。

pBuf=NULL を指定した場合は、データを読み捨てます。

戻り値 ： ≠０：当該ノードデータのバイト数
＝０：ノードなし

35.1.6. 指定ノード削除（AjcXQueRemove）

形　式 ： UI　AjcXQueRemove (HAJCXQUE hXQue, C_VOP pDat);

引　数 ： hXQue - インスタンスハンドル
pDat　- 削除するノードデータのアドレス

説　明 ： pDat で示すノードを削除します。
AjcXQueCreate で cbRemove が指定されている場合は、全てのノードについて、ノードが削除される直前に、当該ノードデータを通知します。

戻り値 ： ≠-1：削除したノードデータのバイト数
＝-1：ノードなし

35.1.7. リスト中の全ノード破棄（AjcXQuePurge）

形　式 ： BOOL　AjcXQuePurge (HAJCXQUE hXQue);

引　数 ： hXQue - インスタンスハンドル

説　明 ： 全てのノードを破棄します。
AjcXQueCreate で cbRemove が指定されている場合は、全てのノードについて、ノードが破棄される直前に、当該ノードデータを通知します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

35.1.8. リスト中の個別ノード破棄（AjcXQueDelAny）

形　式 ： BOOL AjcXQueDelAny (HAJCXQUE hXQue, UX cbp, BOOL (CALLBACK *cbQuery)(VOP pDat, UI len, UX cbp));

引　数 ： hFQue/hVQue - インスタンスハンドル
cbp - コールバックパラメタ
cbQuery - コールバック関数（削除問い合わせ）

説　明 ： 個別にノードを破棄します。
破棄するか否かは、コールバック関数で問い合わせます。

戻り値 ： TRUE - 成功
FALSE - 失敗

コールバック： コールバック関数の仕様は、以下のとおりです。

cbQuery（ノードデータ破棄の問い合わせ）

形　式 ： BOOL CALLBACK ***cbQuery***(VOP pDat, UI len, UX cbp);

引　数 ： pDat - ノードデータのアドレス（ノードのデータ部分を示します）
len - ノードデータのバイト数
cbp - コールバックパラメタ

説　明 ： 通知されたノードを削除する場合は TURE を、削除しない場合は FALSE を返します。

戻り値 ： TRUE - ノードを削除する
FALSE – ノードを削除しない

35.1.9. 先頭ノード取り出し（AjcXQueDequeEx）

形　式 ： UI AjcXQueDeque (HAJCXQUE hXQue, VOP pBuf, UI lBuf);

引　数 ： hXQue - インスタンスハンドル
pBuf - 取り出したノードのデータを格納（コピー）するバッファのアドレス（不要時は NULL）
lBuf - バッファのバイト数

説　明 ： リストの先頭ノードを取り出して、データを pBuf で示されるバッファに格納（コピー）します。
取り出したノードのメモリブロックは解放されます。

実際のノードデータサイズが、lBuf で示されるバイト数よりも大きい場合は、lBuf で示されるバイト数のデータだけをバッファに格納します。

pBuf=NULL を指定した場合は、データを読み捨てます。

戻り値 ： ≠０：当該ノードデータのバイト数
＝-1：ノードなし

35.1.10. ノード数取得（AjcXQueGetCount）

形　式 ： UI AjcXQueGetCount (HAJCXQUE hXQue);

引　数 ： hXQue - インスタンスハンドル

説　明 ： 現在の総ノード数を返します。

戻り値 ： 総ノード数（失敗時は０）

35.1.11. ２つのリストを連結する（AjcXQueMerge）

形　式　：　BOOL　AjcXQueMerge (HAJCXQUE hXQue, HAJCXQUE hSrc);

引　数　：　hXQue - 自インスタンスハンドル
hSrc　- 連結元リストのインスタンスハンドル

説　明　：　hSrc で示されるリストを、hFXqu/hXQue で示される、自リストの後部へ連結します。
hSrc で示されるリストは、空になります。

戻り値　：　TRUE　- 成功
FALSE - 失敗

35.1.12. 先頭ノードデータアドレス取得（AjcXQueTopNode）

形　式　：　VOP　AjcXQueTopNode (HAJCXQUE hXQue, UIP pBytes);

引　数　：　hhXQue - インスタンスハンドル
pBytes - ノードデータのバイト数を格納するバッファのアドレス（不要時は、ＮＵＬＬ）

説　明　：　先頭ノードの情報を取得します。
リスト上にノードがある場合は、先頭ノードのデータ部分のアドレスを返します。
pBytes で示されるバッファには、当該ノードデータのバイト数が設定されます。

リスト上にノードが１つも無い場合は、ＮＵＬＬポインタを返します。

戻り値　：　≠NULL - 当該ノードデータのアドレス(ノード内のデータ部分を示します)
＝NULL - ノードなし

35.1.13. 末尾ノードデータアドレス取得（AjcXQueLastNode）

形　式　：　VOP　AjcXQueLastNode (HAJCXQUE hXQue, UIP pBytes);

引　数　：　hhXQue - インスタンスハンドル
pBytes - ノードデータのバイト数を格納するバッファのアドレス（不要時は、ＮＵＬＬ）

説　明　：　末尾ノードの情報を取得します。
リスト上にノードがある場合は、末尾ノードのデータ部分のアドレスを返します。
pBytes で示されるバッファには、当該ノードデータのバイト数が設定されます。

リスト上にノードが１つも無い場合は、ＮＵＬＬポインタを返します。

戻り値　：　≠NULL - 当該ノードデータのアドレス(ノード内のデータ部分を示します)
＝NULL - ノードなし

35.1.14. 次のノードデータアドレス取得（AjcXQueNextNode）

形　式 ： VOP　AjcXQueNextNode(HAJCXQUE hXQue, C_VOP pCurrentNode, UIP pBytes);

引　数 ： hXQue　　- インスタンスハンドル
pCurrentNode - 現在のノードデータのアドレス
pBytes　　- ノードデータのバイト数を格納するバッファのアドレス（不要時は、ＮＵＬＬ）

説　明 ： pCurrentNode で指定したノードの次のノードアドレスを取得します。
リスト上に、次ノードがある場合は、当該ノードのデータ部分のアドレスを返します。
また、可変長ノードの場合は、pBytes で示されるバッファに当該ノードデータのバイト数が設定されます。

リスト上に次のノードが無い場合は、ＮＵＬＬポインタを返します。

戻り値 ： ≠NULL – 次のノードデータのアドレス
＝NULL – 次のノードなし／エラー

35.1.15. 直前のノードデータアドレス取得（AjcXQuePrevNode）

形　式 ： VOP　AjcXQuePrevNode(HAJCXQUE hXQue, C_VOP pCurrentNode, UIP pBytes);

引　数 ： hXQue　　- インスタンスハンドル
pCurrentNode - 現在のノードデータのアドレス
pBytes　　- ノードデータのバイト数を格納するバッファのアドレス（不要時は、ＮＵＬＬ）

説　明 ： pCurrentNode で指定したノードの直前のノードアドレスを取得します。
リスト上に、直前のノードがある場合は、当該ノードのデータ部分のアドレスを返します。
pBytes で示されるバッファには、当該ノードデータのバイト数が設定されます。

リスト上に直前のノードが無い場合は、ＮＵＬＬポインタを返します。

戻り値 ： ≠NULL – 直前のノードデータのアドレス
＝NULL – 直前のノードなし／エラー

35.1.16. 全ノードへのポインタ配列生成（AjcXQueCreatePtrArr）

形　式 ： PCAJCVXQUPTR　AjcXQueCreatePtrArr(HAJCXQUE hXQue, UIP pNum);

引　数 ： hXQue　－ インスタンスハンドル
pNum　－ 配列要素数を格納するバッファのアドレス（不要時はＮＵＬＬ）

説　明 ： 全ノードへのポインタ配列を生成します。
ポインタ配列は、ノード数＋１の要素が作成され、最終要素には NULL が格納されます。
生成した配列を使用した後は、AjcXQueReleasePtrArr()により配列を解放してください。

戻り値 ： ≠NULL ： 全ノードへのポインタ配列の先頭アドレス
＝NULL ： エラー

備　考 ： 本 API は、以下の構造体の配列の先頭アドレスを返します。

```
typedef struct {
    VOP     pNode; //  ノードデータへのポインタ
    UI      len;   //  ノードのバイト数
} AJCXQUEPTR, *P AJCXQUEPTR;
typedef const AJCXQUEPTR *PC AJCXQUEPTR;
```

また、「MAJCXQUEPTR」マクロにより、構造体名とポインタタイプを変更した、同型の構造体を定義できます。

```
  MAJCXQUEPTR(MYSTRUCT, PTRTYPE)
+ typedef struct {
+     PTRTYPE  pNode; // ノードデータへのポインタ
+     UI       len;   //  ノードのバイト数
+ } MYSTRUCT, *PMYSTRUCT;
+ typedef const MYSTRUCT *PCMYSTRUCT;
```

「MAJCXQUEPTR」マクロを定義し、本 API を呼び出す際に、生成した構造体でキャストできます。

```
  PCMYSTRUCT  p;
  p = (PCMYSTRUCT)AjcXQueCreatePtrArr(・・・);
```

35.1.17. 全ノードへのポインタ配列解放（AjcXQueReleasePtrArr）

形　式 ： VO　AjcXQueReleasePtrArr (HAJCXQUE hXQue, PCAJCVXQUPTR pArr);

引　数 ： hXQue　－ インスタンスハンドル
pArr　－ 全ノードへのポインタ配列の先頭アドレス（Ajc?XquCreatePtrArr ()の戻り値）

説　明 ： AjcXQueCreatePtrArr()で生成した、全ノードへのポインタ配列を解放します。

戻り値 ： なし

35.1.18. マルチスレッドの許可／禁止（AjcXQueEnableMultiThread）

形　式　：　BOOL　AjcXQueEnableMultiThread(HAJCXQUE hXQue, BOOL fEnable);

引　数　：　hXQue　- インスタンスハンドル
fEnable- TRUE　：複数のスレッドからのアクセスを許可
FALSE：複数のスレッドからのアクセスを禁止

説　明　：　fEnable=TRUE とした場合、複数のスレッドによる、アクセスを可能にします。
この場合、各関数の入り口と出口で、クリティカルセクションによるスレッド間の排他制御を行います。
但し、次の関数は、(fEnable=TRUE としても）マルチスレッドでの排他制御を行いません。

・AjcXQueCreate（インスタンス生成）
・AjcXQueDelete（インスタンス消去）
・AjcXQueEnableMultiThread（本関数）

fEnable=FALSE（デフォルト）とした場合は、マルチスレッドでの排他制御を行いません。

戻り値　：　前回の許可／禁止状態

35.2. サンプルプログラム

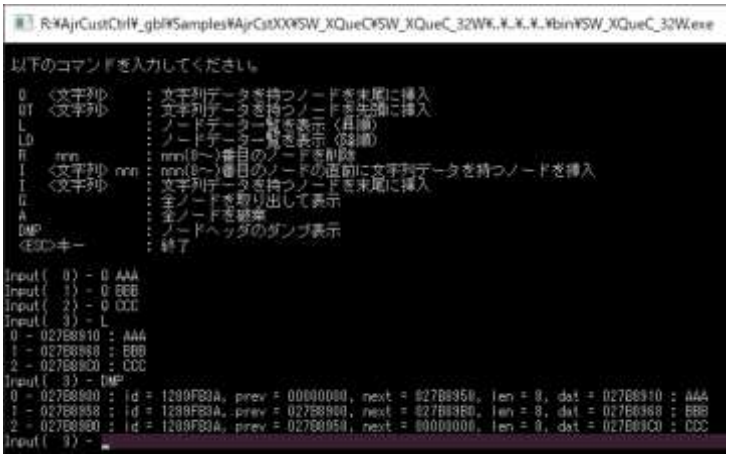
35.2.1. SW_XQueC（双方向リスト）

以下のサンプルプログラムは、コマンド入力により、双方向リストのノード挿入や削除等を行います。
ノードデータは、入力した文字列です。
コマンドの形式は以下の通りです。 ESC キーを押すとプログラムを終了します。

#	コマンド形式	内容
1	Q 〈文字列〉	文字列データを持つノードを末尾に挿入
2	QT 〈文字列〉	文字列データを持つノードを先頭に挿入
3	L	ノードデータ一覧を表示（昇順）
4	LD	ノードデータ一覧を表示（降順）
5	R nnn	nnn(0～)番目のノードを削除
6	I 〈文字列〉 nnn	nnn(0～)番目のノードの直前に文字列データを持つノードを挿入
7	I 〈文字列〉	文字列データを持つノードを末尾に挿入
8	G	全ノードを取り出して表示
9	A	全ノードを破棄
10	S 〈文字列〉	指定した文字列を含むノードを破棄します
11	DMP	ノードヘッダのダンプ表示

※＜文字列＞に空白を含む場合はダブルクォート(")で囲みます（ex. "A B C"）

実行イメージ




```
 1 : //
 2 : //  SW_XQueC.c
 3 : //
 4 : #include    <AjrCstXX.h>
 5 : #include    <stdio.h>
 6 : #include    <string.h>
 7 : #include    <tchar.h>
 8 :
 9 : static  VOP     GetNodeAddr(UI ix);
10 :
11 : //-------------------------------------------------------------------------------------//
12 : //  作業領域                                                                           //
13 : //-------------------------------------------------------------------------------------//
14 : static  HAJCXQUE    hQue;
15 :
16 : //-------------------------------------------------------------------------------------//
17 : //  ノード削除通知                                                                     //
18 : //-------------------------------------------------------------------------------------//
19 : static VO CALLBACK cbRemove(VOP pDat, UI len, UX cbp)
20 : {
21 :     AjcPrintF(TEXT("ノード削除 %p : %s, len = %d¥n"), pDat, pDat, len);
22 : }
23 : //-------------------------------------------------------------------------------------//
24 : //  ノード削除の問い合わせ                                                             //
25 : //-------------------------------------------------------------------------------------//
26 : static BOOL CALLBACK cbDelAny(VOP pDat, UI len, UX cbp)
27 : {
28 :     return ((MAjcStrStr((UTP)pDat, (UTP)cbp)) != NULL);
29 : }
30 : //-------------------------------------------------------------------------------------//
31 : //  テキスト入力通知                                                                   //
```

```
     32 : //-------------------------------------------------------------------------------------//
     33 : static  VO CALLBACK cbNtcInput(int argc, UT *argv[], C_UTP pTxt, UX cbp)
     34 : {
     35 :     VOP pDat;
     36 :     UI  bytes, seq;
     37 :
     38 :     if (argc >= 1) {
     39 :         //  文字列データを持つノードを末尾に挿入
     40 :         if      (_tcsicmp(argv[0], TEXT("Q")) == 0) {
     41 :             if (argc == 2) AjcXQueEnque(hQue, argv[1], ((UI)_tcslen(argv[1]) + 1) * (sizeof(UT)));
     42 :             else           AjcPrintF(TEXT("*** Invalid parameter.¥n"));
     43 :         }
     44 :         //  文字列データを持つノードを先頭に挿入
     45 :         else if (_tcsicmp(argv[0], TEXT("QT")) == 0) {
     46 :             if (argc == 2) AjcXQueEnqTop(hQue, argv[1], ((UI)_tcslen(argv[1]) + 1) * (sizeof(UT)));
     47 :             else           AjcPrintF(TEXT("*** Invalid parameter.¥n"));
     48 :         }
     49 :         //  ノードデータ一覧を表示 (昇順)
     50 :         else if (_tcsicmp(argv[0], TEXT("L")) == 0) {
     51 :             seq = 0;
     52 :             if (pDat = (UTP)AjcXQueTopNode(hQue, &bytes)) {
     53 :                 do {
     54 :                     AjcPrintF(TEXT("%2d - %p : %s¥n"), seq++, pDat, pDat);
     55 :                 } while(pDat = AjcXQueNextNode(hQue, pDat, &bytes));
     56 :             }
     57 :         }
     58 :         //  ノードデータ一覧を表示 (降順)
     59 :         else if (_tcsicmp(argv[0], TEXT("LD")) == 0) {
     60 :             seq = AjcXQueGetCount(hQue) - 1;
     61 :             if (pDat = (UTP)AjcXQueLastNode(hQue, &bytes)) {
     62 :                 do {
     63 :                     AjcPrintF(TEXT("%2d - %p : %s¥n"), seq--, pDat, pDat);
     64 :                 } while(pDat = AjcXQuePrevNode(hQue, pDat, &bytes));
     65 :             }
     66 :         }
     67 :         //  nnn 番目のノードを削除
     68 :         else if (_tcsicmp(argv[0], TEXT("R"))  == 0) {
     69 :             if (argc == 2 && (pDat = GetNodeAddr(_ttoi(argv[1])))) AjcXQueRemove(hQue, pDat);
     70 :             else                                                   AjcPrintF(TEXT("*** Invalid parameter.¥n"));
     71 :         }
     72 :         //  nnn 番目のノードの直前／末尾にノードを挿入
     73 :         else if (_tcsicmp(argv[0], TEXT("I"))  == 0) {
     74 :             if      (argc == 3 && (pDat = GetNodeAddr(_ttoi(argv[2])))) AjcXQueInsert(hQue, argv[1], (UI)(_tcslen(argv[1]) + 1)
* sizeof(UT), pDat);
     75 :             else if (argc == 2                                        ) AjcXQueInsert(hQue, argv[1], (UI)(_tcslen(argv[1]) + 1)
* sizeof(UT), NULL);
     76 :             else                                                        AjcPrintF(TEXT("*** Invalid parameter.¥n"));
     77 :         }
     78 :         //  全ノードを取り出して表示
     79 :         else if (_tcsicmp(argv[0], TEXT("G"))  == 0) {
     80 :             UT  buf[256];
     81 :             seq = 0;
     82 :             while ((AjcXQueDeque(hQue, &buf, sizeof buf)) != -1) {
     83 :                 AjcPrintF(TEXT("%2d - %s¥n"), seq++, buf);
     84 :             }
     85 :         }
     86 :         //  全ノードを破棄
     87 :         else if (_tcsicmp(argv[0], TEXT("A"))  == 0) {
     88 :             AjcXQuePurge(hQue);
     89 :         }
     90 :         //  文字列を含むノードを破棄
     91 :         else if (_tcsicmp(argv[0], TEXT("S"))  == 0) {
     92 :             if (argc == 2) {
     93 :                 AjcXQueDelAny(hQue, (UX)argv[1], cbDelAny);
     94 :             }
     95 :             else AjcPrintF(TEXT("*** Invalid parameter.¥n"));
     96 :         }
     97 :         //  ノードヘッダのダンプ表示
     98 :         else if (_tcsicmp(argv[0], TEXT("DMP"))  == 0) {
     99 :             seq = 0;
    100 :             if (pDat = (UTP)AjcXQueTopNode(hQue, &bytes)) {
    101 :                 do {
    102 :                     PAJCXQNODE pH = ((PAJCXQNODE)pDat) - 1;
    103 :                     AjcPrintF(TEXT("%2d - %p : id = %08X, prev = %p, next = %p, ")
    104 :                               TEXT("len = %d, dat = %p : %s¥n"),
    105 :                               seq++, pH, pH->id, pH->pBfr, pH->pNxt, pH->len, pDat, pDat);
    106 :                 } while(pDat = AjcXQueNextNode(hQue, pDat, &bytes));
    107 :             }
    108 :         }
    109 :         //  その他
```

```
110 :         else {
111 :             AjcPrintF(TEXT("*** Invalid command.¥n"));
112 :         }
113 :     }
114 : }
115 : //-----------------------------------------------------------------------------------------//
116 : //  Ｎ番目のノードデータアドレス取得                                                       //
117 : //-----------------------------------------------------------------------------------------//
118 : MAJCXQUEPTR(PTRARR, UTP)
119 : static  VOP     GetNodeAddr(UI ix)
120 : {
121 :     VOP     rc = NULL;
122 :     UI      n;
123 :     PCPTRARR pArr = (PPTRARR)AjcXQueCreatePtrArr(hQue, &n);
124 :     if (ix < n) {
125 :         rc = (VOP)pArr[ix].pNode;
126 :     }
127 :     AjcXQueReleasePtrArr(hQue, (PCAJCXQUEPTR)pArr);
128 :     return rc;
129 : }
130 :
131 : //-----------------------------------------------------------------------------------------//
132 : //  m  a  i  n                                                                             //
133 : //-----------------------------------------------------------------------------------------//
134 : int  AjcMain(int argc, UTP argv[])
135 : {
136 :     AjcSetStdoutMode();
137 :
138 :     AjcSetConsoleBufSize(128, 64);
139 :     AjcSetConsoleWndRect(0, 0, 127, 30);
140 :
141 :     AjcPrintF(TEXT("¥n 以下のコマンドを入力してください。¥n¥n"));
142 :     AjcPrintF(TEXT("  Q   <文字列>     : 文字列データを持つノードを末尾に挿入¥n"));
143 :     AjcPrintF(TEXT("  QT  <文字列>     : 文字列データを持つノードを先頭に挿入¥n"));
144 :     AjcPrintF(TEXT("  L                : ノードデータ一覧を表示（昇順）¥n"));
145 :     AjcPrintF(TEXT("  LD               : ノードデータ一覧を表示（降順）¥n"));
146 :     AjcPrintF(TEXT("  R    nnn         : nnn(0～)番目のノードを削除¥n"));
147 :     AjcPrintF(TEXT("  I   <文字列> nnn : nnn(0～)番目のノードの直前に文字列データを持つノードを挿入¥n"));
148 :     AjcPrintF(TEXT("  I   <文字列>     : 文字列データを持つノードを末尾に挿入¥n"));
149 :     AjcPrintF(TEXT("  G                : 全ノードを取り出して表示¥n"));
150 :     AjcPrintF(TEXT("  A                : 全ノードを破棄¥n"));
151 :     AjcPrintF(TEXT("  S   <文字列>     : 文字列を含むノードを削除¥n"));
152 :     AjcPrintF(TEXT("  DMP              : ノードヘッダのダンプ表示¥n"));
153 :     AjcPrintF(TEXT("  <ESC>キー        : 終了¥n"));
154 :     AjcPrintF(TEXT("¥n"));
155 :
156 :     //  双方向リストインスタンス生成
157 :     hQue = AjcXQueCreate(0, cbRemove);
158 :
159 :     AjcPrintF(TEXT("Input(%3u) - "), AjcXQueGetCount(hQue));
160 :     while (AjcConInputByNtc(32, 0, cbNtcInput)) {
161 :         AjcPrintF(TEXT("Input(%3u) - "), AjcXQueGetCount(hQue));
162 :     }
163 :     //  双方向リストインスタンス消去
164 :     AjcXQueDelete(hQue);
165 :
166 :     return 0;
167 : }
```

36. スレッド間メールデータ通信

スレッド間でのメールデータ通信を行います。
メールデータ受信側のスレッドは、メールデータが届くまで（あるいは、タイムアウトするまで）自スレッドをウェイトします。
メールデータ送信側のスレッドが、メールデータを送信すると、受信側のスレッドが起床され、メールデータを受け取ることができます。

36.1. サポートＡＰＩ

スレッド間メールデータ通信のサポートＡＰＩ一覧を以下に示します。

#	内容	関数名		備考
		固定長データ	可変長データ	
1	インスタンス生成	AjcFMbxCreate	AjcVMbxCreate	
2	インスタンス消去	AjcFMbxDelete	AjcVMbxDelete	
3	メールデータ通知	AjcFMbxEnque	AjcVMbxEnque	
4	優先メールデータ通知	AjcFMbxEnqTop	AjcVMbxEnqTop	
5	メールデータ取り出し	AjcFMbxDeque	AjcVMbxDeque	メールデータ到着／タイムアウトまでウェイト
6	メールデータ取り出し	－	AjcVMbxDequeEx	AjcVMbxDeque()の改良版
7	全メールデータ破棄	AjcFMbxPurge	AjcVMbxPurge	
8	個別メールデータ破棄	AjcFMbxDelAny	AjcVMbxPDelAny	
9	メールデータの個数取得	AjcFMbxGetCount	AjcVMbxGetCount	
10	メールデータの総バイト数取得	－	AjcVMbxGetTotalBytes	

36.1.1. インスタンス生成（Ajc{F/V}MbxCreate）

形　式 ： HAJCFMBX　AjcFMbxCreate(UI MailSize, UX cbp, VO (CALLBACK *cbRemove)(VOP pDat, UX cbp)); ---　固定長データ用
HAJCVMBX　AjcVMbxCreate(UX cbp, VO (CALLBACK *cbRemove)(VOP pDat, UI len, UX cbp)); ---------- 可変長データ用

引　数 ： MailSize　- メールデータのバイト数
cbp　　- コールバックパラメタ
cbRemove　- メールデータ消去通知用コールバック関数のアドレス（不要時は NULL）

説　明 ： スレッド間データ通信機能のインスタンスを生成／初期化します。
「cbRemove」は、AjcFMbxPurge/ AjcVMbxPurge や AjcFMbxDelete/ AjcVMbxDelete により、消去されるメールデータを通知するためのコールバック関数を指定します。
メールデータ消去時に、何も操作する必要が無い場合は、cbRemove=NULL を指定します。

戻り値 ： ≠NULL - 成功　（インスタンスハンドルを返します）
＝NULL - 失敗

コールバック： コールバック関数の仕様は、以下のとおりです。

cbRemove（消去されるメールデータ通知）

形　式 ： VO CALLBACK ***cbRemove***(VOP pDat, UX cbp); ---------- 固定長データ用
VO CALLBACK ***cbRemove***(VOP pDat, UI len, UX cbp); ----- 可変長データ用

引　数 ： pDat - メールデータのアドレス
len　- メールデータのバイト数
cbp　- コールバックパラメタ

説　明 ： 当該メールデータに関連した後処理（リソースの開放等）を行います。

戻り値 ： なし

36.1.2. インスタンス消去（Ajc{F/V}MbxDelete）

形　式 ： VO AjcFMbxDelete (HAJCFMBX hFMbx); - 固定長データ用
VO AjcVMbxDelete (HAJCVMBX hVMbx); - 可変長データ用

引　数 ： hFMbx/hVMbx - インスタンスハンドル

説　明 ： 全メールデータを削除（開放）し、AjcFMbxCreate/ AjcVMbxCreate で動的に確保した、リソースを解放します。

AjcFMbxCreate/AjcVMbxCreate で cbRemove が指定されている場合は、全てのメールデータについて、破棄される直前に、当該メールデータを通知します。

戻り値 ： なし

36.1.3. メールデータ通知（Ajc{F/V}MbxEnque）

形　式 ： BOOL AjcFMbxEnque (HAJCFMbx hFMbx, C_VOP pDat); ---------- 固定長データ用
BOOL AjcVMbxEnque (HAJCVMbx hVMbx, C_VOP pDat, UI lDat); ----- 可変長データ用

引　数 ： hFMbx/hVMbx - インスタンスハンドル
pDat - 通知するメールデータのアドレス
lDat - 通知するメールデータのバイト数

説　明 ： 他のスレッドへメールデータを通知します。
指定されたメールデータは、メールデータ・キューの末尾に追加されます。
pDat で示されるデータは、メールデータバッファ内にコピーされます。従って、本関数実行後、pDat で示されるデータを保持している必要はありません。

戻り値 ： TRUE - 成功
FALSE - 失敗

36.1.4. 優先メールデータ通知（Ajc{F/V}MbxEnqTop）

形　式 ： BOOL AjcFMbxEnqTop (HAJCFMBX hFMbx, C_VOP pDat); ---------- 固定長データ用
BOOL AjcVMbxEnqTop (HAJCVMBX hVMbx, C_VOP pDat, UI lDat); ----- 可変長データ用

引　数 ： hFMbx/hVMbx - インスタンスハンドル
pDat - 通知するメールデータのアドレス
lDat - 通知するメールデータのバイト数

説　明 ： 他のスレッドへ優先メールデータを通知します。
指定されたメールデータは、メールデータ・キューの先頭に追加されます。
pDat で示されるデータは、新たに作成したノード内にコピーされます。従って、本関数実行後、pDat で示されるデータを保持している必要はありません。

戻り値 ： TRUE - 成功
FALSE - 失敗

36.1.5. メールデータ取り出し（Ajc{F/V}MbxDeque）

形　式 ： UI　AjcFMbxDeque (HAJCFMBX hFMbx, VOP pBuf, UI msTime);　---------- 固定長データ用
UI　AjcVMbxDeque (HAJCVMBX hVMbx, VOP pBuf, UI lBuf, UI msTime);　----- 可変長データ用

引　数 ： hFMbx/hVMbx - インスタンスハンドル
pBuf　　- 取り出したメールデータを格納（コピー）するバッファのアドレス
lBuf　　- バッファのバイト数
msTime　　- メールデータ受信待ちタイマ[ms]

説　明 ： メールデータがある場合は、先頭メールデータを取り出して、pBuf で示されるバッファに格納（コピー）します。
メールデータが無い場合は、メールデータが格納されるまで（他のスレッドがメールデータを通知するまで）待ってから先頭メールデータを取り出して、pBuf で示されるバッファに格納（コピー）します。
msTime で指定された時間が経過してもメールデータが通知されない場合は、何もせずに０を返します。
msTime=-1 とした場合は、メールデータが通知されるまで永久に待ち続けます。

固定長データの場合は、pBuf で示されるバッファのバイト数は、インスタンスの初期化時に指定されたメールデータサイズ(MailSize)でなければなりません。

可変長データの場合は、実際のメールデータサイズが、lBuf で示されるバイト数よりも大きい場合は、lBuf で示されるバイト数のデータだけをバッファに格納します。

戻り値 ： ≠０：当該メールデータのバイト数
＝０：メールデータなし（タイムアウト）

36.1.6. メールデータ取り出し（Ajc{F/V}MbxDeque）

形　式 ： UI　AjcVMbxDequeEx (HAJCVMBX hVMbx, VOP pBuf, UI lBuf, UI msTime);　----- 可変長データ用

引　数 ： hVMbx　　- インスタンスハンドル
pBuf　　- 取り出したメールデータを格納（コピー）するバッファのアドレス
lBuf　　- バッファのバイト数
msTime　　- メールデータ受信待ちタイマ[ms]

説　明 ： メールデータがある場合は、先頭メールデータを取り出して、pBuf で示されるバッファに格納（コピー）します。
メールデータが無い場合は、メールデータが格納されるまで（他のスレッドがメールデータを通知するまで）待ってから先頭メールデータを取り出して、pBuf で示されるバッファに格納（コピー）します。
msTime で指定された時間が経過してもメールデータが通知されない場合は、何もせずに０を返します。
msTime=-1 とした場合は、メールデータが通知されるまで永久に待ち続けます。

固定長データの場合は、pBuf で示されるバッファのバイト数は、インスタンスの初期化時に指定されたメールデータサイズ(MailSize)でなければなりません。

可変長データの場合は、実際のメールデータサイズが、lBuf で示されるバイト数よりも大きい場合は、lBuf で示されるバイト数のデータだけをバッファに格納します。

戻り値 ： ≠０：当該メールデータのバイト数
＝-1：メールデータなし（タイムアウト）

備　考 ： このＡＰＩは、基本的にAjcVMbxDeque()と同じですが、ノードなしの場合は－１を返します。
AjcVMbxDequeEx()では、０を返すため、０バイトデータとノードなしが混同します。
０バイトデータを扱う場合は、AjcVMbxDequeEx()を使用してください。

36.1.7. 全メールデータ破棄（Ajc{F/V}MbxPurge）

形　式 ： VO　AjcFMbxPurge (HAJCFMBX hFMbx)； ---- 固定長データ用
VO　AjcVMbxPurge (HAJCVMBX hVMbx)； ----- 可変長データ用

引　数 ： hFMbx/hVMbx - インスタンスハンドル

説　明 ： キューに蓄積されている、全てのメールデータを破棄します。

AjcFMbxCreate/AjcVMbxCreate で cbRemove が指定されている場合は、全てのメールデータについて、破棄される直前に、当該メールデータを通知します。

戻り値 ： なし

36.1.8. 個別メールデータ破棄（Ajc{F/V}MbxDelAny）

形　式 ： BOOL　AjcFMbxDelAny (HAJCFQUE nFMbx, UX cbp, BOOL (CALLBACK *cbQuery)(VOP pDat, UX cbp))； ------ 固定長ノード用
BOOL　AjcVMbxDelAny (HAJCVQUE hVMbx, UX cbp, BOOL (CALLBACK *cbQuery)(VOP pDat, UI len, UX cbp))；-- 可変長ノード用

引　数 ： hFMbx/hVMbx - インスタンスハンドル
cbp　　- コールバックパラメタ
cbQuery　- コールバック関数（削除問い合わせ）

説　明 ： 個別にノードを破棄します。
破棄するか否かは、コールバック関数で問い合わせます。

戻り値 ： TRUE　- 成功
FALSE - 失敗

コールバック： コールバック関数の仕様は、以下のとおりです。

cbQuery（ノードデータ破棄の問い合わせ）

形　式 ： BOOL CALLBACK ***cbQuery***(VOP pDat, UX cbp)； ---------- 固定長ノード用
BOOL CALLBACK ***cbQuery***(VOP pDat, UI len, UX cbp)； ----- 可変長ノード用

引　数 ： pDat - ノードデータのアドレス（ノードのデータ部分を示します）
len　- ノードデータのバイト数
cbp　- コールバックパラメタ

説　明 ： 通知されたノードを削除する場合は TURE を、削除しない場合は FALSE を返します。

戻り値 ： TRUE　- ノードを削除する
FALSE – 喉を削除しない

36.1.9. メールデータ数取得（Ajc{F/V}MbxGetCount）

形　式 ： UI　AjcFMbxGetCount (HAJCFMBX hFMbx)； ---------- 固定長データ用
UI　AjcVMbxGetCount (HAJCVMBX hVMbx)； ---------- 可変長データ用

引　数 ： hFMbx/hVMbx - インスタンスハンドル

説　明 ： 現在の総メールデータの個数を返します。

戻り値 ： 総メールデータ個数

36.1.10. 現在のメールデータの総バイト数取得（AjcVMbxGetTotalBytes）

形　式　：　ULL　AjcVMbxGetTotalBytes (HAJCFMBX hFMbx);　---------- 可変長データ用のみ

引　数　：　hFMbx/hVMbx - インスタンスハンドル

説　明　：　現在の全メールデータの総バイト数を返します。

戻り値　：　全メールデータの総バイト数

36.2. サンプルプログラム

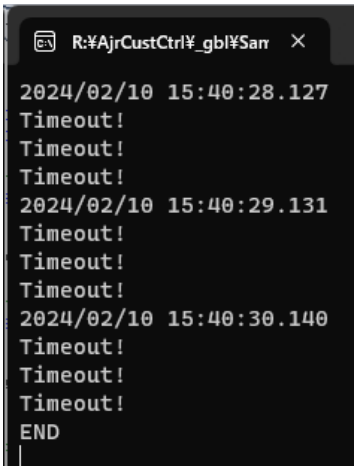
36.2.1. SW_FMbx（スレッド間メールデータ通信 – 固定長データ）

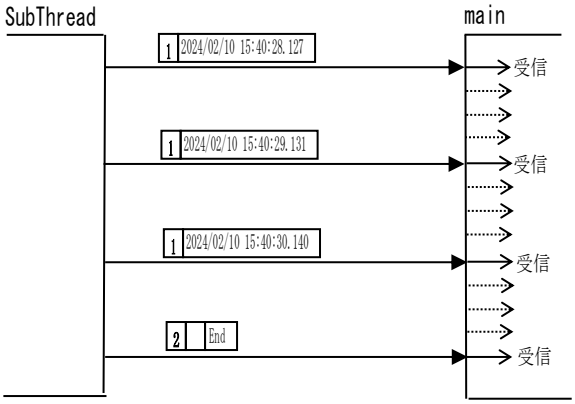
このサンプルプログラムは、サブスレッドを生成し、サブスレッドからメインスレッドへ１秒周期で現在時刻を３回通知します。
メインスレッドは、３００ｍｓ周期でサブスレッドからの通知を待ち受けて、通知された現在時刻を表示します。

通知する固定長メールデータの形式は、以下のとおりです。

cmd	現在時刻

cmd ： 種別（1:現在時刻通知, 2:終了通知）






```
 1 : //
 2 : //  SW_FMbxC.c
 3 : //
 4 : #include    <AjrCstXX.h>
 5 : #include    <process.h>
 6 :
 7 : //----- メールデータ種別 ------------------------------------------------------------------------------------------//
 8 : typedef enum {
 9 :     CMD_TIME    = 1 ,       //  現在時刻
10 :     CMD_END                 //  終了
11 : } CMDCODE;
12 :
13 : //----- メールデータ形式 ------------------------------------------------------------------------------------------//
14 : typedef struct {
15 :     CMDCODE     cmd;        //  メールデータ種別
16 :     SYSTEMTIME  stime;      //  現在時刻
17 : } MAILDATA;
18 :
19 :
20 : //==============================================================================================================//
21 : //                                                                                                              //
22 : //  サブスレッド                                                                                                //
23 : //                                                                                                              //
24 : //==============================================================================================================//
25 : VO      SubThread(VOP param)
26 : {
27 :     HAJCFMBX    hFMbx = (HAJCFMBX)param;
28 :     UI          i;
29 :     MAILDATA    MailData;
30 :
31 :     //-----  １秒周期で現在時刻をメインスレッドに通知（３回）---------//
32 :     for (i = 0; i < 3; i++) {
33 :         MailData.cmd = CMD_TIME;
34 :         GetLocalTime(&MailData.stime);
35 :         AjcFMbxEnque(hFMbx, &MailData);
36 :         Sleep(1000);
37 :     }
38 :     //----- メインスレッドに終了を通知 ------------------------------//
39 :     MailData.cmd = CMD_END;
40 :     AjcFMbxEnque(hFMbx, &MailData);
41 :     //----- サブスレッド終了 ----------------------------------------//
42 :     _endthread();
```

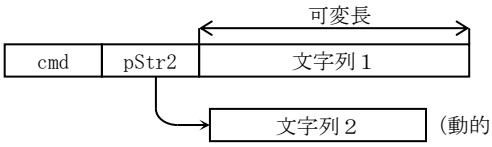
```
43 : }
44 :
45 : //=============================================================================================================//
46 : //                                                                                                             //
47 : //  m a i n                                                                                                    //
48 : //                                                                                                             //
49 : //=============================================================================================================//
50 : int     AjcMain(int argc, UTP argv[])
51 : {
52 :     HAJCFMBX    hFMbx;
53 :     MAILDATA    MailData;
54 :
55 :     AjcSetStdoutMode();
56 :
57 :     //----- メールボックス・インスタンス生成 ------------------------//
58 :     hFMbx = AjcFMbxCreate(sizeof(MAILDATA), 0, NULL);
59 :     //----- サブスレッド生成 ----------------------------------------//
60 :     _beginthread(SubThread, 0, hFMbx);
61 :     Sleep(1);
62 :     //----- サブスレッドからの通知待ちループ ------------------------//
63 :     MailData.cmd = 0;
64 :     while (MailData.cmd != CMD_END) {
65 :         if (AjcFMbxDeque(hFMbx, &MailData, 300)) {      // サブスレッドからの通知あり？
66 :             switch (MailData.cmd) {
67 :                 case CMD_TIME:                          //      ● 現在時刻通知
68 :                     AjcPrintF(TEXT("%d/%02d/%02d %02d:%02d:%02d.%03d¥n"),
69 :                             MailData.stime.wYear, MailData.stime.wMonth , MailData.stime.wDay,
70 :                             MailData.stime.wHour, MailData.stime.wMinute, MailData.stime.wSecond,
71 :                             MailData.stime.wMilliseconds);
72 :                     break;
73 :
74 :                 case CMD_END:                           //      ● 終了通知
75 :                     AjcPrintF(TEXT("END¥n"));
76 :                     break;
77 :             }
78 :         }
79 :         else {                                          //  サブスレッドからの通知なし（タイムアウト）？
80 :             AjcPrintF(TEXT("Timeout!¥n"));
81 :         }
82 :     }
83 :     Sleep(1);
84 :
85 :     //----- メールボックス・インスタンス消去 ------------------------//
86 :     AjcFMbxDelete(hFMbx);
87 :
88 :     getchar();
89 :     return  0;
90 : }
91 :
92 :
```

36.2.2. SW_VMbxC（スレッド間メールデータ通信 – 可変長データ）

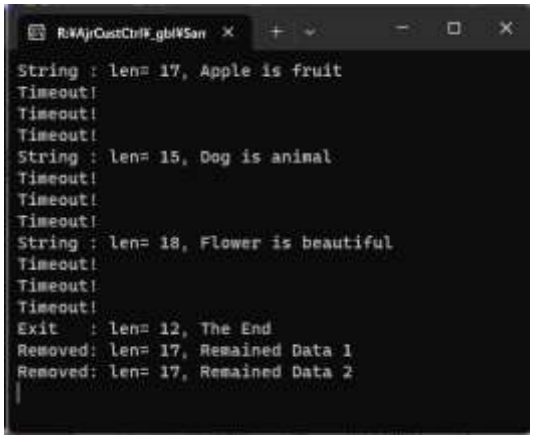
このサンプルプログラムは、サブスレッドを生成し、サブスレッドからメインスレッドへ１秒周期で文字列ペアを３回通知します。
その後、メインスレッドへ終了を通知し、さらに、(残留メールデータとして）文字列ペアを２回通知します。

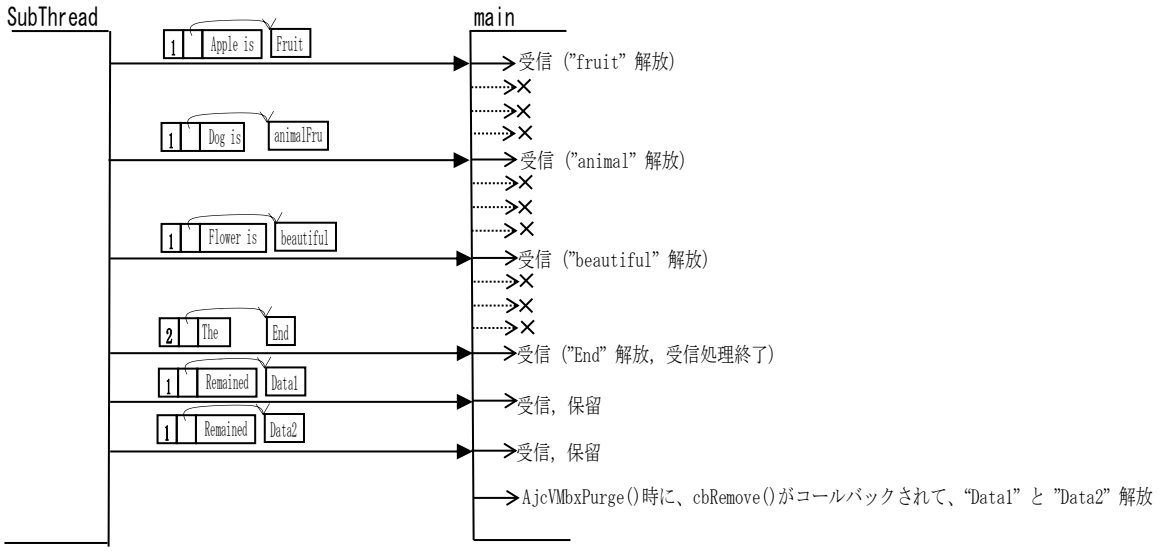
メインスレッドは、３００ｍｓ周期でサブスレッドからの通知を待ち受けて、通知された文字列ペアを表示します。
サブスレッドから、終了通知を受けたら、残留しているメールデータを破棄します。

通知する可変長メールデータの形式は、以下のとおりです。



cmd ： 種別（1:文字列通知, 2:終了通知）
pStr2: 文字列２へのポインタ



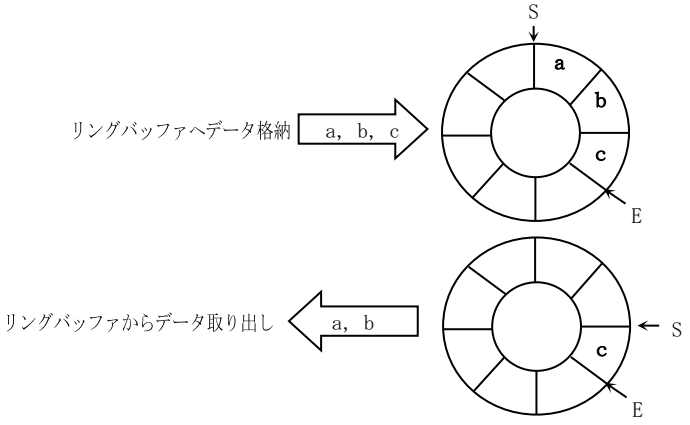


```
 1 : //
 2 : //  SW_VMbxC.c
 3 : //
 4 : #include    <AjrCstXX.h>
 5 : #include    <process.h>
 6 : #include    <tchar.h>
 7 :
 8 : //----- メールデータ種別 ----------------------------------------------------------------------------------------//
 9 : typedef enum {
10 :     CMD_STR = 1 ,           //  文字列通知
11 :     CMD_END                 //  終了
12 : } CMDCODE;
13 :
14 : //----- メールデータ形式 ----------------------------------------------------------------------------------------//
15 : typedef struct {
16 :     CMDCODE     cmd;        //  メールデータ種別
17 :     UTP         pStr2;      //  文字列２へのポインタ
18 :     UT          Str1[64];   //  文字列１
19 : } MAILDATA, *PMAILDATA;
20 :
21 :
22 : //=============================================================================================================//
23 : //                                                                                                             //
24 : //  メールデータ作成                                                                                           //
25 : //                                                                                                             //
26 : //=============================================================================================================//
27 : static  UI      MakeMailData(PMAILDATA pBuf, CMDCODE cmd, C_UTP pStr1, C_UTP pStr2)
28 : {
29 :     UI  lStr1 = (UI)MAjcStrLen(pStr1) + 1;
30 :     UI  lStr2 = (UI)MAjcStrLen(pStr2) + 1;
31 :
32 :     pBuf->cmd   = cmd;
33 :     MAjcStrCpy(pBuf->Str1, AJCTSIZE(pBuf->Str1), pStr1);
34 :     pBuf->pStr2 = AjcTAlloc(lStr2);
35 :     MAjcStrCpy(pBuf->pStr2, lStr2, pStr2);
36 :
37 :     return sizeof(MAILDATA) - sizeof pBuf->Str1 + (lStr1 * sizeof(UT));
38 : }
39 : //=============================================================================================================//
40 : //                                                                                                             //
41 : //  破棄メールデータ通知用コールバック関数                                                                     //
42 : //                                                                                                             //
43 : //=============================================================================================================//
44 : static  VO  CALLBACK    cbRemove(VOP pD, UI lDat, UX cbp)
45 : {
46 :     PMAILDATA   pDat = (PMAILDATA)pD;
47 :     AjcPrintF(TEXT("Removed: len=%3d, %s %s¥n"), lDat, pDat->Str1, pDat->pStr2);
48 :     AjcTFree(pDat->pStr2);
49 : }
50 : //=============================================================================================================//
51 : //                                                                                                             //
52 : //  サブスレッド                                                                                               //
53 : //                                                                                                             //
54 : //=============================================================================================================//
55 : static  VO      SubThread(VOP param)
56 : {
57 :     HAJCVMBX    hVMbx = (HAJCVMBX)param;
58 :     UI          lDat;
59 :     MAILDATA    MailData;
60 :
61 :     //----- １秒周期で文字列をメインスレッドに通知（３回）-----------//
62 :     lDat = MakeMailData(&MailData, CMD_STR, TEXT("Apple is" ), TEXT("fruit"    ));
63 :     AjcVMbxEnque(hVMbx, &MailData, lDat);
64 :     Sleep(1000);
65 :     lDat = MakeMailData(&MailData, CMD_STR, TEXT("Dog is"   ), TEXT("animal"   ));
66 :     AjcVMbxEnque(hVMbx, &MailData, lDat);
67 :     Sleep(1000);
68 :     lDat = MakeMailData(&MailData, CMD_STR, TEXT("Flower is"), TEXT("beautiful"));
69 :     AjcVMbxEnque(hVMbx, &MailData, lDat);
70 :     Sleep(1000);
71 :     //----- メインスレッドに終了を通知 ------------------------------//
72 :     lDat = MakeMailData(&MailData, CMD_END, TEXT("The"), TEXT("End"));
73 :     AjcVMbxEnque(hVMbx, &MailData, lDat);
74 :     //----- 余分なデータ通知 ----------------------------------------//
75 :     lDat = MakeMailData(&MailData, CMD_STR, TEXT("Remained"), TEXT("Data 1"));
76 :     AjcVMbxEnque(hVMbx, &MailData, lDat);
77 :     lDat = MakeMailData(&MailData, CMD_STR, TEXT("Remained"), TEXT("Data 2"));
78 :     AjcVMbxEnque(hVMbx, &MailData, lDat);
79 :     //----- サブスレッド終了 ----------------------------------------//
80 :     _endthread();
```

```
 81 : }
 82 :
 83 : //=============================================================================================================//
 84 : //                                                                                                             //
 85 : //  m a i n                                                                                                    //
 86 : //                                                                                                             //
 87 : //=============================================================================================================//
 88 : int     AjcMain(int argc, UTP argv[])
 89 : {
 90 :     HAJCVMBX    hVMbx;
 91 :     MAILDATA    MailData;
 92 :     UI          lDat;
 93 :     BOOL        fEnd = FALSE;
 94 :
 95 :     AjcSetStdoutMode();
 96 :
 97 :     //----- メールボックス・インスタンス生成 ------------------------//
 98 :     hVMbx = AjcVMbxCreate(0, cbRemove);
 99 :     //----- サブスレッド生成 ----------------------------------------//
100 :     _beginthread(SubThread, 0, hVMbx);
101 :     Sleep(1);
102 :     //----- サブスレッドからの通知待ちループ ------------------------//
103 :     MailData.cmd = 0;
104 :     while (!fEnd) {
105 :         memset(&MailData, 0, sizeof MailData);
106 :         if ((lDat = AjcVMbxDequeEx(hVMbx, &MailData, sizeof MailData, 300)) != -1) {    // サブスレッドからの通知あり？
107 :             switch (MailData.cmd) {
108 :                 case CMD_STR:                                           //      ● 文字列通知
109 :                     AjcPrintF(TEXT("String : len=%3d, %s %s¥n"), lDat, MailData.Str1, MailData.pStr2);
110 :                     AjcTFree(MailData.pStr2);
111 :                     break;
112 :
113 :                 case CMD_END:                                           //      ● 終了通知
114 :                     AjcPrintF(TEXT("Exit   : len=%3d, %s %s¥n"), lDat, MailData.Str1, MailData.pStr2);
115 :                     AjcTFree(MailData.pStr2);
116 :                     fEnd = TRUE;
117 :                     break;
118 :             }
119 :         }
120 :         else {                                                          //  タイムアウト？
121 :             AjcPrintF(TEXT("Timeout!¥n"));
122 :         }
123 :     }
124 :     Sleep(100);
125 :
126 :     //----- 残留メールデータ破棄 ------------------------------------//
127 :     AjcVMbxPurge(hVMbx);
128 :
129 :     //----- メールボックス・インスタンス消去 ------------------------//
130 :     AjcVMbxDelete(hVMbx);
131 :
132 :     getchar();
133 :     return  0;
134 : }
135 :
```

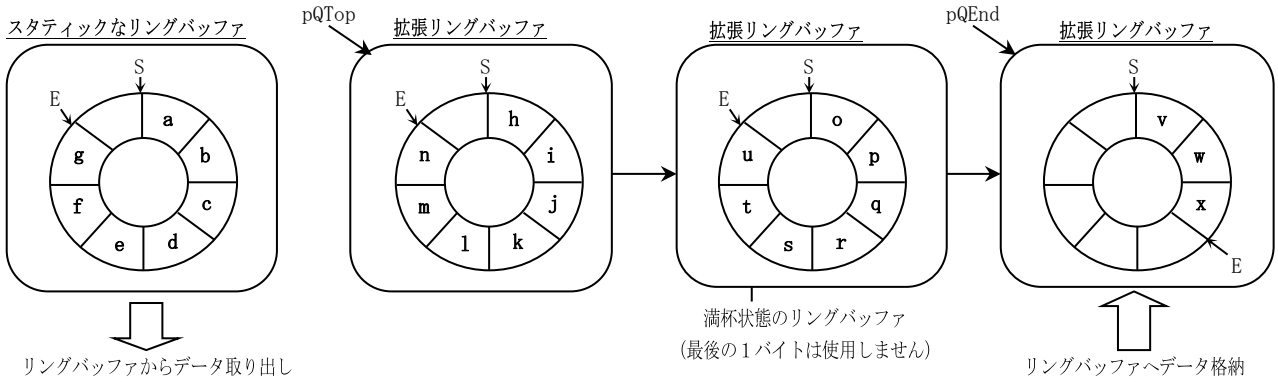
37. リングバッファ制御

リングバッファとは、仮想的にリング状のバッファメモリを構成し、データの格納と取り出しを行うものです。
データ取り出しの際は、最古のデータから順に取り出されます。



本ライブラリでは、スタティックなリングバッファと拡張リングバッファが存在します。
スタティックなリングバッファは、静的な領域のリングバッファであり、１ヶだけ存在します。
スタティックなリングバッファが満杯になった場合は、拡張リングバッファを作成します。
拡張リングバッファへのデータ格納において、１つの拡張リングバッファの容量を越える場合は、満杯になった拡張リングバッファメモリを退避（キューイング）し、新たな拡張リングバッファメモリを割り当てた上でデータの格納を続行します。
また、拡張リングバッファからデータを取り出すことにより、退避（キューイング）していた拡張リングバッファが不要になった場合は、このメモリを開放します。

つまり、本ライブラリでは、スタティックなリングバッファと、リスト構造でチェインした複数の拡張リングバッファを、あたかも１つのリングバッファのように扱います。また、格納データの容量に応じて拡張リングバッファの個数が増減します。



尚、スタティックなリングバッファだけ、あるいは、拡張リングバッファだけ、といった構成にすることもできます。

37.1. サポートＡＰＩ

リングバッファ制御のサポートＡＰＩ一覧を以下に示します。

#	機能	関数名	備考
1	AjcRngCreate	インスタンス生成	
2	AjcRngDelete	インスタンス消去	
3	AjcRngPutData	リングバッファへデータ格納	メモリ不足時は、格納できる分だけ格納する
4	AjcRngPutDataEx	リングバッファへデータ格納	メモリ不足時は、データ全体を格納しない
5	AjcRngGetData	リングバッファからデータ取り出し	
6	AjcRngPeekData	リングバッファ中の最古のデータを見る	リングバッファから、データは取り出さない
7	AjcRngGetDataSize	リングバッファに格納されているデータサイズ取得	
8	AjcRngPurge	リングバッファに格納されている全データ破棄	
9	AjcRngEnableMaltiThread	マルチスレッドの許可／禁止	

37.1.1. インスタンス生成（AjcRngCreate）

形　式 ： HAJCRNG　AjcRngCreate(VOP pStaticBuf, UI lStaticBuf, UI　ChunkSize);

引　数 ： pStaticBuf - スタティックなリングバッファのアドレス（不要時はＮＵＬＬ）
lStaticBuf - スタティックなリングバッファのバイト数（2以上, pStaticBuf=NULL 時は不要）
ChunkSize　- １つの拡張リングバッファメモリのバイト数

説　明 ： リングバッファのインスタンスを生成します。
pStaticBuf=NULL とした場合は、スタティックなリングバッファ無し（拡張リングバッファだけ）の構成となります。
ChunkSize=0 とした場合は、拡張リングバッファなし（スタティックなリングバッファだけ）の構成となります。

ChunkSize は、１つの拡張リングバッファのメモリブロックサイズを指定します。但し、実際にリングバッファとして利用できるバイト数は、ChunkSize - AJCRNGQUE_HEADER_SIZE となります。
ChunkSize は、AJCRNGQUE_HEADER_SIZE +2 以上の値を指定しなければなりません。
AJCRNGQUE_HEADER_SIZE は、処理系により異なりますが、３２ビット系の場合１２で、６４ビット系の場合１６となります。

戻り値 ： ≠NULL - 成功　（インスタンスハンドルを返します）
＝NULL - 失敗

37.1.2. インスタンス消去（AjcRngDelete）

形　式 ： BOOL　AjcRngDelete (HAJCRNG hRng);

引　数 ： hRng - インスタンスハンドル（AjcRngCreate の戻り値）

説　明 ： AjcRngCreate で動的に確保した、インスタンスワーク領域を解放します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

37.1.3. データ格納（AjcRngPutData, 格納できない場合は、格納できる分だけ格納）

形　式 ： UI　AjcRngPutData(HAJCRNG hRng, C_VOP pDat, UI len);

引　数 ： hRng　　- インスタンスハンドル
pDat　　- リングバッファへ格納するデータのアドレス
len　　- リングバッファへ格納するデータのバイト数

説　明 ： pDat, len で示されるデータをリングバッファへ格納します。
メモリ不足により、指定されたデータを格納しきれない場合は、格納できる分だけを格納します。

戻り値 ： 実際に格納されたデータのバイト数
この値が、len より小さい場合は、メモリ確保失敗のために、指定されたデータの前半部分しか格納されなかったことを意味します。

37.1.4. データ格納（AjcRngPutDataEx，格納できない場合は、データ全部を格納しない）

形　式 ： UI　AjcRngPutDataEx(HAJCRNG hRng, C_VOP pDat, UI len);

引　数 ： hRng　　－ インスタンスハンドル
pDat　　－ リングバッファへ格納するデータのアドレス
len　　－ リングバッファへ格納するデータのバイト数

説　明 ： pDat, len で示されるデータをリングバッファへ格納します。
メモリ不足となった場合は、指定されたデータ全部を格納しません。この場合、戻り値は0を返します。

戻り値 ： 実際に格納されたデータのバイト数（0 or len）

37.1.5. データ取り出し（AjcRngGetData）

形　式 ： UI　AjcRngGetData(HAJCRNG hRng, VOP pBuf, UI len);

引　数 ： hRng　　－ インスタンスハンドル
pBuf　　－ リングバッファから取り出したデータを格納するバッファのアドレス（空読みする場合は、ＮＵＬＬ）
len　　－ リングバッファから取り出すデータのバイト数

説　明 ： FIFOバッファからデータを取り出し、pBuf, len で示されるバッファへ格納します。
pBuf=NULL を指定した場合は、空読みします。

戻り値 ： 実際に取り出したデータのバイト数

37.1.6. データ覗き見（AjcRngPeekData）

形　式 ： UI　AjcRngPeekData(HAJCRNG hRng, VOP pBuf, UI len);

引　数 ： hRng　　－ インスタンスハンドル
pBuf　　－ リングバッファから取り出したデータを格納するバッファのアドレス
len　　－ リングバッファから取り出すデータのバイト数

説　明 ： リングバッファ先頭部分（最古部分）のデータを、pBuf, len で示されるバッファへ格納します。
リングバッファからデータは取り出しません。

戻り値 ： 実際に格納したデータのバイト数

37.1.7. 総格納データサイズ取得（AjcRngGetDataSize）

形　式 ： UI　AjcRngGetDataSize(HAJCRNG hRng);

引　数 ： hRng － インスタンスハンドル

説　明 ： リングバッファに格納されている、総データ容量（バイト数）を返します。

戻り値 ： リングバッファに格納されている、総データ容量（バイト数）

37.1.8. 格納データ破棄（AjcRngPurge）

形　式 ： BOOL　AjcRngPurge(HAJCRNG hRng);

引　数 ： hRng - インスタンスハンドル

説　明 ： リングバッファに格納されている、全データを破棄します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

37.1.9. マルチスレッドの許可／禁止（AjcRngEnableMultiThread）

形　式 ： BOOL　AjcRngEnableMultiThread(HAJCRNG hRng, BOOL fEnable);

引　数 ： hRng　　- インスタンスハンドル
fEnable　- TRUE　: 複数のスレッドからのアクセスを許可
FALSE : 複数のスレッドからのアクセスを禁止

説　明 ： fEnable=TRUE とした場合、複数のスレッドによる、リングバッファ・アクセスを可能にします。
この場合、各関数の入り口と出口で、クリティカルセクションによるスレッド間の排他制御を行います。
但し、次の関数は、(fEnable=TRUE としても）マルチスレッドでの排他制御を行いません。

・AjcRngCreate（インスタンス生成）
・AjcRngDelete（インスタンス消去）
・AjcRngEnableMultiThread（本関数）

fEnable=FALSE（デフォルト）とした場合は、マルチスレッドでの排他制御を行いません。

戻り値 ： 前回の許可／禁止状態

37.2. サンプルプログラム

37.2.1. SW_RingBufC（リングバッファのアクセス）

以下のサンプルプログラムは、キーボードから入力した文字列をリングバッファに格納します。
尚、次の入力は、コマンドとして動作します。

- １桁の数字：指定個数の文字列をリングバッファから取り出して表示¥n″);
- 空行　　：全ての文字列をリングバッファから取り出して表示¥n″);
- Ｋ　　：リングバッファ中の最古の文字列を表示¥n″);
- Ｐ　　：リングバッファ中の全データ破棄¥n″);
- ＥＳＣキー：プログラム終了¥n″);

スタティックなリングバッファのサイズは６４バイトで、拡張リングバッファのサイズは３２バイトです。

```
 1 : //
 2 : //  SW_RingBufC.c
 3 : //
 4 : #include    <AjrCstXX.h>
 5 : #include    <stdio.h>
 6 : #include    <tchar.h>
 7 :
 8 : #define STA_SIZE    64          //  スタティック・リングバッファサイズ
 9 : #define EXP_SIZE    32          //  拡張リングバッファサイズ
10 :
11 : static  UB  StaticBuf[STA_SIZE];
12 : static  UI  AllocCount = 0;
13 :
14 : //--------------------------------------------------------------------------------------//
15 : //  m a  i  n                                                                           //
16 : //--------------------------------------------------------------------------------------//
17 : int  AjcMain(int argc, UTP argv[])
18 : {
19 :     UI      i, n, len;
20 :     UT      buf[256] = {0};
21 :     HAJCRNG hRng;
22 :
23 :     AjcSetStdoutMode();
24 :
25 :     AjcPrintF(TEXT("入力したテキストをリングバッファに蓄えます。¥n″));
26 :     AjcPrintF(TEXT("但し、以下の入力はコマンドとして扱います。¥n″));
27 :     AjcPrintF(TEXT("     １桁の数字：指定個数の文字列をリングバッファから取り出して表示¥n″));
28 :     AjcPrintF(TEXT("       空行     ：全ての文字列をリングバッファから取り出して表示¥n″));
29 :     AjcPrintF(TEXT("        K       ：リングバッファ中の最古の文字列を表示¥n″));
30 :     AjcPrintF(TEXT("        P       ：リングバッファ中の全データ破棄¥n″));
31 :     AjcPrintF(TEXT("      ESC キー  ：プログラム終了¥n″));
32 :     AjcPrintF(TEXT("¥n″));
33 :
34 :     //----- インスタンス初期化 ---------------------------------------------------------//
35 :     if (argc > 1) {
36 :         if      (MAjcStrICmp(argv[1], TEXT("S″)) == 0) hRng = AjcRngCreate(StaticBuf, STA_SIZE, 0     );
37 :         else if (MAjcStrICmp(argv[1], TEXT("E″)) == 0) hRng = AjcRngCreate(NULL   , 0       , EXP_SIZE);
38 :         else                                   hRng = AjcRngCreate(StaticBuf, STA_SIZE, EXP_SIZE);
39 :     }
40 :     else {
41 :         hRng = AjcRngCreate(StaticBuf, STA_SIZE, EXP_SIZE);
42 :     }
43 :
44 :     //----- メインループ ---------------------------------------------------------------//
45 :     AjcPrintF(TEXT("Input(%5u) - "), AjcRngGetDataSize(hRng));
46 :     while (AjcConInput(buf, AJCTSIZE(buf))) {
47 :         //----- Ｐ：全データ破棄 -----------------//
48 :         if      (MAjcStrICmp(buf, TEXT("P")) == 0) {
49 :             AjcRngPurge(hRng);
50 :         }
51 :         //----- Ｋ：最古の文字列表示 -------------//
52 :         else if (MAjcStrICmp(buf, TEXT("K")) == 0) {
53 :             struct {int len; BC txt[1024];} s;
54 :             if (AjcRngPeekData(hRng, &s.len, sizeof s.len) == sizeof s.len) {
55 :                 AjcRngPeekData(hRng, &s, sizeof s.len + s.len);
56 :                 s.txt[s.len] = 0;
57 :                 AjcPrintF(TEXT("%s (Remain %u)¥n"), s.txt, AjcRngGetDataSize(hRng));
58 :             }
59 :         }
60 :         //----- 数字：指定個数を取り出し表示 -----//
61 :         else if (isdigit(buf[0]) && buf[1] == 0) {
62 :             n = _ttoi(buf);
63 :             for (i=0; i<n && AjcRngGetData(hRng, &len, sizeof len) == sizeof len; i++) {
```

```
64 :                 len = AjcRngGetData(hRng, buf, len);
65 :                 AjcPrintF(TEXT("%s (Remain %u)¥n"), buf, AjcRngGetDataSize(hRng));
66 :             }
67 :         }
68 :         //----- 空行：全数を取り出し表示 --------//
69 :         else if (buf[0] == 0           ) {
70 :             while (AjcRngGetData(hRng, &len, sizeof len) == sizeof len) {
71 :                 len = AjcRngGetData(hRng, buf, len);
72 :                 AjcPrintF(TEXT("%s (Remain %u)¥n"), buf, AjcRngGetDataSize(hRng));
73 :             }
74 :         }
75 :         //----- その他：データを格納 ------------//
76 :         else {
77 :             struct {UI stl; UT txt[256];} dat;
78 :             MAjcStrCpy(dat.txt, AJCTSIZE(dat.txt), buf);
79 :             dat.stl = (UI)(MAjcStrLen(dat.txt) + 1) * sizeof(UT);
80 :             AjcRngPutDataEx(hRng, &dat , sizeof(UI) + dat.stl);
81 :         }
82 :         AjcPrintF(TEXT("Input(%5u) - "), AjcRngGetDataSize(hRng));
83 :         buf[0] = 0;
84 :     }
85 :
86 :     AjcRngDelete(hRng);
87 :
88 :     return 0;
89 : }
```

38. C言語の字句分解

C／C＋＋言語のソースプログラムテキストを、字句（トークン）の単位に分解し、順次読み出すことができます。
ソースプログラムテキスト中のコメント（「/*…*/」や「//…」）部分を無視することができます。

例えば、以下の①ようなソースプログラムテキストは、②に示す字句に分解して順次読み出されます。

①

```
#include <stdio.h>
main( )
{
    printf("Hellow¥n");  /* ->STDOUT */
}
```

②

```
#
include
<
stdio.h
>
main
(
)
{
printf
(
"Hellow¥n"
)
;
}
```

※フラグ（AJCTKFLG_INCSPACE）を指定していない場合、全ての空白，コメント，改行や行継続記号（¥）はトークンとして読み出されません。

38.1. 空白，コメント，改行や行の継続記号（¥）もトークンとして読み出す

AjcCtkCreate()やAjcCtkSetFlag()で フラグ（AJCTKFLG_INCSPACE）を指定した場合は、全ての空白，改行コード，コメントや行継続記号(¥)もトークンとして読み出されます。

①

```
#define M(a, buffer) a = sizeof¥
buffer;
extern int func(); // 外部関数
```

②

```
#
define
 
M
(
a
,
 
buffer
)
 
a
 
=
 
sizeof
¥
↓
buffer
;
↓
```

```
extern
 
int
 
func
(
)
;
 
// 外部関数
↓
```

 ： 空白(0x20)を示します。

↓ ： 改行(0x0A)を示します。

※フラグ（AJCTKFLG_INCSPACE）を指定した場合、全ての文字をトークンとして読み出すことになります。
　従って、読み出したトークンをそのままファイルに出力すると、元のファイルと同じになります。

38.2. プリプロセス文情報

読み出した字句には、フラグ情報として、以下のプリプロセス文情報が付属します。

#	フラグ名	内容
1	AJCTKF_PREPRO	プリプロセス文中のトークンであることを示します
2	AJCTKF_PPTOP	プリプロセス文の先頭トークン（#）であることを示します
3	AJCTKF_MACNAME	#define 文により定義されたマクロ名
4	AJCTKF_MACWITHARG	引数付きのマクロ名（CTKF_MACNAME も同時にセットされる）
5	AJCTKF_MACARG	引数付きマクロの仮引数部分（前後の '(' と ')' も含む）
6	AJCTKF_MACBODY	マクロボディ
7	AJCTKF_NXTSPC	トークンの次に空白ありを示すフラグ
8	AJCTKF_WIDECHAR	ワイド文字（L'X' / L"XXX"）フラグ

例えば、以下の①ようなソースプログラムテキストは、②に示す字句とフラグ情報に分解して順次読み出されます。

①

```
#define  ADD(A,  B)  ¥
                (A  +  B)
#define SUB     (a  -  b)
```

②

字句	フラグ（ AJCTKF_*xxxxx* ）						
	PREPRO	PPTOP	MACNAME	MACWITHARG	MACARG	MACBODY	NXTSPC
#	1	1	0	0	0	0	0
define	1	0	0	0	0	0	1
ADD	1	0	1	1	0	0	0
(	1	0	0	0	1	0	0
A	1	0	0	0	1	0	0
,	1	0	0	0	1	0	1
B	1	0	0	0	1	0	0
)	1	0	0	0	1	0	1
(	1	0	0	0	0	1	0
A	1	0	0	0	0	1	1
+	1	0	0	0	0	1	1
B	1	0	0	0	0	1	0
)	1	0	0	0	0	1	0
#	1	1	0	0	0	0	0
define	1	0	0	0	0	0	1
SUB	1	0	1	0	0	0	1
(	1	0	0	0	0	1	0
a	1	0	0	0	0	1	1
-	1	0	0	0	0	1	1
b	1	0	0	0	0	1	0
)	1	0	0	0	0	1	0

38.3. #include 文のヘッダファイル名情報

「#include <ファイル名>」で指定された、「ファイル名」は、特別な字句（EAJCTK_PATHNAME：パス名）として認識します。
例えば、以下の①ようなソースプログラムテキストは、②に示す字句情報に分解して順次読み出されます。

①

```
#include <stdio.h>
#include "local.h"
```

※「#include <・・・>」は、行継続記号（¥）を使用しないで、1行で記述されていなければなりません。

②

字句	字句コード
#	EAJCTK_DLM_SHARP
include	EAJCTK_RSV_INCLUDE
<	EAJCTK_DLM_LO
stdio.h	EAJCTK_PATHNAME
>	EAJCTK_DLM_HI
#	EAJCTK_DLM_SHARP
include	EAJCTK_RSV_INCLUDE
"local.h"	EAJCTK_STR_QUOTE

38.4. サポートＡＰＩ

Ｃ言語の字句分解のサポートＡＰＩ一覧を以下に示します。

#	関 数 名	内容	備考
1	AjcCtkCreate	インスタンス生成	
2	AjcCtkReset	リセット（インスタンス生成時の状態に戻す）	
3	AjcCtkSetFlag AjcCtkGetFlag	機能フラグの設定／取得	
4	AjcCtkSetCallBack	コールバック設定	
5	AjcCtkSetCallBackParam	コールバックパラメタ設定	
6	AjcCtkDelete	インスタンス消去	
7	AjcCtkGetToken	Ｃ言語ソースプログラムテキストから、字句情報を順次取り出します。	
8	AjcCtkPeekToken	Ｃ言語ソースプログラムテキストから、現在の字句情報を取得します。	
9	AjcCtkGetTokenString	トークンコードに対応する文字列の取得	
10	AjcCtkGetReplicatedHandle	インスタンスのコピーを取得	

38.4.1. インスタンス生成（AjcCtkCreate）

形　式 ： HAJCTK　AjcCtkCreate (UI flag, PCBAJCTK cbGetLine, UX cbp);

引　数 ： flag　　- 機能フラグ
cbGetLine - ソーステキスト行読み出し用コールバック関数のアドレス（後でコールバックを設定する場合はNULL）
cbp　　- コールバックパラメタ

説　明 ： インスタンスを生成し、初期化します。
flagには、０、あるいは、以下のシンボルを指定します。(論理和で複数指定可)

シンボル	内容	備考
AJCTKFLG_CPLUSPLUS	Ｃ＋＋のトークンを認識する	
AJCTKFLG_INCSPACE	空白，改行，コメント、改行や行継続記号（¥）もトークンに含める	
AJCTKFLG_INCSYM_DOLLAR	シンボルの英字としてドルマーク($)を含める	
AJCTKFLG_INCSYM_ATMARK	シンボルの英字としてアットマーク(@)を含める	
AJCTKFLG_INCSYM_MBSTR	シンボルの英字としてマルチバイト文字を含める	
AJCTKFLG_APOST_NOESC	アポストロフィ(')で囲まれた文字列は、エスケース文字(¥)を認識しない	
AJCTKFLG_DOTSYMBOL	シンボル＋Dot(.)＋数字列を１つのシンボルとみなす（ex. P3.0） （Ｃコンパイラによっては、このようなポート指定を行う場合がある）	
AJCTKFLG_ASMHEX	アセンブラ記述の１６進数（末尾にH付加，ex 0FFh）を認識可能とする	EAJCTK_VAL_HEXA_H
AJCTKFLG_LBF_1K	テキストファイル行の読み出しバッファサイズ ＝　1KB	実際にはBit31～24で、KB数(1～255KB)を指定します。 デフォルトは1KBです
AJCTKFLG_LBF_2K	テキストファイル行の読み出しバッファサイズ ＝　2KB	
AJCTKFLG_LBF_4K	テキストファイル行の読み出しバッファサイズ ＝　4KB	
AJCTKFLG_LBF_8K	テキストファイル行の読み出しバッファサイズ ＝　8KB	
AJCTKFLG_LBF_16K	テキストファイル行の読み出しバッファサイズ ＝　16KB	
AJCTKFLG_LBF_32K	テキストファイル行の読み出しバッファサイズ ＝　32KB	
AJCTKFLG_LBF_64K	テキストファイル行の読み出しバッファサイズ ＝　64KB	
AJCTKFLG_LBF_128K	テキストファイル行の読み出しバッファサイズ ＝　128KB	

戻り値 ： ≠NULL - 成功　（インスタンスハンドルを返します）
＝NULL - 失敗

コールバック

cbGetLine（ソーステキスト行読み出し）

形　式 ： BOOL CALLBACK ***cbGetLine***(UTP pBuf, UI lBuf, UX cbp);

引　数 ： pBuf -読み出したソーステキスト行を格納するバッファのアドレス
lBuf - 読み出したソーステキスト行を格納するバッファのバイト数／文字数
cbp　- CtkInitで指定したｘｐ引数の値

説　明 ： このコールバック関数は、ソースプログラムテキストを１行分読み出すためにコールされます。
読み出したソーステキスト行は、pBuf,lBufで指定されたバッファに格納します。

戻り値 ： TRUE　- 有効なソーステキスト（ライン）を読み出した
FALSE - ＥＯＦ（ソーステキストの終端を検出した）

38.4.2. リセット（AjcCtkReset）

形　式 ： BOOL　AjcCtkReset (HAJCTK hCtk);

引　数 ： hCtk - インスタンスハンドル

説　明 ： AjcCtkCreate()で、インスタンス生成時の状態に戻します。
解析中の字句情報や、読み出した行テキストは破棄されます。

戻り値 ： TRUE　- 成功
FALSE - 失敗

38.4.3. 機能フラグの設定／取得（AjcCtk{Set|Get}Flag）

形　式 ： BOOL　AjcCtkSetFlag (HCTK hCtk, UI flag);　--- 機能フラグ設定
UI　　AjcCtkGetFlag (HCTK hCtk); ---------- 機能フラグ取得

引　数 ： hCtk　　- インスタンスハンドル
flag　　- 機能フラグ (AJCTKFLG_XXXXX, 詳細は AjcCtkCreate()参照)

説　明 ： 機能フラグの設定／取得を行います。
AjcCtkCreate()の「flag」引数で設定されている機能フラグを再設定／取得します。

戻り値 ： 設定時：TRUE　- 成功　　取得時：現在設定されている機能フラグ(AJCTKFLG_XXXXX)
FALSE - 失敗　　エラーの場合は、０を返します。

38.4.4. コールバック設定（AjcCtkSetCallBack）

形　式 ： BOOL　AjcCtkSetCallBack (HAJCTK hCtk, PCBAJCTKA cbGetLine, UX cbp);

引　数 ： hCtk　　- インスタンスハンドル
cbGetLine - ソーステキスト行読み出し用コールバック関数のアドレス
cbp　　- コールバックパラメタ

説　明 ： コールバック情報を設定します。
AjcCtkCreate()の「cbGetLine」「cbp」引数で設定されているコールバック情報を再設定します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

38.4.5. コールバックパラメタ設定（AjcCtkSetCallBackParam）

形　式 ： BOOL　AjcCtkSetCallBackParam (HAJCTK hCtk, UX cbp);

引　数 ： hCtk　　- インスタンスハンドル
cbp　　- コールバックパラメタ

説　明 ： コールバックパラメタを設定します。
AjcCtkCreate()の「cbp」引数で設定されているコールバックパラメタを再設定します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

38.4.6. インスタンス消去（AjcCtkDelete）

形　式 ： BOOL　AjcCtkDelete (HAJCTK hCtk);

引　数 ： hCtk - インスタンスハンドル

説　明 ： AjcCtkCreate で動的に確保した、インスタンスワーク領域を解放します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

38.4.7. 字句の読み出し（AjcCtkGetToken）

形　式 ： BOOL　AjcCtkGetToken (HAJCTK hCtk, UTP pStrBuf, UI lStrBuf);

引　数 ： hCtk　　- インスタンスハンドル
pStrBuf - 読み出した字句（文字列）を格納するバッファのアドレス（不要時はNULL）
lSTrBuf - 読み出した字句（文字列）を格納するバッファの文字数

説　明 ： ソースプログラムテキストから、順次、字句情報を１つづつ読み出します。
本関数を実行後、以下のマクロにより、字句文字列以外にも、字句に関する情報を取得できます。

AJCTK_ERROR(hCtk) -------- エラーコードを返します（※1（本関数はエラーを返さないため、このマクロでエラーをチェックします））
AJCTK_TOKEN(hCtk) -------- トークンコードを返します（※2）
AJCTK_SUFFIX(hCtk)-------- 数値定数のサフィックスコードを返します（※3）
AJCTK_FLAG (hCtk) -------- フラグ情報を返します　　（※4）
AJCTK_LINE (hCtk) -------- 当該字句が存在する、ソースプログラム上の行番号を返します
AJCTK_LOC(hCtk)　-------- 当該字句が存在する、ライン上の桁位置を返します（タブも1桁として計算）（※5）
AJCTK_POS(hCtk, *TabStep*) - 当該字句が存在する、タブ文字(0x09)を考慮したライン上の桁位置を返します（*TabStep*はタブステップ幅）（※5）

※1：エラーコードの内容は、以下の組み合わせです。

#	名　称	値	内容	戻り値
1	AJCTKERR_OK	0	正常	TRUE
2	AJCTKERR_NULLPTR	0x8000	ＮＵＬＬポインタが指定された	
3	AJCTKERR_ZEROSIZE	0x4000	バッファサイズがゼロ	
4	AJCTKERR_MEMALLOC	0x2000	メモリ割り当て失敗（内部ワークメモリの確保失敗）	FALSE
5	AJCTKERR_INVALID_OCTALNUMBER	0x1000	不正な８進数	TRUE
6	AJCTKERR_INVALID_HEXADECIMAL	0x0800	不正な１６進数	
7	AJCTKERR_INVALID_REALNUMBER	0x0400	不正な実数表現	
8	AJCTKERR_INVALID_SUFFIX	0x0200	不正なサフィックス	
9	AJCTKERR_INVALID_DELIMITER	0x0100	不正なデリミタ	
10	AJCTKERR_LFINSTR	0x0080	文字列中に改行が含まれている	
11	AJCTKERR_BUFOVER	0x0040	トークンバッファオーバー	
12	AJCTKERR_EOFINCOMMENT	0x0020	コメント中にＥＯＦ検出（ファイル内でコメントが閉じていない）	

※2：トークンコードは、以下のとおりです。（typedef enum｛・・・｝**EAJCTKCODE**, *LPEAJCTKCODE;）
シンボル（網掛けの部分は、C++用の字句を意味します）

コード名	トークン	コード名	トークン	コード名	トークン
EAJCTK_RSV_ASM	asm	EAJCTK_RSV_EXTERN	extern	EAJCTK_RSV_SHORT	short
EAJCTK_RSV_AUTO	auto	EAJCTK_RSV_FLOAT	float	EAJCTK_RSV_SIGNED	signed
EAJCTK_RSV_BREAK	break	EAJCTK_RSV_FOR	for	EAJCTK_RSV_SIZEOF	sizeof
EAJCTK_RSV_CASE	case	EAJCTK_RSV_FRIEND	friend	EAJCTK_RSV_STATIC	static
EAJCTK_RSV_CATCH	catch	EAJCTK_RSV_GOTO	goto	EAJCTK_RSV_STACAST	static_cast
EAJCTK_RSV_CHAR	char	EAJCTK_RSV_IF	if	EAJCTK_RSV_STRUCT	struct
EAJCTK_RSV_CLASS	class	EAJCTK_RSV_IFDEF	ifdef	EAJCTK_RSV_SWITCH	switch
EAJCTK_RSV_CONST	const	EAJCTK_RSV_IFNDEF	ifndef	EAJCTK_RSV_TEMPLATE	template
EAJCTK_RSV_CONCAST	const_cast	EAJCTK_RSV_INCLUDE	include	EAJCTK_RSV_THIS	this
EAJCTK_RSV_CONTINUE	continue	EAJCTK_RSV_INLINE	inline	EAJCTK_RSV_THROW	throw
EAJCTK_RSV_DEFAULT	default	EAJCTK_RSV_INT	int	EAJCTK_RSV_TRY	try
EAJCTK_RSV_DEFINE	define	EAJCTK_RSV_LONG	long	EAJCTK_RSV_TYPEDEF	typedef
EAJCTK_RSV_DEFINED	defined	EAJCTK_RSV_NEW	new	EAJCTK_RSV_UNION	union
EAJCTK_RSV_DELETE	delete	EAJCTK_RSV_OPERATOR	operator	EAJCTK_RSV_UNDEF	undef
EAJCTK_RSV_DO	do	EAJCTK_RSV_PRAGMA	pragma	EAJCTK_RSV_UNSIGNED	unsigned
EAJCTK_RSV_DOUBLE	double	EAJCTK_RSV_PRIVATE	private	EAJCTK_RSV_VIRTUAL	virtual
EAJCTK_RSV_DYNCAST	dynamic_cast	EAJCTK_RSV_PROTECTED	protected	EAJCTK_RSV_VOID	void
EAJCTK_RSV_ELIF	elif	EAJCTK_RSV_PUBLIC	public	EAJCTK_RSV_VOLATILE	volatile
EAJCTK_RSV_ELSE	else	EAJCTK_RSV_REGISTER	register	EAJCTK_RSV_WCHAR_T	wchar_t
EAJCTK_RSV_ENDIF	endif	EAJCTK_RSV_REICAST	reinterpret_cast	EAJCTK_RSV_WHILE	while
EAJCTK_RSV_ENUM	enum	EAJCTK_RSV_RETURN	return	EAJCTK_USR_NAME	ユーザ定義名

数値定数

コード名	トークン
EAJCTK_VAL_DECIMAL	１０進数
EAJCTK_VAL_HEXA	１６進数
EAJCTK_VAL_OCTAL	８進数
EAJCTK_VAL_REAL	実数
EAJCTK_VAL_HEXA_H	16進数(末尾H)
EAJCTK_VAL_INVALID	不正な数値表現

文字列

コード名	トークン
EAJCTK_STR_QUOTE	″ ･･･ ″
EAJCTK_STR_APOST	’ ･･･ ’

パス名

コード名	トークン
EAJCTK_PATHNAME	ファイルパス名

「#include <ファイルパス名>」の’ファイルパス名’部分を示します。
但し、「#include ″ファイルパス名″」の場合は、ECTK_STR_QUOTEとなります。

デリミタ（網掛けの部分は、C++用の字句を意味します）

コード名	トークン	コード名	トークン	コード名	トークン
EAJCTK_DLM_LSPART	(	EAJCTK_DLM_DOT	.	EAJCTK_DLM_OREQ	|=
EAJCTK_DLM_LMPART	{	EAJCTK_DLM_SHREQ	>>=	EAJCTK_DLM_XOREQ	^=
EAJCTK_DLM_LLPART	[	EAJCTK_DLM_SHLEQ	<<=	EAJCTK_DLM_SHARP2	##
EAJCTK_DLM_RSPART	)	EAJCTK_DLM_SHR	>>	EAJCTK_DLM_PLUS2	++
EAJCTK_DLM_RMPART	}	EAJCTK_DLM_SHL	<<	EAJCTK_DLM_MINUS2	--
EAJCTK_DLM_RLPART	]	EAJCTK_DLM_EQEQ	==	EAJCTK_DLM_SHARP	#
EAJCTK_DLM_SCOPE	::	EAJCTK_DLM_NOTEQ	!=	EAJCTK_DLM_PLUS	+
EAJCTK_DLM_IDDOT	.*	EAJCTK_DLM_LOEQ	<=	EAJCTK_DLM_MINUS	-
EAJCTK_DLM_IDARROW	->*	EAJCTK_DLM_HIEQ	>=	EAJCTK_DLM_MULT	*
EAJCTK_DLM_LAND	&&	EAJCTK_DLM_LO	<	EAJCTK_DLM_DIV	/
EAJCTK_DLM_LOR	||	EAJCTK_DLM_HI	>	EAJCTK_DLM_MOD	%
EAJCTK_DLM_ARROW	->	EAJCTK_DLM_PLUSEQ	+=	EAJCTK_DLM_AND	&
EAJCTK_DLM_QUEST	?	EAJCTK_DLM_MINUSEQ	-=	EAJCTK_DLM_OR	|
EAJCTK_DLM_COLON	:	EAJCTK_DLM_MULTEQ	*=	EAJCTK_DLM_XOR	^
EAJCTK_DLM_SEMICOL	;	EAJCTK_DLM_DIVEQ	/=	EAJCTK_DLM_EQ	=
EAJCTK_DLM_COMMA	,	EAJCTK_DLM_MODEQ	%=	EAJCTK_DLM_NOT	~
EAJCTK_DLM_VARG	...	EAJCTK_DLM_ANDEQ	&=	EAJCTK_DLM_LNOT	!

空白／改行／コメント／行継続記号″￥″（AjcCtkCreate で flag に AJCTKFLG_INCSPACE 指定時のみ）

コード名	トークン	コード名	トークン	コード名	トークン
EAJCTK_SPACE	空白／タブ文字列	EAJCTK_COMMENT	/*･･･*/	EAJCTK_LINECONT	行継続記号(￥)
EAJCTK_EOL	改行文字列	EAJCTK_LINECOMMENT	//･･･		

EAJCTK_COMMENT では、コメントが複数行にまたがる場合は、トークンが分割されます。

また、以下のマクロで取得したトークンコード(tkn)を指定することにより、語句の種別を判定できます。

AJCTKIS_USRSYM(tkn)　- トークンがユーザ定義名である場合、０以外の値を返します。
AJCTKIS_RSVSYM(tkn)　- トークンが予約名である場合、０以外の値を返します。
AJCTKIS_VALUE(tkn)　- トークンが数値定数である場合、０以外の値を返します。
AJCTKIS_STRING(tkn)　- トークンが文字列である場合、０以外の値を返します。
AJCTKIS_PATHNAME(tkn) - トークンがヘッダファイルのパス名である場合、０以外の値を返します。
AJCTKIS_DELIMIT(tkn)　- トークンがデリミタである場合、０以外の値を返します。
AJCTKIS_SYMBOL(tkn)　- トークンがユーザ定義名／予約名である場合、０以外の値を返します。
AJCTKIS_VALSYM(tkn)　- トークンがユーザ定義名／予約名／数値定数である場合、０以外の値を返します。
AJCTKIS_SPACE(tkn)　- トークンが空白／改行／コメント／行継続記号(￥)である場合、０以外の値を返します。

※３：サフィックス情報の内容は、以下の組み合わせです。

#	名称	値	内容	備考
1	EAJCTKSUF_NONE	0	サフィックスなし	
2	EAJCTKSUF_L	1	L	long
3	EAJCTKSUF_UL	2	UL / LU	unsigned long
4	EAJCTKSUF_LL	3	LL	long long
5	EAJCTKSUF_ULL	4	ULL	unsigned long long
6	EAJCTKSUF_FLOAT	5	F	float
7	EAJCTKSUF_DOUBLE	6	D	double
8	EAJCTKSUF_LDOUBLE	7	L	long double

※５：バイト文字モード時は、全角文字を２桁として計算します、
ワイド文字モード時は、全角文字も１桁として計算します。

※4：フラグ情報の内容は、以下の組み合わせです。

#	フラグ名	値	内容
1	AJCTKF_PREPRO	0x80	プリプロセス文中のトークンであることを示します
2	AJCTKF_PPTOP	0x40	プリプロセス文の先頭トークン（#）であることを示します
3	AJCTKF_MACNAME	0x20	#define 文により定義されたマクロ名
4	AJCTKF_MACWITHARG	0x10	引数付のマクロ名（CTKF_MACNAME も同時にセットされる）
5	AJCTKF_WIDECHAR	0x08	ワイド文字列（ L'・・・' / L"・・・" ）

戻り値 ： TRUE - 有効な字句を読み出した（エラーコードは、「AJCTK_ERROR(hCtk)」により取得します）
FALSE - ＥＯＦ ／ 内部ワークメモリ割り当て失敗（「AJCTK_ERROR(hCtk)」により「AJCTKERR_MEMALLOC」を返します）

備　考 ： 内部ワークメモリ割り当て失敗「AJCTKERR_MEMALLOC」は、UNICODE モード時のみ発生します。

38.4.8. 現在の字句情報の取得（AjcCtkPeekToken）

形　式 ： BOOL AjcCtkPeekToken (HCTK hCtk, UTP pStrBuf, UI lStrBuf);

引　数 ： hCtk - インスタンスハンドル
pStrBuf - 読み出した字句（文字列）を格納するバッファのアドレス（不要時は NULL）
lSTrBuf - 読み出した字句（文字列）を格納するバッファの文字数

説　明 ： ソースプログラムテキストから、現在の字句情報を取得します。
「AjcCtkGetToken」は、字句情報を１つずつ進めるのに対し、「AjcCtkPeekToken」は、現在の字句情報を取得するだけであり、次の字句へは進みません。（つまり、「AjcCtkPeekToken」直後の「AjcCtkGetToken / CtkPeekToken」は同一字句情報を返します）

戻り値 ： TRUE - 有効な字句を読み出した（エラーコードは、「AJCTK_ERROR(hCtk)」により取得します）
FALSE - ＥＯＦ ／ 内部ワークメモリ割り当て失敗（「AJCTK_ERROR(hCtk)」により「AJCTKERR_MEMALLOC」を返します）

備　考 ： 内部ワークメモリ割り当て失敗「AJCTKERR_MEMALLOC」は、UNICODE モード時のみ発生します。

38.4.9. トークンコードに対応する文字列の取得（AjcCtkGetTokenString）

形　式 ： UTP AjcCtkGetTokenString (EAJCTKCODE tkn);

引　数 ： tkn - トークンコード（AjcCtkGetToken()後、AJCCTK_TOKEN マクロで得た値）

説　明 ： トークンコードに対応する文字列のアドレスを返します。
トークンコードは、予約名（EAJCTK_RSV_*XXXXX*）あるいは、デリミタ（EAJCTK_DLM_*XXXXX*）でなければなりません。
不正なトークンコードを指定した場合は、NULL を返します。

戻り値 ： ≠NULL - トークンコードに対応する文字列のアドレス
＝NULL - 不正なトークンコード

38.4.10. インスタンスの複製（AjcCtkGetReplicatedHandle）

形　式　：　HAJCTK　AjcCtkGetReplicatedHandle(HAJCTK hCtk)

引　数　：　hCtk　　- インスタンスハンドル

説　明　：　現在のインスタンスを複製したインスタンスハンドル(hDup)を返します。
複製したインスタンスを使用することにより。現在の読み出し位置を停止したまま、先読みを行うことができます。

複製したハンドル(hDup)を使用した場合、元のハンドル(hCtk)操作の続きとなります。
例えば、AjcCtkGetToken(hDup・・)を実行した場合、元のハンドルで最後に取得したトークンの次のトークンを取得します。
元のインスタンスは更新されませんので、AjcCtkGetToken(hCtk・・)は、最後に実行した AjcCtkGetToken(hCtk) の次のトークンを取得します。
複製したインスタンスハンドルは、AjcCtkDelete(hDup)により消去しなければなりません。

戻り値　：　≠NULL – 複製したインスタンスのハンドル(hDup)
＝NULL - 不正なトークンコード

38.5. サンプルプログラム

38.5.1. SW_Ctk01C（C言語ソースファイルからコメント削除）

次のサンプルプログラムは、C／C＋＋言語ソースプログラムファイルを入力し、全てのコメントを削除したテキストを標準出力へ出力します。(コマンドの第1引数にファイルパスを指定します)

入力例

```
// SW_Ctk01C.c

#pragma warning(disable:4996)
#include    <AjrCstXX.h>
#include    <stdio.h>
#include    <tchar.h>

//============================================================
//  ソーステキスト読み出し
//
//  引  数  :  pBuf    - ソーステキストを格納するバッファのアドレス
//             lBif    - ソーステキストを格納するバッファの文字数
//             xp      - コールバックパラメタ
//
//  戻り値  :  TRUE  - 有効なテキストを読み出した
//             FALSE - ＥＯＦ
//============================================================
static BOOL CALLBACK cbGetLine(UTP pBuf, UI lBuf, UX xp)
{
    return _fgetts(pBuf, lBuf, (FILE *)xp) != NULL;
}

//============================================================
//
//  m a i n
//
//============================================================
int  AjcMain(int argc, UTP argv[])
{
    FILE    *hFile;
    HAJCTK  hCtk;
    UI      tc;
    UT      txt[256];

    do {
        //----- コマンドパラメタ チェック --------------------//
        if (argc <= 1) {
            AjcPrintF(TEXT("No source file name¥n"));
            break;
        }
        //----- ソースファイル オープン ----------------------//
        if (!(hFile = _tfopen(argv[1], TEXT("rt")))) {
            AjcPrintF(TEXT("File %s open failure¥n"), argv[1]);
            break;
        }
        //----- コメントを取り除いたテキスト出力 -------------//
        hCtk = AjcCtkCreate(AJCTKFLG_CPLUSPLUS | AJCTKFLG_INCSPACE,
                                        cbGetLine, (UX)hFile);
        while (AjcCtkGetToken(hCtk, txt, sizeof txt)) {
            tc = AJCTK_TOKEN(hCtk);
            if (tc != EAJCTK_COMMENT && tc != EAJCTK_LINECOMMENT) {
                AjcPrintF(TEXT("%s"), txt);
            }
        }
        //----- インスタンス消去 -----------------------------//
        AjcCtkDelete(hCtk);
        //----- ソースファイル クローズ ----------------------//
        fclose(hFile);

    } while(0);
    getchar();
    return 0;
}
```

出力

```
#pragma warning(disable:4996)
#include    <AjrCstXX.h>
#include    <stdio.h>
#include    <tchar.h>

static BOOL CALLBACK cbGetLine(UTP pBuf, UI lBuf, UX xp)
{
    return _fgetts(pBuf, lBuf, (FILE *)xp) != NULL;
}

int  AjcMain(int argc, UTP argv[])
{
    FILE    *hFile;
    HAJCTK  hCtk;
    UI      tc;
    UT      txt[256];

    do {

        if (argc <= 1) {
            AjcPrintF(TEXT("No source file name¥n"));
            break;
        }

        if (!(hFile = _tfopen(argv[1], TEXT("rt")))) {
            AjcPrintF(TEXT("File %s open failure¥n"), argv[1]);
            break;
        }

        hCtk = AjcCtkCreate(AJCTKFLG_CPLUSPLUS | AJCTKFLG_INCSPACE,
                                        cbGetLine, (UX)hFile);
        while (AjcCtkGetToken(hCtk, txt, sizeof txt)) {
            tc = AJCTK_TOKEN(hCtk);
            if (tc != EAJCTK_COMMENT && tc != EAJCTK_LINECOMMENT) {
                AjcPrintF(TEXT("%s"), txt);
            }
        }

        AjcCtkDelete(hCtk);

        fclose(hFile);

    } while(0);
    getchar();
    return 0;
}
```

```
 1 : // SW_Ctk01C.c
 2 : #include    <AjrCstXX.h>
 3 : #include    <stdio.h>
 4 : #include    <tchar.h>
 5 : #include    <fcntl.h>
 6 :
 7 : //=============================================================================================================//
 8 : //  ソーステキスト読み出し                                                                                     //
 9 : //                                                                                                             //
10 : //  引  数  :  pBuf    - ソーステキストを格納するバッファのアドレス                                            //
11 : //             lBif    - ソーステキストを格納するバッファの文字数                                              //
12 : //             xp      - コールバックパラメタ                                                                  //
13 : //                                                                                                             //
14 : //  戻り値  :  TRUE  - 有効なテキストを読み出した                                                              //
15 : //             FALSE - ＥＯＦ                                                                                  //
16 : //=============================================================================================================//
17 : static BOOL CALLBACK cbGetLine(UTP pBuf, UI lBuf, UX xp)
18 : {
19 :     return AjcFGetS((HAJCFILE)xp, pBuf, lBuf) != NULL;      //  TRUE:継続, FALSE:中止
20 : }
21 :
22 : //=============================================================================================================//
23 : //                                                                                                             //
24 : //  m a i n                                                                                                    //
25 : //                                                                                                             //
26 : //=============================================================================================================//
27 : int  AjcMain(int argc, UTP argv[])
28 : {
29 :     HAJCFILE    hFile;
30 :     HAJCTK      hCtk;
31 :     UI          tc;
32 :     UT          txt[256];
33 :     UT          path[MAX_PATH];
34 :
35 :     AjcSetStdoutMode();
36 :     _tsystem(TEXT("cd"));
37 :
38 :     do {
39 :         AjcGetAppPath(path, AJCTSIZE(path));
40 :         SetCurrentDirectory(path);
41 :         //----- コマンドパラメタ チェック --------------------//
42 :         if (argc >= 2) MAjcStrCpy(path, MAX_PATH, argv[1]);
43 :         else           MAjcStrCpy(path, MAX_PATH, TEXT("Sample.c"));
44 :         //----- ソースファイル オープン ----------------------//
45 :         if ((hFile = AjcFOpen(path, AJCTEC_AUTO)) == NULL) {
46 :             AjcPrintF(TEXT("File %s open failure¥n"), argv[1]);
47 :             break;
48 :         }
49 :         //----- コメントを取り除いたテキスト出力 -------------//
50 :         hCtk = AjcCtkCreate(AJCTKFLG_CPLUSPLUS | AJCTKFLG_INCSPACE,
51 :                                             cbGetLine, (UX)hFile);
52 :         while (AjcCtkGetToken(hCtk, txt, sizeof txt)) {
53 :             tc = AJCTK_TOKEN(hCtk);
54 :             if (tc != EAJCTK_COMMENT && tc != EAJCTK_LINECOMMENT) {
55 :                 _tprintf(TEXT("%s"), txt);
56 :             }
57 :         }
58 :         //----- インスタンス消去 ----------------------------//
59 :         AjcCtkDelete(hCtk);
60 :         //----- ソースファイル クローズ ----------------------//
61 :         AjcFClose(hFile);
62 :
63 :     } while(0);
64 :     AjcPrintF(TEXT("Hit Enter Key!"));
65 :     getchar();
66 :
67 :     return 0;
68 : }
```

38.5.2. SW_Ctk02（字句情報のログ表示）

次のサンプルプログラムは、C／C＋＋言語ソースプログラムファイルを入力し、字句分解を行い、各字句の情報を標準出力へ出力します。

また、「P3.12」のように、「シンボル＋Dot(.)＋数字列」も１つのシンボルとみなします。

実行例

入力ソースプログラム

```
 1 : #include    <windows.h>
 2 : #include    <setupapi.h>
 3 : #pragma comment(lib,"setupapi.lib")
 4 : #define     VAL         (1)
 5 : #define     ADD(a, b)   (a + b)
 6 : struct {int a, b;} s;
 7 :
 8 : main()
 9 : {
10 :     s.a  = VAL;
11 :     P3.12 = 0;
12 : }
```

出力

```
      S U R V S P D S P P M M M M N W
      Y S S A T A E P R P A A A A X I
      M R V L R T L A E T C C C C T D
      B S S U I H I C P O N W A B S E
      O Y Y E N N M E R P A I R O P C
      L M M | G A I | O | M T G D C H
      | | | | | M T | | | E H | Y | A
      | | | | | E | | | | A | | | R
Token | | | | | | | | | R | | | |
Code  | | | | | | | | | G | | | | TokenString

2027  . . . . . 1 . 1 1 . . . . . .  #
021D  1 . 1 . . . . . 1 . . . . 1 .  include
201A  . . . . . 1 . 1 . . . . . . .  <
1000  . . . . . 1 . . 1 . . . . . .  windows.h
201B  . . . . . 1 . 1 . . . . . . .  >
2027  . . . . . 1 . 1 1 . . . . . .  #
021D  1 . 1 . . . . . 1 . . . . 1 .  include
201A  . . . . . 1 . 1 . . . . . . .  <
1000  . . . . . 1 . . 1 . . . . . .  setupapi.h
201B  . . . . . 1 . 1 . . . . . . .  >
2027  . . . . . 1 . 1 1 . . . . . .  #
0223  1 . 1 . . . . . 1 . . . . 1 .  pragma
0100  1 1 . . . . . 1 . . . . . . .  comment
2000  . . . . . 1 . 1 . . . . . . .  (
0100  1 1 . . . . . 1 . . . . . . .  lib
200F  . . . . . 1 . 1 . . . . . . .  ,
0800  . . . . 1 . . . 1 . . . . . .  "setupapi.lib"
2003  . . . . . 1 . 1 . . . . . . .  )
2027  . . . . . 1 . 1 1 . . . . . .  #
020B  1 . 1 . . . . . 1 . . . . 1 .  define
0100  1 1 . . . . . 1 . 1 . . . 1 .  VAL
2000  . . . . . 1 . 1 . . . 1 . . .  (
0400  . . . 1 . . . . 1 . . . 1 . .  1
2003  . . . . . 1 . . 1 . . . 1 . .  )
2027  . . . . . 1 . 1 1 . . . . . .  #
020B  1 . 1 . . . . . 1 . . . . 1 .  define
0100  1 1 . . . . . 1 . 1 1 . . . .  ADD
2000  . . . . . 1 . . 1 . . . 1 . .  (
0100  1 1 . . . . . 1 . . . 1 . . .  a
200F  . . . . . 1 . 1 . . . 1 . 1 .  ,
0100  1 1 . . . . . 1 . . . 1 . . .  b
2003  . . . . . 1 . . 1 . . . 1 . 1  )
2000  . . . . . 1 . 1 . . . . 1 . .  (
0100  1 1 . . . . . 1 . . . . 1 1 .  a
2028  . . . . . 1 . 1 . . . . 1 1 .  +
0100  1 1 . . . . . 1 . . . . 1 . .  b
2003  . . . . . 1 . 1 . . . . 1 . .  )
022F  1 . 1 . . . . . . . . . . 1 .  struct
2001  . . . . . 1 . . . . . . . . .  {
021F  1 . 1 . . . . . . . . . . 1 .  int
0100  1 1 . . . . . . . . . . . . .  a
200F  . . . . . 1 . . . . . . . 1 .  ,
0100  1 1 . . . . . . . . . . . . .  b
200E  . . . . . 1 . . . . . . . . .  ;
2004  . . . . . 1 . . . . . . . . 1  }
0100  1 1 . . . . . . . . . . . . .  s
200E  . . . . . 1 . . . . . . . . .  ;
0100  1 1 . . . . . . . . . . . . .  main
2000  . . . . . 1 . . . . . . . . .  (
2003  . . . . . 1 . . . . . . . . .  )
2001  . . . . . 1 . . . . . . . . .  {
0100  1 1 . . . . . . . . . . . . .  s
2011  . . . . . 1 . . . . . . . . .  .
0100  1 1 . . . . . . . . . . 1 .  a
2030  . . . . . 1 . . . . . . . 1 .  =
0100  1 1 . . . . . . . . . . . . .  VAL
200E  . . . . . 1 . . . . . . . . .  ;
0100  1 1 . . . . . . . . . . . 1 .  P3.12
2030  . . . . . 1 . . . . . . . 1 .  =
0400  . . . 1 . . . . . . . . . . .  0
200E  . . . . . 1 . . . . . . . . .  ;
2004  . . . . . 1 . . . . . . . . .  }
```

```
 1 : // SW_Ctk02C.c
 2 : #include    <AjrCstXX.h>
 3 : #include    <stdio.h>
 4 : #include    <tchar.h>
 5 :
 6 : //=============================================================================================================//
 7 : //  ソーステキスト読み出し                                                                                     //
 8 : //                                                                                                             //
 9 : //  引  数  :  pBuf    - ソーステキストを格納するバッファのアドレス                                            //
10 : //             lBif    - ソーステキストを格納するバッファの文字数                                              //
11 : //             xp      - コールバックパラメタ                                                                  //
12 : //                                                                                                             //
13 : //  戻り値  :  TRUE  - 有効なテキストを読み出した                                                              //
14 : //             FALSE - ＥＯＦ                                                                                  //
15 : //=============================================================================================================//
16 : static BOOL CALLBACK cbGetLine(UTP pBuf, UI lBuf, UX xp)
17 : {
18 :     return _fgetts(pBuf, lBuf, (FILE *)xp) != NULL;
19 : }
20 :
21 : //=============================================================================================================//
22 : //                                                                                                             //
23 : //  m a i n                                                                                                    //
24 : //                                                                                                             //
25 : //=============================================================================================================//
26 : int  AjcMain(int argc, UTP argv[])
27 : {
28 :     FILE    *hFile;
29 :     HAJCTK  hCtk;
30 :     UI      tc, flg;
31 :     UT      txt[256];
32 :     UT      path[MAX_PATH];
33 :
34 :     AjcSetStdoutMode();
35 :
36 :     do {
37 :         AjcGetAppPath(path, AJCTSIZE(path));
38 :         SetCurrentDirectory(path);
39 :         //----- コマンドパラメタ チェック --------------------//
40 :         if (argc >= 2) MAjcStrCpy(path, MAX_PATH, argv[1]);
41 :         else           MAjcStrCpy(path, MAX_PATH, TEXT(".¥¥Sample.c"));
42 :         //----- ソースファイル オープン ----------------------//
43 :         if (_tfopen_s(&hFile, path, TEXT("rt")) != 0) {
44 :             AjcPrintF(TEXT("File %s open failure¥n"), argv[1]);
45 :             break;
46 :         }
47 :         //----- ヘッダテキスト出力 ---------------------------//
48 :         AjcPrintF(TEXT("      S U R  V S P D  S  P P  M M M M  N W              ¥n"));
49 :         AjcPrintF(TEXT("      Y S S  A T A E  P  R P  A A A A  X I              ¥n"));
50 :         AjcPrintF(TEXT("      M R V  L R T L  A  E T  C C C C  T D              ¥n"));
51 :         AjcPrintF(TEXT("      B S S  U I H I  C  P O  N W A B  S E              ¥n"));
52 :         AjcPrintF(TEXT("      O Y Y  E N N M  E  R P  A I R O  P C              ¥n"));
53 :         AjcPrintF(TEXT("      L M M  ｜G A I  ｜ O｜  M T G D  C H              ¥n"));
54 :         AjcPrintF(TEXT("      ｜｜｜  ｜｜M T  ｜ ｜｜ E H｜Y  ｜A              ¥n"));
55 :         AjcPrintF(TEXT("      ｜｜｜  ｜｜E｜  ｜ ｜｜ ｜A｜｜ ｜R              ¥n"));
56 :         AjcPrintF(TEXT("Token ｜｜｜  ｜｜｜｜  ｜ ｜｜ ｜R｜｜ ｜｜              ¥n"));
57 :         AjcPrintF(TEXT("Code  ｜｜｜  ｜｜｜｜  ｜ ｜｜ ｜G｜｜ ｜｜ TokenString  ¥n"));
58 :         AjcPrintF(TEXT("¥n"));
59 :         //----- トークン情報出力 -----------------------------//
60 :         hCtk = AjcCtkCreate(AJCTKFLG_CPLUSPLUS | AJCTKFLG_DOTSYMBOL | AJCTKFLG_ASMHEX, cbGetLine, (UX)hFile);
61 :         while (AjcCtkGetToken(hCtk, txt, sizeof txt)) {
62 :             tc = AJCTK_TOKEN(hCtk);
63 :             AjcPrintF(TEXT("%04X  "), tc);
64 :             AjcPrintF(AJCTKIS_SYMBOL  (tc)    ? TEXT("1") : TEXT("."));
65 :             AjcPrintF(TEXT(" "));
66 :             AjcPrintF(AJCTKIS_USRSYM  (tc)    ? TEXT("1") : TEXT("."));
67 :             AjcPrintF(TEXT(" "));
68 :             AjcPrintF(AJCTKIS_RSVSYM  (tc)    ? TEXT("1") : TEXT("."));
69 :             AjcPrintF(TEXT("  "));
70 :             AjcPrintF(AJCTKIS_VALUE   (tc)    ? TEXT("1") : TEXT("."));
71 :             AjcPrintF(TEXT(" "));
72 :             AjcPrintF(AJCTKIS_STRING  (tc)    ? TEXT("1") : TEXT("."));
73 :             AjcPrintF(TEXT(" "));
74 :             AjcPrintF(AJCTKIS_PATHNAME(tc)    ? TEXT("1") : TEXT("."));
75 :             AjcPrintF(TEXT(" "));
76 :             AjcPrintF(AJCTKIS_DELIMIT (tc)    ? TEXT("1") : TEXT("."));
77 :             AjcPrintF(TEXT("  "));
78 :             AjcPrintF(AJCTKIS_SPACE   (tc)    ? TEXT("1") : TEXT("."));
79 :             AjcPrintF(TEXT("  "));
80 :             flg = AJCTK_FLAG(hCtk);
```

```
 81 :             AjcPrintF(flg & AJCTKF_PREPRO     ? TEXT("1") : TEXT("."));
 82 :             AjcPrintF(TEXT(" "));
 83 :             AjcPrintF(flg & AJCTKF_PPTOP      ? TEXT("1") : TEXT("."));
 84 :             AjcPrintF(TEXT("  "));
 85 :             AjcPrintF(flg & AJCTKF_MACNAME    ? TEXT("1") : TEXT("."));
 86 :             AjcPrintF(TEXT(" "));
 87 :             AjcPrintF(flg & AJCTKF_MACWITHARG ? TEXT("1") : TEXT("."));
 88 :             AjcPrintF(TEXT(" "));
 89 :             AjcPrintF(flg & AJCTKF_MACARG     ? TEXT("1") : TEXT("."));
 90 :             AjcPrintF(TEXT(" "));
 91 :             AjcPrintF(flg & AJCTKF_MACBODY    ? TEXT("1") : TEXT("."));
 92 :             AjcPrintF(TEXT("  "));
 93 :             AjcPrintF(flg & AJCTKF_NXTSPC     ? TEXT("1") : TEXT("."));
 94 :             AjcPrintF(TEXT(" "));
 95 :             AjcPrintF(flg & AJCTKF_WIDECHAR   ? TEXT("1") : TEXT("."));
 96 :             AjcPrintF(TEXT("   "));
 97 :
 98 :             AjcPrintF(TEXT("%s¥n"), txt);
 99 :         }
100 :         //----- インスタンス消去 ----------------------------//
101 :         AjcCtkDelete(hCtk);
102 :         //----- ソースファイル クローズ ---------------------//
103 :         fclose(hFile);
104 :
105 :     } while(0);
106 :
107 :     AjcPrintF(TEXT("Hit Enter Key!"));
108 :     getchar();
109 :
110 :     return 0;
111 : }
```

39. C言語のプリコンパイル

C言語のソースプログラムテキストを入力し、プリプロセス文を解決します。
解決するプリプロセス文は以下のとおりです。

- #include
- #define
- #if
- #ifdef
- #ifndef
- #elif
- #else
- #endif

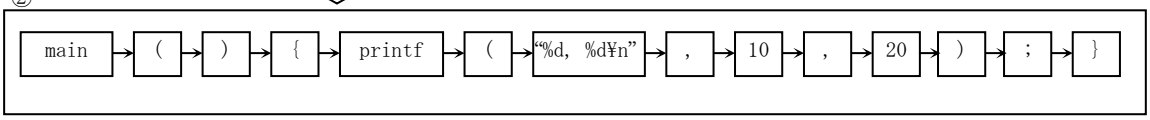
上記以外のプリプロセス文（#pragma 等）の解決は行われません。
プリプロセス文を解決した（プリコンパイルした）結果は、メモリ上にトークンストリームとして出力されます。

例えば、以下の①ようなソースプログラムテキストは、プリコンパイルされ ②に示すトークンストリームとして出力されます。

①

```
#define  MAC(A, B)  printf("%d, %d¥n", A, B)
main( )
{
    #ifdef MAC
        MAC(10, 20);
    #else
        printf("MAC not defined¥n");
    #endif
}
```

②



#define, #ifdef, #else や#endif（マクロや条件コンパイル）が解決されて、以下のようなソースイメージが作成されます。

```
main( )
{
    printf("%d, %d¥n", 10, 20);
}
```

39.1. プリコンパイルオプション

AjcPpcSetOption()により、プリコンパイル動作や出力ファイルの形式に関するオプションを指定できます。
指定できるオプションは以下のとおりです。(JCPPC_FLG_NONE を指定するか、#2～#4 のシンボルの合成値を指定します)

#	オプション	内容
1	AJCPPC_FLG_NONE	オプション無し
2	AJCPPC_FLG_AUTO_SEARCH	インクルードファイル自動検索
3	AJCPPC_FLG_ONCE	同一インクルードファイルを１回だけ読み出す
4	AJCPPC_FLG_GENALL	非生成部分（条件コンパイル＝偽の部分）もファイル出力する

39.1.1. インクルードファイル自動検索 (AJCPPC_FLG_AUTO_SEARCH)

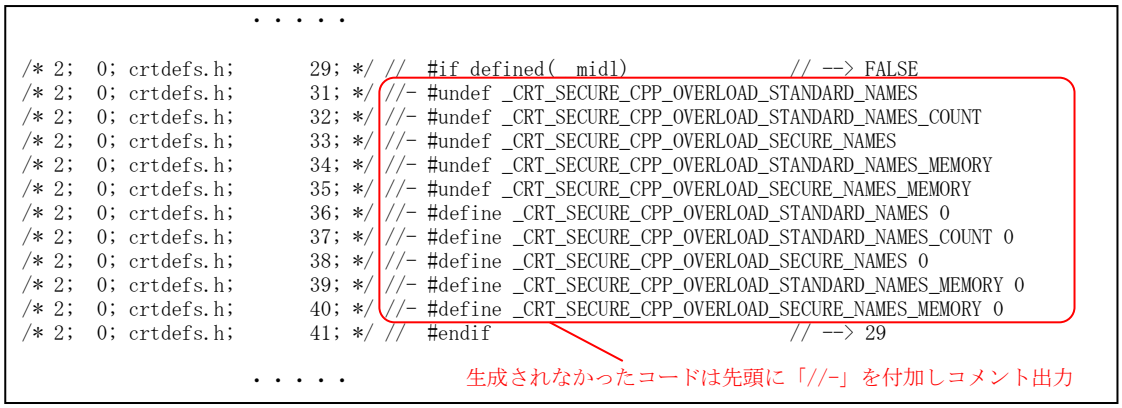
インクルードファイルをベースフォルダの下から自動的に検索します。
ベースフォルダは、インスタンスの生成（AjcPpcCreate()）時に、pBasePath 引数で指定します。
ベースフォルダには、自動的に検索したいインクルードファイル群を含む、最上位フォルダを指定します。
このオプションは、デフォルトで有効となっています。

39.1.2. 同一インクルードファイルを１回だけ読み出す (AJCPPC_FLG_ONCE)

同じインクルードファイルは、最初の１回だけ読み出すように制御します。
このオプションは、デフォルトで有効となっています。

39.1.3. 非生成部分（条件コンパイル＝偽の部分）もファイル出力する (AJCPPC_FLG_GENALL)

条件コンパイル(#if, #elif, #ifdef, #ifndef, #else, #endif)により、生成されなかった部分も、コメント出力するように制御します。
生成されなかった部分のコードは、先頭に「//-」を付加してコメント化します。(下図参照)



このオプションは、デフォルトで無効となっています。

39.2. プリコンパイル結果のファイル出力

AjcPpcCompile()によりプリコンパイルを実行後、AjcPpcTokenStreamToFile()を実行すると、プリコンパイル結果をファイルへ出力します。

元ソース中のコメントは全て削除された結果が、ファイルへ出力されます。また、デリミタとしての空白は必要最小限となります。

以降、出力するファイルの内容について説明します。

39.2.1. インクルードファイルの展開／非展開

AjcPpcTokenStreamToFile()の引数でで、fExpInc=TRUE を指定すると、読み出したインクルードファイルの内容をファイルへ出力します。

この場合、行の先頭に「インクルードファイルのネスト値」「マクロ展開のネスト値」「ファイル名」「行番号」が付加されます。

元ソース部分（インクルードファイルのネスト値＝０の部分）の行番号は、元ソースと一致します。

プリコンパイル結果の出力ファイル（fExpInc=TRUE）

```
/* "..¥SampleInp.c" */
/* Include-Nest; Macro-Nest; SourceFile; LineNumber; */

/* 0;  0; SampleInp.c;      1; */     #pragma warning(disable:4996)
/* 0;  0; SampleInp.c;      2; */
/* 0;  0; SampleInp.c;      3; */ //  #include <stdio.h>
/* 1;  0; stdio.h;         15; */     #pragma once
/* 1;  0; stdio.h;         17; */ //  #ifndef _INC_STDIO                // --> TRUE
/* 1;  0; stdio.h;         18; */ //  #define _INC_STDIO
/* 1;  0; stdio.h;         20; */ //  #include <crtdefs.h>
/* 2;  0; crtdefs.h;       16; */ //  #ifndef _CRTIMP                   // --> TRUE
/* 2;  0; crtdefs.h;       17; */ //  #ifdef _DLL                       // --> FALSE
/* 2;  0; crtdefs.h;       18; */ //- #define _CRTIMP __declspec(dllimport)
/* 2;  0; crtdefs.h;       19; */ //  #else                             // --> TRUE
/* 2;  0; crtdefs.h;       20; */ //  #define _CRTIMP
/* 2;  0; crtdefs.h;       21; */ //  #endif                            // --> 17
/* 2;  0; crtdefs.h;       22; */ //  #endif                            // --> 16
```

・・・・・

ファイル名　　行番号

マクロ展開のネスト値（０はマクロ展開無しを意味する）

インクルードファイルのネスト値（０は、元ソースを意味する）

fExpInc=FALSE を指定した場合は、元ソースのプリコンパイル結果のみファイル出力します。

行の先頭に「インクルードファイルのネスト値」「マクロ展開のネスト値」「ファイル名」「行番号」は付加しません。

出力したファイルの行番号は、元ソースと一致します。（元ソースと出力したファイルの行数は一致します）

プリコンパイル結果の出力ファイル（fExpInc=FALSE）

```
    #pragma warning(disable:4996)

    #include <stdio.h>

//  #define OUTER(V1,V2,OUT)  OUT.x=V1.y*V2.z-V1.z*V2.y;                    ¥ //
//                              OUT.y=V1.z*V2.x-V1.x*V2.z;                  ¥ //
//                              OUT.z=V1.x*V2.y-V1.y*V2.x


    typedef struct{double x,y,z;}VEC;

    int main(int argc,char*argv[])
    {
        VEC v1={1.0,2.0,3.0};
        VEC v2={-3.0,-2.0,-1.0};
        VEC outer;

        outer.x=v1.y*v2.z-v1.z*v2.y;outer.y=v1.z*v2.x-v1.x*v2.z;outer.z=v1.x*v2.y-v1.y*v2.x;

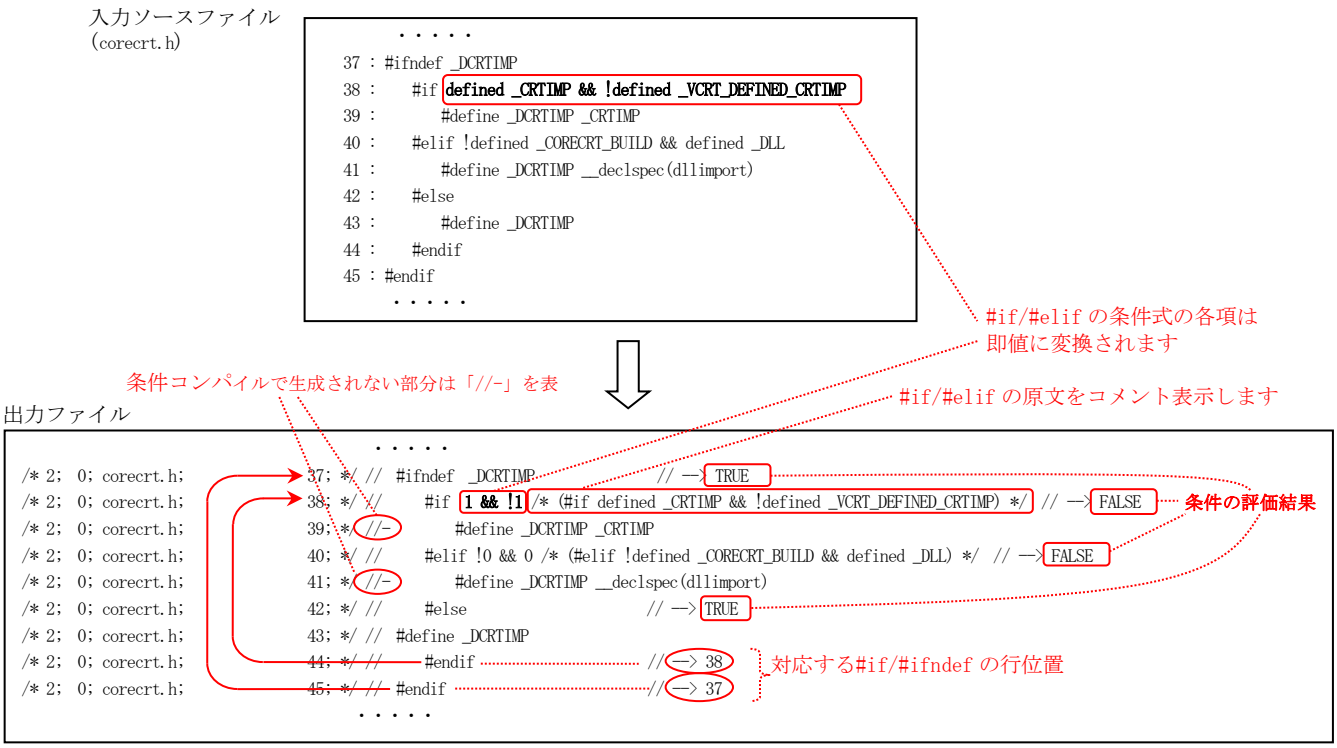
//      #ifdef _DEBUG                   // --> FALSE
//-         printf("_MSC_VER = %d¥n",_MSC_VER);
//      #endif                          // --> 20

        printf("Outer = (%.3f, %.3f, %.3f)¥n",outer.x,outer.y,outer.z);

        return 0;
    }
```

39.2.2. 条件コンパイルの真偽値の表示

#if, #elif, #ifdef, #ifndef, #else の条件の真偽値を「TRUE」「FALSE」でコメント出力します。
#endif の場合は、対応する#if/#ifdef/#ifndef の行番号を表示します。
尚、既に偽条件(FALSE)のネスト中である場合は「During FALSE cond.」と表示します。
その他、下図に示す情報を表示します。



※ #endif で対応する#if/#ifdef/#ifndef の行番号を表示できるのは、行番号の差が 65530 の範囲内である場合に限ります。
65530 行を超えて離れている場合は、「--> ??? (Before nnn)」と表示します。(不明(nnn 行以前) の意味)

※ 以下のプリプロセス文と非生成部分はコメント表示します。

- #if
- #elif
- #ifdef
- #endif
- #ifndef
- #define

39.3. トークン情報の形式

個々のトークンは、以下の構造体により表現されます。

```
typedef struct _AJCPPCTKNNODE {
    struct _AJCPPCTKNNODE *pNxt;          //  次ノードポインタ
    UI          seq;                      //  ノード順序番号
    UBP         pFile;                    //  ファイルパス名へのポインタ
    UBP         pSyl;                     //  語句文字列ポインタ
    UW          tkn;                      //  トークンコード
    UW          kndPP;                    //  プリプロセス文識別(＝0:プリプロセス文以外, ≠0:AJCPPC_PPK_XXXXX)
    UI          lno;                      //  行番号
    UI          pos;                      //  トークンの位置（タブステップ＝４）
    UB          flg;                      //  語句のフラグ情報(AJCTKF_XXXXX)
    UB          inst;                     //  #include のネストレベル
    UB          inc;                      //  #include 種別
    UB          mexp;                     //  マクロ展開ネストレベル（０：非マクロ展開コード, １～：マクロ展開コー
ド)
} AJCPPCTKNNODE, *PAJCPPCTKNNODE;
typedef const AJCPPCTKNNODE *PCAJCPPCTKNNODE;
```

「kndPP」が AJCPPC_PPK_TOKEN(=0)の場合は、プリコンパイルにより生成された通常のトークン(語句)の情報であることを示します。

この場合、pSyl はトークン文字列(ex. “if”, “0xFF”, “&&” 等)をポイントします。

「KndPP」が AJCPPC_PPK_TOKEN(=0)以外である場合は、以下のプリプロセス文情報を示し、「tkn = EAJCTK_PREPRO」となります。

kndPP	内容
AJCPPC_PPK_DEFINE	#define 文を意味します
AJCPPC_PPK_UNDEF	#undef 文を意味します
AJCPPC_PPK_INCLUDE	#include 文を意味します
AJCPPC_PPK_IFDEF	#ifdef 文を意味します
AJCPPC_PPK_IFNDEF	#ifndef 文を意味します
AJCPPC_PPK_IF	#if 文を意味します
AJCPPC_PPK_ELIF	#elif 文を意味します
AJCPPC_PPK_ELSE	#else 文を意味します
AJCPPC_PPK_ENDIF	#endif 文を意味します
AJCPPC_PPK_OTHERS	上記以外のプリプロセス文を示します。(#pragma 等)

