標準ＰＬＭＣ－ＭⅡＥｘ対応
通信ライブラリ
リファレンスマニュアル

ＰＬＭＥＸ－ＣＯＭＮＴ説 明 書

Ｖｅｒ１．３
2011.07.01

－目次－

１．形式

ＰＬＭＥＸＣＯＭ－ＮＴ

２．製品構成

ＰＬＭＥＸＣＯＭＮＴ　．ＤＬＬ	･･･････	ダイナミックリンクライブラリーファイル
ＰＬＭＥＸＣＯＭＮＴ　．ＬＩＢ	･･･････	インポートライブラリーファイル
ＰＬＭＥＸＣＯＭＮＴ　．Ｈ	･･･････	インクルードファイル(関数定義)
ＰＬＭＥＸＤＡＴＡ　．Ｈ	･･･････	インクルードファイル(構造体定義)
ＳＹＮＣＤＥＦ　．Ｈ	･･･････	インクルードファイル(マクロ定義)
ＰＬＭＥＸＣＯＭＡＰＩ．ＴＸＴ	･･･････	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ＰＬＭＣ－ＭⅡＥｘシリーズ対応通信ライブラリ）は、
パソコンとＰＬＭＣ－ＭⅡＥｘコントローラをＦＡＭ３のパソコンリンク機能を利用して
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

「標準ＰＬＭＣ－ＭⅡＥｘ対応 ユーザーズマニュアル」　（ＴＢ００－０９００）
「標準ＰＬＭＣ－ＭⅡＥｘ対応 セッティングＰＣマニュアル」　（ＴＢ００－０９０１）
「標準ＰＬＭＣ－ＭⅡＥｘ対応 ＲＯＭＳＷ設定ソフトマニュアル」　（ＴＢ００－０９０２）
「標準ＰＬＭＣ－ＭⅡＥｘ対応 通信ライブラリリファレンスマニュアル」（ＴＢ００－０９０３）
「標準ＰＬＭＣ－ＭⅡＥｘ対応 送受信データ説明書」　（ＴＢ００－０９０４）
「標準ＰＬＭＣ－ＭⅡＥｘ対応 Ｔコード変換ライブラリリファレンスマニュアル」
（ＴＢ００－０９０５）
「標準ＰＬＭＣ－ＭⅡＥｘ対応 Ｇコード変換ライブラリリファレンスマニュアル」
（ＴＢ００－０９０６）
「標準ＰＬＭＣ－ＭⅡＥｘ対応 サンプルラダープログラム説明書」　（ＴＢ００－０９１７）
「標準ＰＬＭＣ－４０／ＭⅡ対応 サンプルラダープログラム補足説明書」（ＴＢ００－０８８４）

※サンプルアプリケーション（サンプルソース／説明）はテクノホームページからダウンロード
してください。

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ｘ８６ＣＰＵ搭載機
対応コンパイラ　・・・　Ｖｉｓｕａｌ　Ｃ＋＋　Ｖｅｒ６．０
Ｖｉｓｕａｌ　Ｂａｓｉｃ　Ｖｅｒ６．０

６．複数台での使用

１台のパソコンから複数台のＰＬＭＣ－ＭⅡＥｘと送受信を行うことが可能です。
複数台のパソコンから１台のＰＬＭＣ－ＭⅡＥｘに送受信を行うことはできません。

７．インストール

ＤＬＬ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。

８．ライブラリ使用上の注意事項

ＰＬＭシリーズ用通信ライブラリーを使用して、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。

・ 運転プログラムの作成には、弊社プログラム処理ライブラリが必要です。

９．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。
ライブラリー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－ＶＣ＋＋ Ｖｅｒ６．０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。

９－１ ＭＳ－ＶＣ以外のＣ言語処理系での利用方法

ＭＳ－ＶＣ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Plmexcomnt.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、Plmexcomnt.h内で
定義しています。
Plmexcomnt.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 "Plmexcomnt.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ﾎﾟｲﾝﾀ               */
QUITPROC        *QuitCommProc;          /* 通信終了処理関数へのﾎﾟｲﾝﾀ             */

