

標準 S L M 通信ライブラリリファレンスマニュアル

# S L M - C O M N T 説 明 書

Ver 1.2  
2010.11.01

# - 目次 -

1 . 形式 .....	3
2 . 製品構成 .....	3
3 . 概要 .....	3
4 . 関連資料 .....	3
5 . 使用環境 .....	3
6 . インストール .....	3
7 . ライブラリ使用上の注意事項 .....	4
8 . 関数使用例 .....	4
8 - 1 MS - VC 以外の C 言語処理系での利用方法 .....	4
9 . 関数仕様 .....	6
9 - 1 . 通信 .....	6
I n i t C o m m P r o c .....	6
Q u i t C o m m P r o c .....	7
S e n d D a t a .....	8
R e c e i v e D a t a .....	9
S e n d C o m m a n d .....	10
9 - 2 . タイマー .....	11
S e t T i m e O u t .....	11
C h e c k T i m e O u t .....	11
10 . 改版履歴 .....	12
10 - 1 . ( Ver.1.0 Ver.1.1)2004.06.09 .....	12
10 - 2 . ( Ver.1.1 Ver.1.2)2010.11.01 .....	12

## 1. 形式

S L M C O M - N T

## 2. 製品構成

S L M C O M N T	.	D L L	.....	ダイナミックリンクライブラリーファイル
S L M C O M N T	.	L I B	.....	インポートライブラリーファイル
S L M C O M N T	.	H	.....	インクルードファイル(関数定義)
S L M D A T A	.	H	.....	インクルードファイル(構造体定義)
S Y N C D E F	.	H	.....	インクルードファイル(マクロ定義)
S L M C O M A P I	.	T X T	.....	ビジュアルベーシック用データ型・関数定義

## 3. 概要

本ライブラリ(標準 S L M シリーズ対応通信ライブラリ)は、パソコンと S L M コントローラ間で、R S - 2 3 2 C / U S B を利用して、通信を行うための処理をライブラリ化したものです。本ライブラリの機能は、C 言語の関数形式で呼び出して、利用できます。

本ライブラリを利用して以下のような処理が行えます。

- ・データ送信
- ・データ受信
- ・動作指示

各データの詳細は送受信データ説明書を参照下さい。

## 4. 関連資料

「S L M - 4 0 0 0 ユーザーズマニュアル」	( T B 0 0 - 0 8 0 0 )
「標準 S L M 対応 セッティング P C マニュアル」	( T B 0 0 - 0 8 0 2 )
「標準 S L M 対応 R O M S W 設定ソフトマニュアル」	( T B 0 0 - 0 8 0 1 )
「標準 S L M 対応 送受信データ説明書」	( T B 0 0 - 0 8 0 4 )
「標準 S L M 対応 T コード変換ライブラリリファレンスマニュアル」	( T B 0 0 - 0 8 0 5 )
「標準 S L M 対応 G コード変換ライブラリリファレンスマニュアル」	( T B 0 0 - 0 8 0 6 )

## 5. 使用環境

本ライブラリが対応するパソコン及び、C コンパイラは以下の通りです。

対応パソコン	.....	Windows XP/Vista/7が動作する x 8 6 C P U 搭載機
対応 C コンパイラ	.....	Visual C++ Ver 6 . 0 以降

## 6. インストール

D L L ファイルをプログラム検索パスの通っているディレクトリ(フォルダ)に格納して下さい。

U S B バス接続で使用する場合は、デバイスドライバのインストールが必要です。  
詳細は「S L M - 4 0 0 0 ユーザーズマニュアル(TB00-0800)」< 導入編 > “ 3 . U S B  
ドライバーのインストール ” を参照下さい。

