DSH-PLCSim　対PLCシミュレータ

装置コントローラのための

インタフェース関数説明書

DSHPLC.DLL ライブラリ

２００８年6月

株式会社データマップ

文書番号　DSHPLCSIM-08-30320-00

[取り扱い注意]

- この資料ならびにソフトウェアの一部または全部を無断で使用、複製することはできません。
- 本説明書に記述されている内容は予告なしで変更される可能性があります。
- Windows は米国 Microsoft Corporation の登録商標です。
- MELSEC は三菱電機株式会社殿の登録商標です。
- ユーザーが本ソフトウェアの使用によって生じた遺失履歴、（株）データマップの予見の有無を問わず発生した特別損害、付随的損害、間接損害およびその他の拡大損害に対して責任を負いません。

【改訂履歴】

番号	改訂日付	項 目	概 略
1.	2008.6	初版	
2.			
3.			
4.			

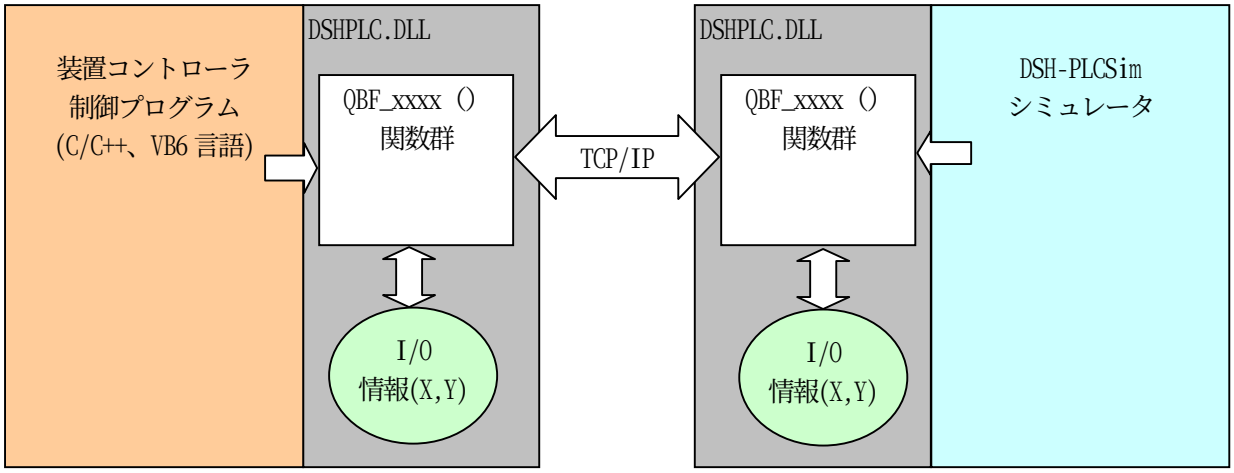
目　次

１．概要

本説明書は、装置コントローラ(EQCON)制御プログラムが DSH-PLCSim シミュレータとの通信で使用するインタフェース関数について説明します。

本シミュレータがサポートする PLC は、**三菱電機(株)製 MELSEC** シリーズの PLC であり、バス・インタフェース関数が使用できるものを対象としています。

EQCON が、PLC の代わりを務める DSH-PLCSim とのインタフェースを取るために、MELSEC のバス・インタフェース関数と同じ機能を実行するライブラリが必要になります。DSH-PLCSim はこのライブラリを EQCON に提供します。
このライブラリは、DLL（Dynamic Link Library）で、名前は、**DSHPLC.DLL** です。
DLL は C/C++、VB6 などの高級言語によるプログラムとのリンクを可能にし、DLL が提供する機能を実行することができます。すなわち、EQCON と DSH-PLCSim との間のインタフェースは、DSHPLC.DLL ダイナミックライブラリに含まれる関数を使って取ります。



[関連文書]

DSH-PLCSim 関連文書一覧表

#	文書番号	文書名	注釈
1	DSHPLCSIM-08-30304-00	DSH-PLCSim ﾕｰｻﾞｰｽﾞ・ｶﾞｲﾄﾞ	機能と操作概要説明書
2	DSHPLCSIM-08-30305-00	DSH-PLCSim 操作ﾏﾆｭｱﾙ	この文書です。
3	DSHPLCSIM-08-30320-00	DSH-PLCSim ｲﾝﾀﾌｪｰｽ関数説明書	装置ｺﾝﾄﾛｰﾗ側が使用できるｲﾝﾀﾌｪｰｽ関数
4	DSHPLCSIM-08-30390-00	DSH-PLCSim PLC 制御デモ・プログラムについて（VB6.0 言語版）	VB6 のﾃﾞﾓﾌﾟﾛｸﾞﾗﾑの説明書です。
5	DSHPLCSIM-08-30391-00	DSH-PLCSim PLC 制御デモ・プログラムについて（VC++6.0 言語版）	Visual C++のﾃﾞﾓﾌﾟﾛｸﾞﾗﾑの説明書です。

[必要なソフトウェア]

ソフトウェア	
ＯＳ	Windows-2000、Windows-XP または Windows-Vista
DSHPLCSIM 関連	dshplcsim.exe - シミュレータ本体プログラムファイル dshplc.dll - PLC 通信プログラムライブラリファイル