// 通信ライブラリ関数ロード処理
int LoadPlmComm(void)
{
    // ＤＬＬのロード
    if(NULL == (hComInst = LoadLibrary(LNCOMDLL))){
          return 0;
    }
    // ＤＬＬが正常にロードできた場合
    // ＤＬＬ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
        // ＤＬＬの解放
        FreeLibrary(hComInst);
        return 0;
    }

    return 1;
}
```

１０．関数仕様

１０－１．通信

ＩｎｉｔＣｏｍｍＰｒｏｃ

機能

通信インタフェースの初期化及び、通信処理の組み込みを実行します。

プロトタイプ

```
#include    "Plmexcomnt.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 {
     long            nSize;        // 通信初期化構造体サイズ
     short           fComType;     // 通信種別フラグ
     short           fShare;       // 共有フラグ
     long            fLogging;     // 通信ロギングフラグ
     char            *pLogFile;    // 通信ロギングファイル名
     // pclink common
     short           nCpuNo;       // CPU番号
     short           nUnitNo;      // PLMC-MⅡEx ユニット番号 [0～7]
     short           nSlotNo;      // PLMC-MⅡEx スロット番号 [1～16]
     short           Reserved;     // 予約
     // pclink Rs232c
     short           nComPort;     // ＰＣ　ＲＳ２３２Ｃ通信ポート番号
     short           nStation;     // ﾊﾟｿｺﾝﾘﾝｸﾓｼﾞｭｰﾙ ｽﾃｰｼｮﾝ番号[1～32]
     short           nComSel;      // ﾊﾟｿｺﾝﾘﾝｸﾓｼﾞｭｰﾙ 通信速度／パリティ選択
     short           fComFlag;     // ﾊﾟｿｺﾝﾘﾝｸﾓｼﾞｭｰﾙ 付加データ選択フラグ
     // pclink Ether
     unsigned char   nIpAddr[4];   // ｲｰｻﾈｯﾄﾓｼﾞｭｰﾙ ＩＰアドレス
     unsigned short  nIpPort;      // ｲｰｻﾈｯﾄﾓｼﾞｭｰﾙ  ポート番号
     short           fBinary;      // ｲｰｻﾈｯﾄﾓｼﾞｭｰﾙ バイナリモードフラグ
} SXDEF, *PSXDEF;
```

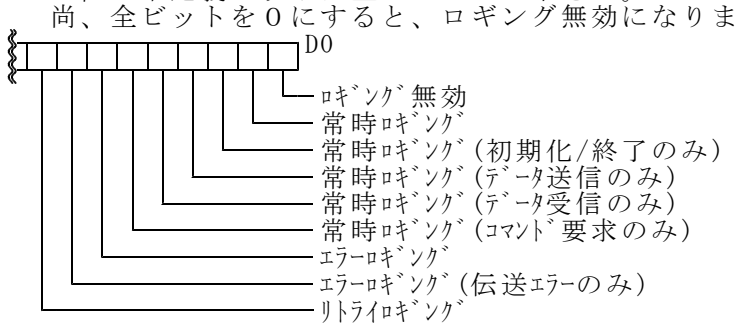
各変数の内容は以下の通りです。

nSize　構造体サイズ
　本構造体（通信定義構造体）のサイズを設定して下さい。
　sizeof():VC++/ LenB():VB 関数でサイズを取得できます。

fComType　通信種別フラグ　【0～1】
　ﾊﾟｿｺﾝﾘﾝｸ(RS232C)(0)：ＲＳ２３２Ｃ通信
　ﾊﾟｿｺﾝﾘﾝｸ(ｲｰｻﾈｯﾄ)(1)：イーサネット通信

fShare　共有フラグ　【１】
　１を指定して下さい。

fLogging　通信ロギングフラグ
　通信ロギングの設定を行います。内容は通りです。
　上位の未定義ビットは全て０にして下さい。
　尚、全ビットを０にすると、ロギング無効になります。



pLogFile　通信ロギングファイル名

ＰＬＭＣ－ＭⅡＥｘとの通信のログを保存するファイル名を指定します。
NULLを指定するとロギングが無効になります。
ファイル名を指定＆通信ロギングフラグ指定でログを有効にした場合、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．「通信ﾛｷﾞﾝｸﾞﾌｧｲﾙ名」で指定したﾌｧｲﾙ
２．「通信ﾛｷﾞﾝｸﾞﾌｧｲﾙ名」で指定したﾌｧｲﾙのﾍﾞ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