「KndPP」が AJCPPC_PPK_DEFINE である場合、pSyl は#define で定義されたマクロ名（文字列）をポイントします。
この場合、マクロ定義自体のトークンストリームは、AjcPpcGetMacroInfo ()により取得します。
「KndPP」が AJCPPC_PPK_UNDEF～AJCPPC_PPK_OTHERS である場合、pSyl はプリプロセス文全体の文字列をポイントします
（ex. “#include <stdio.h>” や “#pragma pack(1)” 等)

39.4. サポートＡＰＩ

Ｃ言語のプリコンパイルの、サポートＡＰＩを以下に示します。

#	関数名	内容
1	AjcPpcCreate	インスタンス生成
2	AjcPpcDelete	インスタンス消去
3	AjcPpcSetOption AjcPpcGetOption	プリコンパイルオプションの設定／取得
4	AjcPpcCompile	プリコンパイルの実行
5	AjcPpcGetObject	プリコンパイル済のオブジェクト（トークンストリーム）を取得
6	AjcPpcReleaseObject	プリコンパイル済のオブジェクト（トークンストリーム）を破棄
7	AjcPpcEnumMacro	マクロ定義情報の列挙（全マクロの定義情報取得）
8	AjcPpcGetMacroInfo	マクロ定義情報の取得
9	AjcPpcStop	プリコンパイル中止
10	AjcPpcTokenStreamToFile	プリコンパイル済のオブジェクト（トークンストリーム）をテキストファイルへ出力します
11	AjcPpcSetTecAtTokenStreamToFile	トークンストリームをファイル出力する際のテキスト文字コード設定
12	AjcPpcSetTextEncode	ソースファイルのテキスト文字コード設定
13	AjcPpcGetTextEncode	ソースファイルのテキスト文字コード取得
14	AjcPpcGetErrMsgText	エラーメッセージテキスト取得

※ 関数は全てバイト文字列用です（ワイド文字列には対応していません）

39.4.1. インスタンス生成（AjcPpcCreate）

形　式 ： HAJCPPC　AjcPpcCreate（C_UBP　pBasePath, PAJCLBXITEMA　pIncPath, PAJCLBXITEMA　pOptSym,
UX cbp, VO (CALLBACK *cbNtc)(AJCPPCNOTIFY ntc, UX p1, UX p2, UX p3, UX cbp),
VO (CALLBACK *cbErr)(AJCPPCERROR　err, UX p1, UX p2, UX p3, UX cbp));

引　数 ： pBasePath　　- ベースパス（NULL を指定した場合はカレントディレクトリ）
pIncPath　　- インクルードパス群　（不要時は NULL）
pOptSym　　- オプションシンボル群（不要時は NULL）
cbp　　- コールバックパラメタ
cbNtc　　- イベント通知用コールバック関数（不要時は NULL）
cbErr　　- エラー通知用コールバック関数　（不要時は NULL）

説　明 ： インスタンスを生成し、初期化します。
pBasePath は、インクルードファイルを検索する際の最上位フォルダを指定します。
pBasePath = NULL の場合は、カレントディレクトリを仮定します。
pBasePath に相対パスを指定した場合は、カレントディレクトリからの相対パスとみなします。
pIncPath は、インクルードファイルのパス群の配列の先頭アドレスを指定します。
AjcPpcSetOption()で AJCPPC_FLG_AUTO_SEARCH（インクルードファイル自動検索フラグ）を指定する場合、
各インクルードパスにワイルドカードを指定することができます。(ex. 「c:¥Program files¥Microsoft*.*」)
pIncPath に相対パスを指定した場合は、ベースパスからの相対パスとみなします。
pOptSym は、オプションシンボル群の配列の先頭アドレスを指定します。
pIncPath, pOptSym は以下の構造体の配列で指定します。

```
typedef struct {
        UBP     pStr;     // インクルードパス ／ オプションシンボル文字列へのポインタ
        UX      data;     // 0 ／ AJCPP_INC_TOPPRIORITY
} AJCLBXITEM, *PAJCLBXITEM;
typedef const AJCLBXITEM *PCAJCLBXITEM;
```

「pIncPath->data == AJCPP_INC_TOPPRIORITY」を設定してあるインクルードパスは、最優先パスとして扱います。
最優先パスは、自フォルダパスよりも優先します。

戻り値 ： ≠NULL - 成功　（インスタンスハンドルを返します）
＝NULL - 失敗

コールバック

cbNtc（イベント通知）

形　式　：　VO CALLBACK ***cbNtc***(AJCPPCNOTIFY ntc, UX p1, UX p2, UX p3, UX cbp);

引　数　：　ntc　　- 通知コード
p1, p2, p3 - 通知パラメタ
cbp　　- コールバックパラメタ

説　明　：　プリコンパイル中に以下のイベントを通知します。

#	ntc(AJCPPCNOTIFY)	内容	p1	p2	p3
1	AJCPPC_NTC_ANYEVT	いずれかのイベントが発生したことを通知（※1） (このイベントの後に、#2以降のイベントが発生します)	イベント識別 (AJCPPCNOTIFY)	－	－
2	AJCPPC_NTC_FILE_LNO	ファイル名，行番号通知　　（※1）	″ファイル名″	行#	ネスト値（0～)
3	AJCPPC_NTC_SRH_START	インクルードファイル検索開始通知	″Incファイル名″	″検索フォルダ″	－
4	AJCPPC_NTC_SRH_DIR	インクルードファイル検索中のフォルダ通知	″フォルダパス″	－	－
5	AJCPPC_NTC_SRH_END	インクルードファイル検索終了通知	″Incファイル名″	0:見つからない 1:見つかった	－
6	AJCPPC_NTC_OPTSYM	プリプロセス用オプションシンボル通知 (#if, #elif, #ifdef, #ifndefでの参照シンボル)	PCAJCPPCTKNNODE (シンボル・トークン)	－	－
7	AJCPPC_NTC_MACDEF	マクロ定義通知　　（※2）	PCAJCPPCTKNNODE (マクロ名トークン)	PCAJCPPCMACINFO (マクロ定義)	－
8	AJCPPC_NTC_MACREF	マクロ参照通知　　（※2）	PCAJCPPCTKNNODE (展開先頭トークン)	PCAJCPPCMACINFO (マクロ定義)	－
9	AJCPPC_NTC_OUTLOOP	トークンストリームをファイルへ出力中通知	0:出力中 1:出力終了	－	－
10	AJCPPC_NTC_SRCTEC	ソースファイルのテキストエンコード通知（※1）	“ソースファイルパス”	テキストエンコード EAJCTEC	TRUE : BOM有 FALSE: BOM無
11	AJCPPC_NTC_TOKEN	トークン通知（ソースから読み出したトークン）	トークンコード (EAJCTKCODE)	“トークン文字列”	－

※1：これらの通知はプリコンパイル中やファイル出力中に頻繁に通知されます。
プリコンパイル(AjcPpcCompile())やファイル出力(AjcPpcTokenStreamToFile())は、処理が終了するまでロックされます。
これらの通知を契機に、キャンセルボタンのチェックやキー入力をチェックし、必要ならば AjcPpcStop()を呼び出して、プリコンパイルやファイル出力処理を中止することができます。

※2：マクロ名，マクロ定義／参照箇所は以下のコードで参照できます

項目		コード	備考
マクロ名		((PAJCPPCMACINFO)**p2**)->pMacName	
定義／参照箇所	ファイルパス	((PAJCPPCTKNNODE)**p1**)->pFile	標準事前定義マクロ（__DATE__等）の場合は　pFile = ”<ANSI-C>” （固定） オプションシンボルで与えたマクロの場合は pFile=“<OptionSymbol>” （固定）
	行番号	((PAJCPPCTKNNODE)**p1**)->lno	標準事前定義マクロ／オプションシンボルで与えたマクロの場合は lno=1（固定）

cbErr（エラー通知）

形　式 ： VO CALLBACK ***cbErr***(AJCPPCERROR　err, UX p1, UX p2, UX p3, UX cbp);

引　数 ： ntc　　- 通知コード
p1, p2, p3 - 通知パラメタ
cbp　　- コールバックパラメタ

説　明 ： 以下のエラーを通知します。

#	名称(err)	内容	p1	p2	p3
1	AJCPPC_ERROR_SRC_OPEN	ソースファイルをオープンできません	"ファイルパス"	－	－
2	AJCPPC_ERROR_INC_OPEN	INCLUDE ファイルをオープンできません	"ファイルパス"	行#	"Include パス"
3	AJCPPC_ERROR_NO_SYMBOL	#ifdef/#ifndef でシンボル名が指定されていません	"ファイルパス"	1	－
4	AJCPPC_ERROR_COND_NOTCLS	条件(#if～#endif)がクローズされていません	"ファイルパス"	行#	－
5	AJCPPC_ERROR_COND_DEEP	条件(#if～#endif)のネストが深すぎます	"ファイルパス"	行#	－
6	AJCPPC_ERROR_NOT_IN_IF	対応する「#if / #ifdef / #ifndef」がありません	"ファイルパス"	行#	－
7	AJCPPC_ERROR_ELIF_IN_ELSE	「#else」条件中に「#elif」は記述できません	"ファイルパス"	行#	－
8	AJCPPC_ERROR_ELSE_IN_ELSE	「#else」条件中に「#else」は記述できません	"ファイルパス"	行#	－
9	AJCPPC_DEFERR_INVALID	「defined」の構文誤り	"ファイルパス"	行#	－
10	AJCPPC_DEFERR_NEED_LP	defined'の後に左括弧 '(' が必要です	"ファイルパス"	行#	－
11	AJCPPC_DEFERR_NEED_SYMBOL	defined'でシンボルが指定されていません	"ファイルパス"	行#	－
12	AJCPPC_DEFERR_NEED_RP	シンボルの次に右括弧 ')' が必要です	"ファイルパス"	行#	"シンボル"
13	AJCPPC_MACERR_NEED_LP	マクロ名の後に左括弧が必要です	"ファイルパス"	行#	"マクロ名"
14	AJCPPC_MACERR_NEED_RP	仮引数の後に右括弧が必要です	"ファイルパス"	行#	"仮引数名"
15	AJCPPC_MACERR_NEED_RP_C	仮引数の後に右括弧かカンマが必要です	"ファイルパス"	行#	"仮引数名"
16	AJCPPC_MACERR_MULTDEF	マクロ二重定義	"ファイルパス"	行#	"マクロ名"
17	AJCPPC_MACERR_INV_NAME	不正なマクロ名です	"ファイルパス"	行#	－
18	AJCPPC_MACERR_NO_NAME	マクロ名がありません	"ファイルパス"	行#	－
19	AJCPPC_MACERR_NEST_OVER	マクロ展開ネストオーバー	－	－	－
20	AJCPPC_MACERR_ARG_LACK	マクロの引数が少なすぎる	"ファイルパス"	行#	"マクロ名"
21	AJCPPC_MACERR_ARG_OVER	マクロの引数が多すぎる	"ファイルパス"	行#	"マクロ名"
22	AJCPPC_MACERR_ARG_INVALID	不正なマクロ引数があります	"ファイルパス"	行#	"マクロ名"
23	AJCPPC_INCERR_NO_FILE	Include ファイル名が指定されていません	"ファイルパス"	行#	－
24	AJCPPC_INCERR_INV_FILE	Include ファイルの記述が不正です	"ファイルパス"	行#	－
25	AJCPPC_INCERR_NOTCLS	Include ファイル名の後に '>' がありません	"ファイルパス"	行#	－
26	AJCPPC_INCERR_NEST_OVER	Include ファイルのネストオーバー	"ファイルパス"	行#	－
27	AJCPPC_INCERR_NOT_FOUND	Include ファイルが見つかりません	"ファイルパス"	行#	"Include ファイル"
28	AJCPPC_SCLERR_INVSYL	無効な語句	"ファイルパス"	行#	"語句"
29	AJCPPC_SCLERR_DIVZERO	ゼロ除算	"ファイルパス"	行#	－
30	AJCPPC_SCLERR_NOTCLS	括弧が閉じられていない	"ファイルパス"	行#	－
31	AJCPPC_SCLERR_EXPRESSION	不当な数式表現	"ファイルパス"	行#	－
32	AJCPPC_SCLERR_OVERNEST	数式表現のネストオーバー	"ファイルパス"	行#	－
33	AJCPPC_ERROR_MEMALLOC	メモリ割り当て失敗	－	－	－

戻り値 ： なし

39.4.2. インスタンス消去（AjcPpcDelete）

形　式 ： VO　AjcPpcDelete (HAJCPPC hPpc);

引　数 ： hPpc - インスタンスハンドル

説　明 ： AjcPpcCreate で動的に確保した、インスタンスワーク領域を解放します。

戻り値 ： なし

39.4.3. プリコンパイル・オプションの設定／取得（AjcPpc{Set|Get}Option）

形　式 ： BOOL　AjcPpcSetOption(HAJCPPC hPpc, UI opt);
UI　　AjcPpcGetOption(HAJCPPC hPpc);

引　数 ： hPpc - インスタンスハンドル
opt　- プリコンパイルオプション

説　明 ： プリコンパイル・オプションを設定／取得します。
プリコンパイルオプションは、以下のシンボルの合成値で指定します。(何も指定しない場合は、０)

シンボル	内容	備考
AJCPPC_FLG_NONE	指定無し	（＝０）
AJCPPC_FLG_AUTO_SEARCH	インクルードファイル自動検索フラグ インクルードファイル群と、ベースパスの下（全サブフォルダを含む）からインクルードファイルを検索します。	
AJCPPC_FLG_ONCE	同一インクルードファイルを１回だけ読み出す	
AJCPPC_FLG_GENALL	非生成部分（条件コンパイル＝偽の部分）もファイル出力する	

戻り値 ： なし

39.4.4. プリコンパイルの実行（AjcPpcCompile）

形　式 ： AJCPPCRESULT　AjcPpcCompile (HAJCPPC hPpc, C_UBP pSrcPath);

引　数 ： hPpc　　- インスタンスハンドル
pSrcPath - ソースプログラムパス（相対パスを指定した場合は、ベースパスからの相対パスとなります）

説　明 ： ソースプログラムをプリコンパイルしたトークン・ストリームを作成します。
このＡＰＩはプリコンパイルが終了するまで、呼び出し元に戻りません。
プリコンパイルを中止するには、プリコンパイル中に頻繁に通知される、ファイル名,行番号通知
(AJCPPC_NTC_FILE_LNO) の通知コールバック中で AjcPpcStop()を呼び出します。

戻り値 ： AJCPPCR_OK　　- 正常
AJCPPCR_NOFILE　- ファイルなし
AJCPPCR_MEMERR　- メモリエラー（メモリ不足）
AJCPPCR_STOP　　- 中止
AJCPPCR_NOTOKEN　-トークン無し
AJCPPCR_PARAM　　-パラメタエラー

備　考 ： プリコンパイルしたトークン・ストリームを取得するには、AjcPpcGetObject()を実行します。

39.4.5. プリコンパイル済オブジェクトの取得（AjcPpcGetObject）

形　式 ： PAJCPPCTKNNODE AjcPpcGetObject (HAJCPPC hPpc, PAJCPPCTKNNODE *ppNoGen);

引　数 ： hPpc　　- インスタンスハンドル
ppNoGen - 非生成部分のトークンストリームの先頭アドレスを格納するバッファアドレス（不要時は NULL）

説　明 ： プリコンパイルしたオブジェクトの(生成部分の)トークンストリームの先頭アドレスを返します。
このトークンストリームには、条件コンパイル(#if, #elif, #ifdef, #ifndef, #else, #endif)で「偽条件」で生成されなかった部分は含まれません。
*ppNoGen には、条件コンパイル(#if, #elif, #ifdef, #ifndef, #else, #endif)で「偽条件」で生成されなかった部分のトークンストリームの先頭アドレスを格納します。
非生成部分が無い場合は NULL が設定されます。

戻り値 ： ≠NULL - プリコンパイルしたトークン・ストリームの先頭アドレス
＝NULL - 指定されたソースプログラムはプリコンパイルされていない

39.4.6. プリコンパイル済オブジェクトの解放（AjcPpcReleaseObject）

形　式 ： BOOL AjcPpcDelete (HAJCPPC pW);

引　数 ： hPpc　　- インスタンスハンドル

説　明 ： プリコンパイルしたトークンストリームを破棄します。

戻り値 ： TRUE - 成功
FALSE - 失敗

39.4.7. マクロ定義情報の列挙（AjcPpcEnumMacro）

形　式 ： UI AjcPpcDEnumMacro (HAJCPPC hPpc, UX cbp, BOOL (CALLBACK *cbNtcMacInfo)(PCAJCPPCMACINFO pMacInfo, UX cbp));

引　数 ： hPpc　　- インスタンスハンドル
cbp　　- コールバックパラメタ
cbNtcMacInfo - マクロ定義情報通知用コールバック関数（不要時は NULL）

説　明 ： 全マクロの定義情報を取得します。
マクロ名の個数だけを取得する場合は、cbNtcMacName=NULL を指定します。

戻り値 ： マクロ名の個数

コールバック

cbNtcMacInfo（マクロ定義情報通知）

形　式 ： BOOL CALLBACK ***cbNtcMacInfo***(PCAJCPPCMACINFO pMacInfo, UX cbp);

引　数 ： pMacInfo　- マクロ定義情報へのポインタ（構造体の形式は「AjcPpcGetMacroInfo()」参照）
cbp　　- コールバックパラメタ

説　明 ： プリコンパイル中に検出した、全マクロの定義情報を通知します。
マクロ名は「pMacInfo->pMacName」に設定されています。

戻り値 ： TRUE - 継続
FALSE – 中止

39.4.8. マクロ定義情報の取得（AjcPpcGetMacroInfo）

形　式 ： BOOL AjcPpcGetMacroInfo (HAJCPPC hPpc, C_BCP pMacName, PAJCPPCMACINFO pMacInfo);

引　数 ： hPpc　　- インスタンスハンドル
pMacName - マクロ名文字列へのポインタ
pMacInfo - マクロ定義情報を格納するバッファ（不要時は NULL）

説　明 ： 指定したマクロ名のマクロ定義情報を取得します。
マクロ定義情報の構造体は以下のとおりです。

```
typedef struct {
    BOOL            fWithArg;           //  引数付マクロ・フラグ（TRUE：引数付, FALSE:引数無し）
    BCP             pMacName;           //  マクロ名文字列へのポインタ
    PAJCPPCTKNNODE  pTknName;           //  マクロ名・トークンノードへのポインタ
    PAJCPPCTKNNODE  pTknBody;           //  マクロボディの先頭トークンノードへのポインタ
    UI              pos;                //  先頭「#」の桁位置
    BOOL            fVaArgs;            //  可変個数引数マクロフラグ（fWithArg=FALSE 時は０）
    UI              nTmpArg;            //  仮引数の個数          （fWithArg=FALSE 時は０）
    HAJCVQUE        hVQueArg;           //  マクロ仮引数名リスト（fWithArg=FALSE 時は NULL）※１
} AJCPPCMACINFO, *PAJCPPCMACINFO;
typedef const AJCPPCMACINFO *PCAJCPPCMACINFO;
```

※１：可変長線形リストのハンドルです。ノードデータは仮引数名の文字列（終端 0x00 付き）です。

戻り値 ： TRUE ： 成功
FALSE ： 指定されたマクロが見つからない

39.4.9. プリコンパイルの中止（AjcPpcStop）

形　式 ： VO AjcPpcStop (HAJCPPC hPpc);

引　数 ： hPpc - インスタンスハンドル

説　明 ： プリコンパイルを中止します。

戻り値 ： なし

39.4.10. トークンストリームのファイル出力（AjcPpcTokenStreamToFile）

形　式 ： BOOL　AjcPpcTokenStreamToFile (HAJCPPC hPpc, C_UBP pOut, BOOL fExpInc, AJCPPCPPKND CommOutOfkndPP);

引　数 ： hPpc　　- インスタンスハンドル
pOut　　- 出力ファイルパス（相対パスを指定した場合は、ベースパスからの相対パスとなります）
fExpInc – TRUE　: インクルードファイルの内容を展開したテキストを作成
　　FALSE : インクルードファイルの内容は展開しない（#include 文をそのまま出力する）
CommOutOfkndPP - コメント出力するプリプロセス文を指定します。

説　明 ： トークンストリームをテキストファイルとして出力します。（プリコンパイルソースの出力）
fExpInc=TRUE を指定した場合は、インクルードした内容も全てファイル出力します。
fExpInc=FALSE を指定した場合は、インクルードした内容は出力しないで「#include」文をそのまま出力します。
CommOutOfkndPP は、コメント出力するプリプロセス文を以下のシンボルで指定します（複数指定時は論理和）

- AJCPPC_PPK_COND　- #if～#endif をコメント出力　（未指定時はコメント化せずにそのまま出力）
- AJCPPC_PPK_DEFINE　- #define をコメント出力　（未指定時はコメント化せずにそのまま出力）
- AJCPPC_PPK_UNDEF　- #undef をコメント出力　（未指定時はコメント化せずにそのまま出力）
- AJCPPC_PPK_ALL　- 上記全てのシンボルの合成値

fExpInc=TRUE （include の内容を展開する　）場合は、#include 文自体はコメント出力します。
fExpInc=FALSE（include の内容を展開しない）場合は、#include 文自体をコメント化せずにそのまま出力します。
上記以外のプリプロセス文（#pragma 等）は、コメント化せずに、そのままファイル出力します。
#define 以外のプリプロセス文は、改行せずに 1 行で出力します。

このＡＰＩはファイル出力が終了するまで、呼び出し元に戻りません。
ファイル出力を中止するには、ファイル出力中に中に頻繁に通知される、トークンストリームをファイルへ出力中通知（AJCPPC_NTC_OUTLOOP）の通知コールバック中で AjcPpcStop()を呼び出します。

戻り値 ： TRUE　- 成功
FALSE – 失敗

39.4.11. トークンストリームをファイル出力する際のテキスト文字コード設定（AjcPpcSetTecAtTokenStreamToFile）

形　式 ： BOOL　AjcPpcSetTecAtTokenStreamToFile (HAJCPPC hPpc, EAJCTEC tec,　BOOL fBOM);

引　数 ： hPpc　　- インスタンスハンドル
tec　　- テキスト文字コード
fBOM　　- ＢＯＭ出力フラグ（TRUE:出力する,　FALSE:出力しない（デフォルト））

説　明 ： トークンストリームをファイル出力する際のテキスト文字コードとＢＯＭ出力に有無を設定します。
tec は、以下のいずれかの値を指定します。

- AJCTEC_MBC　　- マルチバイト(S-JIS)・・・デフォルト値
- AJCTEC_UTF_8　　- ＵＴＦ－８
- AJCTEC_EUC_J　　- 日本語ＥＵＣ
- AJCTEC_UTF_16LE　- ＵＴＦ－１６（リトルエンディアン）
- AJCTEC_UTF_16BE　- ＵＴＦ－１６（ビッグエンディアン）

戻り値 ： TRUE　- 成功
FALSE - 失敗

39.4.12. ソースファイルのテキスト文字コード設定(AjcPpcSetTextEncode)

形　式 ： BOOL　AjcPpcSetTextEncode(HAJCPPC hPpc, EAJCTEC tec);

引　数 ： hPpc　　- インスタンスハンドル
tec　　- 入力テキスト文字コード

説　明 ： ソースファイルを入力する際のテキスト文字コードを設定します。
tec は、以下のいずれかの値を指定します。
・AJCTEC_MBC　　- マルチバイト(S-JIS)・・・デフォルト値
・AJCTEC_UTF_8　- ＵＴＦ－８
・AJCTEC_EUC_J　- 日本語ＥＵＣ
・AJCTEC_UTF_16LE　- ＵＴＦ－１６（リトルエンディアン）
・AJCTEC_UTF_16BE　- ＵＴＦ－１６（ビッグエンディアン）
・AJCTEC_AUTO　　- 自動判別

戻り値 ： TRUE　- 成功
FALSE - 失敗

39.4.13. ソースファイルのテキスト文字コード取得(AjcPpcGetTextEncode)

形　式 ： BOOL　AjcPpcGetTextEncode(HAJCPPC hPpc, PEAJCTEC piTec,　BOOL *pfBOM);

引　数 ： hPpc　　- インスタンスハンドル
pTec　　- 入力ソースファイルの文字コードを格納するバッファのアドレス（不要時は NULL）
pfBOM　- 入力ソースファイルの BOM 有無を格納するバッファのアドレス　（不要時は NULL）

説　明 ： 直前にプリコンパイルしたソースファイルのテキスト文字コードとＢＯＭの有無を取得します。
pTec で示すバッファには以下のいずれかの値が設定されます。
・AJCTEC_MBC　　- マルチバイト(S-JIS)・・・デフォルト値
・AJCTEC_UTF_8　- ＵＴＦ－８
・AJCTEC_EUC_J　- 日本語ＥＵＣ
・AJCTEC_UTF_16LE　- ＵＴＦ－１６（リトルエンディアン）
・AJCTEC_UTF_16BE　- ＵＴＦ－１６（ビッグエンディアン）

pfBOM で示すバッファには、ソースファイルにＢＯＭが存在する(TRUE)か、存在しない(FALSE)かの情報が設定されます。

プリコンパイルが１つも実行されていない場合は、いずれの情報も－１を設定します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

39.4.14. エラーメッセージテキスト取得(AjcPpcGetErrMsgText)

形　式 ： C_BCP　AjcPpcGetErrMsgText (AJCPPCERROR err, UX p1, UX p2, UX p3);

引　数 ： err　　- エラーコード
p1, p2, p3 - エラー情報

説　明 ： エラー通知されたエラーに関するメッセージテキストを取得します。
このＡＰＩは、エラー通知用コールバック関数(*cbErr*)中でのみ実行可能です。
err, p1, p2, p3 はエラー通知時の引数を指定します。

戻り値 ： エラーに関するメッセージテキストへのポインタ

備　考 ： 取得するエラーメッセージテキストは以下の通りです

エラーコード	エラーメッセージテキスト	備考
AJCPPC_ERROR_SRC_OPEN	ソースファイル(*FilePath*)をオープンできません	
AJCPPC_ERROR_INC_OPEN	“*FilePath*”<*nnn*>: インクルードファイルをオープンできません (*IncludeFilePath*)	
AJCPPC_ERROR_NO_SYMBOL	“*FilePath*”<*nnn*>: #ifdef/#ifndef で、シンボル名が指定されていません	
AJCPPC_ERROR_COND_NOTCLS	“*FilePath*”<1>: 条件(#if～#endif)がクローズされていません	
AJCPPC_ERROR_COND_DEEP	“*FilePath*”<*nnn*>: 条件(#if～#endif)のネストが深すぎます	
AJCPPC_ERROR_NOT_IN_IF	“*FilePath*”<*nnn*>: 対応する「#if / #ifdef / #ifndef」がありません	
AJCPPC_ERROR_ELIF_IN_ELSE	“*FilePath*”<*nnn*>: 「#else」条件中に「#elif」は記述できません	
AJCPPC_ERROR_ELSE_IN_ELSE	“*FilePath*”<*nnn*>: 「#else」条件中に「#else」は記述できません	
AJCPPC_DEFERR_INVALID	“*FilePath*”<*nnn*>: ‘defined’の構文誤り	
AJCPPC_DEFERR_NEED_LP	“*FilePath*”<*nnn*>: ’defined’の後に左括弧 ’(’ が必要です	
AJCPPC_DEFERR_NEED_SYMBOL	“*FilePath*”<*nnn*>: ’defined’でシンボルが指定されていません	
AJCPPC_DEFERR_NEED_RP	“*FilePath*”<*nnn*>: シンボル(*XXX*)の後に右括弧 ’)’ が必要です	
AJCPPC_MACERR_NEED_LP	“*FilePath*”<*nnn*>: マクロ名(*XXX*)の後に左括弧が必要です	
AJCPPC_MACERR_NEED_RP	“*FilePath*”<*nnn*>: 仮引数(*XXX*)の後に右括弧が必要です	
AJCPPC_MACERR_NEED_RP_C	“*FilePath*”<*nnn*>: 仮引数(*XXX*)の後に右括弧かカンマが必要です	
AJCPPC_MACERR_MULTDEF	“*FilePath*”<*nnn*>: マクロ(*XXX*)二重定義	
AJCPPC_MACERR_INV_NAME	“*FilePath*”<*nnn*>: 不正なマクロ名です	
AJCPPC_MACERR_NO_NAME	“*FilePath*”<*nnn*>: マクロ名がありません	
AJCPPC_MACERR_NEST_OVER	“*FilePath*”<*nnn*>: マクロ展開ネストオーバー(*MacroName*)	最大ネスト＝３２
AJCPPC_MACERR_ARG_LACK	“*FilePath*”<*nnn*>: マクロ(*XXX*)の引数が少なすぎます	
AJCPPC_MACERR_ARG_OVER	“*FilePath*”<*nnn*>: マクロ(*XXX*)の引数が多すぎます	
AJCPPC_INCERR_NO_FILE	“*FilePath*”<*nnn*>: Include ファイル名が指定されていません	
AJCPPC_INCERR_INV_FILE	“*FilePath*”<*nnn*>: Include ファイルの記述が不正です	
AJCPPC_INCERR_NOTCLS	“*FilePath*”<*nnn*>: Include ファイル名の後に ’>’ がありません	
AJCPPC_INCERR_NEST_OVER	“*FilePath*”<*nnn*>: Include ファイルのネストオーバー	最大ネスト＝３２
AJCPPC_INCERR_NOT_FOUND	“*FilePath*”<*nnn*>: Include ファイルが見つかりません(*FileName*)	
AJCPPC_SCLERR_INVSYL	“*FilePath*”<*nnn*>: 式中に不当な語句があります(*XXX*)	
AJCPPC_SCLERR_DIVZERO	“*FilePath*”<*nnn*>: ゼロ除算エラー	
AJCPPC_SCLERR_NOTCLS	“*FilePath*”<*nnn*>: 右括弧で閉じられていません	
AJCPPC_SCLERR_EXPRESSION	“*FilePath*”<*nnn*>: 不当な数式表現です「*XXX*」	
AJCPPC_SCLERR_EOL	“*FilePath*”<*nnn*>: 数式表現が途中で終了しています	
AJCPPC_SCLERR_OVERNEST	“*FilePath*”<*nnn*>: 数式表現のネストが深すぎます	‘(‘ によるネスト，最大３２
AJCPPC_ERROR_MEMALLOC	“*FilePath*”<*nnn*>:メモリ割り当て失敗	
上記外	未定義エラーを検出しました	

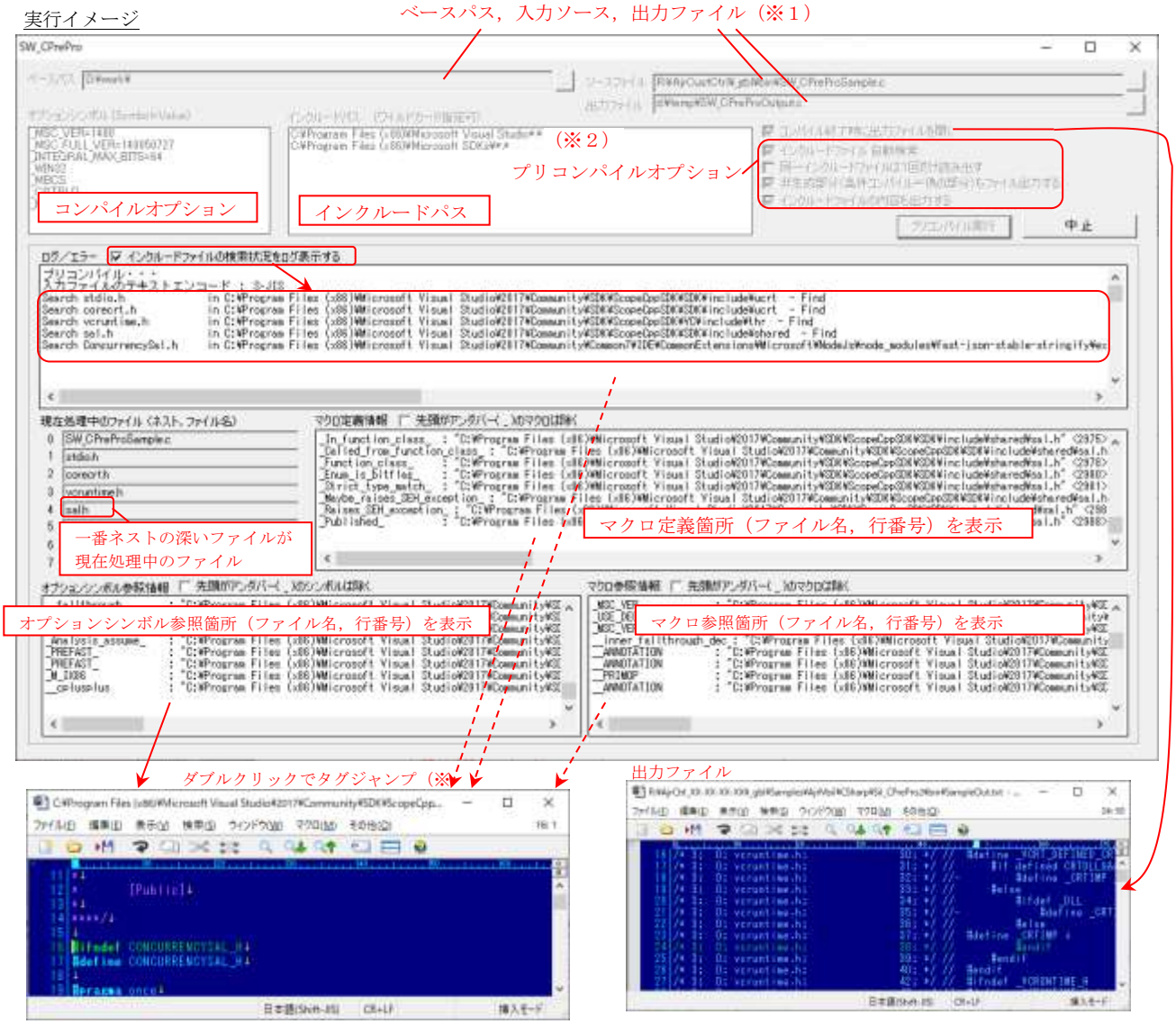
“*FilePath*”： エラーを検出したファイルのパス名
<*nnn*>　　： エラーを検出した行番号

39.5. サンプルプログラム

39.5.1. SW_CPrePro（C言語のプリコンパイル）

以下のサンプルプログラムは、ＧＵＩにより各種設定を行い、プリコンパイルを実行します。

実行イメージ



※１：ベースパス，入力ソース，出力ファイルは、以下の方法で設定／編集します。
- 「...」ボタンにより、ダイアログボックスから選択
- エクスプローラから、ドラッグ＆ドロップ

※２：オプションシンボル，インクルードパスは、以下の方法で設定・編集します。
- エクスプローラから、ドラッグ＆ドロップ
- 右クリックで項目の削除等の操作が可能

※３：タグジャンプは、特定のテキストエディタを起動するように、ハードコードしています。
起動するテキストエディタを変更する場合は、ソースプログラム中 TagJump()関数の以下の**赤字**部分を変更してください。

```
AjcSnPrintF(szOpt, AJCTSIZE(szOpt), "/J%d /m4 %s", lno, pPath);
ShellExecute(NULL, NULL, "Hidemaru.exe", szOpt, "C:¥¥Program Files (x86)¥¥Hidemaru¥¥", SW_SHOWNORMAL);  //  タグジャンプ
```

```
 1 : //
 2 : //  SW_CPrePro.c
 3 : //
 4 :
 5 : #include    <AjrCstXX.h>
 6 : #include    <math.h>
 7 : #include    <tchar.h>
 8 : #include    "resource.h"
 9 :
10 : //-------------------------------------------------------------------------------------------------------------//
11 : //  ワーク                                                                                                     //
12 : //-------------------------------------------------------------------------------------------------------------//
13 : static  HINSTANCE       hInst;                      //  ＤＬＬインスタンスハンドル
14 : static  HWND            hDlgMain;                   //  ダイアログボックスハンドル
15 : static  HWND            hVthLog;
16 : static  HWND            hVthMacDef;
17 : static  HWND            hVthMacRef;
18 : static  HWND            hVthOptSym;
19 :
20 : static  HAJCPPC         hPpc  = NULL;               //  プリコンパイル インスタンスハンドル
21 : static  BOOL            fBusy = FALSE;              //  プリコンパイル中フラグ
22 : static  BOOL            fExit = FALSE;              //  プログラム終了フラグ
23 : static  UI              CurNest = -1;               //  インクルードネスト値
24 : static  BC              SvFileName[MAX_PATH] = {0};
25 :
26 : //-------------------------------------------------------------------------------------------------------------//
27 : //  内部サブ関数                                                                                               //
28 : //-------------------------------------------------------------------------------------------------------------//
29 : AJC_DLGPROC_DEF(Main);
30 : static  VO CALLBACK cbNtcPpcEvent(AJCPPCNOTIFY ntc, UX p1, UX p2, UX p3, UX cbp);
31 : static  VO CALLBACK cbNtcPpcErr  (AJCPPCERROR err, UX p1, UX p2, UX p3, UX cbp);
32 : static  VO      TagJump(BCP pLine);
33 :
34 : //=============================================================================================================//
35 : //                                                                                                             //
36 : //  W i n M a i n                                                                                              //
37 : //                                                                                                             //
38 : //=============================================================================================================//
39 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
40 : {
41 :     MSG     msg;
42 :
43 :     hInst = hInstance;
44 :
45 :     //----- メイン・ダイアログオープン -----------------//
46 :     hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_DLGMAIN), NULL, AJC_DLGPROC_NAME(Main));
47 :     ShowWindow(hDlgMain, SW_SHOW);
48 :
49 :     //----- メッセージループ ---------------------------//
50 :     while (GetMessage(&msg, NULL, 0, 0)) {
51 :         do {
52 :             if (IsDialogMessage(hDlgMain, &msg)) break;
53 :             TranslateMessage(&msg);
54 :             DispatchMessage (&msg);
55 :         } while (0);
56 :     }
57 :
58 :     return (int)msg.wParam ;
59 : }
60 : //=============================================================================================================//
61 : //                                                                                                             //
62 : //  ダイアログ・プロシージャ                                                                                   //
63 : //                                                                                                             //
64 : //=============================================================================================================//
65 : //----- ダイアログ初期化 ----------------------------------------------------------------------------------------//
66 : AJC_DLGPROC(Main, WM_INITDIALOG     )
67 : {
68 :     BC      path[MAX_PATH];
69 :
70 :     hDlgMain   = hDlg;
71 :     hVthLog    = GetDlgItem(hDlg, IDC_VTH_LOG);
72 :     hVthMacDef = GetDlgItem(hDlg, IDC_VTH_MACDEF);
73 :     hVthMacRef = GetDlgItem(hDlg, IDC_VTH_MACREF);
74 :     hVthOptSym = GetDlgItem(hDlg, IDC_VTH_OPTSYM);
75 :
76 :     //----- テキストボックスにドロップ可能とする -----------------//
77 :     AjcEnableDlgItemToDrop(hDlg, IDC_TXT_BASE, AJCDROP_DIR);
78 :     AjcEnableDlgItemToDrop(hDlg, IDC_TXT_SRC , AJCDROP_DIR_AND_FILE);
79 :     AjcEnableDlgItemToDrop(hDlg, IDC_TXT_OUT , AJCDROP_DIR_AND_FILE);
80 :     //----- チェックボックス初期化 -------------------------------//
```

```
 81 :     AjcSetDlgItemChk(hDlg, IDC_CHK_OPENOUTFILE, TRUE);
 82 :     AjcSetDlgItemChk(hDlg, IDC_CHK_AUTOSRH    , TRUE);
 83 :     AjcSetDlgItemChk(hDlg, IDC_CHK_GENALL     , TRUE);
 84 :     AjcSetDlgItemChk(hDlg, IDC_CHK_EXPINC     , TRUE);
 85 :     AjcSetDlgItemChk(hDlg, IDC_CHK_MACDEF     , TRUE);
 86 :     AjcSetDlgItemChk(hDlg, IDC_CHK_OPTSYM     , TRUE);
 87 :     AjcSetDlgItemChk(hDlg, IDC_CHK_MACREF     , TRUE);
 88 :     //----- サンプル用デフォルト設定 ----------------------------//
 89 :     //  入出力ファイル
 90 :     AjcGetAppPath(path, MAX_PATH); AjcPathCat(path, "SW_CPreProSample.c", MAX_PATH);
 91 :     AjcSetDlgItemStr(hDlg, IDC_TXT_SRC, path);
 92 :     GetTempPath(MAX_PATH, path);   AjcPathCat(path, "SW_CPreProOutput.c", MAX_PATH);
 93 :     AjcSetDlgItemStr(hDlg, IDC_TXT_OUT, path);
 94 :     //  オプションシンボル
 95 :     AjcLbxInsertString(GetDlgItem(hDlg, IDC_LBX_OPTSYM), -1, "_MSC_VER=1400"            );
 96 :     AjcLbxInsertString(GetDlgItem(hDlg, IDC_LBX_OPTSYM), -1, "_MSC_FULL_VER=140050727"  );
 97 :     AjcLbxInsertString(GetDlgItem(hDlg, IDC_LBX_OPTSYM), -1, "_INTEGRAL_MAX_BITS=64"    );
 98 :     AjcLbxInsertString(GetDlgItem(hDlg, IDC_LBX_OPTSYM), -1, "_WIN32"                   );
 99 :     AjcLbxInsertString(GetDlgItem(hDlg, IDC_LBX_OPTSYM), -1, "_MBCS"                    );
100 :     AjcLbxInsertString(GetDlgItem(hDlg, IDC_LBX_OPTSYM), -1, "_CRTBLD"                  );
101 :     AjcLbxInsertString(GetDlgItem(hDlg, IDC_LBX_OPTSYM), -1, "_M_IX86"                  );
102 :     AjcLbxInsertString(GetDlgItem(hDlg, IDC_LBX_OPTSYM), -1, "__STDC__=0"               );
103 :     AjcLbxInsertString(GetDlgItem(hDlg, IDC_LBX_OPTSYM), -1, "__STDC_WANT_SECURE_LIB__=0");
104 :     //  インクルードパス
105 :     AjcLbxInsertString(GetDlgItem(hDlg, IDC_LBX_INCPATH), -1, "C:¥¥Program Files (x86)¥¥Microsoft Visual Studio*.*");
106 :     AjcLbxInsertString(GetDlgItem(hDlg, IDC_LBX_INCPATH), -1, "C:¥¥Program Files (x86)¥¥Microsoft SDKs¥¥*.*"      );
107 :     AjcLbxInsertString(GetDlgItem(hDlg, IDC_LBX_INCPATH), -1, "C:¥¥Program Files (x86)¥¥Windows Kits¥¥*.*"        );
108 :     //----- ツールチップ設定 ------------------------------------//
109 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_TXT_BASE),   "インクルードファイルのベースフォルダ¥n"
110 :                                                     "フォルダをドロップするか、右の「...」ボタンで設定します¥n"
111 :                                                     "「インクルードファイル自動検索」をチェックした場合、"
112 :                                                     "このフォルダ下からもインクルードファイルを検索します" );
113 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_CMD_BASE),   "ダイアログにより、ベースパスを設定します"                 );
114 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_TXT_SRC ),   "入力ソースプログラムファイル¥n"
115 :                                                     "ファイルをドロップするか、右の「...」ボタンで設定します"  );
116 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_CMD_SRC ),   "ダイアログにより、ソースファイルを設定します"             );
117 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_TXT_OUT ),   "プリコンパイル結果の出力ファイル¥n"
118 :                                                     "ファイルをドロップするか、右の「...」ボタンで設定します"  );
119 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_LBX_OPTSYM ),"プリコンパイル用オプションシンボル群, 右クリックで各種操作可");
120 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_LBX_INCPATH),"インクルードパス群, ドロップ可, 右クリックで各種操作可"   );
121 :
122 :     //----- 設定値ロード ----------------------------------------//
123 :     AjcLoadAllControlSettings(hDlg, "Settings", AJCCTL_SELACT_ALL | AJCCTL_SELACT_NTCCHK);
124 :
125 :     return TRUE;
126 : }
127 : //----- ウインド破棄 ----------------------------------------------------------------------------------------//
128 : AJC_DLGPROC(Main, WM_DESTROY        )
129 : {
130 :     //----- 設定値セーブ ----------------------------------------//
131 :     AjcSaveAllControlSettings(hDlg);
132 :     //----- プログラム終了 --------------------------------------//
133 :     PostQuitMessage(0);
134 :     return TRUE;
135 : }
136 : //----- ベースパス設定ボタン --------------------------------------------------------------------------------//
137 : AJC_DLGPROC(Main, IDC_CMD_BASE      )
138 : {
139 :     BC      path[MAX_PATH];
140 :
141 :     AjcGetDlgItemStr(hDlg, IDC_TXT_BASE, path, MAX_PATH);
142 :     if (AjcGetFolderName(hDlg, "ベースパスの設定", path, path, MAX_PATH)) {
143 :         AjcSetDlgItemStr(hDlg, IDC_TXT_BASE, path);
144 :     }
145 :     return TRUE;
146 : }
147 : //----- ソースファイル設定ボタン ----------------------------------------------------------------------------//
148 : AJC_DLGPROC(Main, IDC_CMD_SRC       )
149 : {
150 :     BC      path[MAX_PATH];
151 :
152 :     AjcGetDlgItemStr(hDlg, IDC_TXT_SRC, path, MAX_PATH);
153 :     if (AjcGetOpenFile(hDlg, "ソースファイルの設定", "AllFiles(*.*)/*.*/CLangFiles(*.c)/*.c", "c", path, MAX_PATH)) {
154 :         AjcSetDlgItemStr(hDlg, IDC_TXT_SRC, path);
155 :     }
156 :     return TRUE;
157 : }
158 : //----- 出力ファイル設定ボタン ------------------------------------------------------------------------------//
159 : AJC_DLGPROC(Main, IDC_CMD_OUT       )
160 : {
```

```
161 :     BC      path[MAX_PATH];
162 :
163 :     AjcGetDlgItemStr(hDlg, IDC_TXT_OUT, path, MAX_PATH);
164 :     if (AjcGetSaveFile(hDlg, "出力ファイルの設定", "AllFiles(*.*)/*.*/CLangFiles(*.c)/*.c", "c", path, MAX_PATH)) {
165 :         AjcSetDlgItemStr(hDlg, IDC_TXT_OUT, path);
166 :     }
167 :     return TRUE;
168 : }
169 : //----- ベースパス テキスト ----------------------------------------------------------------------------//
170 : AJC_DLGPROC(Main, IDC_TXT_BASE      )
171 : {
172 :     BC      txt[MAX_PATH];
173 :
174 :     if (HIWORD(wParam) == EN_CHANGE) {
175 :         AjcGetDlgItemStr(hDlg, IDC_TXT_BASE, txt, AJCTSIZE(txt));
176 :         AjcLbxSetBasePath(GetDlgItem(hDlg, IDC_LBX_INCPATH), txt);
177 :     }
178 :     return TRUE;
179 : }
180 : //----- インクルードファイルを自動検索・チェックボックス ----------------------------------------------//
181 : AJC_DLGPROC(Main, IDC_CHK_AUTOSRH   )
182 : {
183 :     return TRUE;
184 : }
185 : //----- 実行ボタン ------------------------------------------------------------------------------------//
186 : AJC_DLGPROC(Main, IDC_CMD_EXEC      )
187 : {
188 :     AJCPPCRESULT    rsu = AJCPPCR_OK;
189 :     PAJCPPCTKNNODE  pTkn     = NULL;
190 :     PAJCLBXITEM     pOptSym  = NULL;
191 :     PAJCLBXITEM     pIncPath = NULL;
192 :     UI              i;
193 :     BC              BasPath[MAX_PATH]   = {0};
194 :     BC              SrcPath[MAX_PATH]   = {0};
195 :     BC              OutPath[MAX_PATH]   = {0};
196 :     BC              fname[MAX_PATH], fext[_MAX_EXT];
197 :
198 :     //  ログクリアー
199 :     AjcVthClear(hVthLog);
200 :     AjcVthClear(hVthOptSym);
201 :     AjcVthClear(hVthMacDef);
202 :     AjcVthClear(hVthMacRef);
203 :     //  プリプロセス実行中の旨、設定
204 :     fBusy = TRUE;
205 :     //  インクルードネスト値クリアー
206 :     CurNest = 0;
207 :     //  全コントロールを禁止状態とする（キャンセルボタンを除く）
208 :     AjcEnableCtrlsInWnd(hDlg, FALSE);
209 :     AjcEnableDlgItem(hDlg, IDC_CMD_CANCEL, TRUE);
210 :     //  ログ全体を許可状態とする
211 :     AjcEnableDlgGroup(hDlg, IDC_GRP_LOG, TRUE, TRUE);
212 :     //  ベース，ソース，出力パス設定
213 :     AjcGetDlgItemStr(hDlg, IDC_TXT_BASE, BasPath, MAX_PATH);
214 :     AjcGetDlgItemStr(hDlg, IDC_TXT_SRC , SrcPath, MAX_PATH);
215 :     AjcGetDlgItemStr(hDlg, IDC_TXT_OUT , OutPath, MAX_PATH);
216 :
217 :     //  オプションシンボル，インクルードパス設定
218 :     pOptSym  = AjcLbxGetAllItems(GetDlgItem(hDlg, IDC_LBX_OPTSYM ), NULL);
219 :     pIncPath = AjcLbxGetAllItems(GetDlgItem(hDlg, IDC_LBX_INCPATH), NULL);
220 :
221 :     //  プリコンパイル実行
222 :     if (hPpc = AjcPpcCreate(BasPath,
223 :                             pIncPath,
224 :                             pOptSym,
225 :                             (UX)hDlg,
226 :                             cbNtcPpcEvent,
227 :                             cbNtcPpcErr)) {
228 :
229 :         //  レベル０のソースファイル表示
230 :         MAjcSplitPath(SrcPath, NULL, NULL, fname, fext);
231 :         strcat_s(fname, MAX_PATH, fext);
232 :         AjcSetDlgItemStr(hDlg, IDC_TXT_F0, fname);
233 :
234 :         //  オプションフラグ設定
235 :         AjcPpcSetOption(hPpc,
236 :             (AjcGetDlgItemChk(hDlg, IDC_CHK_AUTOSRH) ? AJCPPC_FLG_AUTO_SEARCH : 0) |    //  インクルードファイル自動検索
237 :             (AjcGetDlgItemChk(hDlg, IDC_CHK_ONCE   ) ? AJCPPC_FLG_ONCE        : 0) |    //  同一 Include-File を１回だけ読み出す
238 :             (AjcGetDlgItemChk(hDlg, IDC_CHK_GENALL ) ? AJCPPC_FLG_GENALL      : 0));    //  非生成部分(偽の部分)もファイル出力する
239 :
```

```
240 :         //  プリコンパイル実行
241 :         AjcVthTimeStamp(hVthLog);
242 :         AjcVthPutText(hVthLog, " プリコンパイル開始¥n", -1);
243 :         rsu = AjcPpcCompile(hPpc, SrcPath);
244 :         AjcSetDlgItemStr(hDlg, IDC_LBL_LNO, "");
245 :         AjcVthPutText(hVthLog, "¥r¥x1B[2K", -1);
246 :         AjcVthTimeStamp(hVthLog);
247 :         switch (rsu) {
248 :             case AJCPPCR_OK:      AjcVthPutText(hVthLog, " プリコンパイル終了¥n", -1); break;
249 :             case AJCPPCR_NOFILE:  AjcVthPutText(hVthLog, " ソースファイルなし¥n", -1); break;
250 :             case AJCPPCR_MEMERR:  AjcVthPutText(hVthLog, " メモリエラー¥n"      , -1); break;
251 :             case AJCPPCR_STOP:    AjcVthPutText(hVthLog, " 中止しました¥n"      , -1); break;
252 :             case AJCPPCR_NOTOKEN: AjcVthPutText(hVthLog, " 内容がありません¥n" , -1); break;
253 :             case AJCPPCR_PARAM:   AjcVthPutText(hVthLog, " パラメタエラー¥n"    , -1); break;
254 :         }
255 :         if (rsu == AJCPPCR_OK) {
256 :             if (pTkn = AjcPpcGetObject (hPpc, NULL)) {
257 :                 //  プリコンパイル結果をファイルへ出力（インクルードファイルの内容を含める）
258 :                 if (AjcPpcTokenStreamToFile(hPpc,  OutPath, AjcGetDlgItemChk(hDlg, IDC_CHK_EXPINC), AJCPPC_PPK_ALL)) {
259 :                     //  出力ファイルを開く
260 :                     if (AjcGetDlgItemChk(hDlg, IDC_CHK_OPENOUTFILE)) {
261 :                         ShellExecute(NULL, "open", OutPath, NULL, NULL, SW_SHOWNORMAL);
262 :                     }
263 :                 }
264 :                 else {
265 :                     AjcVthPutText(hVthLog, "プリコンパイル結果をファイルへ出力できませんでした¥n", -1);
266 :                 }
267 :             }
268 :         }
269 :         //  レベル０のソースファイルクリアー
270 :         AjcSetDlgItemStr(hDlg, IDC_TXT_F0, "");
271 :
272 :         //  プリコンパイルインスタンス消去
273 :         AjcPpcDelete(hPpc);
274 :         hPpc = NULL;
275 :     }
276 :
277 :     //  リストボックス項目群の配列解放
278 :     if (pOptSym  != NULL) AjcLbxRelAllItems(pOptSym );
279 :     if (pIncPath != NULL) AjcLbxRelAllItems(pIncPath);
280 :
281 :     //  プリコンパイル実行中解除
282 :     fBusy = FALSE;
283 :     //  全コントロール禁止状態を解除
284 :     AjcEnableCtrlsInWnd(hDlg, TRUE);
285 :     AjcEnableDlgItem(hDlg, IDC_CMD_CANCEL, FALSE);
286 :     //  処理中のファイル名クリアー
287 :     for (i = 0; i < 8; i++) {
288 :         AjcSetDlgItemStr(hDlg, IDC_TXT_F0 + i, "");
289 :     }
290 :     //  フォームクローズならば、プログラム終了
291 :     if (fExit) {
292 :         DestroyWindow(hDlg);
293 :     }
294 :
295 :     return TRUE;
296 : }
297 : //----- 中止ボタン ----------------------------------------------------------------------------------------//
298 : AJC_DLGPROC(Main, IDC_CMD_CANCEL    )
299 : {
300 :     if (hPpc != NULL) {
301 :         AjcEnableDlgItem(hDlg, IDC_CMD_CANCEL, FALSE);
302 :         AjcPpcStop(hPpc);
303 :     }
304 :     return TRUE;
305 : }
306 : //----- ログウインドからの通知 ----------------------------------------------------------------------------//
307 : AJC_DLGPROC(Main, IDC_VTH_LOG       )
308 : {
309 :     BC  buf[4096];
310 :
311 :     if (HIWORD(wParam) == AJCVTHN_DBLCLK) {
312 :         AjcVthGetDblClickedLine(hVthLog, buf, AJCTSIZE(buf));
313 :         TagJump(buf);
314 :     }
315 :     return TRUE;
316 : }
317 : //----- オプションシンボル参照ウインドからの通知 ----------------------------------------------------------//
318 : AJC_DLGPROC(Main, IDC_VTH_OPTSYM    )
319 : {
```

```
320 :     BC  buf[4096];
321 :
322 :     if (HIWORD(wParam) == AJCVTHN_DBLCLK) {
323 :         AjcVthGetDblClickedLine(hVthOptSym, buf, AJCTSIZE(buf));
324 :         TagJump(buf);
325 :     }
326 :     return TRUE;
327 : }
328 : //----- マクロ定義ウインドからの通知 ----------------------------------------------------------------------//
329 : AJC_DLGPROC(Main, IDC_VTH_MACDEF    )
330 : {
331 :     BC  buf[4096];
332 :
333 :     if (HIWORD(wParam) == AJCVTHN_DBLCLK) {
334 :         AjcVthGetDblClickedLine(hVthMacDef, buf, AJCTSIZE(buf));
335 :         TagJump(buf);
336 :     }
337 :     return TRUE;
338 : }
339 : //----- マクロ参照ウインドからの通知 ----------------------------------------------------------------------//
340 : AJC_DLGPROC(Main, IDC_VTH_MACREF    )
341 : {
342 :     BC  buf[4096];
343 :
344 :     if (HIWORD(wParam) == AJCVTHN_DBLCLK) {
345 :         AjcVthGetDblClickedLine(hVthMacRef, buf, AJCTSIZE(buf));
346 :         TagJump(buf);
347 :     }
348 :     return TRUE;
349 : }
350 : //----- 「Cancel」ボタン ----------------------------------------------------------------------------//
351 : AJC_DLGPROC(Main, IDCANCEL          )
352 : {
353 :     if (fBusy) {
354 :         fExit = TRUE;
355 :         AjcPpcStop(hPpc);
356 :     }
357 :     else DestroyWindow(hDlg);
358 :
359 :     return TRUE;
360 : }
361 : //---------------------------------------------------------------------------------------------------//
362 : AJC_DLGMAP_DEF(Main)
363 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG     )
364 :     AJC_DLGMAP_MSG(Main, WM_DESTROY        )
365 :
366 :     AJC_DLGMAP_CMD(Main, IDC_TXT_BASE      )
367 :     AJC_DLGMAP_CMD(Main, IDC_CMD_BASE      )
368 :     AJC_DLGMAP_CMD(Main, IDC_CMD_SRC       )
369 :     AJC_DLGMAP_CMD(Main, IDC_CMD_OUT       )
370 :     AJC_DLGMAP_CMD(Main, IDC_CHK_AUTOSRH   )
371 :     AJC_DLGMAP_CMD(Main, IDC_CMD_EXEC      )
372 :     AJC_DLGMAP_CMD(Main, IDC_CMD_CANCEL    )
373 :     AJC_DLGMAP_CMD(Main, IDC_VTH_LOG       )
374 :     AJC_DLGMAP_CMD(Main, IDC_VTH_OPTSYM    )
375 :     AJC_DLGMAP_CMD(Main, IDC_VTH_MACDEF    )
376 :     AJC_DLGMAP_CMD(Main, IDC_VTH_MACREF    )
377 :     AJC_DLGMAP_CMD(Main, IDCANCEL          )
378 : AJC_DLGMAP_END
379 :
380 : //---------------------------------------------------------------------------------------------------//
381 : //  ＰＰＣイベント通知                                                                               //
382 : //---------------------------------------------------------------------------------------------------//
383 : VO CALLBACK cbNtcPpcEvent(AJCPPCNOTIFY ntc, UX p1, UX p2, UX p3, UX cbp)
384 : {
385 :     HWND    hDlg = (HWND)cbp;
386 :     static  BC  IncName[MAX_PATH] = {0};
387 :
388 :     switch (ntc) {
389 :         case AJCPPC_NTC_ANYEVT:         //  いずれかのイベントが発生したことを通知      AJCPPCNOTIFY   , -             , -
390 :             AjcDoEvent();
391 :             break;
392 :
393 :         case AJCPPC_NTC_FILE_LNO:       //  ファイル名, 行番号通知                      "ファイル名"  , 行#          , ネスト値
394 :         {   C_BCP   pFile = (C_BCP)p1;
395 :             //  現在処理中のファイル名とネスト値表示
396 :             if (MAjcStrCmp(pFile, SvFileName) != 0) {
397 :                 if (p3 < 8) {
398 :                     if ((UI)p3 < CurNest) AjcSetDlgItemStr(hDlg, IDC_TXT_F0 + CurNest, "");
399 :                     AjcSetDlgItemStr(hDlg, IDC_TXT_F0 + (UI)p3, pFile);
```

```
400 :                   CurNest = (UI)p3;
401 :               }
402 :               MAjcStrCpy(SvFileName, MAX_PATH, pFile);
403 :           }
404 :           AjcSetDlgItemUInt(hDlg, IDC_LBL_LNO, (UI)p2);
405 :           break;
406 :       }
407 :       case AJCPPC_NTC_SRH_START:      //  インクルードファイル検索開始通知            "Inc ファイル名", "検索フォルダ", -
408 :           MAjcStrCpy(IncName, MAX_PATH, (BCP)p1);
409 :           break;
410 :
411 :       case AJCPPC_NTC_SRH_DIR:        //  インクルードファイル検索中のフォルダ通知    "フォルダパス", -               , -
412 :           if (AjcGetDlgItemChk(hDlg, IDC_CHK_INCSRH)) {
413 :               AjcVthPutText(hVthLog, "¥r¥x1B[2K¥r", -1);
414 :               AjcVthPrintF (hVthLog, "Search %-20s in %s", IncName, (BCP)p1);
415 :           }
416 :           break;
417 :
418 :       case AJCPPC_NTC_SRH_END:        //  インクルードファイル検索終了通知            "Inc ファイル名", 1:見つかった    , -
419 :           if (AjcGetDlgItemChk(hDlg, IDC_CHK_INCSRH)) {
420 :               AjcVthPutText(hVthLog, p2 ? "  - Find¥n" : "  - NotFound¥n", -1);
421 :           }
422 :           break;
423 :
424 :       case AJCPPC_NTC_OPTSYM:         //  プリプロセス用オプションシンボル通知        PCAJCPPCTKNNODE, -              , -
425 :       {   PCAJCPPCTKNNODE pTkn = (PCAJCPPCTKNNODE)p1;
426 :           if (!(AjcGetDlgItemChk(hDlg, IDC_CHK_OPTSYM) && *(pTkn->pSyl) == '_')) {
427 :               AjcVthPrintF(hVthOptSym, "%-20s : ¥"%s¥" <%d>¥n", pTkn->pSyl,
428 :                                                                 pTkn->pFile,
429 :                                                                 pTkn->lno);
430 :           }
431 :           break;
432 :       }
433 :       case AJCPPC_NTC_MACDEF:         //  マクロ定義通知                              PCAJCPPCTKNNODE, PCAJCPPCMACINFO, -
434 :       {   PCAJCPPCTKNNODE pTkn = (PCAJCPPCTKNNODE)p1;
435 :           PCAJCPPCMACINFO pMac = (PCAJCPPCMACINFO)p2;
436 :           if (!(AjcGetDlgItemChk(hDlg, IDC_CHK_MACDEF) && *(pMac->pMacName) == '_')) {
437 :               AjcVthPrintF(hVthMacDef, "%-20s : ¥"%s¥" <%d>¥n", pMac->pMacName,
438 :                                                                 pTkn->pFile,
439 :                                                                 pTkn->lno);
440 :           }
441 :           break;
442 :       }
443 :       case AJCPPC_NTC_MACREF:         //  マクロ参照通知                              PCAJCPPCTKNNODE, CAJCPPCMACINFO , -
444 :       {   PCAJCPPCTKNNODE pTkn = (PCAJCPPCTKNNODE)p1;
445 :           PCAJCPPCMACINFO pMac = (PCAJCPPCMACINFO)p2;
446 :           if (!(AjcGetDlgItemChk(hDlg, IDC_CHK_MACREF) && *(pMac->pMacName) == '_')) {
447 :               AjcVthPrintF(hVthMacRef, "%-20s : ¥"%s¥" <%d>¥n", pMac->pMacName,
448 :                                                                 pTkn->pFile,
449 :                                                                 pTkn->lno);
450 :           }
451 :           break;
452 :       }
453 :       case AJCPPC_NTC_OUTLOOP:        //  トークンストリームをファイルへ出力中通知    1:出力終了      , -               , -
454 :           break;
455 :
456 :       case AJCPPC_NTC_SRCTEC:         //  ソースファイルのテキストエンコード通知      ソースパス"     , ﾃｷｽﾄｴﾝｺｰﾄﾞ      , 1:BOM
457 :           AjcVthPutText(hVthLog, "¥r¥x1B[2K", -1);
458 :           AjcVthPutText(hVthLog, "入力ファイルのテキストエンコード : ", -1);
459 :           switch (p2) {
460 :               case AJCTEC_MBC:        AjcVthPutText(hVthLog, AJCLNGSEL("S-JIS¥n", "MBC¥n"), -1);  break;
461 :               case AJCTEC_UTF_8:      AjcVthPutText(hVthLog, "UTF-8¥n"     , -1);                 break;
462 :               case AJCTEC_EUC_J:      AjcVthPutText(hVthLog, "EUC-J¥n"     , -1);                 break;
463 :               case AJCTEC_UTF_16LE:   AjcVthPutText(hVthLog, "UTF-16(LE)¥n", -1);                 break;
464 :               case AJCTEC_UTF_16BE:   AjcVthPutText(hVthLog, "UTF-16(BE)¥n", -1);                 break;
465 :           }
466 :           break;
467 :
468 :       case AJCPPC_NTC_TOKEN:          //  トークン通知（ソースから読み出したトークン）トークンコード，“トークン文字列”，-
469 :           break;
470 :
471 :   }
472 : }
473 : //-------------------------------------------------------------------------------------------------------------//
474 : //  ＰＰＣエラー通知                                                                                           //
475 : //-------------------------------------------------------------------------------------------------------------//
476 : static  VO CALLBACK cbNtcPpcErr  (AJCPPCERROR err, UX p1, UX p2, UX p3, UX cbp)
477 : {
478 :     HWND    hDlg = (HWND)cbp;
479 :
```

```
480 :     AjcVthPutText(hVthLog, "¥r¥x1B[2K", -1);
481 :     AjcVthPutText(hVthLog, "¥x1B[31m" , -1);
482 :     AjcVthPutText(hVthLog, AjcPpcGetErrMsgText(err, p1, p2, p3), -1);
483 :     AjcVthPutText(hVthLog, "¥x1B[0m¥n", -1);
484 : }
485 : //-----------------------------------------------------------------------------------------------------------------//
486 : //  タグジャンプ（秀丸を起動）                                                                                     //
487 : //-----------------------------------------------------------------------------------------------------------------//
488 : static  VO   TagJump(BCP pLine)
489 : {
490 :     BCP     pPath = MAjcStrStr(pLine, "¥"");
491 :     BCP     pLno  = MAjcStrStr(pLine, "<");
492 :     int lno   = 0;
493 :     if (pPath != NULL && pLno != NULL) {
494 :         pPath++; MAjcStrTok(pPath, "¥"");
495 :         pLno++;  MAjcStrTok(pLno , ">" ); lno = AjcAscToInt(pLno);
496 :
497 :         if (AjcPathIsFile(pPath) && lno > 0) {
498 :             BC  szOpt[MAX_PATH + 64];
499 :             AjcSnPrintF(szOpt, AJCTSIZE(szOpt), "/J%d /m4 %s", lno, pPath);
500 :             ShellExecute(NULL, NULL, "Hidemaru.exe", szOpt, "C:¥¥Program Files (x86)¥¥Hidemaru¥¥", SW_SHOWNORMAL);
501 :         }
502 :     }
503 : }
504 :
```

40. 変数管理

インタプリタ的な変数管理ファンクションです。
「変数名（文字列)」に、「タイプ」と「変数値」を関連付けて管理します。
全ての変数は、要素数（１～）を指定し、配列として扱います。
変数のタイプは、32Bit 整数, 64Bit 整数, 実数と文字列を扱います。

40.1. 変数情報

変数の属性

変数の属性は以下のとおりです。

```
#define AJCVMGA_RDONLY  0x8000      //  Bit15   : 読み出し専用
#define AJCVMGA_USER    0x7FFF      //  Bit14-0 : ユーザ属性

#define AJCVMGA_NORMAL  0           //  読み書き可
```

「AJCVMGA_RDONLY」を設定した変数に書き込みはできません。
Bit14-0 はユーザが自由に設定／読み出し可能です。

変数のタイプ

変数のタイプは以下のとおりです。

```
typedef enum {
    AJCVMGT_UNDEF    = 0,       //  未定義／エラー
    AJCVMGT_INT      = 1,       //  ３２ビット整数(signed int)
    AJCVMGT_LONG       ,        //  ６４ビット整数(signed long long)
    AJCVMGT_REAL       ,        //  倍精度浮動小数(double)
    AJCVMGT_STR        ,        //  文字列
} AJCVMGTYPE, *PAJCVMGTYPE;
```

変数情報

変数情報は、以下の構造体で示します。（この情報を取得するには、AjcVmgGetNode()ＡＰＩを使用します）

```
//  変数値（配列）へのポインタ
typedef union {
        SIP      pSI;       //  ３２ビット整数
        SLLP     pSLL;      //  ６４ビット整数
        double  *pDBL;      //  実数
        UTP     *pStr;      //  文字列ポインタ
        VOP      vp;
} AJCVMGARRP, *PAJCVMGARRP;
typedef const AJCVMGARRP *PCAJCVMGARRP;

//  変数情報
typedef struct {
    UTP         pName;      //  変数名（文字列）へのポインタ
    AJCVMGTYPE  type;       //  変数のタイプ
    UI          num;        //  変数（配列）要素数
    AJCVMGARRP  arr;        //  変数値（配列）へのポインタ
} AJCVMGNODE, *PAJCVMGNODE;
typedef const AJCVMGNODE *PCAJCVMGNODE;
```

40.2. サポートＡＰＩ

変数管理のサポートＡＰＩ一覧を以下に示します。

#	関数名	内容
1	AjcVmgCreate	インスタンス生成
2	AjcVmgDelete	インスタンス消去
3	AjcVmgPurge	全変数情報の破棄
4	AjcVmgGenVar	変数の作成
5	AjcVmgSetAtt AjcVmgGetAtt	変数の属性設定／取得
6	AjcVmgDelVar	変数の削除
7	AjcVmgSetInteger AjcVmgSetReal AjcVmgSetString	変数値の設定
8	AjcVmgGetInt32 AjcVmgGetInt64 AjcVmgGetReal AjcVmgGetString	変数値の取得
9	AjcVmgSwap	変数値の交換
10	AjcVmgGetArrNum	配列要素数の取得
11	AjcVmgGetNode	変数情報の取得
12	AjcVmgEnumVar	登録済変数の列挙
13	AjcVmgCopy	変数コピー

40.2.1. インスタンス生成 (AjcVmgCreate)

形　式 ： HAJCVMG　AjcVmgCreate(VO);

引　数 ： なし

説　明 ： 変数管理の為の作業領域を生成します。

戻り値 ： ≠NULL　: 成功（インスタンスハンドル）
＝NULL　: 失敗

40.2.2. インスタンス消去 (AjcVMgDelete)

形　式 ： VO　AjcVmgDelete(HAJCVMG hVmg);

引　数 ： hVmg – インスタンスハンドル（AjcVMgCreate()の戻り値）

説　明 ： 変数管理の為の作業領域を解放します。

戻り値 ： なし

40.2.3. 全変数情報の破棄 (AjcVMgPurge)

形　式 ： VO　AjcVmgPurge(HAJCVMG hVmg);

引　数 ： hVmg – インスタンスハンドル（AjcVMgCreate()の戻り値）

説　明 ： 登録しているすべての変数情報を破棄します。

戻り値 ： なし

40.2.4. 変数の作成 (AjcVmgGenVar)

形　式 ： BOOL　AjcVmgGenVar(HAJCVMG hVmg, C_UTP pVarName, AJCVMGTYPE type, UI num);

引　数 ： hVmg　　- インスタンスハンドル
pVarName - 変数名文字列のアドレス
type　　- 変数のタイプ
num　　- 配列の要素数（０指定時は、要素数＝１として扱う）

説　明 ： pName で指定された変数名とタイプを登録します。
pName で指定された変数名が既に登録されている場合は、エラー(FALSE)を返します。
数値型変数（AJCVMGT_INT, AJCVMGT_LONG, AJCVMGT_REAL）の場合、変数の（各要素の）値はゼロに初期化されます。
文字列型変数の場合、変数の（各要素の）値は空文字列に初期化されます。

戻り値 ： TRUE　: 成功
FALSE : 失敗

40.2.5. 変数の属性設定／取得 (AjcVmg{Set/Get}Att)

形　式 ： BOOL　AjcVmgSetAtt (HAJCVMG hVmg, C_UTP pVarName, UI att); -- 変数の属性設定
UI　AjcVmgGetAtt (HAJCVMG hVmg, C_UTP pVarName); ---------- 変数の属性取得

引　数 ： hVmg – インスタンスハンドル (AjcVMgCreate()の戻り値)
pVarName - 変数名文字列のアドレス
att　　- 変数の属性

説　明 ： 登録済の変数の属性を設定／取得します。
att には以下の属性値を指定します。

#	名称	値	内容
1	AJCVMGA_RDONLY	0x8000 (Bit15)	読み出しのみ可能
2	AJCVMGA_NORMAL	0	読み書き可能

※Bit14-0 は自由に設定／読み出しができます。

戻り値 ： 設定時：TRUE　- 成功　　取得時：≠-1 - 成功 (AjcVmgSetAtt()で設定した、変数の属性値)
FALSE - 失敗　　=-1 - 失敗

40.2.6. 変数の削除 (AjcVmgDelVar)

形　式 ： BOOL　AjcVmgDelVar(HAJCVMG hVmg, C_UTP pVarName);

引　数 ： hVmg　　- インスタンスハンドル
pVarName - 変数名文字列のアドレス

説　明 ： pName で指定された変数を登録解除します。
pName で指定された変数名が登録されていない場合は、エラー(FALSE)を返します。

戻り値 ： TRUE　: 成功
FALSE : 失敗

40.2.7. 変数値の設定 (AjcVmgSetInteger, AjcVmgSetReal, AjcVmgSetString)

形　式 ： BOOL AjcVmgSetInteger (HAJCVMG hVmg, C_BCP pVarName, UI ix, SLL value); ・・・ 整数値設定
BOOL AjcVmgSetReal (HAJCVMG hVmg, C_BCP pVarName, UI ix, double value); ・・・ 実数値設定
BOOL AjcVmgSetString (HAJCVMG hVmg, C_BCP pVarName, UI ix, C_UTP pStr); ・・・ 文字列設定

引　数 ： hVmg - インスタンスハンドル
pVarName - 変数名文字列のアドレス
ix - 配列の要素インデクス（０～）
value - 設定する数値
pStr - 設定する文字列

説　明 ： pName で指定された変数（の配列要素）に値を設定します。
pName で指定された変数名が登録されていない場合は、エラー(FALSE)を返します。
AjcVmgSetInteger()と AjcVmgSetReal()は、数値型（AJCVMGT_INT, AJCVMGT_LONG, AJCVMGT_REAL）の変数へ数値を設定します。この際、設定値(value)と変数のタイプが異なる場合は、変数のタイプに合わせてキャストされます。
AjcVmgSetString()は、文字列型の変数へ文字列を設定します。

戻り値 ： TRUE ： 成功
FALSE ： 失敗

40.2.8. 変数値の取得 (AjcVmgGetInt32, AjcVmgGetInt 64, AjcVmgGetReal, AjcVmgGetString)

形　式 ： AJCVMGTYPE AjcVmgGetInt32 (HAJCVMG hVmg, C_BCP pVarName, UI ix, SIP pValue); ・・ 32Bit 整数値取得
AJCVMGTYPE AjcVmgGetInt64 (HAJCVMG hVmg, C_BCP pVarName, UI ix, SLLP pValue); ・・ 64Bit 整数値取得
AJCVMGTYPE AjcVmgGetReal (HAJCVMG hVmg, C_BCP pVarName, UI ix, double *pValue); ・・ 実数値取得
C_UTP AjcVmgGetString (HAJCVMG hVmg, C_BCP pVarName, UI ix); ・・ 文字列取得

引　数 ： hVmg - インスタンスハンドル
pVarName - 変数名文字列のアドレス
ix - 配列の要素インデクス（０～）
pValue - 数値型整数の値を格納するバッファのアドレス

説　明 ： pName で指定された変数（の配列要素）の値を取得します。
pName で指定された変数名が登録されていない場合や変数のタイプ（数値型／文字型）が異なる場合は、エラー(0 / NULL)を返します。

AjcVmgGetInt32(),AjcVmgGetInt 64()と AjcVmgGetReal()は、数値型（AJCVMGT_INT, AJCVMGT_LONG, AJCVMGT_REAL）の変数（の配列要素）値を取得します。
この際、格納バッファ(pValue)と変数のタイプが異なる場合は、格納バッファに合わせてキャストされます。

AjcVmgGetString()は、文字列型変数（の配列要素）の内容（文字列のアドレス）を返します。

戻り値 ： AjcVmgGetInt32(),AjcVmgGetInt 64()と AjcVmgGetReal()の場合
≠０ ： 成功(AJCVMGT_INT / AJCVMGT_LONG / AJCVMGT_REAL)
＝０ ： 失敗(AJCVMGT_UNDEF)

AjcVmgGetString()の場合
≠NULL：成功（文字列のアドレス）
＝NULL：失敗

40.2.9. 変数値の交換 (AjcVmgSwap)

形　式 ： BOOL AjcVmgSwap (HAJCVMG hVmg, C_UTP pVarName1, UI ix1, C_UTP pVarName2, UI ix2);

引　数 ： hVmg　　- インスタンスハンドル
pVarName1, pVarName2 - 値を交換する２つの変数名文字列のアドレス
ix1　　, ix2　　- 値を交換する２つの変数の配列要素インデクス（０～）

説　明 ： pVarName1,ix1 と pVarName2,ix2 で指定された２つの変数（の配列要素）の値を交換します。
pVarName1 か pVarName2 で指定された変数名が登録されていない、あるいは、タイプ(数値型／文字型)が異なる場合は、エラー(AJCVMGT_UNDEF)を返します。

戻り値 ： TRUE　: 成功
FALSE : 失敗

40.2.10. 配列要素数の取得 (AjcVmgGetArrNum)

形　式 ： UI　AjcVmgGetArrNum (HAJCVMG hVmg, C_UTP pVarName);

引　数 ： hVmg　　- インスタンスハンドル
pVarName - 変数名文字列のアドレス

説　明 ： pVarName で指定された変数の配列要素数を返します。
pVarName で指定された変数名が登録されていない場合は、０を返します。

戻り値 ： ≠０ : 成功(変数の配列要素数)
＝０ : 失敗

40.2.11. 変数情報の取得 (AjcVmgGetNode)

形　式 ： PCAJCVMGNODE AjcVmgGetNode (HAJCVMG hVmg, C_UTP pVarName);

引　数 ： hVmg　　- インスタンスハンドル
pVarName - 変数名文字列のアドレス

説　明 ： pVarName で指定された変数の情報を取得します。
pVarName で指定された変数名が登録されていない場合は、エラー(AJCVMGT_UNDEF)を返します。

戻り値 ： ≠NULL : 成功(ノードのアドレス)
＝NULL : 失敗

40.2.12. 登録済変数の列挙 (AjcVmgEnumVar)

形　式 ： UI AjcVmgEnumVar (HAJCVMG hVmg, UX cbp, BOOL (CALLBACK *cbNtcVar)(PCAJCVMGNODE pVarInfo, UX cbp));

引　数 ： hVmg　　- インスタンスハンドル
cbp　　- コールバックパラメタ
cbNtcVar - 登録済変数の通知用コールバック関数のアドレス

説　明 ： コールバック関数(cbNtcVar)を呼び出すことにより、登録済変数群を通知します。
途中で変数の通知を中止する場合は、コールバック関数で FALSE を返してください。

戻り値 ： 登録済変数の個数

コールバック：

cbNtcVar（登録済変数の通知用コールバック関数）

形　式 ： BOOL CALLBACK *cbNtcVar* (PCAJCVMGNODE pVarInfo, UX cbp);

引　数 ： pVarInfo　- 変数情報へのポインタ
cbp　　- コールバックパラメタ

説　明 ： 登録済変数群の通知を受けます。

戻り値 ： TRUE　: 変数の通知を継続する
FALSE : 変数の通知を中止する

40.2.13. 変数のコピー (AjcVmgCopy)

形　式 ： BOOL AjcVmgEnumVar (HAJCVMG hVmgD, C_BCP pNameD, HAJCVMG hVmgS, C_BCP pNameS);

引　数 ： hVmgD, pNameD – 受け側のインスタンスハンドルと、変数名文字列のアドレス
hVmgS, pNameS – 送り側のインスタンスハンドルと、変数名文字列のアドレス

説　明 ： 変数の内容（全ての配列要素）をコピーします。
受け側に、既に pNameD で指定した変数が存在しない場合は、当該変数を生成します。
受け側に、既に pNameD で指定した変数が存在する場合は、当該変数を一旦削除して、再度生成します。
同一のインスタンス内で変数をコピーする場合は、hVmgD と hVmgS に同じインスタンスを指定します。
異なるインスタンス間で変数をコピーする場合は、各々のインスタンスを指定します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

40.3. サンプルプログラム

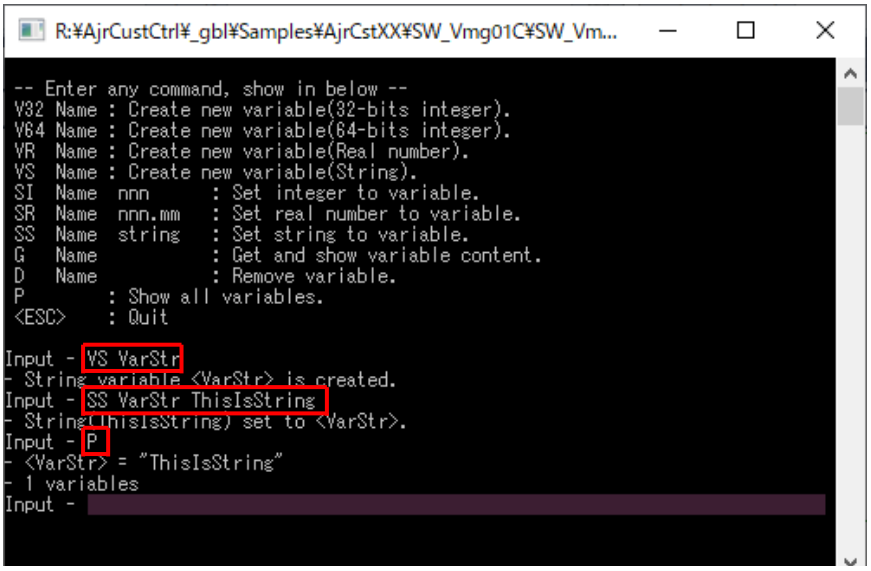
40.3.1. SW_Vmg01C（変数の作成／設定／表示）

このサンプルプログラムは、以下のコマンドにより、変数の作成，値設定等を行います。
（コマンド(CMD) に続けてパラメタを指定します。　コマンドと各パラメタの間は空白で区切ります）

#	CMD	パラメタ１	パラメタ２	備考	例
1	V32	変数名	－	変数生成（32Bit 整数）	V32 Var32
2	V64	変数名	－	〃　（64Bit 整数）	V64 Var64
3	VR	変数名	－	〃　（実数）	VR VarReal
4	VS	変数名	－	〃　（文字列）	VS VarStr
5	SI	変数名	設定値	整数の設定	SI Var32 100 SI Var64 200
6	SR	変数名	設定値	実数の設定	SR VarReal 3.14
7	SS	変数名	設定値	文字列の設定	SS VarStr ThisIsString
8	G	変数名	－	変数の表示	G Var32
9	D	変数名	－	変数の削除	D Var32
10	P	－	－	全変数の表示	P
11	ESC キー	－	－	終了	Q

変数は単一要素の配列（要素数＝１）として扱います。

アプリケーション実行ウインド



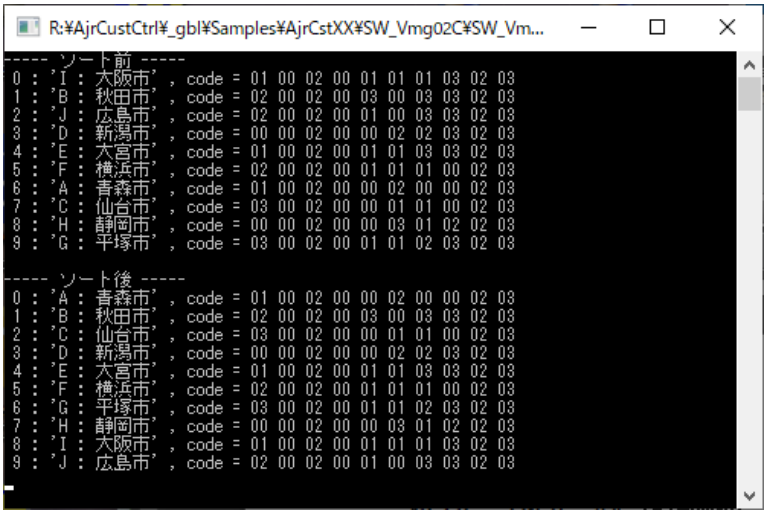

```
 1 : //
 2 : //  SW_Vmg01C.C
 3 : //
 4 : #include    <AjrCstXX.h>
 5 : #include    <stdio.h>
 6 : #include    <conio.h>
 7 : #include    <time.h>
 8 : #include    <tchar.h>
 9 :
10 : //----- コールバック（変数情報の通知）-------------------------------------------------------------------------//
11 : static BOOL CALLBACK cbNtcVar(PCAJCVMGNODE pVarInfo, UX cbp)
12 : {
13 :     switch (pVarInfo->type) {
14 :         case AJCVMGT_INT:   AjcPrintF(TEXT("- <%s> = %d¥n"  )  , pVarInfo->pName, pVarInfo->arr.pSI [0]); break;
15 :         case AJCVMGT_LONG:  AjcPrintF(TEXT("- <%s> = %lld¥n") , pVarInfo->pName, pVarInfo->arr.pSLL[0]); break;
16 :         case AJCVMGT_REAL:  AjcPrintF(TEXT("- <%s> = %f¥n"  )  , pVarInfo->pName, pVarInfo->arr.pDBL[0]); break;
17 :         case AJCVMGT_STR:   AjcPrintF(TEXT("- <%s> = ¥"%s¥"¥n"), pVarInfo->pName, pVarInfo->arr.pStr[0]); break;
18 :     }
19 :     return TRUE;
20 : }
21 :
22 : //=============================================================================================================//
23 : int  AjcMain(int argc, UTP argv[])
24 : {
```

```
     25 :     HAJCVMG     hVmg;
     26 :     UT          InpBuf[256] = {0};
     27 :
     28 :     AjcSetStdoutMode();
     29 :
     30 :     //----- インスタンス生成 ----------------------------------------------------------------------------------------//
     31 :     hVmg = AjcVmgCreate();
     32 :
     33 :     //----- コマンドガイド表示 --------------------------------------------------------------------------------------//
     34 :     AjcPrintF(TEXT("¥n"));
     35 :     AjcPrintF(TEXT(" -- Enter any command, show in below --¥n"));
     36 :     AjcPrintF(TEXT(" V32 Name : Create new variable(32-bits integer).¥n"));
     37 :     AjcPrintF(TEXT(" V64 Name : Create new variable(64-bits integer).¥n"));
     38 :     AjcPrintF(TEXT(" VR  Name : Create new variable(Real number).¥n"));
     39 :     AjcPrintF(TEXT(" VS  Name : Create new variable(String).¥n"));
     40 :     AjcPrintF(TEXT(" SI  Name  nnn      : Set integer to variable.¥n"));
     41 :     AjcPrintF(TEXT(" SR  Name  nnn.mm   : Set real number to variable.¥n"));
     42 :     AjcPrintF(TEXT(" SS  Name  string   : Set string to variable.¥n"));
     43 :     AjcPrintF(TEXT(" G   Name           : Get and show variable content.¥n"));
     44 :     AjcPrintF(TEXT(" D   Name           : Remove variable.¥n"));
     45 :     AjcPrintF(TEXT(" P        : Show all variables.¥n"));
     46 :     AjcPrintF(TEXT(" <ESC>    : Quit¥n"));
     47 :     AjcPrintF(TEXT("¥n"));
     48 :
     49 :     //----- コマンド処理 --------------------------------------------------------------------------------------------//
     50 :     AjcPrintF(TEXT("Input - "));
     51 :     while (AjcConInput(InpBuf, AJCTSIZE(InpBuf))) {
     52 :         UTP p1 = MAjcStrTok(InpBuf, TEXT(" "));
     53 :         UTP p2 = MAjcStrTok(NULL  , TEXT(" "));
     54 :         UTP p3 = MAjcStrTok(NULL  , TEXT(" "));
     55 :         //------------------------------------------------------------------------------------------------------//
     56 :         if      (p1 && p2 && MAjcStrICmp(p1, TEXT("V32")) == 0) {              //● V32 Name : Create new variable(32-bits
integer)
     57 :             if (AjcVmgGenVar(hVmg, p2, AJCVMGT_INT, 1)) {
     58 :                 AjcPrintF(TEXT("- 32-bits integer variable <%s> is created.¥n"), p2);
     59 :             }
     60 :             else AjcPrintF(TEXT("* Variable (<%s>) creation failure.¥n"), p2);
     61 :         }
     62 :         //------------------------------------------------------------------------------------------------------//
     63 :         else if (p1 && p2 && MAjcStrICmp(p1, TEXT("V64")) == 0) {              //● V64 Name : Create new variable(64-bits
integer)
     64 :             if (AjcVmgGenVar(hVmg, p2, AJCVMGT_LONG, 1)) {
     65 :                 AjcPrintF(TEXT("- 64-bits integer variable <%s> is created.¥n"), p2);
     66 :             }
     67 :             else AjcPrintF(TEXT("* Variable (<%s>) creation failure.¥n"), p2);
     68 :         }
     69 :         //------------------------------------------------------------------------------------------------------//
     70 :         else if (p1 && p2 && MAjcStrICmp(p1, TEXT("VR"))  == 0) {              //● VR  Name : Create new variable(Real
number)
     71 :             if (AjcVmgGenVar(hVmg, p2, AJCVMGT_REAL, 1)) {
     72 :                 AjcPrintF(TEXT("- Real number variable <%s> is created.¥n"), p2);
     73 :             }
     74 :             else AjcPrintF(TEXT("* Variable (<%s>) creation failure.¥n"), p2);
     75 :         }
     76 :         //------------------------------------------------------------------------------------------------------//
     77 :         else if (p1 && p2 && MAjcStrICmp(p1, TEXT("VS"))  == 0) {              //● VS  Name : Create new variable(String)
     78 :             if (AjcVmgGenVar(hVmg, p2, AJCVMGT_STR, 1)) {
     79 :                 AjcPrintF(TEXT("- String variable <%s> is created.¥n"), p2);
     80 :             }
     81 :             else AjcPrintF(TEXT("* Variable <%s> creation failure.¥n"), p2);
     82 :         }
     83 :         //------------------------------------------------------------------------------------------------------//
     84 :         else if (p1 && p2 && p3 && MAjcStrICmp(p1, TEXT("SI"))  == 0) {        //● SI  Name nnn    : Set integer to variable.
     85 :             SLL n = AjcAscToLInt(p3);
     86 :             if (AjcVmgSetInteger(hVmg, p2, 0, n)) {
     87 :                 AjcPrintF(TEXT("- Integer value(%lld) set to <%s>.¥n"), n, p2);
     88 :             }
     89 :             else AjcPrintF(TEXT("* Integer value setting failure.¥n"));
     90 :         }
     91 :         //------------------------------------------------------------------------------------------------------//
     92 :         else if (p1 && p2 && p3 && MAjcStrICmp(p1, TEXT("SR"))  == 0) {        //● SR  Name nnn.mm : Set real-number to
variable.
     93 :             double  n = AjcAscToReal(p3);
     94 :             if (AjcVmgSetReal(hVmg, p2, 0, n)) {
     95 :                 AjcPrintF(TEXT("- Real number(%f) set to <%s>.¥n"), n, p2);
     96 :             }
     97 :             else AjcPrintF(TEXT("* Real number setting failure.¥n"));
     98 :         }
     99 :         //------------------------------------------------------------------------------------------------------//
    100 :         else if (p1 && p2 && p3 && MAjcStrICmp(p1, TEXT("SS"))  == 0) {        //● SS  Name string : Set string to variable.
```

```
    101 :             if (AjcVmgSetString(hVmg, p2, 0, p3)) {
    102 :                 AjcPrintF(TEXT("- String(%s) set to <%s>.¥n"), p3, p2);
    103 :             }
    104 :             else AjcPrintF(TEXT("* String setting failure.¥n"));
    105 :         }
    106 :         //-------------------------------------------------------------------------------------------------//
    107 :         else if (p1 && p2 && MAjcStrICmp(p1, TEXT("G"))  == 0) {               //● G   Name        : Get and show variable
content
    108 :             PCAJCVMGNODE  pNode = AjcVmgGetNode(hVmg, p2);
    109 :             if (pNode->type != AJCVMGT_UNDEF) {
    110 :                 SI      si;
    111 :                 SLL     sll;
    112 :                 double  dbl;
    113 :                 switch (pNode->type) {
    114 :                     case AJCVMGT_INT:   AjcVmgGetInt32(hVmg, p2, 0, &si ); AjcPrintF(TEXT("- <%s> = %d¥n"  ), p2, si ); break;
    115 :                     case AJCVMGT_LONG:  AjcVmgGetInt64(hVmg, p2, 0, &sll); AjcPrintF(TEXT("- <%s> = %lld¥n"), p2, sll); break;
    116 :                     case AJCVMGT_REAL:  AjcVmgGetReal (hVmg, p2, 0, &dbl); AjcPrintF(TEXT("- <%s> = %f¥n"  ), p2, dbl); break;
    117 :                     case AJCVMGT_STR:   AjcPrintF(TEXT("- <%s> = ¥"%s¥"¥n"), p2, AjcVmgGetString(hVmg, p2, 0));         break;
    118 :                 }
    119 :             }
    120 :             else AjcPrintF(TEXT("* Variable <%s> not found.¥n"), p2);
    121 :         }
    122 :         //-------------------------------------------------------------------------------------------------//
    123 :         else if (p1 && p2 && MAjcStrICmp(p1, TEXT("D"))  == 0) {               //● D   Name        : Remove variable
    124 :             if (AjcVmgDelVar(hVmg, p2)) {
    125 :                 AjcPrintF(TEXT("- Variable <%s> is removed.¥n"), p2);
    126 :             }
    127 :             else AjcPrintF(TEXT("* Variable <%s> not found.¥n"), p2);
    128 :         }
    129 :         //-------------------------------------------------------------------------------------------------//
    130 :         else if (p1 && MAjcStrICmp(p1, TEXT("P"))   == 0) {                    //● P               : Show all variable
    131 :             UI  n;
    132 :             n = AjcVmgEnumVar(hVmg, 0, cbNtcVar);
    133 :             AjcPrintF(TEXT("- %d variables¥n"), n);
    134 :         }
    135 :         else {
    136 :             AjcPrintF(TEXT("* Missing parameter.¥n"));
    137 :         }
    138 :         InpBuf[0] = 0;
    139 :         AjcPrintF(TEXT("Input - "));
    140 :     }
    141 :     AjcVmgDelete(hVmg);
    142 :
    143 :     return 0;
    144 : }
    145 :
```

40.3.2. SW_Vmg02C（配列変数）

以下のサンプルプログラムは、文字列型の配列変数（地域）を作成し、適当な文字列を設定後、文字列の昇順(北の都市からの順)にソートします。




```
 1 : //
 2 : //  SW_Vmg02C.C
 3 : //
 4 : #include    <AjrCstXX.h>
 5 : #include    <stdio.h>
 6 : #include    <conio.h>
 7 : #include    <time.h>
 8 : #include    <tchar.h>
 9 :
10 : #define     VAR_NAME    TEXT("地域")
11 :
12 :
13 :
14 : //=============================================================================================================//
15 : static VO   SubPrint(HAJCVMG hVmg, UTP pName)
16 : {
17 :     PCAJCVMGNODE    pNode = AjcVmgGetNode(hVmg, VAR_NAME);
18 :     UI              i, j;
19 :     C_UTP           pStr;
20 :
21 :     for (i = 0; i < pNode->num; i++) {
22 :         pStr = AjcVmgGetString(hVmg, pName, i);
23 :         AjcPrintF(TEXT("%2d : '%s' , code ="), i, pStr);
24 :         for (j = 0; pStr[j] != 0; j++) {
25 :             AjcPrintF(TEXT(" %0*X"), (UI)(sizeof(UT) * 2), pStr[j] & ((1 << (sizeof(UT) * 2)) - 1));
26 :         }
27 :         AjcPrintF(TEXT("¥n"));
28 :     }
29 : }
30 :
31 :
32 : //=============================================================================================================//
33 : int  AjcMain(int argc, UTP argv[])
34 : {
35 :     HAJCVMG     hVmg;
36 :     UI          i, j;
37 :
38 :     AjcSetStdoutMode();
39 :
40 :     //----- インスタンス生成 ------------------------------------------------------------------------------------//
41 :     hVmg = AjcVmgCreate();
42 :
43 :     //----- 文字列変数（地域）を生成し、適当な値を設定 ----------------------------------------------------------//
44 :     AjcVmgGenVar(hVmg, VAR_NAME, AJCVMGT_STR, 10);
45 :     AjcVmgSetString(hVmg, VAR_NAME, 0, TEXT("I : 大阪市"));
46 :     AjcVmgSetString(hVmg, VAR_NAME, 1, TEXT("B : 秋田市"));
47 :     AjcVmgSetString(hVmg, VAR_NAME, 2, TEXT("J : 広島市"));
48 :     AjcVmgSetString(hVmg, VAR_NAME, 3, TEXT("D : 新潟市"));
49 :     AjcVmgSetString(hVmg, VAR_NAME, 4, TEXT("E : 大宮市"));
50 :     AjcVmgSetString(hVmg, VAR_NAME, 5, TEXT("F : 横浜市"));
51 :     AjcVmgSetString(hVmg, VAR_NAME, 6, TEXT("A : 青森市"));
```

```
52 :     AjcVmgSetString(hVmg, VAR_NAME, 7, TEXT("C : 仙台市"));
53 :     AjcVmgSetString(hVmg, VAR_NAME, 8, TEXT("H : 静岡市"));
54 :     AjcVmgSetString(hVmg, VAR_NAME, 9, TEXT("G : 平塚市"));
55 :
56 :     //----- ソート前表示 ----------------------------------------------------------------------------------------//
57 :     AjcPrintF(TEXT("----- ソート前 -----¥n"));
58 :     SubPrint(hVmg, VAR_NAME);
59 :
60 :     //----- ソート ----------------------------------------------------------------------------------------------//
61 :     for (i = 0; i < 9; i++) {
62 :         C_UTP pSi = AjcVmgGetString(hVmg, VAR_NAME, i);
63 :         for (j = i + 1; j < 10; j++) {
64 :             C_UTP pSj = AjcVmgGetString(hVmg, VAR_NAME, j);
65 :             if (MAjcStrCmp(pSi, pSj) > 0) {
66 :                 AjcVmgSwap(hVmg, VAR_NAME, i, VAR_NAME, j);
67 :                 pSi = pSj;
68 :             }
69 :         }
70 :     }
71 :
72 :     //----- ソート後表示 ----------------------------------------------------------------------------------------//
73 :     AjcPrintF(TEXT("¥n----- ソート後 -----¥n"));
74 :     SubPrint(hVmg, VAR_NAME);
75 :
76 :     //----- インスタンス消去 ------------------------------------------------------------------------------------//
77 :     AjcVmgDelete(hVmg);
78 :
79 :     _gettchar();
80 :
81 :      return 0;
82 : }
83 :
```

41. 文字列操作

文字列の操作に関する関数群です。

41.1. 多重文字列

複数の文字列を連ねて、終端にヌル文字を付加したものを多重文字列と言うことにします。
単一の文字列はヌル文字（0x00）で終端しますが、多重文字列は、連なった文字列群の終端として、さらにもう１ヶのヌル文字を付加します。

例えば、３つの文字列（″ABC″, ″DEF″ と ″XYZ″）を含む多重文字列は、以下のようになります。

″ABC″, ″DEF″ と ″XYZ″を含む多重文字列

A	B	C	00	D	E	F	00	W	X	Y	Z	00	00

※ 00 は、文字列の終端(0x00)を意味します

上記の多重文字列をＣ言語の文字列定数で表現すると、″ABC¥0DEF¥0XYZ¥0″ となります。

文字列が１つも無い多重文字列は、単に１つのヌル文字となります。

文字列が１つも無い多重文字列

00

41.2. マクロ

マクロ形式, 展開形	内容	使用例

AJCTSIZE(_TEXT_)
+(sizeof _TEXT_ / sizeof _TEXT[0])
 	
あるいは
 	
AJCSZARR(_ARR_)
+(sizeof _ARR_ / sizeof _ARR[0])
	配列の要素数
	(文字列変数の場合は
	変数のバイト数／文字数
		BC bstr[64]; // バイト文字列
		WC wstr[64]; // UNICODE 文字列
		UT ustr[64]; // バイト文字列 ／ UNICODE 文字列
		int blen = AJCTSIZE(bstr); // blen = 64[バイト]
		int wlen = AJCTSIZE(wstr); // wlen = 64[文字]
		int ulen = AJCTSIZE(ustr); // ulen = 64[バイト／文字]
		
		POINT pt[64];
		int ptlen = AJCSZARR(pt); // ptlen = 64

AJCOPTn(PFX, P1, P2, ・・・ Pn) +(PFX_P1 | PFX_P2 | ・・・ PFX_Pn)	接頭語(PFX)と P1～P16 の 合成値（論理和）を返す 「n」は 1～16 で最大 16 個 のシンボルの合成値を返す	#define OPT_FIX 0x10 #define OPT_ADJUST 0x20 #define OPT_SEND 0x40 UI opt = AJCOPT**3**(OPT_, FIX, ADJUST, SEND); // opt = 0x07

41.3. サポートＡＰＩ

文字列操作のサポートＡＰＩ一覧を以下に示します。

#	関数名	内容
1	AjcStrCmpEx	文字列比較方法を指定して文字列を比較する
2	AjcStrIStr, AjcStrStrEx	英文字の大小を区別しないで部分文字列検索（単純バイト比較）
3	AjcMbsIStr, AjcMbsStrEx	英文字の大小を区別しないで部分文字列検索（マルチバイト比較）
4	AjcStr[I]Find	複数の文字列を指定して部分文字列検索
5	AjcStrTok / AjcStrTokS	文字列内のトークンを検索
6	AjcMbsTok / AjcMbsTokS	文字列内のトークンを検索（マルチバイト文字列）
7	AjcStrTokEx / AjcStrTokSEx	区切り文字で区切られた文字列内のトークンを取得（囲み文字指定）
8	AjcStrEnclose	文字列を指定文字で囲む
9	AjcStrStripEnclose	文字列の囲みを取り除く
10	AjcSnPrintF	書式文字列の生成
11	AjcSnPrtSep	書式文字列生成と１０進数文字列の整数部で特定桁数毎に区切り文字を挿入する
12	AjcStrRmvSepChar	１０進数文字列の整数部から区切り文字を削除する
13	AjcVSnPrintF	引数情報を指定した書式文字列の生成
14	AjcMStrMakeArray	多重文字列→文字列ポインタ配列生成
15	AjcMStrReleaseArray	文字列ポインタ配列開放
16	AjcMStrCount	多重文字列中の文字列の個数取得
17	AjcMStrTotalSize	多重文字列の総文字数取得
18	AjcRepPartStr	部分文字列置換
19	AjcStrLTrim[Ex]	文字列の前部空白を除去
20	AjcStrRTrim[Ex]	文字列の後部空白を除去
21	AjcStrTrim[Ex]	文字列の両端の空白を除去
22	AjcRemoveEscInStr	文字列中の ESC シーケンスを除去
23	AjcStrShaping	文字列の整形 (先頭と末尾の不要文字群を削除し、中間の連続した文字群を指定文字列に置き換える)
24	AjcStrChkMbcPart	マルチバイト（シフトＪＩＳ）文字列情報の取得
25	AjcStrChkUtf8Part	ＵＴＦ－８文字情報の取得
26	AjcStrChkEucPart	日本語ＥＵＣ文字情報の取得
27	AjcUtf8ToSJis	ＵＴＦ８→ マルチバイト（シフトＪＩＳ）文字列変換
28	AjcSJisToUtf8	マルチバイト（シフトＪＩＳ） →ＵＴＦ８文字列変換
29	AjcEucToSJis	日本語ＥＵＣ → シフトＪＩＳ文字列変換
30	AjcSJisToEuc	シフトＪＩＳ → 日本語ＥＵＣ文字列変換
31	AjcStrChkCode	文字コード判別（シフトＪＩＳ／日本語ＥＵＣ／ＵＴＦ-８）
32	AjcStrChkCodeEx	文字コード判別（シフトＪＩＳ／日本語ＥＵＣ／ＵＴＦ-８／ＵＴＦ-１６）
33	AjcStrChk{Mbc/Utf8/Euc/ U16Le/U16Be}Len	１文字の長さ（バイト数／ワード）取得
34	AjcStrAdjustLen	文字列の長さ補正
35	AjcStrCnvRevPath	パスとファイル名を反対にした文字列を作成する
36	AjcStrRetRevPath	パスとファイル名を反対にしたパス名を元に戻す
37	AjcStrSameAsAllChar	文字列が全て同一文字で構成されているかチェック

つづく

つづき

#	関数名	内容
38	AjcHexToUI / ULL	１６進文字列を数値に変換
39	AjcOctToUI / ULL	８進文字列を数値に変換
40	AjcHexStrToUB / UW / UI / ULL	１６進文字列を数値に変換［１６進文字チェック付］
41	AjcAscToInt / LInt	１０／１６進文字列を数値に変換
42	AjcAscToReal	１０／１６進文字列を実数値に変換
43	AjcGetSepNumbers	文字列中の区切られた複数の数値を取得する
44	AjcGetAfterPrefix	接頭語に続くパラメタ取得
45	AjcExtractCompletedText	マルチバイトが分断されないようにマルチバイトテキストを取り出す
46	AjcReleaseCompletedText	マルチバイトが分断されないように抽出したマルチバイトテキストを解放
47	AjcCLangStrToBin	Ｃ言語表記文字列をバイナリ化
48	AjcBinToCLangStr	バイナリ文字列をＣ言語表記文字列化
49	AjcTextLen	テキストの表示桁数取得
50	AjcIsBigChar	全角文字チェック
51	AjcTextToCodePoint	UNICODE コードポイント算出
52	AjcCodePointToText	UNICODE コードポイントを文字列に変換
53	AjcLogFontToText	フォント情報(LOGFONT)を文字列に変換
54	AjcTextToLogFont	文字列をフォント情報(LOGFONT) に変換

41.3.1. 比較方法を指定して文字列比較（AjcStrCmpEx）

形　式 ： int AjcStrCmpEx(C_UTP pStr, C_UTP pPart, EAJCCMPMODE CmpMode);

引　数 ： pStr1 － 比較索文字列１の先頭アドレス
pStr2 － 比較索文字列２の先頭アドレス
CmpMode － 文字列比較方法
・AJCCMP_EXACT_WIDTH ： 英大小文字を区別する
・AJCCMP_IGNORE_WIDTH ： 英大小文字を区別しない
・AJCCMP_MIX ： 同一の場合のみ英大小文字を区別する

説　明 ： 文字列の比較方法（英字の扱い）を指定して文字列を比較します。
CmpMode = AJCCMP_EXACT_WIDTH とした場合は、英大小文字を区別して文字列の比較を行います。(”ABC”≠”abc”)
CmpMode = AJCCMP_IGNORE_WIDTH とした場合は、英大小文字を区別せずに文字列の比較を行います。(”ABC”＝”abc”)
CmpMode = AJCCMP_MIX とした場合は、英大小文字を区別せずに文字列の比較を行いますが、同一文字列の場合は、英大小文字を区別して文字列の比較を行います。(アルファベット順に並ぶように比較する　ex. “AAA” > “aaa” > “BBB”)

戻り値 ： ＜ 0 － 比較索文字列１ ＜ 比較索文字列２
＝ 0 － 比較索文字列１ ＝ 比較索文字列２
＞ 0 － 比較索文字列１ ＞ 比較索文字列２

41.3.2. 部分文字列検索（AjcStrIStr，単純バイト比較）

形　式 ： UTP AjcStrIStr (C_UTP pStr, C_UTP pPart);
UTP AjcStrStrEx(C_UTP pStr, C_UTP pPart, BOOL fIgnoreWidth);

引　数 ： pStr － 非検索文字列の先頭アドレス
pPart － 検索する部分文字列の先頭アドレス
fIgnoreWidth － 文字列比較方法(FALSE:英大小を区別する，TRUE: 英大小を区別しない)

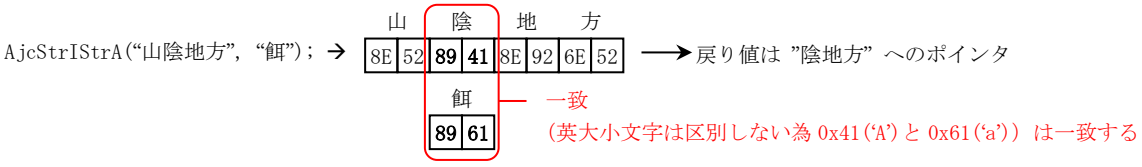
説　明 ： AjcStrIStr()は、英字の大文字／小文字を区別しないで、非検索文字列中の部分バイト文字列を検索します。
AjcStrStrEx()は、文字列の比較方法を指定して非検索文字列中の部分文字列を検索します。
fIgnoreWidth の指定により、以下の関数を実行します。

fIgnoreWidth	実行関数	
	バイト文字	ワイド文字(UNICODE)
FALSE（英大小を区別する）	strstr	wcsstr
TRUE （英大小を区別しない）	AjcStrIStrA	AjcStrIStrW

検索する部分文字列の長さが０（つまり、空文字列）を指定した場合は、非検索文字列の先頭アドレスを返します。

戻り値 ： ≠NULL － 非検索文字列中で、最初に見つかった部分文字列のアドレス
＝NULL － 部分文字列は見つからなかった

備　考 ： AjcStrIStrA()は、単純にバイト単位の文字で比較する為、異なる文字列が一致する場合があります。



41.3.3. 部分文字列検索（AjcMbsIStr，マルチバイト比較）

形　式 ： UTP　AjcMbsIStr（C_UTP pStr, C_UTP pPart);
UTP　AjcMbsStrEx(C_UTP pStr, C_UTP pPart, BOOL fIgnoreWidth);

引　数 ： pStr　　- 非検索文字列の先頭アドレス
pPart　　- 検索する部分文字列の先頭アドレス
fIgnoreWidth - 文字列比較方法(FALSE:英大小を区別する，TRUE: 英大小を区別しない)

説　明 ： AjcMbsIStr()は、英字の大文字／小文字を区別しないで、非検索文字列中の部分マルチバイト文字列を検索します。
fIgnoreWidth の指定により、以下の関数を実行します。

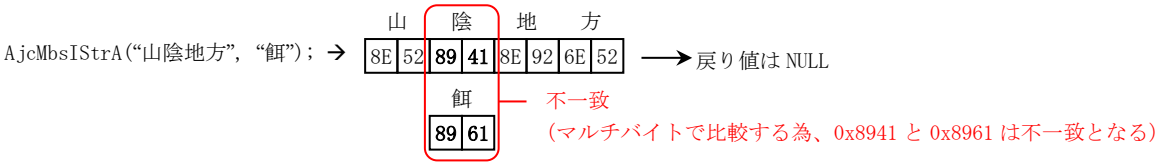
fIgnoreWidth	実行関数	
	バイト文字	ワイド文字(UNICODE)
FALSE（英大小を区別する）	_mbsstr	wcsstr
TRUE　（英大小を区別しない）	AjcMbsIStrA	AjcMbsIStrW

AjcMbsIStrW()は、AjcStrIStrW()と同一処理です

検索する部分文字列の長さが０（つまり、空文字列）を指定した場合は、非検索文字列の先頭アドレスを返します。

戻り値 ： ≠NULL - 非検索文字列中で、最初に見つかった部分文字列のアドレス
＝NULL - 部分文字列は見つからなかった

備　考 ： AjcStrMbsIStrA()は、マルチバイト文字で比較する為、以下ような場合は不一致となります。



41.3.4. 複数の文字列を指定して部分文字列検索（AjcStrFind）

形　式 ： int AjcStrFind (C_UTP pSrc, C_UTP pStr, PAJCSTRFIND pInfo);
int AjcStrIFind (C_UTP pSrc, C_UTP pStr, PAJCSTRFIND pInfo);
int AjcStrFindEx (C_UTP pSrc, C_UTP pStr, C_UTP pDlm, C_UTP pAny, C_UTP pEnc, BOOL fTrim, UI opt, PAJCSTRFIND pInfo);
int AjcStrIFindEx(C_UTP pSrc, C_UTP pStr C_UTP pDlm, C_UTP pAny C_UTP pEnc, BOOL fTrim, UI opt, PAJCSTRFIND pInfo);

引　数 ：
pSrc　　- 非検索文字列へのポインタ
pStr　　- 検索する文字列へのポインタ
pDlm　　- 区切り文字セットへのポインタ（NULL/空文字列：単一文字列検索，未指定時は “;” ）
pAny　　- 任意の0桁以上の文字列とする文字セットへのポインタ（NULL/空文字列：任意の文字列を認識しない，未指定時は”*” ）
pEnc　　- 文字列囲み文字セットへのポインタ（未指定時は NULL）
fTrim　　- 両端の空白除去フラグ（未指定時は”¥”“(ダブルクォート)）
opt　　- オプション（AJCSFO_XXXXX，未指定時は０）
pInfo　　- 文字列が見つかった位置情報を格納するバッファ（不要時は NULL）

説　明 ： pSrc で指定した文字列が、pStr で示す文字列を含むかチェックします。
AjcStrFind()は半角英字の大小を区別して文字列の比較を行います。　　(”ABC” ≠ “abc”)
AjcStrIFind()は半角英字の大小を区別しないで文字列の比較を行います。(”ABC” ＝ “abc”)

検索する文字列を複数指定する場合は、pDlm で分離記号を指定します。
例えば、pDlm=”/;” とした場合、pStr の検索文字列を ’/’ or ‘;’ で区切った複数の文字列を検索します。

検索文字列中に、pAny で指定した文字が存在する場合、当該文字を任意の（０桁以上の）文字列として扱います。
つまり、pAny で指定した文字で分離された文字列群は、文字列の出現順を示すことになります。
例えば、pStr=”BCD*OPQ?VWX”, pAny=”*?”と指定した場合、“BCD”, ”OPQ”, ”VWX” が順に出現することを意味します。
この時、pSrc=”ABCDEFGHIJKLMNOPQRSTUVWXYZ”とした場合、見つかった文字列の長さ(下線部=23)を返します。
検索する文字列（pDlm や pAny で分離された文字列）が空文字列である場合は、すべて「一致」と判断します。
pStr で空文字列(“”)を指定した場合は、「一致」となり０を返します。

pEnc で指定した文字群は、囲み文字として機能し、いかなる文字列も表現できます。
囲み文字は通常開始文字と終端文字が同一(ex = “ABC”)ですが、’<’ 等で始まる場合は終端が ’>’ 等になります。
（囲み終端文字の規則については、AjcStrEnclose()を参照してください）
囲み文字でサンドイッチした文字列は、空白、pDlm で指定した文字や、pAny で指定した文字を含むことができます。
両端の囲み文字は除去されます
囲み文字でサンドイッチした文字列中の「¥」はエスケープ文字となり、「¥」の次の文字を有効とします。
pEnc=”¥”<”; (ダブルクォート(“)と、(<)を囲み文字とした場合、pStr で指定した文字列は以下のように変換されます。

pStr で指定した検索文字列	変換された検索文字列	備考
「< ABC = 123 * 10; >」	「 ABC = 123 * 10; 」	囲み文字列に空白を含む
「This is ¥”Text¥”.」	「This is “Text”.」	「¥」の次の文字が有効
「¥¥100」	「¥100;」	「¥¥」は、「¥」に変換

fTrim=TRUE を指定した場合は、検索する文字列（pDlm や pAny で分離された文字列を含む）の両端の空白を除去します。

opt は、０、あるいは、以下のシンボルの合成値を指定します。(複数の文字列が見つかった場合の動作)

シンボル	指定時	未指定時
AJCSFO_HIGHPOS	高位アドレスの文字列を選択する	低位アドレスの文字列を選択する
AJCSFO_SHORT（同じ位置で見つかった場合の選択）	短い方の文字列を選択する	長い方の文字列を選択する

文字列が見つかった（戻り値≠-1 の）場合、pInfo には以下の情報が設定されます。

メンバ変数	タイプ	内容
ix	int	pDlm で区切られた、何番目の文字列が見つかったかを示す（０：１番目～）
pos	int	見つかった文字列の先頭バイト位置／文字位置（０～）
len	int	見つかった文字列の長さ（バイト数／文字数） - ０の場合、空文字列が見つかったことを意味します

AjcStrFind(), AjcStrIFind()は、pDlm=”;”, pAny=”*”, pEnc=NULL, opt=0 で、AjcStrFindEx(), AjcStrIFindEx()を実行します。

戻り値 ： ≠ -1 - いずれかの文字列が見つかった（見つかった文字列のバイト数／文字数）
＝ -1 - いずれの文字列も見つからない

41.3.5. 文字列内のトークンを検索（AjcStrTok[S]）

形　式 ： UTP　AjcStrTok (UTP pStr, C_UTP pDlm);
UTP　AjcStrTokS(UTP pStr, C_UTP pDlm, UTP *ppContext);

引　数 ： pStr　　- トークンを取得する文字列の先頭アドレス（２つ目以降のトークンを取得する場合は NULL を指定）
pDlm　　- 区切り文字セットへのポインタ
ppContext　- 次のトークン検索ポインタを退避するワークへのポインタ

説　明 ： バイト文字バージョンの場合は、strtok_s()を実行します。
UNICODE バージョンの場合は、wcstok_s()を実行します。

戻り値 ： ≠NULL - 見つかったトークンへのポインタ
＝NULL – トークン無し

備　考 ： 内部で、バイト文字の場合 strtok_s()を、ワイド文字の場合 wcstok_s()を実行します。
AjcStrTok()は、ppContext をスレッドごとに準備するため、スレッドセーフですが、同一スレッドで多重には実行できません。

41.3.6. マルチバイト文字列内のトークンを検索（AjcMbsTok）

形　式 ： UTP　AjcMbsTok(UTP pStr, C_UTP pDlm);
UTP　AjcMbsTokS(UTP pStr, C_UTP pDlm, UTP *ppContext);

引　数 ： pStr　　- トークンを取得する文字列の先頭アドレス（２つ目以降のトークンを取得する場合は NULL を指定）
pDlm　　- 区切り文字セットへのポインタ
ppContext　- 次のトークン検索ポインタを退避するワークへのポインタ

説　明 ： バイト文字バージョンの場合は、_mbstok_s を実行します。
UNICODE バージョンの場合は、wcstok_s()を実行します。

戻り値 ： ≠NULL - 見つかったトークンへのポインタ
＝NULL – トークン無し

備　考 ： 内部で、バイト文字の場合 mbstok_s()を、ワイド文字の場合 wcstok_s()を実行します。
AjcMbsTok()は、ppContext をスレッドごとに準備するため、スレッドセーフですが、同一スレッドで多重には実行できません。

41.3.7. 区切り文字で区切られた文字列内のトークンを取得（AjcStrTok[S]Ex）

形　式 ： UTP AjcStrTokEx (UTP pStr, C_UTP pDlm, C_UTP pEnc);
UTP AjcStrTokSEx(UTP pStr, C_UTP pDlm, C_UTP pEnc, UTP *ppContext);
UTP AjcStrTokREx(UTP pStr, C_UTP pDlm, C_UTP pEnc, UTP *ppContext);

引　数 ： pStr　　- トークンを取得する文字列の先頭アドレス（２つ目以降のトークンを取得する場合はNULLを指定）
pDlm　　- 区切り文字群文字列の先頭アドレス
pEnc　　- 囲み文字群文字列の先頭アドレス（囲み文字が無い場合はNULL）
ppContext - 次のトークン検索ポインタを退避するワークへのポインタ

説　明 ： 区切り文字で区切られた文字列内の次のトークンを取得します。
AjcStrTokEx()は、ppContextをスレッドごとに準備するため、スレッドセーフとなります。
AjcStrTokSEx()は、取得したトークン中の囲み文字を除去しません。
AjcStrTokREx()は、取得したトークン中の囲み文字を除去します。(ex . “<ABC>” -> “ABC”)
囲み文字列中は区切り文字を認識しません。
pEncで囲み文字を指定しますが、囲み文字の先頭と終端が異なる場合は、先頭の囲み文字だけを指定します。
この場合、pEncで指定する先頭文字に対する囲み文字の終端は、以下の文字となります。

先頭	終端	先頭	終端	先頭	終端	先頭	終端
<	>	(	)	[	]	{	}

UNICODE（日本語）バージョン(AjcStrTokExW())の場合は、さらに以下の全角文字が指定可能です。

先頭	終端	先頭	終端	先頭	終端	先頭	終端	先頭	終端
＜	＞	（	）	［	］	｛	｝	「	」
〔	〕	《	》	『	』	【	】		

上記以外の囲み文字を指定した場合は、囲み文字の先頭と終端が同一文字となります。
例えば、pEnc=”¥”[“（ダブルクォートと左大括弧）を指定した場合、「”ABC;DEF”」や「[123, 456]」は囲み文字として認識され、この中にある区切り文字は認識されません。

ex.

```
UT txt[] = TEXT(“ABC;DEF;¥”123;456¥”;[X;Y;Z]”);
C_UTP pDlm = TEXT(“;”);   // 区切文字＝ ;
C_UTP pEnc = TEXT(“¥”[“); // 囲み文字＝ ” と [
AjcStrTokEx(txt , pDlm, pEnc);
```
→ “ABC”

```
AjcStrTokEx(NULL,  pDlm, pEnc);
```
→ “DEF”

```
AjcStrTokEx(NULL,  pDlm, pEnc);
```
→ “¥”123;456¥”” (囲み文字列中の区切り文字(;)は認識しない)

```
AjcStrTokEx(NULL,  pDlm, pEnc);
```
→ “[X;Y;Z]” (囲み文字列中の区切り文字(;)は認識しない)

```
AjcStrTokEx(NULL,  pDlm, pEnc);
```
→ NULL

バイト文字バージョン(AjcStrTok[S]ExA())の場合、pEncで指定する囲み文字に全角文字（２バイト文字）は指定できません。
ワイド文字バージョン(AjcStrTok[S]ExW())の場合、pEncで指定する囲み文字にサロゲートペア文字は指定できません。

トークンを検出する文字列(pStr)中の区切り記号の位置に文字列終端(0x0)を書き込み、トークン文字列（戻り値）はトークンを検出する文字列中へのポインタとなります。

戻り値 ： ≠NULL - 見つかったトークン文字列のアドレス（トークンを検出する文字列中へのポインタ）
＝NULL - トークンなし（文字列終端を検出）

備　考 ： バイト文字バージョン(AjcStrTokExA())の場合は、マルチバイトを認識します。
したがって、pStrで指定する文字列中の全角文字２バイト目が終端文字として認識されることはありません。

ex.

```
BC txt[] = “[江刺;藤原];123-1”;
C_BCP pDlm = “;”;  // 区切文字＝  ;
C_BCP pEnc = “[“;  // 囲み文字＝  [・・]
AjcStrTokExA(txt, pDlm, pEnc);
```
→ “江刺;藤原” （この後、”123-1”, NULL と続く）

txt	[	江		刺	;	藤		原	]	;	1	2	3	-	1	¥0
(16進)	5B	8D	5D	8E	68	3B	93	A1	8C	B4	5D	3B	31	32	33	2D	31	00

囲み文字終端(']')と同じだが、全角２バイト目なので認識しない
囲み文字終端(']')

41.3.8. 文字列を指定文字で囲む (AjcStrEnclose)

形　式 ： UI　　AjcStrEnclose(C_UTP pStr, UTP pBuf, UI lBuf, UI c)

引　数 ： pStr　- ソース文字列のアドレス
pBuf　- 囲み文字で囲んだ文字列を格納するバッファのアドレス（囲み文字付加後の文字長・算出だけの場合はNULL）
lBuf　- 囲み文字で囲んだ文字列を格納するバッファの文字数
c　　- 文字列を囲む文字（括弧等の場合は、左側の文字を指定(右側自動生成)。'¥0' / '¥¥'は指定不可）

説　明 ： pStr で示されるソース文字列を、「c」で指定される文字で囲んだ文字列を作成し、pBuf/lBuf で示されるバッファに格納します。
「ｃ」に以下の文字を指定した場合は、右端側の文字は以下のようになります。

c	右端文字	c	右端文字	c	右端文字	c	右端文字
<	>	(	)	[	]	{	}

尚、UNICODE バージョンの場合は、さらに以下の全角文字が追加されます。

c	右端文字	c	右端文字	c	右端文字	c	右端文字	c	右端文字
＜	＞	(	)	[	]	{	}	「	」
〔	〕	《	》	『	』	【	】		

「ｃ」が上記文字以外である場合は、右端文字は「c」と同じになります。

ソース文字列の中に、「¥」あるいは、右端文字と同じ文字が存在する場合は、その文字の直前に「¥」を挿入します。

```
ex. pSrc=「ABC」 , c=「″」    ---> 「″ABC″」
    pSrc=「ABC¥」, c=「[」    ---> 「[ABC¥¥]」
    pSrc=「A>C」 , c=「<」    ---> 「<A¥>C>」
```

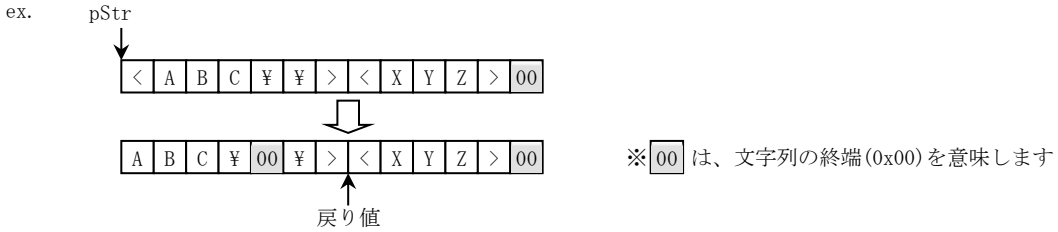
戻り値 ： ≠０ - 成功（囲み文字を付加した文字列の文字長）
＝０ - 失敗

41.3.9. 文字列の囲みを取り除く(AjcStrStripEnclose)

形　式 ： UTP　AjcStrStripEnclose (UTP pStr)

引　数 ： pStr　- 囲み文字を取り除く文字列のアドレス

説　明 ： AjcStrEnclose()で囲み文字を付加された文字列を、元の文字列に戻します。
復元した文字列の終端には、文字列の終端(0x00)が付加されます。
尚、右端文字が見つからない場合は、文字列の終端(0x00)まで処理します。



戻り値 ： ≠NULL - 成功（囲み文字列の次の文字のアドレス（右端文字の次の文字のアドレス））
＝NULL - 失敗

41.3.10. 書式文字列の生成 (AjcSn[V]PrintF)

形　式 ： BOOL　AjcSnPrintF (UTP pBuf, UI lBuf, C_UTP pFmt, ...);
BOOL　AjcVSnPrintF(UTP pBuf, UI lBuf, C_UTP pFmt, va_list vls)

引　数 ： pBuf　- 文字列を生成するバッファのアドレス
lBuf　- 文字列を生成するバッファの文字数
pFmt　- 書式文字列 (C ライブラリの printf()と同一仕様)
vls　- 可変パラメタへのポインタ