２．インタフェース関数概要

（1）バスインタフェース互換関数（QBF_xxx）

DSHPLC.dll ライブラリが準備している MELSEC バスインタフェース関数互換のために準備されている関数は次表の通りです。

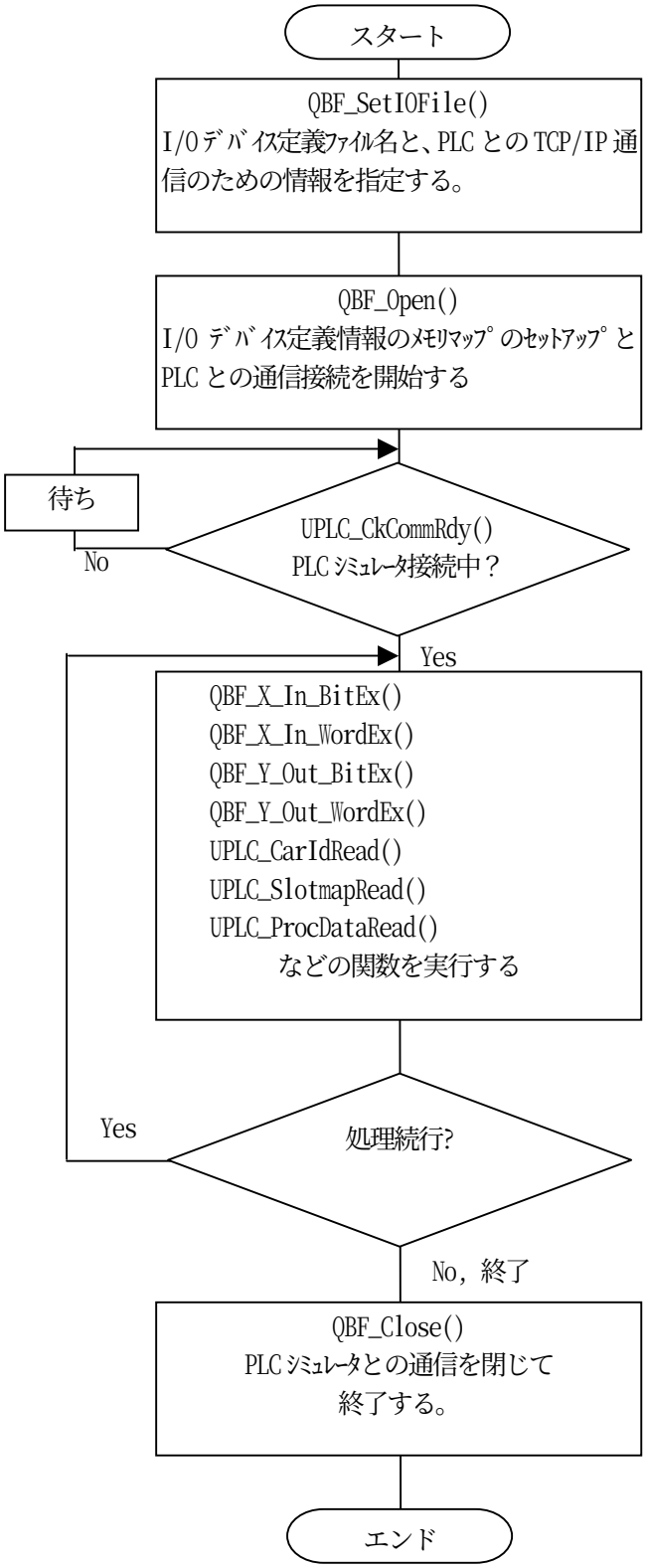
	I/F 関数	機能
1	QBF_SetIOFile()	入出力定義ﾌｧｲﾙ名と DSH-PLCSim との TCP/IP 通信のための情報を設定します。 **DSH-PLCSim 特有の関数です。**
2	QBF_Open()	ﾊﾞｽまたは通信回線をｵｰﾌﾟﾝします。 DSH-PLCSim と接続します。
3	QBF_Close()	ﾊﾞｽまたは通信回線をｸﾛｰｽﾞします。 DSH-PLCSim との接続を切断します。
4	QBF_X_In_BitEx()	入力信号(X)を１点入力します。
5	QBF_X_In_WordEx()	入力信号(X)をﾜｰﾄﾞ単位で入力します。
6	QBF_Y_Out_BitEx()	出力信号(Y)を１点出力します。
7	QBF_Y_Out_WordEx()	出力信号(Y)をﾜｰﾄﾞ単位で出力します。
8	QBF_Y_In_BitEx()	出力信号(Y)を１点出力します。
9	QBF_Y_In_WordEx()	出力信号(Y)をﾜｰﾄﾞ単位で出力します。
10	QBF_WaitUnitEvent()	ｲﾍﾞﾝﾄを指定して PLC からのｲﾍﾞﾝﾄを待機します。
11	QBF_Reset()	DSHPLC.DLL をﾘｾｯﾄします。
12	QBF_RefreshLinkDevice() QBF_WriteLinkDevice() QBF_SEND() QBF_RECV() QBF_UnitInfo() QBF_StartWDT() その他の関数	これらの関数については、何も処理しません。 返却値が必要な関数については、全て = 0 を返却します。

