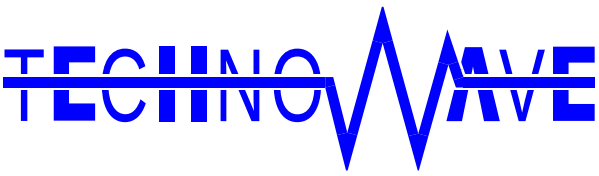
USBM ライブラリ　関数リファレンス



テクノウェーブ株式会社

目次

1. はじめに

□ 『USBM ライブラリ』について

『USBM ライブラリ』は、弊社製品 M3069 マイコンボードシリーズの制御用ライブラリです。ライブラリ内の関数は一部を除いて、対応製品全てでご利用になれます。本リファレンスは『USBM ライブラリ』の個々の関数について使用方法を解説しています。

表 1　対応製品

『USBM3069』 / 『USBM3069F』 / 『USBM3069-S』 / 『USBM3069-SL』 / 『USBM3069-HS』 / 『USBM3069-HSL』 / 『LANM3069』 / 『LANM3069-S』 / 『LANM3069-SL』 / 『LANM3069C』 / 『LANM3069C-S』 / 『LANM3069C-SL』 / 『LANM3069D-S』 / 『LANM3069D-SL』

□ 本リファレンスの対応ファイルとバージョンについて

本リファレンスは下記のバージョンのファイル内容を元に記載されています。過去のバージョンについては記載内容と異なる場合がございますのでご注意ください。

表 2　対応するライブラリのバージョン

ファイル名	バージョン番号	対応 OS
USBM3069.DLL	4.0.x.x [1]	Windows XP 以降

□ 本リファレンス内の表記について

本リファレンス内では対応製品を「デバイス」と表記します。それぞれの製品のホストインタフェースを区別する場合や各関数の対応製品を示す場合には表 3 に従います。

表 3　ホストインタフェースの表記・表示方法

説明文での表記	各関数の対応表示	対応製品
USB デバイス	USB	『USBM3069』 / 『USBM3069F』 / 『USBM3069-S』 / 『USBM3069-SL』
HS デバイス	HS	『USBM3069-HS』 / 『USBM3069-HSL』
LAN デバイス	LAN	『LANM3069』 / 『LANM3069-S』 / 『LANM3069-SL』 / 『LANM3069C』 / 『LANM3069C-S』 / 『LANM3069C-SL』 / 『LANM3069D-S』 / 『LANM3069D-SL』

本リファレンス内でハードウェアの電気的状態について記述する必要がある場合には、下記のように表記します。

[1] x にはより細かなバージョンを示す数値が入ります。

表 4　電気的状態の表記方法

表記	状態
″ON″	電流が流れている状態、スイッチが閉じている状態、オープンコレクタ(オープンドレイン)出力がシンク出力している状態。
″OFF″	電流が流れていない状態、スイッチが開いている状態、オープンコレクタ(オープンドレイン)出力がハイインピーダンスの状態。
″Hi″	電圧がロジックレベルのハイレベルに相当する状態。
″Lo″	電圧がロジックレベルのローレベルに相当する状態。
″Z″	端子がハイインピーダンスの状態。

また、数値について「0x」、「&H」 、「H'」はいずれもそれに続く数値が 16 進数であることを表します。″0x10″、″&H1F″、″H'20″などはいずれも 16 進数です。同様に「B'」に続く数値は 2 進数であることを表します。例えば″B'01000001″のように表記されます。数値の最初に特別な表記が無い場合は 10 進数です。

2. 関数リファレンス

各関数の説明は、C 言語、Visual Basic.NET 以降、Visual Basic® 6.0 以前、それぞれにおけるプロトタイプ、変数の説明、動作説明の順になっています。また、関数で構造体を使用する場合には、その節の先頭で説明を加えています。

ほとんどの関数の戻り値は 32 ビットの整数で関数の実行結果を表します（以下参照）。関数がそれ以外の特別な戻り値を返す場合は、各関数の動作説明の欄で内容を示します。

また、各関数の対応製品を示すために、関数名の後に表 3 の対応表示を行っています。

□ 関数の戻り値の意味

以下に主な戻り値の意味を示します。尚、戻り値を示す各定数は各言語用の定義ファイル（拡張子が「.h」、「.bas」、「.vb」のファイル）中で定義されています。

表 5　関数の戻り値

定数	値	意味
TW_OK	0x00000000	正常終了
TW_INVALID_HANDLE	0x00000001	デバイスのハンドルが無効
TW_DEVICE_NOT_FOUND	0x00000002	デバイスが見つからない
TW_IO_ERROR	0x00000004	送受信中にエラーが発生した
TW_INSUFFICIENT_RESOURCES	0x00000005	リソースエラー（デバイスの最大接続数を超えた場合など）
TW_INVALID_ARGS	0x00000010	関数に渡された引数が無効
TW_NOT_SUPPORTED	0x00000011	サポートされない機能
TW_OTHER_ERROR	0x00000012	その他のエラー
TW_TIMEOUT	0xffff0001	送信または受信処理がタイムアウトした
TW_FILE_ERROR	0xffff0002	ファイル操作に関するエラーが発生した
TW_MEMORY_ERROR	0xffff0003	メモリの確保に失敗した
TW_DATA_NOT_FOUND	0xffff0004	有効なデータが見つからなかった
TW_SOCKET_ERROR	0xffff0005	Winsock のエラー（多くの場合 WSAGetLastError() を呼び出すとさらに詳しい情報を得ることができます）
TW_ACCESS_DENIED	0xffff0006	デバイスとの認証作業に失敗した
TW_NOT_SUPPORTED_MODE	0xffff0007	関数をサポートしないモードでデバイスに接続している
TW_FLASH_MODE_DEVICE	0xffff0008	フラッシュ書換えモードのデバイスのため制御できない
TW_ATF_ERR_FILE_VERSION	0xffff0101	対応するバージョンより新しい ATF ファイルフォーマット
TW_ATF_ERR_ILLEGAL_FILE	0xffff0102	ATF ファイルの内容が不正
TW_ATF_ERR_SERVICE_VERSION	0xffff0103	ファームのバージョンが ATF ファイルの要求より古い
TW_FLASH_ERR_DOWNLOAD	0xffff0201	フラッシュ書換えルーチンの転送エラー
TW_FLASH_ERR_INIT	0xffff0202	フラッシュ書換えルーチンの初期化エラー
TW_FLASH_ERR_ERASE	0xffff0203	フラッシュ消去時に予期しないエラーが発生した
TW_FLASH_ERR_WRITE	0xffff0204	フラッシュ書き込み時に予期しないエラーが発生した
TW_FLASH_ERR_COMMAND	0xffff0205	フラッシュ書換えモードで定義されないコマンドを受信した
TW_FLASH_ERR_CHKSUM	0xffff0206	書き込みデータのチェックサムが合わない
TW_FLASH_ERR_SIZE	0xffff0210	データサイズのエラー（128 バイト単位になっていないなど）
TW_FLASH_ERR_ADDRESS	0xffff0211	アドレス指定のエラー（128 バイト境界になっていないなど）
TW_FLASH_ERR_NOT_ERASED	0xffff0212	未消去の領域に書き込みを行おうとした

□ C/C++での文字列の扱いについて

C/C++でライブラリと文字列の受け渡しを行う場合、Windows CE ではUnicode を使用し、その他の Windows ではマルチバイト文字を使用します。ライブラリでは *TW_CSTR* のような独自の型を定義していますが、これらはコンパイル環境によって以下のように変換されます。

表 6　文字変数の定義

型	WINCE が定義(define)されている	WINCE が定義(define)されていない
TW_CSTR	LPCWSTR	LPCSTR
TW_STR	LPWSTR	LPSTR
TW_CHAR	WCHAR	char

□ マルチスレッドプログラムからの呼び出しについて

USBM3069.DLL の 3.4.2.1 以降では、複数のスレッドからの関数呼び出しをサポートしていますが、デバイスとの通信仕様により、1 つのデバイスを複数のスレッドから同時に制御することはできません。何らかの理由により、複数のスレッドから 1 つのデバイスにアクセスする必要がある場合には、クリティカルセクションなどを使用することにより、ライブラリ関数の呼び出しをシリアル化し、複数のスレッドが同時に 1 つのデバイスにアクセスしないようにプログラムしてください。

1 つのスレッドが 1 つのデバイスのみを制御する場合は、複数のスレッドから同時にライブラリ関数を呼び出しても問題ありません。

□ デバイスへの接続／切断に関する関数

USBM_OpenA() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_OpenA(TW_HANDLE *phDev, TW_CSTR *pIID, long Opt)
VB	Function USBM_OpenA(ByRef phDev As System.IntPtr, ByVal pIID As String, ByVal Opt As Integer) As Integer
VBA	Function USBM_OpenA(ByRef phDev As Long, ByVal pIID As String, ByVal Opt As Long) As Long

phDev ： [出力]ハンドルの格納先
pIID ： [入力]USB のシリアル番号、IP アドレス、または、ドメイン名(NULL で終わる文字列)
Opt ： インタフェースと接続オプションを指定
インタフェースは以下のいずれかを指定
USBM_IF_USB(0x80000000) ： USB デバイスに接続する
USBM_IF_LAN(0x40000000) ： LAN デバイスに接続する
USBM_IF_HS(0x20000000) ： HS デバイスに接続する
USBM_IF_ANY(0xff000000) ： インタフェースを問わない

上記インタフェースに加えて以下のオプションを OR で追加可能
USBM_MODE_FLASH(0x00800000) ： フラッシュ書換えモードのデバイスに接続
USBM_LIST_UPDATE(0x00000001) ： LAN デバイスの情報を保存した内部テーブルを更新
(LAN デバイスのみ有効)
USBM_MODE_BUS16(0x00400000) ： USB インタフェース IC との接続に 16 ビットアクセスを使用
(HS デバイスのみ有効)

pIID が NULL か""の場合、接続できた最初のデバイスのハンドルを取得します。それ以外では指定された ID と一致するデバイスと接続します。pIID に指定できる ID は、USB デバイス、HS デバイスの場合 USB のシリアル番号、LAN デバイスの場合、IP アドレス、または、ドメイン名です。さらに LAN デバイスでは、指定した IP アドレスやドメイン名の後に":"(コロン)とポート番号を続けることで、ポート番号を指定することができます(例:"lanx01.mydomain.co.jp:49153")。

Opt は USBM_IF_USB、USBM_IF_LAN または USBM_IF_HS でインタフェースを指定します。これらは OR で結合して指定することが可能です。

USBM_IF_HS を指定した場合には USBM_MODE_BUS16 が指定できます。このオプションで接続した場合には、マイコンから USB インタフェース IC へのアクセスが 16 ビット幅で行われ、USBM_PortBWrite()などのブロック転送命令が高速になります(それ以外の命令はわずかですが遅くなります)。また、このオプションを指定した場合は、P40-P47 が D0-D7 のデータバスとして使用されポートとして使用できなくなりますのでご注意ください。

USBM_IF_LAN を指定した場合には USBM_LIST_UPDATE を結合することができます。このオプションにより LAN デバイスの情報を格納した内部テーブルを更新することができます。USBM_LIST_UPDATE が指定されない場合には、既に取得した内部テーブルの情報からデバイスを探します。
内部テーブルの更新には UDP のブロードキャストが用いられ、約 1.5 秒の時間を要します。
複数のインタフェースを指定して呼び出した場合、USB デバイス、HS デバイス、LAN デバイスの順に接続を試みます。

USBM_MODE_FLASH を指定するとフラッシュ書換えモードのデバイスに接続します。LAN デイバスの場合は pIID に指定した IP アドレスを一時的にデバイスに割り当てて通信を行います。pIID の指定がない場合には DHCP を利用して IP の取得を試みます。

USBM_Open() USB HS LAN

言語	関数宣言
C/C++	TW_HANDLE USBM_Open(TW_CSTR *pIID)
VB	Function USBM_Open(ByVal pIID As String) As System.IntPtr
VBA	Function USBM_Open(ByVal pIID As String) As Long

pIID　：[入力]USB のシリアル番号または IP アドレス(NULL で終わる文字列)

戻り値：接続できたデバイスのハンドル

pIID が NULL か""の場合、接続できた最初のデバイスのハンドルを返します。それ以外では指定された ID と一致するデバイスと接続します。pIID が指定されない場合、USB デバイス、HS デバイス、LAN デバイスの順で接続を試みます。指定デバイスと接続できなかった場合は 0 を返します。
LAN デバイスとの接続は設定ファイルで抑止することができます。

USBM_OpenByPI() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_OpenByPI(TW_HANDLE *phDev, TW_CSTR *pUuid, DWORD Number, long Opt)
VB	Function USBM_OpenByPI(ByRef phDev As System.IntPtr, ByVal pUuid As String, ByVal Number As Integer, ByVal Opt As Integer) As Integer
VBA	Function USBM_OpenByPI(ByRef phDev As Long, ByVal pUuid As String, ByVal Number As Long, ByVal Opt As Long) As Long

phDev　：[出力]ハンドルの格納先
pUuid　：[入力]UUID を指定します。指定が無い場合には NULL または""とします。
Number：Number を指定します。指定が無い場合には 0 とします。
Opt　　：以下の値の組み合わせ
　　USBM_IF_USB(0x80000000)：USB デバイスに接続する
　　USBM_IF_LAN(0x40000000)：LAN デバイスに接続する
　　USBM_IF_HS(0x20000000)　：HS デバイスに接続する
　　USBM_IF_ANY(0xff000000)：インタフェースを問わない