説　明 ： Ｃライブラリ関数（_snprintf / _vsnprintf）と同様に、書式に従った文字列を生成します。
但し、バッファがオーバーする場合でも、文字列の終端(0x00)を確実に設定します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

41.3.11. 書式文字列生成と１０進数文字列の整数部で特定桁数毎に区切り文字を挿入する (AjcSnPrtSep)

形　式 ： BOOL　AjcSnPrtSep (UTP pBuf, UI lBuf, C_UTP pFmt, ...);

引　数 ： pBuf　- 文字列を生成するバッファのアドレス
lBuf　- 文字列を生成するバッファの文字数
pFmt　- 書式文字列 (C ライブラリの printf()と同一仕様)

説　明 ： Ｃライブラリ関数（_snprintf）と同様に、書式に従った文字列を生成し、生成した文字列の先頭部分が１０進整数表現の文字列（先頭部分に空白と符号を許容）である場合、特定の桁数毎に区切り記号（３桁毎のカンマ（,）等）を挿入します。
区切り文字の挿入は、１０進整数表現に許容されない文字を見つけた時点で終了します。
区切り桁数と区切り文字は、ロケールの情報によります。
この関数は、バッファがオーバーする場合でも、文字列の終端(0x00)を確実に設定します。

ex.　AjcSnPrtSep(buf, sizeof buf, "%.4f (ABC)", -12345678.0034);　---> buf="-12,345,678.0034 (ABC)"

戻り値 ： TRUE　- 成功
FALSE - 失敗

41.3.12. １０進数文字列の整数部から区切り文字を削除する (AjcStrRmvSepChar)

形　式 ： VO　AjcStrRmvSepChar (UTP pStr);

引　数 ： pStr　- 区切り文字を削除する１０進数表現文字列のアドレス

説　明 ： 指定された１０進整数表現の文字列（先頭部分に空白と符号を許容）から区切り文字（３桁毎のカンマ（,）等）を削除します。
区切り文字の削除は、１０進整数表現に許容されない文字を見つけた時点で終了します。
区切り文字は、ロケールの情報によります。

ex.　strcpy(buf, " -12,345,678 (A,B,C)");
AjcStrRmvSepChar(buf);　---> buf=" -12345678 (A,B,C)"

戻り値 ： なし

41.3.13. 引数情報を指定した書式文字列の生成 (AjcVSnPrintF)

形　式 ： BOOL　AjcVSnPrintF (UTP pBuf, UI lBuf, C_UTP pFmt, va_list vls);
BOOL　MAjcVSnPrintF(UTP pBuf, UI lBuf, C_UTP pFmt); --- マクロ

引　数 ： pBuf　- 文字列を生成するバッファのアドレス
lBuf　- 文字列を生成するバッファの文字数
pFmt　- 書式文字列（C ライブラリの printf()と同一仕様）
vls　- 書式文字列以降の引数情報（vs_start()により設定した情報）

説　明 ： Ｃライブラリ関数（_snvprintf）と同様に、書式に従った文字列を生成します。
但し、バッファがオーバーする場合でも、文字列の終端(0x00)を確実に設定します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

備　考 ： このＡＰＩは、呼び出し元の関数に可変個引数がある場合に使用します。

```
VO SubSPrintF(UTP pBuf, UI lBuf, C_BCP pFmt, ...)
{
    va_list vls;

    va_start(vls, pFmt);
    AjcVSnPrintF(pBuf, lBuf, pFmt, vls);
    va_end(vls);
}

VO MyApp( ・・・ )
{
   int   var1 = 123, var2 = 456;;
   BC    buf[100];
    ・・・
    SubSPrintF(buf, sizeof buf, “%d - %d”, var1, var2);
    ・・・
}
```

MAjcVSnPrintF()マクロを使用した場合「va_list」に関する操作を省略できます。
MAjcVSnPrintF()マクロを使用した場合、以下のように記述できます。

```
VO SubSPrintF(UTP pBuf, UI lBuf, C_BCP pFmt, ...)
{
    MAjcVSnPrintF(pBuf, lBuf, pFmt);
}

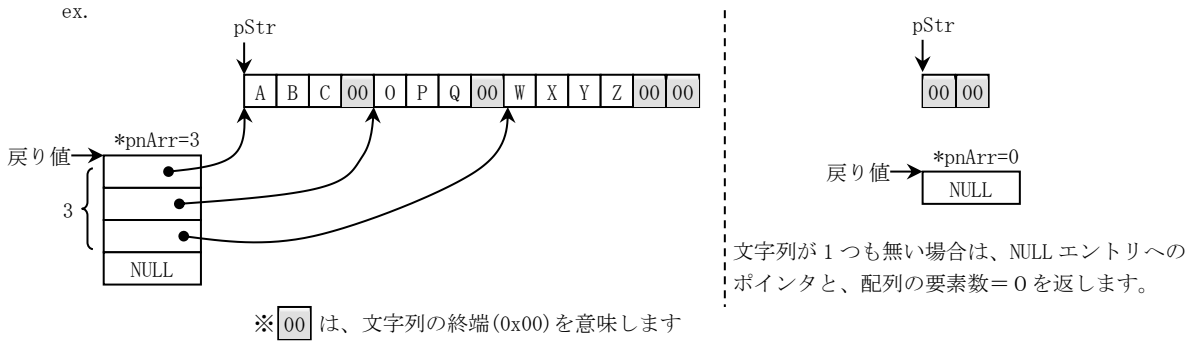
VO MyApp( ・・・ )
{
   int   var1 = 123, var2 = 456;;
   BC    buf[100];
    ・・・
    SubSPrintF(buf, sizeof buf, “%d - %d”, var1, var2);
    ・・・
}
```

41.3.14. 多重文字列→文字列ポインタ配列生成 (AjcMStrMakeArray)

形　式 ： UTP * AjcMStrMakeArray(UTP pStr, UIP pnArr)

引　数 ： pStr　- 多重文字列のアドレス
pnArr - 配列の要素数（文字列の個数）を格納するバッファのアドレス（不要時は NULL）

説　明 ： pStr で指定される多重文字列の各文字列へのポインタ配列を生成します。
生成したポインタ配列は、AjcMStrReleaseArray()で開放します。



文字列が 1 つも無い場合は、NULL エントリへのポインタと、配列の要素数＝０を返します。

※ 00 は、文字列の終端(0x00)を意味します

戻り値 ： ≠NULL - 成功（生成したポインタ配列のアドレス）
＝NULL - 失敗

41.3.15. 文字列ポインタ配列開放(AjcMStrReleaseArray)

形　式 ： VO　AjcMStrReleaseArray (VOP pArr)

引　数 ： pArr　- 開放するポインタ配列のアドレス

説　明 ： AjcMStrMakeArray()で生成した、ポインタ配列を開放します。

戻り値 ： なし

41.3.16. 多重文字列中の文字列の個数取得(AjcMStrCount)

形　式 ： UI　AjcMStrCount(C_UTP pStr)

引　数 ： pStr　- 多重文字列のアドレス

説　明 ： 多重文字列に含まれる文字列の個数を返します。

戻り値 ： 文字列の個数

41.3.17. 多重文字列の総文字数取得(AjcMStrTotalSize)

形　式 ： UI　AjcMStrTotalSize (C_UTP pStr)

引　数 ： pStr　- 多重文字列のアドレス

説　明 ： 多重文字列の総文字数を取得します。

戻り値 ： 総文字数（文字列終端のヌル文字と多重文字列終端のヌル文字を含む）

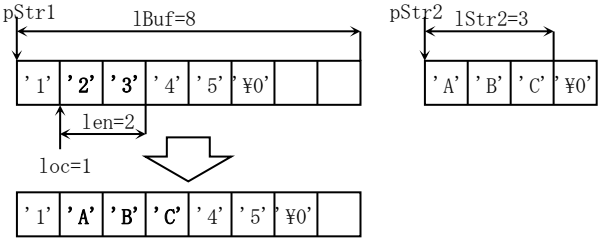
41.3.18. 部分文字列置換（AjcRepPartStr）

形　式 ： UTP　AjcRepPartStr (UTP pStr1, UI lBuf, UI loc, UI len, C_UTP pStr2, UI lStr2);

引　数 ： pStr1 - 文字列１のアドレス
lBuf　- 文字列１のバッファサイズ（文字列の終端（0x00）を含むバイト数／文字数）
loc　- 文字列１中の部分文字列の文字位置（０～）
len　- 文字列１中の部分文字列の文字数（０～）
pStr2 - 文字列２のアドレス
lStr2 - 文字列２の文字数（－１を指定した場合は、本関数内で文字列２の長さを算出する）

説　明 ： 文字列１中の loc, len で示される部分文字列を、文字列２で置換します。
置換後の文字列１の長さは、置換した文字列の長さに応じて増減します。
len=0 を指定した場合は、loc で指定した位置へ、文字列２を挿入することになります。

（例）



文字列の置換が成功した場合は、文字列１のアドレス(=pStr1)を返します。
置換後の文字列の長さが、lBuf で示されるバッファサイズを超える場合、文字列の置換は行われず、NULL を返します。

戻り値 ： ≠NULL - 成功（文字列１の先頭アドレス）
＝NULL - 失敗

41.3.19. 文字列の前部空白／不要文字群を除去(AjcStrLTrim[Ex])

形　式 ： UTP　AjcStrLTrim　(UTP pStr, UTP pBuf, UI lBuf);
UTP　AjcStrLTrimEx(UTP pStr, UTP pBuf, UI lBuf, C_UTP pChars);

引　数 ： pStr　- 不要文字群を除去する文字列のアドレス
pBuf　- 不要文字群を除去した文字列を格納するバッファのアドレス
lBuf　- 不要文字群を除去した文字列を格納するバッファのバイト数／文字数
pChars – 除去する不要文字群のアドレス（文字列で指定）

説　明 ： pStr で指定した文字列の前部分の不要文字群を除去した文字列を pBuf で指定したバッファに格納します。
pStr と pBuf は重複した（あるいは、同一の）領域を指定できます。
pChars は、除去する不要文字群を文字列で指定します。
AjcStrLTrim()は、不要文字を空白（半角と全角）に固定したバージョンです。

ex.　AjcStrLTrim　(TEXT("　東京都　"), pBuf, lBuf);　→　"東京都　"
　　AjcStrLTrimEx(TEXT("*★東京都★*"), pBuf, lBuf, TEXT("*★"));　→　"東京都★*"

pBuf, lBuf で指定したバッファには必ず文字列の終端(¥0)が付加されます。
バッファサイズが除去後の文字列より小さい場合は、文字列の後部がカットされます。

戻り値 ： 空白を除去した文字列を格納したバッファのアドレス（=pBuf）

41.3.20. 文字列の後部空白／不要文字群を除去(AjcStrRTrim[Ex])

形　式 ： UTP　AjcStrRTrim　(UTP pStr, UTP pBuf, UI lBuf);
UTP　AjcStrRTrimEx(UTP pStr, UTP pBuf, UI lBuf, C_UTP pChars);

引　数 ： pStr　- 不要文字群を除去する文字列のアドレス
pBuf　- 不要文字群を除去した文字列を格納するバッファアのアドレス
lBuf　- 不要文字群を除去した文字列を格納するバッファアのバイト数／文字数
pChars – 除去する不要文字群のアドレス（文字列で指定）

説　明 ： pStr で指定した文字列の後ろ分部の不要文字群を除去した文字列を pBuf で指定したバッファに格納します。
pStr と pBuf は重複した（あるいは、同一の）領域を指定できます。
pChars は、除去する不要文字群を文字列で指定します。
AjcStrRTrim()は、不要文字を空白（半角と全角）に固定したバージョンです。

ex.　AjcStrRTrim　(TEXT("　東京都　"), pBuf, lBuf);　→　"　東京都"
AjcStrRTrimEx(TEXT("*★東京都★*"), pBuf, lBuf, TEXT("*★"));　→　"*★東京都"

pBuf, lBuf で指定したバッファには必ず文字列の終端(¥0)が付加されます。
バッファサイズが除去後の文字列より小さい場合は、文字列の後部がカットされます。

戻り値 ： 空白を除去した文字列を格納したバッファのアドレス（=pBuf）

41.3.21. 文字列の両端の空白／不要文字群を除去(AjcStrTrim[Ex])

形　式 ： UTP　AjcStrTrim　(UTP pStr, UTP pBuf, UI lBuf);
UTP　AjcStrTrimEx(UTP pStr, UTP pBuf, UI lBuf, C_UTP pChars);

引　数 ： pStr　- 空白を除去する文字列のアドレス
pBuf　- 空白を除去した文字列を格納するバッファアのアドレス（pStr と重複可）
lBuf　- 空白を除去した文字列を格納するバッファアのバイト数／文字数
pChars – 除去する不要文字群のアドレス（文字列で指定）

説　明 ： pStr で指定した文字列の両端の空白（半角と全角）を除去した文字列を pBuf で指定したバッファに格納します。
pStr と pBuf は重複した（あるいは、同一の）領域を指定できます。
pChars は、除去する不要文字群を文字列で指定します。
AjcStrTrim()は、不要文字を空白（半角と全角）に固定したバージョンです。

ex.　AjcStrTrim　(TEXT("　東京都　"), pBuf, lBuf);　→　"東京都"
AjcStrTrimEx(TEXT("*★東京都★*"), pBuf, lBuf, TEXT("*★"));　→　"東京都"

pBuf, lBuf で指定したバッファには必ず文字列の終端(¥0)が付加されます。
バッファサイズが除去後の文字列より小さい場合は、文字列の後部がカットされます。

戻り値 ： 空白を除去した文字列を格納したバッファのアドレス（=pBuf）

41.3.22. 文字列中の ESC シーケンスを除去(AjcRemoveEscInStr)

形　式 ： UTP　AjcRemoveEscInStr　(UTP pStr, UTP pBuf, UI lBuf);

引　数 ： pStr　- ESC シーケンスを除去する文字列のアドレス
pBuf　- ESC シーケンスを除去した文字列を格納するバッファアのアドレス
lBuf　- ESC シーケンスを除去した文字列を格納するバッファアのバイト数／文字数

説　明 ： pStr で指定した文字列中 ESC シーケンス（'¥x1B'～英字）を除去した文字列を pBuf で指定したバッファに格納します。
pStr と pBuf は重複した（あるいは、同一の）領域を指定できます。

ex.　AjcStrTrim　(TEXT("¥x1B[31mThis is string.¥x1B0m¥n"), pBuf, lBuf);　→　"This is string.¥n"

戻り値 ： ESC シーケンスを除去した文字列を格納したバッファのアドレス（=pBuf）

41.3.23. 文字列の整形(AjcStrShaping)

形　式 ： UTP　AjcStrShaping (BCP pStr, BCP pBuf, UI lBuf, C_BCP pChars, C_BCP pRepStr);

引　数 ： pStr　　- 整形前文字列のアドレス
pBuf　　- 整形した文字列を格納するバッファのアドレス
lBuf　　- バッファのバイト数／文字数
pChars　- 文字群（文字列で指定）
pRepStr - 置き換える文字列

説　明 ： pStr で指定した文字列の両端の文字群（pChars で指定した複数文字）を除去し、中間の連続した文字群を pRepStr で指定した文字列に置き換えます。
pStr と pBuf は同一の領域を指定できます。
pChars は、両端の除去する文字群、あるいは、中間の置換する連続した（複数の）文字を文字列で指定します。

ex.　　BC　str[256] = “,,_,ABC_,_DEF_,_,_,,,XYZ,_,,”;
AjcStrShaping(str, str, 256, ″,_″, ″,″);”

⇩

buf[256] = “ABC,DEF,XYZ”

戻り値 ： 整形した文字列を格納したバッファのアドレス（=pBuf）

41.3.24. マルチバイト（シフトＪＩＳ）文字列情報の取得(AjcStrChkMbcPart)

形　式 ： BOOL　AjcStrChkMbcPart (C_BCP pTop, UI ix, PAJCMBCINFO pBuf);

引　数 ： pTop　　- マルチバイト(シフトＪＩＳ)文字列のアドレス
ix　　- マルチバイト情報をチェックする、文字列中のバイト位置（０～）
pBuf　　- マルチバイト情報を格納するバッファのアドレス

説　明 ： pTop で指定した文字列中の、ix で指定したインデクスのバイト位置に関する情報を、pBuf に格納します。
pTop で示す文字列の先頭は、１バイト文字か、複数バイト文字の１バイト目でなければなりません。
pBuf で示されるバッファには、以下の情報が設定されます。

pBuf->len ： 「ix」の位置のバイトが、ｎバイト文字の構成要素であることを示す
pBuf->ix　： ｎバイト文字中のインデクス（０ ～ ｎ－１）
例えば、「pBuf->len=2」，「pBuf->ix=0」である場合、２バイト文字の１バイト目を意味する

戻り値 ： TRUE　- 成功
FALSE - 失敗

41.3.25. ＵＴＦ－８ 文字列情報の取得(AjcStrChkUtf8Part)

形　式 ： BOOL　AjcStrChkUtf8Part (C_BCP pTop, UI ix, PAJCMBCINFO pBuf);

引　数 ： pTop　　- ＵＴＦ－８文字列のアドレス
ix　　- マルチバイト情報をチェックする、文字列中のバイト位置（０～）
pBuf　　- マルチバイト情報を格納するバッファのアドレス

説　明 ： pTop で指定した文字列中の、ix で指定したインデクス位置のバイトに関する情報を、pBuf に格納します。
pTop で示す文字列の先頭は、１バイト文字か、複数バイト文字の１バイト目でなければなりません。
pBuf で示されるバッファには、以下の情報が設定されます。

pBuf->len ： 「ix」の位置のバイトが、ｎバイト文字の構成要素であることを示す
pBuf->ix　： ｎバイト文字中のインデクス（０ ～ ｎ－１）
例えば、「pBuf->len=3」，「pBuf->ix=1」である場合、３バイト文字の２バイト目を意味する

戻り値 ： TRUE　- 成功
FALSE - 失敗

41.3.26. 日本語ＥＵＣ文字列情報の取得(AjcStrChkEucPart)

形　式 ： BOOL　AjcStrChkEucPart（C_BCP pTop, UI ix, PAJCMBCINFO pBuf);

引　数 ： pTop　　－ 日本語ＥＵＣ文字列のアドレス
ix　　－ マルチバイト情報をチェックする、文字列中のバイト位置（０～）
pBuf　　－ マルチバイト情報を格納するバッファのアドレス

説　明 ： pTop で指定した文字列中の、ix で指定したインデクスのバイト位置に関する情報を、pBuf に格納します。
pTop で示す文字列の先頭は、１バイト文字か、複数バイト文字の１バイト目でなければなりません。
pBuf で示されるバッファには、以下の情報が設定されます。

pBuf->len ：「ix」の位置のバイトが、ｎバイト文字の構成要素であることを示す
pBuf->ix　：ｎバイト文字中のインデクス（０ ～ ｎ－１）
例えば、「pBuf->len=2」，「pBuf->ix=0」である場合、２バイト文字の１バイト目を意味する

戻り値 ： TRUE　－ ＯＫ
FALSE – エラー

41.3.27. ＵＴＦ８→ マルチバイト（シフトＪＩＳ）文字列変換(AjcUtf8ToMbc[Ex])

形　式 ： int　AjcUtf8ToMbc　(C_UBP pUtf8,　　UBP pMbc, UI lMbc);
int　AjcUtf8ToMbcEx(C_UBP pUtf8, UI lUtf8, UBP pMbc, UI lMbc);

引　数 ： pUtf8 – UTF-8 文字列のアドレス
lUtf8 – UTF-8 文字列のバイト数（-1：終端文字(0x00)を含めた文字列長）
pMbc　– 変換したマルチバイト(S-ＪＩＳ）文字列を格納するバッファアのアドレス（バッファサイズの算出だけの場合はNULL）
lMbc　– 変換したマルチバイト(S-ＪＩＳ）文字列を格納するバッファアのバイト数

説　明 ： pUtf8, lUtf8 で指定したUTF-8 文字列をマルチバイト(シフトＪＩＳ)文字列に変換し、pMbc で指定したバッファに格納します。
pMbc=NULL とした場合は、変換に必要なバッファサイズ（バイト数）を返します。
lUtf8=-1 とした場合は、pUtf8 で示される文字列長（文字列終端(0x00)を含む）を仮定します。
lMbc が変換に必要なサイズよりも大きい場合は、文字列終端(0x00)が付加されます。
lMbc が変換に必要なサイズに満たない場合は、変換を中止します。
AjcUtf8ToMbc()は、lUtf8=-1 として、AjcUtf8ToMbcEx()を実行します。

戻り値 ： 変換に必要なバッファサイズ（lUtf8=-1 の場合は、文字列終端(0x00)を含むバイト数）

41.3.28. マルチバイト（シフトＪＩＳ） →ＵＴＦ８文字列変換(AjcMbcToUtf8[Ex])

形　式 ： int　AjcMbcToUtf8　(C_UBP pMbc,　　UBP pUtf8, UI lUtf8);
int　AjcMbcToUtf8Ex(C_UBP pMbc, UI lMbc, UBP pUtf8, UI lUtf8);

引　数 ： pMbc　– マルチバイト(シフトＪＩＳ)文字列のアドレス
lMbc　– マルチバイト(シフトＪＩＳ)文字列のバイト数（-1：終端文字(0x00)を含めた文字列長）
pUtf8　– 変換したUTF-8 文字列を格納するバッファアのアドレス（バッファサイズの算出だけの場合はNULL）
lUtf8　– 変換したUTF-8 文字列を格納するバッファアのバイト数

説　明 ： pMbc, lMbc で指定したマルチバイト(シフトＪＩＳ)文字列をUTF-8 文字列に変換し、pUtf8 で指定したバッファに格納します。
pUtf8=NULL とした場合は、変換に必要なバッファサイズ（バイト数）を返します。
lMbc=-1 とした場合は、pMbc で示される文字列長（文字列終端(0x00)を含む）を仮定します。
lUtf8 が変換に必要なサイズよりも大きい場合は、文字列終端(0x00)が付加されます。
lUtf8 が変換に必要なサイズに満たない場合は、変換を中止します。
AjcMbcToUtf8()は、lMbc=-1 として、AjcMbcToUtf8Ex()を実行します。

戻り値 ： 変換に必要なバッファサイズ（lMbc=-1 の場合は、文字列終端(0x00)を含むバイト数

41.3.29. 日本語ＥＵＣ → シフトＪＩＳ文字列変換(AjcEucToSJis[Ex])

形　式 ： int AjcEucToSJis (C_UBP pEuc, UBP pSJis, UI lSJis);
int AjcEucToSJisEx(C_UBP pEuc, UI lEuc, UBP pSJis, UI lSJis);

引　数 ： pEuc - 日本語ＥＵＣ文字列のアドレス
lEuc - 日本語ＥＵＣ文字列のバイト数（-1：終端文字(0x00)を含めた文字列長）
pSJis - 変換したシフトＪＩＳ文字列を格納するバッファのアドレス（バッファサイズの算出だけの場合は NULL）
lSJis - 変換したシフトＪＩＳ文字列を格納するバッファのバイト数

説　明 ： pEuc, lEuc で指定した日本語ＥＵＣ文字列をシフトＪＩＳ文字列に変換し、pSJis で指定したバッファに格納します。
pSJis=NULL とした場合は、変換に必要なバッファサイズ（バイト数）を返します。
lEuc=-1 とした場合は、pEuc で示される文字列長（文字列終端(0x00)を含む）を仮定します。
lSJis が変換に必要なサイズよりも大きい場合は、文字列終端(0x00)が付加されます。
lSJis が変換に必要なサイズに満たない場合や、文字列終端を検出しな場合は変換を中止します。
変換不能な文字は、'?' を出力します。
AjcEucToSJis()は、lEuc=-1 として、AjcEucToSJisEx()を実行します。

戻り値 ： 変換に必要なバッファサイズ（lEuc=-1 の場合は、文字列終端(0x00)を含むバイト数）

41.3.30. シフトＪＩＳ → 日本語ＥＵＣ文字列変換(AjcSJisToEuc[Ex])

形　式 ： int AjcSJisToEuc (C_UBP pSJis, UBP pEuc , UI lEuc);
int AjcSJisToEucEx(C_UBP pSJis, UI lSJis, UBP pEuc , UI lEuc);

引　数 ： pSJis - シフトＪＩＳ文字列のアドレス
lSJis - シフトＪＩＳ文字列のバイト数（-1：終端文字(0x00)を含めた文字列長）
pEuc - 変換した日本語ＥＵＣ文字列を格納するバッファのアドレス（バッファサイズの算出だけの場合は NULL）
lEuc - 変換した日本語ＥＵＣ文字列を格納するバッファのバイト数

説　明 ： pSJis, lSJis で指定したシフトＪＩＳ文字列を日本語ＥＵＣ文字列に変換し、pEuc で指定したバッファに格納します。
pEuc=NULL とした場合は、変換に必要なバッファサイズ（バイト数）を返します。
lSJis=-1 とした場合は、pSjis で示される文字列長（文字列終端(0x00)を含む）を仮定します。
lEuc が変換に必要なサイズよりも大きい場合は、文字列終端(0x00)が付加されます。
lEuc が変換に必要なサイズに満たない場合はや、文字列終端を検出しな場合は変換を中止します。
変換不能な文字は、'?' を出力します。
AjcSJisToEuc()は、lSJis=-1 として、AjcSjisToEucEx()を実行します。

戻り値 ： 変換に必要なバッファサイズ（lSJis=-1 の場合は、文字列終端(0x00)を含むバイト数）

41.3.31. 文字コード判別（シフトＪＩＳ／日本語ＥＵＣ／ＵＴＦ-８）(AjcStrChkCode)

形　式 ： AJCMBCKIND AjcStrChkCode (C_BCP pStr);

引　数 ： pStr - 文字コードを判別する文字列のアドレス

説　明 ： 文字コードの種別を判別します。

戻り値 ： AJCMBC_SJIS ： シフトＪＩＳ
AJCMBC_EUC ： 日本語ＥＵＣ
AJCMBC_UTF8 ： ＵＴＦ-８
-1 ： 判別不能

41.3.32. 文字コード判別（シフトＪＩＳ／日本語ＥＵＣ／ＵＴＦ-８／ＵＴＦ－１６）(AjcStrChkCodeEx)

形　式 ： EAJCTEC　AjcStrChkCodeEx (C_BCP pStr, UI lStr);

引　数 ： pStr　- 文字コードを判別する文字列のアドレス
lStr　- 文字コードを判別する文字列のバイト数（８以上）

説　明 ： 文字コードの種別を判別します。
文字列のバイト数は８以上でなければなりません。

戻り値 ： AJCTEC_MBC　　：マルチバイト文字（シフトＪＩＳ）／ 判別不能
AJCTEC_EUC_J　：日本語ＥＵＣ
AJCTEC_UTF_8　：ＵＴＦ-８
AJCTEC_UTF_16LE：ＵＴＦ-１６（リトルエンディアン）
AJCTEC_UTF_16BE：ＵＴＦ-１６（ビッグエンディアン）
-1　　：判別不能

注　意 ： lStr は文字列のバイト数を指定しますが、本ＡＰＩはバイト文字，ワイド文字兼用なので文字列のバイト数を「 strlen() 」や、「 wcslen() * 2 」で求めることはできません。
lStr には、(ファイルから読み出したバイト数等の) 文字列のバイナリのバイト数を指定してください。

41.3.33. 文字の長さ（バイト数／ワード数）取得(AjcStrChk{Mbc/Utf8/Euc/U16Le/U16Be}Len)

形　式 ： UI　AjcStrChkMbcLen　(C_BCP pTop, UI len);　--- Ｓ－ＪＩＳ文字の長さ
UI　AjcStrChkUtf8Len (C_BCP pTop, UI len);　--- ＵＴＦ－８文字の長さ
UI　AjcStrChkEucLen　(C_BCP pTop, UI len);　--- ＥＵＣ　　文字の長さ
UI　AjcStrChkU16LeLen(C_WCP pTop, UI len);　--- UNICODE(LittleEndian) 文字の長さ
UI　AjcStrChkU16BeLen(C_WCP pTop, UI len);　--- UNICODE(BigEndian　) 文字の長さ

引　数 ： pTop　- 文字の先頭アドレス
len　- 文字列の長さ（バイト数／ワード数，pTop から有効な文字列終端までの長さ，-1 指定時は自動算出）

説　明 ： pTop で指定した位置から始まる各文字コードでの１文字の長さ（バイト数／ワード数）を取得します。
AjcStrChkMbcLen()は、全角文字は２を、半角文字は１を、文字コードが不正な場合は０を返します。
AjcStrChkU16LeLen(),AjcStrChkU16BeLen()はサロゲートペアの場合２を、それ以外は１を返します。
AjcStrChkUtf8Len()は、１～６を、AjcStrChkEucLen()は１～３を返します。

戻り値 ： １～６：文字のバイト数／ワード数　　(AjcStrChkMbcLen / AjcStrChkU16LeLen / AjcStrChkU16BeLen ：１～２)
(AjcStrChkUtf8Len　　：１～６)
(AjcStrChkEucLen　　：１～３)
０　　：判定不可（len が 1 文字分のバイト数／ワード数より小さい、あるいは、不正な文字コード）

41.3.34. 文字列の長さ補正(AjcStrAdjustLen)

形　式 ： UI　AjcStrAdjustLenA(C_UTP pTxt, UI len)

引　数 ： pTxt　- 文字の先頭アドレス
len　- 文字列の長さ（バイト数／文字数）

説　明 ： len = -1 を指定した場合は、pTxt の文字列長を返します。(strlen(pTxt) / wcslen(pTxt))
len が-1 以外の場合は、文字列長を補正します。 (strnlen(pTxt, len) / wcsnlen(pTxt, len))
len が-1 以外の場合、len で指定された長さだけ文字列を検索し、途中に終端文字（０）がある場合は、終端文字の直前までの長さを返します。途中に終端文字（０）が無い場合は、len をそのまま返します。
pTxt = NULL を指定した場合は、０を返します。

戻り値 ： 調整された文字列長（バイト数／文字数）

41.3.35. パスとファイル名を反対にした文字列を作成する（AjcStrCnvRevPath）

形　式 ： BOOL　AjcStrCnvRevPath(C_UTP pPath, UTP pBuf, int lBuf);

引　数 ： pPath　- パス名文字列のアドレス
pBuf　- 反対にしたしたパス名文字列を格納するバッファのアドレス
lBuf　- 反対にしたしたパス名文字列を格納するバッファの文字数

説　明 ： パス名のパス部分（ドライブとディレクトリ）とファイル名部分を反対にし、セミコロン(;)で区切った文字列を作成します。

（例）　「d:¥Subdir¥MyFile.txt」 → 「MyFile.txt;d:¥SubDir¥」

入力パス名（pPath）と出力バッファ（pBuf）は同一領域でもＯＫです。
返還後の文字列が、pBuf, lBuf で指定したバッファに収まらない場合は、末尾部分はカットされます。
バッファには、文字列の終端(0x00)が必ず設定されます。

戻り値 ： TRUE　- 成功
FALSE – 失敗

41.3.36. パスとファイル名を反対にしたパス名を元に戻す（AjcStrRetRevPath）

形　式 ： BOOL　AjcStrRetRevPath(C_UTP pRev, UTP pBuf, int lBuf)

引　数 ： pPath　- パス部分とファイル名部分が反対になったパス名文字列のアドレス
pBuf　- 元に戻したパス名文字列を格納するバッファのアドレス
lBuf　- 元に戻したパス名文字列を格納するバッファの文字数

説　明 ： AjcStrCnvRevPath()で作成した、パス部分とファイル名部分が反対になったパス名を元のパス名に戻します。

（例）　「MyFile.txt;d:¥SubDir¥」 → 「d:¥Subdir¥MyFile.txt」

入力文字列（pRev）と出力バッファ（pBuf）は同一領域でもＯＫです。
返還後の文字列が、pBuf, lBuf で指定したバッファに収まらない場合は、末尾部分はカットされます。
バッファには、文字列の終端(0x00)が必ず設定されます。

戻り値 ： TRUE　- 成功
FALSE – 失敗

41.3.37. 文字列が全て同一文字で構成されているかチェック（AjcStrRetRevPath）

形　式 ： BOOL　AjcStrSameAsAllChar(C_UTP pStr,UT c);

引　数 ： pStr　- 同一文字で構成されているかチェックする文字列
c　- チェックする文字コード

説　明 ： 文字列(pStr)が全て同一文字(c)で構成されているかチェックします。

戻り値 ： TRUE　- 文字列は全て指定された文字で構成されている
FALSE - 文字列中に指定された文字以外が含まれている（pStr が NULL／空文字列 を含む）

41.3.38. １６進文字列を数値に変換（AjcHexToUI / AjcHexToULL）

形　式 ： UI AjcHexToUI (C_UTP pStr);
ULL AjcHexToULL(C_UTP pStr);

引　数 ： pSrc - １６進文字列の先頭アドレス

説　明 ： pSrc で示される１６進文字列（'0'～'9' / 'A'～'F' / 'a'～'f'）で構成される文字列）を数値に変換します。
文字列の先頭部分がスペース（0x09～0x0D or 0x20）である場合は、スペースをスキップします。
先頭が "0x" / "0X" である場合は、これをスキップして変換します。
１６進文字以外が現れた時点で変換を終了します。
文字列の先頭が１６進文字以外である場合は、０を返します。

戻り値 ： １６進文字列が表す数値

41.3.39. ８進文字列を数値に変換（AjcOctToUI / AjcOctToULL）

形　式 ： UI AjcOctToUI (C_UTP pStr);
ULL AjcOctToULL(C_UTP pStr);

引　数 ： pSrc - ８進文字列の先頭アドレス

説　明 ： pSrc で示される８進文字列（'0'～'7' で構成される文字列）を数値に変換します。
文字列の先頭部分がスペース（0x09～0x0D or 0x20）である場合は、スペースをスキップします。
８進文字以外が現れた時点で変換を終了します。
文字列の先頭が８進文字以外である場合は、０を返します。

戻り値 ： ８進文字列が表す数値

41.3.40. １６進文字列を数値に変換［１６進文字チェック付］（AjcHexStrTo???）

形　式 ： BOOL AjcHexStrToUB (C_UTP pStr, UI nLen, UBP pVal);
BOOL AjcHexStrToUW (C_UTP pStr, UI nLen, UWP pVal);
BOOL AjcHexStrToUI (C_UTP pStr, UI nLen, ULP pVal);
BOOL AjcHexStrToULL(C_UTP pStr, UI nLen, ULLP pVal);

引　数 ： pStr - １６進文字列の先頭アドレス
nLen - １６進文字列の文字数
pVal - 変換した数値を格納するバッファのアドレス

説　明 ： pStr,nLen で示される１６進文字列（'0'～'9' / 'A'～'F' / 'a'～'f' で構成される文字列）を数値に変換します。
nLen で示される文字数の文字は、１６進文字でなければなりません。

戻り値 ： TRUE - ＯＫ
FALSE - エラー（nLen で示される文字数の文字中に１６進文字以外が存在する）

41.3.41. １０／１６進文字列を数値に変換（AjcAscToInt / AjcAscToLInt）

形　式 ： SI AjcAscToInt (C_UTP pStr);
SLL AjcAscToLInt (C_UTP pStr);

引　数 ： pStr - １０／１６進文字列の先頭アドレス

説　明 ： 先頭部分の空白を無視して、最初の文字が「0x」である場合は、当該文字列を１６進数に変換します。
最初の文字が「0x」以外である場合は、当該文字列を１０進数に変換します。

戻り値 ： 変換した数値

41.3.42. １０／１６進文字列を実数値に変換（AjcAscToReal）

形　式 ： double　AjcAscToReal(C_UTP pStr);

引　数 ： pStr - １０／１６進文字列の先頭アドレス

説　明 ： 文字列を実数値に変換します。先行する空白は無視します。
尚、有効な文字列の先頭が「0x」である場合は、１６進数の整数とみなします。

戻り値 ： 変換した数値

41.3.43. 文字列中の区切られた複数の数値を取得する（AjcGetSepNumbers）

形　式 ： UI AjcGetSepNumbers(C_UTP pStr, UT sep, C_UTP pTyp, ...)

引　数 ： pStr - 複数の数値を含む文字列の先頭アドレス
sep　- 区切り文字
pTyp - 数値のタイプを表す文字列のアドレス
...　- 取得した数値を格納する変数群のアドレス

説　明 ： 「pStr」で示される文字列から「sep」で区切られた複数の数値を取得し、変数に設定します。
各数値の前後には空白を置くことができます。
「sep」で区切られた文字列が全て空白である（あるいは文字が無い）場合は、対応する変数に値を設定しません。
「pTyp」は、数値の個数とタイプを指定します。(文字列の文字数が数値の個数を意味します)
「pTyp」で示す文字列は、以下の文字で構成します。

'b' or 'B' : ８ビット整数
'w' or 'W' : １６ビット整数
'i' or 'I' : ３２ビット整数
'l' or 'L' : ６４ビット整数
'f' or 'F' : 実数 (double)

上記以外の文字を指定した場合は、対応する変数に値を設定しません。

（例）

```
int    a;
ULL    b;
double f;
AjcGetSepNumbers("-10 / 0x200 / 3.14", '/', "ilf", &a, &b, &f);  // --> a=-10   , b=0x200 , f=3.14
AjcGetSepNumbers("-10 /       / 3.14", '/', "ilf", &a, &b, &f);  // --> a=-10   , b=未設定, f=3.14
AjcGetSepNumbers("    /       / 3.14", '/', "ilf", &a, &b, &f);  // --> a=未設定, b=未設定, f=3.14
AjcGetSepNumbers("-10 / 0x200 / 3.14", '/', "il.", &a, &b, &f);  // --> a=-10   , b=0x200 , f=未設定
```

戻り値 ： 変数に設定した数値の個数

41.3.44. 接頭語に続くパラメタ取得（AjcGetAfterPrefix）

形　式 ： BOOL　AjcGetAfterPrefix (C_UTP pParam, C_UTP pPrefix, UIP pVal, UTP pBuf, UI lBuf);

引　数 ： pParam　- パラメタ文字列へのポインタ
pPrefix　- パラメタ接頭語へのポインタ
pVal　　- パラメタ接頭語に続く文字列が表す数値を格納するバッファのアドレス
pBuf　　- パラメタ接頭語に続く文字列を格納するバッファのアドレス
lBuf　　- パラメタ接頭語に続く文字列を格納するバッファのバイト数／文字数

説　明 ： 接頭語に続く部分のパラメタ情報を取得します。

（例）

```
UI   val;
BC   buf[128];
AjcGetAfterPrefix (“/P123”, “/P”, &val, buf, sizeof
```

⇒

```
val = 123
buf = “123”
```

戻り値 ： なし

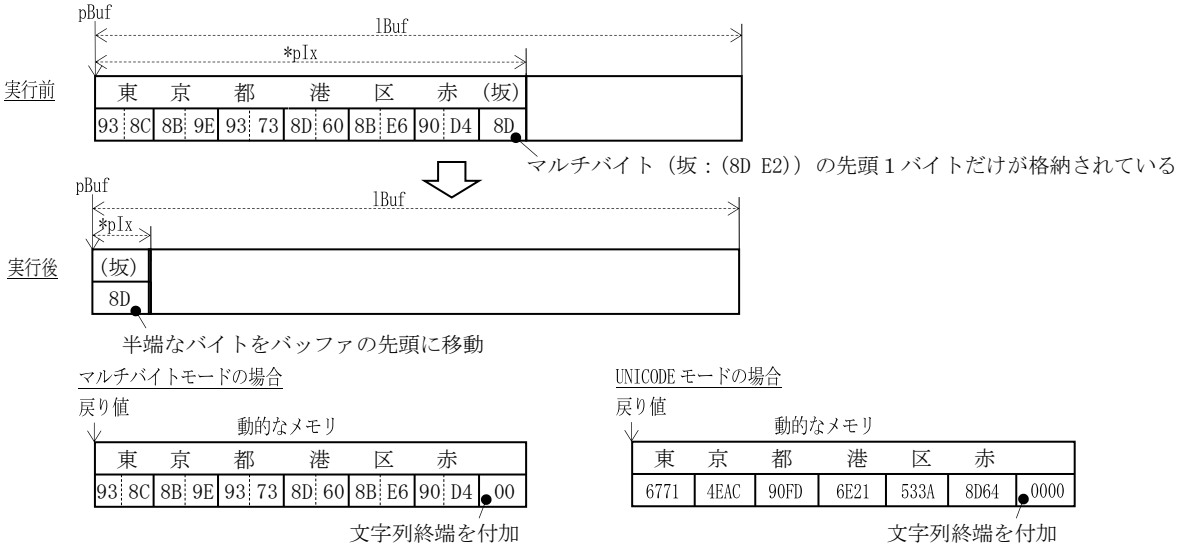
41.3.45. マルチバイトが分断されないようにマルチバイトテキストを抽出（AjcExtractCompletedText）

形　式 ： UTP　AjcExtractCompletedText (BCP pBuf, UI lBuf, UIP pIx, AJCMBCKIND mbk);

引　数 ： pBuf　　- マルチバイト文字列バッファアドレス
lBuf　　- マルチバイト文字列バッファのバイト数
pIx　　- バッファ中の有効なデータのバイト数（バッファ中の空領域の先頭インデクス）を示す変数のアドレス
mbk　　- マルチバイト文字列の文字コード（AJCMBC_SJIS / AJCMBC_EUC / AJCMBC_UTF8）

説　明 ： マルチバイト文字が分断されないように、完結したテキストを取り出します。
*pIx は、バッファの先頭に既に格納済のテキストバイト数を示します。
文字列の末尾が、分断されたマルチバイト文字である場合、当該分断バイトをバッファの先頭に移動し、*pIx に移動したバイト数を設定します。（つまり、関数実行後、*pIx はバッファ中の空領域の先頭を示します）
文字列の末尾が、分断されたマルチバイト文字でない場合は、*pIx=0 を設定します。
抽出した文字列は、末尾に文字列終端(0x00)を付加し、確保した動的なメモリに格納して返します。

（例）mbk=AJCMBC_SJIS, fUnicode=FALSE の場合の例



戻り値 ： ≠ NULL - マルチバイト文字が分断されないように、完結したテキストのアドレス（動的なメモリ）
＝ NULL　- 完結したテキストなし／エラー

注　意 ： *pIx で示すテキスト中に文字列の終端(0x00)がある場合は、終端文字の直前までを有効なテキストとします。

41.3.46. マルチバイトが分断されないように抽出したマルチバイトテキストを解放（AjcReleaseCompletedText）

形　式　：　VO　AjcReleaseCompletedText (BCP pBuf);

引　数　：　pBuf　- AjcExtractCompletedText()の戻り値

説　明　：　AjcExtractCompletedText()で確保した動的なメモリを解放します。

戻り値　：　なし

41.3.47. Ｃ言語表記文字列をバイナリ化（AjcCLangStrToBin）

形　式　：　UI　AjcCLangStrToBin (C_UTP pTxt, UTP pBuf, UI lBuf);

引　数　：　pTxt　- Ｃ言語表記文字列のアドレス
pBuf　- バイナリ化した文字列を格納するバッファのアドレス　　（不要時はNULL）
lBuf　- バイナリ化した文字列を格納するバッファのバイト数／文字数（文字列終端は含む，不要時は０）

説　明　：　Ｃ言語表記で記述された文字列を、（エスケープ表記部分を）バイナリ化した文字列を作成します。
pBuf=NULLを指定した場合は、バイナリ化に必要なバイト数／文字数（文字列終端は含まない）を算出します、
エスケープ文字と変換内容は以下の通りです。

#	エスケープ表記	変換内容	備考
1	¥a	0x07	警告音
2	¥b	0x08	バックスペース
3	¥n	0x0A	改行
4	¥r	0x0D	復帰
5	¥f	0x0C	改ページ
6	¥t	0x09	タブ
7	¥v	0x0B	垂直タブ
8	¥¥	0x5C	文字としての「¥」
9	¥?	0x3F	文字としての「?」
10	¥’	0x27	文字としての「’」
11	¥”	0x22	文字としての「”」
12	¥0	0x00	ヌル文字
13	¥ooo	ooo が示す８進数	ex. ¥377 → 0xFF
14	¥xhh	hh が示す１６進数	ex. ¥xFF → 0xFF
15	¥e　（＊１）	0x1B	エスケープ

＊１：Ｃ言語表記にはありません。本ＡＰＩ独自のエスケープ文字です。

戻り値　：　pBuf≠NULLの場合、pBufで示すバッファへ格納したバイナリ化文字列のバイト数／文字数（文字列終端は含まない）
pBuf＝NULLの場合、バイナリ化に必要なバッファのバイト数／文字数（文字列終端は含まない）

41.3.48. バイナリ文字列Ｃ言語表記文字列化（AjcCLangStrToBin）

形　式　：　UI　AjcBinToCLangStr (C_UTP pTxt, UTP pBuf, UI lBuf);

引　数　：　pTxt　- バイナリ文字列のアドレス
pBuf　- Ｃ言語表記文字列化した文字列を格納するバッファのアドレス　　（不要時はNULL）
lBuf　- Ｃ言語表記文字列化した文字列を格納するバッファのバイト数／文字数（文字列終端は含む，不要時は０）

説　明　：　バイナリ文字列のバイナリ部分をＣ言語表記文字列に変換した文字列を作成します。
このＡＰＩは、AjcCLangStrToBin ()でバイナリ化した文字列を、元のＣ言語表記文字列に戻します。
但し、８進数表記の部分は１６進数表記となります。また、0x1Bは"¥x1B"に変換します。
pBuf=NULLを指定した場合は、Ｃ言語表記文字列化に必要なバイト数／文字数（文字列終端は含まない）を算出します、

戻り値　：　pBuf≠NULLの場合、pBufで示すバッファに格納したＣ言語表記文字列のバイト数／文字数（文字列終端は含まない）
pBuf＝NULLの場合、Ｃ言語表記文字列化に必要なバッファのバイト数／文字数（文字列終端は含まない）

41.3.49. テキストの表示桁数取得(AjcTextLen)

形　式 ： UI　AjcTextLen(C_UTP pText);

引　数 ： pText　　- 表示桁数を算出するテキストのアドレス

説　明 ： 半角文字を１桁，全角文字を２桁として、テキストの桁数を算出します。(文字列の終端(0x00)は含みません)
バイト文字の場合は、文字列のバイト数を返します。
ワイド文字の場合は、EastAsianWidth.txt による日本語での扱い方の指針 により半角／全角を判断します。

戻り値 ： 算出したテキストの表示桁数

備　考 ： UNICODE モードの場合、「EastAsianWidth.txt」に従って全角／半角を判断します。
従って、東アジア以外の地域の文字に関しては、正常に判断できない場合があります。

41.3.50. 文字の全角／半角チェック(AjcIsBigChar)

形　式 ： BOOL　AjcIsBigChar (const UT txt[2]);

引　数 ： txt　　- 全角文字かチェックする文字列（２バイト／２文字)

説　明 ： バイト文字の場合は、テキストの先頭２バイトから全角文字か、半角文字かを判断します。
ワイド文字の場合は、EastAsianWidth.txt による日本語での扱い方の指針 により半角／全角を判断します。

戻り値 ： TRUE　- 全角文字
FALSE – 半角文字

備　考 ： UNICODE モードの場合、「EastAsianWidth.txt」に従って全角／半角を判断します。
従って、東アジア以外の地域の文字に関しては、正常に判断できない場合があります。

41.3.51. UNICODE コードポイント算出(AjcCodePoint)

形　式 ： UI　AjcTextToCodePoint(const UT txt[2]);

引　数 ： txt　- UNICODE コードポイントを算出するための１～２バイト／１～２文字のテキスト

説　明 ： UNICODE モードの場合、テキスト１～２文字から、UNICODE コードポイントを算出します。
バイト文字の場合は、テキストを UNICODE に変換した上で、UNICODE コードポイント(21bit)を算出します

戻り値 ： UNICODE コードポイント

41.3.52. UNICODE コードポイントからテキスト生成(AjcCodePointToText)

形　式 ： UI　AjcCodePointToText(cp, UT txt[2]);

引　数 ： cp　- UNICODE コードポイントコードポイント
txt　- 作成した文字列１～２バイト／１～２文字を格納するバッファ

説　明 ： UNICODE モードの場合、UNICODE コードポイントから UNICODE テキスト１～２文字を作成します。
バイト文字の場合は、UNICODE コードポイントから UNICODE テキスト１～２文字を作成し、これをバイト文字に変換します。
生成した文字列が１バイト／１文字の場合は、２バイト目／２文字目は０が設定されます。

戻り値 ： １～２：作成した文字列の文字数
０　　：エラー

41.3.53. フォント情報(LOGFONT)を文字列に変換(AjcLogFontToText)

形　式 ： UI　AjcLogFontToText (const LOGFONTA * pLogFont, UTP pTxt, UI lTxt, BOOL fKey);

引　数 ： pLogFont　- フォント情報のアドレス
pTxt　- 作成した文字列を格納するバッファのアドレス（不要時は NULL）
lTxt　- 作成した文字列を格納するバッファの文字数
fKey　- 文字列中にキー（FN=, LF=）を含めるか否かのフラグ（TRUE:含める, FALSE:含めない）

説　明 ： pLogFont で示すフォント情報を文字列に変換します。
fKey=TRUE の場合は、以下のような文字列に変換します。

“FN=MS Gothic,LF=10/0/0/0/400/0/0/0/1/1/2/1/1”

「FN=」はフォントフェース名を、「LF=」はフォントフェース名(lfFaceName)を除く LOGFONT 構造体のメンバ値(１０進数)を示します。

fKey=FALSE の場合は、以下のような文字列に変換します。

“MS Gothic,10/0/0/0/400/0/0/0/1/1/2/1/1”

戻り値 ： ≠０：文字列を作成するのに必要な文字数（終端(¥0)を含む）
＝０：エラー

41.3.54. 文字列をフォント情報(LOGFONT) に変換(AjcTextToLogFont)

形　式 ： BOOL　AjcTextToLogFont (C_UTP pTxt, LPLOGFONTA pLogFont);

引　数 ： pTxt　- 文字列のアドレス
pLogFont　- フォント情報を格納するバッファのアドレス

説　明 ： pTxt で示すテキストからフォント情報を作成します。
pTxt は、AjcLogFontToText()で作成した文字列を指定します。

戻り値 ： TRUE　：成功
FALSE：失敗

41.4. マルチバイト文字列用インライン関数

インライン関数で、マルチバイト文字列の引数を「BCP」とすることにより、キャスト（BCP -> UBP）を不要とする

形式				
__inline	int	mbsbtype	(C_BCP P, size_t L)	{return _mbsbtype ((C_UBP)P, (size_t)L);}
__inline	BCP	mbschr	(C_BCP P , int C)	{return (BCP)_mbschr ((C_UBP)P , (int)C);}
__inline	int	mbscmp	(C_BCP P1, C_BCP P2)	{return _mbscmp ((C_UBP)P1, (C_UBP)P2);}
__inline	int	mbsncmp	(C_BCP P1, C_BCP P2, size_t L)	{return _mbsncmp ((C_UBP)P1, (C_UBP)P2, (size_t)L);}
__inline	int	mbsicmp	(C_BCP P1, C_BCP P2)	{return _mbsicmp ((C_UBP)P1, (C_UBP)P2);}
__inline	int	mbsnicmp	(C_BCP P1, C_BCP P2, size_t L)	{return _mbsnicmp ((C_UBP)P1, (C_UBP)P2, (size_t)L);}
__inline	BCP	mbsstr	(C_BCP P1, C_BCP P2)	{return (BCP)_mbsstr ((C_UBP)P1, (C_UBP)P2);}
__inline	size_t	mbslen	(C_BCP P)	{return _mbslen ((C_UBP)P);}
__inline	BCP	mbsistr	(C_BCP P1, C_BCP P2)	{return (BCP)AjcMbsIStrA(P1, P2);}
__inline	BCP	mbstok	(BCP P1, C_BCP P2)	{return (BCP)AjcMbsTokA (P1, P2);}

41.5. 文字列操作マクロ

マルチバイト文字列とワイド文字列で共通に使用できるように、以下のマクロを定義しています。

バイト文字列操作ファンクション（「UNICODE」指定なし）

マクロ形式	展開形	備考
MAjcStrChr (P, C)	strchr ((P) , (C))	バイト文字有無チェック
MAjcStrCmp (P1, P2)	strcmp ((P1), (P2))	バイト文字列比較（英大小区別あり）
MAjcStrNCmp (P1, P2)	strncmp ((P1), (P2), (L))	バイト文字列比較（英大小区別あり，バイト数指定）
MAjcStrICmp(P1, P2)	_stricmp((P1), (P2))	バイト文字列比較（英大小区別なし）
MAjcStrNICmp(P1, P2)	_strnicmp((P1), (P2), (L))	バイト文字列比較（英大小区別なし，バイト数指定）
MAjcStrStr (P1, P2)	strstr ((P1), (P2))	バイト文字列中の文字列検索（英大小区別あり）
MAjcStrIStr(P1, P2)	AjcStrIStrA((P1), (P2))	バイト文字列中の文字列検索（英大小区別なし）
MAjcStrTok (P1, P2)	AjcStrTokA ((P1), (P2)) ※1	バイト文字列トークン分離
MAjcStrLen (P)	strlen ((P))	バイト文字列のバイト数
MAjcStrCpy (P1, L, P2)	strncpy_s ((P1), (L), (P2), (L)-1) ※2	バイト文字列コピー
MAjcStrNCpy(P1, L, P2, N)	strncpy_s((P1), (L), (P2), (N))	バイト文字列コピー（最大バイト数指定）
MAjcStrCat (P1, L, P2)	strcat_s ((P1), (L), (P2))	バイト文字列連結
MAjcStrNCat (P1, L, P2, N)	strcat_s ((P1), (L), (P2), (N))	バイト文字列連結 （最大バイト数指定）
MAjcStrBytes(P)	(strlen(P) + 1)	バイト文字列のバイト数（終端('¥0')を含む）
MAjcMbsBType(P, L)	mbsbtype ((P), (L))	指定バイト位置のマルチバイト種別を取得 戻り値 = _MBC_SINGLE - バイト文字 _MBC_LEAD - 全角１文字目 _MBC_TRAIL - 全角２文字目 _MBC_ILLEGAL - 不当な文字
MAjcMbsChr (P, C)	mbschr ((P) , (UI)(C))	マルチバイト文字有無チェック
MAjcMbsCmp (P1, P2)	mbscmp ((P1), (P2))	マルチバイト文字列比較（英大小区別あり）
MAjcMbsNCmp(P1, P2, L)	mbsncmp ((P1), (P2), (L))	マルチバイト文字列比較（英大小区別あり，文字数指定）
MAjcMbsICmp(P1, P2)	mbsicmp ((P1), (P2))	マルチバイト文字列比較（英大小区別なし）
MAjcMbsNICmp(P1, P2, L)	mbsnicmp ((P1), (P2), (L))	マルチバイト文字列比較（英大小区別なし，文字数指定）
MAjcMbsStr (P1, P2)	mbsstr ((P1), (P2))	マルチバイト文字列中の文字列検索（英大小区別あり）
MAjcMbsLen (P)	mbslen ((P))	マルチバイト文字列の文字数（全角を１文字としてカウント）
MAjcMbsIStr(P1, P2)	AjcMcsIStrA((P1), (P2))	マルチバイト文字列中の文字列検索（英大小区別なし）
MAjcMbsTok (P1, P2)	AjcMbsTokA ((P1), (P2)) ※1	マルチバイトトークン分離
MAjcMakePath	MAjcMakePathW(PATH, DRV, DIR, FNAME, FEXT) ※3	_makepath_s (PATH, sizeof(PATH)/2, DRV, DIR, FNAME, FEXT)
MAjcSplitPath	MAjcSplitPathW(PATH, DRV, DIR, FNAME, FEXT) ※4	_splitpath_s(PATH, DRV, ¥ DRV == NULL ? 0 : sizeof(DRV), ¥ DIR, ¥ DIR == NULL ? 0 : sizeof(DIR), ¥ FNAME, ¥ FNAME == NULL ? 0 : sizeof(FNAME), ¥ FEXT, ¥ FEXT == NULL ? 0 : sizeof(FEXT))

ワイド文字列操作ファンクション（「UNICODE」指定あり）

マクロ形式	展開形	備考
MAjcStrChr (P, C)	wcschr ((P), (UI)(C))	ワイド文字有無チェック
MAjcStrCmp (P1, P2)	wcscmp ((P1), (P2))	ワイド文字列比較（英大小区別あり）
MAjcStrNCmp (P1, P2)	wcsncmp ((P1), (P2), (L))	ワイド文字列比較（英大小区別あり，文字数指定）
MAjcStrICmp(P1, P2)	_wcsicmp ((P1), (P2))	ワイド文字列比較（英大小区別なし）
MAjcStrNICmp(P1, P2)	_wcsnicmp ((P1), (P2), (L))	ワイド文字列比較（英大小区別なし，文字数指定）
MAjcStrStr (P1, P2)	wcsstr ((P1), (P2))	ワイド文字列中の文字列検索（英大小区別あり）
MAjcStrIStr(P1, P2)	AjcStrIStrW((P1), (P2))	ワイド文字列中の文字列検索（英大小区別なし）
MAjcStrTok (P1, P2)	AjcStrTokW ((P1), (P2)) ※1	ワイド文字列トークン分離
MAjcStrLen (P)	wcslen ((P))	ワイド文字列のワード数
MAjcStrCpy (P1, L, P2)	wcsncpy_s ((P1), (L), (P2), (L)-1) ※2	ワイド文字列コピー
MAjcStrNCpy(P1, L, P2, N)	wcsncpy_s((P1), (L), (P2), (N))	ワイド文字列コピー（最大文字数指定
MAjcStrCat (P1, L, P2)	wcscat_s ((P1), (L), (P2))	ワイド文字列連結
MAjcStrNCat (P1, L, P2, N)	wcsncat_s ((P1), (L), (P2), (N))	ワイド文字列連結
MAjcStrBytes(P)	((wcslen(P) + 1) * 2)	ワイド文字列のバイト数（終端('¥0')を含む）
MAjcMbsBType(P, L)	(_MBC_SINGLE)	指定文字位置の種別を取得 _MBC_SINGLE - 単独文字（固定）
MAjcMbsChr (P, C)	wcschr ((P), (UI)(C))	ワイド文字有無チェック
MAjcMbsCmp (P1, P2)	wcscmp ((P1), (P2))	ワイド文字列比較（英大小区別あり）
MAjcMbsNCmp(P1, P2, L)	wcsncmp ((P1), (P2), (L))	ワイド文字列比較（英大小区別あり，文字数指定）
MAjcMbsICmp(P1, P2)	_wcsicmp ((P1), (P2))	ワイド文字列比較（英大小区別なし）
MAjcMbsNICmp(P1, P2, L)	_wcsnicmp ((P1), (P2), (L))	ワイド文字列比較（英大小区別なし，文字数指定）
MAjcMbsStr (P1, P2)	wcsstr ((P1), (P2))	ワイド文字列中の文字列検索（英大小区別あり）
MAjcMbsLen (P)	wcslen ((P))	ワイド文字列の文字数
MAjcMbsIStr(P1, P2)	AjcMbsIStrW((P1), (P2))	ワイド文字列中の文字列検索（英大小区別なし）
MAjcMbsTok (P1, P2)	AjcMbsTokW ((P1), (P2)) ※1	ワイドトークン分離
MAjcMakePath	MAjcMakePathW(PATH, DRV, DIR, FNAME, FEXT) ※3	_wmakepath_s (PATH, sizeof(PATH)/2, DRV, DIR, FNAME, FEXT)
MAjcSplitPath	MAjcSplitPathW(PATH, DRV, DIR, FNAME, FEXT) ※4	_wsplitpath_s(PATH, DRV, ¥ DRV == NULL ? 0 : sizeof(DRV) / 2, ¥ DIR, ¥ DIR == NULL ? 0 : sizeof(DIR) / 2, ¥ FNAME, ¥ FNAME == NULL ? 0 : sizeof(FNAME) / 2,¥ FEXT, ¥ FEXT == NULL ? 0 : sizeof(FEXT) / 2)

※1： 本ライブラリのＡＰＩ「AjcStrTok」「AjcMbsTok」が実行されます。
これらのＡＰＩでは、strtok_s(), _mbstok_s()やwcstok_s()が実行され、context 変数は、内部でスレッドセーフな変数が割り当てられます。
但し、同一スレッドで複数のトークン取得を並列して実行することはできません。
以下のようなコードは、正常に実行できません。

```
p1A = MAjcStrTok(str1, “;”);
p2A = MAjcStrTok(str2, “/”);
p1B = MAjcStrTok(NULL, “;”);
p2B = MAjcStrTok(NULL, “/”);
```

※2：転送元の文字列が長い場合でも、文字列の途中までコピーするように「strncpy_s()」/「wcsncpy_s()」を展開します。
「strcpy_s()」/「wcscpy_s()」では、転送元の文字列が長い場合、C ランタイムエラーとなります。

※3：PATH のサイズは自動計算されます。DRV, DIR, FNAME, FEXT はそれぞれ NULL とすることができます。
PATH は、path[MAX_PATH]のように sizeof 演算子で算出できるようになっていなければなりません。

※4：DRV, DIR, FNAME, FEXT のサイズは自動計算されます。DRV, DIR, FNAME, FEXT はそれぞれ NULL とすることができます。
これらは、drive[_MAX_DRIVE], directory[_MAX_DIR],file_name[_MAX_FNAME], file_ext[_MAX_EXT]のように sizeof 演算子で算出できるよになっていなければなりません。

41.6. 文字コードチェックマクロ

マルチバイト文字とワイド文字で文字の種別チェックを共通に使用できるように、以下のマクロを定義しています。

バイト文字チェックファンクション（「UNICODE」指定なし）

マクロ形式	展開形	備考
MAjcIsLead(C)	(ismbblead(C) != 0)	マルチバイトの１バイト目かチェック
MAjcIsTrail(C)	(ismbbtrail(C) != 0)	マルチバイトの２バイト目かチェック
MAjcIsAlpha(C)	(((C) >= 0 && (C) <= 255) ? (isalpha (C) != 0) : 0)	半角英字チェック
MAjcIsAlNum(C)	(((C) >= 0 && (C) <= 255) ? (isalnum (C) != 0) : 0)	半角英字／半角数字チェック
MAjcIsDigit(C)	(((C) >= 0 && (C) <= 255) ? (isdigit (C) != 0) : 0)	半角数字チェック
MAjcIsXDigit(C)	(((C) >= 0 && (C) <= 255) ? (isxdigit(C) != 0) : 0)	半角１６進文字チェック
MAjcIsUpper(C)	(((C) >= 0 && (C) <= 255) ? (isupper (C) != 0) : 0)	半角英字の大文字（'A' ～ 'Z'）チェック
MAjcIsLower(C)	(((C) >= 0 && (C) <= 255) ? (islower (C) != 0) : 0)	半角英字の小文字（'a' ～ 'z'）かチェック
MAjcIsSpace(C)	(((C) >= 0 && (C) <= 255) ? (isspace (C) != 0) : 0)	半角空白文字（0x20 or 0x09～0x0D）チェック
MAjcIsCntrl(C)	(((C) >= 0 && (C) <= 255) ? (iscntrl (C) != 0) : 0)	制御コード（0x00～0x1F, 0x7F）チェック
MAjcIsPrint(C)	(((C) >= 0 && (C) <= 255) ? (isprint (C) != 0) : 0)	半角印字可能文字チェック
MAjcIsBlank(C)	((C) == ' ' || (C) == '¥t')	半角空白文字チェック
MAjcAscIsAlpha(C)	(((C) >= 0 && (C) <= 255) ? (isalpha (C) != 0) : 0)	半角英字チェック
MAjcAscIsAlNum(C)	(((C) >= 0 && (C) <= 255) ? (isalnum (C) != 0) : 0)	半角英字／半角数字チェック
MAjcAscIsDigit(C)	(((C) >= 0 && (C) <= 255) ? (isdigit (C) != 0) : 0)	半角数字チェック
MAjcAscIsXDigit(C)	(((C) >= 0 && (C) <= 255) ? (isxdigit(C) != 0) : 0)	半角１６進文字チェック
MAjcAscIsPrint(C)	(((C) >= 0 && (C) <= 255) ? (isprint (C) != 0) : 0)	半角印字可能文字チェック
MAjcMbcIsAlpha(C)	(_ismbcalpha(C) != 0)	半角英字　／全角英字チェック
MAjcMbcIsAlNum(C)	(_ismbcalnum(C) != 0)	半角英数字／全角英数字チェック
MAjcMbcIsDigit(C)	(_ismbcdigit(C) != 0)	半角数字　／全角数字チェック
MAjcMbcIsXDigit(C)	(((C) >= '0' && (C) <= '9') || ((C) >= 'A' && (C) <= 'F') || ((C) >= 'a' && (C) <= 'f') || ((C) >= '０' && (C) <= '９') || ((C) >= 'Ａ' && (C) <= 'Ｆ') || ((C) >= 'ａ' && (C) <= 'ｆ'))	半角１６進文字／全角１６進文字チェック
MAjcMbcIsUpper(C)	(_ismbcupper(C) != 0)	半角英字の大文字／全角英字の大文字チェック
MAjcMbcIsLower(C)	(_ismbclower(C) != 0)	半角英字の小文字／全角英字の小文字チェック
MAjcMbcIsSpace(C)	(_ismbcspace(C) != 0)	半角空白文字（0x20 or 0x09～0x0D）/全角空白チェック
MAjcMbcIsBlank(C)	((C) == ' ' || (C) == '　' || (C) == '¥t')	半角空白，TAB ／ 全角空白チェック
MAjcIsBigChar(P)	(MAjcIsBigCharA(P)) // Pはバイト文字2文字へのポインタ	全角１バイト目+２バイト目の場合 TRUE を返す

ワイド文字チェックファンクション（「UNICODE」指定あり）

マクロ形式	展開形	備考
MAjcIsLead(C)	((C) >= 0xD800 && (C) <= 0xDBFF)	上位サロゲートペア
MAjcIsTrail(C)	((C) >= 0xDC00 && (C) <= 0xDFFF)	下位サロゲートペア
MAjcIsAlpha(C)	(iswalpha(C) != 0)	半角英字　／全角英字チェック
MAjcIsAlNum(C)	(iswalnum(C) != 0)	半角英数字／全角英数字チェック
MAjcIsDigit(C)	(iswdigit(C) != 0)	半角数字　／全角数字チェック
MAjcIsXDigit(C)	(iswxdigit(C) != 0)	半角１６進文字／全角１６進文字チェック
MAjcIsUpper(C)	(iswupper(C) != 0)	半角英字の大文字／全角英字の大文字チェック
MAjcIsLower(C)	(iswlower(C) != 0)	半角英字の小文字／全角英字の小文字チェック
MAjcIsSpace(C)	(iswspace(C) != 0)	半角空白文字（0x20 or 0x09～0x0D）／全角空白チェック
MAjcIsCntrl(C)	(iswcntrl(C) != 0)	制御コード（0x00～0x1F, 0x7F）チェック
MAjcIsPrint(C)	(iswprint(C) != 0)	印字可能半角／全角文字チェック
MAjcIsBlank(C)	((C) == L' ' || (C) == L'¥t')	空白(0x20) or ＴＡＢ(0x09) チェック
MAjcAscIsAlpha(C)	(iswascii(C) != 0 && iswalpha (C) != 0)	半角英字チェック
MAjcAscIsAlNum(C)	(iswascii(C) != 0 && iswalnum (C) != 0)	半角英字／半角数字チェック
MAjcAscIsDigit(C)	(iswascii(C) != 0 && iswdigit (C) != 0)	半角数字チェック
MAjcAscIsXDigit(C)	(iswascii(C) != 0 && iswxdigit(C) != 0)	半角１６進文字チェック
MAjcAscIsPrint(C)	(iswascii(C) != 0 && iswprint (C) != 0)	半角印字可能文字チェック
MAjcMbcIsAlpha(C)	(iswalpha(C) != 0)	半角英字　／全角英字チェック
MAjcMbcIsAlNum(C)	(iswalnum(C) != 0)	半角英数字／全角英数字チェック
MAjcMbcIsDigit(C)	(iswdigit(C) != 0)	半角数字　／全角数字チェック
MAjcMbcIsXDigit(C)	(iswxdigit(C)!= 0)	半角１６進文字／全角１６進文字チェック
MAjcMbcIsUpper(C)	(iswupper(C) != 0)	半角英字の大文字／全角英字の大文字チェック
MAjcMbcIsLower(C)	(iswlower(C) != 0)	半角英字の小文字／全角英字の小文字チェック
MAjcMbcIsSpace(C)	(iswspace(C) != 0)	半角空白文字（0x20 or 0x09～0x0D）／全角空白チェック
MAjcMbcIsBlank(C)	((C) == L' ' || (C) == L' ' || (C) == L'¥t')	半角空白，TAB ／ 全角空白チェック
MAjcIsBigChar(P)	(MAjcIsBigCharW(P)) // Pはワイド文字２文字へのポインタ	サロゲート１ワード目+２ワード目の場合TRUEを返す

42. 文字列プール

文字列を溜めておくバッファ制御です。
同一文字列は、バッファに格納しません。

42.1. サポートＡＰＩ

文字列操作のサポートＡＰＩ一覧を以下に示します。

#	関数名	内容
1	AjcSplCreate	文字列プールイのンスタンス生成
2	AjcSplDelete	文字列プールイのンスタンス消去
3	AjcSplSetCompMode AjcSplGetCompMode	文字列の比較モード設定／取得
4	AjcSplRegist	文字列プールへ文字列登録
5	AjcSplFind	文字列プールから文字列を検索
6	AjcSplPartStrInPool	部分文字列一致検索（文字列プールから指定部分文字列が一致するノード検索）
7	AjcSplPoolStrInStr	部分文字列一致検索（指定文字列が文字列プール中の部分文字列と一致するかチェック）
8	AjcSplRemove	文字列プールから文字列を削除
9	AjcSplGetCount	登録済文字列数取得
10	AjcSplReset	文字列プールをリセット
11	AjcSplEnumStr	登録済み全文字列の取得
12	AjcSplGetAvlHandle	ＡＶＬ２分木のハンドルを取得する

42.1.1. 文字列プールのインスタンス生成（AjcSplCreate）

形　式 ： HAJCSPL　AjcSplCreate(EAJCCMPMODE CmpMode);

引　数 ： CmpMode － 文字列の比較方法
・AJCCMP_EXACT_WIDTH　：英大小文字を区別する
・AJCCMP_IGNORE_WIDTH　：英大小文字を区別しない
・AJCCMP_MIX　：同一の場合のみ英大小文字を区別する

説　明 ： 文字列プールのインスタンスを生成します。
CmpMode = AJCCMP_EXACT_WIDTH とした場合は、英大小文字を区別して文字列の比較を行います。(”ABC”≠”abc”)
CmpMode = AJCCMP_IGNORE_WIDTH とした場合は、英大小文字を区別せずに文字列の比較を行います。(”ABC”＝”abc”)
CmpMode = AJCCMP_MIX とした場合は、英大小文字を区別せずに文字列の比較を行いますが、同一文字列の場合は、英大小文字を区別して文字列の比較を行います。(アルファベット順に並ぶように比較する)

戻り値 ： ≠ NULL － 成功（文字列プールハンドル）
＝ NULL － 失敗

42.1.2. 文字列プールのインスタンス消去（AjcSplDelete）

形　式 ： BOOL　AjcSplDelete (HAJCSPL hSpl);

引　数 ： hSpl －　文字列プールのインスタンスハンドル

説　明 ： 文字列プールのインスタンスを消去します。

戻り値 ： TRUE　－ 成功
FALSE － 失敗

42.1.3. 文字列の比較方法設定／取得（AjcSpl{Set/Get}CompMode）

形　式 ： BOOL AjcSplSetCompMode(HAJCSPL hSpl, EAJCCMPMODE CmpMode); -- 文字列の比較方法設定
EAJCCMPMODE AjcSplGetCompMode(HAJCSPL hSpl); ---------- 文字列の比較方法取得

引　数 ： hSpl　- 文字列プールのインスタンスハンドル
CmpMode - 文字列の比較モード
・AJCCMP_EXACT_WIDTH　：英大小文字を区別する
・AJCCMP_IGNORE_WIDTH ：英大小文字を区別しない
・AJCCMP_MIX　　：同一の場合のみ英大小文字を区別する

説　明 ： 文字列の比較方法を設定/取得します。
CmpMode = AJCCMP_EXACT_WIDTH とした場合は、英大小文字を区別して文字列の比較を行います。(”ABC”≠”abc”)
CmpMode = AJCCMP_IGNORE_WIDTH とした場合は、英大小文字を区別せずに文字列の比較を行います。(”ABC”＝”abc”)
CmpMode = AJCCMP_MIX とした場合は、英大小文字を区別せずに文字列の比較を行いますが、同一文字列の場合は、英大小文字を区別して文字列の比較を行います。(アルファベット順に並ぶように比較する)

文字列の比較方法を変更した場合、文字列プールの内容は全て破棄されます。
文字列の比較方法を変更しない（現在の文字列の比較方法と同一の文字列の比較方法を指定した）場合は何もせずに TRUE を返します。

戻り値 ： 設定時：TRUE　- 成功　　取得時：≠-1 - 成功(AJCCMP_IGNORE_WIDTH, AJCCMP_EXACT_WIDTH or AJCCMP_MIX)
FALSE - 失敗　　＝-1 - 失敗

備　考 ： 例えば、４つの文字列(“Tokyo”, “tokyo”, “Sendai”, “sendai”)を登録した場合は、比較モードにより以下のように登録されます。

比較モード	AJCCMP_EXACT_WIDTH	AJCCMP_IGNORE_WIDTH	AJCCMP_MIX
文字列プールへの登録内容	Sendai Tokyo sendai tokyo	Sendai Tokyo	Sendai sendai Tokyo tokyo

42.1.4. 文字列プールへ文字列の登録（AjcSplRegist）

形　式 ： C_UTP AjcSplRegist (HAJCSPL hSpl, C_UTP pStr);

引　数 ： hSpl - 文字列プールのインスタンスハンドル
pStr - 登録する文字列のアドレス

説　明 ： 文字列プールへ文字列を登録し、登録した文字列のアドレスを返します。
文字列プール中に、既に同一文字列が存在しない場合は、文字列を登録し参照カウンタ＝１を設定します。
文字列プール中に、既に同一文字列が存在する場合は、参照カウンタをインクリメントし、新たな登録は行わずに既存の文字列のアドレスを返します。

戻り値 ： ≠NULL - 成功（登録した文字列／既存の文字列のアドレス）
＝NULL - 失敗

42.1.5. 文字列プールから文字列を検索（AjcSplFind）

形　式 ： C_UTP AjcSplFind (HAJCSPL hSpl, C_UTP pStr);

引　数 ： hSpl - 文字列プールのインスタンスハンドル
pStr - 検索する文字列のアドレス

説　明 ： 文字列プール中から、指定された文字列が存在するか検索します。
文字列プール中に、既に同一文字列が存在する場合は、文字列プール中の当該文字列のアドレスを返します。

戻り値 ： ≠NULL - 文字列が見つかった（見つかった文字列のアドレス）
＝NULL - 文字列プールな中に当該文字列は存在しない／エラー

42.1.6. 部分文字列一致検索（文字列プールの文字列が指定部分文字列と一致するかチェック）（AjcSplPartStrInPool）

形　式 ： C_UTP AjcSplPartStrInPool (HAJCSPL hSpl, C_UTP pPartStr , AJCSPL_INSTROPT opt);

引　数 ： hSpl - 文字列プールのインスタンスハンドル
pPartStr - 検索する部分文字列
opt - 検索種別（AJCSPL_ISP_MATCHFIRST：先頭一致, AJCSPL_ISP_INCLUDING：部分一致）

説　明 ： 文字列プール中から pPartStr で指定された部分文字列が一致するノード（文字列）を検索します。
opt= AJCSPL_ISP_MATCHFIRST の場合は、文字列プール中の先頭文字列が pPartStr で指定された文字列と一致するかをチェックします。
opt= AJCSPL_ISP_INCLUDING の場合は、文字列プール中の文字列が、pPartStr で指定された文字列を含むかをチェックします。
文字列プール中に指定された部分文字列を含むノードが存在しない場合は NULL を返します。

戻り値 ： ≠NULL - 文字列プール中の一致したノード（文字列）のアドレス
＝NULL - いずれの部分文字列も一致しない

42.1.7. 部分文字列一致検索（指定文字列が文字列プール中の部分文字列と一致するかチェック）（AjcSplPoolStrInStr）

形　式 ： C_UTP AjcSplPoolStrInStr (HAJCSPL hSpl, C_UTP pStr, AJCSPL_INSTROPT opt);

引　数 ： hSpl - 文字列プールのインスタンスハンドル
pStr - 部分文字列が含まれているかチェックする文字列のアドレス
opt - 検索種別（AJCSPL_ISP_MATCHFIRST：先頭一致, AJCSPL_ISP_INCLUDING：部分一致）

説　明 ： pStr で指定された文字列と、文字列プール中のいずれかの部分文字列が一致するかチェックします。
opt= AJCSPL_ISP_MATCHFIRST の場合は、pStr で指定された文字列の先頭部分が文字列プール中のいずれかの部分文字列と一致するかをチェックします。
opt= AJCSPL_ISP_INCLUDING の場合は、pStr で指定された文字列が、文字列プール中のいずれかの文字列を含むかをチェックします。

戻り値 ： ≠NULL - 文字列プール中の一致したノード（部分文字列）のアドレス
＝NULL - いずれの部分文字列も一致しない

42.1.8. 文字列プールから文字列を削除（AjcSplRemove）

形　式 ： BOOL AjcSplRemove (HAJCSPL hSpl, C_UTP pStr);

引　数 ： hSpl - 文字列プールのインスタンスハンドル
pStr - 削除する文字列のアドレス

説　明 ： 文字列プール中に指定された文字列が存在する場合は、参照カウンタを減算します。
参照カウンタが 0 となった場合は、文字列プール中から当該文字列を削除します。
文字列プール中に、同一文字列が存在しない場合は FALSE を返します。

戻り値 ： TRUE - 参照カウンタを減算あるいは、当該文字列を削除した
FALSE - 文字列プール中に当該文字列は存在しない

42.1.9. 文字列プールの登録済文字列数取得（AjcSplGetCount）

形　式 ： UI AjcSplGetCount (HAJCSPL hSpl);

引　数 ： hSpl - 文字列プールのインスタンスハンドル

説　明 ： 文字列プール中に登録済文字列の個数を取得します。

戻り値 ： 登録済の文字列の個数

42.1.10. 文字列プールリセット（AjcSplReset）

形　式 ： C_UTP AjcSplReset (HAJCSPL hSpl);

引　数 ： hSpl - 文字列プールのインスタンスハンドル

説　明 ： 文字列プール中の全ての文字列を破棄します。

戻り値 ： TRUE - 成功（見つかった文字列のアドレス）
FALSE - 失敗

42.1.11. 登録済み全文字列の取得（AjcSplEnumStr）

形　式 ： UI AjcSplEnumStr (HAJCSPL hSpl, UX cbp, BOOL (CALLBACK *cbNtcStr)(C_UTP pStr, UX cbp), BOOL fDownSeq);

引　数 ： hSpl - 文字列プールのインスタンスハンドル
cbp - コールバックパラメタ
cbNtcStr - 文字列通知用コールバック関数（不要時は NULL）
fDownSeq - 文字列読み出し順序の指定（FALSE：昇順, TRUE：降順）

説　明 ： 文字列プール中の全ての文字列コールバック関数へ通知します。
総文字列数だけを取得する場合は、cbNtcStr=NULL を指定します。

戻り値 ： 総文字列数（エラーの場合は０を返します）

コールバック：

cbNtcStr（文字列通知用コールバック関数）

形　式 ： BOOL CALLBACK *cbNtcStr* (C_UTP pStr, UX cbp);

引　数 ： pStr - 文字列へのポインタ
cbp - コールバックパラメタ

説　明 ： 順次、登録済文字列を通知します。

戻り値 ： TRUE ： 文字列の通知を継続する
FALSE ： 文字列の通知を中止する

42.1.12. ＡＶＬ２分木ハンドル取得（AjcSplGetAvlHandle）

形　式 ： HAJCAVL AjcSplGetAvlHandle (HAJCSPL hSpl);

引　数 ： hSpl - 文字列プールのインスタンスハンドル

説　明 ： 文字列プール中のの実態は、文字列をキーとしたＡＶＬ２分木です。
この関数は、ＡＶＬ２分木のハンドルを返します。

戻り値 ： ≠NULL - 成功（ＡＶＬ２分木のハンドル）
＝NULL - 失敗

43. 日付／時刻

43.1. サポートＡＰＩ

#	関 数 名	内容
1	AjcTime1970ToSysTime	1970/1/1 00:00:00 からの通算秒をシステムタイムに変換
2	AjcSysTimeToTime1970	システムタイムを 1970/1/1 00:00:00 からの通算秒に変換
3	AjcSysTimeToLocalTime	システムタイムをローカルタイムに変換
4	AjcTimeUpdate	指定秒数進めた（あるいは戻した）日時を求める
5	AjcDateStr	日付文字列取得
6	AjcTimeStr	時刻文字列取得
7	AjcSetDateFromText[Ex]	日付文字列から SYSTEMTIME 構造体の日付情報を設定
8	AjcSetTimeFromText[Ex]	時刻文字列から SYSTEMTIME 構造体の時刻情報を設定
9	AjcSetDateAndTimeFromText[Ex]	日時文字列から SYSTEMTIME 構造体の日時情報を設定
10	AjcGetRandomDateAndTime	ランダムな日付と時刻を取得
11	AjcSetRandomDateAndTime	ランダムな日付と時刻の範囲を設定

43.1.1. 1970/1/1 00:00:00 からの通算秒をシステムタイムに変換(AjcTime1970ToSysTime)

形　式 ： BOOL　AjcTime1970ToSysTime（UI　Time1970, LPSYSTEMTIME pStime);
BOOL　AjcTime1970ToSysTimeL(ULL Time1970, LPSYSTEMTIME pStime);

引　数 ： Time1970 - 1970/1/1 00:00:00 からの経過秒数（32Bit 時は、最大 2106/02/07 06:28:14）
pStime　- 日付／時刻を格納するバッファのアドレス

説　明 ： 1970/1/1 00:00:00 からの経過秒数で表わされた現在時刻（最大 0xFFFFFFFE=2106/02/07 06:28:14）を、「年月日・時分秒」に変換して、pStime で示すバッファに格納します。曜日(wDayOfWeek)も設定します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

43.1.2. システムタイムを 1970/1/1 00:00:00 からの通算秒に変換(AjcSysTimeToTime1970)

形　式 ： UI　AjcSysTimeToTime1970（const SYSTEMTIME *pStime);
ULL AjcSysTimeToTime1970L(const SYSTEMTIME *pStime);

引　数 ： pStime　- 経過秒数に変換する日時（1970 年以上、32Bit 時は、最大 2106/02/07 06:28:14 -> 0xFFFFFFFE）

説　明 ： pStime で示される日時を、1970/1/1 00:00:00 からの経過秒数に変換します。
pStime で示される日時の妥当性はチェックしません。

戻り値 ： ≠-1 - 1970/1/1 00:00:00 からの経過秒数
＝-1 - 失敗

43.1.3. システムタイムをローカルタイムに変換(AjcSysTimeToLocalTime)

形　式 ： BOOL　AjcSysTimeToLocalTime（const SYSTEMTIME *pStime, LPSYSTEMTIME pLtime);

引　数 ： pStime　- 現在時刻（ＵＴＣ：世界標準時）
pLtime　- ローカルタイムを格納するバッファのアドレス

説　明 ： pStime で示される世界標準時を、ローカルタイムに変換し、pLTime で示すバッファに格納します。
pStime と pLtime は、同じ実体をポイントしていてもＯＫです。

戻り値 ： TRUE　- 成功
FALSE - 失敗

43.1.4. 指定秒数進めた（あるいは戻した）日時を求める(AjcTimeUpdate)

形　式 ： BOOL AjcTimeUpdate(const SYSTEMTIME *pStime, LPSYSTEMTIME pUpdated, int Second);

引　数 ： pStime - 更新対象日時
pLtime - 更新した日時を格納するバッファのアドレス
Second - 更新秒数（時刻を戻す場合は負数を指定）

説　明 ： pStime で示される時刻から、Sensnd で示される秒数進めた（あるいは戻した）時刻を、pUpdated で示されるバッファに格納します。
pStime と pUpdated は、同じ実体をポイントしていてもＯＫです。

戻り値 ： TRUE - 成功
FALSE - 失敗

43.1.5. 日付文字列取得(AjcDateStr)

形　式 ： UTP AjcDateStr(const SYSTEMTIME *pStime);

引　数 ： pStime - 日付情報のアドレス

説　明 ： pStime で示される日付情報から文字列を作成して、その文字列へのポインタを返します。
日付の順序（年月日の並び）と、区切り記号は、ロケール情報によります。
日本では、"yyyy/mm/dd"形式の１０文字となります。
文字列の終端には、0x00 が付加されます。

戻り値 ： ≠NULL - 成功（作成した日付文字列へのポインタ）
＝NULL - 失敗

43.1.6. 時刻文字列取得(AjcTimeStr)

形　式 ： UTP AjcTimeStr(const SYSTEMTIME *pStime, BOOL fMilliseconds);

引　数 ： pStime - 時刻情報のアドレス
fMilliseconds - ミリ秒単位の時刻を付加するか否かのフラグ（FALSE:付加しない, TRUE:付加する）

説　明 ： pStime で示される時刻情報から文字列を作成して、その文字列へのポインタを返します。
時刻の区切り記号は、ロケール情報によります。
日本では、"hh:mm:ss"形式の８文字、あるいは、"hh:mm:ss.xxx"形式の１２文字となります。
文字列の終端には、0x00 が付加されます。

戻り値 ： ≠NULL - 成功（作成した時刻文字列へのポインタ）
＝NULL - 失敗

43.1.7. 日付文字列から SYSTEMTIME 構造体の日付情報を設定(AjcSetDateFromText[Ex])

形　式 ： BOOL AjcSetDateFromText (C_UTP pDateTxt, LPSYSTEMTIME pSt, UTP *pNext);
BOOL AjcSetDateFromTextEx(C_UTP pDateTxt, LPSYSTEMTIME pSt, UTP *pNext, UI flag. UT dlm);

引　数 ： pDateText - 日付テキストのアドレス（ex. “2022/12/23”）
pSt - 日付情報を設定する SYSTEMTIME 構造体のアドレス
pNext - 日付テキストの直後のアドレスを格納するポインタ変数のアドレス（不要時は NULL, pDateText と重複可）
flag - 以下のフラグの合成値を指定（AjcSetDateFromText()の場合は、下線の項目を仮定）
AJCDATEFLAG_YYMMDD ： 年月日の順で指定
AJCDATEFLAG_MMDDYY ： 月日年の順で指定
AJCDATEFLAG_DDMMYY ： 日月年の順で指定
AJCDATEFLAG_MMYYDD ： 月年日の順で指定
} いずれか１つを指定
AJCDATEFLAG_ADJUST ： 年＝1970～2105、月＝01～12、日＝01～28,29,30 or 31 の範囲内に補正
AJCDATEFLAG_SETDEF ： 未指定の項目に規定値（年=2000, 月=01, 日=01）を設定する
AJCDATEFLAG_SETDOW ： 曜日(wDayOfWeek)を設定する
dlm - 年月日の区切り文字（AjcSetDateFromText()の場合や0指定時は、'/' を仮定, 英数字, 空白／タブ／'*' は不可）

説　明 ： pDateText で示される日付文字列をを評価して、SYSTEMTIME 構造体の年月日を設定します。
文字列先頭部分の空白／タブは無視され、評価中に空白／タブを検出すると評価を終了します。
年は２桁あるいは４ケタの１０進数で、月日は２桁数の１０進数で指定します。
年を２桁で指定した場合は、70～99 → 1970～1999, 00～69 → 2000～2069 となります。
年月日のいずれに ”**” / ”****” を指定した場合当該項目を構造体へ設定しません。
ex. “2022/**/31” → 「月」は設定しない
年月日の項目が３つに満たない場合、未指定の項目は「AJCDATEFLAG_SETDEF」の指定により設定／未設定となります。
ex. “2022/12” → 2022 年 12 月 01 日（AJCDATEFLAG_SETDEF 指定あり, 日=１を設定）
2022 年 12 月 **日（AJCDATEFLAG_SETDEF 指定なし, 日は未設定）

戻り値 ： TRUE - 成功
FALSE - 失敗（pSt で示される構造体の内容は変更されません, *pNext はエラー発生個所を示す）

43.1.8. 時刻文字列から SYSTEMTIME 構造体の時刻情報を設定(AjcSetTimeFromText)

形　式 ： BOOL AjcSetTimeFromText (C_UTP pTimeText LPSYSTEMTIME pSt, UTP *pNext);
BOOL AjcSetTimeFromTextEx(C_UTP pTimeText LPSYSTEMTIME pSt, UTP *pNext, UI flag, UT dlm, UT dms);

引　数 ： pTimeText - 時刻テキストのアドレス（ex. “13:23:51.345”）
pSt - 時刻情報を設定する SYSTEMTIME 構造体のアドレス
pNext - 時刻テキストの直後のアドレスを格納するポインタ変数のアドレス（不要時は NULL, pTimeText と重複可）
flag - 以下のフラグの合成値か０を指定（AjcSetDateFromText()の場合は、下線の項目を仮定）
AJCTIMEFLAG_ADJUST ： 時＝00～23、分＝00～59、秒＝00～58, ミリ秒=000～999 の範囲内に補正
AJCTIMEFLAG_SETDEF ： 未指定の項目に規定値（０）を設定する
dlm - 時分秒の区切り文字（AjcSetTimeFromText()の場合や0指定時は、':' を仮定, 英数字／空白／タブ／'*' は不可）
dms - ミリ秒直前の区切り記号（AjcSetTimeFromText()の場合や0指定時は、'.' を仮定, 英数字／空白／タブ／'*' は不可）

説　明 ： pTimeText で示される日付文字列をを評価して、SYSTEMTIME 構造体の時分秒, ミリ秒を設定します。
文字列先頭部分の空白／タブは無視され、評価中に空白／タブを検出すると評価を終了します。
時分秒は２桁の１０進数, ミリ秒は３桁の１０進数で指定します。
時分秒のいずれかに ”**”, ミリ秒に “***” を指定した場合当該項目を構造体へ設定しません。
ex. “13:**:51.345” → 「分」は設定しない
時分秒. ミリ秒の指定が４ヶに満たない場合は、「AJCTIMEFLAG_SETDEF」の指定により設定／未設定となります。
ex. “15:23” → 15:23:00.000（AJCDATEFLAG_SETDEF 指定あり, 秒=0,ミリ秒=0 を設定）
15:23:**.***（AJCDATEFLAG_SETDEF 指定なし, 秒とミリ秒は未設定）

戻り値 ： TRUE - 成功
FALSE - 失敗（pSt で示される構造体の内容は変更されません, *pNext はエラー発生個所を示す）

43.1.9. 日時文字列から SYSTEMTIME 構造体の日時情報を設定(AjcSetDateAndTimeFromText[Ex])

形　式 ： BOOL AjcSetDateAndTimeFromText (C_UTP pText, LPSYSTEMTIME pSt, UTP *pNext);
BOOL AjcSetDateAndTimeFromTextEx(C_UTP pText, LPSYSTEMTIME pSt, UTP *pNext, UI flag, dlm_d, UT dlm_t, UT dms);

引　数 ： pText - 日時テキストのアドレス（ex. “2022/12/23 13:23:51.345”）
pSt - 日時情報を設定する SYSTEMTIME 構造体のアドレス
pNext - 日時テキストの直後のアドレスを格納するポインタ変数のアドレス（不要時は NULL, pTimeText と重複可）
flag - 以下のフラグの合成値を指定（AjcSetDateFromText()の場合は、下線の項目を仮定）

AJCDATEFLAG_YYMMDD ： 年月日の順で指定
AJCDATEFLAG_MMDDYY ： 月日年の順で指定
AJCDATEFLAG_DDMMYY ： 日月年の順で指定
AJCDATEFLAG_MMYYDD ： 月年日の順で指定
} いずれか１つを指定

AJCDATEFLAG_ADJUST ： 年＝1970～2105、月＝01～12、日＝01～28,29,30 or 31 の範囲内に補正
AJCDATEFLAG_SETDEF ： 未指定の項目に規定値（年=2000, 月=01, 日=01）を設定する
AJCDATEFLAG_SETDOW ： 曜日(wDayOfWeek)を設定する

AJCTIMEFLAG_ADJUST ： 時＝00～23、分＝00～59、秒＝00～58, ミリ秒=000～999 の範囲内に補正
AJCTIMEFLAG_SETDEF ： 未指定の時刻項目に規定値（０）を設定する

dlm_d - 年月日の区切り文字（AjcSetDateAndTimeFromText ()の場合は、’/’ を仮定, 英数字, 空白／タブ／’*’ は不可）
dlm_t - 時分秒の区切り文字（AjcSetDateAndTimeFromText ()の場合は、’:’ を仮定, 英数字／空白／タブ／’*’ は不可）
dms - ミリ秒直前の区切り記号（AjcSetTimeFromText()の場合は、’.’ を仮定, 英数字／空白／タブ／’*’ は不可）

説　明 ： pText で示される日時文字列をを評価して、SYSTEMTIME 構造体の年月日時分秒を設定します。
文字列先頭部分の空白は無視されます。日付と時刻は空白で分離して指定します。
日付, 時刻の評価は、AjcSetTimeFromText[Ex], AjcSetTimeFromText[Ex]と同じです。

戻り値 ： TRUE - 成功
FALSE - 失敗（pSt で示される構造体の内容は変更されません, *pNext はエラー発生個所を示す）

43.1.10. ランダムな日付と時刻を取得(AjcGetRandomDateAndTime)

形　式 ： BOOL AjcGetRandomDateAndTime (LPSYSTEMTIME pSt, LPSYSTEMTIME pStSrt, LPSYSTEMTIME pStEnd);

引　数 ： pSt - ランダムな日時情報を設定する SYSTEMTIME 構造体のアドレス
pStSrt - ランダムな日付と時刻の開始を示す SYSTEMTIME 構造体のアドレス（NULL 指定時は１年前の日時）
pStEnd - ランダムな日付と時刻の終了を示す SYSTEMTIME 構造体のアドレス（NULL 指定時は現在日時）

説　明 ： pStSrt～pStEnd の範囲で、ランダムな日付と時刻を取得します。
日付と時刻の範囲は、各々別々で範囲を認識します。(日付は日付の範囲, 時刻は時刻の範囲)
pStEnd で示される日時は、pStSrt で示される日時より大きいか等しくなければなりません。(pStSrt ≦ pStEnd)

戻り値 ： TRUE - 成功
FALSE - 失敗

43.2. サンプルプログラム

43.2.1. SW_DateAndTime1（日時テキストの評価）

日時表現のテキストを評価し、SYSTEMTIME 構造体に設定します。
入力テキストに日時表現のテキストを入力し、実行ボタンを押します。



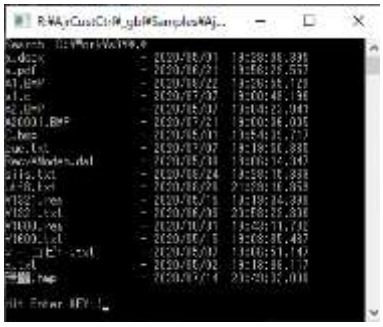

```
 1 : //
 2 : //  SW_DateAndTime1.c
 3 : //
 4 :
 5 : #include    <AjrCstXX.h>
 6 : #include    <math.h>
 7 : #include    <tchar.h>
 8 : #include    "resource.h"
 9 :
10 : //-------------------------------------------------------------------------------------------------------------------//
11 : //  ワーク                                                                                                           //
12 : //-------------------------------------------------------------------------------------------------------------------//
13 : static  HINSTANCE       hInst;                          //  ＤＬＬインスタンスハンドル
14 : static  HWND            hDlgMain;                       //  ダイアログボックスハンドル
15 : static  SYSTEMTIME      SysTime;                        //  SYSTEMTIME 構造体領域
16 : //-------------------------------------------------------------------------------------------------------------------//
17 : //  内部サブ関数                                                                                                     //
18 : //-------------------------------------------------------------------------------------------------------------------//
19 : AJC_DLGPROC_DEF(Main);
20 :
21 : //===================================================================================================================//
22 : //                                                                                                                   //
23 : //  ＷｉｎＭａｉｎ                                                                                                  //
24 : //                                                                                                                   //
25 : //===================================================================================================================//
26 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
27 : {
28 :     MSG     msg;
29 :
30 :     hInst = hInstance;
31 :
32 :     //----- メイン・ダイアログオープン -----------------//
33 :     hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_DLGMAIN), NULL, AJC_DLGPROC_NAME(Main));
34 :     //----- ダイアログ表示 -----------------------------//
35 :     ShowWindow(hDlgMain, SW_SHOW);
36 :
37 :     //----- メッセージループ ---------------------------//
38 :     while (GetMessage(&msg, NULL, 0, 0)) {
39 :         do {
40 :             if (IsDialogMessage(hDlgMain, &msg)) break;
41 :             TranslateMessage(&msg);
42 :             DispatchMessage (&msg);
43 :         } while (0);
44 :     }
45 :
46 :     return (int)msg.wParam ;
47 : }
48 : //===================================================================================================================//
49 : //                                                                                                                   //
50 : //  ダイアログ・プロシージャ                                                                                         //
51 : //                                                                                                                   //
52 : //===================================================================================================================//
53 : //----- ダイアログ初期化 -----------------------------------------------------------------------------------------//
54 : AJC_DLGPROC(Main, WM_INITDIALOG                        )
55 : {
```

```
 56 :     hDlgMain = hDlg;
 57 :     //  テキスト長制限
 58 :     AjcSetDlgItemEdtLimit(hDlg, IDC_TXT_DLMDATE, 1);
 59 :     AjcSetDlgItemEdtLimit(hDlg, IDC_TXT_DLMTIME, 1);
 60 :     AjcSetDlgItemEdtLimit(hDlg, IDC_TXT_DLMMS  , 1);
 61 :     //  デフォルト設定
 62 :     AjcSetDlgItemChk(hDlg, IDC_RBT_AjcSetDateFromText, TRUE);
 63 :     AjcSetDlgItemChk(hDlg, IDC_RBT_DATEFLAG_YYMMDD   , TRUE);
 64 :     AjcSetDlgItemChk(hDlg, IDC_CHK_DATEFLAG_ADJUST   , TRUE);
 65 :     AjcSetDlgItemChk(hDlg, IDC_CHK_DATEFLAG_SETDEF   , TRUE);
 66 :     AjcSetDlgItemChk(hDlg, IDC_CHK_DATEFLAG_SETDOW   , TRUE);
 67 :     AjcSetDlgItemChk(hDlg, IDC_CHK_TIMEFLAG_ADJUST   , TRUE);
 68 :     AjcSetDlgItemChk(hDlg, IDC_CHK_TIMEFLAG_SETDEF   , TRUE);
 69 :     AjcSetDlgItemStr(hDlg, IDC_TXT_DLMDATE           , TEXT("/"));
 70 :     AjcSetDlgItemStr(hDlg, IDC_TXT_DLMTIME           , TEXT(":"));
 71 :     AjcSetDlgItemStr(hDlg, IDC_TXT_DLMMS             , TEXT("."));
 72 :     //  設定値ロード
 73 :     AjcLoadAllControlSettings(hDlg, TEXT("MySect"), AJCCTL_SELACT_ALL | AJCCTL_SELACT_NTCRBT);
 74 :
 75 :     return TRUE;
 76 : }
 77 : //----- ウインド破棄 ----------------------------------------------------------------------------------------//
 78 : AJC_DLGPROC(Main, WM_DESTROY                            )
 79 : {
 80 :     //  設定値セーブ
 81 :     AjcSaveAllControlSettings(hDlg);
 82 :     //  プログラム終了
 83 :     PostQuitMessage(0);
 84 :     return TRUE;
 85 : }
 86 : //-----  「Cancel」ボタン ----------------------------------------------------------------------------------//
 87 : AJC_DLGPROC(Main, IDCANCEL                             )
 88 : {
 89 :     DestroyWindow(hDlg);
 90 :     return TRUE;
 91 : }
 92 : //----- ラジオボタン (IDC_RBT_AjcSetDateFromText) ------------------------------------------------------------//
 93 : AJC_DLGPROC(Main, IDC_RBT_AjcSetDateFromText           )
 94 : {
 95 :     AjcEnableDlgGroup(hDlg, IDC_GRP_FLG, FALSE, FALSE);
 96 :     return TRUE;
 97 : }
 98 : //----- ラジオボタン (IDC_RBT_AjcSetTimeFromText) ------------------------------------------------------------//
 99 : AJC_DLGPROC(Main, IDC_RBT_AjcSetTimeFromText           )
100 : {
101 :     AjcEnableDlgGroup(hDlg, IDC_GRP_FLG, FALSE, FALSE);
102 :     return TRUE;
103 : }
104 : //----- ラジオボタン (IDC_RBT_AjcSetDateAndTimeFromText) -----------------------------------------------------//
105 : AJC_DLGPROC(Main, IDC_RBT_AjcSetDateAndTimeFromText    )
106 : {
107 :     AjcEnableDlgGroup(hDlg, IDC_GRP_FLG, FALSE, FALSE);
108 :     return TRUE;
109 : }
110 : //----- ラジオボタン (IDC_RBT_AjcSetDateFromTextEx) ----------------------------------------------------------//
111 : AJC_DLGPROC(Main, IDC_RBT_AjcSetDateFromTextEx         )
112 : {
113 :     AjcEnableDlgGroup(hDlg, IDC_GRP_FLG    , TRUE , TRUE );
114 :     AjcEnableDlgGroup(hDlg, IDC_GRP_FLGDATE, TRUE , TRUE );
115 :     AjcEnableDlgGroup(hDlg, IDC_GRP_FLGTIME, FALSE, FALSE);
116 :     return TRUE;
117 : }
118 : //----- ラジオボタン (IDC_RBT_AjcSetTimeFromTextEx) ----------------------------------------------------------//
119 : AJC_DLGPROC(Main, IDC_RBT_AjcSetTimeFromTextEx          )
120 : {
121 :     AjcEnableDlgGroup(hDlg, IDC_GRP_FLG    , TRUE , TRUE );
122 :     AjcEnableDlgGroup(hDlg, IDC_GRP_FLGDATE, FALSE, FALSE);
123 :     AjcEnableDlgGroup(hDlg, IDC_GRP_FLGTIME, TRUE , TRUE );
124 :     return TRUE;
125 : }
126 : //----- ラジオボタン (IDC_RBT_AjcSetDateAndTimeFromTextEx) ---------------------------------------------------//
127 : AJC_DLGPROC(Main, IDC_RBT_AjcSetDateAndTimeFromTextEx   )
128 : {
129 :     AjcEnableDlgGroup(hDlg, IDC_GRP_FLG    , TRUE , TRUE );
130 :     return TRUE;
131 : }
132 : //----- 構造体に日時設定ボタン ------------------------------------------------------------------------------//
133 : AJC_DLGPROC(Main, IDC_CMD_SETSYSTEMTIME                 )
134 : {
135 :     UT      txt[256];
136 :
137 :     SysTime.wYear         = (UW)AjcGetDlgItemUInt(hDlg, IDC_TXT_YEAR        );
138 :     SysTime.wMonth        = (UW)AjcGetDlgItemUInt(hDlg, IDC_TXT_MONTH       );
139 :     SysTime.wDay          = (UW)AjcGetDlgItemUInt(hDlg, IDC_TXT_DAY         );
140 :     SysTime.wDayOfWeek    = (UW)AjcGetDlgItemUInt(hDlg, IDC_TXT_DAYOFWEEK   );
141 :     SysTime.wHour         = (UW)AjcGetDlgItemUInt(hDlg, IDC_TXT_HOUR        );
142 :     SysTime.wMinute       = (UW)AjcGetDlgItemUInt(hDlg, IDC_TXT_MINUTE      );
143 :     SysTime.wSecond       = (UW)AjcGetDlgItemUInt(hDlg, IDC_TXT_SECOND      );
144 :     SysTime.wMilliseconds = (UW)AjcGetDlgItemUInt(hDlg, IDC_TXT_MS          );
145 :
```

```
146 :     AjcSnPrintF(txt, AJCTSIZE(txt), TEXT("出力用の構造体領域に日時¥n(%04d/%02d/%02d (曜日=%d), %02d:%02d:%02d.%03d)を設定しました。"),
147 :                 SysTime.wYear, SysTime.wMonth, SysTime.wDay, SysTime.wDayOfWeek,
148 :                 SysTime.wHour, SysTime.wMinute, SysTime.wSecond, SysTime.wMilliseconds);
149 :     AjcTipTextShowCenter(hDlg , txt, 5000, NULL);
150 :
151 :     return TRUE;
152 : }
153 : //----- 実行ボタン ----------------------------------------------------------------------------------------------------//
154 : AJC_DLGPROC(Main, IDOK                                    )
155 : {
156 :     UI      flg = 0;
157 :     UT      dlmDate[16];
158 :     UT      dlmTime[16];
159 :     UT      dlmMs  [16];
160 :     UTP     pNext;
161 :     UT      txt[256];
162 :     BOOL    rsu;
163 :
164 :     if (AjcGetDlgItemChk(hDlg, IDC_RBT_DATEFLAG_YYMMDD  ))  flg |= AJCDATEFLAG_YYMMDD;
165 :     if (AjcGetDlgItemChk(hDlg, IDC_RBT_DATEFLAG_MMDDYY  ))  flg |= AJCDATEFLAG_MMDDYY;
166 :     if (AjcGetDlgItemChk(hDlg, IDC_RBT_DATEFLAG_DDMMYY  ))  flg |= AJCDATEFLAG_DDMMYY;
167 :     if (AjcGetDlgItemChk(hDlg, IDC_RBT_DATEFLAG_MMYYDD  ))  flg |= AJCDATEFLAG_MMYYDD;
168 :     if (AjcGetDlgItemChk(hDlg, IDC_CHK_DATEFLAG_ADJUST  ))  flg |= AJCDATEFLAG_ADJUST;
169 :     if (AjcGetDlgItemChk(hDlg, IDC_CHK_DATEFLAG_SETDEF  ))  flg |= AJCDATEFLAG_SETDEF;
170 :     if (AjcGetDlgItemChk(hDlg, IDC_CHK_DATEFLAG_SETDOW  ))  flg |= AJCDATEFLAG_SETDOW;
171 :     if (AjcGetDlgItemChk(hDlg, IDC_CHK_TIMEFLAG_ADJUST  ))  flg |= AJCTIMEFLAG_ADJUST;
172 :     if (AjcGetDlgItemChk(hDlg, IDC_CHK_TIMEFLAG_SETDEF  ))  flg |= AJCTIMEFLAG_SETDEF;
173 :     AjcGetDlgItemStr(hDlg, IDC_TXT_DLMDATE, dlmDate, AJCTSIZE(dlmDate));
174 :     AjcGetDlgItemStr(hDlg, IDC_TXT_DLMTIME, dlmTime, AJCTSIZE(dlmTime));
175 :     AjcGetDlgItemStr(hDlg, IDC_TXT_DLMMS  , dlmMs  , AJCTSIZE(dlmMs  ));
176 :
177 :     AjcGetDlgItemStr(hDlg, IDC_TXT_INP, txt, AJCTSIZE(txt));
178 :
179 :     if    (AjcGetDlgItemChk(hDlg, IDC_RBT_AjcSetDateFromText      )) rsu = AjcSetDateFromText      (txt, &SysTime, &pNext);
180 :     else if (AjcGetDlgItemChk(hDlg, IDC_RBT_AjcSetDateFromTextEx    )) rsu = AjcSetDateFromTextEx    (txt, &SysTime, &pNext, flg, dlmDate[0]);
181 :     else if (AjcGetDlgItemChk(hDlg, IDC_RBT_AjcSetTimeFromText      )) rsu = AjcSetTimeFromText      (txt, &SysTime, &pNext);
182 :     else if (AjcGetDlgItemChk(hDlg, IDC_RBT_AjcSetTimeFromTextEx    )) rsu = AjcSetTimeFromTextEx    (txt, &SysTime, &pNext, flg, dlmTime[0],
        dlmMs[0]);
183 :     else if (AjcGetDlgItemChk(hDlg, IDC_RBT_AjcSetDateAndTimeFromText )) rsu = AjcSetDateAndTimeFromText (txt, &SysTime, &pNext);
184 :     else if (AjcGetDlgItemChk(hDlg, IDC_RBT_AjcSetDateAndTimeFromTextEx)) rsu = AjcSetDateAndTimeFromTextEx(txt, &SysTime, &pNext, flg, dlmDate[0], dlmTime[0],
        dlmMs[0]);
185 :
186 :     AjcSetDlgItemUInt(hDlg, IDC_TXT_YEAR        , SysTime.wYear);
187 :     AjcSetDlgItemUInt(hDlg, IDC_TXT_MONTH       , SysTime.wMonth);
188 :     AjcSetDlgItemUInt(hDlg, IDC_TXT_DAY         , SysTime.wDay);
189 :     AjcSetDlgItemUInt(hDlg, IDC_TXT_DAYOFWEEK   , SysTime.wDayOfWeek);
190 :     AjcSetDlgItemUInt(hDlg, IDC_TXT_HOUR        , SysTime.wHour);
191 :     AjcSetDlgItemUInt(hDlg, IDC_TXT_MINUTE      , SysTime.wMinute);
192 :     AjcSetDlgItemUInt(hDlg, IDC_TXT_SECOND      , SysTime.wSecond);
193 :     AjcSetDlgItemUInt(hDlg, IDC_TXT_MS          , SysTime.wMilliseconds);
194 :     AjcSetDlgItemUInt(hDlg, IDC_TXT_RC          , rsu);
195 :
196 :     if (pNext != NULL) {
197 :         UX loc = (UX)pNext - (UX)txt;
198 :         AjcSetDlgItemUInt(hDlg, IDC_TXT_LOC     , (UI)loc);
199 :         AjcSetDlgItemStr (hDlg, IDC_TXT_PNEXT   , pNext);
200 :     }
201 :     else {
202 :         AjcSetDlgItemStr (hDlg, IDC_TXT_PNEXT   , TEXT(""));
203 :     }
204 :
205 :     return TRUE;
206 : }
207 :
208 :
209 : //----------------------------------------------------------------------------------------------------------------//
210 : AJC_DLGMAP_DEF(Main)
211 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG                      )
212 :     AJC_DLGMAP_MSG(Main, WM_DESTROY                         )
213 :
214 :     AJC_DLGMAP_CMD(Main, IDCANCEL                           )
215 :     AJC_DLGMAP_CMD(Main, IDC_RBT_AjcSetDateFromText         )
216 :     AJC_DLGMAP_CMD(Main, IDC_RBT_AjcSetTimeFromText         )
217 :     AJC_DLGMAP_CMD(Main, IDC_RBT_AjcSetDateAndTimeFromText  )
218 :     AJC_DLGMAP_CMD(Main, IDC_RBT_AjcSetDateFromTextEx       )
219 :     AJC_DLGMAP_CMD(Main, IDC_RBT_AjcSetTimeFromTextEx       )
220 :     AJC_DLGMAP_CMD(Main, IDC_RBT_AjcSetDateAndTimeFromTextEx)
221 :     AJC_DLGMAP_CMD(Main, IDC_CMD_SETSYSTEMTIME              )
222 :     AJC_DLGMAP_CMD(Main, IDOK                               )
223 : AJC_DLGMAP_END
224 :
```

43.2.2. SW_DateAndTime2（ファイルへランダムなタイムスタンプ設定）

第１パラメタで指定したフォルダ下の全ファイルへランダムなタイムスタンプを設定します。
日付の範囲は、2020/05/01～2020/10/31，時刻の範囲は、19:00:00.000～21:59:59.999 です。



```
 1 : //
 2 : //  SW_DateAndTime2.c
 3 : //
 4 : #include    <AjrCstXX.h>
 5 : #include    <stdio.h>
 6 : #include    <conio.h>
 7 : #include    <time.h>
 8 : #include    <tchar.h>
 9 :
10 : //=============================================================================================================//
11 : int  AjcMain(int argc, UTP argv[])
12 : {
13 :     SYSTEMTIME          st, srt, end;
14 :     UX                  fh;
15 :     struct _tfinddata_t fd;
16 :     UT                  wild[MAX_PATH];
17 :     UT                  path[MAX_PATH];
18 :
19 :     AjcSetStdMode(AJCTEC_UTF_8, FALSE);
20 :
21 :     if (argc == 2 && AjcPathIsDirectory(argv[1])) {
22 :         //  ワイルドカード設定
23 :         AjcSnPrintF(wild, MAX_PATH, TEXT("%s¥¥*.*"), argv[1]);
24 :         AjcPrintF(TEXT("Search  %s¥n"), wild);
25 :         //  設定日時の範囲設定
26 :         srt.wYear         = 2020;   end.wYear         = 2020;
27 :         srt.wMonth        = 5;      end.wMonth        = 10;
28 :         srt.wDay          = 1;      end.wDay          = 31;
29 :         srt.wHour         = 19;     end.wHour         = 21;
30 :         srt.wMinute       = 0;      end.wMinute       = 59;
31 :         srt.wSecond       = 0;      end.wSecond       = 59;
32 :         srt.wMilliseconds = 0;      end.wMilliseconds = 999;
33 :         //  ファイルのタイムスタンプ設定
34 :         if ((fh = _tfindfirst(wild, &fd))  != -1) {
35 :             do {
36 :                 if (!(fd.attrib & _A_SUBDIR)) {
37 :                     AjcSnPrintF(path, MAX_PATH, TEXT("%s¥¥%s"), argv[1], fd.name);
38 :                     AjcGetRandomDateAndTime (&st, &srt, &end);
39 :                     if (AjcSetFileTimeBySysTime(path, &st, TRUE)) {
40 :                         AjcPrintF(TEXT("%-20s - %04d/%02d/%02d  %02d:%02d:%02d.%03d¥n"),
41 :                             fd.name, st.wYear, st.wMonth, st.wDay, st.wHour, st.wMinute, st.wSecond, st.wMilliseconds);
42 :                     }
43 :                     else {
44 :                         AjcPrintF(TEXT("*** file(%s) time-stamp setting failure ***¥n"), path);
45 :                     }
46 :                     Sleep(100);
47 :                 }
48 :             } while (_tfindnext(fh, &fd) != -1);
49 :             _findclose(fh);
50 :         }
51 :     }
52 :     else {
53 :         AjcPrintF(TEXT("フォルダを指定してください"));
54 :     }
55 :
56 :     AjcPrintF(TEXT("¥nHit Enter KEY!!"));
57 :     getchar();
58 :
59 :     return 0;
60 : }
61 :
```

44. 時間計測

マイクロ秒単位の時間計測を行います。
時間の計測は「QueryPerformanceFrequency()」「QueryPerformanceCounter()」により計測しますが、
これらのＡＰＩが使用できない場合は「timeGetTime()」により計測します。

44.1. サポートＡＰＩ

#	関 数 名	内容
1	AjcTimeMeasureStart	時間計測開始（スタティック・インスタンス）
2	AjcTimeMeasureInterval	周期時間計測（スタティック・インスタンス）
3	AjcTimeMeasureElapse	経過時間計測（スタティック・インスタンス）
4	AjcTimeMeasureCreate	時間計測オブジェクト生成
5	AjcMesTimeDelete	時間計測オブジェクト消去
6	AjcMesTimeInterval	周期時間計測
7	AjcMesTimeElapse	経過時間計測

44.1.1. 時間計測開始(AjcTimeMeasureStart)

形　式　：　ULL　AjcTimeMeasureStart(VO);

引　数　：　なし

説　明　：　実行時間の計測を開始します。

戻り値　：　≠０：ハードウェア・カウンタの周波数（μｓ単位での時間計測が可能）
＝０：μｓ単位での時間計測は不可（ｍｓ単位の精度で計測し、μｓ単位の時間に換算）

44.1.2. 周期時間計測(AjcTimeMeasureInterval)

形　式　：　ULL　AjcTimeMeasureInterval(VO);

引　数　：　なし

説　明　：　最後に実行したAjcTimeMeasureStart()／AjcTimeMeasureInterval()からの経過時間を取得し、計測時間をリセットします。
本関数を繰り返し実行することにより、各実行間のラップタイムを計測できます。

戻り値　：　最後に実行したAjcTimeMeasureStart()／AjcTimeMeasureInterval()からの経過時間[μｓ]

44.1.3. 経過時間計測(AjcTimeMeasureElapse)

形　式　：　ULL　AjcTimeMeasureElapse(VO);
ULL　AjcTimeMeasureStop(VO);　-- AjcTimeMeasureElapse()と同じ

引　数　：　なし

説　明　：　最後に実行したAjcTimeMeasureStart()／AjcTimeMeasureInterval()実行後の経過時間を取得します。

戻り値　：　最後に実行したAjcTimeMeasureStart()／AjcTimeMeasureInterval()実行後の経過時間［μｓ］

44.1.4. 時間計測オブジェクト生成(AjcMesTimeCreate)

形　式 ： HAJCMESTIME　AjcMesTimeCreate (ULLP pFreq);

引　数 ： pFreq - 計測周波数[Hz]を格納するバッファのアドレス（不要時はNULL）

説　明 ： 時間計測用のオブジェクトを生成します。
pFreqで示すバッファには、ハードウェア・カウンタの周波数が設定されます。
*pFreqに0以外が設定された場合は、μｓ単位での時間計測が可能であることを意味します。
*pFreqに0が設定された場合は、μｓ単位での時間計測が不能であることを示します。この場合、ｍｓ単位の精度で計測し、μｓ単位の時間に換算します。

戻り値 ： ≠NULL - 成功（インスタンスハンドル）
＝NULL - 失敗

44.1.5. 時間計測オブジェクト消去(AjcMesTimeDelete)

形　式 ： VO　AjcMesTimeDelete(HAJCMESTIME hMesTime);

引　数 ： hMesTime -　AjcTimeMeasureCreate()で取得したインスタンスハンドル

説　明 ： 時間計測用のオブジェクト・リソースを消去します。

戻り値 ： なし

44.1.6. 周期時間計測(AjcMesTimeInterval)

形　式 ： ULL　AjcMesTimeInterval (HAJCMESTIME hMesTime);

引　数 ： hMesTime -　AjcTimeMeasureCreate()で取得したインスタンスハンドル

説　明 ： AjcTimeMeasureCreate()、あるいは、最後に実行したAjcMesTimeInterval()からの経過時間を取得し、計測時間をリセットします。
本関数を繰り返し実行することにより、各実行間のラップタイムを計測できます。

戻り値 ： AjcTimeMeasureCreate()、あるいは、最後に実行したAjcMesTimeInterval()からの経過時間［μｓ］

44.1.7. 経過時間計測(AjcTimeMeasureElapse)

形　式 ： ULL　AjcMesTimeElapse(HAJCMESTIME hMesTime);

引　数 ： hMesTime -　AjcTimeMeasureCreate()で取得したインスタンスハンドル

説　明 ： AjcTimeMeasureCreate()、あるいは、最後に実行したAjcMesTimeInterval()からの経過時間を取得します。

戻り値 ： AjcTimeMeasureCreate()、あるいは、最後に実行したAjcMesTimeInterval()からの経過時間［μｓ］

45. イメージの描画

様々な形式のイメージデータ（「.bmp」や「.jpg」等）のイメージを描画します。
この機能は、VisualStudio2005 以降でのみ使用できます。

45.1. サポートＡＰＩ

イメージの描画のサポートＡＰＩ一覧を以下に示します。

#	関 数 名	内容
1	AjcImgFuncRead	ファイルからイメージデータ読み出し
2	AjcImgFuncDraw	イメージ描画
3	AjcImgFuncRelease	イメージデータ解放

45.1.1. ファイルからイメージデータ読み出し (AjcImgFuncRead)

形　式 ： BOOL　AjcImgFuncRead(PAJC_IMGINFO　　pImgInfo, C_UTP pFilePath)

引　数 ： pImgInfo　－　読み出したイメージデータ情報を格納するバッファのアドレス
pFilePath －　イメージデータファイルのパス名へのポインタ

説　明 ： ファイルからイメージデータを読み出し、確保したメモリに置きます。
読み出し可能なイメージファイルの形式は、以下のとおりです。

.bmp　－　ビットマップ形式
.jpg　－　JPEG 圧縮形式
.png　－　Deflate 圧縮形式
.gif　－　GIF 規格の圧縮形式

読み出したイメージを描画するには、「AjcImgFuncDraw」関数を使用します。
読み出したイメージの使用終了時には、「AjcImgFuncRelease」関数でイメージデータを解放してください。

戻り値 ： TRUE　－ ＯＫ
FALSE － 失敗

備　考 ： pImgInfo で示される構造体の形式は、以下の通りです。

```
typedef struct {
    BOOL        fValid;                 //  有効フラグ（イメージ読み出し済フラグ）
    VOP         hFactory;               //  IShellImageDataFactory  インタフェース
    VOP         hImg;                   //  IShellImageData         インタフェース
    HBITMAP     hBmp;                   //  ビットマップ・ハンドル（ＤＩＢセクション）
    int         width;                  //  イメージの横サイズ
    int         height;                 //  イメージの縦サイズ
} AJC_IMGINFO, *PAJC_IMGINFO;
typedef const AJC_IMGINFO *PCAJC_IMGINFO;
```

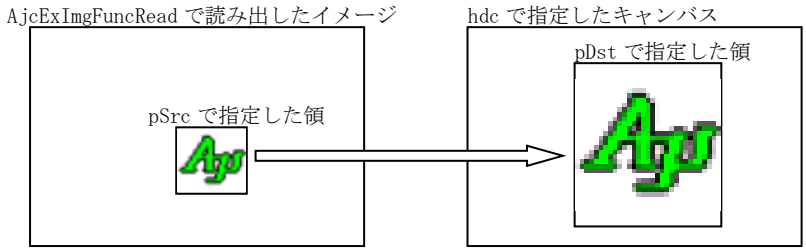
拡張子が「.bmp」である場合は、ビットマップファイルの読み出しを試み、成功した場合は、「hBmp」にビットマップハンドルが設定されます。
拡張子が「.bmp」以外である場合や、ビットマップファイルの読み出しを失敗した場合（圧縮されたビットマップの場合）は、IShellImageData インタフェースによりイメージデータの読み出しを行い、成功した場合は hFactory にインタフェースハンドルが設定されます。
いずれの読み出しも失敗した場合は、「fValid=FALSE」が設定され、「hFactory=NULL」「hBmp=NULL」が設定されます。

45.1.2. イメージ描画 (AjcImgFuncDraw)

形　式 ： BOOL AjcImgFuncDraw(PAJC_IMGINFO pImgInfo, LPRECT pSrc, HDC hdc, LPRECT pDst)

引　数 ： pImgInfo - イメージデータ情報のアドレス
pSrc - イメージデータの描画部分（長方形領域）
hdc - 描画先のデバイスコンテキスト(DC)
pDst - 描画先の描画部分（長方形領域）

説　明 ： 「AjcImgFuncRead」により読み出したイメージデータの一部、あるいは全部を描画します。



戻り値 ： TRUE - 成功
FALSE - 失敗

45.1.3. イメージデータ解放 (AjcImgFuncRelease)

形　式 ： BOOL AjcImgFuncRelease(PAJC_IMGINFO pImgInfo)