## 1.1 7. ライブラリ使用上の注意事項

SLMシリーズ用通信ライブラリを使用して、アプリケーションを作成するときは、以下の事を注意して下さい。

- ・ USB通信はWindows XP/Vista/7で使用可能です。  
Windows NTでは使用できません。
- ・ プログラムデータの作成には、弊社プログラム処理ライブラリが必要です。
- ・ 複数のアプリケーションで通信を利用する場合、以下に注意する必要があります。
  - <RS232C通信>
    - 1) 1つのCOMポートに対しては、1つのアプリケーションしかアクセスすることはできません。
    - 2) 1台のSLMに対して、最大2つのCOMポートから同時にアクセスできます。  
但し、この場合は別途に232通信用基板/ケーブル(AS232/CB-C-012)をご購入いただく必要があります。
  - <USB通信>
    - 1) 1つのUSBに対しては、1つのアプリケーションしかアクセスすることはできません。
  - <共通>
    - 1) 1つのアプリケーションから最大4つの通信を行うことができます。  
例1) 通信1: COM1-SLM1、通信2: COM2-SLM2、通信3: COM3-SLM3、通信4: COM4-SLM4  
例2) 通信1: USB-SLM1、通信2: COM1-SLM1、通信3: USB-SLM2、通信4: COM2-SLM2
    - 2) 複数のアプリケーションから、複数の通信形態を併用して1台のSLMにアクセスすることが可能です。  
例) アプリケーション1: RS232C-SLM1  
アプリケーション2: RS232C-SLM1  
アプリケーション3: USB-SLM1
    - 3) 1台のSLMに対して、複数の通信を利用してデータ書き込みや動作指示を行う場合、それぞれのデータ書き込みや動作指示が干渉することがありますので、運用で干渉を回避する必要があります。

## 8. 関数使用例

本ライブラリの使用法は、以下の2通りがあります。

- ・ インポートライブラリを使用して関数をインポートする。  
ライブラリに付属のインポートライブラリをアプリケーションの作成時にリンクします。  
この方法は、使用言語がMS-V C++ Ver 6.0の時のみ有効です。
- ・ DLLロード関数を使用して関数をインポートする。  
DLLロード関数を使用してDLLをメモリにロードし、関数アドレスを取得して関数を呼び出します。  
この方法は、ウィンドウズ上で動作する言語処理系(32ビット版)全てで有効です。  
詳細は次項を参照して下さい。

### 1.1 8-1 MS-V C以外のC言語処理系での利用方法

MS-V C以外のC言語処理系でDLLを使用する場合は、以下の手順で行います。

1. DLLのロード
2. DLL内の任意の関数のアドレス取得  
⋮  
DLL内の関数の利用  
⋮
3. DLLのアンロード

関数アドレス取得時に指定する関数名は、本説明書(ヘッダーファイル)の関数名と異なります。

アドレス取得時の関数名のフォーマットは、ヘッダーファイルの関数名の前に"\_"(アンダーバー)を付け、関数名の後に@、その後に引数の数×4を付けます。  
具体的には以下の通りとなります。

\_関数名@引数の数×4

例 InitCommProc(PsxDEF psxdef, int \*phCom)の場合  
関数名 ..... \_InitCommProc@8

これらの定義がSlmcomnt.h内にあるので、独自の処理を作成する場合は参考にして下さい。

関数アドレス取得時に指定する関数名 / 関数ポインタのデータ型は、Slmcomnt.h内で定義しています。Slmtcnv.h をインクルードする前に LOADLIBRARY\_MCDLL を定義(define)することによってこれらの定義を使用できるようになります。(LOADLIBRARY\_MCDLLを定義しなかった場合はインポートライブラリを使用する設定になります。)

以下の例は、通信ライブラリの関数取得例を示します。

```
#define LOADLIBRARY_MCDLL          // テクノ製ライブラリ明示的リンク指定
#include "Slmcomnt.h"             // 通信ライブラリ宣言

// グローバルデータ定義
HINSTANCE      hComInst;          /* 通信ライブラリハンドル          */
INITPROC       *InitCommProc;    /* 通信初期化関数へのポインタ      */
QUITPROC       *QuitCommProc;   /* 通信終了処理関数へのポインタ    */

// 通信ライブラリ関数ロード処理
int LoadSlmComm(void)
{
    // D L L のロード
    if(NULL == (hComInst = LoadLibrary("SLMCOMNT.DLL"))){
        return 0;
    }
    // D L L が正常にロードできた場合
    // D L L 関数ポインタの取得
    InitCommProc = (INITPROC *)GetProcAddress(hComInst, FNINITPROC);
    QuitCommProc = (QUITPROC *)GetProcAddress(hComInst, FNQUITPROC);

    if((NULL == InitCommProc) || (NULL == QuitCommProc)){
        // 関数アドレス取得が失敗した場合
        // D L L の解放
        FreeLibrary(hComInst);
        return 0;
    }

    return 1;
}
```

