簡単パラメータ(forVC)
マニュアル

version 1.00(シェアウェア版)

第 1.00 版

株式会社　イメージ・アルファ

はじめに

この度は、弊社パラメータ制御ライブラリ IAESP の購入をご検討頂き、ありがとうございます。

本ソフトウェア“簡単パラメータ forVC”(version 1.00 以上)は、シェアウェアとして提供します。詳細は、後述の「使用制限について」を参照ください。

バグ報告、及び追加機能のご要望につきましては、本件開発担当までメールをお願いいたします。

問合せ先
株式会社　イメージ・アルファ
http://www.image-alpha.com
営業担当　info@image-alpha.com
本件開発担当　mochimaru@image-alpha.com

修正履歴

ﾊﾞｰｼﾞｮﾝ番号	分類	修正内容
0.1.0		新規
0.1.1	【機能拡張】	char 文字列型の追加[5.1 節] 最小値指定は、無効です。最大値指定が他と異なります。[6.1 節]参照。
	【機能拡張】	インデックス取得[4.2 節] インデックス取得用マクロを調整し、char 文字列型対応を行い、 " ep_prot.h"に格納しました。従来マクロも使用可能です。
0.1.2	【機能修正】	"0.1.1"で問題のあった、以下の修正を行いました。 ・テキストボックス char 文字列型 ・コンボボックス int 型 ・インデックス取得用マクロ
0.2.0	【機能修正】	デフォルト値修正 ： 初期文字列数の制約をなくすため、char [20]配列から char *に変更しました。 汎用ポインタ追加 ： 次のパラメータタイプのみリスト内文字列設定に使用します。 EP_TYPE_CMB_FLOAT, EP_TYPE_CMB_STRING ※この変更により、0.1.2 以前のオブジェクト互換性がなくなりました。アプリケーションをリビルドして下さい。
	【機能拡張】	スライダ対応の追加[5.1 節] ・int 型 ・float 型(1 倍,0.1 倍,0.01 倍)
	【機能拡張】	スクロールバー(縦・横共通)対応の追加[5.1 節] ・int 型 ・float 型(1 倍,0.1 倍,0.01 倍)
	【機能拡張】	コンボボックス対応の追加[5.1 節] ・float 型 ・char 文字列型
1.0.0	【機能修正】	セクション/キー名修正 ： 文字列数制約をなくすため、char [20]配列から char *に変更しました。 構造体メンバのサイズ追加 ： パラメータタイプと実際に指定したメンバサイズが異なる場合に、エラーを発生するための追加です。 ※これらの変更により、0.2.0 以前のオブジェクト互換性が再度なくなりました。 ヘッダファイル("ep_prot.h")で差分を吸収するため、ソース変更は不要です。 すみませんが、アプリケーションをリビルドして下さい。
	【機能修正】	汎用ポインタ: 次のパラメータタイプのみリスト内文字列設定に使用 (0.2.0)EP_TYPE_CMB_FLOAT, EP_TYPE_CMB_STRING, (追加)EP_TYPE_CMB_STRING_ID, EP_TYPE_CMB_SEL_LONG, EP_TYPE_LST_MULTI_UCH
	【機能修正】	移植性向上のため構造体のﾒﾝﾊﾞ指定を、CPU 依存性の強い int 型から CPU 依存のない long 型に変更しました。すみませんがﾊﾟﾗﾒｰﾀﾀｲﾌﾟの一部を"_INT"から"_LONG"に変更し、対応ﾒﾝﾊﾞの型を int から long にしてください。
	【機能拡張】	EP_OpenParamTable 6.1 節に、拡張用ﾌﾗｸﾞを追加しました。当面 0 にしてください。
	【機能拡張】	コンボボックス対応の追加[5.1 節] int 型-文字列(項目の上から、0,1,2...と int 型で取得)
	【機能拡張】	コンボボックス→リストボックス対応[5.1 節] コンボボックス指定により、リストボックスに対応可能
	【機能拡張】	double 型各種対応[5.1 節] ・テキストボックス(指数表記)/コンボボックス/スライダ/スクロールバー
	【機能拡張】	ラジオボタン対応[5.1 節] ラジオボタンに対応しました。(-1～)で値を取得できます。
	【機能拡張】	レジストリ[5.1 節] 設定ファイルだけでなく、レジストリに対しても入出力を可能にしました。 レジストリ簡単パラメータ開始時の引数 iniFile に、次の文字列(先頭)を使用することにより、レジストリに対して入出力を行います。 "HKEY_CLASSES_ROOT","HKEY_CURRENT_USER","HKEY_LOCAL_MACHINE", "HKEY_USERS","HKEY_CURRENT_CONFIG"
	【機能拡張】	チェックボックス[5.1 節] 従来、int 型の下位 0～7bit のみアクセスできましたが、long 型の全(0～31)bit をアクセス可能にしました。

使用制限について

本ソフトウェア“簡単パラメータ forVC”は、シェアウェアです。
(株)Vector の登録フォームにて、レジストリキー登録をお願いいたします。

なお、営利を目的としない場合、以下の機能制限はありますが、ご自由にお使いください。

レジストリキー未登録の場合は、次機能に制限があります。
・パラメータ数　1 アプリケーションにつき 1 画面(1 回)（登録済では最大 64 画面）
・(制限項目には、変更の可能性があります)