引　数 ： pImgInfo - イメージデータ情報のアドレス

説　明 ： 「AjcImgFuncRead」により読み出したイメージデータを解放します。

戻り値 ： TRUE - 成功
FALSE - 失敗

45.2. サンプルプログラム

45.2.1. SW_ShowImage01（イメージファイルの表示）

ファイルからイメージデータを読み出して表示します。
右クリックで、イメージデータファイルを選択します。




```
 1 : //
 2 : //  SW_ShowImage.c
 3 : //
 4 : #include    <AjrCstXX.h>
 5 : #include    <tchar.h>
 6 :
 7 : //-------------------------------------------------------------------------------------------------------------//
 8 : //  作業領域                                                                                                   //
 9 : //-------------------------------------------------------------------------------------------------------------//
10 : static  HWND            hWndMain    = NULL;         //  メインウインド
11 : static  AJC_IMGINFO     ImgInfo;                    //  イメージ情報
12 :
13 : //-------------------------------------------------------------------------------------------------------------//
14 : //  内部サブ関数                                                                                               //
15 : //-------------------------------------------------------------------------------------------------------------//
16 : AJC_WNDPROC_DEF(Main);
17 :
18 : //=============================================================================================================//
19 : //  W i n M a i n                                                                                              //
20 : //=============================================================================================================//
21 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
22 : {
23 :     WNDCLASS    wndclass;
24 :     MSG         msg;
25 :
26 :     memset(&ImgInfo, 0, sizeof ImgInfo);
27 :
28 :     //----- ウインド生成 ----------------------------------------------------------------------------------------//
29 :     wndclass.style          = 0;
30 :     wndclass.lpfnWndProc    = AJC_WNDPROC_NAME(Main);
31 :     wndclass.cbClsExtra     = 0;
32 :     wndclass.cbWndExtra     = 0;
33 :     wndclass.hInstance      = hInstance;
34 :     wndclass.hIcon          = NULL;
35 :     wndclass.hCursor        = LoadCursor(NULL, IDC_ARROW);;
36 :     wndclass.hbrBackground  = (HBRUSH)GetStockObject(WHITE_BRUSH);
37 :     wndclass.lpszMenuName   = NULL;
38 :     wndclass.lpszClassName  = TEXT("CSW_ShowImage");
39 :     RegisterClass (&wndclass);
40 :
41 :     hWndMain = CreateWindow(TEXT("CSW_ShowImage"),    // window class name
42 :                             TEXT("SW_ShowImage"),     // window caption
43 :                             WS_OVERLAPPEDWINDOW,      // style
44 :                             0,                        // initial x position
45 :                             0,                        // initial y position
46 :                             512,                      // initial x size
47 :                             384,                      // initial y size
48 :                             NULL,                     // parent window handle
49 :                             NULL,                     // window menu handle
```

```
50 :                            hInstance,              // program instance handle
51 :                            NULL) ;                 // creation parameters
52 :
53 :    //----- ウインド表示 ----------------------------------------------------------------------------------------//
54 :    ShowWindow(hWndMain, iCmdShow);
55 :
56 :    //----- メッセージループ ------------------------------------------------------------------------------------//
57 :    while (GetMessage(&msg, NULL, 0, 0)) {
58 :        TranslateMessage(&msg);
59 :        DispatchMessage (&msg);
60 :    }
61 :
62 :    return (int)msg.wParam ;
63 : }
64 : //=============================================================================================================//
65 : //                                                                                                             //
66 : //  ウインド・プロシージャ                                                                                     //
67 : //                                                                                                             //
68 : //=============================================================================================================//
69 : AJC_WNDPROC(Main, WM_CREATE         )
70 : {
71 :    hWndMain   = hwnd;
72 :
73 :    return 0;
74 : }
75 : //-------------------------------------------------------------------------------------------------------------//
76 : AJC_WNDPROC(Main, WM_DESTROY        )
77 : {
78 :    AjcImgFuncRelease(&ImgInfo);                        //  読み出し済のイメージ解放
79 :    PostQuitMessage(0);
80 :
81 :    return 0;
82 : }
83 : //-------------------------------------------------------------------------------------------------------------//
84 : AJC_WNDPROC(Main, WM_SIZE           )
85 : {
86 :    InvalidateRect(hwnd, NULL, FALSE);
87 :    return 0;
88 : }
89 : //-------------------------------------------------------------------------------------------------------------//
90 : AJC_WNDPROC(Main, WM_RBUTTONDOWN    )      //  右クリックで、表示するイメージファイルを選択
91 : {
92 :    static  UT  path[MAX_PATH] = {0};
93 :    UT          txt[MAX_PATH + 64];
94 :
95 :    if (AjcGetOpenFile(hWndMain, TEXT("イメージファイル設定"),
96 :                    TEXT("All Files (*.*)/*.*/Bitmap File (*.bmp)/*.bmp/.jpg File/*.jpg/.png File/*.png/.gif File/*.gif"),
97 :                    TEXT("bmp"), path, MAX_PATH)) {
98 :
99 :        AjcSnPrintF(txt, MAX_PATH, TEXT("SW_ShowImage (%s)"), path); //  ウインドタイトル表示
100 :       AjcSetWindowText(hwnd, txt);                                 //  ・
101 :       AjcImgFuncRelease(&ImgInfo);                                 // 読み出し済のイメージ解放
102 :       AjcImgFuncRead   (&ImgInfo, path);                           // イメージファイル読み出し
103 :       InvalidateRect(hwnd, NULL, TRUE);                            // 再描画
104 :   }
105 :   return 0;
106 : }
107 : //-------------------------------------------------------------------------------------------------------------//
108 : AJC_WNDPROC(Main, WM_PAINT          )
109 : {
110 :    HDC         hdc;
111 :    PAINTSTRUCT ps;
112 :    RECT        rcS, rcR, rcB;
113 :
114 :    hdc = BeginPaint(hwnd, &ps);
115 :    if (ImgInfo.fValid) {
116 :        SetRect(&rcS, 0, 0, ImgInfo.width, ImgInfo.height);     //  入力側矩形
117 :        SetRect(&rcR, 0, 0, 500          , 500           );     //  出力側矩形
118 :        AjcAspGetZoomedRect(&rcS, &rcR, &rcB);                  //  拡大縮小した矩形算出
119 :        AjcImgFuncDraw(&ImgInfo, &rcS, hdc, &rcB);              //  イメージ描画
120 :        AdjustWindowRect(&rcB, WS_OVERLAPPEDWINDOW, FALSE);     //  ウインドサイズをイメージに合わせる
121 :        SetWindowPos(hwnd, NULL, 0, 0, rcB.right - rcB.left, rcB.bottom - rcB.top, SWP_NOMOVE);
122 :    }
123 :    else {
124 :        UTP     p =  TEXT("右クリックで表示するイメージファイルを選択してください。");
125 :        TextOut(hdc, 10, 10, p,  (UI)MAjcStrLen(p));
126 :    }
127 :    EndPaint(hwnd, &ps);
128 :
129 :    return 0;
```

```
130 : }
131 : //-----------------------------------------------------------------------------------------------------------//
132 : AJC_WNDMAP_DEF(Main)
133 :     AJC_WNDMAP_MSG(Main, WM_CREATE      )
134 :     AJC_WNDMAP_MSG(Main, WM_DESTROY     )
135 :     AJC_WNDMAP_MSG(Main, WM_SIZE        )
136 :     AJC_WNDMAP_MSG(Main, WM_RBUTTONDOWN )
137 :     AJC_WNDMAP_MSG(Main, WM_PAINT       )
138 : AJC_WNDMAP_END
139 :
```

46. ＬＺＨ書庫データのアクセス

ＬＨＡ等のプログラムにより圧縮された書庫ファイル（.LZH ファイル）へのアクセスを行ないます。
本ライブラリでは、ＬＺＨ書庫ファイルに対して、以下の操作が可能です。

・ＬＺＨ書庫ファイル内に格納されているファイル名の検索
・ＬＺＨ書庫ファイル内に格納されているファイルを復元（解凍）しながら読み出す

ファイル名の検索では、ワイルドカードによる複数ファイル検索も可能です。

ファイルの復元（解凍）では、書庫内のファイルを通常のファイルアクセス（fopen～fread～fclose）のように操作可能です。
但し、ＳＥＥＫ操作（アクセスバイト位置の変更）や、書き込み操作はできません。シーケンシャルな読み出し操作のみとなります。

サポートするＬＺＨファイルフォーマットとヘッダ形式は、以下のとおりです。

・ＬＺＨファイルフォーマット　：　"-lh0-"　（非圧縮）
　　"-lh5-"　（８ＫＢのスライディング辞書による圧縮）
・ヘッダ形式　：　レベル０～２

４ＧＢ以上のＬＺＨ書庫ファイルは、アクセスできません。

エラーコード（AJCLZHERR ）

#	名　称	内　容	備　考
1	AJCLZHERR _OK	正常（＝０）	
2	AJCLZHERR _UNARYCODE_OUTOFRANGE	単進符号値が１７以上である	
3	AJCLZHERR _ILLEGAL_BITLEN	不正ビット長	
4	AJCLZHERR _BLEN_OVER19	ビット長のビット長テーブルのエントリ数が１９ヶをオーバー	
5	AJCLZHERR _BLEN_OVER510	ビット長テーブルのエントリ数が５１０ヶをオーバー	
6	AJCLZHERR _ILLEGAL_FORMAT	"-lh0-"，"-lh5-"，"-lhd-" 以外のフォーマット	
7	AJCLZHERR _ILLEGAL_HDLVL	ヘッダレベルが０～２以外	
8	AJCLZHERR _NOTFOUND	ファイルが見つからない	
9	AJCLZHERR_MEMORY	メモリ不足	
10	AJCLZHERR_INVALIDMODE	不正モード	

46.1. サポートＡＰＩ

ＬＺＨ書庫データのアクセスのサポートＡＰＩ一覧を以下に示します。

#	関数名	内容	備考
1	AjcLzhFindFirst	書庫内のファイル名検索（初回）	
2	AjcLzhFindNext	書庫内のファイル名検索（２回目以降）	
3	AjcLzhOpen	書庫内のファイルオープン	
4	AjcLzhRead	書庫内のファイル読み出し	
5	AjcLzhClose	書庫アクセスのクローズ	
6	AjcLzhGetFileInfo	検索したファイル情報取得	
7	AjcLzhGetLastError	最後に発生したエラーコードの取得	

46.1.1. 書庫内のファイル名検索［初回］（AjcLzhFindFirst）

形　式 ： HAJCLZH　AjcLzhFindFirst(C_UTP pWild, UI flag, UX xp, UI (CALLBACK *cbRead)(UX xp, VOP pBuf, UI len),
VO (CALLBACK *cbSeek)(UX xp, UL ofs));

引　数 ： pWild　　－ 検索するファイル名（'*' や'?' を用いたワイルドカードも可，パスの区切り文字は '¥' or '/' ）
flag　　－ 動作指定フラグ（AJCLZHF_SUBDIR / AJCLZHF_SRH_DIR / AJCLZHF_NOSRH_FILE）
xp　　－ コールバックパラメタ
cbRead　　－ ＬＺＨ書庫ファイル入力用コールバック関数のアドレス
cbSeek　　－ ＬＺＨ書庫ファイルのバイト位置設定用コールバック関数のアドレス

説　明 ： 書庫内のファイル／ディレクトリ検索を開始します。
pWild は、検索するファイル名（'*' や'?' を用いたワイルドカードも可）を指定します。
pWild の先頭部分にディレクトリ名を指定した場合は、当該ディレクトリ下のファイル／ディレクトリを検索します。
pWild の先頭部分にディレクトリ名を指定する場合は、先頭のディレクトリから指定しなければなりません。
例えば、「topdir¥subdir¥*.txt」を指定した場合は、「topdir¥subdir¥」下の検索となります。
pWild の先頭部分にディレクトリ名を指定しない場合は、先頭ディレクトリ下の検索となります。(ex.　「*.txt」)
書庫内に、pWild で指定されたファイル／ディレクトリが見つかった場合は、ＬＺＨ検索ハンドルを返します。

flag は、ファイル／ディレクトリの検索ににおける動作を指定します。(以下のシンボルの組み合わせ)

シンボル	内容
AJCLZHF_SUBDIR	サブディレクトリ下のファイル（やディレクトリ）も検索する
AJCLZHF_SRH_DIR	ディレクトリも検索する　（※１）
AJCLZHF_NOSRH_FILE	ファイルの検索を行わない
AJCLZHF_DIRONLY	ディレクトリだけを検索（AJCLZHF_SUBDIR ｜ AJCLZHF_SRH_DIR ｜ AJCLZHF_NOSRH_FILE）

※１：圧縮ツールによっては、書庫ファイル内のヘッダ情報にディレクトリ情報が作成されていない場合があります。
書庫ファイル内にディレクトリ情報が作成されていないものについては検索されません。
圧縮ツールによっては、ディレクトリが空の場合だけディレクトリ情報が作成されている場合があります。
この場合は、「AJCLZHF_SRH_DIR」を指定しても空のディレクトリだけが検索されることになります。

いずれの動作指定も行わない場合（サブフォルダ中のファイルだけを検索する場合）は flag=0 を指定します。

書庫内のファイル名検索は、AjcLzhFindFirst(), AjcLzhFindNext(), AjcLzhClose()により、以下のような手順で行ないます。

```
HAJCLZH  hLzh;

if ((hLzh = AjcLzhFindFirst(・・・) ) != NULL) {
    do {

        /*- 見つかったファイル群の処理 -*/

    } while (AjcLzhFindNext(hLzh));
    AjcLzhClose(hLzh);
}
```

戻り値 ： ≠NULL － ファイルが見つかった（ＬＺＨ検索ハンドル）
＝NULL － 当該ファイルは無い

備　考 ： エラーを取得するには、AjcLzhGetLastError()を実行します

コールバック：コールバック関数の仕様は以下のとおりです。

cbRead（書庫データ読み出し）

形　式　：　UI CALLBACK ***cbRead***(UX xp, VOP pBuf, UI len);

引　数　：　xp　－ コールバックパラメタ
pBuf － 読み出したたデータを格納するバッファのアドレス
len　－ 読み出すバイト数

説　明　：　ＬＺＨ書庫のデータを読み出して、pBuf で示されるバッファに格納します。
このコールバック関数では、ＬＺＨ書庫の読み出しバイト位置を更新します。つまり、(cbSeek により読み出しバイト位置が変更されない限り）次回の cbRead では、今回の読み出したバイトストリームの次のバイトストリームを読み出します。

戻り値　：　実際に読み出したバイト数（書庫の終端を越えた読み出しが要求された場合は、len で指定された値よりも小さい値となります)

cbSeek（書庫データ読み出しバイト位置設定）

形　式　：　VO CALLBACK ***cbSeek***(UX xp, UL ofs);

引　数　：　xp　－ コールバックパラメタ
ofs　－ 設定する読み出しバイト位置

説　明　：　ＬＺＨ書庫データの読み出しバイト位置を設定します。
次回の cbRead では、本コールバックで指定されたバイト位置から、バイトストリームを読み出します。

戻り値　：　なし

46.1.2.　書庫内のファイル名検索［２回目以降］（AjcLzhFindNext）

形　式　：　BOOL　AjcLzhFindNext(HAJCLZH hLzh);

引　数　：　hLzh　　－ ＬＺＨデコードハンドル

説　明　：　AjcLzhFindFirst に続いて、書庫内のファイル検索を継続します。
hLzh は、LzdFindFirst()で取得したＬＺＨデコードハンドルを指定します。
書庫内に、(AjcLzhFindFirst()の pWild 引数で指定された）ワイルドカードの後続ファイルが見つかった場合は、hLzh で示されるバッファに、見つかったファイルの情報（詳細は、AjcLzhFindFirst()参照）を設定し、TRUE を返します。

戻り値　：　TRUE　：ファイルが見つかった
FALSE：当該ファイルはもう無い

備　考　：　エラーを取得するには、AjcLzhGetLastError()を実行します

46.1.3. 書庫内のファイルオープン（AjcLzhOpen）

形　式　：　HAJCLZH　AjcLzhOpen(C_UTP pFilePath, UI flag, UX xp, UI (CALLBACK *cbRead)(UX xp, VOP pBuf, UI len),
VO (CALLBACK *cbSeek)(UX xp, UL ofs));;

引　数　：　pFilePath　- 読み出す書庫内ファイル名のアドレス
flag　　- 動作指定フラグ（AJCLZHF_SUBDIR）
xp　　- コールバックパラメタ
cbRead　　- ＬＺＨ書庫ファイル入力用コールバック関数のアドレス
cbSeek　　- ＬＺＨ書庫ファイルのバイト位置設定用コールバック関数のアドレス

説　明　：　pFilePath で指定された書庫内のファイルを、読み出し可能な状態にセットアップします。
flag= AJCLZHF_SUBDIR を指定するとサブディレクトリ下からも当該ファイルを検索します。
サブディレクトリ下を検索しない場合は、flag=0 を指定してください。
pFilePath にワイルドカードを指定したり、flag= AJCLZHF_SUBDIR を指定してサブディレクトリの検索をすることも可能です。この場合、最初に見つかったファイルが読み出しの対象となります。

書庫内のファイル読み出しは、AjcLzhOpen(), AjcLzhRead()と AjcLzhClose()により、以下のような手順で行ないます。

```
HAJCLZH hLzh;
UI      rbytes;
UB      buf[1024];
・・・・・
if ((hLzh = AjcLzhOpen(・・・)) != NULL) {
    while (rbytes = AjcLzhRead(hLzh, buf, 1024)) {

        /*- 読み出したデータの処理-*/

    }
    AjcLzhClose(hLzh);
```

尚、hLzh が示すメンバ（hLzh->crc）に、解凍後ファイルデータのＣＲＣが設定されます。
ＣＲＣは、ＡＮＳＩ－ＣＲＣ１６（生成多項式＝$X^{16}+X^{15}+X^{2}+1$）の右送りで算出されます。

戻り値　：　≠ＮＵＬＬ：ファイルが見つかった（ＬＺＨデコードハンドル）
＝ＮＵＬＬ：当該ファイルは無い

備　考　：　エラーを取得するには、AjcLzhGetLastError()を実行します

コールバック：コールバック関数の仕様は、AjcLzhFindFirst()と同じです。

46.1.4. 書庫内のファイル読み出し（AjcLzhRead）

形　式 ： UI　AjcLzhRead(HAJCLZH hLzh, VOP pBuf, UI len);

引　数 ： hLzh　　－ ＬＺＨデコードハンドル
pBuf　　－ 読み出したデータを格納するバッファのアドレス
len　　－ 読み出すバイト数

説　明 ： AjcLzhOpen でオープンした書庫内ファイルからデータを読み出し、pBuf,len で指定されるバッファに格納します。
hLzh は、AjcLzhOpen で取得したＬＺＨデコードハンドルを指定します。
読み出しが成功した場合は、実際に読み出したバイト数を返します。
ファイルの終端（ＥＯＦ）を検出、あるいは、エラーを検出した場合は０を返します。

戻り値 ： ≠０：ＥＯＦ以外（実際に読みだしたバイト数）
＝０：ＥＯＦ

備　考 ： エラーを取得するには、AjcLzhGetLastError()を実行します

46.1.5. 書庫アクセスのクローズ（AjcLzhClose）

形　式 ： VO　AjcLzhClose(HAJCLZH hLzh);

引　数 ： hLzh　　－ ＬＺＨデコードハンドル

説　明 ： AjcLzhFindFirst() / AjcLzhOpen()で取得したＬＺＨデコードハンドルをクローズし、リソースを解放します。

戻り値 ： なし

46.1.6. 検索したファイル情報の取得（AjcLzhGetFileInfo）

形　式 ： VO　AjcLzhGetFileInfo(HAJCLZH hLzh, PAJCLZHFILEINFO pInfo);

引　数 ： hLzh　　－ ＬＺＨデコードハンドル
pInfo　　－ 取得したファイル情報を格納するバッファのアドレス

説　明 ： AjcLzhFindFirst() / AjcLzhFindNext() / AjcLzhOpen()で検索されたファイルの情報を取得します。
pInfo で示されるバッファには、以下の情報が設定されます。

#	メンバ名	タイプ	内　容	備　考
1	HuffSize	UL	圧縮データサイズ（バイト数）	
2	FileSize	UL	ファイルサイズ　（バイト数）	
3	FileTime	UL	ファイルタイム（1970-01-01, 00:00:00 からの通算秒数）	
4	fLocalTime	BOOL	ファイルタイムがローカルタイムか、世界標準時かの識別	TRUE=ローカルタイム, FALSE=世界標準時
5	crc	UW	解凍データのＣＲＣ	ANSI-CRC16（右送り，初期値＝０）
6	szPathName	UT [MAX_PATH]	ファイルパス名へのポインタ（ex. “temp¥sample.txt”）	
7	szDirName	UT [MAX_PATH]	ディレクトリ名へのポインタ（ex. “temp¥” ）	ディレクトリ名が無い場合は空文字列 ディレクトリ名がある場合、末尾は常に ”¥”
8	szFileName	UT [MAX_PATH]	ファイル名へのポインタ　　（ex. “sample.txt”）	ディレクトリの場合は空文字列(“”)
9	fDir	BOOL	ディレクトリか、ファイルかを示すフラグ	TRUE ：ディレクトリ FALSE：ファイル

戻り値 ： TRUE　－ 成功
FALSE － 失敗

備　考 ： エラーを取得するには、AjcLzhGetLastError()を実行します

46.1.7. 書庫アクセスのエラーコード取得（AjcLzhGetLastError）

形　式 ： AJCLZHERR AjcLzhGetLastError (VO);

引　数 ： なし

説　明 ： 当該スレッドにおいて、最後に実行したＬＺＨデコードＡＰＩのエラーコードを取得します。

戻り値 ： ＬＺＨデコードで最後に発生したエラーコード

46.2. サンプルプログラム

46.2.1. SW_LzhDecodeC（LZH書庫ファイルの解凍）

以下のサンプルプログラムは、コマンドラインで指定されたＬＺＨ書庫ファイルの全フォルダと全ファイルを書庫ファイルと同一フォルダ内に解凍します。　コマンドの形式は、以下のとおりです。

```
A:> S_LzhDecode　ＬＺＨ書庫ファイル名↓
```

```
 1 : //
 2 : //  SW_LzhDecodeC.c
 3 : //
 4 : #include    <AjrCstXX.h>
 5 : #include    <direct.h>
 6 : #include    <tchar.h>
 7 : #include    <imagehlp.h>
 8 :
 9 : //=============================================================================================================//
10 : //  書庫ファイル読み出し用コールバック                                                                         //
11 : //                                                                                                             //
12 : //  引　数　:  xp   - コールバックパラメタ（本プログラムでは、ＬＺＨファイルのハンドルとして使用）             //
13 : //             pBuf - 読み出したデータを格納するバッファのアドレス                                             //
14 : //             len  - 読み出すバイト数                                                                         //
15 : //                                                                                                             //
16 : //  戻り値　:  実際に読み出したバイト数                                                                        //
17 : //=============================================================================================================//
18 : static UI   CALLBACK    cbRead(UX xp, VOP pBuf, UI len)
19 : {
20 :     return (UI)fread(pBuf, 1, len, (FILE *)xp);
21 : }
22 : //=============================================================================================================//
23 : //  書庫ファイルのバイト位置設定用コールバック                                                                 //
24 : //                                                                                                             //
25 : //  引　数　:  xp   - コールバックパラメタ（本プログラムでは、ＬＺＨファイルのハンドルとして使用）             //
26 : //             ofs  - 設定するバイト位置                                                                       //
27 : //                                                                                                             //
28 : //  戻り値　:  なし                                                                                            //
29 : //=============================================================================================================//
30 : static  VO  CALLBACK    cbSeek(UX xp, UL ofs)
31 : {
32 :     fseek((FILE *)xp, ofs, SEEK_SET);
33 : }
34 : //=============================================================================================================//
35 : //  m a i n                                                                                                    //
36 : //=============================================================================================================//
37 : int  AjcMain(int argc, UTP argv[])
38 : {
39 :     UW          crc;                    //  ＣＲＣ算出ワーク
40 :     AJCLZHERR   err;                    //  ＬＺＨデコード エラーコード
41 :     HAJCLZH     hLzhSearch = NULL;      //  ＬＺＨハンドル（検索用）
42 :     HAJCLZH     hLzhRead   = NULL;      //  ＬＺＨハンドル（読出用）
43 :     FILE        *hFileLzh  = NULL;      //  ＬＺＨ書庫ファイル・ハンドル
44 :     FILE        *hFileOut  = NULL;      //  出力ファイルハンドル
45 :     UT          szPath    [MAX_PATH];   //  パス編集ワーク
46 :     UT          szTopDir  [MAX_PATH];   //  先頭フォルダパス
47 :     BOOL        fIgnoreDir = TRUE;      //  ディレクトリ無視フラグ
48 :     AJCLZHFILEINFO  FInfo;              //  ファイル情報
49 :     UT          drv[_MAX_DRIVE], dir[_MAX_DIR], fname[_MAX_FNAME], fext[_MAX_EXT];
50 :     UI          rbytes;                 //  読み出したバイト数
51 :     UB          buf[1024];              //  読み出しバッファ
52 :
53 :     AjcSetStdoutMode();
54 :
55 :     //  ＬＺＨ書庫(.lzh)ファイルオープン
56 :     if (argc >= 2 && (_tfopen_s(&hFileLzh, (UTP)argv[1], TEXT("rb"))) == 0) {
57 :         //  トップフォルダ名設定
58 :         MAjcSplitPath(argv[1] , drv, dir, fname, fext);
59 :         MAjcMakePath (szTopDir, drv, dir, NULL , NULL);
60 :         //  書庫内のファイル検索初期化
61 :         if ((hLzhSearch = AjcLzhFindFirst(TEXT("*.*"), AJCLZHF_SUBDIR, (UX)hFileLzh, cbRead, cbSeek)) != NULL) {
62 :             //  書庫内の全ファイルループ
63 :             do {
64 :                 //----- ファイル情報取得 ------------------------------------------------------------------//
65 :                 AjcLzhGetFileInfo(hLzhSearch, &FInfo);
66 :                 //----- フォルダ作成 ----------------------------------------------------------------------//
67 :                 if (FInfo.szDirName[0] != 0) {
68 :                     UTP     p;
69 :                     UT      tmp [MAX_PATH];
70 :                     MAjcStrCpy(szPath, AJCTSIZE(szPath), szTopDir);
71 :                     MAjcStrCpy(tmp, AJCTSIZE(tmp), FInfo.szDirName);
72 :                     if (p = MAjcStrTok(tmp, TEXT("¥¥"))) {
73 :                         do {
74 :                             AjcPathCat(szPath, p, MAX_PATH);
75 :                              _tmkdir(szPath);
```

```
76 :                         } while (p = MAjcStrTok(NULL, TEXT("¥¥")));
77 :                     }
78 :                 }
79 :                 //----- ファイル出力 ----------------------------------------------------------------------//
80 :                 if (hLzhRead = AjcLzhOpen(FInfo.szPathName, 0, (UX)hFileLzh, cbRead, cbSeek)) {
81 :                     AjcPrintF(TEXT("Extracting %s - "), FInfo.szFileName);
82 :                     MAjcStrCpy(szPath, MAX_PATH, szTopDir);
83 :                     AjcPathCat(szPath, FInfo.szPathName, MAX_PATH);
84 :                     if (_tfopen_s(&hFileOut, szPath, TEXT("wb")) == 0) {
85 :                         //  ファイル出力とＣＲＣ算出
86 :                         crc = 0;
87 :                         while (rbytes = AjcLzhRead(hLzhRead, buf, sizeof buf)) {
88 :                             crc = AjcPartCrc16R(buf, rbytes, crc);
89 :                             fwrite(buf, 1, rbytes, hFileOut);
90 :                         }
91 :                         //  ＣＲＣチェック
92 :                         if ((err = AjcLzhGetLastError()) == AJCLZHERR_OK) {
93 :                             if (crc == hLzhRead->crc) AjcPrintF(TEXT("OK¥n"));
94 :                             else                      AjcPrintF(TEXT("NG (CRC error)¥n"));
95 :                         }
96 :                         else {
97 :                             AjcPrintF(TEXT("NG (err = 0x%04X)¥n"), err);
98 :                         }
99 :                         //  出力ファイル・クローズ
100 :                        fclose(hFileOut);
101 :                    }
102 :                    else {
103 :                        AjcPrintF(TEXT("NG (Creation failure)¥n"));
104 :                    }
105 :                    // ＬＺＨ読み出しハンドル・クローズ
106 :                    AjcLzhClose(hLzhRead);
107 :                }
108 :                else {
109 :                    AjcPrintF(TEXT("File not found (%s)¥n"), FInfo.szPathName);
110 :                }
111 :            } while(AjcLzhFindNext(hLzhSearch));
112 :            // ＬＺＨ検索ハンドル・クローズ
113 :            AjcLzhClose(hLzhSearch);
114 :        }
115 :        else {
116 :            AjcPrintF(TEXT("ＬＺＨ書庫内にファイルが見つかりません¥n"));
117 :        }
118 :        //  ＬＺＨ書庫(.lzh)ファイル クローズ
119 :        if (hFileLzh != NULL) fclose(hFileLzh);
120 :    }
121 :    else {
122 :        AjcPrintF(TEXT("ＬＺＨ書庫ファイルのオープン失敗¥n"));
123 :    }
124 :
125 :    getchar();
126 :    return 0;
127 : }
```

47. ビットマップ操作

ビットマップ操作に関するＡＰＩ群です。

47.1. サポートＡＰＩ

ビットマップ操作のサポートＡＰＩ一覧を以下に示します。

#	関 数 名	内容
1	AjcChangeBitmapColor	ビットマップの色変更
2	AjcSetBitmapToClipboard	ビットマップデータをクリップボードへ格納する
3	AjcGetBitmapFromClipboard	ビットマップデータをクリップボードから取得する
4	AjcWriteBitmapToFile	ビットマップファイルを作成する
5	AjcCreateDibFromFile	ビットマップファイルの読み出し（ＤＩＢセクション作成）
6	AjcGetBitmapSize	ビットマップサイズの取得
7	AjcGetIconSize	アイコンサイズの取得

47.1.1. ビットマップの色変更(AjcChangeBitmapColor)

形　式　：　BOOL　AjcChangeBitmapColor(HBITMAP hBmp, COLORREF before, COLORREF after);

引　数　：　hBmp　- 色の変更を行うビットマップのハンドル
before - 変更前の色コード
after　- 変更後の色コード

説　明　：　ビットマップデータ中の、「before」で指定された色を、「after」で指定された色に変更します。

戻り値　：　TRUE　- 成功
FALSE - 失敗

47.1.2. ビットマップデータをクリップボードへ格納する(AjcSetBitmapToClipboard)

形　式　：　BOOL　AjcSetBitmapToClipboard (HBITMAP hBitmap);

引　数　：　hBitmap　　- クリップボードへ格納するビットマップデータ

説　明　：　新たに生成およびコピーしたビットマップオブジェクトをクリップボードに格納します。
指定した「hBitmap」は保存する必要はありません。(自己責任でDeleteObject()により削除してください)

戻り値　：　TRUE　- 成功
FALSE – 失敗

47.1.3. クリップボードからビットマップデータを取得する(AjcGetBitmapFromClipboard)

形　式　：　HBITMAP　AjcGetBitmapFromClipboard (VO);

引　数　：　なし

説　明　：　クリップボードうちのビットマップデータを、新たに作成したビットマップオブジェクトにコピーし、そのビットマップハンドルを返します。
取得したビットマップデータ（戻り値）は、使用後に「DeleteObject()」で削除しなければなりません。

戻り値　：　≠NULL - 成功（取得したビットマップデータのハンドル）

＝NULL - 失敗

47.1.4. ビットマップファイルを作成する(AjcWriteBitmapToFile)

形　式 ： BOOL AjcWriteBitmapToFile (HBITMAP hBitmap, C_UTP pFilePath);

引　数 ： hBitmap - ビットマップオブジェクトのハンドル
pFilePath - 出力するビットマップファイルのパス名

説　明 ： 「hBitmap」で示されるビットマップイメージのビットマップファイル(.bmp)を作成します。
pFilePath は作成するビットマップファイルのパス名を指定します。
指定したファイルが既に存在する場合は、上書きします。

戻り値 ： TRUE - 成功
FALSE - 失敗

47.1.5. ビットマップファイルを読み出してＤＩＢセクションを作成((AjcCreateDibFromFile)

形　式 ： HBITMAP AjcCreateDibFromFile(C_UTP pFilePath, UIP pWidth, UIP pHeight, PAJCBITMAPINFO pBmpInfo, VOP *ppImage);

引　数 ： pFilePath - ビットマップファイル(.bmp)のファイルパス名
pWidth - ビットマップの縦ピクセル数を格納するバッファのアドレス (不要時は NULL)
pHeight - ビットマップの縦ピクセル数を格納するバッファのアドレス (不要時は NULL)
pBmpInfo - ビットマップ情報を格納するバッファのアドレス (不要時は NULL)
ppImage - イメージデータへのポインタを格納するバッファのアドレス (不要時は NULL)

説　明 ： ビットマップファイル(.bmp)を読み出して、当該イメージを格納したＤＩＢセクションを作成します。
pWidth で示すバッファには、イメージの横ピクセル数が格納されます。
pHeight 示すバッファには、イメージの縦ピクセル数が格納されます。
ppImage で示されるバッファ(ポインタ)には、イメージデータのアドレスが格納されます。
pBmpInfo で示されるバッファには、以下の構造体(union)データが格納されます。

```
typedef union {
    DWORD             size;   //  構造体のサイズ
    BITMAPCOREHEADER    bc;     //  OS2 スタイルのビットマップ情報   , size = sizeof(BITMAPCOREHEADER) =  12
    BITMAPINFOHEADER    bi;     //  Windows の拡張ビットマップ情報   , size = sizeof(BITMAPINFOHEADER) =  40
    BITMAPV4HEADER      v4;     //  バージョン４ヘッダ(Win95/NT～)  , size = sizeof(BITMAPV4HEADER)    = 108
    BITMAPV5HEADER      v5;     //  バージョン５ヘッダ(Win98/2K～)  , size = sizeof(BITMAPV5HEADER)    = 124
} AJCBITMAPINFO, *PAJCBITMAPINFO;
```

ビットマップファイルは、プラットフォームにより形式が異なり、各々以下の構造体で表されます。

ＯＳ２スタイル

```
typedef struct tagBITMAPCOREHEADER {
    DWORD   bcSize;     // 構造体サイズ(=12)
    WORD    bcWidth;    // イメージの横ピクセル数
    WORD    bcHeight;   // イメージの縦ピクセル数
    WORD    bcPlanes;   // １固定
    WORD    bcBitCount; // ピクセルあたりのカラービット数(1, 4, 8 or 24)
} BITMAPCOREHEADER, FAR *LPBITMAPCOREHEADER, *PBITMAPCOREHEADER;
```

Windows の拡張ビットマップ情報(Windows3.1～)

```
typedef struct tagBITMAPINFOHEADER{
    DWORD       biSize;           // 構造体サイズ(=40)
    LONG        biWidth;          // イメージの横ピクセル数
    LONG        biHeight;         // イメージの縦ピクセル数
    WORD        biPlanes;         // １固定
    WORD        biBitCount;       // カラービット数(1, 4, 8, 16, 24 or 32)
    DWORD       biCompression;    // 圧縮コード
    DWORD       biSizeImage;      // イメージのバイト数
    LONG        biXPelsPerMeter;  // 水平解像度
    LONG        biYPelsPerMeter;  // 垂直解像度
    DWORD       biClrUsed;        // 使われている色の数
    DWORD       biClrImportant;   // 重要な色の数
} BITMAPINFOHEADER, FAR *LPBITMAPINFOHEADER, *PBITMAPINFOHEADER;
```

バージョン４ヘッダ(Windows95/NT4.0～)

```
typedef struct {
    DWORD        bV4Size;           // 構造体サイズ(=108)
    LONG         bV4Width;          // イメージの横ピクセル数
    LONG         bV4Height;         // イメージの縦ピクセル数
    WORD         bV4Planes;         // １固定
    WORD         bV4BitCount;       // カラービット数(1, 4, 8, 16, 24 or 32)
    DWORD        bV4V4Compression;  // 圧縮コード
    DWORD        bV4SizeImage;      // イメージのバイト数
    LONG         bV4XPelsPerMeter;  // 水平解像度
    LONG         bV4YPelsPerMeter;  // 垂直解像度
    DWORD        bV4ClrUsed;        // 使われている色の数
    DWORD        bV4ClrImportant;   // 重要な色の数
    DWORD        bV4RedMask;        // 赤マスク
    DWORD        bV4GreenMask;      // 緑マスク
    DWORD        bV4BlueMask;       // 青マスク
    DWORD        bV4AlphaMask;      // アルファマスク
    DWORD        bV4CSType;         // カラースペースタイプ
    CIEXYZTRIPLE bV4Endpoints;      // ＸＹＺ値
    DWORD        bV4GammaRed;       // 赤ガンマ値
    DWORD        bV4GammaGreen;     // 緑ガンマ値
    DWORD        bV4GammaBlue;      // 青ガンマ値
} BITMAPV4HEADER, FAR *LPBITMAPV4HEADER, *PBITMAPV4HEADER;
```

バージョン５ヘッダ(Windows2000/XP～)

```
typedef struct {
    DWORD        bV5Size;                // 構造体サイズ(=124)
    LONG         bV5Width;               // イメージの横ピクセル数
    LONG         bV5Height;              // イメージの縦ピクセル数
    WORD         bV5Planes;              // １固定
    WORD         bV5BitCount;            // カラービット数(1, 4, 8, 16, 24 or 32)
    DWORD        bV5Compression;         // 圧縮コード
    DWORD        bV5SizeImage;           // イメージのバイト数
    LONG         bV5XPelsPerMeter;       // 水平解像度
    LONG         bV5YPelsPerMeter;       // 垂直解像度
    DWORD        bV5ClrUsed;             // 使われている色の数
    DWORD        bV5ClrImportant;        // 重要な色の数
    DWORD        bV5RedMask;             // 赤マスク
    DWORD        bV5GreenMask;           // 緑マスク
    DWORD        bV5BlueMask;            // 青マスク
    DWORD        bV5AlphaMask;           // アルファマスク
    DWORD        bV5CSType;              // カラースペースタイプ
    CIEXYZTRIPLE bV5Endpoints;           // ＸＹＺ値
    DWORD        bV5GammaRed;            // 赤ガンマ値
    DWORD        bV5GammaGreen;          // 緑ガンマ値
    DWORD        bV5GammaBlue;           // 青ガンマ値
    DWORD        bV5Intent;              // 展開意図
    DWORD        bV5ProfileData;         // プロファイルデータ／ファイル名
    DWORD        bV5ProfileSize;         // 埋め込まれたデータ／ファイル名のサイズ
    DWORD        bV5Reserved;            //
} BITMAPV5HEADER, FAR *LPBITMAPV5HEADER, *PBITMAPV5HEADER;
```

戻り値 ： ≠NULL - 成功（ＤＩＢセクションのビットマップ・ハンドル）
＝NULL - 失敗

47.1.6. ビットマップサイズの取得(AjcGetBitmapSize)

形　式 ： UI　AjcGetBitmapSize(HBITMAP hBitmap, LPSIZE pSize);

引　数 ： hBitmap　- ビットマップオブジェクトのハンドル
pSize　　- ビットマップのサイズを格納するバッファのアドレス（不要時はNULL）

説　明 ： 「hBitmap」で示されるビットマップイメージのサイズを取得します。

戻り値 ： ≠0：ピクセルの色を示すために必要なビット数
＝0：失敗

47.1.7. アイコンサイズの取得(AjcGetIconSize)

形　式 ： BOOL　AjcGetIconSize(HICON hIcon, LPSIZE pSize);

引　数 ： hIcon　- アイコンのハンドル
pSize　- アイコンのサイズを格納するバッファのアドレス（不要時はNULL）

説　明 ： 「hIcon」で示されるアイコンのサイズを取得します。

戻り値 ： ≠0：ピクセルの色を示すために必要なビット数
＝0：失敗

48. ＤＩＢセクション

ＤＩＢセクションによる、ビットマップ操作関数です。

ＤＩＢセクションによるビットマップ操作では、Windows-API を介さずに直接イメージのメモリをアクセスできるため、高速にビットマップ操作を実行することができます。

ここで扱うビットマップは、基本的にフルカラー（２４ビット or ３２ビット／ピクセル，約 1677 万色）のイメージだけです。
また、トップダウン形式（Height が負数）のイメージデータには対応していません。

48.1. サポートＡＰＩ

ＤＩＢセクションのサポートＡＰＩ一覧を以下に示します。

#	関 数 名	内容
1	AjcDibCreate	ＤＩＢセクションの生成
2	AjcDibClear	ＤＩＢセクションを特定の色でクリアー
3	AjcDibCopyBitmap[V]	ＤＩＢセクションへビットマップをコピー
4	AjcDibCorrectedCopy[V]	ＤＩＢセクションからＤＩＢセクションへ特定の色範囲の部分だけをコピー
5	AjcDibExcludedCopy[V]	ＤＩＢセクションからＤＩＢセクションへ特定の色範囲を除くの部分だけをコピー
6	AjcDibSelectedCopy[V]	ＤＩＢセクションからＤＩＢセクションへ複数の色範囲を選択してコピー
7	AjcDibColorCount[V]	ＤＩＢセクション内で特定の色範囲のピクセル数をカウントする
8	AjcDibFillRect[V]	ＤＩＢセクション内矩形領域の塗りつぶし
9	AjcDibGetPixel	ピクセルの取得
10	AjcDibSetPixel	ピクセルの設定
11	AjcDibGetLinrPtr	ラインの先頭アドレス取得
12	AjcDibReadFileAndCreate	ＤＩＢセクションの生成とビットマップファイルの読み出し

48.1.1. ＤＩＢセクションの生成 (AjcDibCreate)

形　式 ： HBITMAP　AjcDibCreate (int width, int height, int BitCount, PAJCDIBINFO pBuf);

引　数 ： width　　- ビットマップの幅　(横ピクセル数)
height　- ビットマップの高さ (縦ピクセル数)
BitCount　- イメージのビット数(24 or 32)
pBuf　　- 生成したＤＩＢセクションの情報を格納するバッファのアドレス (不要時はＮＵＬＬ)

説　明 ： 「width」と「height」で指定されたビットマップサイズの DIB セクション (非圧縮) を作成します。
「BitCount」は１ピクセル当たりのビット数であり、２４あるいは３２を指定します。
「pBuf」は、生成したＤＩＢセクションの情報を格納するバッファのアドレスを指定します。
このバッファは、以下の形式で定義されています。

```
typedef struct {
    HBITMAP    hBmp;                    // ビットマップハンドル
    int        BitCount;                // イメージのビット数
    int        width;                   // ビットマップの幅　(横ピクセル数)
    int        height;                  // ビットマップの高さ (縦ピクセル数)
    int        PixelSize;               // １ピクセル当たりのバイト数
    int        LineSize;                // １ライン当たりのバイト数
    UBP        pImage;                  // イメージデータの先頭アドレス
} AJCDIBINFO, *PAJCDIBINFO;
typedef const *AJCDIBINFO  PCAJCDIBINFO;
```

戻り値 ： ≠NULL - 成功 (ビットマップハンドル)
＝NULL - 失敗

48.1.2. ＤＩＢセクションを特定の色でクリアー (AjcDibClear)

形　式 ： BOOL　AjcDibClear(PAJCDIBINFO pDibInfo. COLORREF rgb);

引　数 ： pDibInfo　- ＤＩＢセクションの情報 (AjcDibCreate で取得したＤＩＢセクションの情報 (pBuf) を指定)
rgb　　- クリアーする色

説　明 ： 指定した DIB セクションのビットマップ全体を「rgb」で指定した色で塗りつぶします。

戻り値 ： TRUE　- 成功
FALSE - 失敗

48.1.3. ＤＩＢセクションへビットマップをコピー (AjcDibCopyBitmap[V])

形　式 ： BOOL AjcDibCopyBitmap (PAJCDIBINFO pDibInfo, int dpx, int dpy, int dcx, dcy,
HBITMAP hBitmap , int spx, int spy, int scx, int scy, int rop);

BOOL AjcDibCopyBitmapV(PAJCDIBINFO pDibInfo, const RECT *rcDest,
HBITMAP hBitmap , const RECT *rcSrc, int rop);

引　数 ： pDibInfo - コピー先ＤＩＢセクションの情報
dpx, dpy、dcx、dcy / rcDest - コピー先矩形の左上位置，幅と高さ
hBitmap - コピー元のビットマップハンドル
spx, spy, scx, scy / rcSrc - コピー元矩形の左上位置，幅と高さ
rop - ラスタオペレーション(単純コピーの場合は「SRCCOPY」)

説　明 ： 「hBitmap」で指定されたビットマップ中の矩形を「pDibInfo」で指定されたビットマップ中の矩形へコピーします。
「dpx, dpy, dcx, dcy」はコピー先の矩形を指定します。(全て０とした場合は、コピー先ビットマップ全体が対象)
「spx, spy, scx, scy」はコピー元の矩形を指定します。(全て０とした場合は、コピー元ビットマップ全体が対象)
転送元の矩形と転送先の矩形サイズが異なる場合は、ビットマップが拡大／縮小されます。
「rop」はラスターオペレーションであり、以下のいずれかのシンボルを指定します。(WindowsAPI「StretchBlt」と同じ)

シンボル	意　味
BLACKNESS	すべての出力を物理パレットのインデクス０（デフォルトは黒）に設定する
DSTINVERT	コピー先の矩形を反転する
MERGECOPY	コピー元とコピー先のビットマップを論理 AND 演算子で結合する
MERGEPAINT	コピー元とコピー先のビットマップを論理 OR 演算子で結合する
NOTSRCCOPY	コピー元のビットマップを反転してからコピーする
NOTSRCERASE	コピー元とコピー先のビットマップを論理 OR 演算子で結合した結果を反転する
PATCOPY	パターンをそのままコピーする
PATINVERT	コピー元とコピー先のビットマップを論理 XOR 演算子で結合する
PATPAINT	パターンのカラーと反転したコピー元矩形のカラーを論理 OR 演算子で結合し、その結果とコピー先ビットマップのカラーを論理 OR 演算子で結合する
SRCAND	コピー元とコピー先のビットマップを論理 AND 演算子で結合する
SRCCOPY	**コピー元をそのままコピーする**
SRCERASE	コピー先ビットマップの反転したカラーとコピー元ビットマップのカラーを論理 AND 演算子で結合する
SRCINVERT	コピー元とコピー先のビットマップを論理 XOR 演算子で結合する
SRCPAINT	コピー元とコピー先のビットマップを論理 OR 演算子で結合する
WHITENESS	すべての出力を物理パレットのインデクス１（デフォルトは白）に設定する

戻り値 ： TRUE - 成功
FALSE - 失敗

48.1.4. ＤＩＢセクションからＤＩＢセクションへ特定の色範囲の部分だけをコピー (AjcDibCorrectedCopy[V])

形　式 ： BOOL AjcDibCorrectedCopy (PAJCDIBINFO pDibDest, int dxp, int dyp, int dcx, int dcy,
PAJCDIBINFO pDibSrc , int spx, int spy , COLORRREF rgbBase, double distance);

BOOL AjcDibCorrectedCopyV(PAJCDIBINFO pDibDest, const RECT *rcDest,
PAJCDIBINFO pDibSrc , const POINT *ptSrc, COLORRREF rgbBase, double distance);

引　数 ：
pDibDest	- コピー先ＤＩＢセクションの情報
dpx, dpy, dcx, dcy / rcDest	- コピー先矩形の左上位置，幅と高さ
pDibSrc	- コピー元ＤＩＢセクションの情報
spx, spy / ptSrc	- コピー元矩形の左上位置
rgbBase	- コピー対象とする色のベース値
distance	- コピー対象とする色の範囲（rgbBase からの距離（0.0 ～ 441.68)）

説　明 ： ＤＩＢセクションから他のＤＩＢセクションへ指定された色範囲の部分だけをコピーします。
「dpx, dpy, dcx, dcy」はコピー先の矩形を指定します。（全て０とした場合は、コピー先ビットマップ全体が対象）

戻り値 ： TRUE - 成功
FALSE - 失敗

48.1.5. ＤＩＢセクションからＤＩＢセクションへ特定の色範囲を除く部分だけをコピー (AjcDibExcludedCopy[V])

形　式 ： BOOL AjcDibExcludedCopy (PAJCDIBINFO pDibDest, int dxp, int dyp, int dcx, int dcy,
PAJCDIBINFO pDibSrc , int spx, int spy , COLORRREF rgbBase, double distance);

BOOL AjcDibExcludedCopyV(PAJCDIBINFO pDibDest, const RECT *rcDest,
PAJCDIBINFO pDibSrc , const POINT *ptSrc, COLORRREF rgbBase, double distance);

引　数 ：
pDibDest	- コピー先ＤＩＢセクションの情報
dpx, dpy, dcx, dcy / rcDest	- コピー先矩形の左上位置，幅と高さ
pDibSrc	- コピー元ＤＩＢセクションの情報
spx, spy / ptSrc	- コピー元矩形の左上位置
rgbBase	- コピー除外とする色のベース値
distance	- コピー除外とする色の範囲（rgbBase からの距離（0.0 ～ 441.68)）

説　明 ： ＤＩＢセクションから他のＤＩＢセクションへ指定された色範囲を除く部分だけをコピーします。
「dpx, dpy, dcx, dcy」はコピー先の矩形を指定します。（全て０とした場合は、コピー先ビットマップ全体が対象）

戻り値 ： TRUE - 成功
FALSE - 失敗

48.1.6. ＤＩＢセクションからＤＩＢセクションへ複数の色範囲を選択してコピー (AjcDibSelectedCopy[V])

形　式 ： BOOL AjcDibSelectedCopy (PAJCDIBINFO pDibDest, int dxp, int dyp, int dcx, int dcy,
PAJCDIBINFO pDibSrc , int spx, int spy , int OpeNum, ...);

BOOL AjcDibSelectedCopyV(PAJCDIBINFO pDibDest, const RECT *rcDest,
PAJCDIBINFO pDibSrc , const POINT *ptSrc, int OpeNum, ...); ・・・マクロ

引　数 ： pDibDest - コピー先ＤＩＢセクションの情報
dpx, dpy, dcx, dcy / rcDest - コピー先矩形の左上位置，幅と高さ
pDibSrc - コピー元ＤＩＢセクションの情報
spx, spy / ptSrc - コピー元矩形の左上位置
OpeNum - 選択／除外する色範囲の個数
（「選択／除外」「色のベース値」「距離」の３つを１セットとした情報の個数）

説　明 ： ＤＩＢセクションから他のＤＩＢセクションへ複数の色範囲を選択してコピーします。
「dpx, dpy, dcx, dcy」はコピー先の矩形を指定します。(全て０とした場合は、コピー先ビットマップ全体が対象)
この関数では、「AjcDibCorrectedCopy」や「AjcDibExcludedCopy」の色選択／除外コピー機能に加えて、コピー時のラスタオペレーション(OR, AND, XOR)を選択することもできます。

「OpeNum」に続けて、選択／除外する色の範囲を複数（OpeNum で示すセット数だけ）指定します。
選択／除外する色の範囲は、以下の３つの情報で１セットとなります。

#	タイプ	内　容
1	int	AJCDIFOPE_CORRECT : 色範囲の選択（単純コピー） AJCDIBOPE_COR_OR : 〃 （論理和 (OR) ） AJCDIBOPE_COR_AND : 〃 （論理積 (AND) ） AJCDIBOPE_COR_XOR : 〃 （排他論理和 (XOR) ） AJCDIFOPE_EXCLUDE : 色範囲の除外（単純コピー） AJCDIBOPE_EXC_OR : 〃 （論理和 (OR) ） AJCDIBOPE_EXC_AND : 〃 （論理積 (AND) ） AJCDIBOPE_EXC_XOR : 〃 （排他論理和 (XOR) ）
2	COLORREF	ベース色(ＲＧＢ値)
3	double	ベース色からの距離

例えば、赤色から距離 100～150 の部分だけをコピーする場合は、以下のように指定します。

```
AjcDibSelectedCopy(pDibDest, 0, 0, 0, 0, pDibSrc, 0, 0,
            2,                                      // 選択／除外する色範囲の個数
            AJCDIBOPE_CORRECT, RGB(255, 0, 0), 150.0,   // 赤から距離１５０以内を選択
            AJCDIBOPE_EXCLUDE, RGB(255, 0, 0), 100.0);  // 赤から距離１００以内を除外
```

戻り値 ： TRUE - 成功
FALSE - 失敗

48.1.7. ＤＩＢセクション内で特定の色範囲のピクセル数をカウントする (AjcDibColorCount[V])

形　式 ： BOOL AjcDibColorCount (PAJCDIBINFO pDibInfo, int x, int y, int cx, int cy, COLORREF rgbBase, double distance);
BOOL AjcDibColorCountV(PAJCDIBINFO pDibInfo, const RECT *rcArea , COLORREF rgbBase, double distance);

引　数 ： pDibInfo - ＤＩＢセクション情報のアドレス
x, y, cx, cy / rcArea - ピクセル数をカウントする矩形の左上位置，幅と高さ／矩形領域を示す構造体
rgbBase - ピクセル数をカウントする色のベース値
distance - ピクセル数をカウントする色の距離（rgbBase からの距離（0.0 ～ 441.68））

説　明 ： DIB セクションの指定された矩形うちで、指定された色範囲のピクセル数をカウントします。
「x, y, cx, cy」はカウントする矩形を指定します。（全て０とした場合は、DIB セクションのビットマップ全体が対象）

戻り値 ： ≧０ - 指定された色範囲のピクセル数
＝-1 - 失敗

48.1.8. 矩形領域の塗りつぶし (AjcDibGetPixel[V])

形　式 ： BOOL AjcDibFillRect (PAJCDIBINFO pDibInfo, int x, int y, int cx, int cy, COLORREF rgb);
BOOL AjcDibFillRectV(PAJCDIBINFO pDibInfo, PAJCDIBINFO pDibInfo, const RECT *pRect, COLORREF rgb);

引　数 ： pDibInfo - ＤＩＢセクションの情報（AjcDibCreate で取得したＤＩＢセクションの情報（pBuf）を指定）
x, y, cx, cy - 矩形の左上ピクセル位置，幅と高さ
pRect - 矩形領域を示す RECT 構造体へのポインタ

説　明 ： イメージ中の指定ピクセル位置の色を取得します。
「x, y, cx, cy」は塗りつぶす矩形を指定します。（全て０とした場合は、DIB セクション全体が対象）

戻り値 ： TRUE - 成功
FALSE - 失敗

48.1.9. ピクセルの取得 (AjcDibGetPixel[V])

形　式 ： COLORREF AjcDibGetPixel (PAJCDIBINFO pDibInfo, int x, int y);
COLORREF AjcDibGetPixelV(PAJCDIBINFO pDibInfo, const POINT *pt);

引　数 ： pDibInfo - ＤＩＢセクションの情報（AjcDibCreate で取得したＤＩＢセクションの情報（pBuf）を指定）
x, y / pt - イメージのピクセル位置（(0, 0) ～ ）

説　明 ： イメージ中の指定ピクセル位置の色を取得します。
ピクセル位置は、イメージの左上をピクセル位置(x)＝０，ライン位置(y)＝０とした値で指定します。

戻り値 ： ≠-1 - ピクセルの色
＝-1 - 失敗

48.1.10. ピクセルの設定 (AjcDibSetPixel[V])

形　式 ： BOOL AjcDibSetPixel (PAJCDIBINFO pDibInfo, int x, int y , COLORREF rgb);
BOOL AjcDibSetPixelV(PAJCDIBINFO pDibInfo, const POINT *pt, COLORREF rgb);

引　数 ： pDibInfo - ＤＩＢセクションの情報（AjcDibCreate で取得したＤＩＢセクションの情報（pBuf）を指定）
x, y / pt - イメージのピクセル位置（(0, 0) ～ ）
rgb - 設定する色

説　明 ： イメージ中の指定ピクセル位置の色を設定します。
ピクセル位置は、イメージの左上をピクセル位置(x)＝０，ライン位置(y)＝０とした値で指定します。

戻り値 ： TRUE - 成功
FALSE - 失敗

48.1.11. ラインの先頭アドレス取得(AjcDibGetLinePtr)

形　式 ： UBP　AjcDibGetLinrPtr (PAJCDIBINFO pDibInfo, int y);

引　数 ： pDibInfo　－ ＤＩＢセクションの情報（AjcDibCreate で取得したＤＩＢセクションの情報（pBuf）を指定）
y　　－ イメージのライン位置　（０～）

説　明 ： イメージ・メモリ中の指定ライン位置の先頭アドレスを取得します。
ライン位置は、イメージの上を０とした値で指定します。

戻り値 ： ≠NULL －　成功（イメージ・メモリ中のライン先頭へのポインタ）
＝NULL －　失敗

48.1.12. ＤＩＢセクションの生成とビットマップファイルの読み出し (AjcDibReadFileAndCreate)

形　式 ： HBITMAP　AjcDibReadFileAndCreate (C_UTP pFilePath, PAJCDIBINFO pBuf);

引　数 ： pFilePath － ビットマップファイル(.bmp)のファイルパス名
pBuf　　－ 生成したＤＩＢセクションの情報を格納するバッファのアドレス（不要時はＮＵＬＬ）

説　明 ： ビットマップファイル(.bmp)を読み出して、当該イメージを格納したＤＩＢセクションを作成します。
「pBuf」は、生成したＤＩＢセクションの情報を格納するバッファのアドレスを指定します。
このバッファは、以下の形式で定義されています。

```
typedef struct {
    HBITMAP    hBmp;                    // ビットマップハンドル
    int        BitCount;                // イメージのビット数
    int        width;                   // ビットマップの幅　(横ピクセル数)
    int        height;                  // ビットマップの高さ (縦ピクセル数)
    int        PixelSize;               // １ピクセル当たりのバイト数
    int        LineSize;                // １ライン当たりのバイト数
    UBP        pImage;                  // イメージデータの先頭アドレス
} AJCDIBINFO, *PAJCDIBINFO;
typedef const *AJCDIBINFO  PCAJCDIBINFO;
```

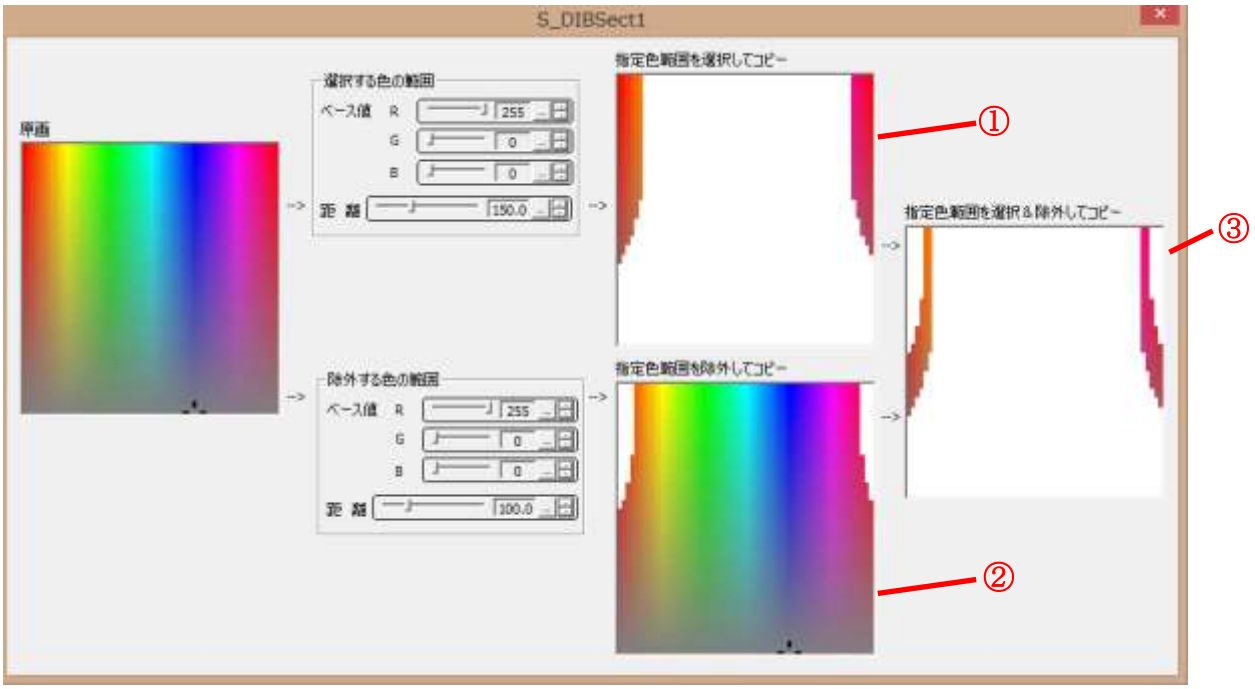
戻り値 ： ≠NULL －　成功（ビットマップハンドル）
＝NULL －　失敗

注　意 ： フルカラー(約 1677 万色)以外のビットマップファイル（2 色,8 色,16 色、256 色あるいは 65536 色）を読み出した場合、ＤＩＢセクションは生成されますが、(pDibInfo を指定する)他のＡＰＩで読み出し，描画やコピーはできません。

48.2.　サンプルプログラム

48.2.1.　SW_DibSect（色の合成）

次のサンプルプログラムは、原画から色を選択してコピーしたり、色を除外してコピーしたりします。

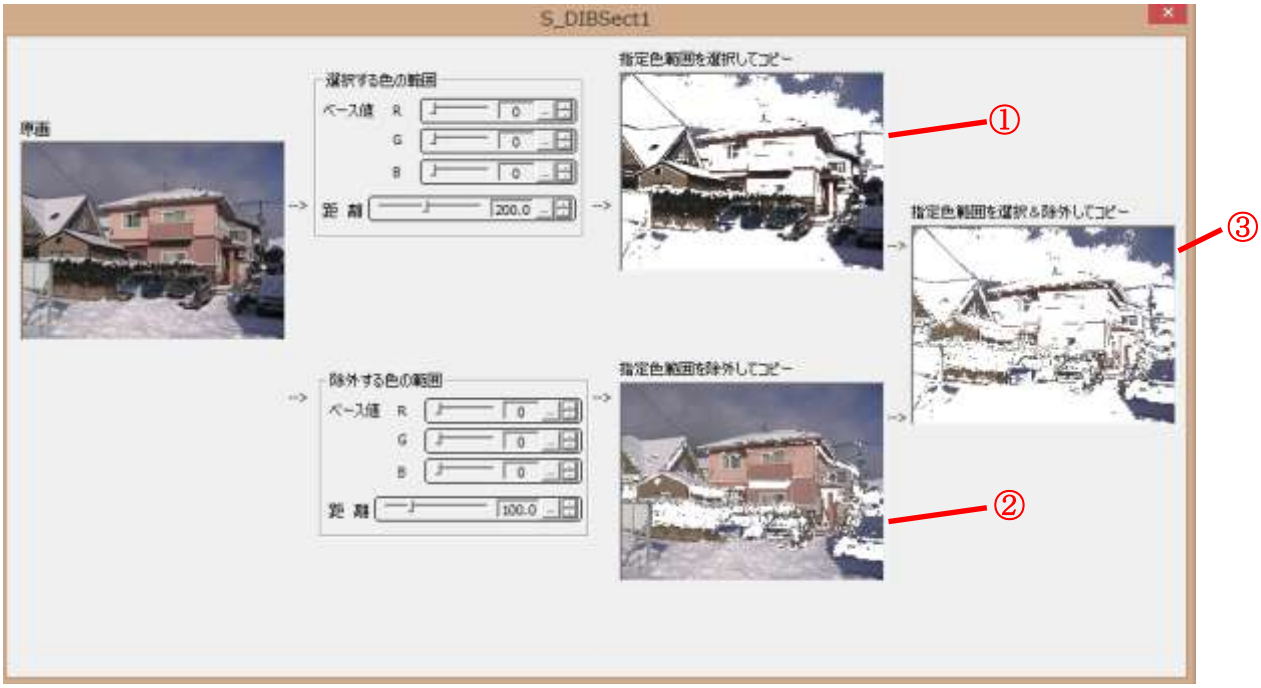


上記設定例では、以下の操作を実行しています。（添付のサンプルプログラムでは、上記原画を使用しています）

① 赤色から距離１５０の範囲を選択してコピー（つまり、赤っぽい部分だけをコピー）

② 赤色から距離１００の範囲を除外してコピー（つまり、赤っぽい部分を除外してコピー）

③ 赤色から距離１５０の範囲を選択後、距離１００の範囲を除外してコピー（赤っぽい部分だけを選択後、さらに赤っぽい部分を除外）



上記設定例では、以下の操作を実行しています。（使用するビットマップファイル（S_DIBSect1.bmp）の内容を変更した例です）

① 黒色から距離２００の範囲を選択してコピー（つまり、暗い部分だけをコピー）

② 黒色から距離１００の範囲を除外してコピー（つまり、暗い部分を除外してコピー）

③ 黒色から距離２００の範囲を選択後、距離１００の範囲を除外してコピー（暗い部分だけを選択後、さらに暗い部分を除外）

```
 1 : //
 2 : //  SW_DibSect.c
 3 : //
 4 : #include    <AjrCstXX.h>
 5 : #include    <tchar.h>
 6 : #include    "resource.h"
 7 :
 8 : #define RGB_WHITE   RGB(255, 255, 255)
 9 :
10 : //-----------------------------------------------------------------------------------------------------------//
11 : //  ワーク                                                                                                   //
12 : //-----------------------------------------------------------------------------------------------------------//
13 : static  HINSTANCE       hInst;
14 : static  HWND            hDlgMain;
15 : static  HBITMAP         hBmpOrg, hBmpCor, hBmpExc, hBmpSel;
16 : static  AJCDIBINFO      DbifOrg, DbifCor, DbifExc, DbifSel;
17 :
18 : //-----------------------------------------------------------------------------------------------------------//
19 : //  内部サブ関数                                                                                             //
20 : //-----------------------------------------------------------------------------------------------------------//
21 : AJC_DLGPROC_DEF(Main);
22 : static VO ShowPic(VO);
23 :
24 : //===========================================================================================================//
25 : //                                                                                                           //
26 : //  Ｗｉｎ Ｍａｉｎ                                                                                          //
27 : //                                                                                                           //
28 : //===========================================================================================================//
29 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
30 : {
31 :     MSG     msg;
32 :
33 :     hInst = hInstance;
34 :
35 :     //----- メイン・ダイアログオープン ----------------//
36 :     hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_DLGMAIN), NULL, AJC_DLGPROC_NAME(Main));
37 :     ShowWindow(hDlgMain, SW_SHOW);
38 :
39 :     //----- メッセージループ --------------------------//
40 :     while (GetMessage(&msg, NULL, 0, 0)) {
41 :         do {
42 :             if (IsDialogMessage(hDlgMain, &msg)) break;
43 :             TranslateMessage(&msg);
44 :             DispatchMessage (&msg);
45 :         } while (0);
46 :     }
47 :
48 :     return (int)msg.wParam ;
49 : }
50 : //===========================================================================================================//
51 : //                                                                                                           //
52 : //  ダイアログ・プロシージャ                                                                                 //
53 : //                                                                                                           //
54 : //===========================================================================================================//
55 : //----- ダイアログ初期化 ----------------------------------------------------------------------------------//
56 : AJC_DLGPROC(Main, WM_INITDIALOG         )
57 : {
58 :     UT      path[MAX_PATH];
59 :
60 :     hDlgMain = hDlg;
61 :
62 :     //----- 原画表示 ---------------------------------//
63 :     AjcGetAppPath(path, MAX_PATH);
64 :     MAjcStrCat(path, AJCTSIZE(path), TEXT("SW_DibSect.bmp"));
65 :     hBmpOrg = AjcDibReadFileAndCreate(path, &DbifOrg);
66 :     SendDlgItemMessage(hDlg, IDC_PIC_ORG, STM_SETIMAGE, IMAGE_BITMAP, (LPARAM)hBmpOrg);
67 :
68 :     //----- ＤＩＢセクション生成 ---------------------//
69 :     hBmpCor = AjcDibCreate(DbifOrg.width, DbifOrg.height, DbifOrg.BitCount, &DbifCor);
70 :     hBmpExc = AjcDibCreate(DbifOrg.width, DbifOrg.height, DbifOrg.BitCount, &DbifExc);
71 :     hBmpSel = AjcDibCreate(DbifOrg.width, DbifOrg.height, DbifOrg.BitCount, &DbifSel);
72 :
73 :     //----- ダイアログ項目の設定値ロード ------------//
74 :     AjcLoadAllControlSettings (hDlg, TEXT("Values"), AJCCTL_SELACT_ALL);
75 :
76 :     //----- 各ピクチャ表示 ---------------------------//
77 :     ShowPic();
78 :
79 :     return TRUE;
80 : }
```

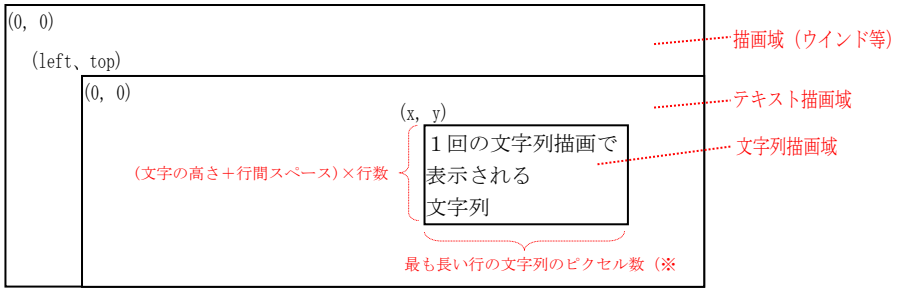
```
 81 : //----- ウインド破棄 ----------------------------------------------------------------------------------------------//
 82 : AJC_DLGPROC(Main, WM_DESTROY            )
 83 : {
 84 :     //----- DIBitmap 削除 ----------------------------//
 85 :     DeleteObject(hBmpCor);
 86 :     DeleteObject(hBmpExc);
 87 :     DeleteObject(hBmpSel);
 88 :     //----- ダイアログ項目のセーブ -------------------//
 89 :     AjcSaveAllControlSettings(hDlg);
 90 :     //----- プログラム終了 ---------------------------//
 91 :     PostQuitMessage(0);
 92 :     return TRUE;
 93 : }
 94 : //----- WM_COMMAND ------------------------------------------------------------------------------------------------//
 95 : AJC_DLGPROC(Main, WM_COMMAND           )
 96 : {
 97 :     int     id  = LOWORD(wParam);
 98 :     int     cmd = HIWORD(wParam);
 99 :
100 :     if ((cmd == AJCIVN_INTVALUE  && (id == IDC_INP_COR_R || id == IDC_INP_COR_G || id == IDC_INP_COR_B ||
101 :                                      id == IDC_INP_EXC_R || id == IDC_INP_EXC_G || id == IDC_INP_EXC_B))  ||
102 :         (cmd == AJCIVN_REALVALUE && (id == IDC_INP_COR_LEN || id == IDC_INP_EXC_LEN))) {
103 :
104 :         ShowPic();
105 :     }
106 :
107 :     return TRUE;
108 : }
109 : //----- ウインドクローズ ------------------------------------------------------------------------------------------//
110 : AJC_DLGPROC(Main, IDCANCEL              )
111 : {
112 :     DestroyWindow(hDlg);
113 :     return TRUE;
114 : }
115 : //-----------------------------------------------------------------------------------------------------------------//
116 : AJC_DLGMAP_DEF(Main)
117 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG )
118 :     AJC_DLGMAP_MSG(Main, WM_DESTROY    )
119 :     AJC_DLGMAP_MSG(Main, WM_COMMAND    )
120 :     AJC_DLGMAP_CMD(Main, IDCANCEL      )
121 : AJC_DLGMAP_END
122 :
123 : //-----------------------------------------------------------------------------------------------------------------//
124 : //  各ピクチャ表示                                                                                               //
125 : //-----------------------------------------------------------------------------------------------------------------//
126 : static VO ShowPic(VO)
127 : {
128 :     //----- 設定値取得 -------------------------------------------------------//
129 :     COLORREF    RgbCor = RGB(AjcGetDlgItemUInt(hDlgMain, IDC_INP_COR_R),
130 :                              AjcGetDlgItemUInt(hDlgMain, IDC_INP_COR_G),
131 :                              AjcGetDlgItemUInt(hDlgMain, IDC_INP_COR_B));
132 :
133 :     COLORREF    RgbExc = RGB(AjcGetDlgItemUInt(hDlgMain, IDC_INP_EXC_R),
134 :                              AjcGetDlgItemUInt(hDlgMain, IDC_INP_EXC_G),
135 :                              AjcGetDlgItemUInt(hDlgMain, IDC_INP_EXC_B));
136 :
137 :     double      LenCor = AjcGetDlgItemReal(hDlgMain, IDC_INP_COR_LEN);
138 :     double      LenExc = AjcGetDlgItemReal(hDlgMain, IDC_INP_EXC_LEN);
139 :
140 :     //----- 各ピクチャ・クリアー ---------------------------------------------//
141 :     AjcDibClear(&DbifCor, RGB_WHITE);
142 :     AjcDibClear(&DbifExc, RGB_WHITE);
143 :     AjcDibClear(&DbifSel, RGB_WHITE);
144 :
145 :     //----- 各ピクチャへ原画から色を選択／除外してコピー ---------------------//
146 :     AjcDibCorrectedCopy(&DbifCor, 0, 0, 0, 0, &DbifOrg, 0, 0, RgbCor, LenCor);
147 :     AjcDibExcludedCopy (&DbifExc, 0, 0, 0, 0, &DbifOrg, 0, 0, RgbExc, LenExc);
148 :     AjcDibSelectedCopy (&DbifSel, 0, 0, 0, 0, &DbifOrg, 0, 0, 2,
149 :                         AJCDIBOPE_CORRECT, RgbCor, LenCor,
150 :                         AJCDIBOPE_EXCLUDE, RgbExc, LenExc);
151 :
152 :     //----- 各ピクチャへ表示 -------------------------------------------------//
153 :     SendDlgItemMessage(hDlgMain, IDC_PIC_COR, STM_SETIMAGE, IMAGE_BITMAP, (LPARAM)hBmpCor);
154 :     SendDlgItemMessage(hDlgMain, IDC_PIC_EXC, STM_SETIMAGE, IMAGE_BITMAP, (LPARAM)hBmpExc);
155 :     SendDlgItemMessage(hDlgMain, IDC_PIC_SEL, STM_SETIMAGE, IMAGE_BITMAP, (LPARAM)hBmpSel);
156 : }
```

49. テキスト描画

ＤＣ（デバイスコンテキスト）へテキストを描画する為の関数群です。
テキストに、改行（'¥n'）や、エスケープシーケンスを含めることができます。

49.1. 描画域とテキスト描画域，文字列描画域

描画域とは、ウインドやＤＣ描画域全体を意味します。
テキスト描画域は、描画域の全体、あるいは、描画域の右下部分に設定します。
テキスト描画域の原点位置は、AjcTxoCreate()の left,top 引数で指定します。
文字列描画域とは、１回の文字列描画で描画される文字列の外枠を意味します。
文字列描画域の大きさは、横幅＝最も長い行の文字列のピクセル数、高さ＝(文字の高さ＋行間スペース)×行数 となります。



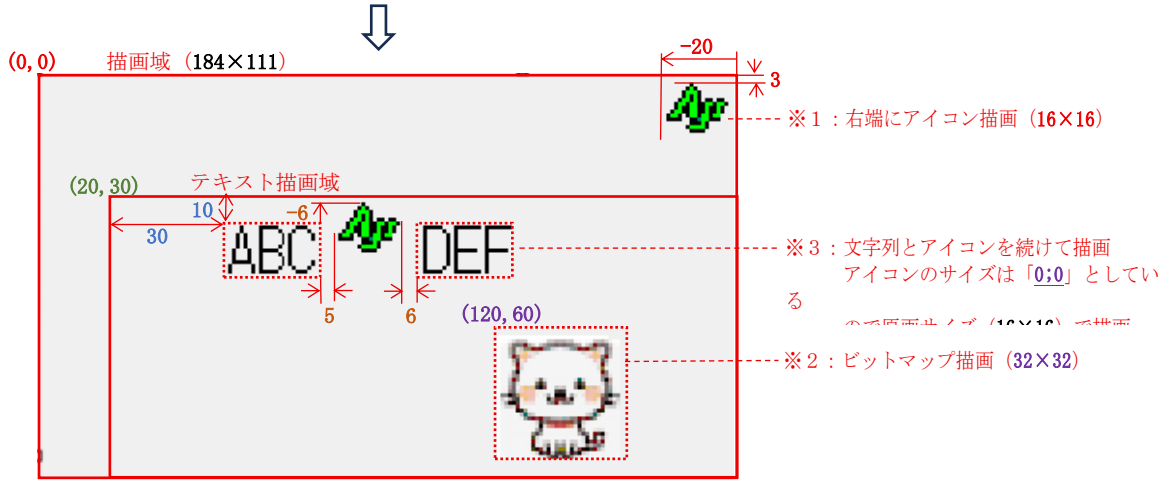
※１：AjcTxoPrintFExEx() / AjcTxoTextOutExEx() の場合は 引数(cx)で指定した値

49.2. イメージ（ビットマップ，アイコン）の描画

イメージ（ビットマップやアイコン）を描画するには、AjcTxoRegist*XXX*()で、あらかじめイメージを登録しておき、文字列中にエスケープシーケンス("¥x1B[*id*・・・**z**" や "¥x1B[*id*・・・**r**")を指定します。
エスケープシーケンスの末尾が 'z' の場合は、描画域の指定位置にイメージを描画します。
エスケープシーケンスの末尾が 'r' の場合は、文字列に続けて（文字列の末尾からの相対位置に）イメージを描画し、文字列描画位置（X）を更新します。

ex.

```
HICON   hIco = (HICON   )LoadImage(hInst, MAKEINTRESOURCE(IDI_BMP), IMAGE_ICON   , 16, 16, LR_DEFAULTCOLOR);
HBITMAP hBmp = (HBITMAP)LoadImage(hInst, MAKEINTRESOURCE(IDB_ICO), IMAGE_BITMAP, 48, 48, LR_DEFAULTCOLOR);
HAJCTXO hTxo;
・・・
if (hTxo = AjcTxoCreate(hdc, 20, 30, -20)) {              // テキスト描画オブジェクト生成（行間スペース：文字高さの 20%）
    AjcTxoSetDcBmpSize(hTxo, 184, 111);                   // 描画域サイズ設定(184 × 111)
    AjcTxoRegistIcon  (hTxo, hIco, 0, 0, 0, 0, 0);        // アイコン登録     (id = 0)
    AjcTxoRegistBitmap(hTxo, hBmp, 1, 0, 0, 0, 0);        // ビットマップ登録(id = 1)
    ・・・
    AjcTxoTextOut(hTxo, TEXT("¥x1B[0;-20;3;16;16z"));                 // 描画域右上にアイコン表示（x = 画面右端 - 16)   ※1
    AjcTxoTextOut(hTxo, TEXT("¥x1B[1;120;60;32;32z"));                // 描画域の位置(120, 60)の位置にビットマップ表示  ※2
    AjcTxoTextOutEx(hTxo, 30, 10, TEXT("ABC¥x1B[0;5;-6;0;0;6rDEF")); // "ABC",アイコンと"DEF"を続けて表示               ※3
    ・・・
    AjcTxoDelete(hTxo, TRUE);                             // テキスト描画オブジェクト破棄
}
```



49.3. エスケープシーケンス

使用できるエスケープシーケンスは、以下のとおりです。

エスケープシーケンス

#	エスケープシーケンス	内容		
1	“¥x1B[0**m**”	文字色, 文字背景色とフォント（太字, 斜字, 下線）をリセット		（※３）
2	“¥x1B[1**m**”	ボールド（太字）設定 **(※4)**		
3	“¥x1B[3**m**”	イタリック（斜字）設定 **(※4)**		
4	“¥x1B[4**m**”	アンダーライン（下線）設定 **(※4)**		
5	“¥x1B[9**m**”	文字色と文字背景色をリセット（パレット０, 透明）		
6	“¥x1B[30**m**” ～ “¥x1B[37**m**”	文字色の設定　（3***X***の ***X***はパレット番号 **※１**）		
7	“¥x1B[38;2;*r*;*g*;*b***m**” **（※２）**	文字色をＲＧＢで指定（r,g,b は、各々 0～255）		
8	“¥x1B[39**m**”	文字色リセット（パレット０（デフォルトは黒））		
9	“¥x1B[40**m**” ～ “¥x1B[47**m**”	文字背景色の設定（4***X***の ***X***はパレット番号 **※１**）		
10	“¥x1B[48;2;*r*;*g*;*b***m**” **（※２）**	文字背景色をＲＧＢで指定（r,g,b は、各々 0～255）		
11	“¥x1B[49**m**”	文字背景色リセット		
12	“¥x1B[**T**”	ボールド（太字）設定（“¥x1B[1m”と同じ） **(※4)**		
13	“¥x1B[**t**”	ボールド（太字）解除 **(※4)**		
14	“¥x1B[**I**”	イタリック（斜字）設定（“¥x1B[3m”と同じ） **(※4)**		
15	“¥x1B[**i**”	イタリック（斜字）解除 **(※4)**		
16	“¥x1B[**U**”	アンダーライン（下線）設定（“¥x1B[3m”と同じ） **(※4)**		
17	“¥x1B[**u**”	アンダーライン（下線）解除 **(※4)**		
18	“¥x1B[**N**”	ボールド（太字）、 イタリック（斜字）、アンダーライン（下線）解除		
19	“¥x1B[*n***L**”	次の改行(¥n)で、行間スペースを、文字高さの*n*[%]とする。(ワンタイム)		
20	“¥x1B[*id*;*x*;*y*;*cx*;*cy***z**”	描画域上に イメージ描画	x,y:描画域上の描画位置（省略時は 0, 0） 負数時はテキスト描画域右端／下端からの減算値	id　：イメージ登録 ID cx,cy ： 描画サイズ 省略時は、(0, 0) 各々 0 は、原画サイズ
21	“¥x1B[*id*;*x*;*y*;*cx*;*cy*;*mrg***r**”	文字列に続けて イメージ描画	x,y:描画位置（省略時は 0, 0） 現描画位置からの相対値 mrg ： イメージ右端の余白（省略時は 0）	

※１：デフォルトのパレット色は、0：黒, 1: 赤 2：緑, 3：黄, 4：青, 5：紫, 6：水色, 7：白

※２：色の成分値は各々 0～255（*r*：赤, *g*：緑, *b*：青）

※３：複数指定時は、セミコロンで区切ってまとめる（ex. “¥x1B[1;31;43m” - ボールド, 文字色＝赤, 文字背景色＝黄）

※４：ボールド、イタリックやアンダーラインの設定／解除はネスト制御します （設定と同数の解除で解除）

上記以外のエスケープシーケンスは、何もせずに読み飛ばします。
エスケープシーケンスとは、’¥x1B’ で始まり、英字(a-z / A-Z)で終了する文字列とします。

49.4. 制御文字

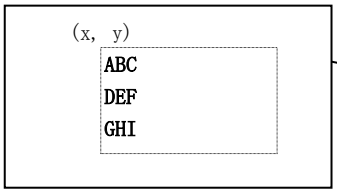
使用できる制御文字（0x00 ～ 0x1F または 0x7F）は、以下のとおりです。

制御コード

#	制御コード		内容
1	'¥x1B'	0x1B	エスケープシーケンスの始まり
2	'¥t'	0x09	タブステップ数の倍数の位置までＸ位置を進める。 あるいは、タブステップ数だけＸ位置を進める。　（後述の「タブ('¥t')の扱い」参照）
3	'¥r'	0x0D	テキスト描画域の左端までＸ位置を戻す（左詰め(AJCTXOALIGN_LEFT)時のみ）
4	'¥n'	0x0A	先に'¥r'があった場合は、テキスト描画域の左端までＸ位置を戻し改行する（ワンタイム） その他の場合は、AjcTxoTextOut[Ex]()や、AjcTxoPrintF()開始時のＸ位置へ戻し、改行する（Y位置を文字の高さ+行間スペース分進める）
5	'¥f'	0x0C	描画位置をＸ方向に１ピクセル進める
6	'¥a'	0x07	描画位置をＸ方向に１０ピクセル進める
7	上記以外の制御文字		無視（何もせずに読み飛ばします）

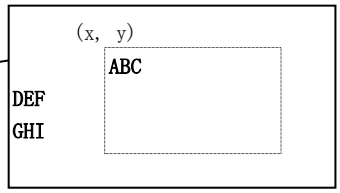
※１：左詰め(AJCTXOALIGN_LEFT)の場合、'¥r'は、テキスト描画域の左端へ移動します。

AjcTxoTextOutEx(hTxo, x, y, "ABC**¥n**DEF**¥n**GHI**¥n**");



'¥n'で、指定したｘ位置へ復帰し改行する

AjcTxoTextOutEx(hTxo, x, y, "ABC**¥r¥n**DEF¥**r¥n**GHI**¥r¥n**");



'¥r¥n'で、テキスト描画域の左端へ復帰し、改行する

テキスト描画域

49.5. テキストを右端，下端や中央に表示

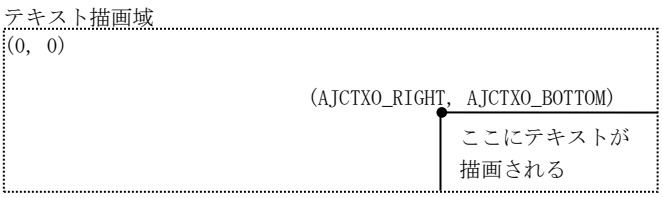
AjcTxoTextOutEx()や、AjcTxoPrintFEx()では、テキスト描画域の右端、下端や中央に文字列を描画することができます。
テキストの描画域の右端や下端に文字列を描画する場合は、AjcTxoSetDcBmpSize()で、描画域のビットマップサイズを設定しておかなければなりません。

文字列をテキスト描画域の右端に表示するには、ｘに「AJCTXO_RIGHT ± n」(n = 0～16000) を指定します。
この場合、指定した「n」は、文字列を右端に表示する位置からの相対位置となります。

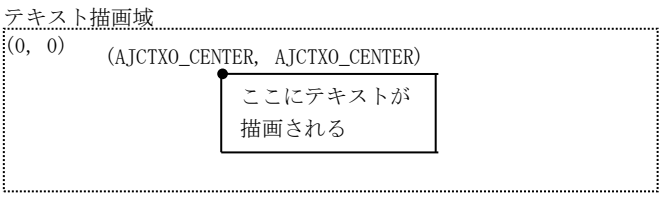
文字列をテキスト描画域の下端に表示するには、ｙに「AJCTXO_BOTTOM ± n」(n = 0～16000) を指定します。
この場合、指定した「n」は、文字列を下端に表示する位置からの相対位置となります。

文字列をテキスト描画域の中央に表示するには、x, ｙに「AJCTXO_CENTER ± n」(n = 0～16000) を指定します。
この場合、指定した「n」は、文字列を中央に表示する位置からの相対位置となります。

つまり、x = AJCTXO_RIGHT, y = AJCTXO_BOTTOM は、文字列を文字列描画域の右下隅に表示するための位置を意味します。



また、x = AJCTXO_CENTER, y = AJCTXO_CENTER は、文字列を文字列描画域の中央に表示するための位置を意味します。



49.6. タブ('¥t')の扱い

タブ('¥t')は、AjcTxoSetAlign()による文字列の整列設定や、文字列をテキスト描画域の右端(x=「AJCTXO_RIGHT ±n」)や中央(x=「AJCTXO_CENTER ±n」)に表示するかにより、以下のように扱われます。

#	文字列の整列 (AJCTXOALIGN_*XXX*)	右端／中央指定 (AJCTXO_RIGHT/CENTER)	タブの扱い	備考
1	LEFT(左詰め)	なし	タブステップ数の倍数の位置までX位置を進める。	
2		あり	タブステップ数だけX位置を進める	
3	CENTER(中央合わせ)	－		
4	RIGHT(右詰め)	－		
5	TAB_RIGHT(TAB 以降右詰め)	－	後続文字列を右詰め	行内で最初のタブのみ有効 2つ目以降のタブは無視

49.7. サポートＡＰＩ

テキスト描画のサポートＡＰＩ一覧を以下に示します。

#	関 数 名	内容
1	AjcTxoCreate	インスタンスの生成
2	AjcTxoDelete	インスタンスの消去
3	AjcTxoSetDcBmpSize	描画域のビットマップサイズ設定
4	AjcTxoSetDefTextColor	デフォルトの文字色設定
5	AjcTxoSetDefBkColor	デフォルトの文字背景色設定
6	AjcTxoSetAlign / AjcTxoGetAlign	文字列の整列
7	AjcTxoLocate / AjcTxoSetPos	描画位置設定
8	AjcTxoGetPos	描画位置取得
9	AjcTxoPrintF[Ex]	書式テキスト描画
10	AjcTxoTextOut[Ex][Ex]	テキスト描画
11	AjcTxoGetLineCount	文字列の行数取得
12	AjcTxoGetExtent[Ex][Ex]	文字列描画の矩形サイズ取得
13	AjcTxoSetPalette / AjcTxoSetPaletteItem	パレットの設定
14	AjcTxoGetPalette / AjcTxoGetPaletteItem	パレットの取得
15	AjcTxoGetCharSize	文字サイズ取得
16	AjcTxo{Set/Get}LineSpace	行間スペース設定／取得
17	AjcTxo{Set/Get}TabStep	タブステップ設定／取得
18	AjcTxo{Set/Get}BkMode	文字背景モード設定／取得
19	AjcTxoEnumEsc	ＥＳＣ文字列の列挙
20	AjcTxoGetRgbInEsc	ＲＧＢ表現文字列の解析
21	AjcTxoRegistBitmap	描画イメージ（ビットマップ）の登録
22	AjcTxoRegistIcon	描画イメージ（アイコン）の登録
23	AjcTxoUnregistImage	描画イメージの登録解除

49.7.1. インスタンスの生成 (AjcTxoCreate)

形　式 ： HAJCTXO　AjcTxoCreate (HDC hdc, int left, int top, int defLsp);

引　数 ： hdc　　- デバイスコンテキスト・ハンドル
left　- テキスト描画域の左端描画開始位置
top　　- テキスト描画域の上端描画開始位置
defLsp - デフォルトの行間スペース（正数：ピクセル数，負数：文字高さのＮ％）

説　明 ： テキスト描画のインスタンスを生成します。
left, top で描画する原点を指定します。



各テキスト描画ＡＰＩでは、(left, top)で指定した位置を原点(0, 0)とします。
defLsp は、改行（"¥n"描画）時の行間スペースを指定します。
defLsp＝正数の場合はｍピクセル数で行間スペースを指定します。(defLsp = 20 → 20 ピクセル)
defLsp＝負数の場合は、文字の高さのＮ％が行間スペースとなります。(defLsp = -20 → 20%)

デフォルト文字色は「黒」に、デフォルト文字背景色は「透明」に設定されます。

戻り値 ： ≠NULL - 成功（インスタンスハンドルを返します）
＝NULL - 失敗

備　考 ： 文字の高さは、本ＡＰＩ実行時にＤＣに割り当てられているフォントから取得します。
AjcTxoCreate()～AjxTxoDelete()の間は、フォントを変更しないでください。

49.7.2. インスタンスの消去 (AjcTxoDelete)

形　式 ： BOOL　AjcTxoDelete(HAJCTXO hTxo, BOOL fRestore);

引　数 ： hTxo　　- インスタンス・ハンドル
fRestore - ＤＣ属性回復フラグ

説　明 ： AjcTxoCreate()で確保したリソースを解放します。
fRestore=TRUE を指定した場合は、ＤＣの属性を、AjcTxoCreate()実行前の状態に戻します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

49.7.3. 描画域のビットマップサイズ設定 (AjcTxoSetDcBmpSize)

形　式 ： BOOL　AjcTxoSetDcBmpSize(HAJCTXO hTxo, int cx, int cy);

引　数 ： hTxo　　- インスタンス・ハンドル
cx　　- 描画域の幅　（ピクセル数）
cy　　- 描画域の高さ（ピクセル数）

説　明 ： 描画域（ウインドのサイズや、ＤＣに割り当てられたビットマップ）のサイズを指定します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

49.7.4. デフォルトの文字色設定(AjcTxoSetDefTextColor)

形　式 ： COLORREF　AjcTxoSetDefTextColor (HAJCTXO hTxo, COLORREF defColor);

引　数 ： hTxo　　- インスタンス・ハンドル
defColor - デフォルト文字色

説　明 ： デフォルトの文字色を設定します。
描画する文字色と文字背景色は、デフォルトにリセットされます。

戻り値 ： ≠ -1　- 成功（前回のデフォルト文字色）
＝ -1　- 失敗

備　考 ： エスケープシーケンスの ”¥x1B[39m” や “¥x1B[0m” でリセットした場合の文字色となります。

49.7.5. デフォルトの文字背景色設定(AjcTxoSetDefBkColor)

形　式 ： COLORREF　AjcTxoSetDefBkColor(HAJCTXO hTxo, COLORREF defColor);

引　数 ： hTxo　　- インスタンス・ハンドル
defColor - デフォルト文字背景色（-1 : 透明）

説　明 ： デフォルトの文字背景色を設定します。defColor = -1 とした場合は、文字背景色は透明となります。
描画する文字色と文字背景色は、デフォルトにリセットされます。

戻り値 ： TRUE　- 成功（前回の文字背景色）
FALSE - 失敗

備　考 ： エスケープシーケンスの ”¥x1B[49m” や “¥x1B[0m” でリセットした場合の文字背景色となります。

49.7.6. 文字列の整列(AjcTxoSetAlign / AjcTxoGetAlign)

形　式 ： BOOL　　AjcTxoSetAlign(HAJCTXO hTxo, AJCTXOALIGN Align);
AJCTXOALIGN　AjcTxoGetAlign(HAJCTXO hTxo);

引　数 ： hTxo　　- インスタンス・ハンドル
Align　- 文字の整列指定 (AJCTXOALIGN_*XXXXX*)

説　明 ： AjcTxoSetAlign()は、複数行の文字列描画時、文字列描画域内での文字列の整列を指定します。
Align には以下のいずれかのシンボルを指定します。

#	シンボル	内容	備考		
1	AJCTXOALIGN_LEFT	左詰め	描画域左詰め	タブは、可変幅(タブステップ倍数位置まで)更新	
			描画域中央／右詰め	タブは、固定幅(タブステップ数)更新	
2	AJCTXOALIGN_CENTER	中央合わせ	タブは、固定幅(タブステップ数)更新		復帰('¥r')は無視
3	AJCTXOALIGN_RIGHT	右詰め			
4	AJCTXOALIGN_TAB_RIGHT	TAB('¥t')以降を右詰め	２つ目以降のタブ('¥t')は無視		

例）
```
AjcTxoTextOut(hTxo,
    TEXT("テキスト整列の描画例¥n")
    TEXT("February¥t(2)¥n")
    TEXT("March¥t(3) "));
```
⇨

AJCTXOALIGN_LEFT
テキスト整列の描画例
February (2)
March (3)

AJCTXOALIGN_CENTER
テキスト整列の描画例
February (2)
March (3)

AJCTXOALIGN_RIGHT
テキスト整列の描画例
February (2)
March (3)

AJCTXOALIGN_TAB_RIGHT
テキスト整列の描画例
February [2]
March [3]

AjcTxoGetAlign()は、現在の文字列整列指定を取得します。

戻り値 ：

AjcTxoSetAlign()	AjcTxoGetAlign()
TRUE　- 成功	≠０：文字列整列指定値
FALSE - 失敗	＝０：失敗

49.7.7. 描画位置設定(AjcTxoLocate / AjcTxoSetPos)

形　式 ： BOOL AjcTxoLocate (HAJCTXO hTxo , int x, int y);
BOOL AjcTxoSetPos ((HAJCTXO hTxo, LPPOINT pPt) ;

引　数 ： hTxo - インスタンス・ハンドル
x - 描画ピクセルＸ位置
y - 描画ピクセルＹ位置
pPt - 描画ピクセル位置へのポインタ

説　明 ： 文字列の描画位置を、「テキスト描画域」の左上を原点(0, 0)とした値で指定します。

戻り値 ： TRUE - 成功
FALSE - 失敗

49.7.8. 描画位置取得(AjcTxoGetPos)

形　式 ： BOOL AjcTxoGetPos(HAJCTXO hTxo, LPPOINT pPt);

引　数 ： hTxo - インスタンス・ハンドル
pPt - 現在の描画ピクセル位置を格納するバッファのアドレス

説　明 ： 現在の描画ピクセル位置を取得します。

戻り値 ： TRUE - 成功
FALSE - 失敗

49.7.9. 書式テキスト描画(AjcTxoPrintF)

形　式 ： BOOL AjcTxoPrintF (HAJCTXO hTxo, C_UTP pFmt, ...);
BOOL AjcTxoPrintFEx (HAJCTXO hTxo, int x, int y, C_UTP pFmt, ...);
BOOL AjcTxoPrintFExEx(HAJCTXO hTxo, int x, int y, int cx, C_UTP pFmt, ...);

引　数 ： hTxo - インスタンス・ハンドル
x - 描画ピクセルX位置 (テキストを右隅／中央に表示する場合は、「AJCTXO_RIGHT ± n」or「AJCTXO_CENTER ± n」を指定)
y - 描画ピクセルY位置 (テキストを下隅／中央に表示する場合は、「AJCTXO_BOTTOM ± n」or「AJCTXO_CENTER ± n」を指定)
cx - 文字列描画域の幅 (＝０：自動算出, ≠０：文字列描画域の幅を指定)
pFmt - 書式文字列テキストへのポインタ (書式の形式は printf()と同じ)

説　明 ： pFmt で書式化されたテキストを描画します。
テキスト描画域の右端を超えた場合、行の折り返しは行われません。
書式文字列の形式は、printf()と同じです。
但し、書式化後の文字数は、最大２０４７バイト／文字であり、超えた部分はカットされます。
描画ピクセル位置は、描画したテキストの直後に更新されます。
cx≠0 を指定した場合は、指定値を文字列描画域の幅として、文字列の整列（右詰め等）を行います。

戻り値 ： TRUE - 成功
FALSE - 失敗

49.7.10. テキスト描画(AjcTxoTextOut / AjcTxoTextOutEx)

形　式 ： BOOL　AjcTxoTextOut　　(HAJCTXO hTxo, C_BCP pText);
BOOL　AjcTxoTextOutEx　(HAJCTXO hTxo, int x, int y, C_BCP pText);
BOOL　AjcTxoTextOutExEx(HAJCTXO hTxo, int x, int y, int cx, C_BCP pText);

引　数 ： hTxo　－ インスタンス・ハンドル
x　　－ 描画ピクセルX位置 (テキストを右隅／中央に表示する場合は、「AJCTXO_RIGHT ± n」or「AJCTXO_CENTER ± n」を指定)
y　　－ 描画ピクセルY位置 (テキストを下隅／中央に表示する場合は、「AJCTXO_BOTTOM ± n」or「AJCTXO_CENTER ± n」を指定)
cx　　－ 文字列描画域の幅 (＝０：自動算出, ≠０：文字列描画域の幅を指定)
pText　－ 描画するテキストへのポインタ

説　明 ： pText で指定されたテキストを描画します。
テキスト描画域の右端を超えた場合、行の折り返しは行われません。
AjcTxoTextOut()は、現在の描画ピクセル位置から描画を開始します。
AjcTxoTextOutEx()は、x, y で指定された描画ピクセル位置から描画を開始します。
描画ピクセル位置は、描画したテキストの直後に更新されます。
cx≠0 を指定した場合は、指定値を文字列描画域の幅として、文字列の整列（右詰め等）を行います。

戻り値 ： TRUE　－ 成功
FALSE － 失敗

49.7.11. 文字列の行数取得(AjcTxoGetLineCount)

形　式 ： UTP　AjcTxoGetLineCount (HAJCTXO hTxo, C_UTP pText);

引　数 ： hTxo　－ インスタンス・ハンドル
pText　－ 文字列へのポインタ

説　明 ： pText で指定した文字列の行数を取得します。
文字列中の改行文字('¥n')の個数＋１の値を返します。
文字列中に改行文字が１つも無い場合は、１を返します。

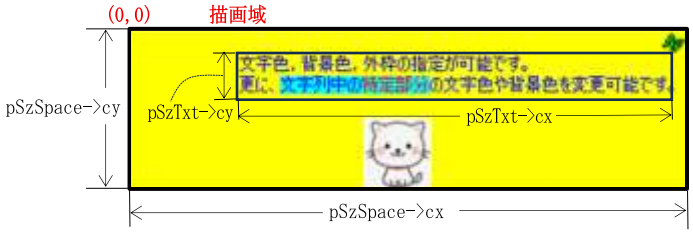
戻り値 ： ≠0 － 文字列の行数（文字列に含まれる '¥n' の数 ＋ １）
＝0 － 失敗

49.7.12. 文字列描画の矩形サイズ取得(AjcTxoGetExtent[Ex][Ex])

形　式 ： UI AjcTxoGetExtent (HAJCTXO pW, C_UTP pText, LPSIZE pSzTxt));
UI AjcTxoGetExtentEx (HAJCTXO pW, int x, int y, C_UTP pText, LPSIZE pSzTxt, LPSIZE pSzSpace);
UI AjcTxoGetExtentExEx(HAJCTXO pW, int x, int y, C_UTP pText, LPSIZE pSzTxt, LPSIZE pSzSpace,
UI LinePixs[], UI nLine);

引　数 ： hTxo – インスタンス・ハンドル
x, y – 描画ピクセル位置（未指定時は、x=0, y=0 を仮定）
pText – 文字列へのポインタ
pSzTxt – 文字列描画の矩形ピクセルサイズを格納するバッファ（不要時は NULL）
pSzSpace – 文字列とイメージ（BitMap/Icon）全体を描画するのに必要な描画域のサイズを格納するバッファ（不要時は NULL）
LinePixs – 各行の描画ピクセル数を格納する配列バッファのアドレス（不要時は NULL）
nLine – 各行の描画ピクセル数を格納する配列バッファの要素数

説　明 ： pText で指定した文字列を描画する場合の、矩形のピクセルサイズを算出し、pSzTxt, pSzImg で示すバッファに格納します。



戻り値 ： ≠0 – 文字列の描画行数（文字列に含まれる ‘¥n’ の数 ＋ 1）
＝0 – 失敗

備　考 ： ・文字列中にＴＡＢ(‘¥t’)が含まれる場合は、AjcTxoGetExtentEx[Ex]()を使用してください。
・文字列中にイメージ描画（“¥x1B[*id*;*x*;*y*;*cx*;*cy*z”）を含む場合、*x*, *y* に負数を指定することはできません。
負数を指定した場合は、ゼロを仮定します。

49.7.13. パレットの設定(AjcTxoSetPalette / AjcTxoSetPaletteItem)

形　式 ： BOOL　AjcTxoSetPalette　　(HAJCTXO hTxo, COLORREF rgb[8]);
BOOL　AjcTxoSetPaletteItem(HAJCTXO hTxo, int ix, COLORREF rgb)

引　数 ： hTxo　 - インスタンス・ハンドル
rgb[8] - 設定するパレット色全８色へのポインタ
ix　　- 設定するパレット色のインデクス（０～７）
rgb　 - 設定するパレット色

説　明 ： AjcTxoSetPalette()は、パレット色全８色を設定します。
AjcTxoSetPaletteItem()は、ix で指定した１色を設定します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

備　考 ： このパレット色は、エスケープシーケンスの “¥x1B[30m” ～ “¥x1B[37m”や、“¥x1B[40m” ～ “¥x1B[47m”で設定する文字色や、文字背景色に対応します。

49.7.14. パレットの取得(AjcTxoGetPalette / AjcTxoGetPaletteItem)

形　式 ： BOOL　　AjcTxoGetPalette　　(HAJCTXO hTxo, BOOL fRestore);
COLORREF　AjcTxoGetPaletteItem(HAJCTXO hTxo, int ix);

引　数 ： hTxo　 - インスタンス・ハンドル
rgb[8] - 取得したするパレット色全８色を格納するバッファへのポインタ
ix　　- 設定するパレット色のインデクス（０～７）

説　明 ： AjcTxoGetPalette()は、パレット色全８色を取得します。
AjcTxoGetPaletteItem()は、ix で指定した１色を取得します。

戻り値 ： <AjcTxoGetPalette()>　　<AjcTxoGetPaletteItem()>
TRUE　- 成功　　≠ -1 ： パレットの色
FALSE - 失敗　　＝ -1 ： 失敗

49.7.15. 文字サイズ取得(AjcTxoGetCharSize)

形　式 ： BOOL　　AjcTxoGetCharSize　(HAJCTXO hTxo, , LPSIZE pSize);

引　数 ： hTxo　 - インスタンス・ハンドル
pSize　- 文字サイズを格納するバッファへのポインタ

説　明 ： 設定されたフォントの文字サイズ（ピクセル数）を取得します・

戻り値 ： TRUE　- 成功
FALSE - 失敗

49.7.16. 行間スペース設定／取得(AjcTxo{Set/Get}LineSpace)

形　式 ： UI　　AjcTxoSetLineSpace (HAJCTXO hTxo, int defLsp);
UI　　AjcTxoGetLineSpace (HAJCTXO hTxo);

引　数 ： hTxo　 - インスタンス・ハンドル
defLsp - デフォルトの行間スペース（正数：ピクセル数, 負数：文字高さのＮ％）

説　明 ： 行間スペース（ピクセル数）を設定／取得します。
defLsp は、改行（"¥n”描画）時の行間スペースを指定します。
defLsp＝正数の場合はｍピクセル数で行間スペースを指定します。(defLsp = 20 → 20 ピクセル)
defLsp＝負数の場合は、文字の高さのＮ％が行間スペースとなります。(defLsp = -20 → 20%)

戻り値 ： ≠-1：成功（行間スペース（AjcTxoSetLineSpace()の場合は前回値））
＝-1： - 失敗

49.7.17. タブステップ設定／取得(AjcTxo{Set/Get}TabStep)

形　式 ： int　AjcTxoSetTabStep (HAJCTXO hTxo, int TabStep);
int　AjcTxoGetTabStep (HAJCTXO hTxo);

引　数 ： hTxo　- インスタンス・ハンドル
TabStep - タブステップ幅（ピクセル数，０は設定不可，正数：可変幅，負数：固定幅）

説　明 ： タブステップ（ピクセル数）を設定／取得します。(デフォルトは、半角４文字分の文字幅)
TabStep＝正数の場合は、Ｘ位置を TabStep の整数倍の位置まで進めます。
TabStep＝負数、あるいは、TAB 以降右詰め(AjcTxoSetAlign()で「AJCTXOALIGN_TAB_RIGHT」を指定)の場合は、Ｘ位置を指定した値（TabStep の絶対値）だけ進めます。

戻り値 ： ≠０：成功（タブステップ（AjcTxoSetTabStep()の場合は前回値））
＝０： - 失敗

49.7.18. 文字背景モード設定／取得(AjcTxo{Set/Get}BkMode)

形　式 ： int　AjcTxoSetBkMode (HAJCTXO hTxo, int TabStep);
int　AjcTxoGetBkMode (HAJCTXO hTxo);

引　数 ： hTxo　- インスタンス・ハンドル
BkMode - タブステップ幅（AJCTXO_TRANSPARENT(透明) / AJCTXO_OPAQUE(不透明)）

説　明 ： 文字背景モードを設定／取得します。

戻り値 ： ≠０：成功（文字背景モード（AjcTxoSetBkMode()の場合は前回値））
＝０： - 失敗

49.7.19. ＥＳＣ文字列の列挙(AjcTxoEnumEsc)

形　式 ： UI　AjcTxoEnumEsc (C_BCP pText, UX cbp, BOOL (CALLBACK *cbNtcEsc)(C_BCP pEsc, UI lEsc, BC cLast, UX cbp));

引　数 ： pText　- インスタンス・ハンドル
cbp　- ＤＣ属性回復フラグ
cbNtcEsc - エスケースシーケンスをを通知するコールバック関数

説　明 ： 複数（あるいは１つ）のエスケースシーケンスで構成するエスケープ文字列から、エスケープシーケンスを列挙します。
各エスケープシーケンスは、コールバック関数（cbNtcEsc）をコールすることにより通知されます。

ex.　〈エスケープ文字列〉　　〈列挙するエスケープシーケンス〉
“¥x1B[31;43m¥x1B[30L”　→　“¥x1B[31;43m”
“¥x1B[30L”

戻り値 ： 列挙したエスケープシーケンスの数

コールバック：

cbNtcEsc（エスケープシーケンス通知）

形　式 ： BOOL CALLBACK *cbNtcEsc* (C_BCP pEsc, UI lEsc, BC cLast, UX cbp);

引　数 ： pEsc　- エスケープシーケンス文字列へのポインタ
lEsc　- エスケープシーケンス文字列のバイト数／文字数
cLast - エスケープシーケンス末尾の英字(‘a’－‘z’ / ‘A’－‘Z’)
cbp　- コールバックパラメタ

説　明 ： １つのエスケープシーケンスを通知します。

戻り値 ： TRUE　：エスケープシーケンスの通知を継続します。
FALSE　：エスケープシーケンスの通知を中止します。

49.7.20. ＲＧＢ表現文字列の解析(AjcTxoGetRgbInEsc)

形　式 ： UTP　AjcTxoGetRgbByText (C_UTP pText, COLORREF *pBuf);

引　数 ： pText　－ ＲＧＢ表現文字列(“*r*;*g*;*b*”)へのポインタ
pBuf　－ ＲＧＢ値を格納するバッファのアドレス

説　明 ： ＲＧＢ表現文字列(“*r*;*g*;*b*”)を解析し、ＲＧＢ値を取得します。
r, g, b は色の要素を示す値（０～２５５の１０進数）です。(r:赤. g：緑, b：青)
r, g, b を省略した場合は、ゼロを仮定します。(″255;;” → “255;0;0” と同じ)

戻り値 ： ≠NULL － ＲＧＢ表現文字列の直後のアドレス
＝NULL － 失敗（ＲＧＢ表現文字列以外）

49.7.21. 描画イメージ（ビットマップ）の登録 (AjcTxoRegistBitmap)

形　式 ： BOOL　AjcTxoRegistBitmap (HBITMAP hBmp , UI id, int x, int y, int cx, int cy);

引　数 ： hBmp　－ ビットマップハンドル
id　　－ 登録ＩＤ（０～）
x, y　－ デフォルト描画位置
cx, cy － デフォルト描画サイズ

説　明 ： エスケープシーケンス（”¥x1B[*id*・・・z”）で描画するビットマップを登録します。
この情報は、インスタンスに関係なく、固定的な（静的な）情報として登録されます。
x, y はデフォルトの描画位置で、エスケープシーケンスで x, y が指定されていない場合の値です。
cx, cy はデフォルトの描画サイズで、エスケープシーケンスで cx, cy が指定されていない場合の値です。
Id で指定したイメージが既に登録済みの場合は、旧イメージの登録は解除されます。

戻り値 ： TRUE　－ 成功
FALSE － 失敗

49.7.22. 描画イメージ（アイコン）の登録 (AjcTxoRegistIcon)

形　式 ： BOOL　AjcTxoRegistIcon (HICON hIcon , UI id, int x, int y, int cx, int cy);

引　数 ： hIcon　－ アイコンハンドル
id　　－ 登録ＩＤ（０～）
x, y　－ デフォルト描画位置
cx, cy － デフォルト描画サイズ

説　明 ： エスケープシーケンス（”¥x1B[*id*・・・z”）で描画するアイコンを登録します。
この情報は、インスタンスに関係なく、固定的な（静的な）情報として登録されます。
x, y はデフォルトの描画位置で、エスケープシーケンスで x, y が指定されていない場合の値です。
cx, cy はデフォルトの描画サイズで、エスケープシーケンスで cx, cy が指定されていない場合の値です。
Id で指定したイメージが既に登録済みの場合は、旧イメージの登録は解除されます。

戻り値 ： TRUE　－ 成功
FALSE － 失敗

49.7.23. 描画イメージの登録解除 (AjcTxoUnregistImage)

形　式 ： BOOL　AjcTxoUnregistImage (UI id);

引　数 ： id　　- 登録ＩＤ（０～、 -1：全て登録解除）

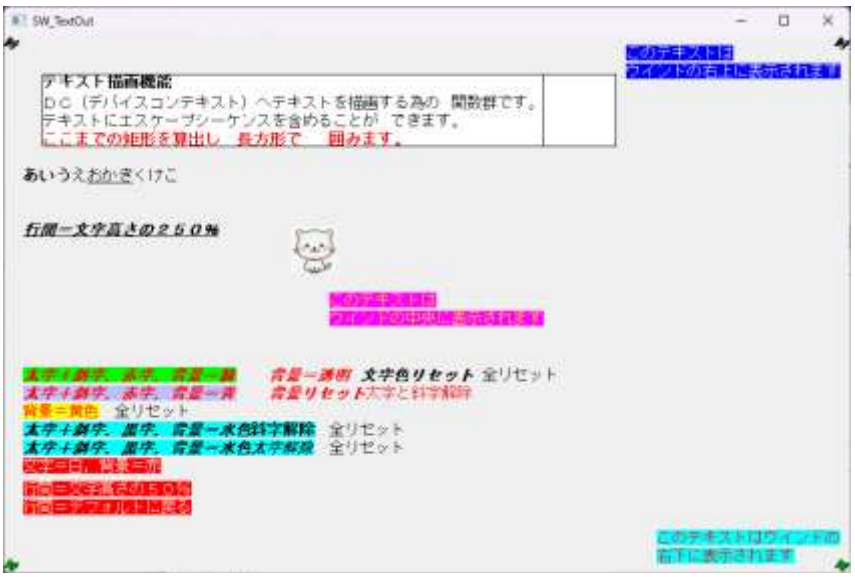
説　明 ： エスケープシーケンス（”¥x1B[*id*・・・z”）で描画するイメージの登録を解除します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

49.8. サンプルプログラム

49.8.1. SW_TextOut

次のサンプルプログラムは、エスケープシーケンスを含む文字列，アイコンとビットマップの描画例です。




```
 1 : //
 2 : //
 3 : //
 4 :
 5 : #include    <AjrCstXX.h>
 6 : #include    <math.h>
 7 : #include    <tchar.h>
 8 : #include    "resource.h"
 9 :
10 : #define LEFT 20
11 : #define TOP  10
12 :
13 : //-------------------------------------------------------------------------------------------------------------//
14 : //  ワーク                                                                                                     //
15 : //-------------------------------------------------------------------------------------------------------------//
16 : HINSTANCE      hInst;                      //  ＤＬＬインスタンスハンドル
17 : HWND           hWndBack;                   //  バックウインドハンドル（ダイアログと VT-100 の親ウインド）
18 : HICON          hIco2Bit;
19 : HICON          hIco4Bit;
20 : HBITMAP        hBmpCat;
21 : //-------------------------------------------------------------------------------------------------------------//
22 : //  内部サブ関数                                                                                               //
23 : //-------------------------------------------------------------------------------------------------------------//
24 : AJC_WNDPROC_DEF(Back);
25 :
26 : //=============================================================================================================//
27 : //                                                                                                             //
28 : //  ＷｉｎＭａｉｎ                                                                                             //
29 : //                                                                                                             //
30 : //=============================================================================================================//
31 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
32 : {
33 :     MSG         msg;
34 :     WNDCLASS    wc;
35 :     HBRUSH      hBru;
36 :
37 :     hInst = hInstance;
38 :
39 :     //  ブラシ生成
40 :     hBru = CreateSolidBrush(GetSysColor(COLOR_BTNFACE));
41 :     //  バックウインド生成
42 :     wc.style        = 0;                        wc.hCursor          = LoadCursor(NULL, IDC_ARROW);
43 :     wc.lpfnWndProc  = AJC_WNDPROC_NAME(Back);   wc.hbrBackground    = hBru;
44 :     wc.cbClsExtra   = 0;                        wc.lpszMenuName     = NULL;
45 :     wc.cbWndExtra   = 0;                        wc.lpszClassName    = TEXT("SW_TextOut");
46 :     wc.hInstance    = hInst;
47 :     wc.hIcon        = NULL;
48 :     RegisterClass(&wc);
```

```
 49 :
 50 :    hWndBack = CreateWindow(TEXT("SW_TextOut"), TEXT("SW_TextOut"),
 51 :                            WS_OVERLAPPEDWINDOW,
 52 :                            0, 0, 900, 600,
 53 :                            NULL, NULL, hInst, NULL);
 54 :
 55 :    // ウインド表示
 56 :    ShowWindow(hWndBack, iCmdShow);
 57 :
 58 :    // メッセージループ
 59 :    while (GetMessage(&msg, NULL, 0, 0)) {
 60 :        do {
 61 :        // if (IsDialogMessage(hDlgMain, &msg)) break;
 62 :            TranslateMessage(&msg);
 63 :            DispatchMessage (&msg);
 64 :        } while (0);
 65 :    }
 66 :
 67 :    // ブラシ破棄
 68 :    DeleteObject(hBru);
 69 :
 70 :    return (int)msg.wParam ;
 71 : }
 72 : //=============================================================================================================//
 73 : //                                                                                                             //
 74 : //  バックウインド・プロシージャ                                                                               //
 75 : //                                                                                                             //
 76 : //=============================================================================================================//
 77 : //----- WM_CREATE -----------------------------------------------------------------------------------------------//
 78 : AJC_WNDPROC(Back, WM_CREATE        )
 79 : {
 80 :    // アイコン読み出し
 81 :    hIco2Bit = (HICON  )LoadImage(hInst, MAKEINTRESOURCE(IDI_2BIT), IMAGE_ICON  , 16, 16, LR_DEFAULTCOLOR);
 82 :    hIco4Bit = (HICON  )LoadImage(hInst, MAKEINTRESOURCE(IDI_4BIT), IMAGE_ICON  , 16, 16, LR_DEFAULTCOLOR);
 83 :    hBmpCat  = (HBITMAP)LoadImage(hInst, MAKEINTRESOURCE(IDB_CAT ), IMAGE_BITMAP, 48, 48, LR_DEFAULTCOLOR);
 84 :
 85 :    // アイコン, ビットマップ登録
 86 :    AjcTxoRegistIcon  (hIco2Bit, 1, 0, 0, 0, 0);
 87 :    AjcTxoRegistIcon  (hIco4Bit, 0, 0, 0, 0, 0);
 88 :    AjcTxoRegistBitmap(hBmpCat , 2, 0, 0, 0, 0);
 89 :
 90 :    return 0;
 91 : }
 92 : //----- WM_DESTROY ----------------------------------------------------------------------------------------------//
 93 : AJC_WNDPROC(Back, WM_DESTROY       )
 94 : {
 95 :    // アイコン, ビットマップ登録解除
 96 :    AjcTxoUnregistImage(-1);
 97 :    // アイコン破棄
 98 :    DeleteObject(hIco2Bit);
 99 :    DeleteObject(hIco4Bit);
100 :    // プログラム終了
101 :    PostQuitMessage(0);
102 :    return 0;
103 : }
104 : //----- WM_SIZE -------------------------------------------------------------------------------------------------//
105 : AJC_WNDPROC(Back, WM_SIZE          )
106 : {
107 :    InvalidateRect(hwnd, NULL, TRUE);
108 :    return 0;
109 : }
110 : //----- WM_PAINT ------------------------------------------------------------------------------------------------//
111 : UT  SampleText[] =  TEXT("¥x1B[T テキスト描画機能¥x1B[t¥x1B[40L¥n")
112 :                     TEXT("ＤＣ（デバイスコンテキスト）へテキストを描画する為の¥t 関数群です。¥n")
113 :                     TEXT("テキストにエスケープシーケンスを含めることが¥t できます。¥n")
114 :                     TEXT("¥x1B[1;31m ここまでの矩形を算出し¥t 長方形で¥t 囲みます。¥x1B[0m");
115 :
116 : AJC_WNDPROC(Back, WM_PAINT         )
117 : {
118 :    PAINTSTRUCT ps;
119 :    HDC         hdc;
120 :    HFONT       hFont;
121 :    HAJCTXO     hTxo;
122 :    AJCTXOALIGN Align;
123 :    int         x, y, w, d, cx, cy;
124 :
125 :    hdc   = BeginPaint(hwnd, &ps);
126 :    hFont = AjcGetStockFont(AJCFID_FIX16);
127 :    hFont = SelectObject(hdc, hFont);
128 :    if (hTxo = AjcTxoCreate(hdc, LEFT, TOP, -20)) { //  左端=10, 改行＝文字高さの２０％
```

```
    SIZE    sz = {0};
    RECT    r;
    //  ＤＣ描画域サイズとしてウインドサイズを設定
    GetClientRect(hwnd, &r);
    AjcTxoSetDcBmpSize(hTxo, r.right - r.left, r.bottom - r.top);
    //--- テキストを描画し、テキスト部分を矩形で囲む ---//
    Align = AJCTXOALIGN_LEFT;   // AJCTXOALIGN_LEFT, AJCTXOALIGN_RIGHT, AJCTXOALIGN_CENTER or AJCTXOALIGN_TAB_RIGHT
    x = 20; y = 30; w = 600; d = w / 2;                   //  描画域のＸ，Ｙ，幅，幅÷２
    AjcTxoSetAlign(hTxo, Align);                          //  テキストアライン設定
    AjcTxoGetExtentEx(hTxo, x, y, SampleText, &sz, NULL);   //  テキスト描画サイズ取得
    cx = sz.cx; cy = sz.cy;
    AjcTxoTextOutExEx(hTxo, x, y, w, SampleText);          //  テキスト描画
    //  テキスト描画域の矩形表示
    SelectObject(hdc, GetStockObject(WHITE_BRUSH));
    Rectangle(hdc, LEFT + x, TOP + y, LEFT + x + w, TOP + y + sz.cy);
    AjcTxoTextOutExEx(hTxo, x, y, w, SampleText);          //  テキスト再描画
    //  テキストを矩形で囲む
    SelectObject(hdc, GetStockObject(NULL_BRUSH));
    switch (Align) {
     case AJCTXOALIGN_LEFT:     Rectangle(hdc, LEFT + x              , TOP + y, LEFT + x + sz.cx       , TOP + y + sz.cy); break;
     case AJCTXOALIGN_RIGHT:    Rectangle(hdc, LEFT + x + w - sz.cx  , TOP + y, LEFT + x + w           , TOP + y + sz.cy); break;
     case AJCTXOALIGN_CENTER:   Rectangle(hdc, LEFT + x + d - sz.cx / 2, TOP + y, LEFT + x + d + sz.cx / 2, TOP + y + sz.cy);
break;
     case AJCTXOALIGN_TAB_RIGHT: Rectangle(hdc, LEFT + x              , TOP + y, LEFT + x + w           , TOP + y + sz.cy); break;
    }

    AjcTxoSetAlign(hTxo, AJCTXOALIGN_LEFT); //  アライン＝左詰め

    AjcTxoTextOut(hTxo, TEXT("¥r¥n¥r¥n¥x1B[T あ¥x1B[T い¥x1B[t う¥x1B[t え¥x1B[U お¥x1B[U か¥x1B[u き¥x1B[u くけこ¥x1B[250L¥r¥n"));
    AjcTxoPrintF(hTxo, TEXT("¥x1B[T¥x1B[I¥x1B[U 行間＝文字高さの２５０%¥x1B[t¥x1B[i¥x1B[u¥r¥n↓¥r¥n↓¥r¥n↓¥r¥n↓¥r¥n↓¥r¥n↓¥r¥n↓
¥r¥n"));



    AjcTxoPrintF(hTxo, TEXT("¥x1B[1;3;31;42m")
                       TEXT("太字＋斜字，赤字，背景＝緑")
                       TEXT("¥x1B[49m")
                       TEXT("¥t 背景＝透明")
                       TEXT("¥x1B[39m")
                       TEXT("¥t 文字色リセット")
                       TEXT("¥x1B[0m")
                       TEXT("¥t 全リセット¥r¥n"));

    AjcTxoPrintF(hTxo, TEXT("¥x1B[T¥x1B[I¥x1B[38;2;255;0;0;48;2;192;192;255m")
                       TEXT("太字＋斜字，赤字，背景＝青")
                       TEXT("¥x1B[49m")
                       TEXT("¥t 背景リセット")
                       TEXT("¥x1B[N")
                       TEXT("太字と斜字解除¥r¥n"));

    AjcTxoPrintF(hTxo, TEXT("¥x1B[43m")
                       TEXT("背景＝黄色")
                       TEXT("¥x1B[0m")
                       TEXT("¥t 全リセット¥r¥n"));

    AjcTxoPrintF(hTxo, TEXT("¥x1B[1;3;30;46m")
                       TEXT("太字＋斜字，黒字，背景＝水色")
                       TEXT("¥x1B[i")
                       TEXT("斜字解除")
                       TEXT("¥x1B[0m")
                       TEXT("¥t 全リセット¥r¥n"));

    AjcTxoPrintF(hTxo, TEXT("¥x1B[1;3;30;46m")
                       TEXT("太字＋斜字，黒字，背景＝水色")
                       TEXT("¥x1B[t")
                       TEXT("太字解除")
                       TEXT("¥x1B[0m")
                       TEXT("¥t 全リセット¥r¥n"));

    AjcTxoPrintF(hTxo, TEXT("¥x1B[37;41m")
                       TEXT("文字＝白，背景＝赤")
                       TEXT("¥x1B[50L¥n")
                       TEXT("行間＝文字高さの５０%"));

    AjcTxoPrintF(hTxo, TEXT("¥r¥n 行間＝デフォルトに戻る¥r¥n"));

    AjcTxoTextOutEx(hTxo, AJCTXO_RIGHT - 10, 0,TEXT("¥x1B[37;44m")
                                               TEXT("このテキストは¥n")
                                               TEXT("ウインドの右上に表示されます"));

    AjcTxoTextOutEx(hTxo, AJCTXO_RIGHT - 10, AJCTXO_BOTTOM - 10,TEXT("¥x1B[31;46m")
```

```
207 :                                                   TEXT("このテキストはウインドの¥n")
208 :                                                   TEXT("右下に表示されます"));
209 :
210 :         AjcTxoTextOutEx(hTxo, AJCTXO_CENTER, AJCTXO_CENTER, TEXT("¥x1B[33;45m")
211 :                                                   TEXT("このテキストは¥n")
212 :                                                   TEXT("ウインドの中央に表示されます"));
213 :         AjcTxoTextOut(hTxo, TEXT("¥x1B[0m"));
214 :         //  四隅にアイコン表示
215 :         AjcTxoTextOut(hTxo, TEXT("¥x1B[0;0;0;0;0z"));       //  左上
216 :         AjcTxoTextOut(hTxo, TEXT("¥x1B[1;0;-16;0;0z"));     //  左下
217 :         AjcTxoTextOut(hTxo, TEXT("¥x1B[0;-16;0;0;0z"));     //  右上
218 :         AjcTxoTextOut(hTxo, TEXT("¥x1B[1;-16;-16;0;0z"));   //  右下
219 :         //  ビットマップ表示
220 :         AjcTxoTextOut(hTxo, TEXT("¥x1B[2;300;200z"));
221 :
222 :         //  インスタンス破棄
223 :         AjcTxoDelete(hTxo, TRUE);
224 :
225 :     }
226 :
227 :     SelectObject(hdc, hFont);
228 :     EndPaint(hwnd, &ps);
229 :
230 :     return 0;
231 : }
232 : //-------------------------------------------------------------------------------------------------------------------//
233 : AJC_WNDMAP_DEF(Back)
234 :     AJC_WNDMAP_MSG(Back, WM_CREATE     )
235 :     AJC_WNDMAP_MSG(Back, WM_DESTROY    )
236 :     AJC_WNDMAP_MSG(Back, WM_SIZE       )
237 :     AJC_WNDMAP_MSG(Back, WM_PAINT      )
238 : AJC_WNDMAP_END
```

50. 時計表示

アナログ時計を表示するＡＰＩです。
時計の表示イメージは、以下のとおりです。

非透明表示　　**透明表示**



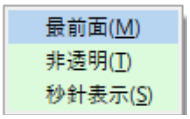


※透明表示時は、（オレンジ色の軸部分を除いて）マウス操作を透過します。
時計ウインド上のクリック等は、時計の下のウインドやデスクトップを
クリックすることになります。

時計をクリック（透明表示時はオレンジ色の軸部分をクリック）すると、時計を非表示します。
時計をドラッグ（透明表示時はオレンジ色の軸部分をドラッグ）すると、時計を移動できます。

ポップアップメニュー

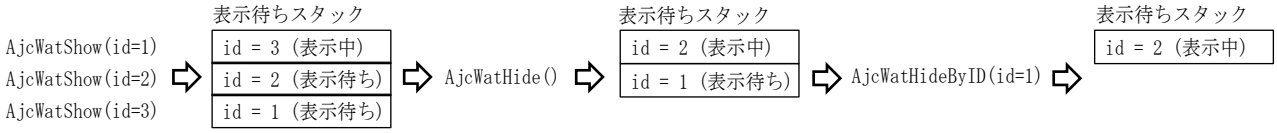
時計を右クリック（透明表示時はオレンジ色の軸部分を右クリック）すると、以下のようなポップアップメニューが表示されます。



ポップアップメニューを選択し、最前面に表示、透明表示や秒針表示の設定（あるいは解除）を切り替えられます。

多重表示

時計表示「AjcWatShow()」を連続した場合、表示が多重化されます。
この場合、時計非表示「AjcWatHide()」を実行すると、１つ前の表示状態が復元されます。



50.1. サポートＡＰＩ

時計表示 のサポートＡＰＩ一覧を以下に示します。

#	関 数 名	内容
1	AjcWatCreate	インスタンス生成
2	AjcWatDelete	インスタンス消去
3	AjcWatShow	時計表示
4	AjcWatHide AjcWatHideByID AjcWatHideAll	時計非表示
5	AjcWatSetShowState	時計表示状態設定
6	AjcWatGetShowState	時計表示状態取得
7	AjcWatGetHWnd	時計のウインドハンドル取得
8	AjcWatAdvance	時計を進める

50.1.1. インスタンス生成(AjcWatCreate)

形　式 ： HANDLE　AjcWatCreate (AjcWatCreate(HWND hOwner, UI dummy, C_UTP pSect);

引　数 ： hOwner - オーナーウインドハンドル（不要時は BNULL）
dummy　- 未使用（０を指定）
pSect　- プロファイルセクション名（ウインド位置の永続化用、不要時は NULL）