　　上記インタフェースに加えて以下のオプションを OR で追加可能
　　USBM_LIST_UPDATE(0x00000001)：LAN デバイスの情報を保存した内部テーブルを更新
　　(LAN デバイスのみ有効)
　　USBM_MODE_BUS16(0x00400000)　：USB インタフェース IC との接続に 16 ビットアクセスを使用
　　(HS デバイスのみ有効)

PI エリア内の UUID と Number が指定と一致するデバイスと接続します（PI エリアについての詳細は製品マニュアルを参照してください)。
Opt は USBM_IF_USB、USBM_IF_LAN または USBM_IF_HS でインタフェースを指定します。これらは OR で結合して指定することが可能です。

USBM_IF_HS を指定した場合には USBM_MODE_BUS16 が指定できます。このオプションで接続した場合には、マイコンから USB インタフェース IC へのアクセスが 16 ビット幅で行われ、USBM_PortBWrite()などのブロック転送命令が高速になります(それ以外の命令はわずかですが遅くなります)。また、このオプションを指定した場合は、P40-P47 が D0-D7 のデータバスとして使用されポートとして使用できなくなりますのでご注意ください。

USBM_IF_LAN を指定した場合には USBM_LIST_UPDATE を結合することができます。このオプションにより LAN デバイスの情報を格納した内部テーブルを更新することができます。USBM_LIST_UPDATE が指定されない場合には、既に取得した内部テーブルの情報からデバイスを探します。
内部テーブルの更新には UDP のブロードキャストが用いられ、約 1.5 秒の時間を要します。

複数のインタフェースを指定して呼び出した場合、USB デバイス、HS デバイス、LAN デバイスの順に接続を試みます。

USBM_Close() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_Close(TW_HANDLE hDev)
VB	Function USBM_Close(ByVal hDev As System.IntPtr) As Integer
VBA	Function USBM_Close(ByVal hDev As Long) As Long

hDev ： デバイスのハンドル

デバイスとの接続を切ります。制御を終了する場合に呼び出します。

USBM_CloseAll() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_CloseAll()
VB	Function USBM_CloseAll() As Integer
VBA	Function USBM_CloseAll() As Long

現在のプロセスが接続している全てのデバイスの制御を終了します。

USBM_ListDevicesA() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_ListDevicesA(long *pnDev, TW_STR *pIID, long *pnIID, long Opt)
VB	Function USBM_ListDevicesA(ByRef pnDev As Integer, ByVal pIID As String, ByRef pnIID As Integer, ByVal Opt As Integer) As Integer
VBA	Function USBM_ListDevicesA(ByRef pnDev As Long, ByVal pIID As String, ByRef pnIID As Long, ByVal Opt As Long) As Long

pnDev ： [出力]デバイス数の格納先
pIID ： [出力]USB のシリアル番号または IP アドレスの格納先
pnIID ： [入力]pIID に用意された文字数
[出力]pIID に必要な文字数、または実際にコピーした文字数
Opt ： 次のオプションのどれか
USBM_IF_USB(0x80000000) ： USB デバイスをリストアップする
USBM_IF_LAN(0x40000000) ： LAN デバイスをリストアップする
USBM_IF_HS(0x20000000) ： HS デバイスをリストアップする

USBM_IF_LAN オプションには下のオプションを OR で追加可能
USBM_LIST_UPDATE(0x00000001) ： LAN デバイスの情報を保存した内部テーブルを更新

接続されているデバイス数を調べ、USB のシリアル番号、または IP アドレスの情報を取得します。
pIID には Opt として USBM_IF_USB または USB_IF_HS を選択した場合 USB デバイスのシリアル番号が USBM_IF_LAN を指定した場合には IP アドレスが格納されます。シリアル番号と IP アドレスは文字列で格納されます。複数のデバイスが見つかった場合には、それぞれのデバイスの ID 文字列間に' '(スペース)が挿入され、最後の要素の後には'¥0'が挿入されます。

pnIID は入力として pIID に用意された文字数を渡します。出力としては全ての ID 文字列が格納された場合にはその文字数が、格納しきれなかった場合には格納するのに十分な文字数を示します。

Opt は USBM_IF_USB、USBM_IF_HS、または USBM_IF_LAN でインタフェースを特定します。複数を同時に指

定することはできません。USBM_IF_LAN を指定した場合には USBM_LIST_UPDATE を結合することができます。このオプションにより LAN デバイスの情報を格納した内部テーブルを更新することができます。USBM_LIST_UPDATE が指定されない場合には、既に取得した内部テーブルの情報からデバイスを探します。内部テーブルの更新には UDP のブロードキャストが用いられ、約 1.5 秒の時間を要します。
Opt 以外の引数は不要な場合 NULL とすることができます。

USBM_ListDevices() USB

言語	関数宣言
C/C++	long USBM_ListDevices(TW_CHAR (*pSerial)[9], long nSerial)
VB	Function USBM_ListDevices(ByRef pSerial() As String) As Integer
VBA	Function USBM_ListDevices(ByRef pSerial() As String) As Long

(*pSerial)[9] : [出力]9 文字の文字列へのポインタ、シリアル番号を格納します
nSerial : シリアル番号を格納する配列の数

戻り値 : パソコンに接続されている USB デバイスの数

接続されている USB デバイスの数を返します。pSerial が NULL でなければ nSerial 個までシリアル番号を pSerial に格納します。シリアル番号は製品に予め付けられている英数字 8 文字(ANSI)の NULL 終端文字列で一意な値です。

USBM_ListDevicesByID() USB

言語	関数宣言
C/C++	long USBM_ListDevicesByID(WORD VID, WORD PID, TW_CHAR (*pSerial)[9], long nSerial)
VB	Function USBM_ListDevicesByID(ByVal VID As Short, ByVal PID As Short, ByRef pSerial() As String) As Integer
VBA	Function USBM_ListDevicesByID(ByVal VID As Integer, ByVal PID As Integer, ByRef pSerial() As String) As Long

VID : ベンダーID
PID : プロダクト ID
(*pSerial)[9] : [出力]9 バイトの文字列へのポインタ、シリアル番号を格納します
nSerial : 上の配列の数

戻り値 : パソコンに接続されているデバイスの数

指定されたベンダーID とプロダクト ID に該当するデバイスを検索し、接続されているデバイスの数を返します。pSerial が NULL でなければ nSerial 個までシリアル番号を pSerial に格納します。シリアル番号は製品に予め付けられている英数字 8 文字(ANSI)の文字列で一意な値です。

USBM_OpenByID() USB

言語	関数宣言
C/C++	TW_HANDLE USBM_OpenByID(WORD VID, WORD PID, TW_CSTR *pSerial)
VB	Function USBM_OpenByID(ByVal VID As Short, ByVal PID As Short, ByVal pSerial As String) As System.IntPtr
VBA	Function USBM_OpenByID(ByVal VID As Integer, ByVal PID As Integer, ByVal pSerial As String) As Long

VID : ベンダーID
PID : プロダクト ID

pSerial　：[入力]シリアル番号､NULL で終わる文字列

戻り値　：接続できたデバイスのハンドル

指定されたベンダーID とプロダクト ID に該当するデバイスを検索しオープンします。pSerial が NULL の場合､または""が渡された場合、オープンできた最初のデバイスのハンドルを返します。それ以外ではシリアル番号が一致したデバイスをオープンします。指定デバイスがオープンできなかった場合は 0 を返します。

USBM_OpenByUA() USB HS

言語	関数宣言
C/C++	TW_HANDLE USBM_OpenByUA(TW_CSTR *pUA, long nUA)
VB	Function USBM_OpenByUA (ByVal pUA As String, ByVal nUA As Integer) As System.IntPtr
VBA	Function USBM_OpenByUA (ByVal pUA As String, ByVal nUA As Long) As Long

pUA　：[入力]ユーザーエリアと比較する文字列
nUA　：比較する文字数

戻り値：接続できたデバイスのハンドル

ユーザーエリアの値を読み出し、指定文字列と一致したデバイスのハンドルを返します。指定デバイスがオープンできなかった場合は 0 を返します。

ユーザーエリアはデバイス上のコンフィギュレーション情報用 EEPROM 領域に確保された領域で、設定ツールを使用して文字列を書き込むことができます。使用できるユーザーエリアは 8 バイトです。

USBM_Listen() LAN

言語	関数宣言
C/C++	TW_STATUS USBM_Listen(UINT_PTR *pListenSocket, TW_CSTR pLocalIP, DWORD PortNumber)
VB	Function USBM_Listen(ByRef pListenSocket As System.IntPtr, ByVal pLocalIP As String, ByVal PortNumber As Integer) As Integer
VBA	Function USBM_Listen(ByRef pListenSocket As Long, ByVal pLocalIP As String, ByVal PortNumber As Long) As Long

pListenSocket：[出力]接続待ちソケットの格納先
pLocalIP　　：[入力]接続待ちを行うローカル(パソコン側)の IP アドレス
PortNumber　：接続待ちを行うポート番号

指定のポート番号をオープンし、クライアントモードのデバイスの接続を待ちます。
pListenSocket に返された値を USBM_Accept() に渡すことで、ここで設定したポートへの接続を受け入れることができます。
pLocalIP はパソコンのネットワークカードが複数ある場合に、接続待ちを行う IP アドレスを指定します。特に指定が無い場合は NULL または""とすることができます。

USBM_Accept() LAN

言語	関数宣言
C/C++	TW_STATUS USBM_Accept(UINT_PTR ListenSocket, TW_HANDLE *phDev)
VB	Function USBM_Accept (ByVal ListenSocket As System.IntPtr, ByRef phDev As System.IntPtr) As Integer
VBA	Function USBM_Accept(ByVal ListenSocket As Long, ByRef phDev As Long) As Long

ListenSocket : 接続待ちソケット
phDev : [出力]デバイスへのハンドルの格納先

USBM_Listen()でオープンした接続待ちソケットに対して、接続要求があれば受け入れてデバイスへのハンドルを返します。
接続要求が無い場合は TW_DEVICE_NOT_FOUND を返します。接続待ちを行う間はこの関数を繰り返し呼び出して、接続要求の有無を確認してください。

USBM_CloseListenSocket() LAN

言語	関数宣言
C/C++	TW_STATUS USBM_CloseListenSocket(UINT_PTR ListenSocket)
VB	Function USBM_CloseListenSocket(ByVal ListenSocket As System.IntPtr) As Integer
VBA	Function USBM_CloseListenSocket(ByVal ListenSocket As Long) As Long

ListenSocket : 接続待ちソケット

USBM_Listen()でオープンした接続待ちのソケットをクローズします。

□ デバイスの状態・初期化に関する関数

USBM_PRODUCT_INFO 構造体

言語	宣言

C/C++	typedef struct
	{
	  WORD Empty;
	  WORD wRsv;
	  UUID ID;
	  DWORD Number;
	  char Description[32];
	  char Manufacture[32];
	  BYTE Reserve[40];
	} USBM_PRODUCT_INFO;

VB	Public Structure USBM_PRODUCT_INFO
	  Public Empty As Short
	  Public wRsv As Short
	  Public ID As TWB_UUID
	  Public Number As Integer
	  <MarshalAs(UnmanagedType.ByValTStr, SizeConst:=32)> _
	  Public Description As String
	  <MarshalAs(UnmanagedType.ByValTStr, SizeConst:=32)> _
	  Public Manufacture As String
	  <MarshalAs(UnmanagedType.ByValArray, SizeConst:=40)> _
	  Public Reserve() As Byte
	End Structure

VBA	Public Type USBM_PRODUCT_INFO
	  Empty As Integer
	  wRsv As Integer
	  ID As TWB_UUID
	  Number As Long
	  Description As String * 32
	  Manufacture As String * 32
	  Reserve(39) As Byte
	End Type

Empty : 管理用の領域です。0 となります
wRsv : 予約
ID : デバイスに書き込まれた UUID です。UUID または TWB_UUID 構造体として格納されています
Number : M3069PIWriter の[Number]に設定された値です
Description : M3069PIWriter の[製品名]に設定された値です
Manufacture : M3069PIWriter の[製造元]に設定された値です
Reserve : 予約

USBM_ReadPI() 関数で使用する構造体です。設定ツール(M3069PIWriter)で書き込んだ製品の識別情報を格納します。PI エリアの詳細については製品マニュアルを参照してください。