（2）DSH-PLCSim との通信のための関数

DSH-PLCSim シミュレータに対する固有の関数

	I/F 関数	機能
1	UPLC_CkCommReady()	DSH-PLCSim との接続が確立しているかどうかを調べるための関数です。
2	UPLC_XBitSyncReq()	DSH-PLCSim が有する入力信号(X)のﾃﾞｰﾀを読出し、ﾃﾞﾊﾞｲｽｺﾝﾄﾛｰﾗ側のﾃﾞﾊﾞｲｽﾒﾓﾘに格納します。
3	UPLC_YBitSyncReq()	DSH-PLCSim が有する出力信号(Y)のﾃﾞｰﾀを読出し、ﾃﾞﾊﾞｲｽｺﾝﾄﾛｰﾗ側のﾃﾞﾊﾞｲｽﾒﾓﾘに格納します。
4	UPLC_CarIdRead()	ｷｬﾘｱ ID を読出します。 読出すｷｬﾘｱ ID は、DSH-PLCSim の画面上に選択設定された値です。
5	UPLC_SlotmapRead()	ｽﾛｯﾄﾏｯﾌﾟ情報（25 ｽﾛｯﾄ分）を読出します。 読出すｽﾛｯﾄﾏｯﾌﾟ情報は、DSH-PLCSim の画面上に選択設定された On/off 情報です。
6	UPLC_ProcDataRead()	指定した番号のﾌﾟﾛｾｽﾃﾞｰﾀを１個読出します。 読出すﾌﾟﾛｾｽﾃﾞｰﾀは文字列ﾃﾞｰﾀです。 ｼﾐｭﾚｰﾀのﾌﾟﾛｾｽﾃﾞｰﾀ画面には 7 個のﾃﾞｰﾀが設定されています。

（3）インタフェース関数の実行フローチャート



３．MELSEC バス・インタフェース互換関数（QBFxxx）

以下説明する関数の書式についてですが、例えば、QBF_Open( )関数については、次のような関数の表現になります。

```
API short WINAPI QBF_Open( short unit, long*path );
```

ここで、API と WINAPI は、呼出規約を指定します。MicroSoft 社 MSC 上で、以下のマクロで規定されるものです。

```
#define  API      __declspec( dllexport )
#define  WINAPI   __stdcall
```

データタイプの表現ですが、次のようになります。

ULONG は unsigned long、
USHORT は unsigned short のタイプを意味します。

３．１　QBF_SetIOFile( ) – DSHPLC.DLL セットアップ情報の設定

（1）呼出書式

```
API short WINAPI QBF_SetIOFile (
    char *pFileName,                // I/Oﾃﾞﾊﾞｲｽ定義ﾌｧｲﾙ名
    char *ip,                       // 通信接続用の IP ｱﾄﾞﾚｽ
    int port                        // TCPﾎﾟｰﾄ番号
);
```

（2）引数

pFileName

I/O デバイス定義ファイル名を指定します。

char *ip

TCP/IP 接続のためのサーバー側の IP アドレスを指定します。

EQCON 側はサーバーの立場になりますから、この値は関数では使用されません。

PLC シミュレータ側は EQCON の IP アドレスを設定する必要があります。

IP は例えば、 “192.168.1.50” のように文字列データで表現します。

int　port

接続したい TCP のポートを指定します。

（3）戻り値

戻り値	意味
0	正常に初期化情報の設定しました。

（4）説明

PLC シミュレータとの接続を行なうための情報の準備をします。

pFileName には、I/O デバイス定義ファイルの名前、そして、ip, port には PLC シミュレータとの接続するための TCP/IP プロトコルの設定情報を指定して呼び出します。

この関数では、PLC シミュレータとの接続は行なわれません。この後の QFB_Open()関数実行時に I/O 定義ファイルの I/O マップメモリへの展開と相手との通信のための接続が行なわれます。

本関数の戻り値として=0 が返却されます。

３．２　QBF_Open( ) – PLC との通信を開く

（1）呼出書式

```
API short WINAPI QBF_ Open(
    short unit,                    // unit 識別
    long *path                     // 接続情報ﾎﾟｲﾝﾀ
);
```

（2）引数

unit

　バスユニット番号です。DSHPLC では私用しません。

path

　ユニットバスの Path ポインタです。DSHPLC は常に０を返します。

（3）戻り値

戻り値	意味
0	正常に初期化と接続処理を開始した。
-218	オープン済エラー

（4）説明

バス・インタフェースを開くための関数です。PLC シミュレータとの通信では、以下のような処理をします。

DSHPLC は、QBF_SetIOFile()関数で設定された I/O デバイス定義ファイルに定義されたデバイスの定義内容を DSHPLC.DLL がデバイスに対応するメモリマップ領域に展開します。

また、QBF_SetIOFile()関数で指定されたポートから PLC からの接続を待機します。本関数が戻ったときに通信接続されているとは限りません。
接続相手である PLC シミュレータからの接続要求がきたときに接続状態になります。

