

Windows カスタムライブラリ

ライブラリ説明書

C# / VB.NET 用

.Net Framework クラスライブラリ編

(32/64 ビット プラットフォーム兼用)

(1. 6. 8. 4 版)

1.	概 要	1
2.	共通定義 (AjrCustCtrl.dll)	4
3.	テキスト表示拡張機能	7
4.	ツールボックス	9
5.	タイムチャート・グラフ表示コントロール (CAjrTimeChart.dll)	10
6.	2D／3Dグラフィック描画コントロール (CAjr3DGraphic.dll)	35
7.	VT-100エミュレーション・ウインド コントロール (CAjrVT100.dll)	77
8.	数値入力コントロール (CAjrInpValue.dll)	103
9.	棒グラフ／折れ線グラフ・コントロール (CAjrBarGraph.dll)	110
10.	通信コントロールで共通な内容の説明	124
11.	シリアル通信コントロール (CAjrSerialComPort.dll)	130
12.	ソケット (TCP/IP) サーバ機能 (CAjrSockServer.dll)	180
13.	ソケット (TCP/IP) クライアント機能 (CAjrSockClient.dll)	209
14.	ファイル検索 (CAjrFileSearch.dll)	236
15.	テキストファイル・アクセス (CAjrTextFile.dll)	242
16.	文字列プール (CAjrStrPool.dll)	253
17.	C言語の字句分解 (CAjrCToken.dll)	262
18.	 C言語のプリコンパイル (CAjrCPrePro.dll)	277
19.	スタティク ファンクション (CAjrStatic.dll)	301
20.	タティク：ツールチップ (CAjrStatic.dll, SAjrTip クラス)	303
21.	スタティク：プロファイル／レジストリアクセス (CAjrStatic.dll, SAjrReg クラス)	317
22.	スタティク：ファイル／フォルダ操作 (CAjrStatic.dll, SAjrFop クラス)	330
23.	スタティク：3D／2Dベクトル等の演算 (CAjrStatic.dll, SAjrMath クラス)	346
24.	スタティク：汎用ファンクション (CAjrStatic.dll, SAjrGsr クラス)	360
25.	スタティク：チェックサム (CAjrStatic.dll, SAjrCS クラス)	371
26.	スタティク：CRC 計算 (CAjrStatic.dll, SAjrCrc クラス)	375
27.	スタティク：ネイティブデータ・アクセス (CAjrStatic.dll, SAjrBin クラス)	381
28.	免責事項	382
29.	問い合わせ先	382

1. 概 要	1
1.1. 実行可能な Windows/VisualStudio/.NETFramework バージョン	2
1.2. 構造体や列挙体（定数）の参照	2
1.3. 言語設定	2
1.4. サンプルプログラム	2
1.5. バージョン更新時の注意事項	2
1.6. 参照の設定	3
1.7. コントロール(DLL)のバージョンを特定しない	3
1.8. ツールボックスアイテムの追加	3
2. 共通定義 (AjrCustCtrl.dll)	4
2.1. 構造体/列挙体（定数）	4
2.1.1. 2Dベクトル	4
2.1.2. 2D線ベクトル（始点と方向ベクトル）	4
2.1.3. 3Dベクトル	4
2.1.4. 3D線ベクトル（始点と方向ベクトル）	4
2.1.5. 3Dラインポイント（始点と終点）	5
2.1.6. 3D三角形情報（三角形の頂点）	5
2.1.7. 3D行列（3×3の行列）	5
2.1.8. 3D/2D平面表示時の横軸/縦軸の種別（3×3の行列）	5
2.1.9. テキストファイル エンコード	6
2.1.10. テキストファイル 改行コード変換モード	6
2.1.11. ファイル属性	6
3. テキスト表示拡張機能	7
3.1. 制御文字	7
3.2. エスケープシーケンス	8
3.3. デザイン時のプロパティ（テキスト）の設定について	8
4. ツールボックス	9
5. タイムチャート・グラフ表示コントロール (CAjrTimeChart.dll)	10
5.1. 機能概要	10
5.1.1. ポップアップメニュー	10
5.1.2. レンジ設定	11
5.1.3. ワンタッチでレンジ設定	11
5.1.4. レンジ自動調整	11
5.1.5. ドラッグ操作によるレンジ設定	11
5.1.6. オフセット設定	12
5.1.7. その他の設定	12
5.1.8. フィルタの設定と表示/非表示	13
5.1.9. 波形の補間表示設定	13
5.1.10. 波形の補間表示	14
5.1.11. 横線描画	15
5.1.12. 縦線描画	15
5.1.13. プロット周期表示	16
5.1.14. 時間計測	16
5.1.15. 描画時間情報表示	17
5.1.16. 右クリック	17
5.1.17. ファイルやディレクトリのドラッグ&ドロップ	17
5.1.18. 表示の高速化	18
5.2. 構造体/列挙体（定数）	19
5.2.1. 線の属性	19
5.2.2. 波形補間表示種別（補間対象データ）	19
5.3. プロパティ	20
5.4. メソッド	21
5.4.1. グラフ表示停止(Stop)	22
5.4.2. グラフ表示開始(Start)	22
5.4.3. データ投与(PutData)	22
5.4.4. データ項目のオフセット値設定(SetItemOffset)	22

5.4.5.	データ項目のオフセット値取得 (GetItemOffset)	22
5.4.6.	データ項目の表示色設定 (SetItemColor)	23
5.4.7.	データ項目の表示色取得 (GetItemColor)	23
5.4.8.	グラフレンジの自動調整 (AdjustRange)	23
5.4.9.	スクロール位置の設定 (SetScrollPos)	23
5.4.10.	スクロール位置の取得 (SetScrollPos)	23
5.4.11.	フィルタの設定 (SetFilter)	24
5.4.12.	フィルタの設定値の取得 (GetFilter)	24
5.4.13.	横線スタイル設定 (SetHoriLineStyle)	24
5.4.14.	横線描画位置設定 (SetHoriLinePos)	24
5.4.15.	横線表示／非表示 (EnableHoriLine)	25
5.4.16.	縦線描画 (SetVertLine)	25
5.4.17.	縦線描画の表示／非表示 (EnableVertLine)	25
5.4.18.	ドロップされたファイル名取得 (GetDroppedFile)	25
5.4.19.	ドロップされたディレクトリ名取得 (GetDroppedDir)	25
5.4.20.	タイトルテキスト表示 (SetTitleText)	26
5.4.21.	現在の設定値をプロファイルへ記録する (SaveToProfile)	26
5.4.22.	設定値をプロファイルから読み出す (LoadFromProfile)	26
5.4.23.	画面表示の停止／再開 (Pause)	26
5.4.24.	プロット周期計測値の表示 (MesPeriShow)	27
5.4.25.	プロット周期計測をリセット (MesPeriReset)	27
5.4.26.	テキスト描画 (TextOut) - ピクセル位置指定	27
5.4.27.	チャートデータ破棄 (PurgePlot)	27
5.4.28.	テキスト描画データ破棄 (PurgeText)	27
5.4.29.	全てのデータ破棄 (Purge)	28
5.5.	イベント	29
5.5.1.	レンジ通知 (OnRangeChanged)	29
5.5.2.	現在のスクロール位置通知 (OnScrPosChanged)	29
5.5.3.	ファイルドロップ通知 (OnFileDrop)	29
5.5.4.	ディレクトリドロップ通知 (OnDirDrop)	30
5.5.5.	右クリック通知 (OnRClick)	30
5.6.	サンプルプログラム	31
5.6.1.	Sil_TimeChart (タイムチャート)	31
6.	2D／3Dグラフィック描画コントロール (CAjr3DGraphic.dll)	35
6.1.1.	プロットデータと図形描画データ	36
6.2.	機能概要	37
6.2.1.	ポップアップメニュー	37
6.2.2.	レンジ設定	38
6.2.3.	ワンタッチでレンジ設定	38
6.2.4.	レンジ自動調整	38
6.2.5.	アスペクト比の設定	39
6.2.6.	フィルタ機能	39
6.2.7.	視点の設定	39
6.2.8.	スケールの表示	40
6.2.9.	描画時間情報表示	41
6.2.10.	奥行き表現	41
6.2.11.	2Dグラフモード	42
6.2.12.	表示の高速化	43
6.2.13.	任意の平面定義と平面上への図形描画	43
6.2.14.	右クリック	44
6.2.15.	ファイルやディレクトリのドラッグ&ドロップ	44
6.3.	プロパティ	45
6.4.	メソッド	46
6.4.1.	プロットデータ投与 (PutData)	47
6.4.2.	レンジ設定 (SetRange)	47
6.4.3.	レンジ自動調整 (AdjustRange)	47
6.4.4.	中心位置設定 (SetCenter)	48
6.4.5.	半径設定 (SetRadius)	48
6.4.6.	各軸の半径を同一値にする (SetSameRadius)	48

6.4.7.	ピクセル描画(Pixel)	49
6.4.8.	ライン始点設定(MoveTo)	49
6.4.9.	ライン終点設定 - ライン/矢印描画(LineTo)	49
6.4.10.	ライン/矢印描画(Line)	50
6.4.11.	三角形描画(Triangle)	50
6.4.12.	四角形描画(Square)	51
6.4.13.	立方体/長方体描画(Cube)	51
6.4.14.	球/楕円球描画(Sphere)	51
6.4.15.	3D空間上に任意の平面を定義(DefPlane)	52
6.4.16.	円/楕円描画(Ellipse)	52
6.4.17.	星形描画(Star)	53
6.4.18.	フィルタの設定(SetFilter)	53
6.4.19.	フィルタの設定値の取得(GetFilter)	54
6.4.20.	最大プロット数設定(SetMaxPlot)	54
6.4.21.	最大プロット数取得(SetMaxPlot)	54
6.4.22.	データ項目の前面表示色設定(SetItemColorP)	54
6.4.23.	データ項目の前面表示色取得(GetItemColorP)	54
6.4.24.	データ項目の後面表示色設定(SetItemColorN)	55
6.4.25.	データ項目の後面表示色取得(GetItemColorN)	55
6.4.26.	視点設定(SetAngle)	55
6.4.27.	視点をX Y平面に設定(SetAngleXY)	56
6.4.28.	視点をX Z平面に設定(SetAngleXZ)	56
6.4.29.	視点をY Z平面に設定(SetAngleYZ)	56
6.4.30.	視点を3Dイメージに設定(SetAngle3D)	56
6.4.31.	視点を任意の平面に設定(SetAnyPlane)	56
6.4.32.	ドロップされたファイル名取得 (GetDroppedFile)	57
6.4.33.	ドロップされたディレクトリ名取得 (GetDroppedDir)	57
6.4.34.	タイトルテキスト表示 (SetTitleText)	57
6.4.35.	現在の設定値をプロファイルへ記録する (SaveToProfile)	57
6.4.36.	設定値をプロファイルから読み出す (LoadFromProfile)	58
6.4.37.	ボーダー色で囲まれた部分の塗りつぶし(FillB)・・・2Dモード専用	58
6.4.38.	連続する白色部分の塗りつぶし(FillS)・・・2Dモード専用	58
6.4.39.	ピクセルの表示色取得 (GetPixel)・・・2Dモード専用	58
6.4.40.	画面表示の停止/再開 (Pause)	59
6.4.41.	テキスト描画 (TextOut) - ピクセル位置指定	59
6.4.42.	テキスト描画 (TextOut) - 2Dモード用	59
6.4.43.	テキスト描画 (TextOut) - 3Dモード用	60
6.4.44.	図形描画データ破棄 (ClearShape)	60
6.4.45.	プロットデータ破棄(PurgePlot)	60
6.4.46.	テキスト描画データ破棄(PurgeText)	60
6.4.47.	全ての描画データ破棄(PurgeData)	61
6.4.48.	全てのデータ破棄(Purge)	61
6.5.	イベント	62
6.5.1.	ファイルドロップ通知 (OnFileDrop)	62
6.5.2.	ディレクトリドロップ通知 (OnDirDrop)	62
6.5.3.	右クリック通知 (OnRClick)	62
6.5.4.	Shift+左クリックで クリックしたプロット点情報の通知(OnPltLst)	63
6.6.	サンプルプログラム	64
6.6.1.	Sil_3DGraphic1 (3Dグラフィック)	64
6.6.2.	Sil_3DGraphic2 (2Dグラフィック)	72
6.6.3.	Sil_3DGraphic3 (視点設定)	75
7.	VT-100エミュレーション・ウインド コントロール (CAjrVT100.d11)	77
7.1.	機能概要	77
7.1.1.	ポップアップメニュー	77
7.1.2.	文字列の検索	78
7.1.3.	テキストの選択とコピー	79
7.1.4.	フォントの設定	79
7.1.5.	その他の設定	79
7.1.6.	ファイルヘセーブ	80

7.1.7.	表示の高速化	80
7.1.8.	描画処理の高速化	81
7.1.9.	ANSIエスケープコード	81
7.1.10.	カーソル位置	82
7.1.11.	スクロールアウトした行のバッファリング	82
7.1.12.	右クリック	82
7.1.13.	オートスクロール	83
7.1.14.	VRAMとバッファを識別して表示	83
7.1.15.	描画テキストの一時保留	83
7.1.16.	マージン	84
7.1.17.	表示内容をファイルへ出力	84
7.1.18.	ファイルやディレクトリのドロップ	85
7.1.19.	固定ピッチ表示	85
7.1.20.	画面クリアボタン	85
7.2.	制御コードとANSIエスケープコード	86
7.4.	プロパティ	87
7.5.	メソッド	88
7.5.1.	1文字描画 (PutChar)	89
7.5.2.	1バイト描画 (PutByte)	89
7.5.3.	文字列描画 (PutText)	89
7.5.4.	書式文字列出力 (PutFormat)	89
7.5.5.	タイムスタンプ描画 (PrintTimeStamp)	90
7.5.6.	16進ダンプ描画 (PrintHexDump)	90
7.5.7.	カーソル位置設定 (Locate)	90
7.5.8.	パレット色設定 (SetPaletteColor)	90
7.5.9.	パレット色取得 (GetPaletteColor)	90
7.5.10.	部分テキスト選択 (Select)	91
7.5.11.	全テキスト選択 (SelectAll)	91
7.5.12.	選択テキストをクリップボードへコピー (Copy)	91
7.5.13.	全テキスト 破棄 (Purge)	91
7.5.14.	ドロップされたファイル名取得 (GetDroppedFile)	92
7.5.15.	ドロップされたディレクトリ名取得 (GetDroppedDir)	92
7.5.16.	タイトルテキスト表示 (SetTitleText)	92
7.5.17.	行テキスト取得 (GetLineText)	92
7.5.18.	ダブルクリック行テキスト取得 (GetDbClickedLineText)	93
7.5.19.	ダブルクリック行位置取得 (GetDbClickedLinePos)	93
7.5.20.	文字列の検索 (SearchBelow)	93
7.5.21.	設定値/フォント情報をプロファイルへ記録 (SaveToProfile, SaveFontToProfile)	94
7.5.22.	設定値/フォント情報をプロファイルから読み出す (LoadFromProfile, LoadFontFromProfile)	94
7.5.23.	画面表示の停止/再開 (Pause)	94
7.5.24.	フォント設定 (SetFont)	95
7.5.25.	フォント取得 (GetFont)	95
7.6.	イベント	96
7.6.1.	ダブルクリック通知 (OnNtcDbClicked)	96
7.6.2.	キー入力通知 (OnNtcKeyIn)	96
7.6.3.	拡張キー押下通知 (OnNtcVKeyIn)	96
7.6.4.	拡張キー離し通知 (OnNtcVKeyOut)	97
7.6.5.	ファイルドロップ通知 (OnFileDrop)	97
7.6.6.	ディレクトリドロップ通知 (OnDirDrop)	97
7.6.7.	文字サイズ変化通知 (OnCharInfo)	97
7.6.8.	右クリック通知 (OnRClick)	98
7.7.	サンプルプログラム	99
8.	数値入力コントロール (CAjrInpValue.dll)	103
8.1.	ツールチップテキスト	104
8.2.	プロパティ	105
8.3.	メソッド	106
8.3.1.	値の設定 (SetValue)	106
8.4.	イベント	107
8.4.1.	整数モードでの数値設定通知 (OnNtcIntValue)	107

8.4.2.	実数モードでの数値設定通知 (OnNtcRealValue)	107
8.4.3.	右クリック通知 (OnNtcRClick)	107
8.5.	サンプルプログラム	108
8.5.1.	Sil_InpVal1 (色や透明度の設定)	108
9.	棒グラフ／折れ線グラフ・コントロール (CAjrBarGraph.dll)	110
9.1.	機能概要	110
9.1.1.	フィルタ機能	111
9.1.2.	折れ線グラフ	111
9.1.3.	右クリック	112
9.1.4.	ファイルやディレクトリのドラッグ&ドロップ	112
9.2.	プロパティ	113
9.3.	メソッド	114
9.3.1.	データ投与(PutData)	115
9.3.2.	データ項目の表示色設定(SetItemColor)	115
9.3.3.	データ項目の表示色取得(GetItemColor)	115
9.3.4.	データ破棄(Purge)	115
9.3.5.	スクロール位置の設定 (SetScrollPos)	116
9.3.6.	スクロール位置の取得 (SetScrollPos)	116
9.3.7.	フィルタの設定(SetFilter)	116
9.3.8.	フィルタの設定値の取得(GetFilter)	116
9.3.9.	横線スタイル設定(SetHoriLineStyle)	117
9.3.10.	横線描画位置設定(SetHoriLinePos)	117
9.3.11.	横線表示／非表示(EnableHoriLine)	117
9.3.12.	ドロップされたファイル名取得 (GetDroppedFile)	117
9.3.13.	ドロップされたディレクトリ名取得 (GetDroppedDir)	118
9.3.14.	タイトルテキスト表示 (SetTitleText)	118
9.3.15.	現在の設定値をプロファイルへ記録する (SaveToProfile)	118
9.3.16.	設定値をプロファイルから読み出す (LoadFromProfile)	118
9.3.17.	テキスト描画 (TextOut) - ピクセル位置指定	119
9.3.18.	グラフデータ破棄(PurgePlot)	119
9.3.19.	テキスト描画データ破棄(PurgeText)	119
9.3.20.	全てのデータ破棄(Purge)	119
9.4.	イベント	120
9.4.1.	レンジ通知 (OnRangeChanged)	120
9.4.2.	ファイルドロップ通知 (OnFileDrop)	120
9.4.3.	ディレクトリドロップ通知 (OnDirDrop)	120
9.4.4.	右クリック通知 (OnRClick)	121
9.5.	サンプルプログラム	122
9.5.1.	Sil_BarGraph1 (棒グラフと折れ線グラフ)	122
10.	通信コントロールで共通な内容の説明	124
10.1.	チャンクデータ	124
10.2.	データ区切りの認識	124
10.3.	テキストデータのマルチバイト文字分断抑止	125
10.4.	受信通知イベント	125
10.4.1.	テキスト チャンク・データ受信通知 (OnRxChunkTxt)	126
10.4.2.	バイナリ チャンク・データ受信通知 (OnRxChunkBin)	126
10.4.3.	不正チャンクテキスト受信通知 (OnRxInvalidChunk)	126
10.4.4.	テキスト受信通知 (OnRxText)	126
10.4.5.	E S Cシーケンス受信通知 (OnRxEsc)	127
10.4.6.	パケットデータ受信通知 (OnRxPkt)	127
10.4.7.	パケット外テキスト受信通知 (OnRxNoPkt)	128
10.4.8.	バイトペアによるワード(14Bit)データ受信通知 (OnRxWord14)	128
10.4.9.	パケットフレーム外での透過制御バイト (DLE)	128
10.5.	送受信動作	129
10.6.	送受信テキストデータの文字コード	129
11.	シリアル通信コントロール (CAjrSerialComPort.dll)	130
11.1.	機能概要	130
11.1.1.	受信データの通知形式と送受信文字コード	130

11.1.2.	イベントの通知方法	130
11.1.3.	COM ポート通信	131
11.1.4.	メールスロット (LAN) 通信	131
11.1.5.	ソケット (TCP/IP クライアント) 通信	131
11.1.6.	ソケット (TCP/IP サーバ) 通信	131
11.1.7.	各通信リソースを切り替えて通信	132
11.1.8.	D C Bとタイムアウト情報	132
11.2.	構造体／列挙体 (定数)	134
11.2.1.	実行モード	134
11.2.2.	通信リソース選択	134
11.2.3.	ポート状態	134
11.2.4.	チャンクデータの扱い	134
11.2.5.	データビット数	134
11.2.6.	パリティビット	134
11.2.7.	ストップビット	134
11.2.8.	イベントコード	135
11.2.9.	エラーコード	135
11.2.10.	受信テキストの文字コード	136
11.2.11.	送信テキストの文字コード	136
11.2.12.	バイトペア受信順序	136
11.3.	プロパティ	137
11.4.	メソッド	138
11.4.1.	初期設定 (Init)	138
11.4.2.	通信回線オープン (Open)	139
11.4.3.	通信回線クローズ (Close)	139
11.4.4.	バイト文字送信 (SendByte)	140
11.4.5.	1文字送信 (SendChar)	140
11.4.6.	テキスト送信 (SendText)	140
11.4.7.	バイナリ送信 (SendBinary)	140
11.4.8.	パケット送信 (SendPacket)	141
11.4.9.	4ビットワード値 (バイトペア) 送信, Low Byte First (SendWord14LF)	141
11.4.10.	14ビットワード値 (バイトペア) 送信, High Byte First (SendWord14HF)	141
11.4.11.	ブレイク信号送出／停止 (SendBreak)	142
11.4.12.	D T R信号設定 (SetDTR)	142
11.4.13.	R T S信号設定 (SetRTS)	142
11.4.14.	D S R信号状態取得 (GetDSR)	142
11.4.15.	C T S信号状態取得 (GetDSR)	142
11.4.16.	R L S D信号状態取得 (GetRLSD)	142
11.4.17.	R I N G信号状態取得 (GetRING)	143
11.4.18.	受信済データデータ破棄 (PurgeRx)	143
11.4.19.	送信待ちデータデータ破棄 (PurgeTx)	143
11.4.20.	送受信データ破棄 (Purge)	143
11.4.21.	ポート種別の設定	143
11.4.22.	ダイアログによる通信パラメタ設定 (SetParamByDialog)	144
11.4.23.	ダイアログによる詳細な通信パラメタ設定 (SetDetailParamByDialog)	145
11.4.24.	通信パラメタ設定ダイアログによるラジオボタンの選択許可／禁止 (EnablePortSelectionInDialog)	145
11.4.25.	イベント待ち (WaitEvent)	146
11.4.26.	イベントマスク設定 (SetEvtMask)	147
11.4.27.	イベントマスク取得 (GetEvtMask)	147
11.4.28.	ポート名取得 (GetPortPathName)	148
11.4.29.	C O Mポートのデバイス名称取得 (GetPortDevName)	148
11.4.30.	自メールスロットパス名取得 (GetMySlotPathName)	148
11.4.31.	自メールスロット生成 (CreateMySlot)	148
11.4.32.	自メールスロット消去 (DeleteMySlot)	149
11.4.33.	メールスロット名情報設定 (DeleteMySlot)	149
11.4.34.	シリアル通信コントロールの消去 (Delete)	149
11.5.	イベント	150
11.5.1.	ポート状態通知 (OnPortState)	150
11.5.2.	テキスト・チャンクデータ受信通知 (OnRxChunkTxt)	150
11.5.3.	バイナリ・チャンクデータ受信通知 (OnRxChunkBin)	151

11. 5. 4.	テキスト受信通知 (OnRxText)	151
11. 5. 5.	E S Cシーケンス受信通知 (OnRxEsc)	151
11. 5. 6.	制御コード受信通知 (OnRxCtrl)	151
11. 5. 7.	パケットデータ受信通知 (OnRxPacket)	151
11. 5. 8.	パケット外テキスト受信通知 (OnRxNoPkt)	152
11. 5. 9.	送信完了通知 (OnTxEmpty)	152
11. 5. 10.	エラー発生通知 (OnError)	152
11. 5. 11.	R I N G信号変化通知 (OnNtcRING)	153
11. 5. 12.	R L S D信号変化通知 (OnNtcRLSD)	153
11. 5. 13.	D S R信号変化通知 (OnNtcDSR)	153
11. 5. 14.	C T S信号変化通知 (OnNtcCTS)	153
11. 5. 15.	バイトペアによるワード (14Bit) データ受信 (OnRxWord14)	154
11. 5. 16.	不正チャンクテキスト受信 (OnRxInvChunk)	154
11. 6.	サンプルプログラム	155
11. 6. 1.	Sil_SerialComPort1 (送受信テスト)	155
11. 6. 2.	Sil_SerialComPort2 (エコーバック)	161
11. 6. 3.	Sil_SerialComPort2C (エコーバック, コンソールアプリ)	163
11. 6. 4.	Sil_SerialComPort3 (ループバックによるフォルダ構造コピー)	166
11. 6. 5.	Sil_SerialComPort4 (メールスロット通信)	171
11. 6. 6.	Sil_SerialComPort5 (テキストとバイナリパケットの多重通信)	173
12.	ソケット (TCP/IP) サーバ機能 (CAjrSockServer.dll)	180
12. 1.	機能概要	180
12. 1. 1.	受信データの通知形式と送受信文字コード	180
12. 1. 2.	イベントの通知方法	180
12. 2.	構造体/列挙体 (定数)	181
12. 2. 1.	実行モード	181
12. 2. 2.	チャンクデータの通知モード	181
12. 2. 3.	イベントコード	181
12. 2. 4.	エラーコード	182
12. 2. 5.	受信テキストの文字コード	182
12. 2. 6.	送信テキストの文字コード	182
12. 2. 7.	アドレスファミリ	182
12. 3.	プロパティ	183
12. 4.	メソッド	184
12. 4. 1.	サーバ開始設定 (Start)	185
12. 4. 2.	サーバ停止 (Stop)	185
12. 4. 3.	クライアント列挙 (EnumClients)	185
12. 4. 4.	イベントマスク設定 (SetEvtMask)	185
12. 4. 5.	イベント待ち (WaitEvent)	186
12. 4. 6.	ソケットサーバ・インスタンス消去 (Delete)	187
12. 4. 7.	1 バイト送信 (SendByte)	187
12. 4. 8.	1 文字送信 (SendChar)	187
12. 4. 9.	テキストデータ送信 (SendChar)	187
12. 4. 10.	バイナリデータ送信 (SendBinary)	188
12. 4. 11.	パケット送信 (SendPacket)	188
12. 4. 12.	受信済データ破棄 (PurgeRx)	188
12. 4. 13.	送信待ちデータ破棄 (PurgeTx)	189
12. 4. 14.	送受信データ破棄 (Purge)	189
12. 4. 15.	クライアントにデータを関連付ける (SetClientData)	189
12. 4. 16.	クライアントに関連付けられたデータ取得 (GetClientData)	189
12. 4. 17.	クライアント接続順序番号取得 (GetSeqNo)	190
12. 4. 18.	クライアントのインデクス値取得 (GetSeqNo)	190
12. 4. 19.	クライアントの I P アドレス文字列取得 (GetIpAddrStr)	190
12. 4. 20.	クライアントスレッドのスレッド I D 取得 (GetThreadId)	190
12. 4. 21.	実際の受信テキストコード取得 (GetActualRxTextCode)	191
12. 4. 22.	実際の送信テキストコード取得 (GetActualTxTextCode)	191
12. 4. 23.	クライアントを切断 (Disconnect)	191
12. 5.	イベント	192
12. 5. 1.	サーバ開始通知 (OnStart)	192

12.5.2.	サーバ停止通知 (OnStop)	192
12.5.3.	クライアント列挙通知 (OnEnumClients)	193
12.5.4.	テキスト受信通知 (OnRxText)	193
12.5.5.	E S C受信通知 (OnRxEsc)	193
12.5.6.	制御コード受信通知 (OnRxEsc)	193
12.5.7.	パケット受信通知 (OnRxPacket)	194
12.5.8.	パケット外テキスト受信通知 (OnRxNoPkt)	194
12.5.9.	送信完了通知 (OnTxEmpty)	194
12.5.10.	テキストチャンク受信通知 (OnRxChunkTxt)	194
12.5.11.	バイナリチャンク受信通知 (OnRxChunkBin)	195
12.5.12.	不正チャンクテキスト受信通知 (OnRxInvChunk)	195
12.5.13.	接続通知 (OnConnect)	195
12.5.14.	切断通知 (OnDisconnect)	195
12.5.15.	受信エラー通知 (OnRecvError)	195
12.5.16.	送信エラー通知 (On SendError)	196
12.5.17.	その他のエラー通知 (OnGeneralError)	196
12.5.18.	クライアント列挙通知 (OnEnumClients)	196
12.6.	サンプルプログラム	197
12.6.1.	Sil_SockServer1 (送受信テスト)	197
12.6.2.	Sil_SockServer2 (エコーサーバ)	204
12.6.3.	Sil_SockServer2C (エコーサーバ, コンソールアプリ)	206
13.	ソケット (TCP/IP) クライアント機能 (CAjrSockClient.dll)	209
13.1.	機能概要	209
13.1.1.	受信データの通知形式と送受信文字コード	209
13.1.2.	イベントの通知方法	209
13.2.	構造体/列挙体 (定数)	210
13.2.1.	実行モード	210
13.2.2.	チャンクデータの通知モード	210
13.2.3.	イベントコード	210
13.2.4.	エラーコード	211
13.2.5.	回線状態	211
13.2.6.	受信テキストの文字コード	211
13.2.7.	送信テキストの文字コード	211
13.2.8.	アドレスファミリ	211
13.3.	プロパティ	212
13.4.	メソッド	212
13.4.1.	初期設定 (Init)	213
13.4.2.	回線接続 (Connect)	213
13.4.3.	回線切断 (Disconnect)	213
13.4.4.	イベントマスク設定/取得 (SetEvtMask)	213
13.4.5.	イベント待ち (WaitEvent)	214
13.4.6.	1バイト送信 (SendByte)	215
13.4.7.	1文字送信 (SendChar)	215
13.4.8.	テキストデータ送信 (SendChar)	215
13.4.9.	バイナリデータ送信 (SendBinary)	215
13.4.10.	パケット送信 (SendPacket)	216
13.4.11.	受信済データ破棄 (PurgeRx)	216
13.4.12.	送信待ちデータ破棄 (PurgeTx)	216
13.4.13.	送受信データ破棄 (Purge)	216
13.4.14.	インスタンス消去 (Delete)	217
13.5.	イベント	218
13.5.1.	テキスト受信通知 (OnRxText)	218
13.5.2.	E S C受信通知 (OnRxEsc)	218
13.5.3.	制御コード受信通知 (OnRxEsc)	218
13.5.4.	パケット受信通知 (OnRxPacket)	219
13.5.5.	パケット外テキスト受信通知 (OnRxNoPkt)	219
13.5.6.	送信完了通知 (OnTxEmpty)	219
13.5.7.	テキストチャンク受信通知 (OnRxChunkTxt)	219
13.5.8.	バイナリチャンク受信通知 (OnRxChunkBin)	220

13. 5. 9.	不正チャンクテキスト受信通知 (OnRxInvChunk)	220
13. 5. 10.	接続通知 (OnConnect)	220
13. 5. 11.	切断通知 (OnDisconnect)	220
13. 5. 12.	接続失敗通知 (OnCnFail)	220
13. 5. 13.	受信エラー通知 (OnRecvError)	221
13. 5. 14.	送信エラー通知 (On SendError)	221
13. 5. 15.	その他のエラー通知 (OnGeneralError)	221
13. 6.	サンプルプログラム	222
13. 6. 1.	Sil_SockClient1 (送受信テスト)	222
13. 6. 2.	Sil_SockClient2 (エコーバック)	227
13. 6. 3.	Sil_SockClient2C (エコーバック, コンソールアプリ)	229
13. 6. 4.	Sil_SockClient3 (ループバックによるフォルダ構造コピー)	231
14.	ファイル検索 (CAjrFileSearch.dll)	236
14. 1.	メソッド	236
14. 1. 1.	ファイル名検索時の検索フォルダの通知設定 (SetNtcSearchingDir)	236
14. 1. 2.	フォルダ下のファイル検索 (SearchFiles)	237
14. 1. 3.	マイコンピュータ内のファイル検索 (SearchMyCompute)	237
14. 2.	イベント/デリゲート	238
14. 2. 1.	ファイル検索通知 (OnFindFile)	238
14. 3.	サンプルプログラム	239
14. 3. 1.	Sil_FileSearch (ファイル検索 - Windows アプリ)	239
14. 3. 2.	Sil_FileSearchC (ファイル検索 - コンソールアプリ)	241
15.	テキストファイル・アクセス (CAjrTextFile.dll)	242
15. 1.	プロパティ	243
15. 2.	メソッド	243
15. 2. 1.	入力テキストファイル オープン (Open)	244
15. 2. 2.	文字列入力 (GetS)	244
15. 2. 3.	1文字入力 (GetC)	245
15. 2. 4.	ファイル読み出しポイント退避 (SavePoint)	245
15. 2. 5.	ファイル読み出しポイント回復 (RecvPoint)	245
15. 2. 6.	ファイル読み出し位置取得 (GetPoint)	246
15. 2. 7.	ファイル読み出し位置設定 (SetPoint)	246
15. 2. 8.	ファイル読み出しバイト位置取得 (GetBytePoint)	246
15. 2. 9.	入力ファイルのEOFチェック (IsEof)	247
15. 2. 10.	出力テキストファイル 生成 (Create)	247
15. 2. 11.	既存ファイルを追記モードでオープン/生成 (Append)	248
15. 2. 12.	文字列出力 (PutS)	248
15. 2. 13.	1文字出力 (PutC)	248
15. 2. 14.	書式文字列出力 (PutFormat)	249
15. 2. 15.	出力データフラッシュ (Flush)	249
15. 2. 16.	テキストファイルクローズ (Close)	249
15. 3.	サンプルプログラム	250
15. 3. 1.	Sil_TextFile (テキストファイルの読み出しと書き込み)	250
15. 3. 2.	Sil_TextFileC (テキストファイルの読み出しと書き込み, コンソールアプリ)	252
16.	文字列プール (CAjrStrPool.dll)	253
16. 1.	構造体/列挙体 (定数)	253
16. 1. 1.	文字列比較パラメタ	253
16. 1. 2.	部分文字列検索パラメタ	253
16. 1. 3.	文字列読み出し順	253
16. 2.	プロパティ	253
16. 3.	メソッド	254
16. 3. 1.	文字列登録 (Regist / RegistPtr)	254
16. 3. 2.	文字列検索 (Find / FindPtr)	254
16. 3. 3.	部分文字列検索 (PartStrInPool / PartStrInPoolPtr)	255
16. 3. 4.	文字列プール中の文字列を部分文字列とした検索 (PoolStrInStr / PoolStrInStrPtr)	255
16. 3. 5.	文字列削除 (Remove)	256
16. 3. 6.	全文字列破棄 (Reset)	256
16. 3. 7.	登録済み文字列の列挙 (EnumStr)	256

16.3.8.	ポインタ→文字列変換 (PtrToString)	256
16.3.9.	文字列プールの消去 (PtrToString)	257
16.4.	イベント/デリゲート	257
16.4.1.	文字列の通知 (OnNtcStr)	257
16.5.	サンプルプログラム	258
16.5.1.	Sil_StrPool (文字列の登録, 削除, 検索, リセット)	258
16.5.2.	Sil_StrPoolC (コンソールアプリ)	261
17.	C言語の字句分解 (CAjrcToken.dll)	262
17.1.	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	262
17.2.	プリプロセス文情報	263
17.3.	#include 文のヘッダファイル名情報	263
17.4.	構造体/列挙体 (定数)	264
17.4.1.	機能フラグ (ECtkOpt)	264
17.4.2.	トークン識別フラグ (ECtkFlg)	264
17.4.3.	エラーコード (ECtkError)	264
17.4.4.	数値定数のサフィックス・コード (ECtkSuf)	264
17.4.5.	トークンコード (ECtkCode)	265
17.5.	プロパティ	266
17.6.	メソッド	266
17.6.1.	コールバックメソッドの設定 (SetCallBack)	267
17.6.2.	字句の読み出し (GetToken)	267
17.6.3.	字句先読み (PeekToken)	268
17.6.4.	字句コードに対応する文字列の取得 (TokenString)	268
17.6.5.	トークンがユーザシンボルかチェックする (IsUserSymbol)	268
17.6.6.	トークンが予約語シンボルかチェックする (IsReservedSymbol)	268
17.6.7.	トークンが数値定数かチェックする (IsNumericValue)	269
17.6.8.	トークンが文字列かチェックする (IsString)	269
17.6.9.	トークンがパス名かチェックする (IsPathName)	269
17.6.10.	トークンがデリミタかチェックする (IsDelimiter)	269
17.6.11.	トークンがシンボルかチェックする (IsSymbol)	269
17.6.12.	トークンがシンボル/数値定数かチェックする (IsValSym)	270
17.6.13.	トークンがシンボル/数値定数かチェックする (IsSpace)	270
17.6.14.	インスタンス退避 (Push)	270
17.6.15.	インスタンス回復 (Pop)	270
17.6.16.	インスタンスリセット (Reset)	271
17.6.17.	インスタンス消去 (Delete)	271
17.7.	イベント/デリゲート	272
17.7.1.	1行読み出し (OnGetS)	272
17.8.	サンプルプログラム	273
17.8.1.	Sil_CToken (トークンとその属性情報表示)	273
17.8.2.	Sil_CTokenC (コメント除去 - コンソールアプリ)	275
18.	 C言語のプリコンパイル (CAjrcPrePro.dll)	277
18.1.	プリコンパイルオプション	277
18.1.1.	インクルードファイル自動検索 (AUTOSRH)	277
18.1.2.	同一インクルードファイルを1回だけ読み出す (ONCE)	278
18.1.3.	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 (GENALL)	278
18.2.	プリコンパイル結果のファイル出力	279
18.2.1.	インクルードファイルの展開/非展開	279
18.2.2.	条件コンパイルの真偽値の表示	280
18.3.	構造体/列挙体 (定数)	281
18.3.1.	プリコンパイル オプション	281
18.3.2.	プリコンパイル結果	281
18.3.3.	通知コード (イベント識別コード)	281
18.3.4.	エラーコード	282
18.4.	プロパティ	283
18.5.	メソッド	284
18.5.1.	ベースパスの設定 (SetBasePath)	284
18.5.2.	オプションシンボル群の設定 (SetOptSy)	284

18. 5. 3.	インクルードパス群の設定 (SetIncPath)	284
18. 5. 4.	プリコンパイルの実行 (PreCompile)	285
18. 5. 5.	プリコンパイルの中止 (Stop)	285
18. 5. 6.	インスタンスの消去 (Delete)	285
18. 5. 7.	コールバックメソッドの設定 (SetCbNtcXXXX)	286
18. 6.	イベント/デリゲート	287
18. 6. 1.	いずれかのイベント発生通知 (OnNtcAnyEvt)	287
18. 6. 2.	現在処理中のファイル名, 行番号通知 (OnNtcFileLno)	287
18. 6. 3.	インクルードファイル検索開始通知 (OnNtcSrhStart)	287
18. 6. 4.	インクルードファイル検索ディレクトリ通知 (OnNtcSrhDir)	288
18. 6. 5.	インクルードファイル検索終了通知 (OnNtcSrhEnd)	288
18. 6. 6.	条件コンパイル用オプションシンボル通知 (OnNtcOptSym)	288
18. 6. 7.	マクロ定義通知 (OnNtcMacDef)	288
18. 6. 8.	マクロ参照通知 (OnNtcMacRef)	289
18. 6. 9.	ファイル出力中通知 (OnNtcOutput)	289
18. 6. 10.	エラー通知 (OnNtcError)	289
18. 6. 11.	ソースファイルのテキストエンコード通知 (OnNtcSrcTec)	290
18. 7.	サンプルプログラム	291
18. 7. 1.	Sil_CPrePro (G U I によるプリコンパイル)	291
18. 7. 2.	Sil_CPreProC (コンソールアプリでプリコンパイル)	298
19.	スタティック ファンクション (CAjrStatic.dll)	301
19. 1.	構造体/列挙体 (定数)	302
19. 1. 1.	言語種別	302
19. 2.	プロパティ	302
20.	タピック: ツールチップ (CAjrStatic.dll, SAjrTip クラス)	303
20. 1.	エスケープシーケンス	304
20. 2.	メソッド	305
20. 2. 1.	デフォルト・テキスト表示色 設定/取得 ({Set Get}DefTextColor)	305
20. 2. 2.	デフォルト外枠表示色 設定/取得 ({Set Get}DefBorderColor)	305
20. 2. 3.	デフォルトウインド背景表示色 設定/取得 ({Set Get}DefBackGround)	306
20. 2. 4.	デフォルト・フォント 設定/取得 ({Set Get}DefFont)	306
20. 2. 5.	デフォルト・表示遅延時間 設定/取得 ({Set Get}DefDelayTime)	306
20. 2. 6.	デフォルト・表示時間 設定/取得 ({Set Get}DefDisplayTime)	306
20. 2. 7.	コントロール間移動猶予時間 設定/取得 ({Set Get}WindowTime)	306
20. 2. 8.	パレット色の設定/取得 ({Set Get}Palette)	307
20. 2. 9.	全コントロールでの、非アクティブ時ツールチップ表示設定/取得 ({Set Get}ShowForOnlyActive)	307
20. 2. 10.	全てのチップテキストの表示許可/禁止 設定/取得 ({Set Get}EnableAll)	307
20. 2. 11.	ツールチップの関連付け追加 (Add)	308
20. 2. 12.	ツールチップの表示条件設定/取得 ({Set Get}ShowAlways)	308
20. 2. 13.	ツールチップの関連付け解除 (Remove)	308
20. 2. 14.	コールバック設定 (SetCallBack)	309
20. 2. 15.	ツールチップ のウインドサイズ取得 (GetSize)	310
20. 2. 16.	ツールチップの表示 (Show)	310
20. 2. 17.	ツールチップの非表示 (Hide)	310
20. 2. 18.	コントロールの中央にツールチップ表示 (ShowCenter)	311
20. 2. 19.	マウスカーソルをツールチップ上へ移動 (MoveCursor)	311
20. 2. 20.	パレット色の設定 (SetPalette)	311
20. 2. 21.	ビットマップ登録 (RegistBitmap)	312
20. 2. 22.	アイコン登録 (RegistIcon)	312
20. 2. 23.	イメージ登録解除 (UnregistImage)	312
20. 3.	サンプルプログラム	313
20. 3. 1.	Sil_TipText1	313
21.	スタティック: プロファイル/レジストリアクセス (CAjrStatic.dll, SAjrReg クラス)	317
21. 1.	構造体/列挙体 (定数)	319
21. 1. 1.	レジストリトップキー (レジストリハイブ)	319
21. 1. 2.	プロファイル記録先	319
21. 1. 3.	レジストリ書き込みモード	319
21. 1. 4.	レジストリタイプ	319

21.1.5.	全コントロール設定値のセーブ／ロードオプション	320
21.2.	メソッド	321
21.2.1.	プロファイル記録先 設定／取得 (SetProfileDev / GetProfileDev)	322
21.2.2.	レジストリ記録モード 設定／取得 (SetRegistryMode / GetRegistryMode)	322
21.2.3.	レジストリトップキー 設定／取得 (SetRegTopKey / GetRegTopKey)	322
21.2.4.	レジストリ・ルートパス 設定／取得 (SetRegRootPath / GetRegRootPath)	322
21.2.5.	レジストリ・ミドルパス 設定／取得 (SetRegMidPath / GetRegMidPath)	323
21.2.6.	.INI ファイルパス 設定／取得 (SetIniFilePath / GetIniFilePath)	323
21.2.7.	レジストリ／INI ファイルのフルパス取得 (GetProfilePath)	323
21.2.8.	プロファイルの読み出し (Read / ReadHex / ReadH64)	324
21.2.9.	プロファイルの書き込み (Write / WriteHex / WriteH64)	324
21.2.10.	プロファイル・セクション削除 (DelSect)	324
21.2.11.	プロファイル・キー削除 (DelKey)	324
21.2.12.	プロファイル・セクション消去／クリーンアップ (RemoveSect / CleanupSect)	325
21.2.13.	ウインド位置を永続化 ({Save/Load}WndPos)	325
21.2.14.	ウインドの位置とサイズを永続化 ({Save/Load}WndRect)	325
21.2.15.	フォーム内 ListBox/CheckedListBox の設定値を永続化 ({Save Load}ListBox)	326
21.2.16.	フォーム内 ComboBox の設定値を永続化 ({Save Load}ComboBox)	326
21.2.17.	フォーム内全コントロールの設定値を永続化 ({Save Load}AllCtrls)	327
21.2.18.	レジストリ中の環境変数の読み出し (GetUserEnv / GetSysEnv)	328
21.2.19.	レジストリ中の環境変数へ書き込み (RegPutUserEnv / RegPutSysEnv)	328
21.2.20.	レジストリ中の環境変数を消去 (RegDelUserEnv / RegDelSysEnv)	328
21.2.21.	レジストリ中の環境変数へ項目を追加 (AddUserEnvItem / AddSysEnvItem)	329
21.2.22.	レジストリ中の環境変数から項目を削除 (DelUserEnvItem / DelSysEnvItem)	329
21.2.23.	環境変数の有効化 (AjrRegEnableEnvironment)	329
22.	スタティク：ファイル／フォルダ操作 (CAjrStatic.dll, SAjrFop クラス)	330
22.1.	メソッド	330
22.1.1.	フォルダ構造のコピー (CopyFolderStruct)	331
22.1.2.	ファイル群のコピー (CopyFiles)	333
22.1.3.	フォルダ削除 (RemoveFolder)	334
22.1.4.	フォルダのクリーンアップ (CleanFolder)	335
22.1.5.	2つのフォルダ下のファイル群・突き合せリスト列挙 (EnumFileMatchingList)	336
22.1.6.	パスがワイルドカード[群]に一致するかチェック (PathMatchSpecs)	340
22.1.7.	パスが文字列[群]を含むかチェックする (PathMatchStrings)	340
22.1.8.	パスが存在するかチェック (IsExistsPath)	340
22.1.9.	パスがディレクトリかチェック (IsPathDirectory)	340
22.1.10.	パスがファイルかチェック (IsPathFile)	341
22.1.11.	ファイルサイズ取得 (AjrGetFileSize)	341
22.1.12.	ファイルタイム取得 (AjrGetFileTime1970)	341
22.1.13.	ファイル比較 (FileCompare)	341
22.2.	サンプルプログラム	342
22.2.1.	Si1_FileDir (ファイル／フォルダ操作)	342
23.	スタティク：3D／2Dベクトル等の演算 (CAjrStatic.dll, SAjrMath クラス)	346
23.1.	メソッド	346
23.1.1.	正弦 (Sin)	347
23.1.2.	双曲線・正弦 (Sinh)	347
23.1.3.	逆正弦 (ASin)	347
23.1.4.	余弦 (Cos)	347
23.1.5.	双曲線・余弦 (Cosh)	347
23.1.6.	逆余弦 (ACos)	348
23.1.7.	正接 (Tan)	348
23.1.8.	双曲線・正接 (Tanh)	348
23.1.9.	逆正接 (ATan)	348
23.1.10.	逆正接 (ATan2)	348
23.1.11.	3Dベクトル加算 (V3dAdd)	349
23.1.12.	3Dベクトル減算 (V3dSub)	349
23.1.13.	3Dベクトル乗算 (V3dMult)	349
23.1.14.	3Dベクトルの除算 (V3dDiv)	349
23.1.15.	3D・線ベクトル設定 (V3dSetLineVec)	349

23.1.16.	3D・線分情報設定(AjcV3dSetLinePoint)	350
23.1.17.	3D・三角形情報設定(AjcV3dSetTriPoint)	350
23.1.18.	3D・ベクトルの長さ算出(V3dLength)	350
23.1.19.	3D・線分の中点算出(V3dVecCenter)	350
23.1.20.	3D・ベクトルの外積算出(V3dOuter)	350
23.1.21.	3D・ベクトルの内積算出(V3dInner)	351
23.1.22.	3D・三角形の法線ベクトル算出(V3dPlaneVec)	351
23.1.23.	3D・単位ベクトル算出(V3dNormal)	351
23.1.24.	3D・ベクトルの開き角度算出(V3dTheta)	351
23.1.25.	3D・ポイントから直線へ直行する方向ベクトル算出(V3dVertVecP2L)	351
23.1.26.	3D・ポイントから直線までの方向ベクトルと距離算出(V3dDistP2L)	352
23.1.27.	3D・2つの3Dポイント間の距離算出(V3dDistP2P)	352
23.1.28.	3D・ラインの交点算出(V3dCrossL2L)	352
23.1.29.	3D・点と平面の交点算出(V3dCrossP2F)	352
23.1.30.	3D・同一平面上で線分に直行するベクトル算出(V3dOrthoVecOnPlane)	353
23.1.31.	3D・ベクトルと行列の掛け算(V3dMultMat)	353
23.1.32.	3D・ベクトルをX軸周りに回転(V3dRotateX)	353
23.1.33.	3D・ベクトルをY軸周りに回転(V3dRotateY)	353
23.1.34.	3D・ベクトルをZ軸周りに回転(V3dRotateZ)	353
23.1.35.	3D・ベクトルを任意の軸周りに回転(V3dRotateAny)	354
23.1.36.	3D・任意の直行ベクトル算出(V3dAnyOrthoVec)	354
23.1.37.	3D・任意の点を平面上で指定角度回転した点を求める(V3dRotateOnPlane)	354
23.1.38.	3D・任意の3点を通る円の中心と半径を算出(V3dCalcCircle)	355
23.1.39.	3D・球面上の任意の4点から球の中心と半径を算出(V3dCalcSphere [V])	356
23.1.40.	2Dベクトル加算(V2dAdd)	357
23.1.41.	2Dベクトル減算(V2dSub)	357
23.1.42.	2Dベクトル乗算(V2dMult)	357
23.1.43.	2Dベクトルの除算(V2dDiv)	357
23.1.44.	2D・ベクトルの長さ算出(V2dLength)	357
23.1.45.	2D・ベクトルの外積算出(V2dOuter)	358
23.1.46.	2D・ベクトルの内積算出(V2dInner)	358
23.1.47.	2D・単位ベクトル算出(V2dNormal)	358
23.1.48.	2D・ベクトルの開き角度算出(V2dTheta)	358
23.1.49.	2D・ポイントから直線へ直行する方向ベクトル算出(V2dVertVecP2L)	358
23.1.50.	2D・ポイントから直線までの方向ベクトルと距離算出(V2dDistP2L)	359
23.1.51.	2D・2つの2Dポイント間の距離算出(V2dDistP2P)	359
23.1.52.	2D・ラインの交点算出(V2dCrossL2L)	359
23.1.53.	2D・ベクトルを回転(V2dRotate)	359
23.1.54.	2D・任意の直行ベクトル算出(V2dAnyOrthoVec)	359
24.	スタティック：汎用ファンクション (CAjrStatic.dll, SAjrGsr クラス)	360
24.1.	メソッド	360
24.1.1.	バージョン文字列取得(GetVersion)	361
24.1.2.	言語種別 設定／取得({Set/Get}Lang)	361
24.1.3.	言語によるテキストの選択(LangSel)	361
24.1.4.	保存ファイル名取得 (GetSaveFile)	362
24.1.5.	オープンファイル名取得 (GetOpenFile)	363
24.1.6.	複数のオープンファイル名取得 (GetOpenFiles)	364
24.1.7.	フォルダ名取得 (GetFolder)	365
24.1.8.	起動したプログラムのフォルダ・パス名取得 (GetAppPath)	365
24.1.9.	パス名の連結 (PathCat)	365
24.1.10.	接頭語に続くパラメタ取得 (AfterPrefix)	366
24.1.11.	フォーム内の全コントロールのイネーブル(EnableAllControls)	366
24.1.12.	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イネーブル(EnableGroup)	366
24.1.13.	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表示／非表示(ShowGroup)	366
24.1.14.	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 (MoveGroup)	367
24.1.15.	2つの角度値の変移角を求める (DifferenceOfTheta)	367
24.1.16.	Windows のシャットダウンやリブートを行う (ExitWindows)	367
24.1.17.	1970/1/1 00:00:00 からの通算秒を日時に変換 (Time1970ToDateAndTime)	368
24.1.18.	日時を 1970/1/1 00:00:00 からの通算秒に変換 (DateAndTimeToTime1970)	368

24.1.19.	UTC日時をローカルタイムに変換 (UtcTimeToLocalTime).....	368
24.1.20.	10進数文字列の整数部で規定桁数毎に区切り文字を挿入する (SepDecimal).....	368
24.1.21.	10進数文字列の整数部から区切り文字を削除する (RmvSepChar).....	369
24.1.22.	テキストボックスにフォルダやファイルをドロップ可能にする (EnableToDrop).....	369
24.1.23.	デスクトップ上のウインド位置取得 (GetWindowPos).....	370
25.	スタティク : チェックサム (CAjrStatic.dll, SAjrCS クラス).....	371
25.1.	メソッド.....	371
25.1.1.	ストリームのバイト/ワードサム算出 (CalcByteSum[N S], CalcWordSum[N S]).....	372
25.1.2.	ストリームのバイト/ワードサム設定 (SetByteSum[N S], SetWordSum[N S]).....	373
25.1.3.	ストリームのバイト/ワードサム検査 (ChkByteSum[N S], ChkWordSum[N S]).....	374
26.	スタティク : CRC 計算 (CAjrStatic.dll, SAjrCrc クラス).....	375
26.1.	メソッド.....	375
26.1.1.	部分CRC算出 (PartCrc??L / PartCrc??R).....	376
26.1.2.	CRC算出 (BlkCrc??L / BlkCrc??R).....	376
26.1.3.	XMODEM-CRC/FCS算出 (BlkXMODEM / BlkCalcFCS).....	377
26.1.3.1.	ストリームへCRC値設定 (SetCrc??...).....	377
26.1.4.	ストリームへXMODEM / FCS 設定 (SetXMODEM... / SetFCS...).....	378
26.1.5.	ストリームのCRC値検査 (ChkCrc??...).....	379
26.1.6.	ストリームのXMODEM / FCS 検査 (ChkXMODEM... / ChkFCS...).....	380
27.	スタティク : ネイティブデータ・アクセス (CAjrStatic.dll, SAjrBin クラス).....	381
27.1.1.	指定アドレスのメモリ間でメモリコピー (MemCopy).....	381
27.1.2.	文字列とバッファ間で文字列のコピー (StrCopy).....	381
27.1.3.	文字列の文字数取得 (StrLen).....	381
28.	免責事項.....	382
29.	問い合わせ先.....	382

1. 概 要

本ライブラリは、WindowsPC の.NETFramework 用 カスタムコントロール群のライブラリです。
Microsoft・VisualStudio (C# やVB.NET 等) で作成したプログラムからの利用を前提としています。

本ライブラリは、AjrCst32.dll/AjrCst64.dll のラッパープログラムであり、実際の処理のほとんどは AjrCst32.dll/AjrCst64.dll で処理されています。

本ライブラリはソースプログラム (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/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)

本ライブラリを使用する場合「.NETFramework Ver4.0 以降」がインストールされていなければなりません。

本ライブラリは、以下のファイルで構成されます。

#	ファイル名	内容	記号名
1	CAjr3DGraphic.dll	2D / 3D グラフィック表示コントロール	g3d
2	CAjrBarGraph.dll	棒グラフ / 折れ線グラフ表示コントロール	bar
3	CAjrCPrePro.dll	C 言語のプリコンパイル	cpp
4	CAjrCToken.dll	C 言語の字句分解	ctk
5	CAjrCustCtrl.dll	共通定義モジュール	—
6	CAjrFileSearch.dll	ファイル検索	fsr
7	CAjrInit.dll	初期設定用内部ファイル (ユーザが直接操作することはありません)	—
8	CAjrInpValue.dll	数値入力コントロール	inp
9	CAjrSerialComPort.dll	シリアル通信コントロール	scp
10	CAjrSockClient.dll	ソケット (TCP/IP) クライアントコントロール	ssv
11	CAjrSockServer.dll	ソケット (TCP/IP) サーバコントロール	ssv
12	CAjrSphereData.dll	サンプルデータ生成コントロール	spd
13	CAjrStatic	汎用スタティックファンクション (プロファイルアクセス, コンソール・・・)	sta
14	CAjrStrPool.dll	文字列プールコントロール	spl
15	CAjeTextFile.dll	テキストファイルアクセスコントロール (テキストエンコード変換機能付き)	txf
16	CAjrTimeChart.dll	タイムチャートグラフ表示コントロール	tch
17	CAjrVT100.dll	VT100 エミュレーションウインド コントロール	vth
18	AjrCst32.dll	ライブラリ処理本体 (32 bit ネイティブコード)	—
19	AjrCst64.dll	ライブラリ処理本体 (64 bit ネイティブコード)	—

「記号名」は、説明文中でプロパティやメソッドを示す場合に、そのプロパティやメソッドがどのコントロールに属するのかが示します。
例えば、「tch.Stop()」は、タイムチャートグラフ表示コントロールの Stop メソッドを意味します。

1.1. 実行可能な Windows／VisualStudio／.NETFramework バージョン

本ライブラリは、Windows10 / Windows11 における 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) の .NETFramework4.0 以降で動作可能です。

1.2. 構造体や列挙体（定数）の参照

構造体や列挙体は、「namespace AjrCustCtrl」の中に定義されています。

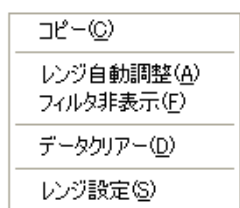
参照する場合は、先頭で「using AjrCustCtrl;」を宣言し「<構造体や列挙体名>[.<メンバ名>]」のように記述するか、あるいは、「AjrCustCtrl.<構造体や列挙体名>[.<メンバ名>]」のようにフルネームで記述します。

(例)	using AjrCustCtrl;	
	ExitWindows(EExitWindows.EWX_SHUTDOWN);	ExitWindows(AjrCustCtrl.EExitWindows.EWX_SHUTDOWN);
	AJC3DVEC v;	AjrCustCtrl.AJC3DVEC v;

1.3. 言語設定

本コントロール群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、日本語 Windows 以外の 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、以下のメソッドを実行します。

SAjrGsr.SetLang(ELangId. Japanese); --- 日本語設定

SAjrGsr.SetLang(ELangId. English); --- 英語設定

または、レジストリキー（HKEY_CURRENT_USER 下の「¥Software¥AjrCstXX¥General」の「Lang」値を設定します。

（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”を設定すると P C の環境に従います。

1.4. サンプルプログラム

本コントロールは、.NET Framework 用のクラスライブラリなので、.Net Framework のアセンブリであれば何からでも使用可能ですが、本書では、C#のサンプルプログラムを掲載しています。

1.5. 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 D L L を同梱してください。

1.6. 参照の設定

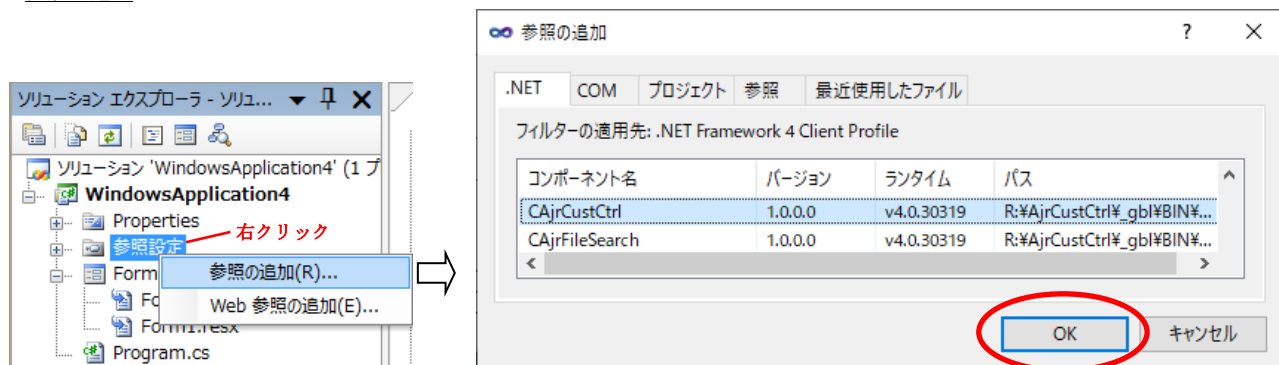
本ライブラリを使用するには、ソースプログラムで「using AjrCustCtrl;」を記述します。

また、コンソールアプリケーション等の場合（フォームにツールアイテムをドラッグしない場合）、VisualStudio でソリューションエクスプローラの「参照設定」を右クリックし、「参照の追加」から「AjrCustCtrl」と必要なコントロールを追加してください。

using の記述

```
using AjrCustCtrl;
```

参照の追加



コンソールアプリで本ライブラリを使用する場合は、必要なコントロールの参照を追加した上で、オブジェクトの生成を行ってください。

```
namespace MyConsoleApplication
{
    class Program
    {
        static CAjrTextFile txf = new CAjrTextFile();
        static CAjrCToken   ctk = new CAjrCToken();
        static CAjrStatic   sta = new CAjrStatic();
        static CtkCbKGetS   m_CtkCbKGetS;
    }
}
```

使用するコントロールの
オブジェクト生成

1.7. コントロール(DLL)のバージョンを特定しない

バージョンを更新した本ライブラリをインストールした場合、コンパイル時に以下のようなメッセージが出力される場合があります。

```
・・・この参照を解決できませんでした。アセンブリ "AjrCustCtrl, Version=n.n.n.n, ・・・
```

このエラーを解決するには、参照設定している本ライブラリ項目について「特定バージョン」プロパティを False に設定します。

※「AjrCustCtrl.dll」の「特定バージョン」を False に設定した例を右図に示します。

その他「CAjr」で始まる項目は全て「False」に設定します。



1.8. ツールボックスアイテムの追加

本コントロール群を VisualStudio のフォームデザイナーで使用するためには、コントロールをツールボックスに追加する必要があります。コントロールをツールボックスに追加する方法については「AjrCstInstall.pdf（インストール手順）」を参照してください。

2. 共通定義 (AjrCustCtrl.dll)

複数のコントロールで参照している、共通の定義モジュールです。

このモジュールは、常に参照設定するようにしてください。(VisualStudio IDE でソリューションエクスプローラの「参照設定」を右クリックし、「参照の追加」から「AjrCustCtrl」を追加してください)

2.1. 構造体/列挙体 (定数)

以下の構造体, 列挙体は、AjrCustCtrl.dll の「namespace AjrCustCtrl」で定義されています。

2.1.1. 2Dベクトル

2次元の座標位置や方向ベクトルを示す構造体です。

```
public struct AJC2DVEC {
    public AJC2DVEC(double x, double y)
    {
        this.x = x;
        this.y = y;
    }
    public double x, y;
}
```

2.1.2. 2D線ベクトル (始点と方向ベクトル)

2次元の線分を示す構造体です。(p:始点, v:方向ベクトル)

```
public struct AJC2DLVEC {
    public AJC2DLVEC(AJC2DVEC p, AJC2DVEC v)
    {
        this.p.x = p.x; this.v.x = v.x;
        this.p.y = p.y; this.v.y = v.y;
    }
    public AJC2DVEC p, v;
}
```

2.1.3. 3Dベクトル

3次元の座標位置や方向ベクトルを示す構造体です。

```
public struct AJC3DVEC
{
    public AJC3DVEC(double x, double y, double z)
    {
        this.x = x;
        this.y = y;
        this.z = z;
    }
    public double x, y, z;
}
```

2.1.4. 3D線ベクトル (始点と方向ベクトル)

3次元の始点位置と、方向を示す構造体です。(p:始点, v:方向ベクトル)

```
public struct AJC3DLVEC
{
    public AJC3DLVEC(AJC3DVEC p, AJC3DVEC v)
    {
        this.p.x = p.x; this.v.x = v.x;
        this.p.y = p.y; this.v.y = v.y;
        this.p.z = p.z; this.v.z = v.z;
    }
    public AJC3DVEC p, v;    // p : 始点, v : 方向
}
```

2.1.5. 3Dラインポイント (始点と終点)

3次元の線分を示す構造体です。

```
public struct AJC3DLINE
{
    public AJC3DLINE(AJC3DVEC p1, AJC3DVEC p2)
    {
        this.p1.x = p1.x;    this.p2.x = p2.x;
        this.p1.y = p1.y;    this.p2.y = p2.y;
        this.p1.z = p1.z;    this.p2.z = p2.z;
    }
    public AJC3DVEC p1, p2;
}
```

2.1.6. 3D三角形情報 (三角形の頂点)

3次元の三角形の頂点を示す構造体です。

```
public struct AJC3DTRI
{
    public AJC3DTRI(AJC3DVEC p1, AJC3DVEC p2, AJC3DVEC p3)
    {
        this.p1.x = p1.x;    this.p1.y = p1.y;    this.p1.z = p1.z;
        this.p2.x = p2.x;    this.p2.y = p2.y;    this.p2.z = p2.z;
        this.p3.x = p3.x;    this.p3.y = p3.y;    this.p3.z = p3.z;
    }
    public AJC3DVEC p1, p2, p3;
}
```

2.1.7. 3D行列 (3×3の行列)

3×3の行列を示す構造体です。

```
public struct AJC3DMAT
{
    public AJC3DMAT(double ini)
    {
        this.m = new double[3,3];
        this.m[0,0] = ini; this.m[0,1] = ini; this.m[0,2] = ini;
        this.m[1,0] = ini; this.m[1,1] = ini; this.m[1,2] = ini;
        this.m[2,0] = ini; this.m[2,1] = ini; this.m[2,2] = ini;
    }
    public AJC3DMAT(double v00, double v01, double v02, double v10, double v11, double v12, double v20, double v21, double v22)
    {
        this.m = new double[3,3];
        this.m[0,0] = v00; this.m[0,1] = v01; this.m[0,2] = v02;
        this.m[1,0] = v10; this.m[1,1] = v11; this.m[1,2] = v12;
        this.m[2,0] = v20; this.m[2,1] = v21; this.m[2,2] = v22;
    }
    public double[,] m;
}
```

2.1.8. 3D／2D平面表示時の横軸／縦軸の種別 (3×3の行列)

```
public enum EAJCPLANEAXIS {
    XP,        // X軸昇順
    YP,        // Y軸  "
    ZP,        // Z軸  "
    XM,        // X軸降順
    YM,        // Y軸  "
    ZM,        // Z軸  "
}
```

2.1.9. テキストファイル エンコード

```
//----- テキストエンコード -----//
public enum ETextEncode : int
{
    TEC_MBC      = 0,      // マルチバイト
    TEC_UTF_8    = 1,      // UTF-8
    TEC_EUC_J    = 2,      // EUC (日本語)
    TEC_UTF_16LE = 3,      // UTF-16LE
    TEC_UTF_16BE = 4,      // UTF-16BE
    TEC_AUTO     = 9,      // AUTO
}

//----- BOM出力 (書き込み用) -----//
public enum EBomMode : int
{
    NOT_WRITE_BOM = 0,      // BOMを書き込まない
    WRITE_BOM     = 1,      // BOMを書き込む
}
```

2.1.10. テキストファイル 改行コード変換モード

```
//-----テキストファイル 改行コード変換モード -----//
public enum ETextLfConv : int
{
    NONE          = 0,      // 変換無し
    LF_TO_CRLF    = ,      // LF → CR, LF (ファイル書き込み時のデフォルト)
    LF_TO_CR      = ,      // LF → CR
    CR_TO_CRLF    = ,      // CR → CR, LF
    CR_TO_LF      = ,      // CR → LF
    CRLF_TO_LF    = ,      // CR, LF → LF (ファイル読み出し時のデフォルト)
    CRLF_TO_CR    = ,      // CR, LF → CR
    LF            = ,      // LFで改行, 変換無し
    CR            = ,      // CRで改行, 変換無し
    LF_CRSKIP     = ,      // LFで改行, CR除去
    CR_LFSKIP     = ,      // CRで改行, LF除去
}
```

2.1.11. ファイル属性

```
public enum EFileAtt : int
{
    _A_ARCH      = 0x20 ,    // アーカイブファイル
    _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
    _A_ALL       = 0x37 ,    // 上記全ファイル属性
}
```

3. テキスト表示拡張機能

ツールチップ等で指定するテキストに制御文字や、エスケープシーケンスを含めて、改行、文字色や太字の指定を行うことができます。ここで解説する制御文字や、エスケープシーケンスは以下の項目で指定するテキストにおいて有効です。

#	クラス	プロパティ／メソッド
1	CAjrTimeChart タイムチャート	ToolTipText ToolTipFilter0~7 TextOut () ツールチップテキスト フィルタ・ツールチップ テキスト描画
2	CAjr3DGraphic 2D／3Dグラフィック	ToolTipText ToolTipFilter00~15 TextOut () ツールチップテキスト フィルタ・ツールチップ テキスト描画
3	CAjrVT100 VT-100エミュレーション (※1)	ToolTipText ツールチップテキスト
4	CAjrInpValue 数値入力コントロール	ToolTipText ツールチップテキスト
5	CAjrBarGraph 棒グラフ／折れ線グラフ	ToolTipText ToolTipFilter0~7 ツールチップテキスト フィルタ・ツールチップ
6	SAjrTip スタティック：ツールチップ	Add () Show () ShowCenter () cbNeedText () ツールチップの関連付け追加 ツールチップの表示 ツールチップを中央に表示 状況依存チップ用コールバック

※1：ここで解説する制御文字や、エスケープシーケンスはCAjrVT100 (VT100 エミュレーション) のPutText ()や PutFormat () メソッドで指定するテキストには適用されません。
これらのメソッドで指定するテキストには、別仕様の制御文字やエスケープシーケンスが割り当てられていますので、「VT-100エミュレーション・ウインド コントロール (CAjrVT100.d11)」の章を参照してください。

3.1. 制御文字

指定可能な制御文字は以下の通りです

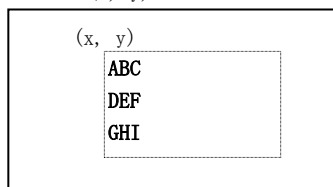
制御文字

#	制御文字	内容
1	'\x1B' 0x1B	エスケープシーケンスの始まり
2	'\t' 0x09	次のタブ位置までX位置を進める
3	'\r' 0x0D	テキスト描画スペースの左端までX位置を戻す
4	'\n' 0x0A	以下の復帰操作を行い、改行する (Y位置を文字の高さ+行間スペース分進める) ・先に'\r' があった場合は、テキスト描画スペースの左端までX位置を戻す (ワントタイム) ・その他の場合は、TextOut () 開始時のX位置へ戻す。

上記以外の制御文字 (0x00~0x1F, 0x7F の内で上記以外の制御文字) は、何もせずに読み飛ばします。

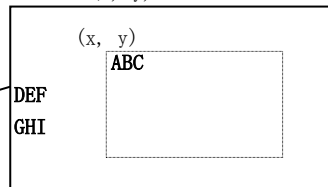
TextOut () メソッドの場合、復帰位置は、'\r' があるかないかで、以下のように異なるので、注意が必要です。

TextOut (x, y, "ABC\nDEF\rGHI\n");



"\n"は、TextOut () 開始時の位置(=x)へ復帰する

TextOut (x, y, "ABC\r\nDEF\r\rGHI\r\r\n");



"\r\n"は、テキスト描画スペースの左端(=0)へ復帰する

ツールチップテキストの表示では、内部で、x=0, y=0 で描画している為、"\n" と "\r\n" は同じ動作となります。

3.2. エスケープシーケンス

指定可能なエスケープシーケンスは以下の通りです

エスケープシーケンス		
#	エスケープシーケンス	内容
1	"¥x1B[0m"	文字色, 文字背景色とフォント (太字, 斜字) をリセット
2	"¥x1B[1m"	ボールド (太字) 設定
3	"¥x1B[3m"	イタリック (斜字) 設定
4	"¥x1B[9m"	文字色と文字背景色をリセット (文字色: パレット0, 背景: 透明)
5	"¥x1B[30m" ~ "¥x1B[37m"	文字色の設定 (3XのXはパレット番号 ※1)
6	"¥x1B[38;2;r;g;b" (※2)	文字色をRGBで指定
7	"¥x1B[39m"	文字色リセット (パレット0)
8	"¥x1B[40m" ~ "¥x1B[47m"	文字背景色の設定 (4XのXはパレット番号 ※1)
9	"¥x1B[48;2;r;g;b" (※2)	文字背景色をRGBで指定
10	"¥x1B[49m"	文字背景色リセット (透明)
11	"¥x1B[T"	ボールド (太字) 設定 ("¥x1B[1m"と同じ)
12	"¥x1B[t"	ボールド (太字) 解除
13	"¥x1B[I"	イタリック (斜字) 設定 ("¥x1B[3m"と同じ)
14	"¥x1B[i"	イタリック (斜字) 解除
15	"¥x1B[N"	ボールド (太字) と イタリック (斜字) 解除
16	"¥x1B[nL"	次の改行("¥n")で、行間スペースを、文字高さのn[%]とする。(ワントタイム)

※1: 各コントロールのTextOut()メソッドの場合は、チャートやグラフィックの描画色と同じ。

その他は、0: 黒, 1: 赤, 2: 緑, 3: 黄, 4: 青, 5: 紫, 6: 水色, 7: 白

※2: 各色の成分値は0~255 (各10進数で、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)

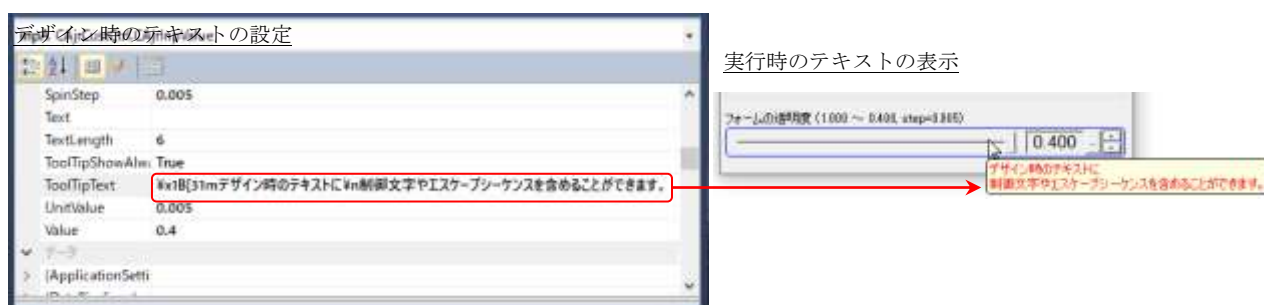
上記以外のエスケープシーケンスは、何もせずに読み飛ばします。

エスケープシーケンスは、"¥x1B" で始まり、英字(a-z / A-Z)で終了する文字列とします。

3.3. デザイン時のプロパティ (テキスト) の設定について

デザイン時に、ToolTipText や ToolTipFilter0~7 等のツールチップテキストのプロパティを設定する際は、直接制御文字を含めることができませんが、本ライブラリでは、デザイン時に設定したテキストを解析し、制御文字を認識します。

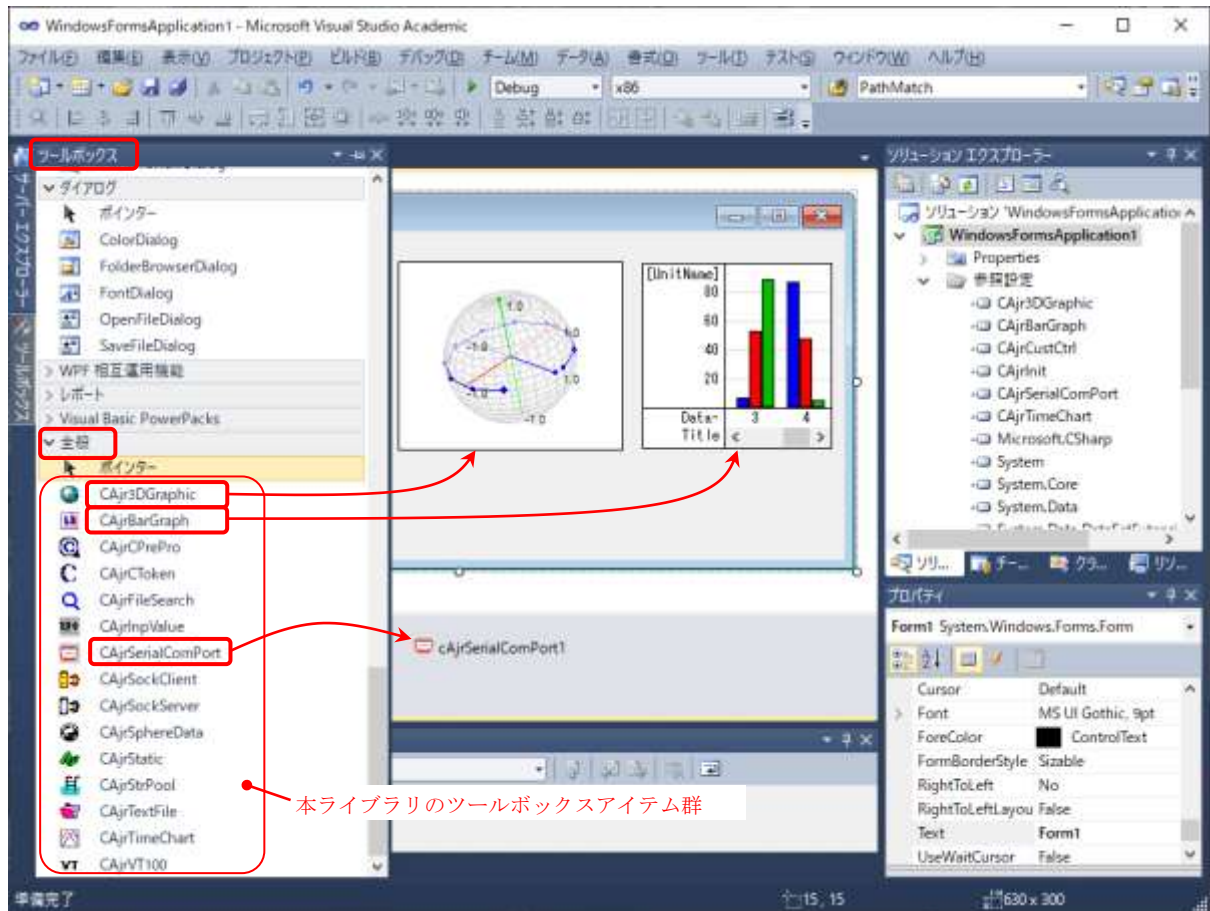
以下の例では、デザイン時のチップテキストに "¥x1B[31m" (文字色=赤) と、"¥n" (改行) を含めています。



制御文字は、C#やC言語と同じ記述 (¥x1B, ¥n, や ¥r 等) ですが、特例として "¥e" (=0x1B) を記述できます。

4. ツールボックス

ツールボックス・アイテムの登録を行うと、ツールボックス中の「全般」タブに以下のツールボックスアイテム群が表示されます。（ツールボックス・アイテムの登録については、インストール手順（AjrCstInstall.pdf / AjrCstParts.pdf）を参照してください）
ここで、各ツールボックスアイテムを選択（クリック）し、フォーム上にドラッグします。（アイテムをダブルクリックでも可）



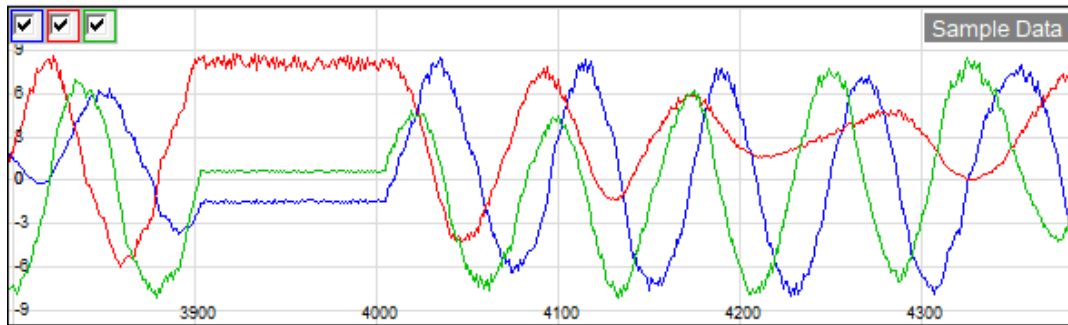
各ツールボックスアイテムの内容は、以下の通りです。

#	名称	内容
1	CAjr3DGraphic	2 D / 3 D グラフィック
2	CAjrBarGraph	棒グラフ / 折れ線グラフ
3	CAjrCPrePro	C 言語プリコンパイル
4	CAjrCToken	C 言語の字句分解
5	CAjrFileSearch	ファイル検索
6	CAjrInpValue	数値入力
7	CAjrSerialComPort	シリアル通信
8	CAjrSockClient	ソケット (TCP/IP) クライアント

#	名称	内容
9	CAjrSockServer	ソケット (TCP/IP) サーバ
10	CAjrSphereData	テストデータ生成
11	CAjrStatic	汎用スタティッククラス
12	CAjrStrPool	文字列プール
13	CAjrTextFile	テキストファイル・アクセス
14	CAjrTimeChart	タイムチャートグラフ表示
15	CAjrVT100	VT 1 0 0 エミュレーションウィンド

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

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、SetItemColor() メソッドで変更できます。

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。



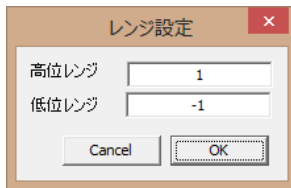
ストップ	: グラフ表示を停止します。次回は「スタート」メニューとなります。（※1）
一時停止	: グラフ表示を一時停止します。次回は「再開」メニューとなります。（※2）
コピー	: グラフ表示内容（ビットマップ）をクリップボードへコピーします。
レンジ設定	: グラフのレンジ（上限値, 下限値）を設定します。
レンジ自動調整	: プロットデータからグラフのレンジを自動算出して設定します。
オフセット設定	: プロットデータに加算するオフセット値を設定します。
その他の設定	: 平均化個数, タイムスケール幅, バッファに格納するデータ数を設定します。
フィルタ非表示	: コントロール左上のフィルタ（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 (AjrTchEnableMesDraw() 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。

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。



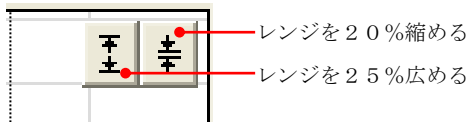
ここで、レンジ値を入力し、「OK」ボタンを押すと、グラフのレンジが設定されます。

「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

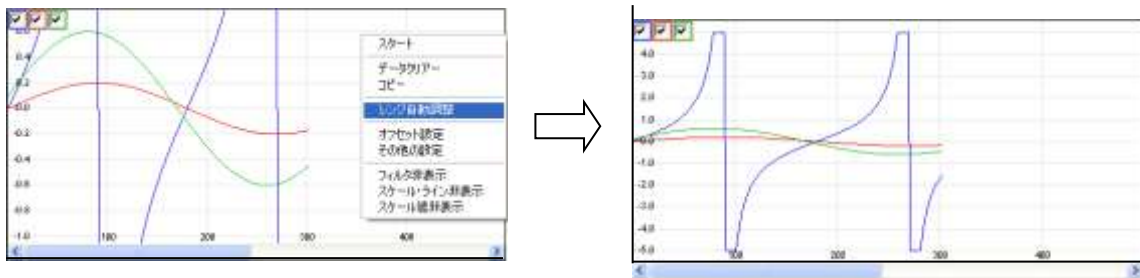
マウスカーソルをコントロールの右上隅に置くと、2つのボタンが表示されます。

これらのボタンで、レンジを30%広めたり、縮めたりすることができます。



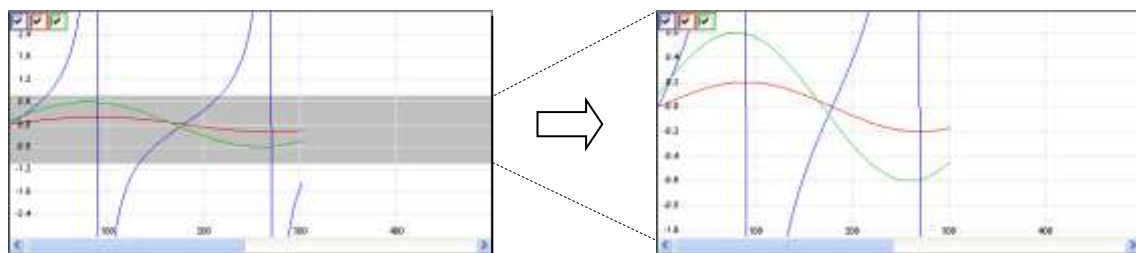
5.1.4. レンジ自動調整

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
（ドラッグされている部分はグレー表示されます）

CTRL キーを押したまま、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）

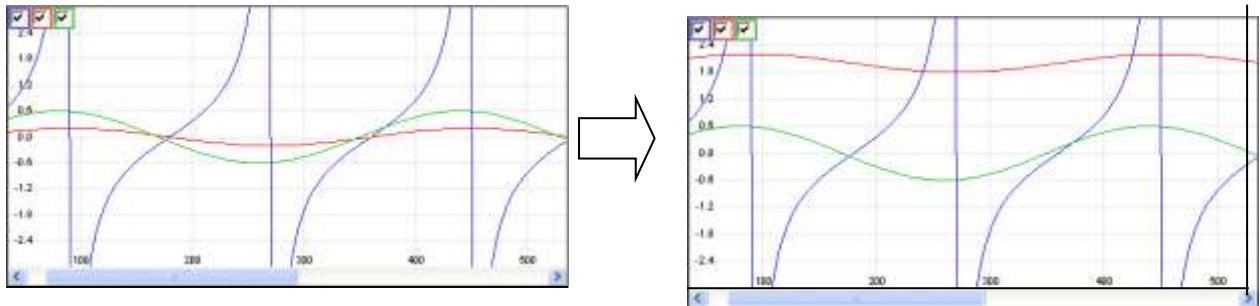
ここで、値を設定すると、当該データ項目のデータ値に、設定値を加算した値でグラフが表示されます。

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。

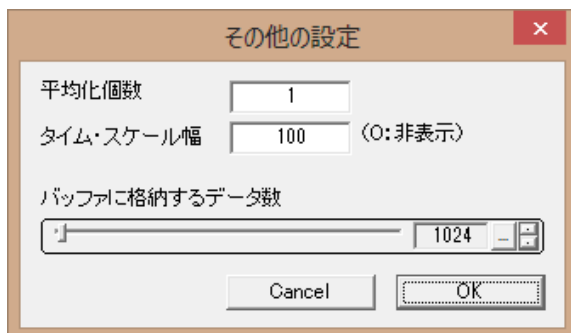
「リセット」ボタンを押すと、全てのオフセット値が「0」に設定されます。

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。

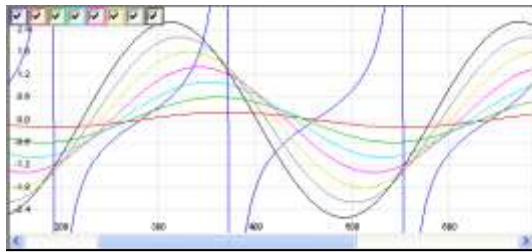
「OK」ボタンを押すと設定内容を確定します。「Cancel」ボタンを押すと設定を中止します。

5.1.8. フィルタの設定と表示／非表示

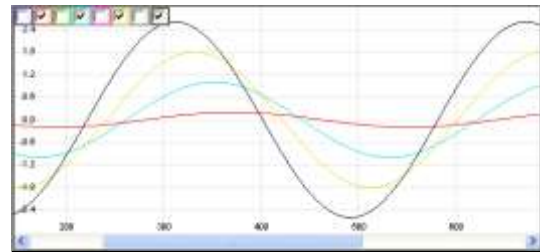
カーソルをウインドの左上隅に置くとチェックボックスが表示されます。

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



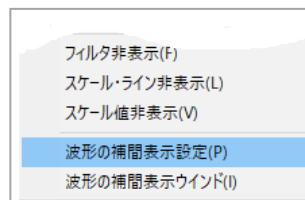
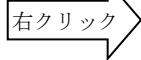
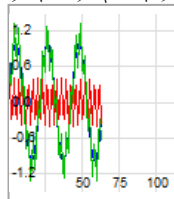
フィルタで、データ項目 0, 2, 4, 6 を非表示

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、プロパティ (IpKnd, IpNum, IpWidth) で設定するか、あるいは、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。

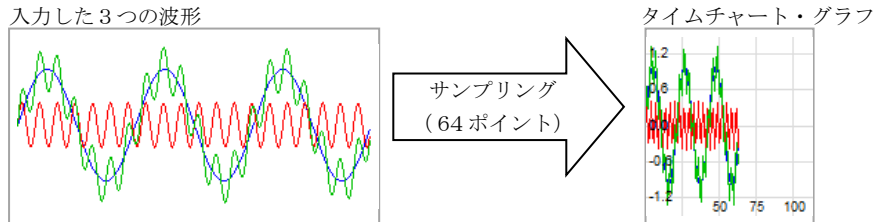
項目		内容	プロパティ
補間表示するデータ (補間表示対象とする データの選択)	表示されているデータ	タイムチャート・コントロールで表示しているデータ	IpKnd=Window
	全データ	バッファに格納されている全データ	IpKnd=All
	最新データ	最新のデータ群	IpKnd= Latest
データ個数		「最新データ」選択時の、データ個数	IpNum
補間幅		プロット点の表示間隔 (ピクセル数)	IpWidth

5.1.10. 波形の補間表示

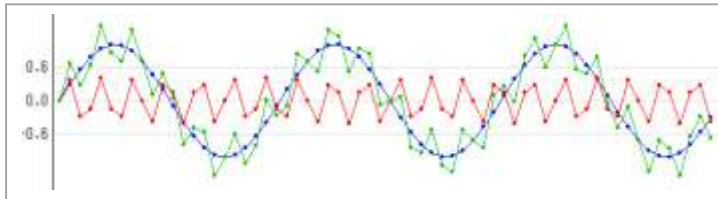
サンプル数が少なく、グラフに波形を正常に表示できない場合、サンプリングしたデータの間を補間することにより、本来の波形を再現して表示することができます。(3次スプライン曲線(サンプリング点を通る曲線)による補間表示)

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。

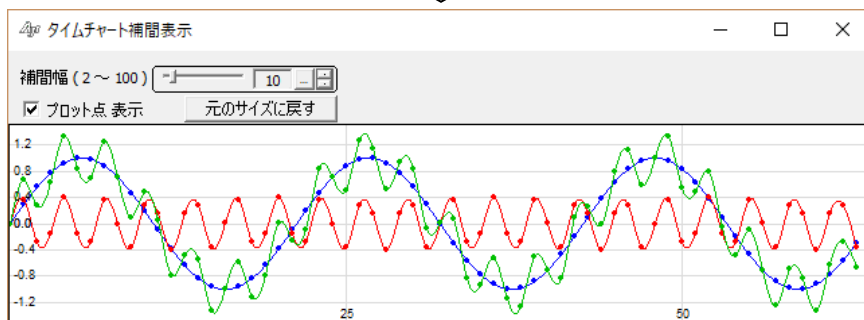
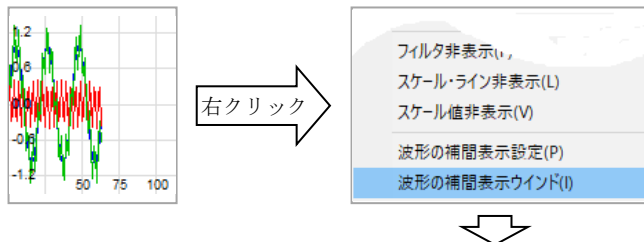


これでは、(タイムチャート・グラフから)元の波形を見ることはできません。そこで、グラフを横に引き伸ばして、プロット点の間を線で結んでみます。(下図)



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。

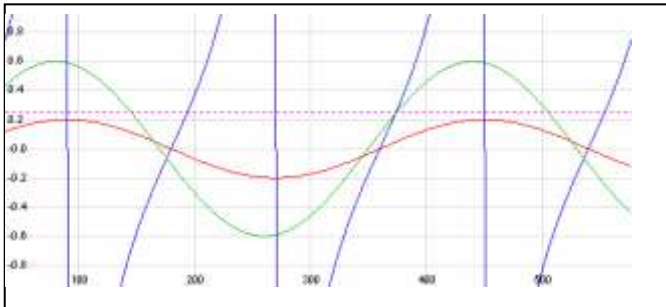
尚、補間表示対象データが少ない(8個未満)の場合は、補間表示できません。

5.1.11. 横線描画

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

- SetLineStyle - 横線の属性（線種，色，太さ）の設定
- SetLinePos - 横線の描画位置
- EnableHLine - 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。



```
tch.SetHoriLinePos (0, 0.25);
tch.SetHoriLineStyle(0, Color.Magenta, 1, TchLineStyle.Dot);
tch.EnableHoriLine (0, true);
```



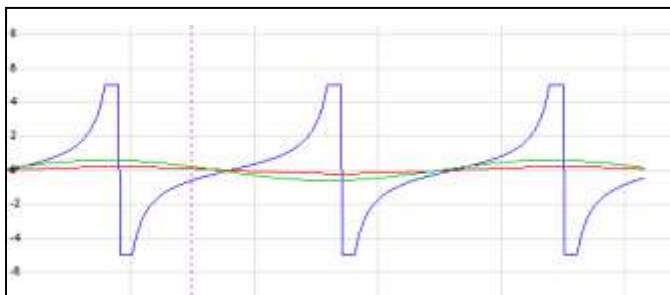
※ 点線等、実線以外の線を描画する場合は、線の太さ = 1 でなければなりません。

5.1.12. 縦線描画

最後に投与したデータの直後に縦の線を描画することができます。
縦線の描画は、PutRealData メソッドによりデータ投与後、以下のメソッドにより行ないます。

- SetVertLine - 縦線の描画
- EnableVertLine - 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。



```
tch.SetVertLine(Color.Magenta, 1,
                ETchLineStyle.Dot);
```



※ 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。

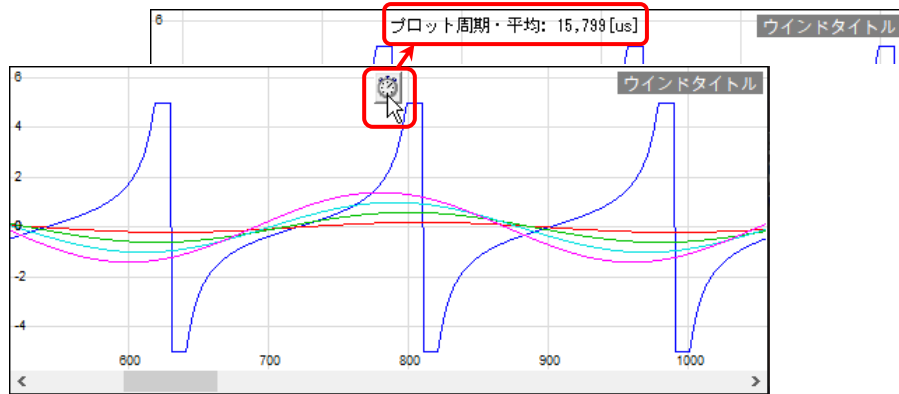
5.1.13. プロット周期表示

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、PutData() メソッド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。

プロット周期はマイクロ秒単位の値で 10 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、以下のメソッド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。

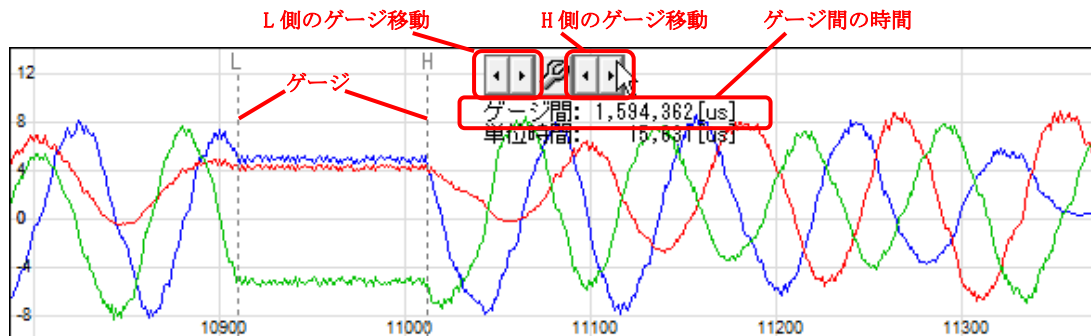
- MesPeriShow() - プロット周期の表示
- MesPeriReset() - プロット周期のリセット

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。

計測された単位時間を使用しない場合は「単位時間を指定する」をチェックし、独自の単位時間を指定します

単位時間を計測し直す

自動的に計測された単位時間

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。(但し、EnableMesDraw プロパティを true に設定要)



平均：1回のグラフィック描画に要する時間の平均[μ s]

最大：最大描画時間[μ s]

最小：最小描画時間[μ s]

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、描画時間だけではなく、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。

5.1.16. 右クリック

右クリック操作による動作は、以下のようになっています。

ポップアップメニュー許可 EnablePopupMenu プロパティ	SHIFT/CTRL キー押下	動作
True (許可)	× (未押下)	ポップアップメニューを表示
True (許可)	○ (押下)	右クリック通知イベント (OnRClick) 発生
False (禁止)	—	右クリック通知イベント (OnRClick) 発生

5.1.17. 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、以下のイベントを発生します。

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

これらのイベントでは、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 GetDroppedFile/GetDroppedDir メソッドにて取得します。

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、AcceptFiles プロパティを true に指定する必要があります。

5.1.18. 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、Pause() メソッド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。

```

        :
        :
        vth. Pause(true);           // 表示停止
        vth. PutData( . . . );      // データ投与
        vth. PutData( . . . );      } ① (この間投与したデータは表示されない)
        :
        :
        vth. Pause(false);         // 表示再開 (①の期間に描画したデータを一気に表示)
        :
        :
    
```

vth. Pause(true) 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)

vth. Pause(true)～vth. Pause(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。

※vth. Pause(true)～vth. Pause(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. 構造体／列挙体 (定数)

5.2.1. 線の属性

横線や縦線の描画属性の列挙体です。

```
public enum ETchLineStyle : int
{
    Solid      = 0,      // 実線
    Dash       = 1,      // -----
    Dot        = 2,      // .....
    DashDot    = 3,      // -.-.-.-
    DashDotDot = 4,      // -.~.-.-
    NullLine   = 5,      // 空線
    InsideFrame = 6      //
}
```

5.2.2. 波形補間表示種別 (補間対象データ)

波形の補間表示する際の、補間対象データです。

```
public enum ETchIpKind : int
{
    Window      = 0,      // ウインド表示データ
    All          = 1,      // 全データ
    Latest       = 2,      // 最新データ
}
```

5.3. プロパティ

タイムチャート・グラフ表示コントロールのプロパティ一覧を示します。

#	名称	タイプ	内容	デフォルト
1	AcceptFiles	bool	ファイル／フォルダのドロップ許可フラグ	false
2	AverageNumber	int	移動平均化個数 (1 以上)	1
3	DataItems	int	データ項目数 (1 ~ 8)	3
4	EnablePopupMenu	bool	ポップアップメニュー許可フラグ (※1)	true
5	FontTxo	Font	テキスト描画時のフォント	MS UI Gothic, 9
6	HoriScaleWidth	int	横スケール線の表示幅	100
7	IpKnd	int	波形補間表示種別 (補間表示対象データ) ・ Window - ウィンドに表示しているデータ ・ All - 全データ ・ Latest - 最新データ (個数を IpNum で指定)	Window
8	IpNum	int	最新データを補間表示する際の最大データ数	500
9	IpWidth	int	補間表示する際の補間幅	10
10	MaxBuf	int	最大バッファ容量 (細田データ数で指定)	4096
11	MaxLineDist	double	最大結線長 (※2)	0
12	RangeHigh	double	グラフ高位レンジ	1
13	RangeLow	double	グラフ低位レンジ	-1
14	ShowBorder	bool	コントロール外枠の表示フラグ	true
15	ShowDummyData	bool	デザイン時のダミーデータ表示フラグ (※3)	true
16	ShowFilter	bool	フィルタ (チェックボックス) 表示フラグ	true
17	ShowScaleLine	bool	スケールライン表示フラグ	true
18	ShowScaluValue	bool	スケール値表示フラグ	true
19	ToolTipFilter0~7	string	フィルタの各チェックボックスのツールヒント (※4)	""
20	ToolTipText	string	コントロールのツールヒント (※4)	""
20	ToolTipShowAlways	bool	ツールチップ表示条件 true - 非アクティブ状態でも表示 false - 非アクティブ状態時は非表示	true
21	EnableMesDraw	bool	ポップアップメニュー「描画時間情報 表示」の許可／禁止	false
22	TitleText	string	タイトルテキスト (文字色=白, 背景=グレーで右上に表示)	""

※

1 : 右クリックによるポップアップメニューの表示(true)／非表示(false)を設定します。

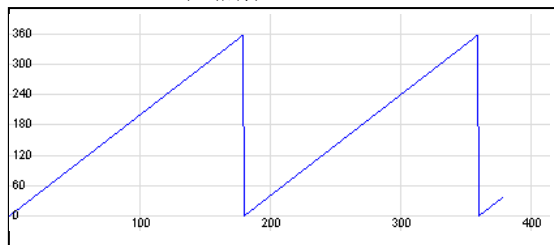
非表示(false)を設定した場合は、右クリックすると「OnRClick」イベントが発生します。

(Shift/Ctrl + 右クリックした場合は、EnablePopupMenu プロパティに関係なく常に「OnRClick」イベントが発生します)

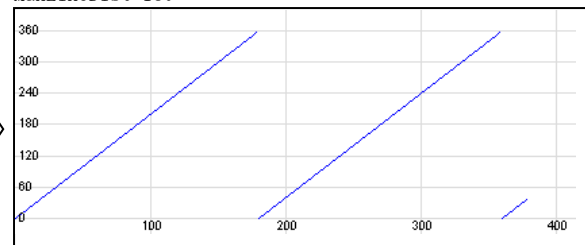
※2 : MaxLineDist プロパティは、2つのデータの差が当該プロパティより大きい場合は、結線しないように制御します。

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、結線してしまうと左図のように 359 → 0 度の変化を結線してしましますが、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



※3 : デザイン時にダミーデータを表示することにより、プロパティの効果を視覚的に確認することができます。

※4 : ToolTipText, ToolTipFilter0~7 プロパティは、制御文字とエスケープシーケンスを含めることができます。

これについては、「テキスト表示拡張機」の章を参照してください。

5.4. メソッド

タイムチャート・グラフ表示コントロールのメソッド一覧を以下に示します。

#	メソッド名	内容	備考
1	Stop	グラフ表示停止	
2	Start	グラフ表示開始	
3	PutData	データ投与	
4	SetItemOffset	データ項目のオフセット設定値設定	
5	GetItemOffset	データ項目のオフセット設定値取得	
6	SetItemColor	データ項目の表示色設定	
7	GetItemColor	データ項目の表示色取得	
8	AdjustRange	グラフレンジの自動調整	
9	SetScrollPos	スクロール位置の設定	
10	GetScrollPos	スクロール位置の取得	
11	SetFilter	フィルタの設定	
12	GetFilter	フィルタの設定値の取得	
13	SetHoriLineStyle	横線スタイル設定	
14	SetHoriLinePos	横線描画位置設定	
15	EnableHoriLine	横線表示／非表示	
16	SetVertLine	縦線描画	
17	EnableVertLine	縦線描画の表示／非表示	
18	GetDroppedFile	ドロップされたファイル名取得	
19	GetDroppedDir	ドロップされたディレクトリ名取得	
20	SetTitleText	タイトルテキスト表示	外面右上に表示
21	SaveToProfile	現在の設定値をプロファイルへ記録する	
22	LoadFromProfile	設定値をプロファイルから読み出す	
23	Pause	画面表示の停止／再開	
24	MesPeriShow	プロット周期計測値の表示	
25	MesPeriReset	プロット周期計測をリセット	
26	TextOut	テキスト描画 (ピクセル位置指定)	
27	PurgePlot	チャートデータ破棄	
28	PurgeText	テキスト描画データ破棄 (描画したテキストのクリアー)	
29	Purge	全てのデータ (チャートデータ, 描画テキスト) 破棄	

5.4.1. グラフ表示停止(Stop)

形 式 : void Stop();

引 数 : なし

説 明 : グラフの表示を停止します。
Start メソッドを実行するまでは、PutData() メソッドにより投与されたデータは無視 (破棄) されます。

戻り値 : なし

5.4.2. グラフ表示開始(Start)

形 式 : void Start();

引 数 : なし

説 明 : グラフの表示を開始します。
Stop メソッドにより表示を停止していた場合は、本メソッドにより表示を再開します。

戻り値 : なし

5.4.3. データ投与(PutData)

形 式 : void PutData(double d1);
void PutData(double d1, double d2);
void PutData(double d1, double d2, double d3);
void PutData(double d1, double d2, double d3, double d4);
void PutData(double d1, double d2, double d3, double d4, double d5);
void PutData(double d1, double d2, double d3, double d4, double d5, double d6);
void PutData(double d1, double d2, double d3, double d4, double d5, double d6, double d7);
void PutData(double d1, double d2, double d3, double d4, double d5, double d6, double d7, double d8);

引 数 : d1～d8 - グラフに投与するデータ

説 明 : データを投与し、グラフを更新します。
DataItems プロパティで指定したデータ項目数のデータを投与してください。

戻り値 : なし

5.4.4. データ項目のオフセット値設定(SetItemOffset)

形 式 : void SetItemOffset(int ix, double offset)

引 数 : ix - データ項目番号 (0～7)
offset - 設定するオフセット値

説 明 : 当該データ項目で、投与したデータ値に offset 値を加算してグラフ表示します。

戻り値 : なし

5.4.5. データ項目のオフセット値取得(GetItemOffset)

形 式 : double GetItemOffset(int ix)

引 数 : ix - データ項目番号 (0～7)

説 明 : 当該データ項目の設定されている offset 値を取得します。

戻り値 : 当該データ項目に設定されているオフセット値

5.4.6. データ項目の表示色設定(SetItemColor)

形 式 : void SetItemColor(int ix, Color color);

引 数 : ix - データ項目番号 (0～7)
Color - 表示色

説 明 : 当該データ項目の表示色を設定します。

戻り値 : なし

5.4.7. データ項目の表示色取得(GetItemColor)

形 式 : Color GetItemColor(int ix);

引 数 : ix - データ項目番号 (0～7)

説 明 : 当該データ項目の表示色を取得します。

戻り値 : 当該データ項目の表示色

5.4.8. グラフレンジの自動調整(AdjustRange)

形 式 : void AdjustRange ();

引 数 : なし

説 明 : フィルタで非表示となっているデータ項目を除く、すべ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、グラフのレンジを設定します。
つまり、すべてのデータがレンジ範囲内に入るように、レンジを設定します。
データがすべて同一値である場合は、当該データ値の -1～+1 でレンジ設定します。

戻り値 : なし

5.4.9. スクロール位置の設定 (SetScrollPos)

形 式 : void SetScrollPos (int pos);

引 数 : なし

説 明 : pos で指定された位置までスクロールして、グラフを表示します。
スクロール位置を設定する場合は、データの投与を停止している状態で行ってください。

戻り値 : なし

5.4.10. スクロール位置の取得 (SetScrollPos)

形 式 : int GetScrollPos ();

引 数 : なし

説 明 : 現在のスクロール位置 (グラフ左端データの最古データからの相対位置) を取得します。

戻り値 : 現在のスクロール位置

5.4.11. フィルタの設定(SetFilter)

形 式 : void SetFilter (int ix, bool fEnable);

引 数 : ix - データ項目番号 (0～7)
 fEnable - フィルタ設定値

説 明 : 当該データ項目のフィイルタ (チェックボックス) を設定します。
 fEnable=True を指定すると当該データ項目は表示され、fEnable=false を指定すると非表示となります。

戻り値 : なし

5.4.12. フィルタの設定値の取得(GetFilter)

形 式 : bool GetFilter (int ix);

引 数 : ix - データ項目番号 (0～7)
 fEnable - フィルタ設定値

説 明 : 当該データ項目のフィイルタ (チェックボックス) の設定値を取得します。

戻り値 : 当該データ項目のフィイルタ (チェックボックス) の設定値

5.4.13. 横線スタイル設定(SetHoriLineStyle)

形 式 : void SetHoriLineStyle (int ix, Color color, int width, ETchLineStyle style);

引 数 : ix - 横線データ識別 (0～7)
 color - 横線の表示色
 width - 線の幅 (1～, 点線等、実線以外のスタイルを指定する場合は1を指定すること)
 style - 横線の描画スタイル

説 明 : 横線の描画属性を指定します。
 グラフ上に描画できる横線は、最大8ヶであり、ix 引数で指定します。

戻り値 : なし

5.4.14. 横線描画位置設定(SetHoriLinePos)

形 式 : void SetHoriLinePos (int ix, double pos);

引 数 : ix - 横線データ識別 (0～7)
 pos - 横線の描画位置

説 明 : pos で指定した位置に横線を描画します。
 グラフ上に描画できる横線は、最大8ヶであり、ix 引数で指定します。

戻り値 : なし

5.4.15. 横線表示／非表示(EnableHoriLine)

形 式 : void EnableHoriLine (int ix, bool fEnable);

引 数 : ix - 横線データ識別 (0～7)
fEnable - 横線の表示／非表示フラグ

説 明 : 横線を表示 (true) あるいは、非表示 (false) します。
グラフ上に描画できる横線は、最大 8 本であり、ix 引数で指定します。

戻り値 : なし

5.4.16. 縦線描画(SetVertLine)

形 式 : void SetVertLine (Color color, int width, ETchLineStyle style);

引 数 : color - 縦線の表示色
width - 線の幅 (1～, 点線等、実線以外のスタイルを指定する場合は 1 を指定すること)
style - 縦線の描画スタイル

説 明 : 最後に投与したデータの位置に、縦の線を描画します。

戻り値 : なし

5.4.17. 縦線描画の表示／非表示(EnableVertLine)

形 式 : void EnableVertLine (bool fEnable);

引 数 : fEnable - 横線の表示／非表示フラグ

説 明 : 描画した縦線を表示 (fEnable =true) あるいは、非表示 (fEnable =false) します。

戻り値 : なし

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

形 式 : string GetDroppedFile();

引 数 : なし

説 明 : OnFileDrop イベント発生時に、順次、ドロップされたファイルのパス名を取得します。
OnFileDrop イベントで通知された、ドロップファイル数の回数だけ本メソッドをコールすることにより、ドロップされた全てのファイルを取得できます。

戻り値 : ≠"" : ドロップされたファイルパス名
="" : 終端(ドロップされたファイルパス名の取得は完了済である)

5.4.19. ドロップされたディレクトリ名取得 (GetDroppedDir)

形 式 : string GetDroppedDir();

引 数 : なし

説 明 : OnDirDrop イベント発生時に、順次、ドロップされたディレクトリのパス名を取得します。
OnDirDrop イベントで通知された、ドロップディレクトリ数の回数だけ本メソッドをこーすることにより、ドロップされた全てのディレクトリを取得できます。

戻り値 : ≠"" : ドロップされたディレクトリパス名
="" : 終端(ドロップされたディレクトリパス名の取得は完了済である)

5.4.20. タイトルテキスト表示 (SetTitleText)

形 式 : void SetTitleText(string TitleText);
 void SetTitleText(string TitleText, Color TextColor);
 void SetTitleText(string TitleText, Color TextColor, Color BackColor);
 void SetTitleText(string TitleText, Color TextColor, Color BackColor, Font TextFont);

引 数 : TitleText - タイトルテキスト (null(あるいは空文字列(""))を指定した場合は非表示)
 TextColor - テキスト描画色 (α 値は無視されます)
 BackColor - テキスト背景色 (α 値は無視されます)
 TextFont - テキストのフォント

説 明 : コントロールウインドの右上にタイトルテキストを表示します。
 TitleText に null(あるいは空文字列(""))を指定した場合、タイトルテキストは非表示となります。

戻り値 : なし

5.4.21. 現在の設定値をプロファイルへ記録する (SaveToProfile)

形 式 : void SaveToProfile (string section);

引 数 : section - プロファイル内のセクション名

説 明 : 現在の設定値 (フィルタ設定やプロパティ) をプロファイルに記録します。

戻り値 : なし

5.4.22. 設定値をプロファイルから読み出す (LoadFromProfile)

形 式 : void LoadFromProfile (string section);

引 数 : section - プロファイル内のセクション名

説 明 : SaveToProfile() によりプロファイルに記録した設定値を読み出します。
 読み出した内容は、そのままフィルタやプロパティに設定されます。

戻り値 : なし

5.4.23. 画面表示の停止／再開 (Pause)

形 式 : void Pause (bool fPause);

引 数 : 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、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 : なし

5.4.24. プロット周期計測値の表示(MesPeriShow)

形 式 : void MesPeriShow (bool fShow);

引 数 : 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 1 0 秒間表示後、自動的に消えます。

戻り値 : なし

5.4.25. プロット周期計測をリセット(MesPeriReset)

形 式 : void MesPeriReset();

引 数 : なし

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 : なし

5.4.26. テキスト描画 (TextOut) - ピクセル位置指定

形 式 : int TextOut(int x, int y, string text);

引 数 : 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、「Txo.Right ± n」 or 「Txo.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、「Txo.Bottom ± n」 or 「Txo.Center ± n」を指定)
text - 描画するテキスト

説 明 : チャートウインド上にテキストを描画します。
描画するテキストには、エスケープシーケンスや制御文字を含めることができます。(「テキスト表示拡張機能」の章参照)
x = Txo.Right , y = Txo.Bottom は、テキストをウインドの右下隅に表示する位置となります。
x = Txo.Center, y = Txo.Center は、テキストをウインドの中央に表示する位置となります。

戻り値 : テキストキー

5.4.27. チャートデータ破棄(PurgePlot)

形 式 : void PurgePlot();

引 数 : なし

説 明 : チャートデータを破棄します。

戻り値 : なし

5.4.28. テキスト描画データ破棄(PurgeText)

形 式 : void PurgeText(int key); // 指定 key の描画テキスト破棄
void PurgeText(); // すべての描画テキスト破棄

引 数 : key - データ項目番号 (TextOut() の戻り値)

説 明 : TextOut() で描画したテキストを破棄します。

戻り値 : なし

5.4.29. 全てのデータ破棄(Purge)

形 式 : void Purge ();

引 数 : なし

説 明 : すべてのデータ (チャートデータ, 描画テキスト) を破棄します。

戻り値 : なし

5.5. イベント

タイムチャート・グラフ表示コントロールのイベント一覧を以下に示します。

#	イベント名	内容
1	OnRangeChanged	グラフのレンジが変更されたことを通知します
2	OnNtcScrollPos	スクロールバーにより、グラフスクロール位置が変更されたことを通知します
3	OnFileDrop	ファイルドロップ通知
4	OnDirDrop	ディレクトリドロップ通知
5	OnRClick	右クリックされたことを通知します

5.5.1. レンジ通知 (OnRangeChanged)

形 式 : void OnRangeChanged (object sender, TchArgRangeChanged e);

パラメタ : double e.low - グラフの低位レンジ
double e.high - グラフの高位レンジ

説 明 : グラフのレンジが変更されたことを通知します。

戻り値 : なし

5.5.2. 現在のスクロール位置通知 (OnScrPosChanged)

形 式 : void OnNtcScrollPos (object sender, TchArgNtcScrollPos e);

パラメタ : int e.pos - スクロール位置

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。

戻り値 : なし

5.5.3. ファイルドロップ通知 (OnFileDrop)

形 式 : void OnFileDrop(object sender, VthArgFileDrop e);

パラメタ : int e.n - ドロップされたファイルの個数

説 明 : コントロールへ、ファイルがドロップされたことを通知します。
GetDroppedFile() メソッドにより、ドロップされたファイルのパス名を取得することができます。

戻り値 : なし

5.5.4. ディレクトリドロップ通知 (OnDirDrop)

形 式 : void OnDirDrop(object sender, VthArgDirDrop e);

パラメタ : int e.n - ドロップされたディレクトリの個数

説 明 : コントロールへ、ディレクトリがドロップされたことを通知します。
GetDroppedDir() メソッドにより、ドロップされたディレクトリのパス名を取得することができます。

戻り値 : なし

5.5.5. 右クリック通知 (OnRClick)

形 式 : void OnRClick (object sender, TchArgRClick e);

パラメタ : int e.x - 右クリックしたXピクセル位置 (コントロールの左端が原点(0))
int e.y - " Y " (" 上端 ")
bool e.shift - Shift キーの押下状態 (true:押下)
bool e.ctrl - Ctrl キーの押下状態 (true:押下)

説 明 : コントロール上で右クリックされたことを通知します。

Shift キーと Ctrl キーが押されていない状態で右クリックした場合、通常はポップアップメニューが表示されます。

但し、EnablePopupMenu プロパティが false (右クリックによるポップアップメニュー禁止) に設定されている場合は、ポップアップメニューが表示されず、右クリック通知イベントが発生します。

Shift キーか Ctrl キーが押されている状態で右クリックした場合は、EnablePopupMenu プロパティに関係なく常に右クリック通知イベントが発生します。

戻り値 : なし

5.6. サンプルプログラム

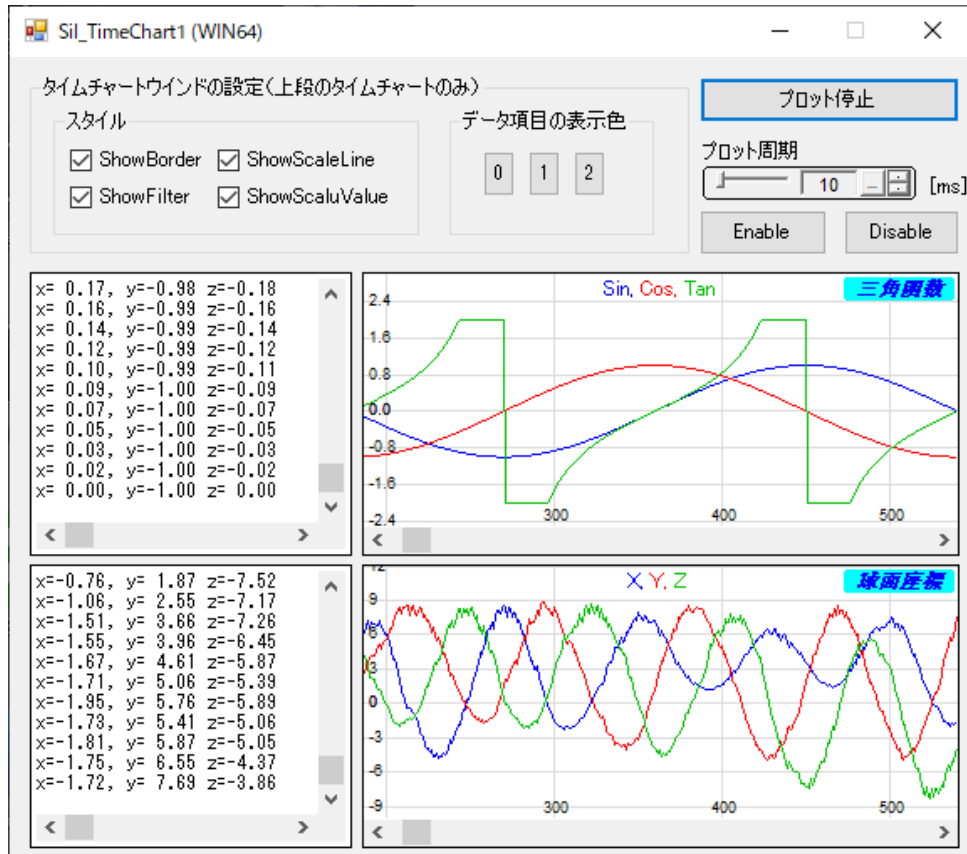
5.6.1. Sil_TimeChart (タイムチャート)

以下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、片方のタイムチャートのスクロールすると、他方のタイムチャートも連動してスクロールします。



```

1 : //
2 : // Sil_TimeChart1
3 : //
4 : using System;
5 : using System.Collections.Generic;
6 : using System.ComponentModel;
7 : using System.Data;
8 : using System.Drawing;
9 : using System.Text;
10 : using System.Windows.Forms;
11 : using CAjrCustCtrl;
12 :
13 : namespace Sil_TimeChart1
14 : {
15 :     public partial class Form1 : Form
16 :     {
17 :         double m_Theta = 0.0;
18 :         bool m_fStop = true;
19 :
20 :         public Form1()
21 :         {
22 :             InitializeComponent();
23 :         }
24 :         //----- 起動時初期設定 -----//