３．232C通信の場合
　　M3LnkRo.tmp
　　　　└──── ﾎﾟｰﾄ番号
　ｲｰｻﾈｯﾄ通信の場合
　　M3LnkEo.tmp
　　　　└──── Ip番号、Unit番号、slot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。

nCpuNo　CPUモジュールスロット番号　【1～4】

通信対象のFA-M3のCPUモジュール/ｱﾄﾞｵﾝCPUﾓｼﾞｭｰﾙのスロット番号を指定します。

nUnitNo　PLMC-MⅡExユニット番号　【0～7】

光FAバス等のサブユニット使用時に対象となるPLMC-MⅡExが実装されているユニット番号を指定します。

nSlotNo　PLMC-MⅡExスロット番号　【1～16】

通信対象のPLMC-MⅡExが実装されているスロット番号を指定します。

nComPort　シリアル通信ポート番号　【1～256】

ＲＳ２３２Ｃ通信に使用するシリアルポート番号を指定します。

nStation　パソコンリングモジュールステーション番号【1～32】※1

通信対象のパソコンリングモジュールのステーション番号を指定します。

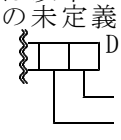
nComSel　通信速度／パリティ選択　【0～9】※1

ＲＳ２３２Ｃ通信時の通信速度とﾊﾟﾘﾃｨﾁｪｯｸの組み合わせを設定します。
設定する値は、以下から選択します。

COM_PCLCS_9600PE	(0)：	9600bps、パリティ偶数
COM_PCLCS_9600PN	(1)：	9600bps、パリティ無し
COM_PCLCS_19200PE	(2)：	19200bps、パリティ偶数
COM_PCLCS_19200PN	(3)：	19200bps、パリティ無し
COM_PCLCS_38400PE	(4)：	38400bps、パリティ偶数
COM_PCLCS_38400PN	(5)：	38400bps、パリティ無し
COM_PCLCS_57600PE	(6)：	57600bps、パリティ偶数
COM_PCLCS_57600PN	(7)：	57600bps、パリティ無し
COM_PCLCS_115200PE	(8)：	115200bps、パリティ偶数
COM_PCLCS_115200PN	(9)：	115200bps、パリティ無し

fComFlag　パソコンリンクプロトコル選択　【0～3】※1

ＲＳ２３２Ｃ通信時のパソコンリングコマンドの付加データ（プロトコル）を選択します。
内容は以下の通りです。
上位の未定義ビットは全て０にして下さい。



nIpAddr　ｲｰｻﾈｯﾄ通信時の接続先IPアドレス
ｲｰｻﾈｯﾄﾓｼﾞｭｰﾙへ接続した場合：
SW1～8で設定したアドレス
CPUﾓｼﾞｭｰﾙ(SP66-4S,SP67-6S)へ接続した場合：
CPUプロパティのEthernet設定へ設定したIPアドレス
例)169.254.205.249を設定する場合
nIpAddr[0]=169、**nIpAddr[1]**=254、
nIpAddr[2]=205、**nIpAddr[3]**=249

nIpPort　接続先ポート番号　【12289/12291】**※1**
ｲｰｻﾈｯﾄ通信時の接続先ポート番号を指定します。
以下のどちらかの値を選択して下さい。
12289：COM_PCLPT_ETHERPORT0
12291：COM_PCLPT_ETHERPORT1
ｲｰｻﾈｯﾄﾓｼﾞｭｰﾙへ接続した場合、機種によっては
ポート番号が選択できないものがあります。

fBinary　バイナリ通信フラグ　【0～1】**※1**
ｲｰｻﾈｯﾄ通信時の接続先ポートのコマンドデータ形式
(0):ASCII形式
(1):バイナリ形式
選択したポート番号のコマンドデータ形式は
ｲｰｻﾈｯﾄﾓｼﾞｭｰﾙへ接続した場合：
SW9の設定により決定
CPUﾓｼﾞｭｰﾙ(SP66-4S,SP67-6S)へ接続した場合：
CPUﾌﾟﾛﾊﾟﾃｨの上位ﾘﾝｸｻｰﾋﾞｽ設定で設定した形式

※1.各設定に関しては
「標準ＰＬＭＣ－ＭⅡＥｘ対応 セッティングＰＣマニュアル
5-4-1.ｲﾝﾀｰﾌｪｰｽ設定画面」も参照にして下さい。

通信の種別により、以下の項目の設定の必要はありません。
ｲｰｻﾈｯﾄ通信時：**nComPort、nComSel、fComFlag**
但し、**nStationには1を設定**
Rs232C通信時：**nIpAddr、nIpPort、fBinary**

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         : 異常終了
```