USBM_ReadPI() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_ReadPI(TW_HANDLE hDev, USBM_PRODUCT_INFO *pInfo)
VB	Function USBM_ReadPI(ByVal hDev As System.IntPtr, ByRef pInfo As USBM_PRODUCT_INFO) As Integer
VBA	Function USBM_ReadPI(ByVal hDev As Long, ByRef pInfo As USBM_PRODUCT_INFO) As Long

hDev ： デバイスのハンドル
pInfo ： [出力]製品情報の格納先

PI エリアから製品情報を読み出します。

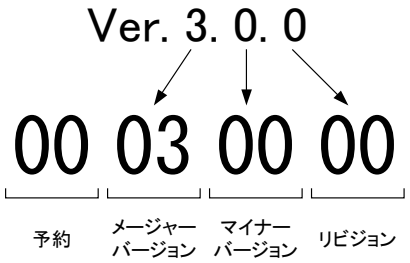
※この関数は Ver.2.1.1 以降のファームウェアが必要です。

USBM_ReadVersionA() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_ReadVersionA(TW_HANDLE hDev, DWORD *pVersion)
VB	Function USBM_ReadVersionA(ByVal hDev As System.IntPtr, ByRef pVersion As Integer) As Integer
VBA	Function USBM_ReadVersionA(ByVal hDev As Long, ByRef pVersion As Long) As Long

hDev ： デバイスのハンドル
pVersion ： [出力]読み出したバージョンの格納先

USBM_ReadVersion()よりも詳しいファームウェアバージョンを取得します。返される値の最上位 8 ビットは予約です。次の 8 ビットはメジャーバージョンです。大幅な機能変更の際に更新されます。次の 8 ビットはマイナーバージョンです。比較的小さな機能変更の際に更新されます。最下位 8 ビットはリビジョンです。バグフィックスや各インタフェース固有の機能変更の際に更新されます。例えばファームウェアバージョンが Ver.3.0.0 の場合、pVersion には H'00030000 が格納されます。



USBM_ReadVersion() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_ReadVersion(TW_HANDLE hDev, DWORD *pVer)
VB	Function USBM_ReadVersion(ByVal hDev As System.IntPtr, ByRef pVer As Integer) As Integer
VBA	Function USBM_ReadVersion(ByVal hDev As Long, ByRef pVer As Long) As Long

hDev ： デバイスのハンドル
pVer ： [出力]読み出したバージョンの格納先

ハードウェアのバージョンの上位 2 桁を読み取ります。バージョンは、x.x の形式を 10 倍にして整数として返します。

USBM_ReadStatus() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_ReadStatus(TW_HANDLE hDev, DWORD *pStat)
VB	Function USBM_ReadStatus(ByVal hDev As System.IntPtr, ByRef pStat As Integer) As Integer
VBA	Function USBM_ReadStatus(ByVal hDev As Long, ByRef pStat As Long) As Long

hDev ： デバイスのハンドル
pStat ： [出力]読み出したステータスの格納先へのポインタ
USBM_STS_TIMEOUT 処理がタイムアウトした
USBM_STS_ILLEGAL_ACCESS 不正なアドレスへのアクセス

デバイスのステータスを読み取ります。上で定義されるステータスビットの組み合わせが返されます。一度読み出すとデバイスのステータスはクリアされます。

USBM_InitializeA() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_InitializeA(TW_HANDLE hDev, DWORD InitOption)
VB	Function USBM_InitializeA(ByVal hDev As System.IntPtr, ByVal InitOption As Integer) As Integer
VBA	Function USBM_InitializeA(ByVal hDev As Long, ByVal InitOption As Long) As Long

hDev ： デバイスのハンドル
InitOption ： 初期化する機能を指定

USBM_INIT_PORT_DIR(0x0001)	ポートの方向、プルアップを初期化
USBM_INIT_PORT_DATA(0x0002)	ポートのデータを初期化
USBM_INIT_BUS(0x0004)	外部バス設定を初期化
USBM_INIT_DMA(0x0008)	DMA を初期化
USBM_INIT_TIMER(0x0010)	16 ビットタイマを初期化
USBM_INIT_AD(0x0020)	AD コンバータを初期化
USBM_INIT_SCI(0x0040)	シリアルポートを初期化
USBM_INIT_PC(0x0080)	パルスカウンタを初期化
USBM_INIT_TCPY(0x0100)	タイマコピーを初期化
USBM_INIT_ALL(0xffffffff)	全ての機能を初期化

デバイスを初期化します。InitOption で初期化する機能を選択できます。

※この関数は Ver.3.2.1 以降のファームウェアが必要です。

USBM_Initialize() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_Initialize(TW_HANDLE hDev)
VB	Function USBM_Initialize(ByVal hDev As System.IntPtr) As Integer
VBA	Function USBM_Initialize(ByVal hDev As Long) As Long

hDev ： デバイスのハンドル

デバイスを初期化します。リセットと違いシステムファームで使用されないレジスタなどは初期化されません。

□ バス操作関数

USBM_AddressEnable() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_AddressEnable(TW_HANDLE hDev, long nBits)
VB	Function USBM_AddressEnable(ByVal hDev As System.IntPtr, ByVal nBits As Integer) As Integer
VBA	Function USBM_AddressEnable(ByVal hDev As Long, ByVal nBits As Long) As Long

hDev ： デバイスのハンドル
nBits ： 有効にするビット数(0,8,16,20)

アドレス出力を有効/無効にします。LSB から指定ビット数が有効になります。アドレスバスはポート 1,ポート 2,ポート 5 と共用です。アドレスを出力すると入力ポートとしては使用できませんので注意してください。アドレスは一部だけを出力できますので出力していないポートは入力ポートのまま使用できます。

USBM_BusSetWait() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_BusSetWait(TW_HANDLE hDev, long Area, BYTE Wait)
VB	Function USBM_BusSetWait(ByVal hDev As System.IntPtr, ByVal Area As Integer, ByVal Wait As Byte) As Integer
VBA	Function USBM_BusSetWait(ByVal hDev As Long, ByVal Area As Long, ByVal Wait As Byte) As Long

hDev ： デバイスのハンドル
Area ： メモリエリア
　　USBM_AREA0(0x01) エリア 0
　　USBM_AREA2(0x04) エリア 2
　　USBM_AREA3(0x08) エリア 3
　　USBM_AREA5(0x10) エリア 5

Wait ： ウェイト数
　　USBM_BUS_WAIT0(0)　ウェイトなし
　　USBM_BUS_WAIT1(1)　ウェイト 1
　　USBM_BUS_WAIT2(2)　ウェイト 2
　　USBM_BUS_WAIT3(3)　ウェイト 3
　　USBM_BUS_2STATE(4) 2 ステートアクセス

メモリエリアのソフトウェアウェイトを設定します。
Wait を 0-3 の範囲とすると指定エリアが 3 ステートアクセスに設定され、指定のウェイトが挿入されます。
Wait を USBM_BUS_2STATE とすると 2 ステートアクセスに設定されます。この場合ウェイトを指定することはできません。
アクセス速度は USBM_BUS_2STATE とした場合が最も高速で、それ以外は 0,1,2,3 の順となり、3 の場合が最も低速です。初期状態ではウェイト 1 に設定されています。

USBM_CSEnable() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_CSEnable(TW_HANDLE hDev, long Area)
VB	Function USBM_CSEnable(ByVal hDev As System.IntPtr, ByVal Area As Integer) As Integer
VBA	Function USBM_CSEnable(ByVal hDev As Long, ByVal Area As Long) As Long

hDev : デバイスのハンドル
Area : CS 信号を有効にするエリア
USBM_AREA2 CS2#を出力
USBM_AREA3 CS3#を出力

CS 信号を有効にするエリアを指定します。使用できる CS 信号のうち CS0#,CS5#は常に出力されますので USBM_AREA2 と USBM_AREA3 が指定できます。両方出力する場合は OR で結合して指定してください(USBM_AREA2 | USBM_AREA3)。CS2#を有効にした場合 PC3#が、CS3#を有効にした場合 PC2#は使用できませんので該当のパルスカウンタは必ず無効にしてください。

USBM_BusSetWidth() HS

言語	関数宣言
C/C++	TW_STATUS USBM_BusSetWidth(TW_HANDLE hDev, long Area, long Bus16)
VB	Function USBM_BusSetWidth(ByVal hDev As System.IntPtr, ByVal Area As Integer, ByVal Bus16 As Integer) As Integer
VBA	Function USBM_BusSetWidth(ByVal hDev As Long, ByVal Area As Long, ByVal Bus16 As Long) As Long

hDev : デバイスのハンドル
Area : バス幅を指定するメモリエリア
USBM_AREA0 エリア 0
USBM_AREA2 エリア 2
USBM_AREA3 エリア 3
USBM_AREA5 エリア 5

Bus16 : 0 の場合 8 ビット、0 以外の場合 16 ビットアクセス空間

外部アドレス空間のバス幅を指定します。

□ ポート操作関数

USBM_PortWrite8A() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_PortWrite8A(TW_HANDLE hDev, DWORD Port, BYTE Data, BYTE Mask)
VB	Function USBM_PortWrite8A(ByVal hDev As System.IntPtr, ByVal Port As Integer, ByVal Data As Byte, ByVal Mask As Byte) As Integer
VBA	Function USBM_PortWrite8A(ByVal hDev As Long, ByVal Port As Long, ByVal Data As Byte, ByVal Mask As Byte) As Long

hDev ： デバイスのハンドル
Port ： 書き込むアドレス
Data ： 書き込むデータ
Mask ： マスクデータ。1 のビットだけ反映されます。

マイコンの Port で指定されるアドレスに 1 バイトのデータを書き込みます。マイコンの制御レジスタにも書き込みが行えますので、定数以外のアドレスを指定する場合には注意が必要です。Mask が 0 となっているビットには影響を与えません。

※この関数には Ver.2.2.1 以降のファームウェアが必要です。

USBM_PortRead8() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_PortRead8(TW_HANDLE hDev, DWORD Port, BYTE *pData)
VB	Function USBM_PortRead8(ByVal hDev As System.IntPtr, ByVal Port As Integer, ByRef pData As Byte) As Integer
VBA	Function USBM_PortRead8(ByVal hDev As Long, ByVal Port As Long, ByRef pData As Byte) As Long

hDev ： デバイスのハンドル
Port ： 読み出すアドレス
pData ： [出力]読み出したデータの格納先

マイコンの Port で指定されるアドレスから 1 バイトのデータを読み出します。

USBM_PortWrite16() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_PortWrite16(TW_HANDLE hDev, DWORD Port, WORD Data)
VB	Function USBM_PortWrite16(ByVal hDev As System.IntPtr, ByVal Port As Integer, ByVal Data As Short) As Integer
VBA	Function USBM_PortWrite16(ByVal hDev As Long, ByVal Port As Long, ByVal Data As Integer) As Long

hDev ： デバイスのハンドル
Port ： 書き込むアドレス
Data ： 書き込むデータ

マイコンの Port で指定されるアドレスに 2 バイトのデータを書き込みます。16 ビットのデータを同時にアクセスしたい場合に使用します。マイコンの制御レジスタにも書き込みが行えますので、定数以外のアドレ

スを指定する場合には注意が必要です。Port アドレスは偶数番地でなければなりません。バイトアクセス専用の空間では 1 バイトずつ 2 回ライトされます。

USBM_PortRead16() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_PortRead16(TW_HANDLE hDev, DWORD Port, WORD *pData)
VB	Function USBM_PortRead16(ByVal hDev As System.IntPtr, ByVal Port As Integer, ByRef pData As Short) As Integer
VBA	Function USBM_PortRead16(ByVal hDev As Long, ByVal Port As Long, ByRef pData As Integer) As Long

hDev ： デバイスのハンドル
Port ： 読み出すアドレス
pData ： [出力]読み出したデータの格納先

マイコンの Port で指定されるアドレスから 2 バイトのデータを読み出します。Port アドレスは偶数番地でなければなりません。バイトアクセス専用の空間では 1 バイトずつ 2 回リードされます。

USBM_PortBWrite() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_PortBWrite(TW_HANDLE hDev, DWORD Port, void *pData, long nData, long Inc, BYTE Dma)
VB	Function USBM_PortBWrite(ByVal hDev As System.IntPtr, ByVal Port As Integer, ByVal Data() As Byte[2], ByVal nData As Integer, ByVal Inc As Integer, ByVal Dma As Byte) As Integer
VBA	Function USBM_PortBWrite(ByVal hDev As Long, ByVal Port As Long, ByRef pData() As Any, ByVal nData As Long, ByVal Inc As Long, ByVal Dma As Byte) As Long

hDev ： デバイスのハンドル
Port ： 書き込むアドレス
pData ： [入力]書き込むデータへのポインタ
nData ： データのバイト数(0～65536)
Inc ： マイコンのアドレスをインクリメントするかどうかを示すフラグ(TRUE,FALSE)
Dma ： DMA を使用するかどうかのフラグ(TURE,FALSE)

マイコンの Port で指定されるアドレスに nData で指定される大きさのデータを転送します。Inc を FALSE とするとマイコン側のアドレスがインクリメントされませんので FIFO のようなデバイスに書き込みを行えます。
Dma を TRUE とするとマイコン内蔵の DMA を使用してデータを転送します。DMA を使用すると転送速度が向上します。ただし、USB デバイスの場合、転送終了時に PA0/TCLKA 端子に TEND#信号を出力するので注意が必要です。PA0/TCLKA 端子に出力が出てはいけない場合には Dma=FALSE として呼び出してください。

[2] Short および Integer のオーバーロードが用意されています。

USBM_PortBRead() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_PortBRead(TW_HANDLE hDev, DWORD Port, void *pData, long nData, long Inc, BYTE Dma)
VB	Function USBM_PortBRead(ByVal hDev As System.IntPtr, ByVal Port As Integer, ByVal pData() As Byte[3], ByVal nData As Integer, ByVal Inc As Integer, ByVal Dma As Byte) As Integer
VBA	Function USBM_PortBRead(ByVal hDev As Long, ByVal Port As Long, ByRef pData() As Any, ByVal nData As Long, ByVal Inc As Long, ByVal Dma As Byte) As Long

hDev　：デバイスのハンドル
Port　：読み出すアドレス
pData　：[出力]読み出したデータの格納先へのポインタ
nData　：データのバイト数(0～65536)
Inc　：マイコンのアドレスをインクリメントするかどうかを示すフラグ(TRUE,FALSE)
Dma　：DMA を使用するかどうかのフラグ(TRUE,FALSE)