## 9 . 関数仕様

### 9 - 1 . 通信

# InitCommProc

#### 機能

通信インタフェース (RS232C または USB) の初期化及び、通信処理の組み込みを実行します。

#### プロトタイプ

```
#include "slmcomnt.h"      関数宣言に必要なインクルードファイル

int WINAPI InitCommProc(PSXDEF psxdef, int *phCom);

PSXDEF          psxdef 通信定義構造体
                  通信処理の各種定義を行います。
                  typedef struct {
                    int      fComType;           // 通信種別フラグ
                    int      nComPort;          // RS232C通信ポート番号
                    int      nLptPort;         // パラレル通信ポート
                    int      nBoardId;         // USB通信ID
                    int      fShare;           // 共有フラグ
                    int      fSxType;          // SXコントローラ種別
                    int      nSize;            // SXコントローラ定義構造体サイズ
                    long      fLogging;         // 通信ロギングフラグ
                    char      *pLogFile;       // 通信ロギングファイル名
                  } SXDEF, *PSXDEF;
```

各変数の内容は以下の通りです。

<b>fComType</b>	通信種別フラグ COMRS232C (0) : RS232C 通信 COMUSB (3) : USB 通信
<b>nComPort</b>	RS232C 通信ポート番号 1 ~ 4
<b>nLptPort</b>	パラレル通信ポート 1 ~ 4 SLMでは使用しません。1を指定して下さい。
<b>nBoardId</b>	USB 通信 ID 0 ~ 4
<b>fShare</b>	共有フラグ 0 ~ 1 1を指定して下さい。
<b>fSxType</b>	SX コントローラ種別 TYPE_SLM4000(0x20) : SLM-4000
<b>nSize</b>	SX コントローラ定義構造体サイズ 常に本構造体のサイズを設定してください。
<b>fLogging</b>	通信ロギングフラグ 内容以下の通りです。 上位の未定義ビットは全て0にして下さい。

ロギング無効  
常時ロギング  
常時ロギング (初期化/終了のみ)  
常時ロギング (データ送信のみ)  
常時ロギング (データ受信のみ)  
常時ロギング (コマンド要求のみ)  
エラーロギング  
エラーロギング (伝送エラーのみ)

## pLogFile

通信ロギングファイル名

S L Xとの通信のログをとる場合にログファイル名を指定します。

ログをとる場合、実行ファイルと同じディレクトリに以下のファイルが作成されます。

1. 「通信時ログファイル名」で指定したファイル
2. 「通信時ログファイル名」で指定したファイルのベースファイル名の最後に1~5の数字を付加したファイル
3. SLM0000.tmp

RS232C通信の場合・・・Com COMポート番号  
USB通信の場合・・・Usb ポートID番号

ログファイル(1のファイル)が512Kバイトをこえると現ログファイルはリネームされて、履歴ファイルとなります。その後、新しいログファイルを作成してロギングを継続します。

履歴ファイル名は、指定されたログファイル名のベース名に履歴番号として1~5を付加した名前です。(最大5世代)

例) ログファイル名として"TMP.LOG"を指定すると、履歴ファイルとしてTMP1.LOG~TMP5.LOGが作成されます。

int

\*phCom 通信ハンドル

初期化が正常に終了した場合に、ハンドルを格納するデータのポインタを指定します。

### 戻り値

通信処理実行ステータスとして以下の値を返します。

E\$OK : 正常終了  
E\$EMPTYHANDLE : ハンドル取得不可  
E\$PARAM : パラメータ設定異常  
E\$ERR : 初期化処理異常終了