正常に初期化が終了した場合、接続されたＰＬＭＣ－ＭⅡＥｘ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。

【参考】
ＶＢ６．０使用時のログファイル名指定方法
【方法１】
SXDEFの初期化で
```
sdf.pLogFile = StrPtr(StrConv("test.log", vbFromUnicode)) '通信ロギングファイル名
```
と指定。
【方法２】
Plmexcomapi.txtのSXDEF構造体にありますpLogFileを
```
pLogFile As String
```
と変更し
SXDEFの初期化で
```
sdf.pLigFile = "test.log"                                    '通信ロギングファイル名
```
と指定。

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    "Plmexcomnt.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         :  異常終了
```

ＳｅｎｄＤａｔａ

機能

ＰＬＭＣ－ＭⅡＥｘ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    "Plmexcomnt.h"        関数宣言に必要なインクルードファイル

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

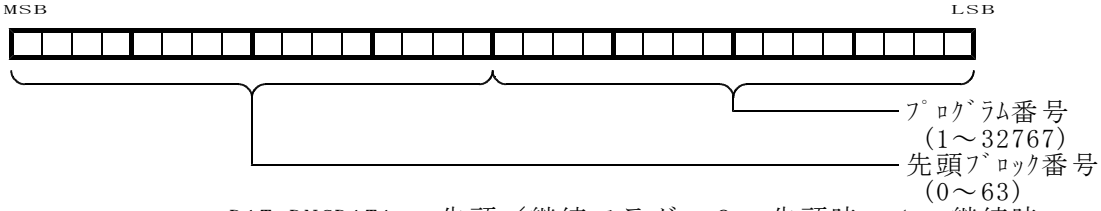
int　　*hCom*　　通信ハンドル
通信初期化処理で取得したハンドルを指定します。

short　　*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。

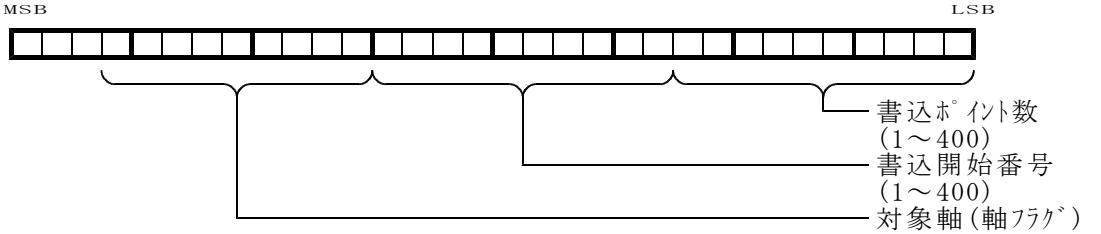
short　　*task*　　タスク番号
設定対象のタスク番号（０～７）を指定します。
タスク指定が関係ない設定の場合は０を指定して下さい。

long　　*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　：ﾌﾟﾛｸﾞﾗﾑ番号(D0～D15)、先頭ﾌﾞﾛｯｸ番号(D16～D31)



・DAT_DNCDATA　：先頭／継続フラグ　０：先頭時、１：継続時
・DAT_MCRREG　：マクロ変数番号
・DAT_ACOPARAM：０：１タスク分の受信、１：全タスク分の受信
・DAT_POINTTABLE：書込ﾎﾟｲﾝﾄ数(D0～D9)、書込開始番号(D10～D19)
軸ﾌﾗｸﾞ(D20～D28)



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ＰＬＭＣ－ＭⅡＥｘ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

ＲｅｃｅｉｖｅＤａｔａ

機能

ＰＬＭＣ－ＭⅡＥｘ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    "Plmexcomnt.h"        関数宣言に必要なインクルードファイル

int WINAPI ReceiveData(int hCom, short type, short task, short prm, LPDWORD size
                                                          , LPVOID data);
```