マイコンのPortで指定されるアドレスからnDataで指定される大きさのデータを読み出します。IncをFALSEとするとマイコン側のアドレスがインクリメントされませんので FIFO のようなデバイスから読み出しを行えます。
Dma を TRUE とするとマイコン内蔵の DMA を使用してデータを転送します。DMA を使用すると転送速度が向上します。ただし、USB デバイスの場合、転送終了時に PA1/TCLKB 端子に TEND#信号を出力するので注意が必要です。PA1/TCLKB 端子に出力が出てはいけない場合には Dma=FALSE として呼び出してください。

USBM_PortCopy8() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_PortCopy8(TW_HANDLE hDev, DWORD SrcPort, DWORD DstPort)
VB	Function USBM_PortCopy8(ByVal hDev As System.IntPtr, ByVal SrcPort As Integer, ByVal DstPort As Integer) As Integer
VBA	Function USBM_PortCopy8(ByVal hDev As Long, ByVal SrcPort As Long, ByVal DstPort As Long) As Long

hDev　：デバイスのハンドル
SrcPort：コピー元
DstPort：コピー先

SrcPort で示されるアドレスから１バイトを読み取り DstPort で示されるアドレスに書込みます。

[3] Short および Integer のオーバーロードが用意されています。

USBM_PortCopy16() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_PortCopy16(TW_HANDLE hDev, DWORD SrcPort, DWORD DstPort)
VB	Function USBM_PortCopy16(ByVal hDev As System.IntPtr, ByVal SrcPort As Integer, ByVal DstPort As Integer) As Integer
VBA	Function USBM_PortCopy16(ByVal hDev As Long, ByVal SrcPort As Long, ByVal DstPort As Long) As Long

hDev　: デバイスのハンドル
SrcPort : コピー元
DstPort : コピー先

SrcPort で示されるアドレスから 16 ビットのデータを読み取り DstPort で示されるアドレスに書込みます。アドレスはどちらも偶数番地でなければなりません。バイトアクセス専用の空間では 1 バイトずつ 2 回アクセスされます。

USBM_PortBCopy() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_PortBCopy(TW_HANDLE hDev, DWORD SrcPort, DWORD DstPort, long nData,long SrcInc,long DstInc)
VB	Function USBM_PortBCopy(ByVal hDev As System.IntPtr, ByVal SrcPort As Integer, ByVal DstPort As Integer, ByVal nData As Integer, ByVal SrcInc As Integer, ByVal DstInc As Integer) As Integer
VBA	Function USBM_PortBCopy(ByVal hDev As Long, ByVal SrcPort As Long, ByVal DstPort As Long, ByVal nData As Long, ByVal SrcInc As Long, ByVal DstInc As Long) As Long

hDev　: デバイスのハンドル
SrcPort : コピー元
DstPort : コピー先
nData　: バイト数(0～65536)
SrcInc　: コピー元アドレスのインクリメント(1,0,-1)
DstInc　: コピー先アドレスのインクリメント(1,0,-1)

SrcPort から DstPort へ指定バイト数だけコピーを行います。DMA のバーストモードで転送しますので、転送が終わるまでマイコンは割込みも含めて他の処理を行えません。

USBM_PortSetDir() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_PortSetDir(TW_HANDLE hDev, DWORD Port, BYTE Bits)
VB	Function USBM_PortSetDir(ByVal hDev As System.IntPtr, ByVal Port As Integer, ByVal Bits As Byte) As Integer
VBA	Function USBM_PortSetDir(ByVal hDev As Long, ByVal Port As Long, ByVal Bits As Byte) As Long

hDev : デバイスのハンドル
Port : 方向を指定するポート
　　USBM_P1 P10～P17 端子の方向を設定
　　USBM_P2 P20～P27 端子の方向を設定

USBM_P4 P40～P47 端子の方向を設定
USBM_P5 P50～P53 端子の方向を設定
USBM_PA PA0～PA7 端子の方向を設定

Bits ： 方向(1 のビット:出力,0 のビット:入力)

ポートの入力/出力を切替えます。Bits の 0 のビットは入力、1 のビットは出力となります。
注:P1,P2,P5 は入力専用で出力にするとバスのアドレス出力になります。

USBM_PortWrite8() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_PortWrite8(TW_HANDLE hDev, DWORD Port, BYTE Data)
VB	Function USBM_PortWrite8(ByVal hDev As System.IntPtr, ByVal Port As Integer, ByVal Data As Byte) As Integer
VBA	Function USBM_PortWrite8(ByVal hDev As Long, ByVal Port As Long, ByVal Data As Byte) As Long

hDev ： デバイスのハンドル
Port ： 書き込むアドレス
Data ： 書き込むデータ

マイコンの Port で指定されるアドレスに 1 バイトのデータを書き込みます。マイコンの制御レジスタにも書き込みが行えますので、定数以外のアドレスを指定する場合には注意が必要です

□ 16 ビットタイマ操作関数

USBM_TimerStartA() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_TimerStartA(TW_HANDLE hDev, BYTE Bits, long TrigPC, long Repeat, long Cmp)
VB	Function USBM_TimerStartA(ByVal hDev As System.IntPtr, ByVal Bits As Byte, ByVal TrigPC As Integer, ByVal Repeat As Integer, ByVal Cmp As Integer) As Integer
VBA	Function USBM_TimerStartA(ByVal hDev As Long, ByVal Bits As Byte, ByVal TrigPC As Long, ByVal Repeat As Long, ByVal Cmp As Long) As Long

hDev　：デバイスのハンドル
Bits　：スタートするタイマチャンネル
　ビット0(LSB) 1の場合、チャンネル0をスタート
　ビット1　　1の場合、チャンネル1をスタート
　ビット2　　1の場合、チャンネル2をスタート

TrigPC：起動のトリガとなるパルスカウンタチャンネル
　-1　直ちにタイマを起動します
　0～3　指定されたPCx#信号でタイマを起動します

Repeat：起動トリガを繰り返し機能させるかどうかのフラグ(TRUE,FALSE)
Cmp　：パルスカウンタのコンペア値(1～2147483647)

16 ビットタイマの指定チャンネルをスタートします。USBM_TimerStart()との違いはスタートするチャンネル以外に影響を与えないことです。
また、TrigPCにパルスカウンタのチャンネルを指定すると、指定したチャンネルはすぐには起動せず、パルスカウンタへの入力待ちになります。パルスカウンタにパルスが入力されると8us～20us以内にタイマが起動されます。この場合、Repeatが真ならパルスカウンタ入力は繰り返し受付られ、その度にタイマが起動されます。
Cmpは通常1ですが、パルスカウンタへ複数のパルス入力によりタイマを起動する場合は回数を設定してください。
パルスカウンタを起動要因とすると、指定のパルスカウンタがカウントを開始しますので不要になった場合はUSBM_PCStop()関数を呼び出して停止してください。

※この関数はVer.2.0.1以降のファームウェアが必要です。

USBM_TimerStop() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_TimerStop(TW_HANDLE hDev, BYTE Bits, long TrigPC, long Repeat, long Cmp)
VB	Function USBM_TimerStop(ByVal hDev As System.IntPtr, ByVal Bits As Byte, ByVal TrigPC As Integer, ByVal Repeat As Integer, ByVal Cmp As Integer) As Integer
VBA	Function USBM_TimerStop(ByVal hDev As Long, ByVal Bits As Byte, ByVal TrigPC As Long, ByVal Repeat As Long, ByVal Cmp As Long) As Long

hDev　：デバイスのハンドル
Bits　：ストップするタイマチャンネル
　ビット0(LSB) 1の場合、チャンネル0を停止
　ビット1　　1の場合、チャンネル1を停止
　ビット2　　1の場合、チャンネル2を停止

TrigPC : 終了のトリガとなるパルスカウンタチャンネル
　　-1　　直ちにタイマを停止します
　　0～3　指定された PCx#信号でタイマを停止します

Repeat : 終了トリガを繰り返し機能させるかどうかのフラグ(TRUE,FALSE)
Cmp　　: パルスカウンタのコンペア値(1～2147483647)

16 ビットタイマの指定チャンネルを停止します。USBM_TimerStart()との違いは、停止するチャンネル以外に影響を与えないことです。
また、TrigPC にパルスカウンタのチャンネルを指定すると、指定したチャンネルはすぐには停止せず、パルスカウンタへの入力待ちになります。パルスカウンタにパルスが入力されると 8us～20us 以内にタイマが停止されます。この場合、Repeat が真ならパルスカウンタ入力は繰り返し受付られ、その度にタイマが停止されます。
Cmp は通常 1 ですが、パルスカウンタへ複数のパルス入力によりタイマを停止する場合は回数を設定してください。
パルスカウンタを停止要因とすると、指定のパルスカウンタがカウントを開始しますので不要になった場合は USBM_PCStop()関数を呼び出して停止してください。

※この関数は Ver.2.0.1 以降のファームウェアが必要です。

USBM_TimerSetCnt() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_TimerSetCnt(TW_HANDLE hDev, long CH, short Cnt)
VB	Function USBM_TimerSetCnt(ByVal hDev As System.IntPtr, ByVal CH As Integer, ByVal Cnt As Short) As Integer
VBA	Function USBM_TimerSetCnt(ByVal hDev As Long, ByVal CH As Long, ByVal Cnt As Integer) As Long

hDev : デバイスのハンドル
CH　 : タイマチャンネル(0,1,2)
Cnt　: 設定する値

タイマのカウンタ値を設定します。チャンネル間で PWM の位相を調整する場合などに使用できます。

USBM_TimerReadCnt() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_TimerReadCnt(TW_HANDLE hDev, long CH, short *pCnt)
VB	Function USBM_TimerReadCnt(ByVal hDev As System.IntPtr, ByVal CH As Integer, ByRef pCnt As Short) As Integer
VBA	Function USBM_TimerReadCnt(ByVal hDev As Long, ByVal CH As Long, ByRef pCnt As Integer) As Long

hDev　: デバイスのハンドル
CH　　: タイマチャンネル(0,1,2)
pCnt　: [出力]カウント値の格納先

タイマのカウンタ値を読み出します。

USBM_TimerSetClk() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_TimerSetClk(TW_HANDLE hDev, long CH, BYTE Bits)
VB	Function USBM_TimerSetClk(ByVal hDev As System.IntPtr, ByVal CH As Integer, ByVal Bits As Byte) As Integer
VBA	Function USBM_TimerSetClk(ByVal hDev As Long, ByVal CH As Long, ByVal Bits As Byte) As Long

hDev : デバイスのハンドル
CH : タイマチャンネル(0,1,2)
Bits : USBM_TCLK25000 25MHz
USBM_TCLK12500 12.5MHz
USBM_TCLK6250 6250kHz
USBM_TCLK3125 3125kHz
USBM_TCLKA TCLKA からの外部入力
USBM_TCLKB TCLKB からの外部入力

16 ビットタイマに使用するクロックを選択します。TCLKA または TCLKB を選択すると PA0 または PA1 ピンをクロック入力として使用します。USB デバイスの場合、TCLKA,TCLKB は USBM_PortBWrite()または USBM_PortBRead()関数で DMA を使用すると一時的に TEND#信号として出力ピンに設定されてしまいますので同時に使用することはできません。

USBM_TimerSetPulse() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_TimerSetPulse(TW_HANDLE hDev, long CH, WORD LtoH, WORD HtoL)
VB	Function USBM_TimerSetPulse(ByVal hDev As System.IntPtr, ByVal CH As Integer, ByVal LtoH As Short, ByVal HtoL As Short) As Integer
VBA	Function USBM_TimerSetPulse(ByVal hDev As Long, ByVal CH As Long, ByVal LtoH As Integer, ByVal HtoL As Integer) As Long

hDev : デバイスのハンドル
CH : タイマチャンネル(0,1,2)
LtoH : タイマ出力(TIOCA ピン)が 1 になるカウント値を指定(1～65535)
HtoL : タイマ出力(TIOCA ピン)が 0 になるカウント値を指定(1～65535)

タイマ出力ピンが 1 になるタイミングと 0 になるタイミングを指定します。デフォルトの設定ではタイマは PWM モードに設定され TIOCA ピンだけが出力となります。すなわち 16 ビットタイマのカウンタが LtoH の値と等しくなると TIOCA ピンに 1 が出力され、HtoL と等しくなると 0 が出力されます。またデフォルトでは 0 になるタイミングでカウンタがリセットされるように設定されます。つまり、HtoL の値でカウンタ出力の周期を LtoH の値でデューティ比を決定できます。チャンネル間の位相差はタイマスタート前に USBM_TimerSetCnt()を呼び出すことで調整できます。パルスの周期は (HtoL+1)/fclk[sec] 、デューティは (1-(LtoH+1)/(HtoL+1))*100[%]となります。