正常に初期化が終了した場合、接続されたS P Xコントローラを識別するためのハンドルをphComの指し示すアドレスへ格納します。

# QuitCommProc

### 機能

関数InitCommProcの実行によって組み込まれた、通信処理を切り放し、組み込み前の状態に戻します。

### プロトタイプ

#include "slmcomnt.h" 関数宣言に必要なインクルードファイル

int WINAPI QuitCommProc(int hCom);

int hCom 通信ハンドル  
通信初期化処理で取得したハンドルを指定します。

### 戻り値

ステータスとして以下の値を返します。

E\$OK : 正常終了  
E\$NOHANDLE : 無効ハンドル  
E\$ERR : 終了処理異常終了

# SendData

## 機能

SLMコントローラに対して、各種データ設定を行います。

## プロトタイプ

```
#include "slmcomnt.h"      関数宣言に必要なインクルードファイル

int WINAPI SendData(int hCom, int type, short prm, DWORD size, LPVOID data);

int          hCom      通信ハンドル
                通信初期化処理で取得したハンドルを指定します。

int          type      データタイプ
                DAT_PARAMETER等、送信するデータのタイプを指定します。

short        prm       付加パラメータ
                データタイプによって、以下のデータを指定して下さい。
                以下に示されていないデータタイプについては0を指定して下さい。

                ・ DAT_PROGRAM   : プログラム番号       1 ~ 12
                ・ DAT_TASKPROG  : タスク番号           0 ~ 4
                ・ DAT_DNCDATA   : 先頭 / 継続フラグ    0 : 先頭時、 1 : 継続時

DWORD        size      送信データサイズ
                0 ~ 65535 バイト

LPVOID       data      送信データ格納バッファへのポインタ
```

本関数のデータタイプに指定できるデータについては、「標準SLM-4000対応送受信データ説明書」の「データ送受信機能」を参照して下さい。

## 戻り値

通信処理実行ステータスとして以下の値を返します。

E\$OK	:	正常終了
E\$DEVNRDY	:	デバイス未初期化
E\$PARAM	:	通信パラメータ設定異常
E\$TIME	:	タイムアウト発生
E\$RTRY	:	リトライオーバー発生
E\$MLTRTRY	:	多重リトライ発生
E\$HARDER	:	通信ハードウェアエラー
E\$PROTECT	:	送信データ書込不可
E\$SEQ	:	通信データフォーマットエラー
E\$PRGTERM	:	プログラム書込中断
E\$PRGBUFF	:	プログラムバッファオーバーフロー
E\$NOHANDLE	:	無効ハンドル



# ReceivedData

## 機能

SLMコントローラ内部の、各種データ読出を行います。

## プロトタイプ

```
#include "slmcomnt.h"      関数宣言に必要なインクルードファイル

int WINAPI ReceiveData(int hCom, int type, LPDWORD size, LPVOID data);

int          hCom      通信ハンドル
                  通信初期化処理で取得したハンドルを指定します。

int          type      データタイプ
                  DAT_PARAMETER等、受信するデータのタイプを指定します。

short       prm        付加パラメータ
                  データタイプによって、以下のデータを指定して下さい。
                  以下に示されていないデータタイプについては0を指定して下さい。

                  ・ DAT_PROGRAM      : プログラム番号   1 ~ 12
                  ・ DAT_TASKPROG     : タスク番号       0 ~ 4

LPDWORD     size      受信データサイズ格納変数へのポインタ
                  ログデータ読み出し時は、1度に読み出すデータ数をセットします。

LPVOID      data      受信データ格納バッファへのポインタ
```

本関数のデータタイプに指定できるデータについては、「標準SLM-4000対応送受信データ説明書」の「データ送受信機能」を参照して下さい。

## 戻り値

通信処理実行ステータスとして以下の値を返します。

```
E$OK          : 正常終了
E$DEVNRDY     : デバイス未初期化
E$PARAM       : 通信パラメータ設定異常
E$TIME        : タイムアウト発生
E$RTRY        : リトライオーバー発生
E$MLTRTRY     : 多重リトライ発生
E$HARDER      : 通信ハードウェアエラー
E$NEXIST      : 要求データが存在しない
E$SEQ         : 通信データフォーマットエラー
E$NEXIST      : 要求データが存在しない
E$NOHANDLE    : 無効ハンドル
```