通信が確立しているかどうかは、UPLC_CkCommRdy()関数を使って確認できます。

正常にオープンできた場合、本関数の戻り値として=0 が返却されます。

３．３　QBF_Close( ) – PLC との通信を閉じる

（1）呼出書式

```
API short WINAPI QBF_Close (
    long  path,                    // Bus のパス
);
```

（2）引数

path

QBF_Open()で得られたバスのパスです。DSHPLC では使用しません。
DSHPLC ライブラリでは使用しません。

（3）戻り値

戻り値	意味
0	正常にｸﾛｰｽﾞした。

（4）説明

PLC シミュレータとの通信を切断します。
同時に、I/O デバイスに設けられたメモリマップ情報も失われます。

正常にクローズできた場合、戻り値として=0 が返却されます。
オープンされていなかった場合でも = 0 を返却します。

３．４　QBF_X_In_BitEx ( ) – 入力信号(X)を１点読み出す

（1）呼出書式

```
API short WINAPI QBF_X_In_BitEx (
    long path,                  // BUS のパス
    short sFlg,                 // ｱｸｾｽﾌﾗｸﾞ
    USHORT usXno,               // 入力ﾃﾞﾊﾞｲｽ番号
    USHORT *pusData             // 入力ﾃﾞﾊﾞｲｽの状態を格納する領域
);
```

（2）引数

path

QBF_Open()で得られたパスです。DSHPLC ライブラリでは使用しません。

sFlg

アクセスフラグです。DSHPLC ライブラリでは使用しません。

usXno

入力デバイスの番号を指定します。

pusData

入力デバイスの状態 On/Off を格納する領域です。

On のとき = 1、Off のとき = 0 が格納返却されます。

（3）戻り値

戻り値	意味
0	正常に読出し処理が終了しました。
-217	Open されていないか PLC と通信接続できていません。
-205	ﾃﾞﾊﾞｲｽ番号が間違っています。

（4）説明

usXno で与えられた入力デバイスの On/Off 状態を 1/0 の値で、pusData で指定された領域に設定します。
正常に読み出せた場合、戻り値として=0 が返却されます。
デバイス番号が正しくなかった（未定義）場合、-205 を返却します。

３．５　QBF_X_In_WordEx ( ) – 入力信号(X)をワード単位で読み出す

（1）呼出書式

```
API short WINAPI QBF_X_In_WordEx (
    long path,                  // BUS のパス
    short sFlg,                 // ｱｸｾｽﾌﾗｸﾞ
    USHORT usXno,               // 先頭入力ﾃﾞﾊﾞｲｽ番号
    USHORT usSize,              // 読み出すﾜｰﾄﾞ数
    USHORT *pusData,            // 入力ﾃﾞﾊﾞｲｽの状態を格納する領域
    USHORT usBufSize            // pusData のﾜｰﾄﾞｻｲｽﾞ
);
```

（2）引数

path

　QBF_Open()で得られたパスです。DSHPLC ライブラリでは使用しません。

sFlg

　アクセスフラグです。DSHPLC ライブラリでは使用しません。

usXno

　読み出す入力デバイスの先頭番号を指定します。

usSize

　読み出すワード数を指定します。

pusData

　入力デバイスのデータを格納する領域です。

usBufSize

　pusData に準備された領域のワードサイズです。

（3）戻り値

戻り値	意味
0	正常に読出し処理が終了しました。
-217	Open されていないか PLC と通信接続できていません。
-205	ﾃﾞﾊﾞｲｽ番号が間違っています。
77	ﾊﾞｯﾌｧｻｲｽﾞが不足しています。

（4）説明

usXno で与えられた先頭入力デバイス番号から usSize ワード分の値で、pusData で指定された領域に設定します。

正常に読み出せた場合、戻り値として=0 が返却されます。

デバイス番号が正しくなかった（未定義）場合、-205 を返却します。

バッファの容量が読出しワード数に満たない場合は 77　を返却します。

３．６　QBF_Y_Out_BitEx ( ) – 出力信号(Y)を１点書込む

（1）呼出書式

```
API short WINAPI QBF_Y_Out_BitEx (
    long path,                  // BUS のパス
    short sFlg,                 // ｱｸｾｽﾌﾗｸﾞ
    USHORT usYno,               // 出力ﾃﾞﾊﾞｲｽ番号
    USHORT usData               // 出力ﾃﾞﾊﾞｲｽのﾃﾞｰﾀ(0/1)
);
```

（2）引数

path

　QBF_Open()で得られたパスです。DSHPLC ライブラリでは使用しません。

sFlg

　アクセスフラグです。DSHPLC ライブラリでは使用しません。

usYno

　出力デバイスの番号を指定します。

usData

　書込む出力デバイスの状態 On/Off を指定します。

　On のとき = 1、Off のとき = 0 を設定します。

（3）戻り値

戻り値	意味
0	正常に書込み処理が終了しました。
-217	Open されていないか PLC と通信接続できていません。
-205	ﾃﾞﾊﾞｲｽ番号が間違っています。