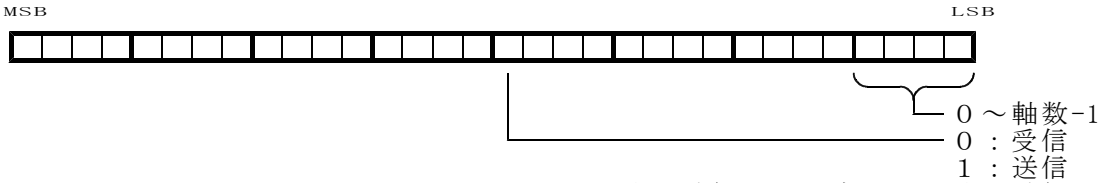
int　　*hCom*　　通信ハンドル
通信初期化処理で取得したハンドルを指定します。

short　　*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。

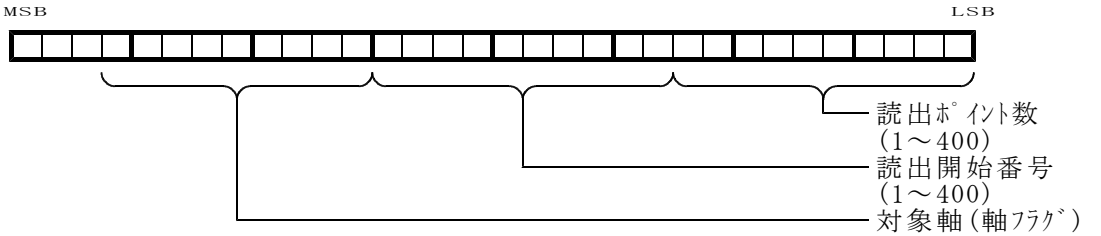
short　　*task*　　タスク番号
読出対象のタスク番号（０～７）を指定します。
タスク指定が関係ない読出の場合は０を指定して下さい。

long　　*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　１～３２７６７
・DAT_ONEBLOCK　：ステップ番号　０～３２７６７
・DAT_MCRREG　：マクロ変数番号
・DAT_TPCDATA　：読み出しデータサイズ　0～65535
・DAT_ML2INFO　：軸番号　０～軸数-1：軸指定(1軸分)
－１　：全軸指定
・DAT_ML2MON　：軸番号(D0～D3)、送受信データフラグ(D15)



・DAT_ACOPARAM：０：１タスク分の受信、１：全タスク分の受信
・DAT_POINTTABLE：書込ﾎﾟｲﾝﾄ数(D0～D9)、書込開始番号(D10～D19)
軸ﾌﾗｸﾞ(D20～D28)



LPDWORD　　*size*　　受信データサイズ格納変数へのポインタ

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ＰＬＭＣ－ＭⅡＥｘ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

ＰＬＭＣ－ＭⅡＥｘ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    "Plmexcomnt.h"        関数宣言に必要なインクルードファイル

int WINAPL SendCommand(int hCom, short cmnd, short task, 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。
short	*cmnd*	動作指示コード REQ_RESET等、動作指示のタイプを指定します。
short	*task*	タスク番号 動作指示対象のタスク番号（０～７）を指定します。 タスク指定が関係ない動作指示の場合は０を指定して下さい。
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ＰＬＭＣ－ＭⅡＥｘ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