```

```

25 : private void Form1_Load(object sender, EventArgs e)
26 : {
27 :     this.Text = this.Text + (IntPtr.Size == 4 ? " (WIN32)" : " (WIN64)");
28 :     //----- フォームの縦サイズ変更禁止 -----//
29 :     int width = this.Width;
30 :     int height = this.Height;
31 :     this.MinimumSize = new Size(width, height);
32 :     this.MaximumSize = new Size(1900, height);
33 :     //----- タイトルテキスト設定 -----//
34 :     tch1.SetTitleText(" 三角関数 ", Color.Blue, Color.Aqua,
35 :                     new Font("MS UI Gothic", 9, FontStyle.Bold | FontStyle.Italic));
36 :     tch2.SetTitleText(" 球面座標 ", Color.Blue, Color.Aqua,
37 :                     new Font("MS UI Gothic", 9, FontStyle.Bold | FontStyle.Italic));
38 :     //----- 注釈テキスト描画 -----//
39 :     tch1.TextOut(Txo.Center, 3, "¥x1B[30mSin ¥x1B[31mCos ¥x1B[32mTan");
40 :     tch2.TextOut(Txo.Center, 3, "¥x1B[30mX ¥x1B[31mY ¥x1B[32mZ");
41 :     //----- 設定値ロード -----//
42 :     SAjrReg.LoadAllCtrls(this);
43 : }
44 : //----- 終了時後処理 -----//
45 : private void Form1_FormClosed(object sender, FormClosedEventArgs e)
46 : {
47 :     // 設定値セーブ
48 :     SAjrReg.SaveAllCtrls(this);
49 : }
50 : //----- タイマイベント -----//
51 : private void tim_Tick(object sender, EventArgs e)
52 : {
53 :     if (!m_fStop) {
54 :         // データ投与 (タイムチャート1)
55 :         double s = SAjrMath.Sin(m_Theta);
56 :         double c = SAjrMath.Cos(m_Theta);
57 :         double t = SAjrMath.Tan(m_Theta);
58 :         t = Math.Min(t, 2.0);
59 :         t = Math.Max(t, -2.0);
60 :         tch1.PutData(s, c, t);
61 :         m_Theta += 1.0;
62 :         // ログ表示
63 :         vth1.PutText(string.Format("x={0,5:f2} y={1,5:f2} z={2,5:f2}¥n", s, c, t));
64 :         // データ投与 (タイムチャート2)
65 :         AJC3DVEC v = spd.Calc();
66 :         tch2.PutData(v.x, v.y, v.z);
67 :         // ログ表示
68 :         vth2.PutText(string.Format("x={0,5:f2} y={1,5:f2} z={2,5:f2}¥n", v.x, v.y, v.z));
69 :     }
70 : }
71 : //----- タイムチャート1・スクロール通知 -----//
72 : private void tch1_OnNtcScrollPos(object sender, TchArgNtcScrollPos e)
73 : {
74 :     if (m_fStop) {
75 :         tch2.SetScrollPos(e.pos);
76 :     }
77 : }
78 : //----- タイムチャート1・レンジ変化通知 -----//
79 : private void tch1_OnRangeChanged(object sender, TchArgRangeChanged e)
80 : {
81 :     SAjrTip.ShowCenter(tch1, "レンジが設定されました。");
82 : }
83 : //----- タイムチャート1・右クリック通知 -----//
84 : private void tch1_OnRClick(object sender, TchArgRClick e)
85 : {
86 :     SAjrTip.ShowCenter(tch1, (e.ctrl ? "CTRL + " : "") + (e.shift ? "SHIFT + " : "") +
87 :                             "右クリック発生 (x = " + e.x.ToString() + ", y = " + e.y.ToString() + ")");
88 : }
89 : //----- タイムチャート1・ディレクトリドロップ通知 -----//
90 : private void tch1_OnDirDrop(object sender, TchArgDirDrop e)
91 : {
92 :     string text = "ディレクトリがドロップされました。";
93 :     for (int i = 0; i < e.n; i++) {
94 :         text = text + "¥n " + tch1.GetDroppedDir();
95 :     }
96 :     SAjrTip.ShowCenter(tch1, text);
97 : }
98 : //----- タイムチャート1・ファイルドロップ通知 -----//
99 : private void tch1_OnFileDrop(object sender, TchArgFileDrop e)
100 : {
101 :     string text = "ファイルがドロップされました。";
102 :     for (int i = 0; i < e.n; i++) {
103 :         text = text + "¥n " + tch1.GetDroppedFile();
104 :     }

```



```

105 :         SAjrTip.ShowCenter(tch1, text);
106 :     }
107 :     //----- タイムチャート2・スクロール通知 -----//
108 :     private void tch2_OnNtcScrollPos(object sender, TchArgNtcScrollPos e)
109 :     {
110 :         if (m_fStop) {
111 :             tch1.SetScrollPos(e.pos);
112 :         }
113 :     }
114 :     //----- タイムチャート2・レンジ変化通知 -----//
115 :     private void tch2_OnRangeChanged(object sender, TchArgRangeChanged e)
116 :     {
117 :         SAjrTip.ShowCenter(tch2, "レンジが設定されました。");
118 :     }
119 :     //----- タイムチャート2・右クリック通知 -----//
120 :     private void tch2_OnRClick(object sender, TchArgRClick e)
121 :     {
122 :         SAjrTip.ShowCenter(tch2, "右クリック発生 (x = " + e.x.ToString() + ", y = " + e.y.ToString() + ")");
123 :     }
124 :     //----- タイムチャート2・ディレクトリドロップ通知 -----//
125 :     private void tch2_OnDirDrop(object sender, TchArgDirDrop e)
126 :     {
127 :         string text = "ディレクトリがドロップされました。";
128 :         for (int i = 0; i < e.n; i++) {
129 :             text = text + "¥n " + tch2.GetDroppedDir();
130 :         }
131 :         SAjrTip.ShowCenter(tch2, text);
132 :     }
133 :     //----- タイムチャート2・ファイルドロップ通知 -----//
134 :     private void tch2_OnFileDrop(object sender, TchArgFileDrop e)
135 :     {
136 :         string text = "ファイルがドロップされました。";
137 :         for (int i = 0; i < e.n; i++) {
138 :             text = text + "¥n " + tch2.GetDroppedFile();
139 :         }
140 :         SAjrTip.ShowCenter(tch2, text);
141 :     }
142 :     //----- プロット開始/停止ボタン -----//
143 :     private void btnStart_Click(object sender, EventArgs e)
144 :     {
145 :         if (m_fStop) {
146 :             tch1.Start();
147 :             tch2.Start();
148 :             tim.Enabled = true;
149 :             m_fStop = false;
150 :             btnStart.Text = "プロット停止";
151 :         }
152 :         else {
153 :             tch1.Stop();
154 :             tch2.Stop();
155 :             tim.Enabled = false;
156 :             m_fStop = true;
157 :             btnStart.Text = "プロット開始";
158 :         }
159 :     }
160 :     //----- プロット周期変更通知 -----//
161 :     private void cAjrInpValue1_OnNtcIntValue(object sender, IvArgNtcIntValue e)
162 :     {
163 :         tim.Interval = e.value;
164 :     }
165 :     //----- Enable ボタン -----//
166 :     private void btnEnable_Click(object sender, EventArgs e)
167 :     {
168 :         tch1.Enabled = true;
169 :         tch2.Enabled = true;
170 :     }
171 :     //----- Disable ボタン -----//
172 :     private void btnDisable_Click(object sender, EventArgs e)
173 :     {
174 :         tch1.Enabled = false;
175 :         tch2.Enabled = false;
176 :     }
177 :     //----- スタイル設定チェックボックス群 -----//
178 :     private void chkShowBorder_CheckedChanged (object sender, EventArgs e) {tch1.ShowBorder = chkShowBorder.Checked;}
179 :     private void chkShowFilter_CheckedChanged (object sender, EventArgs e) {tch1.ShowFilter = chkShowFilter.Checked;}
180 :     private void chkShowScaleLine_CheckedChanged (object sender, EventArgs e) {tch1.ShowScaleLine = chkShowScaleLine.Checked;}
181 :     private void chkShowScaleValue_CheckedChanged(object sender, EventArgs e) {tch1.ShowScaleValue = chkShowScaleValue.Checked;}
182 :     //----- データ項目の表示色設定ボタン群 -----//
183 :     private void btnColor0_Click(object sender, EventArgs e) {SetItemColor(0);}
184 :     private void btnColor1_Click(object sender, EventArgs e) {SetItemColor(1);}

```

```
185 : private void btnColor2_Click(object sender, EventArgs e) {SetItemColor(2);}
186 : //----- データ項目の表示色設定 -----//
187 : private void SetItemColor(int id)
188 : {
189 :     ColorDialog cd = new ColorDialog();
190 :     cd.Color = tchl.GetItemColor(id);
191 :     if (cd.ShowDialog() == DialogResult.OK)
192 :     {
193 :         tchl.SetItemColor(id, cd.Color);
194 :     }
195 : }
196 : }
197 : }
```

6. 2D/3Dグラフィック描画コントロール (CAjr3DGraphic.dll)

3次元座標系 (X, Y, Z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。

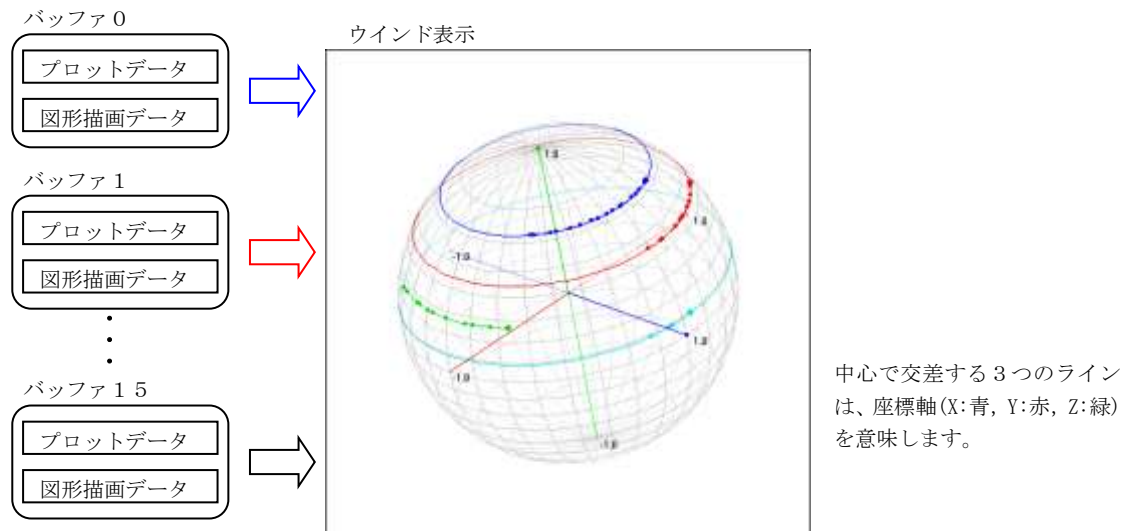
このコントロールは、主に実験データのプロットイングや、演算結果をグラフに描画してグラフィカルに確認することを目的としています。(OpenGL や Direct-X のようにシェーディングやテクスチャマッピングといった高度な描画機能はありません)

プロットデータや図形描画データは、16個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、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)



6.1.1. 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。

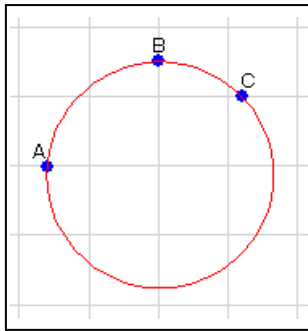
プロットデータとは、投与した座標位置を点で描画するデータであり、蓄積する個数に制限を設け、制限個数を超えた場合は、古いデータから順次破棄されます。つまり、プロットデータは、制限された個数の最新データだけを表示することになります。

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。

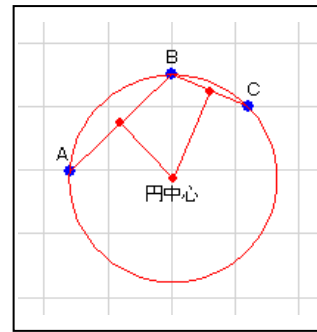
例えば、(2次元で) サンプリングした3つのプロットデータから、円を求めるとします。

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 midpoint 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す。

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。

```
AJC2DVEC    vA = new AJC2DVEC(1234, 350);    // 3つのプロット点 (ここでは定数とする)
AJC2DVEC    vB = new AJC2DVEC(1250, 365);    // .
AJC2DVEC    vC = new AJC2DVEC(1262, 360);    // .
AJC2DVEC    vo0, vo1;                        // 2つのベクトル (A->B, b->C)
AJC2DLVEC   v10, v11;                        // 2つの線分の midpoint と垂線
AJC2DVEC    vCent;                           // 円の中心
double      r;                               // 円の半径

g2d.Pixel(BLUE, vA, 4);                      // 3つのプロット点表示
g2d.Pixel(BLUE, vB, 4);                      // .
g2d.Pixel(BLUE, vC, 4);                      // .

g2d.TextOut(EAJCTXOMD.ABOVE_LEFT, vA, "A");  // 3点の名前 (A, B, C) 表示
g2d.TextOut(EAJCTXOMD.ABOVE_CENTER, vB, "B"); // .
g2d.TextOut(EAJCTXOMD.ABOVE_RIGHT, vC, "C"); // .

v10.p = SAjrMath.V2dSub(vB, vA);    v11.p = SAjrMath.V2dSub(vC, vB);    // 2つの線分の midpoint 算出
v10.p = SAjrMath.V2dMult(v10.p, 0.5); v11.p = SAjrMath.V2dMult(v11.p, 0.5); // .
v10.p = SAjrMath.V2dAdd(vA, v10.p); v11.p = SAjrMath.V2dAdd(vB, v11.p);    // .

vo0 = SAjrMath.V2dSub(vB, vA);    vo1 = SAjrMath.V2dSub(vC, vB);    // 2つの midpoint から垂線を算出
v10.v = SAjrMath.V2dAnyOrthoVec(vo0); v11.v = SAjrMath.V2dAnyOrthoVec(vo1); // .

vCent = SAjrMath.V2dCrossL2L(v10, v11);    // 2つの垂線の交点算出

g2d.Line(RED, vA, vB);    g2d.Line(RED, vB, vC);    // 線分 A-B, B-C 表示
g2d.Pixel(RED, v10.p, 3);    g2d.Pixel(RED, v11.p, 3);    // 線分の midpoint 表示
g2d.Line(RED, v10.p, vCent);    // 2つの midpoint からの垂線表示
g2d.Line(RED, v11.p, vCent);    // .
g2d.Pixel(RED, vCent, 3);    // 円の中心表示
g2d.TextOut(EAJCTXOMD.BELLOW_CENTER, vCent, "円中心");    // .

r = SAjrMath.V2dDistP2P(vCent, vA);    // 円の半径算出
g2d.Ellipse(RED, vCent.x, vCent.y, r, r);    // 円描画
```

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。

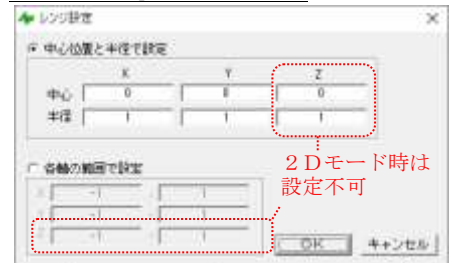
3Dモード時のポップアップメニュー



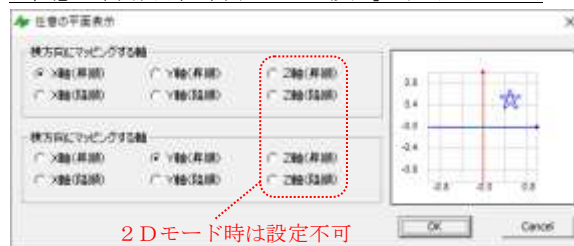
2Dモード時のポップアップメニュー



「レンジ設定」時のダイアログ



「任意の平面表示／平面角度設定」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／平面の角度設定 (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、平面の角度を設定します。 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	X-Y平面スケール表示	X-Y平面に方眼／同心円スケールの表示を許可／禁止
16	X-Z平面スケール表示	X-Z平面に方眼／同心円スケールの表示を許可／禁止
17	Y-Z平面スケール表示	Y-Z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 (EnableMesDraw プロパティで、true を指定したした場合に表示)

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。

「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。

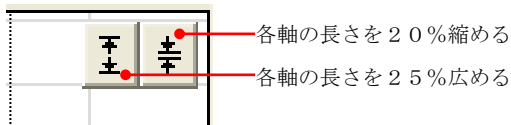
「Cancel」ボタンを押すと、設定を中止します。



6.2.3. ワンタッチでレンジ設定

マウスカursor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。

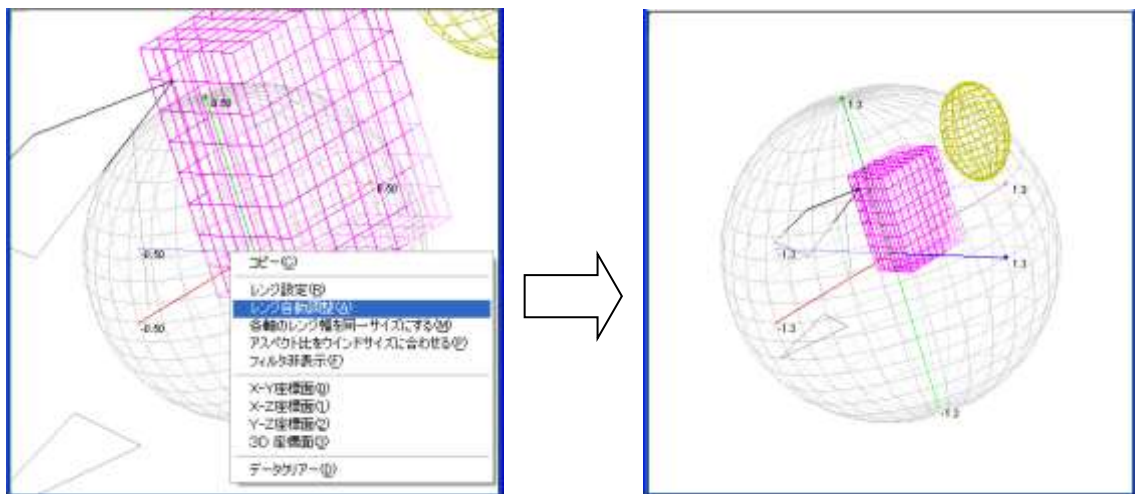
これらのボタンで、各軸の長さ（直径）を30%広めたり、縮めたりすることができます。



6.2.4. レンジ自動調整

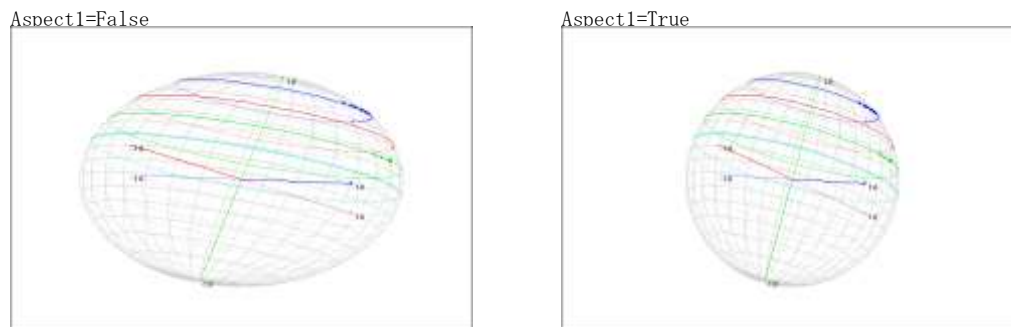
ポップアップメニューで「レンジ自動調整」を選択すると、(フィルタで非表示となっている項目を除く) 全てのデータから、最大値と最小値を算出し、(中心位置は変わらないように) $\pm 5\%$ のマージンを持ってレンジ設定を行います。

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」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。

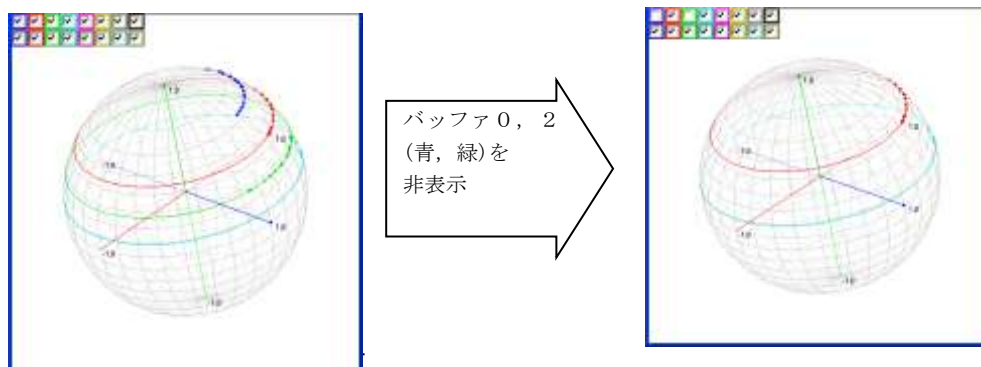
アスペクト比は、プロパティ (fAspect1) によっても設定可能です。

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16個のチェックボックスが表示されます。

こ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／非表示はプロパティの設定によります)



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。

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。

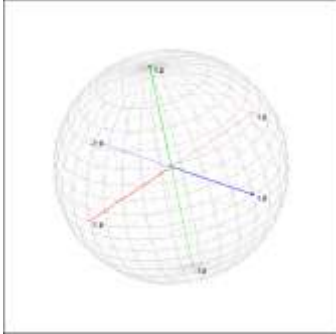
マウスで縦方向にドラッグすると、表示物体がY軸回りに回転します。

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。

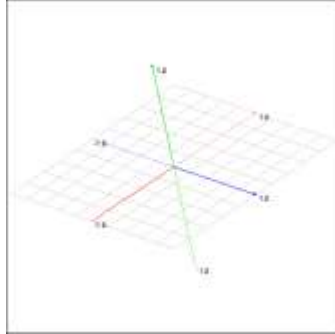
6.2.8. スケールの表示

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。

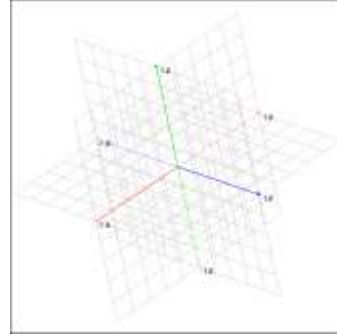
球形スケール



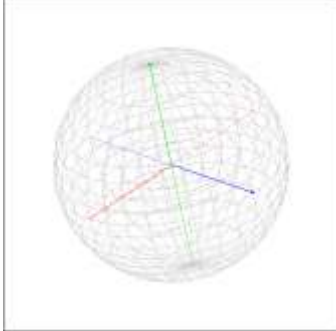
X Y座標面で方眼スケール



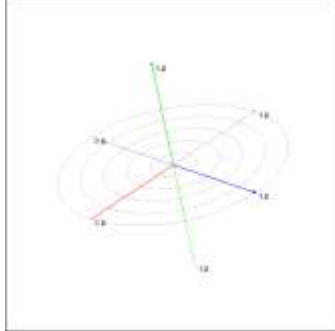
全ての座標面で方眼スケール



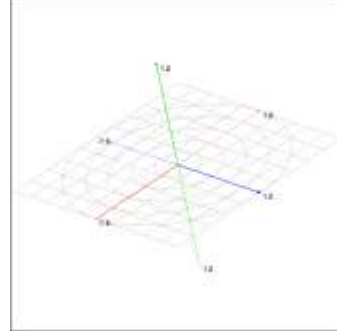
スケール値非表示



X Y座標面で同心円スケール

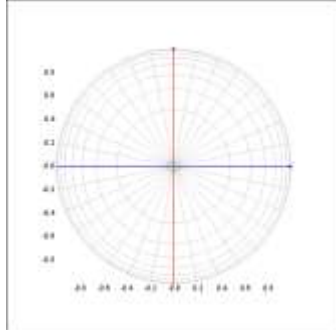


方眼と同心円スケールを併用

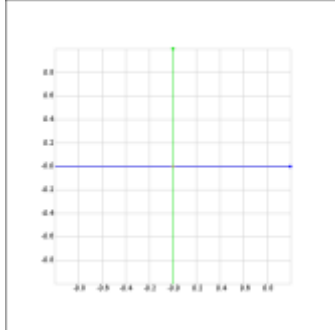


スケール値は、視点がいずれかの座標軸と一致した（つまり、いずれか2軸の平面を見る）場合に限り、中間値も表示されます。その他の視点設定では、各軸の先端値だけを表示します。

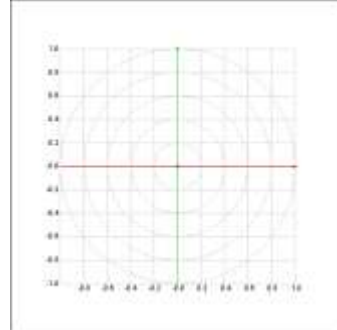
X Y平面（球形スケール）



X Z平面（方眼スケール）



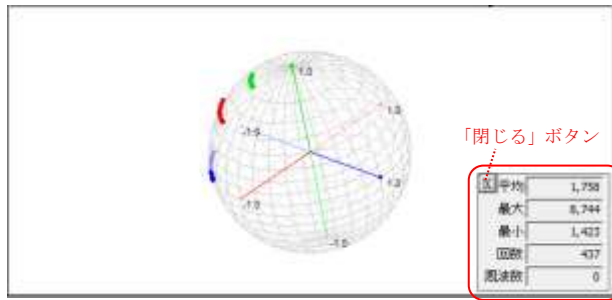
Y Z平面（方眼+同心円スケール）



スケールやスケール値の表示は、プロパティ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[μ s]

最大：最大描画時間[μ s]

最小：最小描画時間[μ s]

回数：計測回数

周波数：計測周波数[Hz]（PCで固定な値）

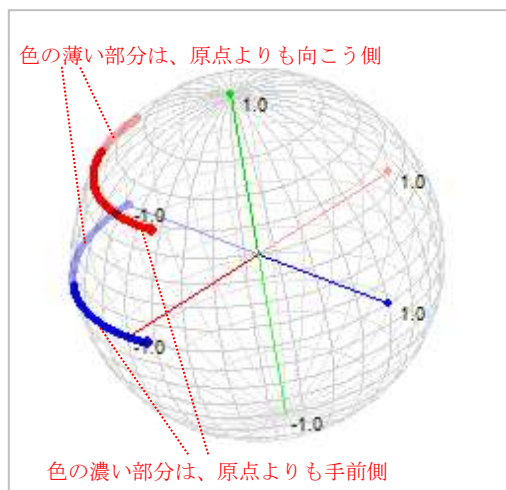
ウインドのサイズを変更した場合は、計測をやり直します。

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、描画時間だけではなく、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	■	■

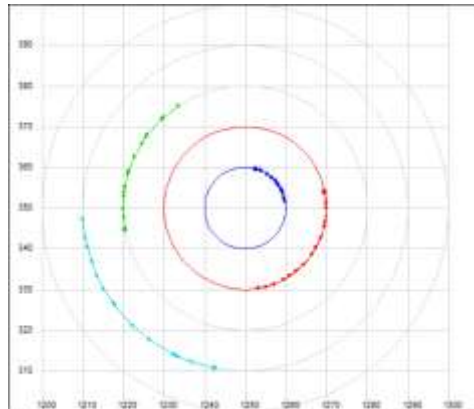
この描画色は、SetItemColorP() / SetItemColorN() メソッドで変更できます。

ポップアップメニューで「奥行き表現禁止」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 _DimMode プロパティを「MODE_2D」に設定します。

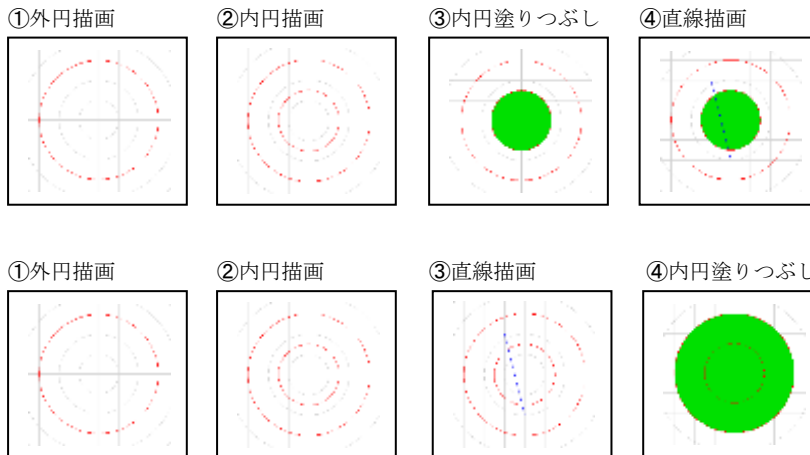
2Dグラフモードでの塗りつぶし

2Dグラフモードでは、塗りつぶしメソッドが使用できます。

FillB() - 閉領域の塗りつぶし

FillS() - 白色部分の塗りつぶし

これらの塗りつぶしメソッドは、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、Pause() メソッド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。

```

        :
        :
g3d. Pause(TRUE);           // 表示停止
g3d. PutData ( . . . );     // データ投与
g3d. Pixel  ( . . . );
g3d. Line   ( . . . );
        :
        :
g3d. Pause(FALSE);         // 表示再開 (①の期間に描画したデータを一気に表示)
        :
    
```

① (この間描画したデータは表示されない)

g3d. Pause(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)

g3d. Pause(true)～g3d. Pause(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。

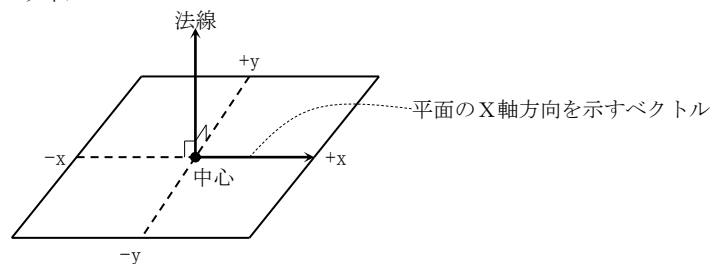
※g3d. Pause(true)～g3d. Pause(false)で表示を抑止している期間でも、ユーザ操作(ウインドのサイズを変えたり、最小化したタスクバーから戻す、等)によりウインドの再描画が必要な場合は、その瞬間の画面状態が表示されます。

6.2.13. 任意の平面定義と平面上への図形描画

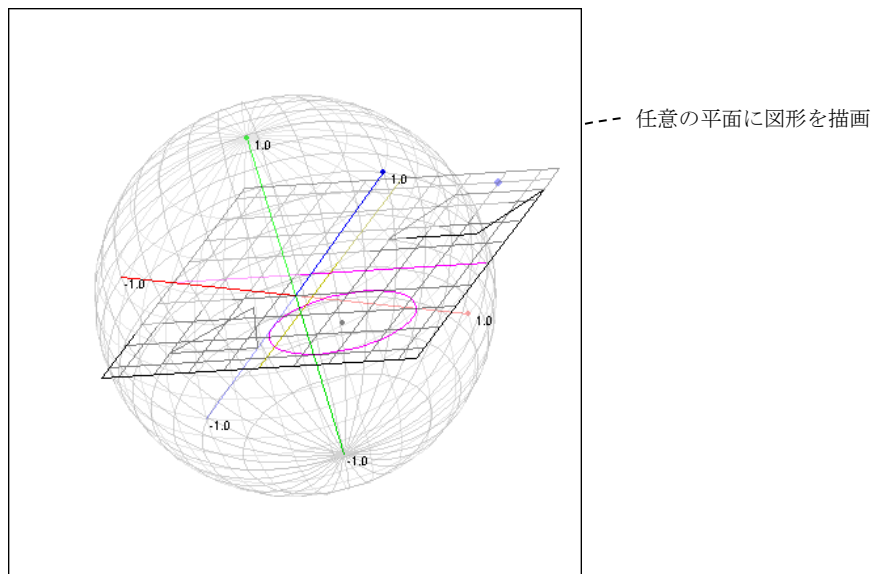
3D空間上に、任意の平面を定義し、この平面上に図形を描画することができます。

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、楕円等を描画することができます。



6.2.14. 右クリック

右クリック操作による動作は、以下のようになっています。

ポップアップメニュー許可 EnablePopupMenu プロパティ	SHIFT/CTRL キー押下	動作
True（許可）	×（未押下）	ポップアップメニューを表示
True（許可）	○（押下）	右クリック通知イベント（OnRClick）発生（右ボタン押下時に発生）
False（禁止）	—	右クリック通知イベント（OnRClick）発生（右ボタン押下時に発生）

6.2.15. 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、以下のイベントが発生します。

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

これらのイベントでは、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GetDroppedFile/GetDroppedDir メソッドにて取得します。

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、AcceptFiles プロパティを true に指定する必要があります。

6.3. プロパティ

2D/3Dグラフィック描画表示コントロールのプロパティ一覧を示します。

#	名称	タイプ	内容	デフォルト
1	_DimMode	EDIMMODE	2Dモード/3Dモードの設定 (※1) ・MODE_2D - 2Dモード ・MODE_3D - 3Dモード	MODE_3D
2	AcceptFiles	bool	ファイル/フォルダのドロップ許可フラグ	false
3	Aspect1	bool	ビューボリュームを常に、立方体となるようにします。	true
4	CenterX, Y, Z	double	ビューボリュームの中心位置	0, 0, 0
5	EnableAngle	bool	マウスドラッグによる視点変更の許可フラグ	true
6	EnableDepthControl	bool	遠近感制御の許可フラグ	true
7	EnableMesDraw	bool	ポップアップメニュー「描画時間情報 表示」の許可/禁止	false
8	EnablePopupMenu	bool	ポップアップメニュー許可フラグ (※2)	true
9	FontTxo	Font	テキスト描画時のフォント	MS UI Gothic, 9
10	PlaneHoriAxis	EAJCPLANEAXIS	平面表示時の横方向にマッピングする軸の種別	XP (X軸昇順)
11	PlaneVertAxis	EAJCPLANEAXIS	平面表示時の縦方向にマッピングする軸の種別	YP (Y軸昇順)
12	PlotLine	bool	プロットデータの結線フラグ	true
13	PlotSize	int	プロットデータのピクセルサイズ	2
14	PlotSizeE	int	最終プロットデータのピクセルサイズ	3
15	RadiusX, Y, Z	double	ビューボリュームの半径	1, 1, 1
16	RotateX, Y, Z	double	視点設定 (物体の回転角度)	60, 10, 45
17	ShowAxisX, Y, Z	bool	X, Y, Z軸の表示フラグ	true, true, true
18	ShowBorder	bool	コントロール外枠の表示フラグ	true
19	ShowDummyData	bool	デザイン時のダミーデータ表示フラグ (※3)	true
20	ShowEllipseScale	bool	同心円スケールの表示フラグ	false
21	ShowFilter	bool	フィルタ (チェックボックス) 表示フラグ	true
22	ShowRectScale	bool	方眼スケールの表示フラグ	false
23	ShowScaleValueX, Y, Z	bool	各軸におけるスケール値の表示フラグ	true, true, true
24	ShowScaleXY	bool	X Y 平面へのスケールを表示フラグ	false
25	ShowScaleXZ	bool	X Z 平面へのスケールを表示フラグ	false
26	ShowScaleYZ	bool	Y Z 平面へのスケールを表示フラグ	false
27	ShowSphereScale	bool	[楕円] 球体スケールの表示フラグ	true
28	TitleText	string	タイトルテキスト (文字色=白, 背景=グレーで右上に表示)	""
29	ToolTipFilter00~15	string	フィルタの各チェックボックスのツールヒント (※4)	""
30	ToolTipText	string	コントロールのツールヒント (※4)	""
31	ViewVolumeRatio	double	ビューボリュームのサイズ (画面に占める割合)	0.7
32	ToolTipShowAlways	bool	ツールチップ表示条件 true - 非アクティブ状態でも表示 false - 非アクティブ状態時は非表示	true

※1: _DimMode プロパティを設定すると、他のプロパティも当該モード用に初期化されます。

このプロパティは、他のプロパティに先駆けて、最初に設定してください。

※2: 右クリックによるポップアップメニューの表示(true)/非表示(false)を設定します。

非表示(false)を設定した場合は、右クリックすると「OnRClick」イベントが発生します。

(Shift/Ctrl + 右クリックした場合は、EnablePopupMenu プロパティに関係なく常に「OnRClick」イベントが発生します)

※3: デザイン時にダミーデータを表示することにより、プロパティの効果を視覚的に確認することができます。

※4: ToolTipText, ToolTipFilter00~15 プロパティは、制御文字とエスケープシーケンスを含めることができます。

これについては、「テキスト表示拡張機」の章を参照してください。

6.4. メソッド

2D/3Dグラフィック描画コントロールメソッド一覧を以下に示します。

#	メソッド名	内容	備考
1	PutData	プロットデータ投与	
2	SetRange	レンジ設定	
3	AdjustRange	レンジ自動調整	
4	SetCenter	中心位置設定	
5	SetRadius	半径設定	
6	SetSameRadius	各軸の半径を同一にする	
7	Pixel	ピクセル描画	
8	MoveTo	ラインの始点設定	
9	LineTo	ラインの終点設定 (ライン/矢印描画)	
10	Line	ライン/矢印描画	
11	Triangle	三角形描画	
12	Square	四角形描画	
13	Cube	立方体/長方体描画	
14	Sphere	球/楕円球描画	
15	DefPlane	3D空間上に任意の平面を定義	
16	Ellipse	円/楕円描画	
17	Star	星形描画	
18	SetFilter	フィルタの設定	
19	GetFilter	フィルタ設定値の取得	
20	SetMaxPlot	最大プロット数設定	
21	GetMaxPlot	最大プロット数取得	
22	SetItemColorP	データ項目の前面表示色設定	
23	GetItemColorP	データ項目の前面表示色取得	
24	SetItemColorN	データ項目の後面表示色設定	
25	GetItemColorN	データ項目の後面表示色取得	
26	SetAngle	視点設定	
27	SetAngleXY	視点をXY平面に設定	
28	SetAngleXZ	視点をXZ平面に設定	
29	SetAngleYZ	視点をYZ平面に設定	
30	SetAngle3D	視点を3Dイメージに設定	
31	SetAnyPlane	視点を任意の平面に設定	
32	GetDroppedFile	ドロップされたファイル名取得	
33	GetDroppedDir	ドロップされたディレクトリ名取得	
34	SetTitleText	タイトルテキストを表示	画面右上に表示
35	SaveToProfile	現在の設定値をプロファイルへ記録する	
36	LoadFromProfile	現在の設定値をプロファイルへ記録する	
37	FillB	ボーダー色で囲まれた部分の塗りつぶし	2Dモード用
38	FillS	連続する白色部分の塗りつぶし	
39	GetPixel	ピクセルの表示色取得	
40	Pause	画面表示の停止/再開	
41	TextOut	テキスト描画 (ピクセル位置指定)	
42	TextOut	テキスト描画 (2Dモード用)	
43	TextOut	テキスト描画 (3Dモード用)	
44	PurgeShape	図形描画データ破棄 (描画した図形データのクリアー)	
45	PurgePlot	プロットデータ破棄 (プロットデータのクリアー)	
46	PurgeText	テキスト描画データ破棄 (描画したテキストのクリアー)	
47	PurgeData	描画データ (図形, プロット) 破棄	
48	Purge	全てのデータ (図形, プロット, テキスト) 破棄	

6.4.1. プロットデータ投与(PutData)

形 式 : void PutData(int id, double x, double y, double z); -- 3Dプロットデータ投与
 void PutData(int id, AJC3DVEC v); ----- 3Dプロットデータ投与 (ベクトル指定)
 void PutData(int id, double x, double y); ----- 2Dプロットデータ投与
 void PutData(int id, AJC2DVEC v); ----- 2Dプロットデータ投与 (ベクトル指定)

引 数 : x, y, z - 投与するデータ値
 v - 投与するベクトルデータ

説 明 : プロットデータを投与します。

戻り値 : なし

6.4.2. レンジ設定(SetRange)

形 式 : void SetRange (double x1, double y1, double z1, double x2, double y2, double z2); --- 3Dレンジ設定
 void SetRange(AJC3DVEC v1, AJC3DVEC v2); ----- 3Dレンジ設定 (ベクトル指定)
 void SetRange(double x1, double y1, double x2, double y2); ----- 2Dレンジ設定
 void SetRange(AJC2DVEC v1, AJC2DVEC v2); ----- 2Dレンジ設定 (ベクトル指定)

引 数 : id - データ項目番号 (0 ~ 15)
 x1, y1, z1 - 各軸の低位値
 x2, y2, z2 - 各軸の高位値
 v1 - 各軸の低位値 (ベクトル指定)
 v2 - 各軸の高位値 (ベクトル指定)

説 明 : 当該データ項目の図形描画データをすべて破棄します。

戻り値 : なし

6.4.3. レンジ自動調整(AdjustRange)

形 式 : void AdjustRange();

引 数 : なし

説 明 : (フィルタで非表示となっている項目を除いて) 全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。但し、各座標軸の中心位置は変化しません。
 つまり、中心位置を変えないで、すべてのデータがビューボックス内に入るようにレンジを調整します。

戻り値 : なし

6.4.4. 中心位置設定(SetCenter)

形 式 : void SetCenter(double xc, double yc, double zc); --- 3Dの中心設定
 void SetCenter(AJC3DVEC vc); ----- 3Dの中心設定 (ベクトル指定)
 void SetRange(double xc, double yc); ----- 2Dの中心設定
 void SetCenter(AJC2DVEC vc); ----- 2Dの中心設定 (ベクトル指定)

引 数 : xc, yc, zc - 各軸の中心位置
 vc - 各軸の中心位置 (ベクトル指定)

説 明 : 各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 : なし

6.4.5. 半径設定(SetRadius)

形 式 : void SetRadius(double xr, double yr, double zr); --- 3Dの半径設定
 void SetRadius(AJC3DVEC r); ----- 3Dの半径設定 (ベクトル指定)
 void SetRadius(double xr, double yr); ----- 2Dの半径設定
 void SetRadius(AJC2DVEC r); -- ----- 2Dの半径設定 (ベクトル指定)

引 数 : xr, yr, zr - 各軸の半径 (中心から端点までの長さ)
 vc - 各軸の半径 (ベクトル指定)

説 明 : 各軸のレンジを半径値で指定します。
 各軸のレンジは、(中心-半径) ~ (中心+半径) となります。

戻り値 : なし

6.4.6. 各軸の半径を同一値にする(SetSameRadius)

形 式 : void SetSameRadius();

引 数 : なし

説 明 : 中心位置を変更しないで、各軸のレンジ範囲 (各軸の長さ) を同一に設定します。(最大のレンジ幅に合わせます)
 各軸のレンジは、(中心-半径) ~ (中心+半径) となります。

戻り値 : なし

6.4.7. ピクセル描画(Pixel)

形 式 : void Pixel(int id, double x, double y, double z, int pixelSize); --- 3D立体空間への描画
 void Pixel(int id, AJC3DVEC v, int pixelSize); ----- 3D立体空間への描画 (ベクトル指定)
 void Pixel(int id, double x, double y, int pixelSize); ----- 2D平面への描画
 void Pixel(int id, AJC2DVEC v, int pixelSize); ----- 2D平面への描画 (ベクトル指定)

引 数 : id - データ項目番号 (0～15)
 x, y, z - 描画位置
 PixelSize - ピクセルのサイズ (1: 1ドットで描画, 2～: 塗りつぶし円の半径)

説 明 : 3D立体空間/2D平面 (DefPlane()により定義された平面/2Dモードでの平面) 上にピクセル (点) を描画します。

戻り値 : なし

6.4.8. ライン始点設定(MoveTo)

形 式 void MoveTo(int id, double x, double y, double z); --- 3D空間の始点設定
 void MoveTo(int id, AJC3DVEC v); ----- 3D空間の始点設定 (ベクトル指定)
 void MoveTo(int id, double x, double y); ----- 2D平面の始点設定
 void MoveTo(int id, AJC2DVEC v); ----- 2D平面の始点設定 (ベクトル指定)

引 数 : id - データ項目番号 (0～15)
 x, y, z - ラインの始点位置
 v - ラインの始点位置 (ベクトル指定)

説 明 : 3D立体空間/2D平面 (DefPlane()により定義された平面/2Dモードでの平面) 上でラインの始点を設定します。

戻り値 : なし

6.4.9. ライン終点設定 - ライン/矢印描画(LineTo)

形 式 void LineTo(int id, double x, double y, double z); --- 3D空間の終点を設定しライン描画
 void LineTo(int id, double x, double y, double z, bool fArrow);

 void LineTo(int id, AJC3DVEC v); ----- 3D空間の終点を設定しライン描画 (ベクトル指定)
 void LineTo(int id, AJC3DVEC v, bool fArrow);

 void LineTo(int id, double x, double y); ----- 2D平面の終点を設定しライン描画
 void LineTo(int id, double x, double y, bool fArrow);

 void LineTo(int id, AJC2DVEC v); ----- 2D平面の終点を設定しライン描画 (ベクトル指定)
 void LineTo(int id, AJC2DVEC v, bool fArrow);

引 数 : id - データ項目番号 (0～15)
 x, y, z - ラインの終点位置
 v - ラインの終点位置 (ベクトル指定)
 fArrow - 矢印描画フラグ

説 明 : 3D立体空間/2D平面 (DefPlane()により定義された平面/2Dモードでの平面) 上でラインの終点を指定しラインを描画します。
 ラインの始点は、MoveTo()メソッドで指定します。
 指定したラインの終点は、次のLineTo()メソッドでラインを描画する際の始点に設定されます。
 fArrow=true を指定した場合は、ラインの終点に矢印を描画します。(但し、ラインが10ピクセル未満時は、矢印を描画しない)

戻り値 : なし

6.4.10. ライン/矢印描画(Line)

形 式 `void Line(int id, double x1, double y1, double z1, double x2, double y2, double z2);` --- 3D空間への描画
`void Line(int id, double x1, double y1, double z1, double x2, double y2, double z2, bool fArrow);`

`void Line(int id, AJC3DVEC v1, AJC3DVEC v2);` ----- 3D空間への描画 (ベクトル指定)
`void Line(int id, AJC3DVEC v1, AJC3DVEC v2, bool fArrow);`

`void Line(int id, double x1, double y1, double x2, double y2);` --- 2D平面への描画
`void Line(int id, double x1, double y1, double x2, double y2, fArrow);`

`void Line(int id, AJC2DVEC v1, AJC2DVEC v2);` ----- 2D平面への描画 (ベクトル指定)
`void Line(int id, AJC2DVEC v1, AJC2DVEC v2, bool fArrow);`

引 数 : `id` - データ項目番号 (0 ~ 15)
 `x1, y1, z1` - ラインの始点位置
 `x2, y2, z2` - ラインの終点位置
 `v1` - ラインの始点位置 (ベクトル指定)
 `v2` - ラインの終点位置 (ベクトル指定)

説 明 : 3D立体空間/2D平面 (DefPlane())により定義された平面/2Dモードでの平面) 上にライン (線分) を描画します。
 `fArrow=true` を指定した場合は、ラインの終点に矢印を描画します。(但し、ラインが10ピクセル未満時は、矢印を描画しない)

戻り値 : なし

6.4.11. 三角形描画(Triangle)

形 式 : --- 3D空間への描画 ---
`void Triangle(int id, double x1, double y1, double z1, double x2, double y2, double z2, double x3, double y3, double z3);`
`void Triangle(int id, AJC3DVEC v1, AJC3DVEC v2, AJC3DVEC v3);`
 --- 2D平面への描画 ---
`void Triangle(int id, double x1, double y1, double x2, double y2, double x3, double y3);`
`void Triangle(int id, AJC2DVEC v1, AJC2DVEC v2, AJC2DVEC v3);`

引 数 : `id` - データ項目番号 (0 ~ 15)
 `x1, y1, z1` - 三角形の頂点位置 1
 `x2, y2, z2` - 三角形の頂点位置 2
 `x3, y3, z3` - 三角形の頂点位置 3
 `v1, v2, v3` - 三角形の頂点位置 (ベクトル指定)

説 明 : 3D立体空間/2D平面 (DefPlane())により定義された平面/2Dモードでの平面) 上に三角形を描画します。

戻り値 : なし

6.4.12. 四角形描画(Square)

形 式 : --- 3D空間への描画 ---

```
void Square(int id, double x1, double y1, double z1, double x2, double y2, double z2, double x3, double y3, double z3,
            double x4, double y4, double z4);
```

```
void Square(int id, AJC3DVEC v1, AJC3DVEC v2, AJC3DVEC v3, AJC3DVEC v4);
```

--- 2D平面への描画 ---

```
void Square(int id, double x1, double y1, double x2, double y2, double x3, double y3, double x4, double y4);
```

```
void Square(int id, AJC2DVEC v1, AJC2DVEC v2, AJC2DVEC v3, AJC2DVEC v4)
```

引 数 : id - データ項目番号 (0 ~ 15)
 x1, y1, z1 - 四角形の頂点位置 1
 x2, y2, z2 - 四角形の頂点位置 2
 x3, y3, z3 - 四角形の頂点位置 3
 x4, y4, z4 - 四角形の頂点位置 4
 v1, v2, v3, v4 - 四角形の頂点位置 (ベクトル指定)

説 明 : 3D立体空間/2D平面 (DefPlane()により定義された平面/2Dモードでの平面) 上に 四角形を描画します。

戻り値 : なし

6.4.13. 立方体/長方体描画(Cube)

形 式 : void Cube(int id, double xc, double yc, double zc, double xr, double yr, double zr, int division);
 void Cube(int id, AJC3DVEC vc, AJC3DVEC vr, int division);

引 数 : id - データ項目番号 (0 ~ 15)
 xc, yc, zc - 立方体/長方体の中心位置
 xr, yr, zr - 立方体/長方体の各軸の長さ
 cv - 立方体/長方体の中心位置 (ベクトル指定)
 vr - 立方体/長方体の各軸の長さ (ベクトル指定)
 division - 分割数

説 明 : 3D立体空間に立方体/長方体を描画します。

戻り値 : なし

6.4.14. 球/楕円球描画(Sphere)

形 式 : void Sphere(int id, double xc, double yc, double zc, double xr, double yr, double zr, int slice, int stack);
 void Sphere(int id, AJC3DVEC vc, AJC3DVEC vr, int slice, int stack);

引 数 : id - データ項目番号 (0 ~ 15)
 xc, yc, zc - 球/楕円球の中心位置
 xr, yr, zr - 球/楕円球の各軸の半径
 cv - 球/楕円球の中心位置 (ベクトル指定)
 vr - 球/楕円球の各軸の半径 (ベクトル指定)
 slice - 水平分割数
 stack - 垂直分割数

説 明 : 3D立体空間に球/楕円球を描画します。

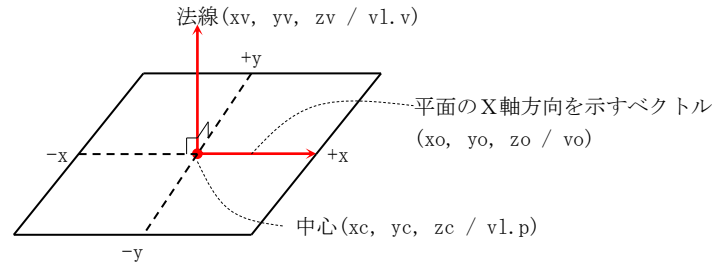
戻り値 : なし

6.4.15. 3D空間上に任意の平面を定義(DefPlane)

形 式 : void DefPlane(int id, double xc, double yc, double zc, double xv, double yv, double zv, double xo, double yo, double zo);
 void DefPlane(int id, AJC3DVEC vl);
 void DefPlane(int id, AJC3DVEC vl, AJC3DVEC vo);

引 数 : 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 - 2D平面の原点(0, 0)とする位置
 xv, yv, zv - 平面の法線ベクトル
 xv, yv, zv - 平面上でX軸 (正方向) とするベクトル
 vl - 2D平面の原点(0, 0)とする位置と法線ベクトル (ベクトル指定)
 vo - 平面上でX軸 (正方向) とするベクトル (ベクトル指定)

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 とした場合や、vo 未指定の場合は、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。
 (X軸方向を示す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不要)
 定義した平面上の座標系は、xc, yc, zc / vl.p で指定した位置が原点(0, 0)となります。

戻り値 : なし

6.4.16. 円/楕円描画(Ellipse)

形 式 : void Ellipse(int id, double xc, double yc, double xr, double yr);
 void Ellipse(int id, AJC2DVEC vc, AJC2DVEC vr);

引 数 : id - データ項目番号 (0 ~ 15)
 xc, yc - 円/楕円の中心位置
 xr, yr - 円/楕円の各軸の半径
 vc - 円/楕円の中心位置 (ベクトル指定)
 vr - 円/楕円の各軸の半径 (ベクトル指定)

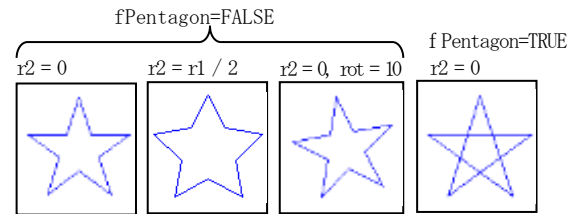
説 明 : DefPlane()により定義された平面 (あるいは2Dモードでの平面) に 円を描画します。

戻り値 : なし

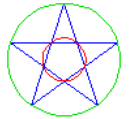
6.4.17. 星形描画(Star)

形 式 : void Star(int id, double xc, double yc, double r);
 void Star(int id, AJC2DVEC vc, double r);
 void Star(int id, double xc, double yc, double r1, double r2, int nVertex, double rot, bool fInscribedLine);
 void Star(int id, AJC2DVEC vc, double r1, double r2, int nVertex, double rot, bool fInscribedLine);

引 数 : id - データ項目番号 (0 ~ 15)
 xc, yc - 星形の中心位置
 xr, yr - 星形の各軸の半径
 vc - 星形の中心位置 (ベクトル指定)
 r, r1 - 星形の外円の半径
 r2 - 星形の内円の半径 (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。
 nVertex, r2, rot, fInscribedLine 未指定時は、nVertex=5, r2=0, rot=0, fInscribedLine=false を仮定します。



戻り値 : なし

6.4.18. フィルタの設定(SetFilter)

形 式 : void SetFilter(int ix, bool fEnable);

引 数 : id - データ項目番号 (0 ~ 15)
 fEnable - フィルタ設定値

説 明 : 当該データ項目のフィィルタ (チェックボックス) を設定します。
 fEnable=True を指定すると当該データ項目は表示され、fEnable=false を指定すると非表示となります。

戻り値 : なし

6.4.19. フィルタの設定値の取得(GetFilter)

形 式 : `bool GetFilter (int ix);`

引 数 : `ix` - データ項目番号 (0～15)
`fEnable` - フィルタ設定値

説 明 : 当該データ項目のフィイルタ (チェックボックス) の設定値を取得します。

戻り値 : 当該データ項目のフィイルタ (チェックボックス) の設定値

6.4.20. 最大プロット数設定(SetMaxPlot)

形 式 : `SetMaxPlot(int ix, int n);`

引 数 : `ix` - データ項目番号 (0～15)
`n` - 最大プロット数

説 明 : プロットデータの最大保留数を設定します。
 投与したプロットデータが、この最大数を超える場合は、最古のデータが破棄されます。

戻り値 : なし

6.4.21. 最大プロット数取得(SetMaxPlot)

形 式 : `int GetMaxPlot(int ix);`

引 数 : `ix` - データ項目番号 (0～15)

説 明 : 現在設定されている、プロットデータの最大保留数を取得します。

戻り値 : プロットデータの最大保留数

6.4.22. データ項目の前面表示色設定(SetItemColorP)

形 式 : `void SetItemColorP(int ix, Color color);`

引 数 : `ix` - データ項目番号 (0～15)
`color` - 設定する表示色

説 明 : 当該データ項目の (原点より手前側の) 表示色を設定します。

戻り値 : なし

6.4.23. データ項目の前面表示色取得(GetItemColorP)

形 式 : `Color GetItemColorP(int ix);`

引 数 : `ix` - データ項目番号 (0～15)

説 明 : 当該データ項目の (原点より手前側の) 表示色を取得します。

戻り値 : なし

6.4.24. データ項目の後面表示色設定(SetItemColorN)

形 式 : void SetItemColorN(int ix, Color color);

引 数 : ix - データ項目番号 (0 ~ 15)
color - 設定する表示色

説 明 : 当該データ項目の (原点より向う側の) 表示色を設定します。

戻り値 : なし

6.4.25. データ項目の後面表示色取得(GetItemColorN)

形 式 : Color GetItemColorN(int ix);

引 数 : ix - データ項目番号 (0 ~ 15)

説 明 : 当該データ項目の (原点より向う側の) 表示色を取得します。

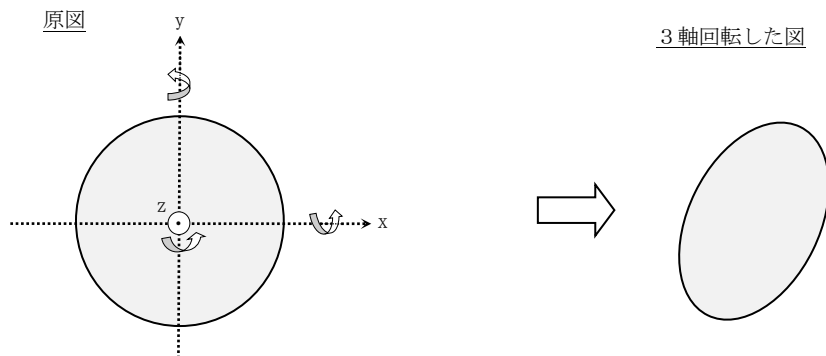
戻り値 : なし

6.4.26. 視点設定(SetAngle)

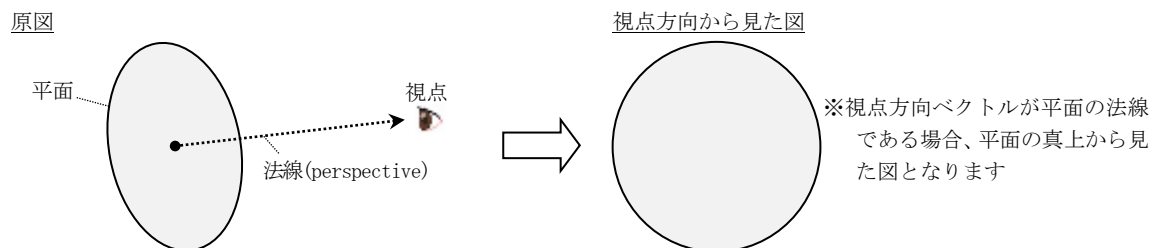
形 式 : void SetAngle(double rtx, double rty, double rtz);
void SetAngle(AJC3DVEC perspective);

引 数 : rtx, rty, rtz - 各軸回りの回転角度 [度]
perspective - 視点方向を示すベクトル

説 明 : rtx, rty, rtz を指定した場合は、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。



perspective を指定した場合は、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。



戻り値 : なし

6.4.27. 視点をX Y平面に設定(SetAngleXY)

形 式 : void SetAngleXY();

引 数 : なし

説 明 : X Y平面が見えるように視点を設定します。

戻り値 : なし

6.4.28. 視点をX Z平面に設定(SetAngleXZ)

形 式 : void SetAngleXZ();

引 数 : なし

説 明 : X Z平面が見えるように視点を設定します。

戻り値 : なし

6.4.29. 視点をY Z平面に設定(SetAngleYZ)

形 式 : void SetAngleYZ();

引 数 : なし

説 明 : Y Z平面が見えるように視点を設定します。

戻り値 : なし

6.4.30. 視点を3 Dイメージに設定(SetAngle3D)

形 式 : void SetAngle3D();

引 数 : なし

説 明 : X, Y, Z軸がすべて見えるように、視点を斜め方向に設定します。

戻り値 : なし

6.4.31. 視点を任意の平面に設定(SetAnyPlane)

形 式 : void SetAnyPlane(EAJCPLANEAXIS HoriAxis, EAJCPLANEAXIS VertAxis);

引 数 : HoriAxis - 横補方向に割り当てる軸と軸の方向 (昇順／降順)
VertAxis - 縦 " (")

説 明 : 横方向と縦方向に任意の軸を割り当てて、平面が見えるように視点を設定します。
横方向と縦方向で、同じ軸を割り当てることはできません。
2Dモードの場合、Z軸を割り当てることはできません。(ZP と ZM は使用不可)
HoriAxis と VertAxis には、以下のいずれかを指定します。

#	シンボル	内容	備考
1	XP	X軸を昇順で設定	横軸:右方向に値が増加 縦軸:上方向に値が増加
2	YP	Y軸を昇順で設定	
3	ZP	Z軸を昇順で設定	

#	シンボル	内容	備考
4	XM	X軸を降順で設定	横軸:右方向に値が減少 縦軸:上方向に値が減少
5	YM	Y軸を降順で設定	
6	ZM	Z軸を降順で設定	

戻り値 : なし

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

形 式 : `string GetDroppedFile();`

引 数 : なし

説 明 : OnFileDrop イベント発生時に、順次、ドロップされたファイルのパス名を取得します。
OnFileDrop イベントで通知された、ドロップファイル数の回数だけ本メソッドをこーするすることにより、ドロップされた全てのファイルを取得できます。

戻り値 : `≠""` : ドロップされたファイルパス名
`=""` : 終端(ドロップされたファイルパス名の取得は完了済である)

6.4.33. ドロップされたディレクトリ名取得 (GetDroppedDir)

形 式 : `string GetDroppedDir();`

引 数 : なし

説 明 : OnDirDrop イベント発生時に、順次、ドロップされたディレクトリのパス名を取得します。
OnDirDrop イベントで通知された、ドロップディレクトリ数の回数だけ本メソッドをこーすることにより、ドロップされた全てのディレクトリを取得できます。

戻り値 : `≠""` : ドロップされたディレクトリパス名
`=""` : 終端(ドロップされたディレクトリパス名の取得は完了済である)

6.4.34. タイトルテキスト表示 (SetTitleText)

形 式 : `void SetTitleText(string TitleText);`
`void SetTitleText(string TitleText, Color TextColor);`
`void SetTitleText(string TitleText, Color TextColor, Color BackColor);`
`void SetTitleText(string TitleText, Color TextColor, Color BackColor, Font TextFont);`

引 数 : TitleText - タイトルテキスト (null(あるいは空文字列(""))を指定した場合は非表示)
TextColor - テキスト描画色 (α値は無視されます)
BackColor - テキスト背景色 (α値は無視されます)
TextFont - テキストのフォント

説 明 : コントロールウインドの右上にタイトルテキストを表示します。
TitleText に null(あるいは空文字列(""))を指定した場合、タイトルテキストは非表示となります。

戻り値 : なし

6.4.35. 現在の設定値をプロファイルへ記録する (SaveToProfile)

形 式 : `void SaveToProfile (string section);`

引 数 : section - プロファイル内のセクション名

説 明 : 現在の設定値 (フィルタ設定やプロパティ) をプロファイルに記録します。

戻り値 : なし

6.4.36. 設定値をプロファイルから読み出す (LoadFromProfile)

形 式 : void LoadFromProfile (string section);

引 数 : section - プロファイル内のセクション名

説 明 : SaveToProfile()によりプロファイルに記録した設定値を読み出します。
読み出した内容は、そのままフィルタやプロパティに設定されます。

戻り値 : なし

6.4.37. ボーダー色で囲まれた部分の塗りつぶし(FillB) ・ ・ ・ 2D モード専用

形 式 : bool FillB(int idFill, int idBorder, double x, double y);

引 数 : 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

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 - エラー

6.4.38. 連続する白色部分の塗りつぶし(FillS) ・ ・ ・ 2D モード専用

形 式 : bool FillS(int idFill, double x, double y);

引 数 : 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

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 ・ ・ ・) の後に実行されます。

戻り値 : true - 成功
false - エラー

6.4.39. ピクセルの表示色取得 (GetPixel) ・ ・ ・ 2D モード専用

形 式 : Color GetPixel(double x, double y);

引 数 : x, y - ピクセルの位置

説 明 : x, y で指定した位置のピクセルの色を取得します。

戻り値 : ピクセルの色

6.4.40. 画面表示の停止／再開 (Pause)

形 式 : void Pause (bool fPause);

引 数 : 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 : なし

6.4.41. テキスト描画 (TextOut) - ピクセル位置指定

形 式 : int TextOut(int x, int y, string text);

引 数 : 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、「Txo.Right ± n」or「Txo.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、「Txo.Bottom ± n」or「Txo.Center ± n」を指定)
text - 描画するテキスト

説 明 : グラフウインド上にテキストを描画します。
描画するテキストには、エスケープシーケンスや制御文字を含めることができます。(「テキスト表示拡張機能」の章参照)
x = Txo.Right , y = Txo.Bottom は、テキストをウインドの右下隅に表示する位置となります。
x = Txo.Center, y = Txo.Center は、テキストをウインドの中央に表示する位置となります。

戻り値 : テキストキー

6.4.42. テキスト描画 (TextOut) - 2Dモード用

形 式 : int TextOut(AJC2DVEC v, string text);
int TextOut(EAJCTXOMD md, AJC2DVEC v, string text);
int TextOut(double x, double y, string text);
int TextOut(EAJCTXOMD md, double x, double y, string text);

引 数 : v, x, y - 描画座標位置
md - テキスト描画方法 (未指定時は、RIGHT)
text - 描画するテキスト

説 明 : 2Dグラフウインド上の、当該座標位置にテキストを描画します。
描画するテキストには、エスケープシーケンスや制御文字を含めることができます。(「テキスト表示拡張機能」の章参照)
md (テキスト描画方法) は以下のいずれかを指定します。

値	md	内容	表示イメージ (青点 (●) が描画位置)
0	RIGHT	描画位置の右側	● 表示テキスト
1	LEFT	描画位置の左側	表示テキスト ●
2	CENTER	描画位置に中央揃え	表示テキスト
3	BELLOW_RIGHT	描画位置の下右側	● 表示テキスト
4	BELLOW_LEFT	描画位置の下左側	表示テキスト ●
5	BELLOW_CENTER	描画位置の下に中央揃え	表示テキスト
6	ABOVE_RIGHT	描画位置の上右側	● 表示テキスト
7	ABOVE_LEFT	描画位置の上左側	表示テキスト ●
8	ABOVE_CENTER	描画位置の上に中央揃え	表示テキスト

戻り値 : テキストキー

6.4.43. テキスト描画 (TextOut) - 3Dモード用

形 式 : `int TextOut(AJC3DVEC v, string text);`
`int TextOut(EAJCTXOMD md, AJC3DVEC v, string text);`
`int TextOut(double x, double y, double z, string text);`
`int TextOut(EAJCTXOMD md, double x, double y, double z, string text);`

引 数 : `v, x, y, z` - 描画ピクセル位置
`md` - テキスト描画方法 (未指定時は、RIGHT)
`text` - 描画するテキスト

説 明 : 3Dグラフウインド上の、当該座標位置にテキストを描画します。
 描画するテキストには、エスケープシーケンスや制御文字を含めることができます。 (「テキスト表示拡張機能」の章参照)
`md` (テキスト描画方法) は「テキスト描画 - 2Dモード用」と同じです。

戻り値 : テキストキー

6.4.44. 図形描画データ破棄 (ClearShape)

形 式 : `void PurgeShape(int id);` // 指定 `id` の図形描画データ破棄 (旧名称「PurgeData」でも可)
`void PurgeShape();` // すべての図形描画データ破棄

引 数 : `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 : なし

6.4.45. プロットデータ破棄(PurgePlot)

形 式 : `void PurgePlot(int id);` // 指定 `id` のプロットデータ破棄
`void PurgePlot();` // すべてのプロットデータ破棄

引 数 : `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 : なし

6.4.46. テキスト描画データ破棄(PurgeText)

形 式 : `void PurgeText(int key);` // 指定 `key` の描画テキスト破棄
`void PurgeText();` // すべての描画テキスト破棄

引 数 : `key` - データ項目番号 (`TextOut()` の戻り値)

説 明 : `TextOut()` で描画したテキストを破棄します。

戻り値 : なし

6.4.47. 全ての描画データ破棄(PurgeData)

形 式 : void PurgeData(int id); // 指定 id の描画データ破棄
void PurgeData(); // すべての描画データ破棄

引 数 : なし

説 明 : 描画データ (図形、プロット) を破棄します。

戻り値 : なし

6.4.48. 全てのデータ破棄(Purge)

形 式 : void Purge ();

引 数 : なし

説 明 : すべてのデータ (図形、プロット, テキスト) を破棄します。

戻り値 : なし

6.5. イベント

2D/3Dグラフ描画コントロールのイベント一覧を以下に示します。

#	イベント名	内容
1	OnFileDrop	ファイルドロップ通知
2	OnDirDrop	ディレクトリドロップ通知
3	OnRClick	右クリックされたことを通知します
4	OnPltLst	Shift+左クリックでクリックしたプロット点情報の通知

6.5.1. ファイルドロップ通知 (OnFileDrop)

形 式 : void OnFileDrop(object sender, VthArgFileDrop e);

パラメタ : int e.n - ドロップされたファイルの個数

説 明 : コントロールへ、ファイルがドロップされたことを通知します。
GetDroppedFile() メソッドにより、ドロップされたファイルのパス名を取得することができます。

戻り値 : なし

6.5.2. ディレクトリドロップ通知 (OnDirDrop)

形 式 : void OnDirDrop(object sender, VthArgDirDrop e);

パラメタ : int e.n - ドロップされたディレクトリの個数

説 明 : コントロールへ、ディレクトリがドロップされたことを通知します。
GetDroppedDir() メソッドにより、ドロップされたディレクトリのパス名を取得することができます。

戻り値 : なし

6.5.3. 右クリック通知 (OnRClick)

形 式 : void OnRClick (object sender, G3dArgRClick e);

パラメタ : int e.x - 右クリックしたX位置 (コントロールの左上が原点)
int e.y - " Y (")
bool e.shift - Shift キーの押下状態
bool e.ctrl - Ctrl キーの押下状態

説 明 : コントロール上で右クリックされたことを通知します。
Shift キーとCtrl キーが押されていない状態で右クリックした場合、通常はポップアップメニューが表示されます。
但し、EnablePopupMenu プロパティが false (右クリックによるポップアップメニュー禁止) に設定されている場合は、ポップアップメニューが表示されず、右クリック通知イベントが発生します。
Shift キーかCtrl キーが押されている状態で右クリックした場合は、EnablePopupMenu プロパティに関係なく常に右クリック通知イベントが発生します。

戻り値 : なし

63

パラメタ :

- int e.max - 実際のプロット点の個数 (64 を超える点については、このパラメタで実際の個数を通知)
- int e.num - 通知するプロット点の個数 (通常は1だが、重複している場合は2~64)
- int e.p[0~63].id - プロット点のデータ項目番号 (0~15)
- int e.p[0~63].ix - プロットデータの場合、バッファ上でのプロット点位置 (0 は最古の点を意味します)
描画データ (ピクセル描画) の場合は -1 固定
- AJC3DVEC[] e.p[0~63].v - プロット点の座標値の配列 (e.num が配列の要素数である)

戻り値 : なし

6.6. サンプルプログラム

6.6.1. Sil_3DGraphic1 (3D グラフィック)

このサンプルプログラムは、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、球の表面を表す座標とします。

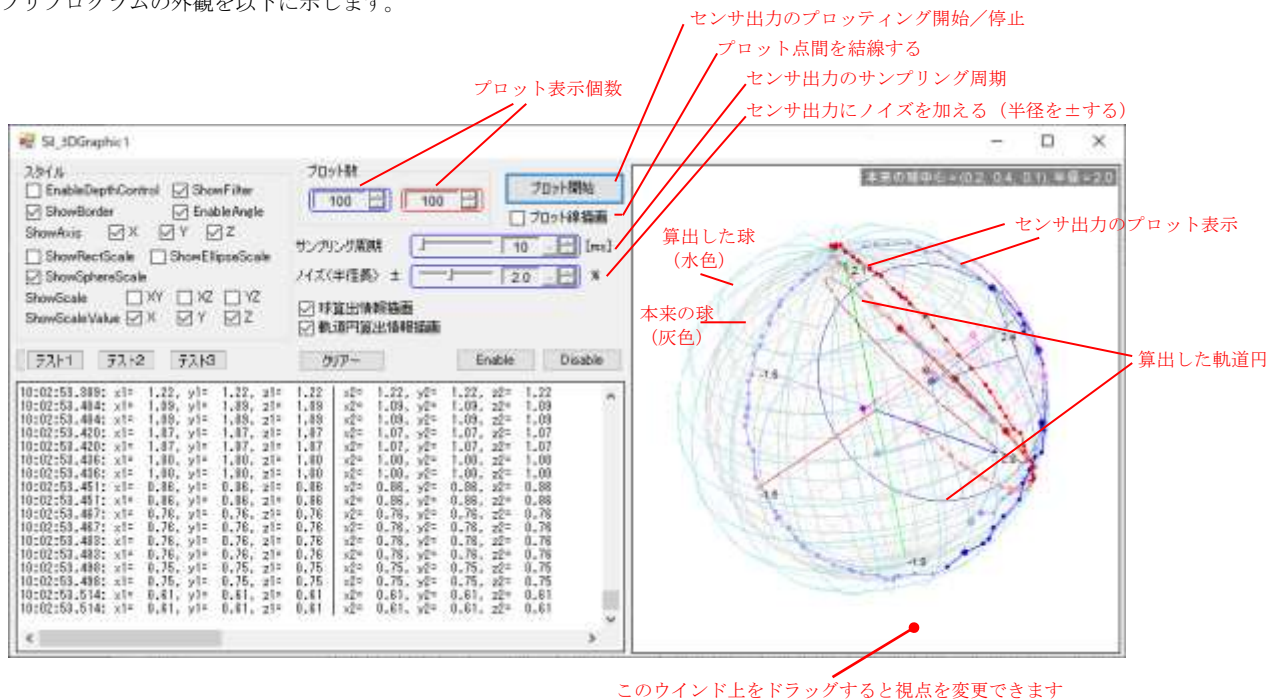
センサ出力は、本来の球(中心=(0.2, 0.4, 0.1), 半径=2.0)にノイズを加えて擬似的に生成します。

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。

- ・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)
- ・球の算出源となった情報を描画 (2つの内接円)
- ・プロット点の軌道を算出して、軌道円を描画 (3点から円を算出)
- ・軌道円の算出源となった情報を描画 (三角形)

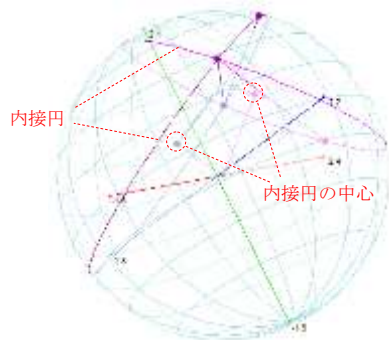
サンプルプログラムの外観を以下に示します。



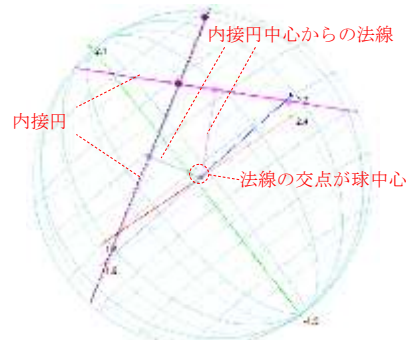
球の算出

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



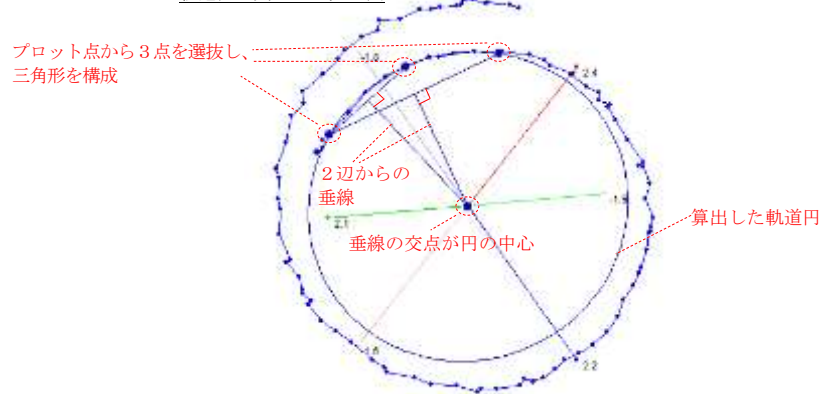
内接円の真横から見た図



軌道円の算出

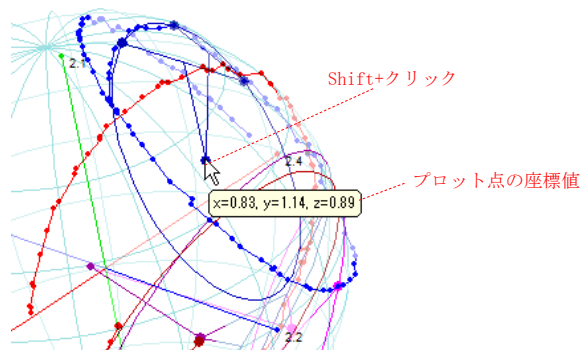
選抜・収集した3点から三角形を構成します。
三角形の2辺の中点から、同一平面上で垂線を引きます。
2つの垂線の交点が、軌道円の中心となります。
以下は、余計な描画を消して、(片方の)軌道円だけをクローズアップし、視点を変えた図です。

軌道円の真上から見た図



座標の表示

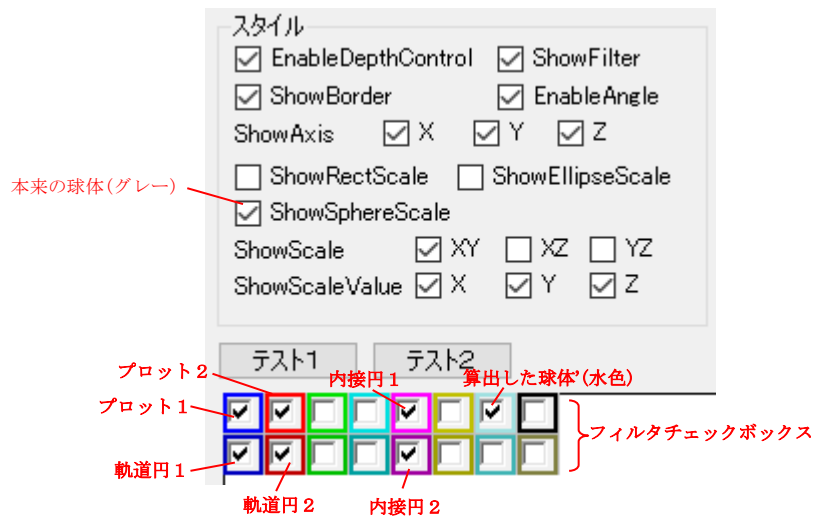
Shift キーを押しながら、3Dグラフィックウインド上のプロット点を左クリックすると当該プロット点の座標値が表示されます。



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、「ShowSphereScale」のチェックを外します。



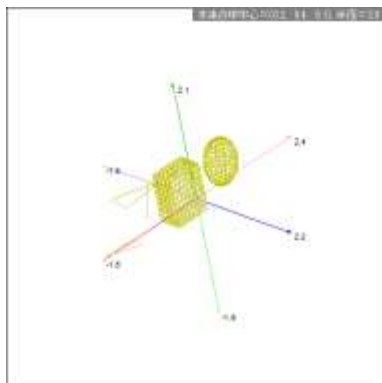
テスト1ボタンと、テスト2ボタンは、単に固定の図形を描画します。

テスト1ボタンは、3D空間上に図形を描画します。

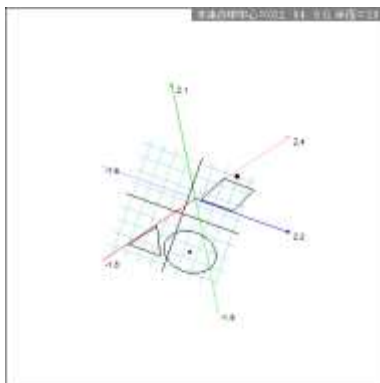
テスト2ボタンは、3D空間上の平面に図形を描画します。

テスト3ボタンは、座標=(1, 1, 1)の位置に、ピクセルとテキストを表示します。

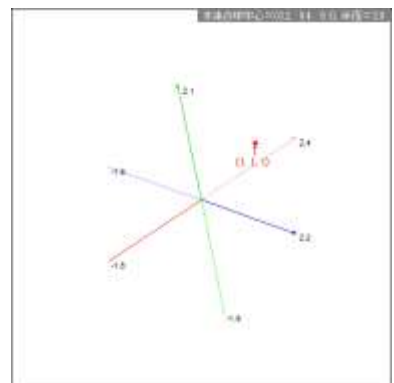
テスト1ボタンによる図形描画



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 : using System;
2 : using System.Collections.Generic;
3 : using System.ComponentModel;
4 : using System.Data;
5 : using System.Drawing;
6 : using System.Linq;
7 : using System.Text;
8 : using System.Windows.Forms;
9 : using CAjrCustCtrl;
10 :
11 : namespace Sil_3DGraphic1
12 : {
13 :     public partial class Form1 : Form
14 :     {
15 :         // 規定球の中心と半径
16 :         const double CX = 0.2;
17 :         const double CY = 0.4;
18 :         const double CZ = 0.1;
19 :         const double R = 2.0;
20 :
21 :         // 描画色 I D
22 :         const int ID_PLOT1 = 0; // プロット点1
23 :         const int ID_PLOT2 = 1; // プロット点2
24 :         const int ID_BALL = 6; // 算出した球
25 :         const int ID_INS1 = 4; // 球の内接円1

```

```

26 :      const int      ID_INS2      = 12;      // 球の内接円2
27 :
28 :      const int      ID_TRACK1    = 8;      // 軌道円1
29 :      const int      ID_TRACK2    = 9;      // 軌道円2
30 :
31 :      const int      ID_2DGRID    = 6;      // 2D方眼
32 :      const int      ID_2DSHAPE   = 7;      // 2D図形
33 :      const int      ID_3DSHAPE   = 5;      // 2D図形
34 :      const int      ID_TXTPOINT  = 1;      // テキスト描画ポイント
35 :
36 :      // プロットデータ収集バッファ情報
37 :      const int      MAX_PLOT     = 4;
38 :      struct PLOTINFO {
39 :      public int      id1, id2, id3; // 描画色 I D
40 :      public double   dist;        // プロット間の最低距離
41 :      public int      num;         // プロット収集個数
42 :      public int      cnt;         // プロットデータカウンタ
43 :      public AJC3DVEC[] plt;      // プロット点収集バッファ
44 :      public AJC3DVEC cent;       // 中心
45 :      public double   radius;     // 半径
46 :      public AJC3DVEC vec;        // 法線
47 :      }
48 :      PLOTINFO PltBall;          // 球算出用プロットバッファ
49 :      PLOTINFO[] PltCir;         // 軌道円算出用プロットバッファ
50 :
51 :      // ワーク
52 :      AJCSPD_PARAM[] m_prm;      // プロット点演算パラメタ
53 :      CAjrSphereData[] m_spd;    // プロット点演算オブジェクト
54 :      Size            m_MinSize;  // ウィンド最小サイズ
55 :      bool            m_fPlot = false; // プロット中フラグ
56 :      string          m_TipText;  // チップテキスト
57 :
58 :      private TipCbKNeedText m_cbNeedText; // チップテキスト取得コールバック
59 :
60 :      public Form1()
61 :      {
62 :          InitializeComponent();
63 :      }
64 :      //----- フォームロード -----//
65 :      private void Form1_Load(object sender, EventArgs e)
66 :      {
67 :          //----- スタイルチェックボックスのツールチップ設定 -----//
68 :          SAjrTip.Add(chkEnableDepthControl, "遠近表現 (原点の手前側と向う側の色分け) 許可");
69 :          SAjrTip.Add(chkShowFilter, "フィルタチェックボックス許可");
70 :          SAjrTip.Add(chkShowBorder, "外枠表示");
71 :          SAjrTip.Add(chkEnableAngle, "ドラッグによる回転許可");
72 :          SAjrTip.Add(chkShowAxisX, "X軸表示");
73 :          SAjrTip.Add(chkShowAxisY, "Y軸表示");
74 :          SAjrTip.Add(chkShowAxisZ, "Z軸表示");
75 :          SAjrTip.Add(chkShowRectScale, "方眼スケール表示 (表示面は ShowScale XY/XZ/YZ で指定)");
76 :          SAjrTip.Add(chkShowEllipseScale, "同心円スケール表示 (表示面は ShowScale XY/XZ/YZ で指定)");
77 :          SAjrTip.Add(chkShowSphereScale, "球体スケール表示");
78 :          SAjrTip.Add(chkShowScaleXY, "XY平面にスケール表示 (スケールは ShowRectScale, ShowEllipseScale で指定)");
79 :          SAjrTip.Add(chkShowScaleXZ, "XZ平面にスケール表示 (スケールは ShowRectScale, ShowEllipseScale で指定)");
80 :          SAjrTip.Add(chkShowScaleYZ, "YZ平面にスケール表示 (スケールは ShowRectScale, ShowEllipseScale で指定)");
81 :          SAjrTip.Add(chkShowScaleValueX, "X軸の値表示");
82 :          SAjrTip.Add(chkShowScaleValueY, "Y軸の値表示");
83 :          SAjrTip.Add(chkShowScaleValueZ, "Z軸の値表示");
84 :          //----- ボタンチップテキスト設定 -----//
85 :          SAjrTip.Add(btnTest1, "3D図形描画/消去");
86 :          SAjrTip.Add(btnTest2, "2D図形描画/消去");
87 :          SAjrTip.Add(btnTest3, "テキスト描画/消去");
88 :          //----- チップテキスト表示設定 (状況依存で球情報表示) -----//
89 :          m_cbNeedText = new TipCbKNeedText(cbNeedText);
90 :          SAjrTip.Add(g3d, "", 1000, 3000);
91 :          SAjrTip.SetCallBack(g3d, (IntPtr)0, m_cbNeedText);
92 :          //----- ウィンド最小サイズ設定 -----//
93 :          m_MinSize = this.Size;
94 :          //----- プロット点演算パラメタ 初期化 -----//
95 :          m_prm = new AJCSPD_PARAM[2];
96 :          m_prm[0] = new AJCSPD_PARAM() {cent_x = CX, cent_y = CY, cent_z = CZ, radius = R, noise = 3.0, xrot = 0.5, yrot = 0.8, pitch = 9.0};
97 :          m_prm[1] = new AJCSPD_PARAM() {cent_x = CX, cent_y = CY, cent_z = CZ, radius = R, noise = 3.0, xrot = 0.6, yrot = 0.7, pitch = 8.5};
98 :          //----- プロット点演算オブジェクト -----//
99 :          m_spd = new CAjrSphereData[2];
100 :          m_spd[0] = spd1;
101 :          m_spd[1] = spd2;
102 :          //----- 球算出用プロットバッファ初期化 -----//
103 :          PltBall = new PLOTINFO() {id1 = ID_INS1, id2 = ID_INS2, id3 = ID_BALL, dist = R * 0.6, num = 4};
104 :          PltBall.plt = new AJC3DVEC[MAX_PLOT];
105 :          PltCir = new PLOTINFO[2];
106 :          PltCir[0] = new PLOTINFO() {id1 = ID_TRACK1, dist = R * 0.4, num = 3};
107 :          PltCir[1] = new PLOTINFO() {id1 = ID_TRACK2, dist = R * 0.4, num = 3};
108 :          PltCir[0].plt = new AJC3DVEC[MAX_PLOT];
109 :          PltCir[1].plt = new AJC3DVEC[MAX_PLOT];
110 :          //----- 演算球表示色設定 -----//
111 :          g3d.SetItemColorP(ID_BALL, Color.FromArgb(0xA0E0E0));
112 :          g3d.SetItemColorN(ID_BALL, Color.FromArgb(0xD0F0F0));
113 :          //----- 設定値ロード -----//
114 :          SAjrReg.LoadAllCtrls(this);
115 :          //----- グラフレンジ設定 -----//

```

```

116 :         g3d.CenterX = CX; g3d.CenterY = CY; g3d.CenterZ = CZ;
117 :         g3d.RadiusX = g3d.RadiusY = g3d.RadiusZ = R;
118 :     }
119 :     //----- フォームクローズ -----//
120 :     private void Form1_FormClosed(object sender, FormClosedEventArgs e)
121 :     {
122 :         // 設定値セーブ
123 :         SAjrReg.SaveAllCtrls(this);
124 :     }
125 :     //----- フォームサイズ変更 -----//
126 :     private void Form1_Resize(object sender, EventArgs e)
127 :     {
128 :         if (this.Width < m_MinSize.Width) this.Width = m_MinSize.Width;
129 :         if (this.Height < m_MinSize.Height) this.Height = m_MinSize.Height;
130 :     }
131 :     //----- コントロールからのイベント -----//
132 :     // スタイル・チェックボックス群
133 :     private void chkEnableDepthControl_CheckedChanged(object sender, EventArgs e) {g3d.EnableDepthControl = chkEnableDepthControl.Checked;}
134 :     private void chkShowFilter_CheckedChanged(object sender, EventArgs e) {g3d.ShowFilter = chkShowFilter.Checked;}
135 :     private void chkShowBorder_CheckedChanged(object sender, EventArgs e) {g3d.ShowBorder = chkShowBorder.Checked;}
136 :     private void chkEnableAngle_CheckedChanged(object sender, EventArgs e) {g3d.EnableAngle = chkEnableAngle.Checked;}
137 :     private void chkShowAxisX_CheckedChanged(object sender, EventArgs e) {g3d.ShowAxisX = chkShowAxisX.Checked;}
138 :     private void chkShowAxisY_CheckedChanged(object sender, EventArgs e) {g3d.ShowAxisY = chkShowAxisY.Checked;}
139 :     private void chkShowAxisZ_CheckedChanged(object sender, EventArgs e) {g3d.ShowAxisZ = chkShowAxisZ.Checked;}
140 :     private void chkShowRectScale_CheckedChanged(object sender, EventArgs e) {g3d.ShowRectScale = chkShowRectScale.Checked;}
141 :     private void chkShowEllipseScale_CheckedChanged(object sender, EventArgs e) {g3d.ShowEllipseScale = chkShowEllipseScale.Checked;}
142 :     private void chkShowSphereScale_CheckedChanged(object sender, EventArgs e) {g3d.ShowSphereScale = chkShowSphereScale.Checked;}
143 :     private void chkShowScaleXY_CheckedChanged(object sender, EventArgs e) {g3d.ShowScaleXY = chkShowScaleXY.Checked;}
144 :     private void chkShowScaleXZ_CheckedChanged(object sender, EventArgs e) {g3d.ShowScaleXZ = chkShowScaleXZ.Checked;}
145 :     private void chkShowScaleYZ_CheckedChanged(object sender, EventArgs e) {g3d.ShowScaleYZ = chkShowScaleYZ.Checked;}
146 :     private void chkShowScaleValueX_CheckedChanged(object sender, EventArgs e) {g3d.ShowScaleValueX = chkShowScaleValueX.Checked;}
147 :     private void chkShowScaleValueY_CheckedChanged(object sender, EventArgs e) {g3d.ShowScaleValueY = chkShowScaleValueY.Checked;}
148 :     private void chkShowScaleValueZ_CheckedChanged(object sender, EventArgs e) {g3d.ShowScaleValueZ = chkShowScaleValueZ.Checked;}
149 :     // プロット線描画チェックボックス
150 :     private void chkPlotLine_CheckedChanged(object sender, EventArgs e) {g3d.PlotLine = chkPlotLine.Checked;}
151 :     // 球算出情報描画チェックボックス
152 :     private void chkBallInfo_CheckedChanged(object sender, EventArgs e) {}
153 :     // 軌道円算出情報描画チェックボックス
154 :     private void chkCirInfo_CheckedChanged(object sender, EventArgs e) {}
155 :     // プロット数0
156 :     private void inpPlotNum0_OnNtcIntValue(object sender, IvArgNtcIntValue e) {g3d.SetMaxPlot(0, e.value);}
157 :     // プロット数1
158 :     private void inpPlotNum1_OnNtcIntValue(object sender, IvArgNtcIntValue e) {g3d.SetMaxPlot(1, e.value);}
159 :     // サンプリング周期
160 :     private void inpPeriod_OnNtcIntValue(object sender, IvArgNtcIntValue e) {tim.Interval = e.value;}
161 :     // ノイズ
162 :     private void inpNoise_OnNtcRealValue(object sender, IvArgNtcRealValue e) {m_prm[0].noise = e.value; m_spd[0].SetParam(m_prm[0]);
163 :                                     m_prm[1].noise = e.value; m_spd[1].SetParam(m_prm[1]);}
164 :     // クリアボタン
165 :     private void btnClear_Click(object sender, EventArgs e) {
166 :         g3d.Purge();
167 :         m_fTest1 = m_fTest2 = m_fTest3 = false;
168 :     }
169 :     // Enable ボタン
170 :     private void btnEnable_Click(object sender, EventArgs e) {g3d.Enabled = true;}
171 :     // Disable ボタン
172 :     private void btnDisable_Click(object sender, EventArgs e) {g3d.Enabled = false;}
173 :     // テスト1ボタン
174 :     bool m_fTest1 = false;
175 :     private void btnTest1_Click(object sender, EventArgs e)
176 :     {
177 :         if (!m_fTest1) {
178 :             g3d.Cube(ID_3DSHAPE, +0.1, +0.2, +0.3, 0.2, 0.3, 0.4, 8);
179 :             g3d.Sphere(ID_3DSHAPE, +0.7, +0.6, +0.8, 0.2, 0.3, 0.4, 12, 8);
180 :             g3d.MoveTo(ID_3DSHAPE, -0.6, 0.2, 0.3);
181 :             g3d.LineTo(ID_3DSHAPE, -0.3, -0.2, -0.1);
182 :             g3d.Triangle(ID_3DSHAPE, -0.8, -0.7, -0.6, -0.4, -0.3, -0.5, -0.4, -0.5, -0.3);
183 :             g3d.Square(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);
184 :             m_fTest1 = true;
185 :         }
186 :         else {
187 :             g3d.PurgeData(ID_3DSHAPE);
188 :             m_fTest1 = false;
189 :         }
190 :     }
191 :     // テスト2ボタン
192 :     bool m_fTest2 = false;
193 :     private void btnTest2_Click(object sender, EventArgs e)
194 :     {
195 :         double x, y;
196 :
197 :         if (!m_fTest2) {
198 :             for (int i = 0; i < 16; i++)
199 :             {
200 :                 g3d.DefPlane(i, 0.2, 0.1, 0.0, 0.6738, -0.5510, 0.4924, 0.1, 0.0, 0.0);
201 :             }
202 :             for (x = -0.8; x < -0.01; x += 0.2) g3d.Line(ID_2DGRID, x, -0.9, x, +0.9);
203 :             for (x = 0.2; x < 0.89; x += 0.2) g3d.Line(ID_2DGRID, x, -0.9, x, +0.9);
204 :             for (y = -0.8; y < -0.01; y += 0.2) g3d.Line(ID_2DGRID, -0.9, y, +0.9, y);
205 :             for (y = 0.2; y < 0.89; y += 0.2) g3d.Line(ID_2DGRID, -0.9, y, +0.9, y);

```

```

206 :         g3d.Line (ID_2DSHAPE, -0.9, 0.0, +0.9, 0.0);
207 :         g3d.Line (ID_2DSHAPE, 0.0, -0.9, 0.0, +0.9);
208 :         g3d.Triangle(ID_2DSHAPE, -0.6, -0.7, -0.3, -0.3, -0.1, -0.7);
209 :         g3d.Square (ID_2DSHAPE, 0.2, 0.3, 0.4, 0.7, 0.9, 0.7, 0.7, 0.3);
210 :         g3d.Pixel (ID_2DSHAPE, 0.3, -0.5, 2);
211 :         g3d.Ellipse (ID_2DSHAPE, 0.3, -0.5, 0.4, 0.3);
212 :         g3d.Pixel (ID_2DSHAPE, 0.6, 0.8, 3);
213 :         m_fTest2 = true;
214 :     }
215 :     else {
216 :         g3d.PurgeData(ID_2DGRID);
217 :         g3d.PurgeData(ID_2DSHAPE);
218 :         m_fTest2 = false;
219 :     }
220 : }
221 : // テスト3 ボタン
222 : bool m_fTest3 = false;
223 : private void btnTest3_Click(object sender, EventArgs e)
224 : {
225 :     if (!m_fTest3) {
226 :         g3d.Pixel(ID_TXTPOINT, 1, 1, 1, 3);
227 :         g3d.TextOut(EAJCTXOMD.BELLOW_CENTER, 1, 1, 1, "¥x1B[1:3]lm ↑ ¥n(1, 1, 1)");
228 :         m_fTest3 = true;
229 :     }
230 :     else {
231 :         g3d.PurgeShape(ID_TXTPOINT);
232 :         g3d.PurgeText();
233 :         m_fTest3 = false;
234 :     }
235 : }
236 : }
237 : // プロット開始/停止ボタン
238 : private void btnStartStop_Click(object sender, EventArgs e)
239 : {
240 :     if (m_fPlot) {
241 :         m_fPlot = false;
242 :         tim.Stop();
243 :         btnStartStop.Text = "プロット開始";
244 :     }
245 :     else {
246 :         m_fPlot = true;
247 :         tim.Start();
248 :         btnStartStop.Text = "プロット停止";
249 :     }
250 : }
251 : // プロット周期タイマ
252 : private void tim_Tick(object sender, EventArgs e)
253 : {
254 :     AJC3DVEC[] v = new AJC3DVEC[2];
255 :     for (int id = 0; id < 2; id++) {
256 :         // ランダムなプロットデータ生成
257 :         v[id] = m_spd[id].Calc();
258 :         // プロットデータ投与
259 :         g3d.PutData(id, v[id]);
260 :         // 球データ収集と算出した球の表示
261 :         BallCorrectAndShow(v[id], ref PltBall);
262 :         // 軌道円データ収集と算出した軌道円の表示
263 :         CirCorrectAndShow(v[id], ref PltCir[id]);
264 :         // ログ表示
265 :         vth.PrintTimeStamp();
266 :         vth.PutText(string.Format("x1={0,6:f2}, y1={0,6:f2}, z1={0,6:f2} | x2={0,6:f2}, y2={0,6:f2}, z2={0,6:f2} ¥n",
267 :             v[0].x, v[0].y, v[0].z, v[1].x, v[1].y, v[1].z));
268 :     }
269 : }
270 : //----- 動的ツールチップ取得用コールバック -----//
271 : private string cbNeedText(IntPtr Handle, IntPtr cbp)
272 : {
273 :     return m_TipText;
274 : }
275 : //----- 3Dグラフ - 右クリック通知 -----//
276 : private void g3d_OnRClick(object sender, G3dArgRClick e)
277 : {
278 :     SAjrTip.ShowCenter(g3d, (e.shift ? "Shift + " : "") + (e.ctrl ? "Ctrl + " : "") +
279 :         "右クリック発生(x = " + e.x.ToString() + ", y = " + e.y.ToString() + ")");
280 : }
281 : //----- 3Dグラフ - ディレクトリドロップ通知 -----//
282 : private void g3d_OnDirDrop(object sender, G3dArgDirDrop e)
283 : {
284 :     string txt = "— Dir dropped —";
285 :     for (int i = 0; i < e.n; i++) {
286 :         txt = txt + "¥n" + g3d.GetDroppedDir();
287 :     }
288 :     SAjrTip.ShowCenter(g3d, txt);
289 : }
290 : //----- 3Dグラフ - ファイルドロップ通知 -----//
291 : private void g3d_OnFileDrop(object sender, G3dArgFileDrop e)
292 : {
293 :     string txt = "— File dropped —";
294 :     for (int i = 0; i < e.n; i++) {
295 :         txt = txt + "¥n" + g3d.GetDroppedFile();

```

```

296 :     }
297 :     SAjrTip.ShowCenter(g3d, txt);
298 : }
299 : //----- 3Dグラフ - プロットリスト通知 -----//
300 : private void g3d_OnPltLst(object sender, G3dArgPltLst e)
301 : {
302 :     Point pt = Control.MousePosition;
303 :     SAjrTip.Show(pt.X, pt.Y + 16, "x=" + e.p[0].v.x.ToString("0.00") + ", " +
304 :     "y=" + e.p[0].v.y.ToString("0.00") + ", " +
305 :     "z=" + e.p[0].v.z.ToString("0.00"), 3000);
306 : }
307 : //-----//
308 : // 球算出用プロット点の収集と描画 //
309 : //-----//
310 : void BallCorrectAndShow(AJC3DVEC Vec, ref PLOTINFO Buf)
311 : {
312 :     int i;
313 :
314 :     // プロット間の最低距離チェック
315 :     for (i = 0; i < Buf.cnt; i++) {
316 :         if (SAjrMath.V3dDistP2P(Vec, Buf.plt[i]) < Buf.dist) {
317 :             break;
318 :         }
319 :     }
320 :     // プロット点の収集
321 :     if (i >= Buf.cnt) {
322 :         Buf.plt[i] = Vec;
323 :         Buf.cnt++;
324 :     }
325 :     // 球の算出と表示
326 :     if (Buf.cnt >= Buf.num) {
327 :         AJC3DSPHINFO sph;
328 :         if ((Buf.radius = SAjrMath.V3dCalcSphere(Buf.plt[0], Buf.plt[1], Buf.plt[2], Buf.plt[3], out Buf.cent, out sph)) != -1) {
329 :             // 2つの平面円の角度が30度以上ならば球描画
330 :             double t = SAjrMath.V3dTheta(sph.cif1.lvc.v, sph.cif2.lvc.v);
331 :             if (t > 30 && t < 180 - 30) {
332 :                 // チップテキスト (球情報) 設定
333 :                 m_TipText = "球算出情報: \n" +
334 :                 "  中心 = " + Buf.cent.x.ToString("0.00") + ", " +
335 :                 Buf.cent.y.ToString("0.00") + ", " +
336 :                 Buf.cent.z.ToString("0.00") + "\n" +
337 :                 "  半径 = " + Buf.radius.ToString("0.00");
338 :                 // 前回の球表示をクリアー
339 :                 g3d.PurgeData(Buf.id3);
340 :                 // 球表示
341 :                 g3d.Sphere(Buf.id3, Buf.cent.x, Buf.cent.y, Buf.cent.z, Buf.radius, Buf.radius, Buf.radius, 8, 8);
342 :                 // 前回の内接円表示をクリアー
343 :                 g3d.PurgeData(Buf.id1);
344 :                 g3d.PurgeData(Buf.id2);
345 :                 // 2つの内接円描画
346 :                 if (chkBallInfo.Checked) {
347 :                     // 球中心点描画
348 :                     g3d.Pixel(Buf.id1, Buf.cent, 3);
349 :                     g3d.Pixel(Buf.id2, Buf.cent, 3);
350 :                     // 球に内接する2つの円の情報を描画
351 :                     DrawInscribedCircle(sph.cif1, Buf.id1);
352 :                     DrawInscribedCircle(sph.cif2, Buf.id2);
353 :                     // 内接円中心から球中心への垂線
354 :                     g3d.Line(Buf.id1, sph.cif1.lvc.p, Buf.cent);
355 :                     g3d.Line(Buf.id2, sph.cif2.lvc.p, Buf.cent);
356 :                 }
357 :             }
358 :         }
359 :         // 球の情報クリアー
360 :         Buf.cnt = 0;
361 :     }
362 : }
363 : //-----//
364 : // 軌道円算出用プロット点の収集と描画 //
365 : //-----//
366 : void CirCorrectAndShow(AJC3DVEC Vec, ref PLOTINFO Buf)
367 : {
368 :     int i;
369 :
370 :     // プロット間の最低距離チェック
371 :     for (i = 0; i < Buf.cnt; i++) {
372 :         if (SAjrMath.V3dDistP2P(Vec, Buf.plt[i]) < Buf.dist) {
373 :             break;
374 :         }
375 :     }
376 :     // プロット点の収集
377 :     if (i >= Buf.cnt) {
378 :         Buf.plt[i] = Vec;
379 :         Buf.cnt++;
380 :     }
381 :
382 :     // 軌道円の算出と表示
383 :     if (Buf.cnt >= Buf.num) {
384 :         AJC3DCIRINFO cif;
385 :         if ((Buf.radius = SAjrMath.V3dCalcCircle(Buf.plt[0], Buf.plt[1], Buf.plt[2], out Buf.cent, out Buf.vec, out cif)) != -1) {

```

```

386 :         if (chkCirInfo.Checked) {
387 :             // 前の軌道円描画クリアー
388 :             g3d.PurgeData(Buf.id1);
389 :             // 3点描画
390 :             g3d.Pixel(Buf.id1, Buf.plt[0], 4);
391 :             g3d.Pixel(Buf.id1, Buf.plt[1], 4);
392 :             g3d.Pixel(Buf.id1, Buf.plt[2], 4);
393 :             // 内接円描画
394 :             DrawInscribedCircle(cif, Buf.id1);
395 :         }
396 :     }
397 :     // 軌道円の情報クリアー
398 :     Buf.cnt = 0;
399 : }
400 : }
401 : //-----//
402 : // 球算出の内接円／軌道円の描画 //
403 : //-----//
404 : void DrawInscribedCircle(AJC3DCIRINFO Cif, int id)
405 : {
406 :     // 円に内接する2つの直線と端点
407 :     g3d.Line (id, Cif.lt1.p1, Cif.lt1.p2);
408 :     g3d.Line (id, Cif.lt2.p1, Cif.lt2.p2);
409 :     g3d.Pixel(id, Cif.lt1.p1, 4);
410 :     g3d.Pixel(id, Cif.lt1.p2, 4);
411 :     g3d.Pixel(id, Cif.lt2.p1, 4);
412 :     g3d.Pixel(id, Cif.lt2.p2, 4);
413 :     // 内接線中点からの垂線
414 :     g3d.Line (id, Cif.lc1.p1, Cif.lc1.p2);
415 :     g3d.Line (id, Cif.lc2.p1, Cif.lc2.p2);
416 :     // 内接円と内接円の中心点
417 :     g3d.DefPlane(id, Cif.lvc, new AJC3DVEC(0, 0, 0));
418 :     g3d.Ellipse (id, 0, 0, Cif.cr, Cif.cr);
419 :     g3d.Pixel (id, 0, 0, 4);
420 : }
421 : }
422 : }

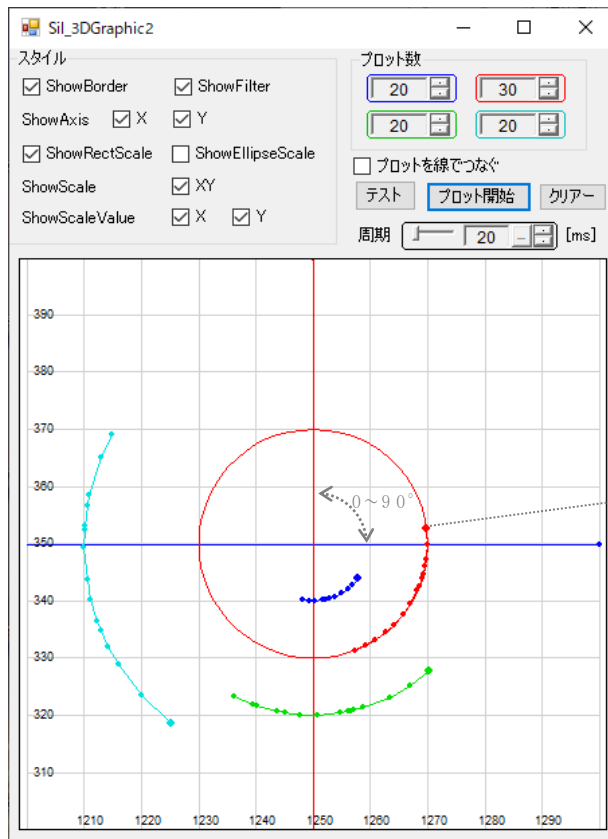
```

6.6.2. Sil_3DGraphic2 (2D グラフィック)

サンプルプログラム2では、2Dグラフィックモードで、データのプロットイ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円を表示します。



プロット点が0～90°以内
である場合、その軌道円を表示

```

1 : using System;
2 : using System.Collections.Generic;
3 : using System.ComponentModel;
4 : using System.Data;
5 : using System.Drawing;
6 : using System.Linq;
7 : using System.Text;
8 : using System.Windows.Forms;
9 : using CAjrCustCtrl;
10 :
11 : namespace Sil_3DGraphic2
12 : {
13 :     public partial class Form1 : Form
14 :     {
15 :         const double X1 = 1200.0;
16 :         const double X2 = 1300.0;
17 :         const double Y1 = 300.0;
18 :         const double Y2 = 400.0;
19 :         const double XC = ((X1 + X2) / 2.0);
20 :         const double YC = ((Y1 + Y2) / 2.0);
21 :         const int RAND_MAX = 10000;
22 :
23 :         Size m_MinSize; // ウインド最小サイズ
24 :         bool m_fPlot = false; // プロット中フラグ
25 :         bool m_fTest = false; // テストボタンで描画中
26 :         double[] m_theta = new double[4] {0, 0, 0, 0};
27 :         Random cRandom = new System.Random();
28 :
29 :         public Form1()

```



```

30 : {
31 :     InitializeComponent();
32 : }
33 : //----- フォームロード -----//
34 : private void Form1_Load(object sender, EventArgs e)
35 : {
36 :     //----- スタイルチェックボックスのツールチップ設定 -----//
37 :     SAjrTip.Add(chkShowFilter, "フィルタチェックボックス許可");
38 :     SAjrTip.Add(chkShowBorder, "外枠表示");
39 :     SAjrTip.Add(chkShowAxisX, "X軸表示");
40 :     SAjrTip.Add(chkShowAxisY, "Y軸表示");
41 :     SAjrTip.Add(chkShowRectScale, "方眼スケール表示 (表示面は ShowScale XY で指定)");
42 :     SAjrTip.Add(chkShowEllipseScale, "同心円スケール表示 (表示面は ShowScale XY で指定)");
43 :     SAjrTip.Add(chkShowScaleXY, "X Y 平面にスケール表示 (スケールは ShowRectScale, ShowEllipseScale で指定)");
44 :     SAjrTip.Add(chkShowScaleValueX, "X軸の値表示");
45 :     SAjrTip.Add(chkShowScaleValueY, "Y軸の値表示");
46 :     //----- ボタンチップテキスト設定 -----//
47 :     SAjrTip.Add(btnTest, "2D図形描画/消去");
48 :     //----- ウィンド最小サイズ設定 -----//
49 :     m_MinSize = this.Size;
50 :
51 :     //----- 設定値ロード -----//
52 :     SAjrReg.LoadAllCtrls(this);
53 :     //----- グラフレンジ設定 -----//
54 :     g3d.SetRange(X1, Y1, X2, Y2);
55 : }
56 : //----- フォームクローズ -----//
57 : private void Form1_FormClosed(object sender, FormClosedEventArgs e)
58 : {
59 :     // 設定値セーブ
60 :     SAjrReg.SaveAllCtrls(this);
61 : }
62 : //----- フォームサイズ変更 -----//
63 : private void Form1_Resize(object sender, EventArgs e)
64 : {
65 :     if (this.Width < m_MinSize.Width) this.Width = m_MinSize.Width;
66 :     if (this.Height < m_MinSize.Height) this.Height = m_MinSize.Height;
67 : }
68 :
69 : //----- コントロールからのイベント -----//
70 : // スタイル・チェックボックス群
71 : private void chkShowFilter_CheckedChanged(object sender, EventArgs e) {g3d.ShowFilter = chkShowFilter.Checked;}
72 : private void chkShowBorder_CheckedChanged(object sender, EventArgs e) {g3d.ShowBorder = chkShowBorder.Checked;}
73 : private void chkShowAxisX_CheckedChanged(object sender, EventArgs e) {g3d.ShowAxisX = chkShowAxisX.Checked;}
74 : private void chkShowAxisY_CheckedChanged(object sender, EventArgs e) {g3d.ShowAxisY = chkShowAxisY.Checked;}
75 : private void chkShowRectScale_CheckedChanged(object sender, EventArgs e) {g3d.ShowRectScale = chkShowRectScale.Checked;}
76 : private void chkShowEllipseScale_CheckedChanged(object sender, EventArgs e) {g3d.ShowEllipseScale = chkShowEllipseScale.Checked;}
77 : private void chkShowScaleXY_CheckedChanged(object sender, EventArgs e) {g3d.ShowScaleXY = chkShowScaleXY.Checked;}
78 : private void chkShowScaleValueX_CheckedChanged(object sender, EventArgs e) {g3d.ShowScaleValueX = chkShowScaleValueX.Checked;}
79 : private void chkShowScaleValueY_CheckedChanged(object sender, EventArgs e) {g3d.ShowScaleValueY = chkShowScaleValueY.Checked;}
80 : // プロット線描画チェックボックス
81 : private void chkPlotLine_CheckedChanged(object sender, EventArgs e) {g3d.PlotLine = chkPlotLine.Checked;}
82 : // プロット数
83 : private void inpPlotNum0_OnNtcIntValue(object sender, IvArgNtcIntValue e) {g3d.SetMaxPlot(0, e.value);}
84 : private void inpPlotNum1_OnNtcIntValue(object sender, IvArgNtcIntValue e) {g3d.SetMaxPlot(1, e.value);}
85 : private void inpPlotNum2_OnNtcIntValue(object sender, IvArgNtcIntValue e) {g3d.SetMaxPlot(2, e.value);}
86 : private void inpPlotNum3_OnNtcIntValue(object sender, IvArgNtcIntValue e) {g3d.SetMaxPlot(3, e.value);}
87 : // サンプリング周期
88 : private void inpPeriod_OnNtcIntValue(object sender, IvArgNtcIntValue e) {tim.Interval = e.value;}
89 : // テストボタン
90 : private void btnTest_Click(object sender, EventArgs e)
91 : {
92 :     if (!m_fTest) {
93 :         g3d.Star(4, 1270, 370, 20.0, 0.0, 5, 0.0, true);
94 :         g3d.FillB(5, 4, 1270, 370);
95 :         g3d.Pixel(4, 1270, 370, 3);
96 :         g3d.Ellipse(5, 1270, 370, 23, 23);
97 :         g3d.Triangle(6, 1210, 350, 1230, 390, 1240, 370);
98 :         g3d.Ellipse(7, 1250, 330, 40, 15);
99 :         m_fTest = true;
100 :     }
101 :     else {
102 :         g3d.PurgeShape();
103 :         m_fTest = false;
104 :     }
105 : }
106 : // プロット開始/停止ボタン
107 : private void btnStartStop_Click(object sender, EventArgs e)
108 : {
109 :     if (m_fPlot) {
110 :         m_fPlot = false;
111 :         tim.Stop();
112 :         btnStartStop.Text = "プロット開始";
113 :     }
114 :     else {
115 :         m_fPlot = true;
116 :         tim.Start();
117 :         btnStartStop.Text = "プロット停止";
118 :     }
119 : }

```

```

120 : // クリアボタン
121 : private void btnClear_Click(object sender, EventArgs e)
122 : {
123 :     m_fTest = false;
124 :     g3d.Purge();
125 : }
126 : // 周期タイマ
127 : private void tim_Tick(object sender, EventArgs e)
128 : {
129 :     double x, y, r, xc, yc;
130 :
131 :     for (int i = 0; i < 4; i++) {
132 :         xc = XC; yc = YC;
133 :         r = 10 * (i + 1);
134 :         x = r * SAjrMath.Cos(m_theta[i]);
135 :         y = r * SAjrMath.Sin(m_theta[i]);
136 :         g3d.PutData(i, XC + x, YC + y);
137 :         if (m_theta[i] >= 0.0 && m_theta[i] <= 90.0) {
138 :             g3d.Ellipse (i, xc, yc, r, r);
139 :         }
140 :         else {
141 :             g3d.PurgeData(i);
142 :         }
143 :         m_theta[i] += 10.0 * ((double)cRandom.Next(RAND_MAX) / (double)RAND_MAX);
144 :         m_theta[i] = (m_theta[i] % 360.0);
145 :     }
146 : }
147 : }
148 : }

```

6.6.3. Sil_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。

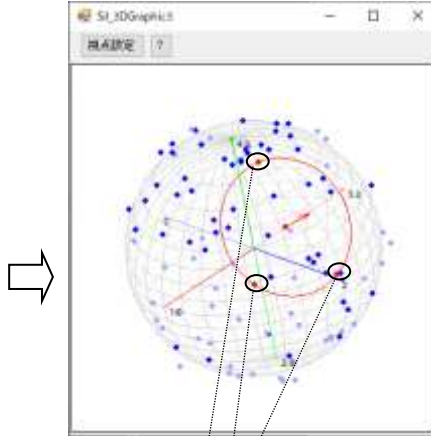
クリックした3点から平面円と平面の法線を算出し、表示します。

「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

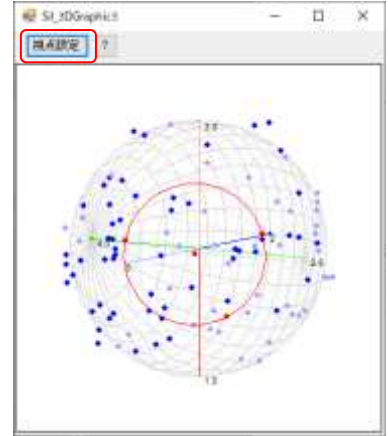


3点選択



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 : using System;
2 : using System.Collections.Generic;
3 : using System.ComponentModel;
4 : using System.Data;
5 : using System.Drawing;
6 : using System.Linq;
7 : using System.Text;
8 : using System.Windows.Forms;
9 : using CAjrCustCtrl;
10 :
11 : namespace Sil_3DGraphic3
12 : {
13 :     public partial class Form1 : Form
14 :     {
15 :         AJC3DVEC    m_cent  = new AJC3DVEC(1.0, 2.0, 3.0);    // 中心位置
16 :         int          m_Count = 0;
17 :         int          m_Ix    = 0;
18 :         AJC3DVEC[]   m_Points = new AJC3DVEC[3];
19 :         AJC3DVEC     m_vh;
20 :
21 :         string m_msg = "SHIFT+クリックで、任意の3点を選択してください。¥n" +
22 :             "3点が選択されたら、各点をとる平面円が表示されます。¥n" +
23 :             "¥n" +
24 :             "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";
25 :
26 :         public Form1()
27 :         {
28 :             InitializeComponent();
29 :         }
30 :         // 起動時初期設定
31 :         private void Form1_Load(object sender, EventArgs e)
32 :         {
33 :             Random rnd = new Random(1000);
34 :             double max;
35 :             AJC3DVEC v;
36 :
37 :             g3d.SetCenter(m_cent);
38 :             // 球面付近にランダムな点を100個表示
39 :             for (int i = 0; i < 100; i++) {
40 :                 v.x = rnd.NextDouble();
41 :                 max = Math.Sqrt(1.0 - Math.Pow(v.x, 2.0));
42 :                 v.y = rnd.NextDouble() * max;
43 :                 v.z = Math.Sqrt(1.0 - Math.Pow(v.x, 2.0) - Math.Pow(v.y, 2.0));
44 :                 if ((rnd.Next(2) & 1) != 0) v.x += (rnd.NextDouble() * 0.1); else v.x -= (rnd.NextDouble() * 0.1);

```

```

45 :         if ((rnd.Next(2) & 1) != 0) v.y += (rnd.NextDouble() * 0.1); else v.y -= (rnd.NextDouble() * 0.1);
46 :         if ((rnd.Next(2) & 1) != 0) v.z += (rnd.NextDouble() * 0.1); else v.z -= (rnd.NextDouble() * 0.1);
47 :         if ((rnd.Next(2) & 1) != 0) v.x *= -1;
48 :         if ((rnd.Next(2) & 1) != 0) v.y *= -1;
49 :         if ((rnd.Next(2) & 1) != 0) v.z *= -1;
50 :         v = SAjrMath.V3dAdd(v, m_cent);
51 :         g3d.Pixel(0, v, 3);
52 :     }
53 :     SAjrTip.ShowCenter(g3d, m_msg);
54 : }
55 : // 3Dグラフィック・プロット点通知
56 : private void cAjr3DGraphic1_OnPltLst(object sender, CAjrCustCtrl.G3dArgPltLst e)
57 : {
58 :     int id = 1;
59 :     AJC3DVEC vc, v;
60 :     double r;
61 :     AJC3DCIRINFO cif;
62 :     // プロット点をバッファへ格納
63 :     m_Points[m_Ix] = e.p[0].v;
64 :     m_Ix = (m_Ix + 1) % 3;
65 :     if (m_Count < 3) m_Count++;
66 :     // プロット点表示
67 :     g3d.PurgeData(id);
68 :     for (int i = 0; i < m_Count; i++) {
69 :         g3d.Pixel(id, m_Points[i], 3);
70 :     }
71 :     // 3点が揃ったら平面円と法線表示
72 :     if (m_Count >= 3) {
73 :         btnViewPoint.Enabled = true;
74 :         r = SAjrMath.V3dCalcCircle(m_Points[m_Ix], m_Points[(m_Ix + 1) % 3], m_Points[(m_Ix + 2) % 3], out vc, out m_vh, out cif);
75 :         g3d.DefPlane(id, cif.lvc);
76 :         g3d.Ellipse(id, 0, 0, cif.cr, cif.cr);
77 :         g3d.Pixel(id, 0, 0, 3);
78 :         v = SAjrMath.V3dNormal(m_vh);
79 :         v = SAjrMath.V3dMult(v, 0.5);
80 :         v = SAjrMath.V3dAdd(v, vc);
81 :         g3d.Line(id, vc, v, true);
82 :     }
83 : }
84 : // 視点設定ボタン
85 : private void btnViewPoint_Click(object sender, EventArgs e)
86 : {
87 :     g3d.SetAngle(m_vh);
88 : }
89 : // ?ボタン
90 : private void btnHelp_Click(object sender, EventArgs e)
91 : {
92 :     SAjrTip.ShowCenter(g3d, m_msg);
93 : }
94 :
95 : }
96 : }