USBM_TimerSetLevel() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_TimerSetLevel(TW_HANDLE hDev, BYTE Bits)
VB	Function USBM_TimerSetLevel(ByVal hDev As System.IntPtr, ByVal Bits As Byte) As Integer
VBA	Function USBM_TimerSetLevel(ByVal hDev As Long, ByVal Bits As Byte) As Long

hDev : デバイスのハンドル
Bits : 各端子に対応するビット
0 ビット(LSB) TIOCA0
1 ビット TIOCB0
2 ビット TIOCA1
3 ビット TIOCB1
4 ビット TIOCA2
5 ビット TIOCB2

タイマ出力に設定されているピンのレベルを変更します。1 にセットすると対応する出力ピンが 1 出力になります。タイマ出力に設定されていないピンは影響を受けません。デフォルトでは有効にしたチャンネルの TIOCA ピンだけが出力になります。

USBM_TimerEnable() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_TimerEnable(TW_HANDLE hDev, BYTE Bits, long MDF)
VB	Function USBM_TimerEnable(ByVal hDev As System.IntPtr, ByVal Bits As Byte, ByVal MDF As Integer) As Integer
VBA	Function USBM_TimerEnable(ByVal hDev As Long, ByVal Bits As Byte, ByVal MDF As Long) As Long

hDev : デバイスのハンドル
Bits : PWM 出力を行うチャンネルを示すビット
0 ビット(LSB) タイマ 0
1 ビット タイマ 1
2 ビット タイマ 2

MDF : 0 以外を指定するとチャンネル 2 を位相計数モードに設定します。

指定チャンネルを PWM モードに設定します。PWM モードに設定したチャンネルの TIOCA ピンはタイマ出力となります。MDF を真にするとチャンネル 2 を位相計数モードに設定します(位相計数モードの詳細は H8 マイコンのドキュメントを参照してください。)。位相計数モードになると自動的に TCLKA,TCLKB ピンがクロック入力となります。

USBM_TimerSetCmp() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_TimerSetCmp(TW_HANDLE hDev, short CmpA, short CmpB)
VB	Function USBM_TimerSetCmp(ByVal hDev As System.IntPtr, ByVal CmpA As Short, ByVal CmpB As Short) As Integer
VBA	Function USBM_TimerSetCmp(ByVal hDev As Long, ByVal CmpA As Integer, ByVal CmpB As Integer) As Long

hDev : デバイスのハンドル
CmpA : コンペアレジスタ A

CmpB　: コンペアレジスタ B

チャンネル 2 を位相計数モードで使用する場合のコンペア値を設定します。コンペア値とチャンネル 2 のカウンタ値が一致したときの動作は USBM_TimerSetCmpOut()関数で設定します。

USBM_TimerSetCmpOut() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_TimerSetCmpOut(TW_HANDLE hDev, long CmpReg, DWORD Port, BYTE Data)
VB	Function USBM_TimerSetCmpOut(ByVal hDev As System.IntPtr, ByVal CmpReg As Integer, ByVal Port As Integer, ByVal Data As Byte) As Integer
VBA	Function USBM_TimerSetCmpOut(ByVal hDev As Long, ByVal CmpReg As Long, ByVal Port As Long, ByVal Data As Byte) As Long

hDev　: デバイスのハンドル
CmpReg : 設定するコンペアレジスタ
　　0 コンペアレジスタ A
　　1 コンペアレジスタ B
Port　: 値を書き込むアドレス
Data　: 書き込むデータ

タイマチャンネル 2 のコンペアアウトを設定します。コンペアアウトを設定したコンペアレジスタの値とタイマチャンネル 2 のカウント値が一致すると Port で指定されたアドレスに Data がライトされます。出力が行われないようにするには Port を 0 にして呼び出してください。タイマチャンネル 2 が PWM モードでも位相計数モードでも機能します。PWM モードでは 1 になるタイミングがコンペアレジスタ A とのマッチングで、0 になるタイミング(同時にカウンタが 0 にクリアされるタイミング)がコンペアレジスタ B とのマッチングです。

USBM_TimerSetCmpClr() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_TimerSetCmpClr(TW_HANDLE hDev, BYTE Bits)
VB	Function USBM_TimerSetCmpClr(ByVal hDev As System.IntPtr, ByVal Bits As Byte) As Integer
VBA	Function USBM_TimerSetCmpClr(ByVal hDev As Long, ByVal Bits As Byte) As Long

hDev : デバイスのハンドル
Bits : コンペアマッチでカウンタをクリアするコンペアレジスタを指定
　　0 ビット(LSB)が 1 のときコンペアレジスタ A とのマッチングでカウンタがクリアされます
　　1 ビットが 1 のときコンペアレジスタ B とのマッチングでカウンタがクリアされます

指定されたコンペアレジスタとコンペアマッチが発生したときタイマチャンネル 2 のカウンタがクリアされます。

USBM_TimerSetCapture() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_TimerSetCapture(TW_HANDLE hDev, long CH, BYTE EdgeA, BYTE EdgeB)
VB	Function USBM_TimerSetCapture(ByVal hDev As System.IntPtr, ByVal CH As Integer, ByVal EdgeA As Byte, ByVal EdgeB As Byte) As Integer
VBA	Function USBM_TimerSetCapture(ByVal hDev As Long, ByVal CH As Long, ByVal EdgeA As Byte, ByVal EdgeB As Byte) As Long

hDev ： デバイスのハンドル
CH ： インプットキャプチャ設定を行うタイマチャンネル(0,1,2)
EdgeA ： TIOCA 端子でキャプチャする信号エッジ
0 キャプチャは行わない
USBM_CAPT_RISE 立ち上がりをキャプチャ
USBM_CAPT_FALL 立ち下がりをキャプチャ
USBM_CAPT_BOTH 立ち上がり、立ち下がり両方でキャプチャ

EdgeB ： TIOCB 端子でキャプチャする信号エッジ(指定できる値は EdgeA と同じ)

16 ビットタイマの指定チャンネルをインプットキャプチャモードに設定します。インプットキャプチャに設定すると該当チャンネルの TIOCA,TIOCB 端子は入力端子となり、引数で指定された信号のエッジを検出すると、そのときのカウンタの値がマイコンの GRA,GRB レジスタに転送されます。
この機能を利用することで、TIOCA 端子で信号を検出してから、TIOCB で信号が検出するまでの時間差をタイマクロックのカウント値で測定することができます。クロックの設定には USBM_TimerSetClk()関数を使用してください。カウンタは TIOCB 端子が信号を検出したときにクリアされます。
動作中は信号が入力される度に、何度でもキャプチャされます。

USBM_TimerReadCaptureCnt() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_TimerReadCaptureCnt(TW_HANDLE hDev, long CH, long *pCntA, long *pCntB)
VB	Function USBM_TimerReadCaptureCnt(ByVal hDev As System.IntPtr, ByVal CH As Integer, ByRef pCntA As Integer, ByRef pCntB As Integer) As Integer
VBA	Function USBM_TimerReadCaptureCnt(ByVal hDev As Long, ByVal CH As Long, ByRef pCntA As Long, ByRef pCntB As Long) As Long

hDev ： デバイスのハンドル
CH ： タイマチャンネル(0,1,2)
pCntA ： [出力]GRA レジスタ値の格納先へのポインタ
pCntB ： [出力]GRB レジスタ値の格納先へのポインタ

インプットキャプチャした結果を読み出します。

USBM_TimerSetCaptureCnt() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_TimerSetCaptureCnt(TW_HANDLE hDev, long CH, long CntA, long CntB)
VB	Function USBM_TimerSetCaptureCnt(ByVal hDev As System.IntPtr, ByVal CH As Integer, ByVal CntA As Integer, ByVal CntB As Integer) As Integer
VBA	Function USBM_TimerSetCaptureCnt(ByVal hDev As Long, ByVal CH As Long, ByVal CntA As Long, ByVal CntB As Long) As Long

hDev : デバイスのハンドル
CH : タイマチャンネル(0,1,2)
CntA : GRA レジスタにセットする値(0～65535)
CntB : GRB レジスタにセットする値(0～65535)

GRA、GRB レジスタに値をセットします。主にインプットキャプチャを行う前にレジスタをクリアするのに使用します。

USBM_TimerSetSinglePulse() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_TimerSetSinglePulse(TW_HANDLE hDev, long CH, long Setup, long Width, long Pos)
VB	Function USBM_TimerSetSinglePulse(ByVal hDev As System.IntPtr, ByVal CH As Integer, ByVal Setup As Integer, ByVal Width As Integer, ByVal Pos As Integer) As Integer
VBA	Function USBM_TimerSetSinglePulse(ByVal hDev As Long, ByVal CH As Long, ByVal Setup As Long, ByVal Width As Long, ByVal Pos As Long) As Long

hDev : デバイスのハンドル
CH : 使用する 16 ビットタイマチャンネル(0,1)
Setup : タイマ起動からパルス出力までの時間をクロック数で指定(1～65534)
Width : パルス幅をクロック数で指定(1～65534)
Pos : TRUE の場合、正のパルス、FALSE の場合、負のパルスが出力されます

16 ビットタイマを利用して単発のパルスを発生させます。Setup と Width を足した値が 65535 を超えてはいけません。

※この関数は Ver.2.0.1 以降のファームウェアが必要です。

USBM_TimerStart() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_TimerStart(TW_HANDLE hDev, BYTE Bits)
VB	Function USBM_TimerStart(ByVal hDev As System.IntPtr, ByVal Bits As Byte) As Integer
VBA	Function USBM_TimerStart(ByVal hDev As Long, ByVal Bits As Byte) As Long

hDev : デバイスのハンドル
Bits : 0 ビット(LSB) タイマ 0
1 ビット タイマ 1
2 ビット タイマ 2

タイマのスタート/ストップを制御します。1 にセットしたビットに対応するタイマがスタートし、0 にセットしたビットに対応するタイマはストップします。

□ AD コンバータ操作関数

USBM_ADRead() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_ADRead(TW_HANDLE hDev, WORD *pData, long CH, long Scan)
VB	Function USBM_ADRead(ByVal hDev As System.IntPtr, ByVal pData() As Short, ByVal CH As Integer, ByVal Scan As Integer) As Integer
VBA	Function USBM_ADRead(ByVal hDev As Long, ByRef pData() As Integer, ByVal CH As Long, ByVal Scan As Long) As Long

hDev ： デバイスのハンドル
pData ： [出力]読み出したデータの格納先へのポインタ
CH ： AD コンバータのチャンネル(0,1,2,3)
Scan ： 複数チャンネルをスキャンするかどうかのフラグ(TRUE,FALSE)

Scan が FALSE のとき指定されたチャンネルの AD 変換結果を pData 位置に格納します。Scan が TRUE のときは 0 から指定されたチャンネルまでを AD 変換し pData の位置から格納します。例えば Scan=TRUE で CH=2 の場合、0～2 チャンネルの 3 チャンネル分の変換結果を pData 位置から 3 ワード分格納します。pData 領域の大きさはチェックされませんのでチャンネル数に合わせて呼び出し側で確保してください。

USBM_ADBRead() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_ADBRead(TW_HANDLE hDev, WORD *pData, DWORD nData, long CH, DWORD *pnRead)
VB	Function USBM_ADBRead(ByVal hDev As System.IntPtr, ByVal pData() As Short, ByVal nData As Integer, ByVal CH As Integer, ByRef pnRead As Integer) As Integer
VBA	Function USBM_ADBRead(ByVal hDev As Long, ByRef pData() As Integer, ByVal nData As Long, ByVal CH As Long, ByRef pnRead As Long) As Long

hDev ： デバイスのハンドル
pData ： [出力]読み出したデータの格納先へのポインタ
nData ： 読み出すデータ数(バイト単位ではなくワードデータの数､1～4294967295)
CH ： AD コンバータのチャンネル(0,1,2,3)
pnRead ： 実際に読み出したデータ数の格納先(バイト単位ではなくワードデータの数)

指定チャンネルの AD コンバータで定期的に AD 変換し、そのデータを読み出します。変換タイミングはマイコン内蔵の 8 ビットタイマで作り出す方法と、外部トリガを入力する方法があります。デフォルトでは外部トリガの入力で変換が開始されますので、タイマを利用する場合には、予め USBM_ADSetCycle()関数でクロックとコンペア値を設定しておいてください。
関数自体は USBM_SetTimeouts()で指定された時間でタイムアウトしますが､デバイスは指定されたデータ数を読み出すまで処理を続けます｡中止させるためには USBM_Abort()関数を使用してください。

USBM_ADStart() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_ADStart(TW_HANDLE hDev, long nCnv, long CH, long Scan, long lByte,long Trig)
VB	Function USBM_ADStart(ByVal hDev As System.IntPtr, ByVal nCnv As Integer, ByVal CH As Integer, ByVal Scan As Integer, ByVal lByte As Integer, ByVal TrigPC As Integer) As Integer
VBA	Function USBM_ADStart(ByVal hDev As Long, ByVal nCnv As Long, ByVal CH As Long, ByVal Scan As Long, ByVal lByte As Long, ByVal Trig As Long) As Long

hDev　：デバイスのハンドル
nCnv　：変換回数。複数チャンネルをスキャンする場合は、それぞれ nCnv 回変換されます。
　　-1 の場合中止されるまで繰り返します。(-1,1～2147483647)

CH　　：AD コンバータのチャンネル(0,1,2,3)
Scan　：複数チャンネルをスキャンするかどうかのフラグ(TRUE,FALSE)
lByte　：バイト読み出しするかどうかのフラグ(TRUE,FALSE)
Trig　：内蔵タイマの起動を ADTRG#で行うかどうかのフラグ(TRUE,FALSE)