説　明 ： 時計表示のインスタンスを生成します。

戻り値 ： ≠NULL – 成功（＝インスタンスハンドル）
＝NULL - 失敗

50.1.2. インスタンス消去(AjcWatDelete)

形　式 ： BOOL　AjcWatDelete (HANDLE hWat);

引　数 ： hWat　- インスタンスハンドル

説　明 ： 時計表示のインスタンスを消去します。
AjcWatCreate()で、プロファイルセクション名が指定された（pSect≠NULL）場合は、ウインドの位置を当該プロファイルセクションに記録し、永続化します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

50.1.3. 時計表示(AjcWatShow)

形　式 ： BOOL　AjcWatShow (HANDLE hWat, UI id, UI state, C_BCP pSect);

引　数 ： hWat　- インスタンスハンドル
id　　- 識別ＩＤ
state　- 表示ステータス（AJCWATSS_*XXXX*）
pSect　- プロファイルセクション名（不要時は NULL）

説　明 ： 時計イメージのウインドを生成／表示し、これに識別ＩＤを付与します。
state は、以下のシンボルの合成値かゼロを指定します。

#	シンボル	内容	備考
1	AJCWATSS_DISLCLK	左クリック（非表示）を禁止	
2	AJCWATSS_DISRCLK	右クリック（ポップアップメニュー）を禁止	
3	AJCWATSS_TOPMOST	最前面に表示	
4	AJCWATSS_TRANSPARENT	透明表示	
5	AJCWATSS_SECHAND	秒針表示	

pSect≠NULL の場合、表示状態をプロファイルに記録し永続化します。
この場合、state は初回の（まだ永続化していない場合の）初期ステータスとなります。

戻り値 ： TRUE　- 成功
FALSE - 失敗

備　考 ：時計表示「AjcWatShow()」を連続した場合、表示が多重化されます。
この場合、時計表示「AjcWatHide()」を実行すると、１つ前の表示状態が復元されます。

50.1.4. 時計非表示(AjcWatHide)

形　式 ： BOOL　AjcWatHide　　(HANDLE hWat);
BOOL　AjcWatHideByID(HANDLE hWat, UI id);
BOOL　AjcWatHideAll (HANDLE hWat);

引　数 ： hWat　- インスタンスハンドル
id　　- 識別ＩＤ

説　明 ： AjcWatHide()は、多重表示中でない場合は、時計表示ウインドを破棄します。
多重表示中の場合は、１つ前の表示状態を復元します。

AjcWatHideByID()は、指定した識別ＩＤを非表示にします。(表示待ちスタックから削除する)
指定した識別ＩＤが、現在表示中である場合は、AjcWatHideByID()と同じです。

AjcWatHideAll()は、すべての識別ＩＤを非表示にします。(結果、時計ウインドは破棄されます)

戻り値 ： TRUE　- 成功
FALSE - 失敗

50.1.5. 時計非表示状態の設定(AjcWatSetShowState)

形　式 ： UI　AjcWatSetShowState (HANDLE hWat, UI id, UI state);

引　数 ： hWat　- インスタンスハンドル
id　　- 識別ＩＤ
state　- 表示ステータス (AjcWatShow()参照)

説　明 ： 指定識別ＩＤの表示状態を設定します。

戻り値 ： ≠-1 - 成功（１つ前の表示状態）
＝-1 - 失敗

50.1.6. 時計表示状態の取得(AjcWatGetShowState)

形　式 ： UI　AjcWatGetShowState (HANDLE hWat, UI id);

引　数 ： hWat　- インスタンスハンドル
id　　- 識別ＩＤ

説　明 ： 指定識別ＩＤの表示状態を取得します。
当該識別ＩＤが現在表示中である場合は、表示中を示すフラグ（AJCWATSS_SHOW）が付加されます。

戻り値 ： ≠-1 - 成功（現在表示中である時計の表示状態、現在表示中である場合は「AJCWATSS_SHOW」を付加）
＝-1 - 失敗

50.1.7. 時計ウインドのハンドル取得(AjcWatSetShowState)

形　式 ： HWND　AjcWatGetHWnd (HANDLE hWat);

引　数 ： hWat　- インスタンスハンドル

説　明 ： 時計表示ウインドのハンドルを取得します。
時計ウインドが表示されていない場合は、NULL を返します。

戻り値 ： ≠-NULL - 成功（時計表示ウインドのハンドル）
＝-NULL - 失敗

50.1.8. 時計を進める(AjcWatAdvance)

形　式 ： VO AjcWatAdvance (SL sec);

引　数 ： sec　- 時計を進める秒数（負数時は戻す秒数）

説　明 ： 時計の表示を指定秒数だけ進める、あるいは戻します。

戻り値 ： なし

51. XMODEM／YMODEM ファイル転送プロトコル

ＸＭＯＤＥＭやＹＭＯＤＥＭは、古くからパソコン通信のファイル転送用プロトコルとして広く用いられています。
また、組み込み装置へのファイル転送用プロトコルとしてもよく使用されます。(組み込み装置の多くのワンチップマイコンには、シリアル(UART)通信のロジックが内蔵されており、多くの組み込み装置ではこれを利用して、パソコンと RS-232C で接続し、パソコンを評価／保守用の端末として使用するケースが多い。また、パソコン用の多くのターミナルソフトにはＸＭＯＤＥＭやＹＭＯＤＥＭプロトコルによるファイル転送機能が組み込まれている)

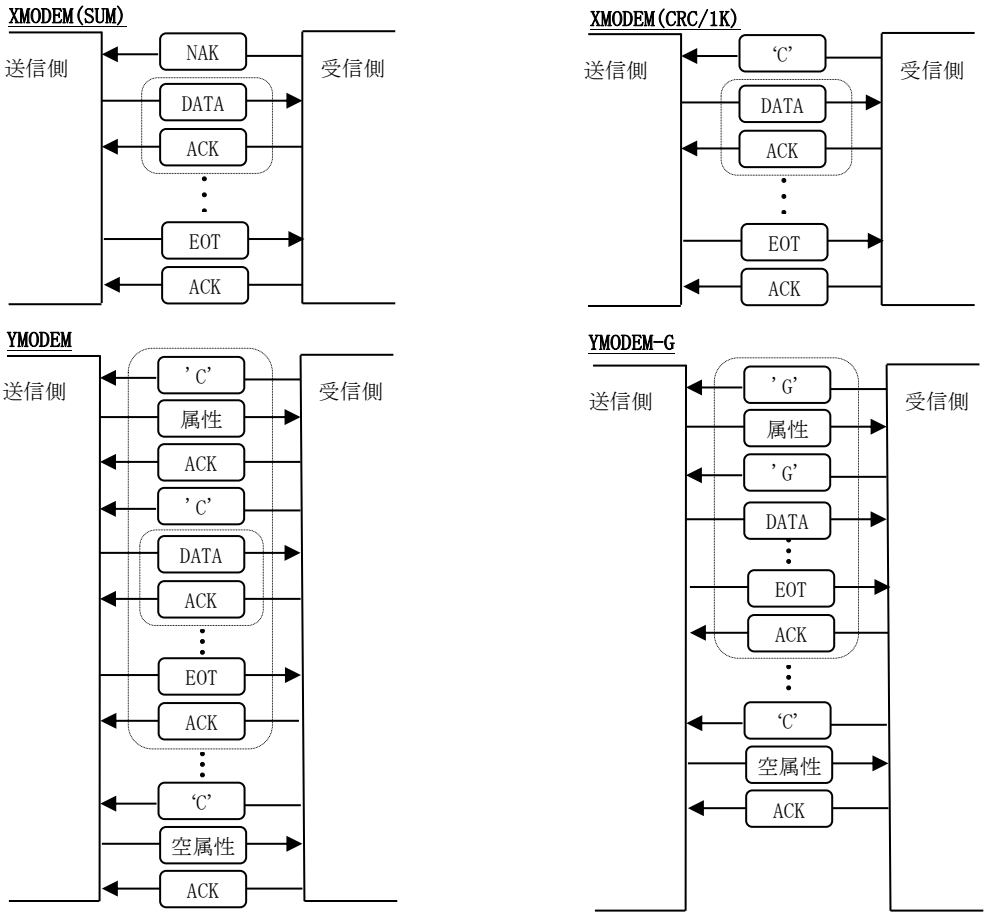
本ライブラリでは、以下のファイル転送プロトコルによる、ファイルの送信、および、受信が可能です。

#	プロトコル	備　考			
		送受信単位	エラー検証方法	ファイル属性情報	再送制御
1	XMODEM(SUM)	１２８バイト	チェックサム	なし	あり
2	XMODEM(CRC)	１２８バイト	ＣＲＣ	なし	あり
3	XMODEM(1K)	１２８／１Ｋバイト	ＣＲＣ	なし	あり
4	YMODEM	１２８／１Ｋバイト	ＣＲＣ	ファイル名，サイズ，日付 等	あり
5	YMODEM-G	１２８／１Ｋバイト	ＣＲＣ	ファイル名，サイズ，日付 等	なし

但し、本ライブラリで提供するのは、上記プロトコル処理部分だけであり、実際にハードウェアを制御し、データの送受信を行なう部分は、ユーザがコールバック関数として作成しなければなりません。

51.1. ファイル転送手順

ＸＭＯＤＥＭ，ＹＭＯＤＥＭ，ＹＭＯＤＥＭ－Ｇの転送手順を以下に示します。



データが不正（チェックサム／ＣＲＣエラー）の場合は、ＡＣＫではなくＮＡＫを返送し、データの再送を要求します。(YMODEM-G 以外)
ファイル転送を中止するには、ＣＡＮ（0x18）を送信します。

51.2. 転送 DATA 形式

SOH ： 0x01　　ACK ： 0x06　　CS ： データ(128 バイト)のチェックサム
STX ： 0x02　　NAK ： 0x15　　CRC ： データ(128/1024 バイト)の CRC
EOT ： 0x04
BN ： レコード番号(0x01～0xFF の繰り返し)
$\overline{BN}$ ： BN の１の補数（ビット反転値）

XMODEM(SUM)

SOH	BN	$\overline{BN}$	データ	CS
1	1	1	128	1

XMODEM(CRC), XMODEM(1K), YMODEM or YMODEM-G

SOH	BN	$\overline{BN}$	データ	CRC
1	1	1	128	2

XMODEM(1K), YMODEM or YMODEM-G

STX	BN	$\overline{BN}$	データ	CRC
1	1	1	1024	2

YMODEM or YMODEM-G の属性データ

ファイルバイト数の 10 進数文字　　1970/1/1 からの通算秒数の８進数文字列

SOH	00	FF	ファイルパス名	00	ファイルサイズ	20	日時(8 進数)	20	属性	20	シリアル番号	00・・・	CRC
1	1	1	←――――――――――――――― 128 ―――――――――――――――→										2

STX	00	FF	ファイルパス名	00	ファイルサイズ	20	日時(8 進数)	20	属性	20	シリアル番号	00・・・	CRC
1	1	1	←――――――――――――――― 1024 ―――――――――――――――→										2

※ファイルパス名の先頭が 0x00 の場合は、「空属性（転送終了)」を意味します。

51.3. プロトコル種別

プロトコル種別は、以下のように定義されています。

```
#define     AJCXYP_XMODEM       0x10
#define     AJCXYP_YMODEM       0x20

typedef enum {
    AJCXYP_XMODEM_SUM   =   (AJCXYP_XMODEM + 1) ,
    AJCXYP_XMODEM_CRC   =   (AJCXYP_XMODEM + 2) ,
    AJCXYP_XMODEM_1K    =   (AJCXYP_XMODEM + 3) ,

    AJCXYP_YMODEM_STD   =   (AJCXYP_YMODEM + 1) ,
    AJCXYP_YMODEM_G     =   (AJCXYP_YMODEM + 2) ,
} AJCXYMPROTOCOL;
```

51.4. ファイル属性情報

AJCXYN_RXFILEINFO イベントで通知されるファイル属性情報は、以下の通りです。

```
  typedef  struct  {
      VOP     pFName;                 /*  ファイル名へのポインタ（但し、ファイル終了の場合はＮＵＬＬ）*/
      ULL     size;                   /*  ファイルサイズ（バイト数）                                  */
      UL      time;                   /*  ファイルタイム（1970-01-01 00:00:00 からの通算秒数（UTC））  */
  } AJCXYMFILEINFO, *PAJCXYMFILEINFO;
  typedef const AJCXYMFILEINFO *PCAJCXYMFILEINFO;
```

51.5. コンソールアプリで使用する場合

ＸＭＯＤＥＭ／ＹＭＯＤＥＭファイル転送では、内部的なウインドを生成してタイマ処理を行っています。

コンソールアプリでＸＭＯＤＥＭ／ＹＭＯＤＥＭファイル転送を行う場合は、ファイル転送中は「AjcDoEvent()」で Windows のメッセージイベントを処理する必要があります。

コンソールアプリの場合は、およそ以下のような処理手順となります。（下記例は、ESC キー押下で転送を中止する処理を含みます）

```
HAJCSCP hScp;               //  ＳＣＰインスタンス（別処理で設定済とする）
BOOL    fBusy = FALSE;      //  ファイル転送中フラグ

//  ファイル送信／受信処理
VO  FileTransferByXModemOrYModem()
{
    WPARAM      wParam;
    LPARAM      lParam;
    UTP         pDat;
    UI          lDat;
    UI          param;

    //  XMODEM/YMODEM インスタンス生成
    HAJCXYM hXym = AjcXymCreate (cbp, cbNotice, cbGetFile, cbGetData, cbSend);

    AjcXymTxStart(hXym, ・・・);        //  ファイル送信開始  } いずれかを実行
    AjcXymRxStart(hXym, ・・・);        //  ファイル受信開始  }

    fBusy = TRUE;
    while (fBusy) {
        if (_kbhit()) {if (_getch() == 0x1B) AjcXymStop();}         //  終了チェック（ESC キー入力チェック）
        //  ＳＣＰイベント待ち
        if (AjcScpWaitEvent(hScp, &wParam, &lParam, 100)) {         //  イベント待ち，イベントあり？
            C_UTP   pPortName = AjcScpGetPortPathName(hScp);
            AjcScpGetEventData(hScp, lParam, &pDat, &lDat, &param); //  イベントデータ取得
                ・・・（ＳＣＰイベント処理）
            AjcScpRelEventData(hScp, lParam);                       //  イベントデータ開放
        }
        AjcDoEvent();                                               //  ウインドメッセージ処理
    }
    AjcXymDelete(hXym);                                             //  XMODEM/YMODEM インスタンス消去
}

//  XMODEM/YMODEM イベント通知　コールバック関数
static VO CALLBACK cbNotice(UI knd, UX Param, UX cbp)
{
    //  イベントハンドル
    switch (knd) {
        case AJCXYN_XXXXX:
                ・・・(XMODEM / YMODEM イベント処理)
    }
    //  ファイル転送終了時の処理
    if ((knd & AJCXYN_END) != 0) {
            ・・・
        fBusy = FALSE;
    }
}
```

51.6. サポートＡＰＩ

ＸＭＯＤＥＭ／ＹＭＯＤＥＭ のサポートＡＰＩ一覧を以下に示します。

#	関 数 名	内容
1	AjcXymCreate	ＸＭＯＤＥＭ／ＹＭＯＤＥＭ通信プロトコル・インスタンス生成
2	AjcXymDelete	ＸＭＯＤＥＭ／ＹＭＯＤＥＭ通信プロトコル・インスタンス消去
3	AjcXym{Set/Get}TxTimeInfo	送信時タイマ情報設定／取得
4	AjcXym{Set/Get}RxTimeInfo	受信時タイマ情報設定／取得
5	AjcXymTxStart	ファイル送信開始
6	AjcXymRxStart	ファイル受信開始
7	AjcXymStop	ファイル転送中止
8	AjcXymPutRxChar	シリアル回線からの受信データ投与
9	AjcXymTxEnd	シリアル回線への送信完了を通知（ＹＭＯＤＥＭ－Ｇ専用）
10	AjcXymGetState	処理状態取得

51.6.1. ＸＭＯＤＥＭ／ＹＭＯＤＥＭ通信プロトコル・インスタンス生成 (AjcXymCreate)

形　式 ： HAJCXYM AjcXymCreate(UX cbp,
VO (CALLBACK *cbNotice)(UI knd, UX Param , UX cbp),
VO (CALLBACK *cbGetFile)(PAJCXYMFILEINFO pBuf , UX cbp),
VO (CALLBACK *cbGetData)(VOP pBuf, UI lBuf, UIP pBytes, UX cbp),
VO (CALLBACK *cbSend)(C_VOP pTxD, UI lTxD , UX cbp))

引　数 ： cbp - コールバックパラメタ
cbNotice - イベント通知用コールバック関数
cbGetFile - ファイル情報取得用コールバック関数　（ファイル受信時は不要(null)）
cbGetData - ファイルデータ取得要コールバック関数（ファイル受信時は不要(null)）
cbSend - シリアル回線へのデータ送信要コールバック関数

説　明 ： ＸＭＯＤＥＭ／ＹＭＯＤＥＭプロトコルのインスタンスを生成します。

戻り値 ： ≠NULL - 成功（インスタンスハンドルを返します）
＝NULL - 失敗

コールバック：

cbNotice（イベント通知）

形　式 ： VO CALLBACK ***cbNotice***(UI knd, UX Param, UX cbp)；

引　数 ： knd - イベント種別
Param - パラメタ
cbp - コールバックパラメタ

説　明 ： このコールバック関数は、以下のイベントが発生した場合に呼び出されます。

#	knd(イベント種別)	値	内容	Param（パラメタ）
1	AJCXYN_RXFILEINFO	0x0010	ファイル属性情報受信通知	ファイル属性情報(AJCXYMFILEINFO)のアドレス
2	AJCXYN_TXREC	0x0020	送信完了通知（YMODEM-G では発生しない）	送信したデータのバイト数（128 / 1024）
3	AJCXYN_RXREC_128	0x0021	１２８バイトレコード受信通知	受信データ(１２８バイト)のアドレス
4	AJCXYN_RXREC_1K	0x0022	１Ｋバイトレコード受信通知	受信データ（１０２４バイト）のアドレス
5	AJCXYN_RETRY	0x0030	再送制御が発生したことを通知	－
6	AJCXYN_EOF	0x0040	ＥＯＦ(End of file) 通知	－
7	AJCXYN_PROTOCOL	0x0050	通信プロトコル変更通知	変更されたプロトコルの種別（AJCXYMPROTOCOL）
8	AJCXYN_COMPLETE	0x8000	ファイル転送完了通知	－
9	AJCXYN_RX_CAN	0x8001	ファイル転送中止（ＣＡＮ受信）	－
10	AJCXYN_USERSTOP	0x8002	〃　(ユーザからの中止要求)	－
11	AJCXYN_TIMEOUT	0x8003	〃　(タイムアウト / リトライアウト)	－
12	AJCXYN_ABORT	0x8004	〃　(YMODEM-G で不正レコード受信)	－

＃１：のファイル属性情報で、ＸＭＯＤＥＭの場合は、ファイル名="XMODEM.TMP", サイズ＝0xFFFFFFFF, タイム＝０で通知されます。
＃８～＃１２は、ファイル転送が終了したことを示します。
ファイル転送終了の条件は、イベント発生時に「 (knd & AJCXYN_END) != 0 」で検出できます。(AJCXYN_END の値は 0x8000)

cbGetFile（ファイル情報取得）

形　式 ： VO CALLBACK ***cbGetFile***(PAJCXYMFILEINFO pBuf, UX cbp)；

引　数 ： pBuf - ファイル情報を格納するバッファのアドレス
cbp - コールバックパラメタ

説　明 ： このコールバック関数は、AjcXymTxStart()実行後、ファイルの送信を開始するタイミングで呼び出されます。
pBuf で示されるバッファに、次に送信するファイルの情報を設定します。ファイル情報の形式は、以下のとおりです。

```
typedef  struct {
    UBP  pFName;    // ファイル名へのポインタ
    ULL  size;      // ファイルのバイト数
    UL   time;      // ファイル日付 (1970-01-01 00:00:00 からの通算秒数)
} AJCXYMFILEINFO, *PAJCXYMFILEINFO;
typedef const AJCXYMFILEINFO *PCAJCXYMFILEINFO;
```

XMODEM の場合は、XymTxStart()実行後 1 回だけ呼び出されます。
YMODEM の場合は、XymTxStart()実行後、複数回呼び出されます。この時、次に送信するファイルが無い場合は、ファイル情報のメンバ「pFName」に NULL を設定します

戻り値 ： なし

cbGetData（ファイルデータ取得）

形　式 ： VO CALLBACK ***cbGetData***(VOP pBuf, UI len, UIP pBytes, UX cbp)；

引　数 ： pBuf - ファイルデータを格納するバッファのアドレス
len - ファイルデータを格納するバッファのバイト数
pBytes - 実際にファイルから読み出したデータのバイト数を格納するバッファのアドレス
cbp - コールバックパラメタ

説　明 ： pBuf,len で示されるバッファに、順次ファイルのデータを読み出し、格納します。
また、pBytes で示されるバッファに、実際に読み出し格納したデータのバイト数を設定します。
ファイルが終端（ＥＯＦ）に達した場合は、pBytes で示されるバッファにゼロ（０）を設定します。

戻り値 ： なし

cbSend（データ送出）

形　式 ： VO CALLBACK ***cbSend***(C_VOP pTxD, UI len, UX cbp)；

引　数 ： pTxD - 送信するデータブロックのアドレス
len - 送信するデータブロックのバイト数
cbp - コールバックパラメタ

説　明 ： pTxD, len で示されるデータブロックを通信回線に送出します。

戻り値 ： なし

51.6.2. ＸＭＯＤＥＭ／ＹＭＯＤＥＭ通信プロトコル・インスタンス消去 (AjcXymDelete)

形　式 ： VO AjcXymDelete(HAJCXYM hXym);

引　数 ： hXym　　- インスタンスハンドル

説　明 ： ＸＭＯＤＥＭ／ＹＭＯＤＥＭプロトコルのインスタンスを消去します。

戻り値 ： なし

51.6.3. 送信時タイマ情報設定／取得 (AjcXym{Set/Get}TxTimeInfo)

形　式 ： VO AjcXymSetTxTimeInfo (HAJCXYM hXym, UI t0, UI r0, UI t1, UI r1); --- 設定
VO AjcXymGetTxTimeInfo (HAJCXYM hXym, UIP t0, UIP r0, UIP t1, UIP r1); --- 取得

引　数 ： hXym　　- インスタンスハンドル
t0, r0　- タイマ０とリトライ回数 (AjcXymGetTxTimeInfo の場合は格納するバッファのアドレス)
t1, r1　- タイマ１とリトライ回数 (AjcXymGetTxTimeInfo の場合は格納するバッファのアドレス)

説　明 ： t0, r0, t1, r1 で以下の情報を設定／取得します。

引数	内容	デフォルト値
t0	受信要求(NAK/C/G)受信待ちタイマ[ms]	60,000
r0	未使用（リトライ無し）	－
t1	ＡＣＫ／ＮＡＫ受信待ちタイマ[ms]	3,000
r1	データ／ＥＯＴ再送限度回数	3

戻り値 ： なし

51.6.4. 受信時タイマ情報設定／取得 (AjcXymGetRxTimeInfo)

形　式 ： VO AjcXymSetRxTimeInfo (HAJCXYM hXym, UI t0, UI r0, UI t1, UI r1); --- 設定
VO AjcXymGetRxTimeInfo (HAJCXYM hXym, UIP t0, UIP r0, UIP t1, UIP r1); --- 取得

引　数 ： hXym　　- インスタンスハンドル
t0, r0　- タイマ０とリトライ回数 (AjcXymGetRxTimeInfo の場合は格納するバッファのアドレス)
t1, r1　- タイマ１とリトライ回数 (AjcXymGetRxTimeInfo の場合は格納するバッファのアドレス)

説　明 ： t0, r0, t1, r1 で以下の情報を設定／取得します。

引数	内容(デフォルト値)	デフォルト値
t0	受信要求(NAK/C/G)送出インタバル[ms]	1,000
r0	未使用（受信要求は永続的に送出）	－
t1	データ／ＥＯＴ受信待ちタイマ[ms]	10,000
r1	再送要求(NAK)送出限度回数	3

戻り値 ： なし

51.6.5. ファイル送信開始 (AjcXymTxStart)

形　式 ： BOOL AjcXymTxStart (HAJCXYM hXym, AJCXYMPROTOCOL Protocol);

引　数 ： hXym　　- インスタンスハンドル
Protocol - ファイル転送プロトコル種別 (AJCXYP_XMODEM_SUM / XMODEM_CRC / XMODEM_1K / YMODEM_STD / YMODEM_G)

説　明 ： ＸＭＯＤＥＭ／ＹＭＯＤＥＭプロトコルによるファイルの送信を開始します。
Protocol には、以下のいずれかのシンボルを指定します。

シンボル	内容（ファイル転送プロトコル）
AJCXYP_XMODEM_SUM	ＸＭＯＤＥＭ（１２８バイト　，チェックサム）
AJCXYP_XMODEM_CRC	〃　　（１２８バイト　，ＣＲＣ）
AJCXYP_XMODEM_1K	〃　　（１０２４バイト，ＣＲＣ）
AJCXYP_YMODEM_STD	ＹＭＯＤＥＭ
AJCXYP_YMODEM_G	ＹＭＯＤＥＭ－Ｇ

ファイル送信中は、コールバック関数（cbNotice, cbGetFile, cbGetData）が適時呼び出されます。
ファイル転送の終了は、cbNotice（knd 引数のビット 15（AJCXYN_END）＝１）により通知されます。

戻り値 ： TRUE - 成功
FALSE - 失敗

51.6.6. ファイル受信開始 (AjcXymRxStart)

形　式 ： BOOL AjcXymTxStart (HAJCXYM hXym, AJCXYMPROTOCOL Protocol);

引　数 ： hXym　　- インスタンスハンドル
Protocol - ファイル転送プロトコル種別 (AJCXYP_XMODEM_SUM / XMODEM_CRC / XMODEM_1K / YMODEM_STD / YMODEM_G)

説　明 ： ＸＭＯＤＥＭ／ＹＭＯＤＥＭプロトコルによるファイルの受信を開始します。
Protocol には、以下のいずれかのシンボルを指定します。

シンボル	内容（ファイル転送プロトコル）
AJCXYP_XMODEM_SUM	ＸＭＯＤＥＭ（１２８バイト　，チェックサム）
AJCXYP_XMODEM_CRC	〃　　（１２８バイト　，ＣＲＣ）
AJCXYP_XMODEM_1K	〃　　（１０２４バイト，ＣＲＣ）
AJCXYP_YMODEM_STD	ＹＭＯＤＥＭ
AJCXYP_YMODEM_G	ＹＭＯＤＥＭ－Ｇ

ファイル受信中は、コールバック関数（cbNotice）が適時呼び出されます。
ファイル受信の開始は、cbNotice（knd＝AJCXYN_RXFILEINFO）により通知されます。
受信したファイルデータは、cbNotice（knd＝AJCXYN_RXREC_128 / AJCXYN_RXREC_1K）により通知されます。
ファイルの終端は、cbNotice（knd＝XYN_EOF）により通知されます。
ファイル転送の終了は、cbNotice（knd 引数のビット 15（AJCXYN_END）＝１）により通知されます。

戻り値 ： TRUE - 成功
FALSE - 失敗

51.6.7. ファイル転送中止(AjcXymStop)

形　式 ： VO　AjcXymStop (HAJCXYM　pW);

引　数 ： hXym　　- インスタンスハンドル

説　明 ： ファイル転送（送信／受信）を中止します。

戻り値 ： なし

51.6.8. シリアル回線からの受信データ投与 (AjcXymPutRxChar)

形　式 ： VO　AjcXymPutRxChar(HAJCXYM　hXym, UI rxd);

引　数 ： hXym　　- インスタンスハンドル
rxd　　- 受信データバイト

説　明 ： 回線から受信したデータ（１バイト）を投与します。

戻り値 ： なし

51.6.9. シリアル回線への送信完了を通知 (AjcXymTxEnd)

形　式 ： VO　AjcXymTxEnd(HAJCXYM　hXym);

引　数 ： hXym　　- インスタンスハンドル

説　明 ： ＹＭＯＤＥＭ－Ｇプロトコルでファイルの送信を行なっている場合、１つのファイルデータを送信完了した時点で、本関数をコールしなければなりません。

ＹＭＯＤＥＭ－Ｇでファイルを送信する場合、データレコードの送出に対するＡＣＫは無いため、cbGetData と、cbSend をくりかえしコールバックすることにより、一気に１ファイル分のデータを送出します。
そして、全ファイルデータの送信完了を待ってからＥＯＴを送出します。
ＹＭＯＤＥＭ－Ｇの通信では、送信の完了も、タイムアウトも検出しない（できない）ため、全てのファイルデータの送信が完了したことを、本関数をコールすることにより通知する必要があります。

ＹＭＯＤＥＭ－Ｇ以外の場合は、本関数をコールする必要はありません。

戻り値 ： なし

51.6.10. 処理状態取得 (AjcXymGetState)

形　式 ： UI　AjcXymGetState (HAJCXYM　hXym);

引　数 ： hXym　　- インスタンスハンドル

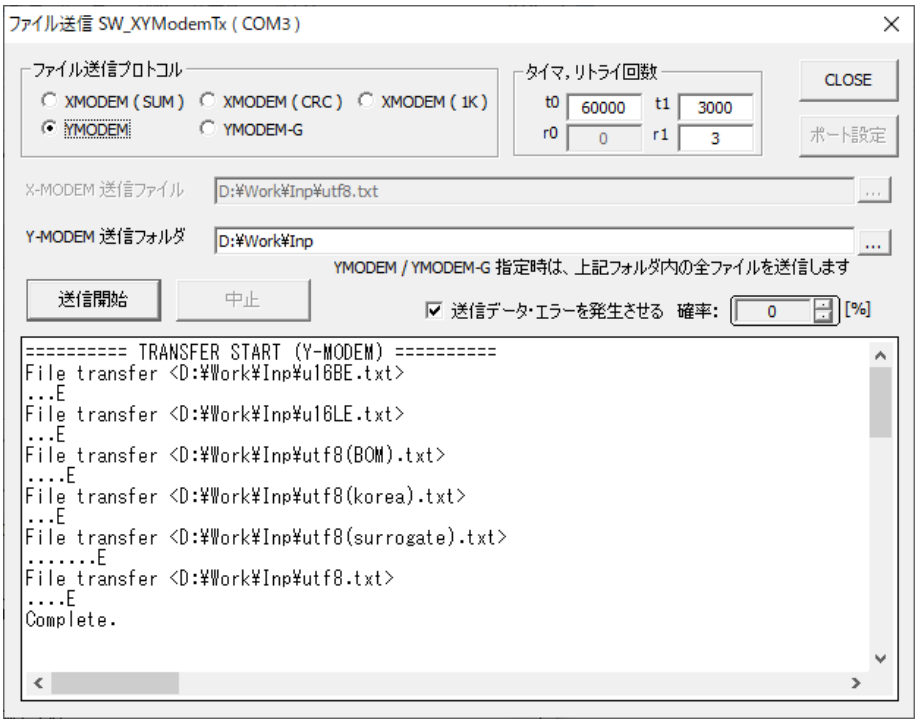
説　明 ： 現在の処理状態を取得します。

戻り値 ： ≠０：通信中
＝０：アイドル状態

51.7. サンプルプログラム

51.7.1. SW_XYModemTx（XMODEM/YMODEM ファイル送信）

以下のサンプルプログラムでは、ＸＭＯＤＥＭ／ＹＭＯＤＥＭによるファイル送信を行います。




```
  1 : //
  2 : //  SW_XYModemTx.c
  3 : //
  4 : #include    <AjrCstXX.h>
  5 : #include    <io.h>
  6 : #include    <tchar.h>
  7 : #include    <sys/stat.h>
  8 : #include    "resource.h"
  9 :
 10 : #define WM_SCPEVENT     (WM_USER + 100)
 11 :
 12 : //-----------------------------------------------------------------------------------------------------//
 13 : //  定数                                                                                               //
 14 : //-----------------------------------------------------------------------------------------------------//
 15 : typedef enum {COMM_232C , COMM_LAN}                ECOMM;
 16 : typedef enum {XM_SUM, XM_CRC, XM_1K, YM_STD, YM_G}  EPROTOCOL;
 17 :
 18 : //-----------------------------------------------------------------------------------------------------//
 19 : //  ワーク                                                                                             //
 20 : //-----------------------------------------------------------------------------------------------------//
 21 : static  HINSTANCE      hInst;                      //  ＤＬＬインスタンスハンドル
 22 : static  HWND           hDlgMain;                   //  ダイアログボックスハンドル
 23 : static  HWND           hVthLog;
 24 :
 25 : static  HAJCXYM        hXym;
 26 : static  HAJCSCP        hScp;
 27 :
 28 : static  BOOL           fFindFirst      = TRUE;      //  初回ファイル検索フラグ
 29 : static  SX             hFind           = -1;        //  ファイル検索ハンドル
 30 : static  struct _tfinddata64_t   FindData;           //  ファイル検索結果データ
 31 : static  HANDLE         hFile = INVALID_HANDLE_VALUE;  //  送信ファイルハンドル
 32 : static  SLL            TxRemBytes;                 //  送信ファイルサイズ
 33 : static  UT             DirPath [MAX_PATH];         //  ディレクトリ・パス名
 34 : static  UT             FilePath[MAX_PATH];         //  ファイルパス名
 35 : static  UT             WildCard[MAX_PATH];         //  ワイルドカード・パス
 36 :
 37 : //-----------------------------------------------------------------------------------------------------//
 38 : //  内部サブ関数                                                                                       //
 39 : //-----------------------------------------------------------------------------------------------------//
 40 : AJC_DLGPROC_DEF(Main);
 41 :
 42 : static VO CALLBACK cbGetFile(PAJCXYMFILEINFO pBuf          , UX cbp);
 43 : static VO CALLBACK cbGetData(VOP pBuf, UI len, UIP pBytes   , UX cbp);
 44 : static VO CALLBACK cbNotice (UI knd, UX Param               , UX cbp);
 45 : static VO CALLBACK cbSend   (C_VOP pTxD, UI len            , UX cbp);
```

```
46 :
47 : static  VO              ShowOnOpened(VO);
48 : static  VO              ShowOnClosed(VO);
49 : static  AJCXYMPROTOCOL  SubGetProtocol      (EPROTOCOL      protocol);
50 : static  UTP             SubProtocolNameByRbt(EPROTOCOL      protocol);
51 : static  UTP             SubProtocolNameByXym(AJCXYMPROTOCOL protocol);
52 :
53 : //=============================================================================================================//
54 : //                                                                                                             //
55 : //  W i n M a i n                                                                                               //
56 : //                                                                                                             //
57 : //=============================================================================================================//
58 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
59 : {
60 :     MSG     msg;
61 :
62 :     hInst = hInstance;
63 :
64 :     //----- メイン・ダイアログオープン ----------------//
65 :     hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_MAIN), NULL, AJC_DLGPROC_NAME(Main));
66 :     ShowWindow(hDlgMain, SW_SHOW);
67 :
68 :     //----- メッセージループ --------------------------//
69 :     while (GetMessage(&msg, NULL, 0, 0)) {
70 :         do {
71 :             if (IsDialogMessage(hDlgMain, &msg)) break;
72 :             TranslateMessage(&msg);
73 :             DispatchMessage (&msg);
74 :         } while (0);
75 :     }
76 :
77 :     return (int)msg.wParam ;
78 : }
79 : //=============================================================================================================//
80 : //                                                                                                             //
81 : //  ダイアログ・プロシージャ                                                                                   //
82 : //                                                                                                             //
83 : //=============================================================================================================//
84 : //----- ダイアログ初期化 ----------------------------------------------------------------------------------------//
85 : AJC_DLGPROC(Main, WM_INITDIALOG    )
86 : {
87 :     UI  t0, t1, r0, r1;
88 :
89 :     hDlgMain = hDlg;
90 :     hVthLog  = GetDlgItem(hDlg, IDC_VTH_LOG);
91 :
92 :     //--- ツールチップ ------------------------------------//
93 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_TXT_T0), TEXT("受信要求(NAK/C/G)受信待ちタイマ[ms]"));
94 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_TXT_T1), TEXT("ＡＣＫ／ＮＡＫ受信待ちタイマ[ms]"));
95 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_TXT_R1), TEXT("データ／ＥＯＴ再送限度回数"));
96 :     //--- ＳＣＰインスタンス生成 --------------------------//
97 :     hScp = AjcScpCreate();
98 :     AjcScpSetMode(hScp, hDlg, WM_SCPEVENT, AJCSCP_CM_BIN);
99 :     AjcScpSetEvtMask(hScp, AJCSCP_EV_PORTSTATE | AJCSCP_EV_RXCHUNK | AJCSCP_EV_TXEMPTY);
100 :    //--- テキストボックスへのドロップを許可 --------------//
101 :    AjcEnableDlgItemToDrop (hDlg, IDC_TXT_XM_FILE, AJCDROP_FILE);
102 :    AjcEnableDlgItemToDrop (hDlg, IDC_TXT_YM_DIR , AJCDROP_DIR );
103 :    //--- ラジオボタングループ化 --------------------------//
104 :    AjcSbcRadioBtns(GetDlgItem(hDlg, IDC_GRP_PROTOCOL));
105 :    //--- X-Modem / Y-Modem インスタンス生成 ---------------//
106 :    hXym = AjcXymCreate(0, cbNotice, cbGetFile, cbGetData, cbSend);
107 :    //--- ダイアログ項目の初期化 ---------------------------//
108 :    AjcXymGetTxTimeInfo(hXym, &t0, &r0, &t1, &r1);
109 :    AjcSetDlgItemUInt(hDlg, IDC_GRP_PROTOCOL, XM_SUM  );
110 :    AjcSetDlgItemStr (hDlg, IDC_TXT_XM_FILE , TEXT(""));
111 :    AjcSetDlgItemStr (hDlg, IDC_TXT_YM_DIR  , TEXT(""));
112 :    AjcSetDlgItemChk (hDlg, IDC_CHK_TXERR   , FALSE  );
113 :    AjcSetDlgItemUInt(hDlg, IDC_INP_TXERR   , 0      );
114 :    AjcSetDlgItemUInt(hDlg, IDC_TXT_T0      , t0     );
115 :    AjcSetDlgItemUInt(hDlg, IDC_TXT_R0      , r0     );
116 :    AjcSetDlgItemUInt(hDlg, IDC_TXT_T1      , t1     );
117 :    AjcSetDlgItemUInt(hDlg, IDC_TXT_R1      , r1     );
118 :    //----- ダイアログ項目のロード ---------------------------//
119 :    AjcLoadAllControlSettings (hDlg, TEXT("DlgSetting"), AJCOPT2(AJCCTL_SELACT_, ALL, NTCRBT));
120 :    //--- ポートオープン -----------------------------------//
121 :    SendMessage(hDlg, WM_COMMAND, MAKELONG(IDC_CMD_OPEN, BN_CLICKED),0);
122 :
123 :    return TRUE;
124 : }
125 : //----- ウインド破棄 --------------------------------------------------------------------------------------------//
126 : AJC_DLGPROC(Main, WM_DESTROY       )
127 : {
128 :    //----- インスタンス破棄 ---------------------------//
129 :    AjcScpDelete(hScp); //  SCP インスタンス消去
130 :    AjcXymDelete(hXym); //  X-Modem / Y-Modem インスタンス
131 :    //----- ダイアログ項目のセーブ ---------------------//
132 :    AjcSaveAllControlSettings(hDlg);
133 :
134 :    PostQuitMessage(0);
135 :    return TRUE;
```

```
136 : }
137 : //----- ＳＣＰイベント通知 ----------------------------------------------------------------------------//
138 : AJC_DLGPROC(Main, WM_SCPEVENT      )
139 : {
140 :     UI      time = GetTickCount();
141 :     UI      len, param;
142 :     union {UBP pBin; UTP pTxt; VOP vp;} u;
143 :     UT      txt[128];
144 :
145 :     AjcScpGetEventData(hScp, lParam, &u.vp, &len, &param);              //  イベントデータ取得
146 :     if(wParam & AJCSCP_EV_PORTSTATE) {                                 //  ●ポート状態通知
147 :         switch (param) {
148 :             case AJCSCP_CLOSED: ShowOnClosed(); break;                 //      ・クローズ状態
149 :             case AJCSCP_OPENED: ShowOnOpened(); break;                 //      ・オープン状態
150 :             case AJCSCP_OPENFAIL:                                      //      ・オープン失敗
151 :                 AjcSnPrintF(txt, AJCTSIZE(txt), TEXT("%s open falure!"), AjcScpGetPortPathName(hScp));
152 :                 MessageBox(hDlgMain, txt, TEXT("Error"), MB_ICONERROR);
153 :                 break;
154 :         }
155 :     }
156 :     if(wParam & AJCSCP_EV_RXCHUNK ) {                                  //  ●チャンクデータ受信通知（バイナリ）
157 :         UI      i;
158 :         UBP     p = u.pBin;
159 :         for (i = 0; i < len; i++) {
160 :             AjcXymPutRxChar(hXym, *p++);
161 :         }
162 :     }
163 :     if(wParam & AJCSCP_EV_TXEMPTY) {                                   //  ●送信完了
164 :         if (hXym->knd == AJCXYP_YMODEM_G && TxRemBytes <= 0) {         //      YMODEM-G and 全データ送出済み？
165 :             AjcXymTxEnd(hXym);                                         //          YMODEM-G ファイル送信完了を通知
166 :         }
167 :     }
168 :     AjcScpRelEventData(hScp, lParam);                                  //  イベントデータ開放
169 :
170 :     return TRUE;
171 : }
172 : //----- ウインドクローズ ------------------------------------------------------------------------------//
173 : AJC_DLGPROC(Main, IDCANCEL          )
174 : {
175 :     DestroyWindow(hDlg);
176 :     return TRUE;
177 : }
178 : //----- オープン／クローズ ボタン ---------------------------------------------------------------------//
179 : AJC_DLGPROC(Main, IDC_CMD_OPEN      )
180 : {
181 :     if (AjcScpIsOpened(hScp)) {
182 :         //  通信中ならば、中止ボタン押下
183 :         if (AjcXymGetState(hXym) != 0) {
184 :             SendMessage(hDlg, WM_COMMAND, MAKELONG(IDC_CMD_STOP, BN_CLICKED), (LPARAM)GetDlgItem(hDlg, IDC_CMD_OPEN));
185 :         }
186 :         //  ポートクローズ
187 :         AjcScpClose(hScp);
188 :     }
189 :     else {
190 :         //  ポートオープン
191 :         AjcScpOpenDefault(hScp);
192 :     }
193 :     return TRUE;
194 : }
195 : //----- ポート設定ボタン ------------------------------------------------------------------------------//
196 : AJC_DLGPROC(Main, IDC_CMD_EASY      )
197 : {
198 :     AjcScpDlgParamEasy(hScp, hDlg);
199 :     return TRUE;
200 : }
201 : //----- プロトコルの選択 ------------------------------------------------------------------------------//
202 : AJC_DLGPROC(Main, IDC_GRP_PROTOCOL  )
203 : {
204 :     if (HIWORD(wParam) == AJCRBTN_SELECT) {
205 :         if (lParam <= XM_1K) {
206 :             AjcEnableGroup(GetDlgItem(hDlg, IDC_GRP_XM_FILE), TRUE , TRUE );
207 :             AjcEnableGroup(GetDlgItem(hDlg, IDC_GRP_YM_DIR ), FALSE, FALSE);
208 :         }
209 :         else {
210 :             AjcEnableGroup(GetDlgItem(hDlg, IDC_GRP_XM_FILE), FALSE, FALSE);
211 :             AjcEnableGroup(GetDlgItem(hDlg, IDC_GRP_YM_DIR ), TRUE , TRUE );
212 :         }
213 :     }
214 :     return TRUE;
215 : }
216 : //----- ＸＭＯＤＥＭ送信ファイル名設定 ----------------------------------------------------------------//
217 : AJC_DLGPROC(Main, IDC_CMD_XM_FILE   )
218 : {
219 :     struct __stat64 st;
220 :     UT              drv[_MAX_DRIVE], dir[_MAX_DIR], fname[_MAX_FNAME], fext[_MAX_EXT];
221 :
222 :     if (_tstat64(FilePath, &st) != 0) {
223 :         FilePath[0] = 0;
224 :     }
225 :     if (AjcGetOpenFile(hDlg, TEXT("送信ファイル"), TEXT("AllFiles(*.*)/*.*"), TEXT("txt"), FilePath, MAX_PATH)) {
```

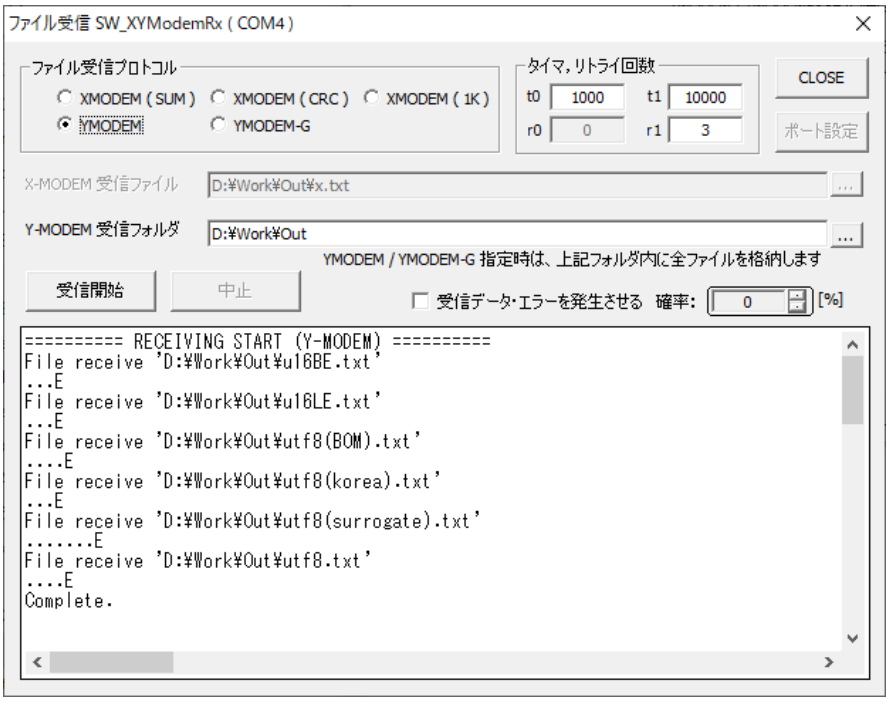
```
226 :        AjcSetDlgItemStr(hDlg, IDC_TXT_XM_FILE, FilePath);
227 :        MAjcSplitPath(FilePath, drv, dir, fname, fext);
228 :        MAjcMakePath (DirPath , drv, dir, NULL , NULL);
229 :    }
230 :
231 :    return TRUE;
232 : }
233 : //----- YMODEM送信フォルダ名設定 ----------------------------------------------------------------------//
234 : AJC_DLGPROC(Main, IDC_CMD_YM_DIR    )
235 : {
236 :    if (AjcGetFolderName(hDlg, TEXT("送信フォルダ"), TEXT("フォルダ内の全ファイルを送信します"), DirPath, MAX_PATH)) {
237 :        AjcSetDlgItemStr(hDlg, IDC_TXT_YM_DIR, DirPath);
238 :    }
239 :    return TRUE;
240 : }
241 : //----- 送信開始ボタン --------------------------------------------------------------------------------//
242 : AJC_DLGPROC(Main, IDC_CMD_SEND      )
243 : {
244 :    EPROTOCOL   protocol = (EPROTOCOL)AjcGetDlgItemUInt(hDlg, IDC_GRP_PROTOCOL);
245 :    UI          t0 = AjcGetDlgItemUInt(hDlg, IDC_TXT_T0);
246 :    UI          r0 = AjcGetDlgItemUInt(hDlg, IDC_TXT_R0);
247 :    UI          t1 = AjcGetDlgItemUInt(hDlg, IDC_TXT_T1);
248 :    UI          r1 = AjcGetDlgItemUInt(hDlg, IDC_TXT_R1);
249 :
250 :    AjcXymSetTxTimeInfo (hXym, t0, r0, t1, r1);                          //  タイマ情報設定
251 :    do {
252 :        if (protocol <= XM_1K) {                                        //  X−MODEM?
253 :            UT      drv[_MAX_DRIVE], dir[_MAX_DIR];
254 :            AjcGetDlgItemStr(hDlg, IDC_TXT_XM_FILE, FilePath, MAX_PATH);    //      転送ファイル名設定
255 :            MAjcStrCpy(WildCard, AJCTSIZE(WildCard), FilePath);
256 :            MAjcSplitPath(FilePath, drv, dir, NULL, NULL);               //      ワイルドカード設定
257 :            MAjcMakePath (DirPath, drv, dir, NULL, NULL);                //      ・
258 :        }
259 :        else {                                                          //  Y−MODEM?
260 :            AjcGetDlgItemStr(hDlg, IDC_TXT_YM_DIR, DirPath, MAX_PATH);      //      転送フォルダ名設定
261 :            MAjcStrCpy(WildCard, AJCTSIZE(WildCard), DirPath);           //      ワイルドカード設定
262 :            AjcPathCat(WildCard, TEXT("*.*"), MAX_PATH);                 //      ・
263 :        }
264 :
265 :        AjcEnableGroup(GetDlgItem(hDlgMain, IDC_GRP_PROTOCOL), FALSE, FALSE);
266 :        EnableWindow(GetDlgItem(hDlgMain, IDC_CMD_SEND), FALSE);
267 :        EnableWindow(GetDlgItem(hDlgMain, IDC_CMD_STOP), TRUE );
268 :
269 :        AjcVthPrintF(hVthLog, TEXT("========= TRANSFER START (%s) =========¥n"), SubProtocolNameByRbt(protocol));
270 :        fFindFirst      = TRUE;                         //  ファイル検索初回フラグ・セット
271 :        AjcXymTxStart(hXym, SubGetProtocol(protocol)); //  送信開始
272 :    } while(0);
273 :
274 :    return TRUE;
275 : }
276 : //----- 送信中止ボタン --------------------------------------------------------------------------------//
277 : AJC_DLGPROC(Main, IDC_CMD_STOP      )
278 : {
279 :    AjcXymStop(hXym);
280 :    AjcEnableGroup(GetDlgItem(hDlgMain, IDC_GRP_PROTOCOL), TRUE , TRUE );
281 :    EnableWindow(GetDlgItem(hDlgMain, IDC_CMD_SEND  ), TRUE );
282 :    EnableWindow(GetDlgItem(hDlgMain, IDC_CMD_STOP  ), FALSE);
283 :    return TRUE;
284 : }
285 : //-----------------------------------------------------------------------------------------------------//
286 : AJC_DLGMAP_DEF(Main)
287 :    AJC_DLGMAP_MSG(Main, WM_INITDIALOG      )
288 :    AJC_DLGMAP_MSG(Main, WM_DESTROY         )
289 :    AJC_DLGMAP_MSG(Main, WM_SCPEVENT        )
290 :    AJC_DLGMAP_CMD(Main, IDCANCEL           )
291 :    AJC_DLGMAP_CMD(Main, IDC_GRP_PROTOCOL   )
292 :    AJC_DLGMAP_CMD(Main, IDC_CMD_XM_FILE    )
293 :    AJC_DLGMAP_CMD(Main, IDC_CMD_YM_DIR     )
294 :    AJC_DLGMAP_CMD(Main, IDC_CMD_OPEN       )
295 :    AJC_DLGMAP_CMD(Main, IDC_CMD_EASY       )
296 :    AJC_DLGMAP_CMD(Main, IDC_CMD_SEND       )
297 :    AJC_DLGMAP_CMD(Main, IDC_CMD_STOP       )
298 : AJC_DLGMAP_END
299 :
300 : //-----------------------------------------------------------------------------------------------------//
301 : //  ファイル情報取得  コールバック関数                                                                 //
302 : //-----------------------------------------------------------------------------------------------------//
303 : static VO CALLBACK cbGetFile(PAJCXYMFILEINFO pBuf, UX cbp)
304 : {
305 :    BOOL    fEndOfFind  = FALSE;
306 :    BOOL    fFileOpened = FALSE;
307 :
308 :    while (!fEndOfFind  &&  !fFileOpened) {
309 :        //---- ファイル検索 ------------------------------------------------------//
310 :        do {
311 :            if (fFindFirst) {if ((hFind = _tfindfirst64(WildCard, &FindData)) == -1) fEndOfFind = TRUE;}
312 :            else            {if (_tfindnext64(hFind, &FindData)               == -1) fEndOfFind = TRUE;}
313 :            fFindFirst = FALSE;
314 :        } while (!fEndOfFind && (FindData.attrib & (_A_HIDDEN | _A_SUBDIR | _A_SYSTEM)));
315 :
```

```
316 :        //----- ファイル情報設定 ------------------------------------------------//
317 :        if (!fEndOfFind) {
318 :            AjcStrCpy(FilePath,MAX_PATH, DirPath);
319 :            AjcPathCat(FilePath, FindData.name, MAX_PATH);
320 :            if (hFile != INVALID_HANDLE_VALUE) {
321 :                CloseHandle(hFile);
322 :            }
323 :            hFile = CreateFile(FilePath, GENERIC_READ, FILE_SHARE_READ, NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
324 :            if (hFile != INVALID_HANDLE_VALUE) {
325 :                GetFileSizeEx(hFile, (LARGE_INTEGER*)&TxRemBytes);  //  ファイルバイトサイズ設定
326 :                MAjcStrCpy(FilePath, MAX_PATH, DirPath);            //  ファイルパス名設定
327 :                AjcPathCat(FilePath, FindData.name, MAX_PATH);      //  ・
328 :                pBuf->pFName = FindData.name;                       //  ファイル名
329 :                GetFileSizeEx(hFile, (PLARGE_INTEGER)&pBuf->size);  //  ファイルサイズ
330 :                pBuf->time   = (UL)(FindData.time_write);           //  ファイル時刻 (ＵＴＣ)
331 :                AjcVthPrintF(hVthLog, TEXT("File transfer <%s>¥n"), FilePath);
332 :                fFileOpened = TRUE;
333 :            }
334 :            else {
335 :                AjcVthPrintF(hVthLog, TEXT("File open failure <%s>¥n"));
336 :                fFileOpened = FALSE;
337 :            }
338 :        }
339 :        else  pBuf->pFName = NULL;
340 :    }
341 : }
342 : //-------------------------------------------------------------------------------------------------------//
343 : //  ファイルデータ取得  コールバック関数                                                                 //
344 : //-------------------------------------------------------------------------------------------------------//
345 : static VO CALLBACK cbGetData(VOP pBuf, UI len, UIP pBytes, UX cbp)
346 : {
347 :     ReadFile(hFile, pBuf, len, (ULP)pBytes, NULL);          //  ファイルデータ読み出し
348 :     TxRemBytes -= (*pBytes);                                //  バイト数カウンタ更新
349 : }
350 : //-------------------------------------------------------------------------------------------------------//
351 : //  イベント通知  コールバック関数                                                                       //
352 : //-------------------------------------------------------------------------------------------------------//
353 : static VO CALLBACK cbNotice(UI knd, UX Param, UX cbp)
354 : {
355 :     switch (knd) {
356 :         case AJCXYN_TXREC:      AjcVthPrintF(hVthLog, TEXT("."));                       break;  //  ●１レコード送信完了
357 :         case AJCXYN_RETRY:      AjcVthPrintF(hVthLog, TEXT("R"));                       break;  //  ●再送発生
358 :         case AJCXYN_EOF:        AjcVthPrintF(hVthLog, TEXT("E¥n"));                     break;  //  ●ファイル終端 (ＥＯＦ)
359 :         case AJCXYN_PROTOCOL:   AjcVthPrintF(hVthLog, TEXT("Protocol change to '%s'¥n"),        //  ●プロトコル変更
360 :                                       SubProtocolNameByXym((AJCXYMPROTOCOL)Param));   break;
361 :         case AJCXYN_COMPLETE:   AjcVthPrintF(hVthLog, TEXT("Complete.¥n"));             break;  //  ●送信正常終了
362 :         case AJCXYN_RX_CAN:     AjcVthPrintF(hVthLog, TEXT("Stop by Received CAN¥n"));  break;  //  ●送信中止 (ＣＡＮ受信)
363 :         case AJCXYN_USERSTOP:   AjcVthPrintF(hVthLog, TEXT("Stop by user¥n"));          break;  //  ●送信中止 (ユーザ停止)
364 :         case AJCXYN_TIMEOUT:    AjcVthPrintF(hVthLog, TEXT("Stop by Timeout¥n"));       break;  //  ●送信中止 (タイムアウト)
365 :         case AJCXYN_ABORT:      AjcVthPrintF(hVthLog, TEXT("Stop by abort¥n"));         break;  //  ●送信中止 (その他)
366 :     }
367 :     if ((knd & AJCXYN_END) != 0) {      // 転送終了
368 :         AjcEnableGroup(GetDlgItem(hDlgMain, IDC_GRP_PROTOCOL), TRUE , TRUE );
369 :         EnableWindow(GetDlgItem(hDlgMain, IDC_CMD_SEND), TRUE );
370 :         EnableWindow(GetDlgItem(hDlgMain, IDC_CMD_STOP), FALSE);
371 :         if (hFind != -1) _findclose(hFind);
372 :         hFind = -1;
373 :         if (hFile != INVALID_HANDLE_VALUE)  CloseHandle(hFile);
374 :         hFile = INVALID_HANDLE_VALUE;
375 :     }
376 : }
377 : //-------------------------------------------------------------------------------------------------------//
378 : //  データ送出  コールバック関数                                                                         //
379 : //-------------------------------------------------------------------------------------------------------//
380 : static VO CALLBACK cbSend(C_VOP pTxD, UI len, UX cbp)
381 : {
382 :     if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_TXERR) && len >= 128) {
383 :         UB sv;
384 :         sv = *((UBP)pTxD + (len - 1));
385 :         if (AjcGetDlgItemSInt(hDlgMain, IDC_INP_TXERR) > (rand() % 100)) {
386 :             *((UBP)pTxD + (len - 1)) ^= 0xFF;
387 :         }
388 :         AjcScpSendBinData(hScp, pTxD, len);
389 :         *((UBP)pTxD + (len - 1))  = sv;
390 :     }
391 :     else {
392 :         AjcScpSendBinData(hScp, pTxD, len);
393 :     }
394 : }
395 :
396 : //-------------------------------------------------------------------------------------------------------//
397 : //  ＣＯＭポートオープン時の表示                                                                         //
398 : //-------------------------------------------------------------------------------------------------------//
399 : static  VO     ShowOnOpened(VO)
400 : {
401 :     UT  txt[64];
402 :
403 :     AjcSnPrintF(txt, AJCTSIZE(txt), TEXT("ファイル送信 SW_XYModemTx ( %s )"), AjcScpGetPortName(hScp));
404 :     SetWindowText(hDlgMain, txt);                           //  ウインドタイトル
405 :     AjcSetDlgItemStr(hDlgMain, IDC_CMD_OPEN, TEXT("CLOSE"));    //  ボタンフェース
```

```
406 :     EnableWindow(GetDlgItem(hDlgMain, IDC_CMD_EASY    ), FALSE);
407 :     EnableWindow(GetDlgItem(hDlgMain, IDC_CMD_SEND    ), TRUE );
408 :     EnableWindow(GetDlgItem(hDlgMain, IDC_CMD_STOP    ), FALSE);
409 : }
410 : //-----------------------------------------------------------------------------------------------------------//
411 : //  ＣＯＭポートクローズ時の表示                                                                             //
412 : //-----------------------------------------------------------------------------------------------------------//
413 : static  VO      ShowOnClosed(VO)
414 : {
415 :     SetWindowText(hDlgMain, TEXT("ファイル送信 SW_XYModemTx ( Disconnect )"));       //  ウインドタイトル
416 :     AjcSetDlgItemStr(hDlgMain, IDC_CMD_OPEN, TEXT("OPEN"));            //  ボタンフェース
417 :     EnableWindow(GetDlgItem(hDlgMain, IDC_CMD_EASY    ), TRUE );
418 :     EnableWindow(GetDlgItem(hDlgMain, IDC_CMD_SEND    ), FALSE);
419 :     EnableWindow(GetDlgItem(hDlgMain, IDC_CMD_STOP    ), FALSE);
420 : }
421 : //-----------------------------------------------------------------------------------------------------------//
422 : //  プロトコル種別値変換                                                                                     //
423 : //-----------------------------------------------------------------------------------------------------------//
424 : static AJCXYMPROTOCOL SubGetProtocol(EPROTOCOL protocol)
425 : {
426 :     AJCXYMPROTOCOL  rc;
427 :
428 :     switch (protocol) {
429 :         default:
430 :         case XM_SUM:    rc = AJCXYP_XMODEM_SUM;     break;
431 :         case XM_CRC:    rc = AJCXYP_XMODEM_CRC;     break;
432 :         case XM_1K:     rc = AJCXYP_XMODEM_1K;      break;
433 :         case YM_STD:    rc = AJCXYP_YMODEM_STD;     break;
434 :         case YM_G:      rc = AJCXYP_YMODEM_G;       break;
435 :     }
436 :     return rc;
437 : }
438 : //-----------------------------------------------------------------------------------------------------------//
439 : //  プロトコル名文字列取得                                                                                   //
440 : //-----------------------------------------------------------------------------------------------------------//
441 : static UTP SubProtocolNameByRbt(EPROTOCOL protocol)
442 : {
443 :     UTP     rc = TEXT("");
444 :
445 :     switch (protocol) {
446 :         case XM_SUM:    rc = TEXT("X-MODEM ( SUM )");   break;
447 :         case XM_CRC:    rc = TEXT("X-MODEM ( CRC )");   break;
448 :         case XM_1K:     rc = TEXT("X-MODEM ( 1KB )");   break;
449 :         case YM_STD:    rc = TEXT("Y-MODEM"         );  break;
450 :         case YM_G:      rc = TEXT("Y-MODEM-G"       );  break;
451 :     }
452 :     return rc;
453 : }
454 : //-----------------------------------------------------------------------------------------------------------//
455 : static  UTP SubProtocolNameByXym(AJCXYMPROTOCOL protocol)
456 : {
457 :     UTP     rc = TEXT("");
458 :
459 :     switch (protocol) {
460 :         case AJCXYP_XMODEM_SUM: rc = TEXT("X-MODEM ( SUM )");   break;
461 :         case AJCXYP_XMODEM_CRC: rc = TEXT("X-MODEM ( CRC )");   break;
462 :         case AJCXYP_XMODEM_1K:  rc = TEXT("X-MODEM ( 1KB )");   break;
463 :         case AJCXYP_YMODEM_STD: rc = TEXT("Y-MODEM"         );  break;
464 :         case AJCXYP_YMODEM_G:   rc = TEXT("Y-MODEM-G"       );  break;
465 :     }
466 :     return rc;
467 : }
```

51.7.2. SW_XYModemRx（XMODEM/YMODEM ファイル受信）

以下のサンプルプログラムでは、ＸＭＯＤＥＭ／ＹＭＯＤＥＭによるファイル受信を行います。




```
 1 : //
 2 : //  SW_XYModemRx.c
 3 : //
 4 : #include    <AjrCstXX.h>
 5 : #include    <io.h>
 6 : #include    <tchar.h>
 7 : #include    <sys/stat.h>
 8 : #include    "resource.h"
 9 :
10 : #define WM_SCPEVENT     (WM_USER + 100)
11 :
12 : //-------------------------------------------------------------------------------------------------------------//
13 : //  定数                                                                                                       //
14 : //-------------------------------------------------------------------------------------------------------------//
15 : typedef enum {COMM_232C , COMM_LAN}               ECOMM;
16 : typedef enum {XM_SUM, XM_CRC, XM_1K, YM_STD, YM_G}  EPROTOCOL;
17 :
18 : //-------------------------------------------------------------------------------------------------------------//
19 : //  ワーク                                                                                                     //
20 : //-------------------------------------------------------------------------------------------------------------//
21 : static  HINSTANCE       hInst;                        //  ＤＬＬインスタンスハンドル
22 : static  HWND            hDlgMain;                     //  ダイアログボックスハンドル
23 : static  HWND            hVthLog;
24 :
25 : static  HAJCXYM         hXym;                         //  XYMedemインスタンスハンドル
26 : static  HAJCSCP         hScp;                         //  ＳＣＭインスタンスハンドル
27 : static  AJCXYMFILEINFO  FileInfo;                     //  ファイル情報（名称，サイズ，日付）
28 :
29 : static  HANDLE          hFile = INVALID_HANDLE_VALUE;  //  受信ファイルハンドル
30 : static  SLL             RxRemBytes;                   //  受信ファイルサイズカウンタ
31 : static  UT              DirPath [MAX_PATH];           //  ディレクトリ・パス名
32 : static  UT              FilePath[MAX_PATH];           //  ファイルパス名
33 :
34 : //-------------------------------------------------------------------------------------------------------------//
35 : //  内部サブ関数                                                                                               //
36 : //-------------------------------------------------------------------------------------------------------------//
37 : AJC_DLGPROC_DEF(Main);
38 :
39 : static VO CALLBACK cbNotice (UI knd, UX Param              , UX cbp);
40 : static VO CALLBACK cbSend   (C_VOP pTxD, UI len            , UX cbp);
41 :
42 : static  VO              ShowOnOpened(VO);
43 : static  VO              ShowOnClosed(VO);
44 : static  AJCXYMPROTOCOL  SubGetProtocol      (EPROTOCOL      protocol);
45 : static  UTP             SubProtocolNameByRbt(EPROTOCOL      protocol);
46 : static  UTP             SubProtocolNameByXym(AJCXYMPROTOCOL protocol);
47 :
48 : //=============================================================================================================//
49 : //                                                                                                             //
```

```
 50 : //  ＷｉｎＭａｉｎ                                                                                    //
 51 : //                                                                                                  //
 52 : //==================================================================================================//
 53 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
 54 : {
 55 :     MSG     msg;
 56 :
 57 :     hInst = hInstance;
 58 :
 59 :     //----- メイン・ダイアログオープン ----------------//
 60 :     hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_MAIN), NULL, AJC_DLGPROC_NAME(Main));
 61 :     ShowWindow(hDlgMain, SW_SHOW);
 62 :
 63 :     //----- メッセージループ --------------------------//
 64 :     while (GetMessage(&msg, NULL, 0, 0)) {
 65 :         do {
 66 :             if (IsDialogMessage(hDlgMain, &msg)) break;
 67 :             TranslateMessage(&msg);
 68 :             DispatchMessage (&msg);
 69 :         } while (0);
 70 :     }
 71 :
 72 :     return (int)msg.wParam ;
 73 : }
 74 : //==================================================================================================//
 75 : //                                                                                                  //
 76 : //  ダイアログ・プロシージャ                                                                        //
 77 : //                                                                                                  //
 78 : //==================================================================================================//
 79 : //----- ダイアログ初期化 ---------------------------------------------------------------------------//
 80 : AJC_DLGPROC(Main, WM_INITDIALOG     )
 81 : {
 82 :     UI  t0, t1, r0, r1;
 83 :
 84 :     hDlgMain = hDlg;
 85 :     hVthLog  = GetDlgItem(hDlg, IDC_VTH_LOG);
 86 :
 87 :     //--- ツールチップ ------------------------------------//
 88 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_TXT_T0), TEXT("受信要求(NAK/C/G)送出周期[ms]"));
 89 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_TXT_T1), TEXT("データ／ＥＯＴ受信待ちタイマ[ms]"));
 90 :     AjcTipTextAdd(GetDlgItem(hDlg, IDC_TXT_R1), TEXT("再送要求(NAK)送出回数"));
 91 :     //--- ＳＣＰインスタンス生成 --------------------------//
 92 :     hScp = AjcScpCreate();
 93 :     AjcScpSetMode(hScp, hDlg, WM_SCPEVENT, AJCSCP_CM_BIN);
 94 :     AjcScpSetEvtMask(hScp, AJCSCP_EV_PORTSTATE | AJCSCP_EV_RXCHUNK | AJCSCP_EV_TXEMPTY);
 95 :     //--- テキストボックスへのドロップを許可 --------------//
 96 :     AjcEnableDlgItemToDrop (hDlg, IDC_TXT_XM_FILE, AJCDROP_FILE);
 97 :     AjcEnableDlgItemToDrop (hDlg, IDC_TXT_YM_DIR , AJCDROP_DIR );
 98 :     //--- ラジオボタングループ化 --------------------------//
 99 :     AjcSbcRadioBtns(GetDlgItem(hDlg, IDC_GRP_PROTOCOL));
100 :     //--- X-Modem / Y-Modem インスタンス生成 ---------------//
101 :     hXym = AjcXymCreate(0, cbNotice, NULL, NULL, cbSend);
102 :     //--- ダイアログ項目の初期化 ---------------------------//
103 :     AjcXymGetRxTimeInfo(hXym, &t0, &r0, &t1, &r1);
104 :     AjcSetDlgItemUInt(hDlg, IDC_GRP_PROTOCOL, XM_SUM  );
105 :     AjcSetDlgItemStr (hDlg, IDC_TXT_XM_FILE , TEXT(""));
106 :     AjcSetDlgItemStr (hDlg, IDC_TXT_YM_DIR  , TEXT(""));
107 :     AjcSetDlgItemChk (hDlg, IDC_CHK_TXERR    , FALSE  );
108 :     AjcSetDlgItemUInt(hDlg, IDC_INP_TXERR    , 0      );
109 :     AjcSetDlgItemUInt(hDlg, IDC_TXT_T0       , t0     );
110 :     AjcSetDlgItemUInt(hDlg, IDC_TXT_R0       , r0     );
111 :     AjcSetDlgItemUInt(hDlg, IDC_TXT_T1       , t1     );
112 :     AjcSetDlgItemUInt(hDlg, IDC_TXT_R1       , r1     );
113 :     //----- ダイアログ項目のロード --------------------------//
114 :     AjcLoadAllControlSettings (hDlg, TEXT("DlgSetting"), AJCOPT2(AJCCTL_SELACT_, ALL, NTCRBT));
115 :     //--- ポートオープン ----------------------------------//
116 :     SendMessage(hDlg, WM_COMMAND, MAKELONG(IDC_CMD_OPEN, BN_CLICKED),0);
117 :
118 :     return TRUE;
119 : }
120 : //----- ウインド破棄 -------------------------------------------------------------------------------//
121 : AJC_DLGPROC(Main, WM_DESTROY        )
122 : {
123 :     //----- インスタンス破棄 --------------------------//
124 :     AjcScpDelete(hScp);     //  SCP インスタンス消去
125 :     AjcXymDelete(hXym);     //  X-Modem / Y-Modem インスタンス消去
126 :     //----- ダイアログ項目のセーブ --------------------//
127 :     AjcSaveAllControlSettings(hDlg);
128 :
129 :     PostQuitMessage(0);
130 :     return TRUE;
131 : }
132 : //----- ＳＣＭイベント通知 -------------------------------------------------------------------------//
133 : AJC_DLGPROC(Main, WM_SCPEVENT       )
134 : {
135 :     UI      time = GetTickCount();
136 :     UI      len, param;
137 :     union {UBP pBin; UTP pTxt; VOP vp;} u;
138 :     UT      txt[128];
139 :
```

```
140 :     AjcScpGetEventData(hScp, lParam, &u.vp, &len, &param);              //  イベントデータ取得
141 :     if(wParam & AJCSCP_EV_PORTSTATE) {                                  //  ●ポート状態通知
142 :         switch (param) {
143 :             case AJCSCP_CLOSED: ShowOnClosed(); break;                  //      ・クローズ状態
144 :             case AJCSCP_OPENED: ShowOnOpened(); break;                  //      ・オープン状態
145 :             case AJCSCP_OPENFAIL:                                       //      ・オープン失敗
146 :                 AjcSnPrintF(txt, AJCTSIZE(txt), TEXT("%s open falure!"), AjcScpGetPortPathName(hScp));
147 :                 MessageBox(hDlgMain, txt, TEXT("Error"), MB_ICONERROR);
148 :                 break;
149 :         }
150 :     }
151 :     if(wParam & AJCSCP_EV_RXCHUNK ) {                                   //  ●チャンクデータ受信通知（バイナリ）
152 :         UI      i;
153 :         UBP     pTop, p;
154 :
155 :         if (pTop = malloc(len)) {
156 :             memcpy(p = pTop, u.pBin, len);
157 :             if (AjcGetDlgItemChk(hDlgMain, IDC_CHK_RXERR) && len >= 5) {
158 :                 if (AjcGetDlgItemSInt(hDlgMain, IDC_INP_RXERR) > (rand() % 100)) {
159 :                     *(p + 4) ^= 0xFF;
160 :                 }
161 :             }
162 :             for (i = 0; i < len; i++) {
163 :                 AjcXymPutRxChar(hXym, *p++);
164 :             }
165 :             free(pTop);
166 :         }
167 :         else {
168 :             p = u.pBin;
169 :             for (i = 0; i < len; i++) {
170 :                 AjcXymPutRxChar(hXym, *p++);
171 :             }
172 :         }
173 :     }
174 :     AjcScpRelEventData(hScp, lParam);                                   //  イベントデータ開放
175 :
176 :     return TRUE;
177 : }
178 : //----- ウインドクローズ ------------------------------------------------------------------------------//
179 : AJC_DLGPROC(Main, IDCANCEL          )
180 : {
181 :     DestroyWindow(hDlg);
182 :     return TRUE;
183 : }
184 : //----- オープン／クローズ ボタン ---------------------------------------------------------------------//
185 : AJC_DLGPROC(Main, IDC_CMD_OPEN      )
186 : {
187 :     if (AjcScpIsOpened(hScp)) {
188 :         //  通信中ならば、中止ボタン押下
189 :         if (AjcXymGetState(hXym) != 0) {
190 :             SendMessage(hDlg, WM_COMMAND, MAKELONG(IDC_CMD_STOP, BN_CLICKED), (LPARAM)GetDlgItem(hDlg, IDC_CMD_OPEN));
191 :         }
192 :         //  ポートクローズ
193 :         AjcScpClose(hScp);
194 :     }
195 :     else {
196 :         //  ポートオープン
197 :         AjcScpOpenDefault(hScp);
198 :     }
199 :     return TRUE;
200 : }
201 : //----- ポート設定ボタン ------------------------------------------------------------------------------//
202 : AJC_DLGPROC(Main, IDC_CMD_EASY      )
203 : {
204 :     AjcScpDlgParamEasy(hScp, hDlg);
205 :     return TRUE;
206 : }
207 : //----- プロトコルの選択 ------------------------------------------------------------------------------//
208 : AJC_DLGPROC(Main, IDC_GRP_PROTOCOL  )
209 : {
210 :     if (HIWORD(wParam) == AJCRBTN_SELECT) {
211 :         if (lParam <= XM_1K) {
212 :             AjcEnableGroup(GetDlgItem(hDlg, IDC_GRP_XM_FILE), TRUE , TRUE );
213 :             AjcEnableGroup(GetDlgItem(hDlg, IDC_GRP_YM_DIR ), FALSE, FALSE);
214 :         }
215 :         else {
216 :             AjcEnableGroup(GetDlgItem(hDlg, IDC_GRP_XM_FILE), FALSE, FALSE);
217 :             AjcEnableGroup(GetDlgItem(hDlg, IDC_GRP_YM_DIR ), TRUE , TRUE );
218 :         }
219 :     }
220 :     return TRUE;
221 : }
222 : //----- ＸＭＯＤＥＭ受信ファイル名設定 ----------------------------------------------------------------//
223 : AJC_DLGPROC(Main, IDC_CMD_XM_FILE   )
224 : {
225 :     struct __stat64 st;
226 :
227 :     if (_tstat64(FilePath, &st) != 0) {
228 :         FilePath[0] = 0;
229 :     }
```

```
230 :     if (AjcGetSaveFile(hDlg, TEXT("受信ファイル"), TEXT("AllFiles(*.*)/*.*"), TEXT("txt"), FilePath, MAX_PATH)) {
231 :         AjcSetDlgItemStr(hDlg, IDC_TXT_XM_FILE, FilePath);
232 :     }
233 :
234 :     return TRUE;
235 : }
236 : //----- ＹＭＯＤＥＭ受信フォルダ名設定 ----------------------------------------------------------------------//
237 : AJC_DLGPROC(Main, IDC_CMD_YM_DIR    )
238 : {
239 :     if (AjcGetFolderName(hDlg, TEXT("受信フォルダ"), TEXT("このフォルダへ全受信ファイルを格納します"), DirPath, MAX_PATH)) {
240 :         AjcSetDlgItemStr(hDlg, IDC_TXT_YM_DIR, DirPath);
241 :     }
242 :     return TRUE;
243 : }
244 : //----- 受信開始ボタン --------------------------------------------------------------------------------------//
245 : AJC_DLGPROC(Main, IDC_CMD_RECV      )
246 : {
247 :     EPROTOCOL   protocol = (EPROTOCOL)AjcGetDlgItemUInt(hDlg, IDC_GRP_PROTOCOL);
248 :     UI          t0 = AjcGetDlgItemUInt(hDlg, IDC_TXT_T0);
249 :     UI          r0 = AjcGetDlgItemUInt(hDlg, IDC_TXT_R0);
250 :     UI          t1 = AjcGetDlgItemUInt(hDlg, IDC_TXT_T1);
251 :     UI          r1 = AjcGetDlgItemUInt(hDlg, IDC_TXT_R1);
252 :
253 :     AjcXymSetRxTimeInfo (hXym, t0, r0, t1, r1);                             //  タイマ情報設定
254 :     do {
255 :         if (protocol <= XM_1K) {                                            //  Ｘ－ＭＯＤＥＭ？
256 :             UT      drv[_MAX_DRIVE], dir[_MAX_DIR];
257 :             AjcGetDlgItemStr(hDlg, IDC_TXT_XM_FILE, FilePath, MAX_PATH);    //      受信ファイルパス名設定
258 :             MAjcSplitPath(FilePath, drv, dir, NULL, NULL);                  //      ディレクトリパス名設定
259 :             MAjcMakePath (DirPath, drv, dir, NULL, NULL);                   //      ・
260 :         }
261 :         else {                                                              //  Ｙ－ＭＯＤＥＭ？
262 :             AjcGetDlgItemStr(hDlg, IDC_TXT_YM_DIR, DirPath, MAX_PATH);      //      格納フォルダ名設定
263 :         }
264 :
265 :         AjcEnableGroup(GetDlgItem(hDlgMain, IDC_GRP_PROTOCOL), FALSE, FALSE);
266 :         EnableWindow(GetDlgItem(hDlgMain, IDC_CMD_SEND), FALSE);
267 :         EnableWindow(GetDlgItem(hDlgMain, IDC_CMD_STOP), TRUE );
268 :
269 :         AjcVthPrintF(hVthLog, TEXT("========== RECEIVING START (%s) ==========¥n"), SubProtocolNameByRbt(protocol));
270 :         AjcXymRxStart(hXym, SubGetProtocol(protocol));  //  受信開始
271 :     } while(0);
272 :
273 :     return TRUE;
274 : }
275 : //----- 受信中止ボタン --------------------------------------------------------------------------------------//
276 : AJC_DLGPROC(Main, IDC_CMD_STOP      )
277 : {
278 :     AjcXymStop(hXym);
279 :     AjcEnableGroup(GetDlgItem(hDlgMain, IDC_GRP_PROTOCOL), TRUE , TRUE );
280 :     EnableWindow(GetDlgItem(hDlgMain, IDC_CMD_SEND  ), TRUE );
281 :     EnableWindow(GetDlgItem(hDlgMain, IDC_CMD_STOP  ), FALSE);
282 :     return TRUE;
283 : }
284 : //-----------------------------------------------------------------------------------------------------------//
285 : AJC_DLGMAP_DEF(Main)
286 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG      )
287 :     AJC_DLGMAP_MSG(Main, WM_DESTROY         )
288 :     AJC_DLGMAP_MSG(Main, WM_SCPEVENT        )
289 :     AJC_DLGMAP_CMD(Main, IDCANCEL           )
290 :     AJC_DLGMAP_CMD(Main, IDC_GRP_PROTOCOL   )
291 :     AJC_DLGMAP_CMD(Main, IDC_CMD_XM_FILE    )
292 :     AJC_DLGMAP_CMD(Main, IDC_CMD_YM_DIR     )
293 :     AJC_DLGMAP_CMD(Main, IDC_CMD_OPEN       )
294 :     AJC_DLGMAP_CMD(Main, IDC_CMD_EASY       )
295 :     AJC_DLGMAP_CMD(Main, IDC_CMD_RECV       )
296 :     AJC_DLGMAP_CMD(Main, IDC_CMD_STOP       )
297 : AJC_DLGMAP_END
298 :
299 : //-----------------------------------------------------------------------------------------------------------//
300 : //  ファイル書き込み                                                                                         //
301 : //-----------------------------------------------------------------------------------------------------------//
302 : static  VO  SubWrite(VOP pDat, UL len)
303 : {
304 :     UL  bytes;
305 :     if (hFile != INVALID_HANDLE_VALUE) {
306 :         if (RxRemBytes > (SLL)len)  {WriteFile(hFile, pDat, (UI)len      , &bytes, NULL); RxRemBytes -= len;}
307 :         else                        {WriteFile(hFile, pDat, (UI)RxRemBytes, &bytes, NULL); RxRemBytes  = 0;  }
308 :     }
309 : }
310 : //-----------------------------------------------------------------------------------------------------------//
311 : //  イベント通知  コールバック関数                                                                           //
312 : //-----------------------------------------------------------------------------------------------------------//
313 : static VO CALLBACK cbNotice(UI knd, UX Param, UX cbp)
314 : {
315 :     switch (knd) {
316 :         case AJCXYN_RXFILEINFO: memcpy(&FileInfo, (VO*)Param, sizeof FileInfo);             //  ●ファイル情報受信
317 :                                 if ( AjcGetDlgItemUInt(hDlgMain, IDC_GRP_PROTOCOL) >= YM_STD) { //    ＹＭＯＤＥＭ？
318 :                                     MAjcStrCpy(FilePath, MAX_PATH, DirPath);        //        受信ファイル名設定
319 :                                     AjcPathCat(FilePath, FileInfo.pFName, MAX_PATH);//        ・
```

```
320 :                               }
321 :                               AjcVthPrintF(hVthLog, TEXT("File receive '%s'¥n"), FilePath);   //      ファイル名ログ表示
322 :                               RxRemBytes  = FileInfo.size;                                    //      受信ファイルサイズ設定
323 :                               if (hFile != INVALID_HANDLE_VALUE) CloseHandle(hFile);          //      （念の為）前ファイルクローズ
324 :                               hFile = CreateFile(FilePath, GENERIC_WRITE, 0, NULL,            //      受信ファイルオープン
325 :                                   CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
326 :                               break;
327 :
328 :           case AJCXYN_RXREC_128:  AjcVthPrintF(hVthLog, TEXT("."));SubWrite((VO*)Param, 128);   //  ●１レコード受信（128 Ｂ）
329 :                               break;
330 :           case AJCXYN_RXREC_1K:   AjcVthPrintF(hVthLog, TEXT("."));SubWrite((VO*)Param, 1024);  //  ●１レコード受信（１ＫＢ）
331 :                               break;
332 :           case AJCXYN_RETRY:      AjcVthPrintF(hVthLog, TEXT("R"));                           //  ●再送発生
333 :                               break;
334 :           case AJCXYN_EOF:        AjcVthPrintF(hVthLog, TEXT("E¥n"));                         //  ●ファイル終端（ＥＯＦ）
335 :                               if (FileInfo.time) {                                            //      ファイル日付あり？
336 :                                   SYSTEMTIME  st;
337 :                                   FILETIME    ft;
338 :                                   AjcTime1970ToSysTime(FileInfo.time, &st);                   //          システムタイムへ変更
339 :                                   SystemTimeToFileTime(&st, &ft);                             //          ファイルタイムに変更
340 :                                   SetFileTime(hFile, &ft, &ft, &ft);                          //          ファイル日時設定
341 :                               }
342 :                               CloseHandle(hFile);
343 :                               hFile = INVALID_HANDLE_VALUE;
344 :                               break;
345 :
346 :           case AJCXYN_PROTOCOL:   AjcVthPrintF(hVthLog, TEXT("Protocol change to '%s'¥n"),    //  ●プロトコル変更
347 :                                       SubProtocolNameByXym((AJCXYMPROTOCOL)Param));
348 :                               break;
349 :           case AJCXYN_COMPLETE:   AjcVthPrintF(hVthLog, TEXT("Complete.¥n"));                 //  ●受信正常終了
350 :                               break;
351 :           case AJCXYN_RX_CAN:     AjcVthPrintF(hVthLog, TEXT("Stop by Received CAN¥n"));      //  ●受信中止（ＣＡＮ受信）
352 :                               break;
353 :           case AJCXYN_USERSTOP:   AjcVthPrintF(hVthLog, TEXT("Stop by user¥n"));              //  ●受信中止（ユーザ停止）
354 :                               break;
355 :           case AJCXYN_TIMEOUT:    AjcVthPrintF(hVthLog, TEXT("Stop by Timeout¥n"));           //  ●受信中止（タイムアウト）
356 :                               break;
357 :           case AJCXYN_ABORT:      AjcVthPrintF(hVthLog, TEXT("Stop by abort¥n"));             //  ●受信中止（その他）
358 :                               break;
359 :       }
360 :       if ((knd & AJCXYN_END) != 0) {      // 転送終了
361 :           AjcEnableGroup(GetDlgItem(hDlgMain, IDC_GRP_PROTOCOL), TRUE, TRUE);
362 :           EnableWindow(GetDlgItem(hDlgMain, IDC_CMD_RECV), TRUE );
363 :           EnableWindow(GetDlgItem(hDlgMain, IDC_CMD_STOP), FALSE);
364 :           if (hFile != INVALID_HANDLE_VALUE)  CloseHandle(hFile);
365 :           hFile = INVALID_HANDLE_VALUE;
366 :       }
367 : }
368 : //-------------------------------------------------------------------------------------------------------------//
369 : //  データ送出  コールバック関数                                                                               //
370 : //-------------------------------------------------------------------------------------------------------------//
371 : static VO CALLBACK cbSend(C_VOP pTxD, UI len, UX cbp)
372 : {
373 :     AjcScpSendBinData(hScp, pTxD, len);
374 : }
375 :
376 : //-------------------------------------------------------------------------------------------------------------//
377 : //  ＣＯＭポートオープン時の表示                                                                               //
378 : //-------------------------------------------------------------------------------------------------------------//
379 : static  VO      ShowOnOpened(VO)
380 : {
381 :     UT  txt[64];
382 :
383 :     AjcSnPrintF(txt, AJCTSIZE(txt), TEXT("ファイル受信 SW_XYModemRx ( %s )"), AjcScpGetPortName(hScp));
384 :     SetWindowText(hDlgMain, txt);                                       //  ウインドタイトル
385 :     AjcSetDlgItemStr(hDlgMain, IDC_CMD_OPEN, TEXT("CLOSE"));            //  ボタンフェース
386 :     EnableWindow(GetDlgItem(hDlgMain, IDC_CMD_EASY    ), FALSE);
387 :     EnableWindow(GetDlgItem(hDlgMain, IDC_CMD_SEND    ), TRUE );
388 :     EnableWindow(GetDlgItem(hDlgMain, IDC_CMD_STOP    ), FALSE);
389 : }
390 : //-------------------------------------------------------------------------------------------------------------//
391 : //  ＣＯＭポートクローズ時の表示                                                                               //
392 : //-------------------------------------------------------------------------------------------------------------//
393 : static  VO      ShowOnClosed(VO)
394 : {
395 :     SetWindowText(hDlgMain, TEXT("ファイル受信 SW_XYModemRx ( Disconnect )"));   //  ウインドタイトル
396 :     AjcSetDlgItemStr(hDlgMain, IDC_CMD_OPEN, TEXT("OPEN"));             //  ボタンフェース
397 :     EnableWindow(GetDlgItem(hDlgMain, IDC_CMD_EASY    ), TRUE );
398 :     EnableWindow(GetDlgItem(hDlgMain, IDC_CMD_SEND    ), FALSE);
399 :     EnableWindow(GetDlgItem(hDlgMain, IDC_CMD_STOP    ), FALSE);
400 : }
401 : //-------------------------------------------------------------------------------------------------------------//
402 : //  プロトコル種別値変換                                                                                       //
403 : //-------------------------------------------------------------------------------------------------------------//
404 : static AJCXYMPROTOCOL SubGetProtocol(EPROTOCOL protocol)
405 : {
406 :     AJCXYMPROTOCOL  rc;
407 :
408 :     switch (protocol) {
409 :         default:
```

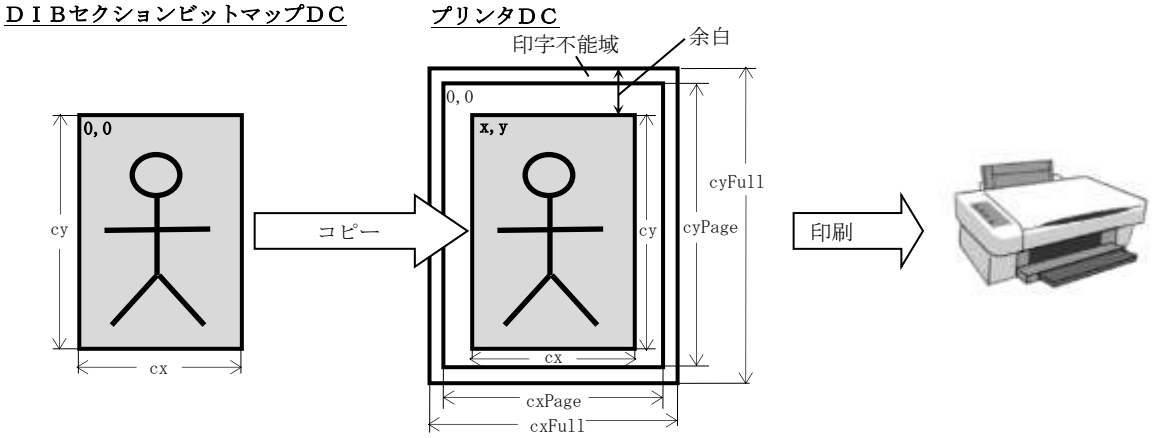
```
410 :         case XM_SUM:    rc = AJCXYP_XMODEM_SUM;     break;
411 :         case XM_CRC:    rc = AJCXYP_XMODEM_CRC;     break;
412 :         case XM_1K:     rc = AJCXYP_XMODEM_1K;      break;
413 :         case YM_STD:    rc = AJCXYP_YMODEM_STD;     break;
414 :         case YM_G:      rc = AJCXYP_YMODEM_G;       break;
415 :     }
416 :     return rc;
417 : }
418 : //-----------------------------------------------------------------------------------------------------------//
419 : //  プロトコル名文字列取得                                                                                   //
420 : //-----------------------------------------------------------------------------------------------------------//
421 : static UTP SubProtocolNameByRbt(EPROTOCOL protocol)
422 : {
423 :     UTP     rc = TEXT("");
424 :
425 :     switch (protocol) {
426 :         case XM_SUM:    rc = TEXT("X-MODEM ( SUM )");   break;
427 :         case XM_CRC:    rc = TEXT("X-MODEM ( CRC )");   break;
428 :         case XM_1K:     rc = TEXT("X-MODEM ( 1KB )");   break;
429 :         case YM_STD:    rc = TEXT("Y-MODEM"         );  break;
430 :         case YM_G:      rc = TEXT("Y-MODEM-G"       );  break;
431 :     }
432 :     return rc;
433 : }
434 : //-----------------------------------------------------------------------------------------------------------//
435 : static  UTP SubProtocolNameByXym(AJCXYMPROTOCOL protocol)
436 : {
437 :     UTP     rc = TEXT("");
438 :
439 :     switch (protocol) {
440 :         case AJCXYP_XMODEM_SUM: rc = TEXT("X-MODEM ( SUM )");   break;
441 :         case AJCXYP_XMODEM_CRC: rc = TEXT("X-MODEM ( CRC )");   break;
442 :         case AJCXYP_XMODEM_1K:  rc = TEXT("X-MODEM ( 1KB )");   break;
443 :         case AJCXYP_YMODEM_STD: rc = TEXT("Y-MODEM"         );  break;
444 :         case AJCXYP_YMODEM_G:   rc = TEXT("Y-MODEM-G"       );  break;
445 :     }
446 :     return rc;
447 : }
```

52. 印 刷

プリンタへの印刷用ＡＰＩ群です。
プリンタへの印刷は、１ページ毎の以下のＤＣ（デバイスコンテキスト）への描画により行います。

・プリンタのＤＣへ直接描画（イメージは余白域を含めたサイズ）
・ＤＩＢセクションによるビットマップをマッピングしたＤＣへ描画（イメージは余白域を除いたサイズ）

ＤＩＢセクションによるビットマップへ描画した場合は、当該ビットマップをプリンタＤＣへコピー後印刷を行います。



52.1. 用紙の向き

用紙の向きは、以下のように定義されています。

```
typedef enum {
    AJCPRN_ORIENT_UNKOWN    = 0                     ,   //  不定
    AJCPRN_ORIENT_PORTRAIT  = DMORIENT_PORTRAIT     ,   //  縦
    AJCPRN_ORIENT_LANDSCAPE = DMORIENT_LANDSCAPE    ,   //  横
} AJCPRN_ORIENT;
```

52.2. 余白情報

余白情報は、以下ように定義されています。

```
typedef struct {
    int     l;      //  左端余白
    int     r;      //  右端余白
    int     u;      //  上端余白
    int     d;      //  下端余白
} AJCPRN_MARGIN, *PAJCPRN_MARGIN;
typedef const AJCPRN_MARGIN *PCAJCPRN_MARGIN;
```

52.3. ページ情報

ページ情報は、以下ように定義されています。

```
typedef struct {
    AJCPRN_MARGIN   mmMrg;                  //  余白サイズ[mm]
    AJCPRN_ORIENT   orientation;            //  用紙の向き
}  AJCPRN_PGINFO, *PAJCPRN_PGINFO;
typedef const AJCPRN_PGINFO *PCAJCPRN_PGINFO;
```

52.4. プリンタ情報

プリンタ情報は、以下ように定義されています。

```
typedef struct {
    //----- プリンタ情報 -----------//
    int                 x , y;                  //  余白を除いた描画左上位置
    int                 cx, cy;                 //  余白を除いた描画ピクセル数
    AJCPRN_MRGINFO      MrgIf;                  //  余白情報
    //----- プリンタ諸元 -----------//
    int                 cxFull, cyFull;         //  用紙サイズのピクセル数
    int                 cxPage, cyPage;         //  印字可能領域のピクセル数
    int                 OffsetX;                //  左右印刷不能域のピクセル数
    int                 OffsetY;                //  上下印刷不能域のピクセル数
    int                 FactorX;                //  プリンタの水平軸の倍率
    int                 FactorY;                //  プリンタの垂直軸の倍率
    int                 LogPixelsX;             //  １インチ当たりの横ピクセル数
    int                 LogPixelsY;             //  １インチ当たりの縦ピクセル数
    AJCPRN_ORIENT       Orientation;            //  AJCPRN_ORIENT_PORTRAIT(縦) /
AJCPRN_ORIENT_LANDSCAPE(横)
} AJCPRN_INFO, *PAJCPRN_INFO;
typedef const AJCPRN_INFO *PCAJCPRN_INFO;
```

52.5. サポートＡＰＩ

印刷のサポートＡＰＩ一覧を以下に示します。

#	関 数 名	内容
1	AjcPrnCreate	印刷インスタンス生成
2	AjcPrnDelete	印刷インスタンス消去
3	AjcPrnSetCallbackParam	コールバックパラメタ設定
4	AjcPrnSetCallbackQueryPage	ページ印刷開始コールバック設定
5	AjcPrnSetCallbackByDibSect	ＤＩＢセクション・ビットマップによる描画コールバック設定
6	AjcPrnSetCallbackByPrinter	プリンタＤＣによる描画コールバック設定
7	AjcPrnSelectDlg	プリンタの選択ダイアログ表示
8	AjcPrnPrintDlg	印刷ダイアログ表示
9	AjcPrnSetMargin	余白サイズ設定
10	AjcPrnStart	印刷開始
11	AjcPrnGetInfo	プリンタ設定情報取得

52.5.1. 印刷インスタンス生成 (AjcPrnCreate)

形　式 ： HAJCXYM　AjcPrnCreate (VO);

引　数 ： hWndOwner　- オーナーウインドのハンドル

説　明 ： 印刷用ののインスタンスを生成します。

戻り値 ： ≠ＮＵＬＬ：成功　（インスタンスハンドルを返します）
＝ＮＵＬＬ：エラー（メモリ確保失敗）

52.5.2. 印刷インスタンス消去 (AjcPrnDelete)

形　式 ： VO　AjcPrnDelete (HAJCPRN hPrn);

引　数 ： hPrn　　- 印刷インスタンスハンドル

説　明 ： 印刷用ののインスタンスを消去します。

戻り値 ： なし

52.5.3. コールバックパラメタ設定(AjcPrnSetCallbackParam)

形　式 ： BOOL　AjcPrnSetCallbackParam (HAJCPRN hPrn , UX cbp);

引　数 ： hPrn　　- 印刷インスタンスハンドル
cbp　　- コールバックパラメタ

説　明 ： AjcPrnSetCallbackQueryPage(),AjcPrnSetCallbackByDibSect()やAjcPrnSetCallbackByPrinter()で設定されたコールバック関数を呼び出す際のコールバックパラメタを設定します。

戻り値 ： TRUE　-　成功
FALSE -　失敗

52.5.4. ページ印刷開始コールバック設定 (AjcPrnSetCallbackQueryPage)

形　式 ： BOOL AjcPrnSetCallbackQueryPage(HAJCPRN hPrn,
BOOL (CALLBACK *cbQueryPage)(PCAJCPRN_INFO pPrnInfo, PAJCPRN_PGINFO pPageInfo, UX cbp));

引　数 ： hPrn – 印刷インスタンスハンドル
cbQueryPage – ページ印刷開始を通知するコールバック関数

説　明 ： ページを印刷する前にコールする、コールバック関数を設定します。
このコールバック関数では、ページの情報（余白サイズ、印刷の向き）を変更することができます。

戻り値 ： TRUE – 成功
FALSE – 失敗

コールバック：

cbQueryPage（ページ印刷開始通知）

形　式 ： BOOL CALLBACK ***cbQueryPage***(PCAJCPRN_INFO pPrnInfo, PAJCPRN_PGINFO pPageInfo, UX cbp)；

引　数 ： pPrnInfo – プリンタ情報
pPageInfo – ページ情報へのポインタ
cbp – コールバックパラメタ

説　明 ： このコールバック関数は、ページの印刷を開始する前にコールされます。
pPageInfo で示される情報を書き換えることにより、余白サイズ，印刷の向きを設定し直すことができます。

戻り値 ：TRUE – 印刷を継続する
FALSE – 印刷を終了する（この後、何も印刷せずに終了します）

52.5.5. ＤＩＢセクション・ビットマップによる描画コールバック設定(AjcPrnSetCallbackByDibSect)

形　式 ： BOOL AjcPrnSetCallbackByDibSect (HAJCPRN hPrn,
BOOL (CALLBACK *cbDrawByDibSectDC)(HDC hdc, PCAJCPRN_INFO pPrnInfo, PCAJCDIBINFO pDibInfo, UX cbp));

引　数 ： hPrn – 印刷インスタンスハンドル
cbDrawByDibSectDC – ＤＩＢセクション・ビットマップによる描画コールバック関数

説　明 ： ページを印刷する際にコールする、コールバック関数を設定します。
このコールバック関数には、ＤＩＢセクション・ビットマップ（余白部分を除く描画域のみ）をマップしたＤＣが渡されますので、このＤＣを使用して１ページ分の描画を行います。
この関数を実行した場合、AjcPrnSetCallbackByPrinter ()により設定されたコールバックは無効となります。

戻り値 ： TRUE – 成功
FALSE – 失敗

コールバック：

cbDrawByDibSectDC（ＤＩＢセクション・ビットマップによる描画コールバック）

形　式 ： BOOL CALLBACK ***cbDrawByDibSectDC***(HDC hdc, PCAJCPRN_INFO pPrnInfo, PCAJCDIBINFO pDibInfo, UX cbp)；

引　数 ： hdc – ＤＩＢセクション・ビットマップを割り当てたＤＣ（余白域を除いた描画領域のみ）
pPrnInfo – プリンタ情報へのポインタ
pDibInfo – ＤＩＢセクション情報へのポインタ（AjcDibCreate()参照）
cbp – コールバックパラメタ

説　明 ： このコールバック関数は、ページを印刷する際にコールされます。
hdc は現ディスプレィ互換ＤＣで、32bit/Pixel の DIB がセレクトされています。
描画可能なピクセル数は、pPrnInfo のメンバ（cx, cy）で示されます。
hdc を使用して、１ページ分の描画を行います。

戻り値 ： TRUE – 印刷を継続する
FALSE – 印刷を終了する（このページを印刷後に終了します）

52.5.6. プリンタＤＣによる描画コールバック設定 (AjcPrnSetCallbackByPrinter)

形　式 ： BOOL AjcPrnSetCallbackByPrinter (HAJCPRN hPrn,
BOOL (CALLBACK *cbDrawByPrinterDC)(HDC hdc, PCAJCPRN_INFO pPrnInfo, UX cbp));

引　数 ： hPrn – 印刷インスタンスハンドル
cbDrawByPrinterDC – プリンタＤＣによる描画コールバック関数

説　明 ： ページを印刷する際にコールする、コールバック関数を設定します。
このコールバック関数には、プリンタのＤＣが渡されますので、このＤＣを使用して１ページ分の描画を行います。
この関数を実行した場合、AjcPrnSetCallbackByDibSect()により設定されたコールバックは無効となります。

戻り値 ： TRUE – 成功
FALSE – エラー

コールバック

cbDrawByPrinterDC（プリンタＤＣによる描画コールバック）

形　式 ： BOOL CALLBACK ***cbDrawByDibPrinterDC***(HDC hdc, PCAJCPRN_INFO pPrnInfo, UX cbp)；

引　数 ： hdc – プリンタのＤＣ（余白域を含む全領域）
pPrnInfo – プリンタ情報へのポインタ
cbp – コールバックパラメタ

説　明 ： このコールバック関数は、ページを印刷する際にコールされます。
描画可能なページ全体のピクセル数は、pPrnInfo のメンバ（cxPage, cyPage）で示されます。
余白域を除いた描画領域の位置とサイズは、pPrnInfo のメンバ（x, y, cx, cy）で示されます。
hdc を使用して、１ページ分の描画を行います。

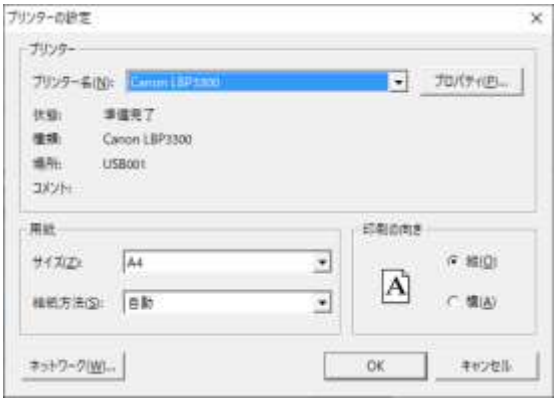
戻り値 ： TRUE – 印刷を継続する
FALSE – 印刷を終了する（このページを印刷後に終了します）

52.5.7. プリンタの選択 (AjcPrnSelectDlg)

形　式 ： BOOL AjcPrnSelectDlg(HAJCPRN hPrn, PAJCPRN_INFO pInfo, UI Flag, HWND hWndOwner);

引　数 ： hPrn – 印刷インスタンスハンドル
pInfo – プリンタ情報を格納するバッファのアドレス（不要時は NULL）
Flag – AJCPRNF_NONETWORKBUTTON を指定で、「ネットワーク」ボタン非表示
hWndOwner – オーナーウインドハンドル

説　明 ： 以下のシステム・ダイアログにより、プリンタの設定を行います。



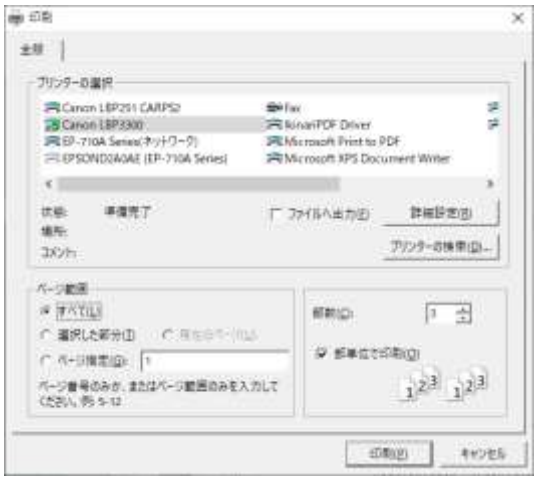
戻り値 ： TRUE –ＯＫボタン押下
FALSE – キャンセルボタン押下

52.5.8. 印刷ダイアログ (AjcPrnPrintDlg)

形　式 ： BOOL　AjcPrnPrintDlg(HAJCPRN hPrn, PAJCPRN_INFO pInfo, PAJCPRN_OPT pOpt, HWND hWndOwner);

引　数 ： hPrn　　－ 印刷インスタンスハンドル
pInfo　　－ プリンタ情報を格納するバッファのアドレス（不要時は NULL）
pOpt　　－ オプション指定情報へのポインタ
hWndOwner　　－ オーナーウインドハンドル

説　明 ： 以下のシステム・ダイアログにより、印刷設定を行います。



or



※オプションフラグで「AJCPRNF_DISABLECOPIES」
「AJCPRNF_DISABLEPAGEINFO」
を指定した場合は上記のダイアログボックスが表示されます。

pOpt は、以下のオプション情報をポイントします。

```
typedef struct {
    ULL     Flags;                  //  オプションフラグ    (入出力)
    UI      nCopies;                //  コピー部数          (  出力)
    UI      nFromPage;              //  開始ページ          (入出力)
    UI      nToPage;                //  終了ページ          (入出力)
    UI      nMinPage;               //  ページ範囲の最小値  (入力  )
    UI      nMaxPage;               //  ページ範囲の最大値  (入力  )
} AJCPRN_OPT, *PAJCPRN_OPT;
typedef const AJCPRN_OPT *PCAJCPRN_OPT;
```

Flags は以下のシンボルの組み合わせです。

シンボル	値(16 進)	入力	出力
AJCPRNF_OUTPUTTOFILE	01 0000 0000	－	「ファイルへ出力」のチェック状態
AJCPRNF_DISABLECOPIES	02 0000 0000	「印刷部数」（グループ）を無効化する	－
AJCPRNF_DISABLEPAGEINFO	04 0000 0000	「印刷範囲」（グループ）を無効にする	－
AJCPRNF_ALLPAGES	0000 0000	「すべて」を選択状態にする	－
AJCPRNF_SELECTION	0000 0001	「選択した部分」を選択状態にする	「選択した部分」の選択状態
AJCPRNF_NOSELECTION	0000 0004	「選択した部分」を無効状態にする	－
AJCPRNF_PAGENUMS	0000 0002	「ページ指定」を選択状態にする	「ページ指定」の選択状態 (1：nFromPage, nToPage が設定されている)
AJCPRNF_NOPAGENUMS	0000 0008	「ページ指定」を無効状態にする	－
AJCPRNF_NOWARNING	0000 0080	既定のプリンタがない場合に警告メッセージ非表示	－
AJCPRNF_PRINTTOFILE	0000 0020	「ファイルへ出力」をチェックする	－
AJCPRNF_DISABLEPRINTTOFILE	0008 0000	「ファイルへ出力」を無効化する	－
AJCPRNF_HIDEPRINTTOFILE	0010 0000	「ファイルへ出力」を非表示	－
AJCPRNF_COLLATE	0000 0010	－	「部単位で印刷」のチェック状態
AJCPRNF_USEDEVMODECOPIES (※1)	0004 0000	アプリで「部単位で印刷」が未サポートである ことを指定	－

※1：「AJCPRNF_USEDEVMODECOPIES」は、アプリケーションが複数のコピーと部単位の印刷をサポートするかどうかを示します。
アプリケーションが複数のコピーと部単位の印刷をサポートしていないことを示すには、入力にこのフラグを設定します。
この場合、オプション情報の **nCopies** メンバーは常に 1 を返し、Flags の **AJCPRNF_COLLATE** は常に 0 が設定されます。
プリンタドライバでコピーと部単位の選択がサポートされている場合もあります。
詳細は、MSDN（**PRINTDLG** 構造体の **PD_USEDEVMODECOPIES / PD_USEDEVMODECOPIESANDCOLLATE**）を参照

戻り値 ： TRUE　－ 印刷ボタン押下
FALSE － キャンセルボタン押下

52.5.9. 余白サイズ設定 (AjcPrnSetMargin)

形　式 ： BOOL　AjcPrnSetMargin (HAJCPRN hPrn, AJCPRN_MARGIN pmmMargin);

引　数 ： hPrn　　- 印刷インスタンスハンドル
pmmMargin - 設定する余白サイズ情報へのポインタ (mm(ミリメートル)単位の値, NULL：最小余白サイズ)

説　明 ： 上下左右の余白域サイズ (mm(ミリメートル)単位の値) を設定します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

52.5.10. 印刷開始 (AjcPrnStart)

形　式 ： int　AjcPrnStart (HAJCPRN hPrn);

引　数 ： hPrn　　- 印刷インスタンスハンドル
pSpace　- 設定する余白サイズ[mm]情報へのポインタ

説　明 ： プリンタの印刷を開始します。
この関数実行中に、cbQueryPage (ページ印刷開始通知), cbDrawByDibSectDC (ＤＩＢセクション・ビットマップによる描画コールバック) or cbDrawByPrinterDC (プリンタＤＣによる描画コールバック) が呼び出され、各ページの描画が行われます。
これらのコールバック関数から FALSE が返されたら印刷を終了します。

戻り値 ： ＝0　- 成功
≠0　- 失敗(GetLastError()で取得した値)

52.5.11. プリンタ設定情報取得 (AjcPrnGetInfo)

形　式 ： C_UTP AjcPrnGetInfo (HAJCPRN hPrn, PAJCPRN_INFO pBuf);

引　数 ： hPrn　　- 印刷インスタンスハンドル
pBuf　　- プリンタ設定情報を格納するバッファのアドレス

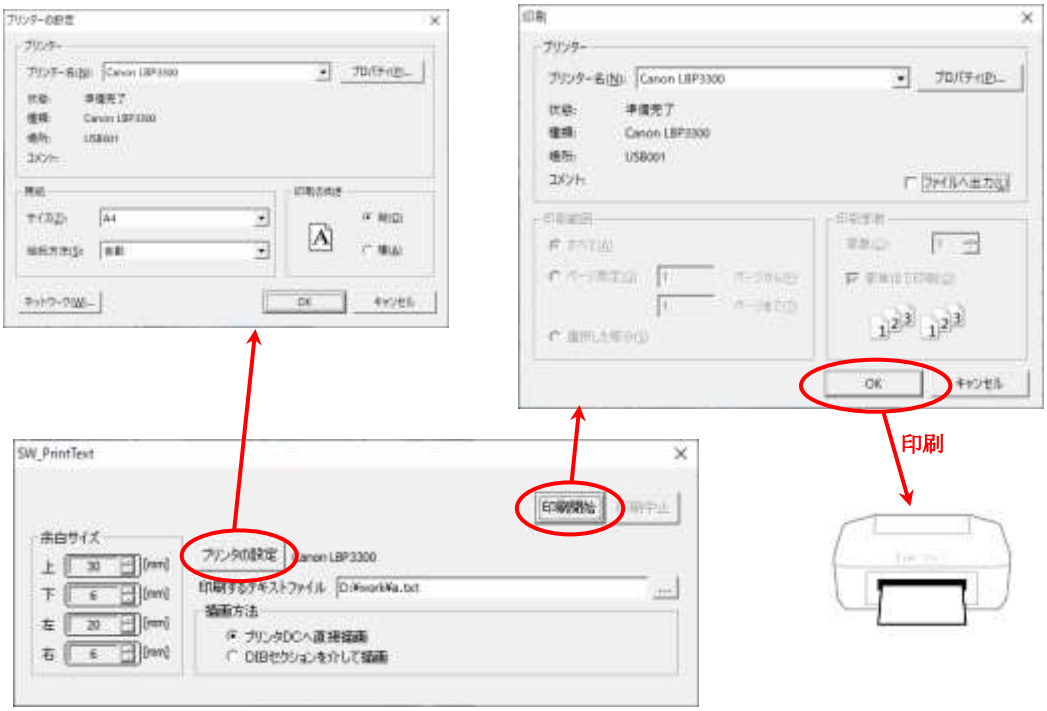
説　明 ： 現在選択されているプリンタの名称と設定情報を取得します。

戻り値 ： ≠NULL - 成功 (プリンタの名称文字列へのポインタ)
＝NULL - 失敗

52.6. サンプルプログラム

52.6.1. SW_PrintText（テキストファイルの印刷）

以下のサンプルプログラムは、テキストファイルを入力し、当該テキストの印刷を行います。
但し、折り返し処理は行わず、右端がはみ出る部分は印刷されません。




```
 1 : //
 2 : //  SW_PrintText.c
 3 : //
 4 : #include    <AjrCstXX.h>
 5 : #include    <Shlwapi.h>
 6 : #include    <io.h>
 7 : #include    <tchar.h>
 8 : #include    <sys/stat.h>
 9 : #include    "resource.h"
10 : #pragma comment(lib, "shlwapi.lib")
11 :
12 : #define     PICL    300
13 :
14 : //-----------------------------------------------------------------------------------------------------------------//
15 : //  ワーク                                                                                                         //
16 : //-----------------------------------------------------------------------------------------------------------------//
17 : static  HINSTANCE       hInst;                    //  ＤＬＬインスタンスハンドル
18 : static  HWND            hDlgMain;                 //  ダイアログボックスハンドル
19 : static  HAJCFILE        hFile = NULL;             //  テキストファイルハンドル
20 : static  HAJCPRN         hPrn;                     //  印刷インスタンス
21 : static  BOOL            fPrintContinue  = TRUE;   //  印刷継続フラグ
22 : static  HFONT           hFont = NULL;             //  フォントオブジェクト
23 : static  AJCPRN_INFO     PrnInfo;                  //  プリンタ情報
24 : static  BOOL            fFileOut = FALSE;         //  ファイルへ出力フラグ
25 :
26 :
27 : //-----------------------------------------------------------------------------------------------------------------//
28 : //  内部サブ関数                                                                                                   //
29 : //-----------------------------------------------------------------------------------------------------------------//
30 : AJC_DLGPROC_DEF(Main);
31 :
32 : static  BOOL CALLBACK cbQueryPage(PCAJCPRN_INFO pPrnInfo, PAJCPRN_PGINFO pPageInfo, UX cbp);
33 : static  BOOL CALLBACK cbDrawByDibSectDC(HDC hdc, PCAJCPRN_INFO pPrnInfo, PCAJCDIBINFO pDibInfo, UX cbp);
34 : static  BOOL CALLBACK cbDrawByPrinterDC(HDC hdc, PCAJCPRN_INFO pPrnInfo, UX cbp);
35 :
36 : //=================================================================================================================//
37 : //                                                                                                                 //
38 : //  ＷｉｎＭａｉｎ                                                                                                //
```

```
 39 : //                                                                                                                  //
 40 : //==========================================================================================================//
 41 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
 42 : {
 43 :     MSG     msg;
 44 :
 45 :     hInst = hInstance;
 46 :
 47 :     //----- メイン・ダイアログオープン ----------------//
 48 :     hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_MAIN), NULL, AJC_DLGPROC_NAME(Main));
 49 :     ShowWindow(hDlgMain, SW_SHOW);
 50 :
 51 :     //----- メッセージループ --------------------------//
 52 :     while (GetMessage(&msg, NULL, 0, 0)) {
 53 :         do {
 54 :             if (IsDialogMessage(hDlgMain, &msg)) break;
 55 :             TranslateMessage(&msg);
 56 :             DispatchMessage (&msg);
 57 :         } while (0);
 58 :     }
 59 :
 60 :     return (int)msg.wParam ;
 61 : }
 62 : //==========================================================================================================//
 63 : //                                                                                                          //
 64 : //  ダイアログ・プロシージャ                                                                                //
 65 : //                                                                                                          //
 66 : //==========================================================================================================//
 67 : //----- ダイアログ初期化 ----------------------------------------------------------------------------------//
 68 : AJC_DLGPROC(Main, WM_INITDIALOG     )
 69 : {
 70 :     AJCPRN_MARGIN   mrg;
 71 :     LOGFONT         lf;
 72 :
 73 :     hDlgMain = hDlg;
 74 :
 75 :     //----- 印刷インスタンス生成 ----------------------//
 76 :     hPrn = AjcPrnCreate();
 77 :     //----- ページ印刷開始コールバック設定 ------------//
 78 :     AjcPrnSetCallbackQueryPage(hPrn, cbQueryPage);
 79 :     //----- プリンタ名表示 ----------------------------//
 80 :     AjcSetDlgItemStr(hDlg, IDC_LBL_PRNAME, AjcPrnGetInfo(hPrn, &PrnInfo));
 81 :     //----- テキストファイル名入力でドロップ可能とする ----//
 82 :     AjcEnableDlgItemToDrop (hDlg, IDC_TXT_FILE, AJCDROP_FILE);
 83 :     //----- ラジオボタン初期化 ------------------------//
 84 :     AjcSetDlgItemChk(hDlg, IDC_RBT_DIB_DC, TRUE);
 85 :     //----- 設定値ロード ------------------------------//
 86 :     AjcLoadAllControlSettings(hDlg, TEXT("Settings"), AJCCTL_SELACT_ALL | AJCCTL_SELACT_NTCALL);
 87 :     //----- 余白設定 ----------------------------------//
 88 :     mrg.u = AjcGetDlgItemUInt(hDlg, IDC_INP_SPC_U);
 89 :     mrg.d = AjcGetDlgItemUInt(hDlg, IDC_INP_SPC_D);
 90 :     mrg.l = AjcGetDlgItemUInt(hDlg, IDC_INP_SPC_L);
 91 :     mrg.r = AjcGetDlgItemUInt(hDlg, IDC_INP_SPC_R);
 92 :     AjcPrnSetMargin(hPrn, &mrg);
 93 :     //----- フォント生成 ------------------------------//
 94 :     lf.lfHeight      = 80;          lf.lfStrikeOut      = 0;
 95 :     lf.lfWidth       = 0;           lf.lfCharSet        = DEFAULT_CHARSET;
 96 :     lf.lfEscapement  = 0;           lf.lfOutPrecision   = OUT_STROKE_PRECIS;
 97 :     lf.lfOrientation = 0;           lf.lfClipPrecision  = CLIP_STROKE_PRECIS;
 98 :     lf.lfWeight      = FW_NORMAL;   lf.lfQuality        = DRAFT_QUALITY;
 99 :     lf.lfItalic      = 0;           lf.lfPitchAndFamily = FF_MODERN | VARIABLE_PITCH;
100 :     lf.lfUnderline   = 0;           MAjcStrCpy(lf.lfFaceName, AJCTSIZE(lf.lfFaceName), TEXT("Arial Unicode MS"));
101 :     hFont = CreateFontIndirect(&lf);
102 :
103 :     return TRUE;
104 : }
105 : //----- ウインド破棄 --------------------------------------------------------------------------------------//
106 : AJC_DLGPROC(Main, WM_DESTROY        )
107 : {
108 :     //----- 設定値セーブ ------------------------------//
109 :     AjcSaveAllControlSettings(hDlg);
110 :     //----- リソース解放 ------------------------------//
111 :     AjcPrnDelete(hPrn);
112 :     DeleteObject(hFont);
113 :     //----- プログラム終了 ----------------------------//
114 :     PostQuitMessage(0);
115 :
116 :     return TRUE;
117 : }
118 : //----- WM_COMMAND ----------------------------------------------------------------------------------------//
```

```
AJC_DLGPROC(Main, WM_COMMAND        )
{
    int     cmd = LOWORD(wParam);

    if (cmd == IDC_INP_SPC_U || cmd == IDC_INP_SPC_D || cmd == IDC_INP_SPC_L || cmd == IDC_INP_SPC_R) {
        if (HIWORD(wParam) == AJCIVN_INTVALUE) {
            AJCPRN_MARGIN   mrg;
            mrg.u = AjcGetDlgItemUInt(hDlg, IDC_INP_SPC_U);
            mrg.d = AjcGetDlgItemUInt(hDlg, IDC_INP_SPC_D);
            mrg.l = AjcGetDlgItemUInt(hDlg, IDC_INP_SPC_L);
            mrg.r = AjcGetDlgItemUInt(hDlg, IDC_INP_SPC_R);
            AjcPrnSetMargin(hPrn, &mrg);
        }
    }
    return TRUE;
}
//----- ウインドクローズ ----------------------------------------------------------------------------------//
AJC_DLGPROC(Main, IDCANCEL          )
{
    DestroyWindow(hDlg);
    return TRUE;
}
//----- プリンタの設定ボタン ------------------------------------------------------------------------------//
AJC_DLGPROC(Main, IDC_CMD_SETPRN    )
{
    if (AjcPrnSelectDlg(hPrn, &PrnInfo, 0, hDlg)) {
        //  プリンタ名表示
        AjcSetDlgItemStr(hDlg, IDC_LBL_PRNAME, AjcPrnGetInfo(hPrn, NULL));
    }
    return TRUE;
}
//----- 印刷開始ボタン ------------------------------------------------------------------------------------//
AJC_DLGPROC(Main, IDC_CMD_START     )
{
    AJCPRN_OPT  opt = {0};
    UT          path[MAX_PATH];

    if (HIWORD(wParam) == BN_CLICKED) {
        AjcGetDlgItemStr(hDlg, IDC_TXT_FILE, path, MAX_PATH);
        if (hFile = AjcFOpen(path, AJCTEC_AUTO)) {
            opt.Flags = (
                AJCPRNF_DISABLECOPIES       |   //  「印刷部数」を無効化する
                AJCPRNF_DISABLEPAGEINFO     |   //  「印刷範囲」を無効にする
            //  AJCPRNF_ALLPAGES            |   //  「すべて」を選択状態にする
            //  AJCPRNF_SELECTION           |   //  「選択した部分」を選択状態にする
            //  AJCPRNF_NOSELECTION         |   //  「選択した部分」を無効状態にする
            //  AJCPRNF_PAGENUMS            |   //  「ページ指定」を選択状態にする
            //  AJCPRNF_NOPAGENUMS          |   //  「ページ指定」を無効状態にする
            //  AJCPRNF_NOWARNING           |   //  既定のプリンタがない場合に警告メッセージ非表示
            //  AJCPRNF_PRINTTOFILE         |   //  「ファイルへ出力」をチェックする
            //  AJCPRNF_DISABLEPRINTTOFILE  |   //  「ファイルへ出力」を無効化する
            //  AJCPRNF_HIDEPRINTTOFILE     |   //  ファイルへ出力」を非表示
            //  AJCPRNF_USEDEVMODECOPIES    |   //  アプリで「部単位で印刷」が未サポートであることを指定
                0);
            opt.nCopies     = 1;                //  コピー部数
            opt.nFromPage   = 1;                //  開始ページ
            opt.nToPage     = 1;                //  終了ページ
            opt.nMinPage    = 1;                //  ページ範囲の最小値
            opt.nMaxPage    = 10;               //  ページ範囲の最小値
            if (AjcPrnPrintDlg(hPrn, NULL, &opt, hDlg)) {
                //--- プリンタ名表示 ------------//
                AjcSetDlgItemStr(hDlg, IDC_LBL_PRNAME, AjcPrnGetInfo(hPrn, NULL));
                //--- 印刷継続フラグ初期化 ------//
                fPrintContinue = TRUE;
                //--- 中止ボタンのみ有効化 ------//
                AjcEnableGroup(GetDlgItem(hDlg, IDC_GRP_ALL ), FALSE, FALSE);
                EnableWindow  (GetDlgItem(hDlg, IDC_CMD_STOP), TRUE);
                //--- 印刷開始 ------------------//
                AjcPrnStart (hPrn, (opt.Flags & AJCPRNF_OUTPUTTOFILE) != 0);
                //--- グレー状態解除 ------------//
                AjcEnableGroup(GetDlgItem(hDlg, IDC_GRP_ALL ), TRUE, TRUE);
                EnableWindow  (GetDlgItem(hDlg, IDC_CMD_STOP), FALSE);
            }
            //--- ファイルクローズ ---------//
            AjcFClose(hFile); hFile = NULL;
        }
        else {
            MessageBox(hDlg, TEXT("ファイルをオープンできません"), TEXT("SW_PrintText"), MB_ICONERROR);
        }
    }
```

```
199 :     return TRUE;
200 : }
201 : //----- 印刷中止ボタン ----------------------------------------------------------------------------------//
202 : AJC_DLGPROC(Main, IDC_CMD_STOP      )
203 : {
204 :     if (HIWORD(wParam) == BN_CLICKED) {
205 :         //--- グレー状態解除 -----------//
206 :         AjcEnableGroup(GetDlgItem(hDlg, IDC_GRP_ALL ), TRUE, TRUE);
207 :         EnableWindow  (GetDlgItem(hDlg, IDC_CMD_STOP), FALSE);
208 :         //--- 印刷終了の旨、設定 -------//
209 :         fPrintContinue = FALSE;
210 :     }
211 :     return TRUE;
212 : }
213 : //----- 印刷テキストファイル設定ボタン ------------------------------------------------------------------//
214 : AJC_DLGPROC(Main, IDC_CMD_FILE      )
215 : {
216 :     UT      path[MAX_PATH];
217 :
218 :     if (HIWORD(wParam) == BN_CLICKED) {
219 :         AjcGetDlgItemStr(hDlg, IDC_TXT_FILE, path, AJCTSIZE(path));
220 :         if (AjcGetOpenFile(hDlg, TEXT("イメージファイル設定"), TEXT("Text File (*.txt)/*.txt/All Files (*.*)/*.*"),
221 :                                                                                 TEXT("txt"), path, MAX_PATH)) {
222 :             AjcSetDlgItemStr(hDlg, IDC_TXT_FILE, path);
223 :         }
224 :     }
225 :     return TRUE;
226 : }
227 : //----- ラジオボタン（プリンタＤＣへ直接描画）----------------------------------------------------------//
228 : AJC_DLGPROC(Main, IDC_RBT_PRT_DC    )
229 : {
230 :     if (HIWORD(wParam) == BN_CLICKED) {
231 :         if (AjcGetDlgItemChk(hDlg, IDC_RBT_PRT_DC)) AjcPrnSetCallbackByPrinter(hPrn, cbDrawByPrinterDC);
232 :         else                                        AjcPrnSetCallbackByDibSect(hPrn, cbDrawByDibSectDC);
233 :     }
234 :     return TRUE;
235 : }
236 : //----- ラジオボタン（ＤＩＢセクションを介して描画）----------------------------------------------------//
237 : AJC_DLGPROC(Main, IDC_RBT_DIB_DC    )
238 : {
239 :     if (HIWORD(wParam) == BN_CLICKED) {
240 :         if (AjcGetDlgItemChk(hDlg, IDC_RBT_PRT_DC)) AjcPrnSetCallbackByPrinter(hPrn, cbDrawByPrinterDC);
241 :         else                                        AjcPrnSetCallbackByDibSect(hPrn, cbDrawByDibSectDC);
242 :     }
243 :     return TRUE;
244 : }
245 : //-------------------------------------------------------------------------------------------------------//
246 : AJC_DLGMAP_DEF(Main)
247 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG      )
248 :     AJC_DLGMAP_MSG(Main, WM_DESTROY         )
249 :     AJC_DLGMAP_MSG(Main, WM_COMMAND         )
250 :     AJC_DLGMAP_CMD(Main, IDCANCEL           )
251 :     AJC_DLGMAP_CMD(Main, IDC_CMD_SETPRN     )
252 :     AJC_DLGMAP_CMD(Main, IDC_CMD_START      )
253 :     AJC_DLGMAP_CMD(Main, IDC_CMD_STOP       )
254 :     AJC_DLGMAP_CMD(Main, IDC_CMD_FILE       )
255 :     AJC_DLGMAP_CMD(Main, IDC_RBT_PRT_DC     )
256 :     AJC_DLGMAP_CMD(Main, IDC_RBT_DIB_DC     )
257 : AJC_DLGMAP_END
258 :
259 : //-------------------------------------------------------------------------------------------------------//
260 : //  コールバック（ページ印刷開始）                                                                       //
261 : //                                                                                                       //
262 : //  引　数　:  hdc         - ＤＩＢセクションビットマップのＤＣ                                         //
263 : //             pPrnInfo    - プリンタ情報へのポインタ                                                   //
264 : //             pPgInfo     - ページ情報へのポインタ                                                     //
265 : //             cbp         - コールバックパラメタ                                                       //
266 : //                                                                                                       //
267 : //  戻り値　:  TRUE  - 印刷継続                                                                          //
268 : //             FALSE - 印刷終了（このページを印刷後に終了）                                              //
269 : //-------------------------------------------------------------------------------------------------------//
270 : static  BOOL CALLBACK cbQueryPage(PCAJCPRN_INFO pPrnInfo, PAJCPRN_PGINFO pPgInfo, UX cbp)
271 : {
272 :     BOOL    rc = FALSE;
273 :
274 :     AjcDoEvent();                                                   //  印刷中止ボタンチェック
275 :     rc = fPrintContinue;                                            //  ・
276 :
277 :     return !AjcFEof(hFile);
278 : }
```

```
279 : //-----------------------------------------------------------------------------------------------------------------//
280 : //  コールバック（ＤＩＢへ描画）                                                                                   //
281 : //                                                                                                                 //
282 : //  引　数　:  hdc         - ＤＩＢセクションビットマップのＤＣ                                                    //
283 : //             pPi - プリンタ情報へのポインタ                                                                      //
284 : //             pDi - ＤＩＢセクション情報へのポインタ                                                              //
285 : //             cbp         - コールバックパラメタ                                                                  //
286 : //                                                                                                                 //
287 : //  戻り値　:  TRUE  - 印刷継続                                                                                    //
288 : //             FALSE - 印刷終了（このページを印刷後に終了）                                                        //
289 : //-----------------------------------------------------------------------------------------------------------------//
290 : static  BOOL CALLBACK cbDrawByDibSectDC(HDC hdc, PCAJCPRN_INFO pPi, PCAJCDIBINFO pDi, UX cbp)
291 : {
292 :     BOOL        rc;
293 :     int         CxChar, CyChar;
294 :     int         y, i, nLine;
295 :     HFONT       hFontSv;
296 :     TEXTMETRIC  tm;
297 :     UT          buf[512];
298 :
299 :     AjcDoEvent();                                                       //  印刷中止ボタンチェック
300 :     rc = fPrintContinue;                                                //  ・
301 :
302 :     //  テキストイメージ描画
303 :     hFontSv = (HFONT)SelectObject(hdc, hFont);                          //  フォント設定
304 :     GetTextMetrics(hdc, &tm);                                           //  テキスト情報取得
305 :     CxChar = __max(1, tm.tmAveCharWidth);                               //  フォントサイズ設定
306 :     CyChar = __max(1, tm.tmHeight + tm.tmExternalLeading);              //  ・
307 :     nLine = pDi->height / CyChar;                                       //  印刷可能行数設定
308 :     for (i = 0, y = 0; i < nLine && AjcFGetS(hFile, buf, AJCTSIZE(buf)); i++) { //  印刷可能行数数ループ
309 :         MAjcStrTok(buf, TEXT("¥x0A"));                                  //          改行文字消去
310 :         TextOut(hdc, 0, y, buf, (int)MAjcStrLen(buf));                  //          テキスト描画
311 :         y += CyChar;                                                    //          ラスタ位置更新
312 :     }
313 :     SelectObject(hdc, hFontSv);
314 :
315 :     //  外枠描画
316 :     {
317 :         HBRUSH hBru = (HBRUSH)SelectObject(hdc, GetStockObject(NULL_BRUSH));
318 :         HPEN   hPen = (HPEN  )SelectObject(hdc, CreatePen(PS_SOLID, 5, RGB(0, 0, 0)));
319 :         Rectangle(hdc, 0, 0, pDi->width, pDi->height);
320 :         SelectObject(hdc, hBru);
321 :         DeleteObject(SelectObject(hdc, hPen));
322 :     }
323 :     return fPrintContinue;      //  FALSE:終了, TRUE：継続
324 : }
325 : //-----------------------------------------------------------------------------------------------------------------//
326 : //  コールバック（プリンタＤＣへ描画）                                                                             //
327 : //                                                                                                                 //
328 : //  引　数　:  hdc         - プリンタのＤＣ                                                                        //
329 : //             pPi         - プリンタ情報へのポインタ                                                          //
330 : //             cbp         - コールバックパラメタ                                                              //
331 : //                                                                                                                 //
332 : //  戻り値　:  TRUE  - 印刷継続                                                                                    //
333 : //             FALSE - 印刷終了（このページを印刷後に終了）                                                        //
334 : //-----------------------------------------------------------------------------------------------------------------//
335 : static  BOOL CALLBACK cbDrawByPrinterDC(HDC hdc, PCAJCPRN_INFO pPi, UX cbp)
336 : {
337 :     BOOL        rc = FALSE;
338 :     int         CxChar, CyChar;
339 :     int         y, i, nLine;
340 :     HFONT       hFontSv;
341 :     TEXTMETRIC  tm;
342 :     UT          buf[512];
343 :
344 :     AjcDoEvent();                                                       //  印刷中止ボタンチェック
345 :     rc = fPrintContinue;                                                //  ・
346 :
347 :     //  テキストイメージ描画
348 :     hFontSv = (HFONT)SelectObject(hdc, hFont);                          //  フォント設定
349 :     GetTextMetrics(hdc, &tm);                                           //  テキスト情報取得
350 :     CxChar = __max(1, tm.tmAveCharWidth);                               //  フォントサイズ設定
351 :     CyChar = __max(1, tm.tmHeight + tm.tmExternalLeading);              //  ・
352 :     nLine = pPi->cy / CyChar;                                           //  印刷可能行数設定
353 :     for (i = 0, y = pPi->y; i < nLine && AjcFGetS(hFile, buf, AJCTSIZE(buf)); i++) {    //  印刷可能行数数ループ
354 :         MAjcStrTok(buf, TEXT("¥x0A"));                                  //          改行文字消去
355 :         TextOut(hdc, pPi->x, y, buf, (int)MAjcStrLen(buf));             //          テキスト描画
356 :         y += CyChar;                                                    //          ラスタ位置更新
357 :     }
358 :     SelectObject(hdc, hFontSv);
```

```
359 :
360 :     //  外枠描画
361 :     {
362 :         HBRUSH hBru = (HBRUSH)SelectObject(hdc, GetStockObject(NULL_BRUSH));
363 :         HPEN   hPen = (HPEN  )SelectObject(hdc, CreatePen(PS_SOLID, 5, RGB(0, 0, 0)));
364 :         Rectangle(hdc, pPi->x, pPi->y, pPi->x + pPi->cx, pPi->y + pPi->cy);
365 :         SelectObject(hdc, hBru);
366 :         DeleteObject(SelectObject(hdc, hPen));
367 :     }
368 :     return fPrintContinue;      //  FALSE:終了, TRUE：継続
369 : }
```

