

標準 P L M C - M 通信ライブラリリファレンスマニュアル

P L M 2 - C O M N T 説 明 書

V e r 1 . 1
2007.1.29

- 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	3
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
InitCommProc	6
QuitCommProc	9
SendData	10
ReceiveData	11
SendCommand	12
DatEndianChange	13
SetErrlog	15
9 - 2 . タイマー	16
SetTimeOut	16
CheckTimeOut	16

1. 形式

PLM2COM-NT

2. 製品構成

PLM2COMNT	.DLL	ダイナミックリンクライブラリーファイル
PLM2COMNT	.LIB	インポートライブラリーファイル
PLM2COMNT	.H	インクルードファイル(関数定義)
PLM2DATA	.H	インクルードファイル(構造体定義)
SYNDEF	.H	インクルードファイル(マクロ定義)
PLM2COMAPI	.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-Mシリーズ対応通信ライブラリ)は、パソコンとPLMC-M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

「PLMC-M ユーザーズマニュアル」	(TB00-0860)
「標準PLMC-M 対応 セッティングPCマニュアル」	(TB00-0861)
「標準PLMC-M 対応 ROMSW設定ソフトマニュアル」	(TB00-0862)
「標準PLMC-M 対応 サンプルラダープログラム説明書」	(TB00-0817)
「標準PLMC-M 対応 通信ライブラリリファレンスマニュアル」	(TB00-0863)
「標準PLMC-M 対応 送受信データ説明書」	(TB00-0864)
「標準PLMC-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」	(TB00-0865)
「標準PLMC-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」	(TB00-0866)

本書

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 . . . Windows98/NT/2000/XPが動作するx86CPU搭載機
対応Cコンパイラ . . . Visual C++ Ver6.0

6. 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。

7. 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。

- ・ 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。

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++ 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。

8 - 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Plm2comnt.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、Plm2comnt.h内で定義しています。

Plm2comnt.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 "Plm2comnt.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 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;
}
```

9 . 関数仕様

9 - 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 "Plm2comnt.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
                // pmlink common
                short       nCpuNo;        // CPU番号
                short       nUnitNo;      // PLMC-M2 ユニット番号 [0 ~ 7]
                short       nSlotNo;      // PLMC-M2 スロット番号 [1 ~ 16]
                short       Reserved;     // 予約
                // pmlink Rs232c
                short       nComPort;     // P C R S 2 3 2 C 通信ポート番号
                short       nStation;     // ハソリンクモジュールステーション番号 [1 ~ 32]
                short       nComSel;      // ハソリンクモジュール通信速度 / パリティ選択
                short       fComFlag;     // ハソリンクモジュール付加データ選択フラグ
                // pmlink Ether
                unsigned char nIpAddr[4]; // イサネットモジュール IP アドレス
                unsigned short nIpPort;   // イサネットモジュールポート番号
                short        fBinary;     // イサネットモジュールバイナリモードフラグ
            } SXDEF, *PSXDEF;
```

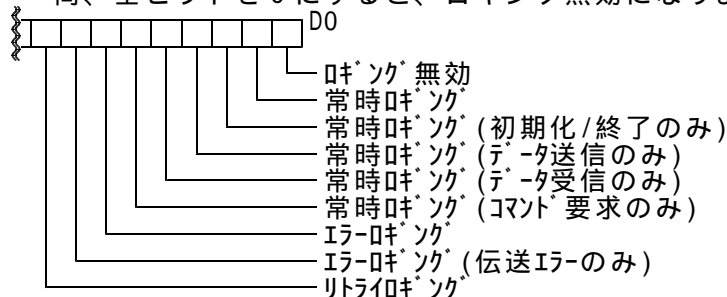
各変数の内容は以下の通りです。

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】
ハソリンク(RS232C) (0) : R S 2 3 2 C 通信
ハソリンク(イサネット) (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-M 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. 「通信ロギングファイル名」で指定したファイル
 2. 「通信ロギングファイル名」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. 232C通信の場合
 M3LnkRg.tmp
 └───┬─── ホスト番号
 └─── IP番号、Unit番号、slot番号
 イーサネット通信の場合
 M3LnkEq.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続けます。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/アドオンCPUモジュールのスロット番号を指定します。

nUnitNo PLMC-M 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-Mが実装されているユニット番号を指定します。

nSlotNo PLMC-M スロット番号 【1~16】
 通信対象のPLMC-M が実装されているスロット番号を指定します。

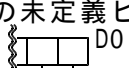
nComPort シリアル通信ポート番号 【1~256】
 RS232C通信に使用するシリアルポート番号を指定します。

nStation パソコンリンク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ク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パ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ク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。



サムチェック有効 : COM_PCLCF_SUMCHK
 終端文字(0Dh)有り : COM_PCLCF_TERMEN

nIpAddr イーサネット通信時の接続先IPアドレス
イーサネットモジュールへ接続した場合：
SW1～8で設定したアドレス
CPUモジュール(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
イーサネットモジ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
イーサネットモジュールへ接続した場合：
SW9の設定により決定
CPUモジュール(SP66-4S, SP67-6S)へ接続した場合：
CPUプロパティの上位リンクバス設定で設定した形式

1. 各設定に関しては
「標準PLMC-M 対応 セッティングPCマニュアル」
(TB00-0861) 5-4-1. インターフェイ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-M 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 "Plm2comnt.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。

<code>ESOK</code>	:	正常終了
<code>ESNOHANDLE</code>	:	無効ハンドル
<code>ESERR</code>	:	異常終了