```

7. VT-100エミュレーション・ウインド コントロール (CAjrVT100.dll)

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。

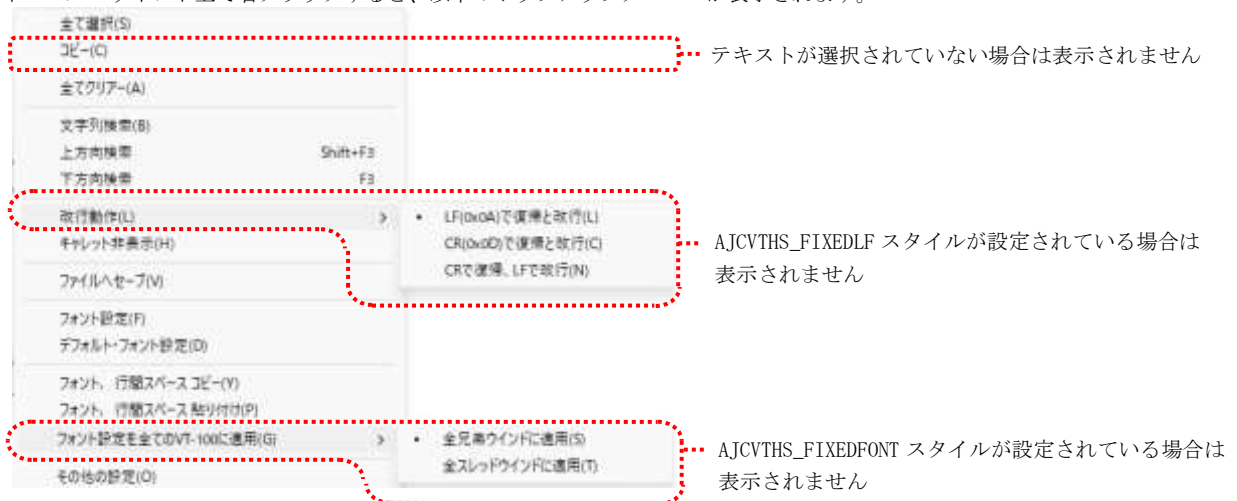
ANSIエスケープコードにより、描画スクリーン上の任意の(文字単位の)位置にグラフィカルに文字列を描画することもできます。ANSIエスケープコードでは、描画色の設定や、スクリーンの部分スクロール等ができます。(サポートするESCコードは、後述します)



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。

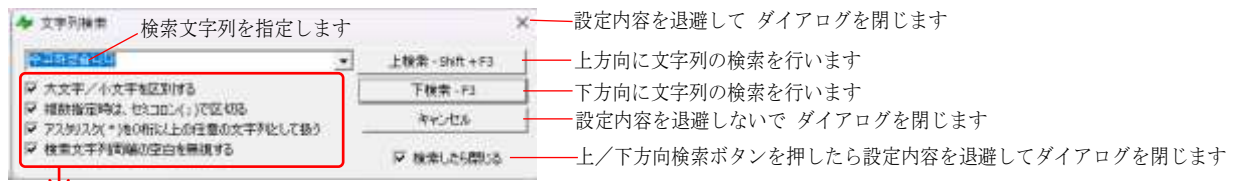


各メニューの内容は、以下のとおりです。

#	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	全てのテキストデータを破棄し、画面をクリアします。
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

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icolonで区切る

複数の文字列を指定する場合は、セミicolon (;) 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 “ABCDEFGH IJKLMNOPQRSTU VWXYZ” が存在する場合、検索文字列が「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icolon (;) 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”)

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。(StrFindKey プロパティ=Keys.F3 の場合)

スクロールバーやマウスホイールでテキストをスクロール、あるいは、新たなテキスト描画を行った場合、下方向検索開始位置はウインド上端に、上方向検索開始位置はウインド下端にリセットされます。

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
マウスホイールを操作			
新たなテキストを表示			

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。

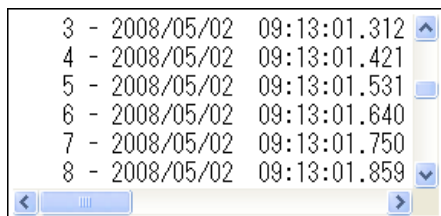
CTRL キーを押下しながらドラッグした場合は、行単位のテキスト選択となります。

選択されたテキストは、ポップアップメニューから「コピー」を選択するか、CTRL+C キーを押すことにより、クリップボードへコピーできます。

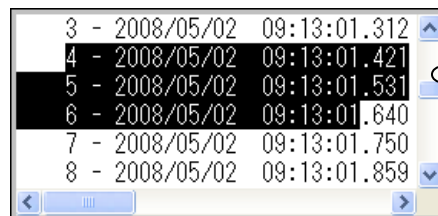
尚、SHIFT キーを押しながらドラッグ操作を行うことにより、スクロール後に、前回のドラッグ操作を継続することができます。

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

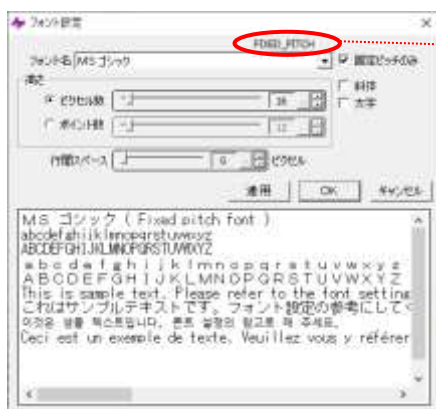


テキストを選択している状態

外枠がグレー表示される

7.1.4. フォントの設定

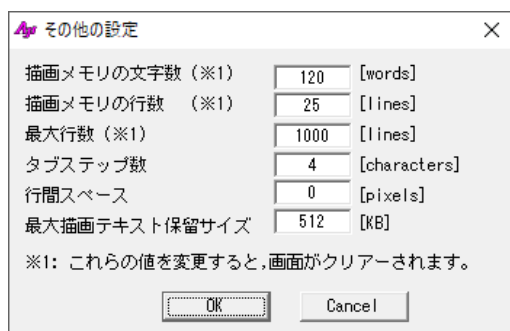
右クリックによるポップアップメニューから「フォント設定」を選択し、任意のフォントを設定することができます。



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 文字エンコード
	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、Pause() メソッド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。

```

        :
        :
        vth. Pause(true);           // 表示停止
        vth. PutText(・・・);       // テキスト描画
        vth. PutText(・・・);
        :
        :
        vth. Pause(false);          // 表示再開 (①の期間に描画したテキストを一気に表示)
        :
        :
    
```

① (この間描画したデータは表示されない)

vth. Pause(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)
 vth. Pause(true)～vth. Pause(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。

※vth. Pause(true)～vth. Pause(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

Fast プロパティが true に設定されている場合、ウインドへの描画処理を高速に実行します。

前述の Pause() メソッドによる高速化は、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、Fast プロパティを設定した場合は、ウインドへの描画処理自体を高速に実行します。

Fast プロパティは、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。

Fast プロパティを設定すると、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。

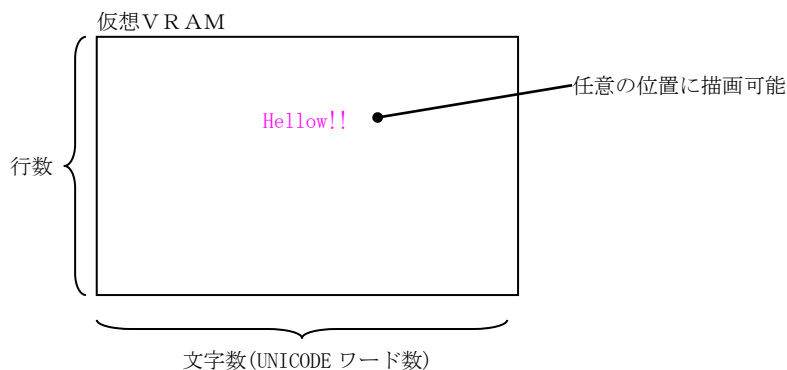
但し、Fast プロパティを設定した場合は、以下の制限があります。

- ・文字色(ForegroundColor)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)
- ・背景色(BackgroundColor)と Separate プロパティは無視されます。

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。



桁位置と行位置は、仮想VRAMの左上位置を原点としますが、本コントロールの Locate メソッドと ANSIエスケープコードでは位置の指定方法が異なります。

本コントロールの Locate メソッドでは、仮想VRAMの左上位置を 0 行, 0 桁とします。

ANSIエスケープコードでは、仮想VRAMの左上位置が 1 行, 1 桁となります。

仮想VRAMの横サイズは、VRamWidth プロパティ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。

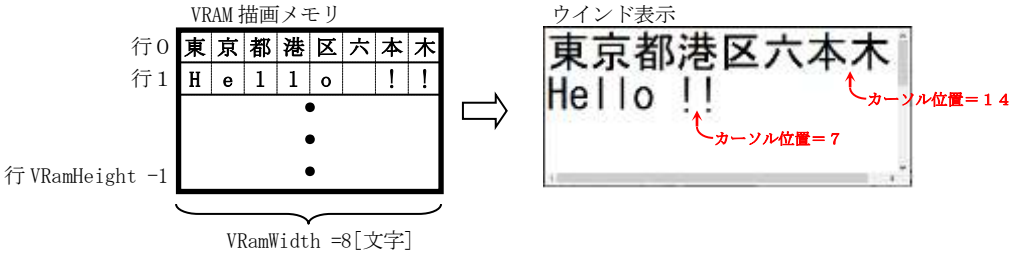
- ・文字は全て UNICODE で格納します。
- ・VRamWidth プロパティは、横方向の最大格納ワード数を意味します。(1 ワード = 2 バイト (16 ビット))
- ・サロゲートペア文字以外の場合は、半角文字 / 全角文字は区別されずに、それぞれ 1 ワード分のメモリを消費します。
- ・全てサロゲートペア文字以外の場合は、1 行に格納できる文字数は VRamWidth プロパティと等しくなります。
- ・サロゲートペア文字は、格納に 2 ワード分のメモリを消費します。
- ・仮に、全てサロゲートペア文字である場合は、1 行に格納できる文字数は VRamWidth プロパティの半分となります。

7.1.10. カーソル位置

テキストは全てカーソル位置から描画します。
 カーソル行位置は、VRAM先頭行を0とし、0～VRamHeight - 1 までとなります。
 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～VRamWidth-1 の範囲となります。
 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～(VRamWidth - 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。

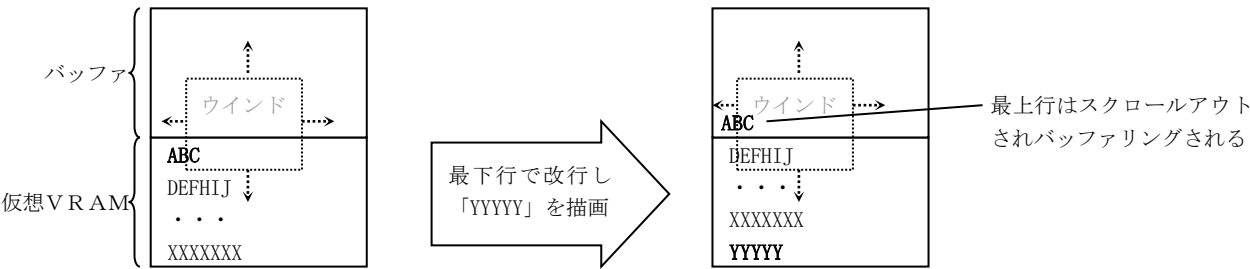
バッファに格納できる文字数は、VRamWidthで制限される (VRamWidthは半角／全角に関わらず1行に格納可能な文字数を意味する) ため、見かけ上、全角文字は半角文字よりも長く表示できます。

例えば、VRamWidth =8 で「東京都港区六本木」と「Hello !!」を描画した場合、表示は以下のようになります。



7.1.11. スクロールアウトした行のバッファリング

EnableScrollOut プロパティが true に設定されている場合、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×桁数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、当該行の有効桁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、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）

EnableScrollOut プロパティが false に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

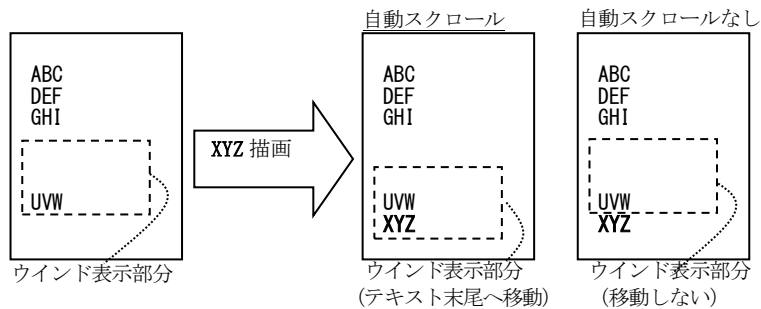
7.1.12. 右クリック

右クリック操作による動作は、以下のようになっています。

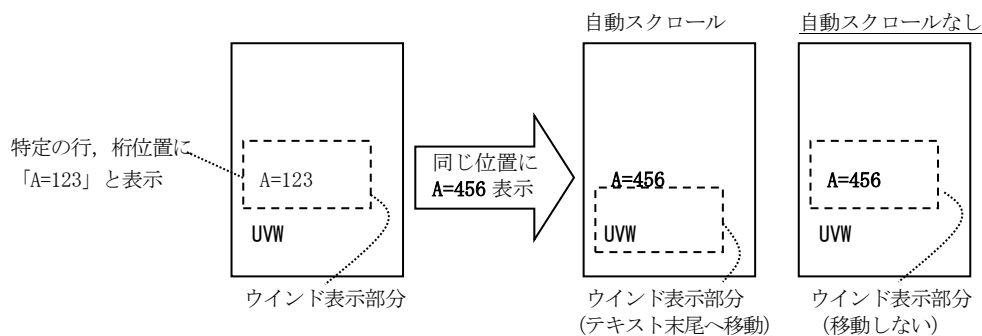
ポップアップメニュー許可 EnablePopupMenu プロパティ	SHIFT/CTRL キー押下	動作
True (許可)	× (未押下)	ポップアップメニューを表示
True (許可)	○ (押下)	右クリック通知イベント (OnRClick) 発生 (右ボタン押下時に発生)
False (禁止)	—	右クリック通知イベント (OnRClick) 発生 (右ボタン押下時に発生)

7.1.13. オートスクロール

AutoScroll プロパティに「true」を設定した場合、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。

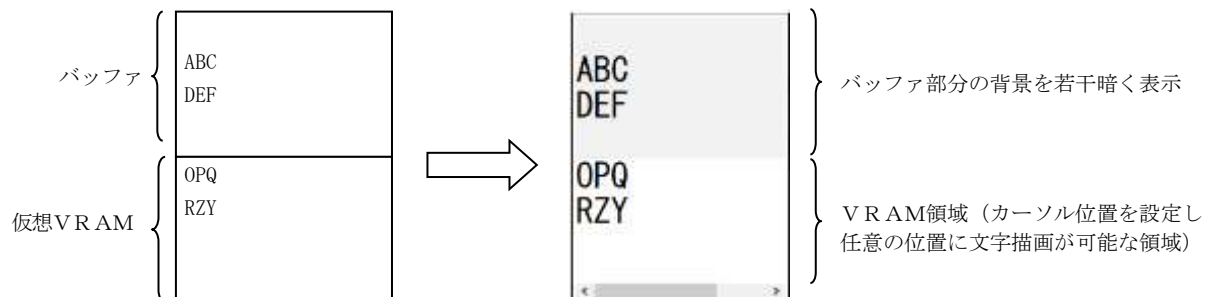


AutoScroll プロパティに「false」を設定した場合、手動でスクロールしない限り、スクロールを行いません。
「オートスクロールなし」は、例えばスクリーンの固定位置（固定の行、桁位置）に情報を表示する場合に有効です。



7.1.14. VRAMとバッファを識別して表示

Separate プロパティに「true」を設定した場合、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。



※実行時にマウスの中ボタン（ホイールボタン）で、VRAM部分の識別をON/OFF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。

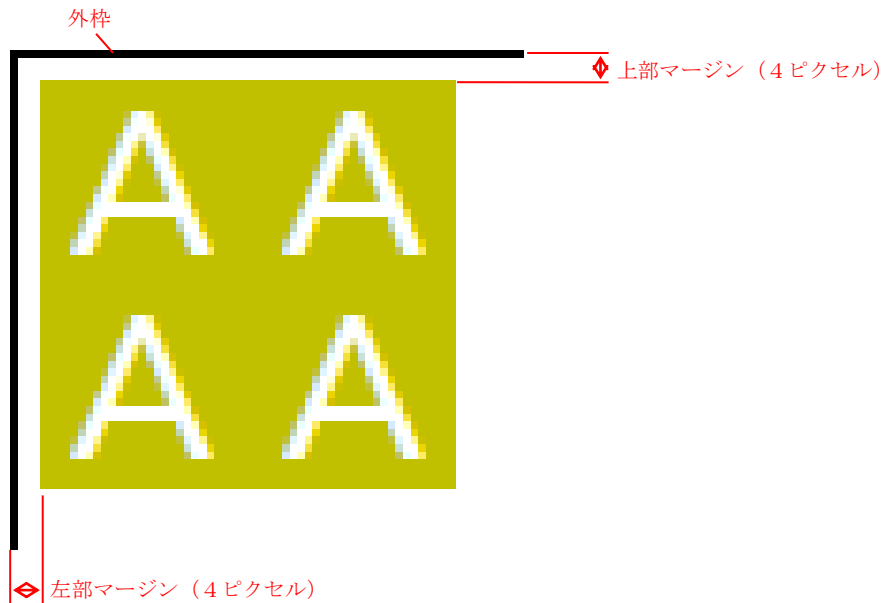
一時保留バッファの容量は、「PendingBufferSize」プロパティで任意のサイズを設定できます。

※ 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

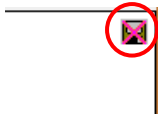
文字がウインドの外枠とくっつかないように、上下左右にマージン（4ピクセル固定）を設けています。
この4ピクセルのマージンは、外枠表示分の1ピクセルを含んでいます。
つまり、外枠を表示している場合、実際のマージンは3ピクセルとなります。

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

「EnableLogFile」プロパティを true に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。
「EnableLogFile」プロパティを true に指定すると、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、以下のイベントが発生します。

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

これらのイベントでは、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 GetDroppedFile/GetDroppedDir メソッドにて取得します。

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ロップを有効とするには、AcceptFiles プロパティを true に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、「FixedPitch」プロパティ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。

「FixedPitch」プロパティを設定した場合、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、「FixedPitch」プロパティによる表示の違いを示します。

FixedPitch = false



FixedPitch = true

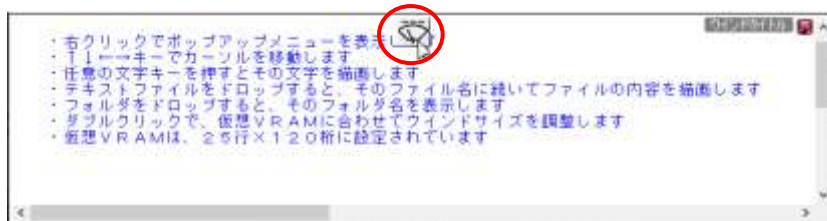


可変ピッチフォントの場合、「FixedPitch」プロパティは無視されます。

7.1.20. 画面クリアボタン

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、画面がクリアされます。



画面クリアボタンは、「EnableClsButton」プロパティが false に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

本コントロールの `PutText()` や `PutFormat()` でサポートする制御文字とエスケープシーケンスは、以下のとおりです。

制御文字

#	制御文字	内 容
1	"¥b" (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	"¥t" (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	"¥n" (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	"¥r" (0x0D)	復帰 (カーソルを行の先頭へ移動)
5	"¥x1B" (0x1B)	エスケープシーケンスの始まり

エスケープシーケンス

#	ESC シーケンス	内 容
1	"¥x1B [0J"	カーソル位置～最終行の右端までクリアー
2	"¥x1B [1J"	先頭行の左端～カーソル位置までクリアー
3	"¥x1B [2J" "¥x1B *"	画面をクリアーし、カーソルをホームへ移動
4	"¥x1B [0K" "¥x1B [K"	カーソル位置～同行右端までをクリアー
5	"¥x1B [1K"	カーソル行の左端～カーソル位置までをクリアー
6	"¥x1B [2K"	カーソル行をクリアー
7	"¥x1B [s"	カーソル位置を退避
8	"¥x1B [u"	カーソル位置を回復
9	"¥x1B [>5l"	カーソル表示
10	"¥x1B [>5h"	カーソル非表示
11	"¥x1B [pI;pcH"	カーソル位置設定 (<i>pI</i> は行位置 (1～), <i>pc</i> は桁位置 (1～))
12	"¥x1B [pnA"	カーソルを上方向に移動 (<i>pn</i> は移動行数であり、省略時は1を仮定)
13	"¥x1B [pnB"	カーソルを下方向に移動 (")
14	"¥x1B [pnC"	カーソルを右方向に移動 (<i>pn</i> 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	"¥x1B [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	"¥x1B [pnM"	カーソル行以降をスクロールアップ (<i>pn</i> はスクロール行数)
17	"¥x1B [pnL"	カーソル行以降をスクロールダウン (")
18	"¥x1B [psm"	描画属性設定 (<i>ps</i> =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	"¥x1B D"	カーソルを1行下へ移動
20	"¥x1B E"	カーソルを1行下の左端へ移動
21	"¥x1B M"	カーソルを1行上へ移動

ToolTipText プロパティも、制御文字とエスケープシーケンスを含めることができますが、これについては、「テキスト表示拡張機」の章を参照してください。

7.4. プロパティ

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一覧を示します。

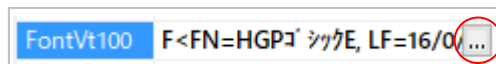
下線 (XXXX) の項目は、ポップアップメニューの「設定をリセット」でデザイン時の状態に戻される項目を意味します。

#	名称	タイプ	内容	デフォルト
1	AcceptFiles	bool	ファイル/フォルダのドロップ許可フラグ	false
2	ForegroundColor	int	文字色のパレット番号 (0～7)	0
3	BackgroundColor	int	文字背景色のパレット番号 (0～7)	7
4	WndBkColor	int	ウインド背景色のパレット番号 (0～7)	7
5	CaretHeight	int	キャレットの高さ (現状は未使用 (旧バージョンと互換の為に存在))	-
6	CharHeight	int	文字の高さ (ピクセル数, 読み出し専用)	-
7	CharWidth	int	文字の幅 (ピクセル数, 読み出し専用)	-
8	EnablePopupMenu	bool	ポップアップメニュー許可フラグ (※1)	true
9	FontVt100	string	フォント定義文字列 (独自) (※2)	-
10	LFActInPopupMenu	bool	ポップアップメニューに改行動作の設定を含める	true
11	LineCount	int	バッファに格納されている行数 (読み出し専用)	-
12	LineHeight	int	行の高さ (ピクセル数, 読み出し専用)	12
13	LineSpace	int	行間スペース (ピクセル数)	0
14	MaxLines	int	最大保留行数 (スクロール可能な行数, VRamHeight 以上の値) (※5)	10000
15	PendingBufferSize	int	一時保留バッファサイズ [Bytes]	524288
16	ScrollPosH	int	横スクロール位置 (ウインド左端の桁位置)	0
17	ScrollPosV	int	縦スクロール位置 (ウインド上端の行位置)	0
18	ShowDummyData	bool	デザイン時のダミーデータ表示フラグ (※3)	true
19	TabStep	int	TAB 文字のステップ数 (2, 4, 8 or 16)	4
20	ToolTipText	string	コントロールのツールヒント (※4)	""
21	ToolTipShowAlways	bool	ツールチップ表示条件 true - 非アクティブ状態でも表示 false - 非アクティブ状態時は非表示	true
22	VRamHeight	int	VRAM 行数 (4 以上, MaxLines 以下の値) (※5)	64
23	VRamWidth	int	VRAM 文字数 (4 以上)	256
24	WndHeight	int	表示ウインド行数 (読み出し専用)	-
25	WndWidth	int	表示ウインド桁数 (読み出し専用)	-
26	CursorLine	int	カーソル行位置 (読み出し専用)	-
27	CursorPos	int	カーソル文字位置 (読み出し専用, 全角文字は2桁でカウント)	-
28	TitleText	string	タイトルテキスト (文字色=白, 背景=グレーで右上に表示)	""
29	EnableClsButton	bool	画面クリアボタンの許可フラグ	true

つづく

※1 : 右クリックによるポップアップメニューの表示(true)/非表示(false)を設定します。
非表示(false)を設定した場合は、右クリックすると「OnRClick」イベントが発生します。
(Shift/Ctrl + 右クリックした場合は、EnablePopupMenu プロパティに関係なく常に「OnRClick」イベントが発生します)

※2 : フォント設定は、文字列を直接編集せずに、ボタンを押下しフォント設定ダイアログで指定してください。



このボタンを押すと、フォント設定ダイアログが表示されます

Font オブジェクトでフォントの設定/取得する場合は、SetFont()/GetFont() メソッドを使用します。

※3 : デザイン時にダミーデータを表示することにより、プロパティの効果を視覚的に確認することができます。

※4 : ToolTipText プロパティは、制御文字とエスケープシーケンスを含めることができます。
これについては、「テキスト表示拡張機能」の章を参照してください。

※5 : VRamHeight と MaxLines を設定する場合は、VRamHeight、MaxLines の順で設定してください。

下線 (XXXX) の項目は、ポップアップメニューの「設定をリセット」で初期状態に戻される項目を意味します。

#	名称	タイプ	内容	デフォルト
29	EnableLogFile	bool	ファイル出力コントロールの表示	false
30	FixedPitch	bool	固定ピッチで表示	false
31	Fast	bool	テキストの描画処理を高速に実行します。 (※6)	false
32	FixedFont	bool	フォント固定 (ポップアップメニューにフォント設定を含めない)	false
33	FixedLF	bool	改行動作固定 (ポップアップメニューに改行動作に関する項目を含めない)	false
34	ShowVScroll	bool	縦スクロールバー表示(true)/非表示(false)	true
35	ShowHScroll	bool	横スクロールバー表示(true)/非表示(false)	true
36	EnableAutoScroll	bool	自動スクロール(描画時末尾へカーソル移動)の許可(true)/禁止(false)	true
37	ShowBorder	bool	コントロール外枠の表示フラグ	true
38	EnableScrollOut	bool	スクロールアウト許可	true
39	LineFeedByCR	bool	C R ("Yr") で復帰・改行(false 時は CR で復帰のみ)	false
40	LineFeedByLF	bool	L F ("Yn") で復帰・改行(false 時は LF で改行のみ)	true
41	Separate	bool	V R A M とその他の領域分離 (VRAM 以外を若干暗くする)	false
42	StrFindInfoSect	string	文字列検索情報を記録するプロファイルセクション名	_DefaultStrFindInfoSect_
43	StrFindKey	Keys	文字列検索の操作キー (Keys.None : キー操作で検索しない)	Keys.F3

※6 : true を設定した場合、描画を高速に実行します。 数秒オーダーの速い周期で、ログを連続的に表示する場合に有効です。

7.5. メソッド

V T - 1 0 0 エミュレーションウインド・コントロールのメソッド一覧を以下に示します。

#	メソッド名	内容	備考
1	PutChar	1 文字描画	
2	PutByte	1 バイト描画	
3	PutText	文字列描画	
4	PutFormat	書式文字列描画	
5	PrintTimeStamp	タイムスタンプ描画	
6	PrintHexDump	1 6 進ダンプ描画	unsafe
7	Locate	カーソル位置設定	
8	SetPaletteColor	パレット色設定	
9	GetPaletteColor	パレット色取得	
10	Select	部分テキスト選択	
11	SelectAll	全テキスト選択	
12	Copy	選択テキストをクリップボードへコピー	
13	Purge	画面クリア (全テキスト 破棄)	
14	GetDroppedFile	ドロップされたファイル名取得	
15	GetDroppedDir	ドロップされたディレクトリ名取得	
16	SetTitleText	タイトルテキスト表示	画面右上に表示
17	GetLineText	行テキスト取得	
18	GetDbClickedLineText	ダブルクリック行テキスト取得	
19	GetDbClickedLinePos	ダブルクリック行位置取得 (バッファ上の行位置)	
20	SearchBelow SearchAbove	文字列の検索	
21	SaveToProfile SaveFontToProfile	設定値をプロファイルへ記録 フォント情報をプロファイルへ記録	
22	LoadFromProfile LoadFontFromProfile	設定値をプロファイルから読み出し フォント情報をプロファイルから読み出し	
23	Pause	画面表示の停止/再開	
24	SetFont	フォント設定	
25	GetFont	フォント取得	

7.5.1. 1文字描画 (PutChar)

形 式 : void PutChar(int c);

引 数 : c - 描画する文字のコード (UNICODE)

説 明 : カーソル位置へ、1文字描画します。

戻り値 : なし

7.5.2. 1バイト描画 (PutByte)

形 式 : void PutByte(int c);

引 数 : c - 描画する文字のバイトコード (S-JIS)

説 明 : カーソル位置へ、1バイト描画します。
マルチバイト文字 (全角文字) の場合、2回で1文字が描画されます。

戻り値 : なし

7.5.3. 文字列描画 (PutText)

形 式 : void PutText(string text);
void PutText(string text, len);

引 数 : text - 描画する文字列
len - 描画する文字数 (先頭部分の文字数, 未指定/-1 指定時は文字列全体)

説 明 : カーソル位置へ、文字列を描画します。

戻り値 : なし

7.5.4. 書式文字列出力 (PutFormat)

形 式 : void PutFormat(IntPtr hFile, string format [, arg]...);

引 数 : hFile - 出力テキストファイルのファイルハンドル
format - 書式文字列 (string.Format()と同じ)
arg - パラメタ (可変個引数)

説 明 : カーソル位置へ、書式化した文字列を描画します。
書式文字列とパラメタは、string.Format()メソッドと同じです、
書式文字列や、パラメタには3つの文字('¥u0000', '¥uFFFA', '¥uFFFB')を含まないようにしてください。

戻り値 : true - 成功
false - 失敗

備 考 : 書式文字列や、パラメタに文字('¥u0000', '¥uFFFA', '¥uFFFB')が含まれる場合は、以下のように処理されます。

- '¥u0000' - 文字列の終端となります。
- '¥uFFFA' - '{' に変換されます。
- '¥uFFFB' - '}' に変換されます。

尚、PutFormat()は、処理効率 (速度やメモリ消費) が良くありません。

処理効率を考慮する必要がある場合は、string.Format()で書式化した文字列を出力するようにしてください。

(例) vth.PutFormat(hFile, "x={0}", x); ⇒ string s = string.Format("x={0}", x);
vth.PutText(hFile, s);

7.5.5. タイムスタンプ描画 (PrintTimeStamp)

形 式 : void PrintTimeStamp();

引 数 : なし

説 明 : カーソル位置へ、現在時刻を表す文字列を描画します。
文字列の形式は“hh:mm:ss.nnn”です。(nnn は、ミリ秒を意味します)

戻り値 : なし

7.5.6. 16進ダンプ描画 (PrintHexDump)

形 式 : unsafe void PrintHexDump(void *pData, int lData);
void PrintHexDump(IntPtr pData, int lData);

引 数 : pData - 16進ダンプ表示するデータのアドレス
lData - 16進ダンプ表示するデータのバイト数

説 明 : カーソル位置へ、指定されたデータを16進でダンプ表示します。
各バイト値の直前に空白を挿入します。(つまり、バイト数×3[桁]の文字列を描画することになります)

戻り値 : なし

7.5.7. カーソル位置設定 (Locate)

形 式 : void Locate(int line, int col);

引 数 : line - 行位置 (0～)
col - 文字位置 (0～)

説 明 : VRAM上の指定された、行位置、桁位置へカーソルを移動します。
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。

戻り値 : なし

7.5.8. パレット色設定 (SetPaletteColor)

形 式 : void SetPaletteColor(int ix, Color color);

引 数 : ix - パレット番号 (0～7)
color - 設定する色

説 明 : 指定された番号のパレットに、表示色を設定します。

戻り値 : なし

7.5.9. パレット色取得 (GetPaletteColor)

形 式 : Color GetPaletteColor(int ix);

引 数 : ix - パレット番号 (0～7)

説 明 : 指定された番号のパレットに設定されている表示色を取得します。

戻り値 : パレットに設定されている表示色

7.5.10. 部分テキスト選択 (Select)

形 式 : void Select(int slp, int scp, int elp, int ecp);

引 数 : 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 : 指定した部分のテキストを選択状態にします。

戻り値 : なし

7.5.11. 全テキスト選択 (SelectAll)

形 式 : void SelectAll();

引 数 : なし

説 明 : 全テキストを選択状態にします。

戻り値 : なし

7.5.12. 選択テキストをクリップボードへコピー (Copy)

形 式 : void Copy();

引 数 : なし

説 明 : 選択状態のテキストをクリップボードへコピーします。

戻り値 : なし

7.5.13. 全テキスト 破棄 (Purge)

形 式 : void Purge();

引 数 : なし

説 明 : 全テキストを破棄し、画面をクリアします。

戻り値 : なし

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

形 式 : `string GetDroppedFile();`

引 数 : なし

説 明 : OnFileDrop イベント発生時に、順次、ドロップされたファイルのパス名を取得します。
OnFileDrop イベントで通知された、ドロップファイル数の回数だけ本メソッドをこーするすることにより、ドロップされた全てのファイルを取得できます。

戻り値 : `≠""` : ドロップされたファイルパス名
`=""` : 終端(ドロップされたファイルパス名の取得は完了済である)

7.5.15. ドロップされたディレクトリ名取得 (GetDroppedDir)

形 式 : `string GetDroppedDir();`

引 数 : なし

説 明 : OnDirDrop イベント発生時に、順次、ドロップされたディレクトリのパス名を取得します。
OnDirDrop イベントで通知された、ドロップディレクトリ数の回数だけ本メソッドをこーすることにより、ドロップされた全てのディレクトリを取得できます。

戻り値 : `≠""` : ドロップされたディレクトリパス名
`=""` : 終端(ドロップされたディレクトリパス名の取得は完了済である)

7.5.16. タイトルテキスト表示 (SetTitleText)

形 式 : `void SetTitleText(string TitleText);`
`void SetTitleText(string TitleText, Color TextColor);`
`void SetTitleText(string TitleText, Color TextColor, Color BackColor);`
`void SetTitleText(string TitleText, Color TextColor, Color BackColor, Font TextFont);`

引 数 : TitleText - タイトルテキスト (null(あるいは空文字列(""))を指定した場合は非表示)
TextColor - テキスト描画色 (α値は無視されます)
BackColor - テキスト背景色 (α値は無視されます)
TextFont - テキストのフォント

説 明 : コントロールウインドの右上にタイトルテキストを表示します。
TitleText に null(あるいは空文字列(""))を指定した場合、タイトルテキストは非表示となります。

戻り値 : なし

7.5.17. 行テキスト取得 (GetLineText)

形 式 : `string GetLineText(int pos);`

引 数 : pos - 行位置 (0～)

説 明 : 指定行位置の行テキストを取得します。
行位置は、バッファ先頭(スクロール最上端行位置)からの相対行位置で指定します。

戻り値 : 行テキスト

7.5.18. ダブルクリック行テキスト取得 (GetDbClickedLineText)

形 式 : `string GetDbClickedLineText();`

引 数 : なし

説 明 : ダブルクリックした行位置の行テキストを取得します。

戻り値 : 行テキスト

7.5.19. ダブルクリック行位置取得 (GetDbClickedLinePos)

形 式 : `int GetDbClickedLinePos();`

引 数 : なし

説 明 : ダブルクリックしたバッファ上の行位置を取得します。
行位置は、バッファ先頭（スクロール最上端行位置）からの相対行位置です。

戻り値 : 行位置（0～）

7.5.20. 文字列の検索 (SearchBelow)

形 式 : `int SearchBelow(string str, char delimiter);` --- 下方向検索（前方検索）
`int SearchAbove(string str, char delimiter);` --- 上方向検索（後方検索）

引 数 : `str` - 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 : $\neq -1$: 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。

7.5.21. 設定値／フォント情報をプロファイルへ記録 (SaveToProfile, SaveFontToProfile)

形 式 : void SaveToProfile(string section);
void SaveFontToProfile(string section);

引 数 : section - プロファイルセクション名

説 明 : SaveToProfile() は、プロファイルへ以下の情報を記録します・

- ・ V R A M サイズ (幅, 高さ)
- ・ キャレットの高さ
- ・ 最大保留行数 (スクロール可能な行数)
- ・ T A B 文字のステップ数
- ・ 行間スペース (ピクセル数)
- ・ 一時保留バッファサイズ [Bytes]
- ・ フォント情報

SaveFontToProfile() は、フォント情報と行間スペースだけをプロファイルに記録します。

戻り値 : なし

7.5.22. 設定値／フォント情報をプロファイルから読み出す (LoadFromProfile, LoadFontFromProfile)

形 式 : void LoadFromProfile(string section);
void LoadFontFromProfile(string section);

引 数 : section - プロファイルセクション名

説 明 : LoadFromProfile () は、プロファイルから以下の情報を読み出して設定します。

- ・ V R A M サイズ (幅, 高さ)
- ・ キャレットの高さ
- ・ 最大保留行数 (スクロール可能な行数)
- ・ T A B 文字のステップ数
- ・ 行間スペース (ピクセル数)
- ・ 一時保留バッファサイズ [Bytes]
- ・ フォント情報

SaveFontToProfile() は、フォント情報と行間スペースだけをプロファイルから読み出して設定します。

戻り値 : なし

7.5.23. 画面表示の停止／再開 (Pause)

形 式 : void Pause (bool fPause);

引 数 : 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 : なし

7.5.24. フォント設定 (SetFont)

形 式 : void SetFont(Font font);

引 数 : font - 設定するフォント

説 明 : フォントを設定します。

戻り値 : なし

7.5.25. フォント取得(GetFont)

形 式 : Font GetFont();

引 数 : なし

説 明 : 現在設定されているフォントを取得します。

戻り値 : なし

7.6. イベント

VT100エミュレーションウインド・コントロールのイベント一覧を以下に示します。

#	イベント名	内容
1	OnNtcDbIcIk	ダブルクリック通知
2	OnNtcKeyIn	キー入力通知
3	OnNtcVKeyIn	拡張キー押下通知
4	OnNtcVKeyOut	拡張キー離し通知
5	OnFileDrop	ファイルドロップ通知
6	OnDirDrop	ディレクトリドロップ通知
7	OnCharInfo	文字サイズ変化通知
8	OnRClick	右クリック通知

7.6.1. ダブルクリック通知 (OnNtcDbIcIk)

形 式 : void OnNtcDbIcIk(object sender, VthArgDbIcIk e);

パラメタ : bool e.shift - Shift キーの押下状態(true : 押下)
bool e.ctrl - Ctrl キーの押下状態 (true : 押下)

説 明 : コントロール上でダブルクリックされたことを通知します。

戻り値 : なし

7.6.2. キー入力通知 (OnNtcKeyIn)

形 式 : void OnNtcKeyIn(object sender, VthArgNtcKeyIn e);

パラメタ : int e.key - 入力されたキーコード
int e.rep - キー押し続け時の繰り返し情報 (0: 初回, 1~ : キー押し続けによる繰り返し回数)

説 明 : コントロールにフォーカスがある状態で、入力されたキーのコードを通知します。
マルチバイト文字の場合は、2回に分けて通知します (第1バイト目, 第2バイト目の順)
Shift, Ctrl や F1 等の特殊キーは通知されません。
テキストが選択状態である場合は、「キー入力通知」は行われません。

戻り値 : なし

7.6.3. 拡張キー押下通知 (OnNtcVKeyIn)

形 式 : void OnNtcVKeyIn(object sender, VthArgNtcVKeyIn e);

パラメタ : Keys e.vkey - 押下された拡張キーコード
int e.rep - キー押し続け時の繰り返し情報 (0: 初回, 1~ : キー押し続けによる繰り返し回数)

説 明 : コントロールにフォーカスがある状態で、押下された拡張キーのコードを通知します。
Shift, Ctrl, Insert や F1 等の特殊キーも通知されます。
テキストが選択状態である場合は、「拡張キー押下通知」は行われません。

戻り値 : なし

7.6.4. 拡張キー離し通知 (OnNtcVKeyOut)

形 式 : void OnNtcVKeyOut(object sender, VthArgNtcVKeyOut e);

パラメタ : Keys e.vkey - 離された拡張キーコード

説 明 : コントロールにフォーカスがある状態で、押下された拡張キーのコードを通知します。
Shift, Ctrl, Insert や F1 等の特殊キーも通知されます。
テキストが選択状態である場合は、「拡張キー離し通知」は行われません。

戻り値 : なし

7.6.5. ファイルドロップ通知 (OnFileDrop)

形 式 : void OnFileDrop(object sender, VthArgFileDrop e);

パラメタ : int e.n - ドロップされたファイルの個数

説 明 : コントロールへ、ファイルがドロップされたことを通知します。
GetDroppedFile() メソッドにより、ドロップされたファイルのパス名を取得することができます。

戻り値 : なし

7.6.6. ディレクトリドロップ通知 (OnDirDrop)

形 式 : void OnDirDrop(object sender, VthArgDirDrop e);

パラメタ : int e.n - ドロップされたディレクトリの個数

説 明 : コントロールへ、ディレクトリがドロップされたことを通知します。
GetDroppedDir() メソッドにより、ドロップされたディレクトリのパス名を取得することができます。

戻り値 : なし

7.6.7. 文字サイズ変化通知 (OnCharInfo)

形 式 : void OnCharInfo(object sender, VthArgCharInfo e);

パラメタ : int e.LineHeight - 行の高さ (ピクセル数)

説 明 : コントロールのフォントが変更され、文字サイズが変更されたことを通知します。
CharWidth や CharaHeight プロパティにより、文字の幅や高さを取得できます。

戻り値 : なし

7.6.8. 右クリック通知 (OnRClick)

形 式 : void OnRClick (object sender, VthArgRClick e);

パラメタ : int e.x - 右クリックした X 位置 (コントロールの左上が原点)
 int e.y - " Y (")
 bool e.shift - Shift キーの押下状態 (true : 押下)
 bool e.ctrl - Ctrl キーの押下状態 (true : 押下)

説 明 : コントロール上で右クリックされたことを通知します。

Shift キーと Ctrl キーが押されていない状態で右クリックした場合、通常はポップアップメニューが表示されます。

但し、EnablePopupMenu プロパティが false (右クリックによるポップアップメニュー禁止) に設定されている場合は、ポップアップメニューが表示されず、右クリック通知イベントが発生します。

Shift キーか Ctrl キーが押されている状態で右クリックした場合は、EnablePopupMenu プロパティに関係なく常に右クリック通知イベントが発生します。

戻り値 : なし

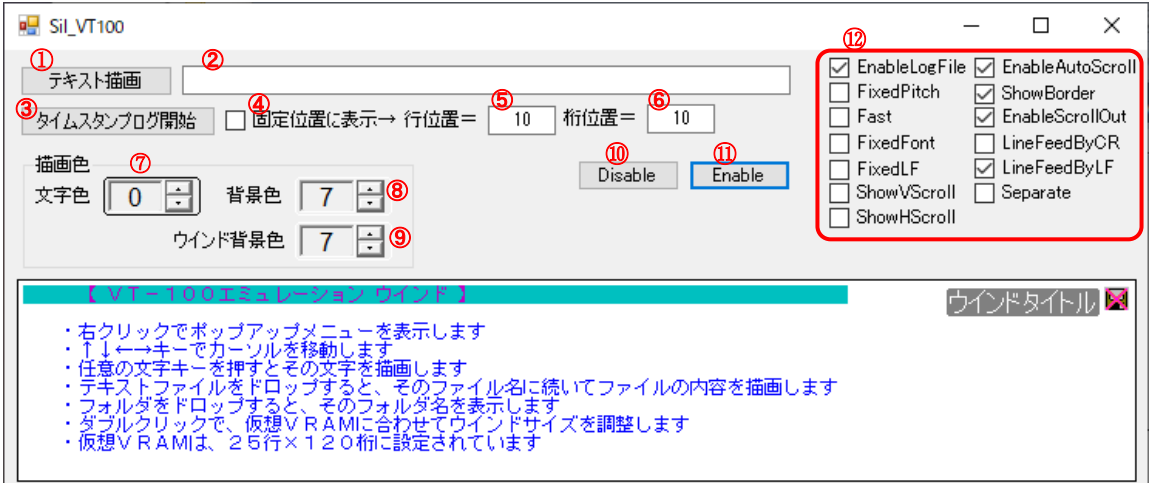
7.7. サンプルプログラム

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。

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。

ファイル（テキストファイル）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。

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	②のテキストをコントロールへ描画します	
②	VT100 コントロールへの描画テキスト	
③	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
④	チェックした場合、タイムスタンプを、⑤、⑥で指定された行／桁位置へ描画します	
⑤	タイムスタンプの描画行位置を設定します	
⑥	タイムスタンプの描画桁位置を設定します	
⑦	文字描画色（パレット番号）を指定します	0～7
⑧	背景描画色（パレット番号）を指定します	0～7
⑨	ウインド背景色（パレット番号）を指定します	0～7
⑩	VT100 コントロールが入力を受け付けないようにします	
⑪	VT100 コントロールが入力を受け付けるようにします	
⑫	各種プロパティ設定	

```
1 : using System;
2 : using System.Collections.Generic;
3 : using System.ComponentModel;
4 : using System.Data;
5 : using System.Drawing;
6 : using System.Linq;
7 : using System.Text;
8 : using System.Windows.Forms;
9 : using CAjrCustCtrl;
10 :
11 : namespace Sil_VT100
12 : {
13 :     public partial class Form1 : Form
14 :     {
15 :         int count = 0;
16 :         public Form1()
```

```

17 :     {
18 :         InitializeComponent();
19 :     }
20 : //----- フォーム初期化 -----//
21 : private void Form1_Load(object sender, EventArgs e)
22 : {
23 :     // スタイル設定チェックボックス初期化
24 :     chkEnableLogFile.Checked = vth.EnableLogFile;
25 :     chkFixedPitch.Checked = vth.FixedPitch;
26 :     chkFast.Checked = vth.Fast;
27 :     chkFixedFont.Checked = vth.FixedFont;
28 :     chkFixedLF.Checked = vth.FixedLF;
29 :     chkShowScroll.Checked = vth.ShowScroll;
30 :     chkShowHScroll.Checked = vth.ShowHScroll;
31 :     chkEnableAutoScroll.Checked = vth.EnableAutoScroll;
32 :     chkShowBorder.Checked = vth.ShowBorder;
33 :     chkEnableScrollOut.Checked = vth.EnableScrollOut;
34 :     chkLineFeedByCR.Checked = vth.LineFeedByCR;
35 :     chkLineFeedByLF.Checked = vth.LineFeedByLF;
36 :     chkSeparate.Checked = vth.Separate;
37 :     // 設定値ロード
38 :     SAjrReg.LoadAllCtrls(this);
39 :     // ウインド位置とサイズロード
40 :     SAjrReg.LoadWndRect(this);
41 :     // 初期メッセージ表示
42 :     vth.PutText("¥x1B[35;46m");
43 :     vth.PutText(" 【 VT-100 エミュレーション ウインド 】 ¥n");
44 :     vth.PutText("¥x1B[0m");
45 :     vth.PutText(" ¥n");
46 :     vth.PutText("¥x1B[34;47m");
47 :     vth.PutText(" ・右クリックでポップアップメニューを表示します ¥n");
48 :     vth.PutText(" ・↑↓←→キーでカーソルを移動します ¥n");
49 :     vth.PutText(" ・任意の文字キーを押すとその文字を描画します ¥n");
50 :     vth.PutText(" ・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");
51 :     vth.PutText(" ・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");
52 :     vth.PutText(" ・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");
53 :     vth.PutText(" ・仮想VRAMは、25行×120桁に設定されています ¥n");
54 :     vth.PutText(" ¥n");
55 :     vth.PutText("¥x1B[0m¥r¥n");
56 : }
57 : //----- フォーム終了 -----//
58 : private void Form1_FormClosed(object sender, FormClosedEventArgs e)
59 : {
60 :     // 設定値セーブ
61 :     SAjrReg.SaveAllCtrls(this);
62 :     // ウインド位置とサイズセーブ
63 :     SAjrReg.SaveWndRect(this);
64 : }
65 : //----- VT100 キー入力 -----//
66 : private void vth_OnNtcKeyIn(object sender, VthArgNtcKeyIn e)
67 : {
68 :     vth.PutChar(e.key);
69 : }
70 : //----- VT100 拡張キー押下 -----//
71 : private void vth_OnNtcVKeyIn(object sender, VthArgNtcVKeyIn e)
72 : {
73 :     if (e.rep == 0) { // キー押す続けによる繰り返し以外?
74 :         Point pt = SAjrGsr.GetWindowPos(vth);
75 :         SAjrTip.Show(pt.X + 2, pt.Y + 2, "¥x1B[34m キーが押されました ( " + e.vkey.ToString() + " ) ", 1000);
76 :     }
77 : }
78 : //----- VT100 キー離し通知 -----//
79 : private void vth_OnNtcVKeyOut(object sender, VthArgNtcVKeyOut e)
80 : {
81 :     Point pt = SAjrGsr.GetWindowPos(vth);
82 :     SAjrTip.Show(pt.X + 2, pt.Y + 2, "¥x1B[35m キーが離されました ( " + e.vkey.ToString() + " ) ", 1000);
83 : }
84 : //----- VT100 ファイルドロップ -----//
85 : private void vth_OnFileDrop(object sender, VthArgFileDrop e)
86 : {
87 :     for (int i = 0; i < e.n; i++) {
88 :         string path = vth.GetDroppedFile();
89 :         vth.PutText("¥n[[[ FILE = " + path + " ]]]¥n¥n");
90 :         try {
91 :             txf.Open(path, TextEncode.TEC_AUTO);
92 :         }
93 :         catch (Exception exc) {
94 :             vth.PutText(exc.ToString() + "¥n");
95 :             break;
96 :         }
97 :     }
98 : }

```

```

97 :         string txt;
98 :         while ((txt = txf.GetS()) != null) {
99 :             vth.PutText(txt);
100 :         }
101 :         txf.Close();
102 :     }
103 : }
104 : //----- VT100 ディレクトリドロップ -----//
105 : private void vth_OnDirDrop(object sender, VthArgDirDrop e)
106 : {
107 :     string txt = "ディレクトリがドロップされました¥n";
108 :     for (int i = 0; i < e.n; i++) {
109 :         txt += (" " + vth.GetDroppedDir() + "¥n");
110 :     }
111 :     SAjrTip.ShowCenter(vth, txt);
112 : }
113 : //----- VT100 右クリック -----//
114 : private void vth_OnRClick(object sender, VthArgRClick e)
115 : {
116 :     SAjrTip.ShowCenter(vth, "¥n 右クリックしました (x=" + e.x.ToString() + ", y=" + e.y.ToString() + ") ¥n");
117 : }
118 : //----- VT100 ダブルクリック -----//
119 : private void vth_OnNtcDb1Clk(object sender, VthArgDb1Clk e)
120 : {
121 :     SAjrTip.ShowCenter(vth, "¥n ダブルクリックしました¥n");
122 : }
123 : //----- タイマ -----//
124 : private void tim_Tick(object sender, EventArgs e)
125 : {
126 :     count++;
127 :     // カーソル移動
128 :     if (chkFixedPos.Checked) {
129 :         int line = int.Parse(txtLine.Text);
130 :         int col = int.Parse(txtCol.Text);
131 :         vth.Locate(line, col);
132 :     }
133 :     // タイムスタンプログ表示
134 :     DateTime dt = DateTime.Now;
135 :     vth.PutText(string.Format("{0,4} : {1:D2}:{2:D2}:{3:D2}.{4:D3}", count,
136 :                               dt.Hour, dt.Minute, dt.Second, dt.Millisecond));
137 :     // 改行, 行クリアー
138 :     if (!chkFixedPos.Checked) {
139 :         vth.PutText("¥n¥x1B[K");
140 :     }
141 : }
142 : //----- テキスト描画ボタン -----//
143 : private void btnPutText_Click(object sender, EventArgs e)
144 : {
145 :     vth.PutText(txtPutText.Text + "¥n");
146 : }
147 : //----- 「タイムスタンプ ログ開始/停止」ボタン -----//
148 : private void btnLog_Click(object sender, EventArgs e)
149 : {
150 :     if (tim.Enabled) {
151 :         tim.Enabled = false;
152 :         btnLog.Text = "タイムスタンプ ログ開始";
153 :     }
154 :     else {
155 :         tim.Enabled = true;
156 :         btnLog.Text = "タイムスタンプ ログ停止";
157 :     }
158 : }
159 : //----- 「文字色」入力 -----//
160 : private void InpTextColor_OnNtcIntValue(object sender, IvArgNtcIntValue e)
161 : {
162 :     vth.ForegroundColor = e.value;
163 :     InpTextColor.BorderColor = vth.GetPaletteColor(e.value);
164 : }
165 : //----- 「背景色」入力 -----//
166 : private void inpBkColor_OnNtcIntValue(object sender, IvArgNtcIntValue e)
167 : {
168 :     vth.BackgroundColor = e.value;
169 :     inpBkColor.BorderColor = vth.GetPaletteColor(e.value);
170 : }
171 : //----- 「ウインド背景色」入力 -----//
172 : private void inpWndBkColor_OnNtcIntValue(object sender, IvArgNtcIntValue e)
173 : {
174 :     vth.WndBkColor = e.value;
175 :     inpWndBkColor.BorderColor = vth.GetPaletteColor(e.value);
176 : }

```

```

177 : //----- 「Enable」 ボタン -----//
178 : private void btnEnable_Click(object sender, EventArgs e)
179 : {
180 :     vth.Enabled = true;
181 : }
182 : //----- 「Disable」 ボタン -----//
183 : private void btnDisable_Click(object sender, EventArgs e)
184 : {
185 :     vth.Enabled = false;
186 : }
187 : //----- スタイル設定チェックボックス -----//
188 : private void chkEnableLogFile_CheckedChanged (object sender, EventArgs e) {vth. EnableLogFile = chkEnableLogFile.Checked;}
189 : private void chkFixedPitch_CheckedChanged (object sender, EventArgs e) {vth. FixedPitch = chkFixedPitch.Checked;}
190 : private void chkFast_CheckedChanged (object sender, EventArgs e) {vth. Fast = chkFast.Checked;}
191 : private void chkFixedFont_CheckedChanged (object sender, EventArgs e) {vth. FixedFont = chkFixedFont.Checked;}
192 : private void chkFixedLF_CheckedChanged (object sender, EventArgs e) {vth. FixedLF = chkFixedLF.Checked;}
193 : private void chkShowVScroll_CheckedChanged (object sender, EventArgs e) {vth. ShowVScroll = chkShowVScroll.Checked;}
194 : private void chkShowHScroll_CheckedChanged (object sender, EventArgs e) {vth. ShowHScroll = chkShowHScroll.Checked;}
195 : private void chkEnableAutoScroll_CheckedChanged (object sender, EventArgs e) {vth. EnableAutoScroll = chkEnableAutoScroll.Checked;}
196 : private void chkShowBorder_CheckedChanged (object sender, EventArgs e) {vth. ShowBorder = chkShowBorder.Checked;}
197 : private void chkEnableScrollOut_CheckedChanged (object sender, EventArgs e) {vth. EnableScrollOut = chkEnableScrollOut.Checked;}
198 : private void chkLineFeedByCR_CheckedChanged (object sender, EventArgs e) {vth. LineFeedByCR = chkLineFeedByCR.Checked;}
199 : private void chkLineFeedByLF_CheckedChanged (object sender, EventArgs e) {vth. LineFeedByLF = chkLineFeedByLF.Checked;}
200 : private void chkSeparate_CheckedChanged (object sender, EventArgs e) {vth. Separate = chkSeparate.Checked;}
201 :
202 :
203 : }
204 : }

```

8. 数値入力コントロール (CAjrInpValue.dll)

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。



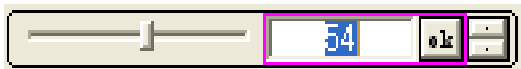
左右のスライダとスピンボタンで、値の増減を行います。

数値が変更されると、OnIntValueChanged／OnRealValueChanged イベントが発生し、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／ESC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

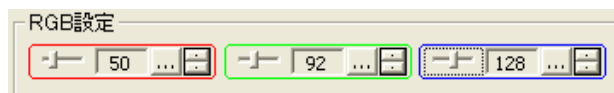
「AutoEdit」プロパティを true に設定した場合、テキストボックスをクリックすると、「...」ボタン押下と同様に)数値の直接入力が可能となります。

この場合、ボタン非表示(ShowButton=false)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、プロパティにより「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。

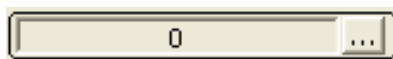


スライダを非表示

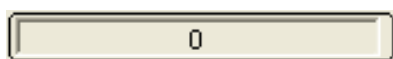


ボタンを非表示

(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ます) ※1

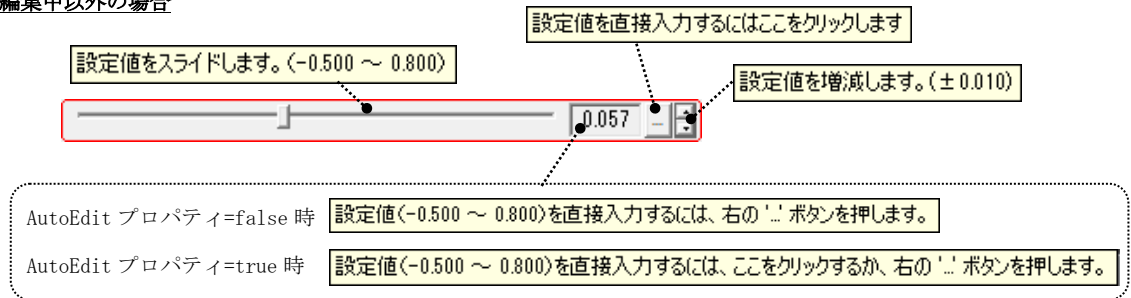
※1 : 「AutoEdit」プロパティが設定されていない場合は、テキストボックスのクリックによる数値の直接入力はできません。

8.1. ツールチップテキスト

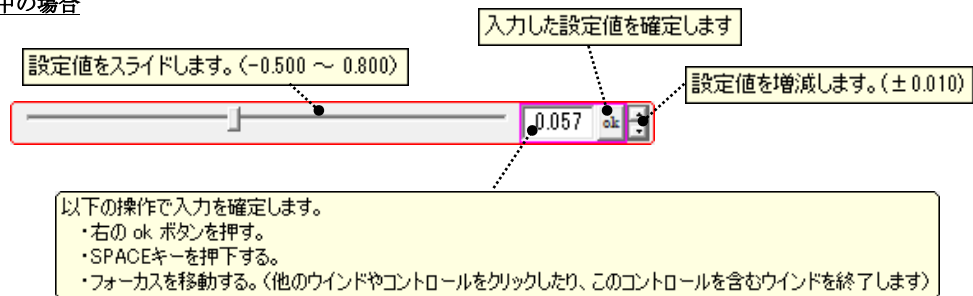
数値入力コントロールでは、デフォルトのツールチップ表示機能が用意されています。
 EnableDefToolTip プロパティを true (許可) に設定すると、デフォルトのツールチップを表示するようになります。

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 (スライダー, 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)

テキスト編集以外の場合



テキスト編集の場合



EnableDefToolTip プロパティを false (禁止) に設定すると、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、false(禁止状態)となっています)

ユーザの自由なツールチップテキストを設定するには、EnableDefToolTip プロパティを false (禁止) に設定し、ToolTipText プロパティにツールチップテキスト文字列を設定します。

あるいは、チップテキストコントロールで、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」参照)

```
TipSetText(inp, "ツールチップテキスト");  
[ TipSetCallBack(inp, 0, cbNeedText); ]
```

※ 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)
 ※ TipSetCallBack () 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が可能です。

8.2. プロパティ

数値入力コントロールのプロパティ一覧を示します。

#	名称	タイプ	内容	デフォルト
1	_RealMode	bool	実数モード・フラグ (※1)	false (整数モード)
2	AutoEdit	bool	テキストボックス・クリックによる数値入力を許可	true
3	BlinkOnInput	bool	数値直接入力時に、確定するまでブリンクする	true
4	BorderColor	Color	コントロール外枠の表示色	黒
5	EnabledDefToolTip	bool	デフォルト・ツールチップの許可(true)／禁止(false)	false
6	HexLen	int	16進数の表示桁数 (ShowHexadecimal=true 時のみ有効)	0
7	IntValue	int	整数値	0
8	MaxValue	double	入力数値の最大値	100
9	MinValue	double	入力数値の最小値	0
10	NotifyWhenSet	bool	数値の設定時にイベント (OnNtc {Int/Real} Value) を発生	false (発生しない)
11	Precision	int	小数部の桁数 (負数字の場合は、有効数字の桁数)	3
12	Separate	bool	true で、整数部分を 3 桁毎にカンマで区切る	false
13	ShowBorder	bool	コントロールの外枠表示フラグ	true
14	ShowButton	bool	数値直接入力ボタンの表示フラグ	true
15	ShowHexadecimal	bool	整数の 16 進表示 (true : 16 進, false: 10 進)	false
16	ShowSlider	bool	スライダ表示フラグ	true
17	ShowSpinButton	bool	スピンボタン表示フラグ	true
18	SliderPageSize	double	スライダのページサイズ (UnitValue 以上の値)	10
19	SpinStep	double	スピンボタンによる増減値 (UnitValue 以上の値)	1
21	TextLength	int	数値テキスト部分の表示桁数	6
22	ToolTipText	string	コントロールのツールヒント文字列 (※2)	"" (空文字列)
23	ToolTipShowAlways	bool	ツールチップ表示条件 true - 非アクティブ状態でも表示 false - 非アクティブ状態時は非表示	true
24	UnitValue	double	数値の最小単位	1
25	Value	double	実数値	0.0

※1 : 実数モードで使用する場合は、最初にこのプロパティを「true」に設定してください。

※2 : ToolTipText プロパティは、制御文字とエスケープシーケンスを含めることができます。
これについては、「テキスト表示拡張機」の章を参照してください。

8.3. メソッド

8.3.1. 値の設定(SetValue)

形 式 : void SetValue(double value); ----- 通知イベントを発生させない
 void SetValue(double value, bool fNtc); -- 通知イベントの発生は「fNtc」の指定に依存

引 数 : value - 設定する値
 fNtc - 通知イベント生成フラグ (true:通知イベントを発生させる, false:発生させない)

説 明 : 数値入力コントロールの値を設定します。
 fNtc=true を指定した場合、値を設定後に通知イベント (OnNtcIntValue / OnNtcRealValue) を発生します。
 fNtc=false を指定した場合 (あるいは fNtc の指定が無い場合) は、値を設定後に通知イベントは発生しません。

このメソッドは整数モード／実数モード兼用です。(整数モードでも本メソッドを使用できます)

戻り値 : なし

8.4. イベント

数値入力・コントロールのイベント一覧を以下に示します。

#	イベント名	内容
1	OnNtcIntValue	数値が設定したことを通知します。(_RealMode プロパティが「false」(整数モード)時) (※1)
2	OnNtcRealValue	数値が設定したことを通知します。(_RealMode プロパティが「true」(実数モード)時) (※1)
3	OnRClick	右クリックされたことを通知します

※1 : Ver1.6.0.3 までは、プログラムから値を設定した場合 (IntValue/Value に値を設定した場合) も、通知イベント (OnNtcIntValue / OnNtcRealValue) を発生していました。
現在では、通知イベント (OnNtcIntValue / OnNtcRealValue) 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、スピンドットで値を設定した場合
- ・SetValue() メソッドで、fNtc=true を指定した場合

8.4.1. 整数モードでの数値設定通知 (OnNtcIntValue)

形 式 : void OnNtcIntValue(object sender, IvArgNtcRealValue e);

パラメタ : int e.value - 変化後の整数値

説 明 : 整数モードで、スライダ/テキストボックス/スピンドットにより値が設定されたことを通知します。
または、SetValue() メソッドで、fNtc=true を指定した場合も発生します。

戻り値 : なし

8.4.2. 実数モードでの数値設定通知 (OnNtcRealValue)

形 式 : void OnNtcRealValue(object sender, IvArgNtcRealValue e);

パラメタ : double e.value - 変化後の実数値

説 明 : 実数モードで、スライダ/テキストボックス/スピンドットにより値が設定されたことを通知します。
または、SetValue() メソッドで、fNtc=true を指定した場合も発生します。

戻り値 : なし

8.4.3. 右クリック通知 (OnNtcRClick)

形 式 : void OnRClick(object sender, IvArgNtcRClick e);

パラメタ : int e.x - 右クリックした X 位置 (コントロールの左上が原点)
int e.y - " Y (")
bool shift - Shift キーの押下状態
bool ctrl - Ctrl キーの押下状態

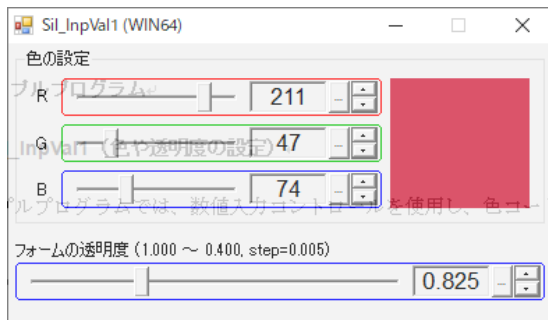
説 明 : コントロール上で右クリックされたことを通知します。

戻り値 : なし

8.5. サンプルプログラム

8.5.1. Sil_InpVal1 (色や透明度の設定)

このサンプルプログラムでは、数値入力コントロールを使用し、色コードの設定と透明度の設定を行います。



```

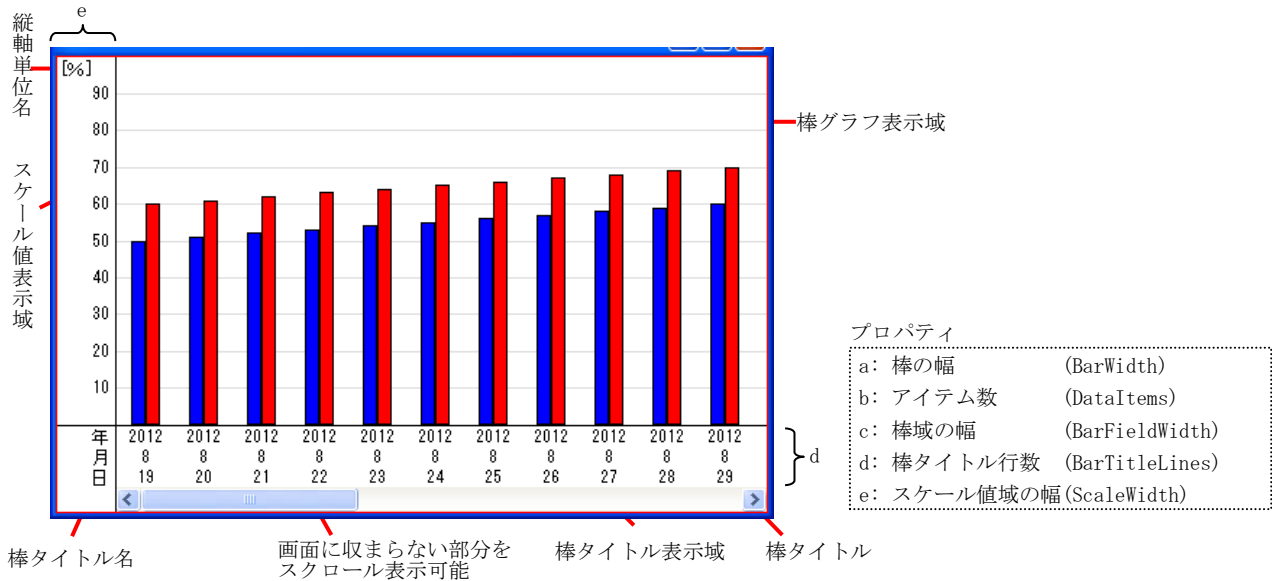
1 : //
2 : // Sil_InpVal1
3 : //
4 : using System;
5 : using System.Collections.Generic;
6 : using System.ComponentModel;
7 : using System.Data;
8 : using System.Drawing;
9 : using System.Text;
10 : using System.Windows.Forms;
11 : using CAjrCustCtrl;
12 :
13 : namespace Sil_InpVal1
14 : {
15 :     public partial class Form1 : Form
16 :     {
17 :         Color m_color = Color.Black;
18 :
19 :         public Form1()
20 :         {
21 :             InitializeComponent();
22 :         }
23 :         //==== 起動時初期設定 =====//
24 :         private void Form1_Load(object sender, EventArgs e)
25 :         {
26 :             this.FormBorderStyle = FormBorderStyle.FixedSingle;
27 :             this.Text = this.Text + (IntPtr.Size == 4 ? " (WIN32)" : " (WIN64)");
28 :             pic.Invalidate();
29 :         }
30 :         //==== ピクチャ描画イベント =====//
31 :         private void pic_Paint(object sender, PaintEventArgs e)
32 :         {
33 :             SolidBrush b = new SolidBrush(m_color);
34 :             e.Graphics.FillRectangle(b, 0, 0, pic.Size.Width, pic.Size.Height);
35 :             b.Dispose();
36 :         }
37 :         //==== R 値通知 =====//
38 :         private void inpR_OnNtcIntValue(object sender, IvArgNtcIntValue e)
39 :         {
40 :             m_color = Color.FromArgb(inpR.IntValue, inpG.IntValue, inpB.IntValue);
41 :             pic.Invalidate();
42 :         }
43 :         //==== G 値通知 =====//
44 :         private void inpG_OnNtcIntValue(object sender, IvArgNtcIntValue e)
45 :         {
46 :             m_color = Color.FromArgb(inpR.IntValue, inpG.IntValue, inpB.IntValue);
47 :             pic.Invalidate();
48 :         }

```

```
49 : //===== B 値通知 =====//
50 : private void inpB_OnNtcIntValue(object sender, IvArgNtcIntValue e)
51 : {
52 :     m_color = Color.FromArgb(inpR.IntValue, inpG.IntValue, inpB.IntValue);
53 :     pic.Invalidate();
54 : }
55 : //===== A 値(フォームの透明度)通知 =====//
56 : private void inpA_OnNtcRealValue(object sender, IvArgNtcRealValue e)
57 : {
58 :     this.Opacity = e.value;
59 : }
60 : }
61 : }
```

9. 棒グラフ／折れ線グラフ・コントロール (CAjrBarGraph.dll)

棒グラフ／折れ線グラフをリアルタイムに表示するコントロールです。
棒グラフ表示時の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。

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、SetItemColor() メソッドで変更できます。

9.1. 機能概要

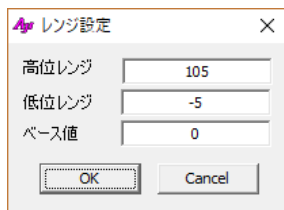
ポップアップメニュー

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(C)	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(R)	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(F)	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(D)	データクリア	: バッファリングされているデータを全て破棄し、画面をクリアします。

レンジ設定

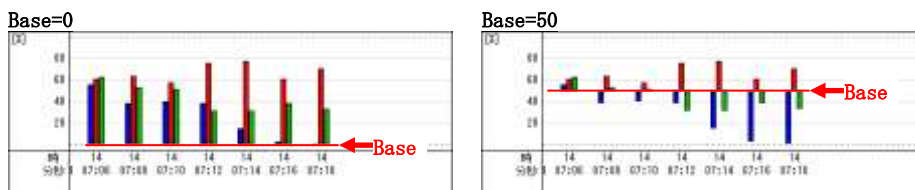
ポップアップメニューで「レンジ設定」を選択すると、以下のダイアログボックスが表示されます。



レンジ設定ダイアログボックスのスクリーンショット。タイトルは「レンジ設定」で、閉じるボタン「X」があります。フィールドには「高位レンジ」が105、「低位レンジ」が-5、「ベース値」が0と入力されています。下部には「OK」と「Cancel」のボタンがあります。

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。

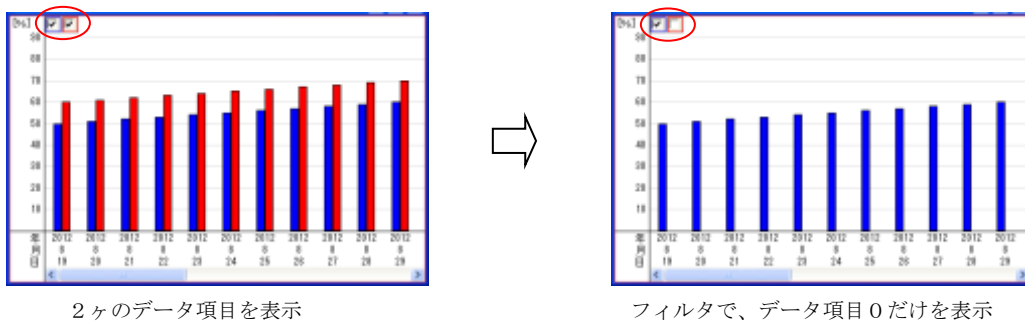
ベース値とは、棒グラフ描画の基点を意味します。



9.1.1. 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ッ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。

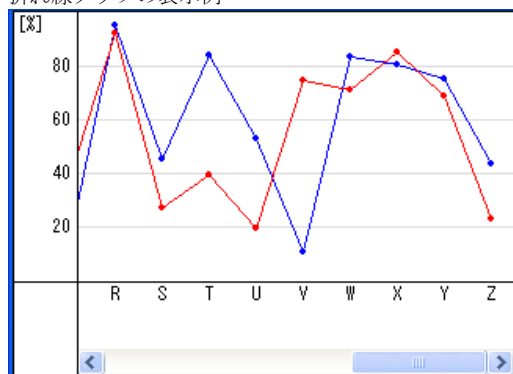


9.1.2. 折れ線グラフ

「LineChart」プロパティを true に設定すると、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「BarFieldWidth」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



9.1.3. 右クリック

右クリック操作による動作は、以下のようになっています。

ポップアップメニュー許可 EnablePopupMenu プロパティ	SHIFT/CTRL キー押下	動作
True (許可)	× (未押下)	ポップアップメニューを表示
True (許可)	○ (押下)	右クリック通知イベント (OnRClick) 発生 (右ボタン押下時に発生)
False (禁止)	—	右クリック通知イベント (OnRClick) 発生 (右ボタン押下時に発生)

9.1.4. 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、以下のイベントが発生します。

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

これらのイベントでは、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 GetDroppedFile/GetDroppedDir メソッド 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、AcceptFiles プロパティを true に指定する必要があります。

9.2. プロパティ

棒グラフ表示/折れ線グラフ・コントロールのプロパティ一覧を示します。

#	名称	タイプ	内容	デフォルト
1	AcceptFiles	bool	ファイル/フォルダのドロップ許可フラグ	false
2	BarFieldWidth	int	棒域の幅 (ピクセル数, 4 以上)	30
3	BarTitleLines	int	棒タイトル行数 (1 以上)	3
4	BarWidth	int	棒の幅 (ピクセル数, 10 以上)	10
5	Base	double	棒グラフのベース値	0
6	CharHeight	int	文字の高さ (ピクセル数, 読み出し専用)	-
7	CharWidth	int	文字の幅 (ピクセル数, 読み出し専用)	-
8	DataItems	int	データ項目数 (アイテム数, 1 ~ 8)	1
9	DataTitle	string	棒タイトル名	"Data-Title"
10	EnablePopupMenu	bool	ポップアップメニュー許可フラグ(※1)	true
11	FontTxo	Font	テキスト描画時のフォント	MS UI Gothic, 9
12	LineChart	bool	折れ線グラフでの表示フラグ	false
13	MaxBuf	int	最大データ保留数	1024
14	RangeHigh	double	グラフ高位レンジ	100
15	RangeLow	double	グラフ低位レンジ	0
16	ScaleWidth	int	スケール値表示域の幅 (ピクセル数, 32 以上)	64
17	ShowBorder	bool	コントロール外枠の表示フラグ	true
18	ShowDummyData	bool	デザイン時のダミーデータ表示フラグ(※2)	true
19	ShowFilter	bool	フィルタ(チェックボックス)表示フラグ	true
20	ToolTipFilter0~7	string	フィルタの各チェックボックスのツールヒント (※3)	""
21	ToolTipText	string	コントロールのツールヒント (※3)	""
22	ToolTipShowAlways	bool	ツールチップ表示条件 true - 非アクティブ状態でも表示 false - 非アクティブ状態時は非表示	true
23	UnitName	string	縦軸単位名	"[UnitName]"
24	TitleText	string	タイトルテキスト(文字色=白, 背景=グレーで右上に表示)	""

※

1 : 右クリックによるポップアップメニューの表示(true)/非表示(false)を設定します。

非表示(false)を設定した場合は、右クリックすると「OnRClick」イベントが発生します。

(Shift/Ctrl + 右クリックした場合は、EnablePopupMenu プロパティに関係なく常に「OnRClick」イベントが発生します)

※2 : デザイン時にダミーデータを表示することにより、プロパティの効果を視覚的に確認することができます。

※3 : ToolTipText, ToolTipFilter0~7 プロパティは、制御文字とエスケープシーケンスを含めることができます。

これについては、「テキスト表示拡張機」の章を参照してください。

9.3. メソッド

棒グラフ／折れ線グラフ・コントロールのメソッド一覧を以下に示します。

#	メソッド名	内容	備考
1	PutData	データ投与	
2	SetItemColor	データ項目の表示色設定	
3	GetItemColor	データ項目の表示色取得	
4	Purge	データ破棄	
5	SetScrollPos	スクロール位置の設定	
6	SetGcrollPos	スクロール位置の取得	
7	SetFilter	フィルタの設定	
8	GetFilter	フィルタの取得	
9	SetHoriLineStyle	横線スタイル設定	
10	SetHoriLinePos	横線描画位置設定	
11	EnableHoriLine	横線描画の許可／禁止	
12	GetDroppedFile	ドロップされたファイル名取得	
13	GetDroppedDir	ドロップされたディレクトリ名取得	
14	SetTitleText	タイトルテキスト表示	画面右上に表示
15	SaveToProfile	現在の設定値をプロファイルへ記録する	
16	LoadFromProfile	設定値をプロファイルから読み出す	
17	TextOut	テキスト描画 (ピクセル位置指定)	
18	PurgePlot	グラフデータ破棄	
19	PurgeText	テキスト描画データ破棄 (描画したテキストのクリアー)	
20	Purge	全てのデータ (グラフデータ, 描画テキスト) 破棄	

9.3.1. データ投与(PutData)

形 式 : void PutData(double d1);
 void PutData(double d1, double d2);
 void PutData(double d1, double d2, double d3);
 void PutData(double d1, double d2, double d3, double d4);
 void PutData(double d1, double d2, double d3, double d4, double d5);
 void PutData(double d1, double d2, double d3, double d4, double d5, double d6);
 void PutData(double d1, double d2, double d3, double d4, double d5, double d6, double d7);
 void PutData(double d1, double d2, double d3, double d4, double d5, double d6, double d7, double d8);

引 数 : d1～d8 - グラフに投与するデータ

説 明 : データを投与し、グラフを更新します。
 DataItems プロパティで指定したデータ項目数のデータを投与してください。

戻り値 : なし

9.3.2. データ項目の表示色設定(SetItemColor)

形 式 : void SetItemColor(int ix, Color color);

引 数 : ix - データ項目番号 (0～7)
 Color - 表示色

説 明 : 当該データ項目の表示色を設定します。

戻り値 : なし

9.3.3. データ項目の表示色取得(GetItemColor)

形 式 : Color GetItemColor(int ix);

引 数 : ix - データ項目番号 (0～7)

説 明 : 当該データ項目の表示色を取得します。

戻り値 : 当該データ項目の表示色

9.3.4. データ破棄(Purge)

形 式 : void Purge();

引 数 : なし

説 明 : バッファに格納されているデータを全て破棄します。

戻り値 : なし

9.3.5. スクロール位置の設定 (SetScrollPos)

形 式 : void SetScrollPos (int pos);

引 数 : なし

説 明 : pos で指定された位置までスクロールして、グラフを表示します。
スクロール位置を設定する場合は、データの投与を停止している状態で行ってください。

戻り値 : なし

9.3.6. スクロール位置の取得 (SetScrollPos)

形 式 : int SetScrollPos ();

引 数 : なし

説 明 : 現在のスクロール位置（グラフ左端データの最古データからの相対位置）を取得します。

戻り値 : 現在のスクロール位置

9.3.7. フィルタの設定(SetFilter)

形 式 : void SetFilter (int ix, bool fEnable);

引 数 : ix - データ項目番号 (0～7)
fEnable - フィルタ設定値

説 明 : 当該データ項目のフィイルタ（チェックボックス）を設定します。
fEnable=True を指定すると当該データ項目は表示され、fEnable=false を指定すると非表示となります。

戻り値 : なし

9.3.8. フィルタの設定値の取得(GetFilter)

形 式 : bool GetFilter (int ix);

引 数 : ix - データ項目番号 (0～7)
fEnable - フィルタ設定値

説 明 : 当該データ項目のフィイルタ（チェックボックス）の設定値を取得します。

戻り値 : 当該データ項目のフィイルタ（チェックボックス）の設定値

9.3.9. 横線スタイル設定(SetHoriLineStyle)

形 式 : void SetHoriLineStyle (int ix, Color color, int width, ETchLineStyle style);

引 数 : ix - 横線データ識別 (0～7)
 color - 横線の表示色
 width - 線の幅 (1～, 点線等、実線以外のスタイルを指定する場合は1を指定すること)
 style - 横線の描画スタイル

説 明 : 横線の描画属性を指定します。
 グラフ上に描画できる横線は、最大8ヶであり、ix 引数で指定します。

戻り値 : なし

9.3.10. 横線描画位置設定(SetHoriLinePos)

形 式 : void SetHoriLinePos (int ix, double pos);

引 数 : ix - 横線データ識別 (0～7)
 pos - 横線の描画位置

説 明 : pos で指定した位置に横線を描画します。
 グラフ上に描画できる横線は、最大8ヶであり、ix 引数で指定します。

戻り値 : なし

9.3.11. 横線表示／非表示(EnableHoriLine)

形 式 : void EnableHoriLine (int ix, bool fEnable);

引 数 : ix - 横線データ識別 (0～7)
 fEnable - 横線の表示／非表示フラグ

説 明 : 横線を表示 (true) あるいは、非表示 (false) します。
 グラフ上に描画できる横線は、最大8ヶであり、ix 引数で指定します。

戻り値 : なし

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

形 式 : string GetDroppedFile();

引 数 : なし

説 明 : OnFileDrop イベント発生時に、順次、ドロップされたファイルのパス名を取得します。
 OnFileDrop イベントで通知された、ドロップファイル数の回数だけ本メソッドをこーするすることにより、ドロップされ
 た全てのファイルを取得できます。

戻り値 : ≠"" : ドロップされたファイルパス名
 ="" : 終端(ドロップされたファイルパス名の取得は完了済である)

9.3.13. ドロップされたディレクトリ名取得 (GetDroppedDir)

形 式 : `string GetDroppedDir();`

引 数 : なし

説 明 : OnDirDrop イベント発生時に、順次、ドロップされたディレクトリのパス名を取得します。
OnDirDrop イベントで通知された、ドロップディレクトリ数の回数だけ本メソッドをこーするすることにより、ドロップされた全てのディレクトリを取得できます。

戻り値 : `≠""` : ドロップされたディレクトリパス名
`=""` : 終端(ドロップされたディレクトリパス名の取得は完了済である)

9.3.14. タイトルテキスト表示 (SetTitleText)

形 式 : `void SetTitleText(string TitleText);`
`void SetTitleText(string TitleText, Color TextColor);`
`void SetTitleText(string TitleText, Color TextColor, Color BackColor);`
`void SetTitleText(string TitleText, Color TextColor, Color BackColor, Font TextFont);`

引 数 : TitleText - タイトルテキスト (null(あるいは空文字列(""))を指定した場合は非表示)
TextColor - テキスト描画色 (α値は無視されます)
BackColor - テキスト背景色 (α値は無視されます)
TextFont - テキストのフォント

説 明 : コントロールウインドの右上にタイトルテキストを表示します。
SetTitleText に null(あるいは空文字列(""))を指定した場合、タイトルテキストは非表示となります。

戻り値 : なし

9.3.15. 現在の設定値をプロファイルへ記録する (SaveToProfile)

形 式 : `void SaveToProfile (string section);`

引 数 : section - プロファイル内のセクション名

説 明 : 現在の設定値 (フィルタ設定やプロパティ) をプロファイルに記録します。

戻り値 : なし

9.3.16. 設定値をプロファイルから読み出す (LoadFromProfile)

形 式 : `void LoadFromProfile (string section);`

引 数 : section - プロファイル内のセクション名

説 明 : SaveToProfile() によりプロファイルに記録した設定値を読み出します。
読み出した内容は、そのままフィルタやプロパティに設定されます。

戻り値 : なし

9.3.17. テキスト描画 (TextOut) - ピクセル位置指定

形 式 : `int TextOut(int x, int y, string text);`

引 数 : `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、「Txo.Right ± n」or「Txo.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、「Txo.Bottom ± n」or「Txo.Center ± n」を指定)
`text` - 描画するテキスト

説 明 : グラフウインド上にテキストを描画します。
 描画するテキストには、エスケープシーケンスや制御文字を含めることができます。(「テキスト表示拡張機能」の章参照)
`x = Txo.Right` , `y = Txo.Bottom` は、テキストをウインドの右下隅に表示する位置となります。
`x = Txo.Center` , `y = Txo.Center` は、テキストをウインドの中央に表示する位置となります。

戻り値 : テキストキー

9.3.18. グラフデータ破棄(PurgePlot)

形 式 : `void PurgePlot();`

引 数 : なし

説 明 : グラフデータを破棄します。

戻り値 : なし

9.3.19. テキスト描画データ破棄(PurgeText)

形 式 : `void PurgeText(int key);` // 指定 key の描画テキスト破棄
`void PurgeText();` // すべての描画テキスト破棄

引 数 : `key` - データ項目番号 (TextOut() の戻り値)

説 明 : TextOut() で描画したテキストを破棄します。

戻り値 : なし

9.3.20. 全てのデータ破棄(Purge)

形 式 : `void Purge();`

引 数 : なし

説 明 : すべてのデータ (グラフデータ, 描画テキスト) を破棄します。

戻り値 : なし

9.4. イベント

棒グラフ／折れ線グラフ・コントロールのイベント一覧を以下に示します。

#	イベント名	内容
1	OnRangeChanged	グラフのレンジが変更されたことを通知します
2	OnFileDrop	ファイルドロップ通知
3	OnDirDrop	ディレクトリドロップ通知
4	OnRClick	右クリックされたことを通知します

9.4.1. レンジ通知 (OnRangeChanged)

形 式 : void OnRangeChanged (object sender, TchArgRangeChanged e);

パラメタ : double e.low - グラフの低位レンジ
double e.high - グラフの高位レンジ

説 明 : グラフのレンジが変更されたことを通知します。

戻り値 : なし

9.4.2. ファイルドロップ通知 (OnFileDrop)

形 式 : void OnFileDrop(object sender, VthArgFileDrop e);

パラメタ : int e.n - ドロップされたファイルの個数

説 明 : コントロールへ、ファイルがドロップされたことを通知します。
GetDroppedFile() メソッドにより、ドロップされたファイルのパス名を取得することができます。

戻り値 : なし

9.4.3. ディレクトリドロップ通知 (OnDirDrop)

形 式 : void OnDirDrop(object sender, VthArgDirDrop e);

パラメタ : int e.n - ドロップされたディレクトリの個数

説 明 : コントロールへ、ディレクトリがドロップされたことを通知します。
GetDroppedDir() メソッドにより、ドロップされたディレクトリのパス名を取得することができます。

戻り値 : なし

9.4.4. 右クリック通知 (OnRClick)

形 式 : void OnRClick (object sender, TchArgRClick e);

パラメタ : int e.x - 右クリックした X 位置 (コントロールの左上が原点)
 int e.y - " Y (")
 bool shift - Shift キーの押下状態
 bool ctrl - Ctrl キーの押下状態

説 明 : コントロール上で右クリックされたことを通知します。

Shift キーと Ctrl キーが押されていない状態で右クリックした場合、通常はポップアップメニューが表示されます。

但し、EnablePopupMenu プロパティが false (右クリックによるポップアップメニュー禁止) に設定されている場合は、ポップアップメニューが表示されず、右クリック通知イベントが発生します。

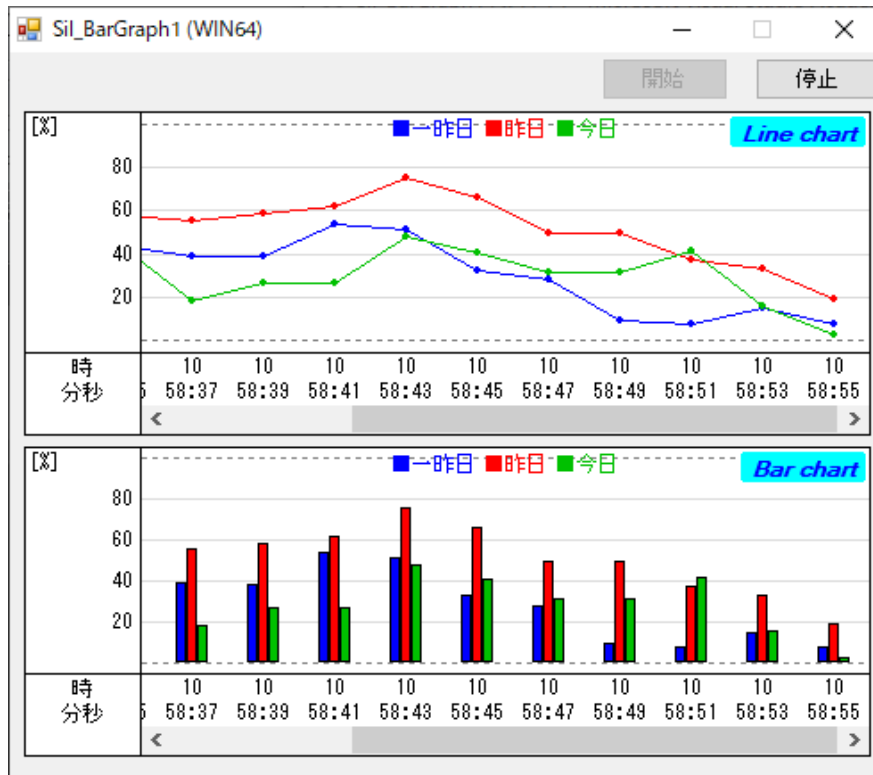
Shift キーか Ctrl キーが押されている状態で右クリックした場合は、EnablePopupMenu プロパティに関係なく常に右クリック通知イベントが発生します。

戻り値 : なし

9.5. サンプルプログラム

9.5.1. Sil_BarGraph1 (棒グラフと折れ線グラフ)

このサンプルプログラムでは、ランダムなデータ (1~100) を発生させ、棒グラフと折れ線グラフを表示します。
現在の日付と時間を各データのタイトルとします。



```

1 : //
2 : // Sil_BarGraph1
3 : //
4 : using System;
5 : using System.Collections.Generic;
6 : using System.ComponentModel;
7 : using System.Data;
8 : using System.Drawing;
9 : using System.Text;
10 : using System.Windows.Forms;
11 : using CAjrCustCtrl;
12 :
13 : namespace Sil_BarGraph1
14 : {
15 :     public partial class Form1 : Form
16 :     {
17 :         double d1 = 40.0;
18 :         double d2 = 50.0;
19 :         double d3 = 60.0;
20 :         Random rnd = new System.Random();
21 :
22 :         public Form1()
23 :         {
24 :             InitializeComponent();
25 :         }
26 :         //----- 起動時初期設定 -----//
27 :         private void Form1_Load(object sender, EventArgs e)
28 :         {
29 :             this.FormBorderStyle = FormBorderStyle.FixedSingle;
30 :             this.Text = this.Text + (IntPtr.Size == 4 ? " (WIN32)" : " (WIN64)");
31 :

```

```

32 : //----- タイトルテキスト設定 -----//
33 : bar.SetTitleText(" Bar chart ", Color.Blue, Color.Aqua,
34 :                 new Font("Terminal", 9, FontStyle.Bold | FontStyle.Italic));
35 : lin.SetTitleText(" Line chart ", Color.Blue, Color.Aqua,
36 :                 new Font("Terminal", 9, FontStyle.Bold | FontStyle.Italic));
37 : //----- テキスト描画 -----//
38 : bar.TextOut(Txo.Center, 3, "¥x1B[30m■一昨日 ¥x1B[31m■昨日 ¥x1B[32m■今日");
39 : lin.TextOut(Txo.Center, 3, "¥x1B[30m■一昨日 ¥x1B[31m■昨日 ¥x1B[32m■今日");
40 : //----- 0.0 と 100.0 の位置に点線表示 -----//
41 : bar.SetHoriLinePos (0, 0.0);
42 : bar.SetHoriLineStyle(0, Color.Gray, 1, EBarLineStyle.Dot);
43 : bar.EnableHoriLine (0, true);
44 :
45 : bar.SetHoriLinePos (1, 100.0);
46 : bar.SetHoriLineStyle(1, Color.Gray, 1, EBarLineStyle.Dot);
47 : bar.EnableHoriLine (1, true);
48 :
49 : lin.SetHoriLinePos (0, 0.0);
50 : lin.SetHoriLineStyle(0, Color.Gray, 1, EBarLineStyle.Dot);
51 : lin.EnableHoriLine (0, true);
52 :
53 : lin.SetHoriLinePos (1, 100.0);
54 : lin.SetHoriLineStyle(1, Color.Gray, 1, EBarLineStyle.Dot);
55 : lin.EnableHoriLine (1, true);
56 :
57 : //----- ウインド位置ロード -----//
58 : SAjrReg.LoadWndPos(this.Handle);
59 : }
60 : //----- 開始ボタン -----//
61 : private void cmdStart_Click(object sender, EventArgs e)
62 : {
63 :     tim.Enabled      = true;
64 :     cmdStart.Enabled = false;
65 :     cmdStop.Enabled  = true;
66 : }
67 : //----- 停止ボタン -----//
68 : private void cmdStop_Click(object sender, EventArgs e)
69 : {
70 :     tim.Enabled      = false;
71 :     cmdStart.Enabled = true;
72 :     cmdStop.Enabled  = false;
73 : }
74 : //----- タイマイベント -----//
75 : private void tim_Tick(object sender, EventArgs e)
76 : {
77 :     DateTime now = DateTime.Now;
78 :     string s = now.Hour.ToString("D2") + "¥n" + now.Minute.ToString("D2") + ":" + now.Second.ToString("D2");
79 :     d1 += RndNum(); d1 = Math.Max(d1, 1.0); d1 = Math.Min(d1, 100.0);
80 :     d2 += RndNum(); d2 = Math.Max(d2, 1.0); d2 = Math.Min(d2, 100.0);
81 :     d3 += RndNum(); d3 = Math.Max(d3, 1.0); d3 = Math.Min(d3, 100.0);
82 :     bar.PutData(s, d1, d2, d3);
83 :     lin.PutData(s, d1, d2, d3);
84 : }
85 : //----- ランダムな加数生成 -----//
86 : private double RndNum()
87 : {
88 :     double a;
89 :     a = rnd.NextDouble();
90 :     a *= rnd.Next(30);
91 :     if ((rnd.Next() & 1) != 0) a *= -1.0;
92 :     return a;
93 : }
94 : //----- フォーム終了 -----//
95 : private void Form1_FormClosed(object sender, FormClosedEventArgs e)
96 : {
97 :     SAjrReg.SaveWndPos(this.Handle);
98 : }
99 : }
100 : }

```

10. 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。

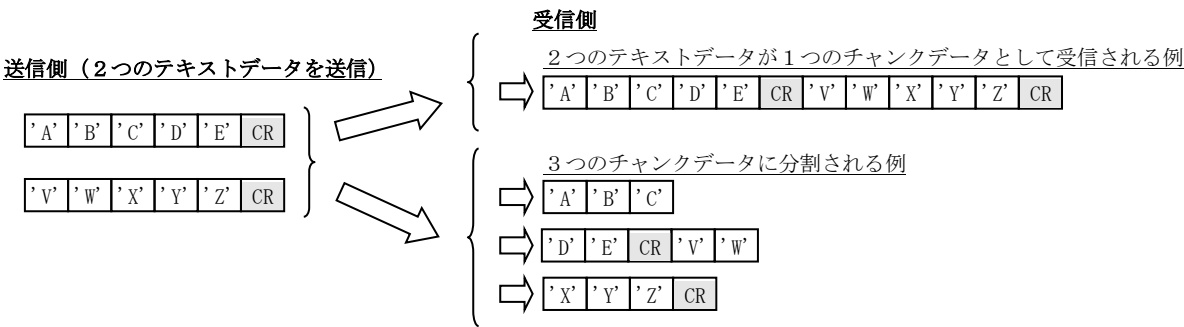
- ・シリアル通信 (COMポート, メールスロット (UDP/IP), ソケット (TCP/IP クライアント))
- ・ソケットサーバ (TCP/IP サーバ機能)
- ・ソケットクライアント (TCP/IP クライアント機能)

10.1. チャンクデータ

チャンクデータとは、1回の読み出し操作で読み出されたデータを意味します。
すぐ後の「データ区切りの認識」で示す、「受信側」の各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的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。

10.2. 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。



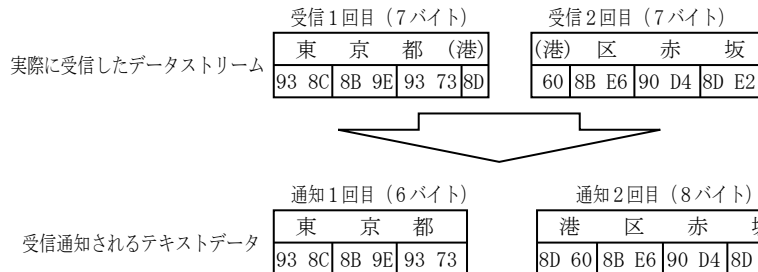
このような場合、受信側では、受信したデータを解析／編集し、2つのテキストデータに復元する必要がありますが、本ライブラリでは、このようなデータストリームの区切りを認識し、各テキストデータの単位で受信通知することができます。
上記の例では、“ABCDE” と “VWXYZ” の2つのテキストデータとして受信通知します。
また、テキストデータのほかにも、以下のデータの区切りを認識し、当該データブロックの単位で受信通知することができます。

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	パケット外テキスト	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

10.3. 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 バイトずつ、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。

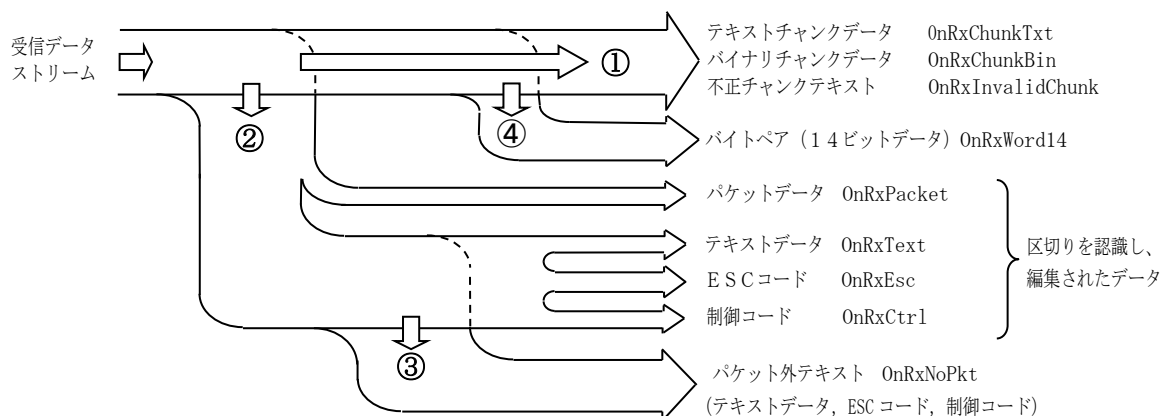
テキスト受信通知の場合は、1 つまたは複数のチャンクデータに渡り、制御コード (TAB (0x09) を除く 0x00～0x1F, 0x7F) の受信によりテキストストリームが完結するまで待つことから、マルチバイト文字の分断は発生しないためマルチバイト文字分断抑止は行いません。

10.4. 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	OnRxChunkTxt	テキストチャンクデータ受信通知	リアルタイム
2	OnRxChunkBin	バイナリチャンクデータ受信通知	リアルタイム
3	OnRxInvalidChunk	不正チャンクテキスト受信通知	リアルタイム
4	OnRxText	テキスト受信通知	
5	OnRxEsc	E S C シーケンス受信通知	
6	OnRxCtrl	制御コード受信通知	
7	OnRxPacket	パケットデータ受信通知	
8	OnRxNoPkt	パケット外データ受信通知	リアルタイム
9	OnRxWord14	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。

10.4.1. テキスト チャンク・データ受信通知 (OnRxChunkTxt)

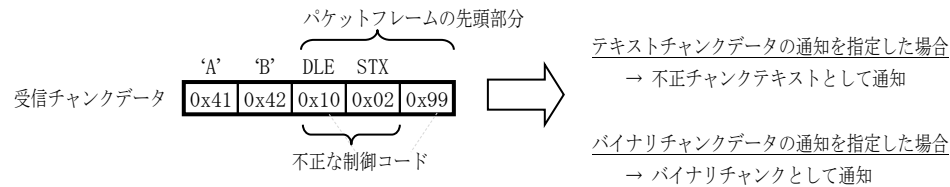
1 回の受信操作で読み出したテキストデータを、テキストエンコードの変換とマルチバイト文字の分断を抑止した上で通知します。
 チャンクデータ中に不正な制御コード (0x07~0x0D, 0x1B 以外) が含まれる場合は、チャンクテキストとしては通知しないで、不正チャンクテキスト (バイナリデータ) として通知します。(詳細は、不正チャンクテキスト受信通知を参照)

10.4.2. バイナリ チャンク・データ受信通知 (OnRxChunkBin)

チャンクデータの通知方法を「バイナリ」とした場合に通知されます。
 1 回の受信操作で読み出したデータを、そのままバイナリデータとして通知します。

10.4.3. 不正チャンクテキスト受信通知 (OnRxInvalidChunk)

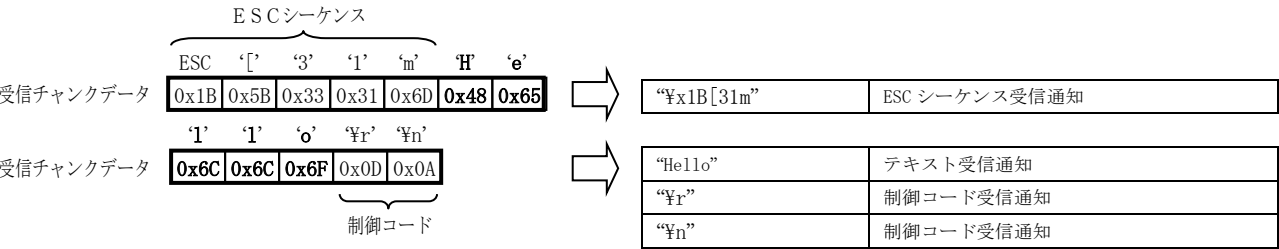
チャンクデータの通知方法を「テキスト」とした場合に通知されます。
 チャンクデータ中に不正な制御コード (0x09~0x0D, 0x1B 以外) が含まれる場合、チャンク テキスト・データ受信通知 (OnRxChunkTxt) ではなく、不正チャンクテキスト受信通知 (OnRxInvalidChunk) で受信したデータが通知されます。
 不正チャンクテキスト受信通知では、受信したチャンクデータ全体がバイナリデータとして通知されます。
 ※ パケットデータとテキストデータを多重化した (混在した) 送受信を行う場合、パケットデータが混入したチャンクデータは不正チャンクテキストとみなされてしまいます。



パケットデータの送受信を行う場合は、(不正チャンクテキストを防止する意味で) チャンクデータをバイナリとして扱うように指定してください。

10.4.4. テキスト受信通知 (OnRxText)

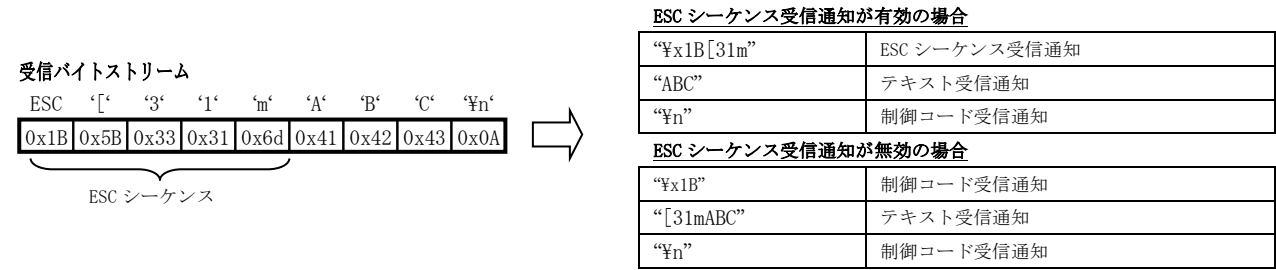
テキストデータを受信した場合に通知されます。
 テキストデータは、制御コード (TAB (0x09) を除く 0x00~0x1F, 0x7F)、パケットフレームや E S C シーケンスによりサンドイッチされた部分のデータです。(TAB (0x09) はテキストデータとして扱います)
 テキストデータは、複数のチャンクデータに渡って認識します。



10.4.5. E S Cシーケンス受信通知 (OnRxEsc)

E S Cシーケンスを受信した場合に通知されます。
E S C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。
E S C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。

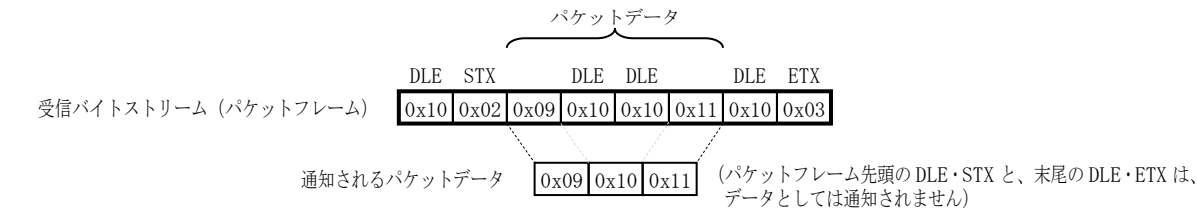
E S Cシーケンスの受信通知が無効である場合は、ESC(0x1B)は制御コードとして認識します。



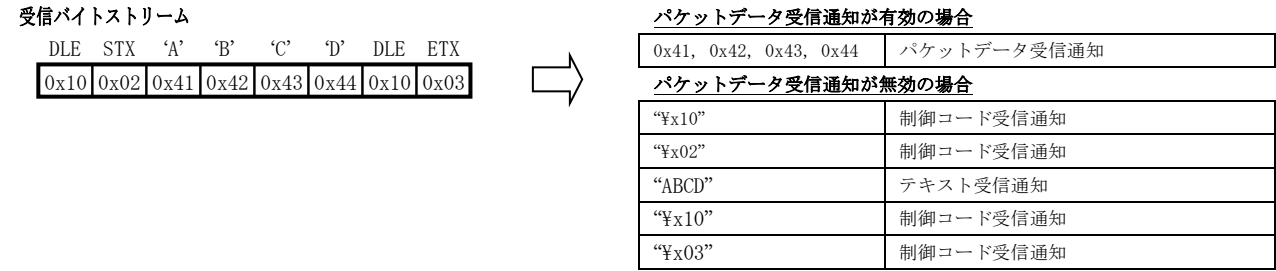
10.4.6. パケットデータ受信通知 (OnRxPkt)

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。



10.4.7. パケット外テキスト受信通知 (OnRxNoPkt)

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。

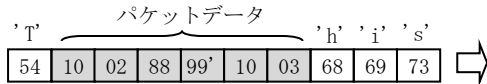
パケットデータの受信イベントが無効である場合は、受信チャンクデータ全体となります。

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。

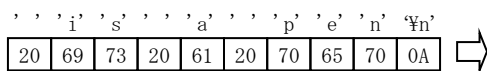
但し、テキストの末尾でマルチバイト文字が分断される場合は、マルチバイト文字の分断を抑止します。

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。

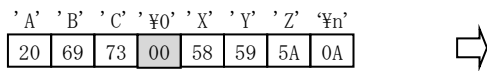
受信チャンクデータ



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



ヌル文字以降はパケット外テキストを受信通知しない

"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

10.4.8. バイトペアによるワード(14Bit)データ受信通知 (OnRxWord14)

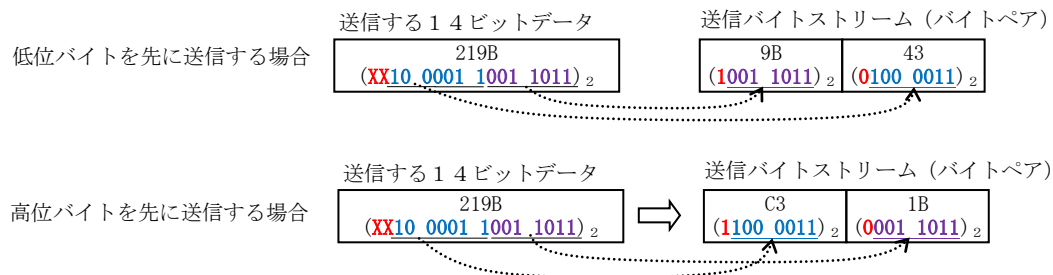
バイトペアデータを認識した場合に通知します。

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。

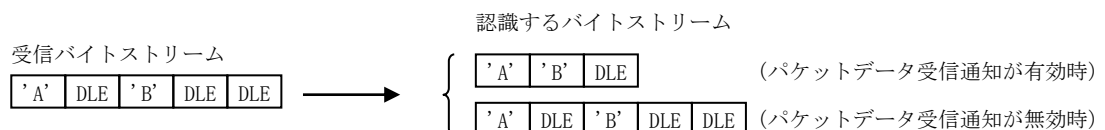


バイトペアの送受信は「シリアル通信」でのみ可能です。(ソケットサーバ、ソケットクライアントではバイトペアを認識しない)

10.4.9. 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つのDLEとして認識します。



10.5. 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。

10.6. 送受信テキストデータの文字コード

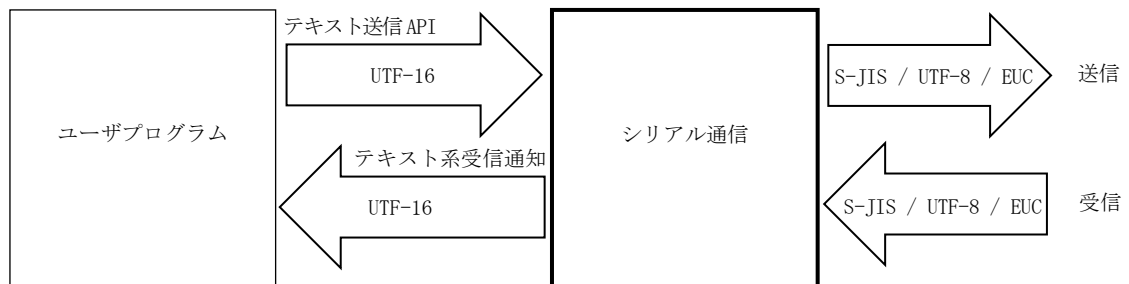
テキストデータは、全てマルチバイト・テキストで送受信します。

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）

Windows(.NET Framework)では、テキストの文字コードをUTF-16として扱います。

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 UTF-16(UNICODE)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 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 UTF-16 → SJIS に変換して送信（デフォルト）
- ・TxTextCode = UTF8 - 送信テキストを UTF-16 → UTF-8 に変換して送信
- ・TxTextCode = EUC - 送信テキストを 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 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。

11. シリアル通信コントロール (CAJrSerialComPort.dll)

COMポート、メールスロット (LAN) やソケット (TCP/IP クライアント) による通信制御モジュールです。
COMポート以外にも、メールスロット (UDP/IP) 通信やソケット (TCP/IP) 通信が可能です。

11.1. 機能概要

11.1.1. 受信データの通知形式と送受信文字コード

本書冒頭の「受信データの通知形式と送受信文字コード」章を参照してください。

11.1.2. イベントの通知方法

イベントの通知方法は、_ScpMode プロパティにより、以下の2つから選択できます。

- ・本コントロールがイベントを発生する (_ScpMode= EScpMode.NotificationByEvent, デフォルト)
- ・ユーザが、その場でイベントの発生を待ち受ける (_ScpMode= EScpMode.WaitingForEvent)

_ScpMode プロパティは、デザインモード時に指定するようにしてください。

コンソールアプリ等で、デザインモードで指定できない場合は、プログラムの最初に (Open() メソッド実行前に) 設定してください。

_ScpMode プロパティを **EScpMode. NotificationByEvent** に設定した場合は、本コントロールがイベント (OnXXXX) を発生し、事象をユーザに通知します。デフォルトでは、このモードに設定されています。

このモードは通常、Windows フォームアプリケーションで使います。

_ScpMode プロパティを **EScpMode. WaitingForEvent** に設定した場合は、WaitEvent() メソッドによりユーザがイベントの発生を待ち受けます。

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、タイムアウト時間を指定して、その場で応答データの着信を待つことが可能です。

このモードは通常、コンソールアプリケーションで使います。

11.1.3. COMポート通信

COMポートで通信するには、以下の Open メソッドでCOMポートをオープンします。

```
scp.Open(PortNo, rate, DataBits, Parity, StopBit);
```

「PortNo」は 1 ～ 255 で COM1 ～ COM255 を示します。その他のパラメタは Open() メソッドの説明を参照してください。

11.1.4. メールスロット (LAN) 通信

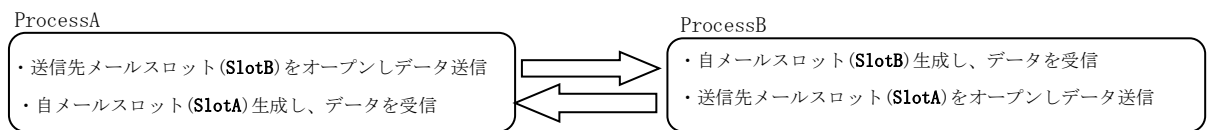
1 つのコンピュータで 2 つのプログラムが通信したり、LAN に接続されている 2 つのコンピュータ間で通信します。

(※ 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、MySlotName プロパティに自メールスロット名を指定し、自メールスロットの生成も行うようにします。

また、RemoteSlotName プロパティに送信先スロット名を指定します。

11.1.5. 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 (TCP/IP クライアント) 通信するには、以下の Open メソッドでソケットをオープンします。

```
scp.Open(ServName, PortNo);
```

「ServName」は、TCP/IP サーバのコンピュータ名 (あるいは IP アドレス) を指定します。

「PortNo」は、TCP/IP サーバのポート番号 (1 ～ 65535) を指定します。

但し、ソケットをオープンしても、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、以下のいずれかの方法で検知することができます。

- PortState (e. state = EScpPortState.Opened) イベントの発生 (サーバへの接続) を待つ
- 適当な周期で IsOpened プロパティを監視し、IsOpened=true となるのを待つ

11.1.6. 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。

接続できるクライアント数は「1」固定です。

ソケット (TCP/IP サーバ) 通信するには、以下の Open メソッドでソケットをオープンします。

```
scp.Open(PortNo);
```

「PortNo」は、TCP/IP サーバのポート番号 (1 ～ 65535) を指定します。

但し、ソケットをオープンしても、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、以下のいずれかの方法で検知することができます。

- OnPortState (e. state = EScpPortState.Opened) イベントの発生 (クライアントの接続) を待つ
- 適当な周期で IsOpened プロパティを監視し、IsOpened=true となるのを待つ

11.1.7. 各通信リソースを切り替えて通信

COMポート通信、メールスロットによる通信、ソケット通信は、随時切り替えることができます。

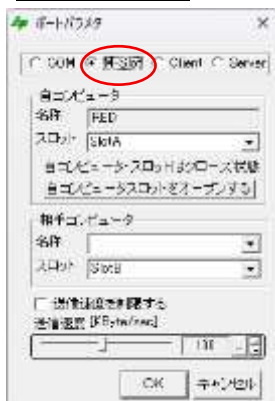
SetParamByDialog() メソッドにより、ダイアログによるポートの設定を行い、(引数無しの) Open() メソッドで、ダイアログで設定されたCOMポート/メールスロット/ソケットをオープンすることができます。

SetParamByDialog() メソッドは、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



通信リソースを選択して通信するには、引数無しの Open メソッドで通信リソースををオープンします。

```
scp.Open();
```

実際にオープンされるリソースは、ダイアログで設定された通信リソースとなります。

sel 引数付きの Open() メソッドは、選択する通信リソースを指定してオープンします。

```
scp.Open(EPortSel, SEL_COMPORT);
```

--- ダイアログ設定内容でCOMポートをオープン

11.1.8. D C Bとタイムアウト情報

本コントロールでは、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、この際にD C Bとタイムアウト情報でCOMポート通信用のパラメタを設定します。

これらの情報は、SetDetailParamByDialog() メソッドにより設定できます。

SetDetailParamByDialog() メソッドは、以下のダイアログを表示します。



「デフォルト設定」ボタンを押すと、設定内容が規定値（上記ダイアログで示している内容）にリセットされます。

DCBの内容 (通常、網掛け部分以外は変更不要)

#	メンバ	内容	規定値
1	BaudRate	通信速度[bps]	9600
2	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
3	fParity	パリティビットの有無 (0:なし, 1:あり)	0
4	fOutxCtsFlow	CTS信号による送信フロー制御 (0:無効, 1:有効)	0
5	fOutxDsrFlow	DSR信号による送信フロー制御 (0:無効, 1:有効)	0
6	fDtrControl	DTR信号による受信フロー制御 (0:無効, 1:有効)	0
7	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
8	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
9	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
10	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
11	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
12	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
13	fRtsControl	RTS信号による受信フロー制御 (0:無効, 1:有効)	0
14	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
15	XonLim	受信フロー制御における低位バッファ容量	512
16	XoffLim	受信フロー制御における高位バッファ容量	1536
17	ByteSize	データビット数 (4~8bit)	8
18	Parity	パリティビット・タイプ (0:None, 1:Odd, 2:Even, 3:Mark, 4:Space)	0
19	StopBits	ストップビット数 (0:1bit, 1:1.5bit, 2:2bit)	1
20	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
21	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
22	ErrorChar	受信エラー時に置換するバイトデータ	0x3F
23	EofChar	ファイル終端文字コード (fBinary=1 の場合は無効)	0x1A
24	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に関するプロパティ (通常、変更は不要です)

#	メンバ	内容	規定値
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

11.2. 構造体／列挙体（定数）

11.2.1. 実行モード

```
public enum EScpMode : int
{
    NotificationByEvent = 0,    // イベントを通知する
    WaitingForEvent     = 1    // イベントを待ち受ける
}
```

11.2.2. 通信リソース選択

```
public enum ESIPort : int
{
    SEL_COMPORT    = 0,    // COMポート
    SEL_MAILSLLOT  = 1,    // メールスロット
    SEL_SOCKET     = 2,    // ソケット (TCP/IP クライアント)
    SEL_SERVER     = 3,    // ソケット (TCP/IP サーバ)
}
```

11.2.3. ポート状態

```
public enum EScpPortState : int
{
    Closed        = 0,    // クローズ状態
    Opened        = 1,    // オープン状態
    OpenFailure   = 2,    // オープン失敗
    PortNumber    = 3,    // ポート番号が変化
    MySlotFail    = 4,    // 自メールスロット生成失敗
    TxFailure     = 5,    // 送信失敗
    ServStart     = 6,    // サーバ開始
    ServStop      = 7,    // サーバ停止
    SrvSrtFail    = 8,    // サーバ開始失敗
}
```

11.2.4. チャンクデータの扱い

```
public enum EScpChunkMode : int
{
    BinaryData    = 0x01,    // バイナリ・チャンク
    TextData      = 0x02,    // テキスト・チャンク
    Both          = 0x03,    // 両方
}
```

11.2.5. データビット数

```
public enum EScpDataBits : int
{
    DataBit_Default = 0,    // 現在の設定値
    DataBit_7       = 7,    // 7ビット長
    DataBit_8       = 8,    // 8ビット長
}
```

11.2.6. パリティビット

```
public enum EScpParity : int
{
    DefaultParity = 0,    // 現在の設定値
    NoParity      = 'N',    // パリティなし
    OddParity     = 'O',    // 奇数パリティ
    EvenParity    = 'E',    // 偶数パリティ
}
```

11.2.7. ストップビット

```
public enum EScpStopBit : int
{
    StopBit_Default = 0,    // 現在の設定値
    StopBit_1       = 1,    // 1ビット
    StopBit_2       = 2,    // 2ビット
}
```

11.2.8. イベントコード

```

public enum EScpEvt : int
{
    EV_NOEVENT      = 0,           // イベント待ちタイムアウト
    EV_PORTSTATE    = 0x8000,      // ポート状態通知
    EV_RXCHUNK      = 0x4000,      // チャンクデータ受信通知
    EV_RXTTEXT      = 0x2000,      // テキスト受信通知
    EV_RXESC        = 0x1000,      // E S C コード受信通知
    EV_RXCTRL       = 0x0800,      // 制御コード受信通知
    EV_RXPKT        = 0x0400,      // パケットデータ受信通知
    EV_TXEMPTY      = 0x0200,      // 送信完了
    EV_RXNOPKT      = 0x0100,      // パケット外テキスト通知

    EV_ERR          = 0x0080,      // エラー通知
    EV_BREAK        = 0x0040,      // BREAK 検出通知
    EV_RING         = 0x0020,      // RING 変化通知
    EV_RLSD         = 0x0010,      // RLSD 変化通知
    EV_DSR          = 0x0008,      // DSR 変化通知
    EV_CTS          = 0x0004,      // CTS 変化通知

    EV_RXWORD14     = 0x0002,      // バイトペアによるワード(14Bit 値)受信通知
    EV_INVCHUNK     = 0x0001,      // 不正チャンクテキスト受信通知

    EV_SSEP         = (EV_RXTTEXT | EV_RXESC | EV_RXCTRL | EV_RXPKT)
    EV_DEFAULT_EVT  = (EV_SSEP | EV_DSR | EV_CTS)
    EV_DEFAULT_POST = (EV_PORTSTATE | EV_DEFAULT_EVT)

    EV_ALL          = ( EV_PORTSTATE |
                        EV_RXCHUNK |
                        EV_RXTTEXT |
                        EV_RXESC |
                        EV_RXCTRL |
                        EV_RXPKT |
                        EV_TXEMPTY |
                        EV_RXNOPKT |
                        EV_ERR |
                        EV_BREAK |
                        EV_RING |
                        EV_RLSD |
                        EV_DSR |
                        EV_CTS |
                        EV_RXWORD14 |
                        EV_INVCHUNK
                      )
}