52.6.2. SW_PrintImage（イメージファイルの印刷 ）

以下のサンプルプログラムは、選択した画像ファイル(*.bmp, *.jpg, *.png, *.gif)を印刷します。
画像ファイルを選択し、印刷開始ボタンを押すと、縦横共にページの半分の大きさの描画域に画像を印刷します。
また、ページ外枠と描画域を表す外枠も加えて印刷します。




```
 1 : //
 2 : //  SW_PrintImage.c
 3 : //
 4 :
 5 : #include    <AjrCstXX.h>
 6 : #include    <Shlwapi.h>
 7 : #include    <io.h>
 8 : #include    <tchar.h>
 9 : #include    <sys/stat.h>
10 : #include    "resource.h"
11 : #pragma comment(lib, "shlwapi.lib")
12 :
13 : #define PICL    400
14 :
15 : //-------------------------------------------------------------------------------------------------------------//
16 : //  ワーク                                                                                                     //
17 : //-------------------------------------------------------------------------------------------------------------//
18 : static  HINSTANCE       hInst;                    //  ＤＬＬインスタンスハンドル
19 : static  HWND            hDlgMain;                 //  ダイアログボックスハンドル
20 : static  HWND            hLbx;
21 :
22 : static  BOOL            fImgValid = FALSE;
23 : static  BOOL            fPrintContinue;
24 : static  HAJCPRN         hPrn;                     //  印刷インスタンス
25 : static  AJCPRN_INFO     PrnInfo;                  //  プリンタ情報
26 : static  AJC_IMGINFO     ImgInfo;                  //  イメージ情報
27 : static  HBITMAP         hBmpPic = NULL;           //  ピクチャビットマップハンドル
28 : static  UT              ImgFile[MAX_PATH];        //  ピクチャファイル
29 :
30 :
31 : //-------------------------------------------------------------------------------------------------------------//
32 : //  内部サブ関数                                                                                               //
33 : //-------------------------------------------------------------------------------------------------------------//
34 : AJC_DLGPROC_DEF(Main);
35 :
36 : static  VO      DrawToPic(VO);
37 : static  BOOL    SearchImageFile(VO);
38 :
39 : static  BOOL CALLBACK cbQueryPage      (PCAJCPRN_INFO pPrnInfo, PAJCPRN_PGINFO pPageInfo, UX cbp);
40 : static  BOOL CALLBACK cbDrawByDibSectDC(HDC hdc, PCAJCPRN_INFO pPrnInfo, PCAJCDIBINFO pDibInfo, UX cbp);
41 :
42 : //=============================================================================================================//
43 : //                                                                                                             //
44 : //  W i n M a i n                                                                                              //
45 : //                                                                                                             //
46 : //=============================================================================================================//
47 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
48 : {
49 :     MSG     msg;
```

```
 50 :
 51 :     hInst = hInstance;
 52 :     memset(&ImgInfo, 0, sizeof ImgInfo);
 53 :
 54 :     //----- メイン・ダイアログオープン -----------------//
 55 :     hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_MAIN), NULL, AJC_DLGPROC_NAME(Main));
 56 :     ShowWindow(hDlgMain, SW_SHOW);
 57 :
 58 :     //----- メッセージループ ---------------------------//
 59 :     while (GetMessage(&msg, NULL, 0, 0)) {
 60 :         do {
 61 :             if (IsDialogMessage(hDlgMain, &msg)) break;
 62 :             TranslateMessage(&msg);
 63 :             DispatchMessage (&msg);
 64 :         } while (0);
 65 :     }
 66 :
 67 :     return (int)msg.wParam ;
 68 : }
 69 : //=============================================================================================================//
 70 : //                                                                                                             //
 71 : //  ダイアログ・プロシージャ                                                                                   //
 72 : //                                                                                                             //
 73 : //=============================================================================================================//
 74 : //----- ダイアログ初期化 -----------------------------------------------------------------------------------//
 75 : AJC_DLGPROC(Main, WM_INITDIALOG     )
 76 : {
 77 :     hDlgMain = hDlg;
 78 :     hLbx     = GetDlgItem(hDlg, IDC_LBX_FILES);
 79 :
 80 :     SetWindowPos(GetDlgItem(hDlg, IDC_PIC_IMAGE), NULL, 0, 0, PICL, PICL, SWP_NOMOVE);
 81 :
 82 :     //----- 印刷インスタンス生成 -----------------------//
 83 :     hPrn = AjcPrnCreate();
 84 :     //----- ページ印刷開始通知コールバック設定 ---------//
 85 :     AjcPrnSetCallbackQueryPage(hPrn, cbQueryPage);
 86 :     //----- １ページ印刷用コールバック設定 -------------//
 87 :     AjcPrnSetCallbackByDibSect(hPrn, cbDrawByDibSectDC);
 88 :     //----- プリンタ情報読み出し -----------------------//
 89 :     AjcSetDlgItemStr(hDlg, IDC_LBL_PRNAME, AjcPrnGetInfo(hPrn, &PrnInfo));
 90 :     //----- ダイアログ初期化 ---------------------------//
 91 :     AjcSetDlgItemUInt(hDlg, IDC_INP_SPC_L, PrnInfo.MrgIf.mm.l);
 92 :     AjcSetDlgItemUInt(hDlg, IDC_INP_SPC_R, PrnInfo.MrgIf.mm.r);
 93 :     AjcSetDlgItemUInt(hDlg, IDC_INP_SPC_U, PrnInfo.MrgIf.mm.u);
 94 :     AjcSetDlgItemUInt(hDlg, IDC_INP_SPC_D, PrnInfo.MrgIf.mm.d);
 95 :     //----- 設定値ロード -------------------------------//
 96 :     AjcLoadAllControlSettings(hDlg, TEXT("Settings"), AJCCTL_SELACT_ALL | AJCCTL_SELACT_NTCALL);
 97 :     //----- 画像表示 -----------------------------------//
 98 :     SendMessage(hDlg, WM_COMMAND, MAKELONG(IDC_LBX_FILES, AJCLBXN_SELCHANGE), (LPARAM)hLbx);
 99 :     //----- 画像選択状態監視タイマ起動 -----------------//
100 :     SetTimer(hDlg, 1, 100, NULL);
101 :
102 :     return TRUE;
103 : }
104 : //----- ウインド破棄 ---------------------------------------------------------------------------------------//
105 : AJC_DLGPROC(Main, WM_DESTROY        )
106 : {
107 :     //----- タイマ停止 ---------------------------------//
108 :     KillTimer(hDlg, 1);
109 :     //----- 設定値セーブ -------------------------------//
110 :     AjcSaveAllControlSettings(hDlg);
111 :     //----- リソース解放 -------------------------------//
112 :     AjcPrnDelete(hPrn);
113 :     AjcImgFuncRelease(&ImgInfo);
114 :     //----- プログラム終了 -----------------------------//
115 :     PostQuitMessage(0);
116 :
117 :     return TRUE;
118 : }
119 : //----- WM_TIMER -------------------------------------------------------------------------------------------//
120 : AJC_DLGPROC(Main, WM_TIMER          )
121 : {
122 :     if (AjcLbxGetCurSel(hLbx) >= 0 && ImgInfo.fValid) {
123 :         ShowWindow(GetDlgItem(hDlg, IDC_PIC_IMAGE), SW_SHOW);
124 :         AjcEnableDlgItem(hDlg, IDC_CMD_START, TRUE);
125 :     }
126 :     else {
127 :         ShowWindow(GetDlgItem(hDlg, IDC_PIC_IMAGE), SW_HIDE);
128 :         AjcEnableDlgItem(hDlg, IDC_CMD_START, FALSE);
129 :     }
```

```
130 :     return TRUE;
131 : }
132 : //----- WM_COMMAND ------------------------------------------------------------------------------------//
133 : AJC_DLGPROC(Main, WM_COMMAND        )
134 : {
135 :     int     cmd = LOWORD(wParam);
136 :
137 :     if (cmd == IDC_INP_SPC_U || cmd == IDC_INP_SPC_D || cmd == IDC_INP_SPC_L || cmd == IDC_INP_SPC_R) {
138 :         if (HIWORD(wParam) == AJCIVN_INTVALUE) {
139 :             AJCPRN_MARGIN   mrg;
140 :             mrg.u = AjcGetDlgItemUInt(hDlg, IDC_INP_SPC_U);
141 :             mrg.d = AjcGetDlgItemUInt(hDlg, IDC_INP_SPC_D);
142 :             mrg.l = AjcGetDlgItemUInt(hDlg, IDC_INP_SPC_L);
143 :             mrg.r = AjcGetDlgItemUInt(hDlg, IDC_INP_SPC_R);
144 :             AjcPrnSetMargin(hPrn, &mrg);
145 :         }
146 :     }
147 :     return TRUE;
148 : }
149 : //----- ウインドクローズ ------------------------------------------------------------------------------//
150 : AJC_DLGPROC(Main, IDCANCEL          )
151 : {
152 :     DestroyWindow(hDlg);
153 :     return TRUE;
154 : }
155 : //----- プリンタの設定ボタン --------------------------------------------------------------------------//
156 : AJC_DLGPROC(Main, IDC_CMD_SETPRN    )
157 : {
158 :     if (AjcPrnSelectDlg(hPrn, &PrnInfo, 0, hDlg)) {
159 :         //  プリンタ名表示
160 :         AjcSetDlgItemStr(hDlg, IDC_LBL_PRNAME, AjcPrnGetInfo(hPrn, NULL));
161 :     }
162 :     return TRUE;
163 : }
164 : //----- 印刷開始ボタン --------------------------------------------------------------------------------//
165 : AJC_DLGPROC(Main, IDC_CMD_START )
166 : {
167 :     AJCPRN_OPT  opt = {0};
168 :
169 :     if (HIWORD(wParam) == BN_CLICKED) {
170 :         opt.Flags = (
171 :         //  AJCPRNF_DISABLECOPIES       |   //  「印刷部数」を無効化する
172 :         //  AJCPRNF_DISABLEPAGEINFO     |   //  「印刷範囲」を無効にする
173 :         //  AJCPRNF_ALLPAGES            |   //  「すべて」を選択状態にする
174 :         //  AJCPRNF_SELECTION           |   //  「選択した部分」を選択状態にする
175 :         //  AJCPRNF_NOSELECTION         |   //  「選択した部分」を無効状態にする
176 :         //  AJCPRNF_PAGENUMS            |   //  「ページ指定」を選択状態にする
177 :         //  AJCPRNF_NOPAGENUMS          |   //  「ページ指定」を無効状態にする
178 :         //  AJCPRNF_NOWARNING           |   //  既定のプリンタがない場合に警告メッセージ非表示
179 :         //  AJCPRNF_PRINTTOFILE         |   //  「ファイルへ出力」をチェックする
180 :         //  AJCPRNF_DISABLEPRINTTOFILE  |   //  「ファイルへ出力」を無効化する
181 :         //  AJCPRNF_HIDEPRINTTOFILE     |   //  ファイルへ出力」を非表示
182 :             AJCPRNF_USEDEVMODECOPIES    |   //  アプリで「部単位で印刷」が未サポートであることを指定
183 :             0);
184 :         opt.nCopies     = 1;                //  コピー部数
185 :         opt.nFromPage   = 1;                //  開始ページ
186 :         opt.nToPage     = 1;                //  終了ページ
187 :         opt.nMinPage    = 1;                //  ページ範囲の最小値
188 :         opt.nMaxPage    = 10;               //  ページ範囲の最小値
189 :         if (AjcPrnPrintDlg(hPrn, NULL, &opt, hDlg)) {
190 :             //--- プリンタ名表示 ------------//
191 :             AjcSetDlgItemStr(hDlg, IDC_LBL_PRNAME, AjcPrnGetInfo(hPrn, NULL));
192 :             //--- 印刷継続フラグ初期化 ------//
193 :             fPrintContinue = TRUE;
194 :             //--- 中止ボタンのみ有効化 ------//
195 :             AjcEnableGroup(GetDlgItem(hDlg, IDC_GRP_ALL ), FALSE, FALSE);
196 :             EnableWindow  (GetDlgItem(hDlg, IDC_CMD_STOP), TRUE);
197 :             //--- 印刷開始 ------------------//
198 :             AjcPrnStart (hPrn, (opt.Flags & AJCPRNF_OUTPUTTOFILE) != 0);
199 :             //--- グレー状態解除 ------------//
200 :             AjcEnableGroup(GetDlgItem(hDlg, IDC_GRP_ALL ), TRUE, TRUE);
201 :             EnableWindow  (GetDlgItem(hDlg, IDC_CMD_STOP), FALSE);
202 :         }
203 :     }
204 :     return TRUE;
205 : }
206 : //----- 印刷中止ボタン --------------------------------------------------------------------------------//
207 : AJC_DLGPROC(Main, IDC_CMD_STOP      )
208 : {
209 :     if (HIWORD(wParam) == BN_CLICKED) {
```

```
210 :         //--- グレー状態解除 -----------//
211 :         AjcEnableGroup(GetDlgItem(hDlg, IDC_GRP_ALL ), TRUE, TRUE);
212 :         EnableWindow  (GetDlgItem(hDlg, IDC_CMD_STOP), FALSE);
213 :         //--- 印刷終了の旨、設定 -------//
214 :         fPrintContinue = FALSE;
215 :     }
216 :     return TRUE;
217 : }
218 : //----- リストボックスからの通知 ----------------------------------------------------------------------------//
219 : AJC_DLGPROC(Main, IDC_LBX_FILES     )
220 : {
221 :     int             ix;
222 :     static UT SvFile[MAX_PATH];
223 :
224 :     if (HIWORD(wParam) == AJCLBXN_SELCHANGE) {
225 :         if ((ix = AjcLbxGetCurSel(hLbx)) >= 0) {                //  選択項目有り
226 :             AjcLbxGetText(hLbx, ix, ImgFile, MAX_PATH);         //  選択ファイルパス取得
227 :             if (MAjcStrCmp(ImgFile, SvFile) != 0) {             //      前回のファイルと異なる？
228 :                 AjcImgFuncRelease(&ImgInfo);                    //          前回読み出し済のイメージ解放
229 :                 if (AjcImgFuncRead(&ImgInfo, ImgFile)) {        //          イメージファイル読み出し，成功？
230 :                     DrawToPic();                                //              イメージ表示
231 :                 }
232 :             //  else MessageBox(hDlg, TEXT("イメージファイルの読み出しを失敗しました。"), TEXT("SW_PrintImage"), MB_ICONERROR);
233 :                 MAjcStrCpy(SvFile, MAX_PATH, ImgFile);          //          イメージファイルパス退避
234 :             }
235 :         }
236 :     }
237 :     return TRUE;
238 : }
239 : //-------------------------------------------------------------------------------------------------------------//
240 : AJC_DLGMAP_DEF(Main)
241 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG      )
242 :     AJC_DLGMAP_MSG(Main, WM_DESTROY         )
243 :     AJC_DLGMAP_MSG(Main, WM_TIMER           )
244 :     AJC_DLGMAP_MSG(Main, WM_COMMAND         )
245 :     AJC_DLGMAP_CMD(Main, IDCANCEL           )
246 :     AJC_DLGMAP_CMD(Main, IDC_CMD_SETPRN     )
247 :     AJC_DLGMAP_CMD(Main, IDC_CMD_START      )
248 :     AJC_DLGMAP_CMD(Main, IDC_CMD_STOP       )
249 :     AJC_DLGMAP_CMD(Main, IDC_LBX_FILES      )
250 : AJC_DLGMAP_END
251 :
252 : //-------------------------------------------------------------------------------------------------------------//
253 : //  ピクチャコントロールににイメージ描画                                                                       //
254 : //                                                                                                             //
255 : //  引　数　：　なし                                                                                           //
256 : //                                                                                                             //
257 : //  戻り値　：　なし                                                                                           //
258 : //-------------------------------------------------------------------------------------------------------------//
259 : static  VO      DrawToPic(VO)
260 : {
261 :     HDC     hdc, mdc;
262 :     HBITMAP hSvBmp;
263 :     HPEN    hSvPen;
264 :     HBRUSH  hSvBru;
265 :     int     pw, ph;
266 :     RECT    rcS, rcD;
267 :
268 :     if(ImgInfo.fValid) {
269 :         //  画像のアスペクトに合わせたピクチャサイズ設定
270 :         if (ImgInfo.height >= ImgInfo.width) {pw = PICL * ImgInfo.width / ImgInfo.height; ph = PICL;}   //  縦長
271 :         else                                 {pw = PICL; ph = PICL * ImgInfo.height / ImgInfo.width;}   //  横長
272 :         //  ＤＣ設定
273 :         hdc = GetDC(hDlgMain);
274 :         mdc = CreateCompatibleDC(hdc);
275 :         //  描画ビットマップオブジェクト作成
276 :         if (hBmpPic != NULL) DeleteObject(hBmpPic);
277 :         hBmpPic = CreateCompatibleBitmap(hdc, pw, ph);
278 :         hSvBmp  = (HBITMAP)SelectObject(mdc, hBmpPic);
279 :         //  用紙域をクリアー
280 :         hSvPen = (HPEN  )SelectObject(mdc, GetStockObject(NULL_PEN));
281 :         hSvBru = (HBRUSH)SelectObject(mdc, CreateSolidBrush(GetSysColor(COLOR_BTNFACE)));
282 :         Rectangle(mdc, 0, 0, pw + 1, ph + 1);
283 :         SelectObject(mdc, hSvPen);
284 :         DeleteObject(SelectObject(mdc, hSvBru));
285 :         //  メモリＤＣへイメージ描画
286 :         SetRect(&rcS, 0, 0, ImgInfo.width, ImgInfo.height);
287 :         SetRect(&rcD, 0, 0, pw, ph);
288 :         AjcImgFuncDraw(&ImgInfo, &rcS, mdc, &rcD);
289 :         //  作成したビットマップのハンドル設定
```

```
290 :         hBmpPic = (HBITMAP)SelectObject(mdc, hSvBmp);
291 :         //  ＤＣ解放
292 :         DeleteDC(mdc);
293 :         ReleaseDC(hDlgMain, hdc);
294 :         //  ピクチャ描画
295 :         SendDlgItemMessage(hDlgMain, IDC_PIC_IMAGE, STM_SETIMAGE, IMAGE_BITMAP, (LPARAM)hBmpPic);
296 :     }
297 : }
298 : //-------------------------------------------------------------------------------------------------------------//
299 : //  コールバック（ページ印刷開始）                                                                             //
300 : //                                                                                                             //
301 : //  引　数　:  pPrnInfo    - プリンタ情報                                                                      //
302 : //             pPgInfo     - ページ情報（余白サイズ，印刷の向き）へのポインタ                                  //
303 : //             cbp         - コールバックパラメタ                                                              //
304 : //                                                                                                             //
305 : //  戻り値　:  TRUE  - 印刷継続                                                                                //
306 : //             FALSE - 印刷終了（以降、何も印刷せずに終了する）                                                //
307 : //-------------------------------------------------------------------------------------------------------------//
308 : static  BOOL CALLBACK cbQueryPage(PCAJCPRN_INFO pPrnInfo, PAJCPRN_PGINFO pPgInfo, UX cbp)
309 : {
310 :     BOOL    rc = TRUE;
311 :
312 :     AjcDoEvent();               //  印刷中止ボタンチェック
313 :     rc = fPrintContinue;        //  ・
314 :
315 :     return rc;
316 : }
317 : //-------------------------------------------------------------------------------------------------------------//
318 : //  コールバック（ＤＩＢへ描画）                                                                               //
319 : //                                                                                                             //
320 : //  引　数　:  hdc          - ＤＩＢセクションビットマップのＤＣ                                               //
321 : //             pPrnInfo     - プリンタ情報へのポインタ                                                         //
322 : //             pDibInfo     - ＤＩＢセクション情報へのポインタ                                                 //
323 : //             cbp          - コールバックパラメタ                                                             //
324 : //                                                                                                             //
325 : //  戻り値　:  TRUE  - 印刷継続                                                                                //
326 : //             FALSE - 印刷終了（このページを印刷後に終了）                                                    //
327 : //-------------------------------------------------------------------------------------------------------------//
328 : static  BOOL CALLBACK cbDrawByDibSectDC(HDC hdc, PCAJCPRN_INFO pPrnInfo, PCAJCDIBINFO pDibInfo, UX cbp)
329 : {
330 :     BOOL    rc = FALSE;
331 :     RECT    rcS, rcR, rcB;
332 :
333 :     //  イメージ描画矩形設定
334 :     rcS.left   = 0;
335 :     rcS.top    = 0;
336 :     rcS.right  = ImgInfo.width;
337 :     rcS.bottom = ImgInfo.height;
338 :     //  イメージ描画矩形算出（ページ中央に 50%のサイズ）
339 :     rcR.left   = pDibInfo->width  / 4;
340 :     rcR.top    = pDibInfo->height / 4;
341 :     rcR.right  = rcR.left + pDibInfo->width  / 2;
342 :     rcR.bottom = rcR.top  + pDibInfo->height / 2;
343 :     //  アスペクトを維持してイメージ描画矩形算出
344 :     AjcAspGetZoomedRect(&rcS, &rcR, &rcB);
345 :     //  イメージ描画
346 :     AjcImgFuncDraw(&ImgInfo, &rcS, hdc, &rcB);
347 :     //  外枠描画
348 :     {
349 :         HBRUSH hBru = (HBRUSH)SelectObject(hdc, GetStockObject(NULL_BRUSH));
350 :         HPEN   hPen = (HPEN  )SelectObject(hdc, CreatePen(PS_SOLID, 5, RGB(0, 0, 0)));
351 :         //  ページ外枠
352 :         Rectangle(hdc, 0, 0, pDibInfo->width, pDibInfo->height);
353 :         //  イメージ域外枠
354 :         Rectangle(hdc, rcR.left, rcR.top, rcR.right, rcR.bottom);
355 :         SelectObject(hdc, hBru);
356 :         DeleteObject(SelectObject(hdc, hPen));
357 :     }
358 :     return FALSE;       //  FALSE:終了, TRUE：継続
359 : }
```

53. 高速フーリエ変換

高速フーリエ変換を行います。
与えるサンプリングデータは、２の整数乗（4, 8, 16, 32 ・・・1,073,741,824）個でなければなりません。

53.1. サポートＡＰＩ

高速フーリエ変換のサポートＡＰＩ一覧を以下に示します。

インスタンスを生成する場合

#	関 数 名	内容
1	AjcFftCreate	ＦＦＴ計算インスタンス生成
2	AjcFftCalc	高速フーリエ変換の計算
	AjcFftDelete	ＦＦＴ計算インスタンス消去

インスタンスを生成しない場合

#	関 数 名	内容
1	AjcFftMakeTable	ＦＦＴ計算用三角関数テーブル，ビット反転テーブル作成
2	AjcFftCalcByTbl	ＦＦＴ計算用（テーブル指定）

53.1.1. ＦＦＴ計算インスタンス生成(AjcFftCreate)

形　式 ： HAJCFFT　AjcFftCreate(int n)

引　数 ： n　－ サンプリングデータの個数（２の整数乗，4 ～ 0x40000000）

説　明 ： ＦＦＥ計算用のインスタンスを作成します。
内部的に三角関数表とビット反転表を作成します。

戻り値 ： ≠NULL - 成功（インスタンスハンドル）
＝NULL - 失敗

53.1.2. ＦＦＴ計算 (AjcFftCalc)

形　式 ： BOOL　AjcFftCreate(HAJCFFT hFft, PAJCCOMPLEX pData, BOOL fInverse);

引　数 ： hFft　　-　ＦＦＴインスタンスハンドル
pData　-　サンプリングデータ（配列）のアドレス
fInverse – FALSE：フーリエ変換を行う
- TRUE ： 逆フーリエ変換を行う

説　明 ： 高速フーリエ変換／逆フーリエ変換の計算を実行します。
pData で示すサンプリングデータの個数は、AjcCreate()で指定した「ｎ」と同じでなければなりません。

戻り値 ： TRUE　- 成功
FALSE - 失敗

53.1.3. ＦＦＴ計算インスタンス消去(AjcFftDelete)

形　式 ： BOOL　AjcFftCreate(HAJCFFT　hFft);

引　数 ： hFft –　ＦＦＴインスタンスハンドル

説　明 ： AjcFftCreate()で生成したＦＦＴインスタンスを解放します。

戻り値 ： TRUE　- 成功
FALSE - 失敗

53.1.4. ＦＦＴ計算用三角関数テーブル，ビット反転テーブル作成(AjcFftMakeTable)

形　式 ： BOOL AjcFftMakeTable(double *pSinTbl, int *pRevTbl, int n)；

引　数 ： pSinTbl - 三角関数表を作成するバッファ（配列）のアドレス
pRevTbl - ビット反転テーブルを作成するバッファ（配列）のアドレス
n　　- サンプリングデータ数（２の整数乗，4 ～ 0x40000000）

説　明 ： ＦＦＴ計算用の三角関数テーブルと、ビット反転テーブルを作成します。
各テーブル（配列）の要素数は以下の通りです。

#	種別	必要なテーブル（配列）の要素数
1	三角関数テーブル	n
2	ビット反転テーブル	n ＋ n ／ 4

戻り値 ： TRUE - 成功
FALSE - 失敗

53.1.5. ＦＦＴ計算（テーブル指定）(AjcFftCalcByTbl)

形　式 ： BOOL AjcFftCalcByTbl(PAJCCOMPLEX pData, int n, const double *pSinTbl, const int *pRevTbl, BOOL fInverse)；

引　数 ： pData - サンプリングデータ（配列）のアドレス
n　　- サンプリングデータ数（２の整数乗，4 ～ 0x40000000）
pSinTbl - 三角関数表を作成するバッファ（配列）のアドレス
pRevTbl - ビット反転テーブルを作成するバッファ（配列）のアドレス
fInverse – FALSE：フーリエ変換を行う
- TRUE ： 逆フーリエ変換を行う

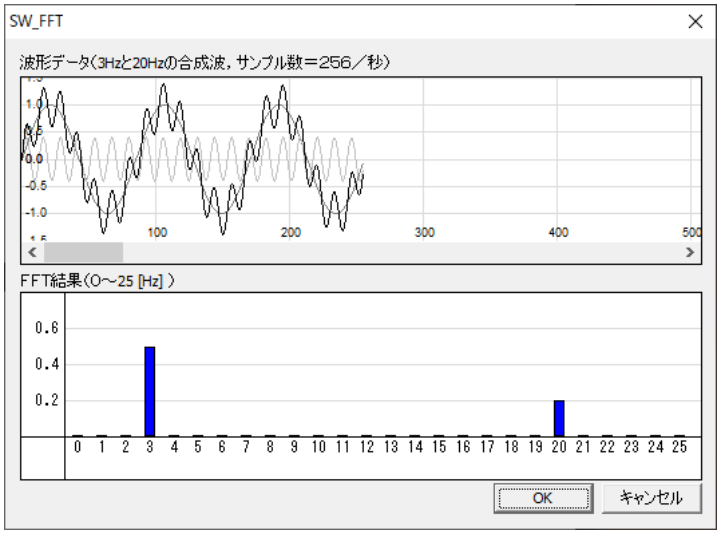
説　明 ： 高速フーリエ変換／逆フーリエ変換の計算を実行します。
pSinTbl と pRevTbl は、AjcFftMakeTable()で作成した三角関数表とビット反転テーブルを指定します。
n（サンプリングデータ数）は、AjcFftMakeTable()で指定した値と同じでなければなりません。

戻り値 ： TRUE - 成功
FALSE - 失敗

53.2. サンプルプログラム

53.2.1. SW_FFT（高速フーリエ変換の演算とグラフ表示）

以下のサンプルプログラムは、３Ｈｚと２０Ｈｚの正弦波を混合した波形をサンプリングデータ（２５６個／秒）として入力し、高速フーリエ変換した結果を表示します。




```
 1 : //
 2 : //  SW_FFT.c
 3 : //
 4 : #include    <AjrCstXX.h>
 5 : #include    <math.h>
 6 : #include    <tchar.h>
 7 : #include    "resource.h"
 8 :
 9 : #define     N       256
10 :
11 : //-------------------------------------------------------------------------------------------------------------//
12 : //  ワーク                                                                                                     //
13 : //-------------------------------------------------------------------------------------------------------------//
14 : HINSTANCE       hInst;
15 : HWND            hDlgMain;
16 :
17 : //-------------------------------------------------------------------------------------------------------------//
18 : //  内部サブ関数                                                                                               //
19 : //-------------------------------------------------------------------------------------------------------------//
20 : AJC_DLGPROC_DEF(Main);
21 :
22 : //=============================================================================================================//
23 : //                                                                                                             //
24 : //  W i n M a i n                                                                                              //
25 : //                                                                                                             //
26 : //=============================================================================================================//
27 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
28 : {
29 :     MSG     msg;
30 :
31 :     hInst = hInstance;
32 :
33 :     //----- メイン・ダイアログオープン -----------------//
34 :     hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_MAIN), NULL, AJC_DLGPROC_NAME(Main));
35 :     ShowWindow(hDlgMain, SW_SHOW);
36 :
37 :     //----- メッセージループ ---------------------------//
38 :     while (GetMessage(&msg, NULL, 0, 0)) {
39 :         do {
40 :             if (IsDialogMessage(hDlgMain, &msg)) break;
41 :             TranslateMessage(&msg);
42 :             DispatchMessage (&msg);
43 :         } while (0);
44 :     }
45 :
```

```
46 :     return (int)msg.wParam ;
47 : }
48 : //=============================================================================================================//
49 : //                                                                                                             //
50 : //  ダイアログ・プロシージャ                                                                                   //
51 : //                                                                                                             //
52 : //=============================================================================================================//
53 : //----- ダイアログ初期化 ------------------------------------------------------------------------------------//
54 : AJC_DLGPROC(Main, WM_INITDIALOG )
55 : {
56 :     hDlgMain = hDlg;
57 :
58 :     return TRUE;
59 : }
60 : //----- ウインド破棄 ----------------------------------------------------------------------------------------//
61 : AJC_DLGPROC(Main, WM_DESTROY    )
62 : {
63 :     PostQuitMessage(0);
64 :     return TRUE;
65 : }
66 : //----- ウインドクローズ ------------------------------------------------------------------------------------//
67 : AJC_DLGPROC(Main, IDCANCEL      )
68 : {
69 :     DestroyWindow(hDlg);
70 :     return TRUE;
71 : }
72 : //----- ＯＫ ------------------------------------------------------------------------------------------------//
73 : AJC_DLGPROC(Main, IDOK          )
74 : {
75 :     HWND        hCtrl;
76 :     AJCTCPROP   prop;
77 :     HAJCFFT     hFft;
78 :     int         i;
79 :     double      t, step;
80 :     double      wav[N][3];
81 :     AJCCOMPLEX  dat[N];
82 :     double      fft[1];
83 :
84 :     //  タイムチャート表示色設定
85 :     hCtrl = GetDlgItem(hDlg, IDC_TMC);
86 :     AjcTchGetProp(hCtrl, &prop);
87 :     prop.Item[0].rgb = RGB(128, 128, 128);
88 :     prop.Item[1].rgb = RGB(192, 192, 192);
89 :     prop.Item[2].rgb = RGB(  0,   0,   0);
90 :     AjcTchSetProp(hCtrl, &prop);
91 :     //  3Hz の波形データ作成
92 :     step = (AJC_PAI * 2.0) / 256.0 * 3.0;
93 :     for (i = 0, t = 0; i < N; i++, t += step) wav[i][0] = sin(t);
94 :     //  20Hz の波形データ作成
95 :     step = (AJC_PAI * 2.0) / 256.0 * 20.0;
96 :     for (i = 0, t = 0; i < N; i++, t += step) wav[i][1] = sin(t) * 0.4;
97 :     //  合成波形作成
98 :     for (i = 0; i < N; i++) wav[i][2] = wav[i][0] + wav[i][1];
99 :     //  波形表示
100 :    AjcTchPurge(hCtrl);
101 :    for (i = 0; i < N; i++) AjcTchPutRealData(hCtrl, wav[i]);
102 :
103 :    //  FFT 解析データ作成
104 :    memset(dat, 0, sizeof dat);
105 :    for (i = 0; i < N; i++) dat[i].re = wav[i][2];
106 :    //  FFT 実行
107 :    hFft = AjcFftCreate(N);
108 :    AjcFftCalc(hFft, dat, FALSE);
109 :    AjcFftDelete(hFft);
110 :    //  振幅データ表示
111 :    hCtrl = GetDlgItem(hDlg, IDC_FFT);
112 :    AjcBarPurge(hCtrl);
113 :    for (i = 0; i < 26; i++) {
114 :        UT  txt[16];
115 :        fft[0] = sqrt(pow(dat[i].re, 2) + pow(dat[i].im, 2));
116 :        AjcSnPrintF(txt, AJCTSIZE(txt), TEXT("%d"), i);
117 :        AjcBarPutRealData(hCtrl, fft, txt);
118 :    }
119 :    //  注釈クリアー
120 :    AjcSetDlgItemStr(hDlg, IDC_LBL_OK, TEXT(""));
121 :
122 :    return TRUE;
123 : }
124 : //-------------------------------------------------------------------------------------------------------------//
125 : AJC_DLGMAP_DEF(Main)
```

```
126 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG  )
127 :     AJC_DLGMAP_MSG(Main, WM_DESTROY     )
128 :
129 :     AJC_DLGMAP_CMD(Main, IDCANCEL       )
130 :     AJC_DLGMAP_CMD(Main, IDOK           )
131 : AJC_DLGMAP_END
132 :
```

54. モニタ情報

モニタ情報に関するＡＰＩ群です。

54.1. サポートＡＰＩ

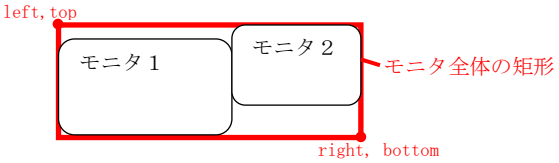
モニタ情報サポートＡＰＩ一覧を以下に示します。

#	関 数 名	内容
1	AjcGetMonitorsRect	マルチモニタ全体の矩形情報情報の取得
2	AjcGetMonitorsInfo	各モニタ矩形情報情報の取得
3	AjcGetPrimaryMonitorInfo	プライマリモニタ矩形情報の取得
4	AjcGetMonitorInfoOfPoint	ポイントが属するモニタ矩形情報の取得
5	AjcGetMonitorInfoOfWindow	ウインドが属するモニタ矩形情報の取得
6	AjcGetWndInfoToFitOnMonitor	ウインドが属するモニタ内に収まるようなウインド情報取得

54.1.1. マルチモニタ全体の矩形情報情報の取得（AjcGetMonitorsRect）

形　式　：　UI　AjcGetMonitorsRect(LPRECT pRect);

引　数　：　pRect -マルチモニタ全体の矩形情報情報を格納するバッファのアドレス（不要時は NULL）



説　明　：　マルチモニタ全体の矩形情報を取得します。
pRect のメンバには、以下の値が設定されます。
left , top　　- 全モニタの矩形情報での左上
right, bottom – 全モニタの矩形情報での右下

戻り値　：　モニタの個数

54.1.2. 各モニタ矩形情報情報の取得（AjcGetMonitorsInfo）

形　式　：　UI　AjcGetMonitorsInfo(PAJCMONITORSINFO pMonInfo);

引　数　：　pMonInfo - 各モニタの矩形情報を格納するバッファのアドレス（不要時は NULL）

説　明　：　各モニタの矩形情報と、各モニタのワークエリア(タスクバーを除く部分)の矩形情報を取得します。
pMonInfo で示すバッファの形式は以下のとおりです。

```
#define     AJCMAX_MONITORS     32

typedef struct {
    int     nMon;                               //  モニタの個数
    RECT    rcMon [AJCMAX_MONITORS];            //  各モニタの矩形情報
    RECT    rcWork[AJCMAX_MONITORS];            //  各モニタ中ワークアリアの矩形情報
} AJCMONITORSINFO, *PAJCMONITORSINFO;
typedef const AJCMONITORSINFO *PCAJCMONITORSINFO;
```

戻り値　：　モニタの個数

54.1.3. プライマリモニタ矩形情報の取得（AjcGetPrimaryMonitorInfo）

形　式 ： BOOL AjcGetPrimaryMonitorInfo(LPRECT pRcMon, LPRECT pRcWork);

引　数 ： pRcMon - モニタ矩形情報を格納するバッファのアドレス （不要時は NULL）
pRcWork - ワークエリアの矩形情報を格納するバッファのアドレス（不要時は NULL）

説　明 ： プライマリモニタの矩形情報を取得します。

戻り値 ： TRUE - 成功
FALSE - 失敗

54.1.4. ポイントが属するモニタ矩形情報の取得（AjcGetMonitorInfoOfPoint）

形　式 ： BOOL AjcGetMonitorInfoOfPoint(int x, int y, LPRECT pRcMon, LPRECT pRcWork);

引　数 ： x, y - 画面の位置を示すポイント情報
pRcMon - モニタ矩形情報を格納するバッファのアドレス （不要時は NULL）
pRcWork - ワークエリアの矩形情報を格納するバッファのアドレス（不要時は NULL）

説　明 ： 指定したポイントを含む、モニタの矩形情報を取得します。
マルチモニタの場合、当該ポイントを含むモニタの矩形情報（スクリーン座標）を取得します。
ポイントがどのモニタにも属さない場合は FALSE を返し、プライマリモニタの矩形情報を設定します。

戻り値 ： TRUE - 成功
FALSE - 失敗

54.1.5. ウインドが属するモニタ矩形情報の取得（AjcGetMonitorInfoOfWindow）

形　式 ： BOOL AjcGetMonitorInfoOfWindow(HWND hwnd, LPRECT pRcMon, LPRECT pRcWork);

引　数 ： hwnd - ウインドハンドル
pRcMon - モニタ矩形情報を格納するバッファのアドレス （不要時は NULL）
pRcWork - ワークエリアの矩形情報を格納するバッファのアドレス（不要時は NULL）

説　明 ： 指定したウインドが属する、モニタの矩形情報を取得します。
ウインドがどのモニタにも属さない場合は FALSE を返し、プライマリモニタの矩形情報を設定します。

戻り値 ： TRUE - 成功
FALSE - 失敗

54.1.6. ウインドが属するモニタ内に収まるようなウインド情報取得（AjcGetWndInfoToFitOnMonitor）

形　式 ： BOOL AjcGetWndInfoToFitOnMonitor(HWND hwnd, LPPOINT pPoint, LPSIZE pSize, UIP pFlag);

引　数 ： hwnd　- ウインドハンドル
pPoint - ウインド移動先の位置を格納するバッファのアドレス（不要時は NULL）
pSize - ウインドの変更サイズを格納するバッファのアドレス（不要時は NULL）
pFlag - ウインド変更オプション（不要時は NULL）

説　明 ： ウインドを、ウインドが属するモニタ（ワークエリア）内に収まるようにウインド位置，サイズを取得します。
ウインドがどのモニタにも属さない場合は、プライマリモニタの内に収まるようにウインド位置，サイズを設定します。
ウインド左端が、モニタ左端を超えている場合は、左方向へ移動するようにウインド位置を設定します。
ウインド下端が、モニタ下端を超えている場合は、下方向へ移動するようにウインド位置を設定します。
ウインドサイズがモニタ内に収まらない場合は、ウインドがモニタに収まるようにウインドサイズを設定します。
戻り値が TRUE の場合、pFlag が示すバッファには、以下のシンボルの合成値が設定されます。

#	シンボル	内容
1	SWP_NOSIZE	ウインドのサイズは変更不要
2	SWP_NOMOVE	ウインドの位置は変更不要

*pFlag に設定された値は、そのままシステムＡＰＩの SetWindowPos()に使用できます。

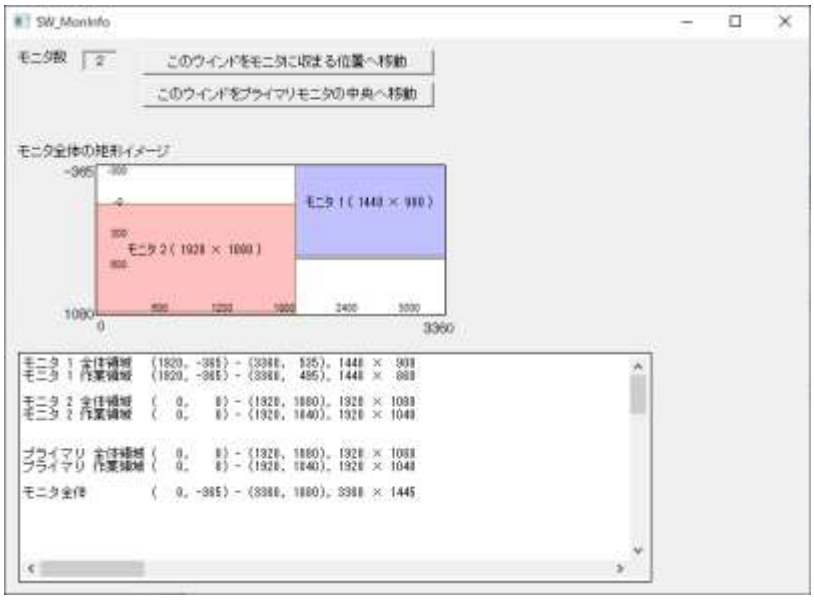
```
ex.
     POINT pt;
     SIZE  sz;
     UI    flag;
     if (AjcGetWindowRectToFitOnMonitor(hwnd, &pt, &sz, &flag)) {
          SetWindowPos(hwnd, NULL, pt.x, pt.y, sz.cx, sz.cy. flag);
     }
```

戻り値 ： TRUE - ウインドがモニタを超えている（ウインドの位置／サイズを変更する必要があることを意味する）
FALSE - ウインドはモニタ内に収まっている（ウインドの位置／サイズを変更する必要がない）

54.2. サンプルプログラム

54.2.1. SW_MonInfo（モニタ情報表示）

以下のようにモニタ情報を表示します。



「このウインドが収まる位置へ移動」ボタンは、このウインドがモニタ境界をまたいでいる場合、ウインドがモニタ内に収まる位置へ移動します。

「このウインドをプライマリモニタの中央へ移動」ボタンを押すと、このウインドをプライマリモニタの中央へ移動します。

```
 1 : //
 2 : //  SW_MonInfo.c
 3 : //
 4 :
 5 : #include    <AjrCstXX.h>
 6 : #include    <math.h>
 7 : #include    <tchar.h>
 8 : #include    "resource.h"
 9 :
10 : //------------------------------------------------------------------------------------------------------------------//
11 : //  ワーク                                                                                                          //
12 : //------------------------------------------------------------------------------------------------------------------//
13 : static  HINSTANCE       hInst;                      //  ＤＬＬインスタンスハンドル
14 : static  HWND            hDlgMain;                   //  ダイアログボックスハンドル
15 : static  HWND            hG2d;
16 : static  HWND            hVth;
17 : //------------------------------------------------------------------------------------------------------------------//
18 : //  内部サブ関数                                                                                                    //
19 : //------------------------------------------------------------------------------------------------------------------//
20 : AJC_DLGPROC_DEF(Main);
21 :
22 : //==================================================================================================================//
23 : //                                                                                                                  //
24 : //  ＷｉｎＭａｉｎ                                                                                                 //
25 : //                                                                                                                  //
26 : //==================================================================================================================//
27 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
28 : {
29 :     MSG     msg;
30 :
31 :     hInst = hInstance;
32 :
33 :     //----- メイン・ダイアログオープン ----------------//
34 :     hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_DLGMAIN), NULL, AJC_DLGPROC_NAME(Main));
35 :     //----- ダイアログ表示 ----------------------------//
36 :     ShowWindow(hDlgMain, SW_SHOW);
37 :
38 :     //----- メッセージループ --------------------------//
39 :     while (GetMessage(&msg, NULL, 0, 0)) {
40 :         do {
41 :             if (IsDialogMessage(hDlgMain, &msg)) break;
42 :             TranslateMessage(&msg);
```

```
43 :             DispatchMessage (&msg);
44 :         } while (0);
45 :     }
46 :
47 :     return (int)msg.wParam ;
48 : }
49 : //=============================================================================================================//
50 : //                                                                                                             //
51 : //  ダイアログ・プロシージャ                                                                                   //
52 : //                                                                                                             //
53 : //=============================================================================================================//
54 : //----- ダイアログ初期化 ------------------------------------------------------------------------------------//
55 : AJC_DLGPROC(Main, WM_INITDIALOG     )
56 : {
57 :     UI      i, nMon;
58 :     RECT    rcMon, rcG2d, rLab, rG2d;
59 :     RECT    rcM, rcW;
60 :     double  cxMon, cyMon, cxG2d, cyG2d;
61 :     double  fact;
62 :     double  cx, cy;
63 :     AJCMONITORSINFO mi;
64 :     HWND        hLab, hG2d;
65 :     AJC3DGPROP prop;
66 :
67 :     hDlgMain = hDlg;
68 :     hG2d = GetDlgItem(hDlg, IDC_G2D);
69 :     hVth = GetDlgItem(hDlg, IDC_VTH);
70 :
71 :     //  パレットを明るい色に変更
72 :     AjcG2dGetProp(hG2d, &prop);
73 :     prop.Item[ 0].rgbP = RGB(192, 192, 255);    prop.Item[ 4].rgbP = RGB(255, 192, 255);
74 :     prop.Item[ 1].rgbP = RGB(255, 192, 192);    prop.Item[ 5].rgbP = RGB(192, 192, 192);
75 :     prop.Item[ 2].rgbP = RGB(192, 224, 192);    prop.Item[ 6].rgbP = RGB(192, 192, 192);
76 :     prop.Item[ 3].rgbP = RGB(192, 224, 224);    prop.Item[ 7].rgbP = RGB(218, 218, 218);
77 :     AjcG2dSetProp(hG2d, &prop);
78 :     //  モニタ全体の矩形情報取得
79 :     nMon = AjcGetMonitorsRect(&rcMon);
80 :     cxMon = rcMon.right - rcMon.left;
81 :     cyMon = rcMon.bottom - rcMon.top;
82 :     //  Ｇ２Ｄウインド矩形情報取得
83 :     GetWindowRect(hG2d, &rcG2d);
84 :     cxG2d = rcG2d.right - rcG2d.left;
85 :     cyG2d = rcG2d.bottom - rcG2d.top;
86 :     //  モニタ数表示
87 :     AjcSetDlgItemUInt(hDlg, IDC_TXT_COUNT, nMon);
88 :     //  レンジ表示
89 :     AjcSetDlgItemSInt(hDlg, IDC_LBL_TOP   , rcMon.top);
90 :     AjcSetDlgItemSInt(hDlg, IDC_LBL_BOTTOM, rcMon.bottom);
91 :     AjcSetDlgItemSInt(hDlg, IDC_LBL_LEFT  , rcMon.left);
92 :     AjcSetDlgItemSInt(hDlg, IDC_LBL_RIGHT , rcMon.right);
93 :     //  ２Ｄグラフサイズ設定
94 :     fact = cxMon / cyMon;
95 :     cx = cyMon * fact;
96 :     cy = cyMon;
97 :     //  アスペクト比に合わせて２Ｄグラフウインドサイズ設定
98 :     fact = cyG2d / cyMon;
99 :     cx *= fact;
100 :    cy *= fact;
101 :    SetWindowPos(hG2d, NULL, 0, 0, (int)cx, (int)cy, SWP_NOMOVE);
102 :    //  右端スケール値移動
103 :    hLab = GetDlgItem(hDlg, IDC_LBL_RIGHT);             hG2d = GetDlgItem(hDlg, IDC_G2D);
104 :    GetWindowRect(hLab, &rLab);                         GetWindowRect(hG2d, &rG2d);
105 :    MapWindowPoints(NULL, hDlg, (LPPOINT)&rLab, 2);     MapWindowPoints(NULL, hDlg, (LPPOINT)&rG2d, 2);
106 :    rLab.left = rG2d.left + (int)cx - 20;
107 :    SetWindowPos(hLab, NULL, rLab.left, rLab.top, 0, 0, SWP_NOSIZE);
108 :    //  ２Ｄグラフモード設定
109 :    AjcG2dInit(hG2d, rcMon.left , rcMon.bottom,
110 :                     rcMon.right, rcMon.top   ,
111 :                     AJC3DGS_2DMODE & ~AJC3DGS_SCALELINE);
112 :    //  平面アングル設定（Ｘ方向＝Ｘ軸昇順，Ｙ方向＝Ｙ軸降順）
113 :    AjcG2dSetPlane(hG2d, AJCG2DAXIS_DIR_XP, AJCG2DAXIS_DIR_YM);
114 :    //  各モニタの矩形描画
115 :    AjcGetMonitorsInfo(&mi);
116 :    for (i = 0; i < nMon; i++) {
117 :        AJC2DVEC v;
118 :        int       w, h;
119 :        AjcG2dRectangle(hG2d, i, mi.rcMon[i].left, mi.rcMon[i].top, mi.rcMon[i].right, mi.rcMon[i].bottom);
120 :        w = (mi.rcMon[i].right  - mi.rcMon[i].left);
121 :        h = (mi.rcMon[i].bottom - mi.rcMon[i].top );
122 :        v.x = mi.rcMon[i].left + w / 2;
```

```
123 :         v.y = mi.rcMon[i].top  + h / 2;
124 :         AjcG2dFillSV(hG2d, i, &v);
125 :         AjcG2dRectangle(hG2d, 15, mi.rcMon[i].left , mi.rcMon[i].top , mi.rcMon[i].right , mi.rcMon[i].bottom );
126 :         AjcG2dRectangle(hG2d, 15, mi.rcWork[i].left, mi.rcWork[i].top, mi.rcWork[i].right, mi.rcWork[i].bottom);
127 :         AjcG2dPrintFV(hG2d,  AJCG2DTXOMD_ABOVE_CENTER, &v, TEXT("モニタ %d ( %d × %d )"), i + 1, w, h);
128 :
129 :         AjcVthPrintF(hVth, TEXT("モニタ %d 全体領域   (%4d, %4d) - (%4d, %4d), %4d × %4d¥n"), i + 1,
130 :                                     mi.rcMon[i].left, mi.rcMon[i].top, mi.rcMon[i].right, mi.rcMon[i].bottom,
131 :                                     mi.rcMon[i].right - mi.rcMon[i].left,  mi.rcMon[i].bottom - mi.rcMon[i].top);
132 :         AjcVthPrintF(hVth, TEXT("モニタ %d 作業領域   (%4d, %4d) - (%4d, %4d), %4d × %4d¥n"), i + 1,
133 :                                     mi.rcWork[i].left, mi.rcWork[i].top, mi.rcWork[i].right, mi.rcWork[i].bottom,
134 :                                     mi.rcWork[i].right - mi.rcWork[i].left,  mi.rcWork[i].bottom - mi.rcWork[i].top);
135 :         AjcVthPrintF(hVth, TEXT("¥n"));
136 :     }
137 :     AjcVthPrintF(hVth, TEXT("¥n"));
138 :     AjcGetPrimaryMonitorInfo(&rcM, &rcW);
139 :     AjcVthPrintF(hVth, TEXT("プライマリ 全体領域 (%4d, %4d) - (%4d, %4d), %4d × %4d¥n"),
140 :                                 rcM.left, rcM.top, rcM.right, rcM.bottom,
141 :                                 rcM.right - rcM.left,  rcM.bottom - rcM.top);
142 :     AjcVthPrintF(hVth, TEXT("プライマリ 作業領域 (%4d, %4d) - (%4d, %4d), %4d × %4d¥n"),
143 :                                 rcW.left, rcW.top, rcW.right, rcW.bottom,
144 :                                 rcW.right - rcW.left, rcW.bottom - rcW.top);
145 :
146 :     AjcVthPrintF(hVth, TEXT("¥n"));
147 :     AjcGetMonitorsRect(&rcM);
148 :     AjcVthPrintF(hVth, TEXT("モニタ全体          (%4d, %4d) - (%4d, %4d), %4d × %4d¥n"),
149 :                                 rcM.left, rcM.top, rcM.right, rcM.bottom,
150 :                                 rcM.right - rcM.left,  rcM.bottom - rcM.top);
151 :
152 :     return TRUE;
153 : }
154 : //----- ウインド破棄 ----------------------------------------------------------------------------------------------//
155 : AJC_DLGPROC(Main, WM_DESTROY        )
156 : {
157 :     //----- プログラム終了 ------------------------------------//
158 :     PostQuitMessage(0);
159 :     return TRUE;
160 : }
161 : //-----  「このウインドをモニタに収まる位置へ移動」ボタン ----------------------------------------------------//
162 : AJC_DLGPROC(Main, IDC_CMD_MOVE      )
163 : {
164 :     POINT   pt;
165 :     SIZE    sz;
166 :     UI      flag;
167 :     AjcGetWndInfoToFitOnMonitor(hDlg, &pt, &sz, &flag);
168 :     SetWindowPos(hDlg, NULL, pt.x, pt.y, sz.cx, sz.cy, flag);
169 :     return TRUE;
170 : }
171 : //-----  「このウインドをプライマリモニタの中央へ移動」ボタン ------------------------------------------------//
172 : AJC_DLGPROC(Main, IDC_CMD_MOVEP     )
173 : {
174 :     int     cx, cy, cxM, cyM;
175 :     RECT    rc, RcMon, RcWork;
176 :     GetWindowRect(hDlg, &rc);
177 :     cx = rc.right - rc.left;
178 :     cy = rc.bottom - rc.top;
179 :     AjcGetPrimaryMonitorInfo(&RcMon, &RcWork);
180 :     cxM = RcWork.right - RcWork.left;
181 :     cyM = RcWork.bottom - RcWork.top;
182 :     SetWindowPos(hDlg, NULL, RcWork.left + (cxM - cx) / 2, RcWork.top + (cyM - cy) / 2, 0, 0, SWP_NOSIZE);
183 :     return TRUE;
184 : }
185 : //-----  「Cancel」ボタン ------------------------------------------------------------------------------------//
186 : AJC_DLGPROC(Main, IDCANCEL          )
187 : {
188 :     DestroyWindow(hDlg);
189 :     return TRUE;
190 : }
191 : //-----------------------------------------------------------------------------------------------------------------//
192 : AJC_DLGMAP_DEF(Main)
193 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG     )
194 :     AJC_DLGMAP_MSG(Main, WM_DESTROY        )
195 :
196 :     AJC_DLGMAP_CMD(Main, IDC_CMD_MOVE      )
197 :     AJC_DLGMAP_CMD(Main, IDC_CMD_MOVEP     )
198 :     AJC_DLGMAP_CMD(Main, IDCANCEL          )
199 : AJC_DLGMAP_END
200 :
```

55. ヒープソート

配列のソートを行うＡＰＩです。
ソートを途中で中止することもできます。

55.1. サポートＡＰＩ

ヒープソートのサポートＡＰＩ一覧を以下に示します。

#	関 数 名	内容
1	AjcHeapSort	配列のソート

55.1.1. 配列のソート（AjcHeapSort）

形　式 ： BOOL AjcHeapSort(VOP pBase, UI num, UI width, UX cbp,
int (CALLBACK *cbComp)(C_VOP pElm1, C_VOP pElm2, BOOL *pStop, UX cbp));

引　数 ： pBase - ソートする配列の先頭アドレス
num - ソートする配列の要素数
width - ソートする配列の要素サイズ（バイト数）
cbp - コールバックパラメタ
cbComp - 要素比較用コールバック関数

説　明 ： 配列をソートします。
コールバック関数で「*pSrtop = TRUE;」を設定することにより、ソートを中止できます。

戻り値 ： TRUE - 成功
FALSE - 中止／エラー

コールバック：

cbComp（要素比較用コールバック関数）

形　式 ： BOOL CALLBACK *cbComp*(C_VOP pElm1, C_VOP pElm2, BOOL *pStop, UX cbp)

引　数 ： pElm1 - 比較する配列要素１のアドレス
pElm2 - 比較する配列要素２のアドレス
pStop - ソート中止フラグへのポインタ
cbp - コールバックパラメタ

説　明 ： 配列の要素を比較し、比較結果を返します。
ソートを中止する場合は、pStop で示されるフラグに TRUE を設定します。(*pStop = TRUE;)

戻り値 ： 配列要素の比較結果
＞ 0 - 配列要素１ ＞ 配列要素２
＝ 0 - 配列要素１ ＝ 配列要素２
＜ 0 - 配列要素１ ＜ 配列要素２

56. 演算

算術演算のＡＰＩ群です。

56.1. サポートＡＰＩ

算術演算のＡＰＩ一覧を以下に示します。

#	ＡＰＩ名	内　容	備考
1	AjcSin	正弦　　(サイン)	角度を度(degree)単位で指定する以外は、Ｃ言語の同名の関数と同じ。
2	AjcSinh	双曲線・正弦　(ハイパーボリック・サイン)	
3	AjcASin	逆正弦　　(アークサイン)	
4	AjcCos	余弦　　(コサイン)	
5	AjcCosh	双曲線・余弦　(ハイパーボリック・コサイン)	
6	AjcACos	逆余弦　　(アークコサイン)	
7	AjcTan	正接　　(タンジェント)	
8	AjcTanh	双曲線・正接　(ハイパーボリック・タンジェント)	
9	AjcATan	逆正接　　(アークタンジェント)	
10	AjcATan2	逆正接　　(アークタンジェント)	
11	AjcV3dAdd	３Ｄ・ベクトル加算	３Ｄベクトル演算
12	AjcV3dSub	３Ｄ・ベクトル減算	
13	AjcV3dMult	３Ｄ・ベクトル乗算（ベクトルをｎ倍する）	
14	AjcV3dDiv	３Ｄ・ベクトルの除算	
15	AjcV3dSetLinePoint	３Ｄ・線ベクトル設定	
16	AjcV3dSetLinePoint	３Ｄ・線分情報設定	
17	AjcV3dSetTriPoint	３Ｄ・三角形情報設定	
18	AjcV3dLength	３Ｄ・ベクトルの長さ算出	
19	AjcV3dLineCenter	３Ｄ・線分の中点算出	
20	AjcV3dOuter	３Ｄ・ベクトルの外積算出	
21	AjcV3dInner	３Ｄ・ベクトルの内積算出	
22	AjcV3dPlaneVec	３Ｄ・三角形の法線ベクトル算出	
23	AjcV3dNormal	３Ｄ・ベクトルの単位ベクトル算出	
24	AjcV3dTheta	３Ｄ・ベクトルの角度算出	
25	AjcV3dVertVecP2L	３Ｄ・ポイントから直線へ垂直な方向ベクトル算出	
26	AjcV3dDistP2L	３Ｄ・ポイントから直線までの長さとベクトル算出	
27	AjcV3dDistP2P	３Ｄ・ポイント間の距離算出	
28	AjcV3dCrossL2L	３Ｄ・ラインの交点算出	
29	AjcV3dCrossL2LEx	３Ｄ・ラインの交点算出（交差しない場合は、最も接近する点間の中点）	
30	AjcV3dCrossP2F	３Ｄ・点からの平面への垂直なベクトルと交点算出	
31	AjcV3dOrthoVecOnPlane	３Ｄ・同一平面上で線分に直行するベクトル算出	
32	AjcV3dMultMat	３Ｄ・ベクトルと行列の掛け算	
33	AjcV3dRotateX	３Ｄ・ベクトルをＸ軸周りに回転	
34	AjcV3dRotateY	３Ｄ・ベクトルをＹ軸周りに回転	
35	AjcV3dRotateZ	３Ｄ・ベクトルをＺ軸周りに回転	
36	AjcV3dRotateAny	３Ｄ・ベクトルを任意の軸周りに回転	
37	AjcV3dAnyOrthoVec	３Ｄ・任意の直角なベクトルを求める	
38	AjcV3dRotateOnPlane	３Ｄ・平面上の点を平面上で指定角度回転	
39	AjcV3dCalcCircle[Ex] AjcV3dCalcCircleV[Ex]	３Ｄ・任意の３点を通る円の中心と半径を求める	
40	AjcV3dCalcSphere[Ex] AjcV3dCalcSphereV[Ex]	３Ｄ・球面上の任意の４点から球の中心と半径を求める	

56.2. ベクトルデータ構造体

３Ｄベクトル演算で使用する、ベクトルデータ構造体の形式を、以下に示します。

３Ｄベクトル

```
typedef struct {
    double  x, y, z;
} AJC3DVEC, *PAJC3DVEC;
typedef const AJC3DVEC *PCAJC3DVEC;
```

３Ｄ始点位置と方向ベクトル

```
typedef struct {
    AJC3DVEC    p, v;
} AJC3DLVEC, *PAJC3DLVEC;
typedef const AJC3DLVEC *PCAJC3DLVEC;
```

p: 始点位置
v: 方向ベクトル

３Ｄ線分情報（始点と終点）

```
typedef struct {
    AJC3DVEC    p1, p2;
} AJC3DLINE, *PAJC3DLINE;
typedef const AJC3DLINE *PCAJC3DLINE;
```

３Ｄ三角形情報（３つの点）

```
typedef struct {
    AJC3DVEC    p1, p2, p3;
} AJC3DTRI, *PAJC3DTRI;
typedef const AJC3DTRI *PCAJC3DTRI;
```

行列（３×３）

```
typedef struct {
    double    m[3][3];
} AJC3DMAT, *PAJC3DMAT;
typedef const AJC3DMAT *PCAJC3DMAT;
```

56.2.1. 正弦 (AjcSin)

形　式　：　double　AjcSin(double degree)

引　数　：　degree　　- 角度［度］

説　明　：　指定された角度の正弦（サイン）値を返します。

戻り値　：　正弦値（サイン値）

56.2.2. 双曲線・正弦 (AjcSinh)

形　式　：　double　AjcSinh(double degree)

引　数　：　degree　　- 角度［度］

説　明　：　指定された角度の双曲線・正弦値（ハイパーボリック・サイン）を返します。

戻り値　：　双曲線・正弦値（ハイパーボリック・サイン値）

56.2.3. 逆正弦 (AjcASin)

形　式　：　double　AjcASin(double　n)

引　数　：　n　　- 逆正弦値を算出する値（-1 ～ +1）

説　明　：　ｎで指定された値の逆正弦値（アークサイン値）を返します。

戻り値　：　逆正弦値（アークサイン値）［-90 ～ +90 度］

56.2.4. 余弦 (AjcCos)

形　式　：　double　AjcCos(double degree)

引　数　：　degree　　- 角度［度］

説　明　：　指定された角度の余弦（コサイン）値を返します。

戻り値　：　余弦値（コサイン値）

56.2.5. 双曲線・余弦(AjcCosh)

形　式　：　double　AjcCosh(double degree)

引　数　：　degree　　- 角度［度］

説　明　：　指定された角度の双曲線・余弦値（ハイパーボリック・コサイン）を返します。

戻り値　：　双曲線・余弦値（ハイパーボリック・コサイン値）

56.2.6. 逆余弦(AjcACos)

形　式　：　double　AjcACos(double　n)

引　数　：　n　　－ 逆余弦値を算出する値（-1 ～ +1）

説　明　：　ｎで指定された値の逆余弦値（アークコサイン値）を返します。

戻り値　：　逆余弦値（アークコサイン値）[0 ～ 180 度]

56.2.7. 正接 (AjcTan)

形　式　：　double　AjcTan(double degree)

引　数　：　degree　　－ 角度［度］

説　明　：　指定された角度の正接（タンジェント）値を返します。

戻り値　：　正接値（タンジェント値）

56.2.8. 双曲線・正接(AjcTanh)

形　式　：　double　AjcTanh(double degree)

引　数　：　degree　　－ 角度［度］

説　明　：　指定された角度の双曲線・正接値（ハイパーボリック・タンジェント）を返します。

戻り値　：　双曲線・正接値（ハイパーボリック・タンジェント値）

56.2.9. 逆正接(AjcATan)

形　式　：　double　AjcATan(double　n)

引　数　：　n　　－ 逆正接値を算出する値

説　明　：　ｎで指定された値の逆正接値（アークタンジェント値）を返します。

戻り値　：　逆正接値（アークタンジェント値）[-90 ～ +90 度]

56.2.10. 逆正接(AjcATan2)

形　式　：　double　AjcATan2(double　x, double y)

引　数　：　x, y　　－ 逆余弦値を算出する値

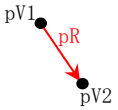
説　明　：　y/x で示される値の逆正接値（アークタンジェント値）を返します。
x,y ともに０を指定した場合は、０を返します。

戻り値　：　逆正接値（アークタンジェント値）[-180 ～ +180 度]

56.2.11. ３Ｄ・ベクトル加算(AjcV3dAdd)

形　式　：　PAJC3DVEC AjcV3dAdd(PCAJC3DVEC pV1, PCAJC3DVEC pV2, PAJC3DVEC pR)；

引　数　：　pV1, pV2　－ 加算する２つのベクトル
pR　　－ 加算結果を格納するバッファのアドレス（pV1 or pV2 と重複可）



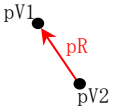
説　明　：　２つのベクトルを加算し、結果を pR で示すバッファへ格納します。

戻り値　：　≠NULL ： 加算結果ベクトルへのポインタ（=pR）
＝NULL ： エラー（NULL ポインタ）

56.2.12. ３Ｄ・ベクトル減算(AjcV3dSub)

形　式　：　PAJC3DVEC AjcV3dSub(PCAJC3DVEC pV1, PCAJC3DVEC pV2, PAJC3DVEC pR)；

引　数　：　pV1, pV2　－ 減算する２つのベクトル
pR　　－ 減算結果を格納するバッファのアドレス（pV1 or pV2 と重複可）



説　明　：　２つのベクトルを減算(V1 － V2)し、結果を pR で示すバッファへ格納します。

戻り値　：　≠NULL ： 加算結果ベクトルへのポインタ（=pR）
＝NULL ： エラー（NULL ポインタが指定された）

56.2.13. ３Ｄ・ベクトル乗算(AjcV3dMult)

形　式　：　PAJC3DVEC AjcV3dMult(PCAJC3DVEC pV, double n, PAJC3DVEC pR)；

引　数　：　pV　　－ 乗算するのベクトル
n　　－ 乗数
pR　　－ 乗算結果を格納するバッファのアドレス（pV と重複可）

説　明　：　ベクトルを乗算し、結果を pR で示すバッファへ格納します。

戻り値　：　≠NULL ： 乗算結果ベクトルへのポインタ（=pR）
＝NULL ： エラー（NULL ポインタが指定された）

56.2.14. ３Ｄ・ベクトル除算(AjcV3dDiv)

形　式　：　PAJC3DVEC AjcV3dDiv(PCAJC3DVEC pV, double n, PAJC3DVEC pR)；

引　数　：　pV　　－ 除算するのベクトル
n　　－ 除数
pR　　－ 除算結果を格納するバッファのアドレス（pV と重複可）

説　明　：　ベクトルを除算し、結果を pR で示すバッファへ格納します。

戻り値　：　≠NULL ： 除算結果ベクトルへのポインタ（=pR）
＝NULL ： エラー（NULL ポインタが指定された）

56.2.15. ３Ｄ・線ベクトル設定(AjcV3dSetLineVec)

形　式 ： PAJC3DLVEC AjcV3dSetLineVec (PCAJC3DVEC p1, PCAJC3DVEC p2, PAJC3DLINE pR);

引　数 ： p1, p2 - 線分を示す２つの点（始点，終点）
pR　　- 線ベクトルを格納するバッファ

説　明 ： pR で示すバッファへ線ベクトル（始点と方向ベクトル）を設定します。
pR.p には、線分の始点(p1)が設定されます
pR.v には、方向ベクトル(p1→p2)が設定されます。

戻り値 ： ≠NULL ： 線ベクトルへのポインタ（=pR）
＝NULL ： エラー（NULL ポインタが指定された）

56.2.16. ３Ｄ・線分情報設定(AjcV3dSetLinePoint)

形　式 ： PAJC3DLINE AjcV3dSetLinePoint (PCAJC3DVEC p1, PCAJC3DVEC p2, PAJC3DLINE pR);

引　数 ： p1, p2 - 線分を示す２つの点（始点，終点）
pR　　- 線分情報を格納するバッファ

説　明 ： pR で示すバッファへ線分情報（始点と終点）を設定します。

戻り値 ： ≠NULL ： 線分情報へのポインタ（=pR）
＝NULL ： エラー（NULL ポインタが指定された）

56.2.17. ３Ｄ・三角形情報設定(AjcV3dSetTriPoint)

形　式 ： PAJC3DTRI AjcV3dSetTriPoint (PCAJC3DVEC p1, PCAJC3DVEC p2, PCAJC3DVEC p3, PAJC3DTRI pR);

引　数 ： p1, p2, p3　- 三角形を示す３つの点
pR　　- 三角形情報を格納するバッファ

説　明 ： pR で示すバッファへ三角形情報（３つの点）を設定します。

戻り値 ： ≠NULL ： 三角形情報へのポインタ（=pR）
＝NULL ： エラー（NULL ポインタが指定された）

56.2.18. ３Ｄ・ベクトルの長さ算出(AjcV3dLength)

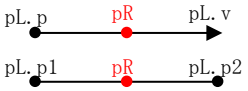
形　式 ： double AjcV3dLength (PCAJC3DVEC pV);

引　数 ： pV　　- 長さを算出するベクトル

説　明 ： pV で示すベクトルの長さを算出します。

戻り値 ： ≠ -1.0 ： ベクトルの長さ
＝ -1.0 ： エラー（NULL ポインタが指定された）

56.2.19. ３Ｄ・線分の中点算出(AjcV3dLineCenter)



形　式 ： PAJC3DVEC AjcV3dLVecCenter (PCAJC3DLVEC pL, PAJC3DVEC pR);
PAJC3DVEC AjcV3dLineCenter (PCAJC3DLINE pL, PAJC3DVEC pR);

引　数 ： pV　　- 中点を算出する線ベクトル／線分情報（始点と方向ベクトル ／ 始点と終点）
pR　　- 中点を格納するバッファのアドレス

説　明 ： 線ベクトル／線分の中点を算出し、pR で示すバッファへ格納します。

戻り値 ： ≠ NULL ： 中点情報へのポインタ(=pR)
＝ NULL ： エラー（NULL ポインタが指定された）

56.2.20. ３Ｄ・外積算出(AjcV3dOuter)

形　式　：　PAJC3DVEC AjcV3dOuter (PCAJC3DVEC pV1, PCAJC3DVEC pV2, PAJC3DVEC pR);

引　数　：　pV1, pV2 - 外積を算出する２つの方向ベクトル
pR　　- 外積を格納するバッファのアドレス（pV1 or pV2 と重複可）

説　明　：　２つのベクトルの外積を算出し、pR で示すバッファへ格納します。

戻り値　：　≠ NULL : 外積情報へのポインタ(=pR)
＝ NULL : エラー（NULL ポインタが指定された）

56.2.21. ３Ｄ・内積算出(AjcV3dInner)

形　式　：　PAJC3DVEC AjcV3dInner (PCAJC3DVEC pV1, PCAJC3DVEC pV2);

引　数　：　pV1, pV2 - 内積を算出する２つのベクトル

説　明　：　２つのベクトルの内積を算出します。

戻り値　：　内積値（エラー時（NULL ポインタが指定された場合）は、０を返します）

56.2.22. ３Ｄ・三角形の法線ベクトル算出(AjcV3dPlaneVec)

形　式　：　PAJC3DVEC AjcV3dPlaneVec(PCAJC3DTRI pT, PAJC3DVEC pR);

引　数　：　pT　　- 法線を算出する三角形情報（３つの点）
pR　　- 法線をベクトルを格納するバッファのアドレス

説　明　：　三角形情報（３つの点）から法線ベクトルを算出します。

戻り値　：　≠ NULL : 法線ベクトル情報へのポインタ(=pR)
＝ NULL : エラー（NULL ポインタが指定された）

56.2.23. ３Ｄ・単位ベクトル算出(AjcV3dNormal)

形　式　：　PAJC3DVEC AjcV3dNormal (PCAJC3DVEC pV , PAJC3DVEC pR);

引　数　：　pV　　- 単位ベクトルを算出するベクトル情報
pR　　- 算出した単位ベクトルを格納するバッファのアドレス（pV と重複可）

説　明　：　pV で指定されたベクトルの単位ベクトルを算出します。

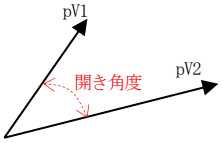
戻り値　：　≠ NULL : 単位ベクトル情報へのポインタ(=pR)
＝ NULL : エラー（NULL ポインタが指定された）

56.2.24. ３Ｄ・ベクトルの角度算出(AjcV3dTheta)

形　式　：　double AjcV3dTheta (PCAJC3DVEC pV1, PCAJC3DVEC pV2);

引　数　：　pV1, pV2　- 開き角度を算出する２つのベクトル

説　明　：　pV1 と ｐV2 で指定されたベクトルの開き角度を算出します。

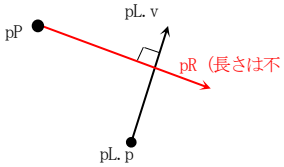


戻り値　：　≠ -1 : 開き角度（0～180 度）
＝ -1 : エラー（NULL ポインタ／ゼロベクトルが指定された）

56.2.25. ３Ｄ・ポイントから直線へ垂直な方向ベクトル算出(AjcV3dVertVecP2L)

形　式 ： PAJC3DVEC AjcV3dVertVecP2L (PCAJC3DLVEC pL , PCAJC3DVEC pP , PAJC3DVEC pR);

引　数 ： pL　　- 線ベクトル（線分の始点と方向ベクトル）
pP　　- 点の位置
pR　　- 点から線分への垂直なベクトルを格納するバッファのアドレス（pP と重複可）

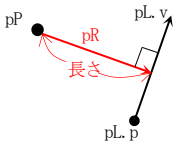


説　明 ： pP で示す点から、pL で示す線分への垂直なベクトルを算出します。
算出したベクトル(pR)の長さは不定です。

戻り値 ： ≠ NULL ： 点から線分への垂直なベクトル情報へのポインタ(=pR)
＝ NULL ： エラー（NULL ポインタが指定された）

56.2.26. ３Ｄ・ポイントから直線までの長さとベクトル算出(AjcV3dDistP2L)

形　式 ： double AjcV3dDistP2L (PCAJC3DLVEC pL , PCAJC3DVEC pP , PAJC3DVEC pR);



引　数 ： pL　　- 線ベクトル（線分の始点と方向ベクトル）
pP　　- 点の位置
pR　　- 点から線分への垂直なベクトルを格納するバッファのアドレス（pP と重複可）

説　明 ： pP で示す点から、pL で示す線ベクトルまでの長さを算出します。
pR には、pP で示す点から、pL で示す線ベクトルへの垂直なベクトルを格納します。（ベクトル長＝線分までの距離）

戻り値 ： ≠ -1 ： 点から線分までの長さ
＝ -1 ： エラー（NULL ポインタが指定された）

56.2.27. ３Ｄ・ポイント間の距離算出(AjcV3dDistP2P)

形　式 ： double AjcV3dDistP2P (PCAJC3DVEC pV1, PCAJC3DVEC pV2);

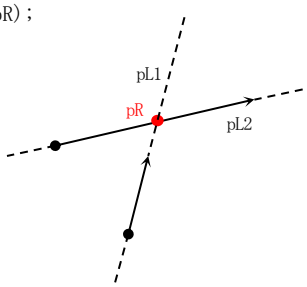
引　数 ： pV1, pV2　- 距離を算出する２つの点

説　明 ： ２つの点間の距離を算出します。

戻り値 ： ≠ -1 ： ２つの点間の距離
＝ -1 ： エラー（NULL ポインタが指定された）

56.2.28. ３Ｄ・ラインの交点算出(AjcV3dCrossL2L)

形　式 ： PAJC3DVEC AjcV3dCrossL2L (PCAJC3DLVEC pL1, PCAJC3DLVEC pL2, PAJC3DVEC pR);



引　数 ： pL1, pL2　- 交点を算出する２つの線ベクトル（始点と方向ベクトル）
pR　　- 交点位置を格納するバッファのアドレス

説　明 ： ２つの線ベクトルの交点位置を算出します。
２つの線ベクトルは、前後に延長した無限の線分とします。
２つの線ベクトルは、必ず交差しなければなりません。

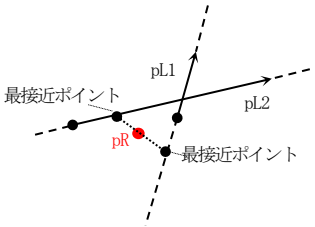
戻り値 ： ≠ NULL ： ２つの線ベクトルの交点位置へのポインタ(=pR)
＝ NULL ： エラー（NULL ポインタが指定された）

56.2.29. ３Ｄ・２つのラインの交点算出(AjcV3dCrossL2LEx)

形　式　：　PAJC3DVEC AjcV3dCrossL2LEx (PCAJC3DLVEC pL1, PCAJC3DLVEC pL2, PAJC3DVEC pR);

引　数　：　pL1, pL2　- 交点を算出する２つの線ベクトル（始点と方向ベクトル）
pR　　- 交点位置を格納するバッファのアドレス

説　明　：　２つの線ベクトルの交点位置を算出します。
２つの線ベクトルは、前後に延長した無限の線分とします。
２つの線ベクトルが交差しない場合は、最も接近する点の中点を求めます。



戻り値　：　≠ NULL ： ２つの線分の交点情報へのポインタ(=pR)
＝ NULL ： エラー（NULL ポインタが指定された）

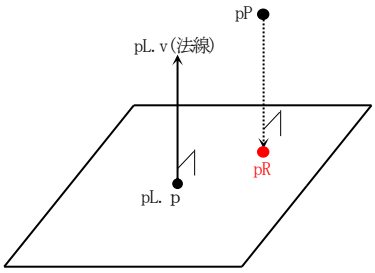
56.2.30. ３Ｄ・点との平面の交点算出(AjcV3dCrossP2F)

形　式　：　PAJC3DVEC AjcV3dCrossP2F (PCAJC3DLVEC pL, PCAJC3DVEC pP, PAJC3DVEC pR);

引　数　：　pL　　- 平面上の任意の点と、平面の法線ベクトル
pP　　- ポイント情報
pR　　- 点と平面の交点を格納するバッファのアドレス

説　明　：　点と平面の交点を算出します。

戻り値　：　≠ NULL ： 点と平面の交点情報へのポインタ(=pR)
＝ NULL ： エラー（NULL ポインタが指定された）



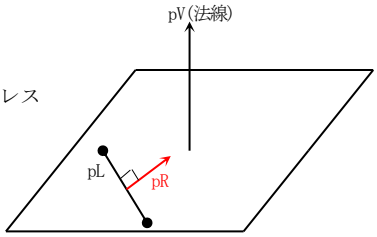
56.2.31. ３Ｄ・同一平面上で線分に直行するベクトル算出(AjcV3dOrthoVecOnPlane)

形　式　：　PAJC3DVEC AjcV3dOrthoVecOnPlane (PCAJC3DLINE pL, PCAJC3DVEC pV, PAJC3DVEC pR);

引　数　：　pL　　- 平面上の線分情報のアドレス（平面上の始点と終点）
pV　　- 平面からの法線ベクトル
pR　　- 同一平面上で直行する方向ベクトル情報を格納するバッファのアドレス

説　明　：　同一平面上で線分に直行するベクトルを算出します。

戻り値　：　≠ NULL ： 同一平面上で直行する方向ベクトル情報へのポインタ(=pR)
＝ NULL ： エラー（NULL ポインタが指定された）



56.2.32. ３Ｄ・ベクトルと行列の掛け算(AjcV3dMultMat)

形　式　：　PAJC3DVEC AjcV3dMultMat (PCAJC3DVEC pV, PCAJC3DMAT pM, PAJC3DVEC pR);

引　数　：　pV　　- 被乗数ベクトル
pM　　- 乗数（行列）
pR　　- 乗算結果のベクトルを格納するバッファのアドレス

説　明　：　ベクトルへ行列を乗算します。

戻り値　：　≠ NULL ： 乗算結果のベクトルへのポインタ(=pR)
＝ NULL ： エラー（NULL ポインタが指定された）

56.2.33. ３Ｄ・ベクトルをＸ軸周りに回転(AjcV3dRotateX)

形　式　：　PAJC3DVEC AjcV3dRotateX (PCAJC3DVEC pV, double t, PAJC3DVEC pR);

引　数　：　pV　　- 回転するベクトル
t　　- 回転角度（度）
pR　　- 回転結果のベクトルを格納するバッファのアドレス

説　明　：　ベクトル（点）をＸ軸回りに回転します。

戻り値　：　≠ NULL : 回転結果のベクトルへのポインタ(=pR)
＝ NULL : エラー（NULL ポインタが指定された）

56.2.34. ３Ｄ・ベクトルをＹ軸周りに回転(AjcV3dRotateY)

形　式　：　PAJC3DVEC AjcV3dRotateY (PCAJC3DVEC pV, double t, PAJC3DVEC pR);

引　数　：　pV　　- 回転するベクトル
t　　- 回転角度（度）
pR　　- 回転結果のベクトルを格納するバッファのアドレス

説　明　：　ベクトル（点）をＹ軸回りに回転します。

戻り値　：　≠ NULL : 回転結果のベクトルへのポインタ(=pR)
＝ NULL : エラー（NULL ポインタが指定された）

56.2.35. ３Ｄ・ベクトルをＺ軸周りに回転(AjcV3dRotateZ)

形　式　：　PAJC3DVEC AjcV3dRotateZ (PCAJC3DVEC pV, double t, PAJC3DVEC pR);

引　数　：　pV　　- 回転するベクトル
t　　- 回転角度（度）
pR　　- 回転結果のベクトルを格納するバッファのアドレス

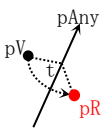
説　明　：　ベクトル（点）をＺ軸回りに回転します。

戻り値　：　≠ NULL : 回転結果のベクトルへのポインタ(=pR)
＝ NULL : エラー（NULL ポインタが指定された）

56.2.36. ３Ｄ・ベクトルを任意の軸周りに回転(AjcV3dRotateAny)

形　式　：　PAJC3DVEC AjcV3dRotateAny (PCAJC3DVEC pV, double t, PCAJC3DVEC pAny, PAJC3DVEC pR);

引　数　：　pV　　- 回転するベクトル
t　　- 回転角度（度）
pAny　　- 任意の回転軸ベクトル（原点(0,0,0)を通るベクトル）
pR　　- 回転結果のベクトルを格納するバッファのアドレス



説　明　：　ベクトル（点）を任意の軸回りに回転します。

戻り値　：　≠ NULL : 回転結果のベクトルへのポインタ(=pR)
＝ NULL : エラー（NULL ポインタが指定された）

56.2.37. ３Ｄ・任意の直角なベクトル算出(AjcV3dAnyOrthoVec)

形　式 ： PAJC3DVEC AjcV3dAnyOrthoVec (PCAJC3DVEC pV, PAJC3DVEC pR);

引　数 ： pV　　- ベクトル
pR　　- ベクトルに直角な任意のベクトルを格納するバッファのアドレス



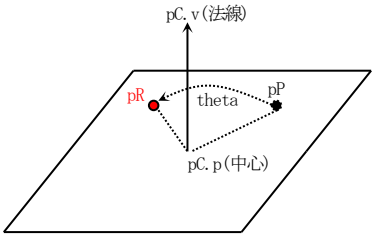
説　明 ： ベクトルに直角な任意のベクトルを算出します。
ベクトル(pV)に直行することは保証されますが、ベクトルの方向は不定です。

戻り値 ： ≠ NULL ： ベクトルに直角な任意のベクトルへのポインタ(=pR)
＝ NULL ： エラー (NULL ポインタが指定された)

56.2.38. ３Ｄ・平面上の点を平面上で指定角度回転 (AjcV3dRotateOnPlane)

形　式 ： PAJC3DVEC AjcV3dRotateOnPlane(PCAJC3DVEC pP, PCAJC3DLVEC pC, double theta, PAJC3DVEC pR);

引　数 ： pP　　- ベクトル
pC　　- 平面の中心と法線情報のアドレス
theta　- 回転角度（度）
pR　　- 回転後の座標を格納するバッファのアドレス



説　明 ： 平面上の点を平面上で指定角度回転します。

戻り値 ： ≠ NULL ： 回転したベクトルへのポインタ(=pR)
＝ NULL ： エラー (NULL ポインタが指定された)

56.2.39. ３Ｄ・任意の３点を通る円の中心と半径を求める(AjcV3dCalcCircle[V])

形　式 ： double AjcV3dCalcCircle (double x1, double y1, double z1,
double x2, double y2, double z2,
double x3, double y3, double z3, PAJC3DVEC pR, PAJC3DVEC pV);

double AjcV3dCalcCircleEx(double x1, double y1, double z1,
double x2, double y2, double z2,
double x3, double y3, double z3, PAJC3DVEC pR, PAJC3DVEC pV, PAJC3DCIRINFO
pCir);

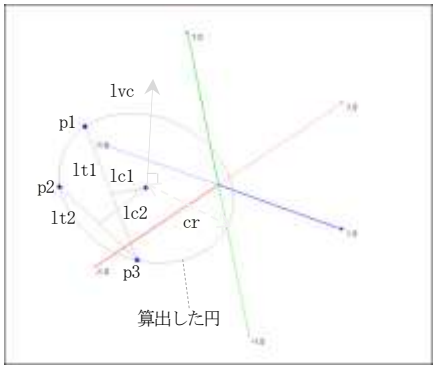
double AjcV3dCalcCircleV (PCAJC3DVEC p1, PCAJC3DVEC p2, PCAJC3DVEC p3, PAJC3DVEC pR, PAJC3DVEC pV);

double AjcV3dCalcCircleVEx(PCAJC3DVEC p1, PCAJC3DVEC p2, PCAJC3DVEC p3, PAJC3DVEC pR, PAJC3DVEC pV , PAJC3DCIRINFO pCir);

引　数 ： x1, y1, z1 ／ p1 － 任意の点１
x2, y2, z2 ／ p2 － 任意の点２
x3, y3, z3 ／ p3 － 任意の点３
pR － 円の中心を格納するバッファのアドレス （不要時は NULL）
pV － 円の法線を格納するバッファのアドレス （不要時は NULL）
pCir － 円の算出情報を格納するバッファのアドレス（不要時は NULL）

説　明 ： ３Ｄ空間上の任意の３点を指定し、３点を通る平面円を算出します。
pCir は円を算出した際の演算情報で、以下の形式です。

```
typedef struct {
    AJC3DLINE   lt1, lt2;          //  円に内接する２つの直線
    AJC3DLINE   lc1, lc2;          //  内接線の中点からの垂線
    AJC3DLVEC   lvc;               //  円の中心と法線
    double      cr;                //  円の半径
} AJC3DCIRINFO, *PAJC3DCIRINFO;
typedef const AJC3DCIRINFO *PCAJC3DCIRINFO;
```



戻り値 ： ≠ -1 ： 円の半径
＝ -1 ： エラー（NULL ポインタが指定された／３点が直線上にある）

56.2.40. ３Ｄ・球面上の任意の４点から球の中心と半径を求める(AjcV3dCalcSphere [V])

形　式 ：
```
double AjcV3dCalcSphere  (double x1, double y1, double z1,
                          double x2, double y2, double z2,
                          double x3, double y3, double z3,
                          double x4, double y4, double z4, PAJC3DVEC pR);

double AjcV3dCalcSphereEx(double x1, double y1, double z1,
                          double x2, double y2, double z2,
                          double x3, double y3, double z3,
                          double x4, double y4, double z4, PAJC3DVEC pR, PAJC3DSPHINFO pSph);

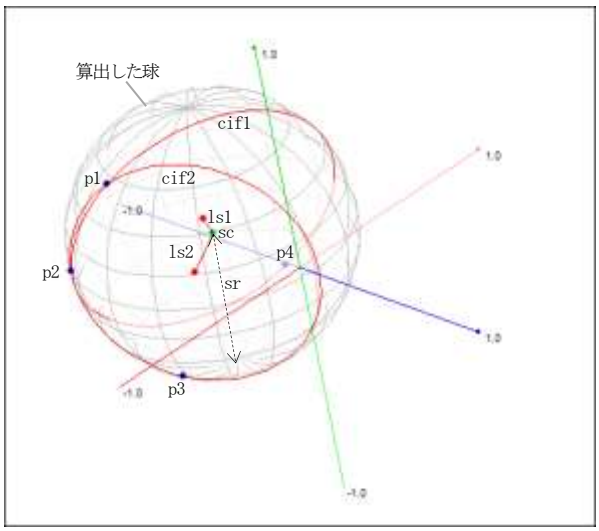
double AjcV3dCalcSphereV  (PCAJC3DVEC p1, PCAJC3DVEC p2, PCAJC3DVEC p3, PCAJC3DVEC p4, PAJC3DVEC pR);

double AjcV3dCalcSphereVEx(PCAJC3DVEC p1, PCAJC3DVEC p2, PCAJC3DVEC p3, PCAJC3DVEC p4, PAJC3DVEC pR, PAJC3DSPHINFO pSph);
```

引　数 ：
```
x1, y1, z1 ／ p1  - 球面上の任意の点１
x2, y2, z2 ／ p2  - 球面上の任意の点２
x3, y3, z3 ／ p3  - 球面上の任意の点３
x4, y4, z4 ／ p4  - 球面上の任意の点４
pR                - 球の中心を格納するバッファのアドレス      （不要時は NULL）
pSph              - 球の算出情報を格納するバッファのアドレス（不要時は NULL）
```

説　明 ： 球面上の任意の４点を指定し、球体を算出します。
pSph は球体を算出した際の演算情報で、以下の形式です。

```
typedef struct {
    AJC3DCIRINFO    cif1, cif2;     //  球に内接する２つの円
    AJC3DLINE       ls1 , ls2;      //  内接円中心から球中心への直線
    AJC3DVEC        sc;             //  球の中心
    double          sr;             //  球の半径
} AJC3DSPHINFO, *PAJC3DSPHINFO;
typedef const AJC3DSPHINFO *PCAJC3DSPHINFO;
```



戻り値 ： ≠ -1 ： 球の半径
＝ -1 ： エラー（NULL ポインタが指定された／いずれかの３点が直線上にある）

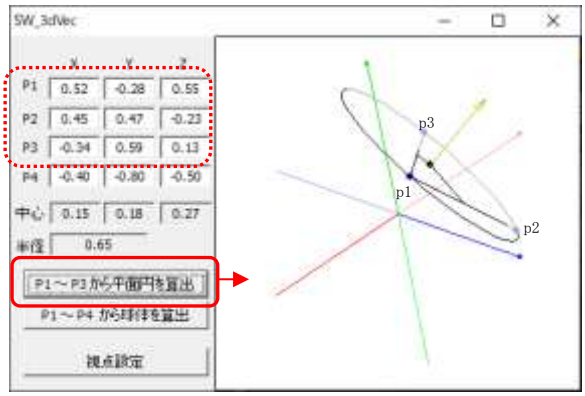
56.3. サンプルプログラム

56.3.1. SW_3dVec（３Ｄ円の算出と３Ｄ球の算出）

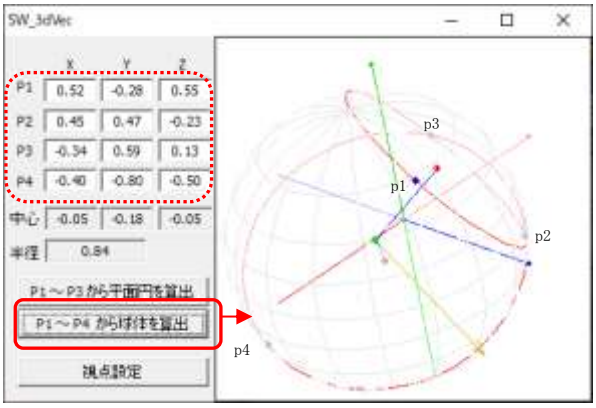
このサンプルプログラムは、以下の演算を行います。

・任意の３点（p1 ～ p3）を通る平面円（円の中心，法線と半径）を算出
・球面上の４点を指定し、球体（球の中心と半径）を算出

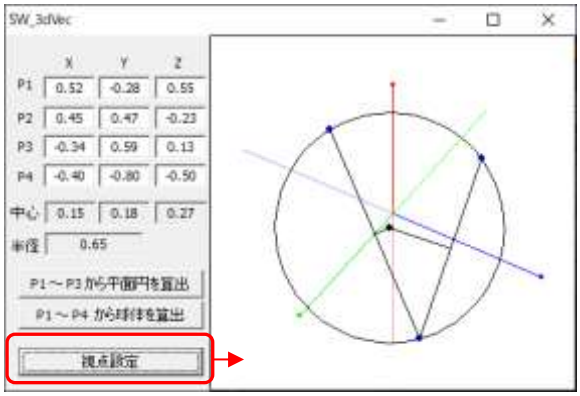
３点(P1～P3)を通る平面円算出



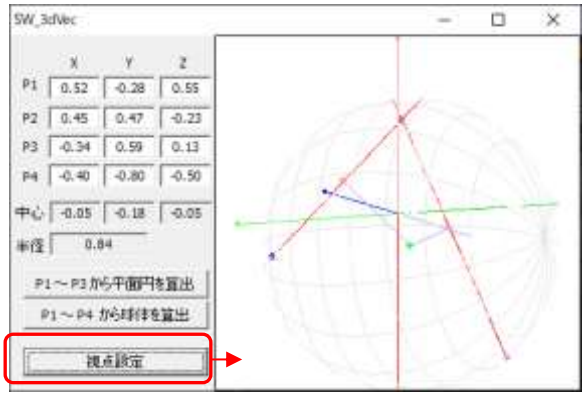
球面上の４点(P1～P4)を通る球体算出



平面円の真上から見た図



２つの平面円の真横から見た図



```
 1 : //
 2 : //  SW_3dVec.c
 3 : //
 4 :
 5 : #include    <AjrCstXX.h>
 6 : #include    <math.h>
 7 : #include    <tchar.h>
 8 : #include    "resource.h"
 9 :
10 : #define     BLUE        0
11 : #define     RED         1
12 : #define     GREEN       2
13 : #define     YELLOW      5
14 : #define     GRAY        6
15 : #define     BLACK       7
16 :
17 : //-------------------------------------------------------------------------------------------------------------------//
18 : //  ワーク                                                                                                           //
19 : //-------------------------------------------------------------------------------------------------------------------//
20 : HINSTANCE       hInst;                      //  ＤＬＬインスタンスハンドル
21 : HWND            hDlgMain;                   //  ダイアログボックスハンドル
22 : HWND            hWndCtrl;                   //  ３Ｄグラフコントロールのウインドハンドル
23 : int             CtrlLeft;                   //  ３Ｄグラフコントロールの上端位置
24 :
25 : AJC3DVEC        vs;                         //  視点ベクトル
26 :
27 : //-------------------------------------------------------------------------------------------------------------------//
28 : //  内部サブ関数                                                                                                     //
29 : //-------------------------------------------------------------------------------------------------------------------//
30 : AJC_DLGPROC_DEF(Main);
31 :
32 : //===================================================================================================================//
33 : //                                                                                                                   //
34 : //  Ｗ ｉ ｎ Ｍ ａ ｉ ｎ                                                                                             //
35 : //                                                                                                                   //
36 : //===================================================================================================================//
37 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
38 : {
39 :     MSG     msg;
40 :     int     sty;
41 :     RECT    rect;
42 :
43 :     hInst = hInstance;
44 :
45 :     //----- メイン・ダイアログオープン -----------------//
46 :     hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_DLGMAIN), NULL, AJC_DLGPROC_NAME(Main));
47 :     sty = (int)MAjcGetWindowLong(hDlgMain, GWL_STYLE);
48 :     MAjcSetWindowLong(hDlgMain, GWL_STYLE, sty | WS_THICKFRAME);
49 :     //----- ３Ｄグラフコントロール位置初期化 -----------//
50 :     GetWindowRect(hWndCtrl, &rect);
51 :     MapWindowPoints(NULL, hDlgMain, (LPPOINT)&rect, 2);
52 :     CtrlLeft = rect.left;
53 :     GetClientRect(hDlgMain, &rect);
54 :     MoveWindow(hWndCtrl, CtrlLeft, 0, rect.right - rect.left - CtrlLeft, rect.bottom - rect.top, TRUE);
55 :     //----- ダイアログ表示 -----------------------------//
56 :     ShowWindow(hDlgMain, SW_SHOW);
57 :
58 :     //----- メッセージループ ---------------------------//
59 :     while (GetMessage(&msg, NULL, 0, 0)) {
60 :         do {
61 :             if (IsDialogMessage(hDlgMain, &msg)) break;
62 :             TranslateMessage(&msg);
63 :             DispatchMessage (&msg);
64 :         } while (0);
65 :     }
66 :
67 :     return (int)msg.wParam ;
68 : }
69 : //===================================================================================================================//
70 : //                                                                                                                   //
71 : //  ダイアログ・プロシージャ                                                                                         //
72 : //                                                                                                                   //
73 : //===================================================================================================================//
74 : //----- ダイアログ初期化 -----------------------------------------------------------------------------------------//
75 : AJC_DLGPROC(Main, WM_INITDIALOG     )
76 : {
77 :     AJC3DVEC    lo  = {-1.0, -1.0, -1.0};
78 :     AJC3DVEC    hi  = { 1.0,  1.0,  1.0};
79 :     AJC3DVEC    rot = {  60,   10,   45};
80 :
```

```
 81 :     hDlgMain = hDlg;
 82 :     hWndCtrl = GetDlgItem(hDlgMain, IDC_3DGRAPH);
 83 :     AjcG3dSetTipText(hWndCtrl, TEXT("３Ｄグラフィック"));
 84 :
 85 :     //----- ３Ｄグラフモード設定 ------------//
 86 :     AjcG3dInitV(hWndCtrl, &lo, &hi, &rot, (AJC3DGS_AXIS));
 87 :
 88 :     //----- 球表示色（グレー）を薄く設定 ----//
 89 :     AjcG3dSetColor (hWndCtrl, GRAY, RGB(230, 230, 230), RGB(250, 250, 250));
 90 :
 91 :     //----- ４点の座標値設定 ----------------//
 92 :     AjcSetDlgItemReal(hDlg, IDC_TXT_X1,  0.52, 2);
 93 :     AjcSetDlgItemReal(hDlg, IDC_TXT_Y1, -0.28, 2);
 94 :     AjcSetDlgItemReal(hDlg, IDC_TXT_Z1,  0.55, 2);
 95 :
 96 :     AjcSetDlgItemReal(hDlg, IDC_TXT_X2,  0.45, 2);
 97 :     AjcSetDlgItemReal(hDlg, IDC_TXT_Y2,  0.47, 2);
 98 :     AjcSetDlgItemReal(hDlg, IDC_TXT_Z2, -0.23, 2);
 99 :
100 :     AjcSetDlgItemReal(hDlg, IDC_TXT_X3, -0.34, 2);
101 :     AjcSetDlgItemReal(hDlg, IDC_TXT_Y3,  0.59, 2);
102 :     AjcSetDlgItemReal(hDlg, IDC_TXT_Z3,  0.13, 2);
103 :
104 :     AjcSetDlgItemReal(hDlg, IDC_TXT_X4, -0.40, 2);
105 :     AjcSetDlgItemReal(hDlg, IDC_TXT_Y4, -0.80, 2);
106 :     AjcSetDlgItemReal(hDlg, IDC_TXT_Z4, -0.50, 2);
107 :
108 :     return TRUE;
109 : }
110 : //----- ウインド破棄 ------------------------------------------------------------------------------------------//
111 : AJC_DLGPROC(Main, WM_DESTROY        )
112 : {
113 :     AjcReleaseDlgProps(hDlg);
114 :     PostQuitMessage(0);
115 :     return TRUE;
116 : }
117 : //----- サイズ変更 --------------------------------------------------------------------------------------------//
118 : AJC_DLGPROC(Main, WM_SIZE           )
119 : {
120 :     int     w = LOWORD(lParam);
121 :     int     h = HIWORD(lParam);
122 :
123 :     MoveWindow  (GetDlgItem(hDlg, IDC_3DGRAPH), CtrlLeft, 0, w - CtrlLeft, h, TRUE);
124 :
125 :     return TRUE;
126 : }
127 : //----- 「平面円算出」ボタン ----------------------------------------------------------------------------------//
128 : AJC_DLGPROC(Main, IDC_CMD_CIR       )
129 : {
130 :     AJC3DVEC        p1, p2, p3, vc, vh, v1, v2;
131 :     double          r;
132 :     AJC3DCIRINFO    cif;
133 :
134 :     //  ３点設定
135 :     p1.x = AjcGetDlgItemReal(hDlg, IDC_TXT_X1);
136 :     p1.y = AjcGetDlgItemReal(hDlg, IDC_TXT_Y1);
137 :     p1.z = AjcGetDlgItemReal(hDlg, IDC_TXT_Z1);
138 :
139 :     p2.x = AjcGetDlgItemReal(hDlg, IDC_TXT_X2);
140 :     p2.y = AjcGetDlgItemReal(hDlg, IDC_TXT_Y2);
141 :     p2.z = AjcGetDlgItemReal(hDlg, IDC_TXT_Z2);
142 :
143 :     p3.x = AjcGetDlgItemReal(hDlg, IDC_TXT_X3);
144 :     p3.y = AjcGetDlgItemReal(hDlg, IDC_TXT_Y3);
145 :     p3.z = AjcGetDlgItemReal(hDlg, IDC_TXT_Z3);
146 :
147 :     //  円の中心と半径算出
148 :     r = AjcV3dCalcCircleVEx(&p1, &p2, &p3, &vc, &vh, &cif);
149 :
150 :     //  円の中心と半径値表示
151 :     AjcSetDlgItemReal(hDlg, IDC_TXT_CX, vc.x, 2);
152 :     AjcSetDlgItemReal(hDlg, IDC_TXT_CY, vc.y, 2);
153 :     AjcSetDlgItemReal(hDlg, IDC_TXT_CZ, vc.z, 2);
154 :     AjcSetDlgItemReal(hDlg, IDC_TXT_R , r   , 2);
155 :
156 :     //  描画クリアー
157 :     AjcG3dClearAllShape(hWndCtrl);
158 :     //  ３点描画
159 :     AjcG3dPixelV (hWndCtrl, BLUE, &p1, 3);
160 :     AjcG3dPixelV (hWndCtrl, BLUE, &p2, 3);
```

```
161 :     AjcG3dPixelV (hWndCtrl, BLUE, &p3, 3);
162 :     //  円に内接する２つの直線描画
163 :     AjcG3dLineV  (hWndCtrl, BLACK, &cif.lc1.p1, &cif.lc1.p2);
164 :     AjcG3dLineV  (hWndCtrl, BLACK, &cif.lc2.p1, &cif.lc2.p2);
165 :     //  ２つの直線の中点からの垂線描画
166 :     AjcG3dLineV  (hWndCtrl, BLACK, &cif.lt1.p1, &cif.lt1.p2);
167 :     AjcG3dLineV  (hWndCtrl, BLACK, &cif.lt2.p1, &cif.lt2.p2);
168 :     //  平面の定義
169 :     AjcG3dDefPlaneV(hWndCtrl, BLACK, &cif.lvc, NULL);
170 :     //  円と円の中心描画
171 :     AjcG2dEllipse(hWndCtrl, BLACK, 0, 0, cif.cr, cif.cr);
172 :     AjcG2dPixel  (hWndCtrl, BLACK, 0, 0, 3);
173 :
174 :     //  平面の法線設定
175 :     AjcV3dSub(&p2, &p1, &v1);           //  v1 = p1→p2 ベクトル
176 :     AjcV3dSub(&p3, &p1, &v2);           //  v2 = p1→p3 ベクトル
177 :     AjcV3dOuter(&v1, &v2, &vs);         //  法線算出
178 :     AjcV3dNormal(&vs, &vs);
179 :     AjcV3dMult  (&vs, 0.5, &vs);
180 :
181 :     //  平面の法線描画
182 :     AjcV3dAdd(&vs, &cif.lvc.p, &v1);
183 :     AjcG3dArrowV(hWndCtrl, YELLOW, &cif.lvc.p, &v1);
184 :
185 :
186 :     return TRUE;
187 : }
188 : //-----  「球体算出」ボタン ----------------------------------------------------------------------------------//
189 : AJC_DLGPROC(Main, IDC_CMD_SPH       )
190 : {
191 :     AJC3DVEC        p1, p2, p3, p4, vc, v1, v2;
192 :     double          r;
193 :     AJC3DSPHINFO    sif;
194 :
195 :     //  ４点設定
196 :     p1.x = AjcGetDlgItemReal(hDlg, IDC_TXT_X1);
197 :     p1.y = AjcGetDlgItemReal(hDlg, IDC_TXT_Y1);
198 :     p1.z = AjcGetDlgItemReal(hDlg, IDC_TXT_Z1);
199 :
200 :     p2.x = AjcGetDlgItemReal(hDlg, IDC_TXT_X2);
201 :     p2.y = AjcGetDlgItemReal(hDlg, IDC_TXT_Y2);
202 :     p2.z = AjcGetDlgItemReal(hDlg, IDC_TXT_Z2);
203 :
204 :     p3.x = AjcGetDlgItemReal(hDlg, IDC_TXT_X3);
205 :     p3.y = AjcGetDlgItemReal(hDlg, IDC_TXT_Y3);
206 :     p3.z = AjcGetDlgItemReal(hDlg, IDC_TXT_Z3);
207 :
208 :     p4.x = AjcGetDlgItemReal(hDlg, IDC_TXT_X4);
209 :     p4.y = AjcGetDlgItemReal(hDlg, IDC_TXT_Y4);
210 :     p4.z = AjcGetDlgItemReal(hDlg, IDC_TXT_Z4);
211 :
212 :     //  球の中心と半径算出
213 :     r = AjcV3dCalcSphereVEx(&p1, &p2, &p3, &p4, &vc, &sif);
214 :
215 :     //  球の中心と半径値表示
216 :     AjcSetDlgItemReal(hDlg, IDC_TXT_CX, vc.x, 2);
217 :     AjcSetDlgItemReal(hDlg, IDC_TXT_CY, vc.y, 2);
218 :     AjcSetDlgItemReal(hDlg, IDC_TXT_CZ, vc.z, 2);
219 :     AjcSetDlgItemReal(hDlg, IDC_TXT_R , r   , 2);
220 :
221 :     //  描画クリアー
222 :     AjcG3dClearAllShape(hWndCtrl);
223 :     //  ４点描画
224 :     AjcG3dPixelV (hWndCtrl, BLUE, &p1, 3);
225 :     AjcG3dPixelV (hWndCtrl, BLUE, &p2, 3);
226 :     AjcG3dPixelV (hWndCtrl, BLUE, &p3, 3);
227 :     AjcG3dPixelV (hWndCtrl, BLUE, &p4, 3);
228 :
229 :     //  ２つの円描画
230 :     AjcG3dDefPlaneV(hWndCtrl, RED, &sif.cif1.lvc, NULL);
231 :     AjcG2dEllipse  (hWndCtrl, RED, 0, 0, sif.cif1.cr, sif.cif1.cr);
232 :     AjcG2dPixel    (hWndCtrl, RED, 0, 0, 3);
233 :
234 :     AjcG3dDefPlaneV(hWndCtrl, RED, &sif.cif2.lvc, NULL);
235 :     AjcG2dEllipse  (hWndCtrl, RED, 0, 0, sif.cif2.cr, sif.cif2.cr);
236 :     AjcG2dPixel    (hWndCtrl, RED, 0, 0, 3);
237 :
238 :     //  ２つの円中心からの法線描画
239 :     AjcG3dLineV  (hWndCtrl, BLUE, &sif.ls1.p1, &sif.ls1.p2);
240 :     AjcG3dLineV  (hWndCtrl, BLUE, &sif.ls2.p1, &sif.ls2.p2);
```

```
241 :
242 :     //  球と球の中心描画
243 :     AjcG3dSphereV(hWndCtrl, GRAY , &vc, r, r, r, 8, 8);
244 :     AjcG3dPixelV (hWndCtrl, GREEN, &vc, 3);
245 :
246 :     //  ２つの円を真横から見るための法線設定
247 :     AjcV3dSub(&vc            , &sif.cif1.lvc.p, &v1);       //  v1 = 球中心 →円１中心 ベクトル
248 :     AjcV3dSub(&sif.cif2.lvc.p, &sif.cif1.lvc.p, &v2);       //  v2 = 円１中心→円２中心 ベクトル
249 :     AjcV3dOuter(&v1, &v2, &vs);                             //  法線算出
250 :     AjcV3dNormal(&vs, &vs);
251 :     AjcV3dMult  (&vs, 0.5, &v1);
252 :
253 :     //  法線描画
254 :     AjcV3dAdd(&vs, &vc, &v1);
255 :     AjcG3dArrowV(hWndCtrl, YELLOW, &vc, &v1);
256 :
257 :     return TRUE;
258 : }
259 : //-----  「視点設定ボタン」ボタン ----------------------------------------------------------------------------------//
260 : AJC_DLGPROC(Main, IDC_CMD_ANGLE     )
261 : {
262 :     AjcG3dSetAngleV(hWndCtrl, &vs);
263 :     return TRUE;
264 : }
265 : //----- キャンセル ------------------------------------------------------------------------------------------------//
266 : AJC_DLGPROC(Main, IDCANCEL          )
267 : {
268 :     DestroyWindow(hDlg);
269 :     return TRUE;
270 : }
271 : //-----------------------------------------------------------------------------------------------------------------//
272 : AJC_DLGMAP_DEF(Main)
273 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG     )
274 :     AJC_DLGMAP_MSG(Main, WM_SIZE           )
275 :     AJC_DLGMAP_MSG(Main, WM_DESTROY        )
276 :     AJC_DLGMAP_CMD(Main, IDC_CMD_CIR       )
277 :     AJC_DLGMAP_CMD(Main, IDC_CMD_SPH       )
278 :     AJC_DLGMAP_CMD(Main, IDC_CMD_ANGLE     )
279 :     AJC_DLGMAP_CMD(Main, IDCANCEL          )
280 : AJC_DLGMAP_END
281 :
```

57. ３Ｄテストデータ生成

本ライブラリのサンプルプログラムで使用している３Ｄデータのテストデータ生成機能です。

球体の球面を進行するランダムなの座標データを生成します。

57.1. パラメタ

球面の座標データを生成するためのパラメタは、以下のとおりです。

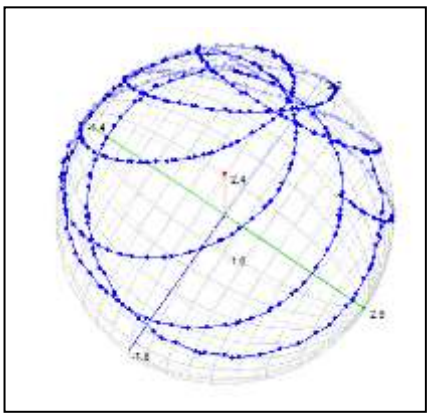
```
typedef struct {
    AJC3DVEC        vCent;              //  球中心（デフォルト＝(0, 0, 0)）
    double          radius;             //  球半径（デフォルト＝1.0）
    double          noise;              //  半径の誤差（±%，デフォルト＝±3.0%）
    double          xrot;               //  回転軸のＸ軸回転角度・最大値（０～最大値［度］，デフォルト＝0.5［度］）
    double          yrot;               //  回転軸のＹ軸回転角度・最大値（０～最大値［度］，デフォルト＝0.5［度］）
    double          pitch;              //  プロット点の回転角度・最大値（０～最大値［度］，デフォルト＝8.0［度］）
} AJCSPD_PARAM, *PAJCSPD_PARAM;
typedef const AJCSPD_PARAM *PCAJCSPD_PARAM;
```

球中心(vCent)と、球半径(radius)から、球面を移動する座標データ（プロット点）を生成します。
算出するプロット点は、noise で指定されたノイズ（半径の長さの誤差）を含みます。(生成する誤差は指定範囲内でランダム)

xrot と yrot は、プロット点の進行方向を変えるパラメタとなります。
プロット点は、回転軸により pitch で回転した方向へと進路を取ります。(回転角度は指定範囲内でランダム)
回転軸自体も、最初ランダムなベクトルとして初期化され、以降、Ｘ軸とＹ軸により回転します。(回転角度は指定範囲内でランダム)

プロット点は、９００回進行し、１００回その場にとどまる（但しノイズは加えられる）といった動作を繰り返します。

以上から、プロット点は以下のように球面を蛇行するような動きとなります。



57.2. サポートＡＰＩ

３Ｄテストデータ生成機能のサポートＡＰＩ一覧を以下に示します。

#	関 数 名	内容
1	AjcSpdCreate	インスタンスの生成
2	AjcSpdDelete	インスタンス消去
3	AjcSpdSetParam AjcSpdGetParam	パラメタ設定／取得
4	AjcSpdCalc	演算

57.2.1. インスタンス生成 (AjcSpdCreate)

形　式 ： HAJCSPD　AjcSpdCreate(UI seed);

引　数 ： seed – 乱数を生成するための初期値（0 以外を指定した場合、srand(seed)が実行されます）

説　明 ： ３Ｄテストデータ生成のインスタンスを生成します。
乱数系列の初期化は一元的なものなので、複数のインスタンスを生成する場合は、最初に一度だけ seed 値を指定し、その後は seed=0 を指定するようにしてください。

戻り値 ： ≠NULL – 成功（=インスタンスハンドル）
＝NULL – 失敗

57.2.2. インスタンス消去 (AjcSpdCreate)

形　式 ： BOOL　AjcSpdDelete (HAJCSPD hSpd);

引　数 ： hSpd　- インスタンスハンドル

説　明 ： ３Ｄテストデータ生成のインスタンスを消去します。

戻り値 ： TRUE　– 成功
FALSE – 失敗

57.2.3. パラメタ設定／取得 (AjcSpd{Set/Get}Param)

形　式 ： BOOL　AjcSpdSetParam(HAJCSPD hSpd, PCAJCSPD_PARAM pParam);　--- 設定
BOOL　AjcSpdGetParam(HAJCSPD hSpd, PAJCSPD_PARAM　pBuf); ------ 取得

引　数 ： hSpd　- インスタンスハンドル
pParam – 設定するパラメタのアドレス
pBuf　- 取得したパラメタを格納するバッファのアドレス

説　明 ： ３Ｄテストデータ生成のパラメタを設定／取得します。

戻り値 ： TRUE　– 成功
FALSE – 失敗

57.2.4. 演算 (AjcSpdCalc)

形　式 ： BOOL　AjcSpdCalc (HAJCSPD pW, double *x, double *y, double *z);
BOOL　AjcSpdCalcV(HAJCSPD pW, PAJC3DVEC pBuf);

引　数 ： hSpd　- インスタンスハンドル
x, y, z – 演算結果（各軸の座標値）を格納するバッファのアドレス
pBuf　- 演算結果（座標値）を格納するバッファのアドレス

説　明 ： １つ進めたプロットデータを取得します。

戻り値 ： TRUE　– 成功
FALSE – 失敗

58. バイトストリーム分離

一連のバイトストリームを、以下の４種類のデータに分離します。
（通常は、何らかの回線から受信した一連のバイトストリームから、データを取り出す為に使用します。）

テキスト	テキストデータ	
ＥＳＣコード	ESC ／ 英字／制御コード以外 ／ 英字	ESC=0x1B
制御コード	CTRL	CTRL=0x00～0x1F or 0x7F（但し、テキストデータに含むものは除く）
パケット・フレー	DLE ／ STX ／ パケットデータ ／ DLE ／ ETX	デフォルトでは、STX=0x02, ETX=0x03, DLE=0x10
パケット外データ	Byte	パケットフレーム以外の全データ（テキスト／ESC／制御コードの通知と重複）

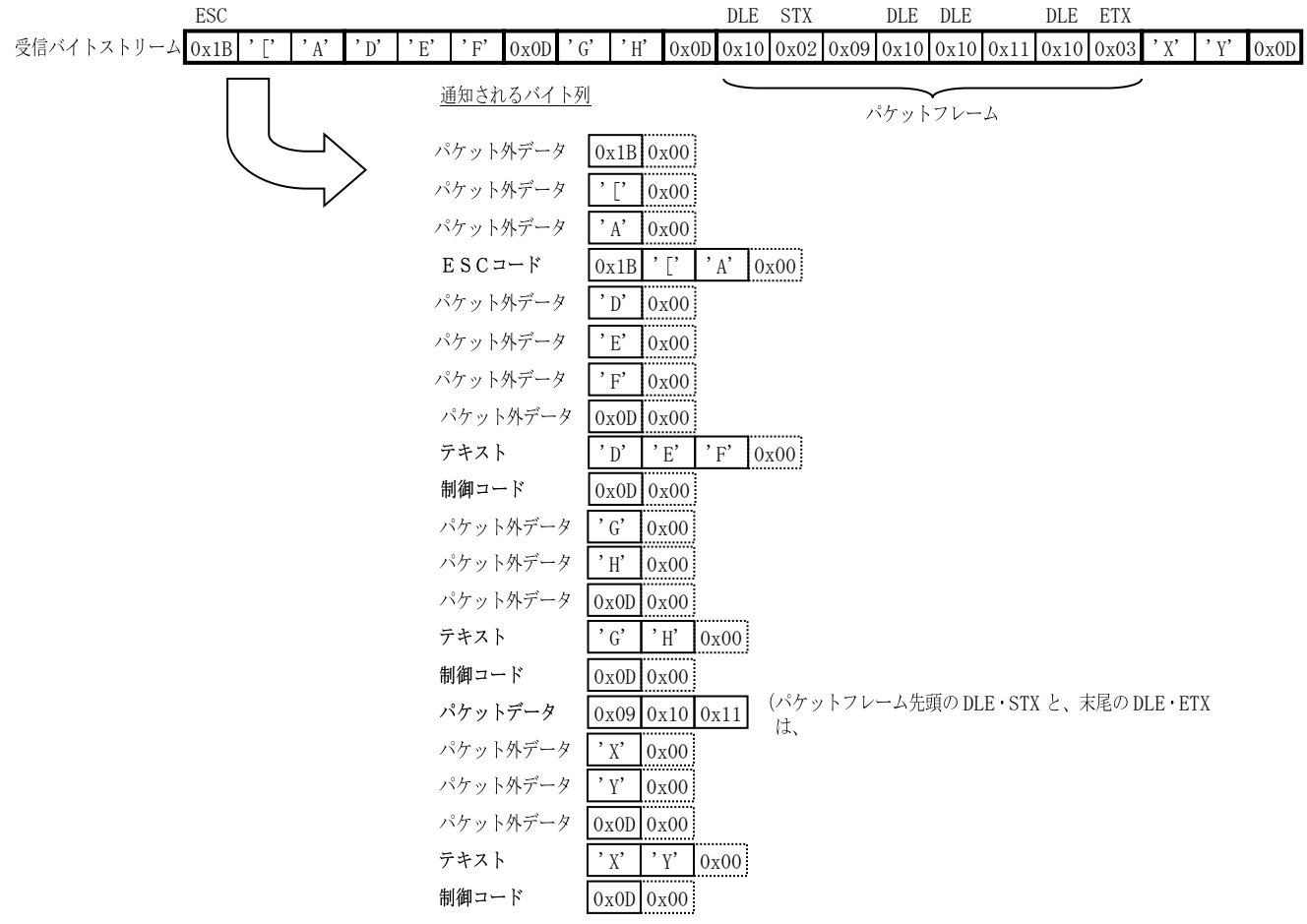
投与するバイトストリームに含まれる テキスト，ＥＳＣコードや制御コードは、マルチバイト文字(日本語の場合はS-JIS)，UTF-8 か日本語EUC でなければなりません。（ワイド文字（UNICODE)を投与した場合は、内部でマルチバイト文字(日本語の場合はS-JIS)に変換します）

