標準ＰＬＭＣ通信ライブラリリファレンスマニュアル

ＰＬＭ－ＣＯＭＮＴ説 明 書

Ｖｅｒ１．２
2007.12.21

－目次－

１．形式

ＰＬＭＣＯＭ－ＮＴ

２．製品構成

ＰＬＭＣＯＭＮＴ　．ＤＬＬ	・・・・・・・・	ダイナミックリンクライブラリーファイル
ＰＬＭＣＯＭＮＴ　．ＬＩＢ	・・・・・・・・	インポートライブラリーファイル
ＰＬＭＣＯＭＮＴ　．Ｈ	・・・・・・・・	インクルードファイル(関数定義)
ＰＬＭＤＡＴＡ　．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)
ＰＬＭＣＯＭＡＰＩ．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シリーズ対応通信ライブラリ）は、パソコンとＰＬＭＣコントローラ間で、ＲＳ－２３２Ｃ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

「ＰＬＭＣ４０ユーザーズマニュアル」	（ＴＢ００－０８１０）
「標準ＰＬＭＣ対応 セッティングＰＣマニュアル」	（ＴＢ００－０８１２）
「標準ＰＬＭＣ対応 ＲＯＭＳＷ設定ソフトマニュアル」	（ＴＢ００－０８１１）
「標準ＰＬＭＣ対応 サンプルラダープログラム説明書」	（ＴＢ００－０８１７）
※「標準ＰＬＭＣ対応 通信ライブラリリファレンスマニュアル」	（ＴＢ００－０８１３）
「標準ＰＬＭＣ対応 送受信データ説明書」	（ＴＢ００－０８１４）
「標準ＰＬＭＣ対応 Ｔコード変換ライブラリリファレンスマニュアル」	（ＴＢ００－０８１５）
「標準ＰＬＭＣ対応 Ｇコード変換ライブラリリファレンスマニュアル」	（ＴＢ００－０８１６）

※ 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が動作するｘ８６ＣＰＵ搭載機
対応Ｃコンパイラ　・・・　Ｖｉｓｕａｌ Ｃ＋＋　Ｖｅｒ６．０以降

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。

・ 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＋＋ 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。

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Plmcomnt.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、Plmcomnt.h内で定義しています。
Plmcomnt.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 "Plmcomnt.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;
}
```

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    "Plmcomnt.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;    // 通信種別ﾌﾗｸﾞ
                        int         nComPort;    // RS232C通信ﾎﾟｰﾄ番号
                        int         nLptPort;    // ﾊﾟﾗﾚﾙ通信ﾎﾟｰﾄ
                        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;    // 通信ﾛｷﾞﾝｸﾞﾌﾗｸﾞ
                        char        *pLogFile;   // 通信ﾛｷﾞﾝｸﾞﾌｧｲﾙ名
                        int         nStation;    // ﾊﾟｿｺﾝﾘﾝｸｽﾃｰｼｮﾝ番号[1～32]
                        int         nCpuNo;      // ﾊﾟｿｺﾝﾘﾝｸCPU番号
                                                 // [1:CPUﾓｼﾞｭｰﾙ、2～4:ｱﾄﾞｵﾝCPUﾓｼﾞｭｰﾙ1～3]
                        int         nUnitNo;     // ﾊﾟｿｺﾝﾘﾝｸﾕﾆｯﾄ番号[0～7]
                        int         nSlotNo;     // ﾊﾟｿｺﾝﾘﾝｸｽﾛｯﾄ番号[1～16]
                        DWORD       dwLnkPrm1;   // ﾊﾟｿｺﾝﾘﾝｸ 通信設定１
                                                 // (COMRSLINK :通信速度/ﾊﾟﾘﾃｨ、
                                                 //  COMETHLINK:IPアドレス)
                        DWORD       dwLnkPrm2;   // ﾊﾟｿｺﾝﾘﾝｸ 通信設定２
                    } SXDEF, *PSXDEF;
```