（4）説明

usYno で与えられた出力デバイスに対し、usData で指定される On/Off 状態を書き込みます。

正常に書込めた場合、戻り値として=0 が返却されます。

デバイス番号が正しくなかった（未定義）場合、-205 を返却します。

３．７　QBF_Y_Out_WordEx ( ) – 出力信号(Y)をワード単位で読み出す

（1）呼出書式

```
API short WINAPI QBF_Y_Out_WordEx (
    long path,                  // BUS のパス
    short sFlg,                 // ｱｸｾｽﾌﾗｸﾞ
    USHORT usYno,               // 先頭出力ﾃﾞﾊﾞｲｽ番号
    USHORT usSize,              // 読み出すﾜｰﾄﾞ数
    USHORT *pusData,            // 出力ﾃﾞﾊﾞｲｽの状態を格納する領域
    USHORT usBufSize            // pusData のﾜｰﾄﾞｻｲｽﾞ
);
```

（2）引数

path

　QBF_Open()で得られたパスです。DSHPLC ライブラリでは使用しません。

sFlg

　アクセスフラグです。DSHPLC ライブラリでは使用しません。

usYno

　書込む出力デバイスの先頭番号を指定します。

usSize

　書込むワード数を指定します。

pusData

　出力デバイスのデータを格納する領域です。

usBufSize

　pusData 領域のワードサイズです。使用しませんので=0 を指定してください。

（3）戻り値

戻り値	意味
0	正常に書込み処理が終了しました。
-217	Open されていないか PLC と通信接続できていません。
-205	ﾃﾞﾊﾞｲｽ番号が間違っています。

（4）説明

usYno で与えられた先頭出力デバイス番号から usSize ワード分の値で、pusData で指定された領域のワードﾃﾞｰﾀを書き込みます。

正常に書込めた場合、戻り値として=0 が返却されます。

デバイス番号が正しくなかった（未定義）場合、-205 を返却します。

３．８　QBF_Y_In_BitEx ( ) – 出力信号(Y)を１点読み出す

（1）呼出書式

```
API short WINAPI QBF_Y_In_BitEx (
    long path,                  // BUS のパス
    short sFlg,                 // ｱｸｾｽﾌﾗｸﾞ
    USHORT usYno,               // 出力ﾃﾞﾊﾞｲｽ番号
    USHORT *pusData             // 出力ﾃﾞﾊﾞｲｽの状態を格納する領域
);
```

（2）引数

path

QBF_Open()で得られたパスです。DSHPLC ライブラリでは使用しません。

sFlg

アクセスフラグです。DSHPLC ライブラリでは使用しません。

usYno

出力デバイスの番号を指定します。

pusData

出力デバイスの状態 On/Off を格納する領域です。

On のとき = 1、Off のとき = 0 が格納返却されます。

（3）戻り値

戻り値	意味
0	正常に読出し処理が終了しました。
-217	Open されていないか PLC と通信接続できていません。
-205	ﾃﾞﾊﾞｲｽ番号が間違っています。

（4）説明

usYno で与えられた出力デバイスの On/Off 状態を 1/0 の値で、pusData で指定された領域に設定します。
正常に読み出せた場合、戻り値として=0 が返却されます。
デバイス番号が正しくなかった（未定義）場合、-205 を返却します。

３．９　QBF_Y_In_WordEx ( ) – 出力信号(Y)をワード単位で読み出す

（1）呼出書式

```
API short WINAPI QBF_Y_In_WordEx (
    long path,                  // BUS のパス
    short sFlg,                 // ｱｸｾｽﾌﾗｸﾞ
    USHORT usYno,               // 先頭出力ﾃﾞﾊﾞｲｽ番号
    USHORT usSize,              // 読み出すﾜｰﾄﾞ数
    USHORT *pusData,            // 出力ﾃﾞﾊﾞｲｽの状態を格納する領域
    USHORT usBufSize            // pusData のﾜｰﾄﾞｻｲｽﾞ
);
```

（2）引数

path

　QBF_Open()で得られたパスです。DSHPLC ライブラリでは使用しません。

sFlg

　アクセスフラグです。DSHPLC ライブラリでは使用しません。

usYno

　読み出す出力デバイスの先頭番号を指定します。

usSize

　読み出すワード数を指定します。

pusData

　出力デバイスのデータを格納する領域です。

usBufSize

　pusData に準備された領域のワードサイズです。

（3）戻り値

戻り値	意味
0	正常に読出し処理が終了しました。
-217	Open されていないか PLC と通信接続できていません。
-205	ﾃﾞﾊﾞｲｽ番号が間違っています。
77	ﾊﾞｯﾌｧｻｲｽﾞが不足しています。

（4）説明

usYno で与えられた先頭出力デバイス番号から usSize ワード分の値で、pusData で指定された領域に設定します。

正常に読み出せた場合、戻り値として=0 が返却されます。

デバイス番号が正しくなかった（未定義）場合、-205 を返却します。

バッファの容量が読出しワード数に満たない場合は 77　を返却します。

３．10　QBF_WaitUnitEvent ( ) – イベントを待機する

（1）呼出書式

```
API short WINAPI QBF_WaitUnitEvent(
    long path,                      // BUS のパス
    short *psEvent,                 // 待ちｲﾍﾞﾝﾄ情報
    ULONG ulTimeout,                // 待ちﾀｲﾑｱｳﾄ時間(ms)
    short *psSetEventNo             // 発生したｲﾍﾞﾝﾄ番号格納領域
);
```