1. 概要

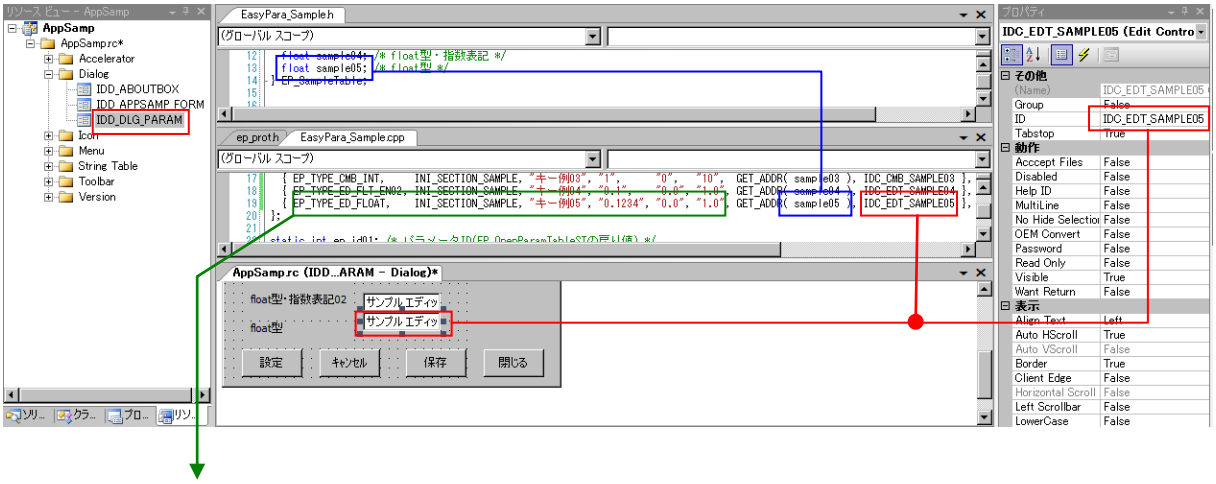
“簡単パラメータ”は、Visual C++による GUI パラメータ制御プログラミングを効率化するライブラリです。Visual C++ 2005 以降に対応しています。

Visual C++による GUI パラメータ制御には、幾つかの方法が用意されています。
しかし、追加は簡単でも削除に手作業が多いなど、効率的とは言えない場面が多くありました。
また、構造体とのやりとりで単純なバグも多く発生しました。

“簡単パラメータ”はこの手作業を軽減し、バグの発生率を抑え、プログラミングを効率化します。
必須部分のインプリメント後は、**パラメータ構造体の定義**、**リソースエディタでの編集**、および**テーブル追加**の三箇所の簡単な変更により、画面、パラメータ、設定ファイル(レジストリ対象可)の入出力を行うことができます。
パラメータ削除も、上記三箇所の削除により行うことができます。

1.1 サンプルプログラム

サンプルプログラム”AppSamp”を提供いたします。組込みの参考にしてください。



左から次のデータを入力します。

パラメータタイプ，セクション名，キー名，基準値，最小値，最大値，**パラメータ構造体メンバ**，**リソース ID**，汎用 char 型ポインタ(一部パラメータタイプで使用・使用しない場合は省略可)

- AppSamp 概要

  SDI (Single Document Interface)

  メインメニューより、[パラメータ 1]画面を開けます。

  [保存]ボタンにより、現在値を”(カレントフォルダ)¥sample.ini”に保存します。

  ※VC2005 の SDI スケルトンに、パラメータ用のダイアログを追加した単純な構成です。

1.2 動作環境

OS　　: Windows XP/Vista/7

利用・動作環境　: Visual C++ 2005 以降 (Visual C++ 2003 以前はサポート対象外)

2. インストール

2.1 インストール

ダウンロードした“EasyPara_*.zip”を解凍し、作成される” IAESP_Installer”フォルダの”setup.exe”を実行してください。(管理者権限で実行してください)

セットアップを完了すると、次のフォルダとファイルが作成されます。
これらのファイルの再配布は禁止します。利用は 2.3 節を参照ください。
“C:¥IA_CLASS¥IAESP” フォルダ

- “ep_prot.h”
- “EasyPara.lib”
- “EasyPara_static.lib” ・・・配布実行ファイル作成時に“EasyPara.lib”の代わりにリンクしてください。

2.2 アンインストール

① Windows XP:
コントロールパネルの[プログラムの追加と削除]から[イメージ・アルファ 簡単パラメータ IAESP]をアンインストールしてください。

② Windows Vista, 7:
コントロールパネルの[プログラムと機能]から[イメージ・アルファ 簡単パラメータ IAESP]をアンインストールしてください。

2.3 VC プロジェクト

“簡単パラメータ”を利用する VC プロジェクトでは、 “ep_prot.h”と“EasyPara.lib”ファイルを以下の様に追加してください。VC2005 の場合です。

① [プロジェクトのプロパティ]により、プロパティ画面を開く
② [構成プロパティ]-[C/C++]-[全般]の[追加のインクルード ディレクトリ]
“C:¥IA_CLASS¥IAESP”を追加してください。
③ [構成プロパティ]-[リンカ]-[入力] の[追加の依存ファイル]
“C:¥IA_CLASS¥IAESP¥EasyPara.lib”を追加してください。
④ “簡単パラメータ”を利用するプログラムソースに“ep_prot.h”を追加してください。

3. ライセンス登録

営利を目的とする場合には、Vector にてライセンスキー登録をして下さい。
折り返し、ライセンスファイルを送付します。
ライセンスファイルがあると、簡単パラメータの全機能が動作します。

3.1 ライセンスファイル

以下の項目の詳細は、送付するライセンスファイルに記載されています。
ライセンスファイルに従ってください。

i) 利用方法
ii) 有効範囲
　このライセンスファイルを営利目的で利用/静的リンクで配布できるのは、Vector に登録した一人に限ります。
　それ以外の利用/配布は認めておりません。営利目的で利用/配布する人は、登録をお願いします。
iii) 利用期限

なお、既に静的リンクで作成した実行ファイルとは対になりますので、更新は必要ありません。

4. インプリメントの手引き

4.1 ラッパー関数

“簡単パラメータ”のライブラリは、構造体の指定に汎用ポインタ(void *)を使用します。これにより、多様なパラメータテーブルに対応できます。

しかし、汎用ポインタはコンパイルでの引数チェックが掛かりません。このため、バグの温床になり易い面があります。

このため、ユーザー定義の構造体を引数とするラッパー関数を用意し、それらを使用する事を強く推奨します。AppSamp には、” EasyPara” フォルダ下に、“EasyPara_Sample.h”, “EasyPara_Sample.cpp”のソースコードを提供しています。