```

11.2.9. エラーコード

```

public enum EScpErr : int
{
    CE_RXOVER       = 0x0001,      // 受信キューオーバフロー
    CE_OVERRUN      = 0x0002,      // オーバーランエラーを検出した
    CE_RXPARITY     = 0x0004,      // パリティエラーを検出した
    CE_FRAME        = 0x0008,      // フレーミングエラーを検出した
    CE_BREAK        = 0x0010,      // ブレーク信号を検出した
    CE_TXFULL       = 0x0100,      // 送信キュー満杯
    CE_IOE          = 0x0400,      // デバイスアクセス中に I/O エラーを検出した

    RXERR           = 0x10000,      // 受信エラー
    TXERR           = 0x20000,      // 送信エラー
}

```

11.2.10. 受信テキストの文字コード

```
public enum ESepRxTextCode : int
{
    SJIS          = 1    ,    // S - J I S
    EUC           = 2    ,    // E U C
    UTF8          = 3    ,    // U T F - 8
    AUTO          = 9    ,    // 自動認識
}
```

11.2.11. 送信テキストの文字コード

```
public enum ESepTxTextCode : int
{
    SJIS          = 1    ,    // S - J I S
    EUC           = 2    ,    // E U C
    UTF8          = 3    ,    // U T F - 8
}
```

11.2.12. バイトペア受信順序

```
public enum ESepByteSeq : int
{
    HIGH_BYTE_FIRST = 0,    // High byte first
    LOW_BYTE_FIRST  = 1 ,    // Low byte first
}
```


11.3. プロパティ

シリアル通信コントロールのプロパティ一覧を示します。

#	名称	タイプ	内容	デフォルト
1	_ProfileSection	string	設定情報を記録するプロファイルセクション名 (※1)	“SerialPortSect”
2	_ScpMode	EScpMode	実行モード (※2) NotificationByEvent : イベントの発生を OnXXXX 通知で受ける WaitingForEvent : イベントの発生を WaitEvent() で待ち受ける	デザイン時のみ有効 NotificationByEvent
3	ChunkMode	EScpChunkMode	チャンクデータの通知モード BinaryData : バイナリデータ TextData : テキストデータ Both : 両方	BinaryData
4	STX	int	パケットデータの先頭識別コード	0x02
5	ETX	int	パケットデータの終端識別コード	0x03
6	DLE	int	パケットデータの透過制御コード	0x10
7	IsCreatedMySlot	bool	自メールスロットの生成状態 (false:未生成, true:生成済)	読出し専用 -
8	IsOpened	bool	ポートのオープン状態 (false:未オープン, true:オープン済) ※ソケットの場合は, true で接続済を意味する	
9	IsServerStarted	bool	サーバ起動状態取得 (false:起動済, true:停止中)	
10	PortKind	ESelPort	現在設定されている、通信ポート種別 ・SEL_COMPOR : COMポート ・SEL_MAILSLLOT : メールスロット ・SEL_CLIENT : ソケット (TCP/IP クライアント) ・SEL_SERVER : ソケット (TCP/IP サーバ)	
11	PortNo	int	現在設定されている、通信ポート番号 ・1~255 : COM1 ~ COM255 ・256 : メールスロット ・257 : ソケット (TCP/IP クライアント) ・258 : ソケット (TCP/IP サーバ)	
12	ActualRxTextCode	EScpRxTextCode	実際の受信テキストコード(SJIS / EUC / UTF8) AUTO 設定時は、実際に受信したテキストから自動判別	SJIS
13	ActualTxTextCode	EScpTxTextCode	実際の送信テキストコード(SJIS / EUC / UTF8) AUTO 設定時は、受信テキストコードと同じ	SJIS
14	RxTextCode	EScpRxTextCode	受信テキストコード(SJIS / EUC / UTF8 / AUTO(自動判別)) (※3)	
15	TxTextCode	EScpTxTextCode	送信テキストコード(SJIS / EUC / UTF8 / AUTO(受信テキストコードと同じ))	SJIS
16	PktTimeout	int	パケットデータ受信タイムアウト時間[ms]	3000[ms]
17	EvtMask	EScpEvt	未使用 (SetEvtMask() メソッドを使用してください)	-
18	MySlotName	string	自スロット名 (※4) (※5) (※6)	“MySlot”
19	RemoteComputerName	string	相手コンピュータ名 (自コンピュータ内で通信する場合は空白) (※6)	“”
20	RemoteSlotName	string	相手スロット名 (※5) (※6)	“RmtSlot”
21	ByteSeq	EScpByteSeq	HIGH_BYTE_FIRST : 上位バイト, 下位バイトの順で受信する (Big Endian) LOW_BYTE_FIRST : 下位バイト, 上位バイトの順で受信する (Little Endian)	LOW_BYTE_FIRST

※1 : 複数のシリアル通信インスタンスを使用する場合は、各々のインスタンスで重複しない名称を設定してください。

空文字列とした場合は、プロファイルへ通信パラメータを記録しません。

※2 : 通常、コンソールアプリの場合、WaitingForEvent を設定し、WaitEvent() メソッドを使用します。

※3 : 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。

※4 : 自スロット名を空文字列とした場合は、起動時に自メールスロットを生成しません。

自コンピュータ内で重複したスロット名称を使用することはできません。

※5 : 自コンピュータ内でプロセス間通信を行う場合 (相手コンピュータ名が空文字列の場合) は、自スロット名と相手スロット名を同じ名称にすることはできません。

※6 : これらの値は、SetParamByDialog() メソッドによる ポート設定ダイアログのデフォルト値です。

引数なしの Open() メソッドの場合、実際の値は、ポート設定ダイアログで設定できます。

11.4. メソッド

シリアル通信コントロールメソッド一覧を以下に示します。

#	メソッド名	内容	備考
1	Init	初期設定（最初に1度だけ、必ず実行する必要があります）	
2	Open	通信回線オープン	
3	Close	通信回線クローズ	
4	SendByte	1 バイト送信	
5	SendChar	1 文字送信	
6	SendText	テキスト送信	
7	SendBinary	バイナリ送信	
8	SendPacket	パケット送信	
9	SendWord14LF	1 4 ビットワード値（バイトペア）送信, Low Byte First	
10	SendWord14HF	1 4 ビットワード値（バイトペア）送信, High Byte First	
11	SendBreak	ブレイク信号送出／停止	
12	SetDTR	D T R 信号設定	
13	SetRTS	R T S 信号設定	
14	GetDSR	D S R 信号取得	
15	GetCTS	C T S 信号取得	
16	GetRLSD	R L S D 信号取得	
17	GetRING	R I N G 信号取得	
18	PurgeRx	受信済データデータ破棄	
19	PurgeTx	送信待ちデータデータ破棄	
20	Purge	送受信データ破棄	
21	SetPortKind	ポート種別の設定	
22	SetParamByDialog	ダイアログによる通信パラメタ設定	
23	SetDetailParamByDialog	ダイアログによる詳細な通信パラメタ設定	
24	EnablePortSelectionInDialog	通信パラメタ設定ダイアログによるCOMポート、メールスロット、ソケット通信の選択許可／禁止	
25	WaitEvent	イベント待ち	
26	SetEvtMask	イベントマスク設定	
27	GetEvtMask	イベントマスク取得	
28	GetPortName GetPortPathName	ポート名称取得	
29	GetPortDevName	COMポートのデバイス名称取得	
30	GetMySlotPathName	自メールスロットパス名取得	
31	CreateMySlot	自メールスロット生成	
32	DeleteMySlot	自メールスロット消去	
33	SetMailSlotNames	メールスロット名情報設定	
34	Delete	シリアル通信インスタンスの消去	

11.4.1. 初期設定 (Init)

形 式 : void Init();

引 数 : なし

説 明 : シリアル通信の初期設定を行います。
このメソッドは、他のメソッドの先駆けて、最初に1度だけ実行する必要があります。

戻り値 : なし

11.4.2. 通信回線オープン (Open)

形 式 : `bool Open();` ----- ダイアログで設定された内容でオープン
`bool Open (EPortSel sel);` - ダイアログで設定された内容でオープン (初回オープンする通信リソース選択)
`bool Open(int PortNo, int rate, ESepDataBits DataBits, ESepParity Parity, ESepStopBit StopBit);` - COMポート
`bool Open(string RemoteHostName, string RemoteSlotName);` --- メールスロットのオープン
`bool Open(string ServName, uint ServPortNo);` ----- ソケット(TCP/IP クライアント)のオープン
`bool Open(ServPortNo);` ----- ソケット(TCP/IP サーバ)のオープン

引 数 : `sel` - 最初にオープンする通信リソース (SEL_COMPORT / SEL_MAILSLLOT / SEL_SOCKET / SEL_SERVER)
`PortNo` - COMポート番号 (1 ~ 255)
`rate` - 通信レート[bps]
`DataBits` - データビット数 (DataBit_Default, DataBit_7, DataBit_8)
`Parity` - パリティビット (DefaultParity, NoParity, OddParity, EvenParity)
`StopBit` - ストップビット長 (StopBit_Default, StopBit_1, StopBit_2)
`RemoteHostName` - 通信相手のコンピュータ名 (自コンピュータ内で通信する場合は、空文字列)
`RemoteSlotName` - 通信相手のスロット名
`ServName` - サーバのコンピュータ名/IP アドレス
`ServPortNo` - サーバのポート番号

説 明 : COMポート/メールスロット/ソケットをオープンし、通信可能状態にします。
既にオープンされている場合は、一旦クローズし、再度オープンします。
引数なしの `Open()` メソッドは、現在設定されている (あるいは、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します。
`sel` 引数を指定した `Open()` メソッドは、オープンする通信リソースを指定し、その後は引数無しの `Open()` メソッドと同じです。
メールスロットのオープンにおいて、自メールスロットが未生成である場合は、自メールスロットの生成も行います。

戻り値 : `true` - オープン成功
`false` - オープン失敗

備 考 : メールスロットをオープンする場合、自メールスロットが未生成ならば、自メールスロットの生成を試みます。

11.4.3. 通信回線クローズ (Close)

形 式 : `void Close();`

引 数 : なし

説 明 : 現在選択されている通信リソースをクローズし、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、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、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 : なし

11.4.4. バイト文字送信 (SendByte)

形 式 : `void SendByte(int ByteCode);`

引 数 : `ByteCode` - 送信するバイトコード (0x00～0xFF)

説 明 : 通信回線へバイト文字（日本語の場合はシフト J I S）を送信します。 マルチバイトの場合は複数回実行してください。
文字データを `TxTextCode` プロパティで指定された文字コードに変換して送信します。
このメソッドは送信用スプールバッファにデータを格納するだけであり、送信完了を待たずに、すぐに制御を返します。

戻り値 : なし

11.4.5. 1文字送信 (SendChar)

形 式 : `void SendChar(char Character);`

引 数 : `Character` - 送信する文字 (0x0000～0xFFFF)

説 明 : 通信回線へ1文字送信します。
文字データを `TxTextCode` プロパティで指定された文字コードに変換して送信します。
このメソッドは送信用スプールバッファにデータを格納するだけであり、送信完了を待たずに、すぐに制御を返します。

戻り値 : なし

11.4.6. テキスト送信 (SendText)

形 式 : `void SendText(string text);`

引 数 : `text` - 送信するテキストデータ

説 明 : 通信回線へテキストデータを送信します。
テキストデータを `TxTextCode` プロパティで指定された文字コードに変換して送信します。
このメソッドは送信用スプールバッファにデータを格納するだけであり、送信完了を待たずに、すぐに制御を返します。

戻り値 : なし

11.4.7. バイナリ送信 (SendBinary)

形 式 : `void SendBinary(Byte[] bin);`
`void SendBinary(IntPtr p, int len);`
`unsafe void SendBinary(void *p, int len);`

引 数 : `bin` - 送信するバイナリデータのバイト配列
`p` - 送信するバイナリデータのアドレス
`len` - 送信するバイナリデータのバイト数

説 明 : 通信回線へバイナリデータを送信します。
このメソッドは送信用スプールバッファにデータを格納するだけであり、送信完了を待たずに、すぐに制御を返します。

戻り値 : なし

11.4.8. パケット送信 (SendPacket)

形 式 : int SendPacket(Byte[] bin);
 int SendPacket(IntPtr p, int len);
 unsafe int SendPacket(void *p, int len);

引 数 : bin - 送信するバイナリデータのバイト配列 (空パケット (DLE, STX, DLE, ETX の 4 バイト) 送信時は null)
 p - 送信するバイナリデータのアドレス (空パケット (DLE, STX, DLE, ETX の 4 バイト) 送信時は 0)
 len - 送信するバイナリデータのバイト数 (空パケット (DLE, STX, DLE, ETX の 4 バイト) 送信時は 0)

説 明 : 通信回線へパケット形式でバイナリデータを送信します。
 つまり、先頭に「DLE・STX」の 2 バイトを、末尾に「DLE・ETX」の 2 バイトを付加し、バイナリデータ中の DLE バイトは、2 つの DLE に変換 (DLE → DLE・DLE) して伝送します。
 STX, ETX, DLE の実際のコード値はプロパティにて設定可能です。(規定値は、STX=0x02, ETX=0x03, DLE=0x10)
 このメソッドは送信用スプールバッファにデータを格納するだけであり、送信完了を待たずに、すぐに制御を返します。

戻り値 : 4 ~ - 正常 ((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 - エラー

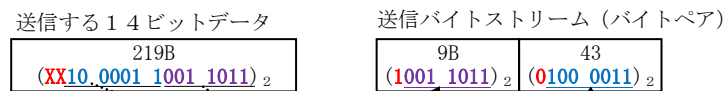
11.4.9. 4 ビットワード値 (バイトペア) 送信, Low Byte First (SendWord14LF)

形 式 : void SendWord14LF (int data);

引 数 : data - 送信するワード (14Bit 値) 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 ビットず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。



このメソッドは送信用スプールバッファにデータを格納するだけであり、送信完了を待たずに、すぐに制御を返します。

戻り値 : なし

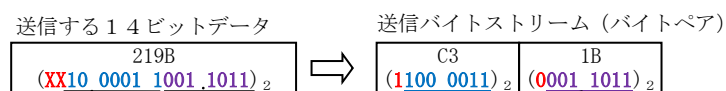
11.4.10. 14 ビットワード値 (バイトペア) 送信, High Byte First (SendWord14HF)

形 式 : void SendWord14HF (int data);

引 数 : data - 送信するワード (14Bit 値) 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 ビットず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。



このメソッドは送信用スプールバッファにデータを格納するだけであり、送信完了を待たずに、すぐに制御を返します。

戻り値 : なし

11.4.11. ブレーク信号送出／停止 (SendBreak)

形 式 : void SendBreak(bool fActive);

引 数 : fActive - true : ブレーク信号送出, false : ブレーク信号送出停止

説 明 : 通信回線へブレーク信号を創出します。
SendBreak(true);でブレーク信号の送出を開始し、SendBreak(false);でブレーク信号の送出を停止します。

戻り値 : なし

11.4.12. D T R信号設定 (SetDTR)

形 式 : void SetDTR(bool fActive);

引 数 : fActive - DTR 信号設定 (true - アクティブ状態, false - 非アクティブ状態)

説 明 : D T R信号の設定を行います。

戻り値 : なし

11.4.13. R T S信号設定 (SetRTS)

形 式 : void SetRTS(bool fActive);

引 数 : fActive - RTS 信号設定 (true - アクティブ状態, false - 非アクティブ状態)

説 明 : R T S信号の設定を行います。

戻り値 : なし

11.4.14. D S R信号状態取得 (GetDSR)

形 式 : bool GetDSR();

引 数 : なし

説 明 : D S R信号の状態を取得します。

戻り値 : true - DSR 信号アクティブ状態
false - DSR 信号非アクティブ状態

11.4.15. C T S信号状態取得 (GetCTS)

形 式 : bool GetCTS();

引 数 : なし

説 明 : C T S信号の状態を取得します。

戻り値 : true - CTS 信号アクティブ状態
false - CTS 信号非アクティブ状態

11.4.16. R L S D信号状態取得 (GetRLSD)

形 式 : bool GetRLSD();

引 数 : なし

説 明 : R L S D信号の状態を取得します。

戻り値 : true - RLSD 信号アクティブ状態
false - RLSD 信号非アクティブ状態

11.4.17. R I N G信号状態取得 (GetRING)

形 式 : `bool GetRING();`

引 数 : なし

説 明 : R I N G信号の状態を取得します。

戻り値 : `true` - RING 信号アクティブ状態
`false` - RING 信号非アクティブ状態

11.4.18. 受信済データデータ破棄 (PurgeRx)

形 式 : `void PurgeRx();`

引 数 : なし

説 明 : 受信済データをすべて破棄します。

戻り値 : なし

11.4.19. 送信待ちデータデータ破棄(PurgeTx)

形 式 : `void PurgeTx();`

引 数 : なし

説 明 : 送信待ちデータをすべて破棄します。

戻り値 : なし

11.4.20. 送受信データ破棄(Purge)

形 式 : `void Purge();`

引 数 : なし

説 明 : 受信済データと送信待ちデータをすべて破棄します。

戻り値 : なし

11.4.21. ポート種別の設定

形 式 : `void SetPortSel(ESelPort kind);`

引 数 : `kind` - ポート種別

• <code>SEL_COMPORT</code> : COMポート	• <code>SEL_CLIENT</code> : ソケット (クライアント)
• <code>SEL_MAILSLLOT</code> : メールスロット	• <code>SEL_SERVER</code> : ソケット (サーバ)

説 明 : 現在のポート種別選択状態を設定します。
`SetParamByDialog()` で通信パラメタ設定ダイアログを表示する場合、設定されたポート種別が既定値となります。

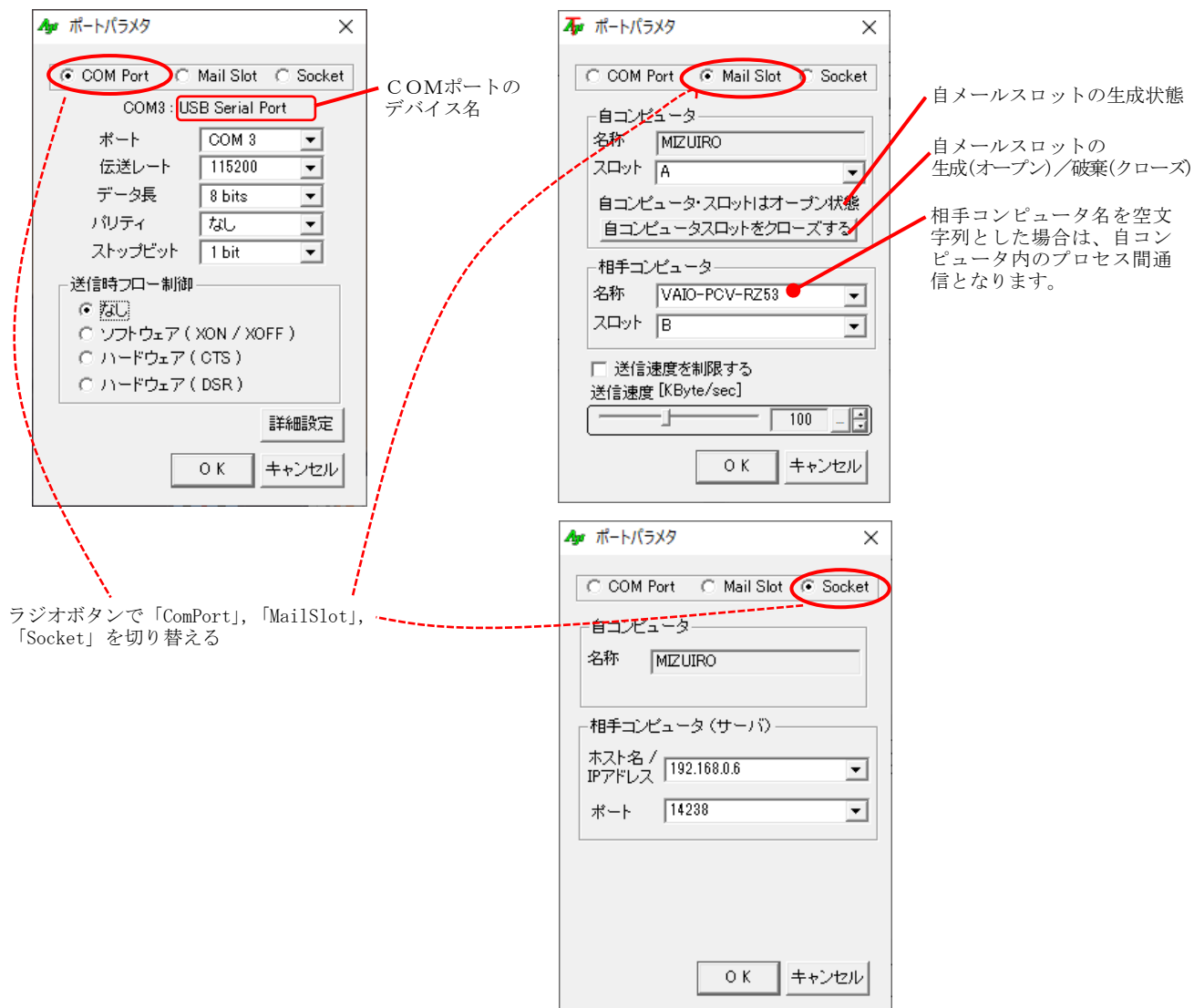
戻り値 : なし

11.4.22. ダイアログによる通信パラメタ設定(SetParamByDialog)

形 式 : void SetParamByDialog();
void SetParamByDialog(IntPtr hOwner);

引 数 : hOwner - オーナーウインドハンドル

説 明 : 以下のダイアログにより、通信パラメタの設定を行います。
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。



「MailSlot」選択時、相手コンピュータ名を空白とした場合は、自コンピュータ内のプロセス間通信となります。
尚、オープン状態で、COM ポート番号の変更や、「ComPort」「MailSlot」「Socket」の選択を変更した場合は、新たな設定内容で通信ポートの再オープンを行います。

戻り値 : なし

11.4.23. ダイアログによる詳細な通信パラメタ設定(SetDetailParamByDialog)

形 式 : void SetDetailParamByDialog;
void SetDetailParamByDialog(IntPtr hOwner);

引 数 : hOwner - オーナーウィンドハンドル

説 明 : 以下のダイアログにより、より詳細なCOMポートパラメタの設定を行います。



戻り値 : なし

11.4.24. 通信パラメタ設定ダイアログによるラジオボタンの選択許可/禁止(EnablePortSelectionInDialog)

形 式 : void EnablePortSelectionInDialog(bool fEnableComPort, bool fEnableMailSlot, bool fEnableClient, bool fEnableServer);

引 数 : fEnableComPort - true: COMポートの選択を許可, false: COMポートの選択を禁止
fEnableMailSlot - true: メールスロット選択を許可, false: メールスロット選択を禁止
fEnableClient - true: クライアント通信の選択を許可, false: クライアント通信の選択を禁止
fEnableServer - true: サーバ通信の選択を許可, false: サーバ通信の選択を禁止

説 明 : SetParamByDialog() によるダイアログで、各ラジオボタン「COM Port」「Mail Slot」「Socket」有効化/無効化します。
「true」を指定したラジオボタンは選択可能となります。
「false」を指定したラジオボタンは、グレー表示し選択不能となります。

11.4.25. イベント待ち(WaitEvent)

形 式 : EScpEvt WaitEvent(out Object EvtData, int msWaitTime);

引 数 : EvtData - 受信データ等を格納するオブジェクト (実際に格納されるタイプはイベントにより異なります)
WaitTime - イベントを待ち受ける時間 (タイムアウト値, ミリ秒)

説 明 : 通信回線からのイベントの発生を待ち受けます。
このメソッドは、通常、コンソールアプリケーションで使います。
このメソッドは、イベント待ち受けモードを設定 (ScpMode プロパティに「EScpMode.WaitingForEvent」を設定) した場合のみ実行可能です。

このメソッドでイベントを待ち受ける場合は、イベントハンドラの登録は意味を成しません。
つまり、「scp.OnRxText += new ScpEvtRxText(scp.OnRxText);」のようにイベントハンドラを登録しても当該ハンドラは呼び出されません。

このメソッドを使用する場合は、対象とするイベントを、SetEvtMask() メソッドによりあらかじめ設定しておかなければなりません。

SetEvtMask() メソッドにより指定するイベントマスクと、イベント発生時に EvtData に格納されるオブジェクトについては SetEvtMask() メソッドを参照してください。

例えば、テキストデータの受信を 1 秒間待ち受ける場合は、以下のようにコードします。

```
//----- 対象とするイベントを設定 -----//
scp.SetEvtMask(EScmEvt.EV_RXTXT);

. . . . .

//----- シリアル通信イベント/タイムアウト待ち -----//
evt = scm.WaitEvent(out obj, 1000);
//----- テキスト受信イベント -----//
if ((evt & EScmEvt.EV_RXTXT) != 0) {
    Console.WriteLine("RXTXT: " + (string)obj);
}
else if (evt == EScmEvt.EVT_NOEVENT) {
    // タイムアウト
}
```

戻り値 : ≠EV_NOEVENT : イベント・マスク値
=EV_NOEVENT : タイムアウト

備 考 : チャンクデータ受信通知(EV_RXCHUNK)で、テキストチャンクと、バイナリチャンクデータの両方を扱う (ChunkMode=Both) 場合は、以下の条件でテキストチャンク/バイナリチャンクを特定してください。

- EvtData.GetType() == typeof(byte[]) -- バイナリチャンクデータ
- EvtData.GetType() == typeof(string) -- テキストチャンクデータ

11.4.26. イベントマスク設定(SetEvtMask)

形 式 : void SetEvtMask(EScpEvt evt);

引 数 : evt : イベントマスク値

説 明 : WaitEvent()メソッドで検出するイベントを設定します。(_ScpMode プロパティ= WaitingForEvent に設定時のみ有効)
イベントマスクと、WaitEvent()で受け取るイベントデータ(EvtData)の内容は以下のとおりです。

#	イベントマスク	イベント	EvtData に格納されるオブジェクト	
			タイプ	内容
1	EV_NOEVENT	イベント待ちタイムアウト	-	null
2	EV_PORTSTATE	ポート状態通知	int	0:クローズ状態 2:オープン失敗 1:オープン状態 3:ポート番号変化 (ソケット時は、1:接続済)
3	EV_RXCHUNK	チャンクデータ受信通知 (テキスト)	string	受信したテキスト・チャンクデータ
		チャンクデータ受信通知 (バイナリ)	Byte[]	受信したバイナリ・チャンクデータ
4	EV_RXTEXT	テキスト受信通知	string	受信したテキストデータ
5	EV_RXESC	E S Cコード受信通知	string	受信したE S C文字列
6	EV_RXCTRL	制御コード受信通知	int	受信した制御コード
7	EV_RXPKT	パケットデータ受信通知	byte[]	受信したパケットデータ
8	EV_RXNOPKT	パケット外データ受信通知	int	受信した文字コード
9	EV_RXWORD14	バイトペアによるワード値受信通知	ushort	受信したワード値 (14ビット値)
10	EV_RXINVCHUNK	不正チャンクテキスト受信	byte[]	受信した不正テキストチャンク
11	EV_TXEMPTY	送信完了	int	0 (未使用)
12	EV_ERR	エラー通知	int	エラー要因マスク値 0x0001: システムの受信キュー満杯 0x0002: オーバーランエラー 0x0004: パリティチェックエラー 0x0008: フレーミングエラー 0x0010: ブレーク検出 0x0100: システムの送信キュー満杯
13	EV_BREAK	BREAK 検出通知	int	0 (未使用)
14	EV_RING	RING 変化通知	int	信号の状態マスク値(0:非Active, 1:Active) 0x10 : CTS 0x40 : RING 0x20 : DSR 0x80 : RLSD
15	EV_RLSD	RLSD 変化通知		
16	EV_DSR	DSR 変化通知		
17	EV_CTS	CTS 変化通知		

イベントマスクは、合成値で指定します。例えば、テキスト受信通知とパケットデータ受信通知イベントを検出する場合は、「EScpEvt.EV_RXTEXT | EScpEvt.EV_RXPKT」と指定します。

戻り値 : なし

11.4.27. イベントマスク取得(GetEvtMask)

形 式 : EScpEvt GetEvtMask();

引 数 : なし

説 明 : 現在設定されているイベントマスク値を取得します。

戻り値 : 現在設定されているイベントマスク値

11.4.28. ポート名取得(GetPortPathName)

形 式 : string GetPortName();
string GetPortPathName();

引 数 : なし

説 明 : 現在設定されている通信ポートの名称を取得します。

設定されている通信 ポート	戻り値	
	GetPortName()	GetPortPathName()
COM1～COM9	“COM1” ～ “COM9”	“COM1” ～ “COM9”
COM10～COM255	“COM10” ～ “COM255”	“¥¥. ¥COM10” ～ “¥¥. ¥COM255”
メールスロット	“mailslot¥RemoteSlotName”	“¥¥. ¥mailslot¥RemoteSlotName” or “¥¥RemoteHostName¥mailslot¥RemoteSlotName”
ソケット	サーバ名/IP アドレス : ポート番号 (ex. “192.168.0.5:14238”)	

戻り値 : 現在設定されている通信ポートの名称

11.4.29. COMポートのデバイス名称取得(GetPortDevName)

形 式 : string GetPortDevName (string ComPortName);

引 数 : ComPortName - COMポート名 (“COM1” ～ “COM255”)

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 : ≠ “” : COMポートのデバイス名
= “” : 当該COMポート無し/エラー

11.4.30. 自メールスロットパス名取得(GetMySlotPathName)

形 式 : string GetMySlotPathName ();

引 数 : なし

説 明 : 現在設定されている、自メールスロットのパス名を取得します。

戻り値 : 自メールスロットのパス名 (¥¥RMyComputerName¥mailslot¥MySlotName)

11.4.31. 自メールスロット生成(CreateMySlot)

形 式 : bool CreateMySlot ();

引 数 : なし

説 明 : MySlotName プロパティに設定されている名称で、自メールスロットを生成します。
既に、自メールスロットが生成されている場合は、自メールスロットを破棄し、(MySlotName プロパティが空文字列以外
ならば) 再度生成します。

戻り値 : true - 成功
false - 失敗

11.4.32. 自メールスロット消去(DeleteMySlot)

形 式 : void DeleteMySlot ();

引 数 : なし

説 明 : 自メールスロットを消去します。
自メールスロットが生成されていない場合は何もしません。

戻り値 : なし

11.4.33. メールスロット名情報設定(DeleteMySlot)

形 式 : void SetMailSlotNames (string MySlot, string RmtHost, string RmtSlot);

引 数 : MySlot - 自メールスロット名 (設定しない場合は null)
RmtHost - リモートコンピュータ名 (設定しない場合は null)
RmtSlot - リモートスロット名 (設定しない場合は null)

説 明 : メールスロット関連の名称 (自メールスロット名, リモートコンピュータ名, リモートスロット名) を設定します。

戻り値 : なし

11.4.34. シリアル通信コントロールの消去 (Delete)

形 式 : void Delete();

引 数 : なし

説 明 : シリアル通信コントロールを消去します。以降、本コントロールへのアクセスはできません。

このメソッドは、コンソールアプリの場合に実行してください。
Windows フォームアプリ等では実行しないでください。(自動的に実行されます)

戻り値 : なし

11.5. イベント

シリアル通信コントロールのイベント一覧を以下に示します。

#	イベント名	内容	備考
1	OnPortState	ポート状態通知	
2	OnRxChunkTxt	チャンクデータ受信通知 (テキスト)	
3	OnRxChunkBin	チャンクデータ受信通知 (バイナリ)	unsafe
4	OnRxText	テキスト受信通知	
5	OnRxEsc	E S C シーケンス受信通知	
6	OnRxCtrl	制御コード受信通知	
7	OnRxPacket	パケットデータ受信通知	unsafe
8	OnRxNoPkt	パケット外データ受信通知	
9	OnTxEmpty	送信完了	
10	OnError	エラー通知	
11	OnNtcRING	RING 変化通知	
12	OnNtcRLSD	RLSD 変化通知	
13	OnNtcDSR	DSR 変化通知	
14	OnNtcCTS	CTS 変化通知	
15	OnRxWord14	バイトペアによるワード (14Bit) データ受信	
16	OnRxInvalidChunk	不正チャンクテキスト受信通知	unsafe

11.5.1. ポート状態通知 (OnPortState)

形 式 : void OnPortState (object sender, ScpArgPortState e);

パラメタ : EScpPortState e.state - ポート状態
string e.name - ポート名 (ex. "COM3", "mailslot¥HostName", "HostName: 14238")

説 明 : 通信回線の状態が変化したことを通知します。
「e.state」で以下の状態を通知します。

#	e.state	内容
1	EScpPortState.Closed	ポートがクローズされた
2	EScpPortState.Opened	ポートがオープンされた (ソケットの場合は、クライアントと接続された)
3	EScpPortState.OpenFailure	ポートのオープンが失敗した
4	EScpPortState.PortChanged	ポート番号／通信リソースが変化した
5	EScpPortState.MySlotFail	自メールスロット生成を失敗した

戻り値 : なし

11.5.2. テキスト・チャンクデータ受信通知 (OnRxChunkTxt)

形 式 : void OnRxChunkTxt (object sender, ScpArgRxChunkTxt e);

パラメタ : string e.text - 受信したチャンク・テキストデータ

説 明 : テキスト・チャンクデータを受信したことを通知します。
このイベントは、_ChunkMode プロパティが TextData に設定されている場合に発生します。

戻り値 : なし

11.5.3. バイナリ・チャンクデータ受信通知 (OnRxChunkBin)

形 式 : void OnRxChunkBin (object sender, ScpArgRxChunkBin e);

パラメタ : Byte[] e.bin - 受信したバイナリ・チャンクデータ

説 明 : バイナリ・チャンクデータを受信したことを通知します。
このイベントは、_ChunkMode プロパティが BinaryData に設定されている場合に発生します。

戻り値 : なし

11.5.4. テキスト受信通知 (OnRxText)

形 式 : void OnRxText (object sender, ScpArgRxText e);

パラメタ : string e.text - 受信したテキストデータ

説 明 : テキストデータを受信したことを通知します。

戻り値 : なし

11.5.5. E S Cシーケンス受信通知 (OnRxEsc)

形 式 : void OnRxEsc (object sender, ScpArgRxEsc e);

パラメタ : string e.esc - 受信したE S Cシーケンス文字列 (先頭は 0x1B, 末尾は英字)

説 明 : E S Cシーケンス文字列を受信したことを通知します。

戻り値 : なし

11.5.6. 制御コード受信通知 (OnRxCtrl)

形 式 : void OnRxCtrl (object sender, ScpArgRxCtrl e);

パラメタ : char e.ctrl - 受信した制御コード (0x00~0x1F or 0x7F)

説 明 : 制御コードを受信したことを通知します。
但し、テキストに含まれる制御コード (デフォルトでは 0x09 (TAB)) は受信通知されません。
テキストに含まれる制御コードは、テキスト受診通知で通知される受信テキストデータに含まれます。

戻り値 : なし

11.5.7. パケットデータ受信通知 (OnRxPacket)

形 式 : void OnRxPacket (object sender, ScpArgRxPacket e);

パラメタ : Byte[] e.bin - 受信したパケットデータ (空パケットの場合は null)

説 明 : パケットデータを受信したことを通知します。
パケットデータは、デコードされたデータだけをを通知します。
つまり、先頭の「DLE・STX」と末尾の「DLE・ETX」は除去され、連続する 2 つの DLE を 1 つの DLE に変換したデータが通知されます。
空のパケット (データ無しで、DLE・STX と DLE・ETX の 4 バイトのみ) の場合は、e.bin = null が設定されます。

戻り値 : なし

11.5.8. パケット外テキスト受信通知 (OnRxNoPkt)

形 式 : void OnRxNoPkt (object sender, ScpArgRxNoPkt e);

パラメタ : string e.text - 受信したテキストデータ

説 明 : 受信したチャンクデータのパケット部分 (STX・DLE～ETX・DLE) を除いた部分のテキストを通知します。
通知されるデータは、テキスト受信通知 (OnRxText) とほぼ同じになりますが、テキスト受信通知の場合はテキスト終端 (0x0D や 0x0A) 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受信通知 (OnTxNoPkt) ではテキスト終端を待たずにリアルタイムに通知されます。また、パケット外テキストには制御コードも含まれます。
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。

戻り値 : なし

11.5.9. 送信完了通知 (OnTxEmpty)

形 式 : void OnRxTxEmpty (object sender, EventArgs e);

パラメタ : なし

説 明 : 送信が完了した (送信スプールバッファが空になった) ことを通知します。
このイベントは、本コントロール内の送信スプールバッファが空となった時点で発生しますが、システムの送信バッファには、まだ送信待ちのデータが存在する可能性があります。

戻り値 : なし

11.5.10. エラー発生通知 (OnError)

形 式 : void OnError (object sender, ScpArgError e);

パラメタ : EScpErr e.err; - エラー要因

説 明 : 通信エラーが発生したことを通知します。
「e.err」は、以下のエラー要因を示します。

#	エラーコード	内容
1	CE_BREAK	ブレーク信号を検出した
2	CE_FRAME	フレーミングエラーを検出した
3	CE_OVERRUN	オーバーランエラーを検出した
4	CE_RXPARITY	パリティエラーを検出した
5	CE_IOE	デバイスアクセス中に I/O エラーを検出した
6	CE_RXOVER	受信キューオーバフロー
7	CE_TXFULL	送信キュー満杯
8	CE_RXERR	受信エラー (ReadFile() / recv() でエラー)
9	CE_TXERR	送信エラー (WriteFile() / send() でエラー)

戻り値 : なし

11.5.11. R I N G信号変化通知 (OnNtcRING)

形 式 : void OnNtcRING (object sender, ScpArgNtcRING e);

パラメタ : bool e.cts - C T S信号の状態 (true:アクティブ, false:非アクティブ)
 bool e.dsr - D S R信号の状態 (true:アクティブ, false:非アクティブ)
 bool e.rlsd - R L S D信号の状態 (true:アクティブ, false:非アクティブ)
 bool e.ring - R I N G信号の状態 (true:アクティブ, false:非アクティブ)

説 明 : R I N G信号が変化したことを通知します。

戻り値 : なし

11.5.12. R L S D信号変化通知 (OnNtcRLSD)

形 式 : void OnNtcRLSD (object sender, ScpArgNtcRLSD e);

パラメタ : bool e.cts - C T S信号の状態 (true:アクティブ, false:非アクティブ)
 bool e.dsr - D S R信号の状態 (true:アクティブ, false:非アクティブ)
 bool e.rlsd - R L S D信号の状態 (true:アクティブ, false:非アクティブ)
 bool e.ring - R I N G信号の状態 (true:アクティブ, false:非アクティブ)

説 明 : R L S D信号が変化したことを通知します。

戻り値 : なし

11.5.13. D S R信号変化通知 (OnNtcDSR)

形 式 : void OnNtcDSR (object sender, ScpArgNtcDSR e);

パラメタ : bool e.cts - C T S信号の状態 (true:アクティブ, false:非アクティブ)
 bool e.dsr - D S R信号の状態 (true:アクティブ, false:非アクティブ)
 bool e.rlsd - R L S D信号の状態 (true:アクティブ, false:非アクティブ)
 bool e.ring - R I N G信号の状態 (true:アクティブ, false:非アクティブ)

説 明 : D S R信号が変化したことを通知します。

戻り値 : なし

11.5.14. C T S信号変化通知 (OnNtcCTS)

形 式 : void OnNtcCTS (object sender, ScpArgNtcCTS e);

パラメタ : bool e.cts - C T S信号の状態 (true:アクティブ, false:非アクティブ)
 bool e.dsr - D S R信号の状態 (true:アクティブ, false:非アクティブ)
 bool e.rlsd - R L S D信号の状態 (true:アクティブ, false:非アクティブ)
 bool e.ring - R I N G信号の状態 (true:アクティブ, false:非アクティブ)

説 明 : C T S信号が変化したことを通知します。

戻り値 : なし

11.5.15. バイトペアによるワード (14Bit) データ受信 (OnRxWord14)

形 式 : void OnRxWord14 (object sender, ScpArgRxWord14 e);

パラメタ : e.data - 受信した 14 ビットデータ

説 明 : 1 バイト目の MSB=1, 2 バイト目の MSB=0 である 2 バイトの受信データから抽出した 14 ビットデータを通知します。

戻り値 : なし

11.5.16. 不正チャンクテキスト受信 (OnRxInvChunk)

形 式 : unsafe void OnRxInvChunk (object sender, ScpArgRxInvChunk e);

パラメタ : Byte[] e.bin - 受信したチャンクテキスト (バイナリデータ)

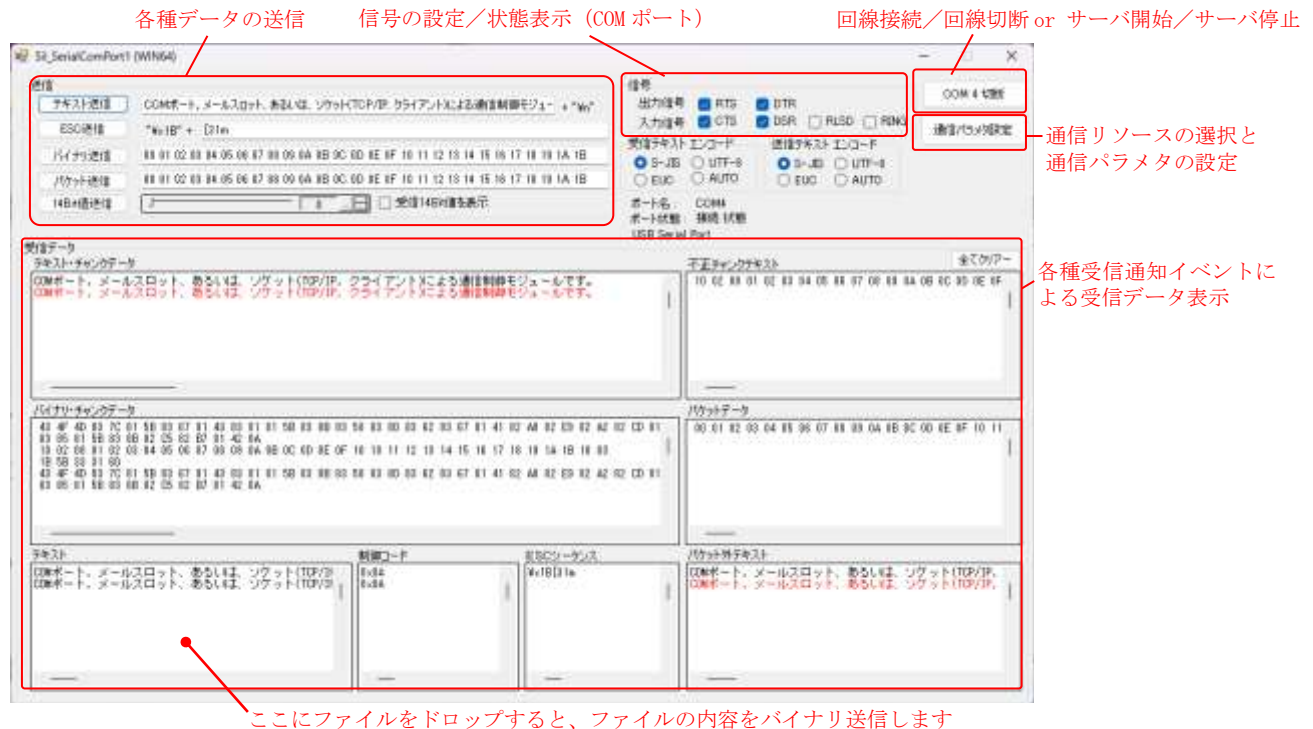
説 明 : テキストチャンクに不正な制御コードが含まれていることを通知します。
 チャンクデータの扱いをテキスト (ChunkMode = TextData) としている場合で、チャンクデータに不正な制御コード (0x09~0x0D, 0x1B 以外) が含まれている場合は、テキストチャンク受信通知 (OnRxChunkTxt) ではなく、このイベントでチャンクデータ (バイナリデータ) が通知されます。
 このバイナリデータはリアルタイムに通知されます。

戻り値 : なし

11.6. サンプルプログラム

11.6.1. Sil_SerialComPort1 (送受信テスト)

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 : // Sil_SerialComPort1
3 : //
4 : using System;
5 : using System.Collections.Generic;
6 : using System.ComponentModel;
7 : using System.Data;
8 : using System.Drawing;
9 : using System.Text;
10 : using System.Windows.Forms;
11 : using System.IO;
12 : using CAjrCustCtrl;
13 :
14 : namespace Sil_SerialComPort1
15 : {
16 :     public partial class Form1 : Form
17 :     {
18 :         bool m_FirstText = true;
19 :         string ApName = "Sil_SerialComPort1";
20 :
21 :         public Form1()
22 :         {
23 :             InitializeComponent();
24 :         }
25 :         //----- 起動時初期設定 -----//
26 :         private void Form1_Load(object sender, EventArgs e)
27 :         {
28 :             this.Text = this.Text + (IntPtr.Size == 4 ? " (WIN32)" : " (WIN64)");
29 :             // サイズ変更禁止
30 :             this.FormBorderStyle = FormBorderStyle.FixedSingle;
31 :             // ツールチップ設定
32 :             SAjrTip.Add(txtSndBin, "バイナリデータ（2桁の16進数を空白で区切って）設定してください");
33 :             SAjrTip.Add(txtSndPkt, "パケットデータ（2桁の16進数を空白で区切って）設定してください");
34 :             SAjrTip.Add(vthTxtChunk, "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" +
35 :                 "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。");
36 :             SAjrTip.Add(vthBinChunk, "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。");
37 :             SAjrTip.Add(vthInv, "リアルタイムに受信したテキストデータに不正な制御コードが含まれる場合は、\\n" +
38 :                 "テキストチャンクではなく、不正チャンクテキストとしてバイナリ表示します。\\n" +

```

```

39 :                                     "不正な制御コードとは、TAB(0x09)～CR(0x0D)以外の制御コードを意味します。");
40 :     SAjrTip.Add(vthPkt      , "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。");
41 :     SAjrTip.Add(vthNoPkt    , "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"+
42 :                                     "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。");
43 :     SAjrTip.Add(vthTxt      , "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" +
44 :                                     "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送信します。");
45 :     SAjrTip.Add(vthCtrl     , "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。");
46 :     SAjrTip.Add(vthEsc      , "受信したストリームから、E S Cシーケンス (0x1B～英字) を抜きだして表示します。");
47 :     // ウインド位置ロード
48 :     SAjrReg.LoadWndPos(this);
49 :     // 設定値ロード
50 :     SAjrReg.LoadAllCtrls(this);
51 :     // S C P 初期化
52 :     scp.Init();
53 :     // 信号出力
54 :     scp.SetRTS(chkRTS.Checked);
55 :     scp.SetDTR(chkDTR.Checked);
56 :     // テキストエンコード設定
57 :     SetTextEncode();
58 :     // 各ラベル表示
59 :     ShowLabel();
60 : }
61 : //----- フォーム終了 -----//
62 : private void Form1_FormClosed(object sender, FormClosedEventArgs e)
63 : {
64 :     // ウインド位置セーブ
65 :     SAjrReg.SaveWndPos(this);
66 :     // 設定値セーブ
67 :     SAjrReg.SaveAllCtrls(this);
68 : }
69 : //----- パラメタ設定ボタン -----//
70 : private void btnSetParam_Click_1(object sender, EventArgs e)
71 : {
72 :     scp.SetParamByDialog(this.Handle);
73 :     ShowLabel();
74 : }
75 : //----- ポート オープン/クローズ ボタン -----//
76 : private void btnOpenClose_Click(object sender, EventArgs e)
77 : {
78 :     if (scp.IsOpened || scp.IsServerStarted) scp.Close();
79 :     else                                     scp.Open();
80 :     ShowLabel();
81 : }
82 : //----- テキスト送信ボタン -----//
83 : private void btnSndTxt_Click_1(object sender, EventArgs e)
84 : {
85 :     scp.SendText(txtSndTxt.Text + "\\n");
86 : }
87 : //----- E S C 送信ボタン -----//
88 : private void btnSndEsc_Click(object sender, EventArgs e)
89 : {
90 :     scp.SendText("\\x1B" + txtSndEsc.Text);
91 : }
92 : //----- バイナリ送信ボタン -----//
93 : private void btnSndBin_Click(object sender, EventArgs e)
94 : {
95 :     string[] arr = txtSndBin.Text.Split(' ');
96 :     int      n = 0;
97 :     // 有効な数算出
98 :     for (int i = 0; i < arr.Length; i++) {
99 :         if (IsByteHexa(arr[i])) n++;
100 :     }
101 :     if (n != 0) {
102 :         // 送信バイナリ作成
103 :         Byte[] BArr = new Byte[n];
104 :
105 :         int ix = 0;
106 :         for (int i = 0; i < arr.Length; i++) {
107 :             if (IsByteHexa(arr[i])) {BArr[ix++] = Convert.ToByte(arr[i], 16);}
108 :         }
109 :         if (ix != 0) {
110 :             scp.SendBinary(BArr);
111 :         }
112 :     }
113 : }
114 : //----- パケット送信ボタン -----//
115 : private void btnSndPkt_Click(object sender, EventArgs e)
116 : {
117 :     string[] arr = txtSndPkt.Text.Split(' ');
118 :     int      n = 0;

```

```

119 :         // 有効な数算出
120 :         for (int i = 0; i < arr.Length; i++) {
121 :             if (IsByteHexa(arr[i])) n++;
122 :         }
123 :         if (n != 0) {
124 :             // 送信バイナリ作成
125 :             Byte[] BArr = new Byte[n];
126 :
127 :             int ix = 0;
128 :             for (int i = 0; i < arr.Length; i++) {
129 :                 if (IsByteHexa(arr[i])) {BArr[ix++] = Convert.ToByte(arr[i], 16);}
130 :             }
131 :             if (ix != 0) {
132 :                 scp.SendPacket(BArr);
133 :             }
134 :         }
135 :     }
136 :     // 14ビット値送信ボタン
137 :     private void btnSnd14_Click(object sender, EventArgs e)
138 :     {
139 :         scp.SendWord14LF(inp14Bit.IntValue);
140 :     }
141 :     // 全てクリアボタン
142 :     private void btnClearAll_Click(object sender, EventArgs e)
143 :     {
144 :         vthTxtChunk.Purge();
145 :         vthBinChunk.Purge();
146 :         vthTxt.Purge();
147 :         vthCtrl.Purge();
148 :         vthEsc.Purge();
149 :         vthInv.Purge();
150 :         vthPkt.Purge();
151 :         vthNoPkt.Purge();
152 :     }
153 :     //----- R T S 信号出力 -----//
154 :     private void chkRTS_CheckedChanged(object sender, EventArgs e)
155 :     {
156 :         scp.SetRTS(chkRTS.Checked);
157 :     }
158 :     //----- D T R 信号出力 -----//
159 :     private void chkDTR_CheckedChanged(object sender, EventArgs e)
160 :     {
161 :         scp.SetDTR(chkDTR.Checked);
162 :     }
163 :     //-----//
164 :     // S C P イベント通知 //
165 :     //-----//
166 :     //----- ポート状態通知 -----//
167 :     private void scp_OnPortState(object sender, ScpArgPortState e)
168 :     {
169 :         SAjrTip.Hide();
170 :         switch (e.state) {
171 :             case EScpPortState.Opened: // ・オープン／接続
172 :                 // COMポートならば、信号表示
173 :                 if (scp.PortKind == ESelPort.SEL_COMPORT) {
174 :                     scp.SetRTS(chkRTS.Checked);
175 :                     scp.SetDTR(chkDTR.Checked);
176 :                     chkCTS.Checked = scp.GetCTS();
177 :                     chkDSR.Checked = scp.GetDSR();
178 :                     chkRLSD.Checked = scp.GetRLSD();
179 :                     chkRING.Checked = scp.GetRING();
180 :                 }
181 :                 // 送受信表示有効化
182 :                 grpSend.Enabled = grpRecv.Enabled = true;
183 :                 break;
184 :             case EScpPortState.Closed: // ・クローズ／切断
185 :                 // 送受信表示無効化
186 :                 grpSend.Enabled = grpRecv.Enabled = false;
187 :                 break;
188 :             case EScpPortState.SrvStart: // ・サーバ開始
189 :                 SAjrTip.ShowCenter(this, "¥n クライアントを接続してください ¥n ");
190 :                 break;
191 :             case EScpPortState.OpenFailure: // ・オープン失敗
192 :                 MessageBox.Show(e.name + "オープン／接続失敗", ApName, MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
193 :                 break;
194 :             case EScpPortState.SrvSrtFail: // ・サーバ開始失敗
195 :                 MessageBox.Show(e.name + "サーバ開始失敗", ApName, MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
196 :                 break;
197 :             case EScpPortState.MySlotFail: // ・自スロット生成失敗
198 :                 MessageBox.Show("自メールスロット(" + e.name + ")の生成失敗", ApName, MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

```

```

199 :         break;
200 :     }
201 :     ShowLabel();
202 : }
203 : //----- テキスト受信通知 -----//
204 : private void scp_OnRxText(object sender, ScpArgRxText e)
205 : {
206 :     if (m_FirstText) {
207 :         vthTxt.Purge();
208 :         m_FirstText = false;
209 :     }
210 :     vthTxt.PutText(e.text + "\n");
211 : }
212 : //----- E S Cデータ受信通知 -----//
213 : private void scp_OnRxEsc(object sender, ScpArgRxEsc e)
214 : {
215 :     vthEsc.PutText("¥¥x1B" + e.esc.Substring(1));
216 :     vthEsc.PutText("\n");
217 : }
218 : //----- 制御コード受信通知 -----//
219 : private void scp_OnRxCtrl(object sender, ScpArgRxCtrl e)
220 : {
221 :     int c = (int)e.ctrl;
222 :     vthCtrl.PutFormat("0x{0:X2}¥n", c);
223 : }
224 : //----- パケットデータ受信通知 -----//
225 : private void scp_OnRxPacket(object sender, ScpArgRxPacket e)
226 : {
227 :     if (e.bin != null) {
228 :         vthPkt.PrintHexDump(e.bin);
229 :         vthPkt.PutText("\n");
230 :     }
231 : }
232 :
233 :
234 :
235 : //----- バイナリチャンクデータ受信通知 -----//
236 : unsafe private void scp_OnRxChunkBin(object sender, ScpArgRxChunkBin e)
237 : {
238 :     vthBinChunk.PrintHexDump(e.bin);
239 :     vthBinChunk.PutText("\n");
240 : }
241 : //----- 不正チャンクテキスト受信通知 -----//
242 : private void scp_OnRxInvChunk(object sender, ScpArgRxInvChunk e)
243 : {
244 :     vthInv.PrintHexDump(e.bin);
245 :     vthInv.PutText("\n");
246 : }
247 : //----- テキストチャンク受信通知 -----//
248 : private void scp_OnRxChunkTxt(object sender, ScpArgRxChunkTxt e)
249 : {
250 :     vthTxtChunk.PutText(e.text);
251 : }
252 : //----- パケット外テキスト受信通知 -----//
253 : private void scp_OnRxNoPkt(object sender, ScpArgRxNoPkt e)
254 : {
255 :     vthNoPkt.PutText(e.text);
256 : }
257 : //----- 14ビット値受信通知 -----//
258 : private void scp_OnRxWord14(object sender, ScpArgRxWord14 e)
259 : {
260 :     if (chkView14.Checked) {
261 :         Point pt = chkView14.ClientRectangle.Location;
262 :         pt = chkView14.PointToScreen(pt);
263 :         SAjrTip.Show(pt.X + chkView14.Size.Width + 5, pt.Y - 4, "14Bit value is received : " + e.data.ToString() +
264 :             " (0x" + e.data.ToString("X4") + ")");
265 :     }
266 : }
267 : //----- C T S信号変化通知 -----//
268 : private void scp_OnNtcCTS(object sender, ScpArgNtcCTS e)
269 : {
270 :     chkCTS.Checked = e.cts;
271 :     chkDSR.Checked = e.dsr;
272 :     chkRLSD.Checked = e.rlsd;
273 :     chkRING.Checked = e.ring;
274 : }
275 : //----- D S R信号変化通知 -----//
276 : private void scp_OnNtcDSR(object sender, ScpArgNtcDSR e)
277 : {
278 :     chkCTS.Checked = e.cts;

```

```

279 :         chkDSR.Checked = e.dsr;
280 :         chkRLSD.Checked = e.rlsd;
281 :         chkRING.Checked = e.ring;
282 :     }
283 :     //----- R L S D信号変化通知 -----//
284 :     private void scp_OnNtcRLSD(object sender, ScpArgNtcRLSD e)
285 :     {
286 :         chkCTS.Checked = e.cts;
287 :         chkDSR.Checked = e.dsr;
288 :         chkRLSD.Checked = e.rlsd;
289 :         chkRING.Checked = e.ring;
290 :     }
291 :     //----- R I N G信号変化通知 -----//
292 :     private void scp_OnNtcRING(object sender, ScpArgNtcRING e)
293 :     {
294 :         chkCTS.Checked = e.cts;
295 :         chkDSR.Checked = e.dsr;
296 :         chkRLSD.Checked = e.rlsd;
297 :         chkRING.Checked = e.ring;
298 :     }
299 :     //----- テキスト表示ウインドにファイルドロップ -----//
300 :     private void vthTxt_OnFileDrop(object sender, VthArgFileDrop e)
301 :     {
302 :         for (int i = 0; i < e.n; i++) {
303 :             string path = vthTxt.GetDroppedFile();
304 :             byte[] buf = new byte[1024];
305 :             long    FileSize;
306 :             int     ReadSize;
307 :             FileStream fs = new FileStream(path, FileMode.Open, FileAccess.Read);
308 :             FileSize = fs.Length;
309 :             while (FileSize >= 1024) {
310 :                 ReadSize = fs.Read(buf, 0, 1024);
311 :                 FileSize -= ReadSize;
312 :                 scp.SendBinary(buf);
313 :             }
314 :             if (FileSize > 0) {
315 :                 buf = new byte[FileSize];
316 :                 fs.Read(buf, 0, (int)FileSize);
317 :                 scp.SendBinary(buf);
318 :             }
319 :             fs.Dispose();
320 :         }
321 :     }
322 :     //----- 1 6進文字列チェック -----//
323 :     private bool IsByteHexa(string s)
324 :     {
325 :         bool    rc = false;
326 :         do {
327 :             if (string.IsNullOrEmpty(s)) break;
328 :             if (s.Length != 2) break;
329 :             if (!Uri.IsHexDigit(s[0])) break;
330 :             if (!Uri.IsHexDigit(s[1])) break;
331 :             rc = true;
332 :         } while (false);
333 :         return rc;
334 :     }
335 :     //----- 受信テキストエンコード設定ラジオボタン -----//
336 :     private void rbtRxSJis_CheckedChanged(object sender, EventArgs e) {SetTextEncode();}
337 :     private void rbtRxUTF8_CheckedChanged(object sender, EventArgs e) {SetTextEncode();}
338 :     private void rbtRxEuc_CheckedChanged(object sender, EventArgs e) {SetTextEncode();}
339 :     private void rbtRxAuto_CheckedChanged(object sender, EventArgs e) {SetTextEncode();}
340 :     //----- 送信テキストエンコード設定ラジオボタン -----//
341 :     private void rbtTxSJis_CheckedChanged(object sender, EventArgs e) {SetTextEncode();}
342 :     private void rbtTxUTF8_CheckedChanged(object sender, EventArgs e) {SetTextEncode();}
343 :     private void rbtTxEuc_CheckedChanged(object sender, EventArgs e) {SetTextEncode();}
344 :     private void rbtTxAuto_CheckedChanged(object sender, EventArgs e) {SetTextEncode();}
345 :     // テキストエンコード設定
346 :     private void SetTextEncode()
347 :     {
348 :         if (rbtRxSJis.Checked) scp.RxTextCode = EScpRxTextCode.SJIS;
349 :         else if (rbtRxUTF8.Checked) scp.RxTextCode = EScpRxTextCode.UTF8;
350 :         else if (rbtRxEuc.Checked) scp.RxTextCode = EScpRxTextCode.EUC;
351 :         else if (rbtRxAuto.Checked) scp.RxTextCode = EScpRxTextCode.AUTO;
352 :
353 :         if (rbtTxSJis.Checked) scp.TxTextCode = EScpTxTextCode.SJIS;
354 :         else if (rbtTxUTF8.Checked) scp.TxTextCode = EScpTxTextCode.UTF8;
355 :         else if (rbtTxEuc.Checked) scp.TxTextCode = EScpTxTextCode.EUC;
356 :         else if (rbtTxAuto.Checked) scp.TxTextCode = EScpTxTextCode.SJIS;
357 :     }
358 : }

```

```

359 : // 各ラベル表示
360 : private void ShowLabel()
361 : {
362 :     lblPortName.Text = scp.GetPortPathName();
363 :     switch (scp.PortKind) {
364 :         case ESelPort.SEL_COMPORT: // ●COMポート
365 :             int pno = scp.PortNo;
366 :             if (scp.IsOpened) { // オープン状態?
367 :                 lblPortState.Text = "接続 状態";
368 :                 btnOpenClose.Text = "COM " + pno.ToString() + " 切断";
369 :                 grpSig.Enabled = true;
370 :             }
371 :             else { // クローズ状態
372 :                 lblPortState.Text = "切断 状態";
373 :                 btnOpenClose.Text = "COM " + pno.ToString() + " 接続";
374 :                 grpSig.Enabled = false;
375 :             }
376 :             lblDevName.Text = scp.GetPortDevName("COM" + pno.ToString());
377 :             lblServ.Text = "";
378 :             break;
379 :         case ESelPort.SEL_MAILSLOT: // ●メールスロット
380 :             grpSig.Enabled = false;
381 :             if (scp.IsOpened) { // オープン状態?
382 :                 lblPortState.Text = "接続 状態";
383 :                 btnOpenClose.Text = "スロット 切断";
384 :             }
385 :             else { // クローズ状態
386 :                 lblPortState.Text = "切断 状態";
387 :                 btnOpenClose.Text = "スロット 接続";
388 :             }
389 :             lblDevName.Text = "";
390 :             lblServ.Text = "";
391 :             break;
392 :         case ESelPort.SEL_CLIENT: // ●ソケット (クライアント)
393 :             grpSig.Enabled = false;
394 :             if (scp.IsOpened) { // オープン状態?
395 :                 lblPortState.Text = "接続 状態";
396 :                 btnOpenClose.Text = "サーバから切断";
397 :             }
398 :             else { // クローズ状態
399 :                 lblPortState.Text = "切断 状態";
400 :                 btnOpenClose.Text = "サーバへ接続";
401 :             }
402 :             lblDevName.Text = "";
403 :             lblServ.Text = "";
404 :             break;
405 :         case ESelPort.SEL_SERVER: // ●ソケット (サーバ)
406 :             grpSig.Enabled = false;
407 :             if (scp.IsServerStarted) {
408 :                 if (scp.IsOpened) lblPortState.Text = "接続 状態";
409 :                 else lblPortState.Text = "切断 状態";
410 :                 btnOpenClose.Text = "サーバ停止";
411 :             }
412 :             else {
413 :                 lblPortState.Text = "切断 状態";
414 :                 btnOpenClose.Text = "サーバ起動";
415 :             }
416 :             lblDevName.Text = "";
417 :             if (scp.IsServerStarted) lblServ.Text = "サーバ動作中";
418 :             else lblServ.Text = "サーバ停止中";
419 :             break;
420 :     }
421 : }
422 : }
423 : }

```


11.6.2. Sil_SerialComPort2 (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返信）します。

```

1 : using System;
2 : using System.Collections.Generic;
3 : using System.ComponentModel;
4 : using System.Data;
5 : using System.Drawing;
6 : using System.Linq;
7 : using System.Text;
8 : using System.Windows.Forms;
9 : using CAjrCustCtrl;
10 :
11 : namespace Sil_SerialComPort2
12 : {
13 :     public partial class Form1 : Form
14 :     {
15 :         int m_ByteCount = 0;
16 :         string ApName = "Sil_serialComPort2";
17 :
18 :         public Form1()
19 :         {
20 :             InitializeComponent();
21 :         }
22 :         //----- 起動時初期設定 -----//
23 :         private void Form1_Load(object sender, EventArgs e)
24 :         {
25 :             // S C P 初期化
26 :             scp.Init();
27 :             // ウィンド位置ロード
28 :             SAjrReg.LoadWndPos(this);
29 :             // 各ラベル表示
30 :             ShowLabel();
31 :         }
32 :         //----- 終了時後処理 -----//
33 :         private void Form1_FormClosed(object sender, FormClosedEventArgs e)
34 :         {
35 :             // ウィンド位置セーブ
36 :             SAjrReg.SaveWndPos(this);
37 :         }
38 :         //----- O P E N ボタン -----//
39 :         private void btnOpen_Click(object sender, EventArgs e)
40 :         {
41 :             if (scp.IsOpened || scp.IsServerStarted) scp.Close();
42 :             else scp.Open();
43 :             ShowLabel();
44 :         }
45 :         //----- 通信パラメタボタン -----//
46 :         private void btnParam_Click(object sender, EventArgs e)
47 :         {
48 :             scp.SetParamByDialog(this.Handle);
49 :             ShowLabel();
50 :         }
51 :         //----- S C P ポート状態通知 -----//
52 :         private void scp_OnPortState(object sender, CAjrCustCtrl.ScArgPortState e)
53 :         {
54 :             SAjrTip.Hide();
55 :             switch (e.state) {
56 :                 case EScpPortState.SrvStart: // ・サーバ開始
57 :                     SAjrTip.ShowCenter(this, "¥n クライアントを接続してください ¥n ");
58 :                     break;
59 :                 case EScpPortState.OpenFailure: // ・オープン失敗
60 :                     MessageBox.Show(e.name + " オープン／接続失敗", ApName, MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
61 :                     break;
62 :                 case EScpPortState.SrvSrtFail: // ・サーバ開始失敗
63 :                     MessageBox.Show(e.name + " サーバ開始失敗", ApName, MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

```

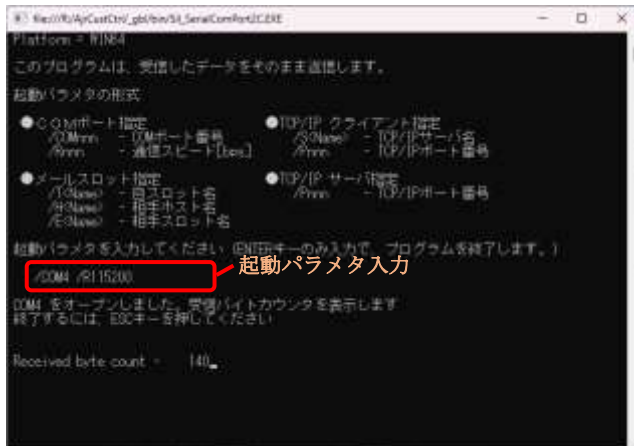
```

64 :         break;
65 :         case ESelPortState.MySlotFail: // ・自スロット生成失敗
66 :             MessageBox.Show("自メールスロット(" + e.name + ")の生成失敗", ApName, MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
67 :             break;
68 :     }
69 :     ShowLabel();
70 : }
71 : //----- S C Pバイナリデータ受信通知 -----//
72 : private void scp_OnRxChunkBin(object sender, CAjrCustCtrl.ScpArgRxChunkBin e)
73 : {
74 :     m_ByteCount += e.bin.Length;
75 :     txtBytes.Text = m_ByteCount.ToString();
76 :     scp.SendBinary(e.bin);
77 : }
78 : //----- ポート名表示 -----//
79 : private bool ShowPortName(string name)
80 : {
81 :     bool rc = false;
82 :     lblPortName.Text = name;
83 :     string pn = scp.GetPortDevName(name);
84 :     if (pn != "") {
85 :         lblDevName.Text = "(" + pn + ")";
86 :     }
87 :     else {
88 :         lblDevName.Text = "";
89 :     }
90 :     return rc;
91 : }
92 : // 各ラベル表示
93 : private void ShowLabel()
94 : {
95 :     lblPortName.Text = scp.GetPortPathName();
96 :     switch (scp.PortKind) {
97 :         case ESelPort.SEL_COMPORT: // ●COMポート
98 :             int pno = scp.PortNo;
99 :             if (scp.IsOpened) { // オープン状態?
100 :                 lblPortState.Text = "接続 状態";
101 :                 btnOpen.Text = "COM " + pno.ToString() + " 切断";
102 :             }
103 :             else { // クローズ状態
104 :                 lblPortState.Text = "切断 状態";
105 :                 btnOpen.Text = "COM " + pno.ToString() + " 接続";
106 :             }
107 :             ShowPortName("COM" + pno.ToString());
108 :             break;
109 :         case ESelPort.SEL_MAILSLLOT: // ●メールスロット
110 :             if (scp.IsOpened) { // オープン状態?
111 :                 lblPortState.Text = "接続 状態";
112 :                 btnOpen.Text = "スロット 切断";
113 :             }
114 :             else { // クローズ状態
115 :                 lblPortState.Text = "切断 状態";
116 :                 btnOpen.Text = "スロット 接続";
117 :             }
118 :             break;
119 :         case ESelPort.SEL_CLIENT: // ●ソケット (クライアント)
120 :             if (scp.IsOpened) { // オープン状態?
121 :                 lblPortState.Text = "接続 状態";
122 :                 btnOpen.Text = "サーバから切断";
123 :             }
124 :             else { // クローズ状態
125 :                 lblPortState.Text = "切断 状態";
126 :                 btnOpen.Text = "サーバへ接続";
127 :             }
128 :             break;
129 :         case ESelPort.SEL_SERVER: // ●ソケット (サーバ)
130 :             if (scp.IsServerStarted) {
131 :                 if (scp.IsOpened) lblPortState.Text = "接続 状態";
132 :                 else lblPortState.Text = "切断 状態";
133 :                 btnOpen.Text = "サーバ停止";
134 :             }
135 :             else {
136 :                 lblPortState.Text = "切断 状態";
137 :                 btnOpen.Text = "サーバ起動";
138 :             }
139 :             break;
140 :     }
141 : }
142 : }
143 : }

```

11.6.3. Sil_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、Sil_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）するコンソールアプリケーションです。



起動パラメータを入力すると、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。

起動パラメータは、Enter キーで確定します。

起動パラメータの形式は、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、相手ホスト名と相手スロット名を指定します。

```
/TMySlotName /HRemoteHostName /ERemoteSlotName
```

TCP/IP クライアントの場合、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、ポート番号だけを指定します。

```
/P14238
```

起動パラメータは、コマンドライン上でも指定可能です。

ESC キーを押すとプログラムを終了します。

```
1 : // Sil_SerialComPort2C
2 :
3 : using System;
4 : using System.Collections.Generic;
5 : using System.Linq;
6 : using System.Text;
7 : using System.Drawing;
8 : using System.Runtime.InteropServices;
9 : using CAjrCustCtrl;
10 : using System.Threading;
11 :
12 : namespace Sil_SerialComPort2C
13 : {
14 :     // 通信デバイス
15 :     enum CommDev : int {COMPORT, MAILSLLOT, CLIENT, SERVER};
16 :
17 :     class Program
18 :     {
19 :         // ヘルプテキスト
20 :         static string HelpText = " 起動パラメータの形式¥n¥n"
21 :             " ●COMポート指定                +
22 :             /COMnnn    - COM ポート番号    /S<Name>  - TCP/IP サーバ名¥n" +
23 :             /Rnnn      - 通信スピード[bps]  /Pnnn      - TCP/IP ポート番号¥n¥n" +
24 :             " ●メールスロット指定                +
25 :             /TSlotName /HRemoteHostName /ERemoteSlotName" +
26 :             " ●TCP/IP クライアント指定¥n" +
27 :             " ●TCP/IP サーバ指定¥n"
```

```

25 :                                     "      /T<Name> - 自スロット名          /Pnnn      - TCP/IP ポート番号¥n" +
26 :                                     "      /H<Name> - 相手ホスト名          ¥n" +
27 :                                     "      /E<Name> - 相手スロット名          ¥n¥n";
28 :
29 : // 通信パラメタ
30 : static CommDev dev = CommDev.COMPORT; // 通信デバイス
31 : static int ComPort = 4; // COMポート番号
32 : static int Speed = 115200; // COMポート速度
33 : static string MySlot = "NoName"; // 自スロット名
34 : static string RmtSlot = "SlotB"; // 相手スロット名
35 : static string RmtHost = ""; // 相手ホスト名
36 : static string TcpServ = ""; // TCP/IP サーバ名
37 : static uint TcpPort = 0; // TCP/IP ポート番号
38 : // バイトカウンタ
39 : static int ByteCount = 0;
40 : // シリアル通信インスタンス
41 : static CAjrSerialComPort scp = new CAjrSerialComPort();
42 : // カーソル位置
43 : static int px, py;
44 :
45 : static void Main(string[] args)
46 : {
47 :     EScpEvt evt;
48 :     object obj;
49 :     ConsoleKeyInfo c = new ConsoleKeyInfo();
50 :
51 :     // コンソールアプリ強制終了ハンドラ登録
52 :     m_CbkConApExit = new CbkConApExit(SsvConApExit);
53 :     SetConsoleCtrlHandler(m_CbkConApExit, true);
54 :
55 :     Console.WriteLine(" Platform = " + (IntPtr.Size == 4 ? "WIN32" : "WIN64") + "¥n");
56 :
57 :     Console.WriteLine(" このプログラムは、受信したデータをそのまま返信します。¥n");
58 :
59 :     do {
60 :         // 起動パラメタ解析
61 :         if (args.Length >= 1) {
62 :             AnalArgs(args.Length, args);
63 :         }
64 :         else {
65 :             Console.Write(HelperText);
66 :             Console.WriteLine(" 起動パラメタを入力してください (ENTER キーのみ入力で、プログラムを終了します。) ¥n¥n");
67 :             Console.WriteLine("");
68 :             string sl = Console.ReadLine();
69 :             if (sl == "") break;
70 :             string[] s = sl.Split(' ');
71 :             AnalArgs(s.Length, s);
72 :             Console.WriteLine("¥n");
73 :         }
74 :         //----- S C P 初期化 -----//
75 :         scp.Init();
76 :         //----- イベント待ち受けモードに設定 -----//
77 :         scp._ScpMode = EScpMode.WaitingForEvent;
78 :         //----- 対象とするイベントを設定 -----//
79 :         scp.SetEvtMask(EScpEvt.EV_PORTSTATE | EScpEvt.EV_RXCHUNK);
80 :         //----- ポート オープン -----//
81 :         bool rsu = false;
82 :         switch (dev) {
83 :             case CommDev.COMPORT: rsu = scp.Open(ComPort, Speed, EScpDataBits.DataBit_8, EScpParity.NoParity,
84 :                 EScpStopBit.StopBit_1); break;
85 :             case CommDev.MAILSLOT: scp.MySlotName = MySlot; scp.CreateMySlot();
86 :                 rsu = scp.Open(RmtHost, RmtSlot); break;
87 :             case CommDev.CLIENT: rsu = scp.Open(TcpServ, TcpPort);
88 :                 Console.WriteLine(" サーバに接続中です。¥n"); break;
89 :             case CommDev.SERVER: rsu = scp.Open(TcpPort);
90 :                 Console.WriteLine(" クライアントを接続してください。¥n"); break;
91 :         }
92 :         //----- E S C キー入力/回線切断までループ -----//
93 :         while (c.Key != ConsoleKey.Escape) {
94 :             obj = null;
95 :             //----- シリアル通信イベント待ち -----//
96 :             switch (evt = scp.WaitEvent(out obj, 1000)) {
97 :                 //----- ポート状態通知 -----//
98 :                 case EScpEvt.EV_PORTSTATE:
99 :                     if ((int)obj == 0) { // ポートクローズ
100 :                         Console.WriteLine(" " + scp.GetPortPathName() + " をクローズしました。ESC キーを押すと、プログラムを終了します¥n");
101 :                     }
102 :                     else if ((int)obj == 1) { // ポートオープン
103 :                         Console.WriteLine(" " + scp.GetPortPathName() + " をオープンしました。受信バイトカウンタを表示します¥n" +
104 :                             " 終了するには、ESC キーを押してください¥n");
105 :                         Console.WriteLine("¥n Received byte count -");

```

```

105 :         px = Console.CursorLeft;
106 :         py = Console.CursorTop;
107 :         Console.Write("    0");
108 :     }
109 :     else if ((int)obj == 2) { // ポートオープン失敗
110 :         Console.Write(" " + scp.GetPortPathName() + " のオープンに失敗しました。ESC キーを押すと、プログラムを終了します\n");
111 :     }
112 :     break;
113 : //----- バイナリチャンク受信 -----//
114 : case ESceEvt.EV_RXCHUNK:
115 :     Type t = obj.GetType();
116 :     if (t == typeof(byte[])) {
117 :         scp.SendBinary((byte[])obj);
118 :         ByteCount += ((byte[])obj).Length;
119 :         Console.SetCursorPosition(px, py);
120 :         Console.Write("    " + ByteCount.ToString());
121 :     }
122 :     break;
123 : }
124 : //--- キー入力 -----//
125 : if (Console.KeyAvailable) c = Console.ReadKey(true);
126 : }
127 : //----- ポートクローズ -----//
128 : if (scp.IsOpened) scp.Close();
129 : } while(false);
130 : //----- シリアル通信オブジェクト削除 -----//
131 : scp.Delete();
132 : }
133 :
134 : // 起動パラメタ解析
135 : static void AnalArgs(int n, string[] args)
136 : {
137 :     int nn;
138 :     foreach (string s in args) {
139 :         string r;
140 :         if ((r = SAjrGsr.AfterPrefix(s, "/COM", out nn)) != null) {dev = CommDev.COMPORT; ComPort = nn;}
141 :         else if ((r = SAjrGsr.AfterPrefix(s, "/R", out nn)) != null) {dev = CommDev.COMPORT; Speed = nn;}
142 :         else if ((r = SAjrGsr.AfterPrefix(s, "/T", out nn)) != null) {dev = CommDev.MAILSLOT; MySlot = r;}
143 :         else if ((r = SAjrGsr.AfterPrefix(s, "/H", out nn)) != null) {dev = CommDev.MAILSLOT; RmtHost = r;}
144 :         else if ((r = SAjrGsr.AfterPrefix(s, "/E", out nn)) != null) {dev = CommDev.MAILSLOT; RmtSlot = r;}
145 :         else if ((r = SAjrGsr.AfterPrefix(s, "/S", out nn)) != null) {dev = CommDev.CLIENT; TcpServ = r;}
146 :         else if ((r = SAjrGsr.AfterPrefix(s, "/P", out nn)) != null) {dev = CommDev.SERVER; TcpPort = (uint)nn;}
147 :         else {
148 :             Console.WriteLine("*** Invalid parameter '" + s + "' ***");
149 :         }
150 :     }
151 :     if (TcpPort != 0) {
152 :         if (TcpServ != "") dev = CommDev.CLIENT;
153 :         else dev = CommDev.SERVER;
154 :     }
155 : }
156 :
157 : // コンソールアプリ終了ハンドラ用デリゲート
158 : [DllImport("Kernel32")]
159 : static extern bool SetConsoleCtrlHandler(CbkConApExit Handler, bool Add);
160 : delegate bool CbkConApExit(EAJCEXITTYPE ExitType);
161 : static CbkConApExit m_CbkConApExit;
162 : // コンソールアプリ終了ハンドラ
163 : static bool SsvConApExit(EAJCEXITTYPE ExitType)
164 : {
165 :     // シリアル通信オブジェクト消去
166 :     scp.Delete();
167 :     // false : 次のイベントハンドラへリンクする
168 :     return false;
169 : }
170 : }
171 : }
172 :
173 :

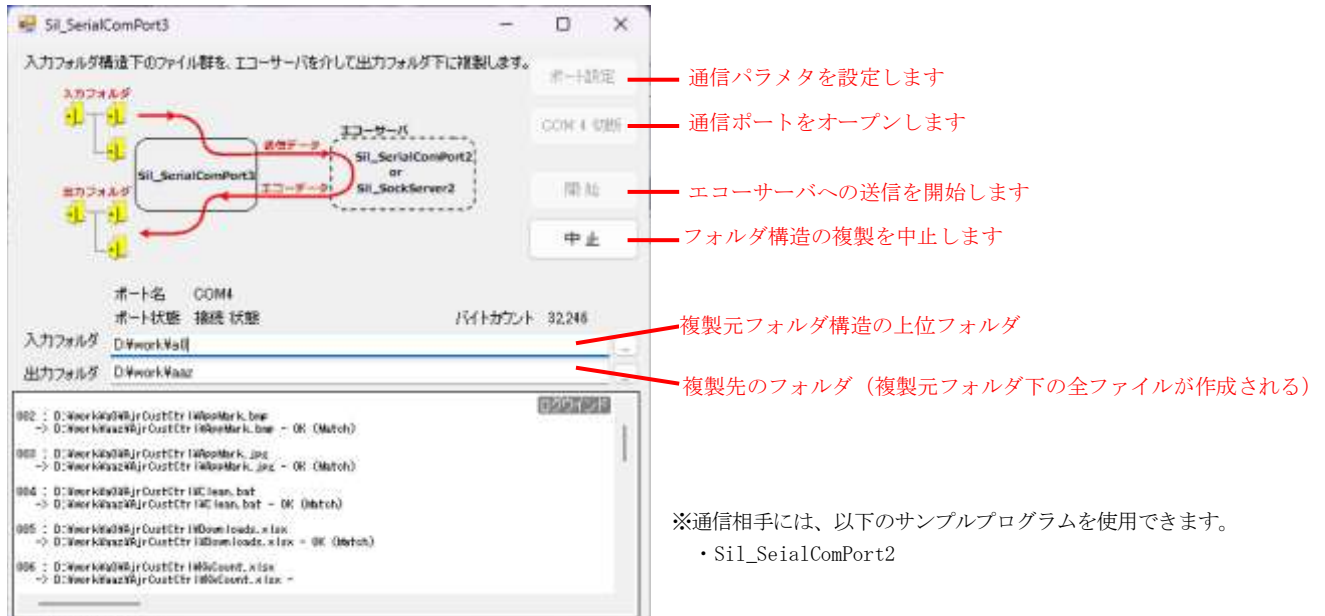
```

11.6.4. Sil_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（データの返信）を行う通信相手を介して、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通信相手に送信し、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パシ、一致／不一致をログ表示します。

複製する前に、複製先フォルダ下の全フォルダやファイルは消去されます。



※通信相手には、以下のサンプルプログラムを使用できます。

・Sil_SerialComPort2

```

1 : using System;
2 : using System.Collections.Generic;
3 : using System.ComponentModel;
4 : using System.Data;
5 : using System.Drawing;
6 : using System.Linq;
7 : using System.Text;
8 : using System.Windows.Forms;
9 : using System.IO;
10 : using System.Collections;
11 : using CAjrCustCtrl;
12 :
13 : namespace Sil_SerialComPort3
14 : {
15 :     public partial class Form1 : Form
16 :     {
17 :         string    ApName = "Sil_SerialComPort3";
18 :         string    m_InpDir;           // 入力フォルダパス
19 :         string    m_OutDir;          // 出力フォルダパス
20 :         string    m_InpPath;         // 入力ファイルパス
21 :         string    m_OutPath;         // 出力ファイルパス
22 :         FileStream m_InpFs;           // 入力ファイルストリーム
23 :         FileStream m_OutFs;           // 出力ファイルストリーム
24 :         long      m_FileSize;         // 入力ファイルサイズ
25 :         int       m_Unmatch;          // 比較不一致ファイル数
26 :         int       m_ListCount;        // 全ファイル数
27 :         int       m_ListIx;           // ファイルインデクス
28 :         ArrayList m_list = new ArrayList(); // 全入出力ファイル リスト
29 :         int       m_BufSize = 1024;   // 送信データバッファサイズ
30 :         int       m_Bytes = 0;        // 送信バイトカウント
31 :         bool      m_fSendBusy = false; // 送信中フラグ
32 :
33 :         public Form1()
34 :         {
35 :             InitializeComponent();
36 :         }
37 :
38 :         private void Form1_Load(object sender, EventArgs e)
39 :         {
40 :             // ビットマップ表示
41 :             System.Reflection.Assembly myAssembly = System.Reflection.Assembly.GetExecutingAssembly();
42 :             Bitmap bmp = new Bitmap(myAssembly.GetManifestResourceStream("Sil_SerialComPort3.ProgImage.bmp"));
43 :             bmp.MakeTransparent();
44 :             pic.Image = bmp;

```

```

45 :         // ウインド位置ロード
46 :         SAjrReg.LoadWndPos(this);
47 :         // 設定値ロード
48 :         SAjrReg.LoadAllCtrls(this);
49 :         // S C P 初期化
50 :         scp.Init();
51 :         // パケット受信タイムアウト (低速通信用に長めに設定する)
52 :         scp.PktTimeout = 60000;
53 :         // ポート名表示
54 :         lblPortName.Text = scp.GetPortName();
55 :         // ラベル表示
56 :         ShowLabel();
57 :     }
58 :     // フォーム終了
59 :     private void Form1_FormClosed(object sender, FormClosedEventArgs e)
60 :     {
61 :         // ウインド位置セーブ
62 :         SAjrReg.SaveWndPos(this);
63 :         // 設定値セーブ
64 :         SAjrReg.SaveAllCtrls(this);
65 :     }
66 :     // 入力パス設定ボタン
67 :     private void btnInpPath_Click(object sender, EventArgs e)
68 :     {
69 :         string path;
70 :         if ((path = SAjrGsr.GetFolder("入力フォルダ", "")) != "") {
71 :             txtInpPath.Text = path;
72 :         }
73 :     }
74 :     // 出力パス設定ボタン
75 :     private void btnOutPath_Click(object sender, EventArgs e)
76 :     {
77 :         string path;
78 :         if ((path = SAjrGsr.GetFolder("出力フォルダ", "")) != "") {
79 :             txtOutPath.Text = path;
80 :         }
81 :     }
82 :     // 通信パラメタ設定ボタン
83 :     private void btnParam_Click(object sender, EventArgs e)
84 :     {
85 :         scp.SetParamByDialog();
86 :         ShowLabel();
87 :     }
88 :     // オープンボタン
89 :     private void btnOpen_Click(object sender, EventArgs e)
90 :     {
91 :         if (scp.IsOpened || scp.IsServerStarted) scp.Close();
92 :         else scp.Open();
93 :         ShowLabel();
94 :     }
95 :     // 開始ボタン
96 :     private void btnStart_Click(object sender, EventArgs e)
97 :     {
98 :         // 入出力パス設定
99 :         m_InpDir = txtInpPath.Text;
100 :         m_OutDir = txtOutPath.Text;
101 :         if (m_InpDir != "" && m_OutDir != "") {
102 :             if (m_InpDir != m_OutDir) {
103 :                 // 入出力パスリストクリアー
104 :                 m_list.Clear();
105 :                 // ファイル検索 (入出力パスリスト作成)
106 :                 m_ListCount = 0;
107 :                 m_ListIx = 0;
108 :                 m_Unmatch = 0;
109 :                 fsr.SearchFiles(m_InpDir, "*.*", true);
110 :                 // フォルダ構造コピー
111 :                 if (m_ListCount != 0) {
112 :                     // 出力フォルダ下の全ファイル削除
113 :                     if (MessageBox.Show(m_OutDir + "下の全ファイルを削除します。よろしいですか?", "Sil_SerialComPort3",
114 :                         MessageBoxButtons.YesNo, MessageBoxIcon.Exclamation) == DialogResult.Yes) {
115 :                         // バイトカウンタクリアー
116 :                         m_Bytes = 0;
117 :                         lblBytes.Text = "0";
118 :                         // ログクリアー
119 :                         vthLog.Purge();
120 :                         // ボタングレー化
121 :                         EnableButtons(false, false, false, true);
122 :                         // フォルダ下クリアー
123 :                         SAjrFop.CleanFolder(m_OutDir);
124 :                         // フォルダ構造コピー

```

```

125 :             SAjrFop.CopyFolderStruct(m_InpDir, m_OutDir);
126 :             // ファイルをオープンし送信開始
127 :             SendNextFile();
128 :         }
129 :     } else SAjrTip.ShowCenter(this, "¥x1B[31m" + "入力フォルダ下にファイルがありません。");
130 :
131 :     } else SAjrTip.ShowCenter(this, "¥x1B[31m" + "入出力フォルダが同じです。");
132 :
133 :     } else SAjrTip.ShowCenter(this, "¥x1B[31m" + "入出力フォルダが設定されていません。");
134 : }
135 : // 中止ボタン
136 : private void btnStop_Click(object sender, EventArgs e)
137 : {
138 :     // 送信中フラグクリア
139 :     m_fSendBusy = false;
140 :     // ポートクローズ
141 :     scp.Close();
142 :     // ファイルストリーム解放
143 :     if (m_InpFs != null) {m_InpFs.Dispose(); m_InpFs = null;}
144 :     if (m_OutFs != null) {m_OutFs.Dispose(); m_OutFs = null;}
145 :     vthLog.PutText("¥n¥n--- ループバックテストを中止しました ---¥n");
146 : }
147 : // ポート状態通知
148 : private void scp_OnPortState(object sender, ScpArgPortState e)
149 : {
150 :     SAjrTip.Hide();
151 :     lblPortName.Text = e.name;
152 :     switch (e.state) {
153 :         case EScpPortState.Opened: // ・オープン
154 :             EnableButtons(false, true, true, false);
155 :             break;
156 :         case EScpPortState.Closed: // ・クローズ
157 :             // データ送信中ならば、中止ボタン押下
158 :             if (m_fSendBusy) {
159 :                 btnStop.PerformClick();
160 :             }
161 :             EnableButtons(true, true, false, false);
162 :             break;
163 :         case EScpPortState.SrvStart: // ・サーバ開始
164 :             SAjrTip.ShowCenter(this, "¥n エコー クライアントを接続してください ¥n ");
165 :             break;
166 :         case EScpPortState.OpenFailure: // ・オープン失敗
167 :             EnableButtons(true, true, false, false);
168 :             MessageBox.Show(e.name + "オープン／接続失敗", ApName, MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
169 :             break;
170 :         case EScpPortState.SrvSrtFail: // ・サーバ開始失敗
171 :             MessageBox.Show(e.name + "サーバ開始失敗", ApName, MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
172 :             break;
173 :         case EScpPortState.MySlotFail: // ・自スロット生成失敗
174 :             MessageBox.Show("自メールスロット(" + e.name + ")の生成失敗", ApName, MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
175 :             break;
176 :     }
177 :     ShowLabel();
178 : }
179 : // パケット受信通知
180 : private void scp_OnRxPacket(object sender, ScpArgRxPacket e)
181 : {
182 :     // 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
183 :     if (e.bin != null) {
184 :         if (m_OutFs != null) {
185 :             m_OutFs.Write(e.bin, 0, e.bin.Length);
186 :         }
187 :     }
188 :     // 空パケット（ファイル終端）ならば、次のファイル送信へ・・・
189 :     else {
190 :         // 出力ファイルクローズ
191 :         if (m_OutFs != null) {m_OutFs.Dispose(); m_OutFs = null;}
192 :         // ファイルコンペア
193 :         if (SAjrFop.FileCompare(m_InpPath, m_OutPath)) {
194 :             vthLog.PutText("OK (Match)¥n");
195 :         }
196 :         else {
197 :             m_Unmatch++;
198 :             vthLog.PutText("¥x1B[31mNG (Unmatch)¥x1B[0m¥n");
199 :         }
200 :         // 次のファイル送信開始
201 :         SendNextFile();
202 :     }
203 : }
204 : // 送信完了通知

```



```

205 : private void scp_OnTxEmpty(object sender, EventArgs e)
206 : {
207 :     // 次のファイルデータ送信
208 :     if (m_InpFs != null) {
209 :         FileSend();
210 :     }
211 : }
212 : // ファイル検索通知
213 : private bool fsr_OnFindFile(object sender, FsrArgFindFile e)
214 : {
215 :     if (e.FileAtt != EFileAtt._A_SUBDIR) {
216 :         // 入出力ファイルパス設定 (セミコロン;)で区切る)
217 :         string dtail, outpath;
218 :         dtail = Path.GetDirectoryName(e.PathName);
219 :         dtail = SAjrGsr.PathCat(dtail, "");
220 :         dtail = dtail.Substring(m_InpDir.Length);
221 :         outpath = SAjrGsr.PathCat(m_OutDir + dtail, e.FileName);
222 :         m_list.Add(e.PathName + ";" + outpath);
223 :         m_ListCount++;
224 :     }
225 :     return true;
226 : }
227 : // 次のファイル送信
228 : private bool SendNextFile()
229 : {
230 :     bool rc = false;
231 :     if (FileOpen()) {
232 :         m_fSendBusy = true;
233 :         FileSend();
234 :         rc = true;
235 :     }
236 :     else {
237 :         m_fSendBusy = false;
238 :         scp.Close();
239 :         vthLog.PutText("Yn--- ループバックテストを終了しました ---Yn");
240 :         vthLog.PutText("   ループバック不一致数 : " + m_Unmatch.ToString() + " / " +
241 :             m_ListCount.ToString() + " [files]Yn");
242 :     }
243 :     return rc;
244 : }
245 : // 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)
246 : private bool FileOpen()
247 : {
248 :     bool rc = false;
249 :     while (m_ListIx < m_ListCount) {
250 :         // 入出力パス名設定
251 :         Object obj = m_list[m_ListIx++];
252 :         string pathes = (string)obj;
253 :         string[] s = pathes.Split(';');
254 :         m_InpPath = s[0];
255 :         m_OutPath = s[1];
256 :         vthLog.PutText("Yn" + m_ListIx.ToString("D3") + " : " + m_InpPath + "Yn");
257 :         vthLog.PutText("  -> " + m_OutPath + " - ");
258 :         // 入出力ファイルオープン
259 :         m_InpFs = new FileStream(m_InpPath, FileMode.Open);
260 :         m_OutFs = new FileStream(m_OutPath, FileMode.Create);
261 :         // ファイルサイズ設定
262 :         m_FileSize = m_InpFs.Length;
263 :         // ゼロサイズならば、ファイルクローズ
264 :         if (m_FileSize == 0) {
265 :             vthLog.PutText("OK (Zero size)Yn");
266 :             if (m_InpFs != null) {m_InpFs.Dispose(); m_InpFs = null;}
267 :             if (m_OutFs != null) {m_OutFs.Dispose(); m_OutFs = null;}
268 :         }
269 :         // ゼロサイズ以外ならば、ファールオープンした旨を返す
270 :         else {
271 :             rc = true;
272 :             break;
273 :         }
274 :     }
275 :     return rc;
276 : }
277 : // ファイル送信
278 : private void FileSend()
279 : {
280 :     // 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
281 :     if (m_FileSize > m_BufSize) {
282 :         byte[] buf = new byte[m_BufSize];
283 :         m_InpFs.Read(buf, 0, m_BufSize);
284 :         m_Bytes += scp.SendPacket(buf);

```

```

285 :         m_FileSize -= m_BufSize;
286 :     }
287 :     // 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
288 :     else if (m_FileSize != 0) {
289 :         byte[] buf = new byte[m_FileSize];
290 :         m_InpFs.Read(buf, 0, (int)m_FileSize);
291 :         m_Bytes += scp.SendPacket(buf);
292 :         m_FileSize = 0;
293 :     }
294 :     // 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
295 :     else {
296 :         // 空パケット送信
297 :         m_Bytes += scp.SendPacket(null);
298 :         // 入力ファイルクローズ
299 :         if (m_InpFs != null) {m_InpFs.Dispose(); m_InpFs = null;}
300 :     }
301 :     // 送信バイトカウント表示
302 :     lblBytes.Text = SAjrGsr.SepDecimal(m_Bytes.ToString());
303 : }
304 : // ボタン群許可
305 : private void EnableButtons(bool fSetServ, bool fConnect, bool fStart, bool fStop)
306 : {
307 :     btnParam.Enabled = fSetServ;
308 :     btnOpen.Enabled = fConnect;
309 :     btnStart.Enabled = fStart;
310 :     btnStop.Enabled = fStop;
311 : }
312 : // 各ラベル表示
313 : private void ShowLabel()
314 : {
315 :     lblPortName.Text = scp.GetPortPathName();
316 :     switch (scp.PortKind) {
317 :         case ESelPort.SEL_COMPORT: // ●COMポート
318 :             int pno = scp.PortNo;
319 :             if (scp.IsOpened) { // オープン状態?
320 :                 lblState.Text = "接続 状態";
321 :                 btnOpen.Text = "COM " + pno.ToString() + " 切断";
322 :             }
323 :             else { // クローズ状態
324 :                 lblState.Text = "切断 状態";
325 :                 btnOpen.Text = "COM " + pno.ToString() + " 接続";
326 :             }
327 :             break;
328 :         case ESelPort.SEL_MAILSLLOT: // ●メールスロット
329 :             if (scp.IsOpened) { // オープン状態?
330 :                 lblState.Text = "接続 状態";
331 :                 btnOpen.Text = "スロット 切断";
332 :             }
333 :             else { // クローズ状態
334 :                 lblState.Text = "切断 状態";
335 :                 btnOpen.Text = "スロット 接続";
336 :             }
337 :             break;
338 :         case ESelPort.SEL_CLIENT: // ●ソケット (クライアント)
339 :             if (scp.IsOpened) { // オープン状態?
340 :                 lblState.Text = "接続 状態";
341 :                 btnOpen.Text = "サーバから切断";
342 :             }
343 :             else { // クローズ状態
344 :                 lblState.Text = "切断 状態";
345 :                 btnOpen.Text = "サーバへ接続";
346 :             }
347 :             break;
348 :         case ESelPort.SEL_SERVER: // ●ソケット (サーバ)
349 :             if (scp.IsServerStarted) {
350 :                 if (scp.IsOpened) lblState.Text = "接続 状態";
351 :                 else lblState.Text = "切断 状態";
352 :                 btnOpen.Text = "サーバ停止";
353 :             }
354 :             else {
355 :                 lblState.Text = "切断 状態";
356 :                 btnOpen.Text = "サーバ起動";
357 :             }
358 :             break;
359 :     }
360 : }
361 : }
362 : }

```

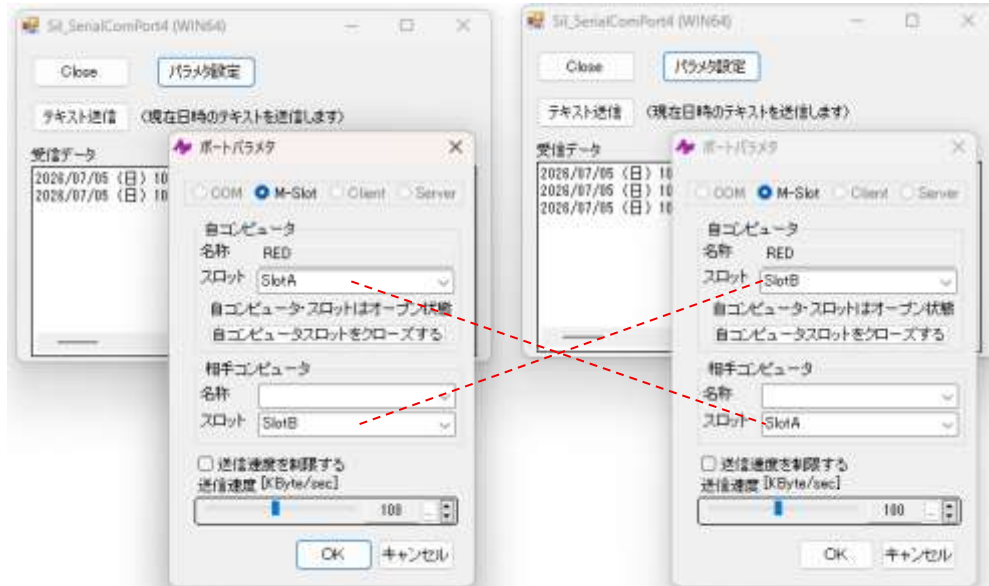
11.6.5. Sil_SerialComPort4 (メールスロット通信)

メールスロットによる通信のサンプルプログラムです。

プログラムを起動すると、自分自身をもう1つ起動します。(Sil_SerialComPort4.exe が2つ起動された状態になります) 互いのプログラムでは、自身のメールスロットと相手のメールスロット情報を(固定で)設定します。

「パラメタ設定」ボタンでは、メールスロットの情報だけを設定可能とします。

「Open」ボタンを押した後、「テキスト送信」ボタンを押すと相手に、現在時刻のテキストを送信します。



```

1 : using System;
2 : using System.Collections.Generic;
3 : using System.ComponentModel;
4 : using System.Data;
5 : using System.Drawing;
6 : using System.Text;
7 : using System.Windows.Forms;
8 : using CAjrCustCtrl;
9 :
10 : namespace Sil_SerialComPort4
11 : {
12 :     public partial class Form1 : Form
13 :     {
14 :         string      ApName = "Sil_SerialComPort4";
15 :
16 :         System.Threading.Mutex m_Mutex = new System.Threading.Mutex(false, "Sil_SerialComPort4");
17 :
18 :         public Form1()
19 :         {
20 :             InitializeComponent();
21 :         }
22 :         //----- 起動時初期設定 -----//
23 :         private void Form1_Load(object sender, EventArgs e)
24 :         {
25 :             this.Text = this.Text + (IntPtr.Size == 4 ? " (WIN32)" : " (WIN64)");
26 :             this.Disposed += OnUnloadMyControl;
27 :             // サイズ変更禁止
28 :             this.FormBorderStyle = FormBorderStyle.FixedSingle;
29 :             // S C P 初期化
30 :             scp.Init();
31 :             // メールスロットのみ設定可とする
32 :             scp.EnablePortSelectionInDialog(false, true, false, false);
33 :             // メールスロット名設定
34 :             if (m_Mutex.WaitOne(0, false)) {
35 :                 scp.SetMailSlotNames("SlotA", null, "SlotB");
36 :                 scp.CreateMySlot();
37 :                 // ウインド位置設定
38 :                 this.Left = 100;
39 :                 this.Top = 100;
40 :                 // 相手局プログラム起動
41 :                 System.Diagnostics.Process.Start("Sil_SerialComPort4.exe");
42 :             }
43 :             else {
44 :                 scp.SetMailSlotNames("SlotB", null, "SlotA");

```

```

45 :         scp.CreateMySlot();
46 :         // ウインド位置設定
47 :         this.Left = 100 + this.Width + 10;
48 :         this.Top = 100;
49 :     }
50 : }
51 : //----- オープン／クローズ ボタン -----//
52 : private void btnOpenClose_Click(object sender, EventArgs e)
53 : {
54 :     if (scp.IsOpened) scp.Close();
55 :     else                scp.Open();
56 : }
57 : //----- パラメタ設定ボタン -----//
58 : private void btnSetParam_Click(object sender, EventArgs e)
59 : {
60 :     scp.SetParamByDialog(this.Handle);
61 : }
62 : //----- テキスト送信ボタン -----//
63 : private void btnSndTxt_Click(object sender, EventArgs e)
64 : {
65 :     DateTime dtNow = DateTime.Now;
66 :     string str = dtNow.ToString("yyyy/MM/dd (ddd) hh:mm:ss");
67 :
68 :     scp.SendText(str + "\n");
69 : }
70 : //----- シリアルポート状態通知 -----//
71 : private void cAjrSerialComPort1_OnPortState(object sender, CAjrCustCtrl.ScpArgPortState e)
72 : {
73 :     if (e.state == EScpPortState.Opened) {
74 :         btnOpenClose.Text = "Close";
75 :         btnSndTxt.Enabled = true;
76 :     }
77 :     else if (e.state == EScpPortState.Closed) {
78 :         btnOpenClose.Text = "Open";
79 :         btnSndTxt.Enabled = false;
80 :     }
81 :     else if (e.state == EScpPortState.OpenFailure) {
82 :         MessageBox.Show(e.name + "オープン／接続失敗", ApName, MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
83 :     }
84 :     else if (e.state == EScpPortState.MySlotFail) {
85 :         MessageBox.Show("自メーカスロット(" + e.name + ")の生成失敗", ApName, MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
86 :     }
87 : }
88 : //----- テキスト受信通知 -----//
89 : private void cAjrSerialComPort1_OnRxText(object sender, CAjrCustCtrl.ScpArgRxText e)
90 : {
91 :     vthRxTxt.PutText(e.text);
92 : }
93 : //----- 制御文字受信通知 -----//
94 : private void cAjrSerialComPort1_OnRxCtrl(object sender, CAjrCustCtrl.ScpArgRxCtrl e)
95 : {
96 :     vthRxTxt.PutChar(e.ctrl);
97 : }
98 : //----- 終了処理 -----//
99 : void OnUnloadMyControl(object sender, EventArgs e)
100 : {
101 :     // ミューテックス解放
102 :     m_Mutex.ReleaseMutex();
103 : }
104 : }
105 : }

```

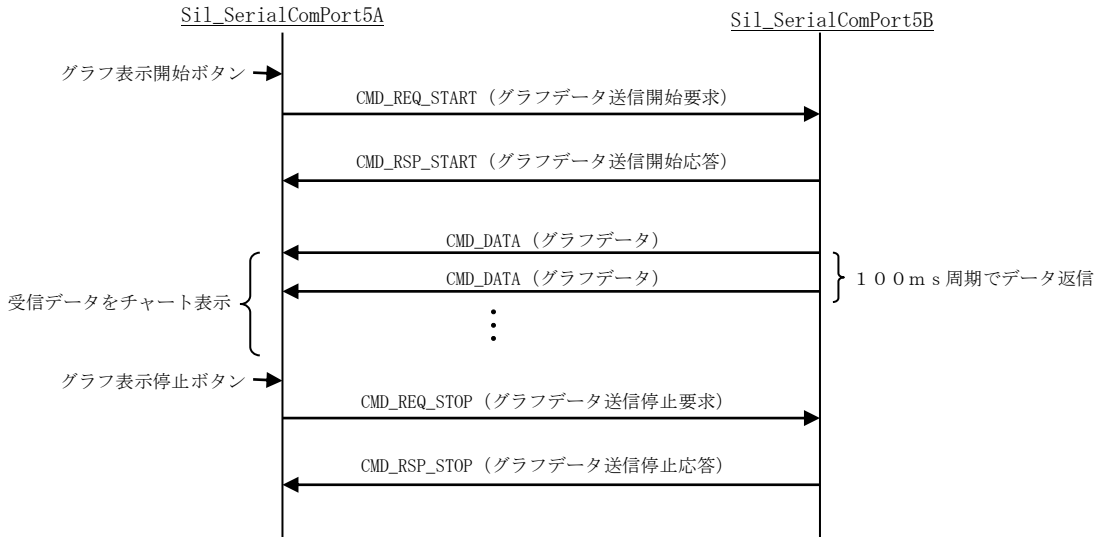
11.6.6. Sil_SerialComPort5 (テキストとバイナリパケットの多重通信)

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。

- Sil_SerialComPort5A - Sil_SerialComPort5B へコマンドを送信します。
- Sil_SerialComPort5B - Sil_SerialComPort5A からのコマンド要求に従い、データを返信します。

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。

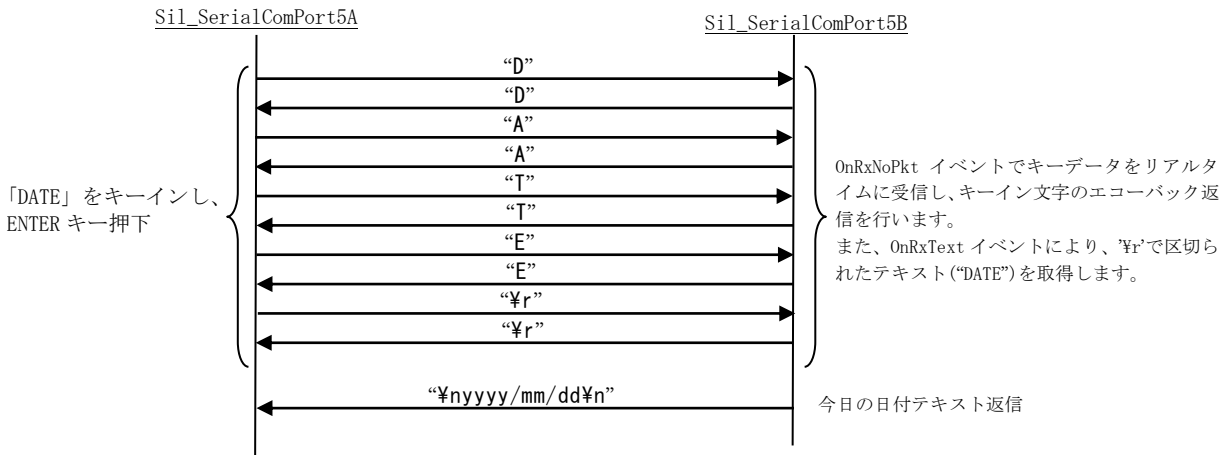


この2つのプログラムは、バイナリパケットでの通信と多重して、以下のテキスト・コマンド通信も行います。

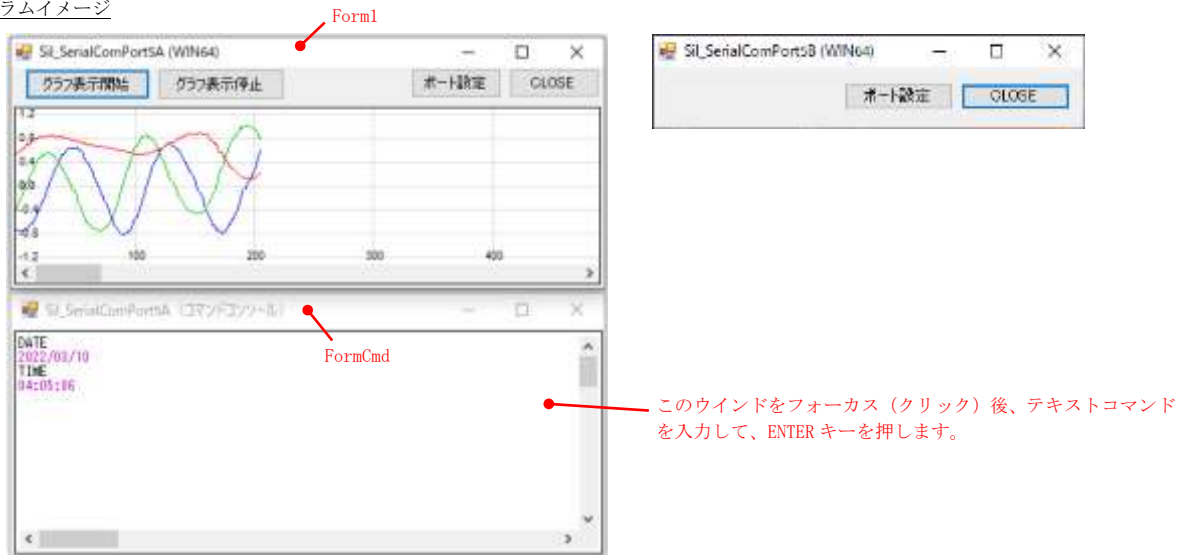
テキスト・コマンドは、以下の2つです。

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



Sil_SerialComPort5A (Form1)

```

1 : using System;
2 : using System.Collections.Generic;
3 : using System.ComponentModel;
4 : using System.Data;
5 : using System.Drawing;
6 : using System.Linq;
7 : using System.Text;
8 : using System.Windows.Forms;
9 : using System.Diagnostics;
10 : using System.Runtime.InteropServices;
11 : using CAjrCustCtrl;
12 :
13 : namespace Sil_SerialComPort5A
14 : {
15 :     public partial class Form1 : Form
16 :     {
17 :         string ApName = "Sil_SerialComPort5A";
18 :         // サブフォーム
19 :         FormCmd m_FormCmd;
20 :         // 通信パケットコマンド
21 :         byte CMD_REQ_START = 0x01; // グラフデータ送信開始要求
22 :         byte CMD_RSP_START = 0x11; // " 応答
23 :         byte CMD_REQ_STOP = 0x02; // グラフデータ送信停止要求
24 :         byte CMD_RSP_STOP = 0x12; // " 応答
25 :         byte CMD_DATA = 0x40; // グラフデータ
26 :         // グラフデータ形式
27 :         [StructLayout(LayoutKind.Sequential, Pack=1)]
28 :         struct CMDDATA {
29 :             public byte cmd; // コマンドコード
30 :             public byte f1, f2, f3; // -
31 :             public AJC3DVEC vec; // グラフデータ
32 :         }
33 :
34 :         System.Threading.Mutex m_Mut5A = new System.Threading.Mutex(true, "Sil_SerialComPort5A");
35 :         System.Threading.Mutex m_Mut5B;
36 :
37 :         public Form1()
38 :         {
39 :             InitializeComponent();
40 :         }
41 :
42 :         private void Form1_Load(object sender, EventArgs e)
43 :         {
44 :             this.Text = this.Text + (IntPtr.Size == 4 ? " (WIN32)" : " (WIN64)");
45 :             // コマンドフォーム表示
46 :             m_FormCmd = new FormCmd();
47 :             m_FormCmd.FormBorderStyle = FormBorderStyle.FixedSingle;
48 :             m_FormCmd.StartPosition = FormStartPosition.Manual;
49 :             m_FormCmd.Left = this.Left;
50 :             m_FormCmd.Top = this.Top + this.Height;
51 :             m_FormCmd.Show();

```

```

52 :         // SCP 初期化
53 :         scp.Init();
54 :         // 相手プログラム起動
55 :         bool fCreateNew;
56 :         m_Mut5B = new System.Threading.Mutex(false, "Sil_SerialComPort5B", out fCreateNew);
57 :         m_Mut5B.Close();
58 :         if (fCreateNew) {
59 :             System.Diagnostics.Process.Start("Sil_SerialComPort5B.exe");
60 :         }
61 :         m_FormCmd.Focus();
62 :         ShowLabel();
63 :     }
64 :     //----- 終了時後処理 -----//
65 :     private void Form1_FormClosed(object sender, FormClosedEventArgs e)
66 :     {
67 :     }
68 :     //----- グラフ表示開始ボタン -----//
69 :     private unsafe void btnStart_Click(object sender, EventArgs e)
70 :     {
71 :         fixed (byte* p = &CMD_REQ_START) {
72 :             scp.SendPacket(p, 1);
73 :         }
74 :         m_FormCmd.Focus();
75 :     }
76 :     //----- グラフ表示停止ボタン -----//
77 :     private unsafe void btnStop_Click(object sender, EventArgs e)
78 :     {
79 :         fixed (byte* p = &CMD_REQ_STOP) {
80 :             scp.SendPacket(p, 1);
81 :         }
82 :         m_FormCmd.Focus();
83 :     }
84 :     //----- ポート設定ボタン -----//
85 :     private void btnSetPort_Click(object sender, EventArgs e)
86 :     {
87 :         scp.SetParamByDialog(this.Handle);
88 :         m_FormCmd.Focus();
89 :         ShowLabel();
90 :     }
91 :     //----- OPEN/CLOSE ボタン -----//
92 :     private void btnOpenClose_Click(object sender, EventArgs e)
93 :     {
94 :         if (scp.IsOpened || scp.IsServerStarted) scp.Close();
95 :         else scp.Open();
96 :         m_FormCmd.Focus();
97 :         ShowLabel();
98 :     }
99 :     //----- SCP : シリアルポート状態通知 -----//
100 :    private void scp_OnPortState(object sender, ScpArgPortState e)
101 :    {
102 :        SAjrTip.Hide();
103 :        switch (e.state) {
104 :            case EScpPortState.Opened: // ・オープン
105 :                btnStart.Enabled = btnStop.Enabled = true;
106 :                break;
107 :            case EScpPortState.Closed: // ・クローズ
108 :                btnStart.Enabled = btnStop.Enabled = false;
109 :                break;
110 :            case EScpPortState.SrvStart: // ・サーバ開始
111 :                SAjrTip.ShowCenter(this, "¥n クライアントを接続してください ¥n ");
112 :                break;
113 :            case EScpPortState.OpenFailure: // ・オープン失敗
114 :                MessageBox.Show(e.name + "オープン/接続失敗", ApName, MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
115 :                break;
116 :            case EScpPortState.SrvSrtFail: // ・サーバ開始失敗
117 :                MessageBox.Show(e.name + "サーバ開始失敗", ApName, MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
118 :                break;
119 :            case EScpPortState.MySlotFail: // ・自スロット生成失敗
120 :                MessageBox.Show("自メールスロット(" + e.name + ")の生成失敗", ApName, MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
121 :                break;
122 :        }
123 :        ShowLabel();
124 :    }
125 :    //----- SCP : パケットデータ受信通知 -----//
126 :    private unsafe void scp_OnRxPacket(object sender, ScpArgRxPacket e)
127 :    {
128 :        if (e.bin[0] == CMD_RSP_START) {
129 :        }
130 :        else if (e.bin[0] == CMD_RSP_STOP) {
131 :        }
132 :        else if (e.bin[0] == CMD_DATA) {
133 :            CMDDATA dat = new CMDDATA();
134 :            CMDDATA *p = &dat;
135 :            fixed (byte *q = &e.bin[0]) {
136 :                SAjrBin.MemCopy(p, q, sizeof(CMDDATA));
137 :            }
138 :            tch.PutData(dat.vec.x, dat.vec.y, dat.vec.z);
139 :        }
140 :    }
141 :    //----- SCP : パケット外テキスト受信通知 -----//

```

```

142 : private void scp_OnRxNoPkt(object sender, ScpArgRxNoPkt e)
143 : {
144 :     m_FormCmd.ShowText(e.text);
145 : }
146 : //----- 1文字送信 -----//
147 : public void SendChar(char c)
148 : {
149 :     scp.SendChar(c);
150 : }
151 : //----- 各ラベル表示 -----//
152 : private void ShowLabel()
153 : {
154 :     switch (scp.PortKind) {
155 :         case ESelPort.SEL_COMPORT: // ●COMポート
156 :             int pno = scp.PortNo;
157 :             if (scp.IsOpened) btnOpenClose.Text = "COM " + pno.ToString() + " 切断";
158 :             else btnOpenClose.Text = "COM " + pno.ToString() + " 接続";
159 :             break;
160 :         case ESelPort.SEL_MAILSLLOT: // ●メールスロット
161 :             if (scp.IsOpened) btnOpenClose.Text = "スロット 切断";
162 :             else btnOpenClose.Text = "スロット 接続";
163 :             break;
164 :         case ESelPort.SEL_CLIENT: // ●ソケット (クライアント)
165 :             if (scp.IsOpened) btnOpenClose.Text = "サーバから切断";
166 :             else btnOpenClose.Text = "サーバへ接続";
167 :             break;
168 :         case ESelPort.SEL_SERVER: // ●ソケット (サーバ)
169 :             if (scp.IsServerStarted) btnOpenClose.Text = "サーバ停止";
170 :             else btnOpenClose.Text = "サーバ起動";
171 :             break;
172 :     }
173 : }
174 : }
175 : }

```


Sil_SerialComPort5A(FormCmd)

```

1 : using System;
2 : using System.Collections.Generic;
3 : using System.ComponentModel;
4 : using System.Data;
5 : using System.Drawing;
6 : using System.Linq;
7 : using System.Text;
8 : using System.Security.Permissions;
9 : using System.Windows.Forms;
10 :
11 : namespace Sil_SerialComPort5A
12 : {
13 :     public partial class FormCmd : Form
14 :     {
15 :         bool m_fFirst = true;
16 :
17 :         public FormCmd()
18 :         {
19 :             InitializeComponent();
20 :         }
21 :         // 起動時初期設定
22 :         private void FormCmd_Load(object sender, EventArgs e)
23 :         {
24 :             // 初期メッセージ
25 :             vth.PutText("相手(Sil_SerialComPort5B)と通信を行います。¥n¥n" +
26 :                 "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" +
27 :                 "「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" +
28 :                 "このウィ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" +
29 :                 "    ・DATE  -- 今日の日付表示¥n" +
30 :                 "    ・TIME  -- 現在時刻表示¥n");
31 :         }
32 :         private void FormCmd_Activated(object sender, EventArgs e)
33 :         {
34 :             // コマンドコンソールをフォーカス
35 :             vth.Focus();
36 :         }
37 :         //----- VTH : キー入力通知 -----//
38 :         private void vth_OnNtcKeyIn(object sender, CAJrCustCtrl.VthArgNtcKeyIn e)
39 :         {
40 :             if (m_fFirst) {
41 :                 vth.Purge();
42 :                 m_fFirst = false;
43 :             }
44 :             ((Form1)Application.OpenForms["Form1"]).SendChar((char)e.key);
45 :         }
46 :         //----- 受信テキスト表示 -----//
47 :         public void ShowText(string text)
48 :         {
49 :             vth.PutText(text);
50 :         }
51 :         //----- フォーカス -----//
52 :         public new void Focus()
53 :         {
54 :             vth.Focus();
55 :         }
56 :         //----- フォームの終了を禁止 -----//
57 :         protected override CreateParams CreateParams
58 :         {
59 :             [SecurityPermission(SecurityAction.Demand, Flags = SecurityPermissionFlag.UnmanagedCode)]
60 :             get {
61 :                 const int CS_NOCLOSE = 0x200;
62 :                 CreateParams cp = base.CreateParams;
63 :                 cp.ClassStyle = cp.ClassStyle | CS_NOCLOSE;
64 :                 return cp;
65 :             }
66 :         }
67 :     }
68 : }
69 : }

```

Sil_SerialComPort5B