各変数の内容は以下の通りです。

fComType　通信種別フラグ

COMRS232C	(0)：	ＲＳ２３２Ｃ通信
COMPARALLEL	(1)：	ＲＳ２３２Ｃ＋セントロ通信
COMDPRAM	(2)：	ＤＰＲＡＭ通信
COMUSB	(3)：	ＵＳＢ通信
COMETHERNET	(4)：	イーサネット通信
COMRSLINK	(5)：	ＲＳ２３２Ｃ上位リンク通信
COMETHLINK	(6)：	イーサネット上位リンク通信

nComPort　ＲＳ２３２Ｃ通信ポート番号【１～２５６】

nLptPort　パラレル通信ポート【１～４】
ＰＬＭＣでは使用しません。１を指定して下さい。

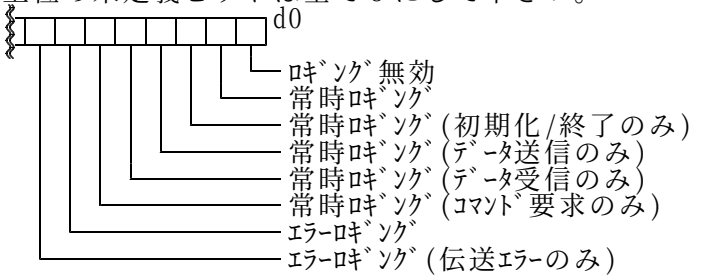
nBoardId　ＵＳＢ通信ＩＤ【０～４】
ＰＬＭＣでは使用しません。０を指定して下さい。

fShare　共有フラグ【０～１】
１を指定して下さい。

fSxType　コントローラ種別
TYPE_PLMC(0x20)：PLMC40

nSize　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。
sizeof():VC++/ LenB():VB 関数でサイズを取得できます。

fLogging　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。



pLogFile　通信ロギングファイル名
ＰＬＭＣ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．PlmComo.tmp
　　└ COMﾎﾟｰﾄ番号
ログファイル（１のファイル）が５１２Ｋバイトを越えると、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。

nStation　パソコンリングステーション番号【1～32】
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。

nCpuNo　CPU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　ﾊﾟｿｺﾝﾘﾝｸﾕﾆｯﾄ番号【0～7】
光FAバス等のサブユニット使用時に対象となるPLMC-40が実装されているユニット番号を指定します。

nSlotNo　ﾊﾟｿｺﾝﾘﾝｸｽﾛｯﾄ番号【1～16】
通信対象のPLMC-40が実装されているスロット番号を指定します。

dwLnkPrm1　ﾊﾟｿｺﾝﾘﾝｸ 通信設定１
(COMRSLINK :通信速度/ﾊﾟﾘﾃｨ、COMETHLINK:IPアドレス)

・COMRSLINK(fComType=COMRSLINK)の場合
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

・COMETHLINK(fComType=COMETHLINK)の場合

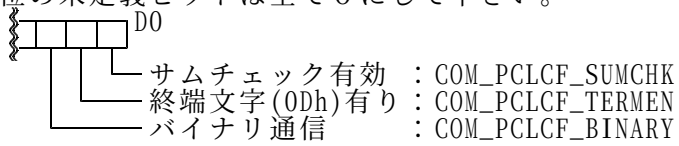
通信対象のｲｰｻﾈｯﾄﾓｼﾞｭｰﾙ／CPUﾓｼﾞｭｰﾙのIPｱﾄﾞﾚｽを
設定します。IPｱﾄﾞﾚｽの４桁を最下位バイトから
順番に格納します。

（例）169.254.255.1 を設定する場合

DWORD dwLnkPrm1

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を設定します。
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アドレスを設定します。

dwLnkPrm2　パソコンリンクプロトコル選択【0～4】
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。



●COMRSLINK：ＲＳ２３２Ｃ上位リンク通信の場合
・”サムチェック有効”/”終端文字(0Dh)有り”を
設定して下さい。

●COMETHLINK：イーサネット上位リンク通信の場合
・”バイナリ通信”を設定して下さい。

・ｲｰｻﾈｯﾄﾓｼﾞｭｰﾙへ接続した場合：
SW9の設定により決定します。
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
形式です。

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ＰＬＭＣ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    "Plmcomnt.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

ＰＬＭＣ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    "Plmcomnt.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ＰＬＭＣ４０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    "Plmcomnt.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ＰＬＭＣ４０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    "Plmcomnt.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 cmnd, 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	*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。
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ＰＬＭ-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