以下にラッパー関数の例を記載します。また、パラメータ ID も隠蔽しています。

“EasyPara_Sample.h”

```
/* パラメータ構造体の例*/
typedef struct _EP_SampleTable {
          long  sample01; /* long型 */
          long  sample02; /* long型Obit : 対象Check-box */
          long  sample03; /* long型: 対象Combo-box */
          float sample04; /* float型・指数表記 */
          float sample05; /* float型 */
          char  sample06[128]; /* char文字列型 */
 } EP_SampleTable;


/* ラッパー関数群 */ /* ST = SampleTable */
void EP_OpenParamTableST( void );
int EP_LoadInifileST( EP_SampleTable *sampleTable );
int EP_SaveInifileST( EP_SampleTable *sampleTable );
int EP_SetParamToDlgST( HWND hwnd, EP_SampleTable *sampleTable );
int EP_GetParamFromDlgST( HWND hwnd, EP_SampleTable *sampleTable );
```

“EasyPara_Sample.cpp”

```
static int ep_id01; /* パラメータ ID(EP_OpenParamTableST の戻り値) */

/* ラッパー関数群 */
void EP_OpenParamTableST( void ) { ep_id01 = EP_OpenParamTable( sizeof(g_EP_ParamTbl) / sizeof(EP_ParamTable),
g_EP_ParamTbl, ".¥¥sample.ini" ); }
int EP_LoadInifileST( EP_SampleTable *sampleTable )                    { return( EP_LoadInifile(      ep_id01,
sampleTable ) ); }
int EP_SaveInifileST( EP_SampleTable *sampleTable )                    { return( EP_SaveInifile(      ep_id01,
sampleTable ) ); }
int EP_SetParamToDlgST( HWND hwnd, EP_SampleTable *sampleTable )    { return( EP_SetParamToDlg(    ep_id01, hwnd,
sampleTable ) ); }
int EP_GetParamFromDlgST( HWND hwnd, EP_SampleTable *sampleTable ) { return( EP_GetParamFromDlg( ep_id01, hwnd,
sampleTable ) ); }
```

4.2 インデックス取得

本ライブラリの特徴です。構造体の各メンバへのインデックスを指定することで、各メンバの値をライブラリ内で直接やり取りします。
下記のコードにより実現します。構造体のダミーポインタが必要です。(0 クリア必須)
構造体のメンバが char[]の文字列型は、必ず下記マクロを利用して最大文字数・インデックスを指定してください。

ただし構造体のメモリ配置は、CPU に近いレベルの知識が必要です。OS が同じ Windows でも、異なる CPU 間で同じメモリ配置を期待してコーディングすると、バッティングする危険性があります。

```
/* パラメータ構造体の例*/
typedef struct _EP_SampleTable {
        long  sample01; /* long型*/
        long  sample02; /* long型0bit : 対象Check-box */
        long  sample03; /* long型: 対象Combo-box */
        float sample04; /* float型・指数表記*/
        float sample05; /* float型*/
        char  sample06[128]; /* char文字列型*/
} EP_SampleTable;


/* インデックス取得用定義 */
static EP_SampleTable *g_dmyPrm = 0;
#define EPG_ADR_BASE(n) (g_dmyPrm->n) /* GET_ADDR, GET_ADDR_STR_MAXの定義に必須 */
#define EPG_ADDR(n)          GET_ADDR(n)          /* 数値の場合のテーブルのインデックス */
#define EPG_ADDR_STR_MAX(n) GET_ADDR_STR_MAX(n) /* 文字列の場合の最大文字数(32bit)とテーブルのインデックス */

static EP_ParamTable g_EP_ParamTbl[] = {
        { EP_TYPE_ED_LONG,      INI_SECTION_SAMPLE, "キー例", "0",      "0",    "3",
EPG_ADDR( sample01 ), IDC_EDT_SAMPLE01 },
        { EP_TYPE_ED_STRING,    INI_SECTION_SAMPLE, "キー例", "C:¥¥IA_CLASS¥¥IAESP", "0",
EPG_ADDR_STR_MAX( sample06 ), IDC_EDT_SAMPLE06 },

/* GET_ADDR, GET_ADDR_STR_MAXの定義を削除(この後に別テーブルを設定可能) */
#undef EPG_ADDR_STR_MAX
#undef EPG_ADDR
#undef EPG_ADR_BASE
```

4.3 パラメータ保存/読込(設定ファイル,レジストリ)

簡単パラメータ開始関数 EP_OpenParamTable 6.1 節により、パラメータの値を保存/読込する対象を指定します。
文字列先頭を特定のものにした場合は、レジストリを対象とします。それ以外は、ファイルが対象です。

4.3.1 ファイル

ファイルを対象とする場合は、次の様に指定してください。

・ファイル例
iniFile(C 言語指定) : "c:¥¥test¥¥sample.ini"
セクション名 : "sec1"
キー名 : "key1"
↓
ファイル名 : "c:¥test¥sample.ini"

```
[sec1]
key1=2
```

4.3.2 レジストリ

レジストリへの出力は、バイナリ型のみサポートしています。

レジストリ対象文字列 :
"HKEY_CLASSES_ROOT","HKEY_CURRENT_USER","HKEY_LOCAL_MACHINE",
"HKEY_USERS","HKEY_CURRENT_CONFIG"

このときセクション名は、ファイル名で指定したキーからのサブキー名になります。
これにより、ファイルアクセスと互換性を持たせています。

・レジストリ例(レジストリのフルパス指定)
iniFile(C 言語指定) : "HKEY_CURRENT_USER¥¥Software¥¥para_tst"
セクション名 : "sec1"
キー名 : "key1"
↓
レジストリキー名 : "HKEY_CURRENT_USER¥Software¥para_tst¥sec1"
サブキー名 : "key1"

ただし Windows OS に深く関わる、以下のレジストリに対するアクセスは禁止しています。
・アクセス禁止レジストリ
"HKEY_CURRENT_USER¥Software¥Microsoft¥Windows" (以下のキー全て)

レジストリには、Windows OS に関わる情報が多く含まれます。その値を書き換えると、PC が異常な動作をする場合があります。このため、" 簡単パラメータ" では一部については書換えを禁止していますが、レジストリの管理・操作は自己責任で行って下さい。