（2）引数

path

QBF_Open()で得られたパスです。DSHPLC ライブラリでは使用しません。

psEvent

待ちイベントの数と待ちイベント番号が格納されている配列領域のﾎﾟｲﾝﾀです。

psEvent[0] = 待ちイベント数

psEvent[1] = 1 番目の待ちイベント番号

以下、順に、2 番目以降のイベント番号を格納しておきます。

MELSEC の QBF 関数ではのイベント番号、割込み要因は次のようになります。

0～15　：QI60 による割込み

16～49　：リザーブ

50～255　：インテリジェント機能ユニット割込み。

C 言語コントローラ設定ユーティリティにて設定をする

ulTimeout

イベントを待機する時間を ms(ミリ秒)で指定します。

値が =0xFFFFFFFF の場合は、永久に待機する指定になります。

psSetEventNo

発生したイベント番号を格納する領域のポインタです。

（3）戻り値

戻り値	意味
>= 0	正常に読出し処理が終了しました。
-217	Open されていないか PLC と通信接続できていません。
-231	ﾃﾞﾊﾞｲｽ番号が間違っています。

（4）説明

PLC からのイベント発生を待機します。

psEvent 領域に指定されたイベントを ulTimeout で指定された時間だけ待機します。

ulTimeout 時間内に、指定されたイベントが発生した場合、psSetEventNo 領域に発生したイベント番号を格納するとともにその値を返却します。

ulTimeout 時間内にイベントが発生しなかった場合には、タイムアウトの値(-231)を返却します。

ulTimeout の値として 0xFFFFFFFF が与えられた場合には､指定されたイベントが発生するまで永久に待機します。

本関数が実行される前に既に指定されたイベントが発生していた場合は、即時そのイベント番号が発生したものとして処理します。

PLC はイベント発生に QBF_GINT()関数を使用します。

３．11　QBF_Reset()－リセット

（1）呼出書式

```
API short WINAPI QBF_ WaitUnitEvent(
    long path                          // BUS のパス
    );
```

（2）引数

path

QBF_Open()で得られたパスです。DSHPLC ライブラリでは使用しません。

（3）戻り値

戻り値	意味
0	常時 0 を返却します。

（4）説明

３．10 で説明した PLC からのイベントで、発生しているものがあればそれらをリセットします。
本関数では、現段階でこの処理だけを実行します。

３．12　その他の QBF 関数

その他 QBF インタフェース関数としている関数については、DSHPLC ライブラリでは、何も処理しないで、単に０を返却します。

```
API short WINAPI QBF_ToBuf( long path, USHORT usIoNo, ULONG ulOffset, ULONG ulSize, USHORT *pusDataBuf,
    ULONG ulBufSize );
API short WINAPI QBF_FromBuf( long path, USHORT usIoNo, ULONG ulOffset, ULONG ulSize, USHORT *pusDataBuf,
    ULONG ulBufSize );
API short WINAPI QBF_RefreshLinkDevice( long path, USHORT usIoNo );
API short WINAPI QBF_WriteLinkDevice( long path, USHORT usIoNo, short sDevType, ULONG ulDevNo,
    ULONG ulSize, USHORT *pusDataBuf, ULONG ulBufSize );
API short WINAPI QBF_ReadLinkDevice( long path, USHORT usIoNo, short sDevType, ULONG ulDevNo, ULONG ulSize,
     USHORT *pusDataBuf, ULONG ulBufSize );
API short WINAPI QBF_SEND( long path, USHORT usIoNo, short *psControlData, USHORT *pusDataBuf, ULONG
     ulBufSize, short *psAddInfo );
API short WINAPI QBF_RECV( long path, USHORT usIoNo, short *psControlData, USHORT *pusDataBuf, ULONG
     ulBufSize, short *psAddInfo );
API short WINAPI QBF_UnitInfo( long path, USHORT pusUnitInfo );
API short WINAPI QBF_StartWDT( long path, short sType, short sInterval );
API short WINAPI QBF_ResetWDT( long path, short sType );
API short WINAPI QBF_StopWDT( long path, short sType );
API short WINAPI QBF_EntryWDTInt( long path, short sType, void *pFuncPtr );
API short WINAPI QBF_EntryTimerEvent ( long path, long plEvent );
API short WINAPI QBF_WaitTimerEvent ( long path, long lEventNo );
API short WINAPI QBF_ReadStatusEx( long path, long *plInfo, ULONG pulBufSize );
API short WINAPI QBF_ControlLED( long path, USHORT usLedInfo );
API short WINAPI QBF_Control( long path, short sCode );
API short WINAPI QBF_ControlEx( long path, long sCpuNo, short sCode);
API short WINAPI QBF_RegistEventLog( long path, short sFlg, char *pcSrcStr, long lEventNo,
     char *pcAddMsg );
API short WINAPI QBF_MountCfCard( long path, short sDrive );
API short WINAPI QBF_UnmountCfCard( long path, short sDrive );
API short WINAPI QBF_ShutdownRom( long path, short sMode );
API short WINAPI QBF_SetTime( long path, short *psSetData );
API short WINAPI QBF_GetTime( long path, short psGetdata );
API short WINAPI QBF_WriteSRAM( long path, short sFlg, ULONG ulOffset, ULONG ulSize, char *pcDataBuf,
     ULONG ulBufSize );
API short WINAPI QBF_ReadSRAM( long path, short sFlg, ULONG ulOffset, ULONG ulSize, char *pcDataBuf,
     ULONG ulBufSize );
API short WINAPI QBF_WaitEvent( long path, short *psEvent, ULONG ulTimeout, short *psSetEventNo);
API short WINAPI QBF_GINT( long path, short sCpuNo, short sEventNo);
API short WINAPI QBF_ControlProgram( long path, short sCpuNo, char *pcProgramName, short sType,
     long lInterval );
API short WINAPI QBF_MotionSFCS( long path, short sCpuNo, short sProgramNo, long ulTimeout);
API short WINAPI QBF_MotionSVST( long path, short sCpuNo, short *psAxisno, short sProgramNo,
     ULONG ulTimeout );
API short WINAPI QBF_MotionCHGA( long path, short sCpuNo, short sAxisNo, long lData, ULONG ulTimeout );
API short WINAPI QBF_MotionCHGV( long path, short sCpuNo, short sAxisNo, long lData, ULONG ulTimeout );
API short WINAPI QBF_MotionCHGT( long path, short sCpuNo, short sAxisNo, long lData, ULONG ulTimeout );

API short WINAPI QBF_MotionDDWR( long path, short sCpuNo, short sDevType, ULONG ulDevNo, ULONG ulSize,
     USHORT *pusDataBuf, ULONG ulBufSize, ULONG ulTimeout);
```

```
API short WINAPI QBF_MotionDDRD( long path, short sCpuNo, short sDevType, ULONG ulDevNo, ULONG ulSize,
      USHORT *pusDataBuf, ULONG ulBufSize, ULONG ulTimeout);
```