テキストデータは、ＥＳＣコード，制御コードやパケットデータにより分離された部分バイト列です。
但し、テキストデータ中に制御コードを含むこともできます。（デフォルトでは、TAB(0x09)をテキストデータとして扱います）

ＥＳＣコードは、ESC(0x1B)で始まり、英字(A～Z／a～z)で終わる部分バイト列です。

パケット・フレームは、DLE・STX で始まり、DLE・ETX で終了する部分バイト列です。
STX,ETX や DLE の実際のコード値は、自由に設定可能です。（デフォルトでは、STX=0x02, ETX=0x03, DLE=0x10）
パケットデータ中の、２つの連続する DLE は、１つの DLE に変換されます。
例えば、（DLE=0x10 として）実際のパケットデータが「0x09, 0x10, 0x10, 0x11」の４バイトである場合、重複する２つの DLE は、１つの DLE に変換され、取得されるパケットデータは「0x09, 0x10, 0x11」の３バイトとなります。

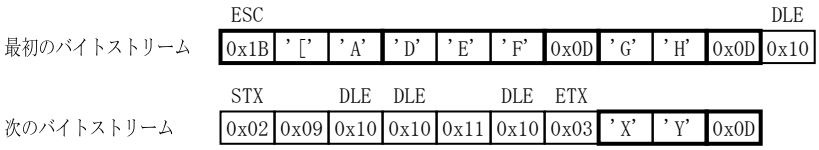
例えば、次のようなバイトストリームは、８つの部分バイト列と、パケット外データに分離されます。



※ 「パケット外データ」は、1 バイトづつ通知され、内容は、（パケットデータを除く）他の通知項目と重複します。

分離したデータ（テキスト／ＥＳＣコード／制御コード／パケット外データ）を通知する際は、末尾に文字列終端（0x00）が付加されます。

一連のバイトストリームは、複数回に分けて投与することができます。
例えば、次のようにバイトストリームを、２回に分けて投与しても、結果は同じになります。



58.1. サポートＡＰＩ

バイトストリーム分離のサポートＡＰＩ一覧を以下に示します。

#	関数名	内 容	備 考
1	AjcSsepCreate	インスタンス生成	
2	AjcSsepDelete	インスタンス消去	
3	AjcSsepPutData	バイトストリーム投与	
4	AjcSsepPutText	テキストデータ投与	
5	AjcSsepReset	リセット	
6	AjcSsepSetEvent AjcSsepGetEvent	生成するイベントコード設定／取得	
7	AjcSsepSetIncTxt	テキストに含める制御コードの設定	複数設定可能
8	AjcSsepSetStx　AjcSsepGetStx AjcSsepSetEtx　AjcSsepGetStx AjcSsepSetDle　AjcSsepGetStx	パケット・フレームを認識する為の制御コード値設定／取得	STX のコード値設定／取得 ETX のコード値設定／取得 DLE のコード値設定／取得
9	AjcSsepSetPktTimOut AjcSsepGetPktTimOut	パケットデータ受信時の、タイムアウト設定／取得	
10	AjcSsepMakePacket	パケット・フレーム・イメージ生成	
11	AjcSsepRelease	データメモリの開放	
12	AjcSsepPutMbc	文字コード判定用（半角）テキスト投与	
13	AjcSsepGetMbcKind	テキストの文字コード種別判定	
14	AjcSsepGetLastMbcKind	最後に判定した文字コード種別取得	
15	AjcSsepMbcReset	文字コード判定バッファ リセット	
16	AjcSsepEnableMultiThread	マルチスレッドの許可／禁止	

イベントコード

分離する（認識する）データ種別は、以下のイベントコードの組み合わせで指定します。

#	シンボル	値	内容	備考	
				通知データ(pDat)のタイプ	lDat
1	AJCSSEP_EV_TXT	0x01	テキスト	UTP	
2	AJCSSEP_EV_ESC	0x02	ＥＳＣコード	UTP	
3	AJCSSEP_EV_PKT	0x04	パケットデータ	UBP	
4	AJCSSEP_EV_CTRL	0x08	制御コード	UBP	
5	AJCSSEP_EV_NOPKT	0x10	パケット外データ	UBP	
6	AJCSSEP_EV_ALL	0x0F	上記#1～#4 の合成値	-	

イベントコードは、次のように定義されています。

```
#define  AJCSSEP_EV_TXT        0x01          //   テキストデータ通知
#define  AJCSSEP_EV_ESC        0x02          //   ＥＳＣコードデータ通知
#define  AJCSSEP_EV_PKT        0x04          //   パケットデータ通知
#define  AJCSSEP_EV_CTRL       0x08          //   制御コード通知
#define  AJCSSEP_EV_NOPKT      0x10          //   パケット外バイトデータ
#define  AJCSSEP_EV_ALL (AJCSSEP_EV_TXT | AJCSSEP_EV_ESC | AJCSSEP_EV_PKT | AJCSSEP_EV_CTRL)
```

58.1.1. インスタンス生成（AjcSsepCreate）

形　式 ： HAJCSSEP　AjcSsepCreate(UI evt, UX cbp,
BOOL (CALLBACK *cbEvent)(UI evt, VOP pDat, UI lDat, UX cbp))

引　数 ： evt　　- 生成するイベントコード（分離するデータ種別）
cbp　　- コールバックパラメタ
cbEvent　- 分離したデータ通知用コールバック関数

説　明 ： ストリーム分離用のインスタンスを生成します。
「evt」は、分離するデータ種別を、イベントコードの組み合わせで指定します。
例えば、ストリームをテキストと制御コードに分離する場合は、「AJCSSEP_EV_TXT | AJCSSEP_EV_CTRL」を指定します。

戻り値 ： ≠NULL - 成功（インスタンスハンドルを返します）
＝NULL - 失敗

コールバック：

cbEvent（分離／抽出データの通知）

形　式 ： BOOL CALLBACK *cbEvent*(UI evt, VOP pDat, UI lDat, UX cbp)

引　数 ： evt　- イベントコード（分離／抽出したデータの種別）
pDat　- 分離抽出したデータのアドレス
lDat　- 分離抽出したデータの文字数／バイト数
（パケットデータ時のみバイト数，その他はバイト数／文字数（終端(0x00)を含まない））
cbp　- コールバックパラメタ（AjcSsepCreate で指定した cbp）

説　明 ： 一連のバイトストリームから分離／抽出された部分バイト列を通知します。
pDat は、通知されたデータの先頭アドレスを示します。
lDat は、パケットデータの場合は、そのバイト数を示し、その他の場合は文字数を示します。
テキスト，ＥＳＣコードや制御コードは、ワイド文字バージョンの場合は、UNICODE でデータを通知します。
各引数の内容は、以下のとおりです。

#	evt	値	内容	pDat のタイプ	lDat
1	AJCSSEP_EV_TXT	0x01	テキスト	UTP	バイト数／文字数
2	AJCSSEP_EV_ESC	0x02	ＥＳＣコード	UTP	
3	AJCSSEP_EV_PKT	0x04	パケットデータ	UBP	バイト数
4	AJCSSEP_EV_CTRL	0x08	制御コード	BCP	1
5	AJCSSEP_EV_NOPKT	0x10	パケット外データ	UBP	1

戻り値 ： TRUE ： データメモリを開放する
FALSE ： データメモリを開放しない（この場合、後でAjcSsepRelease(pDat)を実行し、データメモリを開放しなければならない）

58.1.2. インスタンス消去（AjcSsepDelete）

形　式 ： VO　AjcSsepDelete(HAJCSSEP hSsep);

引　数 ： hSsep　　- インスタンスハンドル

説　明 ： AjcSsepCreate()で動的に確保したインスタンスワークを開放します。

戻り値 ： なし

58.1.3. データストリーム投与（AjcSsepPutData）

形　式 ： AJCSSEP_MODE AjcSsepPutData(HAJCSSEP hSsep, UBP pDat, UI lDat);

引　数 ： hSsep　　- インスタンスハンドル
pDat　　- 投与するデータストリームのアドレス
lDat　　- 投与するデータストリームのバイト数

説　明 ： バイトストリームを投与します。
その結果、AjcSsepCreate()やAjcSsepSetEvent()で指定されている種類のデータを検出した場合は、AjcSsepCreate()の cbEvent 引数で指定されたコールバック関数が呼び出され、当該データを通知します。

戻り値として、現在収集途中のストリーム種別が返されます。

戻り値 ： 収集中のストリーム種別
AJCSSEP_MD_IDLE(=0)　：アイドル状態（いずれのストリームも収集中でない）
AJCSSEP_MD_TXT　：テキストデータ収集中
AJCSSEP_MD_ESC　：ＥＳＣコードデータ収集中
AJCSSEP_MD_PKT　：パケットデータ収集中

備　考 ： 投与するバイトストリームは、テキストデータとパケットデータ（バイナリ）が混在する場合がありますのが、このＡＰＩで投与するテキスト部分はマルチバイト文字（シフトＪＩＳコード）でなければなりません。

58.1.4. テキストストリーム投与（AjcSsepPutText）

形　式 ： AJCSSEP_MODE AjcSsepPutText(HAJCSSEP hSsep, C_UTP pTxt, UI lTxt);

引　数 ： hSsep　　- インスタンスハンドル
pTxt　　- 投与するテキストストリームのアドレス
lTxt　　- 投与するテキストストリームの文字数（－1指定時は、自動算出）

説　明 ： テキストデータストリームを投与します。
その結果、AjcSsepCreate()やAjcSsepSetEvent()で指定されている種類のデータを検出した場合は、AjcSsepCreate()の cbEvent 引数で指定されたコールバック関数が呼び出され、当該データを通知します。

戻り値として、現在収集途中のストリーム種別が返されます。

戻り値 ： 収集中のストリーム種別
AJCSSEP_MD_IDLE(=0)　：アイドル状態（いずれのストリームも収集中でない）
AJCSSEP_MD_TXT　：テキストデータ収集中
AJCSSEP_MD_ESC　：ＥＳＣコードデータ収集中
AJCSSEP_MD_PKT　：パケットデータ収集中

備　考 ： 投与するデータが全てテキスト・データであることが分かっている場合、本ＡＰＩでテキストデータを投与します。

58.1.5. リセット（AjcSsepReset）

形　式 ： VO AjcSsepReset(HAJCSSEP hSsep);

引　数 ： hSsep　　- インスタンスハンドル
pDat　　- 投与するデータストリームのアドレス
lDat　　- 投与するデータストリームのバイト数

説　明 ： 現在認識途中のデータを破棄し、何も認識していない状態にします。

戻り値 ： なし

58.1.6. 生成するイベントコード設定／取得（AjcSsep{Set/Get}Event）

形　式 ： VO　AjcSsepSetEvent(HAJCSSEP hSsep, UI evt); -- 生成するイベントコード設定
UI　AjcSsepGetEvent(HAJCSSEP hSsep); ---------- 生成するイベントコード取得

引　数 ： hSsep　　- インスタンスハンドル
evt　　- 生成するイベントコード（分離するデータ種別）

説　明 ： 生成するイベント（分離するデータ種別）を設定／取得します。
「evt」は、分離するデータ種別を、イベントコードの組み合わせで指定します。
例えば、ストリームをテキストと制御コードに分離する場合は、「AJCSSEP_EV_TXT | AJCSSEP_EV_CTRL」を指定します。

生成するイベントコードを設定した場合は、現在認識途中のデータを破棄し、何も認識していない状態になります。

戻り値 ： 設定時：なし　　取得時：生成するイベントコード

58.1.7. テキストに含める制御コードの設定／取得（AjcSsepSetIncTxt, AjcSsep{Set/Get}IncTxtMap）

形　式 ： BOOL AjcSsepSetIncTxt(HAJCSSEP hSsep, UI num, ...)
BOOL AjcSsepSetIncTxtMap(HAJCSSEP pW, UI map);
UI　AjcSsepGetIncTxtMap(HAJCSSEP pW);

引　数 ： hSsep　　- インスタンスハンドル
num　　- 設定する制御コードの個数
...　　- 設定する制御コード群（0x00～0x1F or 0x7F）
map　　- テキストに含める制御コードマップ(Bit0=0x7F, Bit1～31=0x01～0x1F に対応する)

説　明 ： テキストデータに含める制御コードを設定／取得します。
AjcSsepSetIncTxt()では、設定する制御コード群は、int 型の数値で、0x00～0x1F あるいは 0x7F でなければなりません。
例えば、TAB(0x09)と、BS(0x08)をテキストデータに含める場合は AjcSsepSetIncTxt(hSsep, 2, 0x09, 0x08) とします。

AjcSsepSetIncTxtMap()では、テキストに含める制御コードをビットマップで指定します。
Bit0(0x00000001)が 0x7F に対応し、Bit1～Bit31 が 0x01～0x1F に対応します。
例えば、TAB(0x09)と、BS(0x08)をテキストデータに含める場合は AjcSsepSetIncTxtMap(hSsep, 0x00000300) とします。

AjcSsepSetIncTxtMap()は、テキストに含める制御コードのビットマップを取得します。

デフォルトでは、テキストデータに含める制御コードとして TAB(0x09)が設定されています。

戻り値 ： AjcSsepSetIncTxt/AjcSsepSetIncTxtMap　　AjcSsepGetIncTxtMap
TRUE　- 成功　　テキストに含める制御コードのビットマップ（エラー時は０）
FALSE – エラー

58.1.8. パケット・フレーム認識,制御ｺｰﾄﾞ値設定/取得(AjcSsep{Set/Get}Stx / Etx / Dle)

形　式 ： VO AjcSsepSetStx(HAJCSSEP hSsep, UI stx) ・・・・・ STX コード値設定
VO AjcSsepSetEtx(HAJCSSEP hSsep, UI etx) ・・・・・ ETX コード値設定
VO AjcSsepSetDle(HAJCSSEP hSsep, UI dle) ・・・・・ DLE コード値設定

UI AjcSsepGetStx(HAJCSSEP hSsep); ・・・・・・・・・ STX コード値取得
UI AjcSsepGetEtx(HAJCSSEP hSsep); ・・・・・・・・・ ETX コード値取得
UI AjcSsepGetDle(HAJCSSEP hSsep); ・・・・・・・・・ DLE コード値取得

引　数 ： hSsep - インスタンスハンドル
stx - STX のコード値
etx - ETX のコード値
dle - DLE のコード値

説　明 ： パケット・フレームを認識する為の制御コード（STX / ETX / DLE）のコード値を設定／取得します。
デフォルトでは、STX=0x02, ETX=0x03, DLE=0x10 となっています。

戻り値 ： なし

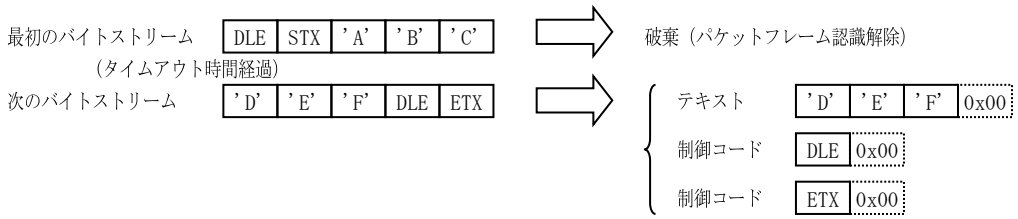
58.1.9. パケットフレームのタイムアウト値設定／取得（AjcSsep{Set/Get}PktTimOut）

形　式 ： VO AjcSsepSetPktTimeOut(HAJCSSEP hSsep, UI time);
UI AjcSsepGetPktTimeOut(HAJCSSEP hSsep);

引　数 ： hSsep - インスタンスハンドル
time - タイムアウト時間[ms]（0＝タイムアウトなし）

説　明 ： パケットフレーム認識中における、タイムアウト時間を設定／取得します。
パケットの受信が指定時間以内に完了しない場合は、タイムアウトとなり、認識中のパケットフレームを破棄し、何も認識していない状態にします。
time=0 とした場合は、タイムアウトを検出しません。（デフォルト）

例えば、２つのストリームを、タイムアウト時間を越えて投与した場合は、以下のようになります。



戻り値 ： なし

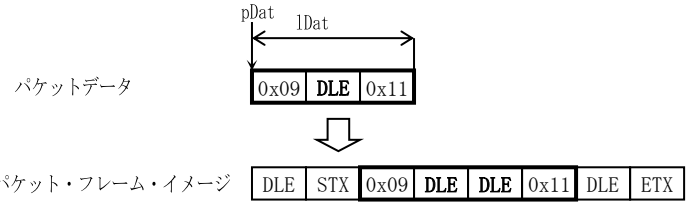
58.1.10. パケット・フレーム・イメージ生成（AjcSsepMakePacket）

形　式 ： VO　AjcSsepMakePacket(HAJCSSEP hSsep, C_VOP pDat, UI lDat, UIP pBytes)

引　数 ： hSsep　　- インスタンスハンドル
pDat　　- パケットデータのアドレス
lDat　　- パケットデータのバイト数
pBytes　- 生成したパケット・フレーム・イメージのバイト数を格納するバッファのアドレス

説　明 ： pDat,lDat で指定されたパケットデータから、パケット・フレーム・イメージを生成します。
つまり、パケットデータの先頭に DLE,STX を追加し、末尾に DLE,ETX を追加します。
また、パケットデータ中の DLE は、２つの DLE に変換します。

（例）



パケット・フレーム・イメージは、内部で動的に確保したバッファに作成されます。
この動的に確保したバッファを開放するにはは、AjcSsepRelease ()を実行します。

戻り値 ： ≠NULL - 生成したパケット・フレーム・イメージのアドレス
＝NULL - 失敗

58.1.11. データメモリ開放（AjcSsepReleasePacket）

形　式 ： VO　AjcSsepRelease (VOP pDat)

引　数 ： pDat　　- 開放するデータメモリのアドレス

説　明 ： コールバック関数で通知されたデータや、AjcSsepMakePacket()で生成した、パケット・フレーム・イメージを開放します。
コールバック関数で FALSE を返した場合や、AjcSsepMakePacket()でパケット・フレーム・イメージを生成した場合は、本関数で当該メモリを開放する必要があります。

戻り値 ： なし

58.1.12. 文字コード判定用テキスト投与（AjcSsepPutMbc）

形　式 ： VO　AjcSsepPutMbc (HAJCSSEP hSsep, C_BCP pTxt)

引　数 ： hSsep　　- インスタンスハンドル
pTxt　　- 投与するテキストデータのアドレス

説　明 ： 文字コード判定用のテキストデータを蓄積します。

戻り値 ： なし

58.1.13. テキストの文字コード種別判定（AjcSsepGetMbcKind）

形　式 ： AJCMBCKIND　AjcSsepGetMbcKind (HAJCSSEP hSsep)

引　数 ： hSsep　　- インスタンスハンドル

説　明 ： AjcSsepPutMbc()により蓄積されたテキストデータから、文字コードを判定します。
文字コードが判定不能である場合は、前回の判定結果を返します。
蓄積されたテキストデータが無い場合は、デフォルト値(AJCMBC_SJIS)を返します。

戻り値 ： 文字コード判定結果 (AJCMBC_SJIS, AJCMBC_UTF8 or AJCMBC_EUC)

58.1.14. 最後に判定した文字コード種別取得（AjcSsepGetLastKind）

形　式 ： AJCMBCKIND　AjcSsepGetLastMbcKind (HAJCSSEP hSsep)

引　数 ： hSsep　　- インスタンスハンドル

説　明 ： 前回の AjcSsepGetMbcKind()による判定結果を返します。
AjcSsepGetMbcKind()が実行されていない場合は、デフォルト値(AJCMBC_SJIS)を返します。

戻り値 ： 文字コード判定結果 (AJCMBC_SJIS, AJCMBC_UTF8 or AJCMBC_EUC)

58.1.15. 文字コード判定バッファ リセット（AjcSsepMbcReset）

形　式 ： BOOL　AjcSsepMbcReset (HAJCSSEP hSsep, C_BCP pTxt)

引　数 ： hSsep　　- インスタンスハンドル

説　明 ： AjcSsepPutMbc()により蓄積された、文字コード判定用テキストデータを破棄します、
AjcSsepGetMbcKind()が実行されていない場合は、デフォルト値(AJCMBC_SJIS)を返します。

戻り値 ：

58.1.16. マルチスレッドの許可／禁止（AjcSsepEnableMultiThread）

形　式 ： BOOL　AjcSsepEnableMultiThread(HAJCRNG hRng, BOOL fEnable);

引　数 ： hRng　　- インスタンスハンドル
fEnable　- TRUE　：複数のスレッドからのアクセスを許可
　　FALSE ：複数のスレッドからのアクセスを禁止

説　明 ： fEnable=TRUE とした場合、複数のスレッドによる、インスタンスへのアクセスを可能にします。
この場合、各関数の入り口と出口で、クリティカルセクションによるスレッド間の排他制御を行います。
但し、次の関数は、(fEnable=TRUE としても）マルチスレッドでの排他制御を行いません。

・AjcSsepCreate（インスタンス生成）
・AjcSsepDelete（インスタンス消去）
・AjcSsepEnableMultiThread（本関数）

fEnable=FALSE（デフォルト）とした場合は、マルチスレッドでの排他制御を行いません。

戻り値 ： 前回の許可／禁止状態

58.2. サンプルプログラム

58.2.1. SW_SSepC（バイトストリーム分離）

このサンプルプログラムでは、単に２つのバイトストリームを投与し、通知されたデータをダンプ表示します。
実行結果は、以下のとおりです。

```
ESC  : "¥x1B[A"
TXT  : "DEF"
CTRL : "¥x0D"
TXT  : "GH"
CTRL : "¥x0D"
PKT  :  09 10 11
TXT  : "XY"
CTRL : "¥x0D"
```

```
 1 : //
 2 : // SW_SSepC.c
 3 : //
 4 : #include    <AjrCstXX.h>
 5 : #include    <stdio.h>
 6 : #include    <tchar.h>
 7 :
 8 : //----- インスタンスハンドル ------------------------------------------//
 9 : static  HAJCSSEP    hSsep;
10 : //----- 制御コード ----------------------------------------------------//
11 : #define STX 0x02
12 : #define ETX 0x03
13 : #define DLE 0x10
14 : #define ESC 0x1B
15 : //----- バイトストリームデータ ----------------------------------------//
16 : static  UB          Data1[] = {0x1B, '[' , 'A' , 'D' , 'E' , 'F' , 0x0D, 'G' , 'H' , 0x0D, DLE};
17 : static  UB          Data2[] = {STX , 0x09, DLE , DLE , 0x11, DLE , ETX , 'X' , 'Y' , 0x0D};
18 :
19 : //-----------------------------------------------------------------------------------------------------//
20 : //  テキストデータのダンプ表示                                                                         //
21 : //-----------------------------------------------------------------------------------------------------//
22 : static  VO  DumpText(UTP pDat, UI lDat)
23 : {
24 :     AjcPrintF(TEXT("¥""));
25 :     while (lDat--) {
26 :         if (_istcntrl(*pDat)) AjcPrintF(TEXT("¥¥x%02X"), *pDat);
27 :         else                  AjcPrintF(TEXT("%c"), *pDat);
28 :         pDat++;
29 :     }
30 :     AjcPrintF(TEXT("¥""));
31 : }
32 : //-----------------------------------------------------------------------------------------------------//
33 : //  パケットデータのダンプ表示                                                                         //
34 : //-----------------------------------------------------------------------------------------------------//
35 : static  VO  DumpPacket(UBP pDat, UI lDat)
36 : {
37 :     while (lDat--) {
38 :         AjcPrintF(TEXT(" %02X"), *pDat++);
39 :     }
40 : }
41 : //-----------------------------------------------------------------------------------------------------//
42 : //  イベント通知用コールバック関数                                                                     //
43 : //-----------------------------------------------------------------------------------------------------//
44 : static  BOOL  CALLBACK cbEvent(UI evt, VOP pDat, UI lDat, UX cbp)
45 : {
46 :     switch (evt) {
47 :         case AJCSSEP_EV_TXT:                    //  ●テキスト通知
48 :             AjcPrintF(TEXT("TXT  : "));
49 :             DumpText((UTP)pDat, lDat);
50 :             AjcPrintF(TEXT("¥n"));
51 :             break;
52 :
53 :         case AJCSSEP_EV_ESC:                    //  ●ＥＳＣコード通知
54 :             AjcPrintF(TEXT("ESC  : "));
55 :             DumpText((UTP)pDat, lDat);
56 :             AjcPrintF(TEXT("¥n"));
57 :             break;
58 :
59 :         case AJCSSEP_EV_CTRL:                   //  ●制御コード通知
60 :             AjcPrintF(TEXT("CTRL : "));
61 :             DumpText((UTP)pDat, lDat);
```

```
62 :             AjcPrintF(TEXT("¥n"));
63 :             break;
64 :
65 :         case AJCSSEP_EV_PKT:                    //  ●パケットデータ通知
66 :             AjcPrintF(TEXT("PKT  : "));
67 :             DumpPacket((UBP)pDat, lDat);
68 :             AjcPrintF(TEXT("¥n"));
69 :             break;
70 :
71 :     }
72 :     return TRUE;    // TRUE:データメモリ開放
73 : }
74 : //-------------------------------------------------------------------------------------------------------------------//
75 : //  m a i n                                                                                                          //
76 : //-------------------------------------------------------------------------------------------------------------------//
77 : int  AjcMain(int argc, UTP argv[])
78 : {
79 :     AjcSetStdoutMode();
80 :
81 :     hSsep = AjcSsepCreate(AJCSSEP_EV_ALL, 0, cbEvent);          //  インスタンス生成
82 :     AjcSsepPutData(hSsep, Data1, sizeof Data1);                 //  バイトストリーム１投与
83 :     AjcSsepPutData(hSsep, Data2, sizeof Data2);                 //  バイトストリーム２投与
84 :     AjcSsepDelete(hSsep);                                       //  インスタンス消去
85 :
86 :     getchar();
87 :     return 0;
88 : }
```

59. 状態遷移制御

状態遷移制御とは、「状態」と「イベント」の２次元表による制御を意味します。
「Ｂ状態」で、「Ｃイベント」が発生した場合は、「Ｘ処理」を実行し、「Ｙ状態」へ遷移する、といった制御を行います。

例えば、通信プログラム風に言い直せば、「ＡＣＫ受信待ち状態」で「ＣＡＮ受信イベント」が発生した場合は、「送信保留データの開放処理」を実行し、「送信要求待ち状態」へ遷移する。といった風になります。

タイマについて

本状態遷移制御では、内部で非表示ウインドを生成し、タイマの起動／停止を Win-API の SetTimer()/KillTimer()により行っています。
本状態遷移制御を、コンソール・アプリケーションのようなメッセージループの無いプログラムで使用する場合は、メッセージループ処理を行って、WM_TIMER メッセージを非表示ウインドに配信するようにしなければなりません。

59.1. 状態遷移表の記述方法

状態遷移表を記述するには、まず、３つの要素「状態番号」「イベント番号」「ファンクション番号」の定義を行います。
以下に、３つの要素の定義例を示します。

状態番号の定義

```
enum {
    AJCSTC_STS_IDLE     ,       //  0: 待機状態
    AJCSTC_STS_ACTIVE   ,       //  1: タイマ動作中
    AJCSTC_STS_RINGING  ,       //  2: 鳴音中

    STS_NUM                     //  状態数
};
```

※ 各状態名の先頭は「AJCSTC_STS_」でなければなりません

イベント番号の定義

```
enum {
    EVT_START   ,   //  0: ＳＴＡＲＴボタン押下
    EVT_TMO_T0  ,   //  1: タイムアウト監視タイマ(t0)タイムアウト
    EVT_TMO_T1  ,   //  2: キッチンタイマ(t1)タイムアウト
    EVT_TMO_T2  ,   //  3: 鳴音インタバルタイマ(t2)タイムアウト
    EVT_OVER    ,   //  4: 鳴音回数満了
    EVT_STOP    ,   //  5: ＳＴＯＰボタン押下

    EVT_NUM         //  イベント数
};
```

※ 各イベント名は、任意です。

ファンクション番号の定義

```
enum {
    AJCSTC_PF_CNT_INI   ,       //  鳴音回数初期化
    AJCSTC_PF_SET_TIM   ,       //  キッチンタイマ時間設定
    AJCSTC_PF_DSP_LAP   ,       //  経過時間表示
    AJCSTC_PF_SOUND     ,       //  鳴音
    AJCSTC_PF_QUIET     ,       //  鳴音停止
    AJCSTC_PF_SWTIME    ,       //  経過時間表示
    AJCSTC_PF_ENASRT    ,       //  スタートボタン許可
    AJCSTC_PF_DISSRT    ,       //  スタートボタン禁止
    AJCSTC_PF_ENASTP    ,       //  ストップボタン許可
    AJCSTC_PF_DISSTP    ,       //  ストップボタン禁止

    PF_NUM                      //  プリセットファンクション数
};
```

※ 各ファンクション名の先頭は「AJCSTC_PF_」でなければなりません

本ライブラリでは、Ｃ言語のマクロ機能を駆使し、状態遷移表そのものをコーディングします。

状態遷移表の記述マクロ一覧

#	マクロ名	内　容	備　考
1	AJCSM	状態遷移先の記述マクロ	
2	AJCTM	タイマ動作の記述マクロ（４ヶのタイマ）	
3	AJCTM6	〃　（６ヶのタイマ）	
4	AJCTM8	〃　（８ヶのタイマ）	
5	AJCPF	ファンクションの記述マクロ	

状態遷移表は、「AJCSM」「AJCTM」「AJCPF」マクロにより、以下のように記述します。

```
#define TM  AJCTM
#define SM  AJCSM
#define PF  AJCPF

static  const AJCSTC_EVT_NF  EvtTbl[ EVT_NUM ][ STS_NUM ] = {
//----- [ 0] - ＳＴＡＲＴボタン押下 ------------------------------------------------------------------------//
//              NextState        t0  t1  t2  t3      Pre-set functions
/* 0:IDLE    */{{SM( ACTIVE  ), TM( C , S , _ , _), PF( CNT_INI , SET_TIM , DSP_LAP , __      , __      , __      )},
/* 1:ACTIVE  */ {SM( __      ), TM( _ , _ , _ , _), PF( __      , __      , __      , __      , __      , __      )},
/* 2:RINGING */ {SM( __      ), TM( _ , _ , _ , _), PF( __      , __      , __      , __      , __      , __      )}},

//----- [ 1] - 監視タイマ(t0)タイムアウト ------------------------------------------------------------------//
//              NextState        t0  t1  t2  t3      Pre-set functions
/* 0:IDLE    */{{SM( __      ), TM( _ , _ , _ , _), PF( __      , __      , __      , __      , __      , __      )},
/* 1:ACTIVE  */ {SM( __      ), TM( _ , _ , _ , _), PF( DSP_LAP , __      , __      , __      , __      , __      )},
/* 2:RINGING */ {SM( __      ), TM( _ , _ , _ , _), PF( __      , __      , __      , __      , __      , __      )}},

                .
                .
                .

//----- [ 5] - ＳＴＯＰボタン押下 ------------------------------------------------------------------------//
//              NextState        t0  t1  t2  t3      Pre-set functions
/* 0:IDLE    */{{SM( __      ), TM( _ , _ , _ , _), PF( __      , __      , __      , __      , __      , __      )},
/* 1:ACTIVE  */ {SM( IDLE    ), TM( _ , _ , _ , _), PF( __      , __      , __      , __      , __      , __      )},
/* 2:RINGING */ {SM( IDLE    ), TM( _ , _ , _ , _), PF( __      , __      , __      , __      , __      , __      )}},
};
```

AJCSM マクロでは、状態遷移先の名称（先に定義した状態名の「AJCSTC_STS_」を除く部分）を指定します。
状態遷移を行わない場合は、「__ 」（２ヶのアンダバー）を指定します。
AJCTM マクロでは、各４つのタイマの動作を指定します。(S:スタート(単発), C:スタート(サイクリック), R:ストップ, _ : 無操作)
「S:スタート(単発)」は、タイムアウト発生時にタイマを停止します。
「C:スタート(サイクリック)」はタイムアウトが発生してもタイマを停止せずに、「R:ストップ」でタイマを停止するまでタイマは動き続けます。
６ヶのタイマを扱う場合は、AJCTM6 マクロを、８ヶのタイマを扱う場合は、AJCTM8 マクロを使用します。
AJCPF マクロでは、実行するファンクション名（先に定義したファンクション名の「AJCSTC_PF_」を除く部分）を指定します。
実際のファンクションの処理内容はユーザが作成します。(本ライブラリでは、コールバックによりファンクション番号を渡します)
ファンクションを実行しない場合は、「__ 」（２ヶのアンダバー）を指定します。

例えば、上記例の１行目（網掛け部分）は、「IDLE 状態」で、「START ボタン押下イベント」が発生したら、以下の動作を行います。

- CNT_INI：カウンタ初期化初期化
- SET_TIM：タイマ設定
- DSP_LAP：経過時間表示
- タイマ０(t0)をサイクリック起動
- タイマ１(t1)を単発起動

59.2. サポートＡＰＩ

状態遷移制御のサポートＡＰＩ一覧を以下に示します。

#	関数名	内　容	備　考
1	AjcStcCreate	インスタンス生成	
2	AjcStcDelete	インスタンス消去	
3	AjcStcSetTimerInfo	タイマ情報設定	
4	AjcStcTimerStart	タイマ・スタート	
5	AjcStcTimerStop	タイマ・ストップ	
6	AjcStcGetCurrentState	現在の状態番号設定／取得	
7	AjcStcExecEvent	イベント実行	
8	AjcStcPurge	保留中の全イベント破棄	

59.2.1. インスタンス生成（AjcStcCreate）

形　式 ： HAJCSTC AjcStcCreate(PCAJCSTC_STSACT pCom, PCAJCSTC_STSACT pRel, PCAJCSTC_EVT_NF pEvt,
int nState, UX cbp,
VO (CALLBACK *cbPsf)(UI pno, int sts, UX cbp),
VO (CALLBACK *cbNtc)(AJCSTC_NTC ntc, UL p1, UL p2, UX cbp));

引　数 ： pCom　- 自状態へ遷移する場合のアクションテーブルのアドレス（不要時はNULL）
pRel　- 自状態を離れる場合のアクションテーブルのアドレス　（不要時はNULL）
pEvt　- 状態遷移表のアドレス
nState - 状態番号の個数
cbp　- コールバックパラメタ
cbPsf　- ファンクションハンドラ関数
cbNtc　- 状態遷移制御状況を通知する為のコールバック関数（不要時はNULL）

説　明 ： 状態遷移制御のインスタンスを作成し、初期化します。
pEvtは、状態遷移表のアドレスを指定します。（前述の例では「EvtTbl」を指定します）
nStateは状態番号の個数を指定します（前述の例では「STS_NUM」を指定します）
cbpは、cbPsfやcbNtcコールバック時に渡すパラメタを指定します。
cbPsfは、ファンクション実行用コールバック関数を指定します。
cbNtcは、状態遷移制御状況を通知する為のコールバック関数を指定します。
cbNtcは、状態の遷移状況やタイマの起動／停止をモニタする等、主にデバッグ用途で使用します。

戻り値 ： ≠NULL - 正常（インスタンスハンドル）
＝NULL - 失敗

コールバック： コールバック関数の仕様は、以下のとおりです。

cbPsf（ファンクションハンドラ関数）

形　式 ： VO *cbPsf*(UI pno, int sts, UX cbp);

引　数 ： pno　- ファンクション番号（ AJCSTC_PF_*XXXXX* ）
sts　- 現在の状態番号（ AJCSTC_STS_*XXXXX* ）
cbp　- コールバックパラメタ

説　明 ： 状態遷移表で記述されたファンクションの実行を行います。

戻り値 ： なし

cbNtc（状態遷移制御状況・通知用コールバック関数）

形　式 ： VO *cbNtc*(AJCSTC_NTC ntc, UL p1, UL p2, UX cbp)

引　数 ： ntc　- 通知コード
p1, p2 - 通知パラメタ
cbp　- コールバックパラメタ

説　明 ： このコールバック関数は、イベント実行時，状態が遷移した場合や、タイマの起動／停止を行った場合にコールされます。通知コードと、通知パラメタの対応は、以下のとおりです。

通知コード(ntc)	p1	p2
AJCSTC_NTC_EVT　（イベント実行通知）	状態番号（AJCSTC_STS_*XXXXX*）	イベント番号（0～）
AJCSTC_NTC_STS　（状態変化通知）	遷移先状態番号（AJCSTC_STS_*XXXXX*）	－
AJCSTC_NTC_TIMSRT（タイマスタート通知）	タイマ番号（0～7）	タイマ値［ms］
AJCSTC_NTC_TIMSTP（タイマストップ通知）	タイマ番号（0～7）	－

戻り値 ： なし

59.2.2. インスタンス消去（AjcStcDelete）

形　式 ： VO AjcStcDelete(HAJCSTC hSTc);

引　数 ： hStc　- インスタンスハンドル

説　明 ： AjcStcCreate()で生成したインスタンスを開放します。

戻り値 ： なし

59.2.3. タイマ情報設定（AjcStcSetTimerInfo）

形　式 ： BOOL AjcStcSetTimerInfo(HAJCSTC hSTc, UI tid, UI msTime, int EvtCode);

引　数 ： hStc　- インスタンスハンドル
tid　- タイマ番号（０～７）
msTime - タイマ値[ms]
EvtCode - タイムアウト・イベント番号（０～）

説　明 ： tid で指定したタイマに対する、タイマ情報を設定します。
本関数は、タイマの情報を設定するだけであり、タイマの起動や停止は行いません。

戻り値 ： TRUE - 正常
FALSE - 失敗

59.2.4. タイマ・スタート（AjcStcTimerStart）

形　式 ： BOOL AjcStcTimerStart (HAJCSTC hSTc, UI tid);

引　数 ： hStc　- インスタンスハンドル
tid　- タイマ番号（０～７）

説　明 ： タイマを起動します。
通常、タイマの起動や停止は状態遷移表の AJCTM マクロで記述しますが、本関数でもタイマを起動することができます。
タイマが起動されてから、（タイマを停止せずに）msTime で指定された時間が経過すると、ファンクションハンドラがコールバックされ、AjcStcSetTimerInfo()の EvtCode で設定されたイベント番号が通知されます。

戻り値 ： TRUE - 正常
FALSE - 失敗

59.2.5. タイマ・ストップ（AjcStcTimerStop）

形　式 ： BOOL AjcStcTimerStop (HAJCSTC hSTc, UI tid);

引　数 ： hStc　- インスタンスハンドル
tid　- タイマ番号（０～７）

説　明 ： タイマを停止します。
通常、タイマの起動や停止は状態遷移表の AJCTM マクロで記述しますが、本関数でもタイマを停止することができます。

戻り値 ： TRUE - 正常
FALSE - 失敗

59.2.6. 現在の状態番号設定／取得（AjcStcSetCurrentState）

形　式 ： BOOL AjcStcSetCurrentState(HAJCSTC hSTc, int sts); --- 設定
int AjcStcGetCurrentState(HAJCSTC hSTc); ---------- 取得

引　数 ： hStc　- インスタンスハンドル
sts　- 設定する状態番号

説　明 ： 現在の状態番号を設定／取得します。
AjcStcCreate()でインスタンス生成時には、状態番号はゼロに初期化されます。
開始時の状態番号がゼロ以外である場合は、AjcStcCreate()実行後に、状態番号の設定を行う必要があります。

戻り値 ： 設定時：TRUE - 正常　　取得時：現在の状態番号（ AJCSTC_STS_*XXXXX* ）
FALSE - エラー（不正な状態番号）

59.2.7. イベントの実行（AjcStcExecEvent）

形　式 ： int AjcStcExecEvent(HAJCSTC hSTc, int evt);

引　数 ： hStc　- インスタンスハンドル
evt　- イベント番号（０～）

説　明 ： 状態遷移表の記述に従って、evt で指定されたイベントを実行します。
ファンクションハンドラ以外から本関数がコールされた場合は、当該イベントを実行し、遷移先の状態番号を返します。
ファンクションハンドラ内から本関数がコールされた場合は、イベントは実行せずに保留され、－１を返します。
保留されたイベントは、現在のイベント処理が終了した時点で、順次実行されます。
保留できるイベントの最大数は６３です。

戻り値 ： 下記以外 - 遷移先状態番号（イベント実行後の状態番号）
－１　- イベントは実行されずに保留された
－２　- エラー（最大イベント保留数オーバー）

59.2.8. 保留中の全イベント破棄（AjcStcPurge）

形　式 ： VO AjcStcPurge(HAJCSTC hSTc);

引　数 ： hStc　- インスタンスハンドル

説　明 ： 保留中の全イベントを破棄します。

戻り値 ： なし

59.3. サンプルプログラム

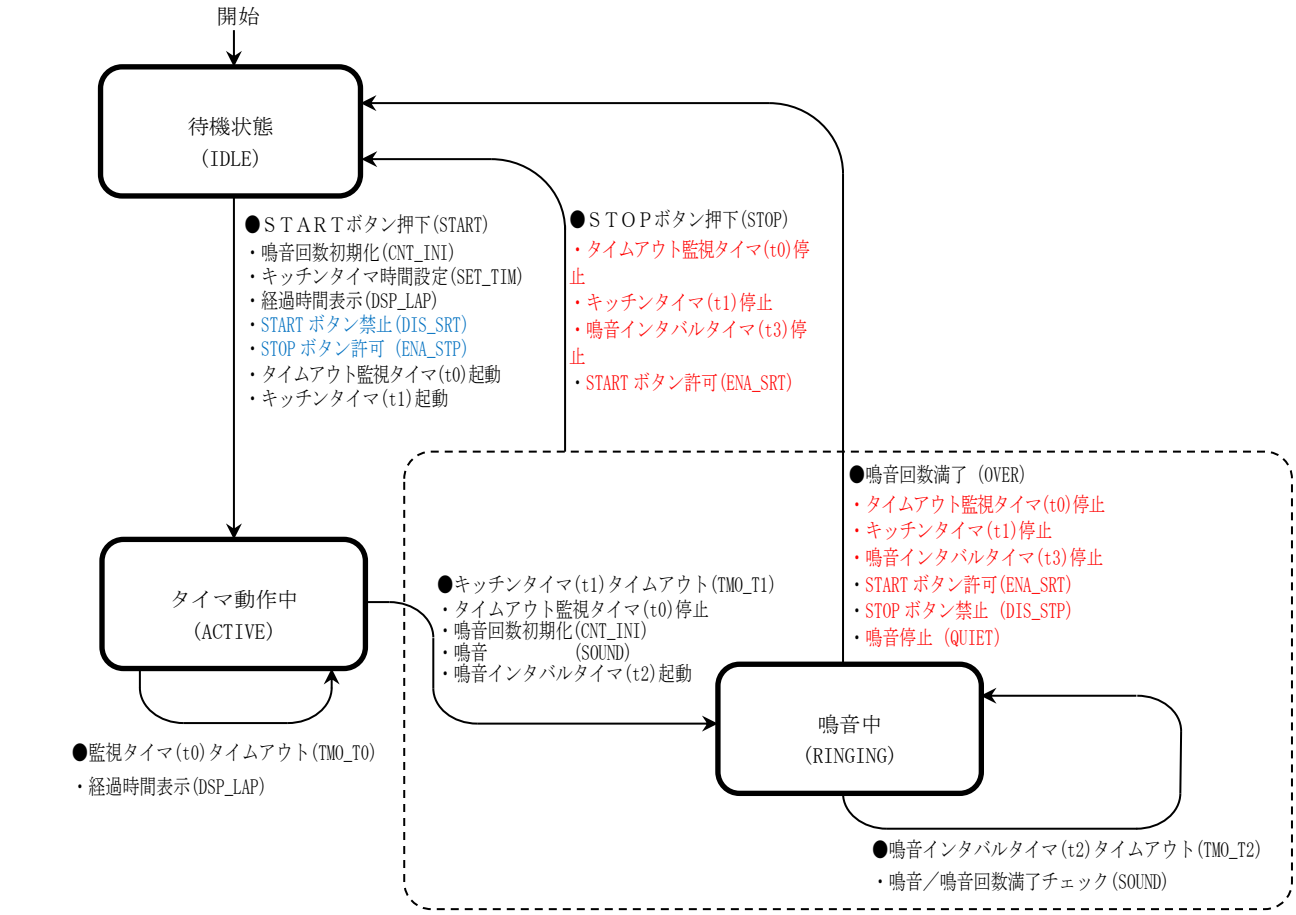
59.3.1. S_StateCtrl（キッチンタイマ）

次のサンプルプログラムでは、キッチンタイマの処理を行います。
タイマを設定し、START ボタンを押すと設定した時間経過時に１０秒間隔で３回鳴音を発生します。
STOP ボタンを押すとタイマを停止します。



本サンプルプログラムでの状態遷移は、以下のとおりです。

状態遷移図



状態遷移表

#	状態 / イベント	0:待機状態(IDLE)	1:タイマ動作中(AVTIVE)	2:鳴音中(RINGING)
0	ＳＴＡＲＴボタン押下 (EVT_START)	・鳴音回数初期化(CNT_INI) ・キッチンタイマ時間設定(SET_TIM) ・経過時間表示(DSP_LAP) ・START ボタン禁止(DIS_SRT) ・STOP ボタン許可 (ENA_STP) ・タイムアウト監視タイマ(t0)起動 ・キッチンタイマ(t1)起動	－	－
		→ 1:タイマ動作中(AVTIVE)	－	－
1	監視タイマ(t0) タイムアウト (EVT_TMO_T0)	－	・経過時間表示(DSP_LAP)	－
		－	－	－
2	キッチンタイマ(t1) タイムアウト (EVT_TMO_T1)	－	・タイムアウト監視タイマ(t0)停止 ・鳴音回数初期化(CNT_INI) ・鳴音　　(SOUND) ・鳴音インタバルタイマ(t2)起動	－
		－	→ 2:鳴音中(RINGING)	－
3	鳴音インタバルタイマ(t2) タイムアウト (EVT_TMO_T2)	－	－	・鳴音／鳴音回数満了チェック (SOUND)
		－	－	－
4	鳴音回数満了 (EVT_OVER)	－	－	・タイムアウト監視タイマ(t0)停止 ・キッチンタイマ(t1)停止 ・鳴音インタバルタイマ(t3)停止 ・START ボタン許可(ENA_SRT) ・STOP ボタン禁止 (DIS_STP) ・鳴音停止 (QUIET)
		－	－	→ 0:待機状態(IDLE)
5	ＳＴＯＰボタン押下 (EVT_STOP)	－	・タイムアウト監視タイマ(t0)停止 ・キッチンタイマ(t1)停止 ・鳴音インタバルタイマ(t3)停止 ・START ボタン許可(ENA_SRT) ・STOP ボタン禁止 (DIS_STP) ・鳴音停止 (QUIET)	・タイムアウト監視タイマ(t0)停止 ・キッチンタイマ(t1)停止 ・鳴音インタバルタイマ(t3)停止 ・START ボタン許可(ENA_SRT) ・STOP ボタン禁止 (DIS_STP) ・鳴音停止 (QUIET)
		－	→ 0:待機状態(IDLE)	→ 0:待機状態(IDLE)

※ 青字の部分は、「アイドル状態から他の状態へ遷移する」場合の共通処理としてくくりだします。
これは、サンプルコード中の「ComTbl[]」で定義されている部分に相当します。

※ 赤字の部分は、「他の状態から IDLE 状態へ遷移する」場合の共通処理としてくくりだします。
これは、サンプルコード中の「RelTbl[]」で定義されている部分に相当します。

```
 1 : //
 2 : //  SW_StateCtrl.c
 3 : //
 4 :
 5 : #include    <AjrCstXX.h>
 6 : #include    <math.h>
 7 : #include    <tchar.h>
 8 : #include    <mmsystem.h>
 9 :
10 : #include    "resource.h"
11 :
12 : #define     SOUND_TIMES     3       //  鳴音回数
13 :
14 : //-------------------------------------------------------------------------------------------------------------//
15 : //  ワーク                                                                                                     //
16 : //-------------------------------------------------------------------------------------------------------------//
17 : static  HINSTANCE       hInst;              //  ＤＬＬインスタンスハンドル
```

```
18 : static  HWND           hDlgMain;          //  ダイアログボックスハンドル
19 : static  HWND           hVth;              //  ＶＴＨウインド
20 : static  HAJCSTC        hStc;              //  状態遷移制御ハンドル
21 : static  UI             CntSound;          //  鳴音回数カウンタ
22 : static  UI             SrtTick;           //  開始時刻
23 : static  UI             msTimer;           //  設定時間[ms]
24 : static  UI             svSec;             //  経過秒数退避
25 : static  HGLOBAL        hWavData;          //  Wav データリソース
26 : static  VOP            pWavData;
27 :
28 : //-------------------------------------------------------------------------------------------------------------------//
29 : //  内部サブ関数                                                                                                     //
30 : //-------------------------------------------------------------------------------------------------------------------//
31 : AJC_DLGPROC_DEF(Main);
32 : static  VO      Log(C_UTP pFmt, ...);
33 :
34 : //-------------------------------------------------------------------------------------------------------------------//
35 : //  状態                                                                                                             //
36 : //-------------------------------------------------------------------------------------------------------------------//
37 : enum {
38 :     AJCSTC_STS_IDLE     ,           //  0: 待機状態
39 :     AJCSTC_STS_ACTIVE   ,           //  1: タイマ動作中
40 :     AJCSTC_STS_RINGING  ,           //  2: 鳴音中
41 :
42 :     STS_NUM                         //  状態数
43 : };
44 :
45 : //-------------------------------------------------------------------------------------------------------------------//
46 : //  イベント                                                                                                         //
47 : //-------------------------------------------------------------------------------------------------------------------//
48 : enum {
49 :     EVT_START   ,                   //  0: ＳＴＡＲＴボタン押下
50 :     EVT_TMO_T0  ,                   //  1: タイムアウト監視タイマ(t0)タイムアウト
51 :     EVT_TMO_T1  ,                   //  2: キッチンタイマ(t1)タイムアウト
52 :     EVT_TMO_T2  ,                   //  3: 鳴音インタバルタイマ(t2)タイムアウト
53 :     EVT_OVER    ,                   //  4: 鳴音回数満了
54 :     EVT_STOP    ,                   //  5: ＳＴＯＰボタン押下
55 :
56 :     EVT_NUM                         //  イベント数
57 : };
58 : //-------------------------------------------------------------------------------------------------------------------//
59 : //  プリセットファンクション                                                                                         //
60 : //-------------------------------------------------------------------------------------------------------------------//
61 : enum {
62 :     AJCSTC_PF_CNT_INI   ,           //  鳴音回数初期化
63 :     AJCSTC_PF_SET_TIM   ,           //  キッチンタイマ時間設定
64 :     AJCSTC_PF_DSP_LAP   ,           //  経過時間表示
65 :     AJCSTC_PF_SOUND     ,           //  鳴音／鳴音回数満了チェック
66 :     AJCSTC_PF_QUIET     ,           //  鳴音停止
67 :     AJCSTC_PF_ENA_SRT   ,           //  スタートボタン許可
68 :     AJCSTC_PF_DIS_SRT   ,           //  スタートボタン禁止
69 :     AJCSTC_PF_ENA_STP   ,           //  ストップボタン許可
70 :     AJCSTC_PF_DIS_STP   ,           //  ストップボタン禁止
71 :
72 :     PF_NUM                          //  プリセットファンクション数
73 : };
74 :
75 : #define TM  AJCTM
76 : #define SM  AJCSM
77 : #define PF  AJCPF
78 :
79 : //-------------------------------------------------------------------------------------------------------------------//
80 : //  他の状態→自状態へ遷移時（IDLE 状態へ遷移時、START ボタン許可, STOP ボタン禁止, 鳴音停止, 全ﾀｲﾏ停止）          //
81 : //-------------------------------------------------------------------------------------------------------------------//
82 : static const AJCSTC_STSACT ComTbl[STS_NUM] = {
83 : //                  t0  t1  t2  t3      Pre-set functions
84 : /* 0:IDLE    */ { TM( R , R , R , _), PF( ENA_SRT , DIS_STP , QUIET   , __      , __      , __      )},
85 : /* 1:ACTIVE  */ { TM( _ , _ , _ , _), PF( __      , __      , __      , __      , __      , __      )},
86 : /* 2:RINGING */ { TM( _ , _ , _ , _), PF( __      , __      , __      , __      , __      , __      )},
87 : };
88 : //-------------------------------------------------------------------------------------------------------------------//
89 : //  自状態→他の状態へ遷移時（IDLE 状態から他の状態へ遷移時、START ボタン禁止, STOP ボタン許可）                    //
90 : //-------------------------------------------------------------------------------------------------------------------//
91 : static const AJCSTC_STSACT RelTbl[STS_NUM] = {
92 : //                  t0  t1  t2  t3      Pre-set functions
93 : /* 0:IDLE    */ { TM( _ , _ , _ , _), PF( DIS_SRT , ENA_STP , __      , __      , __      , __      )},
94 : /* 1:ACTIVE  */ { TM( _ , _ , _ , _), PF( __      , __      , __      , __      , __      , __      )},
95 : /* 2:RINGING */ { TM( _ , _ , _ , _), PF( __      , __      , __      , __      , __      , __      )},
96 : };
97 : //-------------------------------------------------------------------------------------------------------------------//
```

```
 98 : //  状態遷移表                                                                                                    //
 99 : //-------------------------------------------------------------------------------------------------------------//
100 : static  const AJCSTC_EVT_NF  EvtTbl[ EVT_NUM ][ STS_NUM ] = {
101 : //----- [ 0] - ＳＴＡＲＴボタン押下 ------------------------------------------------------------------------//
102 : //              NextState       t0  t1  t2  t3      Pre-set functions
103 : /* 0:IDLE    */{{SM( ACTIVE  ), TM( C , S , _ , _), PF( CNT_INI , SET_TIM , DSP_LAP , __      , __      , __      )},
104 : /* 1:ACTIVE  */ {SM( __      ), TM( _ , _ , _ , _), PF( __      , __      , __      , __      , __      , __      )},
105 : /* 2:RINGING */ {SM( __      ), TM( _ , _ , _ , _), PF( __      , __      , __      , __      , __      , __      )}},
106 :
107 : //----- [ 1] - 監視タイマ(t0)タイムアウト ------------------------------------------------------------------//
108 : //              NextState       t0  t1  t2  t3      Pre-set functions
109 : /* 0:IDLE    */{{SM( __      ), TM( _ , _ , _ , _), PF( __      , __      , __      , __      , __      , __      )},
110 : /* 1:ACTIVE  */ {SM( __      ), TM( _ , _ , _ , _), PF( DSP_LAP , __      , __      , __      , __      , __      )},
111 : /* 2:RINGING */ {SM( __      ), TM( _ , _ , _ , _), PF( __      , __      , __      , __      , __      , __      )}},
112 :
113 : //----- [ 2] - キッチンタイマ(t1)タイムアウト ---------------------------------------------------------------//
114 : //              NextState       t0  t1  t2  t3      Pre-set functions
115 : /* 0:IDLE    */{{SM( __      ), TM( _ , _ , _ , _), PF( __      , __      , __      , __      , __      , __      )},
116 : /* 1:ACTIVE  */ {SM( RINGING ), TM( R , _ , C , _), PF( DSP_LAP , SOUND   , __      , __      , __      , __      )},
117 : /* 2:RINGING */ {SM( __      ), TM( _ , _ , _ , _), PF( __      , __      , __      , __      , __      , __      )}},
118 :
119 : //----- [ 3] - 鳴音インタバルタイマ(t2)タイムアウト ---------------------------------------------------------//
120 : //              NextState       t0  t1  t2  t3      Pre-set functions
121 : /* 0:IDLE    */{{SM( __      ), TM( _ , _ , _ , _), PF( __      , __      , __      , __      , __      , __      )},
122 : /* 1:ACTIVE  */ {SM( __      ), TM( _ , _ , _ , _), PF( __      , __      , __      , __      , __      , __      )},
123 : /* 2:RINGING */ {SM( __      ), TM( _ , _ , _ , _), PF( __      , SOUND   , __      , __      , __      , __      )}},
124 :
125 : //----- [ 4] - 鳴音回数満了 --------------------------------------------------------------------------------//
126 : //              NextState       t0  t1  t2  t3      Pre-set functions
127 : /* 0:IDLE    */{{SM( __      ), TM( _ , _ , _ , _), PF( __      , __      , __      , __      , __      , __      )},
128 : /* 1:ACTIVE  */ {SM( __      ), TM( _ , _ , _ , _), PF( __      , __      , __      , __      , __      , __      )},
129 : /* 2:RINGING */ {SM( IDLE    ), TM( _ , _ , _ , _), PF( __      , __      , __      , __      , __      , __      )}},
130 :
131 : //----- [ 5] - ＳＴＯＰボタン押下 --------------------------------------------------------------------------//
132 : //              NextState       t0  t1  t2  t3      Pre-set functions
133 : /* 0:IDLE    */{{SM( __      ), TM( _ , _ , _ , _), PF( __      , __      , __      , __      , __      , __      )},
134 : /* 1:ACTIVE  */ {SM( IDLE    ), TM( _ , _ , _ , _), PF( __      , __      , __      , __      , __      , __      )},
135 : /* 2:RINGING */ {SM( IDLE    ), TM( _ , _ , _ , _), PF( __      , __      , __      , __      , __      , __      )}},
136 : };
137 : //-------------------------------------------------------------------------------------------------------------//
138 : //  プリセットファンクション実行  コールバック関数                                                              //
139 : //                                                                                                             //
140 : //  引  数  :  pno    - プリセットファンクション番号                                                           //
141 : //             sts    - 現在の状態コード                                                                       //
142 : //             cbp    - コールバックパラメタ (未使用)                                                          //
143 : //                                                                                                             //
144 : //  戻り値  :  なし                                                                                            //
145 : //-------------------------------------------------------------------------------------------------------------//
146 : static VO CALLBACK cbPsf(UI pno, int sts, UX cbp)
147 : {
148 :     switch (pno) {
149 :         //---------------------------------------------------------------------------------------------------//
150 :         case AJCSTC_PF_CNT_INI  :           //  鳴音回数初期化
151 :             //  ログ表示
152 :             Log(TEXT("   Func: CNT_INI    - 鳴音回数初期化¥n"));
153 :             //  鳴音回数カウンタクリアー
154 :             CntSound = 0;
155 :             break;
156 :         //---------------------------------------------------------------------------------------------------//
157 :         case AJCSTC_PF_SET_TIM  :           //  キッチンタイマ時間設定
158 :         {   UI  tm = AjcGetDlgItemUInt(hDlgMain, IDC_INP_MINUTE) * 60000 + AjcGetDlgItemUInt(hDlgMain, IDC_INP_SECOND) * 1000;
159 :             //  ログ表示
160 :             Log(TEXT("   Func: SET_TIM    - キッチンタイマ時間設定(t1=%dms)¥n"), tm);
161 :             //  タイマ情報設定
162 :             SrtTick = GetTickCount();
163 :             AjcStcSetTimerInfo(hStc, 1, tm, EVT_TMO_T1);
164 :             break;
165 :         }
166 :         //---------------------------------------------------------------------------------------------------//
167 :         case AJCSTC_PF_DSP_LAP  :           //  経過時間表示
168 :         {   UI  tm, second, minute;
169 :             UT  txt[128];
170 :             //  ログ表示 (このログは頻繁に出力されるため割愛する)
171 :             //  Log(TEXT("   Func: DSP_LAP    - 経過時間表示¥n"));
172 :             //  経過時間表示
173 :             tm = GetTickCount() - SrtTick;
174 :             second = tm / 1000;
175 :             minute = second / 60;
176 :             if (second == 0 || second != svSec) {
177 :                 AjcSnPrintF(txt, AJCTSIZE(txt), TEXT("%02d : %02d"), minute, second % 60);
```

```
178 :                 AjcSetDlgItemStr(hDlgMain, IDC_TXT_LAP, txt);
179 :             }
180 :             svSec = second;
181 :             break;
182 :         }
183 :         //-----------------------------------------------------------------------------------------------------//
184 :         case AJCSTC_PF_SOUND    :           //  鳴音
185 :         {   UT  txt[128];
186 :             //  鳴音回数更新
187 :             CntSound++;
188 :             //  ログ表示
189 :             AjcSnPrintF(txt, AJCTSIZE(txt), TEXT("鳴音(%d)"), CntSound);
190 :             Log(TEXT("   Func: SOUND     - %s¥n"), txt);
191 :             //  鳴音回数未了
192 :             if (CntSound <= SOUND_TIMES) {
193 :                 RECT    r;
194 :                 int     w, h;
195 :                 SIZE    sz;
196 :                 //  ウインド中央にツールチップで「鳴音」を表示
197 :                 GetWindowRect(hDlgMain, &r);
198 :                 w = r.right - r.left;
199 :                 h = r.bottom - r.top;
200 :                 AjcTipTextGetSize(txt, NULL, TRUE, &sz);
201 :                 AjcTipTextShow(r.left + (w - sz.cx) / 2, r.top + (h - sz.cy) / 2, txt, 1000, NULL);
202 :                 //  ビープ音
203 :                 PlaySound((UTP)pWavData, NULL, SND_MEMORY | SND_ASYNC);
204 :             }
205 :             //  鳴音回数満了
206 :             else {
207 :                 AjcStcExecEvent(hStc, EVT_OVER);
208 :             }
209 :             break;
210 :         }
211 :         //-----------------------------------------------------------------------------------------------------//
212 :         case AJCSTC_PF_QUIET    :           //  鳴音停止
213 :             //  ログ表示
214 :             Log(TEXT("   Func: QUIET     - 鳴音停止¥n"));
215 :             //  ツールチップ消去
216 :             AjcTipTextHide();
217 :             //  鳴音停止
218 :             PlaySound(NULL, NULL, 0);
219 :             break;
220 :         //-----------------------------------------------------------------------------------------------------//
221 :         case AJCSTC_PF_ENA_SRT  :           //  スタートボタン許可
222 :             //  ログ表示
223 :             Log(TEXT("   Func: ENA_SRT   - スタートボタン許可¥n"));
224 :             //  スタートボタン許可
225 :             AjcEnableDlgGroup(hDlgMain, IDC_GRP_START, TRUE, TRUE);
226 :             break;
227 :         //-----------------------------------------------------------------------------------------------------//
228 :         case AJCSTC_PF_DIS_SRT  :           //  スタートボタン禁止
229 :             //  ログ表示
230 :             Log(TEXT("   Func: DIS_SRT   - スタートボタン禁止¥n"));
231 :             //  スタートボタン禁止
232 :             AjcEnableDlgGroup(hDlgMain, IDC_GRP_START, FALSE, FALSE);
233 :             break;
234 :         //-----------------------------------------------------------------------------------------------------//
235 :         case AJCSTC_PF_ENA_STP  :           //  ストップボタン許可
236 :             //  ログ表示
237 :             Log(TEXT("   Func: ENA_STP   - ストップボタン許可¥n"));
238 :             //  ストップボタン許可
239 :             AjcEnableDlgItem(hDlgMain, IDC_CMD_STOP, TRUE);
240 :             break;
241 :         //-----------------------------------------------------------------------------------------------------//
242 :         case AJCSTC_PF_DIS_STP  :           //  ストップボタン禁止
243 :             //  ログ表示
244 :             Log(TEXT("   Func: DIS_STP   - ストップボタン禁止¥n"));
245 :             //  ストップボタン禁止
246 :             AjcEnableDlgItem(hDlgMain, IDC_CMD_STOP, FALSE);
247 :             break;
248 :     }
249 : }
250 : //-----------------------------------------------------------------------------------------------------//
251 : //  状態遷移通知  コールバック関数                                                                     //
252 : //                                                                                                     //
253 : //  引  数  :  ntc    - 通知コード                                                                     //
254 : //             p1, p2 - 通知パラメタ                                                                   //
255 : //             cbp    - コールバックパラメタ (未使用)                                                  //
256 : //                                                                                                     //
257 : //  戻り値  :  なし                                                                                    //
```

```
258 : //-------------------------------------------------------------------------------------------------------------//
259 : static C_UTP    pSts[STS_NUM] = {_T("IDLE   (待機状態)    "),
260 :                                 _T("ACTIVE (タイマ動作中)"),
261 :                                 _T("RINGING(鳴音中)      "),
262 :                 };
263 : static C_UTP    pEvt[EVT_NUM] = {_T("START (ＳＴＡＲＴボタン押下)") ,
264 :                                 _T("TMO_T0(タイムアウト監視タイマ(t0)タイムアウト)"),
265 :                                 _T("TMO_T1(キッチンタイマ(t1)タイムアウト)"),
266 :                                 _T("TMO_T2(鳴音インタバルタイマ(t2)タイムアウト)"),
267 :                                 _T("OVER  (鳴音回数満了)"),
268 :                                 _T("STOP  (ＳＴＯＰボタン押下)"),
269 :                 };
270 :
271 : static  VO CALLBACK cbNtc(AJCSTC_NTC ntc, UL p1, UL p2, UX cbp)
272 : {
273 :     switch (ntc) {
274 :         case AJCSTC_NTC_EVT:    Log(TEXT("状態 = %s, イベント = %s¥n" ), pSts[p1], pEvt[p2]);  break;
275 :         case AJCSTC_NTC_STS:    Log(TEXT("   状態 遷移 --> %s¥n"      ), pSts[p1]          );  break;
276 :         case AJCSTC_NTC_TIMSRT: Log(TEXT("   タイマ %d START (%dms)¥n"), p1, p2           );  break;
277 :         case AJCSTC_NTC_TIMSTP: Log(TEXT("   タイマ %d STOP¥n"        ), p1               );  break;
278 :     }
279 : }
280 :
281 : //=============================================================================================================//
282 : //                                                                                                             //
283 : //  ＷｉｎＭａｉｎ                                                                                            //
284 : //                                                                                                             //
285 : //=============================================================================================================//
286 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
287 : {
288 :     MSG     msg;
289 :
290 :     hInst = hInstance;
291 :
292 :     //  メイン・ダイアログオープン
293 :     hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_DLGMAIN), NULL, AJC_DLGPROC_NAME(Main));
294 :     //  ダイアログ表示
295 :     ShowWindow(hDlgMain, SW_SHOW);
296 :
297 :     //  メッセージループ
298 :     while (GetMessage(&msg, NULL, 0, 0)) {
299 :         do {
300 :             if (IsDialogMessage(hDlgMain, &msg)) break;
301 :             TranslateMessage(&msg);
302 :             DispatchMessage (&msg);
303 :         } while (0);
304 :     }
305 :
306 :     return (int)msg.wParam ;
307 : }
308 : //=============================================================================================================//
309 : //                                                                                                             //
310 : //  ダイアログ・プロシージャ                                                                                   //
311 : //                                                                                                             //
312 : //=============================================================================================================//
313 : //----- ダイアログ初期化 ------------------------------------------------------------------------------------//
314 : AJC_DLGPROC(Main, WM_INITDIALOG     )
315 : {
316 :     hDlgMain = hDlg;
317 :     hVth     = GetDlgItem(hDlgMain, IDC_VTH);
318 :
319 :     //  Wav リソースロード
320 :     hWavData = LoadResource(hInst,
321 :                     FindResource(hInst, MAKEINTRESOURCE(IDR_WAV1), TEXT("WAV")));
322 :     pWavData = LockResource(hWavData);
323 :
324 :     //  チップテキストのフォント設定
325 :     AjcTipTextSetDefFont((HFONT)SendMessage(hDlg, WM_GETFONT, 0, 0));
326 :
327 :     //  状態遷移制御インスタンス生成
328 :     hStc = AjcStcCreate(ComTbl, RelTbl, EvtTbl[0], STS_NUM, 0, cbPsf, cbNtc);
329 :     //  タイマにイベントを関連付け
330 :     AjcStcSetTimerInfo(hStc, 0,  100, EVT_TMO_T0);  //  0: 監視タイマ
331 :     AjcStcSetTimerInfo(hStc, 2, 5000, EVT_TMO_T2);  //  2: 鳴音インタバルタイマ
332 :
333 :     return TRUE;
334 : }
335 : //----- ウインド破棄 ----------------------------------------------------------------------------------------//
336 : AJC_DLGPROC(Main, WM_DESTROY        )
337 : {
```

```
338 :     //      状態遷移制御インスタンス解放
339 :     AjcStcDelete(hStc);
340 :     //  WAV リソース解放
341 :     FreeResource(hWavData);
342 :     //  プログラム終了
343 :     PostQuitMessage(0);
344 :     return TRUE;
345 : }
346 : //----- START ボタン ----------------------------------------------------------------------------------------//
347 : AJC_DLGPROC(Main, IDC_CMD_START     )
348 : {
349 :     //  ログクリアー
350 :     AjcVthClear(hVth);
351 :     //  スタートボタン押下イベント
352 :     AjcStcExecEvent(hStc, EVT_START);
353 :     return TRUE;
354 : }
355 : //----- STOP ボタン -----------------------------------------------------------------------------------------//
356 : AJC_DLGPROC(Main, IDC_CMD_STOP      )
357 : {
358 :     //  ストップボタン押下イベント
359 :     AjcStcExecEvent(hStc, EVT_STOP);
360 :     return TRUE;
361 : }
362 : //----- ウインドクローズ ------------------------------------------------------------------------------------//
363 : AJC_DLGPROC(Main, IDCANCEL          )
364 : {
365 :     //  ストップボタン押下イベント
366 :     AjcStcExecEvent(hStc, EVT_STOP);
367 :     //  ウインドクローズ
368 :     DestroyWindow(hDlg);
369 :     return TRUE;
370 : }
371 : //-----------------------------------------------------------------------------------------------------------//
372 : AJC_DLGMAP_DEF(Main)
373 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG     )
374 :     AJC_DLGMAP_MSG(Main, WM_DESTROY        )
375 :     AJC_DLGMAP_CMD(Main, IDC_CMD_START     )
376 :     AJC_DLGMAP_CMD(Main, IDC_CMD_STOP      )
377 :     AJC_DLGMAP_CMD(Main, IDCANCEL          )
378 : AJC_DLGMAP_END
379 : //-----------------------------------------------------------------------------------------------------------//
380 : //  ログ出力                                                                                                 //
381 : //-----------------------------------------------------------------------------------------------------------//
382 : static  VO      Log(C_UTP pFmt, ...)
383 : {
384 :     static  BOOL fNeedLF = FALSE;
385 :     static  UT   svTxt[1024];
386 :     va_list      vls;
387 :     UT           txt[1024] = {0};
388 :
389 :
390 :     va_start(vls, pFmt);
391 :     AjcVSnPrintF(txt, 1024, pFmt, vls);
392 :     txt[1023] = 0;
393 :     va_end  (vls);
394 :
395 :     AjcVthTimeStamp(hVth);
396 :     AjcVthPutText(hVth, TEXT(" "), -1);
397 :     if (MAjcStrCmp(svTxt, txt) != 0) {
398 :         if (fNeedLF) {
399 :             AjcVthPutText(hVth, TEXT("¥n"), -1);
400 :             fNeedLF = FALSE;
401 :         }
402 :         AjcVthPutText(hVth, TEXT(" "), -1);
403 :         AjcVthPutText(hVth, txt, -1);
404 :         MAjcStrCpy(svTxt, AJCTSIZE(svTxt), txt);
405 :     }
406 :     else {
407 :         AjcVthPutText(hVth, TEXT("・・・同上・・・¥r"), -1);
408 :     }
409 : }
410 :
```

60. ＣＲＣ／チェックサム

ＣＲＣ／チェックサムの計算，設定，チェック関数群です。

ＣＲＣは、以下の２とおりの生成多項式と、さらに左送り演算／右送り演算の、４とおりの演算をサポートします。

ＣＣＩＴＴ　Ｘ．２５　：　$X^{16}+X^{12}+X^{5}+1$
ＡＮＳＩ－ＣＲＣ１６　：　$X^{16}+X^{15}+X^{2}+1$

60.1. サポートＡＰＩ

ＣＲＣ／チェックサムのサポートＡＰＩ一覧を以下に示します。

ＣＲＣ操作

#	内容	関 数 名（ＣＣＩＴＴ　Ｘ．２５）		関 数 名（ＡＮＳＩ－ＣＲＣ１６）	
		（左送り）	（右送り）	（左送り）	（右送り）
1	部分ＣＲＣ算出	AjcPartCrcItL	AjcPartCrcItR	AjcPartCrc16L	AjcPartCrc16R
2	ＣＲＣ値算出	AjcBlkCrcItL	AjcBlkCrcItR	AjcBlkCrc16L	AjcBlkCrc16R
3	ＸＭＯＤＥＭ／ＦＣＳ算出	AjcBlkXMODEM	AjcBlkFCS	－	－
4	ＣＲＣ値設定	AjcSetCrcItL	AjcSetCrcItR	AjcSetCrc16L	AjcSetCrc16R
	（Big Endian）	AjcSetCrcItL_BE	AjcSetCrcItR_BE	AjcSetCrc16L_BE	AjcSetCrc16R_BE
	（Little Endian）	AjcSetCrcItL_LE	AjcSetCrcItR_LE	AjcSetCrc16L_LE	AjcSetCrc16R_LE
5	ＸＭＯＤＥＭ／ＦＣＳ設定	AjcSetXMODEM	AjcSetFCS	－	－
	（Big Endian）	AjcSetXMODEM_BE	AjcSetFCS_BE	－	－
	（Little Endian）	AjcSetXMODEM_LE	AjcSetFCS_LE	－	－
6	ＣＲＣチェック	AjcChkCrcItL	AjcChkCrcItR	AjcChkCrc16L	AjcChkCrc16R
	（Big Endian）	AjcChkCrcItL_BE	AjcChkCrcItR_BE	AjcChkCrc16L_BE	AjcChkCrc16R_BE
	（Little Endian）	AjcChkCrcItL_LE	AjcChkCrcItR_LE	AjcChkCrc16L_LE	AjcChkCrc16R_LE
7	ＸＭＯＤＥＭ／ＦＣＳチェック	AjcChkXMODEM	AjcChkFCS	－	－
	（Big Endian）	AjcChkXMODEM_BE	AJcChkFCS_BE	－	－
	（Little Endian）	AjcChkXMODEM_LE	AjcChkFCS_LE	－	－

チェックサム操作

#	内容	関 数 名		
		バイトサム	ワードサム	ワードサム（逆エンディアン）
8	サム値算出（単純加算）	AjcCalcByteSum	AjcCalcWordSum	AjcCalcWordSumR
	（１の補数）	AjcCalcByteSumN	AjcCalcWordSumN	AjcCalcWordSumNR
	（２の補数）	AjcCalcByteSumS	AjcCalcWordSumS	AjcCalcWordSumSR
	（排他論理和）	AjcCalcByteXumS	AjcCalcWordSumX	AjcCalcWordSumXR
9	サム値設定（単純加算）	AjcSetByteSum	AjcSetWordSum	AjcSetWordSumR
	（１の補数）	AjcSetByteSumN	AjcSetWordSumN	AjcSetWordSumNR
	（２の補数）	AjcSetByteSumS	AjcSetWordSumS	AjcSetWOrdSumSR
	（排他論理和）	AjcSetByteSumX	AjcSetWordSumX	AjcSetWOrdSumXR
10	サム値検査（単純加算）	AjcChkByteSum	AjcChkWordSum	AjcChkWordSumR
	（１の補数）	AjcChkByteSumN	AJcChkWordSumN	AjcChkWordSumNR
	（２の補数）	AjcChkByteSumS	AjcChkWordSumS	AjcChkWordSumSR
	（排他論理和）	AjcChkByteSumX	AjcChkWordSumX	AjcChkWordSumXR

60.1.1. 部分ＣＲＣ算出（AjcPartCrc??L / AjcPartCrc??R）

形　式 ： UW AjcPartCrcItL (C_VOP pRec, UI len, UW ini); --- CCITT 方式, 左送り演算
UW AjcPartCrcItR (C_VOP pRec, UI len, UW ini); --- CCITT 方式, 右送り演算

UW AjcPartCrc16L (C_VOP pRec, UI len, UW ini); --- ANSI-CRC16, 左送り演算
UW AjcPartCrc16R (C_VOP pRec, UI len, UW ini); --- ANSI-CRC16, 右送り演算

引　数 ： pRec - 部分ＣＲＣを算出するデータブロックの先頭アドレス
len - 部分ＣＲＣを算出するデータブロックのバイト数
ini - 前回までの算出結果（本関数の戻り値）を指定する
但し、初回の場合は、0x0000（反転なし）あるいは、0xFFFF（反転操作）を指定する。

説　明 ： この関数は、データブロックのＣＲＣ算出を複数回に分割して算出する場合に使用します。
通常、初回の ini=0xFFFF（反転操作）を指定した場合は、最終的な算出値をさらに反転する必要があります。

戻り値 ： 部分データブロックのＣＲＣ値（失敗時は「ini」の値をそのまま返します）

60.1.2. ＣＲＣ算出（AjcBlkCrc??L / AjcBlkCrc??R）

形　式 ： UW AjcBlkCrcItL (C_VOP pRec, UI len, UW ini); --- CCITT 方式, 左送り演算
UW AjcBlkCrcItR (C_VOP pRec, UI len, UW ini); --- CCITT 方式, 右送り演算

UW AjcBlkCrc16L (C_VOP pRec, UI len, UW ini); --- ANSI-CRC16, 左送り演算
UW AjcBlkCrc16R (C_VOP pRec, UI len, UW ini); --- ANSI-CRC16, 右送り演算

引　数 ： pRec - ＣＲＣを算出するデータブロックの先頭アドレス
len - ＣＲＣを算出するデータブロックのバイト数
ini - 0x0000（反転なし）あるいは、0xFFFF（反転操作）を指定する。

説　明 ： データブロックのＣＲＣ値を算出します。

戻り値 ： データブロックのＣＲＣ値（失敗時は０を返します）

60.1.3. ＸＭＯＤＥＭ－ＣＲＣ／ＦＣＳ算出（AjcBlkXMODEM / AjcBlkCalcFCS）

形　式 ： UW AjcBlkXMODEM (VOP pRec, UI len); --- 左送り演算で、XMODEM 用の CRC 算出
UW AjcBlkFCS (VOP pRec, UI len); --- 右送り演算で、FCS (CCITT 方式の CRC) 算出

引　数 ： pRec - ＣＲＣを算出するデータブロックの先頭アドレス
len - ＣＲＣを算出するデータブロックのバイト数

説　明 ： データブロックのＸＭＯＤＥＭ用ＣＲＣ値／ＦＣＳを算出します。

戻り値 ： データブロックのＸＭＯＤＥＭ用ＣＲＣ値／ＦＣＳ（失敗時は０を返します）

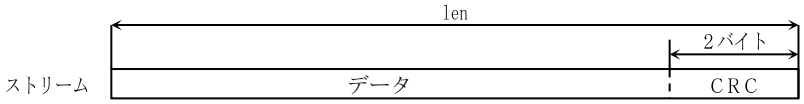
60.1.4. ストリームへＣＲＣ値設定（AjcSetCrc?? ・・・）

形　式 ： BOOL AjcSetCrcItL (VOP pRec, UI len, UW ini); ── CCITT 方式, 左送り演算, CPU 依存のエンディアン形式で設定
BOOL AjcSetCrcItL_BE (VOP pRec, UI len, UW ini); ── CCITT 方式, 左送り演算, ビッグエンディアン形式で設定
BOOL AjcSetCrcItL_LE (VOP pRec, UI len, UW ini); ── CCITT 方式, 左送り演算, リトルエンディアン形式で設定
BOOL AjcSetCrcItR (VOP pRec, UI len, UW ini); ── CCITT 方式, 右送り演算, CPU 依存のエンディアン形式で設定
BOOL AjcSetCrcItR_BE (VOP pRec, UI len, UW ini); ── CCITT 方式, 右送り演算, ビッグエンディアン形式で設定
BOOL AjcSetCrcItR_LE (VOP pRec, UI len, UW ini); ── CCITT 方式, 右送り演算, リトルエンディアン形式で設定

BOOL AjcSetCrc16L (VOP pRec, UI len, UW ini); ── ANSI-CRC16, 左送り演算, CPU 依存のエンディアン形式で設定
BOOL AjcSetCrc16L_BE (VOP pRec, UI len, UW ini); ── ANSI-CRC16, 左送り演算, ビッグエンディアン形式で設定
BOOL AjcSetCrc16L_LE (VOP pRec, UI len, UW ini); ── ANSI-CRC16, 左送り演算, リトルエンディアン形式で設定
BOOL AjcSetCrc16R (VOP pRec, UI len, UW ini); ── ANSI-CRC16, 右送り演算, CPU 依存のエンディアン形式で設定
BOOL AjcSetCrc16R_BE (VOP pRec, UI len, UW ini); ── ANSI-CRC16, 右送り演算, ビッグエンディアン形式で設定
BOOL AjcSetCrc16R_LE (VOP pRec, UI len, UW ini); ── ANSI-CRC16, 右送り演算, リトルエンディアン形式で設定

引　数 ： pRec - ＣＲＣを設定するストリームの先頭アドレス
len - ＣＲＣを設定するストリームのバイト数（２以上）
ini - 0x0000（反転なし）あるいは、0xFFFF（反転操作）を指定する。

説　明 ： ストリームの末尾２バイトへＣＲＣを設定します。
len は、ＣＲＣを計算するデータ部分ではなく、ストリーム全体のバイト数を指定します。



末尾が「_BE」の関数は、ＣＲＣをビッグエンディアン（高位バイト，低位バイトの順）で設定します。
末尾が「_LE」の関数は、ＣＲＣをリトルエンディアン（低位バイト，高位バイトの順）で設定します。

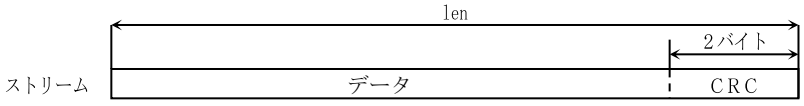
戻り値 ： なし

60.1.5. ストリームへ XMODEM / FCS 設定（AjcSetXMODEM・・・ ／ AjcSetFCS・・・）

形　式 ： VO AjcSetXMODEM (VOP pRec, UI len); --- XMODEM-CRC, CPU 依存のエンディアン形式で設定
VO AjcSetXMODEM_BE (VOP pRec, UI len); --- XMODEM-CRC, ビッグエンディアン形式で設定 (XMODEM 標準)
VO AjcSetXMODEM_LE (VOP pRec, UI len); --- XMODEM-CRC, リトルエンディアン形式で設定
VO AjcSetFCS (VOP pRec, UI len); --- ＦＣＳ, CPU 依存のエンディアン形式で設定
VO AjcSetFCS_BE (VOP pRec, UI len); --- ＦＣＳ, ビッグエンディアン形式で設定
VO AjcSetFCS_LE (VOP pRec, UI len); --- ＦＣＳ, リトルエンディアン形式で設定

引　数 ： pRec - ＸＭＯＤＥＭ用ＣＲＣ／ＦＣＳを設定するストリームの先頭アドレス
len - ＸＭＯＤＥＭ用ＣＲＣ／ＦＣＳを設定するストリームのバイト数（２以上）

説　明 ： ストリームの末尾２バイトへＸＭＯＤＥＭ用ＣＲＣ／ＦＣＳを設定します。
len は、ＣＲＣを計算する部分ではなく、ストリーム全体のバイト数を指定します。



末尾が「_BE」の関数は、ＣＲＣをビッグエンディアン（高位バイト，低位バイトの順）で設定します。
末尾が「_LE」の関数は、ＣＲＣをリトルエンディアン（低位バイト，高位バイトの順）で設定します。

戻り値 ： TRUE - 成功
FALSE - 失敗

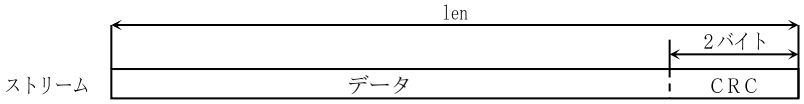
60.1.6. ストリームのＣＲＣ値検査（AjcChkCrc??・・・）

形　式 ： BOOL AjcChkCrcItL　(C_VOP pRec, UI len, UW ini); --- CCITT 方式, 左送り演算, CPU 依存のエンディアン形式として検査
BOOL AjcChkCrcItL_BE(C_VOP pRec, UI len, UW ini); --- CCITT 方式, 左送り演算, ビッグエンディアン形式として検査
BOOL AjcChkCrcItL_LE(C_VOP pRec, UI len, UW ini); --- CCITT 方式, 左送り演算, リトルエンディアン形式として検査
BOOL AjcChkCrcItR　(C_VOP pRec, UI len, UW ini); --- CCITT 方式, 右送り演算, CPU 依存のエンディアン形式として検査
BOOL AjcChkCrcItR_BE(C_VOP pRec, UI len, UW ini); --- CCITT 方式, 右送り演算, ビッグエンディアン形式として検査
BOOL AjcChkCrcItR_LE(C_VOP pRec, UI len, UW ini); --- CCITT 方式, 右送り演算, リトルエンディアン形式として検査

BOOL AjcChkCrc16L　(C_VOP pRec, UI len, UW ini); --- ANSI-CRC16, 左送り演算, CPU 依存のエンディアン形式として検査
BOOL AjcChkCrc16L_BE(C_VOP pRec, UI len, UW ini); --- ANSI-CRC16, 左送り演算, ビッグエンディアン形式として検査
BOOL AjcChkCrc16L_LE(C_VOP pRec, UI len, UW ini); --- ANSI-CRC16, 左送り演算, リトルエンディアン形式として検査
BOOL AjcChkCrc16R　(C_VOP pRec, UI len, UW ini); --- ANSI-CRC16, 右送り演算, CPU 依存のエンディアン形式として検査
BOOL AjcChkCrc16R_BE(C_VOP pRec, UI len, UW ini); --- ANSI-CRC16, 右送り演算, ビッグエンディアン形式として検査
BOOL AjcChkCrc16R_LE(C_VOP pRec, UI len, UW ini); --- ANSI-CRC16, 右送り演算, リトルエンディアン形式として検査

引　数 ： pRec - ＣＲＣを検査するストリームの先頭アドレス
len　- ＣＲＣを検査するストリームのバイト数（２以上）
ini　- 0x0000（反転なし）あるいは、0xFFFF（反転操作）を指定する。

説　明 ： ストリームの末尾２バイトのＣＲＣを検査します。
len は、ＣＲＣを計算する部分ではなく、ストリーム全体のバイト数を指定します。



末尾が「_BE」の関数は、ＣＲＣをビッグエンディアン（高位バイト，低位バイトの順）として検査します。
末尾が「_LE」の関数は、ＣＲＣをリトルエンディアン（低位バイト，高位バイトの順）として検査します。

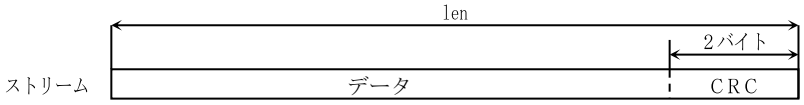
戻り値 ： TRUE　- ＣＲＣは一致した
FALSE - ＣＲＣエラー／失敗

60.1.7. ストリームの XMODEM / FCS 検査（AjcChkXMODEM・・・ ／ AjcChkFCS・・・）

形　式 ： BOOL　AjcChkXMODEM　　(C_VOP pRec, UI len); --- XMODEM-CRC, ＣＰＵ依存のエンディアン形式として検査
BOOL　AjcChkXMODEM_BE (C_VOP pRec, UI len); --- XMODEM-CRC, ビッグエンディアン形式として検査（XMODEM 標準）
BOOL　AjcChkXMODEM_LE (C_VOP pRec, UI len); --- XMODEM-CRC, リトルエンディアン形式として検査
BOOL　AjcChkFCS　　(C_VOP pRec, UI len); --- ＦＣＳ，ＣＰＵ依存のエンディアン形式として検査
BOOL　AjcChkFCS_BE　(C_VOP pRec, UI len); --- ＦＣＳ，ビッグエンディアン形式として検査
BOOL　AjcChkFCS_LE　(C_VOP pRec, UI len); --- ＦＣＳ，リトルエンディアン形式として検査

引　数 ： pRec - ＸＭＯＤＥＭ用ＣＲＣ／ＦＣＳを検査するストリームの先頭アドレス
len　- ＸＭＯＤＥＭ用ＣＲＣ／ＦＣＳを検査するストリームのバイト数（２以上）

説　明 ： ストリームの末尾２バイトのＸＭＯＤＥＭ-ＣＲＣ／ＦＣＳを検査します。
len は、ＣＲＣを計算する部分ではなく、ストリーム全体のバイト数を指定します。



末尾が「_BE」の関数は、ＣＲＣをビッグエンディアン（高位バイト，低位バイトの順）として検査します。
末尾が「_LE」の関数は、ＣＲＣをリトルエンディアン（低位バイト，高位バイトの順）として検査します。

戻り値 ： TRUE　- ＣＲＣは一致した
FALSE - ＣＲＣエラー

60.1.8. ストリームのバイト／ワードサム算出（CalcByteSum[N|S], AjcCalcWordSum[N|S]）

形　式 ： UB　AjcCalcByteSum　(VOP pRec, UI len)； --- バイトサム，単純加算
UB　AjcCalcByteSumN (VOP pRec, UI len)； ---　〃　　，１の補数
UB　AjcCalcByteSumS (VOP pRec, UI len)； ---　〃　　，２の補数
UB　AjcCalcByteSumX (VOP pRec, UI len)； ---　〃　　，排他論理和
UW　AjcCalcWordSum　(VOP pRec, UI len)； --- ワードサム，単純加算
UW　AjcCalcWordSumN (VOP pRec, UI len)； ---　〃　　，１の補数
UW　AjcCalcWordSumS (VOP pRec, UI len)； ---　〃　　，２の補数
UW　AjcCalcWordSumX (VOP pRec, UI len)； ---　〃　　，排他論理和
UW　AjcCalcWordSumR (VOP pRec, UI len)； --- ワードサム，単純加算（自ＣＰＵとは、逆のエンディアン形式で扱う）
UW　AjcCalcWordSumNR(VOP pRec, UI len)； ---　〃　　，１の補数（自ＣＰＵとは、逆のエンディアン形式で扱う）
UW　AjcCalcWordSumSR(VOP pRec, UI len)； ---　〃　　，２の補数（自ＣＰＵとは、逆のエンディアン形式で扱う）
UW　AjcCalcWordSumXR(VOP pRec, UI len)； ---　〃　　，排他論理和（自ＣＰＵとは、逆のエンディアン形式で扱う）

引　数 ： pRec – チェックサムを算出するデータブロックの先頭アドレス
len　– チェックサムを算出するデータブロックのバイト数

説　明 ： データブロックのチェックサムを算出します。
チェックサムの算出方法は、データブロックの全バイト／全ワードを単純に加算した、下位８ビット／１６ビットです。
末尾が「N/NR」の関数は、加算後に１の補数をとります。また、末尾が「S/SR」の関数は、加算後に２の補数をとります。
末尾が「X/XR」の関数は、初期値＝０、で全バイト／全ワードを排他論理和(XOR)します。

尚、「CalcWordSum[N|S][R]」で、len に奇数を指定した場合、データブロックの最終バイトは、データの最終バイトは、Bit15～8＝00h，Bit7～0＝最後の１バイト値として加算します。

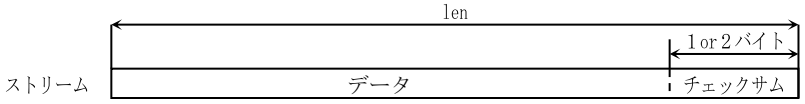
戻り値 ： 算出したチェックサム値（戻り値は、自ＣＰＵのエンディアン形式です。失敗時は０を返します）

60.1.9. ストリームのバイト／ワードサム設定（AjcSetByteSum[N|S], AjcSetWordSum[N|S]）

形　式 ： BOOL　AjcSetByteSum　(VOP pRec, UI len)； --- バイトサム，単純加算
BOOL　AjcSetByteSumN (VOP pRec, UI len)； ---　〃　　，１の補数
BOOL　AjcSetByteSumS (VOP pRec, UI len)； ---　〃　　，２の補数
BOOL　AjcSetByteSumX (VOP pRec, UI len)； ---　〃　　，排他論理和
BOOL　AjcSetWordSum　(VOP pRec, UI len)； --- ワードサム，単純加算
BOOL　AjcSetWordSumN (VOP pRec, UI len)； ---　〃　　，１の補数
BOOL　AjcSetWordSumS (VOP pRec, UI len)； ---　〃　　，２の補数
BOOL　AjcSetWordSumX (VOP pRec, UI len)； ---　〃　　，排他論理和
BOOL　AjcSetWordSumR (VOP pRec, UI len)； --- ワードサム，単純加算（自ＣＰＵとは、逆のエンディアン形式で扱う）
BOOL　AjcSetWordSumNR(VOP pRec, UI len)； ---　〃　　，１の補数（自ＣＰＵとは、逆のエンディアン形式で扱う）
BOOL　AjcSetWordSumSR(VOP pRec, UI len)； ---　〃　　，２の補数（自ＣＰＵとは、逆のエンディアン形式で扱う）
BOOL　AjcSetWordSumXR(VOP pRec, UI len)； ---　〃　　，排他論理和（自ＣＰＵとは、逆のエンディアン形式で扱う）

引　数 ： pRec – チェックサムを設定するストリームの先頭アドレス
len　– チェックサムを設定するストリームのバイト数

説　明 ： ストリームの末尾へチェックサムを設定します。
len は、チェックサムを計算する部分ではなく、ストリーム全体のバイト数を指定します。



チェックサムの算出方法は、データの全バイト／全ワードを単純に加算した、下位８ビット／１６ビットです。
末尾が「N/NR」の関数は、加算後に１の補数をとります。また、末尾が「S/SR」の関数は、加算後に２の補数をとります。
末尾が「X/XR」の関数は、初期値＝０で、全バイト／全ワードを排他論理和(XOR)します。

尚、「SetWordSum[N|S][R]」で、len に奇数を指定した場合は、データの最終バイトは、Bit15～8＝00h，Bit7～0＝最後の１バイト値として加算します。

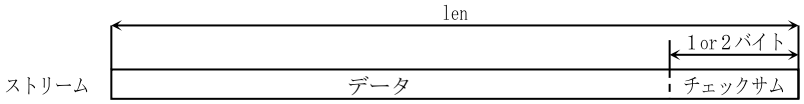
戻り値 ： TRUE　– 成功
FALSE – 失敗

60.1.10. ストリームのバイト／ワードサム検査（ChkByteSum[N|S], ChkWordSum[N|S]）

形　式 ： BOOL　AjcChkByteSum　(VOP pRec, UI len); --- バイトサム, 単純加算
BOOL　AjcChkByteSumN (VOP pRec, UI len); ---　〃　　, １の補数
BOOL　AjcChkByteSumS (VOP pRec, UI len); ---　〃　　, ２の補数
BOOL　AjcChkByteSumX (VOP pRec, UI len); ---　〃　　, 排他論理和
BOOL　AjcChkWordSum　(VOP pRec, UI len); --- ワードサム, 単純加算
BOOL　AjcChkWordSumN (VOP pRec, UI len); ---　〃　　, １の補数
BOOL　AjcChkWordSumS (VOP pRec, UI len); ---　〃　　, ２の補数
BOOL　AjcChkWordSumX (VOP pRec, UI len); ---　〃　　, 排他論理和
BOOL　AjcChkWordSumR (VOP pRec, UI len); --- ワードサム, 単純加算（自ＣＰＵとは、逆のエンディアン形式で扱う）
BOOL　AjcChkWordSumNR(VOP pRec, UI len); ---　〃　　, １の補数（自ＣＰＵとは、逆のエンディアン形式で扱う）
BOOL　AjcChkWordSumSR(VOP pRec, UI len); ---　〃　　, ２の補数（自ＣＰＵとは、逆のエンディアン形式で扱う）
BOOL　AjcChkWordSumXR(VOP pRec, UI len); ---　〃　　, 排他論理和（自ＣＰＵとは、逆のエンディアン形式で扱う）

引　数 ： pRec - チェックサムを検査するストリームの先頭アドレス
len　- チェックサムを検査するストリームのバイト数

説　明 ： ストリーム末尾のチェックサムを検査します。
len は、チェックサムを計算する部分ではなく、ストリーム全体のバイト数を指定します。



チェックサムの算出方法は、データの全バイト／全ワードを単純に加算した、下位８ビット／１６ビットです。
末尾が「N/NR」の関数は、加算後に１の補数をとります。また、末尾が「S/SR」の関数は、加算後に２の補数をとります。
末尾が「X/XR」の関数は、初期値＝０で、全バイト／全ワードを排他論理和(XOR)します。

尚、「ChkWordSum[N|S][R]」で、len に奇数を指定した場合は、データの最終バイトは、Bit15～8＝00h, Bit7～0＝最後の１バイト値として加算します。

戻り値 ： TRUE　- チェックサムは一致した
FALSE - チェックサム不一致／失敗

61. ＨＥＸデータ処理

インテルＨＥＸ／モトローラＳ形式の、ＨＥＸデータの操作関数群です。

エラーコード

#	名称	値	内容	備考
1	AJCIHERR_OK	0	正常	
2	AJCIHERR_DIGIT	1	不正な文字が存在する	
3	AJCIHERR_NOMARK	2	先頭がコロン (':') ／ 'S' でない	
4	AJCIHERR_SUM	3	チェックサム異常	
5	AJCIHERR_TYPE	4	不正なレコードタイプ	
6	AJCIHERR_NOEND	5	ＥＮＤレコードが存在しない	
7	AJCIHERR_FOPEN	6	ファイルオープン失敗	

61.1. サポートＡＰＩ

ＨＥＸデータ処理のサポートＡＰＩ一覧を以下に示します。

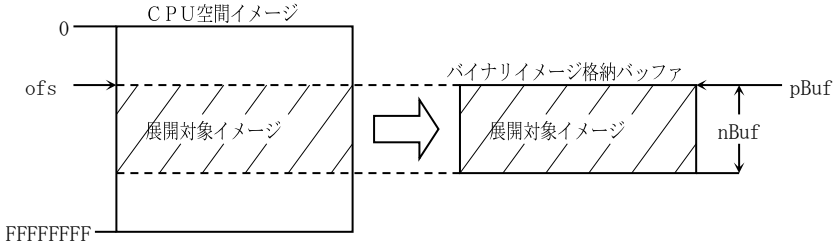
#	関 数 名	内容	備考
1	AjcLoadHexData	インテルＨＥＸ／モトローラＳ形式のデータをメモリ上に展開する	
2	AjcLoadHexFile	インテルＨＥＸ／モトローラＳ形式のＨＥＸファイルをメモリ上に展開する	
3	AjcIHexToBin	インテルＨＥＸ形式のデータをバイナリ化する	
4	AjcSMotToBin	モトローラＳ形式のデータをバイナリ化する	

61.1.1. インテルＨＥＸ／モトローラＳ形式のデータをメモリ上に展開する（AjcLoadHexData）

形　式 ： UI AjcLoadHexData (UBP pBuf, UL nBuf, UL ofs, ULP pSrtAddr,
UX cbp, BOOL (CALLBACK *cbGetRec)(UX xp, UTP pRec, UI nRec));

引　数 ： pBuf　　- 展開したバイナリイメージデータを格納するバッファのアドレス
nBuf　　- 展開したバイナリイメージデータを格納するバッファのバイト数
ofs　　- 展開対象とするバイナリイメージの先頭アドレス
pSrtAddr - 開始アドレスを格納するバッファのアドレス（不要時は、ＮＵＬＬ）
cbp　　- コールバックパラメタ
cbGetRec - インテルＨＥＸ／モトローラＳ形式のレコード取得用コールバック関数のアドレス

説　明 ： インテルＨＥＸ形式／モトローラＳ形式のデータをメモリ上に展開します。
展開対象とするバイナリイメージは、以下のようになります。



バイナリイメージ格納用バッファは、(複数のバイナリイメージを併合可能とするために）バイナリイメージの展開に先立って、0xFF 等のデータで埋めるようなことはしません。従って、バイナリデータがロードされない部分は実行前のままとなります。

pSrtAddr で示されるバッファには、プログラムの開始アドレスが設定されます。但し、プログラムの開始アドレス情報が無い場合は、0xFFFFFFFF が設定されます。

戻り値 ： ＝０：正常
≠０：エラーコード

コールバック

cbGetRec（インテルＨＥＸ／モトローラＳ形式レコード取得）

形　式 ： BOOL CALLBACK ***cbGetRec***(UX cbp, UTP pRec, UI nRec);

引　数 ： cbp - コールバックパラメタ
pRec - インテルＨＥＸ／モトローラＳ形式レコード入力バッファのアドレス
nRec - インテルＨＥＸ／モトローラＳ形式レコード入力バッファのバイト数

説　明 ： pRec, nRec で示されるバッファに、インテルＨＥＸ／モトローラＳ形式レコードを順次読み出します。

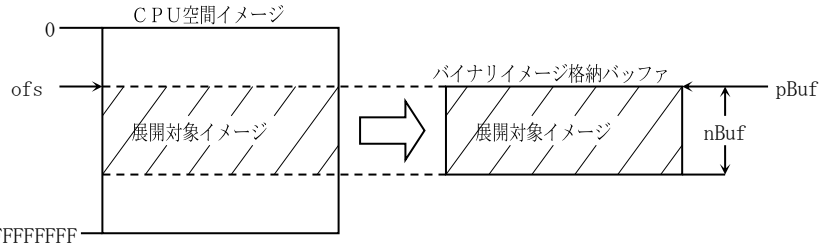
戻り値 ： TRUE　－ Not EOF
FALSE　－ EOF

61.1.2. インテルＨＥＸ／モトローラＳ形式のＨＥＸファイルのロード (AjcLoadHexFile)

形　式 ： UI　　AjcLoadHexFile(C_UTP pFile, UTP pBuf, UL nBuf, UL ofs, ULP pSrtAdr);

引　数 ： pFile　- インテルＨＥＸ／モトローラＳ形式ファイル名のアドレス
pBuf　- ロードしたバイナリイメージデータを格納するバッファのアドレス
nBuf　- ロードしたバイナリイメージデータを格納するバッファのバイト数
ofs　- ロード対象とするバイナリイメージの先頭アドレス
pSrtAdr - 開始アドレスを格納するバッファのアドレス（不要時はＮＵＬＬ）

説　明 ： インテルＨＥＸ形式／モトローラＳ形式のファイルを pBuf,nBuf で指定されたバッファ上に展開します。
展開対象とするバイナリイメージは、以下のようになります。



バイナリイメージ格納用バッファは、(複数のバイナリイメージを併合可能とするために）バイナリイメージの展開に先立って、0xFF 等のデータで埋めるようなことはしません。従って、バイナリデータがロードされない部分は実行前のままとなります。

pSrtAddr で示されるバッファには、プログラムの開始アドレスが設定されます。但し、プログラムの開始アドレス情報が無い場合は、0xFFFFFFFF が設定されます。

戻り値 ： ＝０：正常
≠０：エラーコード

61.1.3. インテルＨＥＸ形式のデータをバイナリ形式に変換する（AjcIHexToBin）

形　式 ： UI　AjcIHexToBin (C_UTP pRec, UBP pBuf, UL nBuf, UBP pTyp, ULP pAdr, UIP pLen);

引　数 ： pRec – インテルＨＥＸ形式のテキストレコードのアドレス
pBuf – 変換したバイナリイメージデータを格納するバッファのアドレス
nBuf – 変換したバイナリイメージデータを格納するバッファのバイト数
pTyp – レコードタイプ情報を格納するバッファのアドレス
pAdr – アドレス情報を格納するバッファのアドレス
pLen – バイナリイメージデータのバイト数を格納するバッファのアドレス

説　明 ： インテルＨＥＸ形式のテキストデータをを解析し、以下の情報を設定します。

*pTyp	pBuf,nBuf で示すバッファ	*pAdr	*pLen
0x00（データレコード）	バイナリイメージデータ	ロケーション	バイナリイメージデータのバイト数
0x01（ＥＮＤレコード）	－	0	－
0x02（拡張アドレス，パラグラフ単位）	－	ベースアドレス（※1）	－
0x03（スタートアドレス）	－	スタートアドレス	－
0x04（拡張アドレス，６４ＫＢ単位）	－	ベースアドレス（※1）	－

※1：バイト単位のアドレス値に変換した値が設定されます。（パラグラフ単位の場合１６倍，６４ＫＢ単位の場合６５５３６倍します）

戻り値 ： ＝０：正常
≠０：エラーコード

61.1.4. モトローラＳ形式のデータをバイナリ形式に変換する（AjcSMotToBin）

形　式 ： UI　AjcSMotToBin (C_UTP pRec, UBP pBuf, UL nBuf, UBP pTyp, ULP pAdr, UIP pLen);

引　数 ： pRec – モトローラＳ形式のテキストレコードのアドレス
pBuf – 変換したバイナリイメージデータを格納するバッファのアドレス
nBuf – 変換したバイナリイメージデータを格納するバッファのバイト数
pTyp – レコードタイプ情報を格納するバッファのアドレス
pAdr – アドレス情報を格納するバッファのアドレス
pLen – バイナリイメージデータのバイト数を格納するバッファのアドレス

説　明 ： モトローラＳ形式のテキストデータを解析し、以下の情報を設定します。

*pTyp	pBuf,nBuf で示すバッファ	*pAdr	*pLen
‘0’（Ｓ０レコード）	－	－	－
‘2’（Ｓ２レコード）	バイナリイメージデータ	バイナリイメージデータのアドレス	バイナリイメージデータのバイト数
‘3’（Ｓ３レコード）			
‘7’（Ｓ７レコード）	－	開始アドレス	－
‘8’（Ｓ８レコード）	－		－

戻り値 ： ＝０：正常
≠０：エラーコード

62. フォント処理とフォント選択ダイアログ

フォント選択ダイアログ等の関数群です。

62.1. サポートＡＰＩ

フォント選択ダイアログ関連のサポートＡＰＩ一覧を以下に示します。

#	関 数 名	内容	備考
1	AjcCfFontDlg	ダイアログボックスによるフォント選択	
2	AjcCfInitFontInfo AjcCfInitPermInfo	フォント情報，ダイアログ設定値の初期化	
3	AjcCfInfoToText	フォント情報，ダイアログ設定値→文字列 変換	
4	AjcCfTextToInfo	文字列→フォント情報，ダイアログ設定値変換	
5	AjcCfMergeToInfo	フォント情報をフォント選択ダイアログ情報に組み入れる	
6	AjcCfEnumFonts	フォント情報の列挙	
7	AjcCfFindFont	フォントフェース名検索	
8	AjcCfSetPreviewText	フォント選択ダイアログのプレビューテキスト設定	
9	AjcCfCreateAllFontInfo	英字名を含む全フォント情報の生成	
10	AjcCfReleaseAllFontInfo	英字名を含む全フォント情報の解放	
11	AjcCfFindFontEx	英字名を含む全フォントの検索	
12	AjcCfEnumFontsEx	英字名を含む全フォントの列挙	
13	AjcCfPointsToPixels	ポイント数 → ピクセル数変換	
14	AjcCfPixelsToPoints	ピクセル数 → ポイント数変換	

62.1.1. ダイアログボックスによるフォント選択 (AjcCfFontDlg)

形　式 ： BOOL AjcCfFontDlg(HWND hOwner, UI opt, PAJCCFFONTINFO pFont, PAJCCFPERMINFO pPerm, UX cbp,
VO (CALLBACK *cbInit)(HWND hDlg, UX cbp),
BOOL (CALLBACK *cbApply)(HFONT hFont, UI lsp, UX cbp));

引　数 ： hOwner　－ オーナーウインドハンドル（不要時は NULL）
opt　－ オプションフラグ（AJCCFF_XXXXX, 未指定時は０）
pFont　－ 選択したフォント情報を格納するバッファへのポインタ
pPerm　－ 入力：フォント選択ダイアログの設定値へのポインタ
出力：フォント選択ダイアログの設定値を格納するバッファへのポインタ
cbp　－ コールバックパラメタ
cbInit　－ ダイアログ起動時のコールバック関数（不要時は NULL）
cbApply　－ 「摘要」ボタン押下時のコールバック関数（不要時は NULL）

説　明 ： ダイアログボックス（右図）により、フォントの選択を行います。
opt には以下のシンボルの合成値を指定します。



#	記号名称	内容
1	AJCCFF_FIXEDONLY	「固定ピッチのみ」を選択
2	AJCCFF_INCVERT	縦書きフォントを含める
3	AJCCFF_NOBOLD	「太字」設定禁止
4	AJCCFF_NOITALIC	「斜字」設定禁止
5	AJCCFF_NOFIXED	「固定ピッチのみ」設定禁止
6	AJCCFF_NOLSPACE	「行間スペース」設定禁止
7	AJCCFF_HIDEBOLD	「太字」非表示
8	AJCCFF_HIDEITALIC	「斜字」非表示
9	AJCCFF_HIDEFIXED	「固定ピッチのみ」設非表示
10	AJCCFF_HIDELSPACE	「行間スペース」非表示

pFont は、選択したフォント情報を格納するバッファを指定します。
pFont で示すバッファの形式は以下のとおりです。

```
typedef struct {
    LOGFONT     lf;             //  フォント情報
    UI          lsp;            //  行間スペース
} AJCCFFONTINFO, *PAJCCFFONTINFO;
typedef const AJCCFFONTINFO *PCAJCCFFONTINFO;
```

pPerm は、永続化情報（ダイアログの設定値情報）を格納するバッファを指定します。
この情報をプロファイル等に記録することにより、ダイアログ設定値を永続化できます。
pPerm で示すバッファの形式は以下のとおりです。

```
//  フォントフェース以外
typedef struct {
    UI          id;             //  情報が有効であることを示すＩＤ(=AJCCFPERMID)
    BOOL        fPixel;         //  高さ＝ピクセル選択
    BOOL        fItalic;        //  斜字 チェック
    BOOL        fBold;          //  太字 チェック
    BOOL        fFixed;         //  固定ピッチ チェック
    UI          Pixels;         //  高さのピクセル数
    UI          Points;         //  高さのポイント数
    UI          LSpace;         //  行間スペース
    AJCCFATT    FixAttr;        //  最後に選択された固定ピッチフォントの属性 (CharSet, PitchAndFamily)
    AJCCFATT    AnyAttr;        //  最後に選択された任意ピッチフォントの属性 (CharSet, PitchAndFamily)
} AJCCFPERMHD, *PAJCCFPERMHD;
//  永続化情報
typedef struct {
    AJCCFPERMHD hd;             //  フォントフェース以外
    WC          FixFace[32];    //  最後に選択された固定ピッチフォントのフォント名
    WC          AnyFace[32];    //  最後に選択された任意ピッチフォントのフォント名
} AJCCFPERMINFO, *PAJCCFPERMINFO;
typedef const AJCCFPERMINFO *PCAJCCFPERMINFO;;
```

pPerm=NULL あるいは、pPerm で示す永続化情報の id≠AJCCFPERMID である場合は、規定値を仮定します。

戻り値 ： TRUE ：ＯＫボタン押下
FALSE：キャンセル

コールバック： コールバック関数の仕様は、以下のとおりです。

cbInit（ダイアログ起動通知）

形　式 ： VO CALLBACK ***cbInit*** (HWND hDlg, UX cbp)；

引　数 ： hDlg　　- ダイアログボックスのウインドハンドル
cbp　　- コールバックパラメタ

説　明 ：ダイアログボックスが起動したことを通知します。
hDlg は、フォント選択ダイアログボックスのハンドルです。

戻り値 ：なし

cbApply（「摘要」ボタン押下通知）

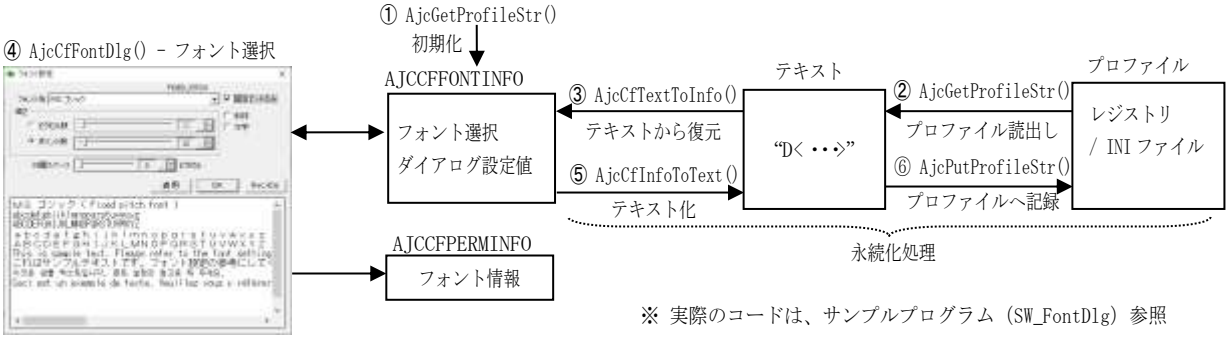
形　式 ： BOOL CALLBACK ***cbApply*** (HFONT hFont, UI lsp, UX cbp)；

引　数 ： hDlg　　- ダイアログボックスのウインドハンドル
lsp　　- 行間スペース
cbp　　- コールバックパラメタ

説　明 ：「摘要」ボタンが押下されたことを通知します。
hFont は、現在選択されているフォント情報により生成されたフォント・ハンドルです。

戻り値 ：TRUE ： フォントハンドルを破棄する
FALSE ： フォントハンドルを破棄しない（後でユーザが破棄する(DeleteObject(hFont);を実行する))

備　考 ： フォント選択ダイアログの設定情報は、以下の手順で（プロファイルにテキストを記録し）永続化できます。



※ 実際のコードは、サンプルプログラム（SW_FontDlg）参照

62.1.2.　フォント情報，ダイアログ設定値の初期化 (AjcCfInitFontInfo, AjcCfInitPermInfo)

形　式 ： BOOL AjcCfInitFontInfo(C_UTP pFace, PAJCCFFONTINFO pFont)； ---------- フォント情報初期化
BOOL AjcCfInitPermInfo(C_UTP pFixFace, C_UTP pAnyFace, PAJCCFPERMINFO pPerm)； - ダイアログ設定値初期化

引　数 ： pFace　　- フォント情報へ設定するフォントフェース名へのポインタ（NULL: "ＭＳ ゴシック"を仮定）
pFont　　- 設定するフォント情報バッファへのポインタ

pFixFace　- 固定ピッチフォントのフォントフェース名へのポインタ（NULL: "ＭＳ ゴシック"を仮定）
pAnyFace　- 任意ピッチフォントのフォントフェース名へのポインタ（NULL: "MS UI Gothic"を仮定）
pPerm　　- 設定するフォント選択ダイアログ情報バッファへのポインタ

説　明 ： 指定したフォントフェース名でフォント情報や、フォント選択ダイアログ情報を初期化します。
フォント情報は、文字サイズ（高さ）＝１２ピクセル，行間スペース＝０に設定します。
フォント選択ダイアログの設定値は、文字サイズ＝１２ピクセル，１２ポイントに設定します。

戻り値 ： TRUE ： 成功
FALSE ： 失敗

62.1.3. フォント情報，ダイアログ設定値 → 文字列 変換 (AjcCfInfoToText)

形　式 ： UI　AjcCfInfoToText(PCAJCCFPERMINFO pPerm, UT pBuf[AJCCFMAX_TEXT]);

引　数 ： pPerm　　- 永続化情報（ダイアログ設定値）へのポインタ
pBuf　　- ダイアログ設定値を表す文字列を格納するバッファへのポインタ

説　明 ： AjcCfFontDlg()で取得したフォント選択ダイアログの設定値を表す文字列を作成します。

戻り値 ： 作成した文字列の長さ

62.1.4. 文字列 → フォント情報，ダイアログ設定値変換 (AjcCfTextToInfo)

形　式 ： BOOL　AjcCfTextToInfo(C_BCP pTxt, PAJCCFPERMINFO pPerm);

引　数 ： pTxt　　- ダイアログ設定値を表す文字列へのポインタ（NULL 指定時は、ダイアログ設定値を初期化）
pPerm　　- 永続化情報（ダイアログ設定値）を格納するバッファのポインタ

説　明 ： フォント選択ダイアログの設定値を表す文字列から、フォント情報，ダイアログ設定値情報を作成します。

戻り値 ： TRUE　：成功
FALSE：失敗

62.1.5. フォント情報をフォント選択ダイアログ情報に組み入れる (AjcCfMergeToInfo)

形　式 ： BOOL　AjcCfMergeToInfo(PCAJCCFFONTINFO pFont, PAJCCFPERMINFO pPerm);

引　数 ： pFont　　- フォント情報へのポインタ
pPerm　　- フォント選択ダイアログの設定値を格納するバッファのポインタ

説　明 ： フォント選択ダイアログを起動した際に当該フォントが選択されている状態となるように、フォント情報をフォント選択ダイアログ情報に組み入れます。

戻り値 ： TRUE　：成功
FALSE：失敗

62.1.6. フォント情報の列挙 (AjcCfEnumFonts)

形　式 ： UI　AjcCfEnumFonts(UX cbp, BOOL (CALLBACK *cbNtcFont)(C_UTP pFaceName, UB CharSet, UB PitchAndFamily, UX cbp));

引　数 ： cbp　　- コールバックパラメタ
cbNtcFont　- フォント情報通知用コールバック関数

説　明 ： フォント情報を列挙します。

戻り値 ： 列挙したフォント数

コールバック： コールバック関数の仕様は、以下のとおりです。

cbNtcFont（フォント情報通知）

形　式 ： BOOL　CALLBACK ***cbNtcFont***(C_UTP pFaceName, UB CharSet, UB PitchAndFamily, UX cbp)；

引　数 ： pFaceName　　- フォントフェース名へのポインタ
CharSet　　- キャラクタセット　　（LOGFONT 構造体の lfCharSet と同じ）
PitchAndFamily - ピッチ&フォントファミリ（LOGFONT 構造体の lfPitchAndFamily と同じ）
cbp　　- コールバックパラメタ

説　明 ： ダイアログボックスが起動したことを通知します。
hDlg は、フォント選択ダイアログボックスのハンドルです。

戻り値 ： TRUE　- 列挙を継続する
FALSE　- 列挙を終了する

備　考 ： システムＡＰＩ（EnumFontFamiliesEx()）により通知されるフォント情報により列挙します。

62.1.7. フォントフェース名検索 (AjcCfFindFont)

形　式 ： BOOL AjcCfFindFont(C_UTP pFaceName, PAJCCFATT pBuf);

引　数 ： pFaceName – 検索するフォントフェース名へのポインタ
pBuf – ピッチ＆ファミリとキャラクタセット情報を格納するバッファへのポインタ

説　明 ： フォント情報から pFaceName で指定されたフォントを検索します。
pBuf 示すバッファの形式は以下のとおりです。

```
typedef union {
    struct {UB PitchAndFamily, CharSet;} s;
    UI                                   v;
} AJCCFATT, *PAJCCFATT;
```

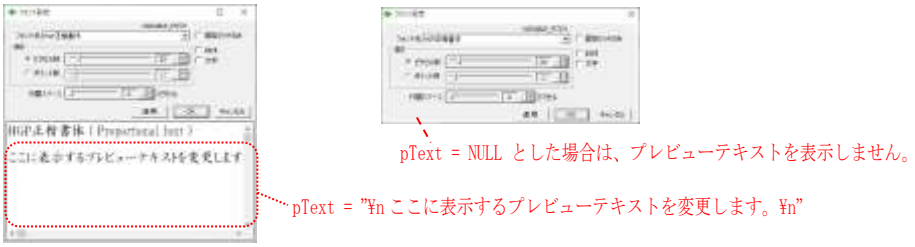
戻り値 ： TRUE ： 成功（見つかった）
FALSE ： 見つからない／失敗

62.1.8. フォント選択ダイアログのプレビューテキスト設定(AjcCfSetPreviewText)

形　式 ： BOOL AjcCfSetPreviewText (C_UTP pText);

引　数 ： pText – プレビューテキストへのポインタ (NULL:プレビューテキストを表示しない)

説　明 ： フォント選択ダイアログ上に表示するプレビューテキスト（下図左の赤枠部分の表示テキスト）を設定します。



プレビューテキストは、プログラム内で共通のスタティックなテキストとして存在します。

戻り値 ： TRUE ： 成功
FALSE ： 失敗

62.1.9. 英字名を含む全フォント情報の生成 (AjcCfCreateAllFontInfo)

形　式 ： HAJCAVS　AjcCfCreateAllFontInfo(VO);

引　数 ： なし

説　明 ： 英字名を含む、全てのフォント情報を収集します。
収拾したフォント情報は、フォント名をキーとした２分木に蓄えられます。
この２分木のノードデータとして以下の情報が作成されます。（２分木については「平衡２分木（文字列キー)」を参照)

```
//  フォント属性情報
typedef struct {
    UB      flag;                   //  フラグ (AJCFIF_XXX)
    UB      PitchAndFamily;         //  ピッチ&ファミリ
    UB      CharSet;                //  キャラクタセット
    UB      filler;                 //  - (未使用)
    C_WCP   pOrgName;               //  英字名→オリジナル名へのポインタ
} AJCFIFNODE, *PAJCFIFNODE;
typedef const AJCFIFNODE *PCAJCFIFNODE;
```

flag には、以下の情報の合成値が設定されます。

#	シンボル	値	内容	
1	AJCFIF_ORG	0x10	元のフォント名を示すフラグ	システムＡＰＩ (EnumFontFamiliesEx()) で通知されたフォント情報から収集したフォント名
2	AJCFIF_ENG	0x08	英字名のフォントファミリ名を示すフラグ	システムＡＰＩ (GetFontData()) により拡張フォント情報から収集した英字名フォント
3	AJCFIF_SUB	0x04	英字名のサブファミリ名を示すフラグ	

pOrgName は、英字名のフォントの場合、元のフォント名へのポインタです。(元のフォント名自身の場合は NULL)
（例えば、英字名のフォント("MS Gothic")である場合、pOrgName は("ＭＳ ゴシック")を示します)

収拾したフォント情報は、AjcCfReleaseAllFontInfo()が実行されるか、プロセスが終了するまで保持されます。

既に、全フォント情報を収集済みである場合は、何もせずに、収集済みの２分木ハンドルを返します。

戻り値 ： ≠NULL：２分木のハンドル
＝NULL：失敗

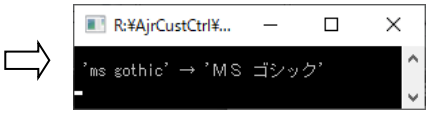
備　考 ： ・同じフォント名が収集される場合は、最初のフォントだけが有効となります。

・２分木ノードデータ中の pOrgName は、UNICODE 文字列へのポインタです。バイト文字モードの場合は注意が必要です。
例えば、以下のコードでは、英字フォント名("ms gothic")を検索し、オリジナルのフォント名を表示する際に AjcPrintFW()を使用しています。

```
    PCAJCFIFNODE p;

    AjcSetStdoutMode();

    if (p = AjcCfFindFontEx(TEXT("ms gothic"))) {
        AjcPrintFW(L"¥n 'ms gothic' → '%s'¥n", p->pOrgName);
    }
```



62.1.10. 英字名を含む全フォント情報の解放 (AjcCfReleaseAllFontInfo)

形　式 ： VO　AjcCfReleaseAllFontInfo (VO);

引　数 ： なし

説　明 ： AjcCfCreateAllFontInfo()で生成したフォント情報を解放します。
本ＡＰＩでフォント情報を解放しない場合は、プロセス終了時に自動的に解放されます。

戻り値 ： なし

62.1.11. 英字名を含む全フォントの検索 (AjcCfFindFontEx)

形　式 ： PCAJCFIFNODE　AjcCfFindFontEx (C_UTP pFontName);

引　数 ： pFontName - フォント名へのポインタ

説　明 ： 英字名を含む全フォント情報から、pFontName で指定されたフォントを検索します。
この関数は、英字名を含む、全フォント情報を収集するために内部で AjcCfCreateAllFontInfo()を実行します。

戻り値 ： ≠NULL：２分木のノードデータへのポインタ（ノードデータの形式は AjcCfCreateAllFontInfo()参照）
＝NULL：失敗

62.1.12. 英字名を含む全フォントの列挙(AjcCfEnumFontsEx)

形　式 ： UI　AjcCfEnumFontsEx(UX cbp, BOOL (CALLBACK *cbNtcFont)(C_UTP pFontName, PCAJCFIFNODE pNode, UX cbp));

引　数 ： cbp　　- コールバックパラメタ
cbNtcFont　- フォント情報通知用コールバック関数

説　明 ： 英字名を含む全フォント情報を列挙します。
この関数は、英字名を含む、全フォント情報を収集するために内部で AjcCfCreateAllFontInfo()を実行します。

戻り値 ： ≠０：列挙したフォント数
＝０：失敗

コールバック： コールバック関数の仕様は、以下のとおりです。

cbNtcFont（フォント情報通知）

形　式 ： BOOL CALLBACK ***cbNtcFont***(C_UTP pFontName, PCAJCFIFNODE pNode, UX cbp)；

引　数 ：pFaceName - フォントフェース名へのポインタ
pNode　　- ２分木のノードデータへのポインタ（ノードデータの形式は AjcCfCreateAllFontInfo()参照）
cbp　　- コールバックパラメタ

説　明 ：ダイアログボックスが起動したことを通知します。
hDlg は、フォント選択ダイアログボックスのハンドルです。

戻り値 ：TRUE　- 列挙を継続する
FALSE　- 列挙を終了する

62.1.13. ポイント数 → ピクセル数変換 (AjcCfPointsToPixels)

形　式 ： int AjcCfPointsToPixels(int points);

引　数 ： points – ポイント数

説　明 ： ディスプレィ・フォントのポイント数をピクセル数に変換します。

戻り値 ： ピクセル数

62.1.14. ピクセル数 →ポイント数変換 (AjcCfPixelsToPoints)

形　式 ： int AjcCfPixelsToPoints(int pixels);