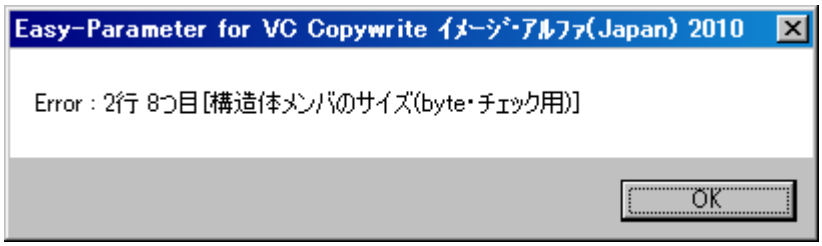
4.4 ファイルアクセス無効

セクション名、もしくはキー名をヌル文字にすると、ファイルに対する入出力を行いません。
画面とテーブルとのやり取りは行いますので、ファイル入出力が不要な場合や、独自形式で入出力を行う場合は利用してください。
なお、全てのパラメータがファイルアクセス無効の場合も、入出力ファイル名は指定してください。

```
{ EP_TYPE_ED_LONG, "", "", "0", "0",    "3",    EPG_ADDR( sample01 ), IDC_EDT_SAMPLE01 },
```

4.5 簡単パラメータテーブルの設定ミス

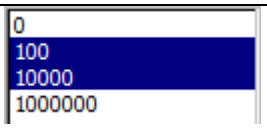
簡単パラメータ開始時に、設定したパラメータテーブルの設定ミスを確認した場合は、次のメッセージを表示します。

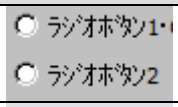
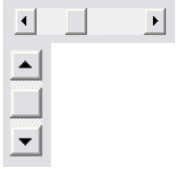


メッセージに従い、設定したパラメータテーブルを確認してください。
なお、設定ミスが見つからないが正常動作していない場合は、良くご確認の上で、その部分のコードを添付しイメージ・アルファに問い合わせてください。

5. 機能

5.1 搭載済み

ｺﾝﾄﾛｰﾙ	long 型	long 型以外	コントロール	備考
ﾃｷｽﾄ ﾎﾞｯｸｽ / ｽﾀﾃｨｯｸ ﾃｷｽﾄ	○	float 型		
		float 型指数表記 (小数点以下 2 桁)	同上	例) “1.00e-001”
		char 文字列型	同上	char[]のサイズで最大配列の自動制限 (4.2 節) 例) ″C:¥IA_CLASS¥IAESP″
		double 型	同上	
		double 型指数表記 (小数点以下 2 桁)	同上	例) “1.00e-001”
ﾁｪｯｸ ﾎﾞｯｸｽ	全 bit 対応 (0～31)	ー		各ビット割付け (初期・最大・最小値・保存/読込は、_0 のみ有効)
ｺﾝﾎﾞ ﾎﾞｯｸｽ / ﾘｽﾄ ﾎﾞｯｸｽ	○	ー		ﾘｿｰｽﾋﾞｭｰでの変更 ・ソート無効 ｺﾝﾎﾞﾎﾞｯｸｽ - ﾘｿｰｽﾋﾞｭｰでの変更 ・ﾄﾞﾛｯﾌﾟﾀﾞｳﾝﾘｽﾄ ・右側の[▼]をクリックして高さ変更 ※指定の最小～最大範囲を選択可 long 型指定の場合は、最小～最大値の全整数から選択できます。 float 型, double 型, char 文字列型指定の場合は､汎用ポインタでリスト内文字列を指定します。次の様にして下さい。 ″1.00¥n2.10¥n3.20″
		float 型 (設定値は文字列で指定)		
		char 文字列型		
		double 型		
	ー	char 文字列指定		ﾘｿｰｽﾋﾞｭｰでの変更 ・ソート無効 ｺﾝﾎﾞﾎﾞｯｸｽ - ﾘｿｰｽﾋﾞｭｰでの変更 ・ﾄﾞﾛｯﾌﾟﾀﾞｳﾝﾘｽﾄ ・右側の[▼]をクリックして高さ変更 リストの上から､0,1,2...と数値を取得/設定します。最小・最大の制限無効。 汎用ポインタで文字列を指定します｡基本的に文字数制限はありません。 ″無効¥n レベル 1¥n レベル 2″ ↑↓ ↑↓ ↑↓ 0 1 2
ﾘｽﾄ ﾎﾞｯｸｽ (ﾏﾙﾁ 選択)	ー	Unsigned char 配列指定		ﾘｿｰｽﾋﾞｭｰでの変更 ・ソート無効 ・Selection : マルチ リストの上から､[0],[1],[2]...と数値(0:非選択/1: 選択)を取得/設定します。最小・最大の制限無効。 汎用ポインタで文字列を指定します｡基本的に文字数制限はありません。 ″無効¥n レベル 1¥n レベル 2″ ↑↓ ↑↓ ↑↓ [0] [1] [2]

ｺﾝﾄﾛｰﾙ	long 型	long 型以外	コントロール	備考
ﾗｼﾞｵ ﾎﾞﾀﾝ	○ (選択なし=-1,選択≧0)	ー		ﾘｿｰｽﾋﾞｭｰでの変更 ・0 番目のデータで、Group=True ｺﾝﾄﾛｰﾙ ID ﾃｰﾌﾞﾙを定義してください。
ｽﾗｲﾀﾞ	○	float 型(1 倍, 0.1 倍,0.01 倍) double 型(1 倍, 0.1 倍,0.01 倍)		WM_HSCROLL、WM_VSCROLL イベントに EP_SetScrollDlgST(6.4 節)関数で、テーブル次のテキストボックスに反映可 ・ｽｸﾛｰﾙ処理 6.4 節を参照
ｽｸﾛｰﾙ ﾊﾞｰ	○	float 型(1 倍, 0.1 倍,0.01 倍) double 型(1 倍, 0.1 倍,0.01 倍)		縦・横ともに、同パラメータタイプ ・EP_TYPE_SCRL_LONG など WM_HSCROLL、WM_VSCROLL イベントに EP_SetScrollDlgST(6.4 節)関数でのスクロール処理必須 ・ｽｸﾛｰﾙ処理 6.4 節を参照