ＤａｔＥｎｄｉａｎＣｈａｎｇｅ

機能

ダイナミックデータローディング（ＤＤＬ）用のデータ変換処理を行います。
ＰＣ／ＦＡ－Ｍ３／ＰＬＭＣ－ＭⅡＥｘでは、多バイトデータをメモリ／ファイルに格納する際のデータの並び順(エンディアン）が異なります。本関数でデータの並び順を変更する事で、データを相互に使用できるようにします。

※ 通信ライブラリを使用してＰＬＭＣ－ＭⅡＥｘと直接通信する場合は、本関数を使用する必要はありません。通信ライブラリ内で自動的に本関数を呼び出します。
ＦＡ－Ｍ３とＰＬＭＣ－ＭⅡＥｘのデータ通信（ダイナミックデータローディング）を使用する場合で、ＰＣとＦＡ－Ｍ３の間でもデータを相互に運用する場合に本関数を使用して下さい。

表１） DWORDデータ 0x12345678 をメモリ／ファイルに格納した場合のデータ並び

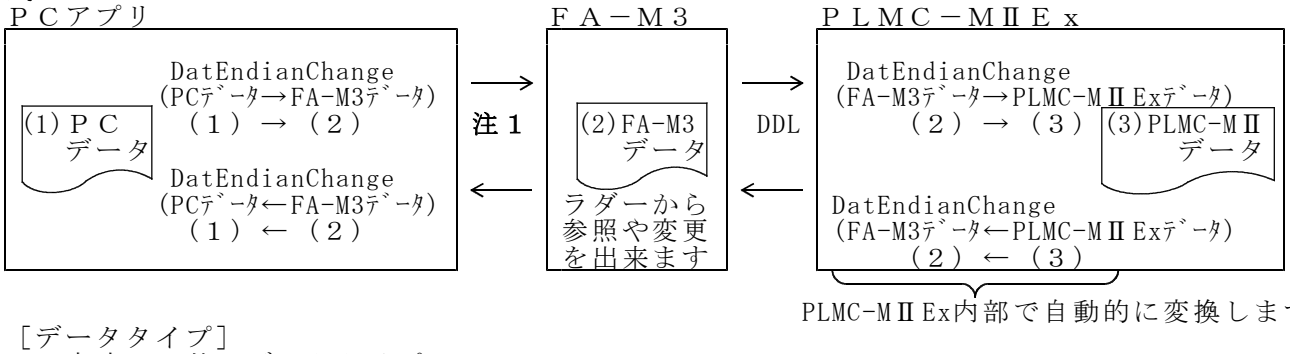
	データ元	アドレス 0	1	2	3	備考
(1)	ＰＣ(Intel系CPU)	７８	５６	３４	１２	リトルエンディアンと呼ばれています
(2)	ＦＡ－Ｍ３	５６	７８	１２	３４	
(3)	ＰＬＭＣ－ＭⅡEx	１２	３４	５６	７８	ビッグエンディアンと呼ばれています

表２） WORDデータ 0x1234 をメモリ／ファイルに格納した場合のデータ並び

	データ元	アドレス 0	1	備考
(1)	ＰＣ(Intel系CPU)	３４	１２	リトルエンディアンと呼ばれています
(2)	ＦＡ－Ｍ３	１２	３４	
(3)	ＰＬＭＣ－ＭⅡEx	１２	３４	ビッグエンディアンと呼ばれています