```

1 : using System;
2 : using System.Collections.Generic;
3 : using System.ComponentModel;
4 : using System.Data;
5 : using System.Drawing;
6 : using System.Linq;
7 : using System.Text;
8 : using System.Windows.Forms;
9 : using System.Runtime.InteropServices;
10 : using CAjrCustCtrl;
11 :
12 : namespace Sil_SerialComPort5B
13 : {
14 :     public partial class Form1 : Form
15 :     {
16 :         string ApName = "Sil_SerialComPort5B";
17 :         // 通信パケットコマンド
18 :         byte CMD_REQ_START = 0x01; // グラフデータ送信開始要求
19 :         byte CMD_RSP_START = 0x11; // " 応答
20 :         byte CMD_REQ_STOP = 0x02; // グラフデータ送信停止要求
21 :         byte CMD_RSP_STOP = 0x12; // " 応答
22 :         byte CMD_DATA = 0x40; // グラフデータ
23 :         // グラフデータ形式
24 :         [StructLayout(LayoutKind.Sequential, Pack=1)]
25 :         struct CMDDATA {
26 :             public byte cmd; // コマンドコード
27 :             public byte f1, f2, f3; // -
28 :             public AJC3DVEC vec; // グラフデータ
29 :         }
30 :
31 :         System.Threading.Mutex m_Mut5A;
32 :         System.Threading.Mutex m_Mut5B = new System.Threading.Mutex(true, "Sil_SerialComPort5B");
33 :
34 :         public Form1()
35 :         {
36 :             InitializeComponent();
37 :         }
38 :         //----- 起動時初期設定 -----//
39 :         private void Form1_Load(object sender, EventArgs e)
40 :         {
41 :             this.Text = this.Text + (IntPtr.Size == 4 ? " (WIN32)" : " (WIN64)");
42 :             // サイズ変更禁止
43 :             this.FormBorderStyle = FormBorderStyle.FixedSingle;
44 :             // S C P 初期化
45 :             scp.Init();
46 :             // 相手プログラム起動
47 :             bool fCreateNew;
48 :             m_Mut5A = new System.Threading.Mutex(false, "Sil_SerialComPort5A", out fCreateNew);
49 :             m_Mut5A.Close();
50 :             if (fCreateNew) {
51 :                 System.Diagnostics.Process.Start("Sil_SerialComPort5A.exe");
52 :             }
53 :             ShowLabel();
54 :         }
55 :         //----- ポート設定ボタン -----//
56 :         private void btnSetPort_Click(object sender, EventArgs e)
57 :         {
58 :             scp.SetParamByDialog(this.Handle);
59 :             ShowLabel();
60 :         }
61 :         //----- OPEN / CLOSE ボタン -----//
62 :         private void btnOpenClose_Click(object sender, EventArgs e)
63 :         {
64 :             if (scp.IsOpened || scp.IsServerStarted) scp.Close();
65 :             else scp.Open();
66 :             ShowLabel();
67 :         }
68 :         //----- SCP : シリアルポート状態通知 -----//
69 :         private void scp_OnPortState(object sender, CAjrCustCtrl.ScpArgPortState e)
70 :         {
71 :             SAjrTip.Hide();
72 :             switch (e.state) {
73 :                 case EScpPortState.SrvStart: // ・サーバ開始
74 :                     SAjrTip.ShowCenter(this, "¥n クライアントを接続してください ¥n ");
75 :                     break;
76 :                 case EScpPortState.OpenFailure: // ・オープン失敗
77 :                     MessageBox.Show(e.name + "オープン／接続失敗", ApName, MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
78 :                     break;
79 :                 case EScpPortState.SrvSrtFail: // ・サーバ開始失敗
80 :                     MessageBox.Show(e.name + "サーバ開始失敗", ApName, MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
81 :                     break;
82 :                 case EScpPortState.MySlotFail: // ・自スロット生成失敗
83 :                     MessageBox.Show("自メールスロット(" + e.name + ")の生成失敗", ApName, MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
84 :                     break;
85 :             }
86 :             ShowLabel();
87 :         }

```

```

88 : //----- SCP : パケットデータ受信通知 -----//
89 : private unsafe void scp_OnRxPacket(object sender, ScpArgRxPacket e)
90 : {
91 :     if (e.bin[0] == CMD_REQ_START) {
92 :         fixed (byte* p = &CMD_RSP_START) {
93 :             scp.SendPacket(p, 1);
94 :             tim.Enabled = true;
95 :         }
96 :     }
97 :     else if (e.bin[0] == CMD_REQ_STOP) {
98 :         fixed (byte* p = &CMD_RSP_STOP) {
99 :             scp.SendPacket(p, 1);
100 :             tim.Enabled = false;
101 :         }
102 :     }
103 : }
104 : //----- SCP : パケット外データ受信通知 -----//
105 : private void scp_OnRxNoPkt(object sender, CAjrCustCtrl.ScpArgRxNoPkt e)
106 : {
107 :     scp.SendText(e.text);
108 : }
109 : //----- テキスト受信通知 -----//
110 : private void scp_OnRxText(object sender, CAjrCustCtrl.ScpArgRxText e)
111 : {
112 :     DateTime dtNow = DateTime.Now;
113 :     string strDate = dtNow.ToString("yyyy/MM/dd") + "¥n";
114 :     string strTime = dtNow.ToString("hh:mm:ss") + "¥n";
115 :     scp.SendText("¥n¥x1B[35m");
116 :     if (e.text.Equals("DATE", StringComparison.OrdinalIgnoreCase)) {
117 :         scp.SendText(strDate);
118 :     }
119 :     else if (e.text.Equals("TIME", StringComparison.OrdinalIgnoreCase)) {
120 :         scp.SendText(strTime);
121 :     }
122 :     else {
123 :         scp.SendText("*** ¥"" + e.text + "¥" は、不正なコマンドです ***¥n");
124 :     }
125 :     scp.SendText("¥x1B[0m");
126 : }
127 : //----- タイマ -----//
128 : private unsafe void tim_Tick(object sender, EventArgs e)
129 : {
130 :     CMDDATA dat = new CMDDATA();
131 :     dat.cmd = CMD_DATA;
132 :     dat.vec = spd.Calc();
133 :     scp.SendPacket(&dat, sizeof(CMDDATA));
134 : }
135 :
136 : //----- 各ラベル表示 -----//
137 : private void ShowLabel()
138 : {
139 :     switch (scp.PortKind) {
140 :         case ESelPort.SEL_COMPORT: // ●COMポート
141 :             int pno = scp.PortNo;
142 :             if (scp.IsOpened) btnOpenClose.Text = "COM " + pno.ToString() + " 切断";
143 :             else btnOpenClose.Text = "COM " + pno.ToString() + " 接続";
144 :             break;
145 :         case ESelPort.SEL_MAILSLLOT: // ●メールスロット
146 :             if (scp.IsOpened) btnOpenClose.Text = "スロット 切断";
147 :             else btnOpenClose.Text = "スロット 接続";
148 :             break;
149 :         case ESelPort.SEL_CLIENT: // ●ソケット (クライアント)
150 :             if (scp.IsOpened) btnOpenClose.Text = "サーバから切断";
151 :             else btnOpenClose.Text = "サーバへ接続";
152 :             break;
153 :         case ESelPort.SEL_SERVER: // ●ソケット (サーバ)
154 :             if (scp.IsServerStarted) btnOpenClose.Text = "サーバ停止";
155 :             else btnOpenClose.Text = "サーバ起動";
156 :             break;
157 :     }
158 : }
159 : }
160 : }

```

12. ソケット (TCP/IP) サーバ機能 (CAjrSockServer.dll)

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。

12.1. 機能概要

12.1.1. 受信データの通知形式と送受信文字コード

本書冒頭の「受信データの通知形式と送受信文字コード」章を参照してください。

12.1.2. イベントの通知方法

イベントの通知方法は、_SsvMode プロパティにより、以下の 2 つから選択できます。

- ・本コントロールがイベントを発生する (_SsvMode= ESsvMode.NotificationByEvent, デフォルト)
- ・ユーザが、その場でイベントの発生を待ち受ける (_SsvMode= ESsvMode.WaitingForEvent)

_SsvMode プロパティは、デザインモード時に指定するようにしてください。
コンソールアプリ等で、デザインモードで指定できない場合は、プログラムの開始時に設定してください。

_SsvMode プロパティを **ESsvMode. NotificationByEvent** に設定した場合は、本コントロールがイベントを発生し、OnXXXXX() イベントで事象をユーザに通知します。デフォルトでは、このモードに設定されています。
このモードは、Windows フォームアプリケーションで使用可能です。

_SsvMode プロパティを **ESsvMode. WaitingForEvent** に設定した場合は、WaitEvent() メソッドによりユーザがイベントの発生を待ち受けます。
このモードは、コンソール・アプリケーションで使います。

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、タイムアウト時間を指定して、その場で応答データの着信を待つことが可能です。

12.2. 構造体／列挙体 (定数)

12.2.1. 実行モード

```
public enum ESsvMode : int
{
    NotificationByEvent = 0,    // イベントを通知する (OnXXXXイベントで通知する)
    WaitingForEvent     = 1,    // イベントを待ち受ける
}
```

12.2.2. チャンクデータの通知モード

```
public enum ESsvChunkMode : int
{
    BinaryData    = 0x01,    // バイナリ・チャンク
    TextData      = 0x02,    // テキスト・チャンク
    Both          = 0x03,    // 両方
}
```

12.2.3. イベントコード

```
public enum ESsvEvt : int
{
    EV_RXTEXT      = 0x00000001,    // テキストデータ通知
    EV_RXESC       = 0x00000002,    // E S C コードデータ通知
    EV_RXPKT       = 0x00000004,    // パケットデータ通知
    EV_RXCTRL      = 0x00000008,    // 制御コード通知
    EV_RXNOPKT     = 0x00000010,    // パケット外テキストデータ

    EV_RXCHUNK     = 0x00000100,    // チャンクデータ受信通知
    EV_INVCHUNK    = 0x00000200,    // 不正チャンクテキスト受信通知
    EV_TXEMPTY     = 0x00000400,    // 送信完了通知
    EV_CONNECT     = 0x00000800,    // 接続通知
    EV_DISCONNECT  = 0x00001000,    // 切断通知
    EV_START       = 0x00002000,    // サーバ開始通知
    EV_STOP        = 0x00004000,    // サーバ終了通知
    EV_RXERR       = 0x00010000,    // 受信エラー通知
    EV_TXERR       = 0x00020000,    // 送信エラー通知
    EV_ERR         = 0x08000000,    // エラー通知

    EV_SSEP        = (EV_RXTEXT | EV_RXESC | EV_RXCTRL | EV_RXPKT),
    EV_COMM        = (EV_RXCHUNK | EV_INVCHUNK | EV_TXEMPTY),
    EV_GENERAL     = (EV_CONNECT | EV_DISCONNECT | EV_START | EV_STOP),
    EV_ERRS        = (EV_RXERR | EV_TXERR | EV_ERR),
    EV_CLIENT      = (EV_SSEP | EV_COMM | EV_CONNECT | EV_DISCONNECT | EV_RXERR | EV_TXERR),
    EV_DEFAULT     = (EV_SSEP | EV_COMM | EV_GENERAL | EV_ERRS),
    EV_ALL         = (EV_SSEP | EV_COMM | EV_GENERAL | EV_ERRS | EV_RXNOPKT)
}
```

12.2.4. エラーコード

```

public enum ESsvErrorCode {
    ERR_CREEVT      = 10,    // 終了通知用イベントオブジェクト生成失敗
    ERR_SOCKET      = 20,    // 接続要求待機用ソケット生成失敗(socket)
    ERR_BIND        = 30,    // バインド失敗 (bindt)
    ERR_LISTEN      = 40,    // LISTEN 失敗(listen)
    ERR_ACCEPT      = 50,    // ACCEPT 失敗 (accept)
    ERR_ADDRINFO    = 60,    // アドレス情報取得失敗 (getaddrinfo)
    ERR_SOCKOPT     = 70,    // ソケットオプション設定失敗(setsockopt)
    ERR_THREADPOWCTRL = 80,  // 電源制御スレッド開始失敗
    ERR_THREADLISTEN = 90,  // 接続待機スレッド開始失敗
    ERR_THREADCLIENT = 100,  // クライアント通信スレッド開始失敗
    ERR_CRESSEP     = 110,   // ストリーム分離インスタンス生成失敗
    ERR_CREMBXTXD   = 120,   // 送信データ用メールボックス生成失敗
    ERR_TIMEOUT     = 130,   // サーバ終了タイムアウト
    ERR_CRETHREADWORK = 140, // クライアントスレッド用ワーク確保失敗
    ERR_CONNOVER    = 150,   // クライアント接続数オーバー
    ERR_CREIXMAP    = 160,   // インデクス割り当て用マップテーブル生成失敗
}

```

12.2.5. 受信テキストの文字コード

```

public enum ESsvRxTextCode : int
{
    SJIS      = 1,    // S - J I S
    EUC       = 2,    // E U C
    UTF8      = 3,    // U T F - 8
    AUTO      = 9,    // 自動認識
}

```

12.2.6. 送信テキストの文字コード

```

public enum ESsvTxTextCode : int
{
    SJIS      = 1,    // S - J I S
    EUC       = 2,    // E U C
    UTF8      = 3,    // U T F - 8
    AUTO      = 9,    // 受信テキストコードに合わせる
}

```

12.2.7. アドレスファミリ

```

public enum ESsvFamily : int
{
    INET      = 0x02,    // IPV4
    INET6     = 0x17,    // IPV6
}

```

12.3. プロパティ

TCP/IP サーバ機能のプロパティ一覧を示します。
プロパティは、Start() メソッドを実行する前に設定してください。

#	名称	タイプ	内容	デフォルト
1	_ScpMode	ESsvMode	実行モード (Start() メソッド実行前に設定すること) (※1) <u>NotificationByEvent</u> : イベントの発生を通知で受ける <u>WaitingForEvent</u> : イベントの発生をその場で待ち受ける	NotificationByEvent
2	SsvOption	ESsvServOpt	オプション ・ <u>NOOPT</u> - オプション無し ・ <u>SUP_SLEEP</u> - スリープ抑止 ・ <u>SUP_DISPOFF</u> - ディスプレイ OFF 抑止 ・ <u>SUP_ALL</u> - スリープ&ディスプレイ OFF 抑止	
3	ChunkMode	ESsvChunkMode	チャンクデータの通知モード ・ <u>BinaryData</u> : バイナリデータとして扱う ・ <u>TextData</u> : テキストデータとして扱う ・ <u>Both</u> : 両方	BinaryData
4	STX	int	パケットデータの先頭識別コード	0x02
5	ETX	int	パケットデータの終端識別コード	0x03
6	DLE	int	パケットデータの投下制御コード	0x10
10	PktTimeout	int	パケットデータ受信タイムアウト時間[ms]	3000[ms]
11	RxTextCode	ESsvRxTextCode	受信テキストコード(SJIS / EUC / UTF8 / AUTO(自動判別)) (※2)	SJIS
12	TxTextCode	ESsvTxTextCode	送信テキストコード(SJIS / EUC / UTF8 / AUTO(受信テキストコードと同じ))	SJIS
13	Clients	int	接続中のクライアント数 (読み出し専用)	-
14	IsStarted	bool	サーバ起動状態 (読み出し専用)	

※1 : 通常、コンソールアプリの場合、WaitingForEvent を設定し、WaitEvent() メソッドでイベントを待ち受けてください。

※2 : 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、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。

12.4. メソッド

TCP/IP サーバ機能のメソッド一覧を以下に示します。

サーバ機能メソッド

#	メソッド名	内容	備考
1	Start	サーバ開始	
2	Stop	サーバ停止	
3	EnumClients	クライアント列挙	
4	SetEvtMask	イベントマスク設定 (コンソールアプリ用)	
5	WaitEvent	イベント待ち	
6	Delete	ソケットサーバ・インスタンス消去	コンソールアプリ用

クライアント機能メソッド

7	SendByte	1 バイト送信	
8	SendChar	1 文字送信	
9	SendText	テキストデータ送信	
10	SendBinary	バイナリデータ送信	
11	SendPacket	パケット送信	
12	PurgeRx	受信済データ破棄	
13	PurgeTx	送信待ちデータ破棄	
14	Purge	送受信データ破棄	
15	SetClientData	クライアントにデータを関連付ける	
16	GetClientData	クライアントに関連付けられたデータ取得	
17	GetSeqNo	クライアント接続順序番号取得	
18	GetIndex	クライアントのインデクス値取得	
19	GetIpAddrStr	クライアントの I P アドレス文字列取得	
20	GetThreadId	クライアントスレッドのスレッド I D 取得	
21	GetActualRxTextCode	実際の受信テキストコード取得	
22	GetActualTxTextCode	実際の送信テキストコード取得	
23	Disconnect	クライアント切断	

※クライアント機能メソッドは、回線切断イベント (OnDisconnect) 発生以降は、実行できません。

12.4.1. サーバ開始設定 (Start)

形 式 : void Start(uint PortNo);
 void Start(uint PortNo, int MaxClients);
 void Start(uint PortNo, ESsvFamily AddressFamily, int MaxClients);

引 数 : PortNo - TCP/IP ポート番号
 AddressFamily - アドレスファミリー (INET:IPV4, INET6:IPV6)・・・省略時は、INET を仮定
 MaxClients - 接続可能な最大クライアント数 ・・・・・・・省略時は、10 を仮定

説 明 : ソケットサーバ・コントロールを開始します。
 このメソッドを実行すると、サーバ開始通知イベントが発生します。

戻り値 : なし

12.4.2. サーバ停止(Stop)

形 式 : void Stop();
 void Stop(int msTimeout);

引 数 : msTimeout - サーバ終了待ち時間[ms] (省略時は、10000 (10 秒)を仮定)

説 明 : ソケットサーバ・コントロールを停止します。
 このメソッドを実行すると、全接続済クライアントについて切断通知イベント後、サーバ終了通知イベントが発生します。

戻り値 : なし

12.4.3. クライアント列挙(EnumClients)

形 式 : int EnumClients(IntPtr param);

引 数 : param - クライアント列挙通知イベントへのパラメタ

説 明 : 接続中の全クライアントを列挙します。
 このメソッドを実行すると、接続中のクライアントについて、クライアント列挙通知イベントが発生します。

戻り値 : 接続中のクライアント数

12.4.4. イベントマスク設定(SetEvtMask)

形 式 : void SetEvtMask(ESsvEvt evt);

引 数 : evt - イベントマスク (EV_XXXXX の合成値)

説 明 : _SsvMode プロパティ = WaitingForEvent (コンソールアプリ) の場合のイベントマスクを設定します。
 使用するイベントを EV_XXXXX の合成値で指定し、Start() メソッドよりも前に実行します。

戻り値 : なし

備 考 : _SsvMode プロパティ = NotificationByEvent (Windows フォームアプリ) の場合は、使用するイベントをイベントハンドラ (OnXXXXX) で指定する為、イベントマスクを指定する必要はありません。

12.4.5. イベント待ち(WaitEvent)

形 式 : ESsvEvt WaitEvent(out Object EvtData, int msWaitTime, ref IntPtr hClient)

引 数 : EvtData - 受信データ (出力)
 msWaitTime - イベント待ち時間
 hClient - クライアントハンドル (出力)
 ClientInfo - クライアント情報 (出力)

説 明 : イベントの発生を待ちます。

このメソッドを実行する場合は、_SsvMode (実行モード) プロパティを WaitingForEvent に設定していなければなりません。

hClient にはクライアントハンドルが格納されます。

EvtData には、以下の情報が格納されます。

イベント	イベントデータ (EvtData)		備考
	タイプ	内容	
EV_NOEVT	-	null	イベント待ちタイムアウト
EV_RXTEXT	String	受信テキストデータ	TAB (0x09) を含む
EV_RXESC	String	受信 ESC シーケンス	
EV_RXCTRL	char	受信制御コード	TAB (0x09) を除く
EV_RXPKT	Byte[]	受信パケットデータ	バイナリデータ
EV_RXNOPKT	String	受信パケット外テキスト	
EV_RXCHUNK	String	受信チャンクデータ (テキスト)	ChunkMode=TextData/Both
	Byte[]	" (バイナリ)	ChunkMode= BinaryData/Both
EV_INVCHUNK	Byte[]	受信不正チャンクテキスト	ChunkMode=TextData/Both
EV_TXEMPTY	-	null	
EV_CONNECT	-	null	
EV_DISCONNECT	object	null / クライアントに関連付けられたデータ	※ 1
EV_RXERR	int	エラー発生種別 0:WSARecv 1:GetOverlappedResult	
EV_TXERR	int	エラー発生種別 0:WSASend 1: GetOverlappedResult	
EV_START	-	null	hClient=0(無効)が設定される
EV_STOP	-	null	
EV_ERR	ESsvErrorCode	エラーコード (ERR_XXXX)	

※ 1 : EV_DISCONNECT (切断通知) の場合は、SetClientData() でクライアントに関連付けられたデータがあれば、EvtData に当該関連付けデータが設定されます。クライアントに関連付けられたデータが無い場合は、null が設定されます。
 尚、EV_DISCONNECT イベント時は、GetClientData() を実行できません。

イベント発生待ちタイムアウトの場合は、EV_NOEVENT を返し、EvtData=null が設定されます。

戻り値 : 発生したイベントのイベントコード (EV_XXXX)

備 考 : チャンクデータ受信通知 (EV_RXCHUNK) で、テキストチャンクと、バイナリチャンクデータの両方を扱う (ChunkMode= Both) 場合は、以下の条件でテキストチャンク/バイナリチャンクを特定してください。

- EvtData.GetType() == typeof(byte[]) -- バイナリチャンクデータ
- EvtData.GetType() == typeof(string) -- テキストチャンクデータ

12.4.6. ソケットサーバ・インスタンス消去(Delete)

形 式 : void Delete()

引 数 : なし

説 明 : ソケットサーバ・インスタンスを消去します。
 接続中の全クライアントを全て切断し、ソケットサーバを停止後にインスタンスを消去します。
 このメソッドは、コンソールアプリの場合に実行してください。
 Windows フォームアプリ等では実行しないでください。(自動的に実行されます)

戻り値 : なし

備 考 : Stop() メソッドを実行済の場合は、既にソケットサーバは停止している為、インスタンスの消去だけが行われます。

12.4.7. 1バイト送信(SendByte)

形 式 : void SendChar(IntPtr hClient, byte ByteData);

引 数 : hClient - クライアントハンドル
 Character - 送信する文字データ

説 明 : hClient で指定されたクライアントへ1文字/1バイト送信します。
 クライアントへByteData で指定された1バイトを送信します。
 このメソッドは送信用スプールバッファにデータを格納するだけであり、送信完了を待たずに、すぐに制御を返します。

戻り値 : なし

例 外 : ArgumentException - 不正なクライアントハンドル

12.4.8. 1文字送信(SendChar)

形 式 : void SendChar(IntPtr hClient, char Character);

引 数 : hClient - クライアントハンドル
 Character - 送信する文字データ

説 明 : hClient で指定されたクライアントへ1文字/1バイト送信します。
 文字データ(Character)を TxTextCode プロパティで指定された文字コードに変換して送信します。
 このメソッドは送信用スプールバッファにデータを格納するだけであり、送信完了を待たずに、すぐに制御を返します。

戻り値 : なし

例 外 : ArgumentException - 不正なクライアントハンドル

12.4.9. テキストデータ送信(SendChar)

形 式 : void SendText(IntPtr hClient, string text);

引 数 : hClient - クライアントハンドル
 text - 送信するテキストデータ

説 明 : hClient で指定されたクライアントへテキストデータを送信します。
 テキストデータを TxTextCode プロパティで指定された文字コードに変換して送信します。
 このメソッドは送信用スプールバッファにデータを格納するだけであり、送信完了を待たずに、すぐに制御を返します。

戻り値 : なし

例 外 : ArgumentException - 不正なクライアントハンドル

12.4.10. バイナリデータ送信(SendBinary)

形 式 : void SendBinary(IntPtr hClient, Byte[] bin);
 void SendBinary(IntPtr hClient, IntPtr p, int len);
 unsafe void SendBinary(IntPtr hClient, void *p, int len);

引 数 : hClient - クライアントハンドル
 bin - 送信するバイナリデータのバイト配列
 p - 送信するバイナリデータのアドレス
 len - 送信するバイナリデータのバイト数

説 明 : hClient で指定されたクライアントへバイナリデータを送信します。
 このメソッドは送信用スプールバッファにデータを格納するだけであり、送信完了を待たずに、すぐに制御を返します。

戻り値 : なし

例 外 : ArgumentException - 不正なクライアントハンドル

12.4.11. パケット送信(SendPacket)

形 式 : int SendPacket(IntPtr hClient, Byte[] bin);
 int SendPacket(IntPtr hClient, IntPtr p, int len);
 unsafe int SendPacket(IntPtr hClient, void *p, int len);

引 数 : hClient - クライアントハンドル
 bin - 送信するバイナリデータのバイト配列 (空パケット (DLE, STX, DLE, ETX の 4 バイト)送信時は null)
 p - 送信するバイナリデータのアドレス (空パケット (DLE, STX, DLE, ETX の 4 バイト)送信時は 0)
 len - 送信するバイナリデータのバイト数 (空パケット (DLE, STX, DLE, ETX の 4 バイト)送信時は 0)

説 明 : hClient で指定されたクライアントへパケット形式でバイナリデータを送信します。
 つまり、先頭に「DLE・STX」の 2 バイトを、末尾に「DLE・ETX」の 2 バイトを付加し、バイナリデータ中の DLE バイトは、2 つの DLE に変換 (DLE → DLE・DLE) して伝送します。
 STX, ETX, DLE の実際のコード値はプロパティにて設定可能です。(規定値は、STX=0x02, ETX=0x03, DLE=0x10)
 このメソッドは送信用スプールバッファにデータを格納するだけであり、送信完了を待たずに、すぐに制御を返します。

戻り値 : 4 ~ - 正常 ((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 - エラー

例 外 : ArgumentException - 不正なクライアントハンドル

12.4.12. 受信済データ破棄(PurgeRx)

形 式 : void PurgeRx(IntPtr hClient);

引 数 : hClient - クライアントハンドル

説 明 : hClient で指定されたクライアントからの受信済みデータを破棄します。

戻り値 : なし

例 外 : ArgumentException - 不正なクライアントハンドル

12.4.13. 送信待ちデータ破棄(PurgeTx)

形 式 : void PurgeTx(IntPtr hClient);

引 数 : hClient - クライアントハンドル

説 明 : hClient で指定されたクライアントへの送信待ちデータを破棄します。

戻り値 : なし

例 外 : ArgumentException - 不正なクライアントハンドル

12.4.14. 送受信データ破棄(Purge)

形 式 : void Purge(IntPtr hClient);

引 数 : hClient - クライアントハンドル

説 明 : hClient で指定されたクライアントからの受信済データと、送信待ちデータを破棄します。

戻り値 : なし

例 外 : ArgumentException - 不正なクライアントハンドル

12.4.15. クライアントにデータを関連付ける(SetClientData)

形 式 : void SetClientData(IntPtr hClient, object obj);

引 数 : hClient - クライアントハンドル
obj - クライアントに関連付けるデータ

説 明 : hClient で指定されたクライアントへデータを関連付けます

戻り値 : なし

例 外 : ArgumentException - 不正なクライアントハンドル

12.4.16. クライアントに関連付けられたデータ取得(GetClientData)

形 式 : void GetClientData(IntPtr hClient, ref object obj);

引 数 : hClient - クライアントハンドル
obj - クライアントに関連付けられたデータを格納するオブジェクト

説 明 : hClient で指定されたクライアントへ関連付けられているデータを取得します

戻り値 : なし

注 意 : WaitEvent()メソッドでイベントを待つ場合、切断通知(EV_DISCONNECT)時は、本メソッドを実行できません。
WaitEvent()メソッドでイベントを待つ場合は、cif 引数の ClientData メンバにクライアントに関連付けられたデータが設定されています。

例 外 : ArgumentException - 不正なクライアントハンドル／クライアントに関連付けられたデータ無し

12.4.17. クライアント接続順序番号取得(GetSeqNo)

形 式 : int GetSeqNo (IntPtr hClient);

引 数 : hClient - クライアントハンドル

説 明 : hClient で指定されたクライアントの接続順序番号 (サーバ開始後、何番目に接続したか) を取得します。

戻り値 : クライアントの接続順序番号 (1 ~)

例 外 : ArgumentException - 不正なクライアントハンドル

12.4.18. クライアントのインデクス値取得(GetSeqNo)

形 式 : int GetIndex (IntPtr hClient);

引 数 : hClient - クライアントハンドル

説 明 : hClient で指定された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 : 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)

例 外 : ArgumentException - 不正なクライアントハンドル

12.4.19. クライアントの I P アドレス文字列取得(GetIpAddrStr)

形 式 : string GetIpAddrStr(IntPtr hClient);

引 数 : hClient - クライアントハンドル

説 明 : hClient で指定されたクライアントの I P アドレス (文字列) を取得します。

戻り値 : なし

注 意 : WaitEvent() メソッドでイベントを待つ場合、切断通知 (EV_DISCONNECT) 時は、本メソッドを実行できません。
WaitEvent() メソッドでイベントを待つ場合は、cif 引数の IpAddr メンバに I P アドレスが設定されています。

例 外 : ArgumentException - 不正なクライアントハンドル

12.4.20. クライアントスレッドのスレッド I D 取得(GetThreadId)

形 式 : uint GetThreadId(IntPtr hClient);

引 数 : hClient - クライアントハンドル

説 明 : hClient で指定されたクライアントのスレッド I D を取得します。

戻り値 : なし

例 外 : ArgumentException - 不正なクライアントハンドル

12.4.21. 実際の受信テキストコード取得(GetActualRxTextCode)

形 式 : ESsvRxTextCode GetActualRxTextCode(IntPtr hClient);

引 数 : hClient - クライアントハンドル

説 明 : hClient で指定されたクライアントの受信テキストコードを取得します。
RxTextCode=AUTO が設定されている場合、実際に受信データから判別したテキストコードを返します。

戻り値 : 受信テキストコード(SJIS / EUC / UTF8)

例 外 : ArgumentException - 不正なクライアントハンドル

12.4.22. 実際の送信テキストコード取得(GetActualTxTextCode)

形 式 : ESsvTxTextCode GetActualTxTextCode(IntPtr hClient);

引 数 : hClient - クライアントハンドル

説 明 : hClient で指定されたクライアントの送信テキストコードを取得します。
TxTextCode=AUTO が設定されている場合、受信テキストコードを返します。

戻り値 : 送信テキストコード(SJIS / EUC / UTF8)

例 外 : ArgumentException - 不正なクライアントハンドル

12.4.23. クライアントを切断(Disconnect)

形 式 : void Disconnect(IntPtr hClient);

引 数 : hClient - クライアントハンドル

説 明 : hClient で指定されたクライアントとの接続を切断します。

戻り値 : なし

例 外 : ArgumentException - 不正なクライアントハンドル

12.5. イベント

TCP/IP サーバ機能のイベント一覧を以下に示します。

#	イベント名	内容	備考
1	OnStart	サーバ開始通知	
2	OnStop	サーバ停止通知	
3	OnEnumClients	クライアント列挙通知	
4	OnRxText	テキスト受信通知	
5	OnRxEsc	E S C 受信通知	
6	OnRxCtrl	制御コード受信通知	
7	OnRxPacket	パケット受信通知	
8	OnRxNoPkt	パケット外テキスト受信通知	
9	OnTxEmpty	送信完了通知	
10	OnRxChunkTxt	テキストチャンク受信通知	
11	OnRxChunkBin	バイナリチャンク受信通知	
12	OnRxInvChunk	不正チャンクテキスト受信通知	
13	OnConnect	接続通知	
14	OnDisconnect	切断通知	
15	OnRecvError	受信エラー通知	
16	OnSendError	送信エラー通知	
17	OnGeneralError	その他のエラー通知	

12.5.1. サーバ開始通知 (OnStart)

形 式 : void OnStart (object sender, SsvArgStart e);

パラメタ : なし

説 明 : Start() メソッド実行後、ソケット・サーバが開始したことを通知します。

戻り値 : なし

12.5.2. サーバ停止通知 (OnStop)

形 式 : void OnStop (object sender, SsvArgStop e);

パラメタ : なし

説 明 : Stop() メソッド実行後、ソケット・サーバが停止したことを通知します。
Start() メソッドを実行すると、ソケット・サーバが再開します。

戻り値 : なし

12.5.3. クライアント列挙通知 (OnEnumClients)

形 式 : `bool OnStop (object sender, SsvArgEnumClients e);`

パラメタ : `IntPtr e.hClient` - クライアントハンドル
`IntPtr e.param` - `EnumClients()` メソッド実行時の `param` 引数の内容

説 明 : `EnumClients()` メソッド実行後、接続中のクライアントを通知します。
`true` を返すと (全てのクライアントを通知するまで) 次のクライアントを通知すべく、本イベントが再び発生します。
`false` を返すと、クライアントの通知を停止します。

戻り値 : なし

12.5.4. テキスト受信通知 (OnRxText)

形 式 : `void OnRxText (object sender, SsvArgRxText e);`

パラメタ : `IntPtr e.hClient` - クライアントハンドル
`string e.text` - 受信したテキストデータ

説 明 : クライアントから受信したテキストデータを通知します。
テキストデータとは、`TAB (0x09)` を除く制御コードで区切られたデータを意味します。
`TAB (0x09)` はテキストデータ内に含めます。

戻り値 : なし

12.5.5. E S C受信通知 (OnRxEsc)

形 式 : `void OnRxEsc (object sender, SsvArgRxEsc e);`

パラメタ : `IntPtr e.hClient` - クライアントハンドル
`string e.text` - 受信したエスケープシーケンスのテキスト

説 明 : クライアントから受信したエスケープシーケンスのテキストを通知します。
エスケープシーケンスとは、`0x1B` で始まり英字で終わるテキストを意味します。

戻り値 : なし

12.5.6. 制御コード受信通知 (OnRxCtrl)

形 式 : `void OnRxCtrl (object sender, SsvArgRxCtrl e);`

パラメタ : `IntPtr e.hClient` - クライアントハンドル
`char e.ctrl` - 受信した制御コード

説 明 : クライアントから受信した制御コードを通知します。
制御コードとは、`TAB (0x09)` を除く、`0x00~0x1F` と `0x7F` を意味します。

戻り値 : なし

12.5.7. パケット受信通知 (OnRxPacket)

形 式 : void OnRxPacket (object sender, SsvArgRxPacket e);

パラメタ : IntPtr e.hClient - クライアントハンドル
 Byte[] e.bin - パケットデータのアドレス (空パケットの場合は null)

説 明 : パケットデータを受信したことを通知します。
 パケットデータは、デコードされたデータだけをを通知します。
 つまり、先頭の「DLE・STX」と末尾の「DLE・ETX」は除去され、連続する 2 つの DLE を 1 つの DLE に変換したデータが通知されます。
 空のパケット (データ無しで、DLE・STX と DLE・ETX の 4 バイトのみ) の場合は、e.bin = null が設定されます。

戻り値 : なし

12.5.8. パケット外テキスト受信通知 (OnRxNoPkt)

形 式 : void OnRxNoPkt (object sender, SsvArgRxNoPkt e);

パラメタ : IntPtr e.hClient - クライアントハンドル
 string e.text - パケット外テキストデータ

説 明 : 受信したチャンクデータのパケット部分 (STX・DLE～ETX・DLE) を除いた部分のテキストを通知します。
 通知されるデータは、テキスト受信通知 (OnRxText) とほぼ同じになりますが、テキスト受信通知の場合はテキスト終端 (0x0D や 0x0A) 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受信通知 (OnTxNoPkt) ではテキスト終端を待たずにリアルタイムに通知されます。また、パケット外テキストには制御コードも含まれます。
 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。

戻り値 : なし

備 考 : 受信データ中のマルチバイト文字が分断されないように制御されます。

12.5.9. 送信完了通知 (OnTxEmpty)

形 式 : void OnTxEmpty (object sender, SsvArgTxEmpty e);

パラメタ : IntPtr e.hClient - クライアントハンドル

説 明 : 送信が完了したことを通知します。
 つまり、送信待ちとなっているデータが無くなったことを意味します。

戻り値 : なし

12.5.10. テキストチャンク受信通知 (OnRxChunkTxt)

形 式 : void OnRxChunkTxt (object sender, SsvArgRxChunkTxt e);

パラメタ : IntPtr e.hClient - クライアントハンドル
 string e.text - テキストチャンクデータ

説 明 : 受信したテキストチャンクデータを通知します。
 このテキストデータはリアルタイムに通知されます。

戻り値 : なし

備 考 : 受信データ中のマルチバイト文字が分断されないように制御されます。

12.5.11. バイナリチャンク受信通知 (OnRxChunkBin)

形 式 : void OnRxChunkBin (object sender, SsvArgRxChunkBin e);

パラメタ : IntPtr e.hClient - クライアントハンドル
Byte[] e.bin - バイナリチャンクデータ

説 明 : 受信したバイナリチャンクデータを通知します。
このバイナリデータはリアルタイムに通知されます。

戻り値 : なし

12.5.12. 不正チャンクテキスト受信通知 (OnRxInvChunk)

形 式 : void OnRxInvChunk (object sender, SsvArgRxInvChunk e);

パラメタ : IntPtr e.hClient - クライアントハンドル
Byte[] e.bin - 受信したチャンクテキスト (バイナリデータ)

説 明 : テキストチャンクに不正な制御コードが含まれていることを通知します。
チャンクデータの扱いをテキスト (ChunkMode = TextData) としている場合で、チャンクデータに不正な制御コード (0x09~0x0D, 0x1B 以外) が含まれている場合は、テキストチャンク受信通知 (OnRxChunkTxt) ではなく、このイベントでチャンクデータ (バイナリデータ) が通知されます。
このバイナリデータはリアルタイムに通知されます。

戻り値 : なし

12.5.13. 接続通知 (OnConnect)

形 式 : void OnConnect (object sender, SsvArgConnect e);

パラメタ : IntPtr e.hClient - クライアントハンドル

説 明 : クライアントと接続したことを通知します。

戻り値 : なし

12.5.14. 切断通知 (OnDisconnect)

形 式 : void OnDisconnect (object sender, SsvArgDisconnect e);

パラメタ : IntPtr e.hClient - クライアントハンドル

説 明 : クライアントとの回線が切断したことを通知します。

戻り値 : なし

12.5.15. 受信エラー通知 (OnRecvError)

形 式 : void OnRecvError (object sender, SsvArgRecvError e);

パラメタ : IntPtr e.hClient - クライアントハンドル
bool e.overlapped - true : WSARecv() のエラー
false : WSAGetOverlappedResult() のエラー

説 明 : 受信エラーが発生したことを通知します。

戻り値 : なし

12.5.16. 送信エラー通知 (On SendError)

形式 : void On SendError (object sender, SsvArgSendError e);

パラメタ :	IntPtr	e.hClient	- クライアントハンドル
	bool	e.overlapped	- true : WSASend() のエラー false : WSAGetOverlappedResult() のエラー

説明 : 送信エラーが発生したことを通知します。

戻り値 : なし

12.5.17. その他のエラー通知 (OnGeneralError)

形式 : void OnGeneralError (object sender, SsvArgGeneralError e);

パラメタ: ESsvErrorCode e. ErrorCode - エラーコード

説明 : 送受信エラー以外のエラーを通知します。

戻り値 : なし

12.5.18. クライアント列挙通知 (OnEnumClients)

形式 : `bool OnEnumClients (object sender, SsvArgEnumClients e);`

パラメタ :	IntPtr e.hClient	- クライアントハンドル
	IntPtr e.param	- イベントへのパラメタ (EnumClients() で指定した parama 引数の値)

説 明 : 送受信エラー以外のエラーを通知します。
EnumClients() メソッドの実行による、クライアントを通知します。
このイベントは、(true を返す限り) 全クライアントの数だけ発生します。

戻り値 : true - クライアントの通知を継続する
false - クライアントの通知を中止する

12.6. サンプルプログラム

12.6.1. Sil_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 や Sil_SerialComPort1) と対向して通信できます。

メインフォーム



接続クライアント毎のフォーム (クライアントが接続されると表示されます)



メインフォーム (Form1.cs)

```

1 : using System;
2 : using System.Collections.Generic;
3 : using System.ComponentModel;
4 : using System.Data;
5 : using System.Drawing;
6 : using System.Linq;
7 : using System.Text;
8 : using System.Windows.Forms;
9 : using CAjrCustCtrl;
10 :
11 : namespace Sil_SockServer1
12 : {
13 :     public partial class Form1 : Form
14 :     {
15 :         public Form1()
16 :         {
17 :             InitializeComponent();
18 :         }
19 :         // 起動時初期設定
20 :         private void Form1_Load(object sender, EventArgs e)
21 :         {
22 :             // サイズ変更禁止
23 :             this.FormBorderStyle = FormBorderStyle.FixedSingle;
24 :             // ホスト名表示
25 :             lblMyName.Text = Environment.MachineName;
26 :             // ウィンド位置ロード
27 :             SAjrReg.LoadWndPos(this);
28 :             // 設定値ロード
29 :             SAjrReg.LoadAllCtrls(this);
30 :             // テキストエンコード設定
31 :             SetTextEncode();
32 :         }
33 :         // 終了処理
34 :         private void Form1_FormClosed(object sender, FormClosedEventArgs e)
35 :         {
36 :             // ウィンド位置セーブ
37 :             SAjrReg.SaveWndPos(this);
38 :             // 設定値セーブ
39 :             SAjrReg.SaveAllCtrls(this);
40 :         }
41 :         // サーバ開始ボタン
42 :         private void btnStart_Click(object sender, EventArgs e)
43 :         {
44 :             uint pno;
45 :             uint.TryParse(txtPortNo.Text, out pno);
46 :             int mxc;
47 :             int.TryParse(txtMaxClients.Text, out mxc);
48 :             ssv.Start(pno, mxc);
49 :         }
50 :         // サーバ停止ボタン
51 :         private void btnStop_Click(object sender, EventArgs e)
52 :         {
53 :             ssv.Stop();
54 :         }
55 :         // サーバ開始通知
56 :         private void ssv_OnStart(object sender, CAjrCustCtrl.SsvArgStart e)
57 :         {
58 :             btnStart.Enabled = false;
59 :             btnStop.Enabled = true;
60 :         }
61 :         // サーバ停止通知
62 :         private void ssv_OnStop(object sender, CAjrCustCtrl.SsvArgStop e)
63 :         {
64 :             btnStart.Enabled = true;
65 :             btnStop.Enabled = false;
66 :         }
67 :         // 接続通知
68 :         private void ssv_OnConnect(object sender, CAjrCustCtrl.SsvArgConnect e)
69 :         {
70 :             // クライアントフォーム表示
71 :             ClientForm cf = new ClientForm();
72 :             cf.Text = ssv.GetIpAddrStr(e.hClient);
73 :             cf.Show();
74 :             // クライアントにフォームを関連付ける
75 :             ssv.SetClientData(e.hClient, (object)cf);
76 :             // リストボックスにクライアントのIPアドレス追加
77 :             lbxClient.Items.Add(ssv.GetIpAddrStr(e.hClient));

```

```

78 :         // フォームにクライアントハンドル退避
79 :         cf.SaveHClient(ssv, e.hClient);
80 :     }
81 :     // 切断通知
82 :     private void ssv_OnDisconnect(object sender, CAjrCustCtrl.SsvArgDisconnect e)
83 :     {
84 :         // クライアントに関連付けたフォームを取得
85 :         ClientForm cf = GetClientForm(e.hClient);
86 :         // フォームを閉じる
87 :         cf.Close();
88 :         // リストボックスのクライアントの I P アドレス削除
89 :         lbxCliet.Items.Remove(ssv.GetIpAddrStr(e.hClient));
90 :     }
91 :     // バイナリチャンク受信通知
92 :     private void ssv_OnRxChunkBin(object sender, CAjrCustCtrl.SsvArgRxChunkBin e)
93 :     {
94 :         // クライアントに関連付けたフォームを取得
95 :         ClientForm cf = GetClientForm(e.hClient);
96 :         // バイナリチャンクデータ表示
97 :         cf.ShowBinaryChunk(e.bin);
98 :     }
99 :     // テキストチャンク受信通知
100 :    private void ssv_OnRxChunkTxt(object sender, CAjrCustCtrl.SsvArgRxChunkTxt e)
101 :    {
102 :        // クライアントに関連付けたフォームを取得
103 :        ClientForm cf = GetClientForm(e.hClient);
104 :        // テキストチャンクデータ表示
105 :        cf.ShowTextChunk(e.text);
106 :    }
107 :    // 不正テキストチャンク受信通知
108 :    private void ssv_OnRxInvChunk(object sender, CAjrCustCtrl.SsvArgRxInvChunk e)
109 :    {
110 :        // クライアントに関連付けたフォームを取得
111 :        ClientForm cf = GetClientForm(e.hClient);
112 :        // 不正テキストチャンク表示
113 :        cf.ShowInvChunk(e.bin);
114 :    }
115 :    // テキスト受信通知
116 :    private void ssv_OnRxText(object sender, CAjrCustCtrl.SsvArgRxText e)
117 :    {
118 :        // クライアントに関連付けたフォームを取得
119 :        ClientForm cf = GetClientForm(e.hClient);
120 :        // テキスト表示
121 :        cf.ShowText(e.text);
122 :    }
123 :    // 制御コード受信通知
124 :    private void ssv_OnRxCtrl(object sender, CAjrCustCtrl.SsvArgRxCtrl e)
125 :    {
126 :        // クライアントに関連付けたフォームを取得
127 :        ClientForm cf = GetClientForm(e.hClient);
128 :        // 制御コード表示
129 :        cf.ShowCtrl(e.ctrl);
130 :    }
131 :    // エスケープシーケンス受信通知
132 :    private void ssv_OnRxEsc(object sender, CAjrCustCtrl.SsvArgRxEsc e)
133 :    {
134 :        // クライアントに関連付けたフォームを取得
135 :        ClientForm cf = GetClientForm(e.hClient);
136 :        // エスケープシーケンス表示
137 :        cf.ShowEsc(e.esc);
138 :    }
139 :    // パケット受信通知
140 :    private void ssv_OnRxPacket(object sender, CAjrCustCtrl.SsvArgRxPacket e)
141 :    {
142 :        if (e.bin != null) {
143 :            // クライアントに関連付けたフォームを取得
144 :            ClientForm cf = GetClientForm(e.hClient);
145 :            // パケットデータ表示
146 :            cf.ShowPacket(e.bin);
147 :        }
148 :    }
149 :    // パケット外テキスト受信通知
150 :    private void ssv_OnRxNoPkt(object sender, CAjrCustCtrl.SsvArgRxNoPkt e)
151 :    {
152 :        // クライアントに関連付けたフォームを取得
153 :        ClientForm cf = GetClientForm(e.hClient);
154 :        // パケット外テキスト表示
155 :        cf.ShowNoPkt(e.text);
156 :    }
157 :    // 受信エラー通知

```

```

158 : private void ssv_OnRecvError(object sender, CAjrCustCtrl.SsvArgRecvError e)
159 : {
160 :     vthError.PutText(ssv.GetIpAddrStr(e.hClient) + " : ");
161 :     if (e.overlapped) vthError.PutText("受信エラー(GetOverlappedResult)¥n");
162 :     else vthError.PutText("受信エラー(WSARecv)¥n");
163 : }
164 : // 送信エラー通知
165 : private void ssv_OnSendError(object sender, CAjrCustCtrl.SsvArgSendError e)
166 : {
167 :     vthError.PutText(ssv.GetIpAddrStr(e.hClient) + " : ");
168 :     if (e.overlapped) vthError.PutText("送信エラー(GetOverlappedResult)¥n");
169 :     else vthError.PutText("送信エラー(WSASend)¥n");
170 : }
171 : // その他のエラー通知
172 : private void ssv_OnGeneralError(object sender, CAjrCustCtrl.SsvArgGeneralError e)
173 : {
174 :     switch (e.ErrorCode) {
175 :         case ESsvErrorCode.ERR_CREEVT: vthError.PutText("CreateEvent() 失敗¥n" ); break;
176 :         case ESsvErrorCode.ERR_SOCKET: vthError.PutText("ソケット生成失敗¥n" ); break;
177 :         case ESsvErrorCode.ERR_BIND: vthError.PutText("bind 失敗¥n" ); break;
178 :         case ESsvErrorCode.ERR_LISTEN: vthError.PutText("listen 失敗¥n" ); break;
179 :         case ESsvErrorCode.ERR_ACCEPT: vthError.PutText("accept 失敗¥n" ); break;
180 :         case ESsvErrorCode.ERR_ADDRINFO: vthError.PutText("getaddrinfo 失敗¥n" ); break;
181 :         case ESsvErrorCode.ERR_SOCKETOPT: vthError.PutText("setsockopt() 失敗¥n" ); break;
182 :         case ESsvErrorCode.ERR_THREADLISTEN: vthError.PutText("接続待機スレッド開始失敗¥n" ); break;
183 :         case ESsvErrorCode.ERR_THREADCLIENT: vthError.PutText("クライアント通信スレッド開始失敗¥n" ); break;
184 :         case ESsvErrorCode.ERR_CRESSEP: vthError.PutText("ストリーム分離生成失敗¥n" ); break;
185 :         case ESsvErrorCode.ERR_CREMBXTXD: vthError.PutText("送信データ用メールボックス生成失敗¥n" ); break;
186 :         case ESsvErrorCode.ERR_TIMEOUT: vthError.PutText("サーバ終了タイムアウト¥n" ); break;
187 :     }
188 : }
189 : // クライアントに関連付けたフォームを取得
190 : private ClientForm GetClientForm(IntPtr hClient)
191 : {
192 :     object obj;
193 :     ssv.GetClientData(hClient, out obj);
194 :     return (ClientForm)obj;
195 : }
196 : //----- 受信テキストエンコード設定ラジオボタン -----//
197 : private void rbtRxSJis_CheckedChanged(object sender, EventArgs e) {SetTextEncode();}
198 : private void rbtRxUTF8_CheckedChanged(object sender, EventArgs e) {SetTextEncode();}
199 : private void rbtRxEuc_CheckedChanged (object sender, EventArgs e) {SetTextEncode();}
200 : private void rbtRxAuto_CheckedChanged(object sender, EventArgs e) {SetTextEncode();}
201 : //----- 送信テキストエンコード設定ラジオボタン -----//
202 : private void rbtTxSJis_CheckedChanged(object sender, EventArgs e) {SetTextEncode();}
203 : private void rbtTxUTF8_CheckedChanged(object sender, EventArgs e) {SetTextEncode();}
204 : private void rbtTxEuc_CheckedChanged (object sender, EventArgs e) {SetTextEncode();}
205 : private void rbtTxAuto_CheckedChanged(object sender, EventArgs e) {SetTextEncode();}
206 : // テキストエンコード設定
207 : private void SetTextEncode()
208 : {
209 :     if (rbtRxSJis.Checked) ssv.RxTextCode = ESsvRxTextCode.SJIS;
210 :     else if (rbtRxUTF8.Checked) ssv.RxTextCode = ESsvRxTextCode.UTF8;
211 :     else if (rbtRxEuc.Checked) ssv.RxTextCode = ESsvRxTextCode.EUC;
212 :     else if (rbtRxAuto.Checked) ssv.RxTextCode = ESsvRxTextCode.AUTO;
213 :
214 :     if (rbtTxSJis.Checked) ssv.TxTextCode = ESsvTxTextCode.SJIS;
215 :     else if (rbtTxUTF8.Checked) ssv.TxTextCode = ESsvTxTextCode.UTF8;
216 :     else if (rbtTxEuc.Checked) ssv.TxTextCode = ESsvTxTextCode.EUC;
217 :     else if (rbtTxAuto.Checked) ssv.TxTextCode = ESsvTxTextCode.SJIS;
218 : }
219 : }
220 : }

```


接続クライアント毎のフォーム (ClientForm.cs)

```

1 : using System;
2 : using System.Collections.Generic;
3 : using System.ComponentModel;
4 : using System.Data;
5 : using System.Drawing;
6 : using System.Linq;
7 : using System.Text;
8 : using System.Windows.Forms;
9 : using System.IO;
10 : using CAjrCustCtrl;
11 :
12 : namespace Sil_SockServer1
13 : {
14 :     public partial class ClientForm : Form
15 :     {
16 :         IntPtr      m_hClient;
17 :         CAjrSockServer m_Ssv;
18 :
19 :         public ClientForm()
20 :         {
21 :             InitializeComponent();
22 :         }
23 :         // フォーム開始
24 :         private void ClientForm_Load(object sender, EventArgs e)
25 :         {
26 :             // サイズ変更禁止
27 :             this.FormBorderStyle = FormBorderStyle.FixedSingle;
28 :             // ツールチップ設定
29 :             SAjrTip.Add(txtSndBin, "バイナリデータ (2桁の16進数を空白で区切って) 設定してください");
30 :             SAjrTip.Add(txtSndPkt, "パケットデータ (2桁の16進数を空白で区切って) 設定してください");
31 :             SAjrTip.Add(vthTxtChunk, "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" +
32 :                 "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 :             SAjrTip.Add(vthBinChunk, "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。");
34 :             SAjrTip.Add(vthInv, "リアルタイムに受信したテキストデータに不正な制御コードが含まれる場合は、¥n" +
35 :                 "テキストチャンクではなく、不正チャンクテキストとしてバイナリ表示します。¥n" +
36 :                 "不正な制御コードとは、TAB(0x09)～CR(0x0D)以外の制御コードを意味します。");
37 :             SAjrTip.Add(vthPkt, "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。");
38 :             SAjrTip.Add(vthNoPkt, "リアルタイムに受信したデータ内のシットデータ (DLE・STX～DLE・ETX) 以外の部分をテキストとして表示します。¥n" +
39 :                 "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。");
40 :             SAjrTip.Add(vthTxt, "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" +
41 :                 "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送信します。");
42 :             SAjrTip.Add(vthCtrl, "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。");
43 :             SAjrTip.Add(vthEsc, "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。");
44 :             // 設定値ロード
45 :             SAjrReg.LoadAllCtrls(this);
46 :         }
47 :         // フォーム終了
48 :         private void ClientForm_FormClosed(object sender, FormClosedEventArgs e)
49 :         {
50 :             // ウインド位置セーブ
51 :             SAjrReg.SaveWndPos(this, m_Ssv.GetIpAddrStr(m_hClient) + "_");
52 :             // 設定値セーブ
53 :             SAjrReg.SaveAllCtrls(this);
54 :             // クライアント切断
55 :             m_Ssv.Disconnect(m_hClient);
56 :         }
57 :         // クライアントハンドドル退避
58 :         public void SaveHClient(CAjrSockServer ssv, IntPtr hClient)
59 :         {
60 :             m_Ssv = ssv;
61 :             m_hClient = hClient;
62 :             this.Text = "Sil_SockServer1 (" + m_Ssv.GetIpAddrStr(m_hClient) + ")";
63 :             // ウインド位置ロード
64 :             SAjrReg.LoadWndPos(this, m_Ssv.GetIpAddrStr(m_hClient) + "_");
65 :         }
66 :         // 受信バイナリチャンク表示
67 :         public void ShowBinaryChunk(Byte[] bin)
68 :         {
69 :             vthBinChunk.PutText("----- Binary Chunk " + bin.Length.ToString() + " Bytes -----¥n");
70 :             vthBinChunk.PrintHexDump(bin);
71 :             vthBinChunk.PutText("¥n");
72 :         }
73 :         // 受信テキストチャンク表示
74 :         public void ShowTextChunk(string text)
75 :         {
76 :             vthTxtChunk.PutText(text);
77 :         }

```

```

78 : // 受信不正チャンクテキスト表示
79 : public void ShowInvChunk(byte[] bin)
80 : {
81 :     vthInv.PrintHexDump(bin);
82 :     vthInv.PutText("¥n");
83 : }
84 : // 受信テキスト表示
85 : public void ShowText(string text)
86 : {
87 :     vthTxt.PutText(text + "¥n");
88 : }
89 : // 受信ESC表示
90 : public void ShowEsc(string esc)
91 : {
92 :     vthEsc.PutText("¥¥x1B" + esc.Substring(1));
93 :     vthEsc.PutText("¥n");
94 : }
95 : // 受信制御コード表示
96 : public void ShowCtrl(char ctrl)
97 : {
98 :     int c = (int)ctrl;
99 :     vthCtrl.PutFormat("0x{0:X2}¥n", c);
100 : }
101 : // 受信パケットデータ表示
102 : public void ShowPacket(byte[] bin)
103 : {
104 :     if (bin != null) {
105 :         vthPkt.PrintHexDump(bin);
106 :         vthPkt.PutText("¥n");
107 :     }
108 : }
109 : // 受信パケット外テキスト表示
110 : public void ShowNoPkt(string text)
111 : {
112 :     vthNoPkt.PutText(text);
113 : }
114 : // テキスト表示ウインドにファイルドロップ (ファイル送信)
115 : private void vthTxt_OnFileDrop(object sender, VthArgFileDrop e)
116 : {
117 :     for (int i = 0; i < e.n; i++) {
118 :         string path = vthTxt.GetDroppedFile();
119 :         SubReadAndSend(path);
120 :     }
121 : }
122 : // Disconnect ボタン
123 : private void btnDisconnect_Click(object sender, EventArgs e)
124 : {
125 :     m_Ssv.Disconnect(m_hClient);
126 : }
127 : // テキスト送信ボタン
128 : private void btnSndTxt_Click(object sender, EventArgs e)
129 : {
130 :     m_Ssv.SendText(m_hClient, txtSndTxt.Text + "¥n");
131 : }
132 : // ESC送信ボタン
133 : private void btnSndEsc_Click(object sender, EventArgs e)
134 : {
135 :     m_Ssv.SendText(m_hClient, "¥x1B" + txtSndEsc.Text);
136 : }
137 : // バイナリ送信ボタン
138 : private void btnSndBin_Click(object sender, EventArgs e)
139 : {
140 :     string[] arr = txtSndBin.Text.Split(' ');
141 :     int n = 0;
142 :     // 有効な数算出
143 :     for (int i = 0; i < arr.Length; i++) {
144 :         if (IsByteHexa(arr[i])) n++;
145 :     }
146 :     if (n != 0) {
147 :         // 送信バイナリ作成
148 :         Byte[] BArr = new Byte[n];
149 :         int ix = 0;
150 :         for (int i = 0; i < arr.Length; i++) {
151 :             if (IsByteHexa(arr[i])) {BArr[ix++] = Convert.ToByte(arr[i], 16);}
152 :         }
153 :         if (ix != 0) {
154 :             m_Ssv.SendBinary(m_hClient, BArr);
155 :         }
156 :     }
157 : }

```

```

158 : // パケット送信ボタン
159 : private void btnSndPkt_Click(object sender, EventArgs e)
160 : {
161 :     string[] arr = txtSndPkt.Text.Split(' ');
162 :     int n = 0;
163 :     // 有効な数算出
164 :     for (int i = 0; i < arr.Length; i++) {
165 :         if (IsByteHexa(arr[i])) n++;
166 :     }
167 :     if (n != 0) {
168 :         // 送信バイナリ作成
169 :         Byte[] BArr = new Byte[n];
170 :         int ix = 0;
171 :         for (int i = 0; i < arr.Length; i++) {
172 :             if (IsByteHexa(arr[i])) {BArr[ix++] = Convert.ToByte(arr[i], 16);}
173 :         }
174 :         if (ix != 0) {
175 :             m_Ssv.SendPacket(m_hClient, BArr);
176 :         }
177 :     }
178 : }
179 : // 全てクリアーボタン
180 : private void btnClearAll_Click(object sender, EventArgs e)
181 : {
182 :     vthTxtChunk.Purge();
183 :     vthBinChunk.Purge();
184 :     vthTxt .Purge();
185 :     vthCtrl .Purge();
186 :     vthEsc .Purge();
187 :     vthInv .Purge();
188 :     vthPkt .Purge();
189 :     vthNoPkt .Purge();
190 : }
191 : // バイナリファイルの読み出しと送信
192 : private void SubReadAndSend(string FilePath)
193 : {
194 :     FileStream fs = new FileStream(FilePath, FileMode.Open, FileAccess.Read);
195 :     int FileSize = (int)fs.Length; // ファイルのサイズ
196 :     byte[] buf = new byte[FileSize]; // データ格納用配列
197 :     fs.Read(buf, 0, FileSize); // ファイル読み出し
198 :     m_Ssv.SendBinary(m_hClient, buf); // バイナリデータ送信
199 :     fs.Dispose();
200 : }
201 : // 16進文字列チェック
202 : private bool IsByteHexa(string s)
203 : {
204 :     bool rc = false;
205 :     do {
206 :         if (string.IsNullOrEmpty(s)) break;
207 :         if (s.Length != 2) break;
208 :         if (!Uri.IsHexDigit(s[0])) break;
209 :         if (!Uri.IsHexDigit(s[1])) break;
210 :         rc = true;
211 :     } while (false);
212 :     return rc;
213 : }
214 : }
215 : }

```

12.6.2. Sil_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。



接続しているクライアントの
IP アドレスと送受信バイト数

```

1 : using System;
2 : using System.Collections.Generic;
3 : using System.ComponentModel;
4 : using System.Data;
5 : using System.Drawing;
6 : using System.Linq;
7 : using System.Text;
8 : using System.Windows.Forms;
9 : using CAjrCustCtrl;
10 :
11 : namespace Sil_SockServer2
12 : {
13 :     public partial class Form1 : Form
14 :     {
15 :         uint    MaxCli = 10;           // 最大クライアント数
16 :         long[]  total = new long[10]; // 受信バイト数
17 :         Label[] ipa;                  // IP アドレス・コントロール
18 :         Label[] cnt;                  // 受信バイト数・コントロール
19 :
20 :         public Form1()
21 :         {
22 :             InitializeComponent();
23 :         }
24 :         // フォーム開始
25 :         private void Form1_Load(object sender, EventArgs e)
26 :         {
27 :             // ホスト名表示
28 :             lblMyName.Text = Environment.MachineName;
29 :             // ウインド位置ロード
30 :             SAjrReg.LoadWndPos(this);
31 :             // テーブル初期化
32 :             ipa = new Label[] { lblIpA0, lblIpA1, lblIpA2, lblIpA3, lblIpA4, lblIpA5, lblIpA6, lblIpA7, lblIpA8, lblIpA9 };
33 :             cnt = new Label[] { lblCnt0, lblCnt1, lblCnt2, lblCnt3, lblCnt4, lblCnt5, lblCnt6, lblCnt7, lblCnt8, lblCnt9 };
34 :             // テーブルクリアー
35 :             for (int i = 0; i < MaxCli; i++) {
36 :                 ipa[i].Text = "-";
37 :                 cnt[i].Text = "-";
38 :             }
39 :             // サーバ開始
40 :             ssv.Start(14238, 10);
41 :         }
42 :
43 :         // フォーム終了
44 :         private void Form1_FormClosed(object sender, FormClosedEventArgs e)
45 :         {
46 :             // ウインド位置セーブ
47 :             SAjrReg.SaveWndPos(this);
48 :             // サーバ停止
49 :             ssv.Stop();

```

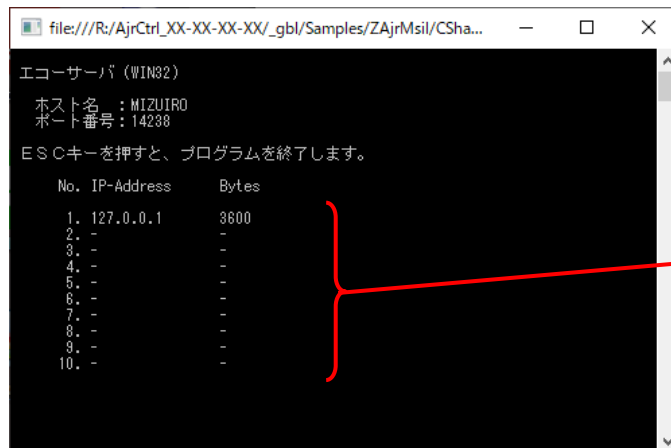
```

50 :     }
51 :     // クライアント接続
52 :     private void cAjrSockServer1_OnConnect(object sender, CAjrCustCtrl.SsvArgConnect e)
53 :     {
54 :         int i = ssv.GetIndex(e.hClient);
55 :         // バイト数カウンタクリア
56 :         total[i] = 0;
57 :         // クライアント IP アドレス表示
58 :         ipa[i].Text = ssv.GetIpAddrStr(e.hClient);
59 :         cnt[i].Text = "0";
60 :     }
61 :     // クライアント切断
62 :     private void cAjrSockServer1_OnDisconnect(object sender, CAjrCustCtrl.SsvArgDisconnect e)
63 :     {
64 :         // クライアント表示クリア
65 :         int i = ssv.GetIndex(e.hClient);
66 :         ipa[i].Text = "-";
67 :         cnt[i].Text = "-";
68 :     }
69 :     // バイナリチャンク受信
70 :     private void cAjrSockServer1_OnRxChunkBin(object sender, CAjrCustCtrl.SsvArgRxChunkBin e)
71 :     {
72 :         int i = ssv.GetIndex(e.hClient);
73 :         // 受信データを返送
74 :         ssv.SendBinary(e.hClient, e.bin);
75 :         // バイトカウンタ更新
76 :         total[i] += e.bin.Length;
77 :         cnt[i].Text = total[i].ToString("N0");
78 :     }
79 : }
80 : }

```

12.6.3. Sil_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、Sil_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。



接続しているクライアントの
IPアドレスと送受信バイト数

```

1 : // Sil_SockServer2C
2 :
3 : using System;
4 : using System.Collections.Generic;
5 : using System.Linq;
6 : using System.Text;
7 : using System.Runtime.InteropServices;
8 : using CAjrCustCtrl;
9 :
10 : namespace Sil_SockServer2C
11 : {
12 :     // クライアント情報
13 :     public struct ClientsInfo
14 :     {
15 :         public string ipaddr;
16 :         public bool busy;
17 :         public int bytes;
18 :     }
19 :
20 :     class Program
21 :     {
22 :         // ソケットサーバ インスタンス
23 :         static CAjrSockServer ssv = new CAjrSockServer();
24 :
25 :         static void Main(string[] args)
26 :         {
27 :             IntPtr hClient; // クライアントハンドル
28 :             ESsvEvt evt; // イベントマスク
29 :             object obj; // イベントデータ
30 :             string IpAddr = ""; // クライアントの IP アドレス
31 :
32 :             // コンソールキーオブジェクト生成
33 :             ConsoleKeyInfo c = new ConsoleKeyInfo();
34 :
35 :             // クライアント情報テーブル生成
36 :             ClientsInfo[] CliTbl = new ClientsInfo[10];
37 :
38 :             // コンソールアプリ終了ハンドラ登録
39 :             m_CbkConApExit = new CbkConApExit(SsvConApExit);
40 :             SetConsoleCtrlHandler(m_CbkConApExit, true);
41 :
42 :             // 初期表示
43 :             Console.Clear();
44 :             Console.WriteLine("");
45 :             Console.WriteLine(" エコーサーバ (" + (IntPtr.Size == 4 ? "WIN32" : "WIN64") + ")");
46 :             Console.WriteLine("");
47 :             Console.WriteLine(" ホスト名 : " + Environment.MachineName);
48 :             Console.WriteLine(" ポート番号 : 14238");
49 :             Console.WriteLine("");
50 :             Console.WriteLine(" E S C キーを押すと、プログラムを終了します。");
51 :             Console.WriteLine("");

```

```

52 : Console.WriteLine("      No. IP-Address      Bytes");
53 : Console.WriteLine("");
54 : Console.WriteLine("      1. -              -");
55 : Console.WriteLine("      2. -              -");
56 : Console.WriteLine("      3. -              -");
57 : Console.WriteLine("      4. -              -");
58 : Console.WriteLine("      5. -              -");
59 : Console.WriteLine("      6. -              -");
60 : Console.WriteLine("      7. -              -");
61 : Console.WriteLine("      8. -              -");
62 : Console.WriteLine("      9. -              -");
63 : Console.WriteLine("     10. -             -");
64 :
65 : // 実行モード設定 (イベントを「WaitEvent」で待つ)
66 : ssv._SsvMode = ESsvMode.WaitingForEvent;
67 : // 対象とするイベントを設定
68 : ssv.SetEvtMask(ESsvEvt.EV_CONNECT | ESsvEvt.EV_DISCONNECT | ESsvEvt.EV_RXCHUNK | ESsvEvt.EV_STOP);
69 : // サーバ開始 (ポート番号: 14238, 最大クライアント数: 10)
70 : ssv.Start(14238, 10);
71 : // ループ
72 : while (true) {
73 :     // E S C キー入力でサーバ終了
74 :     if (c.Key == ConsoleKey.Escape) {
75 :         ssv.Stop();
76 :     }
77 :     // イベント待ち
78 :     obj = null;
79 :     if ((evt = ssv.WaitEvent(out obj, 1000, out hClient)) != ESsvEvt.EV_NOEVENT) {
80 :         // クライアントの IP アドレス取得
81 :         if (hClient != (IntPtr)0) {
82 :             IpAddr = ssv.GetIpAddrStr(hClient);
83 :         }
84 :         // 各イベント処理
85 :         // ●接続通知
86 :         if ((evt & ESsvEvt.EV_CONNECT) != 0) {
87 :             // クライアントテーブルの空きエントリ検索
88 :             int ix;
89 :             for (ix = 0; ix < 10 && CliTbl[ix].busy; ix++);
90 :             if (ix < 10) {
91 :                 // テーブルインデックスをクライアントに関連付け
92 :                 ssv.SetClientData(hClient, ix);
93 :                 // テーブルエントリ設定
94 :                 CliTbl[ix].ipaddr = ssv.GetIpAddrStr(hClient);
95 :                 CliTbl[ix].busy = true;
96 :                 CliTbl[ix].bytes = 0;
97 :                 // クライアント情報初期表示
98 :                 // SAjrCon.SetCursorPos(10, 10 + ix); Console.Write(CliTbl[ix].ipaddr);
99 :                 // SAjrCon.SetCursorPos(26, 10 + ix); Console.Write(CliTbl[ix].bytes.ToString());
100 :                 Console.SetCursorPosition(10, 10 + ix); Console.Write(CliTbl[ix].ipaddr);
101 :                 Console.SetCursorPosition(26, 10 + ix); Console.Write(CliTbl[ix].bytes.ToString());
102 :             }
103 :         }
104 :         // ●切断通知
105 :         if ((evt & ESsvEvt.EV_DISCONNECT) != 0) {
106 :             // クライアントインデックス取得
107 :             int ix = (int)obj;
108 :             // クライアントテーブルエントリに「空き」をマーク
109 :             CliTbl[ix].busy = false;
110 :             // クライアント情報を消す
111 :             // SAjrCon.SetCursorPos(10, 10 + ix); Console.Write("-");
112 :             // SAjrCon.SetCursorPos(26, 10 + ix); Console.Write("-");
113 :             Console.SetCursorPosition(10, 10 + ix); Console.Write("-");
114 :             Console.SetCursorPosition(26, 10 + ix); Console.Write("-");
115 :         }
116 :         // ●バイナリチャンク受信通知
117 :         if ((evt & ESsvEvt.EV_RXCHUNK) != 0) {
118 :             Byte[] rxd = (Byte[])obj;
119 :             // クライアントインデックス取得
120 :             object oix;
121 :             ssv.GetClientData(hClient, out oix);
122 :             int ix = (int)oix;
123 :             // 受信バイト数を表示
124 :             CliTbl[ix].bytes += rxd.Length;
125 :             Console.SetCursorPosition(26, 10 + ix); Console.Write(CliTbl[ix].bytes.ToString());
126 :             // 受信データを返信
127 :             ssv.SendBinary(hClient, (byte[])obj);
128 :         }
129 :         // ●サーバ終了通知
130 :         if ((evt & ESsvEvt.EV_STOP) != 0) {
131 :             break;

```

```

132 :         }
133 :         // イベント処理終了
134 :         ssv.EndEvent();
135 :     }
136 :     // キー入力
137 :     if (Console.KeyAvailable) c = Console.ReadKey(true);
138 : }
139 : // ソケットサーバ消去
140 : ssv.Delete();
141 :
142 : Console.SetCursorPosition(0, 20);
143 : }
144 :
145 : // コンソールアプリ終了ハンドラ用デリゲート
146 : [DllImport("Kernel32")]
147 : static extern bool SetConsoleCtrlHandler(CbkConApExit Handler, bool Add);
148 : delegate bool CbkConApExit(EAJCEXITTYPE ExitType);
149 : static CbkConApExit m_CbkConApExit;
150 : // コンソールアプリ終了ハンドラ
151 : static bool SsvConApExit(EAJCEXITTYPE ExitType)
152 : {
153 :     // ソケットサーバ消去
154 :     ssv.Delete();
155 :     // false : 次のイベントハンドラへリンクする
156 :     return false;
157 : }
158 : }
159 : }

```


13. ソケット (TCP/IP) クライアント機能 (CAjrSockClient.dll)

ソケット (TCP/IP) のクライアント側通信制御モジュールです。
サーバとソケット接続し通信することができます。

13.1. 機能概要

13.1.1. 受信データの通知形式と送受信文字コード

本書冒頭の「受信データの通知形式と送受信文字コード」章を参照してください。

13.1.2. イベントの通知方法

イベントの通知方法は、_SsvMode プロパティにより、以下の 2 つから選択できます。

- ・本コントロールがイベントを発生する (_SctMode= ESctpMode.NotificationByEvent, デフォルト)
- ・ユーザが、その場でイベントの発生を待ち受ける (_SctMode= ESctMode.WaitingForEvent)

_SctMode プロパティは、デザインモード時に指定するようにしてください。
コンソールアプリ等で、デザインモードで指定できない場合は、プログラムの開始時に設定してください。

_SctMode プロパティを **ESctMode. NotificationByEvent** に設定した場合は、本コントロールがイベントを発生し、OnXXXXX() イベントで事象をユーザに通知します。デフォルトでは、このモードに設定されています。
このモードは、Windows フォームアプリケーションで使用可能です。

_SctMode プロパティを **ESctMode. WaitingForEvent** に設定した場合は、WaitEvent() メソッドによりユーザがイベントの発生を待ち受けます。
このモードは、コンソール・アプリケーションで使用します。

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、タイムアウト時間を指定して、その場で応答データの着信を待つことが可能です。

13.2. 構造体／列挙体 (定数)

13.2.1. 実行モード

```
public enum ESctMode : int
{
    NotificationByEvent = 0,    // イベントを通知する (OnXXXXイベントで通知する)
    WaitingForEvent     = 1    // イベントを待ち受ける
}
```

13.2.2. チャンクデータの通知モード

```
public enum ESctChunkMode : int
{
    BinaryData    = 0x01,    // バイナリ・チャンク
    TextData      = 0x02,    // テキスト・チャンク
    Both          = 0x03,    // 両方
}
```

13.2.3. イベントコード

```
public enum ESctEvt : int
{
    EV_RXTEXT      = 0x00000001,    // テキストデータ通知
    EV_RXESC       = 0x00000002,    // E S C コードデータ通知
    EV_RXPKT       = 0x00000004,    // パケットデータ通知
    EV_RXCTRL      = 0x00000008,    // 制御コード通知
    EV_RXNOPKT     = 0x00000010,    // パケット外テキストデータ

    EV_RXCHUNK     = 0x00000100,    // チャンクデータ受信通知
    EV_INVCHUNK    = 0x00000200,    // 不正チャンクテキスト受信通知
    EV_TXEMPTY     = 0x00000400,    // 送信完了通知
    EV_CONNECT     = 0x00000800,    // 接続通知
    EV_DISCONNECT  = 0x00001000,    // 切断通知
    EV_CNFAIL      = 0x00002000,    // 接続失敗通知

    EV_RXERR       = 0x00010000,    // 受信エラー通知
    EV_TXERR       = 0x00020000,    // 送信エラー通知
    EV_ERR         = 0x08000000,    // エラー通知

    EV_SSEP        = (EV_RXTEXT | EV_RXESC | EV_RXCTRL | EV_RXPKT),
    EV_COMM        = (EV_RXCHUNK | EV_INVCHUNK | EV_TXEMPTY),
    EV_GENERAL     = (EV_CONNECT | EV_DISCONNECT),
    EV_ERRS        = (EV_RXERR | EV_TXERR | EV_ERR),
    EV_CLIENT      = (EV_SSEP | EV_COMM | EV_CONNECT | EV_DISCONNECT | EV_RXERR | EV_TXERR),
    EV_DEFAULT     = (EV_SSEP | EV_COMM | EV_GENERAL | EV_ERRS),
    EV_ALL         = (EV_SSEP | EV_COMM | EV_GENERAL | EV_ERRS | EV_RXNOPKT)
}
```

13.2.4. エラーコード

```

public enum ESctErrorCode {
    ERR_CREEVT      = 10,          // 終了通知用イベントオブジェクト生成失敗
    ERR_SOCKET      = 20 ,         // 接続要求待機用ソケット生成失敗(socket)
    ERR_CONNECT     = 40 ,         // CONNECT 失敗(connect)
    ERR_ADDRINFO    = 60 ,         // アドレス情報取得失敗 (getaddrinfo)
    ERR_SOCKOPT     = 70 ,         // ソケットオプション設定失敗(setsockopt)
    ERR_SUBTHREAD   = 90 ,         // クライアント通信サブスレッド開始失敗
    ERR_CRESSEP     = 100 ,        // ストリーム分離インスタンス生成失敗
    ERR_CREMBXTXD   = 110 ,        // 送信データ用メールボックス生成失敗
    ERR_TIMEOUT     = 120 ,        // サブスレッド終了タイムアウト
}

```

13.2.5. 回線状態

```

public enum ESctState {
    DISCONNECT      = 0 ,          // 切断状態
    CONNECTING      = 1 ,          // 接続中
    CONNECT         = 2 ,          // 接続状態
    DISCONNECTING   = 3 ,          // 切断中
}

```

13.2.6. 受信テキストの文字コード

```

public enum ESctRxTextCode : int
{
    SJIS            = 1 ,          // S - J I S
    EUC              = 2 ,          // E U C
    UTF8            = 3 ,          // U T F - 8
    AUTO            = 9 ,          // 自動認識
}

```

13.2.7. 送信テキストの文字コード

```

public enum ESctTxTextCode : int
{
    SJIS            = 1 ,          // S - J I S
    EUC              = 2 ,          // E U C
    UTF8            = 3 ,          // U T F - 8
    AUTO            = 9 ,          // 受信テキストコードに合わせる
}

```

13.2.8. アドレスファミリ

```

public enum ESctFamily : int
{
    INET            = 0x02,        // IPV4
    INET6           = 0x17,        // IPV6
}

```

13.3. プロパティ

TCP/IP クライアント機能のプロパティ一覧を示します。
プロパティは、Start() メソッドを実行する前に設定してください。

#	名称	タイプ	内容	デフォルト
1	_SctMode	ESctMode	実行モード (※1) <u>NotificationByEvent</u> : イベントの発生を通知で受ける <u>WaitingForEvent</u> : イベントの発生をその場で待ち受ける	NotificationByEvent
2	ChunkMode	ESctChunkMode	チャンクデータの通知モード <u>BinaryData</u> : バイナリデータとして扱う <u>TextData</u> : テキストデータとして扱う <u>Both</u> : 両方	BinaryData
3	State	ESctState	回線状態 (読み出し専用) ・ DISCONNECT : 切断状態 ・ CONNECT : 接続状態 ・ CONNECTING : 接続中 ・ DISCONNECTING : 切断中	
4	STX	int	パケットデータの先頭識別コード	0x02
5	ETX	int	パケットデータの終端識別コード	0x03
6	DLE	int	パケットデータの投下制御コード	0x10
7	PktTimeout	int	パケットデータ受信タイムアウト時間 [ms]	3000 [ms]
8	RxTextCode	ESctRxTextCode	受信テキストコード (SJIS / EUC / UTF8 / AUTO (自動判別)) (※2)	SJIS
9	TxTextCode	ESctTxTextCode	送信テキストコード (SJIS / EUC / UTF8 / AUTO (受信テキストコードと同じ))	SJIS
10	ActualRxTextCode	ESctRxTextCode	実際の受信テキストコード (SJIS / EUC / UTF8) AUTO 設定時は、実際に受信したテキストから自動判別	読み出し専用 SJIS
10	ActualTxTextCode	ESctTxTextCode	実際の送信テキストコード (SJIS / EUC / UTF8) AUTO 設定時は、受信テキストコードと同じ	

※1 : コンソールアプリの場合、WaitingForEvent を設定し、WaitEvent() メソッドでイベントを待ち受けてください。

※2 : 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、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。

13.4. メソッド

TCP/IP クライアント機能のソッド一覧を以下に示します。

#	メソッド名	内容	備考
1	Init	初期設定 (最初に1度だけ、必ず実行する必要があります)	
2	Connect	回線接続	
3	Disconnect	回線切断	
4	SetEvtMask	イベントマスク設定 (コンソールアプリ用)	
5	WaitEvent	イベント待ち	
6	SendByte	1 バイト送信	
7	SendChar	1 文字送信	
8	SendText	テキストデータ送信	
9	SendBinary	バイナリデータ送信	
10	SendPacket	パケット送信	
11	PurgeRx	受信済データ破棄	
12	PurgeTx	送信待ちデータ破棄	
13	Purge	送受信データ破棄	
14	Delete	インスタンス消去	

13.4.1. 初期設定 (Init)

形 式 : void Init();

引 数 : なし

説 明 : ソケット(TCP/IP)クライアント機能の初期化を行います。
このメソッドは、他のメソッドの先駆けて、最初に1度だけ実行する必要があります。

戻り値 : なし

13.4.2. 回線接続 (Connect)

形 式 : void Start(string Serv, uint PortNo);
void Start(string Serv, uint PortNo, ESctFamily AddressFamily);

引 数 : Serv - サーバ名/IP アドレス
PortNo - TCP/IP ポート番号
AddressFamily - アドレスファミリ (INET:IPV4, INET6:IPV6)・・・省略時は、INET を仮定

説 明 : サーバとの回線接続を開始します。

戻り値 : なし

13.4.3. 回線切断(Disconnect)

形 式 : void Disconnect();
void Disconnect (int msTimeout);

引 数 : msTimeout - 回線切断終了待ち時間[ms] (省略時は、10000 (10 秒)を仮定)

説 明 : サーバとの回線を切断します。

戻り値 : なし

13.4.4. イベントマスク設定／取得(SetEvtMask)

形 式 : void SetEvtMask(ESctEvt evt); --- 設定
ESctEvt GetEvtMask(); ----- 取得

引 数 : evt - イベントマスク (EV_XXXXXの合成値)

説 明 : _SctMode プロパティ = WaitingForEvent (コンソールアプリ) の場合のイベントマスクを設定します。
使用するイベントを EV_XXXXX の合成値で指定し、Connect() メソッドよりも前に実行します。

戻り値 : なし

備 考 : _SctMode プロパティ = NotificationByEvent (Windows フォームアプリ) の場合は、使用するイベントをイベントハンドラ (OnXXXXX) で指定する為、イベントマスクを指定する必要はありません。

13.4.5. イベント待ち(WaitEvent)

形 式 : ESctEvt WaitEvent(out Object EvtData, int msWaitTime)

引 数 : EvtData - 受信データ
 msWaitTime - イベント待ち時間

説 明 : イベントの発生を待ちます。
 このメソッドを実行する場合は、_SctMode (実行モード) プロパティを WaitingForEvent に設定していなければなりません。
 EvtData には、以下の情報が格納されます。

イベント	イベントデータ (EvtData)		備考
	タイプ	内容	
EV_NOEVT	-	null	イベント待ちタイムアウト
EV_RXTEXT	String	受信テキストデータ	TAB(0x09)を含む
EV_RXESC	String	受信 ESC シーケンス	
EV_RXCTRL	char	受信制御コード	TAB(0x09)を除く
EV_RXPKT	Byte[]	受信パケットデータ	バイナリデータ
EV_RXNOPKT	String	受信パケット外テキスト	
EV_RXCHUNK	String	受信チャンクデータ (テキスト)	ChunkMode=TextData/Both
	Byte[]	" (バイナリ)	ChunkMode= BinaryData/Both
EV_INVCHUNK	Byte[]	受信不正チャンクテキスト	ChunkMode=TextData/Both
EV_TXEMPTY	-	null	
EV_CONNECT	-	null	
EV_DISCONNECT	-	null	
EV_CNFAIL	int	エラーコード	Win32-API connect() 直後の GetLastError() の値
EV_RXERR	int	エラー発生種別 0:WSARecv 1:GetOverlappedResult	
EV_TXERR	int	エラー発生種別 0:WSASend 1: GetOverlappedResult	
EV_ERR	ESctErrorCode	エラーコード (ERR_XXXX)	

イベント発生待ちタイムアウトの場合は、EV_NOEVENT を返し、EvtData=null が設定されます。

戻り値 : 発生したイベントのイベントコード (EV_XXXX)

備 考 : チャンクデータ受信通知 (EV_RXCHUNK) で、テキストチャンクと、バイナリチャンクデータの両方を扱う (ChunkMode= Both) 場合は、以下の条件でテキストチャンク/バイナリチャンクを特定してください。

- EvtData.GetType() == typeof(byte[]) -- バイナリチャンクデータ
- EvtData.GetType() == typeof(string) -- テキストチャンクデータ

13.4.6. 1バイト送信(SendByte)

形 式 : void SendChar(byte ByteData);

引 数 : Character - 送信するバイトデータ

説 明 : ByteData で指定された 1 バイトを送信します。
このメソッドは送信用スプールバッファにデータを格納するだけであり、送信完了を待たずに、すぐに制御を返します。

戻り値 : なし

13.4.7. 1文字送信(SendChar)

形 式 : void SendChar(IntPtr hClient, char Character);

引 数 : Character - 送信する文字データ

説 明 : 文字データ (Character) を TxTextCode プロパティで指定された文字コードに変換して送信します。
このメソッドは送信用スプールバッファにデータを格納するだけであり、送信完了を待たずに、すぐに制御を返します。

戻り値 : なし

13.4.8. テキストデータ送信(SendChar)

形 式 : void SendText(string text);

引 数 : text - 送信するテキストデータ

説 明 : テキストデータを TxTextCode プロパティで指定された文字コードに変換して送信します。
このメソッドは送信用スプールバッファにデータを格納するだけであり、送信完了を待たずに、すぐに制御を返します。

戻り値 : なし

13.4.9. バイナリデータ送信(SendBinary)

形 式 : void SendBinary(Byte[] bin);
void SendBinary(IntPtr p, int len);
unsafe void SendBinary(void *p, int len);

引 数 : bin - 送信するバイナリデータのバイト配列
p - 送信するバイナリデータのアドレス
len - 送信するバイナリデータのバイト数

説 明 : バイナリデータを送信します。
このメソッドは送信用スプールバッファにデータを格納するだけであり、送信完了を待たずに、すぐに制御を返します。

戻り値 : なし

13.4.10. パケット送信(SendPacket)

形 式 : int SendPacket(Byte[] bin);
 int SendPacket(IntPtr p, int len);
 unsafe int SendPacket(void *p, int len);

引 数 : bin - 送信するバイナリデータのバイト配列 (空パケット (DLE, STX, DLE, ETX の 4 バイト)送信時は null)
 p - 送信するバイナリデータのアドレス (空パケット (DLE, STX, DLE, ETX の 4 バイト)送信時は 0)
 len - 送信するバイナリデータのバイト数 (空パケット (DLE, STX, DLE, ETX の 4 バイト)送信時は 0)

説 明 : パケット形式でバイナリデータを送信します。
 つまり、先頭に「DLE・STX」の 2 バイトを、末尾に「DLE・ETX」の 2 バイトを付加し、バイナリデータ中の DLE バイトは、2 つの DLE に変換 (DLE → DLE・DLE) して伝送します。
 STX, ETX, DLE の実際のコード値はプロパティにて設定可能です。(規定値は、STX=0x02, ETX=0x03, DLE=0x10)
 このメソッドは送信用スプールバッファにデータを格納するだけであり、送信完了を待たずに、すぐに制御を返します。

戻り値 : 4 ~ - 正常 ((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 - エラー

13.4.11. 受信済データ破棄(PurgeRx)

形 式 : void PurgeRx();

引 数 : なし

説 明 : 受信済みデータを破棄します。

戻り値 : なし

13.4.12. 送信待ちデータ破棄(PurgeTx)

形 式 : void PurgeTx();

引 数 : なし

説 明 : 送信待ちデータを破棄します。

戻り値 : なし

13.4.13. 送受信データ破棄(Purge)

形 式 : void Purge();

引 数 : なし

説 明 : 受信済データと、送信待ちデータを破棄します。

戻り値 : なし

13.4.14. インスタンス消去(Delete)

形 式 : void Delete()

引 数 : なし

説 明 : インスタンスを消去します。
回線を切断し、送受信スレッドを停止後にインスタンスを消去します。

このメソッドは、コンソールアプリの場合に実行してください。
Windows フォームアプリ等では実行しないでください。(自動的に実行されます)

戻り値 : なし

13.5. イベント

シリアル通信コントロールのイベント一覧を以下に示します。

#	イベント名	内容	備考
1	OnRxText	テキスト受信通知	
2	OnRxEsc	E S C 受信通知	
3	OnRxCtrl	制御コード受信通知	
4	OnRxPacket	パケット受信通知	
5	OnRxNoPkt	パケット外テキスト受信通知	
6	OnTxEmpty	送信完了通知	
7	OnRxChunkTxt	テキストチャンク受信通知	
8	OnRxChunkBin	バイナリチャンク受信通知	
9	OnRxInvChunk	不正チャンクテキスト受信通知	
10	OnConnect	接続通知	
11	OnDisconnect	切断通知	
12	OnCnFail	接続失敗通知	
13	OnRecvError	受信エラー通知	
14	OnSendError	送信エラー通知	
15	OnGeneralError	その他のエラー通知	

13.5.1. テキスト受信通知 (OnRxText)

形 式 : void OnRxText (object sender, SctArgRxText e);

パラメタ : string e.text - 受信したテキストデータ

説 明 : 受信したテキストデータを通知します。
 テキストデータとは、T A B (0x09)を除く制御コードで区切られたデータを意味します。
 T A B (0x09)はテキストデータ内に含めます。

戻り値 : なし

13.5.2. E S C 受信通知 (OnRxEsc)

形 式 : void OnRxEsc (object sender, SctArgRxEsc e);

パラメタ : string e.text - 受信したエスケープシーケンスのテキスト

説 明 : 受信したエスケープシーケンスのテキストを通知します。
 エスケープシーケンスとは、0x1B で始まり英字で終わるテキストを意味します。

戻り値 : なし

13.5.3. 制御コード受信通知 (OnRxCtrl)

形 式 : void OnRxCtrl (object sender, SctArgRxCtrl e);

パラメタ : char e.ctrl - 受信した制御コード

説 明 : クライアントから受信した制御コードを通知します。
 制御コードとは、T A B (0x09)を除く、0x00～0x1F と 0x7F を意味します。

戻り値 : なし

13.5.4. パケット受信通知 (OnRxPacket)

形 式 : void OnRxPacket (object sender, SctArgRxPacket e);

パラメタ : Byte[] e.bin - パケットデータのアドレス (空パケットの場合は null)

説 明 : パケットデータを受信したことを通知します。
 パケットデータは、デコードされたデータだけをを通知します。
 つまり、先頭の「DLE・STX」と末尾の「DLE・ETX」は除去され、連続する 2 つの DLE を 1 つの DLE に変換したデータが通知されます。
 空のパケット (データ無しで、DLE・STX と DLE・ETX の 4 バイトのみ) の場合は、e.bin = null が設定されます。

戻り値 : なし

13.5.5. パケット外テキスト受信通知 (OnRxNoPkt)

形 式 : void OnRxNoPkt (object sender, SctArgRxNoPkt e);

パラメタ : string e.text - パケット外テキストデータ

説 明 : 受信したチャンクデータの packets 部分 (STX・DLE~ETX・DLE) を除いた部分のテキストを通知します。
 通知されるデータは、テキスト受信通知 (OnRxText) とほぼ同じになりますが、テキスト受信通知の場合はテキスト終端 (0x0D や 0x0A) 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受信通知 (OnTxNoPkt) ではテキスト終端を待たずにリアルタイムに通知されます。また、パケット外テキストには制御コードも含まれます。
 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。

戻り値 : なし

備 考 : 受信データ中のマルチバイト文字が分断されないように制御されます。

13.5.6. 送信完了通知 (OnTxEmpty)

形 式 : void OnTxEmpty (object sender, EventArgs e);

パラメタ : なし

説 明 : 送信が完了した (送信待ちとなっているデータが無くなった) ことを通知します。

戻り値 : なし

13.5.7. テキストチャンク受信通知 (OnRxChunkTxt)

形 式 : void OnRxChunkTxt (object sender, SctArgRxChunkTxt e);

パラメタ : string e.text - テキストチャンクデータ

説 明 : 受信したテキストチャンクデータを通知します。
 このテキストデータはリアルタイムに通知されます。

戻り値 : なし

備 考 : 受信データ中のマルチバイト文字が分断されないように制御されます。

13.5.8. バイナリチャンク受信通知 (OnRxChunkBin)

形 式 : void OnRxChunkBin (object sender, SctArgRxChunkBin e);

パラメタ : Byte[] e.bin - バイナリチャンクデータ

説 明 : 受信したバイナリチャンクデータを通知します。
このバイナリデータはリアルタイムに通知されます。

戻り値 : なし

13.5.9. 不正チャンクテキスト受信通知 (OnRxInvChunk)

形 式 : void OnRxInvChunk (object sender, SctArgRxInvChunk e);

パラメタ : Byte[] e.bin - 受信したチャンクテキスト (バイナリデータ)

説 明 : テキストチャンクに不正な制御コードが含まれていることを通知します。
チャンクデータの扱いをテキスト (ChunkMode = TextData) としている場合で、チャンクデータに不正な制御コード (0x09~0x0D, 0x1B 以外) が含まれている場合は、テキストチャンク受信通知 (OnRxChunkTxt) ではなく、このイベントでチャンクデータ (バイナリデータ) が通知されます。
このバイナリデータはリアルタイムに通知されます。

戻り値 : なし

13.5.10. 接続通知 (OnConnect)

形 式 : void OnConnect (object sender, EventArgs e);

パラメタ : なし

説 明 : サーバと接続したことを通知します・

戻り値 : なし

13.5.11. 切断通知 (OnDisconnect)

形 式 : void OnDisconnect (object sender, EventArgs e);

パラメタ : なし

説 明 : サーバとの回線が切断したことを通知します・

戻り値 : なし

13.5.12. 接続失敗通知 (OnCnFail)

形 式 : void OnCnFail (object sender, SctArgCnFail e);

パラメタ : int e.err - エラーコード (Win32API connect()直後の GetLastError() 値)

説 明 : サーバとの回線が失敗したことを通知します。

戻り値 : なし

13.5.13. 受信エラー通知 (OnRecvError)

形 式 : void OnRecvError (object sender, SctArgRecvError e);

パラメタ : bool e.overlapped - true : WSARecv() のエラー
false : WSAGetOverlappedResult() のエラー

説 明 : 受信エラーが発生したことを通知します。

戻り値 : なし

13.5.14. 送信エラー通知 (On SendError)

形 式 : void On SendError (object sender, SctArgSendError e);

パラメタ : bool e.overlapped - true : WSASend() のエラー
false : WSAGetOverlappedResult() のエラー

説 明 : 送信エラーが発生したことを通知します。

戻り値 : なし

13.5.15. その他のエラー通知 (OnGeneralError)

形 式 : void OnGeneralError (object sender, SctArgGeneralError e);

パラメタ : ESctErrorCode e. ErrorCode - エラーコード

説 明 : 送受信エラー以外のエラーを通知します。

戻り値 : なし

13.6. サンプルプログラム

13.6.1. Sil_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 : using System;
2 : using System.Collections.Generic;
3 : using System.ComponentModel;
4 : using System.Data;
5 : using System.Drawing;
6 : using System.Text;
7 : using System.Windows.Forms;
8 : using System.IO;
9 : using CAjrCustCtrl;
10 :
11 : namespace Sil_SockClient1
12 : {
13 :     public partial class Form1 : Form
14 :     {
15 :         bool m_fConnect = false;
16 :
17 :         public Form1()
18 :         {
19 :             InitializeComponent();
20 :         }
21 :         //----- 起動時初期設定 -----//
22 :         private void Form1_Load(object sender, EventArgs e)
23 :         {
24 :             this.Text = this.Text + (IntPtr.Size == 4 ? " (WIN32)" : " (WIN64)");
25 :             // サイズ変更禁止
26 :             this.FormBorderStyle = FormBorderStyle.FixedSingle;
27 :             // ツールチップ設定
28 :             SAjrTip.Add(txtSndBin, "バイナリデータ (2桁の16進数を空白で区切って) 設定してください");
29 :             SAjrTip.Add(txtSndPkt, "パケットデータ (2桁の16進数を空白で区切って) 設定してください");
30 :             SAjrTip.Add(vthTxtChunk, "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。 \r\n" +
31 :                 "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 :             SAjrTip.Add(vthBinChunk, "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。");
33 :             SAjrTip.Add(vthInv, "リアルタイムに受信したテキストデータに不正な制御コードが含まれる場合は、 \r\n" +
34 :                 "テキストチャンクではなく、不正チャンクテキストとしてバイナリ表示します。 \r\n" +
35 :                 "不正な制御コードとは、TAB(0x09)～CR(0x0D)以外の制御コードを意味します。");
36 :             SAjrTip.Add(vthPkt, "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。");
37 :             SAjrTip.Add(vthNoPkt, "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。 \r\n" +

```

```

38 :                                     "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。");
39 :     SAjrTip.Add(vthTxt      , "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" +
40 :                               "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送信します。");
41 :     SAjrTip.Add(vthCtrl    , "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。");
42 :     SAjrTip.Add(vthEsc     , "受信したストリームから、E S Cシーケンス (0x1B〜英字) を抜きだして表示します。");
43 :     // ウインド位置ロード
44 :     SAjrReg.LoadWndPos(this);
45 :     // 設定値ロード
46 :     SAjrReg.LoadAllCtrls(this);
47 :     // S C P 初期化
48 :     sct.Init();
49 :     // テキストエンコード設定
50 :     SetTextEncode();
51 : }
52 : //----- フォーム終了 -----//
53 : private void Form1_FormClosed(object sender, FormClosedEventArgs e)
54 : {
55 :     // ウインド位置セーブ
56 :     SAjrReg.SaveWndPos(this);
57 :     // 設定値セーブ
58 :     SAjrReg.SaveAllCtrls(this);
59 : }
60 : //----- 接続/切断 ボタン -----//
61 : private void btnConnect_Click(object sender, EventArgs e)
62 : {
63 :     if (m_fConnect) {
64 :         sct.Disconnect();
65 :     }
66 :     else {
67 :         sct.Connect(txtServ.Text, uint.Parse(txtPort.Text));
68 :     }
69 : }
70 : //----- テキスト送信ボタン -----//
71 : private void btnSndTxt_Click(object sender, EventArgs e)
72 : {
73 :     sct.SendText(txtSndTxt.Text + "¥n");
74 : }
75 : //----- E S C 送信ボタン -----//
76 : private void btnSndEsc_Click(object sender, EventArgs e)
77 : {
78 :     sct.SendText("¥x1B" + txtSndEsc.Text);
79 : }
80 : //----- バイナリ送信ボタン -----//
81 : private void btnSndBin_Click(object sender, EventArgs e)
82 : {
83 :     string[] arr = txtSndBin.Text.Split(' ');
84 :     int      n = 0;
85 :     // 有効な数算出
86 :     for (int i = 0; i < arr.Length; i++) {
87 :         if (IsByteHexa(arr[i])) n++;
88 :     }
89 :     if (n != 0) {
90 :         //      送信バイナリ作成
91 :         Byte[] BArr = new Byte[n];
92 :
93 :         int      ix = 0;
94 :         for (int i = 0; i < arr.Length; i++) {
95 :             if (IsByteHexa(arr[i])) {BArr[ix++] = Convert.ToByte(arr[i], 16);}
96 :         }
97 :         if (ix != 0) {
98 :             sct.SendBinary(BArr);
99 :         }
100 :     }
101 : }
102 : //----- パケット送信ボタン -----//
103 : private void btnSndPkt_Click(object sender, EventArgs e)
104 : {
105 :     string[] arr = txtSndPkt.Text.Split(' ');
106 :     int      n = 0;
107 :     // 有効な数算出
108 :     for (int i = 0; i < arr.Length; i++) {
109 :         if (IsByteHexa(arr[i])) n++;

```

```

110 :         }
111 :         if (n != 0) {
112 :             //      送信バイナリ作成
113 :             Byte[] BArr = new Byte[n];
114 :
115 :             int ix = 0;
116 :             for (int i = 0; i < arr.Length; i++) {
117 :                 if (IsByteHexa(arr[i])) {BArr[ix++] = Convert.ToByte(arr[i], 16);}
118 :             }
119 :             if (ix != 0) {
120 :                 sct.SendPacket(BArr);
121 :             }
122 :         }
123 :     }
124 :     //----- 全てクリアー ボタン -----//
125 :     private void btnClearAll_Click(object sender, EventArgs e)
126 :     {
127 :         vthTxtChunk.Purge();
128 :         vthBinChunk.Purge();
129 :         vthTxt.Purge();
130 :         vthCtrl.Purge();
131 :         vthEsc.Purge();
132 :         vthInv.Purge();
133 :         vthPkt.Purge();
134 :         vthNoPkt.Purge();
135 :     }
136 :     //-----//
137 :     // SCT イベント通知 //
138 :     //-----//
139 :     //----- 接続通知 -----//
140 :     private void sct_OnConnect(object sender, EventArgs e)
141 :     {
142 :         m_fConnect = true;
143 :         btnConnect.Text = "切 断";
144 :         ShowMsgTip("サーバと接続しました。");
145 :     }
146 :     //----- 切断通知 -----//
147 :     private void sct_OnDisconnect(object sender, EventArgs e)
148 :     {
149 :         m_fConnect = false;
150 :         btnConnect.Text = "接 続";
151 :         ShowMsgTip("サーバとの回線を切断しました。");
152 :     }
153 :     //----- 接続失敗通知 -----//
154 :     private void sct_OnCnFail(object sender, SctArgCnFail e)
155 :     {
156 :         ShowMsgTip("¥x1B[31m サーバとの接続を失敗しました。");
157 :     }
158 :     //----- バイナリチャンクデータ受信通知 -----//
159 :     private void sct_OnRxChunkBin(object sender, SctArgRxChunkBin e)
160 :     {
161 :         vthBinChunk.PutText("----- Binary Chunk " + e.bin.Length.ToString() + " Bytes -----¥n");
162 :         vthBinChunk.PrintHexDump(e.bin);
163 :         vthBinChunk.PutText("¥n");
164 :     }
165 :     //----- 不正チャンクテキスト受信通知 -----//
166 :     private void sct_OnRxInvChunk(object sender, SctArgRxInvChunk e)
167 :     {
168 :         vthInv.PrintHexDump(e.bin);
169 :         vthInv.PutText("¥n");
170 :     }
171 :     //----- テキストチャンク受信通知 -----//
172 :     private void sct_OnRxChunkTxt(object sender, SctArgRxChunkTxt e)
173 :     {
174 :         vthTxtChunk.PutText(e.text);
175 :     }
176 :     //----- テキスト受信通知 -----//
177 :     private void sct_OnRxText(object sender, SctArgRxText e)
178 :     {
179 :         vthTxt.PutText(e.text + "¥n");
180 :     }
181 :     //----- E S C データ受信通知 -----//

```



```

182 : private void sct_OnRxEsc(object sender, SctArgRxEsc e)
183 : {
184 :     vthEsc.PutText("¥¥x1B" + e.esc.Substring(1));
185 :     vthEsc.PutText("¥n");
186 : }
187 : //----- 制御コード受信通知 -----//
188 : private void sct_OnRxCtrl(object sender, SctArgRxCtrl e)
189 : {
190 :     int c = (int)e.ctrl;
191 :     vthCtrl.PutFormat("0x{0:X2}¥n", c);
192 : }
193 : //----- パケットデータ受信通知 -----//
194 : private void sct_OnRxPacket(object sender, SctArgRxPacket e)
195 : {
196 :     if (e.bin != null) {
197 :         vthPkt.PrintHexDump(e.bin);
198 :         vthPkt.PutText("¥n");
199 :     }
200 : }
201 : //----- パケット外テキスト受信通知 -----//
202 : private void sct_OnRxNoPkt(object sender, SctArgRxNoPkt e)
203 : {
204 :     vthNoPkt.PutText(e.text);
205 : }
206 : //----- 受信エラー通知 -----//
207 : private void sct_OnRecvError(object sender, SctArgRecvError e)
208 : {
209 :     if (e.overlapped) SAjrTip.ShowCenter(this, "¥x1B[31m" + "受信エラー(GetOverlappedResult)");
210 :     else SAjrTip.ShowCenter(this, "¥x1B[31m" + "受信エラー(WSARecv)" );
211 : }
212 : //----- 送信エラー通知 -----//
213 : private void sct_OnSendError(object sender, SctArgSendError e)
214 : {
215 :     if (e.overlapped) SAjrTip.ShowCenter(this, "¥x1B[31m" + "送信エラー(GetOverlappedResult)");
216 :     else SAjrTip.ShowCenter(this, "¥x1B[31m" + "送信エラー(WSASend)" );
217 : }
218 : //----- その他のエラー通知 -----//
219 : private void sct_OnGeneralError(object sender, SctArgGeneralError e)
220 : {
221 :     switch (e.ErrorCode) {
222 :         case ESctErrorCode.ERR_CREEVT: SAjrTip.ShowCenter(this, "¥x1B[31m" + "CreateEvent() 失敗" ); break;
223 :         case ESctErrorCode.ERR_SOCKET: SAjrTip.ShowCenter(this, "¥x1B[31m" + "ソケット生成失敗" ); break;
224 :         case ESctErrorCode.ERR_ADDRINFO: SAjrTip.ShowCenter(this, "¥x1B[31m" + "getaddrinfo 失敗" ); break;
225 :         case ESctErrorCode.ERR_SOCKOPT: SAjrTip.ShowCenter(this, "¥x1B[31m" + "setsockopt() 失敗" ); break;
226 :         case ESctErrorCode.ERR_SUBTHREAD: SAjrTip.ShowCenter(this, "¥x1B[31m" + "サブスレッド開始失敗" ); break;
227 :         case ESctErrorCode.ERR_CRESSEP: SAjrTip.ShowCenter(this, "¥x1B[31m" + "ストリーム分離生成失敗" ); break;
228 :         case ESctErrorCode.ERR_CREMBTXD: SAjrTip.ShowCenter(this, "¥x1B[31m" + "送信データ用メールボックス生成失敗" ); break;
229 :         case ESctErrorCode.ERR_TIMEOUT: SAjrTip.ShowCenter(this, "¥x1B[31m" + "サブスレッド終了タイムアウト" ); break;
230 :     }
231 : }
232 : //----- テキスト表示ウィンドにファイルドロップ -----//
233 : private void vthTxt_OnFileDrop(object sender, VthArgFileDrop e)
234 : {
235 :     for (int i = 0; i < e.n; i++) {
236 :         string path = vthTxt.GetDroppedFile();
237 :         byte[] buf = new byte[1024];
238 :         long FileSize;
239 :         int ReadSize;
240 :         FileStream fs = new FileStream(path, FileMode.Open, FileAccess.Read);
241 :         FileSize = fs.Length;
242 :         while (FileSize >= 1024) {
243 :             ReadSize = fs.Read(buf, 0, 1024);
244 :             FileSize -= ReadSize;
245 :             sct.SendBinary(buf);
246 :         }
247 :         if (FileSize > 0) {
248 :             buf = new byte[FileSize];
249 :             fs.Read(buf, 0, (int)FileSize);
250 :             sct.SendBinary(buf);
251 :         }
252 :         fs.Dispose();
253 :     }

```

```

254 :     }
255 :     //----- 16進文字列チェック -----//
256 :     private bool IsByteHexa(string s)
257 :     {
258 :         bool    rc = false;
259 :         do {
260 :             if (string.IsNullOrEmpty(s)) break;
261 :             if (s.Length != 2) break;
262 :             if (!Uri.IsHexDigit(s[0])) break;
263 :             if (!Uri.IsHexDigit(s[1])) break;
264 :             rc = true;
265 :         } while (false);
266 :         return rc;
267 :     }
268 :     //----- 受信テキストエンコード設定ラジオボタン -----//
269 :     private void rbtRxSJis_CheckedChanged(object sender, EventArgs e) {SetTextEncode();}
270 :     private void rbtRxUTF8_CheckedChanged(object sender, EventArgs e) {SetTextEncode();}
271 :     private void rbtRxEuc_CheckedChanged (object sender, EventArgs e) {SetTextEncode();}
272 :     private void rbtRxAuto_CheckedChanged(object sender, EventArgs e) {SetTextEncode();}
273 :     //----- 送信テキストエンコード設定ラジオボタン -----//
274 :     private void rbtTxSJis_CheckedChanged(object sender, EventArgs e) {SetTextEncode();}
275 :     private void rbtTxUTF8_CheckedChanged(object sender, EventArgs e) {SetTextEncode();}
276 :     private void rbtTxEuc_CheckedChanged (object sender, EventArgs e) {SetTextEncode();}
277 :     private void rbtTxAuto_CheckedChanged(object sender, EventArgs e) {SetTextEncode();}
278 :     // テキストエンコード設定
279 :     private void SetTextEncode()
280 :     {
281 :         if      (rbtRxSJis.Checked) sct.RxTextCode = ESctRxTextCode.SJIS;
282 :         else if (rbtRxUTF8.Checked) sct.RxTextCode = ESctRxTextCode.UTF8;
283 :         else if (rbtRxEuc .Checked) sct.RxTextCode = ESctRxTextCode.EUC;
284 :         else if (rbtRxAuto.Checked) sct.RxTextCode = ESctRxTextCode.AUT0;
285 :
286 :         if      (rbtTxSJis.Checked) sct.TxTextCode = ESctTxTextCode.SJIS;
287 :         else if (rbtTxUTF8.Checked) sct.TxTextCode = ESctTxTextCode.UTF8;
288 :         else if (rbtTxEuc .Checked) sct.TxTextCode = ESctTxTextCode.EUC;
289 :         else if (rbtTxAuto.Checked) sct.TxTextCode = ESctTxTextCode.SJIS;
290 :
291 :     }
292 :     //----- メッセージチップ表示 -----//
293 :     private void ShowMsgTip(string txt)
294 :     {
295 :         Size sz = SAjrTip.GetSize(txt, null, true);
296 :         Point pt = grpRecv.ClientRectangle.Location;
297 :         pt      = grpRecv.PointToScreen(pt);
298 :         int  x = pt.X + grpRecv.ClientRectangle.Width - sz.Width;
299 :         int  y = pt.Y - sz.Height - 5;
300 :         SAjrTip.Show(x, y, txt);
301 :     }
302 : }
303 : }

```

13.6.2. Sil_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 : using System;
2 : using System.Collections.Generic;
3 : using System.ComponentModel;
4 : using System.Data;
5 : using System.Drawing;
6 : using System.Linq;
7 : using System.Text;
8 : using System.Windows.Forms;
9 : using CAjrCustCtrl;
10 :
11 : namespace Sil_SockClient2
12 : {
13 :     public partial class Form1 : Form
14 :     {
15 :         bool    m_fConnect = false;
16 :         uint    m_bytes    = 0;
17 :         string[] LoadExc = {"txtMsg"};
18 :
19 :         public Form1()
20 :         {
21 :             InitializeComponent();
22 :         }
23 :         // 起動時初期設定
24 :         private void Form1_Load(object sender, EventArgs e)
25 :         {
26 :             this.Text = this.Text + (IntPtr.Size == 4 ? " (WIN32)" : " (WIN64)");
27 :             // サイズ変更禁止
28 :             this.FormBorderStyle = FormBorderStyle.FixedSingle;
29 :             // ウインド位置ロード
30 :             SAjrReg.LoadWndPos(this);
31 :             // 設定値ロード
32 :             SAjrReg.LoadAllCtrlsExc(this, LoadExc);
33 :             // S C T初期化
34 :             sct.Init();
35 :         }
36 :         // フォーム終了
37 :         private void Form1_FormClosed(object sender, FormClosedEventArgs e)
38 :         {
39 :             // ウインド位置セーブ
40 :             SAjrReg.SaveWndPos(this);
41 :             // 設定値セーブ
42 :             SAjrReg.SaveAllCtrlsExc(this, LoadExc);
43 :         }
44 :         // 接続／切断ボタン
45 :         private void btnConnect_Click(object sender, EventArgs e)
46 :         {
47 :             if (m_fConnect) {
48 :                 sct.Disconnect();
49 :             }
50 :             else {
51 :                 m_bytes = 0;
52 :                 sct.Connect(txtServ.Text, uint.Parse(txtPort.Text));
53 :             }
54 :         }

```

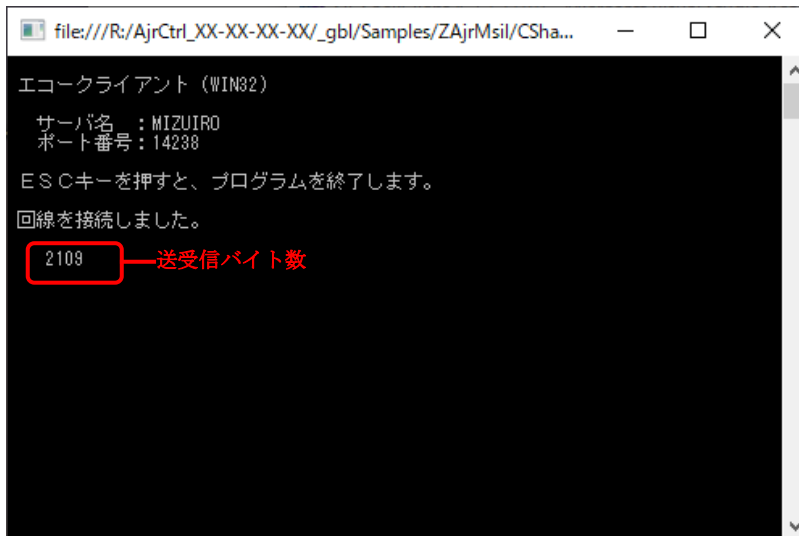
```

55 :
56 : //-----//
57 : // S C T イベント通知 //
58 : //-----//
59 : // 接続通知
60 : private void sct_OnConnect(object sender, EventArgs e)
61 : {
62 :     m_fConnect = true;
63 :     btnConnect.Text = "切 断";
64 :     txtMsg.Text = "¥n サーバと接続状態です。";
65 : }
66 : // 切断通知
67 : private void sct_OnDisconnect(object sender, EventArgs e)
68 : {
69 :     m_fConnect = false;
70 :     btnConnect.Text = "接 続";
71 :     txtMsg.Text = "¥n 接続ボタンを押して、サーバと接続してください。";
72 : }
73 : // 接続失敗通知
74 : private void sct_OnCnFail(object sender, SctArgCnFail e)
75 : {
76 :     txtMsg.Text = "接続を失敗しました。¥r¥n" +
77 :         "接続ボタンを押して、サーバと接続してください。";
78 : }
79 : // バイナリチャンク受信通知
80 : private void sct_OnRxChunkBin(object sender, SctArgRxChunkBin e)
81 : {
82 :     m_bytes += (uint)e.bin.Length;
83 :     txtBytes.Text = m_bytes.ToString();
84 :     sct.SendBinary(e.bin);
85 : }
86 : }
87 : }

```

13.6.3. Sil_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、Sil_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 : using System;
2 : using System.Collections.Generic;
3 : using System.Linq;
4 : using System.Text;
5 : using System.Runtime.InteropServices;
6 : using AjrCustCtrl;
7 :
8 : namespace Sil_SockClient2C
9 : {
10 :     class Program
11 :     {
12 :         static CAjrSockClient sct = new CAjrSockClient();
13 :         static string      ServName = "MIZUIRO";
14 :         static uint        bytes    = 0;
15 :
16 :         static void Main(string[] args)
17 :         {
18 :             ESctEvt evt;          // イベントマスク
19 :             object obj;           // イベントデータ
20 :
21 :             // コンソールキーオブジェクト生成
22 :             ConsoleKeyInfo c = new ConsoleKeyInfo();
23 :
24 :             // コンソールアプリ終了ハンドラ登録
25 :             m_CbkConApExit = new CbkConApExit(SsvConApExit);
26 :             SetConsoleCtrlHandler(m_CbkConApExit, true);
27 :
28 :             // 初期表示
29 :             Console.Clear();
30 :             Console.WriteLine("");
31 :             Console.WriteLine(" エコークライアント (" + (IntPtr.Size == 4 ? "WIN32" : "WIN64") + ")");
32 :             Console.WriteLine("");
33 :             Console.WriteLine("   サーバ名   : " + ServName);
34 :             Console.WriteLine("   ポート番号 : 14238");
35 :             Console.WriteLine("");
36 :             Console.WriteLine(" ESCキーを押すと、プログラムを終了します。");
37 :             Console.WriteLine("");
38 :             // 実行モード設定 (イベントを「WaitEvent」で待つ)
39 :             sct._SctMode = ESctMode.WaitingForEvent;
40 :             // 対象とするイベントを設定
41 :             sct.SetEvtMask(ESctEvt.EV_CONNECT | ESctEvt.EV_DISCONNECT | ESctEvt.EV_CNFAIL | ESctEvt.EV_RXCHUNK);
42 :             // 接続
43 :             sct.Connect(ServName, 14238);
44 :             // ループ
45 :             while (true) {

```

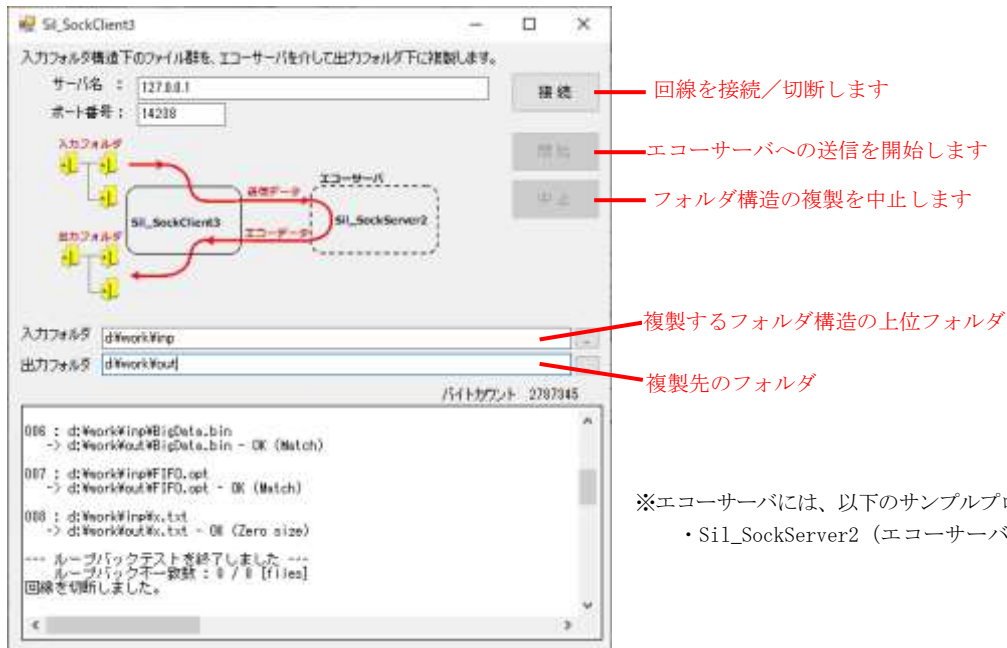
```

46 :         // E S C キー入力で回線切断
47 :         if (c.Key == ConsoleKey.Escape) {
48 :             sct.Disconnect();
49 :         }
50 :         // イベント待ち
51 :         obj = null;
52 :         if ((evt = sct.WaitEvent(out obj, 1000)) != ESctEvt.EV_NOEVENT) {
53 :             // 各イベント処理
54 :             // ●接続通知
55 :             if ((evt & ESctEvt.EV_CONNECT) != 0) {
56 :                 Console.WriteLine(" 回線を接続しました。¥n");
57 :             }
58 :             // ●切断通知
59 :             if ((evt & ESctEvt.EV_DISCONNECT) != 0) {
60 :                 break;
61 :             }
62 :             // ●接続失敗通知
63 :             if ((evt & ESctEvt.EV_CNFAIL) != 0) {
64 :                 Console.Write(" 接続を失敗しました。 ENTER キーを押してください。");
65 :                 Console.ReadLine();
66 :                 break;
67 :             }
68 :             // ●バイナリチャンク受信通知
69 :             if ((evt & ESctEvt.EV_RXCHUNK) != 0) {
70 :                 Byte[] rxd = (Byte[])obj;
71 :                 // 受信バイト数を表示
72 :                 bytes += (uint)rx.Length;
73 :                 Console.Write(" " + bytes.ToString() + "¥r");
74 :                 // 受信データを返信
75 :                 sct.SendBinary((byte[])obj);
76 :             }
77 :         }
78 :         // キー入力
79 :         if (Console.KeyAvailable) c = Console.ReadKey(true);
80 :     }
81 :     // ソケットクライアント消去
82 :     sct.Delete();
83 : }
84 : // コンソールアプリ終了ハンドラ用デリゲート
85 : [DllImport("Kernel32")]
86 : static extern bool SetConsoleCtrlHandler(CbkConApExit Handler, bool Add);
87 : delegate bool CbkConApExit(EAJCEXITTYPE ExitType);
88 : static CbkConApExit m_CbkConApExit;
89 : // コンソールアプリ終了ハンドラ
90 : static bool SsvConApExit(EAJCEXITTYPE ExitType)
91 : {
92 :     // ソケットクライアント消去
93 :     sct.Delete();
94 :     // false : 次のイベントハンドラへリンクする
95 :     return false;
96 : }
97 : }
98 : }