４． DSH-PLCSim シミュレータ専用関数

DSH-PLC が準備する専用関数について説明します。

４．１ UPLC_CｋCommRdy() – 対 PLC シミュレータ通信接続状態チェックする

（1）呼出書式

```
API int WINAPI UPLC_CkCommRdy( );
```

（2）引数

なし。

（3）戻り値

戻り値	意味
0	未接続です。
1	接続状態です。

（4）説明

PLC シミュレータとの間で通信が接続されているかどうかを調べるための関数です。
接続していれば = 1 を返却します。接続できていない場合は = 0 を返します。

４．２　UPLC_XBitSyncReq（）- PLC シミュレータとの入力デバイス同期要求関数

（1）呼出書式

```
API int WINAPI UPLC_XBitSyncReq( );
```

（2）引数

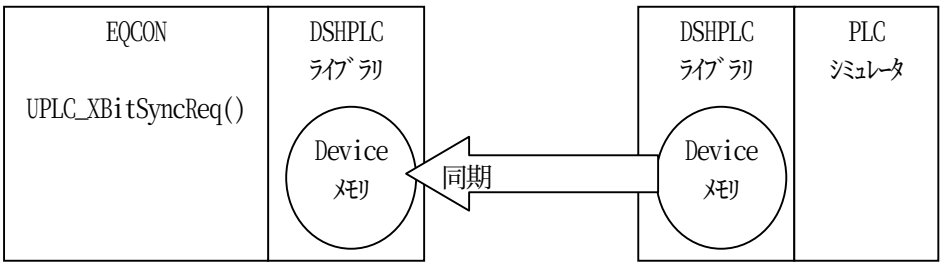
なし。

（3）戻り値

戻り値	意味
0	正常に同期がとれました。
-1	通信が接続していなかった。

（4）説明

PLC シミュレータが有する入力デバイスの On/Off 状態情報を送信要求するための関数です。
本関数は、PLC シミュレータに同期要求を行なうだけで、実際に同期のための通信が行なわれるのはその後です。本関数を実行約１秒間経過後にデバイス On/Off の参照をするようにしてください。



４．３　UPLC_YBitSyncReq（）- PLC シミュレータとの出力デバイス同期要求関数

（1）呼出書式

```
API int WINAPI UPLC_YBitSyncReq( );
```

（2）引数

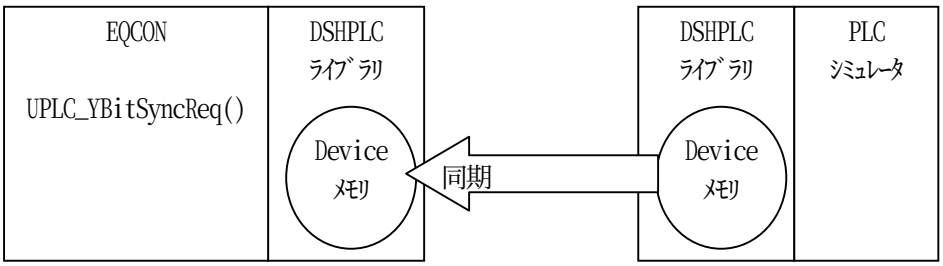
なし。

（3）戻り値

戻り値	意味
0	正常に同期がとれました。
-1	通信が接続していなかった。

（4）説明

PLC シミュレータが有する出力デバイスの On/Off 状態情報を送信要求するための関数です。
本関数は、PLC シミュレータに同期要求を行なうだけで、実際に同期のための通信が行なわれるのはその後です。本関数を実行約１秒間経過後にデバイス On/Off の参照をするようにしてください。