※ｺﾝﾎﾞﾎﾞｯｸｽ、ﾘｽﾄﾎﾞｯｸｽ、ﾗｼﾞｵﾎﾞﾀﾝを利用する際は、備考欄のﾘｿｰｽﾋﾞｭｰでの変更を必ず行ってください。
登録ﾃｰﾌﾞﾙ例は、**EP_OpenParamTable** 6.1 節の機能欄、もしくはｻﾝﾌﾟﾙﾌﾟﾛｸﾞﾗﾑ” AppSamp” を参照してください。

5.2 未搭載

・設定ファイル高速保存機能
設定ファイルを対象として入出力を行う場合は、出力数が多くなると、SSD, CF などのデバイスへの出力に極めて時間が掛かります。
またこのため、それらのデバイスの書換可能回数を浪費する事になります。
簡単パラメータではこの状態を緩和するため、高速・書換回数低減の保存機能を搭載予定です。

なおレジストリに対して入出力を行う場合は、OS に依存するためこの対応は行いません。

5.3 全パラメータタイプ定義

```
enum EP_ParamType { /* パラメータタイプ */
    /* 対象EditControl / TextControl */
    EP_TYPE_ED_LONG = 0x0, /* long型 */
    EP_TYPE_ED_FLOAT,      /* float型 */
    EP_TYPE_ED_FLT_ENO2,   /* float型・指数表記 */
    EP_TYPE_ED_STRING,     /* char文字列型 */
    EP_TYPE_ED_DOUBLE = 0x80, /* double型 */
    EP_TYPE_ED_DBL_ENO2,       /* double型・指数表記 */

    /* 対象CheckBox */
    /* CheckBox指定は、long型bitの下位n bitで指定する。*/
    EP_TYPE_CHKBOX_0 = 0x1000, /* long型0bit */ /* iniファイルとの入出力のためテーブル中に必須(def, min, max値有効、他CheckBoxは無効) */
    EP_TYPE_CHKBOX_1, EP_TYPE_CHKBOX_2, EP_TYPE_CHKBOX_3, EP_TYPE_CHKBOX_4, EP_TYPE_CHKBOX_5, /* long型 1～5bit */
    EP_TYPE_CHKBOX_6, EP_TYPE_CHKBOX_7, EP_TYPE_CHKBOX_8, EP_TYPE_CHKBOX_9, EP_TYPE_CHKBOX10, /* long型 6～10bit */
    EP_TYPE_CHKBOX11, EP_TYPE_CHKBOX12, EP_TYPE_CHKBOX13, EP_TYPE_CHKBOX14, EP_TYPE_CHKBOX15, /* long型11～15bit */
    EP_TYPE_CHKBOX16, EP_TYPE_CHKBOX17, EP_TYPE_CHKBOX18, EP_TYPE_CHKBOX19, EP_TYPE_CHKBOX20, /* long型16～20bit */
    EP_TYPE_CHKBOX21, EP_TYPE_CHKBOX22, EP_TYPE_CHKBOX23, EP_TYPE_CHKBOX24, EP_TYPE_CHKBOX25, /* long型21～25bit */
    EP_TYPE_CHKBOX26, EP_TYPE_CHKBOX27, EP_TYPE_CHKBOX28, EP_TYPE_CHKBOX29, EP_TYPE_CHKBOX30, /* long型26～30bit */
    EP_TYPE_CHKBOX31, /* long型31bit */

    /* 対象RadioButton */
    EP_TYPE_RADBTN   = 0x1080, /* long型: 対象radio button */

    /* 対象ComboBox / ListBox */
    EP_TYPE_CMB_LONG = 0x1101, /* long型: 数値対応 */
    EP_TYPE_CMB_FLOAT,         /* float型 */
    EP_TYPE_CMB_STRING,        /* char文字列型 */
    EP_TYPE_CMB_STRING_ID,     /* long型- 文字列(項目の上から、0,1,2...とlong型で取得) */
    EP_TYPE_CMB_SEL_LONG,       /* long型: 汎用ポインタ指定数値対応("1¥n100¥n10000¥n100000"と指定した数値を取得) */
    EP_TYPE_CMB_DOUBLE = 0x1180,  /* double型 */

    /* 対象ListBox : マルチ選択対応 */
    EP_TYPE_LST_MULTI_UCH = 0x11c0, /* UChar配列型: ("0;2",0,"1¥n100¥n10000", ';'はIndex初期値、設定ﾌｧｲﾙの区切りとする), */
    EP_TYPE_LST_MULTI_STR,          /* 未・文字列配列型: ("100;10000",0,"1¥n100¥n10000", ';'は初期値、設定ﾌｧｲﾙの区切りとする), */

    /* 対象SliderControl */
    EP_TYPE_SLID_LONG = 0x1200,  /* long型: インデックス対応(最小値～最大値の数値指定) */
    EP_TYPE_SLID_FLOAT,          /* float型・1刻み */
    EP_TYPE_SLID_FLT_D10,        /* float型・0.1刻み */
    EP_TYPE_SLID_FLT_D100,       /* float型・0.01刻み */
    EP_TYPE_SLID_DOUBLE = 0x1280, /* double型・1刻み */
    EP_TYPE_SLID_DBL_D10,         /* double型・0.1刻み */
    EP_TYPE_SLID_DBL_D100,        /* double型・0.01刻み */

    /* 対象ScrollBar (縦・横共通: EP_SetScrollDlg関数の実行必須) */
    EP_TYPE_SCRL_LONG = 0x1300,  /* long型: インデックス対応(最小値～最大値の数値指定) */
    EP_TYPE_SCRL_FLOAT,          /* float型・1刻み */
    EP_TYPE_SCRL_FLT_D10,        /* float型・0.1刻み */
    EP_TYPE_SCRL_FLT_D100,       /* float型・0.01刻み */
    EP_TYPE_SCRL_DOUBLE = 0x1380, /* double型・1刻み */
    EP_TYPE_SCRL_DBL_D10,         /* double型・0.1刻み */
    EP_TYPE_SCRL_DBL_D100,        /* double型・0.01刻み */


    EP_TYPE_CHK_MAX, /* 最大値チェック用*/
};
```