```

13.6.4. Sil_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。



```

1 : using System;
2 : using System.Collections.Generic;
3 : using System.ComponentModel;
4 : using System.Data;
5 : using System.Drawing;
6 : using System.Linq;
7 : using System.Text;
8 : using System.Windows.Forms;
9 : using System.IO;
10 : using System.Collections;
11 : using CAjrCustCtrl;
12 :
13 : namespace Sil_SockClient3
14 : {
15 :     public partial class Form1 : Form
16 :     {
17 :         bool        m_fConnect = false;        // 接続状態
18 :         string       m_InpDir;                  // 入力フォルダパス
19 :         string       m_OutDir;                  // 出力フォルダパス
20 :         string       m_InpPath;                 // 入力ファイルパス
21 :         string       m_OutPath;                 // 出力ファイルパス
22 :         FileStream    m_InpFs;                  // 入力ファイルストリーム
23 :         FileStream    m_OutFs;                  // 出力ファイルストリーム
24 :         long          m_FileSize;               // 入力ファイルサイズ
25 :         int           m_Unmatch;               // 比較不一致ファイル数
26 :         int           m_ListCount;             // 全ファイル数
27 :         int           m_ListIx;                // ファイルインデクス
28 :         ArrayList     m_list = new ArrayList(); // 全入出力ファイル リスト
29 :         int           m_BufSize = 1024;        // 送信データバッファサイズ
30 :         int           m_Bytes = 0;             // 送信バイトカウント
31 :         bool          m_fSendBusy = false;     // 送信中フラグ
32 :
33 :         public Form1()
34 :         {
35 :             InitializeComponent();
36 :         }
37 :         // フォーム開始
38 :         private void Form1_Load(object sender, EventArgs e)

```

```

39 :     {
40 :         // ビットマップ表示
41 :         System.Reflection.Assembly myAssembly = System.Reflection.Assembly.GetExecutingAssembly();
42 :         Bitmap bmp = new Bitmap(myAssembly.GetManifestResourceStream("Sil_SockClient3.ProgImage.bmp"));
43 :         bmp.MakeTransparent();
44 :         pic.Image = bmp;
45 :         // ウィンド位置ロード
46 :         SAjrReg.LoadWndPos(this);
47 :         // 設定値ロード
48 :         SAjrReg.LoadAllCtrls(this);
49 :         // S C P 初期化
50 :         sct.Init();
51 :     }
52 :     // フォーム終了
53 :     private void Form1_FormClosed(object sender, FormClosedEventArgs e)
54 :     {
55 :         // ウィンド位置セーブ
56 :         SAjrReg.SaveWndPos(this);
57 :         // 設定値セーブ
58 :         SAjrReg.SaveAllCtrls(this);
59 :     }
60 :     // 入力パス設定ボタン
61 :     private void btnInpPath_Click(object sender, EventArgs e)
62 :     {
63 :         string path;
64 :         if ((path = SAjrGsr.GetFolder("入力フォルダ", "")) != "") {
65 :             txtInpPath.Text = path;
66 :         }
67 :     }
68 :     // 出力パス設定ボタン
69 :     private void btnOutPath_Click(object sender, EventArgs e)
70 :     {
71 :         string path;
72 :         if ((path = SAjrGsr.GetFolder("出力フォルダ", "")) != "") {
73 :             txtOutPath.Text = path;
74 :         }
75 :     }
76 :     // 接続/切断ボタン
77 :     private void btnConnect_Click(object sender, EventArgs e)
78 :     {
79 :         // 接続ボタン禁止
80 :         btnConnect.Enabled = false;
81 :         // 接続/切断
82 :         if (m_fConnect) {
83 :             sct.Disconnect();
84 :         }
85 :         else {
86 :             m_Bytes = 0;
87 :             sct.Connect(txtServ.Text, uint.Parse(txtPort.Text));
88 :         }
89 :     }
90 :     // 開始ボタン
91 :     private void btnStart_Click(object sender, EventArgs e)
92 :     {
93 :         // 入出力パス設定
94 :         m_InpDir = txtInpPath.Text;
95 :         m_OutDir = txtOutPath.Text;
96 :         if (m_InpDir != "" && m_OutDir != "") {
97 :             if (m_InpDir != m_OutDir) {
98 :                 // 入出力パスリストクリアー
99 :                 m_list.Clear();
100 :                 // ファイル検索 (入出力パスリスト作成)
101 :                 m_ListCount = 0;
102 :                 m_ListIx = 0;
103 :                 m_Unmatch = 0;
104 :                 fsr.SearchFiles(m_InpDir, "*.*", true);
105 :                 // フォルダ構造コピー
106 :                 if (m_ListCount != 0) {
107 :                     // 出力フォルダ下の全ファイル削除
108 :                     if (MessageBox.Show(m_OutDir + "下の全ファイルを削除します。よろしいですか?", "S_SerialComPort4",
109 :                         MessageBoxButtons.YesNo, MessageBoxIcon.Exclamation) == DialogResult.Yes) {
110 :                         // バイトカウンタクリアー

```



```

111 :             m_Bytes = 0;
112 :             lblBytes.Text = "0";
113 :             // ログクリアー
114 :             vthLog.Purge();
115 :             // ボタングレー化
116 :             EnableButtons(false, false, true);
117 :             // フォルダ下クリアー
118 :             SAjrFop.CleanFolder(m_OutDir);
119 :             // フォルダ構造コピー
120 :             SAjrFop.CopyFolderStruct(m_InpDir, m_OutDir);
121 :             // ファイルをオープンし送信開始
122 :             SendNextFile();
123 :         }
124 :     } else SAjrTip.ShowCenter(this, "¥x1B[31m" + "入力フォルダ下にファイルがありません。");
125 :
126 :     } else SAjrTip.ShowCenter(this, "¥x1B[31m" + "入出力フォルダが同じです。");
127 :
128 :     } else SAjrTip.ShowCenter(this, "¥x1B[31m" + "入出力フォルダが設定されていません。");
129 : }
130 : // 中止ボタン
131 : private void btnStop_Click(object sender, EventArgs e)
132 : {
133 :     // 送信中フラグクリアー
134 :     m_fSendBusy = false;
135 :     // 回線切断
136 :     sct.Disconnect();
137 :     // ファイルストリーム解放
138 :     if (m_InpFs != null) {m_InpFs.Dispose(); m_InpFs = null;}
139 :     if (m_OutFs != null) {m_OutFs.Dispose(); m_OutFs = null;}
140 :     vthLog.PutText("¥n¥n--- ループバックテストをキャンセルしました ---¥n");
141 :     // 全ボタン禁止
142 :     EnableButtons(false, false, false);
143 : }
144 : // ファイル検索通知
145 : private bool fsr_OnFindFile_1(object sender, FsrArgFindFile e)
146 : {
147 :     if (e.FileAtt != EFileAtt._A_SUBDIR) {
148 :         // 入出力ファイルパス設定 (セミコロン;)で区切る)
149 :         string dtail, outpath;
150 :         dtail = Path.GetDirectoryName(e.PathName);
151 :         dtail = SAjrGsr.PathCat(dtail, "");
152 :         dtail = dtail.Substring(m_InpDir.Length);
153 :         outpath = SAjrGsr.PathCat(m_OutDir + dtail, e.FileName);
154 :         m_list.Add(e.PathName + ";" + outpath);
155 :         m_ListCount++;
156 :     }
157 :     return true;
158 : }
159 : //-----//
160 : // S C T イベント //
161 : //-----//
162 : // 接続通知
163 : private void sct_OnConnect(object sender, EventArgs e)
164 : {
165 :     // 接続状態の旨、フラグ設定
166 :     m_fConnect = true;
167 :     // ボタンフェース変更
168 :     btnConnect.Text = "切 断";
169 :     vthLog.PutText("サーバと接続しました¥n");
170 :     // 接続ボタン禁止, 開始/中止ボタン許可
171 :     EnableButtons(false, true, true);
172 : }
173 : // 切断通知
174 : private void sct_OnDisconnect(object sender, EventArgs e)
175 : {
176 :     // 動作中の切断ならば、中止ボタン押下
177 :     if (m_fSendBusy) {
178 :         btnStop.PerformClick();
179 :     }
180 :     // 切断状態の旨、フラグ設定
181 :     m_fConnect = false;
182 :     // ボタンフェース変更

```

```

183 :         btnConnect.Text = "接 続";
184 :         vthLog.PutText("回線を切断しました。¥n");
185 :         // 接続ボタン許可, 開始/中止ボタン禁止
186 :         EnableButtons(true, false, false);
187 :     }
188 :     // 接続失敗通知
189 :     private void sct_OnCnFail(object sender, SctArgCnFail e)
190 :     {
191 :         vthLog.PutText("接続を失敗しました。¥n");
192 :         // 接続ボタン許可, 開始/中止ボタン禁止
193 :         EnableButtons(true, false, false);
194 :     }
195 :     // パケット受信通知
196 :     private void sct_OnRxPacket(object sender, SctArgRxPacket e)
197 :     {
198 :         // 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
199 :         if (e.bin != null) {
200 :             if (m_OutFs != null) {
201 :                 m_OutFs.Write(e.bin, 0, e.bin.Length);
202 :             }
203 :         }
204 :         // 空パケット (ファイル終端) ならば、次のファイル送信へ・・・
205 :         else {
206 :             // 出力ファイルクローズ
207 :             if (m_OutFs != null) {m_OutFs.Dispose(); m_OutFs = null;}
208 :             // ファイルコンペア
209 :             if (SAjrFop.FileCompare(m_InpPath, m_OutPath)) {
210 :                 vthLog.PutText("OK (Match)¥n");
211 :             }
212 :             else {
213 :                 m_Unmatch++;
214 :                 vthLog.PutText("¥x1B[31mNG (Unmatch)¥x1B[0m¥n");
215 :             }
216 :             // 次のファイル送信開始
217 :             SendNextFile();
218 :         }
219 :     }
220 :     // 送信完了通知
221 :     private void sct_OnTxEmpty(object sender, EventArgs e)
222 :     {
223 :         // 次のファイルデータ送信
224 :         if (m_InpFs != null) {
225 :             FileSend();
226 :         }
227 :     }
228 :     //-----//
229 :     // サブファンクション //
230 :     //-----//
231 :     // 次のファイル送信開始
232 :     private bool SendNextFile()
233 :     {
234 :         bool rc = false;
235 :         if (FileOpen()) {
236 :             m_fSendBusy = true;
237 :             FileSend();
238 :             rc = true;
239 :         }
240 :         else {
241 :             m_fSendBusy = false;
242 :             sct.Disconnect();
243 :             vthLog.PutText("¥n--- ループバックテストを終了しました ---¥n");
244 :             vthLog.PutText("    ループバック不一致数 : " + m_Unmatch.ToString() + " / " +
245 :                 m_ListCount.ToString() + " [files]¥n");
246 :             // 接続ボタン許可, 開始/中止ボタン禁止
247 :             EnableButtons(false, false, false);
248 :         }
249 :         return rc;
250 :     }
251 :     // 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:ゼロサイズファイル)
252 :     private bool FileOpen()
253 :     {
254 :         bool rc = false;

```

```

255 :         while (m_ListIx < m_ListCount) {
256 :             // 入出力パス名設定
257 :             Object obj = m_list[m_ListIx++];
258 :             string pathes = (string)obj;
259 :             string[] s = pathes.Split(';');
260 :             m_InpPath = s[0];
261 :             m_OutPath = s[1];
262 :             vthLog.PutText("Yn" + m_ListIx.ToString("D3") + " : " + m_InpPath + "Yn");
263 :             vthLog.PutText("  -> " + m_OutPath + " - ");
264 :             // 入出力ファイルオープン
265 :             m_InpFs = new FileStream(m_InpPath, FileMode.Open);
266 :             m_OutFs = new FileStream(m_OutPath, FileMode.Create);
267 :             // ファイルサイズ設定
268 :             m_FileSize = m_InpFs.Length;
269 :             // ゼロサイズならば、ファイルクローズ
270 :             if (m_FileSize == 0) {
271 :                 vthLog.PutText("OK (Zero size)Yn");
272 :                 if (m_InpFs != null) {m_InpFs.Dispose(); m_InpFs = null;}
273 :                 if (m_OutFs != null) {m_OutFs.Dispose(); m_OutFs = null;}
274 :             }
275 :             // ゼロサイズ以外ならば、ファールオープンした旨を返す
276 :             else {
277 :                 rc = true;
278 :                 break;
279 :             }
280 :         }
281 :         return rc;
282 :     }
283 :     // ファイル送信
284 :     private void FileSend()
285 :     {
286 :         // バッファサイズを超える残データ有ならば、データ送信し、バイト数減算
287 :         if (m_FileSize > m_BufSize) {
288 :             byte[] buf = new byte[m_BufSize];
289 :             m_InpFs.Read(buf, 0, m_BufSize);
290 :             m_Bytes += sct.SendPacket(buf);
291 :             m_FileSize -= m_BufSize;
292 :         }
293 :         // バッファサイズ以下の残データ有ならば、ファイル末尾データ送信
294 :         else if (m_FileSize != 0) {
295 :             byte[] buf = new byte[m_FileSize];
296 :             m_InpFs.Read(buf, 0, (int)m_FileSize);
297 :             m_Bytes += sct.SendPacket(buf);
298 :             m_FileSize = 0;
299 :         }
300 :         // 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
301 :         else {
302 :             // 空パケット送信
303 :             m_Bytes += sct.SendPacket(null);
304 :             // 入力ファイルクローズ
305 :             if (m_InpFs != null) {m_InpFs.Dispose(); m_InpFs = null;}
306 :         }
307 :         // 送信バイトカウント表示
308 :         lblBytes.Text = SAjrGsr.SepDecimal(m_Bytes.ToString());
309 :     }
310 :     // ボタン群許可／禁止
311 :     private void EnableButtons(bool fConnect, bool fStart, bool fStop)
312 :     {
313 :         btnConnect.Enabled = fConnect; // 接続ボタン
314 :         btnStart.Enabled = fStart; // 開始ボタン
315 :         btnStop.Enabled = fStop; // 中止ボタン
316 :     }
317 : }
318 : }

```

14. ファイル検索 (CAjrFileSearch.dll)

特定のフォルダ下、あるいは、マイコンピュータ内からファイルを検索します。
検索条件 (ワイルドカード) を複数指定可能です。

14.1. メソッド

プロファイルアクセスのメソッド一覧を示します。

#	メソッド名	内容	備考
1	SetNtcSearchingDir	ファイル名検索時の検索フォルダの通知設定	
2	SearchFiles	フォルダ下のファイル検索	
3	SearchMyComputer	マイコンピュータ内のファイル検索	

コールバックに関する注意

SearchFiles ()/SearchMyComputer () で、cb (ファイル検索通知用コールバックメソッド) を指定する場合は、当該メソッドをデリゲート変数に格納し、格納した変数を指定するようにしてください。(ガーベージコレクションでデリゲート自体が移動したり削除されないようにする必要があります)

```

・FsrCbkJndFile cb = new FsrCbkJndFile(cbFind);
  fs.SearchFiles(@"d:¥temp", "*.bmp;*.png", true, (IntPtr)0, cb); ----- OK

・fs.SearchFiles(@"d:¥temp", "*.bmp;*.png", true, (IntPtr)0, new FsrCbkJndFile(cbFind)); --- NG

```

14.1.1. ファイル名検索時の検索フォルダの通知設定 (SetNtcSearchingDir)

形 式 : void SetNtcSearchingDir(bool fNotify);

引 数 : fNotify - ファイル名検索時の検索フォルダの通知フラグ (true:通知, false : 非通知)

説 明 : ファイル検索 (SearchFiles, SearchMyComputer) 実行時、検索中のフォルダ名を通知するか否かを設定します。
fNotify=true を指定した場合、SearchFiles(), SearchMyComputer() メソッドにおいて、OnFindFile イベントで検索中のフォルダ名を通知します。
fNotify=false とした場合は、検索中のフォルダ名は通知せずに、見つかったファイルだけを通知します。

戻り値 : なし

14.1.2. フォルダ下のファイル検索(SearchFiles)

形 式 : `int SearchFiles(string dir, string wild, bool fSubDir);`
`int SearchFiles(string dir, string wild, bool fSubDir, IntPtr cbp);`
`int SearchFiles(string dir, string wild, bool fSubDir, IntPtr cbp, FsrCbkJFindFile cb);`

引 数 : `dir` - 検索するフォルダパス
`wild` - 検索条件 (ワイルドカード, 複数指定時はセミコロン(;))で区切る)
`fSubDir` - サブフォルダの検索フラグ (true:サブフォルダも検索する, false:サブフォルダは検索しない)
`cbp` - コールバックパラメタ (OnFindFile イベント時の引数)
`cb` - ファイル検索通知用コールバックメソッド

説 明 : 指定フォルダ下のファイルを検索します。
見つかったファイルは、OnFindFile イベント／FsrCbkJFindFile デリゲートにて通知します。

戻り値 : 見つかったファイルの個数

備 考 : 「wild」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、`dir=@"d:\work"`, `wild=@"¥Sub1¥Sub2¥*.txt"` と指定した場合、「`d:\work¥ ¥ ¥ ¥Sub1¥Sub2¥`」フォルダ直下の「*.txt」で示されるファイルが検索されます。(「¥ ¥ ¥」は 0 個以上の任意のサブフォルダを意味します)

14.1.3. マイコンピュータ内のファイル検索(SearchMyCompute)

形 式 : `int SearchMyComputer(string path, string wild, bool fRemote);`
`int SearchMyComputer(string path, string wild, bool fRemote, IntPtr cbp);`
`int SearchMyComputer(string path, string wild, bool fRemote, IntPtr cbp, FsrCbkJFindFile cb);`

引 数 : `dir` - 検索するフォルダパス (マイコンピュータ内の全ドライブで共通, null 指定時は全フォルダ検索)
`wild` - 検索条件 (ワイルドカード, 複数指定時はセミコロン(;))で区切る)
`fRemote` - ネットワークドライブの検索フラグ (true:ネットワークドライブも検索する, false:検索しない)
`cbp` - コールバックパラメタ (OnFindFile イベント時の引数)
`cb` - ファイル検索通知コールバックメソッド

説 明 : マイコンピュータ内のファイルを検索します。
`dir` でフォルダ名を指定した場合は、当該フォルダ下のファイルを検索します。
`dir` で指定したフォルダが複数ある場合は、その全てのフォルダが検索対象となります。
見つかったファイルは、OnFindFile イベント／FsrCbkJFindFile デリゲートにて通知します。

戻り値 : 見つかったファイルの個数

備 考 : 「wild」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、`dir=@"work"`, `wild=@"¥Sub1¥Sub2¥*.txt"` と指定した場合、「`work¥ ¥ ¥ ¥Sub1¥Sub2¥`」フォルダ直下の「*.txt」で示されるファイルが検索されます。(「¥ ¥ ¥」は 0 個以上の任意のサブフォルダを意味します)

14.2. イベント／デリゲート

ファイル検索のイベントを示します。

コンソールアプリの場合は、イベントが発生しない為、デリゲートを介してコールバックメソッドにより通知を受け取ります。
以降の説明中「パラメタ」はイベントのパラメタ形式を示します。デリゲートの場合は先頭の「e.」を削除した形式となります。

14.2.1. ファイル検索通知 (OnFindFile)

形 式 : `bool OnFindFile(object sender, GsrArgFindFile e);`
`delegate bool FsrCbKFindFile( int nest,`
`string PathName,`
`string FileName,`
`EFileAtt FileAtt,`
`uint FTime,`
`IntPtr cbp );`

パラメタ :

<code>int e.nest</code>	- サブディレクトリの深さ (0～) / 検索中のフォルダパス通知識別 (-1)
<code>string e.PathName</code>	- <code>e.nest >= 0</code> の場合は、見つかったファイルのパス名 <code>e.nest == -1</code> の場合は、検索中のフォルダパス名
<code>string e.FileName</code>	- 見つかったファイルの名称 (ドライブやフォルダ名を除いたファイル名部分のみ)
<code>EFileAtt e.FileAtt</code>	- ファイル属性
<code>uint e.FTime</code>	- ファイルタイム (ファイルの最終更新日時, 1970/1/1 00:00:00 からの通算秒数)
<code>IntPtr e.cbp</code>	- コールバックパラメタ (SearchFiles, SearchMyComputer コール時に指定した値)

説 明 : ファイル検索 (SearchFiles, SearchMyComputer) 実行時、検索中のフォルダ名を通知するか否かを設定します。
`e.nest == -1` の場合は、検索中のフォルダ・パス名の通知を意味します。
 検索中のフォルダパス名は、SetNtcSearchingDir() メソッドで、fNotify=true を指定した場合のみ通知されます。
 検索中のフォルダパス名通知の場合、e.FileName, e.FileAtt および e.FileTime は設定されません。

`e.nest ≥ 0` の場合は、見つかったファイル・パス名の通知を意味します。

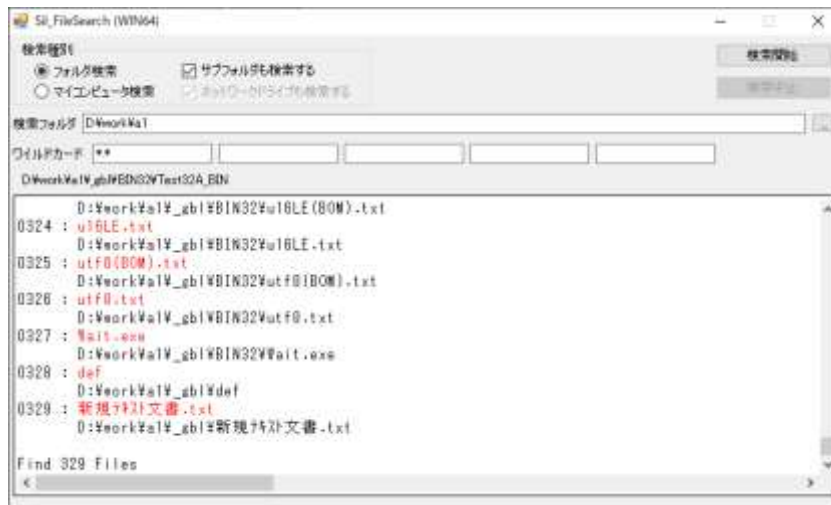
戻り値 :

<code>true</code>	- ファイルの検索を継続する
<code>false</code>	- ファイルの検索を中止する

14.3. サンプルプログラム

14.3.1. Sil_FileSearch (ファイル検索 - Windows アプリ)

このサンプルプログラムは、Windows アプリによるファイル検索の例です。



```

1 : //
2 : // Sil_FileSearch
3 : //
4 : using System;
5 : using System.Collections.Generic;
6 : using System.ComponentModel;
7 : using System.Data;
8 : using System.Drawing;
9 : using System.Text;
10 : using System.Windows.Forms;
11 : using AjrCustCtrl;
12 :
13 : namespace Sil_FileSearch
14 : {
15 :     public partial class Form1 : Form
16 :     {
17 :         int    Count;
18 :         bool   fCont;
19 :
20 :         public Form1()
21 :         {
22 :             InitializeComponent();
23 :         }
24 :         //----- 起動時初期設定 -----//
25 :         private void Form1_Load(object sender, EventArgs e)
26 :         {
27 :             this.FormBorderStyle = FormBorderStyle.FixedSingle;
28 :             this.Text = this.Text + (IntPtr.Size == 4 ? " (WIN32)" : " (WIN64)");
29 :             lblSrhDir.Text = "";
30 :             // 設定値ロード
31 :             SAjrReg.LoadAllCtrls(this);
32 :             //----- 検索中のフォルダ通知設定 -----//
33 :             fsr.SetNtcSearchingDir(true);
34 :         }
35 :         //----- 終了時の処理 -----//
36 :         private void Form1_FormClosing(object sender, FormClosingEventArgs e)
37 :         {
38 :             fCont = false;
39 :             // 設定値セーブ
40 :             SAjrReg.SaveAllCtrls(this);
41 :         }
42 :         //----- フォルダ検索 ラジオボタン -----//
43 :         private void rbtFolder_CheckedChanged(object sender, EventArgs e)
44 :         {
45 :             if (rbtFolder.Checked) {
46 :                 chkSubDir.Enabled = true;
47 :                 chkNwDrive.Enabled = false;
48 :                 lblDir.Enabled = true;

```

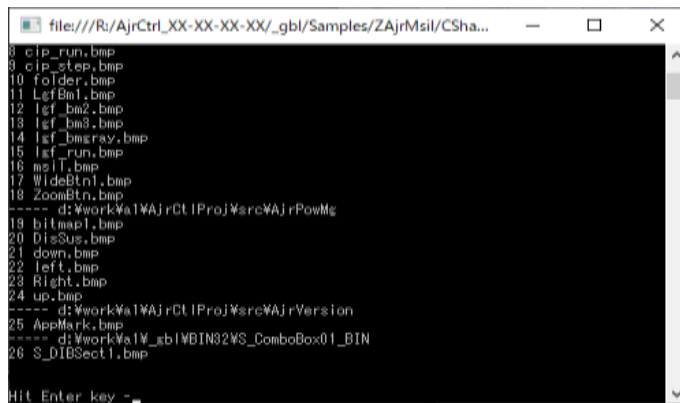
```

49 :         txtDir.Enabled = true;
50 :         cmdDir.Enabled = true;
51 :     }
52 : }
53 : //----- マイコンピュータ検索 ラジオボタン -----//
54 : private void rbtMyComp_CheckedChanged(object sender, EventArgs e)
55 : {
56 :     if (rbtMyComp.Checked) {
57 :         chkSubDir.Enabled = false;
58 :         chkNwDrive.Enabled = true;
59 :         lblDir.Enabled = false;
60 :         txtDir.Enabled = false;
61 :         cmdDir.Enabled = false;
62 :     }
63 : }
64 : //----- フォルダ設定ボタン -----//
65 : private void cmdDir_Click(object sender, EventArgs e)
66 : {
67 :     string dir = SAjrGsr.GetFolder("検索フォルダ設定", "");
68 :     if (dir != "") {
69 :         txtDir.Text = dir;
70 :     }
71 : }
72 : //----- 検索開始ボタン -----//
73 : private void cmdStart_Click(object sender, EventArgs e)
74 : {
75 :     string wild = "";
76 :     int n;
77 :
78 :     vth.Purge();
79 :     if (txtWild0.Text != "") wild = wild + txtWild0.Text;
80 :     if (txtWild1.Text != "") wild = wild + (wild != "" ? ";" : "") + txtWild1.Text;
81 :     if (txtWild2.Text != "") wild = wild + (wild != "" ? ";" : "") + txtWild2.Text;
82 :     if (txtWild3.Text != "") wild = wild + (wild != "" ? ";" : "") + txtWild3.Text;
83 :     if (txtWild4.Text != "") wild = wild + (wild != "" ? ";" : "") + txtWild4.Text;
84 :     cmdStart.Enabled = false;
85 :     cmdStop.Enabled = true;
86 :     Count = 0;
87 :     fCont = true;
88 :     if (rbtFolder.Checked) n = fsr.SearchFiles(txtDir.Text, wild, chkSubDir.Checked);
89 :     else n = fsr.SearchMyComputer(null, wild, chkNwDrive.Checked);
90 :     vth.PutText("¥nFind " + n.ToString() + " Files");
91 :     cmdStart.Enabled = true;
92 :     cmdStop.Enabled = false;
93 : }
94 : //----- 検索中止ボタン -----//
95 : private void cmdStop_Click(object sender, EventArgs e)
96 : {
97 :     fCont = false;
98 : }
99 : //----- ファイル検索通知 -----//
100 : private bool fsr_OnFindFile(object sender, FsrArgFindFile e)
101 : {
102 :     if (e.nest != -1) {
103 :         Count++;
104 :         vth.PutText(Count.ToString("D4") + " : ¥x1B[31m" + e.FileName + "¥n" + "¥x1B[0m" + " " + e.PathName + "¥n");
105 :     }
106 :     else {
107 :         lblSrhDir.Text = e.PathName;
108 :     }
109 :
110 :     System.Windows.Forms.Application.DoEvents();
111 :
112 :     return fCont; // true:検索継続, false:検索中止
113 : }
114 : }
115 : }

```


14.3.2. Sil_FileSearchC (ファイル検索 - コンソールアプリ)

このサンプルプログラムは、コンソールアプリによるファイル検索の例です。
検索フォルダやワイルドカードは、プログラム内にハードコードしています。



```

1 : using System;
2 : using System.Collections.Generic;
3 : using System.Linq;
4 : using System.Text;
5 : using CAjrCustCtrl;
6 :
7 : namespace Sil_FileSearchC
8 : {
9 :     class Program
10 :    {
11 :        static CAjrFileSearch fs = new CAjrFileSearch(); // ファイル検索オブジェクト生成
12 :        static ConsoleKeyInfo c = new ConsoleKeyInfo(); // コンソールキー情報オブジェクト生成
13 :        static string dir = null;
14 :        static uint Count = 0;
15 :
16 :        static void Main(string[] args)
17 :        {
18 :            string path = @"d:\\";
19 :            string wild = "*.bmp;*.png";
20 :            Count = 0;
21 :            Console.WriteLine("/パス(" + path + ")からファイル(" + wild + ")を検索します。");
22 :            Console.WriteLine("ENTER キーを押すと開始します。 ESC キーを押すと検索を終了します。 \n");
23 :            Console.ReadLine();
24 :            // ファイル検索実行
25 :            FsrCbFindFile cb = new FsrCbFindFile(cbFind);
26 :            fs.SetNtcSearchingDir(true);
27 :            fs.SearchFiles(path, wild, true, (IntPtr)0, cb);
28 :            // キー入力待ち
29 :            Console.WriteLine("\n\nHit Enter key -");
30 :            Console.ReadLine();
31 :        }
32 :        // ファイル検索通知コールバック
33 :        static private bool cbFind(int nest, string path, string file, EFileAtt att, uint ftime, IntPtr cbp)
34 :        {
35 :            // 検索中のフォルダパス通知
36 :            if (nest == -1) {
37 :                dir = path;
38 :            }
39 :            // 見つかったファイルパス通知
40 :            else {
41 :                if (dir != null) {
42 :                    Console.WriteLine("----- " + dir);
43 :                    dir = null;
44 :                }
45 :                Count++;
46 :                Console.WriteLine(Count.ToString() + " " + file);
47 :            }
48 :            // E S Cキー押下ならば検索中止
49 :            if (Console.KeyAvailable) c = Console.ReadKey(true);
50 :            return (c.Key != ConsoleKey.Escape) ? true : false;
51 :        }
52 :    }
53 : }

```

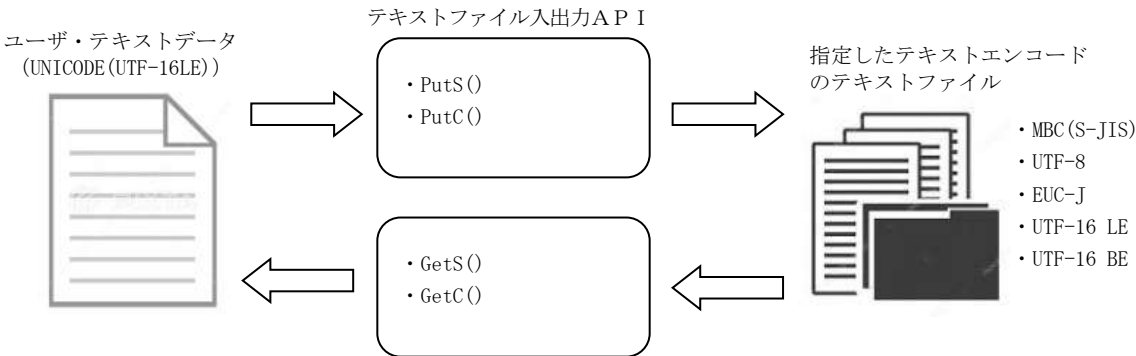
15. テキストファイル・アクセス (CAjrTextFile.dll)

テキストファイルのエンコードを指定し、テキストファイルの入出力を行います。
指定可能なテキストエンコードは、以下の通りです。

- ・マルチバイト (MBC(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)

「MBC(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、UTF-16LEのテキストを扱います。
つまり、各APIでは、読み出したデータや書き込むデータは通常の文字列 (string) となります。



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)に変換されます。
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文字に変換されます。

マルチバイト文字／サロゲートペア

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、上位サロゲートである場合、この半端な文字は退避され、次の書き込みテキストと連結します。

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エンコードを表す符号 (2～3バイト)。
BOMの実際のコードは、以下のとおりです。

テキストエンコード	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

15.1. プロパティ

テキストファイル・アクセスのプロパティ一覧を示します。

#	名称	タイプ	内容	デフォルト
1	TextEncodeAtRead	ETextEncode	入力ファイルのテキストエンコード ※1 (TextEncode を指定しない Open() メソッドで有効)	TEC_MBC
2	TextEncodeAtWrite	ETextEncode	出力ファイルのテキストエンコード (TextEncode を指定しない Create() メソッドで有効)	TEC_MBC
3	BOM	EBomMode	ファイル出力時の BOM書き込み (BomMode を指定しない Create() メソッドで有効)	NOT_WRITE_BOM
4	TextLfConvAtRead	ETextLfConv	テキストファイル読み出し時の改行コード変換	CRLF_TO_LF
4	TextLfConvAtWrite	ETextLfConv	テキストファイル書き込み時の改行コード変換	LF_TO_CRLF

※1 : TextEncodeAtRead プロパティは、設定時と取得時で値が異なる場合があります。

Open() メソッドで ETextEncode.TEC_AUTO を指定した場合や、入力ファイルに BOMがある場合は、テキストファイルの読み出しで、実際に読み出しに使用されたテキストエンコード値が取得されます。

つまり、Open() メソッド実行後に、TextEncodeAtRead プロパティを読み出すことにより、実際に読み出し時に決定したテキストエンコードを取得できます。

Open() メソッドを実行していない場合は、単に設定した値（あるいは、デフォルト値）を返します。

15.2. メソッド

テキストファイル・アクセスのメソッド一覧を示します。

#	機能	関数名	備考
1	入力テキストファイルオープン	Open	
2	文字列入力	GetS	
3	1文字入力	GetC	
4	ファイル読み出し位置退避	SavePoint	
5	ファイル読み出し位置回復	RecvPoint	
6	ファイル読み出し位置取得	GetPoint	
7	ファイル読み出し位置設定	SetPoint	
8	ファイル読み出しバイト位置取得	GetBytePoint	
9	入力ファイルの EOFチェック	IsEof	
10	出力テキストファイル生成	Create / Append	
11	ファイルを追記モードでオープン/生成	Append	
12	文字列出力	PutS	
13	1文字出力	PutC	
14	書式文字列出力	PutFormat	
15	出力データフラッシュ	Flush	
16	テキストファイルクローズ	Close	

例 外 : `InvalidOperationException` - ファイルはオープンされていない

15.2.3. 1文字入力 (GetC)

形 式 : char GetC ();

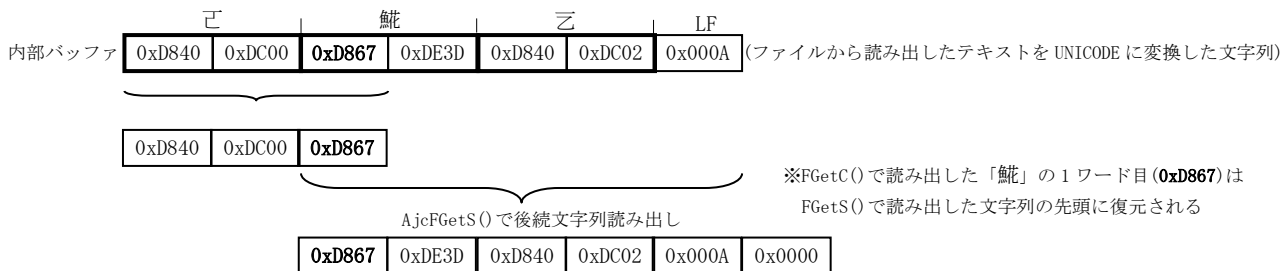
引 数 : なし

説 明 : テキストファイルから1文字読み出します。
サロゲートペア文字を読み出す場合は、本メソッドを2回実行します。

戻り値 : ≠0xFFFF : 読み出した文字コード
=0xFFFF : EOF

例 外 : InvalidOperationException - ファイルはオープンされていない

注 意 : AjcFGetC()で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が取得されます。



15.2.4. ファイル読み出しポイント退避 (SavePoint)

形 式 : long SavePoint ();

引 数 : なし

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。
RecvPoint()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 - 退避したファイルポイント (バイト位置/文字位置)
=-1 - 失敗

例 外 : InvalidOperationException - ファイルはオープンされていない

15.2.5. ファイル読み出しポイント回復 (RecvPoint)

形 式 : void RecvPoint ();

引 数 : なし

説 明 : SavePoint()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

例 外 : InvalidOperationException - ファイルはオープンされていない

15.2.6. ファイル読み出し位置取得 (GetPoint)

形 式 : long GetPoint ();

引 数 : なし

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。
SetPoint () で、読み出し位置を設定し、再度、読み出し位置から読み出すことができます。

戻り値 : ≠-1 - 退避したファイルポイント (バイト位置 / 文字位置)
=-1 - 失敗

例 外 : InvalidOperationException - ファイルはオープンされていない

15.2.7. ファイル読み出し位置設定 (SetPoint)

形 式 : void SetPoint (long point);

引 数 : point - 設定する読み出し位置

説 明 : 現在の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でも、ファイルの先頭が 0 となります。

戻り値 : ≠-1 - 退避したファイルポイント (バイト位置 / 文字位置)
=-1 - 失敗

注 意 : point には、GetPoint () 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)

例 外 : InvalidOperationException - ファイルはオープンされていない

15.2.8. ファイル読み出しバイト位置取得 (GetBytePoint)

形 式 : long GetBytePoint ();

引 数 : なし

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。

戻り値 : ≠-1 - 退避したファイルポイント (バイト位置 / 文字位置)
=-1 - 失敗

例 外 : InvalidOperationException - ファイルはオープンされていない

15.2.9. 入力ファイルのEOFチェック (IsEof)

- 形 式** : `bool IsEof ();`
- 引 数** : なし
- 説 明** : 入力ファイルがEOF (ファイルの終端) に達したかチェックします。
- 戻り値** : `TRUE` - EOF
`FALSE` - EOF以外
- 例 外** : `InvalidOperationException` - ファイルはオープンされていない

15.2.10. 出力テキストファイル 生成 (Create)

- 形 式** : `void Create (string FilePath);`
`void Create (string FilePath, ETextEncode TextEncode);`
`void Create (string FilePath, ETextEncode TextEncode, EBomMode BomMode);`
`void Create (string FilePath, ETextEncode TextEncode, EBomMode BomMode, System.IO.FileShare share);`
- 引 数** : `FilePath` - 出力テキストファイルのパス名
`TextEncode` - 出力テキストファイルの文字コード (未指定時は「TextEncodeAtWrite」プロパティ値を採用)
`BomMode` - UTF-8/UTF-16 の BOM 出力指定 (未指定時は「BOM」プロパティ値を採用)
`Share` - 共有モード (None / Read / ReadWrite / Write - 他の同一ファイルへのアクセスを許可)
- 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。
「TextEncode」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	内容
<code>TEC_MBC</code>	マルチバイト (S-JIS)
<code>TEC_UTF_8</code>	UTF-8
<code>TEC_EUC_J</code>	EUC (日本語)
<code>TEC_UTF_16LE</code>	UTF-16 (リトルエンディアン)
<code>TEC_UTF_16BE</code>	UTF-16 (ビッグエンディアン)
<code>TEC_AUTO</code>	最後にオープンしたファイルのテキストエンコード (未オープンの場合は <code>TEC_MBC</code>)

`PutS()`、`PutFormat()`や、`PutC()`で、ファイルへ出力する場合は、UNICODE から「TextEncode」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。

「BomMode」は、「TextEncode」に「TEC_UTF_8, TEC_UTF_16LE or TEC_UTF_16BE」を指定した場合だけ有効であり、「WRITE_BOM」を指定すると、ファイルの先頭に UTF-8/UTF-16 の BOM を出力します。

- 戻り値** : なし
- 例 外** : `InvalidOperationException` - ファイルは既にオープン／生成されている
`FileCreationFailureException` - ファイルの生成を失敗した

15.2.11. 既存ファイルを追記モードでオープン／生成 (Append)

形 式 : void Append (string FilePath);
 void Append (string FilePath, ETextEncode TextEncode);
 void Append (string FilePath, ETextEncode TextEncode, EBomMode BomMode);
 void Append (string FilePath, ETextEncode TextEncode, EBomMode BomMode, System.IO.FileShare share);

引 数 : FilePath - 出力テキストファイルのパス名
 TextEncode - 出力テキストファイルの文字コード (未指定時は「TextEncodeAtWrite」プロパティ値を採用)
 BomMode - UTF-8/UTF-16 の BOM 出力指定 (未指定時は「BOM」プロパティ値を採用)
 Share - 共有モード (None / Read / ReadWrite / Write - 他の同一ファイルへのアクセスを許可)

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。
 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。(Create() と同じ)

戻り値 : なし

例 外 : InvalidOperationException - ファイルは既にオープン／生成されている
 FileCreationFailureException - ファイルの生成を失敗した

15.2.12. 文字列出力 (PutS)

形 式 : void PutS (string str);
 void PutS (string str, int len);

引 数 : str - 出力テキストデータ (文字列)
 len - 出力する文字数 (未指定時は文字列全体を出力)

説 明 : 指定したテキストデータをファイルへ書き込みます。
 「str」で指定されたテキストは、所定の文字コード (FCreate() の「TextEncode」で指定された文字コード) に変換されてテキストファイルに書き込まれます。

戻り値 : なし

例 外 : InvalidOperationException - ファイルは生成されていない
 FileAccessFailureException - ファイルへの書き込みを失敗した

15.2.13. 1文字出力 (PutC)

形 式 : void PutC (char c);

引 数 : c - 出力する1文字

説 明 : 指定したテキスト1文字をファイルへ書き込みます。
 サロゲートペア文字を出力するには、本メソッドを2回実行します。

戻り値 : なし

例 外 : InvalidOperationException - ファイルは生成されていない
 FileAccessFailureException - ファイルへの書き込みを失敗した

15.2.14. 書式文字列出力 (PutFormat)

形 式 : void PutFormat(string format [, arg]...);

引 数 : format - 書式文字列 (string.Format()と同じ)
arg - パラメタ (可変個引数)

説 明 : 書式化した文字列をファイルへ書き込みます。
書式文字列とパラメタは、string.Format()メソッドと同じです、
書式文字列や、パラメタには3つの文字('¥u0000', '¥uFFFA', '¥uFFFB')を含まないようにしてください。

戻り値 : なし

例 外 : InvalidOperationException - ファイルは生成されていない
FileAccessFailureException - ファイルへの書き込みを失敗した
FormatException - 不正な書式文字列

備 考 : 書式文字列や、パラメタに文字('¥u0000', '¥uFFFA', '¥uFFFB')が含まれる場合は、以下のように処理されます。
・'¥u0000' - 文字列の終端となります。
・'¥uFFFA' - '{'に変換されます。
・'¥uFFFB' - '}'に変換されます。

尚、FPutFormat()は、処理効率(速度やメモリ消費)が良くありません。
処理効率を考慮する必要がある場合は、string.Format()で書式化した文字列を出力するようにしてください。

(例) txf.FPutFormat(hFile, "x={0}", x); ⇒ string s = string.Format("x={0}", x);
txf.FPutS(hFile, s);

15.2.15. 出力データフラッシュ (Flush)

形 式 : void Flush();

引 数 : なし

説 明 : バッファリングされているデータを書き出します。

例 外 : InvalidOperationException - ファイルは生成されていない
FileAccessFailureException - ファイルへの書き込みを失敗した

戻り値 : なし

備 考 : 出力ファイルの書き込みデータは、一定サイズ バッファリングされています。
Flush()は、バッファリングされているデータをファイルへ書き込みます。

15.2.16. テキストファイルクローズ (Close)

形 式 : void Close();

引 数 : なし

説 明 : Open() / Create() でオープン/生成したテキストファイルをクローズします。
再度 Open() / Create() でファイルをオープン/作成するまで、テキストファイルをアクセスすることはできません。

例 外 : InvalidOperationException - ファイルはオープン/生成されていない
FileCloseFailureException - ファイルのクローズを失敗した

戻り値 : なし


```

39 :     }
40 :
41 : // ファイル読み出しボタン
42 : private void cmdRead_Click(object sender, EventArgs e)
43 : {
44 :     // テキストエンコード設定
45 :     ETextEncode tec = ETextEncode.TEC_MBC;
46 :     if (rbtMBC_I.Checked) tec = ETextEncode.TEC_MBC;
47 :     else if (rbtUTF8_I.Checked) tec = ETextEncode.TEC_UTF_8;
48 :     else if (rbtEUCJ_I.Checked) tec = ETextEncode.TEC_EUC_J;
49 :     else if (rbtUTF16LE_I.Checked) tec = ETextEncode.TEC_UTF_16LE;
50 :     else if (rbtUTF16BE_I.Checked) tec = ETextEncode.TEC_UTF_16BE;
51 :     else if (rbtAUTO_I.Checked) tec = ETextEncode.TEC_AUTO;
52 :     // ファイルを読み出して表示
53 :     vth.Purge();
54 :     try {
55 :         txf.Open(txtPath_I.Text, tec);
56 :     }
57 :     catch (Exception exc) {
58 :         vth.PutText("¥x1B[31m" + exc.ToString() + "¥x1B[0m¥n");
59 :         return;
60 :     }
61 :     switch (txf.TextEncodeAtRead) {
62 :         case ETextEncode.TEC_MBC: txtRxTec.Text = "MBC(S-JIS)"; break;
63 :         case ETextEncode.TEC_UTF_8: txtRxTec.Text = "UTF-8"; break;
64 :         case ETextEncode.TEC_EUC_J: txtRxTec.Text = "EUC-J"; break;
65 :         case ETextEncode.TEC_UTF_16LE: txtRxTec.Text = "UTF-16LE"; break;
66 :         case ETextEncode.TEC_UTF_16BE: txtRxTec.Text = "UTF-16BE"; break;
67 :     }
68 :     for (string s = txf.GetS(); s != null; s = txf.GetS()) {
69 :         vth.PutText(s);
70 :     }
71 :     txf.Close();
72 : }
73 :
74 : // ファイル書き込みボタン
75 : private void cmdWrite_Click(object sender, EventArgs e)
76 : {
77 :     // テキストエンコード設定
78 :     ETextEncode tec = ETextEncode.TEC_MBC;
79 :     if (rbtMBC_O.Checked) tec = ETextEncode.TEC_MBC;
80 :     else if (rbtUTF8_O.Checked) tec = ETextEncode.TEC_UTF_8;
81 :     else if (rbtEUCJ_O.Checked) tec = ETextEncode.TEC_EUC_J;
82 :     else if (rbtUTF16LE_O.Checked) tec = ETextEncode.TEC_UTF_16LE;
83 :     else if (rbtUTF16BE_O.Checked) tec = ETextEncode.TEC_UTF_16BE;
84 :     // 表示テキストをファイルへ出力
85 :     try {
86 :         txf.Create(txtPath_O.Text, tec, chkWriteBOM.Checked ? EBomMode.WRITE_BOM : EBomMode.NOT_WRITE_BOM);
87 :     }
88 :     catch (Exception exc) {
89 :         vth.PutText("¥x1B[31m" + exc.ToString() + "¥x1B[0m¥n");
90 :         return;
91 :     }
92 :     for (int i = 0; i < vth.LineCount; i++) {
93 :         string s = vth.GetLineText(i);
94 :         txf.PutS(s + "¥n");
95 :     }
96 :     txf.Close();
97 : }
98 : }
99 : }

```

15.3.2. Sil_TextFileC (テキストファイルの読み出しと書き込み, コンソールアプリ)

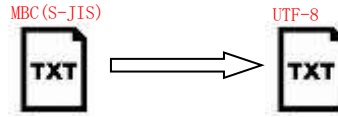
このサンプルプログラムは、ファイルのエンコードを指定してテキストファイルの読み出し／書き込みを行う
コンソールアプリのサンプルです。

テキストファイルのエンコードやパス名は、プログラム内にハードコードしています。

```

1 : //
2 : // Sil_TextFileC
3 : //
4 : using System;
5 : using System.Collections.Generic;
6 : using System.Linq;
7 : using System.Text;
8 : using System.Runtime.InteropServices;
9 : using AjrCustCtrl;
10 :
11 : namespace Sil_TextFileC
12 : {
13 :     class Program
14 :     {
15 :         static void Main(string[] args)
16 :         {
17 :             CAjrTextFile txfInp = new CAjrTextFile();
18 :             CAjrTextFile txfOut = new CAjrTextFile();
19 :
20 :             Console.WriteLine(@"Text file copy and change encode, -..¥sjis.txt to ..¥out_utf8.txt.");
21 :
22 :             txfInp.TextEncodeAtRead = ETextEncode.TEC_MBC;
23 :             txfOut.TextEncodeAtWrite = ETextEncode.TEC_UTF_8;
24 :             txfOut.BOM = EBomMode.WRITE_BOM;
25 :             try {
26 :                 txfInp.Open (@"..¥sjis.txt");
27 :                 txfOut.Create(@"..¥out_utf8.txt");
28 :                 for (string s = txfInp.GetS(); s != null; s = txfInp.GetS()) {
29 :                     txfOut.PutS(s);
30 :                 }
31 :                 txfInp.Close();
32 :                 txfOut.Close();
33 :             }
34 :             catch (Exception e) {
35 :                 Console.WriteLine(e.ToString() + "¥n");
36 :             }
37 :             Console.Write("Hit Enter key -");
38 :             Console.ReadLine();
39 :         }
40 :     }
41 : }

```



16. 文字列プール (CAjrStrPool.dll)

文字列を蓄積するコントロールです。
同一文字列は蓄積せずに、文字列の検索や削除ができます。

16.1. 構造体／列挙体 (定数)

16.1.1. 文字列比較パラメタ

```
public enum ESplComp {
    RECOGNIZE_WIDTH = 0,      // 英大小文字を区別する
    IGNORE_WIDTH    = 1,      // 英大小文字を区別しない
    ALPHABETIC      = 2,      // 英大小文字を区別する (英字順 ※1)
}
```

※1 : 英大小文字を区別して比較しますが、英字の大小にかかわらずアルファベット順に並べるようにします。

< RECOGNIZE_WIDTH >	< ALPHABETIC >
AAA	AAA
BBB	aaa
aaa	BBB

16.1.2. 部分文字列検索パラメタ

```
public enum ESplInStr {
    MATCHFIRST = 1,          // 先頭部分が文字列と一致
    INCLUDING   ,           // 文字列を含む
}
```

16.1.3. 文字列読み出し順

```
public enum ESplSeq {
    Ascending = 0,           // 昇順 ( false )
    Descending = 1,          // 降順 ( true  )
}
```

16.2. プロパティ

テキストファイル・アクセスのプロパティ一覧を示します。

#	名称	タイプ	内容	デフォルト
1	CompMode	ESplComp	文字列の登録や検索を行う際の文字列比較方法 ※1	MIXTURE
2	Count	int	蓄積されている文字列の個数 (読み出し専用)	—

※1 : CompMode プロパティを変更した場合、蓄積されている文字列は全て破棄されます。

16.3. メソッド

テキストファイル・アクセスのメソッド一覧を示します。

#	機能	メソッド名	備考
1	文字列登録	Regist , RegistPtr	
2	文字列検索	Find , FindPtr	
3	部分文字列検索	PartStrInPool , PartStrInPoolPtr	
4	文字列プール中の文字列を部分文字列とした検索	PoolStrInStr , PoolStrInStrPtr	
5	文字列削除	Remove	
6	全文字列破棄	Reset	
7	登録済み全文字列の列挙	EnumStr	
8	ポインタ→文字列変換	PtrToString	
9	文字列プールの消去	Delete	

16.3.1. 文字列登録 (Regist / RegistPtr)

形 式 : void Regist (string str);
IntPtr RegistPtr (string str);

引 数 : str - 文字列プールに登録する文字列

説 明 : 指定した文字列を文字列プールに登録します。
既に、同じ文字列が登録されている場合は、新たな登録は行いません。
登録可能な文字列の最大長は、3 2, 7 6 6 文字です。

戻り値 : Regist - なし
RegistPtr : 登録した文字列、あるいは、既に存在する同一文字列のアドレス

備 考 : RegistPtr() は、登録した文字列（あるいは、同一文字列）の場所を示す数値（アドレス）を返します。
PtrToString() メソッドで、この戻り値（0 以外）を文字列に変換できます。

例 外 : OverflowException - 文字列の長さが 32766 文字を超えた
RegistrationFailureException - 文字列の登録を失敗した

16.3.2. 文字列検索 (Find / FindPtr)

形 式 : bool Find (string str);
IntPtr FindPtr (string str);

引 数 : str - 検索する文字列

説 明 : 文字列プール中から、「str」と一致する文字列を検索します。

戻り値 : Find : true - 同一文字列が見つかった
false - 同一文字列は見つからなかった

FindPtr : ≠0 : 同一文字列が見つかった
=0 : 同一文字列は見つからなかった

備 考 : FindPtr() は、見つかった文字列の場所を示す数値（アドレス）を返します。
PtrToString() メソッドで、この戻り値（0 以外）を文字列に変換できます。

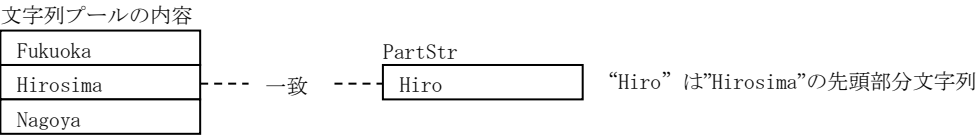
16.3.3. 部分文字列検索 (PartStrInPool / PartStrInPoolPtr)

形 式 : bool PartStrInPool (string PartStr);
bool PartStrInPool (string PartStr , ESplInStr param);
IntPtr PartStrInPoolPtr(string PartStr);
IntPtr PartStrInPoolPtr(string PartStr , ESplInStr param);

引 数 : PartStr - 検索する部分文字列
param - 文字列比較パラメタ (MATCHFIRST=先頭部分が一致, INCLUDING=文字列を含む)

説 明 : 文字列プール中から「PartStr」で示す部分文字列を含む文字列を検索します。

(例, MATCHFIRST=先頭部分が一致)



戻り値 : PartStrInPool : true - 部分文字列を含む文字列が見つかった
false - 部分文字列を含む文字列は見つからなかった

PartStrInPoolPtr : ≠0 : 部分文字列を含む文字列が見つかった
=0 : 部分文字列を含む文字列は見つからなかった

備 考 : PartStrInPoolPtr ()は、見つかった文字列の場所を示す数値 (アドレス) を返します。
PtrToString() メソッドで、この戻り値 (0 以外) を文字列に変換できます。

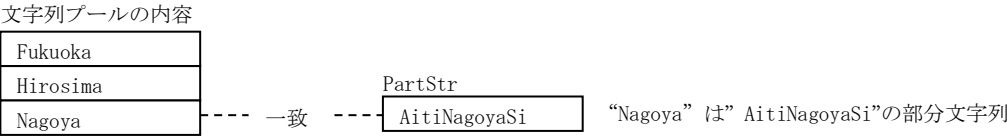
16.3.4. 文字列プール中の文字列を部分文字列とした検索 (PoolStrInStr / PoolStrInStrPtr)

形 式 : bool PoolStrInStr (string PartStr);
bool PoolStrInStr (string PartStr , ESplInStr param);
IntPtr PoolStrInStrPtr(string PartStr);
IntPtr PoolStrInStrPtr(string PartStr , ESplInStr param);

引 数 : PartStr - 検索される文字列
param - 文字列比較パラメタ (MATCHFIRST=先頭部分が一致, INCLUDING=文字列を含む)

説 明 : 文字列プール中の文字列を部分文字列として「PartStr」で示す文字列の当該部分文字列となる文字列を検索します。

(例, INCLUDING=文字列を含む)



戻り値 : PoolStrInStr : true - 部分文字列となる文字列が見つかった
false - 部分文字列となる文字列は見つからなかった

PoolStrInStr Ptr : ≠0 : 部分文字列となる文字列が見つかった
=0 : 部分文字列となる文字列は見つからなかった

備 考 : PoolStrInStrPtr ()は、見つかった文字列プール中の文字列の場所を示す数値 (アドレス) を返します。
PtrToString() メソッドで、この戻り値 (0 以外) を文字列に変換できます。

16.3.5. 文字列削除 (Remove)

形 式 : `bool Remove (string str);`

引 数 : `str` - 削除する文字列

説 明 : 文字列プールから「`str`」で示される文字列を削除します。

戻り値 : `true` - 文字列削除成功
`false` - 文字列削除失敗 (当該文字列は見つからなかった)

16.3.6. 全文字列破棄 (Reset)

形 式 : `void Reset ();`

引 数 : なし

説 明 : 文字列プールに登録されているすべての文字列を破棄します。

戻り値 : なし

16.3.7. 登録済み文字列の列挙 (EnumStr)

形 式 : `int EnumStr();`
`int EnumStr(ESplSeq seq);`
`int EnumStr(ESplSeq seq, IntPtr cbp);`
`int EnumStr( SplCbKntcStr cb);`
`int EnumStr(ESplSeq seq, SplCbKntcStr cb);`
`int EnumStr(ESplSeq seq, IntPtr cbp, SplCbKntcStr cb);`

引 数 : `seq` - 文字列の読み出し順 (Ascending=昇順, , Descending=降順, 省略時は昇順)
`cbp` - コールバックパラメタ (省略時は0)
`cb` - 文字列列挙通知用コールバックメソッド (コンソールアプリ用)

説 明 : 文字列プールに登録されている全ての文字列を読み出します。
各文字列は、`OnNtcStr()` イベントにより通知されます。

戻り値 : 文字列の個数

16.3.8. ポインタ→文字列変換 (PtrToString)

形 式 : `string PtrToString(IntPtr ptr);`

引 数 : `ptr` - 文字列プール中の文字列の場所を示す数値 (アドレス)

説 明 : 以下のメソッドの戻り値 (0 以外) を文字列に変換します。

- `RegistPtr`
- `FindPtr`
- `PartStrInPoolPtr`
- `PoolStrInStrPtr`

戻り値 : `≠null` : 変換した文字列
`=null` : `ptr=0` が指定された

16.3.9. 文字列プールの消去 (PtrToString)

形 式 : `string PtrToString(IntPtr ptr);`

引 数 : なし

説 明 : 文字列プールコントロールを消去します。

このメソッドは、コンソールアプリの場合に実行してください。
Windows フォームアプリ等では実行しないでください（自動的に実行されます）

戻り値 : なし

16.4. イベント／デリゲート

文字列プールのイベントを示します。

コンソールアプリの場合は、イベントが発生しない為、デリゲートを介してコールバックメソッドにより通知を受け取ります。
以降の説明中「パラメタ」はイベントのパラメタ形式を示します。デリゲートの場合は先頭の「e.」を削除した形式となります。

16.4.1. 文字列の通知 (OnNtcStr)

形 式 : `bool OnNtcStr(object sender, SplArgNtcStr e);`
`delegate bool SplCbKntcStr(string str, IntPtr cbp);`

パラメタ : `string e.str` - 列挙通知文字列
`IntPtr e.cbp` - コールバックパラメタ

説 明 : このイベントは `EnumStr()` メソッドを実行することにより発生します。
文字列プール中の文字列を順次通知します。

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

16.5. サンプルプログラム

16.5.1. Sil_StrPool (文字列の登録, 削除, 検索, リセット)

次のプログラムは、文字列の登録、検索、削除やリセットを行います。

C

```

1 : using System;
2 : using System.Collections.Generic;
3 : using System.ComponentModel;
4 : using System.Data;
5 : using System.Drawing;
6 : using System.Linq;
7 : using System.Text;
8 : using System.Windows.Forms;
9 : using CAjrCustCtrl;
10 :
11 : namespace Sil_StrPool
12 : {
13 :     public partial class Form1 : Form
14 :     {
15 :         public Form1()
16 :         {
17 :             InitializeComponent();
18 :         }
19 :         // フォーム開始
20 :         private void Form1_Load(object sender, EventArgs e)
21 :         {
22 :             // 設定値ロード
23 :             SAjrReg.LoadAllCtrls(this);
24 :             // 初期文字列登録
25 :             SetInitialValue();
26 :         }
27 :         // フォーム終了
28 :         private void Form1_FormClosed(object sender, FormClosedEventArgs e)
29 :         {
30 :             // 設定値セーブ
31 :             SAjrReg.SaveAllCtrls(this);
32 :         }
33 :         // RECOGNIZE_WIDTH (英大小区別あり)
34 :         private void rbtRECOGNIZEWIDTH_CheckedChanged(object sender, EventArgs e)
35 :         {
36 :             spl.CompMode = ESplComp.RECOGNIZE_WIDTH;
37 :             ShowAll();
38 :         }
39 :         // IGNORE_WIDTH (英大小区別なし)
40 :         private void rbtIGNOREWIDTH_CheckedChanged(object sender, EventArgs e)
41 :         {
42 :             spl.CompMode = ESplComp.IGNORE_WIDTH;

```

```

43 :         ShowAll();
44 :     }
45 :     // ALPHABETIC (英大小区別あり, 英字順)
46 :     private void rbtMIXTURE_CheckedChanged(object sender, EventArgs e)
47 :     {
48 :         spl.CompMode = ESplComp.ALPHABETIC;
49 :         ShowAll();
50 :     }
51 :     // Ascending (昇順)
52 :     private void rbtAscending_CheckedChanged(object sender, EventArgs e)
53 :     {
54 :         ShowAll();
55 :     }
56 :     // Descending (降順)
57 :     private void rbtDescending_CheckedChanged(object sender, EventArgs e)
58 :     {
59 :         ShowAll();
60 :     }
61 :
62 :     // 文字列登録
63 :     private void cmdRegist_Click(object sender, EventArgs e)
64 :     {
65 :         IntPtr ptr = spl.RegistPtr(txtRegist.Text);
66 :         lblRegist.Text = spl.PtrToString(ptr);
67 :         ShowAll();
68 :     }
69 :     // 文字列検索
70 :     private void cmdFind_Click(object sender, EventArgs e)
71 :     {
72 :         IntPtr ptr = spl.FindPtr(txtFind.Text);
73 :         if (ptr != (IntPtr)0) lblFind.Text = spl.PtrToString(ptr);
74 :         else
75 :             lblFind.Text = "Not found";
76 :     }
77 :     // 文字列プール中の部分文字列検索
78 :     private void cmdInPool_Click(object sender, EventArgs e)
79 :     {
80 :         IntPtr ptr;
81 :         if (rbtInPoolMATCHFIRST.Checked) ptr = spl.PartStrInPoolPtr(txtInPool.Text, ESplInStr.MATCHFIRST);
82 :         else
83 :             ptr = spl.PartStrInPoolPtr(txtInPool.Text, ESplInStr.INCLUDING );
84 :         if (ptr != (IntPtr)0) lblInPool.Text = spl.PtrToString(ptr);
85 :         else
86 :             lblInPool.Text = "Not found";
87 :     }
88 :     // 文字列プールの各文字列を部分文字列として検索
89 :     private void cmdInStr_Click(object sender, EventArgs e)
90 :     {
91 :         IntPtr ptr;
92 :         if (rbtInStrMATCHFIRST.Checked) ptr = spl.PoolStrInStrPtr(txtInStr.Text, ESplInStr.MATCHFIRST);
93 :         else
94 :             ptr = spl.PoolStrInStrPtr(txtInStr.Text, ESplInStr.INCLUDING );
95 :         if (ptr != (IntPtr)0) lblInStr.Text = spl.PtrToString(ptr);
96 :         else
97 :             lblInStr.Text = "Not found";
98 :     }
99 :     // 文字列削除
100 :     private void cmdRemove_Click(object sender, EventArgs e)
101 :     {
102 :         if (spl.Remove(txtRemove.Text)) lblRemove.Text = "Success";
103 :         else
104 :             lblRemove.Text = "Failure";
105 :         ShowAll();
106 :     }
107 :     // 文字列プールリセット
108 :     private void cmdReset_Click(object sender, EventArgs e)
109 :     {
110 :         spl.Reset();
111 :         ShowAll();
112 :     }
113 :     // 文字列プールの内容表示
114 :     private void ShowAll()
115 :     {
116 :         vth.Purge();
117 :         if (rbtAscending.Checked) spl.EnumStr(ESplSeq.Ascending);
118 :         else
119 :             spl.EnumStr(ESplSeq.Descending);
120 :     }
121 :     // 文字列通知イベント
122 :     private bool spl_OnNtcStr(object sender, CAjrCustCtrl.SplArgNtcStr e)
123 :     {
124 :         vth.PutText(e.str + "¥n");
125 :         return true;
126 :     }
127 :     // 初期値登録
128 :     private void cmdInit_Click(object sender, EventArgs e)
129 :     {
130 :

```

```
123 :         SetInitialValue();
124 :     }
125 :     private void SetInitialValue()
126 :     {
127 :         vth.Purge();
128 :         spl.Regist("Sapporo");
129 :         spl.Regist("Sendai");
130 :         spl.Regist("Niigata");
131 :         spl.Regist("Yokohama");
132 :         spl.Regist("Nagoya");
133 :         spl.Regist("Hiroshima");
134 :         spl.Regist("Fukuoka");
135 :         ShowAll();
136 :     }
137 : }
138 : }
```

16.5.2. Sil_StrPoolC (コンソールアプリ)

次のプログラム (コンソールアプリ) は、入力した文字列を文字列プールに登録します。

“E” を入力すると、登録されている文字列を列挙します。

“R” を入力すると文字列プールをリセットします。

“Q” を入力すると、プログラムを終了します。

```

file:///R:/AjrCtrl_XX-XX-XX/_gbl/Samples/AjrMsil/CSharp...
入力した文字列を文字列プールに登録します。
“E” を入力すると登録した文字列を列挙します。
“R” を入力すると文字列プールをリセットします。
“Q” を入力するプログラムを終了します。

CCC
BBB
GGG
AAA
AAA
AAA
E
EnumStr - AAA
EnumStr - BBB
EnumStr - CCC
EnumStr - GGG
R
JJJ
E
EnumStr - JJJ

```

```

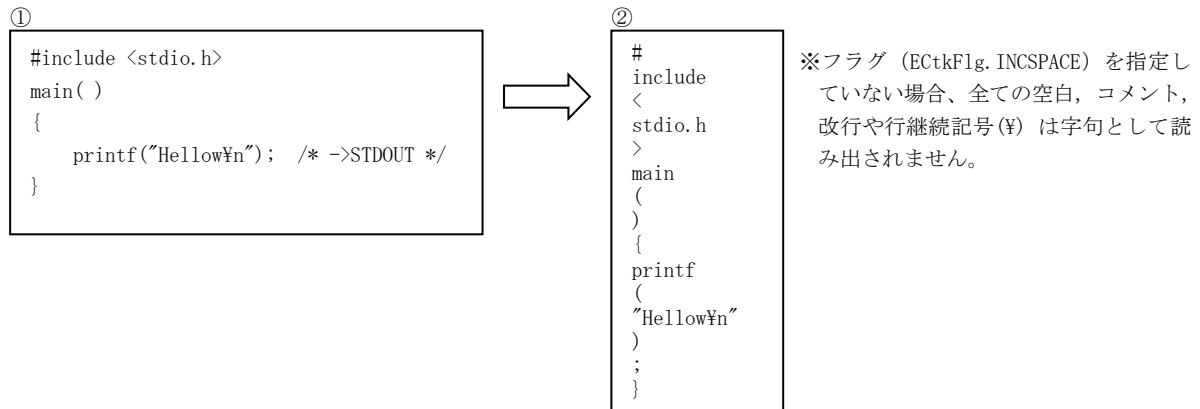
1 : using System;
2 : using System.Collections.Generic;
3 : using System.Linq;
4 : using System.Text;
5 : using CAjrCustCtrl;
6 :
7 : namespace Sil_StrPoolC
8 : {
9 :     class Program
10 :    {
11 :        static CAjrStrPool spl = new CAjrStrPool(); // 文字列プールオブジェクト生成
12 :
13 :        static void Main(string[] args)
14 :        {
15 :            Console.WriteLine(" 入力した文字列を文字列プールに登録します。");
16 :            Console.WriteLine(" 'E' を入力すると登録した文字列を列挙します。");
17 :            Console.WriteLine(" 'R' を入力すると文字列プールをリセットします。");
18 :            Console.WriteLine(" 'Q' を入力するとプログラムを終了します。");
19 :            Console.WriteLine("");
20 :
21 :            string s = "";
22 :            while (s != "Q" && s != "q") {
23 :                s = Console.ReadLine();
24 :                switch (s) {
25 :                    case "Q": // Q : 終了
26 :                        case "q":
27 :                            break;
28 :                    case "R": // R : リセット
29 :                        case "r":
30 :                            spl.Reset();
31 :                            break;
32 :                    case "E": // E : 文字列列挙
33 :                        case "e":
34 :                            SplCbKntcStr cb = new SplCbKntcStr(cbNtcStr);
35 :                            spl.EnumStr(cb);
36 :                            break;
37 :                    default: // その他 : 文字列登録
38 :                        spl.Regist(s);
39 :                        break;
40 :                }
41 :            }
42 :            spl.Delete();
43 :        }
44 :        // 文字列列挙通知
45 :        static bool cbNtcStr(string str, IntPtr cbp)
46 :        {
47 :            Console.WriteLine("EnumStr - " + str);
48 :            return true;
49 :        }
50 :    }
51 : }

```

17. C言語の字句分解 (CAjrCToken.dll)

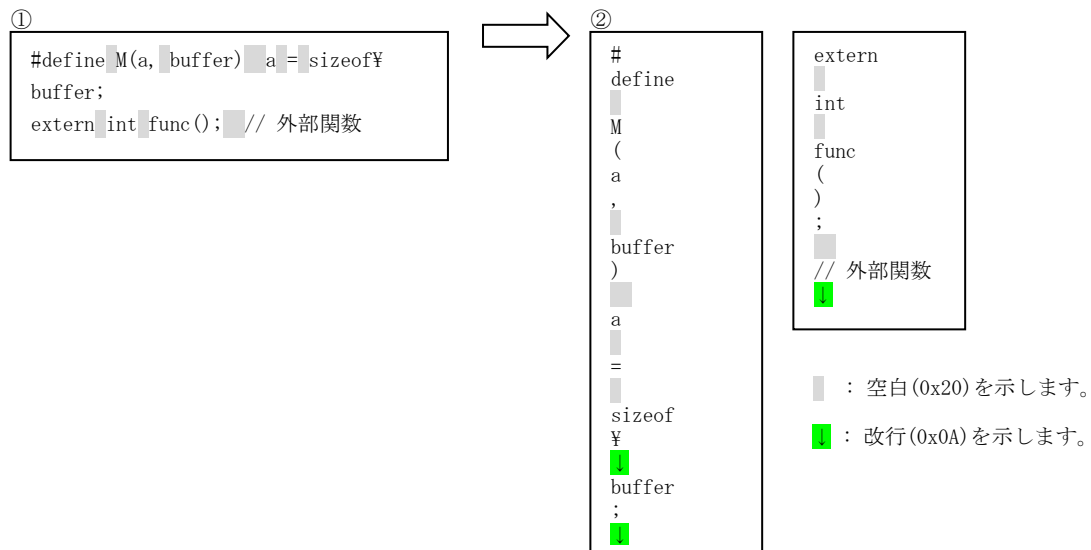
C/C++言語のソースプログラムテキストを、字句（トークン）の単位に分解し、順次読み出すことができます。
ソースプログラムテキスト中のコメント（「/* ... */」や「// ...」）部分を無視することができます。

例えば、以下の①の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。



17.1. 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 (ECtkFlg. 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 (¥) も字句として読み出されます。



※ フラグ (ECtkFlg. INCSPACE) を指定した場合、全ての文字を字句として読み出すことになります。
従って、読み出した字句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。

17.2. 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。

ECtkFlg 列挙体

#	フラグ名	内容
1	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	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	MACNAME	#define 文により定義されたマクロ名
4	MACWITHARG	引数付きのマクロ名 (CTKF_MACNAME も同時にセットされる)
5	MACARG	引数付きマクロの仮引数部分 (前後の ‘(’ と ‘)’ も含む)
6	MACBODY	マクロボディ

例えば、以下の①の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 (ECtkFlg. xxxxx)					
	PREPRO	PPTOP	MACNAME	MACWITHARG	MACARG	MACBODY
#	1	1	0	0	0	0
define	1	0	0	0	0	0
ADD	1	0	1	1	0	0
(	1	0	0	0	1	0
A	1	0	0	0	1	0
,	1	0	0	0	1	0
B	1	0	0	0	1	0
)	1	0	0	0	1	0
(	1	0	0	0	0	1
A	1	0	0	0	0	1
+	1	0	0	0	0	1
B	1	0	0	0	0	1
)	1	0	0	0	0	1
#	1	1	0	0	0	0
define	1	0	0	0	0	0
SUB	1	0	1	0	0	0
(	1	0	0	0	0	1
a	1	0	0	0	0	1
-	1	0	0	0	0	1
b	1	0	0	0	0	1
)	1	0	0	0	0	1

17.3. #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 (ECtkCode. PATHNAME: パス名) として認識します。

例えば、以下の①の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
#	ECtkCode. DLM_SHARP
include	ECtkCode. RSV_INCLUDE
<	ECtkCode. DLM_LO
stdio.h	ECtkCode. PATHNAME
>	ECtkCode. DLM_HI
#	ECtkCode. DLM_SHARP
include	ECtkCode. RSV_INCLUDE
"local.h"	ECtkCode. STR_QUOTE

17.4. 構造体／列挙体 (定数)**17.4.1. 機能フラグ (ECtkOpt)**

```

public enum ECtkOpt : uint {
    // Bit 7 - 0
    CPLUSPLUS      = 0x00000001 ,    // C++のトークンを認識する
    INCSpace       = 0x00000002 ,    // 空白／改行／コメント／行継続記号(¥) もトークンに含める
    INCSYM_DOLLAR  = 0x00000004 ,    // シンボルの英字としてドルマーク($)を含める
    INCSYM_ATMARK  = 0x00000008 ,    // シンボルの英字としてアットマーク(@)を含める
    INCSYM_MBSTR   = 0x00000010 ,    // シンボルの英字としてマルチバイト文字 (全角文字) を含める
    APOST_NOESC    = 0x00000020 ,    // 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
    DOTSYMBOL     = 0x00000040 ,    // シンボル+Dot(.)+数字列を 1つのシンボルとみなす(Ex. P3.12)
    ASMHEX        = 0x00000080 ,    // アセンブラ記法の 16進数 (末尾が H, ex 0FFH)を許可する
}

```

17.4.2. トークン識別フラグ (ECtkFlg)

```

public enum ECtkFlg {
    PREPRO        = 0x0080,    // プリプロセス文
    PPTOP         = 0x0040,    // プリプロセス文の先頭トークン (#)
    MACNAME       = 0x0020,    // マクロ名
    MACWITHARG    = 0x0010,    // 引数付のマクロ名
    MACARG        = 0x0008,    // マクロ引数' ( . . . ) '
    MACBODY       = 0x0004,    // マクロボディ
    NXTSPC        = 0x0004,    // このトークンの次に空白ありを示すフラグ
    WIDECCHAR     = 0x0001,    // ワイド文字 (L'X' / L"XXX") フラグ
}

```

17.4.3. エラーコード (ECtkError)

```

public enum ECtkError {
    OK              = 0 ,    // OK
    NULLPTR         = 0x8000 ,    // NULLポインタが指定された
    ZEROSIZE        = 0x4000 ,    // バッファサイズがゼロ
    MEMALLOC        = 0x2000 ,    // メモリ割り当て失敗
    INVALID_OCTALNUMBER = 0x1000 ,    // 不正な 8進数
    INVALID_HEXADECIMAL = 0x0800 ,    // 不正な 16進数
    INVALID_REALNUMBER = 0x0400 ,    // 不正な実数表現
    INVALID_SUFFIX   = 0x0200 ,    // 不正なサフィックス
    INVALID_DELIMITER = 0x0100 ,    // 不正なデリミタ
    LFINSTR         = 0x0080 ,    // 文字列中に改行が含まれている
    BUFOVER         = 0x0040 ,    // トークンバッファオーバー
    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)
}

```

17.4.4. 数値定数のサフィックス・コード (ECtkSuf)

```

public enum ECtkSuf {
    SUF_NONE      = 0 ,    // サフィックスなし
    SUF_L         ,    // L
    SUF_UL        ,    // UL / LU
    SUF_LL        ,    // LL
    SUF_ULL       ,    // ULL
    SUF_FLOAT     ,    // F
    SUF_DOUBLE    ,    // D
    SUF_LDOUBLE   ,    // L
}

```


17.4.5. トークンコード (EctkCode)

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
RSV_ASM	asm	RSV_EXTERN	extern	RSV_SHORT	short
RSV_AUTO	auto	RSV_FLOAT	float	RSV_SIGNED	signed
RSV_BREAK	break	RSV_FOR	for	RSV_SIZEOF	sizeof
RSV_CASE	case	RSV_FRIEND	friend	RSV_STATIC	static
RSV_CATCH	catch	RSV_GOTO	goto	RSV_STACAST	static_cast
RSV_CHAR	char	RSV_IF	if	RSV_STRUCT	struct
RSV_CLASS	class	RSV_IFDEF	ifdef	RSV_SWITCH	switch
RSV_CONST	const	RSV_IFNDEF	ifndef	RSV_TEMPLATE	template
RSV_CONCAST	const_cast	RSV_INCLUDE	include	RSV_THIS	this
RSV_CONTINUE	continue	RSV_INLINE	inline	RSV_THROW	throw
RSV_DEFAULT	default	RSV_INT	int	RSV_TRY	try
RSV_DEFINE	define	RSV_LONG	long	RSV_TYPEDEF	typedef
RSV_DEFINED	defined	RSV_NEW	new	RSV_UNION	union
RSV_DELETE	delete	RSV_OPERATOR	operator	RSV_UNDEF	undef
RSV_DO	do	RSV_PRAGMA	pragma	RSV_UNSIGNED	unsigned
RSV_DOUBLE	double	RSV_PRIVATE	private	RSV_VIRTUAL	virtual
RSV_DYNCAST	dynamic_cast	RSV_PROTECTED	protected	RSV_VOID	void
RSV_ELIF	elif	RSV_PUBLIC	public	RSV_VOLATILE	volatile
RSV_ELSE	else	RSV_REGISTER	register	RSV_WCHAR_T	wchar_t
RSV_ENDIF	endif	RSV_REICAST	reinterpret_cast	RSV_WHILE	while
RSV_ENUM	enum	RSV_RETURN	return	USR_NAME	ユーザ定義名

数値定数

コード名	トークン
VAL_DECIMAL	10進数
VAL_HEX	16進数
VAL_OCTAL	8進数
VAL_REAL	実数
VAL_INVALID	不正な数値表現

文字列

コード名	トークン
STR_QUOTE	" ... "
STR_APOST	' ... '
VAL_HEX_H	16進数

パス名

コード名	トークン
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。 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
DLM_LSPART	(	DLM_DOT	.	DLM_OREQ	=
DLM_LMPART	{	DLM_SHREQ	>>=	DLM_XOREQ	^=
DLM_LLPART	[	DLM_SHLEQ	<<=	DLM_SHARP2	##
DLM_RSPART	)	DLM_SHR	>>	DLM_PLUS2	++
DLM_RMPART	}	DLM_SHL	<<	DLM_MINUS2	--
DLM_RLPART	]	DLM_EQEQ	==	DLM_SHARP	#
DLM_SCOPE	::	DLM_NOTEQ	!=	DLM_PLUS	+
DLM_IDDOT	.*	DLM_LOEQ	<=	DLM_MINUS	-
DLM_IDARROW	->*	DLM_HIEQ	>=	DLM_MULT	*
DLM_LAND	&&	DLM_LO	<	DLM_DIV	/
DLM_LOR		DLM_HI	>	DLM_MOD	%
DLM_ARROW	->	DLM_PLUSEQ	+=	DLM_AND	&
DLM_QUEST	?	DLM_MINUSEQ	-=	DLM_OR	
DLM_COLON	:	DLM_MULTEQ	*=	DLM_XOR	^
DLM_SEMICOL	;	DLM_DIVEQ	/=	DLM_EQ	=
DLM_COMMA	,	DLM_MODEQ	%=	DLM_NOT	~
DLM_VARG	...	DLM_ANDEQ	&=	DLM_LNOT	!

空白／改行／コメント／行継続記号“¥” (AjcCtkCreate で flag に 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
SPACE	空白／タブ文字列	COMMENT	/*...*/	LINECONT	行継続記号 (¥)
EOL	改行文字列	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。

17.5. プロパティ

C言語の字句分解のサポートAPI一覧を以下に示します。

#	タイプ	プロパティ	内容	備考
1	ECtkTabWidth	TabWidth	TABW_2 - タブ幅=2 TABW_8 - タブ幅=8 TABW_4 - タブ幅=4 TABW_16 - タブ幅=16	
2	bool	optCPlusPlus	true - C++のトークンを認識する	
3	bool	optIncSpace	true - 空白, 改行, コメントや行継続記号 (¥) もトークンに含める	
4	bool	optIncDollar	true - シンボルの英字としてドルマーク (\$) を含める	
5	bool	optIncAtMark	true - シンボルの英字としてアットマーク (@) を含める	
6	bool	optIncMbs	true - シンボルの英字としてマルチバイト文字を含める	
7	bool	optIncDotSym	true - シンボル+Dot(.)+数字列を1つのシンボルとみなす (ex. P3.0)	
8	bool	optIncAsmHex	true - 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	
9	ECtkCode	CurToken	現在のトークンコード	GetToken() or PeekToken() メソッド実行後のみ有効 (読み出し専用)
10	ECtkSuf	CurSuffix	現在の数値定数のサフィックス	
11	ECtkFlg	CurFlag	現在のトークンのフラグ情報	
12	int	CurLineNumber	現在のトークンの行番号	
13	int	CurPosition	現在のトークンの桁位置	
14	ECtkError	CurError	現在のエラーコード	

17.6. メソッド

C言語・字句分解のメソッド一覧を以下に示します。

#	関数名	内容	備考
1	SetCallBack	1行読み出しコールバックメソッドの設定	
2	GetToken	字句の読み出し	
3	PeekToken	字句先読み	
4	TokenString	字句コードに対応する文字列の取得	
5	IsUserSymbol	トークンがユーザシンボルかチェックする	
6	IsReservedSymbol	トークンが予約語かチェックする	
7	IsNumericValue	トークンが数値定数かチェックする	
8	IsString	トークンが文字列かチェックする	
9	IsPathName	トークンがパス名かチェックする	
10	IsDelimiter	トークンがデリミタかチェックする	
11	IsSymbol	トークンがシンボルかチェックする	シンボルとは、先頭が英字かアンダバーで
12	IsValSym	トークンがシンボル/数値定数かチェックする	後続が、英数字かアンダバー
13	IsSpace	トークンがスペースかチェックする	スペースは、空白/コメント/行継続記号(¥)
14	Push	インスタンス退避	
15	Pop	インスタンス回復	
16	Reset	インスタンスリセット	
17	Delete	インスタンス消去	

17.6.1. コールバックメソッドの設定 (SetCallBack)

形 式 : void SetCallBack (CtkCbkJGetS cbGetS);

引 数 : cbGetS - 1行入力用コールバックメソッド

説 明 : ソースプログラムを1行読み出すためのコールバックメソッドを設定します。
コンソールアプリでは、イベントが発生しない為、コールバックメソッドを介してイベントを受け取ります。
コールバックメソッドについては、後述の「デリゲート」を参照してください。

戻り値 : なし

コールバック : コールバックメソッドの形式は、以下の通りです。

cbGetS (ソーステキスト1行読み出し)

形 式 : bool cbGetS(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文字数 (文字列終端(0x0)を含む)
cbp - コールバックパラメタ

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 (1行) を読み出した
false - EOF (ソーステキストの終端を検出した)

17.6.2. 字句の読み出し (GetToken)

形 式 : bool GetToken (out string syl);

引 数 : syl - 読み出したトークン文字列 (出力)

説 明 : ソースプログラムテキストから、順次、字句情報を1つつ読み出します。
本メソッドを実行後、以下のプロパティにより、字句文字列以外にも、字句に関する情報を取得できます。

CurError ----- エラーコードを返します (※1 (本メソッド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))
CurToken ----- トークンコードを返します (※2)
CurSuffix ----- 値定数のサフィックスコードを返します (※3)
CurFlag ----- フラグ情報を返します (※4)
CurLineNumber ----- 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
CurPosition ----- 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文字として計算)

また、以下のメソッドにより CurToken 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。

IsUserSymbol (tkn) - トークンがユーザ定義名である場合、trueを返します。
IsReservedSymbol (tkn) - トークンが予約名である場合、trueを返します。
IsNumericValue (tkn) - トークンが数値定数である場合、trueを返します。
IsString (tkn) - トークンが文字列である場合、trueを返します。
IsPathName (tkn) - トークンがヘッダファイルのパス名である場合、trueを返します。
IsDelimiter (tkn) - トークンがデリミタである場合、trueを返します。
IsSymbol (tkn) - トークンがユーザ定義名/予約名である場合、trueを返します。
IsValSym (tkn) - トークンがユーザ定義名/予約名/数値定数である場合、trueを返します。
IsSpace (tkn) - トークンが空白/改行/コメント/行継続記号(¥)である場合、true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、「CurError プロパティ」により取得します)
false - EOF

17.6.3. 字句先読み (PeekToken)

形 式 : `bool PeekToken ( out string syl);`

引 数 : `syl` - 読み出したトークン文字列 (出力)

説 明 : 字句情報を先読みします。(直前に `GetToken()` を実行している場合、その次の字句情報を取得します。)
「`GetToken()`」は、字句情報を1つずつ進めるのに対し、「`PeekToken()`」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、「`PeekToken()`」直後の「`GetToken()` / `PeekToken()`」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` - `E O F`

17.6.4. 字句コードに対応する文字列の取得 (TokenString)

形 式 : `string TokenString (ECtkCode tkn );`

引 数 : `tkn` - トークンコード

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 (`RSV_XXXX`) あるいは、デリミタ (`DLM_XXXX`) 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

17.6.5. トークンがユーザシンボルかチェックする (IsUserSymbol)

形 式 : `bool IsUserSymbol (ECtkCode tkn );`

引 数 : `tkn` - トークンコード

説 明 : トークンコードがユーザシンボル (予約語以外のシンボル) かチェックします。

戻り値 : `true` - トークンはユーザシンボルである
`false` - トークンはユーザシンボル以外

17.6.6. トークンが予約語シンボルかチェックする (IsReservedSymbol)

形 式 : `bool IsReservedSymbol (ECtkCode tkn );`

引 数 : `tkn` - トークンコード

説 明 : トークンコードが予約語シンボルかチェックします。

戻り値 : `true` - トークンは予約語シンボルである
`false` - トークンは予約語シンボル以外

17.6.7. トークンが数値定数かチェックする (IsNumericValue)

形 式 : `bool IsNumericValue (ECtkCode tkn );`

引 数 : `tkn` - トークンコード

説 明 : トークンコードが数値定数かチェックします。

戻り値 : `true` - トークンは数値定数である
`false` - トークンは数値定数以外

17.6.8. トークンが文字列かチェックする (IsString)

形 式 : `bool IsString (ECtkCode tkn );`

引 数 : `tkn` - トークンコード

説 明 : トークンコードが文字列 ("`xxxx`" or '`xxxx`') かチェックします。

戻り値 : `true` - トークンは文字列である
`false` - トークンは文字列以外

17.6.9. トークンがパス名かチェックする (IsPathName)

形 式 : `bool IsPathName (ECtkCode tkn );`

引 数 : `tkn` - トークンコード

説 明 : トークンコードがパス名 (`#include <XXXXX>` の '`XXXXX`') かチェックします。

戻り値 : `true` - トークンはパス名である
`false` - トークンはパス名以外

17.6.10. トークンがデリミタかチェックする (IsDelimiter)

形 式 : `bool IsDelimiter (ECtkCode tkn );`

引 数 : `tkn` - トークンコード

説 明 : トークンコードがデリミタ (英数字以外の '`==`' や '`%`' 等) かチェックします。

戻り値 : `true` - トークンはデリミタである
`false` - トークンはデリミタ以外

17.6.11. トークンがシンボルかチェックする (IsSymbol)

形 式 : `bool IsSymbol (ECtkCode tkn );`

引 数 : `tkn` - トークンコード

説 明 : トークンコードがシンボルかチェックします。
 シンボルとは、先頭が英字かアンダバーで、後続が英数字かアンダバーで構成する名称 (ex. `Type01`)

戻り値 : `true` - トークンはシンボルである
`false` - トークンはシンボル以外

17.6.12. トークンがシンボル／数値定数かチェックする (IsValSym)

形 式 : `bool IsValSym (ECtkCode tkn );`

引 数 : `tkn` - トークンコード

説 明 : トークンコードがシンボル／数値定数かチェックします。

戻り値 : `true` - トークンはシンボル／数値定数である
`false` - トークンはシンボル, 数値定数以外

17.6.13. トークンがシンボル／数値定数かチェックする (IsSpace)

形 式 : `bool IsSpace (ECtkCode tkn );`

引 数 : `tkn` - トークンコード

説 明 : トークンコードが空白, コメント, あるいは, 行継続記号(¥)かチェックします。

戻り値 : `true` - トークンは空白, コメント, あるいは, 行継続記号(¥)である
`false` - トークンは空白, コメント, および , 行継続記号(¥)以外

17.6.14. インスタンス退避 (Push)

形 式 : `bool Push ();`

引 数 : なし

説 明 : インスタンスを一時退避します。(最大32ネスト)

戻り値 : なし

例 外 : `InvalidOperationException` - ネスト オーバーフロー

17.6.15. インスタンス回復 (Pop)

形 式 : `bool Pop ();`

引 数 : なし

説 明 : `Push()`により退避したインスタンスを回復します。

戻り値 : なし

例 外 : `InvalidOperationException` - ネスト アンダーフロー

17.6.16. インスタンスリセット (Reset)

形 式 : `void Reset();`

引 数 : なし

説 明 : C言語の字句分解インスタンスを初期状態に戻します。
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 : なし

17.6.17. インスタンス消去 (Delete)

形 式 : `void Delete();`

引 数 : なし

説 明 : C言語の字句分解インスタンスを解放します。
コンソールアプリの場合、プログラムの最後に、本メソッドでインスタンスを解放してください。
Windows フォームアプリでは、本メソッドを実行する必要はありません。(暗黙的に実行されます)

戻り値 : なし

17.7. イベント／デリゲート

C言語の字句分解のイベントを示します。

コンソールアプリの場合は、イベントが発生しない為、デリゲートを介して、コールバックメソッドにより通知を受け取ります。以降の説明中「パラメタ」はイベントのパラメタ形式を示します。デリゲートの場合は先頭の「e.」を削除した形式となります。

17.7.1. 1行読み出し (OnGetS)

形 式 : `bool OnGetS(object sender, CtkArgGetS e);`
`delegate bool CtkCbKGetS(IntPtr pBuf, int lBuf);`

パラメタ : e. pBuf - 読み出した行テキストを格納するバッファのアドレス
 e. lBuf - 読み出した行テキストを格納するバッファの文字数 (文字列終端(0x0)を含む)

説 明 : このイベントは、GetToken()/PeekToken()メソッドの実行中に発生します。
 C言語ソースから 順次1行分のデータを読み出してバッファに格納します。
 true を返すと、テキスト行の読み出しを継続します。
 false を返すと、テキスト行の読み出しを終了します。(結果、GetToken()/PeekToken()メソッドが false を返します)

戻り値 : true - 文字列の通知を継続する
 false - EOF／読み出し中止

17.8. サンプルプログラム

17.8.1. Sil_CToken (トークンとその属性情報表示)

次のサンプルプログラムは、C言語ソースを入力し、分解したトークンと、トークン種別、トークンフラグを表示します。



ソースプログラムを入力するには
ここにC言語ソースプログラムを
ドロップします

```

1 : using System;
2 : using System.Collections.Generic;
3 : using System.ComponentModel;
4 : using System.Data;
5 : using System.Drawing;
6 : using System.Linq;
7 : using System.Text;
8 : using System.Windows.Forms;
9 : using CAjrCustCtrl;
10 :
11 : namespace Sil_CToken
12 : {
13 :     public partial class Form1 : Form
14 :     {
15 :         public Form1()
16 :         {
17 :             InitializeComponent();
18 :         }
19 :         // 初期設定
20 :         private void Form1_Load(object sender, EventArgs e)
21 :         {
22 :             //----- ヘッダテキスト出力 -----//
23 :             vthTtl.PutText("    S U R V S P D S P P M M M M W           \n");
24 :             vthTtl.PutText("    Y S S A T A E P R P A A A A I           \n");
25 :             vthTtl.PutText("    M R V L R T L A E T C C C C D           \n");
26 :             vthTtl.PutText("    B S S U I H I C P O N W A B E           \n");
27 :             vthTtl.PutText("    O Y Y E N N M E R P A I R O C           \n");
28 :             vthTtl.PutText("    L M M | G A I | O | M T G D H           \n");
29 :             vthTtl.PutText("    | | | | M T | | | E H | Y A           \n");
30 :             vthTtl.PutText("    | | | | E | | | | A | | R           \n");
31 :             vthTtl.PutText("Token | | | | | | | | | R | | |           \n");
32 :             vthTtl.PutText("Code  | | | | | | | | | G | | | TokenString");
33 :
34 :             vthLog.PutText("\n");
35 :             vthLog.PutText(" C言語ソースプログラムの字句を解析します。 \n");
36 :             vthLog.PutText("ここにソースファイルをドロップしてください。 \n");
37 :         }
38 :         // イベント (1行入力)
39 :         private bool cAjrCToken1_OnGetS(object sender, CAjrCustCtrl.CtkArgGetS e)
40 :         {

```

```

41 :         bool rc = false;
42 :         rc = txfInp.GetS(e.pBuf, e.lBuf);
43 :         return rc;
44 :     }
45 :
46 :     private void vthLog_OnFileDrop(object sender, VthArgFileDrop e)
47 :     {
48 :         string syl;
49 :         ECtkCode tkn;
50 :         ECtkFlg flg;
51 :
52 :         do {
53 :             // 画面クリアー
54 :             vthLog.Purge();
55 :             // 入力ファイルエンコード設定
56 :             txfInp.TextEncodeAtRead = ETextEncode.TEC_AUTO;
57 :             // ファイルオープン
58 :             txfInp.Open(vthLog.GetDroppedFile());
59 :             // トークン入力ループ
60 :             while (ctk.GetToken(out syl)) {
61 :                 tkn = ctk.CurToken;
62 :                 vthLog.PutText(((uint)tkn).ToString("X4"));
63 :                 vthLog.PutText(" ");
64 :                 vthLog.PutText(ctk.IsSymbol(tkn) ? "1" : "."); vthLog.PutText(" ");
65 :                 vthLog.PutText(ctk.IsUserSymbol(tkn) ? "1" : "."); vthLog.PutText(" ");
66 :                 vthLog.PutText(ctk.IsReservedSymbol(tkn) ? "1" : "."); vthLog.PutText(" ");
67 :                 vthLog.PutText(ctk.IsIsNumericValue(tkn) ? "1" : "."); vthLog.PutText(" ");
68 :                 vthLog.PutText(ctk.IsString(tkn) ? "1" : "."); vthLog.PutText(" ");
69 :                 vthLog.PutText(ctk.IsPathName(tkn) ? "1" : "."); vthLog.PutText(" ");
70 :                 vthLog.PutText(ctk.IsDelimiter(tkn) ? "1" : "."); vthLog.PutText(" ");
71 :                 vthLog.PutText(ctk.IsSpace(tkn) ? "1" : "."); vthLog.PutText(" ");
72 :
73 :                 flg = ctk.CurFlag;
74 :                 vthLog.PutText((flg & ECtkFlg.PREPRO) != 0 ? "1" : "."); vthLog.PutText(" ");
75 :                 vthLog.PutText((flg & ECtkFlg.PPTOP) != 0 ? "1" : "."); vthLog.PutText(" ");
76 :                 vthLog.PutText((flg & ECtkFlg.MACNAME) != 0 ? "1" : "."); vthLog.PutText(" ");
77 :                 vthLog.PutText((flg & ECtkFlg.MACWITHARG) != 0 ? "1" : "."); vthLog.PutText(" ");
78 :                 vthLog.PutText((flg & ECtkFlg.MACARG) != 0 ? "1" : "."); vthLog.PutText(" ");
79 :                 vthLog.PutText((flg & ECtkFlg.MACBODY) != 0 ? "1" : "."); vthLog.PutText(" ");
80 :                 vthLog.PutText((flg & ECtkFlg.WIDECHAR) != 0 ? "1" : "."); vthLog.PutText(" ");
81 :                 vthLog.PutText("'" + syl + "'\n");
82 :
83 :                 Application.DoEvents();
84 :             }
85 :             // ファイルクローズ
86 :             txfInp.Close();
87 :         } while(false);
88 :
89 :     }
90 : }
91 : }

```

17.8.2. Sil_CTokenC (コメント除去 - コンソールアプリ)

以下のサンプルプログラム (コンソールアプリ) は、C言語ソースを入力し、コメントを除去したテキストをコンソールに出力します。尚、行番号は元ソースと同じにします。

元ソース

```

1 : #pragma warning(disable:4996)
2 : #include <AjrCstXX.h>
3 : //
4 : // 先頭フォルダ下にある <削除フォルダ>下の全ファイルを削除する>
5 : //
6 : // DelUnderDir <先頭フォルダ> <削除フォルダ名> ...
7 : //
8 : // フォルダ名の末尾が、削除フォルダと等しいフォルダ下の全ファイルを削除します。
9 : //
10 : // ex. DelUnderDir d:\work sub1 sub2
11 : //
12 : // →「d:\work」下のサブフォルダ「sub1」と「sub2」内の全ファイルを削除する
13 : //
14 : BOOL CALLBACK cbFind(UI nest, BCP pPath, BCP pName, UI attrib, UI wtime, UX cbp)
15 : {
16 :     BCP pDelDir = (BCP)cbp;
17 :
18 :     if (!(attrib & _A_SUBDIR)) {
19 :         UI lCur, lDel;
20 :         BC CurDir[MAX_PATH];
21 :         BC DelDir[MAX_PATH];
22 :
23 :         // 削除する DIR の末尾文字列と長さ設定
24 :         AjcSnPrintf(DelDir, sizeof DelDir, "%s%s", pDelDir);
25 :         lDel = strlen(DelDir);
26 :         // 検索した DIR の文字列と長さ設定
27 :         _splitpath(pPath, NULL, CurDir, NULL, NULL);
28 :         lCur = (UI)strlen(CurDir);
29 :
30 :         if (lCur > lDel && stricmp(&CurDir[lCur - lDel], DelDir) == 0) {
31 :             DeleteFile(pPath);
32 :             printf("Delete %s\n", pPath);
33 :         }
34 :     }
35 :     return TRUE;
36 : }
37 :
38 :
39 :
40 : int main(int argc, char *argv[])
41 : {
42 :     int i;
43 :
44 :     if (argc >= 3) {
45 :         for (i = 2; i < argc; i++) {
46 :             AjcSearchFiles(argv[i], ".*", TRUE, (UX)argv[i], cbFind);
47 :         }
48 :     }
49 :
50 :     return 0;
51 : }

```

実行結果

```

1 : #pragma warning(disable:4996)
2 : #include<AjrCstXX.h>
3 :
4 :
5 :
6 :
7 :
8 :
9 :
10 :
11 :
12 :
13 :
14 :
15 : BOOL CALLBACK cbFind(UI nest,BCP pPath,BCP pName,UI attrib,UI wtime,UX cbp)
16 : {
17 :     BCP pDelDir=(BCP)cbp;
18 :
19 :     if(!(attrib&_A_SUBDIR)){
20 :         UI lCur,lDel;
21 :         BC CurDir[MAX_PATH];
22 :         BC DelDir[MAX_PATH];
23 :
24 :         AjcSnPrintf(DelDir,sizeof DelDir,"%s%s",pDelDir);
25 :         lDel=strlen(DelDir);
26 :
27 :         _splitpath(pPath,NULL,CurDir,NULL,NULL);
28 :         lCur=(UI)strlen(CurDir);
29 :
30 :         if(lCur>lDel&&stricmp(&CurDir[lCur-lDel],DelDir)==0){
31 :             DeleteFile(pPath);
32 :             printf("Delete %s\n",pPath);
33 :         }
34 :     }
35 :     return TRUE;
36 : }
37 :
38 :
39 :
40 : int main(int argc,char*argv[])
41 : {
42 :     int i;
43 :
44 :     if(argc>3){
45 :         for(i=2;i<argc;i++){
46 :             AjcSearchFiles(argv[i],".*.",TRUE,(UX)argv[i],cbFind);
47 :         }
48 :     }
49 :
50 :     return 0;
51 : }

```

```

1 : //
2 : // Sil_CTokenC
3 : //
4 : using System;
5 : using System.IO;
6 : using System.Reflection;
7 : using System.Collections.Generic;
8 : using System.Linq;
9 : using System.Text;
10 : using System.Runtime.InteropServices;
11 : using CAjrCustCtrl;
12 :
13 : namespace Sil_CTokenC
14 : {
15 :     class Program
16 :     {
17 :         static CAjrTextFile txInp = new CAjrTextFile();
18 :         static CAjrCToken ctk = new CAjrCToken();
19 :         static string InpFile;
20 :         static CAjrStatic sta = new CAjrStatic();
21 :         static CtkCbGetS m_CtkCbGetS;
22 :
23 :         static void Main(string[] args)
24 :         {
25 :             string s;
26 :             int lno, svLno = 0;
27 :             ECtkCode tkn, SvTkn = (ECtkCode)0;
28 :             ECtkFlg flg, SvFlg = (ECtkFlg)0;
29 :
30 :             // コンソールアプリ終了ハンドラ登録
31 :             m_CbkConApExit = new CbkConApExit(SsvConApExit);
32 :             SetConsoleCtrlHandler(m_CbkConApExit, true);
33 :
34 :             // コールバックメソッド設定
35 :             m_CtkCbGetS = new CtkCbGetS(cbGetS);
36 :             ctk.SetCallBack(m_CtkCbGetS);
37 :             // 入力ソースのフルパス名設定
38 :             Assembly myAssembly = Assembly.GetEntryAssembly();

```

```

39 : InpFile = Path.GetFullPath(Path.GetDirectoryName(myAssembly.Location) + @"¥Sample.c");
40 : // 入力ファイルエンコード設定
41 : txfInp.TextEncodeAtRead = ETextEncode.TEC_AUTO;
42 : // 開始メッセージ
43 : Console.WriteLine("¥nC言語ソース(" + InpFile + ")からコメントを削除して表示します。¥nEnterキーを押すと開始します。");
44 : Console.ReadLine();
45 : do {
46 :     // ファイルオープン
47 :     try {txfInp.Open(InpFile);}
48 :     catch (Exception e) {Console.WriteLine(e.ToString() + "¥n"); break;}
49 :     // トークン入力ループ
50 :     while (ctk.GetToken(out s)) {
51 :         // 行番号, トークンコード, フラグ情報設定
52 :         lno = ctk.CurLineNumber;
53 :         tkn = ctk.CurToken;
54 :         flg = ctk.CurFlag;
55 :         // 行番号が変わったら改行する
56 :         if (lno != svLno) {
57 :             for (int i = svLno; i < lno; i++) {
58 :                 // 改行
59 :                 if (i != 0) Console.WriteLine("");
60 :                 // 行番号表示
61 :                 Console.Write((i+1).ToString().PadLeft(5) + " : ");
62 :             }
63 :             // 行頭合わせ
64 :             int pos = ctk.CurPosition;
65 :             Console.Write("".PadLeft((int)pos));
66 :         }
67 :         // 行番号が同じならば、語句を表示
68 :         else {
69 :             // 語句間空白 (連続したシンボル/数値定数 or マクロボディ先頭)
70 :             if (ctk.IsValSym(SvTkn) && ctk.IsValSym(tkn) ||
71 :                 ((SvFlg & ECtkFlg.MACBODY) == 0 && (flg & ECtkFlg.MACBODY) != 0)) {
72 :                 Console.Write(" ");
73 :             }
74 :         }
75 :         // 語句表示
76 :         Console.Write(s);
77 :         // 行番号, トークンコード, フラグ情報退避
78 :         svLno = lno;
79 :         SvTkn = tkn;
80 :         SvFlg = flg;
81 :     }
82 :     // ファイルクローズ
83 :     txfInp.Close();
84 : } while(false);
85 :
86 : // インスタンス消去
87 : ctk.Delete();
88 :
89 : Console.Write("¥n¥nHit Enter key -");
90 : Console.ReadLine();
91 : }
92 : // コールバック (1行入力)
93 : static private bool cbGetS(IntPtr pBuf, int lBuf, IntPtr cbp)
94 : {
95 :     bool rc;
96 :     rc = txfInp.GetS(pBuf, lBuf);
97 :     return rc;
98 : }
99 : // コンソールアプリ終了ハンドラ用デリゲート
100 : [DllImport("Kernel32")]
101 : static extern bool SetConsoleCtrlHandler(CbkConApExit Handler, bool Add);
102 : delegate bool CbkConApExit(EAJCEXITTYPE ExitType);
103 : static CbkConApExit m_CbkConApExit;
104 : // コンソールアプリ終了ハンドラ
105 : static bool SsvConApExit(EAJCEXITTYPE ExitType)
106 : {
107 :     // インスタンス消去
108 :     ctk.Delete();
109 :     // false : 次のイベントハンドラへリンクする
110 :     return false;
111 : }
112 : }
113 : }

```

18. C言語のプリコンパイル (CAjrCPrePro.dll)

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	• #ifndef
• #define	• #elif
• #if	• #else
• #ifdef	• #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、指定したファイルに出力されます。

例えば、以下の①の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に示すテキストファイルとして出力されます。

①

```
#define MAC(A, B) printf("%d, %d\n", A, B)
main( ) // 主制御
{
    #ifdef MAC
        MAC(10, 20);
    #else
        printf("MAC not defined\n");
    #endif
}
```



② 出力ファイルに、以下のようなソーステキストが作成されます。

```
// #define MAC(A, B) printf("%d, %d\n", A, B)
main()
{
//     #ifdef MAC                      // --> TRUE
        printf("%d, %d\n", 10, 20);
//     #else                          // --> FALSE
//~     printf("MAC not defined\n");
//     #endif                          // --> 4
}
```

18.1. プリコンパイルオプション

Option プロパティにより、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については、Option プロパティを参照してください。

18.1.1. インクルードファイル自動検索 (AUTOSRH)

インクルードファイルをベースフォルダの以下のサブフォルダ群から自動的に検索します。
ベースフォルダは、SetBasePath() メソッドで指定します。
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。

18.1.2. 同一インクルードファイルを1回だけ読み出す (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。

18.1.3. 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 (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)

```

. . . . .
/* 2; 0; crtdefs.h;      29; */ // #if defined( _midl)           // --> FALSE
/* 2; 0; crtdefs.h;      31; */ // - #undef _CRT_SECURE_CPP_OVERLOAD_STANDARD_NAMES
/* 2; 0; crtdefs.h;      32; */ // - #undef _CRT_SECURE_CPP_OVERLOAD_STANDARD_NAMES_COUNT
/* 2; 0; crtdefs.h;      33; */ // - #undef _CRT_SECURE_CPP_OVERLOAD_SECURE_NAMES
/* 2; 0; crtdefs.h;      34; */ // - #undef _CRT_SECURE_CPP_OVERLOAD_STANDARD_NAMES_MEMORY
/* 2; 0; crtdefs.h;      35; */ // - #undef _CRT_SECURE_CPP_OVERLOAD_SECURE_NAMES_MEMORY
/* 2; 0; crtdefs.h;      36; */ // - #define _CRT_SECURE_CPP_OVERLOAD_STANDARD_NAMES 0
/* 2; 0; crtdefs.h;      37; */ // - #define _CRT_SECURE_CPP_OVERLOAD_STANDARD_NAMES_COUNT 0
/* 2; 0; crtdefs.h;      38; */ // - #define _CRT_SECURE_CPP_OVERLOAD_SECURE_NAMES 0
/* 2; 0; crtdefs.h;      39; */ // - #define _CRT_SECURE_CPP_OVERLOAD_STANDARD_NAMES_MEMORY 0
/* 2; 0; crtdefs.h;      40; */ // - #define _CRT_SECURE_CPP_OVERLOAD_SECURE_NAMES_MEMORY 0
/* 2; 0; crtdefs.h;      41; */ // #endif                       // --> 29
. . . . .