４．４　UPLC_CarIdRead（）- キャリア ID 読込み関数

（1）呼出書式

```
API int WINAPI UPLC_CarIdRead(
    char  *buff                    // CARID 格納領域
);
```

（2）引数

buff

読み込んだキャリア ID 情報を格納するバッファのポインタです。

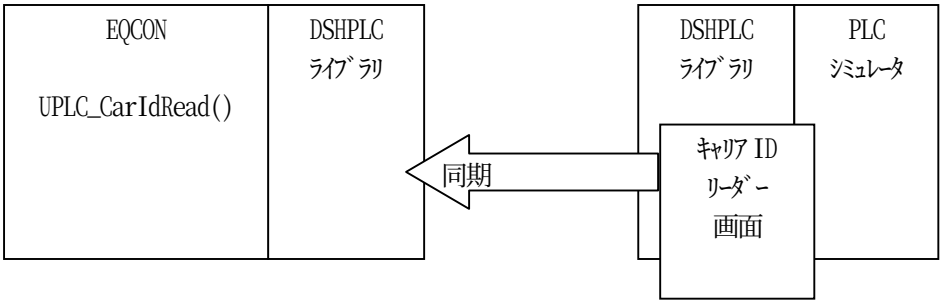
（3）戻り値

戻り値	意味
0	正常に読込みました。。
-1	通信が接続していなかった。

（4）説明

PLC シミュレータが有する仮想キャリア ID リーダーの画面で、現在選択されているキャリア ID を読み込みます。キャリア ID 情報は文字列情報であり、buff 領域に格納されます。

buff にはキャリア ID を格納するに必要なサイズのバッファを準備してください。



４．５　UPLC_SlotmapRead（）- スロットマップ読込み関数

（1）呼出書式

```
API int WINAPI UPLC_SlotmapRead(
    int    *slotmap                    // ｽﾛｯﾄﾏｯﾌﾟ(25 ｽﾛｯﾄ分)格納領域
);
```

（2）引数

slotmap

読み込んだスロットマップ２５個分の情報を格納するバッファのポインタです。

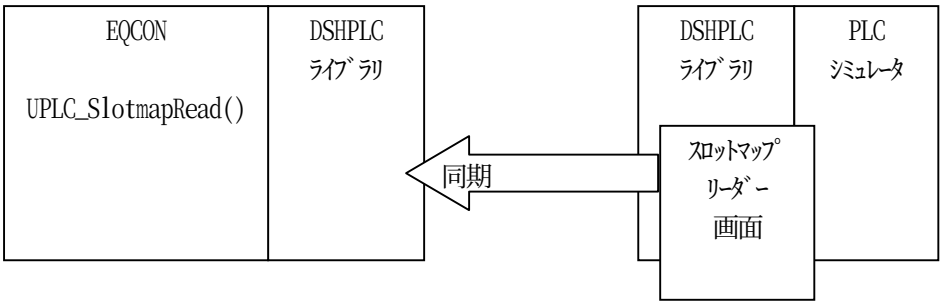
（3）戻り値

戻り値	意味
0	正常に読込みました。
-1	通信が接続していなかった。

（4）説明

PLC シミュレータが有する仮想スロットマップリーダーの画面で、25 スロット分のスロット情報を読み込みます。個々のスロットマップ情報は、整数の１/0 情報であり、slotmap 領域に格納されます。

slotmap には 25 スロット分の情報を格納するに必要なサイズの領域を準備してください。



４．６ UPLC_ProcDataRead（）- プロセスデータ読込み関数

（1）呼出書式

```
API int WINAPI UPLC_ProcDataRead(
    int    index;                    // 0,1,2..6 読込むﾌﾟﾛｾｽﾃﾞｰﾀの順位
    char   *buff                     // ﾌﾟﾛｾｽﾃﾞｰﾀ格納領域
);
```

（2）引数

index

PLC シミュレータ画面上にある７個あるデータの中の１つを表示順位で指定します。
０が先頭で、６が最後のデータになります。

buff

読み込んだプロセスデータ情報を格納するバッファのポインタです。

（3）戻り値

戻り値	意味
0	正常に読込みました。
-1	通信が接続していなかった。

（4）説明

PLC シミュレータのプロセスデータ表示画面に表示されている７個の中の１個のプロセスデータを表示されている順位の番号０から６の値で指定し、そのプロセスデータを buff 領域に読込みます。
順位の値は、7 の mod(7 で割った余り)した値を使用します。
プロセスデータは文字列情報です。

buff にはプロセスデータを格納するに必要なサイズのバッファを準備してください。