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

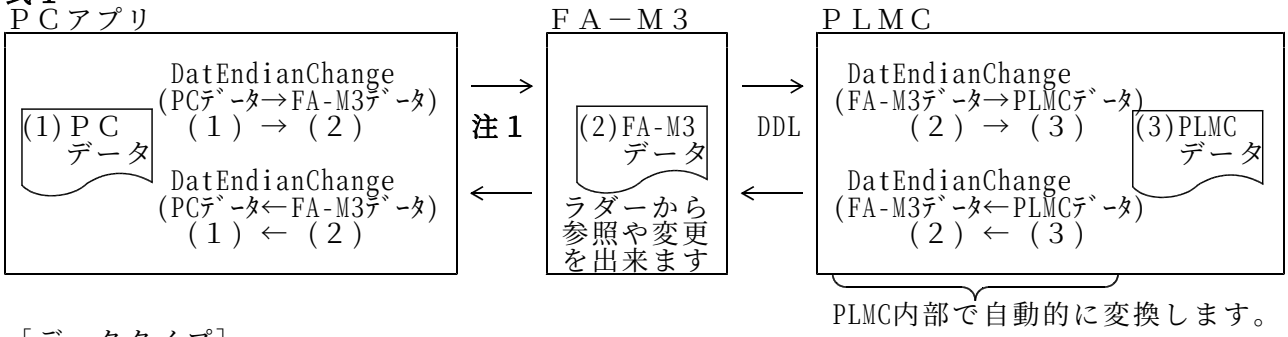
	データ元	アドレス 0	1	2	3	備考
(1)	ＰＣ(Intel系CPU)	７８	５６	３４	１２	リトルエンディアンと呼ばれています
(2)	ＦＡ－Ｍ３	５６	７８	１２	３４	
(3)	ＰＬＭＣ	１２	３４	５６	７８	ビッグエンディアンと呼ばれています

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	１２	３４	ビッグエンディアンと呼ばれています

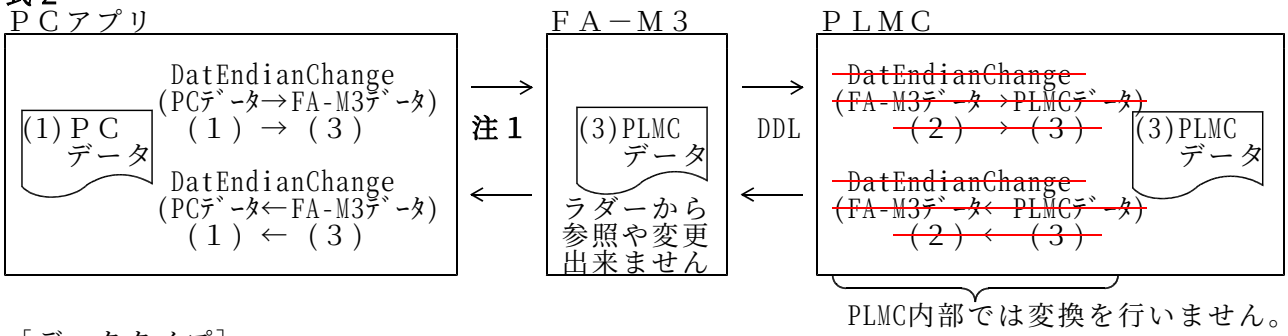
【注意点】
データタイプ（DAT_PARAMETER等）に応じて以下の２通りの変換方法があります。
どちらの変換方法になるかは、データタイプにより決まりますので、御注意下さい。

○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 ：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    "Plmcomnt.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 → ＰＣ　*/ LBC_PcToMc：　/* ＰＣ → PLMC　*/ LBC_PcToM3：　/* ＰＣ → FA-M3　*/ LBC_M3ToPc：　/* FA-M3 → ＰＣ　*/
int	*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。
DWORD	*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ＰＬＭＣ４０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

９－２．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    "Plmcomnt.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    "Plmcomnt.h"     関数宣言に必要なインクルードファイル

BOOl WINAPI  CheckeTimeOut(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  "Plmcomnt.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              */
    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。