```

生成されなかったコードは先頭に「//」を付加しコメント出力

このオプションは、デフォルトで無効となっています。

18.2. プリコンパイル結果のファイル出力

プリコンパイル結果をファイルへ出力する際は、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。

18.2.1. インクルードファイルの展開／非展開

fExpInc プロパティを true とした場合、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 プロパティを false とした場合は、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;

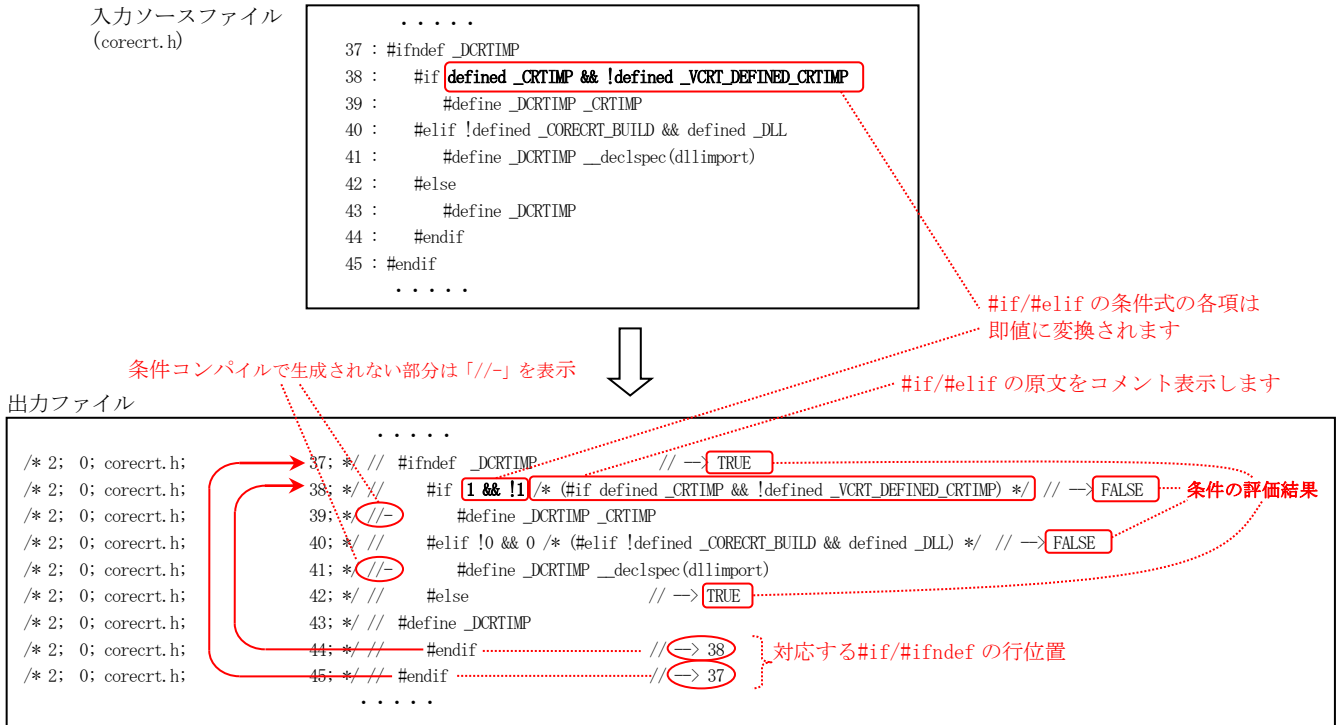
    // #ifdef _DEBUG // --> FALSE
    // printf("_MSC_VER = %d¥n",_MSC_VER);
    // #endif // --> 20

    printf("Outer = (%.3f, %.3f, %.3f)¥n",outer.x,outer.y,outer.z);

    return 0;
}
```

18.2.2. 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です。
 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
- #ifdef
- #ifndef
- #elif
- #endif
- #define

18.3. 構造体／列挙体 (定数)

18.3.1. プリコンパイル オプション

```
public enum EPpcOption {
    NONE          = 0x0000,          // オプション無し
    AUTOSRH        = 0x0001,          // インクルードファイル自動検索フラグ
    ONCE           = 0x0002,          // 同一インクルードファイルを1回だけ読み出すフラグ
    GENALL         = 0x0004,          // 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
    AUTOSRH_ONCE   = (AUTOSRH | ONCE ),
    ONCE_GENALL    = (ONCE | GENALL),
    AUTOSRH_GENALL = (AUTOSRH | GENALL),
    ALL            = (AUTOSRH | ONCE | GENALL),
}
```

18.3.2. プリコンパイル結果

```
public enum EPpcResult {
    OK          = 0 ,          // OK
    NOFILE      ,          // ファイル無し
    MEMERR      ,          // メモリエラー
    STOP        ,          // 中止
    NOTOKEN     ,          // トークン無し
    PARAM       ,          // パラメタエラー
    OUTFILE     = 99,          // 出力ファイル生成失敗
}
```

18.3.3. 通知コード (イベント識別コード)

```
public enum EPpcNtc {
    FILE_LNO      = 1 ,          // ファイル名, 行番号通知
    SRH_START     ,          // インクルードファイル検索開始通知
    SRH_DIR       ,          // インクルードファイル検索中のフォルダ通知
    SRH_END       ,          // インクルードファイル検索終了通知
    OPTSYM        ,          // プリプロセス用オプションシンボル通知
    MACDEF        ,          // マクロ定義通知
    MACREF        ,          // マクロ参照通知
    OUTLOOP       ,          // トークンストリームをファイルへ出力ループ中
    SRCTEC        ,          // ソースファイルのエンコード通知
}
```

18.3.4. エラーコード

```

public enum EPpcErr {
    //--- 一般エラー -----//
    ERROR_OK          = 0 ,    // 正常
    ERROR_SRC_OPEN    ,    // ソースファイルをオープンできません
    ERROR_INC_OPEN    ,    // INCLUDE ファイルをオープンできません
    ERROR_NO_SYMBOL    ,    // #ifdef/#ifndef でシンボル名が指定されていません
    ERROR_COND_NOTCLS  ,    // 条件(#if~#endif)がクローズされていません
    ERROR_COND_DEEP    ,    // 条件(#if~#endif)のネストが深すぎます
    ERROR_NOT_IN_IF    ,    // 対応する「#if / #ifdef / #ifndef」がありません
    ERROR_ELIF_IN_ELSE ,    // 「#else」条件中に「#elif」は記述できません
    ERROR_ELSE_IN_ELSE ,    // 「#else」条件中に「#else」は記述できません
    //--- defined -----//
    DEFERR_INVALID    = 1000 , // defined の構文誤り
    DEFERR_NEED_LP     ,    // defined' の後に左括弧 '(' が必要です
    DEFERR_NEED_SYMBOL ,    // defined' でシンボルが指定されていません
    DEFERR_NEED_RP     ,    // シンボルの次に右括弧 ')' が必要です
    //--- MACRO -----//
    MACERR_NEED_LP     = 2000 , // マクロ名の後に左括弧が必要です
    MACERR_NEED_RP     ,    // 仮引数の後に右括弧が必要です
    MACERR_NEED_RP_C   ,    // 仮引数の後に右括弧かカンマが必要です
    MACERR_MULTDEF     ,    // マクロ二重定義
    MACERR_INV_NAME    ,    // 不正なマクロ名です
    MACERR_NO_NAME     ,    // マクロ名がありません
    MACERR_NEST_OVER   ,    // マクロ展開ネストオーバー
    MACERR_ARG_LACK    ,    // マクロの引数が少なすぎる
    MACERR_ARG_OVER    ,    // マクロの引数が多すぎる
    //--- INCLUDE -----//
    INCERR_NO_FILE     = 3000 , // Include ファイル名が指定されていません
    INCERR_INV_FILE    ,    // Include ファイルの記述が不正です
    INCERR_NOTCLS      ,    // Include ファイル名の後に ' > ' がありません
    INCERR_NEST_OVER   ,    // Include ファイルのネストオーバー
    INCERR_NOT_FOUND   ,    // Include ファイルが見つかりません
    //--- 数式評価エラー -----//
    SCLERR_INVSYL      = 5000 , // 無効な語句
    SCLERR_DIVZERO     ,    // ゼロ除算
    SCLERR_NOTCLS      ,    // 括弧が閉じられていない
    SCLERR_EXPRESSION  ,    // 不当な数式表現
    SCLERR_EOL         ,    // 数式表現が途中で終了している
    SCLERR_OVERNEST    ,    // 数式表現のネストオーバー
    //--- メモリ割り当て失敗 ---//
    ERROR_MEMALLOC     = 9999   // メモリ割り当て失敗
}

```

18.4. プロパティ

C言語プリコンパイルのプロパティ一覧を示します。

#	名称	タイプ	内容
1	Option	EPpcOption	プリコンパイル動作の指定 EPpcOption. NONE : オプション無し EPpcOption. AUTOSRH : インクルードファイル自動検索 (※1) EPpcOption. ONCE : 同一インクルードファイルを1回だけ読み出す EPpcOption. 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 EPpcOption. AUTOSRH_ONCE : AUTOSRH ONCE EPpcOption. ONCE_GENALL : ONCE GENALL EPpcOption. AUTOSRH_GENALL : AUTOSRH GENALL EPpcOption. ALL : AUTOSRH ONCE GENALL (デフォルト)
2	fExpInc	bool	true : 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。(デフォルト) false: 元ソースのプリコンパイル結果のみファイルへ出力します。
3	TextEncodeAtRead	ETextEncode	入力ソースファイルのテキストエンコード ETextEncode. TEC_MBC : マルチバイト (日本語時はS-JIS) ETextEncode. TEC_UTF_8 : UTF-8 ETextEncode. TEC_EUC_J : 日本語 EUC ETextEncode. TEC_UTF_16LE : UTF-16 (Little Endian) ETextEncode. TEC_UTF_16BE : UTF-16 (Big Endian) ETextEncode. TEC_AUTO : 自動判別 (デフォルト)
4	ErrorMessaageText	string	エラーメッセージテキスト (読み出し専用) エラー発生個所 (ファイル名, 行番号) が特定できている場合は、先頭にファイルパス名と行番号が付加されます。 ・エラー発生個所が特定できる場合 : 「“ファイルパス名”<nn>: エラーメッセージ」 ・エラー発生個所が特定できない場合 : 「エラーメッセージ」

※1 : 「インクルードファイル自動検索」とした場合、インクルードファイルを、設定したインクルードパス群とベースパスの下 (サブディレクトリを含む) から自動的に検索します。また、インクルードパス群にワイルドカードを指定可能となります。

18.5. メソッド

C言語プリコンパイルの、メソッド一覧を以下に示します。

#	メソッド名	内容
1	SetBasePath	ベース・パスの設定
2	SetOptSym	オプションシンボル群の設定
3	SetIncPath	インクルード・パス群の設定
4	PreCompile	プリコンパイルの実行
5	Stop	プリコンパイルの中止
6	Delete	インスタンスの消去
7	SetCbNtcFileLno SetCbNtcOptSym SetCbNtcMacDef SetCbNtcMacRef SetCbNtcOutput SetCbNtcError SetCbNtcSrcTec	コールバックメソッドの設定

18.5.1. ベースパスの設定 (SetBasePath)

形 式 : void SetBasePath (string BasePath);

引 数 : BasePath - ベースパス (null を指定した場合はカレントディレクトリ)

説 明 : インクルードファイル検索用のベースパスを設定します。
Option プロパティで、EPpcOption. AUTOARH が指定されている場合、インクルードファイルをベースフォルダ以下のサブフォルダ群から自動的に検索します。
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。
また、ソースファイル等を相対パスで指定した場合は、ベースフォルダからの相対パスとなります。

戻り値 : なし

18.5.2. オプションシンボル群の設定 (SetOptSy)

形 式 : void SetOptSy (string[] OptSym);

引 数 : OptSym - オプションシンボル群

説 明 : プリコンパイル用オプションシンボルを文字列の配列で指定します。
オプションシンボルとは、#if, #elif, #ifdef or #ifndef で参照するシンボルを意味します。
オプションシンボルに値を設定する場合は、「オプションシンボル=値」のように、「=」に続けて値を指定します。

戻り値 : なし

18.5.3. インクルードパス群の設定 (SetIncPath)

形 式 : void SetIncPath(string[] IncPath);

引 数 : IncPath - インクルードパス群 (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 : プリコンパイル用インクルードパス群を文字列の配列で指定します。
プロパティ (EPpcOption. AUTOARH) を指定した場合、インクルードパスにワイルドカードが指定可能となります。

戻り値 : なし

18.5.4. プリコンパイルの実行 (PreCompile)

形 式 : EPpcResult PreCompile (string SrcPath, string OutPath);

引 数 : SrcPath - 入力ソースファイルのパス名
 OutPath - プリコンパイル結果を出力するファイルのパス名 (出力しない場合は、null か空文字列(“”)を指定する)

説 明 : SrcPath で指定されたファイルを読み出して、プリコンパイルを実行します。
 プリコンパイル結果は、OutPath で指定されたファイルへ出力されます。
 このメソッドは、プリコンパイルが終了するが、中止され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、以下のイベントの実行中に Stop() メソッドを呼び出します。

- OnNtcFileLno() - 現在実行中のファイルと行番号を通知するイベント
- OnNtcOutput() - プリコンパイル結果をファイルへ出力中であることを通知するイベント

SrcPath や OutPath 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なります。

戻り値 : EPpcResult. OK - 正常終了
 EPpcResult. NOFILE - ファイルなし
 EPpcResult. MEMERR - メモリエラー
 EPpcResult. STOP - 中止
 EPpcResult. PARAM - パラメタエラー
 EPpcResult. OUTFILE - 出力ファイル生成失敗

18.5.5. プリコンパイルの中止 (Stop)

形 式 : void Stop ();

引 数 : なし

説 明 : プリコンパイルを中止します。
 このメソッドを実行すると、PreCompile() メソッドの実行を中止し、呼び出し元に戻ります。

戻り値 : なし

備 考 : Stop() メソッドは、PreCompile() メソッドの実行中に呼び出す必要があります。
 そのため、PreCompile() メソッド実行中に頻繁には発生するイベント (OnNtcFileLno() / OnNtcOutput()) のイベント処理の中で呼び出すか、あるいは、PreCompile() メソッドを呼び出したスレッドとは別のスレッドから呼び出す必要があります。

18.5.6. インスタンスの消去 (Delete)

形 式 : void Delete();

引 数 : なし

説 明 : C言語プリコンパイルのインスタンスを消去します。
 コンソールアプリの場合プログラム終了時に、このAPIを実行してください。
 Windows フォームアプリの場合は、このAPIを実行する必要はありません。(暗黙的に実行されます)

戻り値 : なし

18.5.7. コールバックメソッドの設定 (SetCbNtcXXXX)

形 式 : void SetCbNtcAnyEvt (PpcCbKntcAnyEvt cb); - いずれかのイベント発生通知用コールバックの設定
 void SetCbNtcFileLno (PpcCbKntcFileLno cb); - 現在処理中のファイル名と行番号通知用コールバックの設定
 void SetCbNtcSrhStart (PpcCbKntcSrhStart cb); - インクルードファイル検索開始通知用コールバックの設定
 void SetCbNtcSrhDir (PpcCbKntcSrhDir cb); - インクルードファイル検索ディレクトリ通知用コールバックの設定
 void SetCbNtcSrhEnd (PpcCbKntcSrhEnd cb); - インクルードファイル検索終了通知用コールバックの設定
 void SetCbNtcOptSym (PpcCbKntcOptSym cb); - オプションシンボル参照の通知用コールバックの設定
 void SetCbNtcMacDef (PpcCbKntcMacDef cb); - マクロ定義の通知用コールバックの設定
 void SetCbNtcMacRef (PpcCbKntcMacRef cb); - マクロ参照の通知用コールバックの設定
 void SetCbNtcOutput (PpcCbKntcOutput cb); - プリコンパイル結果をファイルへ出力中の通知用コールバックの設定
 void SetCbNtcError (PpcCbKntcError cb); - エラー通知用コールバックの設定
 void SetCbNtcSrcTec (PpcCbKntcSrcTec cb); - ソースファイルのテキストエンコード通知用コールバックの設定

引 数 : cb - コールバックメソッドのデリゲート・オブジェクト

説 明 : 各種コールバックメソッドを設定します。
 コンソールアプリではイベントが発生しない為、コールバックメソッドを設定することにより通知を受け取ります。

戻り値 : なし

18.6. イベント／デリゲート

C言語プリコンパイルコントロールのイベント／デリゲート一覧を以下に示します。

#	イベント名	デリゲート名	内容	備考
1	OnNtcAnyEvt	PpcCbkNtcAnyEvt	いずれかのイベント発生通知	
2	OnNtcFileLno	PpcCbkNtcFileLno	現在処理中のファイル名, 行番号通知	
3	OnNtcSrshStart	PpcCbkNtcSrshStart	インクルードファイル検索開始通知	
4	OnNtcSrshDir	PpcCbkNtcSrshDir	インクルードファイル検索ディレクトリ通知	
5	OnNtcSrshEnd	PpcCbkNtcSrshEnd	インクルードファイル検索終了通知	
6	OnNtcOptSym	PpcCbkNtcOptSym	条件コンパイル用オプションシンボル通知	
7	OnNtcMacDef	PpcCbkNtcMacDef	マクロ定義通知	
8	OnNtcMacRef	PpcCbkNtcMacRef	マクロ参照通知	
9	OnNtcOutput	PpcCbkNtcOutput	ファイルへ出力中通知	
10	OnNtcError	PpcCbkNtcError	エラー通知	
11	OnNtcSrcTec	PpcCbkNtcSrcTec	ソースファイルエンコード通知	

コンソールアプリの場合は、イベントが発生しない為、デリゲートを介してコールバックメソッドにより通知を受け取ります。
以降の説明中「パラメタ」はイベントのパラメタ形式を示します。デリゲートの場合は先頭の「e.」を削除した形式となります。

18.6.1. いずれかのイベント発生通知 (OnNtcAnyEvt)

形 式 : void OnNtcAnyEvt (object sender, PpcArgNtcFileLno e);
 delegate void PpcCbkNtcFileLno (string FileName, int lno, int nest);

パラメタ : EPpcNtc e.ntc - 通知コード (イベント識別コード)

説 明 : いずれかのイベントが発生したことを通知します。
このイベントは、他のイベントと重複して発生します。
このイベント発生後に、以降のイベント (OnNtcFileLno・・等) が発生します。

戻り値 : なし

18.6.2. 現在処理中のファイル名, 行番号通知 (OnNtcFileLno)

形 式 : void OnNtcFileLno (object sender, PpcArgNtcFileLno e);
 delegate void PpcCbkNtcAnyEvt (EPpcNtc ntc);

パラメタ : string e.FileName - ファイル名
 int e.lno - 行番号
 int e.nest - ファイルのネスト値 (0:元ソース, 1~:インクルードファイルのネスト)

説 明 : 現在処理中のファイル名と行番号を通知します。
この通知は、プリコンパイル中に頻繁に通知されます。

戻り値 : なし

18.6.3. インクルードファイル検索開始通知 (OnNtcSrshStart)

形 式 : void OnNtcSrshStart (object sender, PpcArgNtcSrshStart e);
 delegate void PpcCbkNtcSrshStart (string IncName, string TopPath);

パラメタ : string e.IncName - インクルードファイル名
 string e.TopPath - 検索開始ディレクトリパス名 (最上位ディレクトリ／ワイルドカード)

説 明 : インクルードファイルの検索を開始したことを通知します。

戻り値 : なし

18.6.4. インクルードファイル検索ディレクトリ通知 (OnNtcSrhDir)

形 式 : void OnNtcSrhDir (object sender, PpcArgNtcSrhDir e);
 delegate void PpcCbKntcSrhDir (string IncName, string DirPath);

パラメタ : string e.IncName - インクルードファイル名
 string e.DirPath - インクルードファイルを検索するディレクトリパス名

説 明 : DirPathで示すディレクトリからインクルードファイルを検索することを通知します。

戻り値 : なし

18.6.5. インクルードファイル検索終了通知 (OnNtcSrhEnd)

形 式 : void OnNtcSrhEnd (object sender, PpcArgNtcSrhDir e);
 delegate void PpcCbKntcSrhDir (string IncName, bool fFind);

パラメタ : string e.IncName - インクルードファイル名
 bool e.fFind - インクルードファイルの検索結果 (true:見つかった, false:見つからない)

説 明 : DirPathで示すディレクトリからインクルードファイルを検索することを通知します。

戻り値 : なし

18.6.6. 条件コンパイル用オプションシンボル通知 (OnNtcOptSym)

形 式 : void OnNtcOptSym (object sender, PpcArgNtcOptSym e);
 delegate void PpcCbKntcOptSym (string FilePath, int lno, string OptSym);

パラメタ : string e.FilePath - ファイルパス名
 int e.lno - 行番号
 string e.OptSym - オプションシンボル名

説 明 : #if, #elif, #ifdef あるいは #ifndef で参照しているシンボルを通知します。
 FilePathとlnoは、オプションシンボルを参照している個所を示します。
 オプションシンボルとは、例えば、以下のようなプリプロセス文の「_DEBUG」や「_MSC_VER」を示します。

```
#ifdef _DEBUG
#if _MSC_VER > 1300
```

戻り値 : なし

18.6.7. マクロ定義通知 (OnNtcMacDef)

形 式 : void OnNtcMacDef (object sender, PpcArgNtcMacDef e);
 delegate void PpcCbKntcMacDef (string FilePath, int lno, string MacName);

パラメタ : string e.FilePath - ファイルパス名
 int e.lno - 行番号
 string e.MacName - オプションシンボル名

説 明 : #defineにより定義されているマクロを通知します。
 FilePathとlnoは、マクロを定義している個所を示します。

戻り値 : なし

18.6.8. マクロ参照通知 (OnNtcMacRef)

形 式 : void OnNtcMacRef (object sender, PPpcArgNtcMacRef e);
 delegate void PpcCbKntcMacRef(string FilePath, int lno, string MacName);

パラメタ : string e.FilePath - ファイルパス名
 int e.lno - 行番号
 string e.MacName - オプションシンボル名

説 明 : マクロを参照している個所を通知します。
 FilePath と lno は、マクロを参照している個所を示します。

戻り値 : なし

18.6.9. ファイル出力中通知 (OnNtcOutput)

形 式 : void OnNtcMacRef (object sender, PpcArgNtcOutput e);
 delegate void PpcCbKntcOutput();

パラメタ : なし

説 明 : プリコンパイル結果をファイルへ出力中であることを通知します。
 この通知は、ファイル出力中に頻繁に通知されます。

戻り値 : なし

18.6.10. エラー通知 (OnNtcError)

形 式 : void OnNtcError (object sender, PpcArgNtcError e);
 delegate void PpcCbKntcError(EPpcErr err, string FilePath, int lno, string param);

パラメタ : EPpcErr e.err - エラーコード
 string e.FilePath - ファイルパス名
 int e.lno - 行番号 (行番号を特定できない場合は不定値)
 string e.param - パラメタ

説 明 : プリコンパイル中にエラーを検出したことを通知します。
 FilePath と lno は、エラーを検出した個所を示します。
 param は、エラーの詳細情報で、以下のとおりです。

#	エラーコード(EPpcErr)	内容	param
1	AJCPPC_ERROR_INC_OPEN	INCLUDE ファイルをオープンできません	インクルードファイルのパス名
2	AJCPPC_DEFERR_NEED_RP	シンボルの次に右括弧 ')' が必要です	シンボル名
3	AJCPPC_MACERR_NEED_LP	マクロ名の後に左括弧が必要です	マクロ名
4	AJCPPC_MACERR_NEED_RP	仮引数の後に右括弧が必要です	仮引数名
5	AJCPPC_MACERR_NEED_RP_C	仮引数の後に右括弧かカンマが必要です	仮引数名
6	AJCPPC_MACERR_MULTDEF	マクロ二重定義	マクロ名
7	AJCPPC_MACERR_ARG_LACK	マクロの引数が少なすぎる	マクロ名
8	AJCPPC_MACERR_ARG_OVER	マクロの引数が多すぎる	マクロ名
9	AJCPPC_MACERR_ARG_INVALID	不正なマクロ引数があります	マクロ名
10	AJCPPC_INCERR_NOT_FOUND	Include ファイルが見つかりません	インクルードファイル名
11	AJCPPC_SCLERR_INVSYL	無効な語句	語句の文字列

上記以外のエラーでは、param は設定されません。(空文字列("")が設定されます)

戻り値 : なし

18.6.11. ソースファイルのテキストエンコード通知 (OnNtcSrcTec)

形 式 : void OnNtcSrcTec (object sender, PpcArgNtcSrcTec e);
delegate void PpcCbKntcOutput();

パラメタ : string e.FilePath - ソースファイルパス
ETextEncode e.tec - ソースファイルのテキストエンコード (TEC_{MBC/UTF_8/EUC_J/UTF_16LE/UTF_16BE})
EBomMode e.bom - ソースファイルのBOM有無

説 明 : ソースファイルのテキストエンコード種別を通知します。

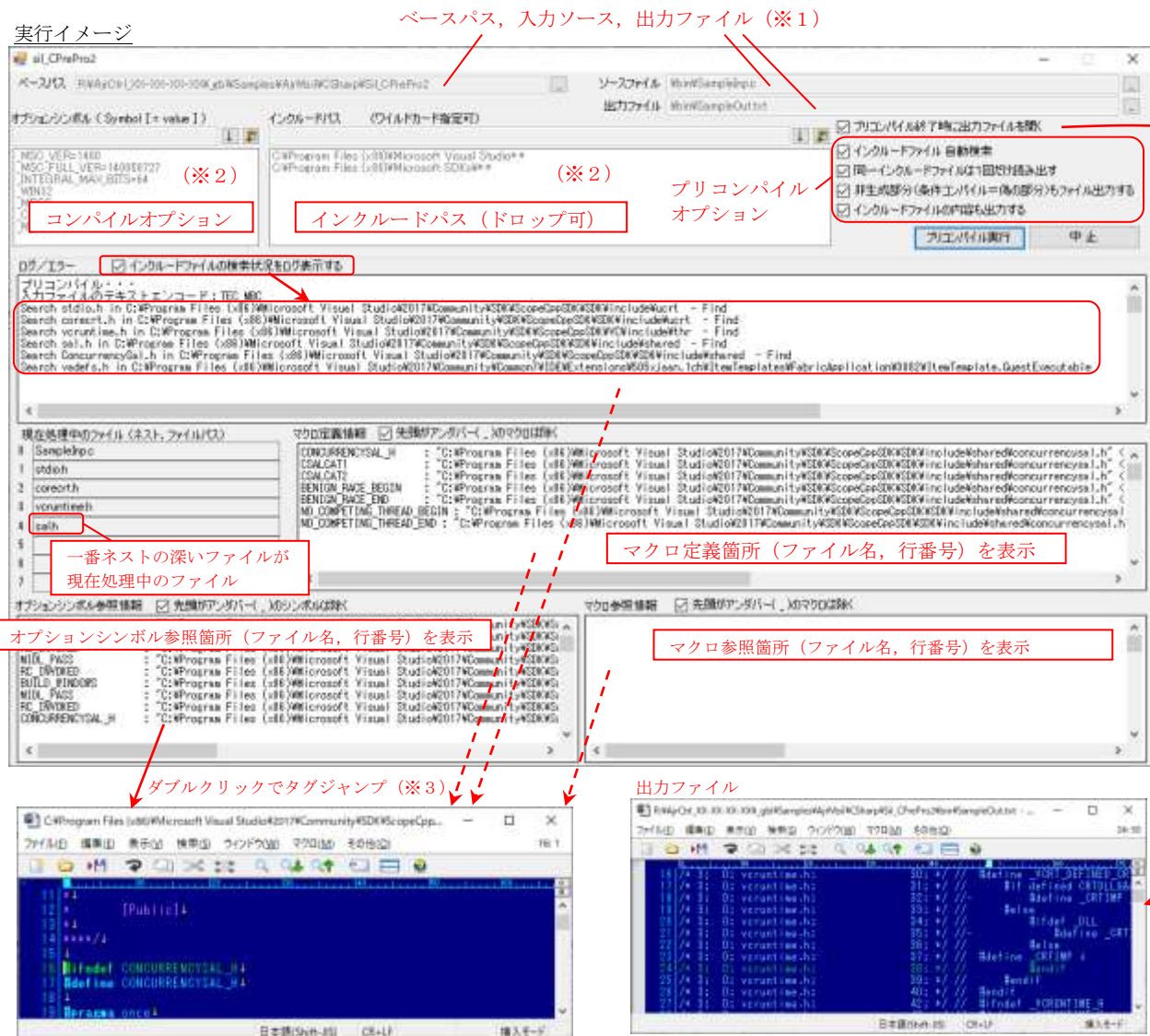
戻り値 : なし

18.7. サンプルプログラム

18.7.1. Sil_CPrePro (GUIによるプリコンパイル)

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。

- ・上部のテキストボックスに入力／編集し、「↓」キー押下
- ・「...」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。

- ・テキストボックスに入力し、「↓」キー押下
- ・クリップボード・テキストを「Ctrl+V」キーで貼り付け（テキストは各行に1項目）
- ・エクスプローラから、ドラッグ&ドロップ（インクルードパスのみ）
- ・削除する項目を選択し、Deleteキーで削除

※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() メソッドの以下の赤字部分を変更してください。

```
// タグジャンプ
private void TagJump(string line)
{
    .....
    pInfo.WorkingDirectory = @"C:\Program Files (x86)\Hidemaru\";
    pInfo.FileName = "Hidemaru.exe";
    pInfo.Arguments = @"/J" + lno.ToString() + " /m4 " + path;
    .....
}
```

```

1 : using System;
2 : using System.IO;
3 : using System.Reflection;
4 : using System.Collections.Generic;
5 : using System.Diagnostics;
6 : using System.ComponentModel;
7 : using System.Data;
8 : using System.Drawing;
9 : using System.Linq;
10 : using System.Text;
11 : using System.Windows.Forms;
12 : using CAjrCustCtrl;
13 :
14 : namespace Sil_CPrePro
15 : {
16 :     public partial class Form1 : Form
17 :     {
18 :         const string MySect = "MainSect";
19 :         TextBox[] m_FPath;
20 :         string m_SvFileName = "";
21 :         bool m_fBusy = false;
22 :         bool m_fExit = false;
23 :         int m_CurNest = 0;
24 :
25 :         public Form1()
26 :         {
27 :             InitializeComponent();
28 :         }
29 :         // フォームのロード (プログラム開始)
30 :         private void Form1_Load(object sender, EventArgs e)
31 :         {
32 :             m_FPath = new TextBox[] {txtF0, txtF1, txtF2, txtF3, txtF4, txtF5, txtF6, txtF7};
33 :             LoadProfile();
34 :             lblWild.Visible = chkAutoSrh.Checked;
35 :             // ツールチップ設定
36 :             SAjrTip.Add(txtBase, "インクルードファイルのベースフォルダ¥n" +
37 :                 "「インクルードファイル自動検索」をチェックした場合、" +
38 :                 "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¥n" +
39 :                 "フォルダをドロップするか、右の「...」ボタンで設定します");
40 :             SAjrTip.Add(cmdBase, "ダイアログにより、ベースパスを設定します");
41 :             SAjrTip.Add(txtSrc, "入力ソースプログラムファイル¥n" +
42 :                 "ファイルをドロップするか、右の「...」ボタンで設定します");
43 :             SAjrTip.Add(cmdSrc, "ダイアログにより、ソースファイルを設定します");
44 :             SAjrTip.Add(txtOut, "プリコンパイル結果の出力ファイル¥n" +
45 :                 "ファイルをドロップするか、右の「...」ボタンで設定します");
46 :             SAjrTip.Add(txtOptSym, "オプションシンボルを入力します");
47 :             SAjrTip.Add(cmdOptSymSet, "オプションシンボルへ、左のテキストを追加します");
48 :             SAjrTip.Add(cmdOptSymPaste, "クリップボード・テキストをオプションシンボルへ追加");
49 :             SAjrTip.Add(lbxOptSym, "プリコンパイル用オプションシンボル群、Del キーで選択項目削除");
50 :             SAjrTip.Add(cmdIncPathSet, "左のテキストをインクルードパスに追加");
51 :             SAjrTip.Add(cmdIncPathPaste, "クリップボード・テキストをインクルードパスに追加");
52 :             SAjrTip.Add(lbxIncPath, "設定するインクルードパスをドロップしてください、Del キーで選択項目を削除");
53 :         }
54 :         // フォームのクロー징
55 :         private void Form1_FormClosing(object sender, FormClosingEventArgs e)
56 :         {
57 :             SaveProfile();
58 :
59 :             if (m_fBusy) { // プリコンパイル実行中?
60 :                 ppc.Stop(); // プリコンパイル中止
61 :                 m_fExit = true; // プログラム終了する旨、設定
62 :                 e.Cancel = true; // フォームのクローズ中止
63 :             }
64 :         }
65 :         // ベースパス設定ボタン
66 :         private void cmdBase_Click(object sender, EventArgs e)
67 :         {
68 :             string s = SAjrGsr.GetFolder("ベースパスの設定", txtBase.Text);
69 :             if (s != "") txtBase.Text = s;
70 :         }
71 :         // ベースパス (DragEnter)
72 :         private void txtBase_DragEnter(object sender, DragEventArgs e)
73 :         {
74 :             if (e.Data.GetDataPresent(DataFormats.FileDrop)) e.Effect = DragDropEffects.All;
75 :             else e.Effect = DragDropEffects.None;
76 :         }
77 :         // ベースパス (DragDrop)
78 :         private void txtBase_DragDrop(object sender, DragEventArgs e)
79 :         {
80 :             string[] files = (string[])e.Data.GetData(DataFormats.FileDrop, false);

```

```

81 :         string fileName = files[0];
82 :         txtBase.Text    = fileName+"¥r¥n";
83 :     }
84 :     // ソースファイル設定ボタン
85 :     private void cmdSrc_Click(object sender, EventArgs e)
86 :     {
87 :         string s = SAjrGsr.GetOpenFile(txtSrc.Text, "ソースファイルの設定", "AllFiles(*.*)/*.*/*CLangFiles(*.c)/*.*.c", "c");
88 :         if (s != "") txtSrc.Text = s;
89 :     }
90 :     // ソースファイル(DragEnter)
91 :     private void txtSrc_DragEnter(object sender, DragEventArgs e)
92 :     {
93 :         if (e.Data.GetDataPresent(DataFormats.FileDrop)) e.Effect = DragDropEffects.All;
94 :         else
95 :             e.Effect = DragDropEffects.None;
96 :     }
97 :     // ソースファイル(DragDrop)
98 :     private void txtSrc_DragDrop(object sender, DragEventArgs e)
99 :     {
100 :         string[] files = (string[])e.Data.GetData(DataFormats.FileDrop, false );
101 :         string fileName = files[0];
102 :         txtSrc.Text    = fileName;
103 :     }
104 :     // 出力ファイル設定ボタン
105 :     private void cmdOut_Click(object sender, EventArgs e)
106 :     {
107 :         string s = SAjrGsr.GetSaveFile(txtOut.Text, "出力ファイルの設定", "AllFiles(*.*)/*.*/*TextFiles(*.txt)/*.*.txt", "txt");
108 :         if (s != "") txtOut.Text = s;
109 :     }
110 :     // 出力ファイル(DragEnter)
111 :     private void txtOut_DragEnter(object sender, DragEventArgs e)
112 :     {
113 :         if (e.Data.GetDataPresent(DataFormats.FileDrop)) e.Effect = DragDropEffects.All;
114 :         else
115 :             e.Effect = DragDropEffects.None;
116 :     }
117 :     // 出力ファイル(DragDrop)
118 :     private void txtOut_DragDrop(object sender, DragEventArgs e)
119 :     {
120 :         string[] files = (string[])e.Data.GetData(DataFormats.FileDrop, false );
121 :         string fileName = files[0];
122 :         txtOut.Text    = fileName;
123 :     }
124 :     // オプションシンボル追加ボタン ( ↓ )
125 :     private void cmdOptSymSet_Click(object sender, EventArgs e)
126 :     {
127 :         lbxOptSym.Items.Add(txtOptSym.Text);
128 :     }
129 :     // オプションシンボル貼り付けボタン
130 :     private void cmdOptSymPaste_Click(object sender, EventArgs e)
131 :     {
132 :         IDataObject ClipboardData = Clipboard.GetDataObject();
133 :         if (ClipboardData.GetDataPresent(DataFormats.Text))
134 :         {
135 :             char[] sep = new char[] { '¥r', '¥n' };
136 :             string[] txts = ((string)ClipboardData.GetData(DataFormats.Text)).Split(sep, StringSplitOptions.RemoveEmptyEntries);
137 :             for (int i = 0; i < txts.Length; i++) {
138 :                 lbxOptSym.Items.Add(txts[i]);
139 :             }
140 :         }
141 :     }
142 :     // オプションシンボル・リストボックス(KeyDown)
143 :     private void lbxOptSym_KeyDown(object sender, KeyEventArgs e)
144 :     {
145 :         int ix;
146 :         if ((ix = lbxOptSym.SelectedIndex) != -1 && e.KeyData == Keys.Delete) {
147 :             lbxOptSym.Items.RemoveAt(ix);
148 :         }
149 :     }
150 :     // インクルードパス追加ボタン ( ↓ )
151 :     private void cmdIncPathSet_Click(object sender, EventArgs e)
152 :     {
153 :         lbxIncPath.Items.Add(txtIncPath.Text);
154 :     }
155 :     // インクルードパス貼り付けボタン
156 :     private void cmdIncPathPaste_Click(object sender, EventArgs e)
157 :     {
158 :         IDataObject ClipboardData = Clipboard.GetDataObject();
159 :         if (ClipboardData.GetDataPresent(DataFormats.Text))
160 :         {
161 :             char[] sep = new char[] { '¥r', '¥n' };
162 :             string[] txts = ((string)ClipboardData.GetData(DataFormats.Text)).Split(sep, StringSplitOptions.RemoveEmptyEntries);

```

```

161 :         for (int i = 0; i < txts.Length; i++) {
162 :             lbxIncPath.Items.Add(txts[i]);
163 :         }
164 :     }
165 : }
166 : // インクルードパス・リストボックス (DragEnter)
167 : private void lbxIncPath_DragEnter(object sender, DragEventArgs e)
168 : {
169 :     if (e.Data.GetDataPresent(DataFormats.FileDrop)) e.Effect = DragDropEffects.All;
170 :     else e.Effect = DragDropEffects.None;
171 : }
172 : // インクルードパス・リストボックス (DragDrop)
173 : private void lbxIncPath_DragDrop(object sender, DragEventArgs e)
174 : {
175 :     string[] files = (string[])e.Data.GetData(DataFormats.FileDrop, false);
176 :     for (int i = 0; i < files.Length; i++) {
177 :         lbxIncPath.Items.Add(files[i]);
178 :     }
179 : }
180 : // インクルードパス・リストボックス (KeyDown)
181 : private void lbxIncPath_KeyDown(object sender, KeyEventArgs e)
182 : {
183 :     int ix;
184 :     if ((ix = lbxIncPath.SelectedIndex) != -1 && e.KeyData == Keys.Delete) {
185 :         lbxIncPath.Items.RemoveAt(ix);
186 :     }
187 : }
188 : // いずれかのイベント発生通知
189 : private void ppc_OnNtcAnyEvt(object sender, PpcArgNtcAnyEvt e)
190 : {
191 :     // キャンセルチェック (システムのイベント処理)
192 :     Application.DoEvents();
193 : }
194 : // プリコンパイル エラー通知
195 : private void ppc_OnNtcError(object sender, CAjrCustCtrl.PpcArgNtcError e)
196 : {
197 :     vthLog.PutText("¥r¥x1B[2K");
198 :     vthLog.PutText("¥x1B[31m");
199 :     vthLog.PutText(ppc.ErrorMessageText);
200 :     vthLog.PutText("¥x1B[0m¥n");
201 : }
202 : // プリコンパイル 現在処理中のファイル／行番号通知
203 : private void ppc_OnNtcFileLno(object sender, CAjrCustCtrl.PpcArgNtcFileLno e)
204 : {
205 :     // 現在処理中のファイル名とネスト値表示
206 :     if (e.FileName != m_SvFileName) {
207 :         if (e.nest < 8) {
208 :             if (e.nest < m_CurNest) m_FPath[m_CurNest].Text = "";
209 :             m_FPath[e.nest].Text = e.FileName;
210 :             m_CurNest = e.nest;
211 :         }
212 :         m_SvFileName = e.FileName;
213 :     }
214 :     lblLno.Text = "L#: " + e.lno.ToString();
215 : }
216 : // インクルードファイル検索開始通知
217 : private void ppc_OnNtcSrhStart(object sender, PpcArgNtcSrhStart e)
218 : {
219 :     if (chkIncSrh.Checked) {
220 :     }
221 : }
222 : // インクルードファイル検索フォルダ通知
223 : private void ppc_OnNtcSrhDir(object sender, PpcArgNtcSrhDir e)
224 : {
225 :     if (chkIncSrh.Checked) {
226 :         vthLog.PutText("¥r¥x1B[2K¥r");
227 :         vthLog.PutText("Search " + e.IncName + " in " + e.DirPath);
228 :     }
229 : }
230 : // インクルードファイル検索終了通知
231 : private void ppc_OnNtcSrhEnd(object sender, PpcArgNtcSrhEnd e)
232 : {
233 :     if (chkIncSrh.Checked) {
234 :         vthLog.PutText(e.fFind ? " - Find¥n" : " - NotFound¥n");
235 :     }
236 : }
237 : // プリコンパイル ファイル出力中通知
238 : private void ppc_OnNtcOutput(object sender, CAjrCustCtrl.PpcArgNtcOutput e)
239 : {
240 : }

```

```

241 : // プリコンパイル ソースファイルテキストエンコード通知
242 : private void ppc_OnNtcSrcTec(object sender, CAjrCustCtrl.PpcArgNtcSrcTec e)
243 : {
244 :     vthLog.PutText("¥r¥x1B[2K");
245 :     vthLog.PutText("入力ファイルのテキストエンコード : " + e.tec.ToString() + "¥n");
246 : }
247 : // プリコンパイル マクロ定義通知
248 : private void ppc_OnNtcMacDef(object sender, CAjrCustCtrl.PpcArgNtcMacDef e)
249 : {
250 :     if (!(chkMacDef.Checked && e.MacName.StartsWith("_"))) {
251 :         vthMacDef.PutText(e.MacName.PadRight(20) + " : " + "¥" + e.FilePath + "¥" < " + e.lno.ToString() + ">¥n");
252 :     }
253 : }
254 : // プリコンパイル マクロ参照通知
255 : private void ppc_OnNtcMacRef(object sender, CAjrCustCtrl.PpcArgNtcMacRef e)
256 : {
257 :     if (!(chkMacRef.Checked && e.MacName.StartsWith("_"))) {
258 :         vthMacRef.PutText(e.MacName.PadRight(20) + " : " + "¥" + e.FilePath + "¥" < " + e.lno.ToString() + ">¥n");
259 :     }
260 : }
261 : // プリコンパイル オプションシンボル参照通知
262 : private void ppc_OnNtcOptSym(object sender, CAjrCustCtrl.PpcArgNtcOptSym e)
263 : {
264 :     if (!(chkOptSym.Checked && e.OptSym.StartsWith("_"))) {
265 :         vthOptSym.PutText(e.OptSym.PadRight(20) + " : " + "¥" + e.FilePath + "¥" < " + e.lno.ToString() + ">¥n");
266 :     }
267 : }
268 : // インクルードファイルを自動検索・チェックボックス
269 : private void chkAutoSrh_CheckedChanged(object sender, EventArgs e)
270 : {
271 :     lblWild.Visible = chkAutoSrh.Checked;
272 : }
273 : // 実行ボタン
274 : private void cmdExec_Click(object sender, EventArgs e)
275 : {
276 :     EPpcResult rsu = EPpcResult.OK;
277 :     DateTime dt;
278 :     // ログクリアー
279 :     vthLog.Purge();
280 :     vthMacDef.Purge();
281 :     vthMacRef.Purge();
282 :     vthOptSym.Purge();
283 :     // プリプロセス実行中の旨、設定
284 :     m_fBusy = true;
285 :     // インクルードネスト値クリアー
286 :     m_CurNest = 0;
287 :     // 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)
288 :     SAjrGsr.EnableAllControls(this, false);
289 :     cmdCancel.Enabled = true;
290 :     // ログ全体を許可状態とする
291 :     SAjrGsr.EnableGroup(grpLog, true, true);
292 :
293 :     // ベースパス設定
294 :     ppc.SetBasePath(txtBase.Text);
295 :     // オプションシンボル設定
296 :     string[] sOptSym = lbxOptSym.Items.Cast<string>().ToArray();
297 :     ppc.SetOptSym(sOptSym);
298 :     // インクルードパス設定
299 :     string[] sIncPath = lbxIncPath.Items.Cast<string>().ToArray();
300 :     ppc.SetIncPath(sIncPath);
301 :     // プリコンパイル実行
302 :     ppc.fExpInc = chkExpInc.Checked;
303 :     ppc.Option = ((chkAutoSrh.Checked ? EPpcOption.AUTOSRH : EPpcOption.NONE) |
304 :                  (chkOnce.Checked ? EPpcOption.ONCE : EPpcOption.NONE) |
305 :                  (chkGenAll.Checked ? EPpcOption.GENALL : EPpcOption.NONE));
306 :     dt = DateTime.Now;
307 :     vthLog.PutText(dt.ToString("HH:mm:ss") + "プリコンパイル開始¥n");
308 :     rsu = ppc.PreCompile(txtSrc.Text, txtOut.Text);
309 :     lblLno.Text = "";
310 :     if (rsu != EPpcResult.OK) vthLog.PutText("¥x1B[31m");
311 :     vthLog.PutText("¥r¥x1B[2K");
312 :     dt = DateTime.Now;
313 :     vthLog.PutText(dt.ToString("HH:mm:ss"));
314 :     switch (rsu) {
315 :         case EPpcResult.OK: vthLog.PutText("プリコンパイル終了¥n"); break;
316 :         case EPpcResult.NOFILE: vthLog.PutText("ソースファイルなし¥n"); break;
317 :         case EPpcResult.MEMERR: vthLog.PutText("メモリエラー¥n"); break;
318 :         case EPpcResult.STOP: vthLog.PutText("中止しました¥n"); break;
319 :         case EPpcResult.NOTOKEN: vthLog.PutText("内容がありません¥n"); break;
320 :         case EPpcResult.PARAM: vthLog.PutText("パラメタエラー¥n"); break;

```

```

321 :         case EPpcResult.OUTFILE: vthLog.PutText("出力ファイル生成失敗\n"); break;
322 :     }
323 :     if (rsu != EPpcResult.OK) vthLog.PutText("¥x1B[0m");
324 :     // プリコンパイル実行中解除
325 :     m_fBusy = false;
326 :     // 出力ファイルを開く
327 :     if (rsu == EPpcResult.OK && chkOpenOutFile.Checked) {
328 :         try {Process.Start(txtOut.Text);}
329 :         catch (Exception ex) {vthLog.PutText("¥x1B31m" + ex.ToString() + "¥x1B[0m¥n");}
330 :     }
331 :     // 全コントロール禁止状態を解除
332 :     SAjrGsr.EnableAllControls(this, true);
333 :     cmdCancel.Enabled = false;
334 :     // 処理中のファイル名クリアー
335 :     for (int i = 0; i < 8; i++) {
336 :         m_FPath[i].Text = "";
337 :     }
338 :     // フォームクローズならば、プログラム終了
339 :     if (m_fExit) {
340 :         this.Close();
341 :     }
342 : }
343 : // 中止ボタン
344 : private void cmdCancel_Click(object sender, EventArgs e)
345 : {
346 :     cmdCancel.Enabled = false;
347 :     ppc.Stop();
348 : }
349 : // ログウインド ダブルクリック
350 : private void vthLog_OnNtcDb1Clk(object sender, VthArgDb1Clk e)
351 : {
352 :     string line = vthLog.GetDb1CkickedLineText();
353 :     TagJump(line);
354 : }
355 : // オプションシンボル参照ウインド ダブルクリック
356 : private void vthOptSym_OnNtcDb1Clk(object sender, VthArgDb1Clk e)
357 : {
358 :     string line = vthOptSym.GetDb1CkickedLineText();
359 :     TagJump(line);
360 : }
361 : // マクロ定義ウインド ダブルクリック
362 : private void vthMacDef_OnNtcDb1Clk(object sender, VthArgDb1Clk e)
363 : {
364 :     string line = vthMacDef.GetDb1CkickedLineText();
365 :     TagJump(line);
366 : }
367 : // マクロ参照ウインド ダブルクリック
368 : private void vthMacRef_OnNtcDb1Clk(object sender, VthArgDb1Clk e)
369 : {
370 :     string line = vthMacRef.GetDb1CkickedLineText();
371 :     TagJump(line);
372 : }
373 : // タグジャンプ
374 : private void TagJump(string line)
375 : {
376 :     int sPath = line.IndexOf(' ');
377 :     int ePath = line.IndexOf(' ', sPath + 1);
378 :     int sLno = line.IndexOf('<');
379 :     int eLno = line.IndexOf('>', sLno + 1);
380 :     int lno = 0;
381 :     if (sPath >= 0 && ePath >= 0 && sLno >= 0 && eLno >= 0) {
382 :         string path = line.Substring(sPath, ePath - sPath + 1);
383 :         string lStr = line.Substring(sLno + 1, eLno - sLno - 1);
384 :         int.TryParse(lStr, out lno);
385 :         if (path != "" && lno > 0) {
386 :             ProcessStartInfo pInfo = new ProcessStartInfo();
387 :             pInfo.WorkingDirectory = @"C:¥Program Files (x86)¥Hidemaru¥";
388 :             pInfo.FileName = "Hidemaru.exe";
389 :             pInfo.Arguments = @" /J" + lno.ToString() + " /m4 " + path;
390 :             try {Process.Start(pInfo);}
391 :             catch (Exception e) {vthLog.PutText("¥x1B31m" + e.ToString() + "¥x1B[0m¥n");}
392 :         }
393 :     }
394 : }
395 : // 設定値ロード
396 : void LoadProfile()
397 : {
398 :     // デフォルト値設定
399 :     // ・ベースパス (自プログラムフォルダの2つ上)
400 :     txtBase.Text = SAjrGsr.GetAppPath();

```



```

401 :         txtSrc .Text = @".¥SampleInp.c";
402 :         txtOut .Text = @".¥SampleOut.txt";
403 :         //      ・ オプションシンボル
404 :         lbxOptSym.Items.Add("_MSC_VER=1400"           );
405 :         lbxOptSym.Items.Add("_MSC_FULL_VER=140050727" );
406 :         lbxOptSym.Items.Add("_INTEGRAL_MAX_BITS=64"   );
407 :         lbxOptSym.Items.Add("_WIN32"                  );
408 :         lbxOptSym.Items.Add("_MBCS"                   );
409 :         lbxOptSym.Items.Add("_CRTBLD"                  );
410 :         lbxOptSym.Items.Add("_M_IX86"                  );
411 :         lbxOptSym.Items.Add("__STDC__=0"               );
412 :         lbxOptSym.Items.Add("__STDC_WANT_SECURE_LIB__=0");
413 :         //      ・ インクルードパス ・ リストボックス
414 :         lbxIncPath.Items.Add(@"C:¥Program Files (x86)¥Microsoft Visual Studio*.");
415 :         lbxIncPath.Items.Add(@"C:¥Program Files (x86)¥Microsoft SDKs¥*.");
416 :         //      設定値ロード
417 :         SAjrReg.LoadAllCtrls(this);
418 :
419 :     }
420 :     //      設定値セーブ
421 :     void SaveProfile()
422 :     {
423 :         SAjrReg.SaveAllCtrls(this);
424 :     }
425 : }
426 : }

```

18.7.2. Sil_CPreProC (コンソールアプリでプリコンパイル)

次のサンプルプログラム (コンソールアプリ) は、C 言語ソースを入力しプリコンパイルしたテキストを出力します。

各種設定や、入力ソースはハードコード (プログラム内で固定で指定) しています。

コンソールにマクロ定義箇所、マクロ参照箇所、オプションシンボル参照箇所を表示します。

(但し、先頭がアンダバー (_) である マクロ名や、シンボル名は除きます)

実行中にいずれかのキーを押下すると、処理を中止します。

```

Nest File
0 : SampleInp.c
1 : stdio.h
2 : corecrt.h
3 : vcruntime.h
4 : sal.h
5 : concurrencysal.h
3 : vcruntime.h
4 : vdefs.h
3 : vcruntime.h
2 : corecrt.h
1 : stdio.h
2 : corecrt_wstdio.h
3 : corecrt_stdio_config.h
2 : corecrt_wstdio.h
1 : stdio.h
0 : SampleInp.c

マクロ定義情報 -----
Def : CONCURRENT_CSPAL_H C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\SDK\ScopeCp
Def : CSALCAT1 C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\SDK\ScopeCp
Def : CSALCAT2 C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\SDK\ScopeCp
Def : BENIGN_RACE_BEGIN C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\SDK\ScopeCp
Def : BENIGN_RACE_END C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\SDK\ScopeCp
Def : NO_COMPETING_THREAD_BEGIN C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\SDK\ScopeCp
Def : NO_COMPETING_THREAD_END C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\SDK\ScopeCp
Def : NULL C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\SDK\ScopeCp
Def : stdin C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\SDK\ScopeCp
Def : stdout C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\SDK\ScopeCp
Def : stderr C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\SDK\ScopeCp
Def : WEOF C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\SDK\ScopeCp
Def : BUFSIZ C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\SDK\ScopeCp
Def : EOF C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\SDK\ScopeCp
Def : L_tmpnam C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\SDK\ScopeCp
Def : SEEK_CUR C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\SDK\ScopeCp
Def : SEEK_END C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\SDK\ScopeCp
Def : SEEK_SET C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\SDK\ScopeCp
Def : FILENAME_MAX C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\SDK\ScopeCp
Def : FOPEN_MAX C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\SDK\ScopeCp
Def : TMP_MAX C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\SDK\ScopeCp
Def : SYS_OPEN C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\SDK\ScopeCp
Def : OUTER R:\AjrCustCtrl\%_gbl\Samples\AjrMsi\%CSharp\Sil_CPreProC\bin\SampleInp.c <
Def : M_GSPUTPROFLENUM R:\AjrCustCtrl\%_gbl\Samples\AjrMsi\%CSharp\Sil_CPreProC\bin\SampleInp.c <

マクロ参照情報 -----
Ref : NULL C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\SDK\ScopeCp
Ref : NULL C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\SDK\ScopeCp
Ref : stdout C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\SDK\ScopeCp
Ref : stdout C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\SDK\ScopeCp
Ref : NULL C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\SDK\ScopeCp
Ref : stdout C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\SDK\ScopeCp

```

```

1 : //
2 : // Sil_CPreProC
3 : //
4 : using System;
5 : using System.Reflection;
6 : using System.Diagnostics;
7 : using System.Collections;
8 : using System.Collections.Generic;
9 : using System.Linq;
10 : using System.Text;
11 : using System.Runtime.InteropServices;
12 : using CAjrCustCtrl;
13 :
14 : namespace Sil_CPreProC
15 : {
16 :     class Program
17 :     {
18 :         const string InpFile = @"%.¥bin¥SampleInp.c";
19 :         const string OutFile = @"%.¥bin¥SampleOut.txt";
20 :         static CAjrCPrePro ppc = new CAjrCPrePro();
21 :         static CAjrStatic sta = new CAjrStatic();
22 :         static ConsoleKeyInfo c = new ConsoleKeyInfo();
23 :         static string m_svFileName = "";
24 :         static ETextEncode m_SrcTec = ETextEncode.TEC_AUTO;
25 :         static EBomMode m_SrcBom = EBomMode.NOT_WRITE_BOM;
26 :         static ArrayList m_OptSym = new ArrayList();
27 :         static ArrayList m_MacDef = new ArrayList();
28 :         static ArrayList m_MacRef = new ArrayList();
29 :
30 :

```

```

31 : // インクルード・パス群
32 : static string[] IncPath = {
33 :     @"C:\Program Files (x86)\Microsoft Visual Studio*.\"",
34 :     @"C:\Program Files (x86)\Microsoft SDKs*.\"",
35 : };
36 : // オプション・シンボル群
37 : static string[] OptSym = {
38 :     "_MSC_VER=1400" ,
39 :     "_MSC_FULL_VER=140050727" ,
40 :     "_INTEGRAL_MAX_BITS=64" ,
41 :     "_WIN32" ,
42 :     "_MBCS" ,
43 :     "_CRTBLD" ,
44 :     "_M_IX86" ,
45 :     "_STDC__=0" ,
46 :     "_STDC_WANT_SECURE_LIB__=0",
47 : };
48 :
49 : static void Main(string[] args)
50 : {
51 :     // コンソールアプリ終了ハンドラ登録
52 :     m_CbkConApExit = new CbkConApExit(SsvConApExit);
53 :     SetConsoleCtrlHandler(m_CbkConApExit, true);
54 :
55 :     // プリコンパイル用コールバック設定
56 :     ppc.SetCbNtcAnyEvt (new PpcCbKntcAnyEvt (cbNtcAnyEvt )); // いずれかのイベント発生通知
57 :     ppc.SetCbNtcSrcTec (new PpcCbKntcSrcTec (cbNtcSrcTec )); // ソースファイルエンコード通知
58 :     ppc.SetCbNtcError (new PpcCbKntcError (cbNtcError )); // エラー通知
59 :     ppc.SetCbNtcMacDef (new PpcCbKntcMacDef (cbNtcMacDef )); // マクロ定義通知
60 :     ppc.SetCbNtcMacRef (new PpcCbKntcMacRef (cbNtcMacRef )); // マクロ参照通知
61 :     ppc.SetCbNtcOptSym (new PpcCbKntcOptSym (cbNtcOptSym )); // オプションシンボル参照通知
62 :     ppc.SetCbNtcFileLno(new PpcCbKntcFileLno(cbNtcFileLno)); // 処理中のファイル行番号通知
63 :     ppc.SetCbNtcSrcTec (new PpcCbKntcSrcTec (cbNtcSrcTec )); // ソースファイルエンコード通知
64 :
65 :     // ベースパス設定 (自プログラムフォルダの2つ上)
66 :     Assembly myAssembly = Assembly.GetEntryAssembly();
67 :     string BasePath = System.IO.Path.GetDirectoryName(myAssembly.Location) + @"¥..";
68 :     ppc.SetBasePath(BasePath);
69 :
70 :     // インクルードパス設定
71 :     ppc.SetIncPath(IncPath);
72 :
73 :     // オプションシンボル設定
74 :     ppc.SetOptSym(OptSym);
75 :
76 :     // プリコンパイル実行
77 :     Console.WriteLine(" Nest File");
78 :     ppc.PreCompile(InpFile, OutFile);
79 :
80 :     // プリコンパイルインスタンス消去
81 :     ppc.Delete();
82 :
83 :     // マクロ定義情報表示
84 :     Console.WriteLine("¥n----- マクロ定義情報 -----");
85 :     for (int i = 0; i < m_MacDef.Count; i++) {
86 :         Console.WriteLine((string)m_MacDef[i]);
87 :     }
88 :
89 :     // マクロ参照情報表示
90 :     Console.WriteLine("¥n----- マクロ参照情報 -----");
91 :     for (int i = 0; i < m_MacRef.Count; i++) {
92 :         Console.WriteLine((string)m_MacRef[i]);
93 :     }
94 :
95 :     // オプションシンボル参照情報表示
96 :     Console.WriteLine("¥n----- オプションシンボル参照情報 -----");
97 :     for (int i = 0; i < m_OptSym.Count; i++) {
98 :         Console.WriteLine((string)m_OptSym[i]);
99 :     }
100 :
101 :     // 出力ファイルを開く
102 :     Process.Start(BasePath + @"¥¥" + OutFile);
103 :
104 :     Console.WriteLine("¥n¥n Hit Enter key -");
105 :     Console.ReadLine();
106 : }
107 : // 何らかのイベント発生通知
108 : static void cbNtcAnyEvt (EPpcNtc ntc)
109 : {
110 :     // いずれかのキー押下でプリコンパイル中止

```

```

111 :         if (Console.KeyAvailable) {
112 :             c = Console.ReadKey(true);
113 :             ppc.Stop();
114 :         }
115 :     }
116 :     // ソースファイルエンコード通知
117 :     static void cbNtcSrcTec(string FilePath, ETextEncode tec, EBomMode bom)
118 :     {
119 :         m_SrcTec = tec;
120 :         m_SrcBom = bom;
121 :     }
122 :     // エラー通知
123 :     static void cbNtcError(EPpcErr err, string FilePath, int lno, string param)
124 :     {
125 :         Console.WriteLine(ppc.ErrorMessageText);
126 :     }
127 :     // マクロ定義通知
128 :     static void cbNtcMacDef(string FilePath, int lno, string MacName)
129 :     {
130 :         if (!MacName.StartsWith("_")) {
131 :             m_MacDef.Add(" Def : " + MacName.PadRight(20) + " " + FilePath + " <" + lno.ToString() + ">");
132 :         }
133 :     }
134 :     // マクロ参照通知
135 :     static void cbNtcMacRef(string FilePath, int lno, string MacName)
136 :     {
137 :         if (!MacName.StartsWith("_")) {
138 :             m_MacRef.Add(" Ref : " + MacName.PadRight(20) + " " + FilePath + " <" + lno.ToString() + ">");
139 :         }
140 :     }
141 :     // オプションシンボル参照通知
142 :     static void cbNtcOptSym(string FilePath, int lno, string OptSym)
143 :     {
144 :         if (!OptSym.StartsWith("_")) {
145 :             m_OptSym.Add(" Opt : " + OptSym.PadRight(20) + " " + FilePath + " <" + lno.ToString() + ">");
146 :         }
147 :     }
148 :     // 処理中のファイル行番号通知
149 :     static void cbNtcFileLno(string FileName, int lno, int nest)
150 :     {
151 :         // ファイル名表示
152 :         if (m_svFileName != FileName) {
153 :             Console.WriteLine(nest.ToString().PadLeft(5) + " : " + FileName);
154 :             m_svFileName = FileName;
155 :         }
156 :     }
157 :     // コンソールアプリ終了ハンドラ用デリゲート
158 :     [DllImport("Kernel32")]
159 :     static extern bool SetConsoleCtrlHandler(CbkConApExit Handler, bool Add);
160 :     delegate bool CbkConApExit(EAJCEXITTYPE ExitType);
161 :     static CbkConApExit m_CbkConApExit;
162 :     // コンソールアプリ終了ハンドラ
163 :     static bool SsvConApExit(EAJCEXITTYPE ExitType)
164 :     {
165 :         // インスタンス消去
166 :         ppc.Delete();
167 :         // false : 次のイベントハンドラへリンクする
168 :         return false;
169 :     }
170 : }
171 : }

```

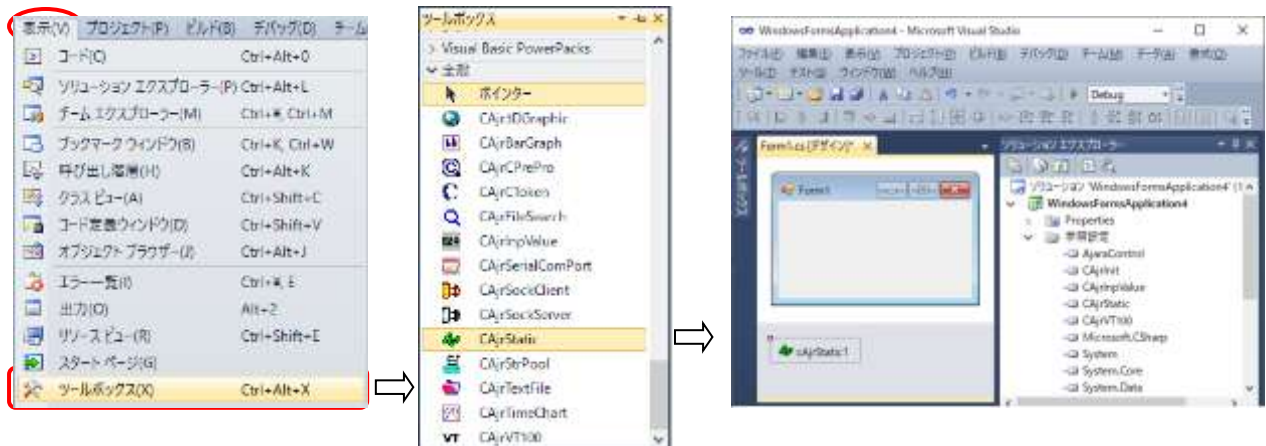
19. スタティク ファンクション (CAjrStatic.dll)

CAjrStatic.dll は、スタティク・クラスの集合モジュールです。
CAjrStatic.dll には、以下のスタティク・クラスが含まれます。

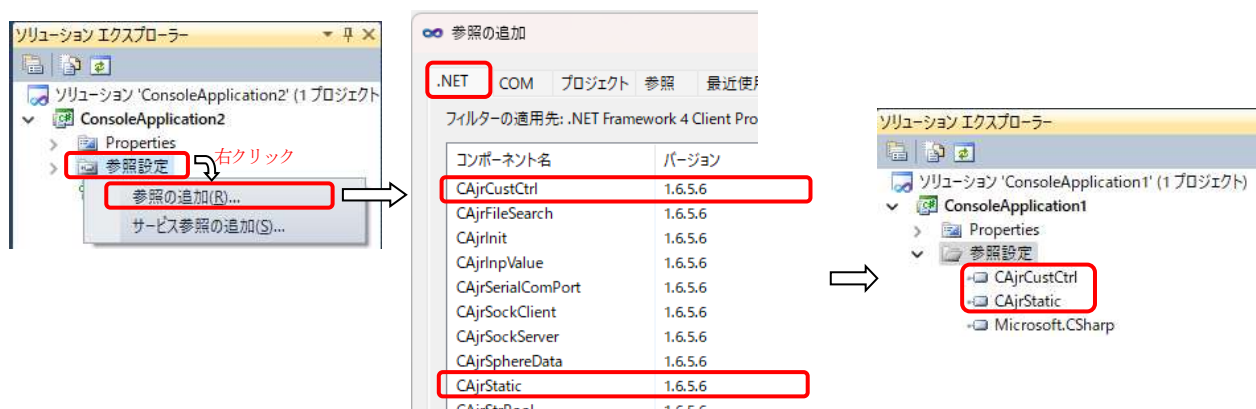
#	クラス名	内容	備考
1	SAjrTip	ツールチップ	
2	SAjrReg	プロファイル, レジストリアクセス	
3	SAjrFop	ファイル/フォルダ操作	
4	SAjrMath	演算ファンクション	3 Dベクトル演算
5	SAjrGsr	汎用ファンクション	
6	SAjrCS	チェックサム計算	
7	SAjrCrc	C R C 計算	
8	SAjrBin	ネイティブメモリアクセス	

各スタティク・クラスの機能詳細については、次章以降で説明します。

これらのスタティク・クラスのメソッドを使用するには、「CAjrStatic」への参照を追加する必要があります。
フォームアプリケーションの場合は、ツールボックス中の「CAjrStatic」をダブルクリックするか、あるいは、フォーム上へドラッグ&ドロップします。



コンソールアプリケーションの場合は、ソリューションエクスプローラの「参照設定」を右クリック→参照の追加から参照の追加ダイアログを表示し、「.NET」タブで、「CAjrCustCtrl」と「CAjrStatic」を選択します。



名前空間の参照

using (C#) / Imports (VB) で、名前空間「CAjrCustCtrl」を参照します。(using CAjrCustCtrl; / Imports CAjrCustCtrl)

19.1. 構造体／列挙体（定数）

19.1.1. 言語種別

```
public enum ELangId : int
{
    English = 0,           // 英語
    Japanese = 0x411      // 日本語
}
```

19.2. プロパティ

CAjrStatic クラスには、以下のプロパティが含まれています。

CAjrStatic クラス自身は、通常の抽象クラスですが、プロパティでアクセスする内容はスタティクな情報です。

フォームアプリケーションの場合は、デザイン時に、これらのプロパティを設定しておくことができます。

コンソールアプリケーションの場合や実行時は、対応するメソッドでアクセスします。

#	プロパティ	タイプ	内容	対応するメソッド	備考
1	DefTextColor	Color	ツールチップのデフォルト テキスト表示色	void SAjrTip.SetDefTextcolor(Color) Color SAjrTip.GetDefTextcolor()	
2	DefBorderColor	Color	ツールチップのデフォルト 外枠表示色	void SAjrTip.SetDefBorderColor(Color) Color SAjrTip.GetDefBorderColor()	
3	DefBackGround	Color	ツールチップのデフォルト ウインド背景色	void SAjrTip.SetDefBackGround(Color) Color SAjrTip.GetDefBackGround()	
4	DefFont	Font	ツールチップのデフォルトフォント	void SAjrTip.SetDefFont(Font) Font SAjrTip.GetDefFont()	
5	DefDelayTime	int	ツールチップのデフォルト 表示遅延時間	void SAjrTip.SetDefDelayTime(int) int SAjrTip.GetDefDelayTime()	
6	DefDisplayTime	int	ツールチップのデフォルト 表示時間	void SAjrTip.SetDefDisplayTime(int) int SAjrTip.GetDefDisplayTime()	
7	WindowTime	int	ツールチップのコントロール間 移動猶予時間	void SAjrTip.SetWindowTime(int) int SAjrTip.GetWindowTime()	
8	ShowForOnlyActive	bool	全コントロールでの、非アクティブ 時ツールチップ表示設定	void SAjrTip.SetShowForOnlyActive(bool) bool SAjrTip.GetShowForOnlyActive()	
9	Version	string	バージョン文字列	Version SAjrGsr.GetVersion()	読み出し専用
10	Lang	ELangId	言語種別	void SAjrGsr.SetLang(ELangId) ELangId SAjrGsr.GetLang()	Japanese / English

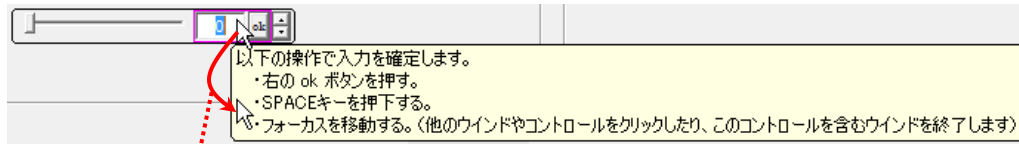
20. タテック：ツールチップ (CAjrStatic.dll, SAjrTip クラス)

ツールチップに関するスタティックメソッド群です。クラス名は「SAjrTip」です。

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 標準のツールチップと同等の機能に、以下の機能を追加しています。
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。

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。



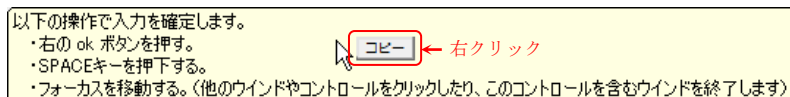
マウスカーソルを素早くツールチップ上に移動するとツールチップの表示を継続します。
その後、マウスカーソルをツールチップ外に移動すると、ツールチップは消えます。

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

ツールチップにイメージ (ビットマップやアイコン) を描画するには、RegistBitmap()/RegistIcon() 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.

```
private Icon    m_ico = Properties.Resources.logo;
private Bitmap m_bmp = Properties.Resources.cat;
...
SAjrTip.RegistIcon (m_ico, 0, 0, 0, 0, 0);           // アイコン登録 (id = 0)
SAjrTip.RegistBitmap(m_bmp, 1, 0, 0, 0, 0);         // ビットマップ登録(id = 1)
...
SAjrTip.Add(ctrl, "イメージ表示ツールチップのサンプル" + //
    "\x1B[0;-20;2;0;0z" +                               // (右端-20, 2) の位置にアイコン表示
    "\x1B[1;30;20;0;0z";                                   // (30, 20)の位置にビットマップ表示
...

```

20.1. 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[0m"	文字色、文字背景色とフォント（太字、斜字、下線）をリセット
2	"¥x1B[1m"	ボールド（太字）設定 (※4)
3	"¥x1B[3m"	イタリック（斜字）設定 (※4)
4	"¥x1B[4m"	アンダーライン（下線）設定 (※4)
5	"¥x1B[9m"	文字色と文字背景色をリセット（パレット0、透明）
6	"¥x1B[30m" ~ "¥x1B[37m"	文字色の設定 (3XのXはパレット番号 ※1)
7	"¥x1B[38;2;r;g;b" (※2)	文字色をRGBで指定
8	"¥x1B[39m"	文字色リセット（パレット0（デフォルトは黒））
9	"¥x1B[40m" ~ "¥x1B[47m"	文字背景色の設定 (4XのXはパレット番号 ※1)
10	"¥x1B[48;2;r;g;b" (※2)	文字背景色をRGBで指定
11	"¥x1B[49m"	文字背景色リセット
12	"¥x1B[I"	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[nL"	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;cyz"	イメージ描画 (id：イメージ登録ID, x,y：表示サイズ, cx,cy：描画サイズ (正数) x yは、描画域の位置を指定します。(文字列描画域ではない) x/yが負数時は右端/下端からの減算値, cx/cyがゼロ時は原画サイズ, x~cyは省略可)

※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：各色の成分値は0~255（各10進数で、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。

20.2. メソッド

チップテキストのメソッド一覧を示します。

#	メソッド名	内容	備考
1	{Set/Get}DefTextColor	デフォルトテキスト表示色 設定／取得	
2	{Set/Get}DefBorderColor	デフォルト外枠表示色 設定／取得	
3	{Set/Get}DefBackGround	デフォルトウインド背景表示色 設定／取得	
4	{Set/Get}DefFont	デフォルトフォント設定／取得	
5	{Set/Get}DefDelayTime	デフォルト表示遅延時間 設定／取得	
6	{Set/Get}DefDisplayTime	デフォルト表示時間 設定／取得	
7	{Set/Get}WindowTime	コントロール間移動猶予時間 設定／取得	
8	{Set/Get}Palette	パレット色の設定／取得	
9	{Set/Get}ShowForOnlyActive	アクティブ時のみツールチップ表示の設定／取得	
10	{Set/Get}EnableAll	全てのチップテキストの表示許可／禁止設定／取得	
11	Add	ツールチップの関連付け追加	
12	{Set/Get}ShowAlways	ツールチップの表示条件設定	
13	Remove	チップテキストの関連付け解除	
14	SetCallBack	コールバック設定	
15	GetSize	ツールチップ・ウインドサイズ取得	
16	Show	ツールチップの表示	マウスカーソルに関係なく単独表示
17	Hide	ツールチップを非表示	
18	ShowCenter	コントロールの中央にツールチップ表示	マウスカーソルに関係なく単独表示
19	MoveCursor	マウスカーソルをツールチップ上へ移動	
20	SetPalette	パレット色の設定	
21	RegistBitmap	ビットマップ登録	
22	RegistIcon	アイコン登録	
23	UnregistImage	イメージ登録解除	

20.2.1. デフォルト・テキスト表示色 設定／取得({Set|Get}DefTextColor)

形 式 : static void SetDefTextColor(Color color); --- 設定
static Color GetDefTextColor(); ----- 取得

引 数 : color - デフォルトテキスト表示色

説 明 : デフォルトのテキスト表示色を設定／取得します。

戻り値 : デフォルト・テキスト表示色 (取得時)

20.2.2. デフォルト外枠表示色 設定／取得({Set|Get}DefBorderColor)

形 式 : static void SetDefBorderColor(Color color) --- 設定
static Color GetDefBorderColor(); ----- 取得

引 数 : color - デフォルト外枠表示色

説 明 : デフォルトの外枠表示色を設定／取得します。

戻り値 : デフォルト外枠表示色 (取得時)

20.2.3. デフォルトウインド背景表示色 設定／取得({Set|Get}DefBackGround)

形 式 : static void SetDefBackGround(Color color); --- 設定
static Color GetDefBackGround(); ----- 取得

引 数 : color - デフォルト外枠表示色

説 明 : デフォルトのウインド背景表示色を設定／取得します。

戻り値 : デフォルト・ウインド背景表示色 (取得時)

20.2.4. デフォルト・フォント 設定／取得({Set|Get}DefFont)

形 式 : static void SetDefFont(Font font); --- 設定
static Font GetDefFont(); ----- 取得

引 数 : color - デフォルト外枠表示色

説 明 : デフォルトのフォントを設定／取得します。

戻り値 : デフォルト・フォント (取得時)

20.2.5. デフォルト・表示遅延時間 設定／取得({Set|Get}DefDelayTime)

形 式 : static void SetDefDelayTime(int msTime); --- 設定
static int GetDefDelayTime(); ----- 取得

引 数 : msTime - デフォルト表示遅延時間[ms] (規定値は、1000[ms])

説 明 : デフォルトの表示遅延時間を設定／取得します。
表示遅延時間とは、マウスポインタがコントロール上で停止してからツールチップが表示されるまでの時間を意味します。

戻り値 : デフォルト表示遅延時間 (取得時)

20.2.6. デフォルト・表示時間 設定／取得({Set|Get}DefDisplayTime)

形 式 : static void SetDefDisplayTime(int msTime); --- 設定
static int GetDefDisplayTime(); ----- 取得

引 数 : msTime - デフォルト表示遅延時間[ms] (規定値は、5000[ms])

説 明 : デフォルトの表示時間を設定／取得します。
表示時間とは、ツールチップが表示されてから、自動的に消えるまでの時間を意味します。

戻り値 : デフォルト表示時間 (取得時)

20.2.7. コントロール間移動猶予時間 設定／取得({Set|Get}WindowTime)

形 式 : static void SetWindowTime(int msTime); --- 設定
static int GetWindowTime(); ----- 取得

引 数 : msTime - コントロール間移動猶予時間[ms] (規定値は、500[ms])

説 明 : デフォルトのコントロール間移動猶予時間を設定／取得します。
カーソルがコントロールを離れても、すぐにツールチップは消えずに「コントロール間移動猶予時間」だけ表示され続けます。
そして、ツールチップが消えないうちに、マウスカーソルが別のコントロールへ移動した場合は表示遅延時間を待たずに該当ツールチップが表示されます。つまり、カーソルを素早く移動すれば、次のツールチップが、すぐに表示されます。
また、「コントロール間移動猶予時間」以内に、ツールチップ上にカーソルを移動すれば、カーソルがツールチップを離れるまで表示し続けます。

戻り値 : コントロール間移動猶予時間 (取得時)

20.2.8. パレット色の設定／取得({Set|Get}Palette)

形 式 : static void SetPalette(int ix, Color color); --- 設定
 static Color GetPalette(); ----- 取得

引 数 : ix - パレット番号 (0～7)
 color - パレットに設定する色

説 明 : ix で指定したパレット番号に色を設定／取得します。

戻り値 : パレットの色 (取得時)

20.2.9. 全コントロールでの、非アクティブ時ツールチップ表示設定／取得({Set|Get}ShowForOnlyActive)

形 式 : static void SetShowForOnlyActive (bool fShowForOnlyActive); --- 設定
 static bool GetShowForOnlyActive(); ----- 取得

引 数 : fShowForOnlyActive - 全コントロールでの非アクティブ時の表示設定
 true : 全てのコントロールで、アクティブな状態時のみツールチップを表示する
 false : ShowAlways() メソッドの指定に従う

説 明 : 全てのコントロールにおいて、アクティブな状態時のみツールチップを表示するか否かを設定／取得します。
 fShowForOnlyActive=true の場合は、全てのコントロールにおいて (ShowAlways() メソッドの設定に関わらず) アクティブな状態時のみツールチップを表示します。
 fShowForOnlyActive=false の場合は、ShowAlways() メソッドでの設定に従います。

戻り値 : 全コントロールでの非アクティブ時の表示設定 (取得時)

20.2.10. 全てのチップテキストの表示許可／禁止 設定／取得({Set|Get}EnableAll)

形 式 : static void SetEnableAll (bool fEnable); ----- 設定
 static bool GetEnableAll(); ----- 取得

引 数 : fEnable - 表示許可(true)／表示禁止(false)

説 明 : 全てのコントロールにおいて、ツールチップを表示するか否かを設定／取得します。
 fEnable=false を設定すると、ツールチップを表示しません。
 fEnable=true を設定すると、再び、ツールチップを表示するようになります。

戻り値 : 全コントロールでの表示許可状態 (取得時)

20.2.11. ツールチップの関連付け追加 (Add)

形 式 : static void Add(object ctrl, string text);
 — 時間指定 —
 static void Add (object ctrl, string text, int msDelay, int msShow);
 — フォント指定 —
 static void Add (object ctrl, string text, Font font);
 — 色指定 —
 static void Add (object ctrl, string text, Color TextColor, Color BackGround, Color BorderColor);
 static void Add (object ctrl, string text, Color TextColor, Color BackGround, bool fBorder);
 — 時間, フォント指定 —
 static void Add (object ctrl, string text, int msDelay, int msShow, Font font);
 — 時間, 色指定 —
 static void Add (object ctrl, string text, int msDelay, int msShow, Color TextColor, Color BackGround, Color BorderColor);
 static void Add (object ctrl, string text, int msDelay, int msShow, Color TextColor, Color BackGround, bool fBorder);
 — フォント, 色指定 —
 static void Add (object ctrl, string text, Font font, Color TextColor, Color BackGround, Color BorderColor);
 static void Add (object ctrl, string text, Font font, Color TextColor, Color BackGround, bool fBorder);
 — 全て指定 —
 static void Add (object ctrl, string text, int msDelay, int msShow, Font font,
 Color TextColor, Color BackGround, Color BorderColor);
 static void Add (object ctrl, string text, int msDelay, int msShow, Font font,
 Color TextColor, Color BackGround, bool fBorder);

引 数 : ctrl - ツールチップを関連付けするコントロール
 text - ツールチップテキスト ("MyWndText" を指定した場合は、自コントロールのテキストがチップテキストとなります)
 msDelay - 表示遅延時間[ms] (-1:デフォルトの表示遅延時間)
 msShow - 表示時間[ms] (-1:デフォルトの表示時間)
 font - テキストフォント (null: デフォルトフォント)
 TextColor - テキストの表示色 (Color.Empty : デフォルトテキスト表示色)
 BackGround - ウィンド背景色 (Color.Empty : デフォルトのウィンド背景色)
 BorderColor - 外枠の色 (Color.Empty : デフォルトの外枠表示色)
 fBorder - 外枠表示フラグ (true : 外枠表示 (デフォルトの外枠表示色), false:外枠非表示)

説 明 : コントロールにツールチップを関連付けします。

戻り値 : なし

20.2.12. ツールチップの表示条件設定／取得({Set/Get}ShowAlways)

形 式 : static void SetShowAlways(object ctrl, bool fShowAlways); --- 設定
 static bool GetShowAlways(object ctrl); ----- 取得

引 数 : ctrl - ツールチップが関連付けられているコントロール
 fShowAlways - 非アクティブな状態時のツールチップ表示フラグ (true:表示する, false:表示しない)

説 明 : コントロールが非アクティブな状態時における、ツールチップの表示／非表示を設定します。
 デフォルトでは、「fShowAlways=true」となっていますので、非アクティブな状態時にツールチップを表示しない場合に fShowAlways=false を設定してください。
 SetShowForOnlyActive() で、fShowForOnlyActive=false (全てのコントロールで、アクティブな状態時のみツールチップを表示する) に設定されている場合は、本メソッドの設定に関わらず、非アクティブな状態時はツールチップを表示しません。

戻り値 : 非アクティブな状態時のツールチップ表示フラグ (取得時)

20.2.13. ツールチップの関連付け解除(Remove)

形 式 : static void Remove(object ctrl); --- 特定のツールチップ関連付け解除
 static void Remove(); ----- 全てのツールチップ関連付け解除

引 数 : ctrl - ツールチップが関連付けられているコントロール

説 明 : ツールチップの関連付けを解除し、コントロール上にマウスカーソルを置いてもツールチップを表示しないようにします。

戻り値 : なし

20.2.14. コールバック設定(SetCallBack)

形 式 : `static void SetCallBack(object ctrl, IntPtr cbp, TipCbKNeedText cbNeedText);`
`static void SetCallBack(object ctrl, IntPtr cbp, TipCbKChkPoint cbChkPoint);`
`static void SetCallBack(object ctrl, IntPtr cbp, TipCbKNeedText cbNeedText, TipCbKChkPoint cbChkPoint);`

引 数 : `ctrl` - ツールチップが関連付けられているコントロール
`cbNeedText` - ツールチップテキスト取得用コールバック
`cbChkPoint` - カーソル位置チェック用コールバック

説 明 : 以下の事象に関するユーザコールバック・メソッドを設定します。

- ・ ツールチップテキストの要求 (cbNeedText)
- ・ マウスカーソル位置のチェック (cbChkPoint)

コントロール上にカーソルを置いてツールチップを表示する際には、cbNeedText で指定したメソッドが呼び出されます。このコールバックメソッドは、表示するべきツールチップテキストを返します。cbNeedText でチップテキスト取得用コールバックが設定されていない場合は、Add() メソッドで設定したツールチップテキストが表示されます。

マウスカーソルがツールチップを表示するべき位置にあるかをチェックする必要がある場合は、cbChkPoint でカーソル位置チェック用のコールバック・メソッドを指定します。cbChkPoint でカーソル位置チェック用コールバック・メソッドが設定されていない場合は、当該コントロール全体がツールチップの表示対象となります。

戻り値 : なし

コールバック : コールバックメソッドの形式は、以下の通りです。

cbNeedText (チップテキスト取得用コールバック)

形 式 : `string cbNeedText(IntPtr Handle, IntPtr cbp);`

引 数 : `Handle` - 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 : 表示すべきツールチップテキストを返します。
空文字列("")を返した場合は、ツールチップを表示しません。

戻り値 : 表示するツールチップテキスト

cbChkPoint (カーソル位置チェック用コールバック)

形 式 : `bool cbChkPoint(IntPtr Handle, ref AJCPOINT ptClient, IntPtr cbp);`

引 数 : `Handle` - コントロールのウインドハンドル
`ptClient` - マウスカーソル位置 (コントロールの左上を (0, 0) としたクライアント座標)
`cbp` - コールバックパラメタ

説 明 : マウスカーソル位置をチェックします。
マウスカーソルが自コントロール中のツールチップ表示対象域である場合は true を返します。

戻り値 : `true` : ツールチップを表示する
`false` : ツールチップを表示しない

20.2.15. ツールチップ のウインドサイズ取得(GetSize)

```
形 式 : static Size GetSize(string text, Font font, bool fBorder); ----- ①
        static void GetSize(string text, Font font, bool fBorder, out Size size); -- ②
```

引 数 : font - ツールチップのフォント (null:デフォルトフォント)
fBorder - 外枠の有無 (true:外枠あり, false:外枠なし)
size - ツールチップのウインドのサイズ (ピクセル数) を設定する変数

説明 : 指定のツールチップを表示する際の、ウインドのサイズを取得します。

戻り値 : ① - チップテキストウインドのサイズ (ピクセル数)
② - なし

20.2.16. ツールチップの表示 (Show)

```

形式 : static void Show(int x, int y, string text);
      — 時間指定 —————
static void Show(int x, int y, string text, int msTime);
      — フォント指定 —————
static void Show(int x, int y, string text, Font font);
      — 色指定 —————
static void Show(int x, int y, string text, Color TextColor, Color BackGround, Color BorderColor);
static void Show(int x, int y, string text, Color TextColor, Color BackGround, int BorderColor);
      — 時間, フォント指定 —
static void Show(int x, int y, string text, int msTime, Font font);
      — 時間, 色指定 —————
static void Show(int x, int y, string text, int msTime, Color TextColor, Color BackGround, Color BorderColor);
static void Show(int x, int y, string text, int msTime, Color TextColor, Color BackGround, bool fBorder);
      — フォント, 色指定 —
static void Show(int x, int y, string text, Font font, Color TextColor, Color BackGround, Color BorderColor);
static void Show(int x, int y, string text, Font font, Color TextColor, Color BackGround, bool fBorder);
      --- 全指定 -----
static void Show(int x, int y, int cx, int cy, string text, int msTime, Font font,
                  Color TextColor, Color BackGround, Color BorderColor);
static void Show(int x, int y, int cx, int cy, string text, int msTime, Font font,
                  Color TextColor, Color BackGround, bool fBorder);

```

引 数	:	x, y	- ツールチップの表示位置
		cx	- ツールチップウインドの幅 (ピクセル数, 0 指定時は自動調整)
		cy	- ツールチップウインドの高さ (ピクセル数, 0 指定時は自動設定)
		text	- ツールチップテキスト
		msTime	- 表示時間[ms]
		font	- テキストフォント (null: デフォルトフォント)
		TextColor	- テキストの表示色 (Color.Empty : デフォルトテキスト表示色)
		BackGround	- ウインド背景色 (Color.Empty : デフォルトのウインド背景色)
		BorderColor	- 外枠の色 (Color.Empty : デフォルトの外枠表示色)
		fBorder	- 外枠表示フラグ (true : 外枠表示 (デフォルトの外枠表示色), false:外枠非表示)

説明 : マウскарソルに関係なく、単独で指定位置にツールチップを表示します。
ツールチップを非表示とするには、TipTextHide()を実行します。

戻り値 : なし

20.2.17. ツールチップの非表示 (Hide)

形 式 : static void Hide();

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

20.2.18. コントロールの中央にツールチップ表示(ShowCenter)

形 式 : static void ShowCenter(Control ctl, string text);
 ----- 時間指定 -----
 static void ShowCenter(Control ctl, string text, int msTime);
 ----- フォント指定 -----
 static void ShowCenter(Control ctl, string text, Font font);
 ----- 色指定 -----
 static void ShowCenter(Control ctl, string text, Color TextColor, Color BackGround, Color BorderColor);
 static void ShowCenter (Control ctl, string text, Color TextColor, Color BackGround, bool fBorder);
 ----- 時間, フォント指定 -----
 static void ShowCenter (Control ctl, string text, int msTime, Font font);
 ----- 時間, 色指定 -----
 static void ShowCenter (Control ctl, string text, int msTime, Color TextColor, Color BackGround, Color BorderColor);
 static void ShowCenter (Control ctl, string text, int msTime, Color TextColor, Color BackGround, bool fBorder);
 ----- フォント, 色指定 -----
 static void ShowCenter (Control ctl, string text, Font font, Color TextColor, Color BackGround, Color BorderColor);
 static void ShowCenter (Control ctl, string text, Font font, Color TextColor, Color BackGround, bool fBorder);
 ----- 全指定 -----
 static void ShowCenter (Control ctl, int cx, int cy, string text, int msTime, Font font,
 Color TextColor, Color BackGround, Color BorderColor);
 static void ShowCenter (Control ctl, int cx, int cy, string text, int msTime, Font font,
 Color TextColor, Color BackGround, bool fBorder);

引 数 : ctl - ツールチップを表示するコントロール (このコントロールの中央にツールチップを表示)
 cx - ツールチップウインドの最小幅 (ピクセル数, 0 指定時はツールチップテキストに合わせて自動調整)
 cy - ツールチップウインドの高さ (ピクセル数, 0 指定時は文字の高さや行数に合わせて自動設定)
 text - ツールチップテキスト
 msTime - 表示時間[ms]
 font - テキストフォント (null: デフォルトフォント)
 TextColor - テキストの表示色 (Color.Empty : デフォルトテキスト表示色)
 BackGround - ウインド背景色 (Color.Empty : デフォルトのウインド背景色)
 BorderColor - 外枠の色 (Color.Empty : デフォルトの外枠表示色)
 fBorder - 外枠表示フラグ (true : 外枠表示 (デフォルトの外枠表示色), false:外枠非表示)

説 明 : マウ斯卡ーソルに関係なく、単独で指定コントロールの中央にツールチップを表示します。
 ツールチップを非表示とするには、TipTextHide() を実行します。

戻り値 : なし

20.2.19. マウ斯卡ーソルをツールチップ上へ移動(MoveCursor)

形 式 : static void MoveCursor(); ----- 中央へ移動
 static void MoveCursor(ETipCurMove mvc); -- 指定位置へ移動

引 数 : mvc - マウ斯卡ーソル移動先 (CENTER:中央, LU : 左上, RU: 右上, LD: 左下, RD : 右下)

説 明 : マウ斯卡ーソルをツールチップ上へ移動します。
 マウ斯卡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。

戻り値 : なし

20.2.20. パレット色の設定 (SetPalette)

形 式 : static void SetPalette (int ix, Color color);

引 数 : ix - パレット番号 (0 ~ 7)
 color - パレットに設定する色

説 明 : パレットの色を設定します。(エスケープシーケンス (“%x1B[3Xm” や “%x1B[4Xm” で指定する表示色の設定)

戻り値 : なし

20.2.21. ビットマップ登録 (RegistBitmap)

形 式 : static void RegistBitmap (Bitmap bmp, uint id, int x, int y, int cx, int cy);

引 数 : bmp - ビットマップ
 id - 識別 I D (0 ~)
 x, y - デフォルトの描画位置
 cx, cy - デフォルトの描画サイズ

説 明 : チップテキストで表示するビットマップを登録します。
 ビットマップを表示するには文字列に以下の (末尾が 'z' の) エスケープシーケンスを含めます。

"¥x1B[id;x;y;cx;cyz" id- 識別 I D x,y - 描画位置 (省略可) cx, cy - 描画サイズ (省略可)

戻り値 : なし

20.2.22. アイコン登録 (RegistIcon)

形 式 : static void RegistIcon (Icon ico, uint id, int x, int y, int cx, int cy);

引 数 : ico - アイコン
 id - 識別 I D (0 ~)
 x, y - デフォルトの描画位置
 cx, cy - デフォルトの描画サイズ

説 明 : チップテキストで表示するアイコンを登録します。
 アイコンを表示するには文字列に以下の (末尾が 'z' の) エスケープシーケンスを含めます。

"¥x1B[id;x;y;cx;cyz" id- 識別 I D x,y - 描画位置 (省略可) cx, cy - 描画サイズ (省略可)

戻り値 : なし

20.2.23. イメージ登録解除 (UnregistImage)

形 式 : static void UnregistImage (int id);

引 数 : id - 識別 I D (0 ~)

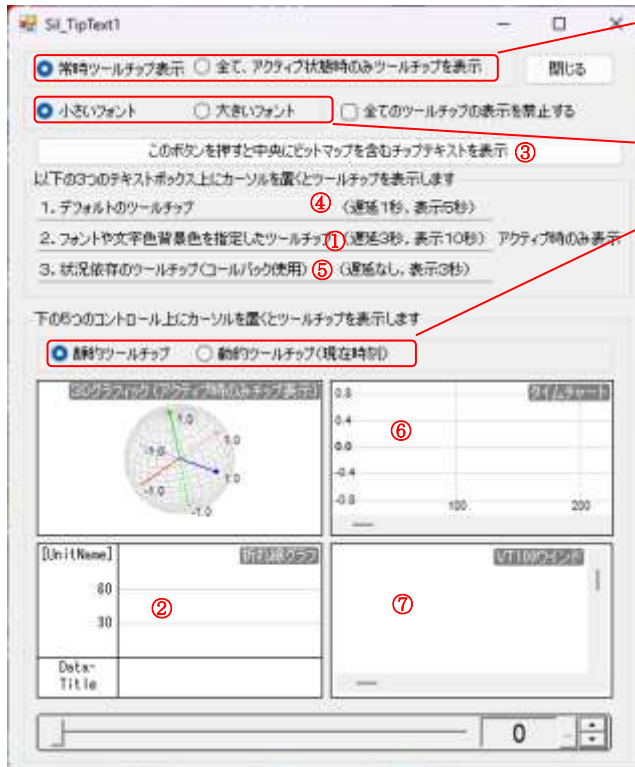
説 明 : ビットマップやアイコンの登録を解除します。

戻り値 : なし

20.3. サンプルプログラム

20.3.1. Sil_TipText1

このサンプルプログラムでは、単に、さまざまなツールチップを表示します。
いずれかの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を表示します。



ツールチップを常時表示するか、自アプリがアクティブな時のみ表示するかを選択します。

但し、「2. フォントや・・・」と「3 Dグラフィック」だけは、常に、自アプリがアクティブな時のみ表示するようにプログラムしています。

ツールチップで表示するテキストのフォントを選択します。

ツールチップの表示種別を選択します。

「静的ツールチップ」は、あらかじめ設定してあるツールチップを表示します。

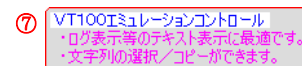
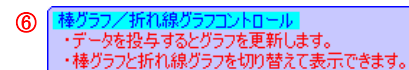
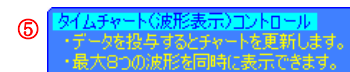
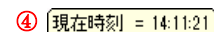
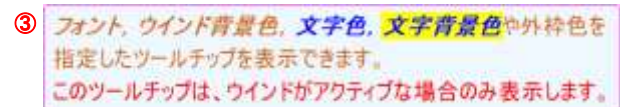
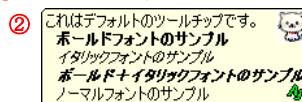
「動的ツールチップ」を選択すると、(5つのコントロールで) ツールチップとして現在時刻を表示します。

SAjrTip.Add() によるチップテキスト設定を行います。

①をクリックした場合のツールチップ



②～⑦にカーソルを置いた場合のツールチップ



```

1 : using System;
2 : using System.Collections.Generic;
3 : using System.ComponentModel;
4 : using System.Data;
5 : using System.Drawing;
6 : using System.Linq;
7 : using System.Text;
8 : using System.Windows.Forms;
9 : using System.Runtime.InteropServices;
10 : using CAjrCustCtrl;
11 :
12 : namespace Sil_TipText1
13 : {
14 :     public partial class Form1 : Form
15 :     {
16 :         private TipCbKNeedText m_cbNeedText;
17 :         private Font m_SFont = new Font("MS UI Gothic", 9);
18 :         private Font m_LFont = new Font("MS UI Gothic", 12);
19 :         private Bitmap m_skr = Properties.Resources.sakura;
20 :         private Bitmap m_bmp = Properties.Resources.cat;
21 :         private Icon m_ico = Properties.Resources.logo;
22 :
23 :         public Form1()
24 :         {
25 :             InitializeComponent();
26 :         }
27 :         //----- 閉じるボタン押下 -----//
28 :         private void btnClose_Click(object sender, EventArgs e)
29 :         {
30 :             Close();
31 :         }
32 :         //----- 起動時初期設定 -----//
33 :         private void Form1_Load(object sender, EventArgs e)
34 :         {
35 :             //----- チップテキスト用イメージ登録 -----//
36 :             SAjrTip.RegistBitmap(m_skr, 0, 0, 0, 0, 0);
37 :             SAjrTip.RegistBitmap(m_bmp, 1, 0, 0, 0, 0);
38 :             SAjrTip.RegistIcon(m_ico, 2, 0, 0, 0, 0);
39 :
40 :             //--- 常時／アクティブ時選択ラジオボタンのツールチップ -----//
41 :             SAjrTip.Add(rbtShowAlways, "常時ツールチップ表示します。¥n" +
42 :                 "但し、以下のコントロールは、アクティブ時のみツールチップを表示します。¥n" +
43 :                 "・「2. フォントや文字色背景色を指定したツールチップ」¥n" +
44 :                 "・「3Dグラフィック」");
45 :             SAjrTip.Add(rbtShowOnActive, "全てのコントロールで、アクティブ状態時のみツールチップを表示します。¥n" +
46 :                 "非アクティブ時はツールチップを表示しません。");
47 :             //--- 小さいフォント ラジオボタンのツールチップ -----//
48 :             SAjrTip.Add(rbtSFont, "小さい文字でツールチップを表示します。¥n" +
49 :                 "但し、「2. フォントや文字色・」は独自のフォントで表示します。");
50 :             //--- 大きいフォント ラジオボタンのツールチップ -----//
51 :             SAjrTip.Add(rbtLFont, "大きい文字でツールチップを表示します。¥n" +
52 :                 "但し、「2. フォントや文字色・」は独自のフォントで表示します。");
53 :             //--- テキストボックス1のツールチップ(ボールド, イタリックの例) ---//
54 :             SAjrTip.Add(textBox1, "これはデフォルトのツールチップです。¥n" +
55 :                 " ¥x1B[T ボールドフォントのサンプル¥x1B[t¥n" +
56 :                 " ¥x1B[I イタリックフォントのサンプル¥x1B[i¥n" +
57 :                 " ¥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" +
58 :                 " ノーマルフォントのサンプル" +
59 :                 " ¥x1B[1;-26;2;24;24z" + // (右端-26, 2) の位置に 24×24 サイズでビットマップ表示
60 :                 " ¥x1B[2;-18;-18z"); // (右端-18, 下端-18)の位置に原画サイズでアイコン表示
61 :             //--- テキストボックス2のツールチップ -----//
62 :             SAjrTip.Add(textBox2,
63 :                 "フォント, ウィンド背景色, ¥x1B[34m 文字色, ¥x1B[43m 文字背景色¥x1B[0m や外枠色を¥n 指定したツールチップを表示できます。¥n" +
64 :                 "¥x1B[31m このツールチップは、ウィンドがアクティブな場合のみ表示します。",
65 :                 3000, 10000, new Font("HGP 楷書体", 18, FontStyle.Bold | FontStyle.Italic), Color.Chocolate, Color.AliceBlue, Color.Fuchsia);
66 :             SAjrTip.SetShowAlways(textBox2, false); // アクティブ時のみ表示
67 :             //--- テキストボックス3のツールチップ (コールバック設定) -----//
68 :             m_cbNeedText = new TipCbKNeedText(cbNeedText);
69 :             SAjrTip.Add(textBox3, "Dummy", 0, 3000);
70 :             SAjrTip.SetCallBack(textBox3, (IntPtr)0, m_cbNeedText);
71 :             //--- 閉じるボタンのツールチップ設定 -----//

```

```

72 :     SAjrTip.Add(btnClose, "ウインドを閉じてプログラムを終了します。");
73 :     //--- 静的／動的選択ラジオボタンのツールチップ -----//
74 :     SAjrTip.Add(rbtStatic, "あらかじめ設定されているツールチップを表示します。");
75 :     SAjrTip.Add(rbtDynamic, "コールバックにより現在時刻をツールチップ表示します。");
76 :     //---- 全ての設定値ロード (ラジオボタンは Checked() イベントを発生する)
77 :     SAjrReg.LoadAllCtrls(this, ELoadAllOpt.RbtClickEvent);
78 : }
79 : //---- フォームクローズ -----//
80 : private void Form1_FormClosed(object sender, FormClosedEventArgs e)
81 : {
82 :     //---- 全ての設定値セーブ -----//
83 :     SAjrReg.SaveAllCtrls(this, ESaveAllOpt.ExcReadOnlyText);
84 : }
85 : //---- 常時ツールチップ表示・ラジオボタン -----//
86 : private void rbtShowAlways_Click(object sender, EventArgs e)
87 : {
88 :     sta.ShowForOnlyActive = false;
89 : }
90 : //---- 全て、アクティブ状態時のみツールチップを表示・ラジオボタン -----//
91 : private void rbtShowOnActive_Click(object sender, EventArgs e)
92 : {
93 :     sta.ShowForOnlyActive = true;
94 : }
95 : // コールバック (ツールチップテキスト要求)
96 : private string cbNeedText(IntPtr Handle, IntPtr cbp)
97 : {
98 :     DateTime dt = DateTime.Now;
99 :     string text = "現在時刻 = " + dt.ToString("HH:mm:ss");
100 :     return text;
101 : }
102 : //---- 静的ツールチップ・ラジオボタン -----//
103 : private void rbtStatic_Click(object sender, EventArgs e)
104 : {
105 :     //--- 3Dグラフィックのツールチップ -----//
106 :     SAjrTip.Add(g3d, "¥x1B[64;64;255F" + "¥x1B[255;200;255B" +
107 :         "¥x1B[37;45m 3D／2Dグラフィックコントロール ¥n¥x1B[0m" +
108 :         "・データを投与すると当該座表にプロット表示します。¥n" +
109 :         "・3D空間上に図形を描画できます。¥n" +
110 :         "・ウインドをドラッグで視点を変更できます。¥n" +
111 :         "¥x1B[31m このツールチップは、ウインドがアクティブな場合のみ表示します。");
112 :     SAjrTip.SetShowAlways(g3d, false); // アクティブ時のみ表示
113 :     //--- 棒グラフのツールチップ -----//
114 :     SAjrTip.Add(bar, "¥x1B[0;0;255F" + "¥x1B[200;200;255B" +
115 :         "¥x1B[34;46m 棒グラフ／折れ線グラフコントロール ¥n¥x1B[0m" +
116 :         "¥x1B[31m" +
117 :         "・データを投与するとグラフを更新します。¥n" +
118 :         "・棒グラフと折れ線グラフを切り替えて表示できます。");
119 :     //--- タイムチャートのツールチップ -----//
120 :     SAjrTip.Add(tch, "¥x1B[0;0;255F" + "¥x1B[64;128;255B" +
121 :         "¥x1B[34;46m タイムチャート (波形表示) コントロール ¥n¥x1B[0m" +
122 :         "¥x1B[33m" +
123 :         "・データを投与するとチャートを更新します。¥n" +
124 :         "・最大8つの波形を同時に表示できます。");
125 :     //--- VT100ウインドのツールチップ -----//
126 :     SAjrTip.Add(vth, "¥x1B[255;0;0F" + "¥x1B[220;220;220B" +
127 :         "¥x1B[34;47m VT100エミュレーションコントロール ¥n¥x1B[0m" +
128 :         "¥x1B[35m" +
129 :         "・ログ表示等のテキスト表示に最適です。¥n" +
130 :         "・文字列の選択／コピーができます。");
131 :     //--- 数値入力コントロールのツールチップ -----//
132 :     SAjrTip.Add(inp, "数値入力コントロール ¥n" +
133 :         "・整数モードと実数モードがあります。¥n" +
134 :         "・直接入力／スライダ／スピンボタン操作ができます。");
135 :     // コールバック解除
136 :     SAjrTip.SetCallBack(g3d, (IntPtr)0, null, null);
137 :     SAjrTip.SetCallBack(bar, (IntPtr)0, null, null);
138 :     SAjrTip.SetCallBack(tch, (IntPtr)0, null, null);
139 :     SAjrTip.SetCallBack(vth, (IntPtr)0, null, null);
140 :     SAjrTip.SetCallBack(inp, (IntPtr)0, null, null);
141 : }
142 : //---- 動的ツールチップ・ラジオボタン -----//
143 : private void rbtDynamic_Click(object sender, EventArgs e)

```

```

144 :     {
145 :         // コールバック設定
146 :         SAjrTip.Add(g3d, null); SAjrTip.SetCallBack(g3d, (IntPtr)0, m_cbNeedText);
147 :         SAjrTip.Add(bar, null); SAjrTip.SetCallBack(bar, (IntPtr)0, m_cbNeedText);
148 :         SAjrTip.Add(tch, null); SAjrTip.SetCallBack(tch, (IntPtr)0, m_cbNeedText);
149 :         SAjrTip.Add(vth, null); SAjrTip.SetCallBack(vth, (IntPtr)0, m_cbNeedText);
150 :         SAjrTip.Add(inp, null); SAjrTip.SetCallBack(inp, (IntPtr)0, m_cbNeedText);
151 :     }
152 :     //----- 動的ツールチップ取得用コールバック -----//
153 :     private string cbGetText(IntPtr Handle, IntPtr cbp)
154 :     {
155 :         return "¥x1B[46m 現在時刻 : ¥x1B[0m " + DateTime.Now;
156 :     }
157 :     //----- 小さいフォントラジオボタン -----//
158 :     private void rbtSFont_Click(object sender, EventArgs e)
159 :     {
160 :         SAjrTip.SetDefFont(m_SFont);
161 :     }
162 :     //----- 大きいフォントラジオボタン -----//
163 :     private void rbtLFont_Click(object sender, EventArgs e)
164 :     {
165 :         SAjrTip.SetDefFont(m_LFont);
166 :     }
167 :     //----- 全てのツールチップの表示を禁止する -----//
168 :     private void chkEnableAll_Click(object sender, EventArgs e)
169 :     {
170 :         SAjrTip.SetEnableAll(!chkEnableAll.Checked);
171 :     }
172 :     //----- チップテキスト表示ボタン -----//
173 :     private void btnShowBmp_Click(object sender, EventArgs e)
174 :     {
175 :         string txt = "ビットマップ表示ツールチップのサンプル" +
176 :             "¥x1B[0;7;20;0;0z";
177 :         Point pt = btnShowBmp.PointToScreen(Point.Empty);
178 :         Size sz = btnShowBmp.Size;
179 :         Size ts = SAjrTip.GetSize(txt, null, true);
180 :
181 :         SAjrTip.Show(pt.X + sz.Width - ts.Width, pt.Y + 35, 0, 0,
182 :             txt, 10000, null, Color.Empty, Color.Empty, true);
183 :     }
184 : }
185 : }

```

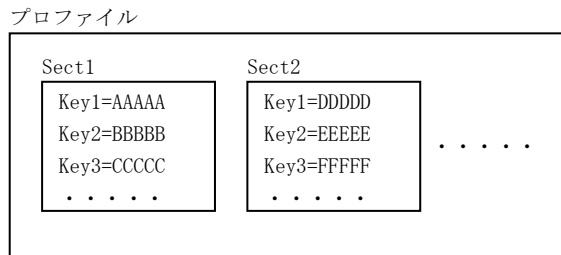
21. スタティック：プロファイル/レジストリアクセス (CAjrStatic.dll, SAjrReg クラス)

プロファイル、レジストリへのアクセススタティック関数群です。クラス名は「SAjrReg」です。

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。（キー名称は、個々の値の名称となります）



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 ファイルかレジストリかを意識することなく、セクション名とキー名称で一元的にアクセスできます。

.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、ToRegistry プロパティの設定によります。
（デフォルトでは、レジストリ(true)となっています）

.ini ファイルへアクセスする場合は、ToRegistry プロパティを false に設定します。

.ini ファイルのパス名は、デフォルトでは自プログラムファイル名の拡張子を「.ini」に変更した名称となっていますが、IniFilePath プロパティにより変更できます。

レジストリへアクセスする場合は、ToRegistry プロパティを true に設定します。

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、「HKEY_CURRENT_USER」となっています。

<ルートパス名>は、デフォルトで「Software¥AjrCstXX」となっていますが、RegRootPath メソッド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、RegMiddlePath メソッドにより変更できます。

<セクション名>は、各プロファイル・アクセスメソッドの「sec」引数で指定します。

プロファイル・アクセスコントロールは、本ライブラリ内で共通のオブジェクトとなっております。

プロファイルアクセス先や、プロファイルのパス名の設定はライブラリやプログラム全体に影響します。

以下の、プロパティ/メソッドによる設定は、ライブラリ全体に影響します。

• ToRegistry プロパティ	（プロファイルの記録先	）
• Volatile プロパティ	（レジストリに記録する際のモード	）
• RegTopKey プロパティ	（レジストリ・トップキー設定	）
• RegRootPath プロパティ	（レジストリ・ルートパス設定	）
• RegMiddlePath プロパティ	（レジストリ・ミドルパス設定	）
• IniFilePath プロパティ	（INI ファイルパス設定	）

例えば、タイムチャートグラフ表示コントロールの SaveToProfile() メソッドは、プロファイルへ設定値を記録しますが、本コントロールで「ToRegistry=false」に設定すると、SaveToProfile()メソッドは 設定内容を INI ファイルへ記録します。

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)

プロファイル先をレジストリとする場合のアクセス手順

```
[SAjrReg. SetProfileDev(EProfileDev.REGISTRY); ] // プロファイルアクセス先をレジストリとする
[SAjrReg. SetRegistryMode(ERegRecMode.VOLATILE); ] // レジストリへの記録を一時的なものとする

[SAjrReg. SetRegTopKey(EAjcHKey.LOCALMACHINE); ] // レジストリトップキー設定
[SAjrReg. SetRegRootPath("Software¥¥SpecialRoot"); ] // ルートパス設定 (省略時は "Software¥¥AjrCst¥¥")
[SAjrReg. SetRegMidPath("SpecialMidPath"); ] // ミドルパス設定 (省略時は、自プログラムのファイル名)

SAjrReg. Read(...); // プロファイル情報読み出し
.
.
SAjrReg. Write(...); // プロファイル情報書き込み
.
.
```

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

```
SAjrReg. SetProfileDev(EProfileDev.INIFILE); // プロファイルアクセス先をINIファイルとする
[SAjrReg. SetIniFilePath ("c:¥¥MyDir¥¥MyProg.ini"); ] // INIファイルのパス名設定

SAjrReg. Read(...); // プロファイル情報読み出し
.
.
SAjrReg. Write(...); // プロファイル情報書き込み
.
.
```

21.1. 構造体/列挙体 (定数)

21.1.1. レジストリトップキー (レジストリハイブ)

```
public enum EAjchKey : uint
{
    CLASSES_ROOT           = (0x80000000),
    CURRENT_USER           = (0x80000001),
    LOCAL_MACHINE          = (0x80000002),
    USERS                  = (0x80000003),
    PERFORMANCE_DATA      = (0x80000004),
    PERFORMANCE_TEXT      = (0x80000050),
    PERFORMANCE_NLSTEXT   = (0x80000060),
    CURRENT_CONFIG         = (0x80000005),
    DYN_DATA               = (0x80000006),
    CURRENT_USER_LOCAL_SETTINGS = (0x80000007),
}
```

21.1.2. プロファイル記録先

```
public enum EProfileDev
{
    INIFILE      = 0,          // INI ファイル
    REGISTORY    = 1,          // レジストリ
}
```

21.1.3. レジストリ書き込みモード

```
public enum ERegRecMode
{
    NON_VOLATILE = 0,          // 永続的なレジストリキー生成
    VOLATILE     = 1,          // 一時的なレジストリキー生成
}
```

21.1.4. レジストリタイプ

```
public enum ERegType
{
    NONE           = 0,        // 特定の型を持たない値
    SZ             = 1,        // 文字列値
    EXPAND_SZ      = 2,        // 展開可能な文字列値
    BINARY         = 3,        // バイナリ値
    DWORD          = 4,        // 32ビット値 (リトルエンディアン)
    DWORD_BIG_ENDIAN = 5,      // 32ビット値 (ビッグエンディアン)
    LINK           = 6,        // シンボリックリンク値
    MULTI_SZ       = 7,        // 複数の文字列値
    RESOURCE_LIST  = 8,        // リソースマップ値
    FULL_RESOURCE_DESCRIPTOR = 9, // ハードウェアリソースリスト値
    RESOURCE_REQUIREMENTS_LIST = 10, // ネストされた一連の配列値
    QWORD          = 11,       // 64ビット値 (リトルエンディアン)
}
```

21.1.5. 全コントロール設定値のセーブ/ロードオプション

```
//----- セーブオプション -----//
public enum ESaveAllOpt : int {
    NoOption          = 0          ,      // 指定無し
    ExcReadOnlyText = 0x0001      ,      // 読み出し専用テキストを除外
}

//----- ロードオプション -----//
public enum ELoadAllOpt : int {
    NoOption          = 0          ,      // 指定無し
    RbtClickEvent    = 0x0001      ,      // ラジオボタンで Click() イベント生成する
}
```


21.2. メソッド

プロファイルアクセスのメソッド一覧を示します。

#	メソッド名	内容	備考
1	SetProfileDev GetProfileDev	プロファイル記録先 設定／取得	デフォルト : REGISTRY
2	SetRegistryMode GetRegistryMode	レジストリ記録モード 設定／取得	デフォルト : NON_VOLATILE
3	SetRegTopKey GetRegTopKey	レジストリトップキー 設定／取得	デフォルト : CURRENT_USER
4	SetRegRootPath GetRegRootPath	レジストリ・ルートパス 設定／取得	デフォルト : Software¥AjrCst32 or Software¥AjrCst64
5	SetRegMidPath GetRegMidPath	レジストリ・ミドルパス 設定／取得	デフォルト : 自プログラムのファイル名
6	SetIniFilePath GetIniFilePath	.INI ファイルパス 設定／取得	デフォルト : 自プログラムのパス (拡張子は「.ini」)
7	GetProfilePath	レジストリ/INI ファイルのフルパス取得	プロファイルアクセス
8	Read	プロファイル読み出し	
9	Write	プロファイル書き込み	
10	DelSect	プロファイルセクション削除	
11	DelKey	プロファイルキー削除	
12	RemoveSect CleanupSect	プロファイルセクション消去／クリーンアップ	
13	SaveWndPos LoadWndPos	ウインドの位置を永続化	
14	SaveWndRect LoadWndRect	ウインドの位置とサイズを永続化	
15	SaveListBox LoadListBox	フォーム内 ListBox/CheckedListBox の設定値を永続化	
16	SaveComboBox LoadComboBox	フォーム内 ComboBox の設定値を永続化	
17	SaveAllCtrls LoadAllCtrls	フォーム内全コントロールの設定値を永続化	
18	RegGetUserEnv RegGetSysEnv	レジストリ中のユーザ／システム環境変数の読み出し	レジストリ内の環境変数への アクセス UserEnv: ユーザ環境変数 SysEnv : システム環境変数
19	RegPutUserEnv RegPutSysEnv	ユーザ／システム環境変数書き込み	
20	RegDelUserEnv RegDelSysEnv	ユーザ／システム環境変数消去	
21	RegAddUserEnvItem RegAddSysEnvItem	ユーザ／システム環境変数へ項目追加	
22	RegDelUserEnvItem RegDelSysEnvItem	ユーザ／システム環境変数から項目削除	
23	EnableEnvironment	環境変数の有効化	

21.2.1. プロファイル記録先 設定／取得 (SetProfileDev / GetProfileDev)

形 式 : `static void SetProfileDev(EProfileDev ProfileDev);`
`static EProfileDev GetProfileDev();`

引 数 : ProfileDev - プロファイル記録先 (REGISTRY / INIFILE)

説 明 : プロファイル記録先を設定／取得します。(デフォルト: REGISTRY)

戻り値 : なし／プロファイル記録先 (REGISTRY / INIFILE)

21.2.2. レジストリ記録モード 設定／取得 (SetRegistryMode / GetRegistryMode)

形 式 : `static void SetRegistryMode(ERegRecMode mode);`
`static ERegRecMode GetRegistryMode();`

引 数 : mode - レジストリ記録モード (NON_VOLATILE / VOLATILE)

説 明 : レジストリ記録モードを設定取得します。(デフォルト: NON_VOLATILE)
 プロファイルのアクセス先が INI ファイルの場合は、意味をなしません。

戻り値 : なし／レジストリ記録モード

21.2.3. レジストリトップキー 設定／取得 (SetRegTopKey / GetRegTopKey)

形 式 : `static void SetRegTopKey(EAjchKey TopKey);`
`static EAjchKey GetRegTopKey();`

引 数 : TopKey - レジストリトップキー (CURRENT_USER / LOCAL_MACHINE . . .)

説 明 : レジストリトップキーを設定／取得します。
 プロファイルのアクセス先が INI ファイルの場合は、意味をなしません。

戻り値 : レジストリトップキー

21.2.4. レジストリ・ルートパス 設定／取得 (SetRegRootPath / GetRegRootPath)

形 式 : `static void SetRegRootPath(string RootPath);`
`static string GetProfilePath();`

引 数 : RootPath - レジストリ・ルートパス

説 明 : レジストリ・ルートパスを設定／取得します。(デフォルト: @"Software¥AjrCst32" / @"Software¥AjrCst64")
 プロファイルのアクセス先が INI ファイルの場合は、意味をなしません。

戻り値 : なし／レジストリ・ルートパス

21.2.5. レジストリ・ミドルパス 設定／取得 (SetRegMidPath / GetRegMidPath)

形 式 : `static void SetRegMidPath(string MidPath);`
`static string GetRegMidPath();`

引 数 : MidPath - レジストリ・ミドルパス

説 明 : レジストリ・ミドルパスを設定／取得します。(デフォルト：自プログラムのパス (但し、拡張子は「.ini」))
 プロファイルのアクセス先が INI ファイルの場合は、意味をなしません。

戻り値 : なし／レジストリ・ミドルパス

21.2.6. .INI ファイルパス 設定／取得 (SetIniFilePath / GetIniFilePath)

形 式 : `static void SetIniFilePath(string IniPath);`
`static string GetIniFilePath();`

引 数 : なし

説 明 : .INI ファイルパスを設定／取得します。

説 明 : .INI ファイルのパスを設定／取得します。(デフォルト：自プログラムのパス (但し、拡張子は「.ini」))
 プロファイルのアクセス先がレジストリの場合は、意味をなしません。

戻り値 : なし／.INI ファイルパス

21.2.7. レジストリ/INI ファイルのフルパス取得 (GetProfilePath)

形 式 : `static string GetProfilePath();`

引 数 : なし

説 明 : 現在設定されているフルパス名を取得します。

プロファイルのアクセス先がレジストリの場合は、HKEY_CURRENT_USER 下のフルパス名 (ルートパス+ミドルパス) を取得します。

プロファイルのアクセス先が INI ファイルの場合は、INI ファイルのフルパス名を取得します。

戻り値 : フルパス名

21.2.8. プロファイルの読み出し (Read / ReadHex / ReadH64)

形式 : static bool Read (string sec, string key, bool defValue); -- ブール値
 static uint Read (string sec, string key, uint defValue); -- 符号なし 32 ビット整数
 static int Read (string sec, string key, int defValue); -- 符号付き 32 ビット整数
 static uint ReadHex(string sec, string key, uint defValue); -- 32 ビット 16 進数
 static ulong Read (string sec, string key, ulong defValue); -- 符号なし 64 ビット整数
 static long Read (string sec, string key, long defValue); -- 符号付き 64 ビット整数
 static ulong ReadH64(string sec, string key, ulong defValue); -- 64 ビット 16 進数
 static double Read (string sec, string key, double defValue); -- 実数
 static string Read (string sec, string key, string defValue); -- 文字列

引数 : sec - プロファイル・セクション名
 key - プロファイル・キー名称 (値の名称)
 defValue - デフォルト値

説明 : プロファイルから値を読み出します。
 プロファイル内に、当該セクション／キーが無い場合は、defValue で指定したデフォルト値を返します。

戻り値 : プロファイルから読み出した値、あるいは、デフォルト値

21.2.9. プロファイルの書き込み (Write / WriteHex / WriteH64)

形式 : static void Write (string sec, string key, bool value); -- ブール値
 static void Write (string sec, string key, uint value); -- 符号なし 32 ビット整数
 static void Write (string sec, string key, int value); -- 符号付き 32 ビット整数
 static void WriteHex(string sec, string key, uint value); -- 32 ビット 16 進数
 static void Write (string sec, string key, ulong value); -- 符号なし 64 ビット整数
 static void Write (string sec, string key, long value); -- 符号付き 64 ビット整数
 static void WriteH64(string sec, string key, ulong value); -- 64 ビット 16 進数
 static void Write (string sec, string key, double value); -- 実数
 static void Write (string sec, string key, string value); -- 文字列

引数 : sec - プロファイル・セクション名
 key - プロファイル・キー名称 (値の名称)
 value - プロファイルへ書き込む値

説明 : プロファイル内の指定セクション、キーへ値を書き込みます。

戻り値 : なし

21.2.10. プロファイル・セクション削除 (DelSect)

形式 : static void DelSect(string sec);

引数 : sec - 削除するプロファイル・セクション名

説明 : プロファイル内の当該セクションと、セクションに属するすべてのキーを削除します。

戻り値 : なし

21.2.11. プロファイル・キー削除 (DelKey)

形式 : static void DelKey(string sec, string key);

引数 : sec - 削除するキーが属するプロファイル・セクション名
 key - 削除するキーの名称

説明 : プロファイル・セクション内の指定キーを削除します。

戻り値 : なし

21.2.12. プロファイル・セクション消去/クリーンアップ (RemoveSect / CleanupSect)

- 形 式** : `static void RemoveSect(string sec);`
`static void CleanupSect(string sec);`
- 引 数** : `sec` - 削除するキーが属するプロファイル・セクション名
`key` - 削除するキーの名称
- 説 明** : `CleanupSect()` は、プロファイル・セクション内のすべてのキーやサブセクションを削除します。
`RemoveSect()` は、指定したセクション自体も削除します。
- 戻り値** : なし

21.2.13. ウインド位置を永続化 ({Save/Load}WndPos)

- 形 式** : **―― ウインド位置をプロファイルにセーブ ――**
`static void SaveWndPos(object wnd);`
`static void SaveWndPos(object wnd, string prefix);`
`static void SaveWndPos(object wnd, string sec, string prefix);`
―― ウインド位置をプロファイルからロード ――
`static void LoadWndPos(object wnd);`
`static void LoadWndPos(object wnd, string prefix);`
`static void LoadWndPos(object wnd, string sec, string prefix);`
- 引 数** : `wnd` - 位置を退避するウインド (フォーム等)
`sec` - ウインド位置を退避するプロファイルのセクション名 (省略時は “WndPos”)
`prefix` - ウインド位置を退避するプロファイル・キーの先頭部分 (省略時は “WP”)
- 説 明** : ウインドの位置を、プロファイルにセーブ/ロードし永続化します。
`SaveWndPos()` は、ウインド位置をプロファイルに記録します。
`LoadWndPos()` は、ウインド位置をプロファイルから読み出して設定します。
ウインドがモニタ外である (タイトルバーが 15%以下しか見えない) 場合は、ウインドを原点に移動します。
ウインド位置が記録されていない場合は、何もしません。
- 戻り値** : なし

21.2.14. ウインドの位置とサイズを永続化 ({Save/Load}WndRect)

- 形 式** : **―― ウインド位置とサイズをプロファイルにセーブ ――**
`static void SaveWndRect(object wnd);`
`static void SaveWndRect(object wnd, string prefix);`
`static void SaveWndRect(object wnd, string sec, string prefix);`
―― ウインド位置とサイズをプロファイルからロード ――
`static void LoadWndRect(object wnd);`
`static void LoadWndRect(object wnd, string prefix);`
`static void LoadWndRect(object wnd, string sec, string prefix);`
- 引 数** : `wnd` - 位置を退避するウインド (フォーム等)
`sec` - ウインド位置を退避するプロファイルのセクション名 (省略時は “WndRect”)
`prefix` - ウインド位置を退避するプロファイル・キーの先頭部分 (省略時は “WP”)
- 説 明** : ウインドの位置とサイズを、プロファイルにセーブ/ロードし永続化します。
`SaveWndRect()` は、ウインド位置とサイズをプロファイルに記録します。
`LoadWndRect()` は、ウインド位置とサイズをプロファイルから読み出して設定します。
ウインドがモニタ外である (タイトルバーが 15%以下しか見えない) 場合は、ウインドを原点に移動します。
ウインド位置とサイズが記録されていない場合は、何もしません。
- 戻り値** : なし

21.2.15. フォーム内 ListBox/CheckedListBox の設定値を永続化 ({Save|Load}ListBox)

形 式 : static void SaveListBox(Control from, Control list_box); --- セーブ
static void LoadListBox(Control form, Control list_box); --- ロード

引 数 : form - フォームオブジェクト
list_box - ListBox/CheckedListBox オブジェクト

説 明 : フォーム内 ListBox/CheckedListBox の設定値を永続化します。
SaveListBox() は、フォーム内 ListBox/CheckedListBox の設定値をプロファイルに記録します。
LoadListBox() は、プロファイルから ListBox/CheckedListBox の設定値を読み出して設定します。
セーブ/ロードする内容は以下の通りです。

- ・リストボックス項目 (但し、ToString() で文字列化します)
- ・選択状態
- ・チェック状態 (CheckedListBox のみ)

ListBox/CheckedListBox 以外のオブジェクトを指定した場合は何もしません。

戻り値 : なし

21.2.16. フォーム内 ComboBox の設定値を永続化 ({Save|Load}ComboBox)

形 式 : static void SaveComboBox(Control from, Control combo_box); --- セーブ
static void LoadComboBox(Control form, Control combo_box); --- ロード

引 数 : form - フォームオブジェクト
combo_box - ComboBox オブジェクト

説 明 : SaveComboBox() は、フォーム内 ComboBox の設定値をプロファイルに記録します。
LoadComboBox() は、プロファイルから ComboBox の設定値を読み出して設定します。
セーブ/ロードする内容は以下の通りです。

- ・コンボボックス項目 (但し、ToString() で文字列化します)
- ・選択状態

ComboBox 以外のオブジェクトを指定した場合は何もしません。

戻り値 : なし

21.2.17. フォーム内全コントロールの設定値を永続化({Save|Load}AllCtrls)

形 式 : **―― プロファイルにセーブ ――**

```
static void SaveAllCtrls (Control from); ----- 全てセーブ
static void SaveAllCtrls (Control form, string[] names); ----- セーブする名称を指定
static void SavrAllCtrlsExc(Control form, string[] excludes); -- セーブしない名称を指定
```

―― プロファイルにセーブ(オプション指定) ――

```
static void SaveAllCtrls (Control from, ESaveAllOpt opt); ----- 全てセーブ
static void SaveAllCtrls (Control form, string[] names, ESaveAllOpt opt); ----- セーブする名称を指定
static void SavrAllCtrlsExc(Control form, string[] excludes, ESaveAllOpt opt); -- セーブしない名称を指定
```

―― プロファイルからロード ――

```
static void LoadAllCtrls (Control from); ----- 全てロード
static void LoadAllCtrls (Control from, string[] names); ----- ロードする名称を指定
static void LoadAllCtrlsExc(Control from, string[] excludes); -- ロードしない名称を指定
```

―― プロファイルからロード (オプション指定) ――