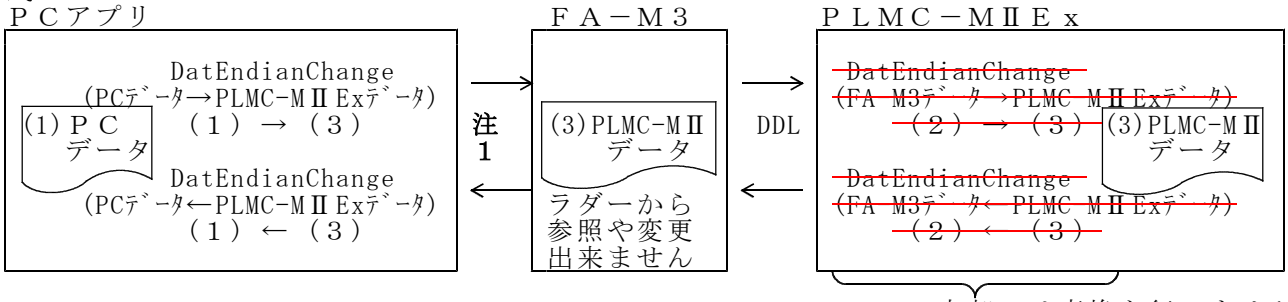
【注意点】
データタイプ（DAT_PARAMETER等）に応じて以下の２通りの変換方法があります。
どちらの変換方法になるかは、データタイプにより決まりますので、御注意下さい。

○方式１



［データタイプ］
方式２以外のデータタイプ

○方式２



［データタイプ］
DAT_PROGRAM ：運転プログラム（バイナリ）送受信
DAT_DNCDATA ：ＤＮＣ運転プログラム（バイナリ）送信
DAT_ADLOG ：ＡＤロギングデータ受信
DAT_TPCDATA ：ＴＰＣロギングデータ受信

注１：メモリモジュール(ＣＦカード)／イーサネットモジュール等のＦＡ－Ｍ３の機能を使用して、ＰＣとＦＡ－Ｍ３の間でデータを授受します。

プロトタイプ

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

int WINAPL DatEndianChange(int lbcf, short type, long size, void *src, void *dst)
```

int	*lbcf*	データ変換フラグ データの変換方法を指定します。 変換方法はデータタイプに応じて決まりますので、御注意下さい。 詳細は上記「機能」説明を参照下さい。

```
LBC_McToPc :     /* PLMC-MⅡEx → ＰＣ              */
LBC_PcToMc :     /* ＰＣ      → PLMC-MⅡEx        */
LBC_PcToM3 :     /* ＰＣ      → FA-M3            */
LBC_M3ToPc :     /* FA-M3     → ＰＣ              */
```

short	*type*	データタイプ DAT_PARAMETER等、変換するデータのタイプを指定します。
long	*size*	変換データサイズ ０～６５５３５バイト
void	**src*	変換元データ格納バッファへのポインタ
void	**dst*	変換先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ＰＬＭＣ－ＭⅡＥｘ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

ＦＡＬＳＥ：変換失敗
ＴＲＵＥ　：変換成功

ＳｅｔＥｒｒｌｏｇ

機能

PLMC-MⅡExコントローラに対して、通信エラーログファイル名を指定します。
通信初期化処理で設定した項目と同じです。
この関数によっても、エラーログファイルを指定できます。
エラーログファイル名やフラグのみを変更する場合に使用します。

プロトタイプ

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

int WINAPI SetErrLog(int hCom, char *fname, short flg);
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。

char                 *fname  通信ロギングファイル名
                               InitComProcのpLogFileの項をご参照下さい。

short                flg     通信ロギングフラグ
                               InitComProcのfLoggingの項をご参照下さい。
```

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$PARAM       : 通信パラメータ設定異常
E$NOHANDLE    : 無効ハンドル
E$ERR         : 初期化処理異常終了
```

１０－２．タイマー

Ｗｉｎｄｏｗｓが持つ、１ｍｓｅｃ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ディレイや同期、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    "Plmexcomnt.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    "Plmexcomnt.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：１ｍｓｅｃ)
```

戻り値

```
      ＝  ＦＡＬＳＥ：タイムアウトしていない
      ＝  ＴＲＵＥ  ：タイムアウトしている
```

例

```
#include  <conio.h>
#include  "Plmexcomnt.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,1100L));/*１．１秒待ち           */
    putch(ch);                            /*入力キーコード表示      */
}
```

このプログラムは、キー入力があってから１．１秒後に、入力キーを表示します。