AD コンバータを起動し、断続的に変換を行います。変換結果は USBM_Read()関数で取り出してください。変換タイミングはマイコン内蔵の 8 ビットタイマで作り出す方法と、ADTRG#端子に信号を入力する方法があります。ADTRG#を選択した場合、ADC はトリガ入力待ちとなり、1 回のトリガにつき 1 回の変換を行います。タイマを選択した場合、タイマを起動し予め設定したサイクル毎に変換を行います。デフォルトでは外部トリガの入力で 変換が開始されますので、タイマを利用する場合には、予め USBM_ADSetCycle()関数でクロックとコンペア値を設定します。

Scan が FALSE のときは指定チャンネルの変換を指定された回数行います。
Scan が TRUE のときは 1 回のトリガにつき、0 から指定されたチャンネルまでを 1 回ずつ AD 変換します。例えば Scan=TRUE で CH=2 の場合、0～2 チャンネルの 3 チャンネル分の変換をトリガ毎に行いホストに転送します。

lByte が TRUE の場合は AD 変換結果の上位 8 ビットのみをホストに転送します。16 ビット転送を行う場合よりも、変換レートを上げることが可能となります。詳しくは「AD コンバータ」を参照してください。

変換タイミングをマイコンの内蔵タイマで作る場合でも、Trig を TRUE に設定すると外部トリガに変換を同期させることができます。この場合、USBM_ADSetCycle()で外部トリガを指定した場合と違い、ADTRG#入力で内蔵タイマを起動し、AD のトリガ信号自体はタイマから供給されますので、ADTRG#への入力は 1 度でよくなります。

ADBRead()関数との違いは、ADBRead()関数は指定したデータ数の変換が全て終了するまでデータが取り出せないのに対して、終了したデータだけを逐次取り出せることです。また、複数チャンネルの連続変換も可能です。

デバイスはこの関数を呼び出した後は、指定の変換回数が終了するまで、USBM_Abort()以外の関数を受付ませんのでご注意ください。また、変換終了後はバッファ内のデータを確実に取り出してください。データが残っている場合、後の関数が誤動作します。

※この関数は Ver.2.0.1 以降のファームウェアが必要です。

USBM_ADSetCycle() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_ADSetCycle(TW_HANDLE hDev, BYTE Cmp, long CLK)
VB	Function USBM_ADSetCycle(ByVal hDev As System.IntPtr, ByVal Cmp As Byte, ByVal CLK As Integer) As Integer
VBA	Function USBM_ADSetCycle(ByVal hDev As Long, ByVal Cmp As Byte, ByVal CLK As Long) As Long

hDev ： デバイスのハンドル
Cmp ： コンペアレジスタの値(1～255)
CLK ： クロックの選択
USBM_TCLK3125 3125kHz
USBM_TCLK390 390.625kHz
USBM_TCLK3 約 3052Hz

USBM_ADBRead()、USBM_ADStart()で連続して AD 変換を行う場合の周期を設定します。CLK でクロックの周波数を選択し、Cmp の値でそのクロックを何回カウントしたときに変換を開始するかを決定します。CLK を 0 にして呼び出すとタイマではなく外部トリガ入力で変換を開始する設定となります。

USBM_ADCopy() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_ADCopy(TW_HANDLE hDev, DWORD DstPort, long nCnv, long CH, long CKS, long DMA_CH, long DstInc)
VB	Function USBM_ADCopy(ByVal hDev As System.IntPtr, ByVal DstPort As Integer, ByVal nCnv As Integer, ByVal CH As Integer, ByVal CKS As Integer, ByVal DMA_CH As Integer, ByVal DstInc As Integer) As Integer
VBA	Function USBM_ADCopy(ByVal hDev As Long, ByVal DstPort As Long, ByVal nCnv As Long, ByVal CH As Long, ByVal CKS As Long, ByVal DMA_CH As Long, ByVal DstInc As Long) As Long

hDev ： デバイスのハンドル
DstPort ： 変換結果を格納するマイコン内のアドレス
nCnv ： 変換回数。複数チャンネルをスキャンする場合は、それぞれ nCnv 回変換されます。(1～65536)
CH ： 終了チャンネル(0,1,2,3)
CKS ： 変換ステート(TRUE,FALSE)
DMA_CH ： 使用する DMA チャンネル(0,1)
DstInc ： 転送先アドレスをインクリメントするかどうかのフラグ(TRUE,FALSE)

AD 変換を連続で行い、変換結果をマイコン内のメモリに転送します。呼び出すと、USBM_ADSetCycle()で指定した変換周期はクリアされ、起動要因は外部トリガに固定されます。設定終了後、変換が終了したかどうかにかかわらず、関数はリターンします。その後、デバイスが ADTRG#の立ち下がりを検出すると、指定回数の変換を自動で行います。このとき変換は常に最大レートで行われます。また、変換は必ず 0 チャンネルから開始され、CH で指定したチャンネルまで変換されます。例えば、CH=2 のとき 0～2 までの 3 チャンネルの変換を nCnv 回行います。
指定回数の変換が終了したかどうかを調べるには USBM_ADReadCopyStatus()を使用してください。中断して終了する場合、及び変換終了後は USBM_ADStopCopy()を呼び出してください。

CKS が FALSE のときは最初の 1 サイクルの変換時間が最大 5.36us で、以降、連続変換中は 5.12us/チャンネルです。CKS を TRUE とすることで精度を犠牲にして、変換レートを上げることができます。この場合、最初の 1 サイクルの変換時間が最大で 2.8us で、以降、連続変換中は 2.64us/チャンネルとなります。
変換精度については製品マニュアルを参照してください。

DstPort と nCnv の設定は慎重に行ってください。変換結果を格納する領域が、ユーザーメモリの範囲を超えてもチェックは行われません。使用できる範囲を超えて DMA 転送が行われると、マイコンの内部メモリの情報を破壊して正常に動作しなくなります。

DMA チャンネルを 1 つ利用しますので、該当チャンネルの DMA 転送は行えなくなります DMA0 を使用する場合には USBM_PortBWrite()の DMA 転送と USBM_PortBCopy()が使用できなくなり、DMA1 を使用した場合には USBM_PortBRead()の DMA 転送が使えません。また、DA コンバータへの自動転送も同じ番号のチャンネルが使用できなくなります。

※この関数は Ver.2.0.1 以降のファームウェアが必要です。

USBM_ADReadCopyBuffer() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_ADReadCopyBuffer(TW_HANDLE hDev, DWORD Port, WORD *pData, long nData, long Inc, BYTE Dma)
VB	Function USBM_ADReadCopyBuffer(ByVal hDev As System.IntPtr, ByVal Port As Integer, ByVal pData() As Short, ByVal nData As Integer, ByVal Inc As Integer, ByVal Dma As Byte) As Integer
VBA	Function USBM_ADReadCopyBuffer(ByVal hDev As Long, ByVal Port As Long, ByRef ppData() As Integer, ByVal nData As Long, ByVal Inc As Long, ByVal Dma As Byte) As Long

hDev : デバイスのハンドル
Port : 読み出すアドレス
pData : [出力]読み出したデータの格納先へのポインタ
nData : データの数(0～32768、ワード単位)
Inc : マイコンのアドレスをインクリメントするかを示すフラグ(TRUE,FALSE)
Dma : DMA を使用するかどうかのフラグ(TRUE,FALSE)

USBM_ADCopy()関数で変換されたデータをデバイス上のバッファから読み出すのに使用します。USBM_PortBRead()関数でも読み出せますが、マイコン上のデータは Windows パソコンとバイトオーダーが違うため変換が必要になります。この関数では読み出したデータを、パソコンと同じバイトオーダー(リトルエンディアン)に並び替えて返します。
Inc を FALSE とするとマイコン側のアドレスがインクリメントされませんので FIFO のようなデバイスから読み出しを行えます。
Dma を TRUE とするとマイコン内蔵の DMA を使用してデータを転送します。DMA を使用すると転送速度が向上します。ただし、USB デバイスの場合、転送終了時に PA1/TCLKB 端子に TEND#信号を出力するので注意が必要です。PA1/TCLKB 端子に出力が出てはいけない場合には Dma=FALSE として呼び出してください。

USBM_ADStopCopy() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_ADStopCopy(TW_HANDLE hDev, long DMA_CH)
VB	Function USBM_ADStopCopy(ByVal hDev As System.IntPtr, ByVal DMA_CH As Integer) As Integer
VBA	Function USBM_ADStopCopy(ByVal hDev As Long, ByVal DMA_CH As Long) As Long

hDev : デバイスのハンドル
DMA_CH : USBM_ADCopy()で指定した DMA チャンネル(0,1)

USBM_ADCopy()の終了処理をします。中断して終了する場合にもこの関数を呼び出してください。

※この関数は Ver.2.0.1 以降のファームウェアが必要です。

USBM_ADReadCopyStatus() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_ADReadCopyStatus(TW_HANDLE hDev, long *pnCnv, long DMA_CH)
VB	Function USBM_ADReadCopyStatus(ByVal hDev As System.IntPtr,ByRef pnCnv As Integer, ByVal DMA_CH As Integer) As Integer
VBA	Function USBM_ADReadCopyStatus(ByVal hDev As Long, ByRef pnCnv As Long, ByVal DMA_CH As Long) As Long

hDev　: デバイスのハンドル
pnCnv　: [出力]残りの変換数の格納先
DMA_CH : USBM_ADCopy()で指定したDMAチャンネル(0,1)

USBM_ADCopy()の残りの変換回数を調べます。

※この関数はVer.2.0.1以降のファームウェアが必要です。

□ DA コンバータ操作関数

USBM_DASetParm() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_DASetParm(TW_HANDLE hDev,long CH,DWORD SrcPort,long nData,long lLoop)
VB	Function USBM_DASetParm(ByVal hDev As System.IntPtr, ByVal CH As Integer, ByVal SrcPort As Integer, ByVal nData As Integer, ByVal lLoop As Integer) As Integer
VBA	Function USBM_DASetParm(ByVal hDev As Long, ByVal CH As Long, ByVal SrcPort As Long, ByVal nData As Long, ByVal lLoop As Long) As Long

hDev　　：デバイスのハンドル
CH　　：DA チャンネル(0,1)
SrcPort ：転送元アドレス
nData　：データ数
　　繰り返さない場合、1～65536
　　繰り返し転送する場合、1～255

lLoop　：繰り返し転送するかのフラグ(TRUE,FALSE)

DA コンバータへの DMA 転送設定を行います。転送(変換)の周期は USBM_DASetCycle()で指定してください。転送元アドレスは一般にユーザーメモリ内のアドレスを指定し、予めデジタルデータを設定しておく必要があります。lLoop が真の場合、nData の変換終了後に、再びデータの先頭から変換を繰り返します。
DA0 の転送には DMA0 チャンネル、DA1 の転送には DMA1 チャンネルを利用しますので、該当チャンネルの DMA 転送は行えなくなります。具体的には DA0 の DMA 転送時には USBM_PortBWrite()の DMA 転送と USBM_PortBCopy()が使用できなくなり、DA1 の DMA 転送時には USBM_PortBRead()の DMA 転送が使えません。また、DA チャンネルと同じチャンネルの 16 ビットタイマも使用できなくなります。

※この関数は Ver.2.0.1 以降のファームウェアが必要です。

USBM_DAStart() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_DAStart(TW_HANDLE hDev, BYTE Bits)
VB	Function USBM_DAStart(ByVal hDev As System.IntPtr, ByVal Bits As Byte) As Integer
VBA	Function USBM_DAStart(ByVal hDev As Long, ByVal Bits As Byte) As Long

hDev ：デバイスのハンドル
Bits ：スタートするチャンネル
　　ビット 0(LSB) 1 の場合 DA0 への転送(変換)を開始します。
　　ビット 1　　1 の場合 DA1 への転送(変換)を開始します。

DA コンバータへのデータ転送を開始します。転送を開始する前に毎回 USBM_DASetParm()を呼び出してください。

※この関数は Ver.2.0.1 以降のファームウェアが必要です。

USBM_DAStop() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS _stdcall USBM_DAStop(TW_HANDLE hDev, BYTE Bits)
VB	Function USBM_DAStop(ByVal hDev As System.IntPtr, ByVal Bits As Byte) As Integer
VBA	Function USBM_DAStop(ByVal hDev As Long, ByVal Bits As Byte) As Long

hDev : デバイスのハンドル
Bits : 終了するチャンネル
　　ビット0(LSB) 1の場合 DA0 への転送(変換)を終了します。
　　ビット1　　1の場合 DA1 への転送(変換)を終了します。