```
static void LoadAllCtrls (Control from, ELoadAllOpt opt); ----- 全てロード
static void LoadAllCtrls (Control form, string[] names, ELoadAllOpt opt); ----- ロードする名称を指定
static void LoadAllCtrlsExc(Control from, string[] excludes, ELoadAllOpt opt); -- ロードしない名称を指定
```

引 数 : form - フォームオブジェクト
names - 対象とするコントロール名/タイプ名 (null:全てセーブ/ロード)
excludes - 対象外とするコントロール名/タイプ名
opt - ロード/セーブオプション (未指定時はNoOptionを仮定)

説 明 : SaveAllCtrls()/SavrAllCtrlsExc()は、フォーム内の全コントロールの設定値をプロファイルに記録します。
SaveAllCtrls()/SavrAllCtrlsExc()では、以下のオプションを指定できます。

opt の指定	値	内容
NoOption	0	オプション指定無し
ExcReadOnlyText	0x0001	読み出し専用テキストは除外する

LoadAllCtrls()/LoadAllCtrlsExc()は、プロファイルからフォーム内の全コントロールの設定値を読み出して設定します。
names で対象とする名称を指定した場合は、当該コントロールの設定値をセーブ/ロードします。

excludes で対象外とする名称を指定した場合は、当該コントロールの設定値はセーブ/ロードしません。

通常、names/excludes は、セーブ/ロードで同じ名称を指定します。

SaveAllCtrls()/SavrAllCtrlsExc()が実行されていない場合、LoadAllCtrls()/LoadAllCtrlsExc()を実行しても各コントロールの値は変化しません。

LoadAllCtrls()/LoadAllCtrlsExc()で以下のオプションを指定できます。

opt の指定	値	内容
NoOption	0	オプション指定無し
RbtClickEvent	0x0001	全てのチェックされているラジオボタンでClick()イベント生成する

対象となるコントロールのタイプ名とセーブ/ロードする内容は以下の通りです。

#	コントロールのタイプ名	セーブする内容	備考 (プロパティ)	読出順
1	TextBox	設定テキスト	Text プロパティ	1
2	CheckBox	チェック状態	Checked プロパティ	
3	RadioButton	チェック状態	Checked プロパティ	
4	ListBox / CheckedListBox	項目名, 選択状態, チェック状態	{Save/Load}ListBox() 実行	2
5	ComboBox	項目名, 選択状態	{Save/Load}ComboBox() 実行	
6	CAjrInpValue	設定数値	Value / IntValue プロパティ	3
7	CAjrTimeChart	フィルタ選択状態 ゲージ情報 (ゲージ位置, 単位時間)		
8	CAjrVT100	フォント 行間スペース	FontVT100 LineSpace	

- ・上記表で示すコントロールタイプのみがセーブ/ロードの対象となります。
- ・names や excludes で指定する名称は、上記コントロール内でのタイプ名やコントロールの名称を指定します。
- ・「読出順」は、プロファイルからコントロールの設定値を読み出す際の順番です。(同一読出順内では順不同)

戻り値 : なし

備 考 : ロード時にテキストボックスで、値が変化した場合はTextChanged() イベントが。チェックボックスやラジオボタンで、値が変化した場合はClickedChaned() イベントが発生します。

21.2.18. レジストリ中の環境変数の読み出し (GetUserEnv / GetSysEnv)

形 式 : static string GetUserEnv(string name); ----- HKEY_CURRENT_USER へのアクセス
 static string GetUserEnv(string name, out ERegType type);
 static string GetSysEnv (string name); ----- HKEY_LOCAL_MACHINE へのアクセス
 static string GetSysEnv (string name, out ERegType type);

引 数 : name - 読み出すレジストリ内の環境変数名
 type - 読み出したレジストリのタイプを格納する変数

説 明 : レジストリに記録されている環境変数を取得します。

戻り値 : ≠null : レジストリから読み出した環境変数の内容
 =null : 読み出し失敗

21.2.19. レジストリ中の環境変数へ書き込み (RegPutUserEnv / RegPutSysEnv)

形 式 : static bool PutUserEnv(string name, string text); ----- HKEY_CURRENT_USER へのアクセス
 static bool PutUserEnv(string name, string text, ERegType type);
 static bool PutSysEnv (string name, string text); ----- HKEY_LOCAL_MACHINE へのアクセス
 static bool PutSysEnv (string name, string text, ERegType type);

引 数 : name - 書き込むレジストリ内の環境変数名
 text - 書き込み環境変数の内容
 type - 新規作成となった場合のレジストリタイプ (ERegType.SZ (デフォルト) / ERegType.EXPAND_SZ)

説 明 : レジストリ内の環境変数を書き込みます。
 当該環境変数が存在しない場合は、環境変数を新規作成します。

戻り値 : true - 成功
 false - 失敗

21.2.20. レジストリ中の環境変数を消去 (RegDelUserEnv / RegDelSysEnv)

形 式 : static bool DelUserEnv(string name; ----- HKEY_CURRENT_USER へのアクセス
 static bool DelSysEnv (string name); ----- HKEY_LOCAL_MACHINE へのアクセス

引 数 : name - 消去するレジストリ内の環境変数名

説 明 : レジストリ内の環境変数を消去します。

戻り値 : true - 成功
 false - 失敗

21.2.21. レジストリ中の環境変数へ項目を追加 (AddUserEnvItem / AddSysEnvItem)

形 式 : static bool AddUserEnvItem (string name, string item, char dlm); ----- HKEY_CURRENT_USER へのアクセス
 static bool AddUserEnvItem (string name, string item, char dlm, bool fFront);
 static bool AddUserEnvItem (string name, string item, char dlm, ERegType type);
 static bool AddUserEnvItem (string name, string item, char dlm, bool fFront, ERegType type);

static bool AddSysEnvItem (string name, string item, char dlm); ----- HKEY_LOCAL_MACHINE へのアクセス
 static bool AddSysEnvItem (string name, string item, char dlm, bool fFront);
 static bool AddSysEnvItem (string name, string item, char dlm, ERegType type);
 static bool AddSysEnvItem (string name, string item, char dlm, bool fFront, ERegType type);

引 数 : name - レジストリ内の環境変数名
 item - 追加する項目文字列
 dlm - 区切り文字
 fFront - 追加位置 (true:先頭, false:末尾 (デフォルト))
 type - 新規作成となった場合のレジストリタイプ (ERegType.SZ (デフォルト) / ERegType.EXPAND_SZ)

説 明 : レジストリの環境変数へ「item」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”)
 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、何もしません。

戻り値 : true - 成功
 false - 失敗

21.2.22. レジストリ中の環境変数から項目を削除 (DelUserEnvItem / DelSysEnvItem)

形 式 : static bool DelUserEnvItem(string name, string item, char dlm); ----- HKEY_CURRENT_USER へのアクセス
 static bool DelUserEnvItem(string name, string item, char dlm, bool fErase);

static bool DelSysEnvItem (string name, string item, char dlm); ----- HKEY_LOCAL_MACHINE へのアクセス
 static bool DelSysEnvItem (string name, string item, char dlm, bool fErase);

引 数 : name - レジストリ内の環境変数名
 item - 削除する項目文字列
 dlm - 区切り文字
 fErase - 削除の結果項目が1つも無くなった場合の動作 (true:環境変数を削除, false:空文字列とする (デフォルト))

説 明 : レジストリの環境変数から「item」で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(;)で区切られた項目”BBB”を削除 → “AAA; CCC”)
 環境変数中に指定した項目が無い場合は、false を返します。

戻り値 : true - 成功
 false - 失敗

21.2.23. 環境変数の有効化 (AjrRegEnableEnvironment)

形 式 : void EnableEnvironment ();

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

22. スタティク : ファイル/フォルダ操作 (CAjrStatic.dll, SAjrFop クラス)

ファイルやフォルダのコピー, 削除等の操作スタティクメソッド群です。クラス名は「SAjrFop」です。

22.1. メソッド

ファイル/フォルダ操作のメソッド一覧を示します。

#	メソッド名	内容	備考
1	CopyFolderStruct	フォルダ構造のコピー	
2	RemoveFolder	フォルダ削除	
3	CopyFiles	ファイル群のコピー	
4	CleanFolder	フォルダのクリーンアップ (フォルダ下の全ファイルとディレクトリ削除)	
5	EnumFileMatchingList	2つのフォルダ下のファイル群・突合せリスト生成	
6	PathExists	パスが存在するかチェック	
7	IsPathDirectory	パスがディレクトリかチェック	
8	IsPathFile	パスがファイルかチェックする	
9	GetFileSize	ファイルサイズ取得	
10	GetFileTime1970	ファイルタイム取得 (1970/01/01 00:00:00 からの通算秒数)	
11	FileCompare	ファイル比較	

22.1.1. フォルダ構造のコピー (CopyFolderStruct)

形 式 : static bool CopyFolderStruct(string dirFrom, string dirTo);
 static bool CopyFolderStruct(string dirFrom, string dirTo, IntPtr cbp);
 static bool CopyFolderStruct(string dirFrom, string dirTo, FopCbKScpNtc cbCpyNtc);
 static bool CopyFolderStruct(string dirFrom, string dirTo, IntPtr cbp, FopCbKScpNtc cbCpyNtc);
 static bool CopyFolderStruct(string dirFrom, string dirTo, FopCbKScpNtc cbCpyNtc, FopCbKScpQry cbCpyQry);
 static bool CopyFolderStruct(string dirFrom, string dirTo, IntPtr cbp, FopCbKScpNtc cbCpyNtc, FopCbKScpQry cbCpyQry);

static bool CopyFolderStruct(string dirFrom, string dirTo, EFscOpt opt, FopCbKScpNtc cbCpyNtc);
 static bool CopyFolderStruct(string dirFrom, string dirTo, EFscOpt opt, FopCbKScpNtc cbCpyNtc, FopCbKScpQry cbCpyQry);
 static bool CopyFolderStruct(string dirFrom, string dirTo, EFscOpt opt, IntPtr cbp, FopCbKScpNtc cbCpyNtc, FopCbKScpQry cbCpyQry);

引 数 : dirFrom - コピー元先頭フォルダパス名
 dirTo - コピー先頭フォルダパス名
 cbp - コールバックパラメタ (省略時は 0)
 cbScpNtc - フォルダコピー通知用コールバックメソッド
 cbScpQry - フォルダコピー可否/フォルダ名変更問い合わせ用コールバックメソッド
 opt - オプション (EFscOpt)

説 明 : 「PathTo」で指定したフォルダの下に、「PathFrom」下のフォルダ構造を作成します。
 「PathTo」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(opt)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。

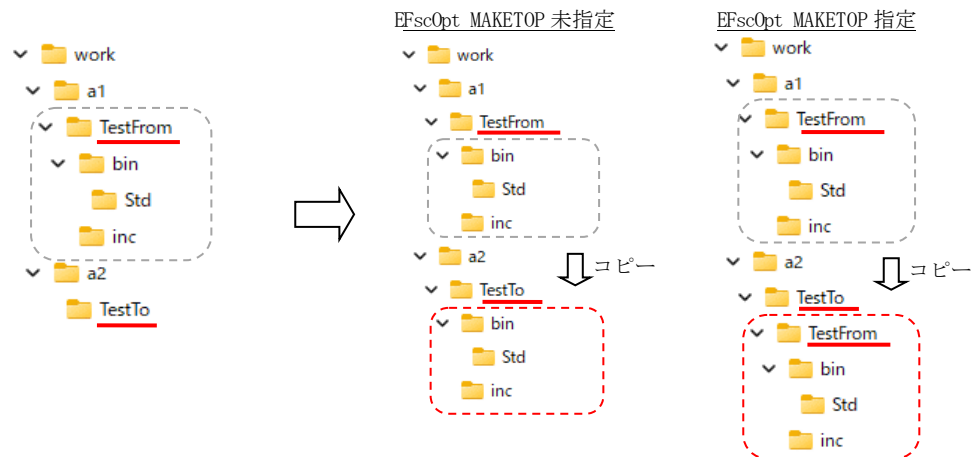
opt	コピー先フォルダ	同一フォルダ無し (新規作成)	同一フォルダあり
NONE		実際にフォルダを作成した現在日時	既存フォルダの日時のまま
SAMETIME		コピー元フォルダの日時と同じ (※1)	既存フォルダの日時のまま
FORCETIME		コピー元フォルダの日時と同じ (※1)	コピー元フォルダの日時と同じ (※1)

※1 : エクスプローラ等で、当該フォルダを表示/参照している場合、日時が設定されない場合があります。

●その他のオプション

Opt	内容
MAKETOP	転送元末尾フォルダ名をの転送先の先頭フォルダとして作成する (下記、例参照)
ALLOWEXIST	同一フォルダが存在してもエラーとしない

(例) CopyFolderStruct(@"f:¥work¥a1¥TestFrom", @"f:¥work¥a2¥TestTo", EFscOpt. XXXXX, ...);



コールバックメソッド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「EntcDirCopy. 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、(opt=ALLOWEXIST 指定では)処理を続行します。

既に同一名称の「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

コールバック : コールバックメソッドの形式は、以下の通りです。

cbScpNtc (フォルダコピー通知)

形 式 : `bool cbScpNtc(string dirFrom, string dirTo, ENtcDirCopy ntc, IntPtr cbp);`

パラメタ :

<code>dirFrom</code>	- コピー元フォルダパス名
<code>dirTo</code>	- コピー先フォルダパス名
<code>ntc</code>	- 通知コード
	SUCCESS : フォルダ作成成功
	FAILURE : フォルダ作成失敗
	EXIST : 既に同一フォルダが存在する
<code>cbp</code>	- コールバックパラメタ

説 明 : `CopyFolderStruct()`により、フォルダがコピーされたことを通知します。

戻り値 : `true` - フォルダ構造のコピー操作を継続する
`false` - フォルダ構造のコピー操作を中止する

cbScpQry (フォルダコピー可否/フォルダ名変更問い合わせ)

形 式 : `string cbScpQry(string dirFront, string dirNew, IntPtr cbp);`

パラメタ :

<code>dirFront</code>	- 作成するフォルダの位置 (上位フォルダのパス名)
<code>dirNew</code>	- 作成するフォルダ名
<code>cbp</code>	- コールバックパラメタ

説 明 : フォルダコピーの可否/フォルダ名変更を問い合わせるために呼び出されます。
 フォルダを作成する場合は、引数の「DirNew」をそのまま返します。
 フォルダ名を変更して作成する場合は、変更するフォルダ名を返します。
 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` - フォルダを作成します (作成するフォルダの名称)
`=null` - フォルダを作成しない

22.1.2. ファイル群のコピー(CopyFiles)

形 式 : static bool CopyFiles(string PathFrom, string PathTo);
 static bool CopyFiles(string PathFrom, string PathTo, string WildCard);
 static bool CopyFiles(string PathFrom, string PathTo, string WildCard, EcpyfOpt opt);
 static bool CopyFiles(CopyFiles(string PathFrom, string PathTo, string WildCard, EcpyfOpt opt, FopCbKfcpNtc cbFcpNtc);
 static bool CopyFiles(string PathFrom, string PathTo, string WildCard, EcpyfOpt opt, IntPtr cbp, FopCbKfcpNtc cbFcpNtc);

 static bool CopyFiles(string PathFrom, string PathTo, string WildCard, EcpyfOpt opt,
 FopCbKfcpNtc cbFcpNtc, FopCbKfcpQry cbFcpQry);

 static bool CopyFiles(string PathFrom, string PathTo, string WildCard, EcpyfOpt opt, IntPtr cbp,
 FopCbKfcpNtc cbFcpNtc, FopCbKfcpQry cbFcpQry);

引 数 : PathFrom - コピー元先頭フォルダパス名
 PathTo - コピー先頭フォルダパス名
 WildCard - コピー/除外するファイルをワイルドカードで指定する(省略/null 指定時は全ファイルをコピー);
 (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
 cbp - コールバックパラメタ
 cbFcpNtc - ファイルコピー通知用コールバックメソッド
 cbFcpQry - ファイルコピー可否/ファイル名変更問い合わせ用コールバックメソッド

説 明 : 「PathFrom」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「PathTo」で指定したフォルダ下へコピーします。
 「PathFrom」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しません。

opt(オプション)は、以下の合成値を指定します。

名称	内容
NONE	オプション無し
CREATEALWAYS	既にファイルが存在する場合、上書きする
WILDExc	ワイルドカードに一致するファイルを除外する

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。)
 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)

コールバックメソッドを介して、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

コールバック : コールバックメソッドの形式は、以下の通りです。

cbFcpNtc (ファイルコピー通知)

形 式 : bool cbFcpNtc(string FileFrom, string FileTo, ENtcFileCopy ntc, IntPtr cbp);

パラメタ : FileFrom - コピー元ファイルパス名
 FileTo - コピー先ファイルパス名
 ntc - 通知コード
 SUCCESS : ファイル作成成功
 FAILURE : ファイル作成失敗
 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 : CopyFiles()により、ファイルがコピーされたことを通知します。

戻り値 : true - ファイルのコピー操作を継続する
 false - ファイルのコピー操作を中止する

cbFcpQry (ファイルコピー可否/ファイル名変更問い合わせ)

形 式 : string cbFcpQry(string FileFrom, string FileTo, string FileName, ref EFileAtt att, IntPtr cbp);

パラメタ : FileFrom - コピー元ファイルのパス名
 FileTo - コピー先ファイルのパス名 (ファイル名の部分は、FileFrom と同じ)
 FileName - コピー先ファイル名 (パス名を含まないファイル名部分のみ (ex. "Sample.txt"))
 att - コピー元ファイル属性
 cbp - コールバックパラメタ

説 明 : ファイルコピーの可否/ファイル名変更を問い合わせるために呼び出されます。
 コピー先ファイル名を変更しないでコピーする場合は、引数の「FileNew」をそのまま返します。
 コピー先ファイル名を変更してコピーする場合は、変更するファイル名を返します。
 このファイルをコピーしない場合は、null を返します。
 コピー先のファイル属性を変更する場合は、att の内容を変更してください。

戻り値 : ≠null - ファイルをコピーします (作成するファイルの名称)
 =null - ファイルをコピーしない

22.1.3. フォルダ削除(RemoveFolder)

形 式 : static bool RemoveFolder(string path);
 static bool RemoveFolder(string path, IntPtr cbp);
 static bool RemoveFolder(string path, FopCbKrmvDir cbRmv);
 static bool RemoveFolder(string path, IntPtr cbp, FopCbKrmvDir cbRmv);

引 数 : path - 削除するフォルダパス名
 cbp - OnRemoveDir イベント発生時の通知コード (省略時は 0)
 cbRmv - 削除したフォルダ/ファイル通知用コールバックメソッド

説 明 : 「path」で指定したフォルダ自体とその下の全てのサブフォルダとファイルを削除します。
 フォルダ中に「読み取り専用」や「隠しファイル」属性を持っているファイルも全て削除します。

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、コールバックメソッド (cbRmv()) 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

コールバック : コールバックメソッドの形式は、以下の通りです。

cbRmv (フォルダ削除通知)

形 式 : bool cbkRmv(string PathName, uint ntc, IntPtr cbp);

パラメタ : PathName - 削除するフォルダ/ファイルのパス名
 Ntc - 通知情報 bit0 : ファイル/フォルダ識別 (0 : ファイル, 1 : フォルダ)
 bit7 : エラーフラグ (0 : OK, 1 : エラー)
 cbp - コールバックパラメタ

説 明 : RemoveFolder()により、フォルダ/ファイルを削除することを通知します。

戻り値 : true - フォルダの削除操作を継続する
 false - フォルダの削除操作を中止する

22.1.4. フォルダのクリーンアップ (CleanFolder)

形 式 : bool CleanFolder (string path);
 bool CleanFolder (string path, IntPtr cbp);
 bool CleanFolder (string path, IntPtr cbp, FopCbKrmvDir cbRmv);

引 数 : path - クリーンアップするフォルダパス名
 cbp - コールバックパラメタ
 cb - 削除したフォルダ/ファイル通知用コールバックメソッド

説 明 : 「path」で指定したフォルダの下の全てのディレクトリとファイルを削除します。
 「path」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」や「隠しファイル」属性を持っているファイルも全て削除します。

戻り値 : true - 成功
 false - 失敗

コールバック : コールバックメソッドの形式は、以下の通りです。

cbRmv (フォルダ削除通知)

形 式 : bool cbkRmv(string PathName, bool fDir, IntPtr cbp);

パラメタ : PathName - 削除するフォルダ/ファイルのパス名
 fDir - フォルダ(true)/ファイル(false)識別フラグ
 cbp - コールバックパラメタ

説 明 : CleanFolder() により、フォルダ/ファイルを削除することを通知します。

戻り値 : true - フォルダのクリーンアップ操作を継続する
 false - フォルダのクリーンアップ操作を中止する

22.1.5. 2つのフォルダ下のファイル群・突き合せリスト列挙 (EnumFileMatchingList)

形 式 : int EnumFileMatchingList (string path1, string path2, IntPtr cbp, FopCbKfmlEnum cbEnum);
 int EnumFileMatchingList (string path1, string path2, EFmlOpt opt, IntPtr cbp, FopCbKfmlQuery cbQuery, FopCbKfmlEnum cbEnum);

引 数 : path1, path2 - 突き合せリストを作成する2つのフォルダ
 opt - オプション
 cbp - コールバックパラメータ
 cbQuery - ファイル通知用コールバックメソッド (不要時はnull)
 cbEnum - 突き合せリスト項目通知用コールバック (不要時はnull)

説 明 : 「path1」と「path2」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。
 「path1」と「path2」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つのフォルダからの相対パス (2つのパス下のサブフォルダ) とファイル名/末尾のディレクトリ名が同じファイル/ディレクトリのペアで作成されます。
 「path1」か「path2」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、以下の以下の値の合成値を指定します。

シンボル	値	内容
BOTH	0	突き合せリストにファイルとディレクトリを含める
NOFILE	0x01	突き合せリストにファイルを含めない
NODIR	0x02	突き合せリストにディレクトリを含めない (デフォルト)

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。(EFmlOpt.NODIR 指定時)

D:¥work + Path1 - a.txt, b.txt + Child - c1.bmp + grandson ----- gs.pdf + granddaughter -	D:¥temp + Path2 - a.txt + Child - c2.bmp + grandson ----- gs.pdf + granddaughter - gd.docx

リストアップ内容

#	リスト項目内容	備考
1	D:¥work¥Path1¥a.txt D:¥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	- D:¥temp¥Path2¥Child¥c2.bmp	.¥Child¥c2.bmp はPath2 側にだけ存在する
5	- D:¥temp¥Path2¥Child¥ 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/ディレクトリ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順次通知されます。

戻り値 : ≥ 0 - 成功 (リストアップした項目数)
 < 0 - 失敗 (-1): pPath1, pPath2 が共に NULL (-4): システムAPIエラー
 (-2): pPath1 で指定したフォルダが存在しない (-5): メモリエラー
 (-3): pPath2 で指定したフォルダが存在しない (-9): コールバックから中止指示

コールバック : コールバックメソッドの形式は、以下の通りです。

cbQuery (検出したファイルの通知)

形 式 : `bool cbQuery(int id, string path, FileAttributes att, ref bool fDiscard, IntPtr cbp);`

引 数 :

- `id` : 2つのパスの識別 (0:パス1側, 1:パス2側)
- `path` : ファイルパス名
- `att` : ファイル属性
- `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を突合せリストから除外する場合は、`fDiscard=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

cbEnum (突き合せリスト項目の通知)

形 式 : `bool cbEnum(bool valid1, uint utc1, FileAttributes att1, long size1, string path1, string name1, string tail1, bool valid2, uint utc2, FileAttributes att2, long size2, string path2, string name2, string tail2, IntPtr cbp);`

引 数 :

- `valid1` : パス1下ファイル/ディレクトリの有無 (true:あり、false:なし)
- `utc1` : " " のタイムスタンプ (UTC 時刻, 1970/1/1 00:00:00 からの通算秒数)
- `att1` : " " のファイル属性
- `size1` : " " のファイルサイズ (バイト数)
- `path1` : " " のファイル/ディレクトリパス名
- `name1` : " " のファイル名+拡張子 (ディレクトリの場合は末尾パス名)
- `tail1` : " " のファイル/ディレクトリパス名の後部部分 (指定したパスに後続する部分)
- `valid2` : パス2下ファイル/ディレクトリの有無 (true:あり、false:なし)
- `utc2` : " " のタイムスタンプ (UTC 時刻, 1970/1/1 00:00:00 からの通算秒数)
- `att2` : " " のファイル属性
- `size2` : " " のファイルサイズ (バイト数)
- `path2` : " " のファイル/ディレクトリパス名
- `name1` : " " のファイル名+拡張子 (ディレクトリの場合は末尾パス名)
- `tail1` : " " のファイル/ディレクトリパス名の後部部分 (指定したパスに後続する部分)
- `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

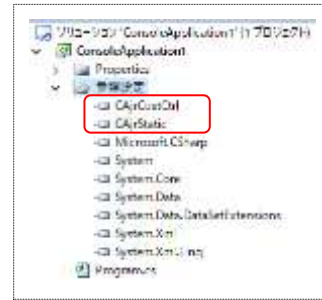
使用例 1 : 次のプログラムは、「d:¥work¥path1」と「d:¥temp¥path2」フォルダからの相対パス（2つのパス下のサブフォルダ）とファイル名が同じファイルのペアで双方のファイル情報を表示します。
ファイルだけの突合せリストを作成します。（ディレクトリの突合せ情報は除外します）

※参照設定に「CAjrCustCtrl」と「CAjrStatic」を追加してください。

```

1 : using System;
2 : using System.IO;
3 : using System.Collections.Generic;
4 : using System.Linq;
5 : using System.Text;
6 : using CAjrCustCtrl;
7 :
8 : namespace ConsoleApplication1
9 : {
10 :     class Program
11 :     {
12 :         static CAjrStatic m_CAjrStatic = new CAjrStatic();
13 :         static void Main(string[] args)
14 :         {
15 :             int rsu;
16 :             FopCbkFmlEnum CbkEnum = new FopCbkFmlEnum(CbkFmlEnum);
17 :             rsu = SAjrFop.EnumFileMatchingList(@"d:¥work¥path1", @"d:¥temp¥path2", (IntPtr)0, CbkFmlEnum);
18 :             Console.WriteLine("Result = " + rsu.ToString() + "¥n");
19 :             Console.ReadLine();
20 :         }
21 :         private static bool CbkFmlEnum(bool fValid1, uint utc1, FileAttributes att1, long size1, string path1,
22 :                                         bool fValid2, uint utc2, FileAttributes att2, long size2, string path2, IntPtr cbp)
23 :         {
24 :             Console.WriteLine("-----¥n");
25 :             if (valid1) Console.WriteLine(" 1: path=" + path1 + ", name=" + name1 + ", tail=" + tail1 + "¥n");
26 :             else Console.WriteLine(" 1: -¥n");
27 :             if (valid2) Console.WriteLine(" 2: path=" + path2 + ", name=" + name2 + ", tail=" + tail2 + "¥n");
28 :             else Console.WriteLine(" 2: -¥n");
29 :             return true;
30 :         }
31 :     }
32 : }

```



表示出力

```

File:///R:/AjrCustCtrl/_obj/bin/5il_FileSearchC.EXE
-----
1: path=d:¥work¥path1¥a.txt, name=a.txt, tail=a.txt
2: path=d:¥temp¥path2¥a.txt, name=a.txt, tail=a.txt
-----
1: path=d:¥work¥path1¥b.txt, name=b.txt, tail=b.txt
2: -
-----
1: path=d:¥work¥path1¥Child¥c1.bmp, name=c1.bmp, tail=Child¥c1.bmp
2: -
-----
1: -
2: path=d:¥temp¥path2¥Child¥c2.bmp, name=c2.bmp, tail=Child¥c2.bmp
-----
1: -
2: path=d:¥temp¥path2¥Child¥grounddaughter¥d.docx, name=d.docx, tail=Child¥grounddaughter¥d.docx
-----
1: path=d:¥work¥path1¥Child¥groundsun¥g.pdf, name=g.pdf, tail=Child¥groundsun¥g.pdf
2: path=d:¥temp¥path2¥Child¥groundsun¥g.pdf, name=g.pdf, tail=Child¥groundsun¥g.pdf
Result = 6

```

.¥a.txt は、
双方のパスに存在する

.¥Child¥b.txt は、
パス 1 側だけに存在する

.¥Child¥c2.bmp は、
パス 2 側だけに存在する

使用例 2 : 使用例 1 に加えて、「*.txt」と「*.docx」を除外します。

```

1 : using System;
2 : using System.IO;
3 : using System.Collections.Generic;
4 : using System.Linq;
5 : using System.Text;
6 : using CAjrCustCtrl;
7 :
8 : namespace ConsoleApplication1
9 : {
10 :     class Program
11 :     {
12 :         static CAjrStatic m_CAJrStatic = new CAjrStatic();
13 :         static void Main(string[] args)
14 :         {
15 :             int rsu;
16 :             FopCbKfmlQuery CbkQuery = new FopCbKfmlQuery(CbkFmlQuery);
17 :             FopCbKfmlEnum CbkEnum = new FopCbKfmlEnum(CbkFmlEnum);
18 :             rsu = SAjrFop.EnumFileMatchingList(@"d:\work\path1", @"d:\temp\path2", EFmlOpt.NODIR, (IntPtr)0, CbkQuery, CbkFmlEnum);
19 :             Console.WriteLine("Result = " + rsu.ToString() + "\n");
20 :             Console.ReadLine();
21 :         }
22 :         private static bool CbkFmlQuery(int id, string path, FileAttributes att, ref bool fDiscard, IntPtr cbp)
23 :         {
24 :             if (SAjrFop.PathMatchSpecs(path, "*.txt ; *.docx"))
25 :             {
26 :                 fDiscard = true;
27 :             }
28 :             return true;
29 :         }
30 :         private static bool CbkFmlEnum(bool valid1, uint utc1, FileAttributes att1, long size1, string path1, string name1, string tail1,
31 :             bool valid2, uint utc2, FileAttributes att2, long size2, string path2, string name2, string tail2, IntPtr cbp)
32 :         {
33 :             Console.WriteLine("-----\n");
34 :             if (valid1) Console.WriteLine(" 1: path=" + path1 + ", name=" + name1 + ", tail=" + tail1 + "\n");
35 :             else Console.WriteLine(" 1: -\n");
36 :             if (valid2) Console.WriteLine(" 2: path=" + path2 + ", name=" + name2 + ", tail=" + tail2 + "\n");
37 :             else Console.WriteLine(" 2: -\n");
38 :             return true;
39 :         }
40 :     }
41 : }

```

表示出力 (使用例 1 から、*.txt と *.doc が除外される)

```

1: path=d:\work\path1\Child\c1.bmp, name=c1.bmp, tail=Child\c1.bmp
2: -

-----

1: -
2: path=d:\temp\path2\Child\c2.bmp, name=c2.bmp, tail=Child\c2.bmp

-----

1: path=d:\work\path1\Child\cgrounddsun\gs.pdf, name=gs.pdf, tail=Child\cgrounddsun\gs.pdf
2: path=d:\temp\path2\Child\cgrounddsun\gs.pdf, name=gs.pdf, tail=Child\cgrounddsun\gs.pdf
Result = 0

```

.Child\c1.bmp は、パス 1 側だけに存在する

.Child\c2.bmp は、パス 2 側だけに存在する

.Child\cgrounddsun\gs.pdf は、双方のパスに存在する

22.1.6. パスがワイルドカード[群]に一致するかチェック (PathMatchSpecs)

形 式 : bool PathMatchSpecs (string path, string specs);

引 数 : path - チェックするパス名
 specs - ワイルドカード (複数指定する場合はセミコロン(;))か スラッシュ(/)で区切る)

説 明 : 「path」で指定したパスが「specs」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。
 複数のワイルドカードを指定する場合は、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されます。

戻り値 : true - パスは (いずれかの) ワイルドカードに一致する
 false - パスは (いずれの) ワイルドカードにも一致しない

22.1.7. パスが文字列[群]を含むかチェックする (PathMatchStrings)

形 式 : bool PathMatchStrings (string path, string strings);

引 数 : path - チェックするパス名
 specs - 文字列 (複数指定する場合はセミコロン(;))か スラッシュ(/)で区切る)

説 明 : 「path」で指定したパスが「strings」で指定した文字列を含むかチェックします。
 複数の文字列を指定する場合は、セミコロン(;))か スラッシュ(/)で区切って指定します。(ex. @"¥Debug¥ ; ¥Release¥")
 各文字列の前後の空白は無視されます。

戻り値 : true - パスは (いずれかの) 文字列を含む
 false - パスは (いずれの) 文字列も含まない

22.1.8. パスが存在するかチェック(IsExistsPath)

形 式 : bool IsExistsPath (string path);

引 数 : path - チェックするパス名

説 明 : 「path」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

22.1.9. パスがディレクトリかチェック(IsPathDirectory)

形 式 : bool IsPathDirectory (string path);

引 数 : path - チェックするパス名

説 明 : 「path」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 (あるいは、パスが存在しない)

22.1.10. パスがファイルかチェック(IsPathFile)

形 式 : `bool IsPathFile (string path);`

引 数 : `path` - チェックするパス名

説 明 : 「path」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)

22.1.11. ファイルサイズ取得(AjcGetFileSize)

形 式 : `long GetFileSize (string path);`

引 数 : `path` - ファイルパス名

説 明 : 「path」で指定したファイルのサイズを取得します。

戻り値 : `≠-1` - ファイルサイズ
`=-1` - エラー

22.1.12. ファイルタイム取得(AjcGetFileTime1970)

形 式 : `uint GetFileTime1970 (string path);`

引 数 : `path` - ファイルパス名

説 明 : 「path」で指定したファイルの最終更新タイム (UTC 時刻) を取得します。

戻り値 : `≠0xFFFFFFFF` - ファイルタイム (1970/01/01 00:00:00 からの通算秒数, 最大 `0xFFFFFFFFE` = 2106/02/07 06:28:14)
`=0xFFFFFFFF` - エラー

22.1.13. ファイル比較(FileCompare)

形 式 : `bool FileCompare (string path1, string path2);`

引 数 : `path1, path2` - 比較する2つのファイルパス名

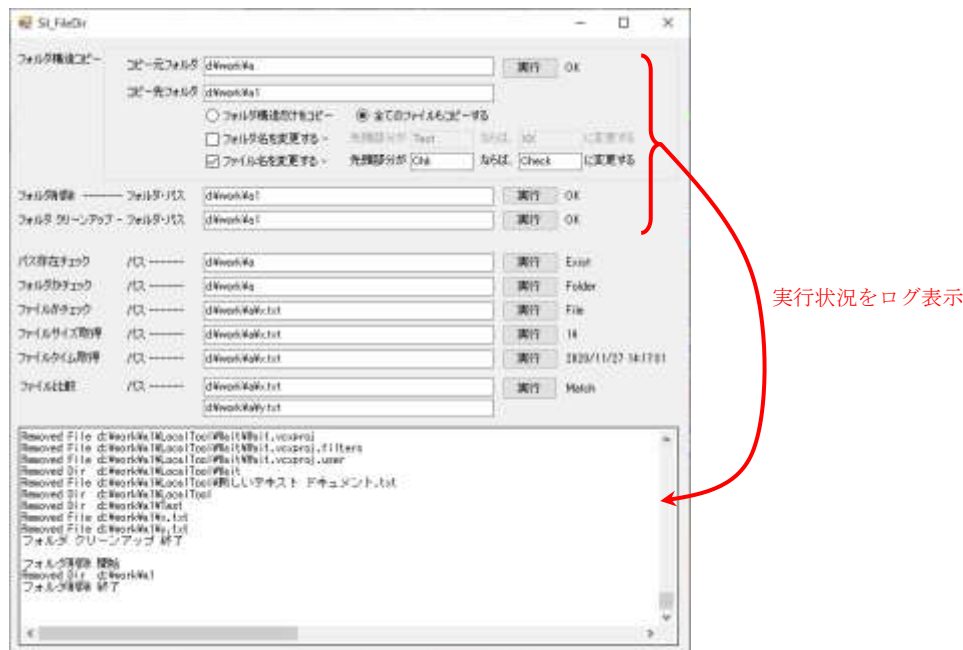
説 明 : 2つのファイルをバイナリ比較します。

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

22.2. サンプルプログラム

22.2.1. Sil_FileDir (ファイル/フォルダ操作)

このサンプルプログラムでは、ファイル/フォルダ操作ファンクションを実行します。



```

1 : using System;
2 : using System.Collections.Generic;
3 : using System.ComponentModel;
4 : using System.Data;
5 : using System.Drawing;
6 : using System.Linq;
7 : using System.Text;
8 : using System.Windows.Forms;
9 : using CAjrCustCtrl;
10 :
11 : namespace Sil_FileDir
12 : {
13 :     public partial class Form1 : Form
14 :     {
15 :
16 :         FopCbkSepNtc m_CbkSepNtc;   FopCbkFcpNtc m_CbkFcpNtc;
17 :         FopCbkSepQry m_CbkSepQry;   FopCbkFcpQry m_CbkFcpQry;
18 :         FopCbkRmvDir m_CbkRmvDir;
19 :
20 :         public Form1()
21 :         {
22 :             InitializeComponent();
23 :         }
24 :         // フォーム開始
25 :         private void Form1_Load(object sender, EventArgs e)
26 :         {
27 :             // テキストボックスへのファイル/ディレクトリ ドロップ許可
28 :             SAjrGsr.EnableDropToTextBox(this);
29 :
30 :             // 設定値ロード
31 :             SAjrReg.LoadAllCtrls(this);
32 :             // コールバック設定
33 :             m_CbkSepNtc = new FopCbkSepNtc(cbSepNtc );           // フォルダコピー通知
34 :             m_CbkSepQry = new FopCbkSepQry(cbSepQry );          // フォルダコピー/名称変更問い合わせ
35 :             m_CbkFcpNtc = new FopCbkFcpNtc(cbFcpNtc );          // ファイルコピー通知
36 :             m_CbkFcpQry = new FopCbkFcpQry(cbFcpQry );          // ファイルコピー/名称変更問い合わせ
37 :             m_CbkRmvDir = new FopCbkRmvDir(cbRmvPath);           // フォルダ/ファイル削除通知
38 :             // ツールチップ
39 :             SAjrTip.Add(cmdCpy , "指定フォルダ下のディレクトリ構造をコピーします。¥n" +

```

```

40 :                                     "「すべてのファイルもコピーする」を選択すると、ファイルもコピーします。");
41 :     SAjrTip.Add(cmdRmv , "指定ディレクトリとその下の全フォルダ、ファイルを削除します。");
42 :     SAjrTip.Add(cmdCln , "指定フォルダ下の全ディレクトリ、ファイルを削除します。");
43 :     SAjrTip.Add(cmdIsPath, "指定したパスが存在するかチェックします。");
44 :     SAjrTip.Add(cmdIsDir , "指定したパスがディレクトリかチェックします。");
45 :     SAjrTip.Add(cmdIsFile, "指定したパスがファイルかチェックします。");
46 :     SAjrTip.Add(cmdFSize , "指定したファイルのサイズを表示します。");
47 :     SAjrTip.Add(cmdFTime , "指定したファイルの更新日時を表示します。");
48 :     SAjrTip.Add(cmdFCmp , "指定した2つのファイルを比較します。");
49 :
50 :     // 初期表示
51 :     ShowCopy();
52 : }
53 : // フォーム終了
54 : private void Form1_FormClosed(object sender, FormClosedEventArgs e)
55 : {
56 :     // 設定値セーブ
57 :     SAjrReg.SaveAllCtrls(this);
58 : }
59 : // コールバック (フォルダ作成可否/フォルダ名変更の問い合わせ)
60 : private string cbScpQry(string DirFront, string DirNew, IntPtr cbp)
61 : {
62 :     string s = DirNew;
63 :     int len = txtDirOld.Text.Length;
64 :     if (chkDirRen.Checked && len != 0) {
65 :         if (DirNew.Length >= len && DirNew.Substring(0, len) == txtDirOld.Text) {
66 :             s = txtDirNew.Text + DirNew.Substring(len);
67 :         }
68 :     }
69 :     return s;
70 : }
71 : // コールバック (フォルダ構造コピー通知)
72 : private bool cbScpNtc(string DirFrom, string DirTo, ENtcDirCopy ntc, IntPtr cbp)
73 : {
74 :     // コピー結果ログ表示
75 :     vth.PutText("Copy From " + DirFrom + "¥n" +
76 :         " to " + DirTo + " --- " + ntc.ToString() + "¥n");
77 :     if (rbtAllFiles.Checked) {
78 :         // このフォルダ下のファイルコピー
79 :         SAjrFop.CopyFiles(DirFrom, DirTo, "*.*", ECpyfOpt.CREATEALWAYS, m_CbkFcpNtc, m_CbkFcpQry);
80 :     }
81 :     Application.DoEvents();
82 :     return true;
83 : }
84 : // コールバック (ファイルコピー可否/ファイル名変更の問い合わせ)
85 : private string cbFcpQry(string FileFrom, string FileTo, string FileName, ref EFileAtt att, IntPtr cbp)
86 : {
87 :     string s = FileName;
88 :     int len = txtFileOld.Text.Length;
89 :     if (chkFileRen.Checked && len != 0) {
90 :         if (FileName.Length >= len && FileName.Substring(0, len) == txtFileOld.Text) {
91 :             s = txtFileNew.Text + FileName.Substring(len);
92 :         }
93 :     }
94 :     return s;
95 : }
96 : // コールバック (ファイルコピー通知)
97 : private bool cbFcpNtc(string FileFrom, string FileTo, ENtcFileCopy ntc, IntPtr cbp)
98 : {
99 :     vth.PutText("File Copy " + FileFrom + " To " + FileTo + " --- " + ntc.ToString() + "¥n");
100 :     Application.DoEvents();
101 :     return true;
102 : }
103 : // コールバック (フォルダ/ファイル削除通知)
104 : private bool cbRmvPath(string PathName, uint ntc, IntPtr cbp)
105 : {
106 :     if ((ntc & 0x01) != 0) vth.PutText("Removed Dir " + PathName + "¥n");
107 :     else vth.PutText("Removed File " + PathName + "¥n");
108 :     Application.DoEvents();
109 :     return true;
110 : }
111 : // フォルダ構造コピー ボタン

```

```

112 : private void cmdCpy_Click(object sender, EventArgs e)
113 : {
114 :     bool    rsu;
115 :     vth.PutText("フォルダ構造コピー 開始\n");
116 :     lblCpy.Text = "";
117 :     if (rsu = SAjrFop.CopyFolderStruct(txtCpyFrom.Text, txtCpyTo.Text, m_CbkScpNtc, m_CbkScpQry)) lblCpy.Text = "OK";
118 :     else
119 :         // 最上位フォルダ下のファイルコピー
120 :         if (rbtAllFiles.Checked && rsu) {
121 :             SAjrFop.CopyFiles(txtCpyFrom.Text, txtCpyTo.Text, " *.*", ECpyfOpt.CREATEALWAYS, m_CbkFcpNtc, m_CbkFcpQry);
122 :         }
123 :     vth.PutText("フォルダ構造コピー 終了\n");
124 : }
125 : // フォルダ構造だけをコピー ラジオボタン
126 : private void rbtStruct_Click(object sender, EventArgs e) {ShowCopy();}
127 : // 全てのファイルもコピーする ラジオボタン
128 : private void rbtAllFiles_Click(object sender, EventArgs e) {ShowCopy();}
129 : // フォルダ名を変更する チェックボックス
130 : private void chkDirRen_Click(object sender, EventArgs e) {ShowCopy();}
131 : // ファイル名を変更する チェックボックス
132 : private void chkFplrRen_Click(object sender, EventArgs e) {ShowCopy();}
133 : // フォルダ構造コピー部分の表示設定
134 : private void ShowCopy()
135 : {
136 :     lblDirCpy1.Enabled = lblDirCpy2.Enabled = lblDirCpy3.Enabled = txtDirOld.Enabled = txtDirNew.Enabled = chkDirRen.Checked;
137 :     if (rbtAllFiles.Checked) {
138 :         chkFileRen.Enabled = true;
139 :         lblFileCpy1.Enabled = lblFileCpy2.Enabled = lblFileCpy3.Enabled =
140 :             txtFileOld.Enabled = txtFileNew.Enabled = chkFileRen.Checked;
141 :     }
142 :     else {
143 :         chkFileRen.Enabled = false;
144 :         lblFileCpy1.Enabled = lblFileCpy2.Enabled = lblFileCpy3.Enabled = txtFileOld.Enabled = txtFileNew.Enabled = false;
145 :     }
146 : }
147 : // フォルダ削除 ボタン
148 : private void cmdRmv_Click(object sender, EventArgs e)
149 : {
150 :     vth.PutText("フォルダ削除 開始\n");
151 :     lblRmv.Text = "";
152 :     if (SAjrFop.RemoveFolder(txtRmv.Text, m_CbkRmvDir)) lblRmv.Text = "OK";
153 :     else
154 :         vth.PutText("フォルダ削除 終了\n");
155 : }
156 : // フォルダ クリーンアップ ボタン
157 : private void cmdCln_Click(object sender, EventArgs e)
158 : {
159 :     vth.PutText("フォルダ クリーンアップ 開始\n");
160 :     lblCln.Text = "";
161 :     if (SAjrFop.CleanFolder(txtCln.Text, m_CbkRmvDir)) lblCln.Text = "OK";
162 :     else
163 :         vth.PutText("フォルダ クリーンアップ 終了\n");
164 : }
165 : // パスの存在チェック ボタン
166 : private void cmdIsPath_Click(object sender, EventArgs e)
167 : {
168 :     if (SAjrFop.IsExistsPath(txtIsPath.Text)) lblIsPath.Text = "Exist";
169 :     else
170 :         lblIsPath.Text = "Not Exist";
171 : }
172 : // フォルダかのチェック ボタン
173 : private void cmdIsDir_Click(object sender, EventArgs e)
174 : {
175 :     if (SAjrFop.IsPathDirectory(txtIsDir.Text)) lblIsDir.Text = "Folder";
176 :     else
177 :         lblIsDir.Text = "Not Folder";
178 : }
179 : // ファイルかのチェック ボタン
180 : private void cmdIsFile_Click(object sender, EventArgs e)
181 : {
182 :     if (SAjrFop.IsPathFile(txtIsFile.Text)) lblIsFile.Text = "File";
183 :     else
184 :         lblIsFile.Text = "Not File";
185 : }
186 : // ファイルサイズ取得 ボタン

```



```

184 :     private void cmdFSize_Click(object sender, EventArgs e)
185 :     {
186 :         long sz = SAjrFop.GetFileSize(txtFSize.Text);
187 :         if (sz != -1) lblFSize.Text = sz.ToString();
188 :         else         lblFSize.Text = "Error";
189 :     }
190 :     // ファイルタイム取得 ボタン
191 :     private void cmdFTime_Click(object sender, EventArgs e)
192 :     {
193 :         uint tm = SAjrFop.GetFileTime1970(txtFTime.Text);
194 :         if (tm != 0xFFFFFFFF) {
195 :             DateTime dt = SAjrGsr.Time1970ToDateAndTime(tm);
196 :             dt = SAjrGsr.UtcTimeToLocalTime(dt);
197 :             lblFTime.Text = dt.ToString();
198 :         }
199 :         else lblFTime.Text = "Error";
200 :     }
201 :     // ファイル比較ボタン
202 :     private void cmdFCmp_Click(object sender, EventArgs e)
203 :     {
204 :         bool rsu = SAjrFop.FileCompare(txtFCmp1.Text, txtFCmp2.Text);
205 :         if (rsu) lblFCmp.Text = "Match";
206 :         else     lblFCmp.Text = "Unmatch";
207 :     }
208 : }
209 : }

```

23. スタティク : 3D / 2Dベクトル等の演算 (CAjrStatic.dll, SAjrMath クラス)

3Dベクトル演算等の算術演算スタティックメソッド群です。クラス名は「SAjrMath」です。

23.1. メソッド

算術演算のメソッド一覧を以下に示します。

#	メソッド名	内 容	備考
1	Sin	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	Sinh	双曲線・正弦 (ハイパーボリック・サイン)	
3	ASin	逆正弦 (アークサイン)	
4	Cos	余弦 (コサイン)	
5	Cosh	双曲線・余弦 (ハイパーボリック・コサイン)	
6	ACos	逆余弦 (アークコサイン)	
7	Tan	正接 (タンジェント)	
8	Tanh	双曲線・正接 (ハイパーボリック・タンジェント)	
9	ATan	逆正接 (アークタンジェント)	
10	ATan2	逆正接 (アークタンジェント)	
11	V3dAdd	3D・ベクトル加算	3Dベクトル演算
12	V3dSub	3D・ベクトル減算	
13	V3dMult	3D・ベクトル乗算 (ベクトルをn倍する)	
14	V3dDiv	3D・ベクトルの除算	
15	V3dSetLineVec	3D・線ベクトル設定	
16	V3dSetLinePoint	3D・線分情報設定	
17	V3dSetTriPoint	3D・三角形情報設定	
18	V3dLength	3D・ベクトルの長さ算出	
19	V3dLineCenter	3D・線分の中点算出	
20	V3dOuter	3D・ベクトルの外積算出	
21	V3dInner	3D・ベクトルの内積算出	
22	V3dPlaneVec	3D・三角形の法線ベクトル算出	
23	V3dNormal	3D・単位ベクトル算出	
24	V3dTheta	3D・ベクトルの開き角度算出	
25	V3dVertVecP2L	3D・ポイントから直線へ直行する方向ベクトル算出	
26	V3dDistP2L	3D・ポイントから直線へ直行する方向ベクトルと距離算出	
27	V3dDistP2P	3D・2つの3Dポイント間の距離算出	
28	V3dCrossL2L	3D・ラインの交点算出	
29	V3dCrossP2F	3D・点と平面の交点算出	
30	V3dOrthoVecOnPlane	3D・同一平面上で線分に直行するベクトル算出	
31	V3dMultMat	3D・ベクトルと行列の掛け算	
32	V3dRotateX	3D・ベクトルをX軸周りに回転	
33	V3dRotateY	3D・ベクトルをY軸周りに回転	
34	V3dRotateZ	3D・ベクトルをZ軸周りに回転	
35	V3dRotateAny	3D・ベクトルを任意の軸周りに回転	
36	V3dAnyOrthoVec	3D・任意の直行ベクトル算出	
37	V3dRotateOnPlane	3D・任意の点を平面上で指定角度回転した点を求める	
38	V3dCalcCircle	3D・任意の3点を通る円の中心と半径を算出	
39	V3dCalcSphere	3D・球面上の任意の4点から球の中心と半径を算出	
40	V2dAdd	2D・ベクトル加算	2Dベクトル演算
41	V2dSub	2D・ベクトル減算	
42	V2dMult	2D・ベクトル乗算 (ベクトルをn倍する)	
43	V2dDiv	2D・ベクトルの除算	
44	V2dLength	2D・ベクトルの長さ算出	
45	V2dOuter	2D・ベクトルの外積算出	
46	V2dInner	2D・ベクトルの内積算出	
47	V2dNormal	2D・単位ベクトル算出	
48	V2dTheta	2D・ベクトルの開き角度算出	
49	V2dVertVecP2L	2D・ポイントから直線へ直行する方向ベクトル算出	
50	V2dDistP2L	2D・ポイントから直線へ直行する方向ベクトルと距離算出	
51	V2dDistP2P	2D・2つの2Dポイント間の距離算出	
52	V2dCrossL2L	2D・ラインの交点算出	
53	V2dRotate	2D・ベクトルをX軸周りに回転	
54	V2dAnyOrthoVec	2D・任意の直行ベクトル算出	

23.1.1. 正弦 (Sin)

形 式 : `static double Sin(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)

23.1.2. 双曲線・正弦 (Sinh)

形 式 : `static double Sinh(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)

23.1.3. 逆正弦 (ASin)

形 式 : `static double ASin(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]

23.1.4. 余弦 (Cos)

形 式 : `static double Cos(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)

23.1.5. 双曲線・余弦(Cosh)

形 式 : `static double Cosh(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)

23.1.6. 逆余弦(ACos)

形 式 : `static double ACos(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]

23.1.7. 正接 (Tan)

形 式 : `static double Tan(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)

23.1.8. 双曲線・正接(Tanh)

形 式 : `static double Tanh(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)

23.1.9. 逆正接(ATan)

形 式 : `static double ATan(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]

23.1.10. 逆正接(ATan2)

形 式 : `static double ATan2(double x, double y)`

引 数 : `x, y` - 逆正接値を算出する値

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]

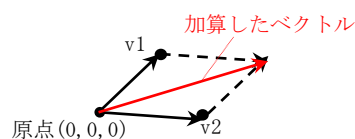
23.1.11. 3Dベクトル加算(V3dAdd)

形 式 : `static AJC3DVEC V3dAdd(AJC3DVEC v1, AJC3DVEC v2);`

引 数 : `v1, v2` - 加算する2つのベクトル

説 明 : ベクトルの加算値を行います。

戻り値 : 加算したベクトル

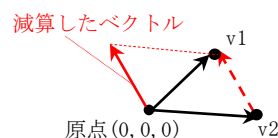
**23.1.12. 3Dベクトル減算(V3dSub)**

形 式 : `static AJC3DVEC V3dSub(AJC3DVEC v1, AJC3DVEC v2);`

引 数 : `p1` - 減算されるベクトル
`p2` - 減算するベクトル

説 明 : ベクトルの減算 ($v1 - v2$) を行います。
 減算したベクトルは、 $v2 \rightarrow v1$ へ方向ベクトルとなります。

戻り値 : 減算した方向ベクトル ($v2 \rightarrow v1$ ベクトル)

**23.1.13. 3Dベクトル乗算 (V3dMult)**

形 式 : `static AJC3DVEC V3dMult(AJC3DVEC v, double n);`

引 数 : `v` - 非乗算ベクトル
`n` - 乗算値

説 明 : ベクトルの乗算 (各ベクトル要素に n を乗算) を行います。

戻り値 : 乗算したベクトル

23.1.14. 3Dベクトルの除算(V3dDiv)

形 式 : `static AJC3DVEC V3dDiv(AJC3DVEC v, double n);`

引 数 : `v` - 非除算ベクトル
`n` - 除算値

説 明 : ベクトルの除算 (各ベクトル要素を n で除算) を行います。

戻り値 : 除算したベクトル

23.1.15. 3D・線ベクトル設定(V3dSetLineVec)

形 式 : `static AJC3DVEC AjcV3dSetLineVec (ref AJC3DVEC p1, ref AJC3DVEC p2);`

引 数 : `p1, p2` - 線分を示す2つの点 (始点, 終点)
`v1` - 線ベクトルを格納するバッファ

説 明 : 始点(`p1`)と方向ベクトル($p1 \rightarrow p2$)を設定した線ベクトルを返します。

戻り値 : 線ベクトル (始点と方向ベクトル)

23.1.16. 3D・線分情報設定(AjcV3dSetLinePoint)

形 式 : static PAJC3DLINE V3dSetLinePoint (PCAJC3DVEC p1, PCAJC3DVEC p2);

引 数 : p1, p2 - 線分を示す2つの点 (始点, 終点)

説 明 : 始点(p1)と、終点(p2)を設定した線分情報を返します。

戻り値 : 線分情報

23.1.17. 3D・三角形情報設定(AjcV3dSetTriPoint)

形 式 : static AJC3DTRI AjcV3dSetTriPoint (AJC3DVEC p1, AJC3DVEC p2, AJC3DVEC p3);

引 数 : p1, p2, p3 - 三角形を示す3つの点 (3つの頂点)

説 明 : 三角形情報 (3つの点) を設定した三角形情報を返します。

戻り値 : 三角形情報

23.1.18. 3D・ベクトルの長さ算出(V3dLength)

形 式 : static double V3dLength(AJC3DVEC v);

引 数 : v - 長さを求めるベクトル

説 明 : ベクトルの長さを算出します。

戻り値 : ベクトルの長さ

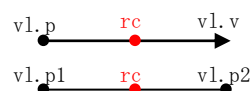
23.1.19. 3D・線分の中点算出(V3dLVecCenter)

形 式 : static AJC3DVEC V3dLineCenter (AJC3DLVEC vl);
static AJC3DVEC V3dLineCenter (AJC3DLINE vl);

引 数 : vl - 中点を求める線ベクトル／線分情報 (始点と方向ベクトル／始点と終点)

説 明 : 線ベクトル／線分の中点を算出します。

戻り値 : 線ベクトル／線分情報の中点

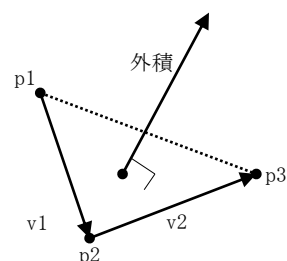
**23.1.20. 3D・ベクトルの外積算出(V3dOuter)**

形 式 : static AJC3DVEC V3dOuter(AJC3DVEC v1, AJC3DVEC v2);
static AJC3DVEC V3dOuter(AJC3DVEC p1, AJC3DVEC p2, AJC3DVEC p3);

引 数 : v1, v2 - 外積を求める2つのベクトル
p1, p2, p3 - 外積を求める3つの点

説 明 : 2つのベクトル／3つの点からの外積を算出します。
3つの点を指定した場合は、p1→p2 と p2→p3 の2つのベクトルから外積を算出します。
つまり、2つのベクトルで表される平面に垂直なベクトル (法線) を求めます。

戻り値 : 2つのベクトルの外積値 (法線ベクトル)



23.1.21. 3D・ベクトルの内積算出(V3dInner)

形 式 : `static double V3dInner(AJC3DVEC v1, AJC3DVEC v2);`
`static double V3dInner(AJC3DVEC p1, AJC3DVEC p2, AJC3DVEC p3)`

引 数 : `v1, v2` - 内積を求める2つのベクトル
`p1, p2, p3` - 内積を求める3つの点

説 明 : 2つのベクトルの内積を算出します。
 3つの点を指定した場合は、`p1→p2` と `p2→p3` の2つのベクトルから内積を算出します。

戻り値 : 2つのベクトルの内積値

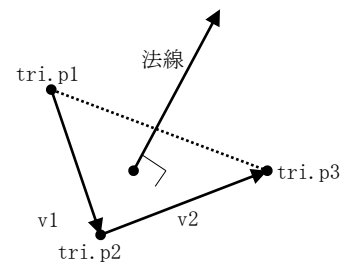
23.1.22. 3D・三角形の法線ベクトル算出(V3dPlaneVec)

形 式 : `static AJC3DVEC V3dPlaneVec (AJC3DTRI tri);`

引 数 : `tri` - 法線を算出する三角形情報 (3つの頂点)

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 : 法線ベクトル

**23.1.23. 3D・単位ベクトル算出(V3dNormal)**

形 式 : `static AJC3DVEC V3dNormal (AJC3DVEC v);`

引 数 : `v` - 単位ベクトルを求めるベクトル

説 明 : 単位ベクトル (ベクトル長=1のベクトル) を算出します。

戻り値 : 単位ベクトル

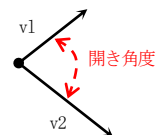
23.1.24. 3D・ベクトルの開き角度算出(V3dTheta)

形 式 : `static double V3dTheta(AJC3DVEC v1, AJC3DVEC v2);`

引 数 : `v1, v2` - 角度を求める2つのベクトル

説 明 : 2つのベクトルの開き角度を算出します。

戻り値 : 2つのベクトルの開き角度 [度]

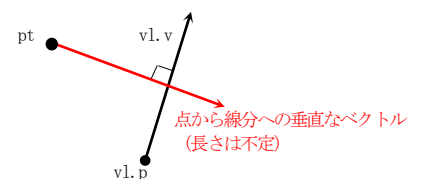
**23.1.25. 3D・ポイントから直線へ直行する方向ベクトル算出(V3dVertVecP2L)**

形 式 : `static AJC3DVEC V3dVertVecP2L (AJC3DVEC v1, AJC3DVEC pt);`

引 数 : `v1` - 線ベクトル (線分の始点と方向ベクトル)
`pt` - 点の位置

説 明 : `pt` で示す点から、`v1` で示す線ベクトルへの垂直なベクトルを算出します。
 算出したベクトル(`vr`)の長さは不定です。

戻り値 : 点から線分への垂直なベクトル



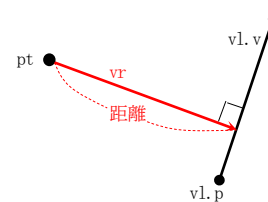
23.1.26. 3D・ポイントから直線までの方向ベクトルと距離算出(V3dDistP2L)

形式 : static double V3dDistP2L(AJC3DVEC v1, AJC3DVEC pt, out AJC3DVEC vr);

引数 : v1 - 直線の始点位置と方向ベクトル
 pt - ポイント
 vr - ポイントから直線へ垂直なベクトルを格納する変数

説明 : ポイントから直線へ垂直なベクトルと、ポイントから直線までの距離を算出します。

戻り値 : ポイントから直線までの距離

**23.1.27. 3D・2つの3Dポイント間の距離算出(V3dDistP2P)**

形式 : static double V3dDistP2P(AJC3DVEC p1, AJC3DVEC p2);

引数 : p1, p2 - 距離を求める2つのポイント

説明 : 2つのポイント間の距離を算出します。

戻り値 : 2つのポイント間の距離

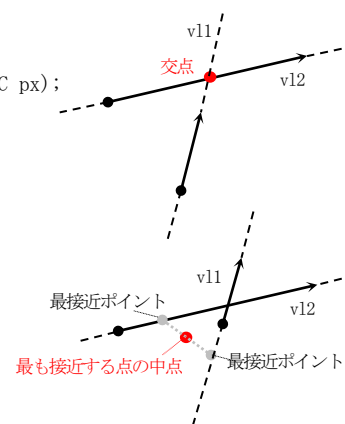
23.1.28. 3D・ラインの交点算出(V3dCrossL2L)

形式 : static AJC3DVEC V3dCrossL2L(AJC3DVEC v11, AJC3DVEC v12, out AJC3DVEC px);

引数 : v11, v12 - 交点を求める2つの直線 (始点と方向ベクトル)

説明 : 2つの直線の交点を算出します。
 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 : 2つの直線の交点

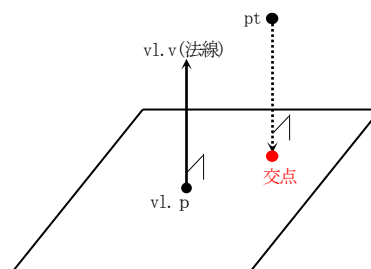
**23.1.29. 3D・点と平面の交点算出(V3dCrossP2F)**

形式 : static AJC3DVEC V3dCrossP2F(AJC3DVEC v1, AJC3DVEC pt);

引数 : v1 - 平面を表す情報 (平面の位置と法線)
 pt - ポイント

説明 : ポイントから平面に垂直に交わる交点を算出します。

戻り値 : 点と平面の交点



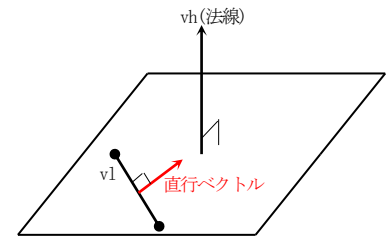
23.1.30. 3D・同一平面上で線分に直行するベクトル算出(V3dOrthoVecOnPlane)

形 式 : `static AJC3DVEC AjcV3dOrthoVecOnPlane (AJC3DLINE v1, AJC3DVEC vh);`

引 数 : `v1` - 平面上の線分情報のアドレス (平面上の始点と終点)
`vh` - 平面からの法線ベクトル

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 : 同一平面上で線分に直行するベクトル

**23.1.31. 3D・ベクトルと行列の掛け算(V3dMultMat)**

形 式 : `static AJC3DVEC AjcV3dMultMat (AJC3DVEC v, AJC3DMAT m);`

引 数 : `v` - 被乗数ベクトル
`m` - 乗数 (行列)

説 明 : ベクトルへ行列を乗算します。

戻り値 : 行列を乗算したベクトル

23.1.32. 3D・ベクトルをX軸周りに回転(V3dRotateX)

形 式 : `static AJC3DVEC V3dRotateX(AJC3DVEC pt, double t);`

引 数 : `pt` - X軸周りに回転するポイント
`t` - 回転角度 [度]

説 明 : X軸周りに回転 (X軸方向に対して右回り) したポイントを算出します。

戻り値 : X軸周りに回転したポイント

23.1.33. 3D・ベクトルをY軸周りに回転(V3dRotateY)

形 式 : `static AJC3DVEC V3dRotateY(AJC3DVEC pt, double t);`

引 数 : `pt` - Y軸周りに回転するポイント
`t` - 回転角度 [度]

説 明 : Y軸周りに回転 (Y軸方向に対して右回り) したポイントを算出します。

戻り値 : Y軸周りに回転したポイント

23.1.34. 3D・ベクトルをZ軸周りに回転(V3dRotateZ)

形 式 : `static AJC3DVEC V3dRotateZ(AJC3DVEC pt, double t);`

引 数 : `pt` - Z軸周りに回転するポイント
`t` - 回転角度 [度]

説 明 : Z軸周りに回転 (Z軸方向に対して右回り) したポイントを算出します。

戻り値 : Z軸周りに回転したポイント

23.1.35. 3D・ベクトルを任意の軸周りに回転(V3dRotateAny)

形式 : `static AJC3DVEC V3dRotateAny(AJC3DVEC pt, double t, AJC3DVEC vs);`

引数 : `pt` - 任意の軸周りに回転するポイント
`t` - 回転角度 [度]
`vs` - 回転軸ベクトル (原点 (0, 0, 0) を通るベクトル)

説明 : 任意の軸周りに回転 (回転軸ベクトル方向に対して右回り) したポイントを算出します。

戻り値 : 任意の軸周りに回転したポイント

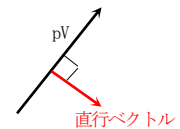
23.1.36. 3D・任意の直行ベクトル算出(V3dAnyOrthoVec)

形式 : `static AJC3DVEC V3dAnyOrthoVec(AJC3DVEC v);`

引数 : `v` - ベクトル

説明 : 指定されたベクトルに直行する、任意のベクトルを算出します。
 ベクトル (`v`) 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 : ベクトルに直行するベクトル

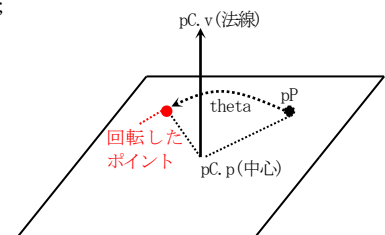
**23.1.37. 3D・任意の点を平面上で指定角度回転した点を求める(V3dRotateOnPlane)**

形式 : `static AJC3DVEC V3dRotateOnPlane(AJC3DVEC pt, AJC3DLVEC vl, double t);`

引数 : `pt` - 平面上のポイント
`vl` - 平面を表す情報 (平面の中心位置と法線)
`t` - 回転角度 [度]

説明 : 3D空間上の平面上で回転したポイントを算出します。

戻り値 : 回転後のポイント



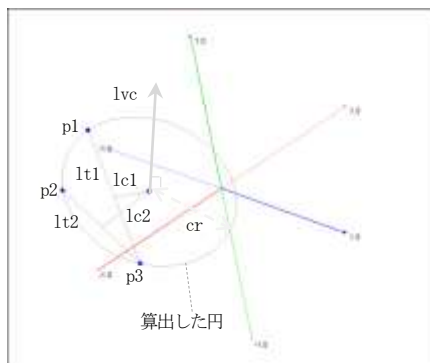
23.1.38. 3D・任意の3点を通る円の中心と半径を算出(V3dCalcCircle)

形 式 : static double V3dCalcCircle(AJC3DVEC p1, AJC3DVEC p2, AJC3DVEC p3, out AJC3DVEC vc);
 static double V3dCalcCircle(AJC3DVEC p1, AJC3DVEC p2, AJC3DVEC p3, out AJC3DVEC vc, out AJC3DVEC vh);
 static double V3dCalcCircle(AJC3DVEC p1, AJC3DVEC p2, AJC3DVEC p3, out AJC3DVEC vc, out AJC3DVEC vh, out AJC3DCIRINFO ci);

引 数 : p1~p3 - 3D空間上の3つのポイント
 vc - 円の中心を格納する変数
 vh - 円の法線を格納する変数
 ci - 円の算出情報を格納する変数

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。
 ciは円を算出した際の演算中間情報で、以下の形式です。

```
struct AJC3DCIRINFO
{
    public AJC3DLINE    lt1, lt2;           // 円に内接する2つの直線
    public AJC3DLINE    lc1, lc2;           // 内接線の中点からの垂線
    public AJC3DLVEC    lvc;                // 円の中心と法線
    public double        cr;                // 円の半径
}
```



戻り値 : ≠-1 : 平面円の半径
 =-1 : エラー (3点が直線上にある)

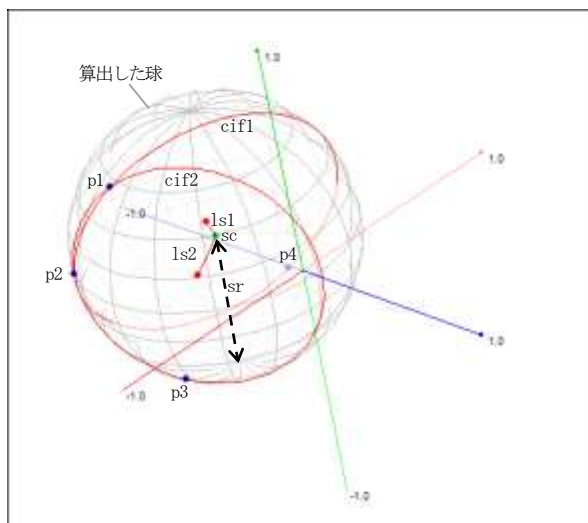
23.1.39. 3D・球面上の任意の4点から球の中心と半径を算出(V3dCalcSphere [V])

形 式 : static double V3dCalcSphere(AJC3DVEC p1, AJC3DVEC p2, AJC3DVEC p3, AJC3DVEC p4, out AJC3DVEC vc);
 static double V3dCalcSphere(AJC3DVEC p1, AJC3DVEC p2, AJC3DVEC p3, AJC3DVEC p4, out AJC3DVEC vc, out AJC3DSPHINFO si);

引 数 : p1~p4 - 球面上の任意の4点
 vc - 球の中心を格納する変数
 si - 球の算出情報を格納する変数

説 明 : 球面上の任意の4点を指定し、球体を算出します。
 siは球を算出した際の演算中間情報で、以下の形式です。

```
struct AJC3DSPHINFO {
    public AJC3DCIRINFO cif1, cif2;           // 球に内接する2つの円
    public AJC3DLINE   ls1, ls2;             // 内接円中心から球中心への直線
    public AJC3DVEC     sc;                   // 球の中心
    public double       sr;                   // 球の半径
}
```



戻り値 : ≠-1 : 球の半径
 =-1 : エラー (いずれかの3点が直線上にある)

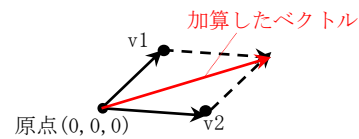
23.1.40. 2Dベクトル加算(V2dAdd)

形 式 : `static AJC2DVEC V2dAdd(AJC2DVEC v1, AJC2DVEC v2);`

引 数 : `v1, v2` - 加算する2つのベクトル

説 明 : ベクトルの加算値を行います。

戻り値 : 加算したベクトル

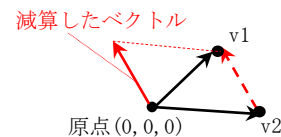
**23.1.41. 2Dベクトル減算(V2dSub)**

形 式 : `static AJC2DVEC V2dSub(AJC2DVEC v1, AJC2DVEC v2);`

引 数 : `p1` - 減算されるベクトル
`p2` - 減算するベクトル

説 明 : ベクトルの減算 ($v1 - v2$) を行います。
 減算したベクトルは、 $v2 \rightarrow v1$ への方方向ベクトルとなります。

戻り値 : 減算した方向ベクトル ($v2 \rightarrow v1$ ベクトル)

**23.1.42. 2Dベクトル乗算 (V2dMult)**

形 式 : `static AJC2DVEC V2dMult(AJC2DVEC v, double n);`

引 数 : `v` - 非乗算ベクトル
`n` - 乗算値

説 明 : ベクトルの乗算 (各ベクトル要素に n を乗算) を行います。

戻り値 : 乗算したベクトル

23.1.43. 2Dベクトルの除算(V2dDiv)

形 式 : `static AJC2DVEC V2dDiv(AJC2DVEC v, double n);`

引 数 : `v` - 非除算ベクトル
`n` - 除算値

説 明 : ベクトルの除算 (各ベクトル要素を n で除算) を行います。

戻り値 : 除算したベクトル

23.1.44. 2D・ベクトルの長さ算出(V2dLength)

形 式 : `static double V2dLength(AJC2DVEC v);`

引 数 : `v` - 長さを求めるベクトル

説 明 : ベクトルの長さを算出します。

戻り値 : ベクトルの長さ

23.1.45. 2D・ベクトルの外積算出(V2dOuter)

形 式 : `static AJC2DVEC V2dOuter(AJC2DVEC v1, AJC2DVEC v2);`

引 数 : `v1, v2` - 外積を求める2つのベクトル

説 明 : 2つのベクトル/3つの点からの外積を算出します。

戻り値 : 2つのベクトルの外積値

23.1.46. 2D・ベクトルの内積算出(V2dInner)

形 式 : `static double V2dInner(AJC2DVEC v1, AJC2DVEC v2);`

引 数 : `v1, v2` - 内積を求める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 : 2つのベクトルの内積値

23.1.47. 2D・単位ベクトル算出(V2dNormal)

形 式 : `static AJC2DVEC V2dNormal(AJC2DVEC v);`

引 数 : `v` - 単位ベクトルを求めるベクトル

説 明 : 単位ベクトル (ベクトル長=1のベクトル) を算出します。

戻り値 : 単位ベクトル

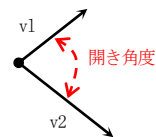
23.1.48. 2D・ベクトルの開き角度算出(V2dTheta)

形 式 : `static double V2dTheta(AJC2DVEC v1, AJC2DVEC v2);`

引 数 : `v1, v2` - 角度を求める2つのベクトル

説 明 : 2つのベクトルの開き角度を算出します。

戻り値 : 2つのベクトルの開き角度 [度]

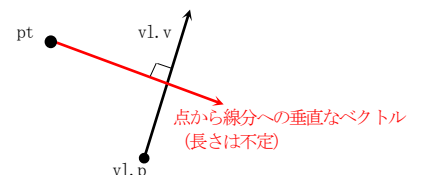
**23.1.49. 2D・ポイントから直線へ直行する方向ベクトル算出(V2dVertVecP2L)**

形 式 : `static AJC2DVEC V2dVertVecP2L (AJC2DVEC v1, AJC2DVEC pt);`

引 数 : `v1` - 線ベクトル (線分の始点と方向ベクトル)
`pt` - 点の位置

説 明 : `pt` で示す点から、`v1` で示す線ベクトルへの垂直なベクトルを算出します。
 算出したベクトル(`vr`)の長さは不定です。

戻り値 : 点から線分への垂直なベクトル



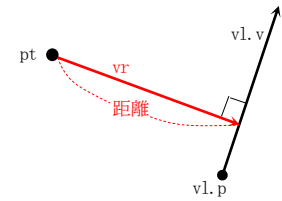
23.1.50. 2D・ポイントから直線までの方向ベクトルと距離算出(V2dDistP2L)

形 式 : `static double V2dDistP2L(AJC2DVEC v1, AJC2DVEC pt, out AJC2DVEC vr);`

引 数 : `v1` - 直線の始点位置と方向ベクトル
`pt` - ポイント
`vr` - ポイントから直線へ垂直なベクトルを格納する変数

説 明 : ポイントから直線へ垂直なベクトルと、ポイントから直線までの距離を算出します。

戻り値 : ポイントから直線までの距離

**23.1.51. 2D・2つの2Dポイント間の距離算出(V2dDistP2P)**

形 式 : `static double V2dDistP2P(AJC2DVEC p1, AJC2DVEC p2);`

引 数 : `p1, p2` - 距離を求める2つのポイント

説 明 : 2つのポイント間の距離を算出します。

戻り値 : 2つのポイント間の距離

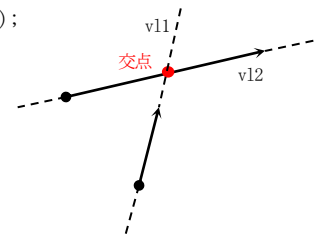
23.1.52. 2D・ラインの交点算出(V2dCrossL2L)

形 式 : `static AJC2DVEC V2dCrossL2L(AJC2DVEC v11, AJC2DVEC v12, out AJC2DVEC px);`

引 数 : `v11, v12` - 交点を求める2つの直線 (始点と方向ベクトル)

説 明 : 2つの直線の交点を算出します。
 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。

戻り値 : 2つの直線の交点

**23.1.53. 2D・ベクトルを回転(V2dRotate)**

形 式 : `static AJC2DVEC V3dRotate(AJC2DVEC pt, double t);`

引 数 : `pt` - X軸回りに回転するポイント
`t` - 回転角度 [度]

説 明 : 指定したポイントを反時計回りに回転したポイントを算出します。
 時計回りに回転する場合は、`t`に負数を指定します。

戻り値 : 回転したポイント

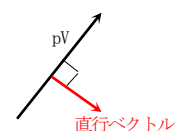
23.1.54. 2D・任意の直行ベクトル算出(V2dAnyOrthoVec)

形 式 : `static AJC2DVEC V2dAnyOrthoVec(AJC2DVEC v);`

引 数 : `v` - ベクトル

説 明 : 指定されたベクトルに直行する、任意のベクトルを算出します。
 ベクトル(`v`)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 : ベクトルに直行するベクトル



24. スタティク : 汎用ファンクション (CAjrStatic.dll, SAjrGsr クラス)

その他の汎用的なスタティックメソッド群です。クラス名は「SAjrGsr」です。

24.1. メソッド

汎用ファンクションのメソッド一覧を示します。

#	メソッド名	内 容	備考
1	GetVersion	バージョン文字列の取得	
2	{Set/Get}Lang	言語種別の設定／取得	
3	LangSel	言語によるテキストの選択	
4	GetSaveFile	保存ファイル名取得	
5	GetOpenFile	オープンファイル名取得	
6	GetOpenFiles	複数のオープンファイル名取得	
7	GetFolder	フォルダ名取得	
8	GetAppPath	起動したプログラムのフォルダ・パス名取得	
9	PathCat	パス名の連結	
10	AfterPrefix	接頭語に続くパラメタ取得	
11	EnableAllControls	フォーム内の全コントロールのイネーブル	
12	ShowGroup	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表示／非表示	
13	EnableGroup	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イネーブル	
14	MoveGroup	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	
15	DifferenceOfTheta	2つの角度の変位量を算出する	
16	ExitWindows	Windows のシャットダウンやリブートを行う	
17	Time1970ToDateAndTime	1970/1/1 00:00:00 からの通算秒を日時に変換	
18	DateAndTimeToTime1970	日時を 1970/1/1 00:00:00 からの通算秒に変換	
19	UtcTimeToLocalTime	UTC 日時をローカルタイムに変換	
20	SepDecimal	10 進数文字列の整数部で規定桁数毎に区切り文字を挿入する	
21	RmvSepChar	10 進数文字列の整数部から区切り文字を削除する	
22	EnableDropToTextBox	テキストボックスにフォルダやファイルをドロップ可能にします	
23	GetWindowPos	デスクトップ上のウインド位置取得	

24.1.1. バージョン文字列取得(GetVersion)

形 式 : `string GetVersion();`

引 数 : なし

説 明 : 本ライブラリのバージョン文字列を取得します。

戻り値 : 本ライブラリのバージョン

24.1.2. 言語種別 設定／取得({Set/Get}Lang)

形 式 : `void SetLang (ELangId lang);` --- 設定
`ELangId GetLang();` ----- 取得

引 数 : `lang` - 言語種別 (Japanese / English)

説 明 : 言語種別を設定／取得します。

本ライブラリで表示する (ポップアップメニュー等の) テキストは、この言語種別に従います。
 (English が設定された場合は、全て英語で、Japanese が設定された場合は全て日本語で表示します)
 言語種別のデフォルトは、Windows のエディションに依存します。

戻り値 : 設定時 : なし 取得時 : 言語種別

24.1.3. 言語によるテキストの選択(LangSel)

形 式 : `string LangSel(string Jpn, string Eng);`

引 数 : `Jpn` - 日本語テキスト
`Eng` - 英語テキスト

説 明 : 言語種別(Language プロパティ)により選択された、日本語テキスト(Jpn)／英語テキスト(Eng)を返します。

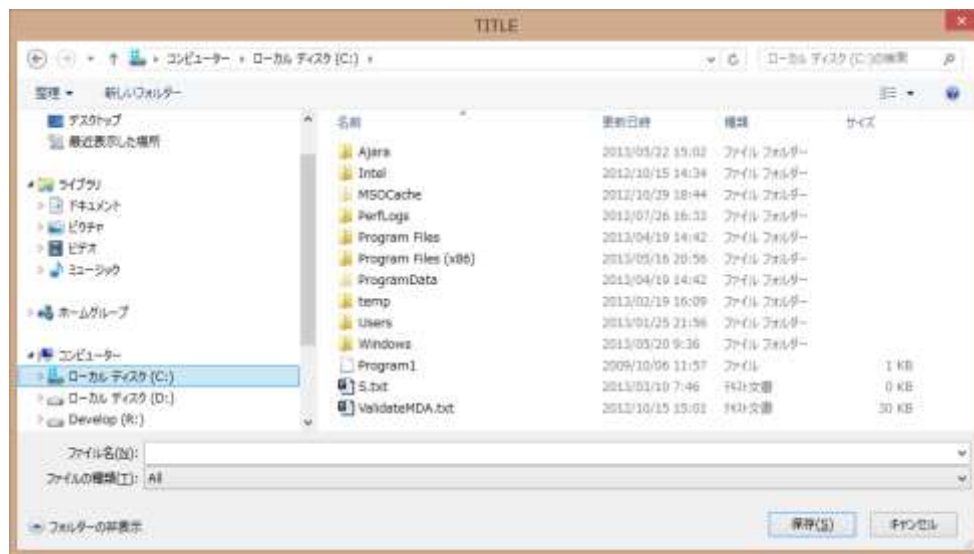
戻り値 : 日本語テキスト／英語テキスト

24.1.4. 保存ファイル名取得 (GetSaveFile)

形 式 : static string GetSaveFile(string title, string filter, string defExt);
 static string GetSaveFile(string initialPath, string title, string filter, string defExt);
 static string GetSaveFile(object owner , string title, string filter, string defExt);
 static string GetSaveFile(object owner , string initialPath, string title, string filter, string defExt);
 static string GetSaveFile(IntPtr hOwner, string title, string filter, string defExt);
 static string GetSaveFile(IntPtr hOwner, string initialPath, string title, string filter, string defExt);

引 数 : owner - オーナーウインドオブジェクト (フォーム等)
 hOwner - オーナーウインドハンドル (フォームの Handle プロパティ等を指定します。)
 initialPath - 初期ファイルパス (指定されたパスが選択されている状態から始める)
 title - 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)
 filter - フィルタ (null を指定した場合は、全てのファイルが対象)
 defExt - 既定の拡張子 (ピリオド (.) は指定不要、既定の拡張子なしの場合は null)

説 明 : 以下のダイアログボックス操作により、保存ファイル・パス名を取得します。



filter 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します。

「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)/*.*")

filter で指定した内容は、ダイアログ中「ファイルの種類」での選択項目となります。

defExt 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。

戻り値 : 空文字列以外 : OK ボタンで終了 (取得したファイルのパス名が返されます)

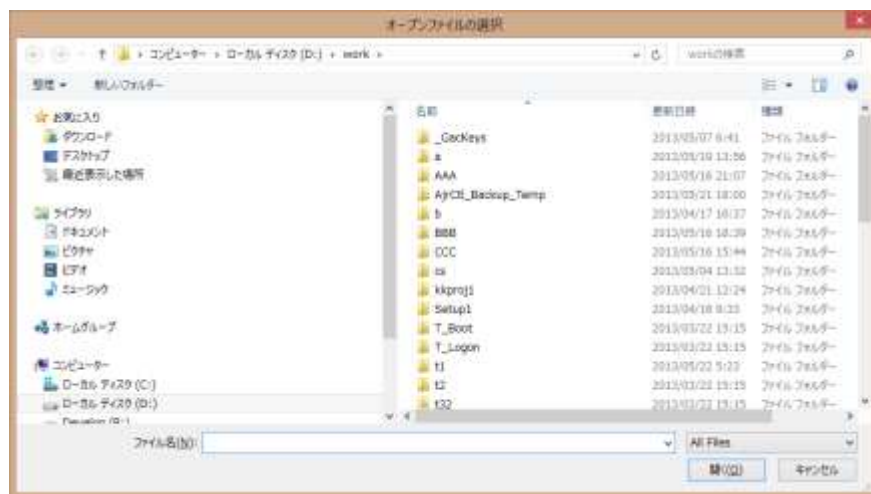
空文字列("") : キャンセルボタンで終了

24.1.5. オープンファイル名取得 (GetOpenFile)

形 式 : static string GetOpenFile(string title, string filter, string defExt);
 static string GetOpenFile(string InitialPath, string title, string filter, string defExt);
 static string GetOpenFile(object owner , string title, string filter, string defExt);
 static string GetOpenFile(object owner , string initialPath, string title, string filter, string defExt);
 static string GetOpenFile(IntPtr hOwner, string title, string filter, string defExt);
 static string GetOpenFile(IntPtr hOwner, string initialPath, string title, string filter, string defExt);

引 数 : owner - オーナーウインドオブジェクト (フォーム等)
 hOwner - オーナーウインドハンドル (フォームの Handle プロパティ等を指定します。)
 title - 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)
 filter - フィルタ (null を指定した場合は、全てのファイルが対象)
 defExt - 既定の拡張子 (ピリオド (.) は指定不要、デフォルト拡張子なしの場合は null)

説 明 : 以下のダイアログボックス操作により、オープンするファイル・パス名を取得します。



filter 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します。

「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)/*.txt")

filter で指定した内容は、ダイアログ中での選択項目となります。

defExt 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。

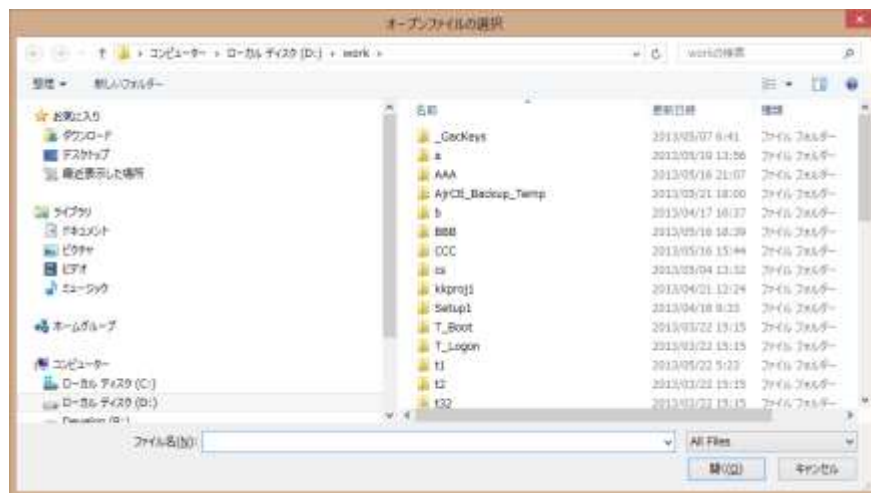
戻り値 : 空文字列以外 : OKボタンで終了 (取得したファイルのパス名が返されます)
 空文字列("") : キャンセルボタンで終了

24.1.6. 複数のオープンファイル名取得 (GetOpenFiles)

形 式 : static string[] GetOpenFiles(string title, string filter, string defExt);
 static string[] GetOpenFiles(string InitialPath, string title, string filter, string defExt);
 static string[] GetOpenFiles(object owner , string initialPath, string title, string filter, string defExt);
 static string[] GetOpenFiles(IntPtr hOwner, string initialPath, string title, string filter, string defExt);

引 数 : owner - オーナーウインドオブジェクト (フォーム等)
 hOwner - オーナーウインドハンドル (フォームの Handle プロパティ等を指定します。)
 initialPath - 初期ファイルパス (指定されたパスが選択されている状態から始める)
 title - 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)
 filter - フィルタ (null を指定した場合は、全てのファイルが対象)
 defExt - 既定の拡張子 (ピリオド (.) は指定不要、デフォルト拡張子なしの場合は null)

説 明 : 以下のダイアログボックス操作により、複数のオープンするファイル・パス名を取得します。



filter 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します。

「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)/*.*")

filter で指定した内容は、ダイアログ中での選択項目となります。

defExt 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。

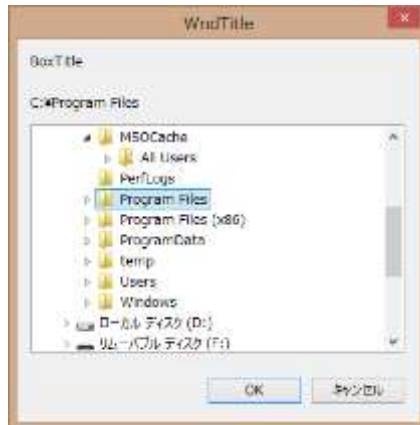
戻り値 : ≠null : OK ボタンで終了 (取得したファイルのパス名の文字列配列が返されます)
 =null : キャンセルボタンで終了

24.1.7. フォルダ名取得 (GetFolder)

形 式 : static string GetFolder(string wndTitle, string boxTitle);
 static string GetFolder(object owner , string wndTitle, string boxTitle);
 static string GetFolder(IntPtr hOwner, string wndTitle, string boxTitle);

引 数 : owner - オーナーウインドオブジェクト (フォーム等)
 hOwner - オーナーウインドハンドル (フォームの Handle プロパティ等を指定します。)
 wndTitle - ウィンドタイトルバーに表示するテキスト
 boxTitle - ダイアログボックス内に表示するタイトルテキスト

説 明 : 以下のダイアログボックス操作により、フォルダ・パス名を取得します。



戻り値 : 空文字列以外 : OKボタンで終了 (取得したフォルダのパス名が返されます)
 空文字列("") : キャンセルボタンで終了

24.1.8. 起動したプログラムのフォルダ・パス名取得 (GetAppPath)

形 式 : static string GetAppPath();

引 数 : なし

説 明 : 起動したプログラムのフォルダ・パス名を取得します。
 取得したフォルダ名の末尾には、常に「¥」が付加されます。

戻り値 : 起動したプログラムのフォルダ・パス名

24.1.9. パス名の連結 (PathCat)

形 式 : static string PathCat(string s1, string s2)

引 数 : s1, s2 - 連結する2つのパス名

説 明 : 2つのパス名の間に、必ず1つの「¥」が挿入されるように、2つの文字列を連結します。
 つまり、s1 で示される文字列の末尾と s2 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。
 s1 で示される文字列の末尾と s2 で示される文字列の先頭文字がともに「¥」である場合は、s2 の先頭文字である「¥」を除去します。

ex. 「d:¥work¥sub」 + 「Sample.txt」 ---> 「d:¥work¥sub¥Sample.txt」
 「d:¥work¥sub¥」 + 「Sample.txt」 ---> 「d:¥work¥sub¥Sample.txt」
 「d:¥work¥sub」 + 「¥Sample.txt」 ---> 「d:¥work¥sub¥Sample.txt」
 「d:¥work¥sub¥」 + 「¥Sample.txt」 ---> 「d:¥work¥sub¥Sample.txt」

戻り値 : 連結した文字列

24.1.10. 接頭語に続くパラメタ取得 (AfterPrefix)

形 式 : static string AfterPrefix(string str, string prefix, out int val)

引 数 : str - 文字列
 prefix - 接頭語
 val - 接頭語に続く部分の文字列が表す数値を格納する変数

説 明 : 文字列(str)の先頭部分が接頭語(prefix)である場合、接頭語に続く部分の文字列と数値を取得します。
 接頭語の比較は、英字の大小を区別しません。

ex. str = "/P123", prefix = "/p" ---> val = 123 , 戻り値 = "123"
 str = "/P123", prefix = "/Q" ---> val は変化なし, 戻り値 = null

戻り値 : ≠null - 接頭語に続く部分の文字列 (文字列の先頭部分が接頭語と一致した)
 =null - 文字列の先頭部分は接頭語と一致しない

24.1.11. フォーム内の全コントロールのイネーブル(EnableAllControls)

形 式 : static void EnableAllControls (object form, bool fEnable);
 static void EnableAllControls (IntPtr hWnd, bool fEnable);

引 数 : form - フォーム・オブジェクト
 hWnd - フォームのウインドハンドル (Handle プロパティ)
 fEnable - フォーム内全コントロールの有効化(true)/無効化(false)フラグ

説 明 : フォーム内の全コントロールを有効化/無効化します。

戻り値 : なし

24.1.12. 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イネーブル(EnableGroup)

形 式 : static void EnableGroup(object gbox, bool fEnableGrpBox, bool fEnableCtrls);
 static void EnableGroup(IntPtr hWnd, bool fEnableGrpBox, bool fEnableCtrls);

引 数 : gbox - グループボックス
 hWnd - グループボックスのウインドハンドル (Handle プロパティを指定します)
 fEnableGrpBox - グループボックス自身の有効化(true)/無効化(false)フラグ
 fEnableCtrls - グループボックス内の全コントロールの有効化(true)/無効化(false)フラグ

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有効化(true)/無効化(false)します。

戻り値 : なし

24.1.13. 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表示/非表示(ShowGroup)

形 式 : static void ShowGroup(object gbox , bool fShowGrpBox, bool fShowCtrls);
 static void ShowGroup(IntPtr hWnd, bool fShowGrpBox, bool fShowCtrls);

引 数 : gbox - グループボックス・オブジェクト
 hWnd - グループボックスのウインドハンドル (Handle プロパティを指定します)
 fShowGrpBox - グループボックス自身の表示(true)/非表示(false)フラグ
 fShowCtrls - グループボックス内の全コントロールの表示(true)/非表示(false)フラグ

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。

戻り値 : なし

24.1.14. 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 (MoveGroup)

形 式 : static void MoveGroup(object gbox, int x, int y); -- 指定位置へ移動
static void MoveGroup(object gbox, Point pt); -- "
static void MoveGroup(IntPtr hWnd, int x, int y); -- "
static void MoveGroup(IntPtr hWnd, Point pt); -- "

static void MoveGroup(object gbox);----- 元の位置へ戻る
static void MoveGroup(IntPtr hWnd); ----- "

引 数 : gbox - グループボックス
hWnd - グループボックスのウインドハンドル (Handle プロパティを指定します)
x, y - 移動先のピクセル位置 (フォームの左上が原点(, 0, 0))
pt - 同上

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移動します。
移動先 (x, y) を指定した場合は、当該位置へ移動します。
移動先を指定しない場合は、元の位置へ戻ります。

戻り値 : なし

24.1.15. 2つの角度値の変移角を求める (DifferenceOfTheta)

形 式 : static double DifferenceOfTheta(double t1, double t2);

引 数 : t1, t2 - 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。
周回遅れの角度は同一角度とみなします。
例えば、0度と360度は、同一角度とみなし、差は0度となります。

(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 (-180 ~ +180 [度])

24.1.16. Windows のシャットダウンやリブートを行う(ExitWindows)

形 式 : static bool ExitWindows (EExitWindows flag); ----- プロセスを強制終了しない
static bool ExitWindows (EExitWindows flag, force); -- プロセスの強制終了を指定

引 数 : flag - 動作フラグ (EWX_XXXX)
force - プロセスの強制終了フラグ (true:強制終了する, false: 強制終了しない)

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、以下のいずれかのシンボルを指定します。

#	シンボル	内容
1	EWX_LOGOFF	現在のユーザをログオフします
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

戻り値 : true - OK
false - エラー

24.1.17. 1970/1/1 00:00:00 からの通算秒を日時に変換 (Time1970ToDateAndTime)

形 式 : static DateTime Time1970ToDateAndTime(uint Time1970);
 static DateTime Time1970ToDateAndTime(ulong Time1970);

引 数 : Time1970 - 1970/01/01 00:00:00 からの通算秒数

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 0xFFFFFFFF=2106/02/07 06:28:14) を、「年月日・時分秒」に変換します。

戻り値 : 変換した日時の DateTime オブジェクト

24.1.18. 日時を 1970/1/1 00:00:00 からの通算秒に変換 (DateAndTimeToTime1970)

形 式 : static uint DateAndTimeToTime1970 (DateTime dt);
 static ulong DateAndTimeToTime1970L(DateTime dt);

引 数 : dt - 1970/01/01 00:00:00 からの通算秒数に変換する日時 (32Bit 時の最大は 2106/02/07 06:28:14 -> 0xFFFFFFFF)

説 明 : dt 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。

戻り値 : 1970/1/1 00:00:00 からの経過秒数

24.1.19. UTC 日時をローカルタイムに変換 (UtcTimeToLocalTime)

形 式 : static DateTime UtcTimeToLocalTime(DateTime dt);

引 数 : dt - UTC 日時

説 明 : dt で示される UTC 日時を、ローカルタイムに変換します。

戻り値 : ローカルタイムの DateTime オブジェクト

24.1.20. 10進数文字列の整数部で規定桁数毎に区切り文字を挿入する(SepDecimal)

形 式 : static string SepDecimal(string strDecimal);

引 数 : strDecimal - 10進数の文字列 (ex. "1234567.89012")

説 明 : 指定された 10 進数文字列の整数部で、既定桁数毎に区切り文字 (ロケールに依存) を挿入します。
 10 進数文字列は、以下の形式でなければなりません。

[空白]・・[+ -]nnnnnnnn.fffff[任意]・・

「nnnnnnnn」: 整数部, 「fffff」: 小数部

ex. string s = CAjrGsr.SepDecimal(" -1234567.14159 "); // s = " 1,234,567.14159 " (日本の場合)

戻り値 : 既定の区切り文字を挿入した 10 進数文字列

備 考 : 日本の場合、区切り文字はカンマ(,)で、3 桁毎に挿入します。

24.1.21. 10進数文字列の整数部から区切り文字を削除する(RmvSepChar)

形 式 : static string RmvSepChar(string strSepDecimal);

引 数 : strSepDecimal - 整数部が規定文字で区切られた10進数の文字列 (ex. “1,234,567.14159”)

説 明 : 指定された10進数文字列の整数部から既定の区切り文字 (ロケールに依存) を削除します。

ex. string s = CAjrGsr. RmvSepChar(“-1,234,567.14159”); // s = “-1234567.14159” (日本の場合)

戻り値 : 既定の区切り文字を削除した10進数文字列

備 考 : 日本の場合、区切り文字はカンマ(,)です。

24.1.22. テキストボックスにフォルダやファイルをドロップ可能にする(EnableToDrop)

形 式 : static void EnableDropToTextBox(object ctrl, EDropMode dm); ——— ① 個別指定 (テキストボックス指定)
 static void EnableDropToTextBox (IntPtr hWnd, EDropMode dm); ——— ② 個別指定 (テキストボックスのハンドル指定)
 static void EnableDropToTextBox (Control form); ——— ③ 一括指定
 static void EnableDropToTextBox (Control form, string kw, Char d1, Char d2); - ④ 一括指定 (キーワード, 区切り文字指定)

引 数 : ctrl - テキストボックスコントロール
 dm - ドロップモード
 form - フォームコントロール
 kw - キーワード (③では「Drop」を仮定)
 d1 - 区切り文字1 (③ではカンマ(,)を仮定)
 d2 - 区切り文字2 (③ではコロン(:)を仮定)

説 明 : 指定されたテキストボックスに (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、テキストボックスにディレクトリやファイルのパス名が設定されます。
 ①と②は、1つのテキストボックスを指定します。
 ドロップモード (dm) は、以下のシンボルの合成値で指定します。

シンボル	内容	備考
DIR	ディレクトリをドロップ可能とする	いずれか1つを指定
FILE	ファイルをドロップ可能とする	
BOTH	ディレクトリとファイルをドロップ可能とする	
CONNECT	複数個をドロップ可能とする	全ドロップ項目をセミコロン(;)で連結
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	

DIR を指定してファイルをドロップ、あるいは、FILE を指定してディレクトリをドロップした場合は、以下のエラーチップを表示します。(但し、NOERRTIP を指定した場合は表示しません)

- ・DIR を指定し、ファイルをドロップした場合 ----- ディレクトリをドロップしてください
- ・FILE を指定し、ディレクトリをドロップした場合 --- ファイルをドロップしてください

③と④は、フォーム内のテキストボックス群に一括してドロップモードを指定します。
 ③では、ドロップモードを各テキストボックスの Tag プロパティで以下の文字列により指定します。(英字の大小は区別しない)

Tag= “[他のオプション, ...], Drop:[D|F|B][C][N] [,他のオプション, ...]”

D, F, B, C, N は、ドロップモードで指定するシンボル (DIR, FILE, BOTH, CONNECT, NOERRTIP) の先頭文字を意味します。
 Tag プロパティで他のオプションと併用する場合は、カンマ(,)で区切ります。

例えば、複数個のディレクトリをドロップ可能とするには、Tag=“XXX, Drop:DC, YYY” とします。(XXX, YYYは他のオプション)

④では、他のオプションと併用する場合のキーワード(Drop)を kw で、区切り記号(,)を d1 で、“Drop”に続く区切り記号(:)を d2 で指定します。(d1 と d2 は、異なる区切り文字を指定しなければなりません)

戻り値 : なし

24.1.23. デスクトップ上のウインド位置取得(GetWindowPos)

形 式 : static Point GetWindowPos(Control ctl);

引 数 : ctl - ウインド位置を取得するコントロール

説 明 : コントロールのデスクトップ上のウインド位置を取得します。

戻り値 : コントロールのデスクトップ上のウインド位置

25. スタティク：チェックサム (CAjrStatic.dll, SAjrCS クラス)

チェックサム算出に関するスタティクメソッド群です。クラス名は「SAjrCS」です。

25.1. メソッド

#	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)
1	サム値算出 (単純加算)	CalcByteSum	CalcWordSum	CalcWordSumR
	(1 の補数)	CalcByteSumN	CalcWordSumN	CalcWordSumNR
	(2 の補数)	CalcByteSumS	CalcWordSumS	CalcWordSumSR
	(排他論理和)	CalcByteXumS	CalcWordSumX	CalcWordSumXR
2	サム値設定 (単純加算)	SetByteSum	SetWordSum	SetWordSumR
	(1 の補数)	SetByteSumN	SetWordSumN	SetWordSumNR
	(2 の補数)	SetByteSumS	SetWordSumS	SetWordSumSR
	(排他論理和)	SetByteSumX	SetWordSumX	SetWordSumXR
3	サム値検査 (単純加算)	ChkByteSum	ChkWordSum	ChkWordSumR
	(1 の補数)	ChkByteSumN	ChkWordSumN	ChkWordSumNR
	(2 の補数)	ChkByteSumS	ChkWordSumS	ChkWordSumSR
	(排他論理和)	ChkByteSumX	ChkWordSumX	ChkWordSumXR

25.1.1. ストリームのバイト／ワードサム算出 (CalcByteSum[N|S], CalcWordSum[N|S])

形 式 : static unsafe byte CalcByteSum (void *p, int len); — バイトサム, 単純加算
 static unsafe byte CalcByteSumN (void *p, int len); — " , 1の補数
 static unsafe byte CalcByteSumS (void *p, int len); — " , 2の補数
 static unsafe byte CalcByteSumX (void *p, int len); — " , 排他論理和
 static unsafe ushort CalcWordSum (void *p, int len); — ワードサム, 単純加算
 static unsafe ushort CalcWordSumN (void *p, int len); — " , 1の補数
 static unsafe ushort CalcWordSumS (void *p, int len); — " , 2の補数
 static unsafe ushort CalcWordSumX (void *p, int len); — " , 排他論理和
 static unsafe ushort CalcWordSumR (void *p, int 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)
 static unsafe ushort CalcWordSumNR (void *p, int len); — " , 1の補数 (自CPUとは、逆のエンディアン形式で扱う)
 static unsafe ushort CalcWordSumSR (void *p, int len); — " , 2の補数 (自CPUとは、逆のエンディアン形式で扱う)
 static unsafe ushort CalcWordSumXR (void *p, int len); — " , 排他論理和 (自CPUとは、逆のエンディアン形式で扱う)

static byte CalcByteSum (IntPtr p, int len); — バイトサム, 単純加算
 static byte CalcByteSumN (IntPtr p, int len); — " , 1の補数
 static byte CalcByteSumS (IntPtr p, int len); — " , 2の補数
 static byte CalcByteSumX (IntPtr p, int len); — " , 排他論理和
 static ushort CalcWordSum (IntPtr p, int len); — ワードサム, 単純加算
 static ushort CalcWordSumN (IntPtr p, int len); — " , 1の補数
 static ushort CalcWordSumS (IntPtr p, int len); — " , 2の補数
 static ushort CalcWordSumX (IntPtr p, int len); — " , 排他論理和
 static ushort CalcWordSumR (IntPtr p, int 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)
 static ushort CalcWordSumNR (IntPtr p, int len); — " , 1の補数 (自CPUとは、逆のエンディアン形式で扱う)
 static ushort CalcWordSumSR (IntPtr p, int len); — " , 2の補数 (自CPUとは、逆のエンディアン形式で扱う)
 static ushort CalcWordSumXR (IntPtr p, int 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 : p - 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」のメソッドは、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」のメソッドは、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)

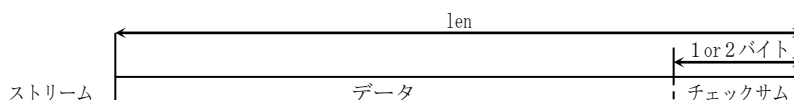
25.1.2. ストリームのバイト／ワードサム設定 (SetByteSum[N|S], SetWordSum[N|S])

形 式 : static unsafe void SetByteSum (void *p, int len); --- バイトサム, 単純加算
static unsafe void SetByteSumN (void *p, int len); --- " , 1の補数
static unsafe void SetByteSumS (void *p, int len); --- " , 2の補数
static unsafe void SetByteSumX (void *p, int len); --- " , 排他論理和
.....
static unsafe void SetWordSum (void *p, int len); --- ワードサム, 単純加算
static unsafe void SetWordSumN (void *p, int len); --- " , 1の補数
static unsafe void SetWordSumS (void *p, int len); --- " , 2の補数
static unsafe void SetWordSumX (void *p, int len); --- " , 排他論理和
.....
static unsafe void SetWordSumR (void *p, int 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)
static unsafe void SetWordSumNR(void *p, int len); --- " , 1の補数 (自CPUとは、逆のエンディアン形式で扱う)
static unsafe void SetWordSumSR(void *p, int len); --- " , 2の補数 (自CPUとは、逆のエンディアン形式で扱う)
static unsafe void SetWordSumXR(void *p, int len); --- " , 排他論理和自CPUとは、逆のエンディアン形式で扱う

```
static void SetByteSum (IntPtr p, int len); --- バイトサム, 単純加算
static void SetByteSumN (IntPtr p, int len); --- " , 1の補数
static void SetByteSumS (IntPtr p, int len); --- " , 2の補数
static void SetByteSumX (IntPtr p, int len); --- " , 排他論理和
.....
static void SetWordSum (IntPtr p, int len); --- ワードサム, 単純加算
static void SetWordSumN (IntPtr p, int len); --- " , 1の補数
static void SetWordSumS (IntPtr p, int len); --- " , 2の補数
static void SetWordSumX (IntPtr p, int len); --- " , 排他論理和
.....
static void SetWordSumR (IntPtr p, int 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)
static void SetWordSumNR(IntPtr p, int len); --- " , 1の補数 (自CPUとは、逆のエンディアン形式で扱う)
static void SetWordSumSR(IntPtr p, int len); --- " , 2の補数 (自CPUとは、逆のエンディアン形式で扱う)
static void SetWordSumXR(IntPtr p, int 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 : p - 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」のメソッドは、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」のメソッドは、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 : なし

25.1.3. ストリームのバイト／ワードサム検査 (ChkByteSum[N|S], ChkWordSum[N|S])

形 式 : static unsafe bool ChkByteSum (void *p, int len); --- バイトサム, 単純加算
static unsafe bool ChkByteSumN (void *p, int len); --- " , 1の補数
static unsafe bool ChkByteSumS (void *p, int len); --- " , 2の補数
static unsafe bool ChkByteSumX (void *p, int len); --- " , 排他論理和

static unsafe bool ChkWordSum (void *p, int len); --- ワードサム, 単純加算
static unsafe bool ChkWordSumN (void *p, int len); --- " , 1の補数
static unsafe bool ChkWordSumS (void *p, int len); --- " , 2の補数
static unsafe bool ChkWordSumX (void *p, int len); --- " , 排他論理和

static unsafe bool ChkWordSumR (void *p, int 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)
static unsafe bool ChkWordSumNR (void *p, int len); --- " , 1の補数 (自CPUとは、逆のエンディアン形式で扱う)
static unsafe bool ChkWordSumSR (void *p, int len); --- " , 2の補数 (自CPUとは、逆のエンディアン形式で扱う)
static unsafe bool ChkWordSumXR (void *p, int len); --- " , 排他論理和 (自CPUとは、逆のエンディアン形式で扱う)

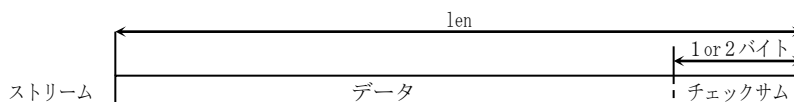
static bool ChkByteSum (IntPtr p, int len); --- バイトサム, 単純加算
static bool ChkByteSumN (IntPtr p, int len); --- " , 1の補数
static bool ChkByteSumS (IntPtr p, int len); --- " , 2の補数
bool ChkByteSumX (IntPtr p, int len); --- " , 排他論理和

static bool ChkWordSum (IntPtr p, int len); --- ワードサム, 単純加算
static bool ChkWordSumN (IntPtr p, int len); --- " , 1の補数
static bool ChkWordSumS (IntPtr p, int len); --- " , 2の補数
static bool ChkWordSumX (IntPtr p, int len); --- " , 排他論理和

static bool ChkWordSumR (IntPtr p, int 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)
static bool ChkWordSumNR (IntPtr p, int len); --- " , 1の補数 (自CPUとは、逆のエンディアン形式で扱う)
static bool ChkWordSumSR (IntPtr p, int len); --- " , 2の補数 (自CPUとは、逆のエンディアン形式で扱う)
static bool ChkWordSumXR (IntPtr p, int 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 : p - 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」のメソッドは、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」のメソッドは、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

26. スタティック : CRC 計算 (CAjrStatic.dll, SAjrcrc クラス)

CRC 算出に関するスタティックメソッド群です。クラス名は「SAjrcrc」です。

26.1. メソッド

#	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	PartCrcItL	PartCrcItR	PartCrc16L	PartCrc16R
2	CRC値算出	BlkCrcItL	BlkCrcItR	BlkCrc16L	BlkCrc16R
3	XMODEM/FCS算出	BlkXMODEM	BlkFCS	—	—
4	CRC値設定	SetCrcItL	SetCrcItR	SetCrc16L	SetCrc16R
	(Big Endian)	SetCrcItL_BE	SetCrcItR_BE	SetCrc16L_BE	SetCrc16R_BE
	(Little Endian)	SetCrcItL_LE	SetCrcItR_LE	SetCrc16L_LE	SetCrc16R_LE
5	XMODEM/FCS設定	SetXMODEM	SetFCS	—	—
	(Big Endian)	SetXMODEM_BE	SetFCS_BE	—	—
	(Little Endian)	SetXMODEM_LE	SetFCS_LE	—	—
6	CRCチェック	ChkCrcItL	ChkCrcItR	ChkCrc16L	ChkCrc16R
	(Big Endian)	ChkCrcItL_BE	ChkCrcItR_BE	ChkCrc16L_BE	ChkCrc16R_BE
	(Little Endian)	ChkCrcItL_LE	ChkCrcItR_LE	ChkCrc16L_LE	ChkCrc16R_LE
7	XMODEM/FCSチェック	ChkXMODEM	ChkFCS	—	—
	(Big Endian)	ChkXMODEM_BE	ChkFCS_BE	—	—
	(Little Endian)	ChkXMODEM_LE	ChkFCS_LE	—	—

26.1.1. 部分CRC算出 (PartCrc??L / PartCrc??R)

形 式 : static unsafe ushort PartCrcItL (void *p, int len, ushort ini); --- CCITT 方式, 左送り演算
static unsafe ushort PartCrcItR (void *p, int len, ushort ini); --- CCITT 方式, 右送り演算

static unsafe ushort PartCrc16L (void *p, int len, ushort ini); --- ANSI-CRC16, 左送り演算
static unsafe ushort PartCrc16R (void *p, int len, ushort ini); --- ANSI-CRC16, 右送り演算

static ushort PartCrcItL (IntPtr p, int len, ushort ini); --- CCITT 方式, 左送り演算
static ushort PartCrcItR (IntPtr p, int len, ushort ini); --- CCITT 方式, 右送り演算

static ushort PartCrc16L (IntPtr p, int len, ushort ini); --- ANSI-CRC16, 左送り演算
static ushort PartCrc16R (IntPtr p, int len, ushort ini); --- ANSI-CRC16, 右送り演算

引 数 : p - 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

26.1.2. CRC算出 (BlkCrc??L / BlkCrc??R)

形 式 : static unsafe ushort BlkCrcItL (void *p, int len, ushort ini); --- CCITT 方式, 左送り演算
static unsafe ushort BlkCrcItR (void *p, int len, ushort ini); --- CCITT 方式, 右送り演算

static unsafe ushort BlkCrc16L (void *p, int len, ushort ini); --- ANSI-CRC16, 左送り演算
static unsafe ushort BlkCrc16R (void *p, int len, ushort ini); --- ANSI-CRC16, 右送り演算

static ushort BlkCrcItL (IntPtr p, int len, ushort ini); --- CCITT 方式, 左送り演算
static ushort BlkCrcItR (IntPtr p, int len, ushort ini); --- CCITT 方式, 右送り演算

static ushort BlkCrc16L (IntPtr p, int len, ushort ini); --- ANSI-CRC16, 左送り演算
static ushort BlkCrc16R (IntPtr p, int len, ushort ini); --- ANSI-CRC16, 右送り演算

引 数 : p - 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

26.1.3. XMODEM-CRC/FCS算出 (BlkXMODEM / BlkCalcFCS)

形 式 : static unsafe ushort BlkXMODEM (void *p, int len); --- 左送り演算で、XMODEM 用の CRC 算出
static unsafe ushort BlkFCS (void *p, int len); --- 右送り演算で、FCS (CCITT 方式の CRC) 算出

static ushort BlkXMODEM (IntPtr p, int len); --- 左送り演算で、XMODEM 用の CRC 算出
static ushort BlkFCS (IntPtr p, int len); --- 右送り演算で、FCS (CCITT 方式の CRC) 算出

引 数 : p - 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

26.1.3.1. ストリームへCRC値設定 (SetCrc??・・・)

形 式 : static unsafe void SetCrcItL (void *p, int len, ushort ini); --- CCITT 方式, 左送り演算, CPU 依存のエンディアン形式で設定
static unsafe void SetCrcItL_BE(void *p, int len, ushort ini); --- CCITT 方式, 左送り演算, ビッグエンディアン形式で設定
static unsafe void SetCrcItL_LE(void *p, int len, ushort ini); --- CCITT 方式, 左送り演算, リトルエンディアン形式で設定
static unsafe void SetCrcItR (void *p, int len, ushort ini); --- CCITT 方式, 右送り演算, CPU 依存のエンディアン形式で設定
static unsafe void SetCrcItR_BE(void *p, int len, ushort ini); --- CCITT 方式, 右送り演算, ビッグエンディアン形式で設定
static unsafe void SetCrcItR_LE(void *p, int len, ushort ini); --- CCITT 方式, 右送り演算, リトルエンディアン形式で設定

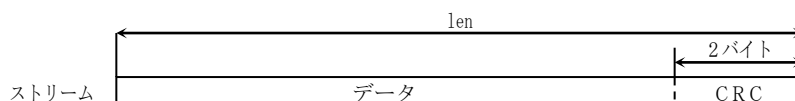
static unsafe void SetCrc16L (void *p, int len, ushort ini); --- ANSI-CRC16, 左送り演算, CPU 依存のエンディアン形式で設定
static unsafe void SetCrc16L_BE(void *p, int len, ushort ini); --- ANSI-CRC16, 左送り演算, ビッグエンディアン形式で設定
static unsafe void SetCrc16L_LE(void *p, int len, ushort ini); --- ANSI-CRC16, 左送り演算, リトルエンディアン形式で設定
static unsafe void SetCrc16R (void *p, int len, ushort ini); --- ANSI-CRC16, 右送り演算, CPU 依存のエンディアン形式で設定
static unsafe void SetCrc16R_BE(void *p, int len, ushort ini); --- ANSI-CRC16, 右送り演算, ビッグエンディアン形式で設定
static unsafe void SetCrc16R_LE(void *p, int len, ushort ini); --- ANSI-CRC16, 右送り演算, リトルエンディアン形式で設定

static void SetCrcItL (IntPtr p, int len, ushort ini); --- CCITT 方式, 左送り演算, CPU 依存のエンディアン形式で設定
static void SetCrcItL_BE(IntPtr p, int len, ushort ini); --- CCITT 方式, 左送り演算, ビッグエンディアン形式で設定
static void SetCrcItL_LE(IntPtr p, int len, ushort ini); --- CCITT 方式, 左送り演算, リトルエンディアン形式で設定
static void SetCrcItR (IntPtr p, int len, ushort ini); --- CCITT 方式, 右送り演算, CPU 依存のエンディアン形式で設定
static void SetCrcItR_BE(IntPtr p, int len, ushort ini); --- CCITT 方式, 右送り演算, ビッグエンディアン形式で設定
static void SetCrcItR_LE(IntPtr p, int len, ushort ini); --- CCITT 方式, 右送り演算, リトルエンディアン形式で設定

static void SetCrc16L (IntPtr p, int len, ushort ini); --- ANSI-CRC16, 左送り演算, CPU 依存のエンディアン形式で設定
static void SetCrc16L_BE(IntPtr p, int len, ushort ini); --- ANSI-CRC16, 左送り演算, ビッグエンディアン形式で設定
static void SetCrc16L_LE(IntPtr p, int len, ushort ini); --- ANSI-CRC16, 左送り演算, リトルエンディアン形式で設定
static void SetCrc16R (IntPtr p, int len, ushort ini); --- ANSI-CRC16, 右送り演算, CPU 依存のエンディアン形式で設定
static void SetCrc16R_BE(IntPtr p, int len, ushort ini); --- ANSI-CRC16, 右送り演算, ビッグエンディアン形式で設定
static void SetCrc16R_LE(IntPtr p, int len, ushort ini); --- ANSI-CRC16, 右送り演算, リトルエンディアン形式で設定

引 数 : p - 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
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」のメソッドは、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」のメソッドは、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

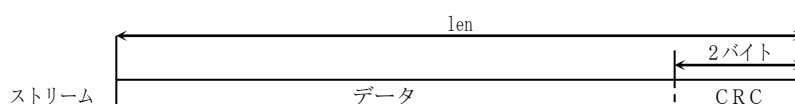
26.1.4. ストリームへ XMODEM / FCS 設定 (SetXMODEM . . . / SetFCS . . .)

形 式 : static unsafe void SetXMODEM (void *p, int len); --- XMODEM-CRC, CPU 依存のエンディアン形式で設定
 static unsafe void SetXMODEM_BE (void *p, int len); --- XMODEM-CRC, ビッグエンディアン形式で設定 (XMODEM 標準)
 static unsafe void SetXMODEM_LE (void *p, int len); --- XMODEM-CRC, リトルエンディアン形式で設定
 static unsafe void SetFCS (void *p, int len); --- F C S, CPU 依存のエンディアン形式で設定
 static unsafe void SetFCS_BE (void *p, int len); --- F C S, ビッグエンディアン形式で設定
 static unsafe void SetFCS_LE (void *p, int len); --- F C S, リトルエンディアン形式で設定

static void SetXMODEM (IntPtr p, int len); --- XMODEM-CRC, CPU 依存のエンディアン形式で設定
 static void SetXMODEM_BE (IntPtr p, int len); --- XMODEM-CRC, ビッグエンディアン形式で設定 (XMODEM 標準)
 static void SetXMODEM_LE (IntPtr p, int len); --- XMODEM-CRC, リトルエンディアン形式で設定
 static void SetFCS (IntPtr p, int len); --- F C S, CPU 依存のエンディアン形式で設定
 static void SetFCS_BE (IntPtr p, int len); --- F C S, ビッグエンディアン形式で設定
 static void SetFCS_LE (IntPtr p, int len); --- F C S, リトルエンディアン形式で設定

引 数 : p - 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

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」のメソッドは、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」のメソッドは、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

26.1.5. ストリームのCRC値検査 (ChkCrc??・・・・)

形式 : static unsafe bool ChkCrcItL (void *, int len, ushort 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
 static unsafe bool ChkCrcItL_BE(void *, int len, ushort 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
 static unsafe bool ChkCrcItL_LE(void *, int len, ushort 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
 static unsafe bool ChkCrcItR (void *, int len, ushort 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
 static unsafe bool ChkCrcItR_BE(void *, int len, ushort 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
 static unsafe bool ChkCrcItR_LE(void *, int len, ushort 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

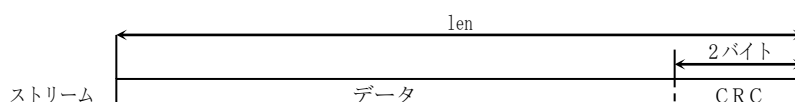
static unsafe bool ChkCrc16L (void *, int len, ushort ini); — ANSI-CRC16, 左送り演算, CPU 依存のエンディアン形式として検査
 static unsafe bool ChkCrc16L_BE(void *, int len, ushort ini); — ANSI-CRC16, 左送り演算, ビッグエンディアン形式として検査
 static unsafe bool ChkCrc16L_LE(void *, int len, ushort ini); — ANSI-CRC16, 左送り演算, リトルエンディアン形式として検査
 static unsafe bool ChkCrc16R (void *, int len, ushort 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
 static unsafe bool ChkCrc16R_BE(void *, int len, ushort ini); — ANSI-CRC16, 右送り演算, ビッグエンディアン形式として検査
 static unsafe bool ChkCrc16R_LE(void *, int len, ushort ini); — ANSI-CRC16, 右送り演算, リトルエンディアン形式として検査

static bool ChkCrcItL (IntPtr p, int len, ushort 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
 static bool ChkCrcItL_BE(IntPtr p, int len, ushort 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
 static bool ChkCrcItL_LE(IntPtr p, int len, ushort 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
 static bool ChkCrcItR (IntPtr p, int len, ushort 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
 static bool ChkCrcItR_BE(IntPtr p, int len, ushort 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
 static bool ChkCrcItR_LE(IntPtr p, int len, ushort 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

static bool ChkCrc16L (IntPtr p, int len, ushort 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
 static bool ChkCrc16L_BE(IntPtr p, int len, ushort ini); — ANSI-CRC16, 左送り演算, ビッグエンディアン形式として検査
 static bool ChkCrc16L_LE(IntPtr p, int len, ushort ini); — ANSI-CRC16, 左送り演算, リトルエンディアン形式として検査
 static bool ChkCrc16R (IntPtr p, int len, ushort 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
 static bool ChkCrc16R_BE(IntPtr p, int len, ushort ini); — ANSI-CRC16, 右送り演算, ビッグエンディアン形式として検査
 static bool ChkCrc16R_LE(IntPtr p, int len, ushort 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 : p - 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
 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」のメソッドは、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」のメソッドは、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

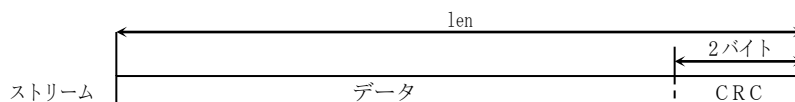
26.1.6. ストリームの XMODEM / FCS 検査 (ChkXMODEM . . . / ChkFCS . . .)

形 式 : static unsafe bool ChkXMODEM (void *p, int len); — XMODEM-CRC, CPU依存のエンディアン形式として検査
 static unsafe bool ChkXMODEM_BE (void *p, int len); — XMODEM-CRC, ビッグエンディアン形式として検査 (XMODEM 標準)
 static unsafe bool ChkXMODEM_LE (void *p, int len); — XMODEM-CRC, リトルエンディアン形式として検査
 static unsafe bool ChkFCS (void *p, int len); — FCS, CPU依存のエンディアン形式として検査
 static unsafe bool ChkFCS_BE (void *p, int len); — FCS, ビッグエンディアン形式として検査
 static unsafe bool ChkFCS_LE (void *p, int len); — FCS, リトルエンディアン形式として検査

static bool ChkXMODEM (IntPtr p, int len); — XMODEM-CRC, CPU依存のエンディアン形式として検査
 static bool ChkXMODEM_BE (IntPtr p, int len); — XMODEM-CRC, ビッグエンディアン形式として検査 (XMODEM 標準)
 static bool ChkXMODEM_LE (IntPtr p, int len); — XMODEM-CRC, リトルエンディアン形式として検査
 static bool ChkFCS (IntPtr p, int len); — FCS, CPU依存のエンディアン形式として検査
 static bool ChkFCS_BE (IntPtr p, int len); — FCS, ビッグエンディアン形式として検査
 static bool ChkFCS_LE (IntPtr p, int len); — FCS, リトルエンディアン形式として検査

引 数 : p - 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

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」のメソッドは、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」のメソッドは、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

27. スタティク：ネイティブデータ・アクセス (CAjrStatic.dll, SAjrBin クラス)

ネイティブデータメモリのアクセス用スタティクメソッド群です。クラス名は「SAjrBin」です。

27.1.1. 指定アドレスのメモリ間でメモリコピー(MemCopy)

形 式 : IntPtr MemCopy(IntPtr pDest, IntPtr pSrc, int len);
 unsafe IntPtr MemCopy(void* pDest, void* pSrc, int len);
 unsafe IntPtr MemCopy(IntPtr pDest, void* pSrc, int len);
 unsafe IntPtr MemCopy(void *pDest, IntPtr pSrc, int len);

引 数 : pDest - コピー先メモリアドレス
 pSrc - コピー元メモリアドレス
 len - コピーするバイト数

説 明 : メモリをコピーします。

戻り値 : コピー先メモリのアドレス (= pDest)

27.1.2. 文字列とバッファ間で文字列のコピー(StrCopy)

形 式 : bool StrCopy(IntPtr pDest, string pSrc, int lDest);
 bool StrCopy(IntPtr pDest, StringBuilder pSrc, int lDest);
 bool StrCopy(IntPtr pDest, IntPtr pSrc, int lDest);
 unsafe bool StrCopy(IntPtr pDest, void* pSrc, int lDest);
 unsafe bool StrCopy(void* pDest, string pSrc, int lDest);
 unsafe bool StrCopy(void* pDest, StringBuilder pSrc, int lDest);
 unsafe bool StrCopy(void* pDest, IntPtr pSrc, int lDest);
 unsafe bool StrCopy(void* pDest, void* pSrc, int lDest);
 bool StrCopy(StringBuilder pDest, IntPtr pSrc);
 unsafe bool StrCopy(StringBuilder pDest, void* pSrc);

引 数 : pDest - コピー先のバッファ (バッファのアドレス/StringBuilder)
 pSrc - コピー元の文字列 (文字列のアドレス/StringBuilder)
 lDest - コピー先のバッファ文字数 (文字列終端(0x0)を含む)

説 明 : 文字列を指定されたバッファへコピーします。
 バッファサイズが小さい場合は、文字列の先頭部分だけがコピーされます。

戻り値 : true - 文字列全体ををバッファへコピーした
 false - バッファサイズが小さい

27.1.3. 文字列の文字数取得(StrLen)

形 式 : int StrLen(IntPtr pStr);
 unsafe int StrLen(void* pStr);

引 数 : pStr - 文字列のアドレス

説 明 : 文字列の文字数を取得します。

戻り値 : 文字列の文字数 (文字列の終端 '¥0' は含まない)

28. 免責事項

本ソフトウェアは、一通りの動作チェックを行っていますが、動作を完全に保障するものではありません。
本ソフトウェアを使用したアプリケーションの運用結果については、一切の責任を負いかねますのでご了承ください。

29. 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 で確認してください。

<http://www.ne.jp/asahi/ajara/kojara/>