6. 関数一覧

表 6-1 簡単パラメータ関数一覧

節番号	関数名	種類	備考
6.1	EP_OpenParamTable	簡単パラメータ開始	
	EP_CloseParamTable	簡単パラメータ終了	
6.2	EP_LoadInifile	構造体の初期化 or ファイル読込み	
	EP_SaveInifile	構造体のファイル保存	
6.3	EP_SetParamToDlg	パラメータ画面に構造体の値セット	
	EP_GetParamFromDlg	パラメータ画面から構造体に値セット	
6.4	EP_SetScrollDlg	スクロール処理(スライドバー対応・スクロールバー対応必須)	WM_HSCROLL, WM_VSCROLL イベントに対応
6.5	EP_SetListToDlg	リストボックス, コンボボックスへのリスト設定	

簡単パラメータを使用するには、図 6-1 の様に、開始と終了の間に、他の簡単パラメータ関数を置いてください。

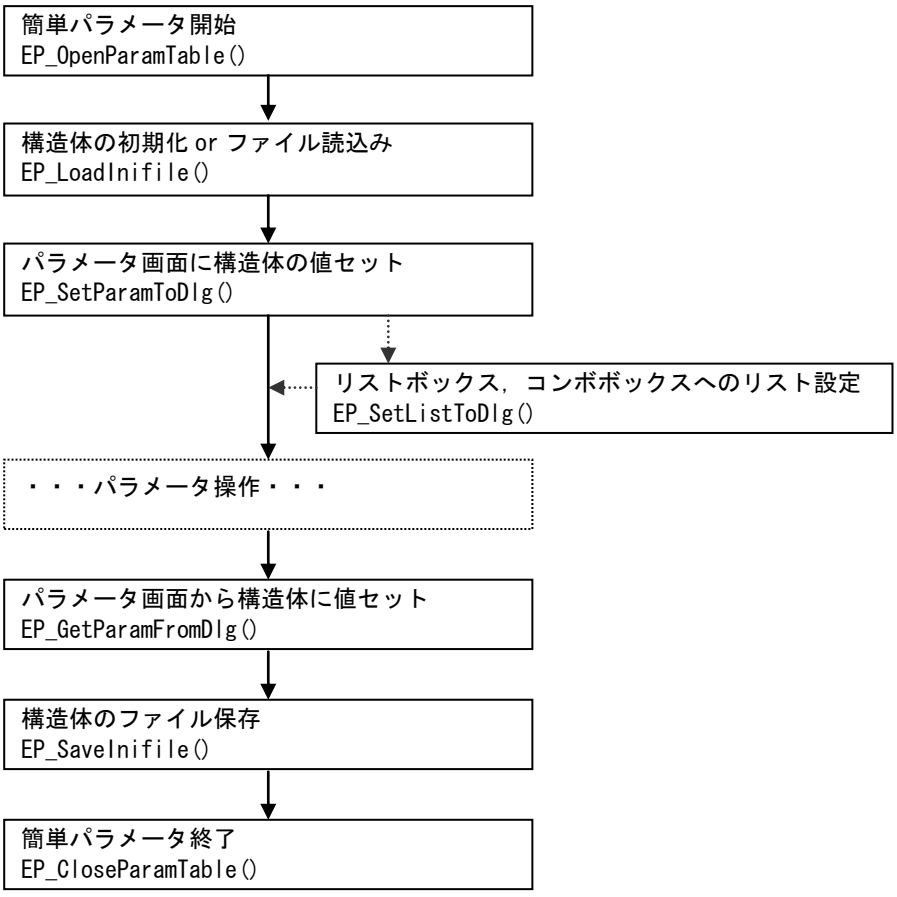


図 6-1 IALIB 通常処理

6.1 簡単パラメータ開始，終了 [EP_OpenParamTable, EP_CloseParamTable]

インタフェース

```
int EP_OpenParamTable(
     int tableNum, EP_ParamTable *paramTable, char *iniFile, long flag
 );
int EP_CloseParamTable( int ep_id );
```

パラメータ

名称	タイプ	入/出	許容値	意味
tableNum	int	入力	1 ～	パラメータ数※
paramTable	EP_ParamTable *	入力		パラメータテーブル(機能参照)
iniFile	char *	入力		設定ファイル(レジストリ対応)
flag	long	入力	0	各種フラグ 将来拡張用(必ず、0にしてください。)
ep_id	int	入力		パラメータ ID

※通常、次の様に定義し、テーブルの拡張に応じて値が追随する様にしてください。

```
sizeof(g_EP_ParamTbl) / sizeof(EP_ParamTable)
```

リターン値

> 1～64　：パラメータ ID

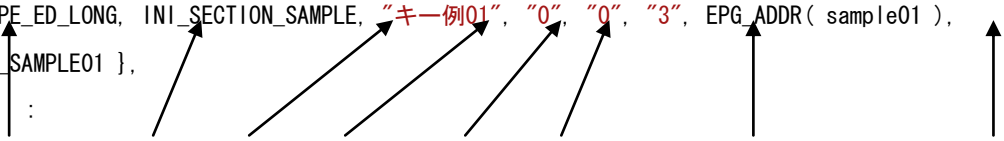
< 0　　：エラー値

機能

最大 64 種類の簡単パラメータのテーブルを設定します。戻り値のパラメータ ID で、設定済のテーブルを指定します。

●テキストボックス - int, float 型指定

```
static EP_ParamTable g_EP_ParamTbl[] = {
{ EP_TYPE_ED_LONG, INI_SECTION_SAMPLE, "キー例01", "0", "0", "3", EPG_ADDR( sample01 ),
IDC_EDT_SAMPLE01 },
        :
};
```