# SendCommand

## 機能

SLMコントローラに対して、各種動作指示を行います。

## プロトタイプ

```
#include "slmcomnt.h"      関数宣言に必要なインクルードファイル
int WINAPL SendCommand(int hCom, int cmd, LPVOID data);
int          hCom      通信ハンドル
                  通信初期化処理で取得したハンドルを指定します。
int          cmd       動作指示コード
                  REQ_RESET等、動作指示のタイプを指定します。
LPVOID      data       動作パラメータ格納バッファへのポインタ
```

本関数の動作指示コードに指定できるデータについては、「標準SLM-4000対応送受信データ説明書」の「動作コマンド付加データ詳細」を参照して下さい。

## 戻り値

通信処理実行ステータスとして以下の値を返します。

E\$OK	: 正常終了
E\$DEVNRDY	: デバイス未初期化
E\$PARAM	: 通信パラメータ設定異常
E\$TIME	: タイムアウト発生
E\$RTRY	: リトライオーバー発生
E\$MLTRTRY	: 多重リトライ発生
E\$HARDER	: 通信ハードウェアエラー
E\$SEQ	: 通信データフォーマットエラー
E\$CMDNOT	: コマンド実行不可
E\$NOHANDLE	: 無効ハンドル

## 9 - 2 . タイマー

Windows が持つ、1 m s e c 間隔のインターバルタイマーをもとに、タイマーによる処理の遅延や同期、またタイムアウト処理などが可能となります。「タイマーチェック起点関数」及び「タイムアウトチェック関数」によりアプリケーションにて通信処理以外の目的で、タイマー処理を行うことができます。

# SetTimeOut

## 機能

タイムアウトチェックの起点設定を行います。

## プロトタイプ

```
#include "slmcomnt.h" 関数宣言に必要なインクルードファイル
void WINAPI SetTimeOut(LPDWORD origin);
LPDWORD origin タイムアウトチェック起点時間格納ポインタ
```

# CheckTimeOut

## 機能

タイムアウトの判定を行います。

## プロトタイプ

```
#include "slmcomnt.h" 関数宣言に必要なインクルードファイル
BOOL WINAPI CheckTimeOut(DWORD origin, DWORD limit);
DWORD origin 関数SetTimeOut()で取得したタイムアウトチェック起点時間
DWORD limit タイムリミット値(単位: 1 m s e c)
```

## 戻り値

= F A L S E : タイムアウトしていない  
= T R U E : タイムアウトしている

## 例

```
#include <conio.h>
#include "slmcomnt.h"

main()
{
    int ch;
    DWORD StartTime;

    ch = getch(); /*キー入力 */
    SetTimeOut(&StartTime); /*タイマーチェック起点設定*/
    While(!CheckTimeOut(StartTime,20L)); /*1.1秒待ち */
    getch(ch); /*入力キーコード表示 */
}
```

このプログラムは、キー入力があったから 1 . 1 秒後に、入力キーを表示します。

## 1 0 . 改版履歴

1 0 - 1 . (Ver.1.0 Ver.1.1)2004.06.09

### 表紙

Ver.1.0 Ver.1.1  
2002.04.27 2004.06.09

### 本文

項番	新ページ	内 容
7.	0803-4	[ 複数のアプリケーションで通信を利用する場合、以下に注意する必要があります。 ]の注意書きを追加
8-1	0803-5	拡張子[SLMDLL]を[MCDLL]に変更

1 0 - 2 . (Ver.1.1 Ver.1.2)2010.11.01

### 表紙

Ver.1.1 Ver.1.2  
2004.06.09 2010.11.01

### 本文

項番	新ページ	内 容
5.	0803-3	対応OSの変更。
7.	0803-3	USB通信の対応OSの変更。
9-2.	0803-11	" SetTimeout " 説明文の中のOS名を変更。