SendData

機能

PLMC - M 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 "Plm2comnt.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 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       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 ~ 6
                  ・ 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

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 - M 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

PLMC - M 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 "Plm2comnt.h" 関数宣言に必要なインクルードファイル
```

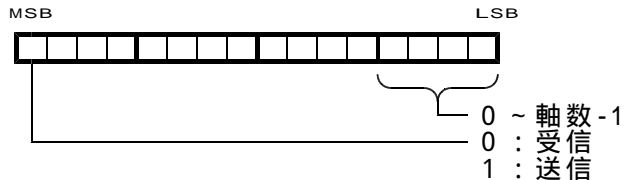
```
int WINAPI ReceiveData(int hCom, short type, short prm, LPDWORD size, LPVOID data);
```

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 *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 ~ 6
- DAT_TASKPRGSTS : タスク番号 0 ~ 6
- DAT_ONEBLOCK : ステップ番号 0 ~ 6 1 9
- 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)



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 - M 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 - M 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 "Plm2comnt.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 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。

short *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PLMC - M 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 - M 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 - M 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 - M 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				備考
		0	1	2	3	
(1)	PC (Intel系CPU)	7 8	5 6	3 4	1 2	リトルエンディアンと呼ばれています
(2)	FA - M3	5 6	7 8	1 2	3 4	
(3)	PLMC - M	1 2	3 4	5 6	7 8	ビッグエンディアンと呼ばれています

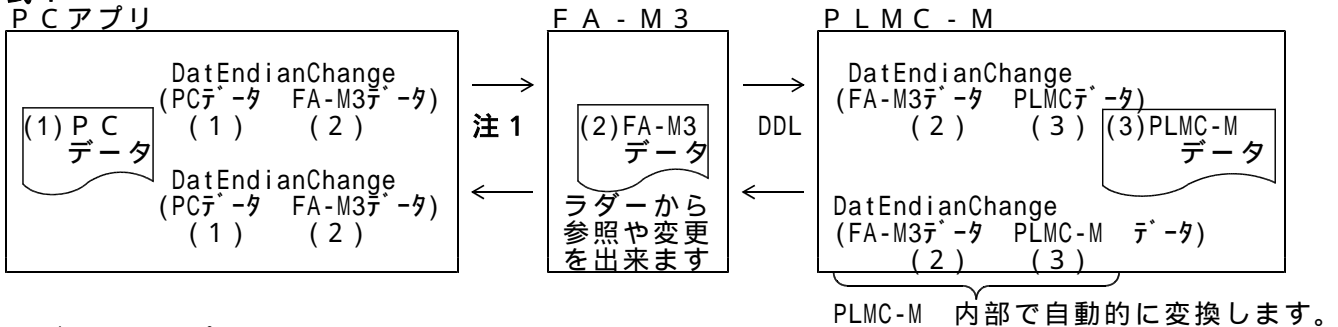
表 2) WORDデータ 0x1234 をメモリ/ファイルに格納した場合のデータ並び

	データ元	アドレス		備考
		0	1	
(1)	PC (Intel系CPU)	3 4	1 2	リトルエンディアンと呼ばれています
(2)	FA - M3	1 2	3 4	
(3)	PLMC - M	1 2	3 4	ビッグエンディアンと呼ばれています

【注意】

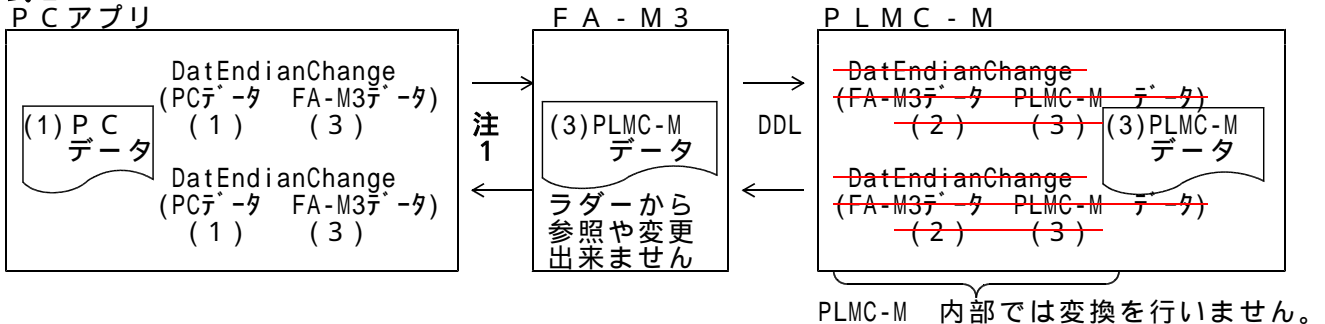
データタイプ (DAT_PARAMETER等) に応じて以下の2通りの変換方法があります。
 どちらの変換方法になるかは、データタイプにより決まりますので、御注意下さい。

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 : 動作プログラム (バイナリ) 送受信
 DAT_TASKPROG : マルチタスク動作プログラム (バイナリ) 送受信
 DAT DNCDATA : DNC 動作プログラム (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 "Plm2comnt.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-M      P C          */  
                LBC_PcToMc : /* P C          PLMC-M      */  
                LBC_PcToM3 : /* P C          FA-M3       */  
                LBC_M3ToPc : /* FA-M3       P C          */  
  
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 ~ 6 5 5 3 5 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-M 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-M 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 "plm2comnt.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

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デレイや同期、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 "Plm2comnt.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 "Plm2comnt.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 "Plm2comnt.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*/
    putchar(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。