DA コンバータへのデータ転送を終了します。

※この関数は Ver.2.0.1 以降のファームウェアが必要です。

USBM_DAReadStatus() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_DAReadStatus(TW_HANDLE hDev, long CH, long *pnData)
VB	Function USBM_DAReadStatus(ByVal hDev As System.IntPtr, ByVal CH As Integer, ByRef pnData As Integer) As Integer
VBA	Function USBM_DAReadStatus(ByVal hDev As Long, ByVal CH As Long, ByRef pnData As Long) As Long

hDev　　: デバイスのハンドル
CH　　: DA のチャンネル(0,1)
pnData　: [出力]残りの変換データ数の格納先

DA コンバータの残り変換データ数を調べます。繰り返し変換中は常に 0 以外の値が返ります。

※この関数は Ver.2.0.1 以降のファームウェアが必要です。

USBM_DASetCycle() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_DASetCycle(TW_HANDLE hDev,long CH,long Cmp,long CLK)
VB	Function USBM_DASetCycle(ByVal hDev As System.IntPtr, ByVal CH As Integer, ByVal Cmp As Integer, ByVal CLK As Integer) As Integer
VBA	Function USBM_DASetCycle(ByVal hDev As Long, ByVal CH As Long, ByVal Cmp As Long, ByVal CLK As Long) As Long

hDev : デバイスのハンドル
CH　: 周期を設定する DA チャンネル
Cmp　: クロックをカウントする回数
CLK　: クロックを選択
　　USBM_TCLK25000 25MHz
　　USBM_TCLK12500 12.5MHz
　　USBM_TCLK6250　6250kHz
　　USBM_TCLK3125　3125kHz

DA コンバータの変換周期を設定します。変換周期は下のようになります。

Tc = (Cmp + 1) / fclk [sec] (fclk:CLK で選択した周波数)

DA0 のタイミング生成には 0 チャンネルの、DA1 のタイミングには 1 チャンネルの 16 ビットタイマを利用しますので該当チャンネルの 16 ビットタイマは他の用途には使用できなくなります。

※この関数は Ver.2.0.1 以降のファームウェアが必要です。

□ SCI（シリアルインタフェース）操作関数

USBM_SCIRead() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_SCIRead(TW_HANDLE hDev, long CH, void *pData, long nData, long *pnRead)
VB	Function USBM_SCIRead(ByVal hDev As System.IntPtr, ByVal CH As Integer, ByVal pData() As Byte, ByVal nData As Integer, ByRef pnRead As Integer) As Integer
VBA	Function USBM_SCIRead(ByVal hDev As Long, ByVal CH As Long, ByRef pData() As Byte, ByVal nData As Long, ByRef pnRead As Long) As Long

hDev　: デバイスのハンドル
CH　　: チャンネル(0,1)
pData : [出力]格納先へのポインタ
nData : データのバイト数(0～255)
pnRead : 実際に読み出したバイト数の格納先

マイコンボード上のシリアルポートからデータを読み出します。関数自体はUSBM_SetTimeouts()で指定された時間でタイムアウトしますが、デバイスは指定されたデータ数を読み出すまで処理を続けます。中止させるためにはUSBM_Abort()関数を使用してください。

USBM_SCIWrite() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_SCIWrite(TW_HANDLE hDev, long CH, void *pData, long nData)
VB	Function USBM_SCIWrite(ByVal hDev As System.IntPtr, ByVal CH As Integer, ByVal pData() As Byte, ByVal nData As Integer) As Integer
VBA	Function USBM_SCIWrite(ByVal hDev As Long, ByVal CH As Long, ByRef pData() As Byte, ByVal nData As Long) As Long

hDev　: デバイスのハンドル
CH　　: チャンネル(0,1)
pData : [入力]データへのポインタ
nData : データのバイト数(0～255)

マイコンボード上のシリアルポートからデータを出力します。

USBM_SCISetMode() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_SCISetMode(TW_HANDLE hDev, long CH, long Mode, long Baud)
VB	Function USBM_SCISetMode(ByVal hDev As System.IntPtr, ByVal CH As Integer, ByVal Mode As Integer, ByVal Baud As Integer) As Integer
VBA	Function USBM_SCISetMode(ByVal hDev As Long, ByVal CH As Long, ByVal Mode As Long, ByVal Baud As Long) As Long

hDev ： デバイスのハンドル
CH ： SCI チャンネル(0,1)
Mode ： キャラクタのビット数,パリティ,ストップビットの設定の OR で結合します。
USBM_SCI_DATA8 ： 8 ビットデータ
USBM_SCI_DATA7 ： 7 ビットデータ
USBM_SCI_NOPARITY ： パリティなし
USBM_SCI_EVEN ： 偶数パリティ
USBM_SCI_ODD ： 奇数パリティ
USBM_SCI_STOP1 ： 1 ストップビット
USBM_SCI_STOP2 ： 2 ストップビット

Baud ： ボーレート
USBM_SCI_BAUD300 ： 300bps
USBM_SCI_BAUD600 ： 600bps
USBM_SCI_BAUD1200 ： 1200bps
USBM_SCI_BAUD2400 ： 2400bps
USBM_SCI_BAUD4800 ： 4800bps
USBM_SCI_BAUD9600 ： 9600bps
USBM_SCI_BAUD14400 ： 14400bps
USBM_SCI_BAUD19200 ： 19200bps
USBM_SCI_BAUD38400 ： 38400bps

SCI チャンネルの設定を行います。使用できるのは調歩同期のみです。
ユーザーファーム開発時に SCI1 をデバッグ通信や標準入出力に使用している場合、1 チャンネルに対してこの関数を呼び出さないでください。

USBM_SCISetDelimiter() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_SCISetDelimiter(TW_HANDLE hDev, long CH, char *pDelimiter, long nDelimiter)
VB	Function USBM_SCISetDelimiter(ByVal hDev As System.IntPtr, ByVal CH As Integer, ByVal pData() As Byte, ByVal nDelimiter As Integer) As Integer
VBA	Function USBM_SCISetDelimiter(ByVal hDev As Long, ByVal CH As Long, ByRef ppData() As Byte, ByVal nDelimiter As Long) As Long

hDev ： デバイスのハンドル
CH ： SCI チャンネル(0,1)
pDelimiter ： [入力]デリミタ文字へのポインタ
nDelimiter ： デリミタ文字の数(0～2)

SCI チャンネルのデリミタ文字を指定します。デリミタ文字(1 文字または 2 文字)が現れると、USBM_SCIRead()関数はシリアルポートからの読み取りを中止し、読み取り指定バイトまで 0 を埋めてデータを返します。

※この関数は Ver.2.0.1 以降のファームウェアが必要です。

USBM_SCIReadStatus() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_SCIReadStatus(TW_HANDLE hDev, long CH, BYTE *pStatus, long *pnReceive)
VB	Function USBM_SCIReadStatus(ByVal hDev As System.IntPtr, ByVal CH As Integer, ByRef pStatus As Byte, ByRef pnReceive As Integer) As Integer
VBA	Function USBM_SCIReadStatus(ByVal hDev As Long, ByVal CH As Long, ByRef pStatus As Byte, ByRef pnReceive As Long) As Long

hDev : デバイスのハンドル
CH : チャンネル(0,1)
pStatus : [出力]ステータスの格納先へのポインタ
0(LSB)ビット～2 ビット 0 です
3 ビット パリティエラーが起こった場合に 1 になります
4 ビット フレーミングエラーが起こった場合に 1 になります
5 ビット オーバーランエラーが起こった場合に 1 になります
6 ビット～7 ビット(MSB) 0 です
pnReceive : [出力]受信データ数の格納先

SCI のステータス情報と受信バッファに格納されているデータ数を読み出します。

□ パルスカウンタ操作関数

USBM_PCSetCnt() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_PCSetCnt(TW_HANDLE hDev, long CH, long Cnt)
VB	Function USBM_PCSetCnt(ByVal hDev As System.IntPtr, ByVal CH As Integer, ByVal Cnt As Integer) As Integer
VBA	Function USBM_PCSetCnt(ByVal hDev As Long, ByVal CH As Long, ByVal Cnt As Long) As Long

hDev ： デバイスのハンドル
CH　 ： パルスカウンタのチャンネル(0～3)
Cnt　： カウンタの値(-2147483648～2147483647)

パルスカウンタに値を設定します。

USBM_PCSetCmp() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_PCSetCmp(TW_HANDLE hDev, long CH, long Cmp)
VB	Function USBM_PCSetCmp(ByVal hDev As System.IntPtr, ByVal CH As Integer, ByVal Cmp As Integer) As Integer
VBA	Function USBM_PCSetCmp(ByVal hDev As Long, ByVal CH As Long, ByVal Cmp As Long) As Long

hDev ： デバイスのハンドル
CH　 ： パルスカウンタのチャンネル(0～3)
Cmp　： コンペア値(-2147483648～2147483647)

パルスカウンタのコンペア値を設定します。設定を行うことでパルスカウンタのカウント値がこの値と一致したときにカウンタ値がリセットされるようにしたり、指定ポートにデータを出力したりするようにできます。

USBM_PCReadCnt() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_PCReadCnt(TW_HANDLE hDev, long CH, long *pCnt)
VB	Function USBM_PCReadCnt(ByVal hDev As System.IntPtr, ByVal CH As Integer, ByVal pCnt() As Integer) As Integer
VBA	Function USBM_PCReadCnt(ByVal hDev As Long, ByVal CH As Long, ByRef pCnt() As Long) As Long

hDev ： デバイスのハンドル
CH　 ： パルスカウンタのチャンネル(0～3)
　　USBM_PC_ALL を指定すると 0～3 チャンネル全てを読み出します
pCnt ： [出力]カウンタの値の格納先

パルスカウンタの値を読み出します。全部のチャンネルを呼び出す場合は 4 チャンネル分の領域を pCnt に指定してください。

USBM_PCSetControl() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_PCSetControl(TW_HANDLE hDev, long CH, BYTE Bits)
VB	Function USBM_PCSetControl(ByVal hDev As System.IntPtr, ByVal CH As Integer, ByVal Bits As Byte) As Integer
VBA	Function USBM_PCSetControl(ByVal hDev As Long, ByVal CH As Long, ByVal Bits As Byte) As Long

hDev : デバイスのハンドル
CH : チャンネル(0,1,2,3)
Bits : 0 ビット(LSB) '1'にすると PC0 の入力でカウンタが 0 にクリアされます
1 ビット '1'にするとコンペアマッチでカウンタが 0 にクリアされます
4,5 ビット '00'にするとパルス入力で無条件にカウントアップします
'01'にすると条件ビットが真のときダウン、偽のときアップ
'10'にすると条件ビットが真のときアップ、偽のときダウン
'11'にすると条件ビットが真のときアップ、偽のときカウントしません
上記以外のビットは予約です。0 にしてください。

パルスカウンタの制御方法を指定します。Bits 下位 2 ビットでリセット条件を 4,5 ビットでカウントの方法を指定します。下位 2 ビットがどちらも 0 の場合、カウンタはクリアされません。条件ビットとは USBM_PCSetCondBit()関数で指定されるビットです。

USBM_PCSetCondBit() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_PCSetCondBit(TW_HANDLE hDev, long CH, DWORD Port, BYTE Mask)
VB	Function USBM_PCSetCondBit(ByVal hDev As System.IntPtr, ByVal CH As Integer, ByVal Port As Integer, ByVal Mask As Byte) As Integer
VBA	Function USBM_PCSetCondBit(ByVal hDev As Long, ByVal CH As Long, ByVal Port As Long, ByVal Mask As Byte) As Long

hDev : デバイスのハンドル
CH : チャンネル(0,1,2,3)
Port : コンディションを読み取るアドレス
Mask : Port アドレスの値とアンドを取るマスク

パルスが入力されたときにカウンタの増減を決める条件ビットを指定します。条件ビットは Port アドレスから読み取られた値と Mask 値の AND(論理積)をとって決定されます。例えば 2 相のロータリーエンコーダー入力の位相関係でカウンタの増減を切替えるような場合に利用できます。

USBM_PCSetCmpOut() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_PCSetCmpOut(TW_HANDLE hDev, long CH, DWORD Port, BYTE Data)
VB	Function USBM_PCSetCmpOut(ByVal hDev As System.IntPtr, ByVal CH As Integer, ByVal Port As Integer, ByVal Data As Byte) As Integer
VBA	Function USBM_PCSetCmpOut(ByVal hDev As Long, ByVal CH As Long, ByVal Port As Long, ByVal Data As Byte) As Long

hDev : デバイスのハンドル
CH : チャンネル(0,1,2,3)
Port : コンペアマッチ時に書き込むアドレス
Data : コンペアマッチ時に書き込むデータ

コンペアマッチ時に行うポート出力を設定します。パルスカウンタの値がコンペア値と一致すると Port アドレスに Data を出力します。ポート操作が必要ない場合は Port=0 としてください。

USBM_PCStartA() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_PCStartA(TW_HANDLE hDev, BYTE Bits)
VB	Function USBM_PCStartA(ByVal hDev As System.IntPtr, ByVal Bits As Byte) As Integer
VBA	Function USBM_PCStartA(ByVal hDev As Long, ByVal Bits As Byte) As Long

hDev : デバイスのハンドル
Bits : スタートするチャンネルをビットで指定
USBM_PC0(0x10) : チャンネル 0 を許可
USBM_PC1(0x20) : チャンネル 1 を許可
USBM_PC2(0x02) : チャンネル 2 を許可
USBM_PC3(0x04) : チャンネル 3 を許可

パルスカウントを許可します。許可するチャンネルが複数ある場合には上記の定数を OR で結合して Bits に指定してください。USBM_PCStart()関数との違いは、スタートするように指定したチャンネル以外に影響を与えないことです。