引　数 ： pixels – ピクセル数

説　明 ： ディスプレィ・フォントのピクセル数をポイント数に変換します。
pixels が負数の場合は、絶対値に変換して計算します。

戻り値 ： ポイント数

62.2. サンプルプログラム

62.2.1. SW_FontDlg（フォント選択）

以下のサンプルプログラムは、FontDlg()の使用例です。




```
 1 : //
 2 : //  SW_FontDlg.c
 3 : //
 4 :
 5 : #include    <AjrCstXX.h>
 6 : #include    <math.h>
 7 : #include    <tchar.h>
 8 : #include    "resource.h"
 9 :
10 : #define CTRL_V  0x001B
11 :
12 : //-------------------------------------------------------------------------------------------------------------//
13 : //  ワーク                                                                                                     //
14 : //-------------------------------------------------------------------------------------------------------------//
15 : static  HINSTANCE       hInst;                          //  ＤＬＬインスタンスハンドル
16 : static  HWND            hDlgMain;                       //  ダイアログボックスハンドル
17 : static  HWND            hVth;
18 :
19 : static  AJCCFPERMINFO   PermInfo = {0};
20 : static  UI              lsp = 0;
21 : static  int             VthTop;
22 :
23 : //-------------------------------------------------------------------------------------------------------------//
24 : //  内部サブ関数                                                                                               //
25 : //-------------------------------------------------------------------------------------------------------------//
26 : AJC_DLGPROC_DEF(Main);
27 : static  VO   CALLBACK cbInit (HWND hDlg, UX cbp);
28 : static  BOOL CALLBACK cbApply(HFONT hFont, UI lsp, UX cbp);
29 : //=============================================================================================================//
30 : //                                                                                                             //
31 : //  W i n M a i n                                                                                              //
32 : //                                                                                                             //
33 : //=============================================================================================================//
34 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
35 : {
36 :     MSG     msg;
37 :
38 :     hInst = hInstance;
39 :
40 :     //----- プロファイルを INI ファイルに設定 ------------//
41 :     AjcSetProfileIsRegistory(FALSE);
42 :
43 :     //----- メイン・ダイアログオープン -----------------//
44 :     hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_DLGMAIN), NULL, AJC_DLGPROC_NAME(Main));
45 :     //----- ダイアログ表示 -----------------------------//
46 :     ShowWindow(hDlgMain, SW_SHOW);
47 :
48 :     //----- メッセージループ ---------------------------//
49 :     while (GetMessage(&msg, NULL, 0, 0)) {
50 :         do {
51 :             if (IsDialogMessage(hDlgMain, &msg)) break;
52 :             TranslateMessage(&msg);
53 :             DispatchMessage (&msg);
54 :         } while (0);
55 :     }
56 :
57 :     return (int)msg.wParam ;
58 : }
59 : //=============================================================================================================//
60 : //                                                                                                             //
61 : //  ダイアログ・プロシージャ                                                                                   //
62 : //                                                                                                             //
63 : //=============================================================================================================//
64 : //----- ダイアログ初期化 -------------------------------------------------------------------------------------//
65 : AJC_DLGPROC(Main, WM_INITDIALOG     )
66 : {
```

```
67 :     RECT        r;
68 :     UT          txt[AJCCFMAX_TEXT];
69 :
70 :     hDlgMain = hDlg;
71 :     hVth     = GetDlgItem(hDlg, IDC_VTH);
72 :
73 :     //  VTH 上位置退避
74 :     GetWindowRect(hVth, &r);
75 :     MapWindowPoints(NULL, hDlg, (LPPOINT)&r, 2);
76 :     VthTop = r.top;
77 :     //  VTH サイズ設定
78 :     GetClientRect(hDlg, &r);
79 :     SendMessage(hDlg, WM_SIZE, SIZE_RESTORED, MAKELONG(r.right - r.left, r.bottom - r.top));
80 :     //  VT100 右クリック禁止, キャレット非表示
81 :     AjcVthSetNtcRClk(hVth, TRUE, 0, 0);
82 :     AjcVthShowCaret(hVth, FALSE);
83 :     //  サンプルテキスト表示
84 :     AjcVthPrintF(hVth,  TEXT("¥n")
85 :                         TEXT("ここにテキストファイルをドロップすると、ファイルの内容を表示します。¥n")
86 :                         TEXT("¥n")
87 :                         TEXT("また、CTRL+V キーを押すと、クリップボードのテキストを表示します。¥n"));
88 :     //  自ダイアログ設定値ロード
89 :     AjcLoadAllControlSettings(hDlg, TEXT("Settings"), AJCCTL_SELACT_ALL);
90 :     //  フォント選択ダイアログ設定値 (永続化情報) 初期化
91 :     AjcCfInitPermInfo(NULL, NULL, &PermInfo);
92 :     //  フォント選択ダイアログ設定値 (永続化情報) ロード
93 :     AjcGetProfileStr(TEXT("FontSec"), TEXT("FontKey"), TEXT(""), txt, AJCCFMAX_TEXT);
94 :     AjcCfTextToInfo(txt, &PermInfo);
95 :
96 :     return TRUE;
97 : }
98 : //----- ウインド破棄 ----------------------------------------------------------------------------------------------//
99 : AJC_DLGPROC(Main, WM_DESTROY        )
100 : {
101 :     UT      txt[AJCCFMAX_TEXT];
102 :     //  ダイアログ設定値セーブ
103 :     AjcSaveAllControlSettings(hDlg);
104 :     //  フォント選択ダイアログ設定値 (永続化情報) セーブ
105 :     AjcCfInfoToText(&PermInfo, txt);
106 :     AjcPutProfileStr(TEXT("FontSec"), TEXT("FontKey"), txt);
107 :
108 :     //----- プログラム終了 --------------------------------------//
109 :     PostQuitMessage(0);
110 :     return TRUE;
111 : }
112 : //----- サイズ変更 ------------------------------------------------------------------------------------------------//
113 : AJC_DLGPROC(Main, WM_SIZE           )
114 : {
115 :     int     dw = LOWORD(lParam);
116 :     int     dh = HIWORD(wParam);
117 :     RECT    r;
118 :
119 :     GetClientRect(hDlg, &r);
120 :     SetWindowPos(hVth, NULL, 4, VthTop, r.right - 8, r.bottom - VthTop - 4, SWP_NOZORDER);
121 :
122 :     return TRUE;
123 : }
124 : //----- フォント設定ボタン ----------------------------------------------------------------------------------------//
125 : AJC_DLGPROC(Main, IDC_CMD_FONT      )
126 : {
127 :     AJCCFFONTINFO   fi;
128 :
129 :     if (AjcCfFontDlg(hDlg, 0, &fi, &PermInfo, (UX)hDlg, cbInit, cbApply)) {
130 :         AjcVthSetFontInfo(hVth, &fi.lf, fi.lsp);
131 :     }
132 :     return TRUE;
133 : }
134 : //----- INI ファイルを開くボタン ----------------------------------------------------------------------------------//
135 : AJC_DLGPROC(Main, IDC_CMD_INIOPEN   )
136 : {
137 :     UT      path[MAX_PATH];
138 :     AjcGetIniFilePath(path, MAX_PATH);
139 :     ShellExecute(NULL, TEXT("open"), path, NULL, NULL, SW_SHOWNORMAL);
140 :     return TRUE;
141 : }
142 : //----- VT100 ウインドからの通知 ----------------------------------------------------------------------------------//
143 : //  ダイアログ下の VT100 コントロールの場合、AJCVTHN_KEYIN は使用不可の為、AJCVTHN_VKEYIN,AJCVTHN_VKEYOUT で処理する
144 : AJC_DLGPROC(Main, IDC_VTH           )
145 : {
146 :     static BOOL fCtrl = FALSE;
147 :     UT      path[MAX_PATH];
148 :
149 :     switch (HIWORD(wParam)) {
150 :         case AJCVTHN_VKEYIN:                            //  ●キー押下
151 :             //  CTRL+V : 貼り付け
152 :             if (lParam == TEXT('V') && fCtrl) {
153 :                 UTP pTxt;
154 :                 if (pTxt = AjcCreateClipboardText()) {
155 :                     AjcVthClear(hVth);
156 :                     AjcVthPutText(hVth, pTxt, -1);
```

```
157 :                     AjcReleaseClipboardText(pTxt);
158 :                 }
159 :             }
160 :             //  CTRL キー押下処理
161 :             if (lParam == VK_CONTROL) fCtrl = TRUE;
162 :             break;
163 :
164 :         case AJCVTHN_VKEYOUT:                           //  ●キー離し
165 :             //  CTRL キー離し処理
166 :             if (lParam == VK_CONTROL) fCtrl = FALSE;
167 :             break;
168 :
169 :         case AJCVTHN_DROPFILE:                          //  ●ファイル ドロップ（ファイルの内容を描画）
170 :             AjcVthClear(hVth);
171 :             while (AjcVthGetDroppedFile(hVth, path)) {
172 :                 HAJCFILE    hFile;
173 :                 UT          wbuf[256];
174 :                 AjcVthPrintF(hVth, TEXT("¥r¥n¥x1B[34m[[[ FILE = %s ]]]¥x1B[0m¥r¥n"), path);
175 :                 if (hFile = AjcFOpen(path, AJCTEC_AUTO)) {
176 :                     while (AjcFGetS(hFile, wbuf, 256)) {
177 :                         AjcVthPutText(hVth, wbuf, -1);
178 :                     }
179 :                     AjcFClose(hFile);
180 :                 }
181 :             }
182 :             break;
183 :     }
184 :     return 0;
185 : }
186 : //-----  「Cancel」ボタン ----------------------------------------------------------------------------------------//
187 : AJC_DLGPROC(Main, IDCANCEL          )
188 : {
189 :     DestroyWindow(hDlg);
190 :     return TRUE;
191 : }
192 : //-----------------------------------------------------------------------------------------------------------------//
193 : AJC_DLGMAP_DEF(Main)
194 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG     )
195 :     AJC_DLGMAP_MSG(Main, WM_DESTROY        )
196 :     AJC_DLGMAP_MSG(Main, WM_SIZE           )
197 :
198 :     AJC_DLGMAP_CMD(Main, IDC_CMD_FONT      )
199 :     AJC_DLGMAP_CMD(Main, IDC_CMD_INIOPEN   )
200 :     AJC_DLGMAP_CMD(Main, IDC_VTH           )
201 :     AJC_DLGMAP_CMD(Main, IDCANCEL          )
202 : AJC_DLGMAP_END
203 :
204 : //-----------------------------------------------------------------------------------------------------------------//
205 : //  初期化コールバック                                                                                             //
206 : //-----------------------------------------------------------------------------------------------------------------//
207 : static  VO CALLBACK cbInit (HWND hDlgChoFont, UX cbp)
208 : {
209 :     HWND    hDlg = (HWND)cbp;
210 :     RECT    r;
211 :     GetWindowRect(hDlg, &r);
212 :     SetWindowPos(hDlgChoFont, NULL, r.right, r.top, 0, 0, SWP_NOSIZE);
213 : }
214 : //-----------------------------------------------------------------------------------------------------------------//
215 : //  適用コールバック                                                                                               //
216 : //-----------------------------------------------------------------------------------------------------------------//
217 : static  BOOL CALLBACK cbApply(HFONT hFont, UI lsp, UX cbp)
218 : {
219 :     HWND    hDlg = (HWND)cbp;
220 :     LOGFONT lf;
221 :     GetObject(hFont, sizeof lf, &lf);
222 :     AjcVthSetFontInfo(hVth, &lf, lsp);
223 :     return TRUE;
224 : }
```

63. その他

その他の汎用ＡＰＩ群です。

63.1. サポートＡＰＩ

その他のサポートＡＰＩ一覧を以下に示します。

#	関 数 名	内容
1	AjcSetLangId AjcGetLangId	言語種別設定／取得
2	AjcSetAppName	アプリケーション名の設定
3	AjcSetAppIcon	アプリケーション・アイコンの設定
4	AjcSetCmdWithHdl	コントロールからの通知時の lParam モード設定
5	AjcGetLastErrorText	GetLastErrot()で得たエラーコードが表す内容のテキストを取得
6	AjcDecToUI / ULL	符号なし１０進文字列を数値に変換
7	AjcAdjustMultValue	指定された数値を単位数値の整数倍に調整する
8	AjcAdjustPow10	指定された値の絶対値以下で、最も近い１０のｎ乗値を求める
9	AjcAdjustMultPow10 AjcRoundMultPow10	指定された値の絶対値以上で、最も近い「１０ⁿ×Ｎ」を求める（Nは1桁の整数） 指定された値の絶対値以上で、最も近い「１０ⁿ×Ｎ」を求める（末桁 = 0 or 5）
10	AjcExcWord/Long/LLong/Float/Double	エンディアン形式変換
11	AjcMAlloc	バイト単位のメモリ割り当て
12	AjcMFree	割り当てたメモリの解放
13	AjcSetMemError	メモリ割り当てエラー通知コールバックの設定
14	AjcMemSwap	メモリブロックの内容交換
15	AjcMemSet{8/16/32/64}	指定データでバッファーを埋める
16	AjcTAlloc	テキストメモリの確保
17	AjcTFree	テキストメモリの開放
18	AjcChgBright	明るさを変更した色を算出
19	AjcRgbBright	色の明るさ算出
20	AjcRgbIntensity	色の輝度算出
21	AjcRgbIntensityRatio	輝度比算出
22	AjcRgbDeffrence	色の差を算出
23	AjcComplementaryColor	補色／反対色を算出
24	AjcParseCmdArgs	コマンド引数のポインタ配列作成
25	AjcReleaseCmdArgs	コマンド引数ポインタ配列の開放
26	AjcParseCmdArgsEx	コマンド引数のポインタ配列作成・拡張版
27	AjcSkipCmdArgs	コマンドライン先頭の引数をスキップする
28	AjcGetCpuId	ＣＰＵのＩＤ情報取得
29	AjcDifferenceOfTheta	２つの角度値の変移角を求める
30	AjcCreateDllExportInfo	ＤＬＬのエクスポート情報取得
31	AjcReleaseDllExportInfo	ＤＬＬのエクスポート情報の開放
32	AjcCreateModImportInfo	自プロセス／ＤＬＬのインポート情報取得
33	AjcReleaseModImportInfo	自プロセス／ＤＬＬのインポート情報の開放
34	AjcGetSidStringByCurrentUser	現在のユーザのＳＩＤ文字列取得
35	AjcGetRidInSid	ＳＩＤ文字列からＲＩＤ値取得
36	AjcGetDupRect	２つの矩形の重なり合う部分を取得する
37	AjcAspGetZoomed{Rect/Size/Info}	アスペクト比を維持し拡大／縮小した矩形／サイズを求める
38	AjcGetParentProcess	親プロセスの情報取得
39	AjcExitWindows	Windows のシャットダウンやリブートを行う
40	AjcTrace	デバッガの出力ウインドへ書式文字列出力
41	AjcGetStockFont	ストック フォントの取得
42	AjcCreateFont	フォントハンドル生成
43	AjcGetVerInfo	実行ファイルのバージョン情報取得
44	AjcGetClipboardText	クリップボードテキストの取得
45	AjcCreateClipboardText	クリップボードテキストの生成
46	AjcReleaseClipboardText	クリップボードテキストの解放
47	AjcPutClipboardText	クリップボードへテキスト格納
48	AjcEnumMenuItems	メニューアイテムの列挙

63.1.1. 言語種別設定／取得 (Ajc{Set/Get}LangId)

形　式 ： VO　　AjcSetLangId(AJCLANGID Lid); - 言語種別設定
AJCLANGID AjcGetLangId(VO); ---------- 言語種別取得

引　数 ： Lid - AJCLID_JPN ： 日本語設定
AJCLID_ENG ： 英語設定

説　明 ： 本ライブラリ中で使用されるダイアログボックス等における、テキストの言語種別を設定／取得します。

戻り値 ： 設定時：なし　　取得時：AJCLID_JPN ： 日本語設定
AJCLID_ENG ： 英語設定

63.1.2. アプリケーション名の設定 (AjcSetAppName)

形　式 ： VO　　AjcSetAppName(C_UTP pName)

引　数 ： pName - アプリケーション名文字列のアドレス

説　明 ： アプリケーションの名称を設定します。
この名称は、本ライブラリ中で使用されるメッセージボックスのウインドタイトルに使用されます。

戻り値 ： なし

63.1.3. アプリケーション・アイコンの設定 (AjcSetAppIcon)

形　式 ： VO　　AjcSetAppIcon(HICON hIcon)

引　数 ： hIcon – アイコンハンドル

説　明 ： アプリケーションのアイコンを設定します。
このアイコンは、本ライブラリ中で使用されるダイアロボックス・タイトルバーに表示されます。

戻り値 ： なし
AJCLID_ENG ： 英語設定

63.1.4. コントロールからの通知時の lParam モード設定(AjcSetCmdWithHdl)

形　式 ： VO AjcSetCmdWithHdl (BOOL fWithHandle)

引　数 ： fWithHandle - コントロールからの通知 (WM_COMMAND) モード
FALSE ： lParam = イベントパラメタ
TRUE　： lParam = コントロールのウインドハンドル

説　明 ： 本ライブラリ中のコントロールからの通知(WM_COMMAND)における lParam 設定モードをセットします。
fWithHandle=FALSE とした場合は、lParam に当該イベントのパラメタを設定します。
fWithHandle=TRUE とした場合は、lParam にコントロールのウインドハンドルを設定します。

戻り値 ： なし

63.1.5. GetLastErrot()で得たエラーコードが表す内容のテキストを取得 (AjcGetLastErrorText)

形　式 ： UI AjcGetLastErrorText(int LastError, UTP pBuf, UI lBuf);

引　数 ： LastError - GetLastError()で取得したエラーコード
pBuf　　- エラーの内容を表すテキストを格納するバッファのアドレス
lBuf　　- エラーの内容を表すテキストを格納するバッファの文字数

説　明 ： GetLastError()で取得したエラーコードの内容を表すテキストを取得します。

戻り値 ： エラーの内容を表すテキストの文字数

63.1.6. 符号なし１０進文字列を数値に変換（AjcDecToUI / AjcDecToULL）

形　式 ： UI AjcDecToUI (C_UTP pStr);
ULL AjcDecToULL(C_UTP pStr);

引　数 ： pSrc – 符号なし１０進文字列の先頭アドレス

説　明 ： pSrc で示される１０進文字列（'0'～'9'）で構成される文字列）を数値に変換します。
文字列の先頭部分がスペース（0x09～0x0D or 0x20）である場合は、スペースをスキップします。
１０進文字以外が現れた時点で変換を終了します。
文字列の先頭が１０進文字以外である場合は、０を返します。
AjcDecToUI()の場合、数値が UINT_MAX を超える場合は、UINT_MAX を返します。

戻り値 ： １０進文字列が表す数値

63.1.7. 指定された、数値を単位数値の整数倍に調整する（AjcAdjustMultValue）

形　式 ： double AjcAdjustMultValue(double value, double unit)

引　数 ： value - 整数倍に調整する値
unit - 単位数値（正の数値）

説　明 ： unit を整数倍した値で、value に最も近い値を返します。
unit にゼロや負数を指定した場合は、ゼロを返します。

ex. AjcAdjustMultValue(98.6, 0.5); --> return = 98.5

戻り値 ： unit を整数倍した値で、value に最も近い値

63.1.8. 指定された値の絶対値以下で、最も近い１０のｎ乗値を求める（AjcAdjustPow10）

形　式 ： double AjcAdjustPow10(double val, UIP pDigit, UIP pPrec)

引　数 ： val - 入力値
pDigit - 整数部の桁数を格納するバッファのアドレス（不要時は NULL）
pPrec - 小数部の桁数を格納するバッファのアドレス（不要時は NULL）

説　明 ： val の絶対値以下で、最も近い「１０^{n}（ｎは整数)」の値を返します。
pDigit で示されるバッファには、戻り値の１０進表現の整数部の桁数（符号を含まない）が格納されます。
pPrec で示されるバッファには、戻り値の１０進表現の小数部の桁数（小数点を含まない）が格納されます。

ex. AjcAdjustPow10(2578.12, &digit, &prec); --> return = 1000.0, digit = 4, prec = 0
AjcAdjustPow10(-0.0234, &digit, &prec); --> return = 0.01 , digit = 1, prec = 2

戻り値 ： ｎの絶対値以下で、最も近い１０^{n}値を返します。

63.1.9. 指定された値の絶対値以上で、最も近い「１０ⁿ×Ｎ」を求める（Ajc[Adjust | Round]MultPow10）

形　式 ： double AjcAdjustMultPow10(double val, UIP pDigit, UIP pPrec, BOOL fEven);
double AjcRoundMultPow10 (double val, UIP pDigit, UIP pPrec);

引　数 ： val　　- 入力値
pDigit - 整数部の桁数を格納するバッファのアドレス（不要時はNULL）
pPrec　- 小数部の桁数を格納するバッファのアドレス（不要時はNULL）
fEven　- 末桁を偶数(0, 2, 4, 6 or 8)とする

説　明 ： valの絶対値以上で、最も近い「１０ⁿ×Ｎ（Ｎは１桁の整数)」の値を返します。
fEven=TRUEとした場合は、Ｎを偶数（２，４，６，８）とします。
fEven=FALSEとした場合は、Ｎを１～９とします。
pDigitで示されるバッファには、戻り値の１０進表現の整数部の桁数（符号を含まない）が格納されます。
pPrecで示されるバッファには、戻り値の１０進表現の小数部の桁数（小数点を含まない）が格納されます。
AjcRoundMultPow10()は、Ｎが「５」or「０」になるように調整します。

```
ex.  AjcAdjustMultPow10(2578.12, &digit, &prec, FALSE);  -->  return = 3000.0, digit = 4, prec = 0
     AjcAdjustMultPow10(-0.0234, &digit, &prec, TRUE );  -->  return = 0.04  , digit = 1, prec = 2

     AjcRoundMultPow10(-0.0234, &digit, &prec);  ---------->  return = 0.05  , digit = 1, prec = 2
```

戻り値 ： ｎの絶対値以上で、最も近い「１０ⁿ×Ｎ」を返します。

63.1.10. エンディアン形式変換（AjcExcWord / Long / LLong / Float / Double）

形　式 ： UW　　AjcExcWord　(UI　　x);
UL　　AjcExcLong　(UL　　x);
ULL　　AjcExcLLong (ULL　　x);
float　AjcExcFloat (float　x);
double　AjcExcDouble(double x);

引　数 ： x　- エンディアン形式を変換する数値

説　明 ： ｘで指定された数値のエンディアン形式（BigEndian / LittleEndian）を変換した値を返します。

戻り値 ： エンディアン形式を変換した値

63.1.11. バイト単位のメモリ割り当て（AjcMAlloc / AJCMEM）

形　式 ： VOP　　AjcMAlloc(size_t bytes, C_UTP pFName, UI lno);
　　AJCMEM(bytes);　--- マクロ

引　数 ： bytes　- 割り当てるメモリのバイト数
　　pFName - 呼び出し元のファイル名
　　lno　　- 呼び出し元の行番号

説　明 ： bytes で指定されたバイト数のメモリを割り当てて、その先頭アドレスを返します。
　　メモリの割り当てに失敗した場合は、AjcSetMemError()で設定されたメモリ割り当てエラー通知コールバック関数を呼び出して、NULL を返します。
　　AJCMEM()マクロは、以下のように展開します。

```
 AJCMEM(size);
+AjcMAlloc(size, __FILE__, __LINE__ );
```

戻り値 ： ≠NULL ： 割り当てたメモリの先頭アドレス
　　＝NULL ： メモリ割り当て失敗

コールバック：　メモリ割り当てエラー通知コールバック関数の仕様は、以下の通りです

形式 ： VO CALLBACK *cbNtcMemError*(C_UTP pFName, UI lno, C_UTP pVer, UX cbp) ; 、

引数 ： pFName - メモリエラーが発生したソースファイル名へのポインタ
　　lno　　- メモリエラーが発生したソースの行番号
　　pVer　 - ライブラリ・バージョン文字列へのポインタ
　　cbp　　- コールバックパラメタ

戻り値： なし

63.1.12. 割り当てたメモリの解放（AjcMFree）

形　式 ： VO　　AjcMFree(VOP pMem);
　　AJCFREE(pMem);　--- マクロ

引　数 ： pMem　- AjcMAlloc ()により割り当てたメモリ・アドレス (NULL 指定時は何もしない)

説　明 ： AjcMAlloc()により割り当てたメモリを解放します。
　　AJCFREE()マクロは、以下のように展開します。

```
 AJCFREE(pMem);
+AjcMFree(pMem);
```

戻り値 ： なし

63.1.13. メモリ割り当てエラー通知コールバックの設定（AjcSetMemError）

形　式 ： VO　　AjcSetMemError(UX cbp, VO (CALLBACK *cbNtcMemError)(C_UTP pFName, UI lno, C_UTP pVer, UX cbp));

引　数 ： cbp　　　　- コールバックパラメタ
　　cbNtcMemError - メモリ割り当てエラーを通知するコールバック関数のアドレス

説　明 ： AjcMAlloc()によるメモリ割り当てを失敗した場合に、メモリエラーを通知するコールバック関数を登録します。
　　本ライブラリ内でメモリエラーが発生した場合も、このコールバック関数が呼び出されます。
　　本ＡＰＩによりコールバック関数を設定しない場合は、メモリエラーを無視して処理を続行します。

戻り値 ： なし

63.1.14. メモリブロックの内容交換（AjcMemSwap）

形　式 ： VO　AjcMemSwap (VOP pMem1, VOP pMem2, UI len);

引　数 ： pMem1 – メモリブロック１のアドレス
pMem2 – メモリブロック２のアドレス
len　– メモリブロックのバイト数

説　明 ： ２つのメモリブロックを、(退避バッファを使用しないで）交換します。

戻り値 ： なし

63.1.15. 指定データでバッファーを埋める（AjcMemSet{8/16/32}）

形　式 ： VOP　AjcMemSet8 (VOP pBuf, UB dat, UI len);
VOP　AjcMemSet16(VOP pBuf, UW dat, UI len);
VOP　AjcMemSet32(VOP pBuf, UL dat, UI len);

引　数 ： pBuf – 指定データで埋めるバッファのアドレス
dat　– 埋めるデータ(8/16/32 bit)
len　– バッファのバイト数/ワード数／ロングワード数

説　明 ： pBuf で示すバッファを指定データ(dat)で埋めます。
len はバッファのデータ長をバイト(8bit)数／ワード(16bit)数／ロングワード(32bit)数で指定します。

戻り値 ： バッファの先頭アドレス (= pBuf)

63.1.16. テキストメモリの確保（AjcTAlloc）

形　式 ： UTP　AjcTAlloc(UI TextLength);

引　数 ： TextLength　- テキスト バイト数／文字数

説　明 ： 指定されたテキストのバイト数／文字数分のメモリブロックを確保します。
バイト文字バージョンの場合は、TextLength で指定されたバイト数のメモリブロックを確保します。
ワイド文字バージョンの場合は、TextLength で指定された文字数（TextLength×２バイト）のメモリブロックを確保します。
メモリの割り当てに失敗した場合は、AjcSetMemError()で設定されたコールバック関数を呼び出して、NULL を返します。

戻り値 ： ≠NULL：確保したメモリブロックのアドレス
＝NULL：メモリブロック確保失敗

63.1.17. テキストメモリの開放（AjcTFree）

形　式 ： VO　AjcTFree(UTP pMemBlk);

引　数 ： pMemBlk – 開放するメモリブロックのアドレス

説　明 ： AjcTAlloc で確保したメモリブロックを開放します。
pMemBlk には、AjcTAlloc で確保したメモリブロックのアドレス（AjcTAlloc の戻り値）を指定します。

戻り値 ： なし

63.1.18. 明るさを変更した色を算出 (AjcChgBright)

形　式 ： COLORREF　AjcChgBright (COLORREF rgb, UI per);

引　数 ： rgb　　－ ＲＧＢ色値
Per　　－ 明るさを変更する割合（%）

説　明 ： 色の要素（R，G，B）を、per で指定した割合に変更した色を取得します。
各要素が２５５を超える場合は、２５５とします。

戻り値 ： 明るさを変更した色

63.1.19. 色の明るさ算出(AjcRgbBright)

形　式 ： UI　AjcRgbBright(COLORREF rgb);

引　数 ： rgb　　－ ＲＧＢ色値

説　明 ： rgb で指定された色の明るさを、次式で求めます。

明るさ＝((R × 299) + (G × 587) + (B × 114)) / 1000

戻り値 ： 色の明るさ（０～２５５）

備　考 ： 見やすい明るさの差は、１２５以上が目安。

63.1.20. 色の輝度算出(AjcRgbIntensity)

形　式 ： double　AjcRgbIntensity(COLORREF rgb);

引　数 ： rgb　　－ ＲＧＢ色値

説　明 ： rgb で指定された色の輝度（ガンマ補正したリニアＲＧＢ値）を、次式で求めます。

$sX = X / 255$ （Xは、R, G, B）
$X' = (sX <= 0.03928) ? sX / 12.92 : ((sX + 0.055) / 1.055)^{2.4}$
} 「X」は、R,G,B

輝度 = R' × 0.2126 + G' × 0.2126 + B' × 0.2126

戻り値 ： 色の輝度（0.0 ～ 1.0）

63.1.21. 輝度比算出(AjcRgbIntensityRatio)

形　式 ： double　AjcRgbIntensityRatio(double L1, double L2);

引　数 ： L1, L2　－ 色の輝度

説　明 ： L1, L2 で指定された色の輝度比を、次式で求めます。

hi = max(L1, L2)
lo = min(L1, l2)
輝度比 = (hi + 0.05) / (lo + 0.05)

戻り値 ： 輝度比

備　考 ： 見やすい輝度比は、５～７以上が目安。

63.1.22. 色の差を算出(AjcRgbIntensityRatio)

形　式 ： double　AjcRgbDeffrence (COLORREF rgb1, COLORREF rgb2);

引　数 ： rgb１, rgb2　－ 差を算出するＲＧＢ値

説　明 ： 色の差を、右式で求めます。　　(色の差　= abs(R1 - R2) + abs(G1 - G2) + abs(B1 - B2))

戻り値 ： 色の差

備　考 ： 見やすい色の差は、５００以上が目安。

63.1.23. 補色／反対色を算出(AjcComplementaryColor)

形　式 ： COLORREF　AjcComplementaryColor (COLORREF rgb);

引　数 ： rgb　－ 補色を算出するＲＧＢ値

説　明 ： rgb の補色、あるいは、反対色を算出します。
補色と反対色を求め、色の差が大きい方の色を返します。

戻り値 ： 補色、あるいは、反対色

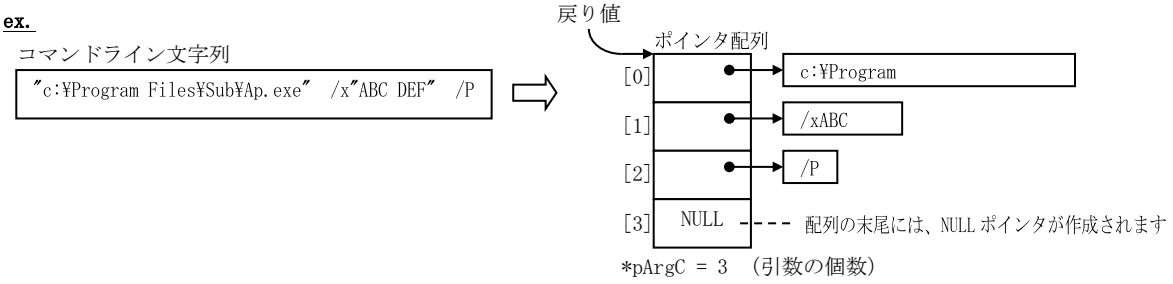
63.1.24. コマンド引数のポインタ配列作成 (AjcParseCmdArgs)

形　式 ： UTP　*AjcParseCmdArgs(C_UTP pCmd, UIP pArgC);

引　数 ： pCmd　－ コマンドライン文字列のアドレス
pArgC － 引数の個数を格納するバッファのアドレス（不要時は NULL）

説　明 ： コマンドライン文字列から、引数の配列を作成します。
pCmd で示される文字列から、空白やタブで区切られたトークン文字列へのポインタ配列を作成します。
但し、ダブルクオート(")で囲まれた文字列は（空白を含んでいても）一連の文字列として認識します。
ダブルクオート(")自身は、トークン文字列から除外されます。

ex.



正常に終了した場合は、(動的に確保した）引数へのポインタ配列の先頭アドレスを返します。
この、引数へのポインタ配列は、後で、AjcReleaseCmdArgs()により開放しなければなりません。
ポインタ配列の生成を失敗した場合は、NULL を返します。

戻り値 ： ≠NULL － 成功（引数へのポインタ配列の先頭アドレス）
＝NULL － 失敗

サンプルコード ： 次のコードは、コマンドラインの引数をすべて表示します。

```
UI   argc, i;
UTP  *argv;

argv = AjcParseCmdArgs(GetCommandLine(), &argc);
for (i=0; i<argc; i++) {
    AjcPrintF(TEXT("%s¥n"), argv[i]);
}
AjcReleaseCmdArgs(argv);
```

63.1.25. コマンド引数ポインタ配列の開放(AjcReleaseCmdARgs)

形　式　：　VO　AjcReleaseCmdArgs (VOP pArgV);

引　数　：　pArgV　- 引数へのポインタ配列の先頭アドレス（AjcParseCmdArgs()の戻り値）

説　明　：　AjcParseCmdArgs()で生成した、引数へのポインタ配列を開放します。

戻り値　：　なし

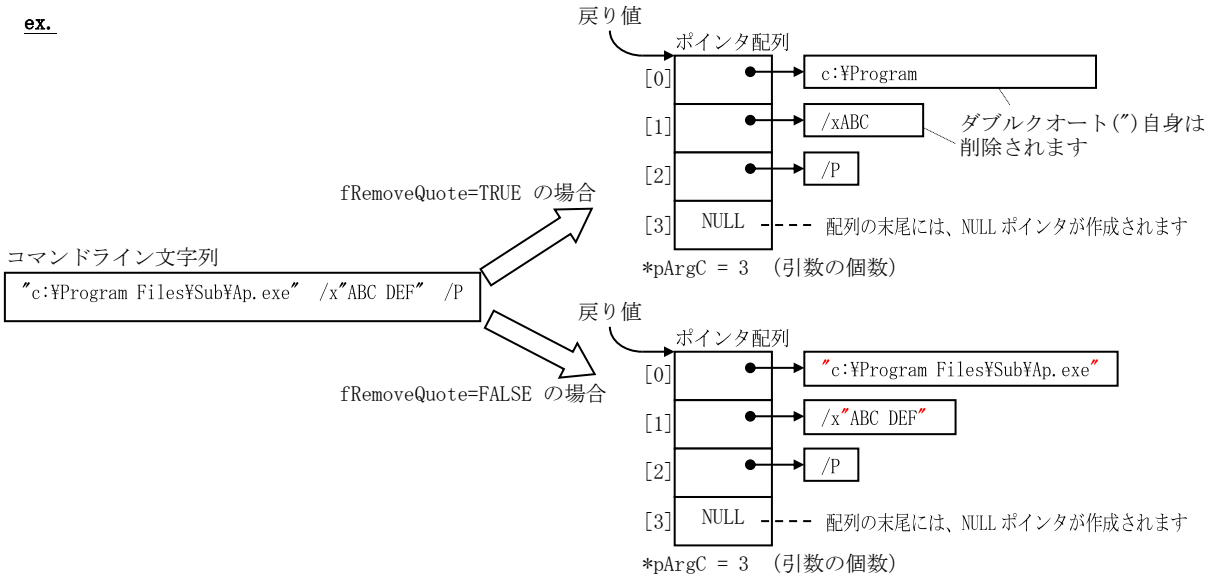
63.1.26. マンド引数のポインタ配列作成・拡張版 (AjcParseCmdArgsEx)

形　式　：　UTP　*AjcParseCmdArgsEx(C_UTP pCmd, UIP pArgC, BOOL fRemoveQuote);

引　数　：　pCmd　　- コマンドライン文字列のアドレス
pArgC　　- 引数の個数を格納するバッファのアドレス（不要時はNULL）
fRemoveQuote - ダブルクォート(")で囲まれた文字列の、両端のダブルクォート(")を削除するか否かの指定
（TRUE: 両端のダブルクォート(")を削除する，FALSE:削除しない）

説　明　：　コマンドライン文字列から、引数の配列を作成します。
pCmd で示される文字列から、空白（0x20, 0x09～0x0D）で区切られたトークン文字列へのポインタ配列を作成します。
但し、ダブルクオート(")で囲まれた文字列は（空白を含んでいても）一連の文字列として認識します。
fRemoveQuote=TRUE の場合、ダブルクオート(")自身は、トークン文字列から除外されます。

ex.



正常に終了した場合は、（動的に確保した）引数へのポインタ配列の先頭アドレスを返します。
この、引数へのポインタ配列は、後で、AjcReleaseCmdArgs()により開放しなければなりません。
ポインタ配列の生成を失敗した場合は、NULL を返します。

戻り値　：　≠NULL - 成功（引数へのポインタ配列の先頭アドレス）
＝NULL - 失敗

サンプルコード　：　次のコードは、コマンドラインの引数をすべて（ダブルクォート(")を削除しないで）表示します。

```
UI   argc, i;
UTP  *argv;

argv = AjcParseCmdArgsEx(GetCommandLine(), &argc, FALSE);
for (i=0; i<argc; i++) {
    AjcPrintF(TEXT("%s¥n"), argv[i]);
}
AjcReleaseCmdArgs(argv);
```

備　考　：　本関数は、「AjcParseCmdArgs」を拡張し、ダブルクオート(")の削除指定を追加したものであり、fRemoveQuote=TRUE とした場合は「AjcParseCmdArgs」とほぼ同じ操作を行います。
但し、「AjcParseCmdArgs」では、0x20 だけを空白として扱いますが、「AjcParseCmdArgsEx」では 0x20 と 0x09～0x0D を空白として扱います。

63.1.27. コマンドライン先頭の引数をスキップする (AjcSkipCmdARgs)

形　式 ： C_UTP　AjcSkipCmdArgs (C_UTP pCmd, UI n);

引　数 ： pCmd　- コマンドラインの先頭アドレス
n　　- スキップする引数の個数

説　明 ： コマンドラインの先頭引数をスキップしたポインタを返します。
引数は空白で木々られたものとします。
ダブルクォート(")で囲まれた文字列は、空白を含んでも連続した文字列と見なします。

戻り値 ： ≠NULL – 成功（先頭の引数（ｎ個）をスキップした次の引数先頭）
＝NULL – 失敗

63.1.28. ＣＰＵのＩＤ情報取得(AjcGetCpuId) ・・・Win32 のみ

形　式 ： UTP　AjcGetCpuId(UIP pSignature, UIP pParam, ULLP pSerialNo);

引　数 ： pSignature　- プロセッサのシグネチャ（あるいはシリアル番号の上位３２ビット）を格納するバッファのアドレス（不要時は NULL)
pParam　　- プロセッサ構成パラメタを格納するバッファのアドレス（不要時は NULL)
pSerialNo　- プロセッサシリアル番号の下位６４ビットを格納するバッファのアドレス（不要時は NULL)

説　明 ： CPUID 命令を実行し、ＣＰＵのＩＤ情報を取得します。
ＣＰＵのＩＤ情報を取得した場合は、各引数で指定されたバッファに当該情報を設定し、ベンダ名文字列の先頭アドレスを返します。(当該情報を取得できない場合は、０を格納します)
ＣＰＵのＩＤ情報を取得できない（CPUID 命令を実行できない）場合は、NULL を返します。

戻り値 ： ≠NULL ： ベンダ名文字列のアドレス
＝NULL ： ＣＰＵのＩＤ情報を取得できない

63.1.29. ２つの角度値の変移角を求める (AjcDifferenceOfTheta)

形　式 ： double AjcDifferenceOfTheta(double t1, double t2);

引　数 ： t1, t2　- 、角度の差を求める２つの角度値[度]

説　明 ： ２つの角度値 t1→t2 の変移角を求めます。

```
（例）  t1 =   10, t2 =   20 -->  10.0
        t1 =   20, t2 =   10 --> -10.0
        t1 =  350, t2 =   10 -->  20.0
        t1 =   10, t2 =  350 --> -20.0
        t1 = -170, t2 = +150 --> -40.0
        t1 = +150, t2 = -170 -->  40.0
        t1 = -180, t2 = +180 -->   0.0
        t1 =  720, t2 =  360 -->   0.0
```

戻り値 ： t1→t2 の変移角度値［度］

63.1.30. ＤＬＬのエクスポート情報取得(AjcCreateDllExportInfo)

形　式　：　PAJCDLLEXPORTINFO　AjcCreateDllExportInfo(HMODULE hDll, UIP pNum);

引　数　：　hDll　－ ＤＬＬのモジュールハンドル（LoadLibrary()で取得したハンドル）
pNum　－ エクスポート情報の個数を格納するバッファのアドレス

説　明　：　ＤＬＬがエクスポートしている関数の情報を取得します。
正常に終了した場合は、AJCDLLEXPORTINFO 型の配列のアドレスを返し、pNum で示されるバッファに配列のエントリ数を格納します。
本関数が返した AJCDLLEXPORTINFO 型の配列は、使用後に、AjcReleaseDllExportInfo()により開放する必要があります。
エクスポート情報が無い場合や、エラーが発生した場合は、NULL を返します。
AJCDLLEXPORTINFO 構造体の形式は、以下のとおりです。

```
typedef struct {
    int     Ordinal;            //  序数
    C_UTP   pFuncName;          //  エクスポート関数名（文字列）へのポインタ
    VOP     pFuncAddr;          //  関数のアドレス
} AJCDLLEXPORTINFO, *PAJCDLLEXPORTINFO;
```

※ 序数だけのエクスポート関数の場合は、pFuncName=NULL となります。

戻り値　：　≠NULL － ＤＬＬエクスポート情報（AJCDLLEXPORTINFO 型の配列）のアドレス
＝NULL － エラー／エクスポート情報なし

（例）　：　以下のサンプルコードは、「XXX.dll」のエクスポート情報を取得し、プリントします。

```
#include    <AjrCst32.h>
#include    <stdio.h>
#include    <tchar.h>

main()
{
    HMODULE             hDll;
    PAJCDLLEXPORTINFO   p, pInfo;
    UI                  i, n;

    hDll = LoadLibrary(TEXT("XXX.dll"));
    if (pInfo = AjcCreateDllExportInfo(hDll, &n)) {
        for (i=0, p=pInfo; i<n; i++, p++) {
            _tprintf(TEXT("%5d, %-30s, %p¥n"), p->Ordinal, p->pFuncName, p-
>pFuncAddr);
        }
        AjcReleaseDllExportInfo(pInfo);
    }
    FreeLibrary(hDll);
}
```

63.1.31. ＤＬＬのエクスポート情報の開放(AjcReleaseDllExportInfo)

形　式　：　VO　AjcReleaseDllExportInfo(PAJCDLLEXPORTINFO pInfo);

引　数　：　pInfo　－ ＤＬＬエクスポート情報のアドレス（AjcCreateDllExportInfo()の戻り値）

説　明　：　AjcCreateDllExportInfo()で取得した、ＤＬＬエクスポート情報（配列領域）を開放します。

戻り値　：　なし

63.1.32. モジュール（自プロセス／ＤＬＬ）のインポート情報取得(AjcCreateModImportInfo)

形　式 ： PAJCMODIMPORTINFO　AjcCreateModImportInfo(HMODULE hMod, UIP pNum);

引　数 ： hMod　- 自プロセス／ＤＬＬのモジュールハンドル（GetModuleHandle() / LoadLibrary() で取得したハンドル）
pNum　- インポート情報の個数を格納するバッファのアドレス

説　明 ： 自プロセス／ＤＬＬがインポートしている関数（参照しているＤＬＬ関数）の情報を取得します。
自プロセスのインポート情報を取得する場合は、hMod = GetModuleHandle(NULL); とします。
ＤＬＬのインポート情報を取得する場合は、hMod = LoadLibrary(TEXT("*XXX*.dll")); とします。
正常に終了した場合は、AJCMODIMPORTINFO 型の配列のアドレスを返し、pNum で示されるバッファに配列のエントリ数を格納します。
本関数が返した AJCMODIMPORTINFO 型の配列は、使用後に、AjcReleaseModImportInfo() により開放する必要があります。
エクスポート情報が無い場合や、エラーが発生した場合は、NULL を返します。
AJCMODIMPORTINFO 構造体の形式は、以下のとおりです。

```
typedef struct {
    C_UTP   pDllName;           //  ＤＬＬ名へのポインタ
    int     Ordinal;            //  序数
    C_UTP   pFuncName;          //  エクスポート関数名へのポインタ
    VOP     pFuncAddr;          //  関数のアドレス
} AJCMODIMPORTINFO, *PAJCMODIMPORTINFO;
```

※ 関数名によるインポート関数の場合は、Ordinal=0 となります。
※ 序数によるインポート関数の場合は、pFuncName=NULL となります。

戻り値 ： ≠NULL -モジュールのインポート情報（AJCMODIMPORTINFO 型の配列）のアドレス
＝NULL - エラー／インポート情報なし

（例） ： 以下のサンプルコードは、自プロセスのインポート情報を取得し、プリントします。

```
#include    <AjrCst32.h>
#include    <stdio.h>
#include    <tchar.h>

main()
{
    HMODULE             hMod;
    PAJCMODIMPORTINFO   p, pInfo;
    UI                  i, n;

    hMod = GetModuleHandle(NULL);
    pInfo = AjcCreateModImportInfo(hMod, &n);
    for (i=0, p=pInfo; i<n; i++, p++) {
        _tprintf(TEXT("%-20s, %5d, %-30s, %p¥n"), p->pDllName, p->Ordinal, p->pFuncName, p-
>pFuncAddr);
    }
    AjcReleaseModImportInfo(pInfo);
}
```

63.1.33. モジュール（自プロセス／ＤＬＬ）のインポート情報の開放(AjcReleaseModImportInfo)

形　式 ： VO　AjcReleaseModImportInfo(PAJCMODIMPORTINFO pInfo);

引　数 ： pInfo　- モジュールのインポート情報のアドレス（AjcCreateModImportInfo()の戻り値）

説　明 ： AjcCreateModImportInfo()で取得した、モジュールのインポート情報（配列領域）を開放します。

戻り値 ： なし

63.1.34. 現在のユーザ（アカウント）のＳＩＤ取得(AjcGetSidStringByCurrentUser)

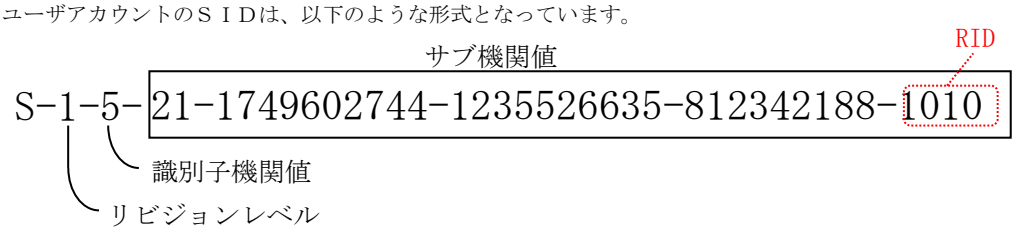
形　式 ： UI　AjcGetSidStringByCurrentUser (UTP pSidBuf, UI lSidBuf);

引　数 ： pSidBuf　－ ＳＩＤ文字列を格納するバッファのアドレス（不要時はＮＵＬＬ）
lSidBuf　－ ＳＩＤ文字列を格納するバッファの文字数　　（不要時は０）

説　明 ： 現在のユーザ名（アカウント名）に対するＳＩＤを文字列で取得します。
pSidBuf=NULL とした場合は、単にＳＩＤ文字列長を返します。
必要なバッファサイズは、このＳＩＤ文字列長に、文字列終端分の１文字分追加したサイズとなります。

戻り値 ： ≠０ － 成功（ＳＩＤの文字列長）
＝０ － 失敗

備　考 ： ユーザアカウントのＳＩＤは、以下のような形式となっています。



63.1.35. ＳＩＤ文字列からＲＩＤ値取得(AjcGetRidInSid)

形　式 ： int　AjcGetRidInSid (UTP pSid);

引　数 ： pSid　－ ＳＩＤ文字列のアドレス

説　明 ： ＳＩＤ内のＲＩＤ値を取得します。

戻り値 ： ≠０ － 成功（ＲＩＤ値）
＝０ － 失敗

63.1.36. ２つの矩形の重なり合う部分を取得する (AjcGetDupRect)

形　式 ： UI　AjcGetDupRect(LPCRECT pRect1, LPCRECT pRect2, LPRECT pDupRect);

引　数 ： pRect1, pRect2 － ２つの矩形情報
pDupRect　　－ ２つの矩形が重なり合う部分の矩形情報（不要時は NULL）

説　明 ： ２つの矩形の重なり合う部分の情報を取得します。
重なり合う部分がある場合は、その部分矩形の情報を pDupRect で示すバッファに格納し、矩形の面積を返します。
重なり合う部分が無い場合は、pDupRect にすべて０を設定し、面積＝０を返します。

戻り値 ： ２つの矩形が重なり合う部分の面積（ピクセル数）

63.1.37. アスペクト比を維持し拡大縮小した矩形／サイズを求める(AjcAspGetZoomed{Rect/Size/Info})

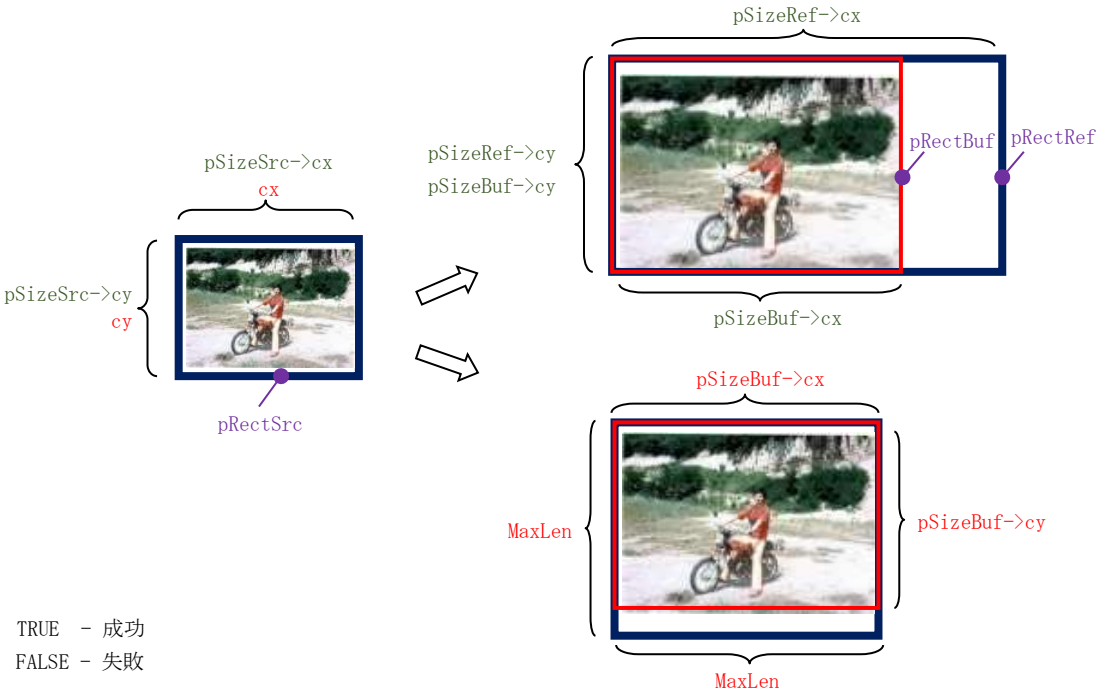
形　式 ： BOOL　AjcAspGetZoomedRect(LPCRECT pRectSrc, LPCRECT pRectRef, LPRECT pRectBuf);
BOOL　AjcAspGetZoomedSize(const SIZE *pSizeSrc, const SIZE *pSizeRef, LPRECT pSizeBuf);
BOOL　AjcAspGetZoomedInfo(int cx, int cy, int MaxLen, LPSIZE pSizeBuf);

引　数 ： pRectSrc　- 入力側矩形情報
pRectRef　- 出力側矩形情報
pRectBuf　- アスペクト比を維持し拡大／縮小した矩形情報を格納するバッファのアドレス

pSizeSrc　- 入力側サイズ情報
pSizeRef　- 出力側サイズ情報
pSizeBuf　- アスペクト比を維持し拡大／縮小したサイズ情報を格納するバッファのアドレス

cx　- 入力側の幅
cy　- 入力側の幅
MaxLen　- 出力側の一辺の最大長
pSizeBuf　- アスペクト比を維持し拡大／縮小したサイズ情報を格納するバッファのアドレス

説　明 ： アスペクト比を維持し、拡大／縮小した矩形／サイズを算出します。
AjcAspGetZoomedRect()／AjcAspGetZoomedSize()の場合は、短辺，長編において、拡大縮小率の小さい方で拡大／縮小する。
AjcAspGetZoomedInfo()の場合は、入力長編を MaxLen に合わせて拡大／縮小します。



戻り値 ： TRUE　- 成功
FALSE - 失敗

63.1.38. 親プロセスの情報取得 (AjcGetParentProcess)

形　式 ： BOOL　AjcGetParentProcess(UL pid, ULP pParent, UTP pBuf, UI lBuf);

引　数 ： pid　　- 子プロセスＩＤ（０の場合は、自プロセスＩＤを仮定）
pParent - 親プロセスＩＤを格納するバッファのアドレス（不要時はＮＵＬＬ）
pBuf　　- 親プロセスの実行ファイル名を格納するバッファのアドレス（不要時はＮＵＬＬ）
lBuf　　- 親プロセスの実行ファイル名を格納するバッファのバイト数／文字数

説　明 ： 親プロセスの情報（プロセスＩＤ，実行ファイル名）を取得します。
pBuf で示すバッファには、親プロセスのファイル名が設定されます。

戻り値 ： TRUE　：成功
FALSE：親プロセスが見つからない

備　考 ： エクスプローラで起動したプログラムから本ＡＰＩを実行すると pBuf に“explorer.exe”が設定されます。
VisualStudio のプロジェクトで起動したプログラムから本ＡＰＩを実行すると、pBuf に“devenv.exe”が設定されます。

63.1.39. Windows のシャットダウンやリブートを行う(AjcExitWindows[Ex])

形　式 ： BOOL　AjcExitWindows　(UI flag);
BOOL　AjcExitWindowsEx(UI flag, UI reason);

引　数 ： flag　- 動作フラグ
reason - シャットダウンする理由（詳細は MSDN 参照）

説　明 ： システム(Windows)のシャットダウンや、再起動を行います。
flag は、動作指定であり、以下のシンボルを指定します。(EWX_FORCE は他のシンボルと組み合わせて指定)

#	シンボル	内容	備考
1	EWX_LOGOFF	現在のユーザをログオフする	
2	EWX_POWEROFF	システムをシャットダウンし、電源を切ります	
3	EWX_REBOOT	システムをシャットダウンした後、再起動します	
4	EWX_SHUTDOWN	システムをシャットダウンし、電源を切っても良い状態にします	
5	EWX_FORCE	プロセスを強制的に終了させます	

Reason は、シャットダウンする理由を指定します。AjcExitWindows()の場合は、以下の合成値が仮定されます。

SHTDN_REASON_MAJOR_APPLICATION　- アプリケーションの問題
SHTDN_REASON_MINOR_MAINTENANCE　- メンテナンス
SHTDN_REASON_FLAG_PLANNED　　- シャットダウンが計画されました

戻り値 ： TRUE　- 成功（シャットダウンが開始された。シャットダウンが成功したかどうかは示されない）
FALSE - 失敗

63.1.40. デバッガの出力ウインドへ書式文字列出力(AjcTrace)

形　式 ： UI　AjcTrace(C_BCP pFmt, ...);

引　数 ： pFmt - 書式文字列（printf と同じ）

説　明 ： デバッガ（VisualStudio）の出力ウインドに書式化した文字列（最大 1023 バイト／文字）を出力します。

戻り値 ： TRUE　：成功
FALSE：失敗

63.1.41. ストック フォントの取得(AjcGetStockFont)

形　式 ： HFONT　AjcGetStockFont(AJCFONTID fid);

引　数 ： fid - フォント種別

説　明 ： ライブラリで保持しているフォントハンドルを取得します。
fid には、以下のいずれかのシンボルを指定します。

シンボル	フォント名
AJCFID_FIX10	MS Gothic （固定ピッチ）
AJCFID_FIX12	
AJCFID_FIX14	
AJCFID_FIX16	
AJCFID_FIX18	
AJCFID_FIX20	
AJCFID_FIX22	
AJCFID_FIX24	

シンボル	フォント名
AJCFID_VAR10	Arial Unicode MS （可変ピッチ）
AJCFID_VAR12	
AJCFID_VAR14	
AJCFID_VAR16	
AJCFID_VAR18	
AJCFID_VAR20	
AJCFID_VAR22	
AJCFID_VAR24	

※ シンボルの末尾の数字（10～24）はフォントの高さ（ピクセル）を意味します。

戻り値 ： ≠NULL ： 成功（フォントハンドル）
＝NULL ： 失敗

63.1.42. フォントハンドル生成(AjcCreateFont)

形　式 ： HFONT　AjcCreateFont(LPLOGFONT pLf);

引　数 ： pLf - フォント情報へのポインタ

説　明 ： 指定されたフォント情報からフォントハンドルを生成します。
pLf=NULL を指定した場合や、フォントが生成できない場合は、システムフォント(“System”)と同様のフォントハンドルを生成します。

pLf で指定されたフォント情報の以下の項目は、正しい値に訂正されます。

#	項目名	内容
1	lfPitchAndFamily	ピッチとファミリ情報
2	lfFaceName	フォントフェース名（英字名の場合は、元のフェース名に変換される）

戻り値 ： ≠NULL ： 成功（フォントハンドル）
＝NULL ： 失敗（基本的に NULL を返すことは無い）

63.1.43. 実行ファイルのバージョン情報取得(AjcGetVerInfo)

形　式 ： UI　AjcGetVerInfo(C_UTP pPath, C_UTP pInfoName, UTP pInfoBuf, int lInfoBuf)；

引　数 ： pPath　　- 実行ファイルのパス名　　(ex. "C:¥¥WINDOWS¥¥system32¥¥cmd.exe")
pInfoName - 取得するバージョン情報の名称 (ex. "ProductVersion")
pInfoBuf　- 取得したバージョン情報を格納するバッファのアドレス (不要時は NULL)
lInfoBuf　- 取得したバージョン情報を格納するバッファのサイズ (バイト数／文字数)

説　明 ： 実行ファイル(.exe や .dll)内のバージョンリソースから情報を読み出します。
pInfoName には、以下の記号名称／文字列を指定します。

記号名称	文字列
AJCVERINFO_COMMENTS	"Comments"
AJCVERINFO_INTERNAL_NAME	"InternalName"
AJCVERINFO_PRODUCT_NAME	"ProductName"
AJCVERINFO_COMPANY_NAME	"CompanyName"
AJCVERINFO_LEGAL_COPYLIGHT	"LegalCopyright"
AJCVERINFO_PRODUCT_VERSION	"ProductVersion"

記号名称	文字列
AJCVERINFO_FILE_DESCRIPTION	"FileDescription"
AJCVERINFO_LEGAL_TRADEMARKS	"LegalTrademarks"
AJCVERINFO_PRIVATE_BUILD	"PrivateBuild"
AJCVERINFO_FILE_VERSION	"FileVersion"
AJCVERINFO_ORIGINAL_FILENAME	"OriginalFilename"
AJCVERINFO_SPECIAL_BUILD	"SpecialBuild"

pInfoBuf=NULL とした場合は、単に、当該情報の文字列終端を含むサイズ (バイト数／文字数) を返します、

戻り値 ： ≠０：成功 (文字列を格納するのに必要なサイズ (バイト数／文字数))
＝０：失敗／当該情報無し

63.1.44. クリップボードテキストの取得(AjcGetClipboardText)

形　式 ： UI　AjcGetClipboardText(UTP pBuf, UI lBuf);

引　数 ： pBuf　　- クリップボードテキストを格納するバッファへのポインタ (不要時は NULL)
lBuf　　- クリップボードテキストを格納するバッファの文字数

説　明 ： クリップボードからテキストを読み出して pBuf で示すバッファに格納します。
pBuf=NULL とした場合は、単にクリップボードテキストの文字数を返します。

戻り値 ： ≠０：クリップボードテキストの文字数 (終端('¥0')を含む)
＝０：失敗

63.1.45. クリップボードテキストの生成(AjcCreateClipboardText)

形　式 ： UTP　AjcCreateClipboardText(VO);

引　数 ： なし

説　明 ： クリップボードから読み出したテキストを動的に確保したメモリに格納し、テキストへのポインタを返します。
このポインタは、読み出したテキスト使用後に、AjcReleaseClipboardText()で解放しなければなりません。

戻り値 ： ≠NULL：クリップボードから読み出したテキストへのポインタ
＝NULL：失敗

63.1.46. クリップボードテキストの解放(AjcReleaseClipboardText)

形　式 ： BOOL　AjcReleaseClipboardText(UTP pTxt);

引　数 ： pTxt - 生成したクリップボードテキストへのポインタ (AjcCreateClipboardText()の戻り値)

説　明 ： AjcCreateClipboardText()で生成したテキストを解放します。

戻り値 ： TRUE　：成功
FALSE：失敗

63.1.47. クリップボードへテキスト格納(AjcPutClipboardText)

形　式 ： BOOL　AjcPutClipboardText (C_UTP pTxt, UI lTxt);

引　数 ： pTxt - クリップボードへ格納するテキストへのポインタ
lTxt - クリップボードへ格納するテキストの文字数（-1：自動設定）

説　明 ： pTxt で指定されたテキストをクリップボードへ設定します。

戻り値 ： TRUE ：成功
FALSE：失敗

63.1.48. メニューアイテムの列挙(AjcEnumMenuItems)

形　式 ： UI　AjcEnumMenuItems (HMENU hMenu, BOOL fSubMenu, UX cbp,
BOOL (CALLBACK *cbEnumMenu)(HMENU hMenu, UI ix, UX id, UI type, UI state, HMENU hSubMenu, UX cbp));

引　数 ： hMenu　　- メニューハンドル
fSUbMenu　- サブメニュー列挙フラグ（TRUE:サブメニューも列挙する，FALSE:サブメニューは列挙しない）
cbp　　- コールバックパラメタ
cbEnumMenu - 個々のメニューアイテム通知用コールバック関数

説　明 ： hMenu で指定されたメニューのメニュー項目を列挙します。

戻り値 ： 列挙した個数

コールバック： コールバック関数の仕様は、以下のとおりです。

cbEnumMenu（メニューアイテム通知）

形　式 ： BOOL　CALLBACK ***cbNEnumMenu*** (HMENU hMenu, UI ix, UX id, UI type, UI state, HMENU hSubMenu, UX cbp)；

引　数 ： hMenu　- メニューハンドル（サブメニュー項目の場合は、サブメニューハンドル）
ix　　- メニュー項目の位置（先頭項目を０としたメニュー項目の位置）
id　　- メニューＩＤ
type　- メニュー項目の種類（MFT_*XXXXX*）
state　- メニュー項目の状態（MFS_*XXXXX*）
hSubMenu - サブメニューハンドル（このメニュー項目にサブメニューが無い場合は NULL）
cbp　　- コールバックパラメタ

説　明 ： メニュー項目を通知します。
FALSE を返すと、メニュー項目の列挙を中止します。

戻り値 ： TRUE　- メニュー項目列挙継続
FALSE - メニュー項目列挙中止

64. ＭＦＣでの利用

ＭＦＣ(Microsoft Foundation Class)から本ライブラリ(AjrCst32.dll/AjrCst64.dll)のカスタムコントロールを呼び出す場合は、プログラムの最初に以下のコードを実行してください。

```
AjcSetCmdWithHdl(TRUE);
```

（C++ラッパークラスライブラリ「AjxCpp32.dll/AjxCpp64.dll」を使用する場合は、上記コードを実行する必要はありません。）

本ライブラリのカスタムコントロールからの通知メッセージ（WM_COMMAND）では、lParam に通知内容に伴うパラメタが設定されますが、ＭＦＣを使用する場合、コントロールからの通知メッセージの lParam がウインドハンドルかチェックされます。
lParam がウインドハンドルでない場合は ASSERT されて、プログラムを実行することができません。

「AjcSetCmdWithHdl(TRUE);」を実行すると、カスタムコントロールからの通知メッセージ（WM_COMMAND）の lParam は常に当該コントロールのウインドハンドルが設定されるようになります。
この場合、通知内容に伴うパラメタは、各カスタムコントロールの「通知情報取得」ＡＰＩで取得します。

カスタムコントロールからの通知は、メッセージマップ（BEGIN_MESSAGE_MAP～END_MESSAGE_MAP）中に ON_CONTROL マクロで直接編集します。(カスタムコントロールからの通知は、ウイザードで生成できません)

64.1. ＭＦＣサンプルプログラム１（S_MFC_01）

ＭＦＣで本ライブラリ（AjrCst32.dll/AjrCst64.dll）を使用したサンプルプログラムを示します。
このサンプルプログラムは、シリアルポート，メールスロットやソケットサーバで回線を接続し、受信したテキストデータを表示します。
受信テキストデータは、テキストコード（S-JIS / UTF-8 / EUC-J）を自動判別します。
ファイルをドロップすると、当該ファイルの内容を UTF-8 テキストデータとして、接続した回線に送信します。
フォルダをドロップすると、当該フォルダのパスを表示します。

プログラムの外観は以下のとおりです。



このサンプルプログラムでは、本ライブラリの C++ラッパークラスを使用しています。
ラッパークラスについては「AjrCppClass.pdf」を参照してください。

●S_MFC_01.cpp

```
 1 :
 2 : // S_MFC_01.cpp : アプリケーションのクラス動作を定義します。
 3 : //
 4 :
 5 : #include "stdafx.h"
 6 : #include "S_MFC_01.h"
 7 : #include "S_MFC_01Dlg.h"
 8 :
 9 : #include <AjxCpp.h>
10 :
11 : #ifdef _DEBUG
12 : #define new DEBUG_NEW
13 : #endif
14 :
15 :
16 : // CS_MFC_01App
17 :
18 : BEGIN_MESSAGE_MAP(CS_MFC_01App, CWinApp)
19 :     ON_COMMAND(ID_HELP, &CWinApp::OnHelp)
20 : END_MESSAGE_MAP()
21 :
22 :
23 : // CS_MFC_01App コンストラクション
24 :
25 : CS_MFC_01App::CS_MFC_01App()
26 : {
27 :     // TODO: この位置に構築用コードを追加してください。
28 :     // ここに InitInstance 中の重要な初期化処理をすべて記述してください。
29 :
30 :     AjcSetCmdWithHdl(TRUE);             //  ●ＭＦＣで実行する場合に必要な初期化
31 :     AjcSetProfileIsRegistory(TRUE);     //  ●プロファイルの記録先＝レジストリ
32 : }
33 :
34 :
35 : // 唯一の CS_MFC_01App オブジェクトです。
36 :
37 : CS_MFC_01App theApp;
38 :
39 :
40 : // CS_MFC_01App 初期化
41 :
42 : BOOL CS_MFC_01App::InitInstance()
43 : {
44 :     CWinApp::InitInstance();
45 :
46 :
47 :     // ダイアログにシェル ツリー ビューまたはシェル リスト ビュー コントロールが
48 :     // 含まれている場合にシェル マネージャーを作成します。
49 :     CShellManager *pShellManager = new CShellManager;
50 :
51 :     // 標準初期化
```

```
52 :     // これらの機能を使わずに最終的な実行可能ファイルの
53 :     // サイズを縮小したい場合は、以下から不要な初期化
54 :     // ルーチンを削除してください。
55 :     // 設定が格納されているレジストリ キーを変更します。
56 :     // TODO: 会社名または組織名などの適切な文字列に
57 :     // この文字列を変更してください。
58 :     SetRegistryKey(_T("アプリケーション ウィザードで生成されたローカル アプリケーション"));
59 :
60 :     CS_MFC_01Dlg dlg;
61 :     m_pMainWnd = &dlg;
62 :     INT_PTR nResponse = dlg.DoModal();
63 :     if (nResponse == IDOK)
64 :     {
65 :         // TODO: ダイアログが〈OK〉で消された時のコードを
66 :         //  記述してください。
67 :     }
68 :     else if (nResponse == IDCANCEL)
69 :     {
70 :         // TODO: ダイアログが〈キャンセル〉で消された時のコードを
71 :         //  記述してください。
72 :     }
73 :
74 :     // 上で作成されたシェル マネージャーを削除します。
75 :     if (pShellManager != NULL)
76 :     {
77 :         delete pShellManager;
78 :     }
79 :
80 :     // ダイアログは閉じられました。アプリケーションのメッセージ ポンプを開始しないで
81 :     //  アプリケーションを終了するために FALSE を返してください。
82 :     return FALSE;
83 : }
84 :
```

●S_MFC_01Dlg.cpp

```
 1 :
 2 : // S_MFC_01Dlg.cpp : 実装ファイル
 3 : //
 4 :
 5 : #include "stdafx.h"
 6 : #include "S_MFC_01.h"
 7 : #include "S_MFC_01Dlg.h"
 8 : #include "afxdialogex.h"
 9 :
10 : #ifdef _DEBUG
11 : #define new DEBUG_NEW
12 : #endif
13 :
14 : // CS_MFC_01Dlg ダイアログ
15 :
16 : CS_MFC_01Dlg::CS_MFC_01Dlg(CWnd* pParent /*=NULL*/)
17 :     : CDialogEx(CS_MFC_01Dlg::IDD, pParent)
18 : {
19 :
20 :     m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
21 : }
22 :
23 : void CS_MFC_01Dlg::DoDataExchange(CDataExchange* pDX)
24 : {
25 :     CDialogEx::DoDataExchange(pDX);
26 : }
27 :
28 : BEGIN_MESSAGE_MAP(CS_MFC_01Dlg, CDialogEx)
29 :     ON_WM_PAINT()
30 :     ON_WM_QUERYDRAGICON()
31 :     ON_BN_CLICKED(IDC_CMD_OPEN , &CS_MFC_01Dlg::OnBnClickedCmdOpen )
32 :     ON_BN_CLICKED(IDC_CMD_CLOSE, &CS_MFC_01Dlg::OnBnClickedCmdClose)
33 :     ON_BN_CLICKED(IDC_CMD_PORT , &CS_MFC_01Dlg::OnBnClickedCmdPort )
34 : END_MESSAGE_MAP()
35 :
36 : // CS_MFC_01Dlg メッセージ ハンドラー
37 :
38 : BOOL CS_MFC_01Dlg::OnInitDialog()
39 : {
40 :     CDialogEx::OnInitDialog();
41 :
42 :     // このダイアログのアイコンを設定します。アプリケーションのメイン ウィンドウがダイアログでない場合、
43 :     //  Framework は、この設定を自動的に行います。
44 :     SetIcon(m_hIcon, TRUE);         // 大きいアイコンの設定
45 :     SetIcon(m_hIcon, FALSE);        // 小さいアイコンの設定
46 :
47 :     // TODO: 初期化をここに追加します。
48 :     m_VthScp.SetUp(std::bind(&CS_MFC_01Dlg::EnableOpenButton , this, std::placeholders::_1),
49 :                    std::bind(&CS_MFC_01Dlg::EnableCloseButton, this, std::placeholders::_1));
50 :     m_VthScp.Attach(GetDlgItem(IDC_VTH)->m_hWnd);
51 :     m_VthScp.Init   (TEXT("MyScpParam"), AJCSCP_CM_TEXT);
52 :     m_VthScp.SetTxTextCode(AJCSCP_TXT_UTF8);
```

```
53 :     m_VthScp.SetRxTextCode(AJCSCP_TXT_AUTO);
54 :     m_VthScp.SetEvtMask(AJCSCP_EV_PORTSTATE | AJCSCP_EV_RXCHUNK);
55 :
56 :     m_VthScp.PrintF(TEXT("・シリアル回線からの受信データ(S-JIS / UTF-8 / EUC-J)をテキスト表示します。¥n"));
57 :     m_VthScp.PrintF(TEXT("・ここにファイルをドロップすると、ファイルの内容を UTF-8 テキストとして送信します。¥n"));
58 :     m_VthScp.PrintF(TEXT("・ここにフォルダをドロップすると、ドロップしたフォルダのパス名を表示します。¥n"));
59 :     m_VthScp.PrintF(TEXT("¥n"));
60 :
61 :     m_VthScp.SetTitleText(TEXT(" Closed "));
62 :
63 :     return TRUE;  // フォーカスをコントロールに設定した場合を除き、TRUE を返します。
64 : }
65 :
66 : // ダイアログに最小化ボタンを追加する場合、アイコンを描画するための
67 : //  下のコードが必要です。ドキュメント/ビュー モデルを使う MFC アプリケーションの場合、
68 : //  これは、Framework によって自動的に設定されます。
69 :
70 : void CS_MFC_01Dlg::OnPaint()
71 : {
72 :     if (IsIconic())
73 :     {
74 :         CPaintDC dc(this); // 描画のデバイス コンテキスト
75 :
76 :         SendMessage(WM_ICONERASEBKGND, reinterpret_cast<WPARAM>(dc.GetSafeHdc()), 0);
77 :
78 :         // クライアントの四角形領域内の中央
79 :         int cxIcon = GetSystemMetrics(SM_CXICON);
80 :         int cyIcon = GetSystemMetrics(SM_CYICON);
81 :         CRect rect;
82 :         GetClientRect(&rect);
83 :         int x = (rect.Width() - cxIcon + 1) / 2;
84 :         int y = (rect.Height() - cyIcon + 1) / 2;
85 :
86 :         // アイコンの描画
87 :         dc.DrawIcon(x, y, m_hIcon);
88 :     }
89 :     else
90 :     {
91 :         CDialogEx::OnPaint();
92 :     }
93 : }
94 :
95 : // ユーザーが最小化したウィンドウをドラッグしているときに表示するカーソルを取得するために、
96 : //  システムがこの関数を呼び出します。
97 : HCURSOR CS_MFC_01Dlg::OnQueryDragIcon()
98 : {
99 :     return static_cast<HCURSOR>(m_hIcon);
100 : }
101 :
102 : //  ●  回線オープンボタン
103 : void CS_MFC_01Dlg::OnBnClickedCmdOpen()
104 : {
105 :     // TODO: ここにコントロール通知ハンドラー コードを追加します。
106 :     if (!m_VthScp.IsOpened()) {
107 :         m_VthScp.Open();
108 :     }
109 : }
110 :
111 : //  ●  回線クローズボタン
112 : void CS_MFC_01Dlg::OnBnClickedCmdClose()
113 : {
114 :     // TODO: ここにコントロール通知ハンドラー コードを追加します。
115 :     if (m_VthScp.IsOpened()) {
116 :         m_VthScp.Close();
117 :     }
118 : }
119 :
120 : //  ●  ポート設定ボタン
121 : void CS_MFC_01Dlg::OnBnClickedCmdPort()
122 : {
123 :     // TODO: ここにコントロール通知ハンドラー コードを追加します。
124 :     m_VthScp.DlgParamEasy(m_hWnd);
125 : }
126 :
127 : //  ●  アプリ終了ボタン
128 : void CS_MFC_01Dlg::OnCancel()
129 : {
130 :     // TODO: ここに特定なコードを追加するか、もしくは基本クラスを呼び出してください。
131 :     CDialogEx::OnCancel();
132 : }
133 :
134 : //  ●  オープンボタン許可／禁止
135 : void CS_MFC_01Dlg::EnableOpenButton(bool fEnable)
136 : {
137 :     AjcEnableDlgItem(m_hWnd, IDC_CMD_OPEN , fEnable);
138 : }
139 :
140 : //  ●  クローズボタン許可／禁止
141 : void CS_MFC_01Dlg::EnableCloseButton(bool fEnable)
142 : {
```

```
143 :     AjcEnableDlgItem(m_hWnd, IDC_CMD_CLOSE, fEnable);
144 : }
```

「CAjxVth, CAjxScp」クラスの継承

●VthScp.h（新規作成）

```
 1 : #pragma once
 2 : #include <functional>
 3 : #include <AjxCpp.h>
 4 :
 5 : class CVthScp :
 6 :     public CAjxVth,
 7 :     public CAjxScp
 8 : {
 9 : public:
10 :     CVthScp(void);
11 :     ~CVthScp(void);
12 :
13 : private:
14 :     std::function<void(bool)> m_EnableOpenButton;          //  オープンボタン許可／禁止
15 :     std::function<void(bool)> m_EnableCloseButton;         //  クローズボタン許可／禁止
16 :
17 : public:
18 :     VO  SetUp(std::function<void(bool)> EnableOpenButton,
19 :               std::function<void(bool)> EnableCloseButton);
20 :
21 :     VO  OnNtcDropDir  (UI nDirs ) override;                //  ディレクトリドロップ通知
22 :     VO  OnNtcDropFile (UI nFiles) override;                //  ファイルドロップ通知
23 :
24 :     VO  OnNtcPortState(C_UTP pPortName, UI Param) override; //  ポート状態通知
25 :     VO  OnNtcRxTextChunk(C_UTP pText) override;            //  テキストチャンク受信通知
26 :
27 : };
28 :
```

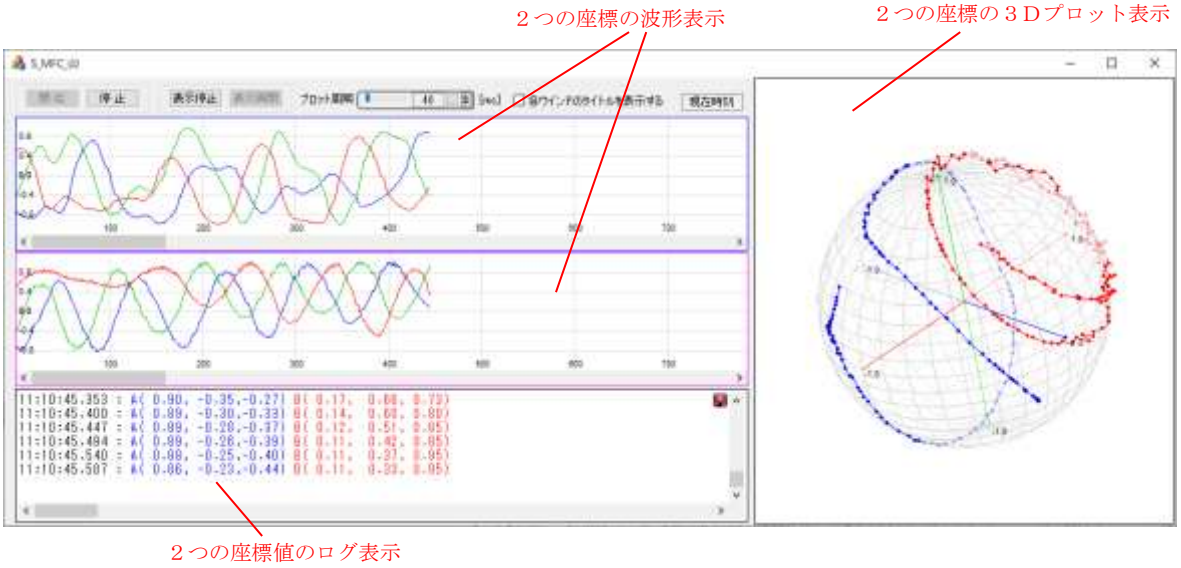
●VthScp.cpp（新規作成）

```
 1 : #include "StdAfx.h"
 2 : #include "VthScp.h"
 3 :
 4 :
 5 : CVthScp::CVthScp(void)
 6 : {
 7 : }
 8 :
 9 :
10 : CVthScp::~CVthScp(void)
11 : {
12 : }
13 :
14 :     VO  CVthScp::SetUp(std::function<void(bool)> EnableOpenButton,
15 :               std::function<void(bool)> EnableCloseButton)
16 : {
17 :     m_EnableOpenButton  = EnableOpenButton;
18 :     m_EnableCloseButton = EnableCloseButton;
19 : }
20 :
21 :
22 : // ●ディレクトリドロップ
23 : VO  CVthScp::OnNtcDropDir (UI nFiles)
24 : {
25 :     // ドロップされたディレクトリ表示
26 :     UT path[MAX_PATH] = {0};
27 :     while (GetDroppedDir(path)) {
28 :         PrintF(TEXT("Directory dropped : %s¥n"), path);
29 :     }
30 : }
31 : // ●ファイルドロップ
32 : VO  CVthScp::OnNtcDropFile(UI nFiles)
33 : {
34 :     CAjxFile f;
35 :     UT  path[MAX_PATH];
36 :     UT  buf[1024];
37 :     while (GetDroppedFile(path)) {
38 :         PrintF(TEXT("¥x1B[34m【 %s 】¥x1B[0m¥n"), path);
39 :         if (f.FOpen(path)) {
40 :             PrintF(TEXT("Send File : %s¥n"), path);
41 :             while (f.FGetS(buf, AJCTSIZE(buf))) {
42 :                 SendText(buf, (UI)MAjcStrLen(buf));
43 :             }
44 :             f.FClose();
45 :         }
46 :     }
47 : }
```

```
48 : //  ●ポート状態通知
49 : VO  CVthScp::OnNtcPortState(C_UTP pPortName, UI Param)
50 : {
51 :     switch (Param) {
52 :         case AJCSCP_CLOSED:     SetTitleText(TEXT(" Closed ")); m_EnableOpenButton(true ); m_EnableCloseButton(false); break;
53 :         case AJCSCP_OPENED:     SetTitleText(TEXT(" Opened ")); m_EnableOpenButton(false); m_EnableCloseButton(true ); break;
54 :         case AJCSCP_OPENFAIL:   SetTitleText(TEXT(" Closed "));
55 :           PrintF(TEXT("¥x1b[31m ポートのオープンを失敗しました¥x1b[0m¥n"));  break;
56 :     }
57 : }
58 : //  ●  テキストチャンク受信通知
59 : VO  CVthScp::OnNtcRxTextChunk(C_UTP pText)
60 : {
61 :     PutText(pText);
62 : }
63 :
```

64.2.　ＭＦＣサンプルプログラム２（S_MFC_02）

ＭＦＣで、C++クラスライブラリ(AjxCpp32.dll/AjxCpp64.dll)を使用したサンプルプログラムを示します。
このサンプルプログラムは、内部で２つの３Ｄテストデータ(x, y, z)を生成し、その波形，３Ｄプロットとログを表示します。
２つの座標値は、中心(0, 0, 0)、半径=1.0の球面上の座標となります。
各座標値は、それぞれ、半径に±１％と±５％のノイズを含めた値となります。



「開始」ボタンを押すと、テストデータの生成を開始し、「停止」ボタンを押すとテストデータの生成を停止します。
「表示停止」ボタンを押すと、各ウインドの表示を停止します。(テストデータの生成は停止せずに継続します)
「表示再開」ボタンを押すと、各ウインドの表示を開会します。この時、表示停止中に蓄積されたデータは一気に表示されます。

２つの波形表示ウインドの片方のスクロールバーを操作すると、他方のウインドも同期してスクロールします。

「現在時刻」の部分にカーソルを置くと現在時刻をチップ表示します。(状況依存ツールチップのサンプル)

このサンプルプログラムでは、本ライブラリのC++ラッパークラスを使用しています。
ラッパークラスについては「AjrCppClass.pdf」を参照してください。

●S_MFC_02.cpp

```
 1 :
 2 : // S_MFC_02.cpp : アプリケーションのクラス動作を定義します。
 3 : //
 4 :
 5 : #include "stdafx.h"
 6 : #include "S_MFC_02.h"
 7 : #include "S_MFC_02Dlg.h"
 8 :
 9 : #ifdef _DEBUG
10 : #define new DEBUG_NEW
11 : #endif
12 :
13 :
14 : // CS_MFC_02App
15 :
16 : BEGIN_MESSAGE_MAP(CS_MFC_02App, CWinApp)
17 :     ON_COMMAND(ID_HELP, &CWinApp::OnHelp)
18 : END_MESSAGE_MAP()
19 :
20 : // CS_MFC_02App コンストラクション
21 :
22 : CS_MFC_02App::CS_MFC_02App()
23 : {
24 :     // 再起動マネージャーをサポートします
25 :     m_dwRestartManagerSupportFlags = AFX_RESTART_MANAGER_SUPPORT_RESTART;
26 :
27 :     // TODO: この位置に構築用コードを追加してください。
28 :     // ここに InitInstance 中の重要な初期化処理をすべて記述してください。
29 :     AjcSetCmdWithHdl(TRUE);             //  ●ＭＦＣで実行する場合に必要な初期化
30 : }
```

```
31 :
32 :
33 : // 唯一の CS_MFC_02App オブジェクトです。
34 :
35 : CS_MFC_02App theApp;
36 :
37 :
38 : // CS_MFC_02App 初期化
39 :
40 : BOOL CS_MFC_02App::InitInstance()
41 : {
42 :     // アプリケーション マニフェストが visual スタイルを有効にするために、
43 :     // ComCtl32.dll Version 6 以降の使用を指定する場合は、
44 :     // Windows XP に InitCommonControlsEx() が必要です。さもなければ、ウィンドウ作成はすべて失敗します。
45 :     INITCOMMONCONTROLSEX InitCtrls;
46 :     InitCtrls.dwSize = sizeof(InitCtrls);
47 :     // アプリケーションで使用するすべてのコモン コントロール クラスを含めるには、
48 :     // これを設定します。
49 :     InitCtrls.dwICC = ICC_WIN95_CLASSES;
50 :     InitCommonControlsEx(&InitCtrls);
51 :
52 :     CWinApp::InitInstance();
53 :
54 :
55 :     AfxEnableControlContainer();
56 :
57 :     // ダイアログにシェル ツリー ビューまたはシェル リスト ビュー コントロールが
58 :     // 含まれている場合にシェル マネージャーを作成します。
59 :     CShellManager *pShellManager = new CShellManager;
60 :
61 :     // 標準初期化
62 :     // これらの機能を使わずに最終的な実行可能ファイルの
63 :     // サイズを縮小したい場合は、以下から不要な初期化
64 :     // ルーチンを削除してください。
65 :     // 設定が格納されているレジストリ キーを変更します。
66 :     // TODO: 会社名または組織名などの適切な文字列に
67 :     // この文字列を変更してください。
68 :     SetRegistryKey(_T("アプリケーション ウィザードで生成されたローカル アプリケーション"));
69 :
70 :     CS_MFC_02Dlg dlg;
71 :     m_pMainWnd = &dlg;
72 :     INT_PTR nResponse = dlg.DoModal();
73 :     if (nResponse == IDOK)
74 :     {
75 :         // TODO: ダイアログが <OK> で消された時のコードを
76 :         //  記述してください。
77 :     }
78 :     else if (nResponse == IDCANCEL)
79 :     {
80 :         // TODO: ダイアログが <キャンセル> で消された時のコードを
81 :         //  記述してください。
82 :     }
83 :
84 :     // 上で作成されたシェル マネージャーを削除します。
85 :     if (pShellManager != NULL)
86 :     {
87 :         delete pShellManager;
88 :     }
89 :
90 :     // ダイアログは閉じられました。アプリケーションのメッセージ ポンプを開始しないで
91 :     //  アプリケーションを終了するために FALSE を返してください。
92 :     return FALSE;
93 : }
94 :
```

●S_MFC_02Dlg.h : ヘッダー ファイル

```
 1 :
 2 : // S_MFC_02Dlg.h : ヘッダー ファイル
 3 : //
 4 :
 5 : #pragma once
 6 :
 7 : #include    <AjxCpp.h>
 8 : #include    "AjxTchEx.h"
 9 : #include    "AjxVthEx.h"
10 : #include    "AjxInpEx.h"
11 : #include    "AjxG3dEx.h"
12 : using namespace AjxControl;
13 :
14 : // CS_MFC_02Dlg ダイアログ
15 : class CS_MFC_02Dlg : public CDialogEx
16 : {
17 : // コンストラクション
18 : public:
19 :     CS_MFC_02Dlg(CWnd* pParent = NULL); // 標準コンストラクター
20 :
```

```
21 : // ダイアログ データ
22 :     enum { IDD = IDD_S_MFC_02_DIALOG };
23 :
24 :     protected:
25 :     virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV サポート
26 :
27 :
28 : // 実装
29 : protected:
30 :     HICON m_hIcon;
31 :
32 :     // 生成された、メッセージ割り当て関数
33 :     virtual BOOL OnInitDialog();
34 :     afx_msg void OnPaint();
35 :     afx_msg HCURSOR OnQueryDragIcon();
36 :     DECLARE_MESSAGE_MAP()
37 :     virtual void OnCancel();
38 : public:
39 :     afx_msg void OnTimer(UINT_PTR nIDEvent);
40 :     afx_msg void OnBnClickedCmdStart();
41 :     afx_msg void OnBnClickedCmdStop();
42 :     afx_msg void OnBnClickedCmdPause();
43 :     afx_msg void OnBnClickedCmdRestart();
44 :     afx_msg void OnSize(UINT nType, int cx, int cy);
45 :     afx_msg void OnSizing(UINT fwSide, LPRECT pRect);
46 :     afx_msg void OnBnClickedChkShowttl();
47 : private:
48 :     bool m_fReady;                  //  プログラムが実行状態であることを示すフラグ
49 :     SIZE m_szDlg;                   //  ダイアログぼっくの初期サイズ
50 :     int  m_TickStart;               //  テストデータ生成状態を示すフラグ
51 :     CAjxDlg     m_dlg;              //  ダイアログ項目アクセスオブジェクト
52 :     CAjxInpEx   m_inp;              //  数値入力オブジェクト
53 :     CAjxG3dEx   m_g3d;              //  ３Ｄグラフィックオブジェクト
54 :     CAjxVthEx   m_vth;              //  ＶＴＨオブジェクト（ログ表示ウインド）
55 :     CAjxTchEx   m_tch1, m_tch2;     //  タイムチャートグラフオブジェクト
56 :     CAjxSpd     m_spd1, m_spd2;     //  テストデータ生成オブジェクト
57 :     CAjxTip     m_tip;              //  状況依存チップテキスト
58 :     void    ArrengeControls(void);  //  各コントロールの配置
59 : };
```

●S_MFC_02Dlg.cpp

```
 1 :
 2 : // S_MFC_02Dlg.cpp : 実装ファイル
 3 : //
 4 :
 5 : #include "stdafx.h"
 6 : #include "S_MFC_02.h"
 7 : #include "S_MFC_02Dlg.h"
 8 : #include "afxdialogex.h"
 9 :
10 : #ifdef _DEBUG
11 : #define new DEBUG_NEW
12 : #endif
13 :
14 : //  プログラムワーク
15 : BOOL    fBusy = FALSE;
16 :
17 :
18 :
19 : // CS_MFC_02Dlg ダイアログ
20 :
21 :
22 :
23 :
24 : CS_MFC_02Dlg::CS_MFC_02Dlg(CWnd* pParent /*=NULL*/)
25 :     : CDialogEx(CS_MFC_02Dlg::IDD, pParent)
26 :     , m_fReady(false)
27 :     , m_TickStart(0)
28 : {
29 :     m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
30 : }
31 :
32 : void CS_MFC_02Dlg::DoDataExchange(CDataExchange* pDX)
33 : {
34 :     CDialogEx::DoDataExchange(pDX);
35 : }
36 :
37 : BEGIN_MESSAGE_MAP(CS_MFC_02Dlg, CDialogEx)
38 :     ON_WM_PAINT()
39 :     ON_WM_QUERYDRAGICON()
40 :     ON_WM_TIMER()
41 :     ON_BN_CLICKED(IDC_CMD_START, &CS_MFC_02Dlg::OnBnClickedCmdStart)
42 :     ON_BN_CLICKED(IDC_CMD_STOP, &CS_MFC_02Dlg::OnBnClickedCmdStop)
43 :     ON_BN_CLICKED(IDC_CMD_PAUSE, &CS_MFC_02Dlg::OnBnClickedCmdPause)
44 :     ON_BN_CLICKED(IDC_CMD_RESTART, &CS_MFC_02Dlg::OnBnClickedCmdRestart)
45 :     ON_WM_SIZE()
46 :     ON_WM_SIZING()
```

```
47 :     ON_BN_CLICKED(IDC_CHK_SHOWTTL, &CS_MFC_02Dlg::OnBnClickedChkShowttl)
48 : END_MESSAGE_MAP()
49 :
50 :
51 : //  状況依存ツールチップ(IDC_TXT_TIME)
52 : class CAjxTip_TEXT_TIME : public CAjxTip
53 : {
54 :     VO      OnNeedText (HWND hCtrl, UTP pBuf, UI lBuf) override
55 :     {
56 :         CTime   t = CTime::GetCurrentTime();
57 :         CString s = t.Format("%Y/%m/%d %H:%M:%S");
58 :         _tcscpy_s(pBuf, lBuf, (LPCTSTR)s);
59 :     }
60 : };
61 :
62 : CAjxTip_TEXT_TIME   tipTextTime;
63 :
64 :
65 : // CS_MFC_02Dlg メッセージ ハンドラー
66 :
67 : BOOL CS_MFC_02Dlg::OnInitDialog()
68 : {
69 :     CDialogEx::OnInitDialog();
70 :
71 :     // このダイアログのアイコンを設定します。アプリケーションのメイン ウィンドウがダイアログでない場合、
72 :     //  Framework は、この設定を自動的に行います。
73 :     SetIcon(m_hIcon, TRUE);         // 大きいアイコンの設定
74 :     SetIcon(m_hIcon, FALSE);        // 小さいアイコンの設定
75 :
76 :     // TODO: 初期化をここに追加します。
77 :     //  ダイアログのサイズ設定
78 :     RECT    rc;
79 :     GetWindowRect(&rc);
80 :     m_szDlg.cx = rc.right - rc.left;
81 :     m_szDlg.cy = rc.bottom - rc.top;
82 :     //  各コントロールへハンドルをアタッチ
83 :     m_dlg .CAjxDlg  ::Attach(m_hWnd);
84 :     m_tch1.CAjxTchEx::Attach(GetDlgItem(IDC_TCH1)->m_hWnd);
85 :     m_tch2.CAjxTchEx::Attach(GetDlgItem(IDC_TCH2)->m_hWnd);
86 :     m_inp .CAjxInpEx::Attach(GetDlgItem(IDC_INP )->m_hWnd);
87 :     m_vth .CAjxVthEx::Attach(GetDlgItem(IDC_VTH )->m_hWnd);
88 :     m_g3d .CAjxG3dEx::Attach(GetDlgItem(IDC_G3D )->m_hWnd);
89 :     //  タイムチャートコントロール初期化
90 :     m_tch1.SetOther(&m_tch2);               //  他方のタイムチャートオブジェクト設定
91 :     m_tch2.SetOther(&m_tch1);
92 :     m_tch1.ShowBorder(TRUE, RGB(  0,   0, 255));
93 :     m_tch2.ShowBorder(TRUE, RGB(255,   0,   0));
94 :     //  ３Ｄグラフィックコントロール初期化
95 :     m_g3d.SetAngle3D();                     //  視点を３Ｄイメージに設定
96 :     m_g3d.SetPlotNumber(0, 150);            //  プロット数設定
97 :     m_g3d.SetPlotNumber(1, 150);
98 :     //  ログウインド初期化
99 :     //  ツールチップテキスト設定
100 :    m_tch1.SetTipText(TEXT("¥x1B[34m¥x1B[Ｔサンプルデータ Ａ：¥x1B[0m¥x1B[t¥n")
101 :                      TEXT("球の表面を示すプロットデータ (x, y, z) を生成し、その波形を表示します。¥n")
102 :                      TEXT("プロットデータは、半径に±１％のノイズを加えます。¥n")
103 :                      TEXT("CTRL+中ボタンでゲージ間計測モードになります。"));
104 :    m_tch1.SetChkBoxTipText(0, TEXT("x の波形"));
105 :    m_tch1.SetChkBoxTipText(1, TEXT("y の波形"));
106 :    m_tch1.SetChkBoxTipText(2, TEXT("z の波形"));
107 :    m_tch2.SetTipText(TEXT("¥x1B[31m¥x1B[Ｔサンプルデータ Ｂ：¥x1B[0m¥x1B[t¥n")
108 :                      TEXT("球の表面を示すプロットデータ (x, y, z) を生成し、その波形を表示します。¥n")
109 :                      TEXT("プロットデータは、半径に±５％のノイズを加えます。¥n")
110 :                      TEXT("CTRL+中ボタンでゲージ間計測モードになります。"));
111 :    m_tch2.SetChkBoxTipText(0, TEXT("x の波形"));
112 :    m_tch2.SetChkBoxTipText(1, TEXT("y の波形"));
113 :    m_tch2.SetChkBoxTipText(2, TEXT("z の波形"));
114 :    m_vth .SetTipText(TEXT("生成した２つの座標値 (x, y, z) をログ表示します。¥n")
115 :                      TEXT("¥x1B[34m 青は上段の波形のデータ値です。¥x1B[0m¥n")
116 :                      TEXT("¥x1B[31m 赤は下段の波形のデータ値です。¥x1B[0m"));
117 :
118 :    m_g3d .SetTipText(TEXT("２つの球面プロットデータを３Ｄイメージで表示します。¥n")
119 :                      TEXT("¥x1B[34m 青は上段の波形に対応します。¥x1B[0m¥n")
120 :                      TEXT("¥x1B[31m 赤は下段の波形に対応します。¥x1B[0m"));
121 :    m_g3d .SetChkBoxTipText(0, TEXT("ノイズ１％データのプロット"));
122 :    m_g3d .SetChkBoxTipText(1, TEXT("ノイズ５％データのプロット"));
123 :
124 :    SAjxTip::Add(GetDlgItem(IDC_CMD_START  )->m_hWnd, TEXT("テストデータの生成とプロット表示を開始します。"));
125 :    SAjxTip::Add(GetDlgItem(IDC_CMD_STOP   )->m_hWnd, TEXT("テストデータの生成とプロット表示を停止します。"));
126 :    SAjxTip::Add(GetDlgItem(IDC_CMD_PAUSE  )->m_hWnd, TEXT("各グラフやログの表示を停止します。¥n")
127 :                                                      TEXT("表示は停止しますが、データは更新を続けます。"));
128 :    SAjxTip::Add(GetDlgItem(IDC_CMD_RESTART)->m_hWnd, TEXT("各グラフやログの表示を再開します。¥n")
129 :                                                      TEXT("蓄積されているデータの末尾から表示します。"));
130 :    //  状況依存ツールチップテキスト設定 (現在時刻)
131 :    m_dlg.SetDlgItemStr(IDC_TXT_TIME, TEXT("現在時刻"));
132 :    tipTextTime.Attach(GetDlgItem(IDC_TXT_TIME)->m_hWnd);
133 :
134 :    //  テストデータ生成条件を設定
135 :    AJCSPD_PARAM prm1, prm2;
136 :    m_spd1.GetParam(&prm1);                         m_spd2.GetParam(&prm2);
```

```
137 :     prm1.noise = 1; prm1.xrot = prm1.yrot = 10;     prm2.noise = 5;
138 :     m_spd1.SetParam(&prm1);                         m_spd2.SetParam(&prm2);
139 :
140 :     //  各コントロール設定値読み出し
141 :     m_dlg.LoadAllControlSettings(TEXT("Settings_*"));
142 :
143 :     m_fReady = TRUE;
144 :     ArrengeControls();
145 :
146 :     return TRUE;  // フォーカスをコントロールに設定した場合を除き、TRUE を返します。
147 : }
148 :
149 : // ダイアログに最小化ボタンを追加する場合、アイコンを描画するための
150 : //  下のコードが必要です。ドキュメント/ビュー モデルを使う MFC アプリケーションの場合、
151 : //  これは、Framework によって自動的に設定されます。
152 :
153 : void CS_MFC_02Dlg::OnPaint()
154 : {
155 :     if (IsIconic())
156 :     {
157 :         CPaintDC dc(this); // 描画のデバイス コンテキスト
158 :
159 :         SendMessage(WM_ICONERASEBKGND, reinterpret_cast<WPARAM>(dc.GetSafeHdc()), 0);
160 :
161 :         // クライアントの四角形領域内の中央
162 :         int cxIcon = GetSystemMetrics(SM_CXICON);
163 :         int cyIcon = GetSystemMetrics(SM_CYICON);
164 :         CRect rect;
165 :         GetClientRect(&rect);
166 :         int x = (rect.Width() - cxIcon + 1) / 2;
167 :         int y = (rect.Height() - cyIcon + 1) / 2;
168 :
169 :         // アイコンの描画
170 :         dc.DrawIcon(x, y, m_hIcon);
171 :     }
172 :     else
173 :     {
174 :         CDialogEx::OnPaint();
175 :     }
176 : }
177 :
178 : // ユーザーが最小化したウィンドウをドラッグしているときに表示するカーソルを取得するために、
179 : //  システムがこの関数を呼び出します。
180 : HCURSOR CS_MFC_02Dlg::OnQueryDragIcon()
181 : {
182 :     return static_cast<HCURSOR>(m_hIcon);
183 : }
184 :
185 :
186 :
187 : //----- キャンセル ------------------------------------------------------------------------------------------//
188 : void CS_MFC_02Dlg::OnCancel()
189 : {
190 :     // TODO: ここに特定なコードを追加するか、もしくは基本クラスを呼び出してください。
191 :
192 :     //  各コントロール設定値書き込み
193 :     m_dlg.SaveAllControlSettings();
194 :
195 :     CDialogEx::OnCancel();
196 : }
197 :
198 :
199 : //----- タイマ ----------------------------------------------------------------------------------------------//
200 : void CS_MFC_02Dlg::OnTimer(UINT_PTR nIDEvent)
201 : {
202 :     // TODO: ここにメッセージ ハンドラー コードを追加するか、既定の処理を呼び出します。
203 :     double  x1, y1, z1, x2, y2, z2;
204 :     m_vth.TimeStamp();
205 :
206 :     m_spd1.Calc     (&x1, &y1, &z1);
207 :     m_tch1.PutData  ( x1,  y1,  z1);
208 :     m_g3d.PutPlotData (0,  x1,  y1,  z1);
209 :     m_vth.PrintF(TEXT(" : ¥x1B[34mA(%5.2f, %5.2f,%5.2f)"), x1, y1, z1);
210 :
211 :     m_spd2.Calc     (&x2, &y2, &z2);
212 :     m_tch2.PutData  ( x2,  y2,  z2);
213 :     m_g3d.PutPlotData (1,  x2,  y2,  z2);
214 :     m_vth.PrintF(TEXT("¥x1B[31m B(%5.2f, %5.2f,%5.2f)¥x1B[0m¥n"), x2, y2, z2);
215 :
216 : //  CDialogEx::OnTimer(nIDEvent);
217 : }
218 : //----- ダイアログサイズ変更中 ------------------------------------------------------------------------------//
219 : void CS_MFC_02Dlg::OnSizing(UINT fwSide, LPRECT pRect)
220 : {
221 :     CDialogEx::OnSizing(fwSide, pRect);
222 :
223 :     // TODO: ここにメッセージ ハンドラー コードを追加します。
224 :     int cx = pRect->right - pRect->left;
225 :     int cy = pRect->bottom - pRect->top;
226 :
```

```
227 :     if (cx < m_szDlg.cx) pRect->right  = pRect->left + m_szDlg.cx;
228 :     if (cy < m_szDlg.cy) pRect->bottom = pRect->top  + m_szDlg.cy;
229 : }
230 : //----- ダイアログサイズ変更 ------------------------------------------------------------------------------------//
231 : void CS_MFC_02Dlg::OnSize(UINT nType, int cx, int cy)
232 : {
233 :     CDialogEx::OnSize(nType, cx, cy);
234 :
235 :     // TODO: ここにメッセージ ハンドラー コードを追加します。
236 :     if (m_fReady) {
237 :         ArrengeControls();
238 :     }
239 : }
240 :
241 : //----- 開始ボタン ----------------------------------------------------------------------------------------------//
242 : void CS_MFC_02Dlg::OnBnClickedCmdStart()
243 : {
244 :     // TODO: ここにコントロール通知ハンドラー コードを追加します。
245 :     SetTimer(1, m_inp.GetValueInt(), NULL);
246 :     GetDlgItem(IDC_CMD_START  )->EnableWindow(FALSE);
247 :     GetDlgItem(IDC_CMD_STOP   )->EnableWindow(TRUE );
248 :     GetDlgItem(IDC_CMD_PAUSE  )->EnableWindow(TRUE );
249 :     GetDlgItem(IDC_CMD_RESTART)->EnableWindow(FALSE);
250 :     fBusy = TRUE;
251 :     m_TickStart = GetTickCount();
252 : }
253 :
254 :
255 : //----- 停止ボタン ----------------------------------------------------------------------------------------------//
256 : void CS_MFC_02Dlg::OnBnClickedCmdStop()
257 : {
258 :     // TODO: ここにコントロール通知ハンドラー コードを追加します。
259 :     KillTimer(1);
260 :     m_tch1.Pause(FALSE);
261 :     m_tch2.Pause(FALSE);
262 :     m_vth .Pause(FALSE);
263 :     m_g3d .Pause(FALSE);
264 :     GetDlgItem(IDC_CMD_START  )->EnableWindow(TRUE);
265 :     GetDlgItem(IDC_CMD_STOP   )->EnableWindow(FALSE);
266 :     GetDlgItem(IDC_CMD_PAUSE  )->EnableWindow(FALSE);
267 :     GetDlgItem(IDC_CMD_RESTART)->EnableWindow(FALSE);
268 :     fBusy = FALSE;
269 : }
270 :
271 :
272 : //----- 一時停止ボタン ------------------------------------------------------------------------------------------//
273 : void CS_MFC_02Dlg::OnBnClickedCmdPause()
274 : {
275 :     // TODO: ここにコントロール通知ハンドラー コードを追加します。
276 :     m_tch1.Pause(TRUE);
277 :     m_tch2.Pause(TRUE);
278 :     m_vth .Pause(TRUE);
279 :     m_g3d .Pause(TRUE);
280 :     GetDlgItem(IDC_CMD_START  )->EnableWindow(FALSE);
281 :     GetDlgItem(IDC_CMD_STOP   )->EnableWindow(TRUE );
282 :     GetDlgItem(IDC_CMD_PAUSE  )->EnableWindow(FALSE);
283 :     GetDlgItem(IDC_CMD_RESTART)->EnableWindow(TRUE );
284 : }
285 :
286 :
287 : //----- 再開ボタン ----------------------------------------------------------------------------------------------//
288 : void CS_MFC_02Dlg::OnBnClickedCmdRestart()
289 : {
290 :     // TODO: ここにコントロール通知ハンドラー コードを追加します。
291 :     m_tch1.Pause(FALSE);
292 :     m_tch2.Pause(FALSE);
293 :     m_vth .Pause(FALSE);
294 :     m_g3d .Pause(FALSE);
295 :     GetDlgItem(IDC_CMD_START  )->EnableWindow(FALSE);
296 :     GetDlgItem(IDC_CMD_STOP   )->EnableWindow(TRUE );
297 :     GetDlgItem(IDC_CMD_PAUSE  )->EnableWindow(TRUE );
298 :     GetDlgItem(IDC_CMD_RESTART)->EnableWindow(FALSE);
299 : }
300 : //----- チェックボックス（各ウインドのタイトルを表示する）------------------------------------------------------//
301 : void CS_MFC_02Dlg::OnBnClickedChkShowttl()
302 : {
303 :     // TODO: ここにコントロール通知ハンドラー コードを追加します。
304 : //  if (IsDlgButtonChecked(IDC_CHK_SHOWTTL)) {
305 :     if (m_dlg.GetDlgItemChk(IDC_CHK_SHOWTTL)) {
306 :         m_tch1.SetTitleText(TEXT("A: ノイズ１%"));
307 :         m_tch2.SetTitleText(TEXT("B: ノイズ５%"));
308 :         m_vth .SetTitleText(TEXT("ログ表示"));
309 :         m_g3d .SetTitleText(TEXT("２つのデータの３Ｄプロット"));
310 :     }
311 :     else {
312 :         m_tch1.SetTitleText("");
313 :         m_tch2.SetTitleText("");
314 :         m_vth .SetTitleText("");
315 :         m_g3d .SetTitleText("");
316 :     }
```

```
317 : }
318 : //----- 各コントロールの配置 ------------------------------------------------------------------------------//
319 : void CS_MFC_02Dlg::ArrengeControls(void)
320 : {
321 :     POINT       ptB;                //  コントロール表示域のベース位置
322 :     int         dhL;                //  左側表示域の高さ
323 :     int         chL;                //  左側各コントロールの高さ
324 :     SIZE        szCur;              //  ダイアログのサイズ
325 :     RECT        rc;
326 :     int         x, y, cx, cy;
327 :
328 :     GetClientRect(&rc);
329 :     szCur.cx = rc.right - rc.left;
330 :     szCur.cy = rc.bottom - rc.top;
331 :
332 :     m_tch1.GetWindowRect(&rc);
333 :     ::MapWindowPoints(NULL, m_hWnd, (LPPOINT)&rc, 2);
334 :     ptB.x = rc.left;  ptB.y = rc.top;
335 :
336 :     cx  = rc.right - rc.left;
337 :     dhL = szCur.cy - ptB.y;
338 :     chL = dhL / 3;
339 :     cy  = chL - 2;
340 :
341 :     x  = ptB.x; y = ptB.y;
342 :     m_tch1.SetWindowPos(NULL, x, y, cx, cy, SWP_NOZORDER); y += chL;
343 :     m_tch2.SetWindowPos(NULL, x, y, cx, cy, SWP_NOZORDER); y += chL;
344 :     m_vth .SetWindowPos(NULL, x, y, cx, cy, SWP_NOZORDER);
345 :
346 :     m_g3d.GetWindowRect(&rc);
347 :     ::MapWindowPoints(NULL, m_hWnd, (LPPOINT)&rc, 2);
348 :     ptB.x = rc.left;  ptB.y = rc.top;
349 :     cx = szCur.cx - ptB.x - 2;  cy = szCur.cy - ptB.y - 2;
350 :     m_g3d.SetWindowPos(NULL, 0, 0, cx, cy, SWP_NOMOVE);
351 : }
352 :
353 :
```

「CAjxTch」クラスの継承（タイムチャート・クラス）

●AjxTxhEx.h（新規作成）

```
1 : #pragma once
2 : #include <AjxCpp.h>
3 : using namespace AjxControl;
4 :
5 : class CAjxTchEx :
6 :     public CAjxTch,
7 :     public CWnd
8 : {
9 : private:
10 :     CAjxTchEx *m_tchOther;  //  他方のタイムチャート
11 : public:
12 :     CAjxTchEx(void);
13 :     ~CAjxTchEx(void);
14 :     VO  SetOther        (CAjxTchEx* other);
15 :     VO  Attach          (HWND hwnd)                 override;
16 :     VO  OnNtcRange      (PCAJCTC_NTC_RANGE pRange)  override;
17 :     VO  OnNtcScrPos     (UI                ScrPos)  override;
18 :     VO  OnNtcClear      ()                          override;
19 :     VO  OnNtcDblClk     ()                          override;
20 :     VO  OnNtcDropFile   (UI              nFiles)    override;
21 :     VO  OnNtcDropDir    (UI              nDirs )    override;
22 :     VO  OnNtcRClick     (PCAJCTCRCLK     pRClk )    override;
23 :
24 : };
25 :
```

●AjxTxhEx.cpp（新規作成）

```
1 : #include "StdAfx.h"
2 : #include "AjxTchEx.h"
3 :
4 : //----- コンストラクタ ------------------------------------------------------------------------------------//
5 : CAjxTchEx::CAjxTchEx(void)
6 : {
7 : }
8 : //----- デストラクタ --------------------------------------------------------------------------------------//
9 : CAjxTchEx::~CAjxTchEx(void)
10 : {
11 :     //  CWnd からウインドを切り離す（CWnd でウインドを破棄しないようにする）
12 :     if (CWnd::m_hWnd != NULL) {
13 :         CWnd::Detach();
14 :     }
15 : }
16 : //----- ウインドハンドルをアタッチ ------------------------------------------------------------------------//
```

```
17 : VO  CAjxTchEx::Attach (HWND hwnd)
18 : {
19 :     CWnd::   Attach(hwnd);
20 :     CAjxTch::Attach(hwnd);
21 : }
22 : //---- 他方のタイムチャートコントロール設定 ----------------------------------------------------------------//
23 : VO  CAjxTchEx::SetOther        (CAjxTchEx* other)
24 : {
25 :     m_tchOther = other;
26 : }
27 : //---- グラフレンジ通知 ------------------------------------------------------------------------------------//
28 : VO  CAjxTchEx::OnNtcRange       (PCAJCTC_NTC_RANGE pRange)
29 : {
30 : }
31 : //---- スクロール位置通知 ----------------------------------------------------------------------------------//
32 : VO  CAjxTchEx::OnNtcScrPos      (UI                ScrPos)
33 : {
34 :     //  他方のタイムチャートのスクロール位置設定（スクロールの連動）
35 :     m_tchOther->CAjxTch::SetScrollPos(ScrPos);
36 : }
37 : //---- データクリアー通知 ----------------------------------------------------------------------------------//
38 : VO  CAjxTchEx::OnNtcClear       ()
39 : {
40 : }
41 : //---- ダブルクリック通知 ----------------------------------------------------------------------------------//
42 : VO  CAjxTchEx::OnNtcDblClk      ()
43 : {
44 : }
45 : //---- ファイルドロップ通知 --------------------------------------------------------------------------------//
46 : VO  CAjxTchEx::OnNtcDropFile    (UI                nFiles)
47 : {
48 :     UT  path[MAX_PATH];
49 :     while (GetDroppedFile(path)) {
50 :         //・・・
51 :     }
52 : }
53 : //---- フォルダドロップ通知 --------------------------------------------------------------------------------//
54 : VO  CAjxTchEx::OnNtcDropDir     (UI                nDirs )
55 : {
56 :     UT  path[MAX_PATH];
57 :     while (GetDroppedDir(path)) {
58 :         //・・・
59 :     }
60 : }
61 : //---- 右クリック通知 --------------------------------------------------------------------------------------//
62 : VO  CAjxTchEx::OnNtcRClick      (PCAJCTCRCLK       pRClk )
63 : {
64 : }
```

「CAjxG3d」クラスの継承（３Ｄグラフィック・クラス）

●AjxG3dEx.h（新規作成）

```
 1 : #pragma once
 2 : #include <AjxCpp.h>
 3 : using namespace AjxControl;
 4 :
 5 : class CAjxG3dEx :
 6 :     public CAjxG3d,
 7 :     public CWnd
 8 : {
 9 : public:
10 :     CAjxG3dEx(void);
11 :     ~CAjxG3dEx(void);
12 :
13 :     VO      Attach          (HWND hWnd)                 override;   //  ハンドルを関連付け
14 :
15 :     VO      OnNtcRotTheta   (PCAJC3DVEC pVec)           override;   //  視点角度通知
16 :     VO      OnNtcPlotList   (PCAJC3DGPLOTLIST pList)    override;   //  プロットリスト通知
17 :     VO      OnNtcClear      (int Factor)                override;   //  データクリアー通知
18 :     VO      OnNtcDblClk     ()                          override;   //  ダブルクリック通知
19 :     VO      OnNtcDropFile   (UI n)                      override;   //  ファイルドロップ通知
20 :     VO      OnNtcDropDir    (UI n)                      override;   //  フォルダドロップ通知
21 :     VO      OnNtcRClick     (PCAJC3DGRCLK pRClk)        override;   //  右クリック通知
22 :
23 : };
24 :
```

●AjxG3dEx.cpp（新規作成）

```
1 : #include "StdAfx.h"
2 : #include "AjxG3dEx.h"
3 :
4 : //---- コンストラクタ --------------------------------------------------------------------------------------//
```

```
 5 : CAjxG3dEx::CAjxG3dEx(void)
 6 : {
 7 : }
 8 : //---- デストラクタ ----------------------------------------------------------------------------//
 9 : CAjxG3dEx::~CAjxG3dEx(void)
10 : {
11 :     //  CWnd からウインドを切り離す (CWnd でウインドを破棄しないようにする)
12 :     if (CWnd::m_hWnd != NULL) {
13 :         CWnd::Detach();
14 :     }
15 : }
16 : //---- ウインドハンドルをアタッチ --------------------------------------------------------------//
17 : VO  CAjxG3dEx::Attach (HWND hwnd)
18 : {
19 :     CWnd::   Attach(hwnd);
20 :     CAjxG3d::Attach(hwnd);
21 : }
22 : //---- 視点角度通知 ----------------------------------------------------------------------------//
23 : VO  CAjxG3dEx::OnNtcRotTheta    (PCAJC3DVEC pVec)
24 : {
25 : }
26 : //---- プロットリスト通知 ----------------------------------------------------------------------//
27 : VO  CAjxG3dEx::OnNtcPlotList    (PCAJC3DGPLOTLIST pList)
28 : {
29 : }
30 : //---- データクリアー通知 ----------------------------------------------------------------------//
31 : VO  CAjxG3dEx::OnNtcClear       (int Factor)
32 : {
33 : }
34 : //---- ダブルクリック通知 ----------------------------------------------------------------------//
35 : VO  CAjxG3dEx::OnNtcDblClk     ()
36 : {
37 : }
38 : //---- ファイルドロップ通知 --------------------------------------------------------------------//
39 : VO  CAjxG3dEx::OnNtcDropFile    (UI n)
40 : {
41 : }
42 : //---- フォルダドロップ通知 --------------------------------------------------------------------//
43 : VO  CAjxG3dEx::OnNtcDropDir     (UI n)
44 : {
45 : }
46 : //---- 右クリック通知 --------------------------------------------------------------------------//
47 : VO  CAjxG3dEx::OnNtcRClick      (PCAJC3DGRCLK pRClk)
48 : {
49 : }
50 :
```

「CAjxVth」クラスの継承（ＶＴ１００ウインド・クラス）

●AjxVthEx.h（新規作成）

```
 1 : #pragma once
 2 : #include <AjxCpp.h>
 3 : using namespace AjxControl;
 4 :
 5 : class CAjxVthEx :
 6 :     public CAjxVth,
 7 :     public CWnd
 8 : {
 9 : public:
10 :     CAjxVthEx(void);
11 :     ~CAjxVthEx(void);
12 :
13 :     VO  Attach          (HWND hWnd)                 override;       //  ハンドルを関連付け
14 :     VO  OnNtcDblClk     (UI flag)                   override;       //  ダブルクリック通知
15 :     VO  OnNtcKeyIn      (UI key, UI rep)            override;       //  キー入力通知
16 :     VO  OnNtcVKeyIn     (UI key, UI rep)            override;       //  拡張キー押下通知
17 :     VO  OnNtcVKeyOut    (UI key)                    override;       //  拡張キー離し通知
18 :     VO  OnNtcDropFile   (UI             nFiles )    override;       //  ファイルドロップ通知
19 :     VO  OnNtcDropDir    (UI             nDirs  )    override;       //  ディレクトリドロップ通知
20 :     VO  OnNtcCharInfo   (UI             height )    override;       //  文字サイズ情報の変化通知
21 :     VO  OnNtcHScroll    (UI             left   )    override;       //  横スクロール通知
22 :     VO  OnNtcVScroll    (UI             top    )    override;       //  縦スクロール通知
23 :     VO  OnNtcRClick     (PCAJCVTHRCLK   pRClk  )    override;       //  右クリック通知
24 :     VO  OnNtcClear      ()                          override;       //  画面クリアー通知
25 :
26 : };
27 :
```

●AjxVthEx.cpp（新規作成）

```
 1 : #include "StdAfx.h"
 2 : #include "AjxVthEx.h"
 3 :
 4 : //---- コンストラクタ --------------------------------------------------------------------------//
```

```
CAjxVthEx::CAjxVthEx(void)
{
}
//---- デストラクタ ----------------------------------------------------------------------------//
CAjxVthEx::~CAjxVthEx(void)
{
    //  CWnd からウインドを切り離す (CWnd でウインドを破棄しないようにする)
    if (CWnd::m_hWnd != NULL) {
        CWnd::Detach();
    }
}
//---- ウインドハンドルをアタッチ --------------------------------------------------------------//
VO  CAjxVthEx::Attach (HWND hwnd)
{
    CWnd::   Attach(hwnd);
    CAjxVth::Attach(hwnd);
}
//---- ダブルクリック通知 ----------------------------------------------------------------------//
VO  CAjxVthEx::OnNtcDblClk  (UI flag)
{
}
//---- キー入力通知 ----------------------------------------------------------------------------//
VO  CAjxVthEx::OnNtcKeyIn   (UI key, UI rep)
{
}
//---- 拡張キー押下通知 ------------------------------------------------------------------------//
VO  CAjxVthEx::OnNtcVKeyIn  (UI key, UI rep)
{
}
//---- 拡張キー離し通知 ------------------------------------------------------------------------//
VO  CAjxVthEx::OnNtcVKeyOut (UI key)
{
}
//---- ファイルドロップ通知 --------------------------------------------------------------------//
VO  CAjxVthEx::OnNtcDropFile(UI             nFiles )
{
}
//---- ディレクトリドロップ通知 ----------------------------------------------------------------//
VO  CAjxVthEx::OnNtcDropDir (UI             nDirs  )
{
}
//---- 文字サイズ情報の変化通知 ----------------------------------------------------------------//
VO  CAjxVthEx::OnNtcCharInfo(UI             height )
{
}
//---- 横スクロール通知 ------------------------------------------------------------------------//
VO  CAjxVthEx::OnNtcHScroll (UI             left   )
{
}
//---- 縦スクロール通知 ------------------------------------------------------------------------//
VO  CAjxVthEx::OnNtcVScroll (UI             top    )
{
}
//---- 右クリック通知 --------------------------------------------------------------------------//
VO  CAjxVthEx::OnNtcRClick  (PCAJCVTHRCLK   pRClk  )
{
}
//---- 画面クリアー通知 ------------------------------------------------------------------------//
VO  CAjxVthEx::OnNtcClear   ()
{
}
```

65. Windows API の不具合

本ライブラリの開発中に 発見した Windows API の不具合と思われる現象を報告します。
尚、当方の勘違いや、あるいは、将来の Windows バージョンにて解消されることも想定されるため、内容については、
自身でもご確認の上、参考にして頂ければと思います。

65.1. コンソールから全角文字入力時 _kbhit(), _getch() が誤動作

1）コンソールから全角文字を入力すると、最後の _kbhit()が常に TRUE を返し、_getch()は最後の全角文字の１バイト目は返さずに、永久に２バイト目の文字コードを返す。（つまり、全角文字を入力すると無限ループしてしまう）

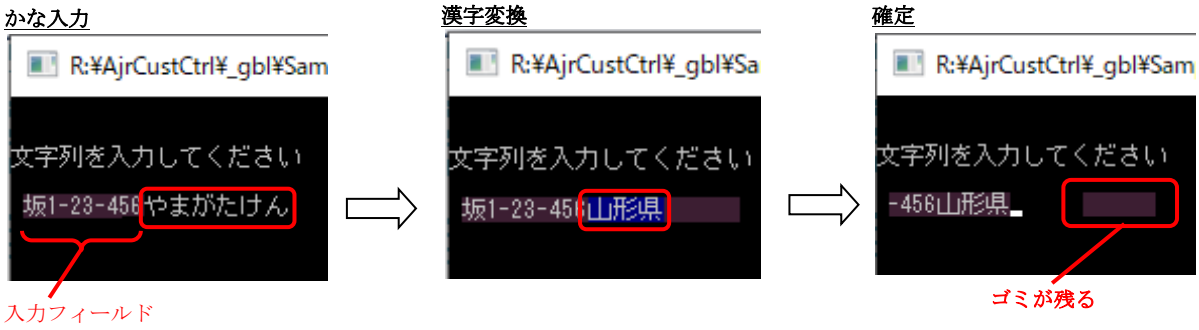
例えばコンソールから「仙台」（バイト順で 90, E5, 91, E4）と入力すると、_getch()は 90, E5 を返した後、永久に E4 を返す。

```
int main()
{
    UI  c;
    while (TRUE) {
        if (_kbhit()) {
            c = _getch();
            printf("%02X ",
c);
        }
    }
    return 0;
}
```

「Alt+漢字」押下後
「仙台」を入力

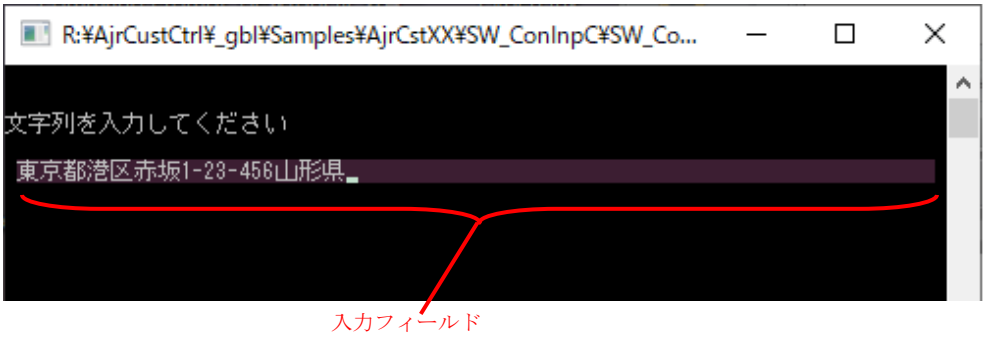

R:¥AjrCustCtrl¥LocalTool¥Test1¥..¥..¥_gbl¥bin¥Test1.exe
00 29 90 E5 E4 E4 E4

３）ＩＭＥのかな漢字変換でコンソール入力する場合、下図のように、かな表示部分にゴミが残る場合があります。（ＩＭＥの仕様？）
（下図は、AjcConInputEx()で入力した場合の現象）



対処方法

AjcConInputEx()でコンソール入力する場合、入力フィールドをコマンドプロンプト・ウインドの右端までとなるように指定する。
（lInpField 引数に大きな値（例えば 512）を指定する）



この現象は、ＩＭＥに起因すると思われます。
上記の対処方法は（ＩＭＥの挙動を精査した訳ではありませんので）補償の限りではありません。

４）コンソールバッファ情報設定

コンソールバッファ情報の設定（SetConsoleScreenBufferInfoEx）で、コンソール表示域の矩形情報(srWindowRect)の下端行位置(Bottom)の値を実際の値＋１とする必要がある。（＋１しないと、コンソールの行数が１つ少なくなる）

例えば、取得したコンソールバッファ情報をそのまま設定する場合、以下のようにします。

```
CONSOLE_SCREEN_BUFFER_INFOEX bi;

GetConsoleScreenBufferInfoEx(handle, &bi);
bi.srWindow.Bottom++;  // 設定する場合 Bottom を+1 する（WindowsAPI バグ？）
SetConsoleScreenBufferInfoEx(handle, &bi);
```

66. 問い合わせ先

本ソフトウェアに関するお問い合わせは、件名の先頭を「Ajara:」として、以下のメールアドレスに送付してください。

xxxajarakojara@kk.email.ne.jpxxx

［注］先頭と末尾の「xxx」は削除してください。
「@」は、全角となっていますので、半角に訂正してください。

メールアドレスは変更される場合がありますので、以下の URL（→お問い合わせ）で確認してください。

http://www.ne.jp/asahi/ajara/kojara/