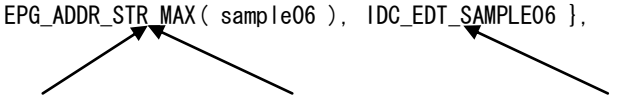


ﾊﾟﾗﾒｰﾀﾀｲﾌﾟ, ｾｸｼｮﾝ名, ｷｰ名, ﾃﾞﾌｫﾙﾄ値, 最小値, 最大値, 構造体ﾒﾝﾊﾞへのｲﾝﾃﾞｯｸｽ, ﾀﾞｲｱﾛｸﾞのｺﾝﾄﾛｰﾙ ID

●テキストボックス - char 文字列型

パラメータタイプ, セクション名, キー名, デフォルト値, 最小値(無効)

```
{ EP_TYPE_ED_STRING,  INI_SECTION_SAMPLE, "キー例06", "C:¥¥IA_CLASS¥¥IAESP", "0",
EPG_ADDR_STR_MAX( sample06 ), IDC_EDT_SAMPLE06 },
```



最大値, 構造体メンバへのインデックス, ダイアログのコントロール ID

最大値, 構造体メンバへのインデックスを同時に行う#define 文を提供しています。

最大値の指定が他と異なるため、必ずこの GET_ADDR_STR_MAX マクロを利用して下さい。

●ラジオボタン

複数のコントロールは、char配列を利用して渡します。

まずコントロールID配列を定義し、2段階で設定する必要があります。

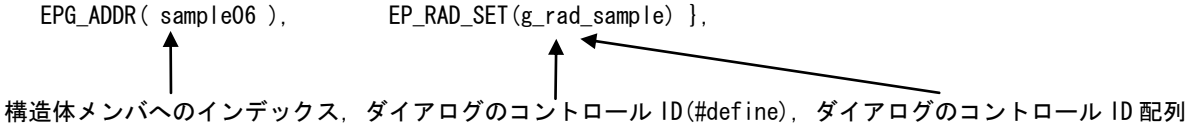
デフォルト値/最小値を"-1"にすると、未選択状態になります。

ダイアログのコントロールID配列

```
static char g_rad_sample[] = {EP_RAD(IDC_RAD_SAMPLE1), EP_RAD(IDC_RAD_SAMPLE2)};
```

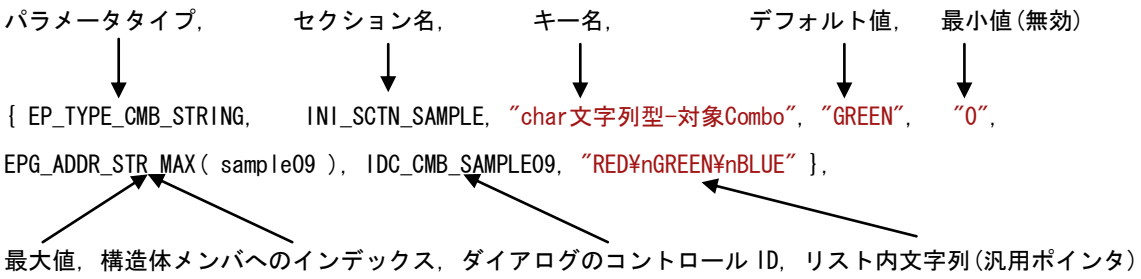
パラメータタイプ, セクション名, キー名, デフォルト値,最小値(無効),最大値(無効)

```
{ EP_TYPE_RADBTN,    INI_SCTN_SAMPLE,  "int 型・ﾗｼﾞｵﾎﾞﾀﾝ",      "-1",    "-1",    "1",
   EPG_ADDR( sample06 ),      EP_RAD_SET(g_rad_sample) },
```

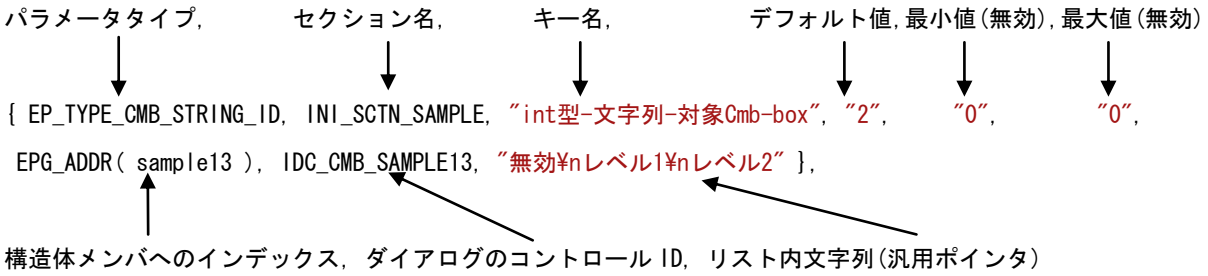


構造体メンバへのインデックス, ダイアログのコントロール ID(#define), ダイアログのコントロール ID 配列

●コンボボックス/リストボックス - char 文字列型



●コンボボックス - int 型-文字列



●設定ファイル(レジストリ対応)

設定ファイル名を指定して、指定のファイルに対して入出力が可能です。4.3 節を参照ください。

設定ファイル名に対してレジストリのキー名を指定して、レジストリにアクセスすることも可能です。

6.2 設定ファイル読込み，設定ファイル保存 [EP_LoadInifile, EP_SaveInifile]

インタフェース

```
int EP_LoadInifile (
     int ep_id, void *paramRec
 );
```

※EP_SaveInifile も同じパラメータ

パラメータ

名称	タイプ	入/出	許容値	意味
ep_id	int	入力	1 ～64	パラメータ ID
paramRec	void *	入力		パラメータテーブル(汎用ポインタ)

リターン値

< 0 : エラー値

機能

登録した設定ファイルと簡単パラメータのテーブルに従い、ファイルとの入出力を行います。

① EP_LoadInifile
　設定ファイル　　→　パラメータテーブル
② EP_SaveInifile
　パラメータテーブル　→　設定ファイル

本コマンドは、簡単パラメータ開始とパラメータテーブルの実体確保が完了していれば、どこで行っても問題ありません。

例）簡単パラメータ開始直後、システムのパラメータテーブルテーブルに EP_LoadInifile でロード。

6.3 画面への設定，画面から取得 [EP_SetParamToDlg，EP_GetParamFromDlg]

インタフェース

```
int EP_SetParamToDlg (
     int ep_id, HWND hwnd, void *paramRec
);
```

※EP_GetParamFromDlg も同じパラメータ

パラメータ

名称	タイプ	入/出	許容値	意味
ep_id	int	入力	1 ～64	パラメータ ID
hwnd	HWND	入力		画面のウィンドウハンドル
paramRec	void *	入力		パラメータテーブル(汎用ポインタ)

リターン値

< 0 : エラー値

機能

登録した画面のコントロールと簡単パラメータのテーブルに従い､画面コントロールとの入出力を行います。

① EP_SetParamToDlg

パラメータテーブル → 画面コントロール

② EP_GetParamFromDlg

画面コントロール → パラメータテーブル

画面のウィンドウハンドルを引数として渡す必要があります。

このため本コマンドは､画面のウィンドウハンドルが有効となる､OnInitDialog()～DestoryWindow()の間で行う必要があります。

6.4 スクロール処理(スライドバー対応・スクロールバー対応必須) [EP_SetScrollDlg]

インタフェース

```
int EP_SetScrollDlg (
   int ep_id, HWND hwnd, HWND hwndBar, int nSBCode, UINT nPos, void *paramRec, int *data, float *dataF
 );
```

パラメータ

名称	タイプ	入/出	許容値	意味
ep_id	int	入力	1 ～64	パラメータ ID
hwnd	HWND	入力		画面のウィンドウハンドル
hwndBar	HWND	入力		バーのウィンドウハンドル
nSBCode	int	入力		スクロール要求を示すスクロール バーのコード
nPos	UINT	入力		スクロール バーのコードが SB_THUMBPOSITION, SB_THUMBTRACK の場合の、スクロール ボックス現在位置
paramRec	void *	入力		パラメータテーブル(汎用ポインタ)
data	int *	出力		内部 int 値
dataF	float *	出力		内部 float 値

リターン値

< 0 : エラー値

機能

登録した画面のコントロールと簡単パラメータのテーブルに従い､画面コントロールとの入出力を行います。

画面、およびスライダ・スクロールバーのウィンドウハンドルを引数として渡す必要があります。
このため本コマンドは、OnInitDialog()～DestoryWindow()の間の下記イベントで行う必要があります。
・WM_HSCROLL
・WM_VSCROLL

パラメータテーブルで、スライダ・スクロールバーのテーブルの後に、一部テキストボックスを置いた場合は、自動的にテキストボックスの内容が更新されます。パラメータタイプとコントロール ID 以外の各種データは、必ず一致させて下さい。

```
{EP_TYPE_SLID_FLT_D100,INI_SCTN_SAMPLE,"ｽﾗｲﾄﾞﾊﾞｰ(float)-0.01刻み","0.2","0.0","1.1",EPG_ADDR(sample08),IDC_SLID_SAMPLE08},
{EP_TYPE_ED_FLOAT,     INI_SCTN_SAMPLE,"ｽﾗｲﾄﾞﾊﾞｰ(float)-0.01刻み","0.2","0.0","1.1",EPG_ADDR(sample08),IDC_EDT_SAMPLE08},
{EP_TYPE_SCRL_LONG,INI_SCTN_SAMPLE, "横ｽｸﾛｰﾙﾊﾞｰ(整数)", "10", "3", "30", EPG_ADDR( sample11 ),  IDC_SCRL_H_SAMPLE11},
{EP_TYPE_ED_LONG,  INI_SCTN_SAMPLE, "横ｽｸﾛｰﾙﾊﾞｰ(整数)", "10", "3", "30", EPG_ADDR( sample11 ),  IDC_EDT_SAMPLE11},
{EP_TYPE_SCRL_LONG,INI_SCTN_SAMPLE, "縦ｽｸﾛｰﾙﾊﾞｰ(整数)", "15", "3", "30", EPG_ADDR( sample12 ),  IDC_SCRL_V_SAMPLE12},
{EP_TYPE_ED_LONG,  INI_SCTN_SAMPLE, "縦ｽｸﾛｰﾙﾊﾞｰ(整数)", "15", "3", "30", EPG_ADDR( sample12 ),  IDC_EDT_SAMPLE12},
```

6.5 リストボックス，コンボボックスへのリスト設定 [EP_SetListToDlg]

インタフェース

```
int EP_SetListToDlg (
     int ep_id, int dlg_no, char *listTbl, void *paramRec
);
```

パラメータ

名称	タイプ	入/出	許容値	意味
ep_id	Int	入力	1 ～64	パラメータ ID
dlg_no	Int	入力		コントロール ID
listTbl	char *	入力		リスト内文字列を指定します。次の様にして下さい。 "1.00¥n2.10¥n3.20"
paramRec	void *	入力		パラメータテーブル(汎用ポインタ)

リターン値

< 0 : エラー値

機能

リストボックス、コンボボックスのリスト文字列を変更します。

簡単パラメータ開始 6.1 節で登録したテーブルと、画面への設定 6.3 節を行った画面、及びコントロール ID とパラメータテーブルを引数とします。

この関数は、実行直後に画面に反映します。
このため本コマンドは、OnInitDialog()～DestoryWindow()の間のイベントで行う必要があります。
通常は、画面への設定 6.3 節の後に実行してください。

●対応済パラメータタイプ

EP_TYPE_CMB_STRING, EP_TYPE_CMB_FLOAT, EP_TYPE_CMB_STRING

7. エラーー覧

関数内部でエラーを確認、発生した場合は、戻り値として負の値を返します。
これらは、デフォルトでは次のファイルにあります。

“C:¥IA_CLASS¥IAESP” フォルダ

- “ep_prot.h”

```
enum EP_ParamError {
        EPERR_PARAM_MNB_TBL = -10000,     /* 最大画面パラメータ数エラー */
        EPERR_PARAM_TBL     = -10001,     /* パラメータ管理構造体エラー */
        EPERR_PARAM_INIFILE = -10002,     /* 設定ファイル名 */
        EPERR_PARAM_ID      = -10003,     /* パラメータID範囲外 */

        EPERR_PARAM_DLG_INP = -10004,     /* ダイアログ入力値範囲外 */

        EPERR_PARAM_CTLID   = -10005,     /* コントロールID範囲外 */
        EPERR_PARAM_MALLOC  = -10006,     /* 内部メモリ確保エラー */



        EPERR_LICENSE       = -20000,     /* ライセンスエラー */
};
```