USBM_PCStop() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_PCStop(TW_HANDLE hDev, BYTE Bits)
VB	Function USBM_PCStop(ByVal hDev As System.IntPtr, ByVal Bits As Byte) As Integer
VBA	Function USBM_PCStop(ByVal hDev As Long, ByVal Bits As Byte) As Long

hDev : デバイスのハンドル
Bits : ストップするチャンネルをビットで指定
USBM_PC0(0x10) : チャンネル 0 のカウントを終了
USBM_PC1(0x20) : チャンネル 1 のカウントを終了
USBM_PC2(0x02) : チャンネル 2 のカウントを終了
USBM_PC3(0x04) : チャンネル 3 のカウントを終了

指定チャンネルのパルスカウントを終了します。終了するチャンネルが複数ある場合には上記の定数を OR で結合して Bits に指定してください。USBM_PCStart()関数との違いは、終了するように指定したチャンネル以外に影響を与えないことです。

USBM_PCStart() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_PCStart(TW_HANDLE hDev, BYTE Bits)
VB	Function USBM_PCStart(ByVal hDev As System.IntPtr, ByVal Bits As Byte) As Integer
VBA	Function USBM_PCStart(ByVal hDev As Long, ByVal Bits As Byte) As Long

hDev : デバイスのハンドル
Bits : スタートするチャンネルをビットで指定
USBM_PC0(0x10) : チャンネル 0 を許可
USBM_PC1(0x20) : チャンネル 1 を許可
USBM_PC2(0x02) : チャンネル 2 を許可
USBM_PC3(0x04) : チャンネル 3 を許可

パルスカウントを許可します。許可するチャンネルが複数ある場合には上記の定数を OR で結合して Bits に指定してください。指定されたチャンネル以外はカウントを終了します。

□ タイマコピー操作関数

USBM_TCPYSetClk() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_TCPYSetClk(TW_HANDLE hDev, long CH, long CLK)
VB	Function USBM_TCPYSetClk(ByVal hDev As System.IntPtr, ByVal CH As Integer, ByVal CLK As Integer) As Integer
VBA	Function USBM_TCPYSetClk(ByVal hDev As Long, ByVal CH As Long, ByVal CLK As Long) As Long

hDev : デバイスのハンドル
CH : クロックを設定するチャンネル(0,1)
CLK : USBM_TCLK3125 3125kHz
USBM_TCLK390 390.625kHz
USBM_TCLK3 3052Hz
USBM_TCLKUP 外部クロックの立ち上がり
USBM_TCLKDOWN 外部クロックの立ち下がり
USBM_TCLKBOTH 外部クロックの両エッジ

タイマコピーに使用するクロックを選択します。外部クロックはチャネル 0 では TCLKA がチャンネル 1 では TCLKB が使用されます。タイマコピーが動作している間は呼び出さないで下さい。

USBM_TCPYSetCmp() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_TCPYSetCmp(TW_HANDLE hDev, long CH, BYTE Cmp)
VB	Function USBM_TCPYSetCmp(ByVal hDev As System.IntPtr, ByVal CH As Integer, ByVal Cmp As Byte) As Integer
VBA	Function USBM_TCPYSetCmp(ByVal hDev As Long, ByVal CH As Long, ByVal Cmp As Byte) As Long

hDev : デバイスのハンドル
CH : 設定するチャンネル(0,1)
Cmp : コンペア値(1～255)

8 ビットタイマがコンペア値と一致すると USBM_TCPYSetParm()で指定した転送元から転送先アドレスに 1 バイトのデータがコピーされます。またタイマのカウント値はリセットされ最初からカウントされますので転送の周期をこの関数で決定します。
転送の周期は T = (Cmp + 1) / CLK となります。ただし CLK は USBM_TCPYSetClk()で指定したクロック周波数です。

USBM_TCPYSetCycle() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_TCPYSetCycle(TW_HANDLE hDev, long CH, BYTE Cmp, long CLK)
VB	Function USBM_TCPYSetCycle(ByVal hDev As System.IntPtr, ByVal CH As Integer, ByVal Cmp As Byte, ByVal CLK As Integer) As Integer
VBA	Function USBM_TCPYSetCycle(ByVal hDev As Long, ByVal CH As Long, ByVal Cmp As Byte, ByVal CLK As Long) As Long

hDev : デバイスのハンドル
CH : クロックを設定するチャンネル(0,1)
Cmp : コンペア値(1～255)

CLK : USBM_TCLK3125 3125kHz
USBM_TCLK390 390.625kHz
USBM_TCLK3 約 3052Hz
USBM_TCLKUP 外部クロックの立ち上がり
USBM_TCLKDOWN 外部クロックの立ち下がり
USBM_TCLKBOTH 外部クロックの両エッジ

タイマコピーのコピーサイクルを設定します。8 ビットタイマのカウンタは CLK で選択されたクロックでカウントされカウンタの値がコンペア値と一致すると USBM_TCPYSetParm()で指定した転送元から転送先アドレスに 1 バイトのデータがコピーされます。またタイマのカウント値はリセットされ最初からカウントされますので転送の周期をこの関数で決定します。
転送の周期は T = (Cmp + 1) / CLK となります。外部クロックはチャネル 0 では TCLKA がチャンネル 1 では TCLKB が使用されます。タイマコピーが動作している間は呼び出さないで下さい。

USBM_TCPYSetParm() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_TCPYSetParm(TW_HANDLE hDev, long CH, DWORD SrcPort, DWORD DstPort, long nCopy, long SrcInc, long DstInc, long lLoop)
VB	Function USBM_TCPYSetParm(ByVal hDev As System.IntPtr, ByVal CH As Integer, ByVal SrcPort As Integer, ByVal DstPort As Integer, ByVal nCopy As Integer, ByVal SrcInc As Integer, ByVal DstInc As Integer, ByVal lLoop As Integer) As Integer
VBA	Function USBM_TCPYSetParm(ByVal hDev As Long, ByVal CH As Long, ByVal SrcPort As Long, ByVal DstPort As Long, ByVal nCopy As Long, ByVal SrcInc As Long, ByVal DstInc As Long, ByVal lLoop As Long) As Long

hDev : デバイスのハンドル
CH : 設定するチャンネル(0,1)
SrcPort : 転送元アドレス
DstPort : 転送先アドレス
nCopy : コピーするバイト数(1～65535)
SrcInc : 転送毎に転送元アドレスを増加するバイト数(-128～127)
DstInc : 転送毎に転送先アドレスを増加するバイト数(-128～127)
lLoop : 真の場合転送終了後に再び最初の状態に戻って転送を再開します(TRUE,FALSE)

タイマコピーのパラメータを設定します。タイマコピーでは 8 ビットタイマがコンペア値と一致する度に転送元に指定したアドレスから転送先に指定したアドレスに 1 バイトずつデータをコピーします。タイマコピーが動作している間は呼び出さないで下さい。

USBM_TCPYSetPatternCtrl() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_TCPYSetPatternCtrl(TW_HANDLE hDev, long CH, DWORD SrcPort, DWORD DstPort, long nCopy, long nSrc, long SrcInc, long Start, BYTE Mask)
VB	Function USBM_TCPYSetPatternCtrl(ByVal hDev As System.IntPtr, ByVal CH As Integer, ByVal SrcPort As Integer, ByVal DstPort As Integer, ByVal nCopy As Integer, ByVal nSrc As Integer, ByVal SrcInc As Integer, ByVal Start As Integer, ByVal Mask As Byte) As Integer
VBA	Function USBM_TCPYSetPatternCtrl(ByVal hDev As Long, ByVal CH As Long, ByVal SrcPort As Long, ByVal DstPort As Long, ByVal nCopy As Long, ByVal nSrc As Long, ByVal SrcInc As Long, ByVal Start As Long, ByVal Mask As Byte) As Long

hDev　　：デバイスのハンドル
CH　　：設定するチャンネル(0,1)
SrcPort：パターンデータの先頭アドレス
DstPort：転送先アドレス
nCopy　：転送回数
　1～65535　指定回数だけコピーを行います
　-1　　停止されるまでコピーを繰り返します

nSrc　　：パターンデータのバイト数(1～65535)
SrcInc　：転送毎に転送元アドレスを増減させるバイト数(-128～127)
Start　：開始時の転送元を示す 0 から始まるインデックス値(0～65534)
Mask　　：転送先に反映するビットを指定

タイマコピーの機能を利用して、任意のポートへのパルスパターン出力を設定します。タイマコピーでは 8 ビットタイマがコンペア値と一致する度に転送元に指定したアドレスから転送先に指定したアドレスに 1 バイトずつデータをコピーされます。
USBM_TCPYSetParm()との違いは転送元に転送回数とは別にパターン数を指定できる点です。転送元アドレスはパターンデータの最後尾を超えると、自動的に先頭に移動します。逆に先頭アドレスより小さくなると、自動的に最後尾に移動します。また、転送先アドレスは常に固定です。
Start は転送開始時の転送元の位置を指定します。転送元として SrcPort+Start から開始されます。
Mask は転送先に反映するビットを指定します。0 のビットは書換えられません。Mask は出力専用ポート(USBM_POUT)に対しては無効です。
この関数は該当チャンネルのタイマコピーが動作している間は呼び出さないで下さい

※この関数は Ver.2.0.1 以降のファームウェアが必要です。

USBM_TCPYSetTrig() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_TCPYSetTrig(TW_HANDLE hDev, long CH, long StartPC, long StopPC,long Repeat)
VB	Function USBM_TCPYSetTrig(ByVal hDev As System.IntPtr, ByVal CH As Integer, ByVal StartPC As Integer, ByVal StopPC As Integer, ByVal flgRepeat As Integer) As Integer
VBA	Function USBM_TCPYSetTrig(ByVal hDev As Long, ByVal CH As Long, ByVal StartPC As Long, ByVal StopPC As Long, ByVal flgRepeat As Long) As Long

hDev　　：デバイスのハンドル
CH　　：設定するタイマコピーのチャンネル(0,1)
StartPC：起動要因となるパルスカウンタチャンネルを指定します。
　　-1　パルスカウンタでは起動しません。
　　0～3 指定された PCx#信号の立ち下がりで起動します。

StopPC　：終了要因となるパルスカウンタチャンネルを指定します。
　　-1　パルスカウンタ入力では終了しません。
　　0～3 指定された PCx#信号の立ち下がりで終了します。

Repeat　：パルスカウンタ入力を繰り返し受け付けるかのフラグ(TRUE,FALSE)

タイマコピーの起動要因、終了要因に PC0#～PC3#のパルスカウンタ入力を指定します。起動要因と終了要因は、別々のチャンネルを指定してください。
この関数を呼び出すと、指定のパルスカウンタがカウントを開始しますので、不要になった場合には USBM_PCStop()関数を呼び出してパルスカウンタを停止してください。

USBM_TCPYReadStatus() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_TCPYReadStatus(TW_HANDLE hDev, long CH, long *pnCopy, long *pSrcPos)
VB	Function USBM_TCPYReadStatus(ByVal hDev As System.IntPtr, ByVal CH As Integer, ByRef pnCopy As Integer, ByRef pSrcPos As Integer) As Integer
VBA	Function USBM_TCPYReadStatus(ByVal hDev As Long, ByVal CH As Long, ByRef pnCopy As Long, ByRef pSrcPos As Long) As Long

hDev　　：デバイスのハンドル
CH　　：設定するタイマコピーのチャンネル(0,1)
pnCopy　：[出力]現在の転送回数の格納先
pSrcPos　：[出力]現在の転送元位置を示す 0 から始まるインデックス

指定チャンネルのタイマコピーの終了した転送回数と転送元の位置を返します。pSrcPos には設定時に指定した転送元アドレスから何バイトの位置を転送中であるかが返ります。
転送中に呼び出すと、pnCopy の値と pSrcPos の値が同期していない場合があります。

USBM_TCPYStart() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_TCPYStart(TW_HANDLE hDev, BYTE Bits)
VB	Function USBM_TCPYStart(ByVal hDev As System.IntPtr, ByVal Bits As Byte) As Integer
VBA	Function USBM_TCPYStart(ByVal hDev As Long, ByVal Bits As Byte) As Long

hDev : デバイスのハンドル
Bits : ビット0(LSB) 1にするとチャンネル0をスタートします。
　　ビット1　　1にするとチャンネル1をスタートします。

Bits で指定されたチャンネルのタイマコピーをスタートします。ストップする場合は対応するビットを0にして呼び出します。Ver.1.0.1 のファームウェアでは呼び出し時に初期化が行われましたが、Ver.2.0.1 以降では中断した位置から再開可能なように、初期化されない仕様に変更されました。新しく転送を開始する場合は USBM_TCPYSetParm()、または USBM_TCPYSetPatternCtrl()関数を呼び出してください。

□ ユーザーファームサポート関数

USBM_ATF_INFO 構造体

言語	宣言

C/C++	typedef struct
	{
	  DWORD FormatVer;
	  DWORD FirmwareVer;
	  char Description[32];
	  char Manufacture[32];
	  DWORD ProgramVer;
	  DWORD AddressTop;
	  DWORD AddressBottom;
	  DWORD CommandAddress;
	  WORD EncryptMode;
	  WORD wRsv;
	  DWORD MainAddress;
	} USBM_ATF_INFO;

VB	Public Structure USBM_ATF_INFO
	  Public FormatVer As Integer
	  Public FirmwareVer As Integer
	  <MarshalAs(UnmanagedType.ByValTStr, SizeConst:=32)> _
	  Public Description As String
	  <MarshalAs(UnmanagedType.ByValTStr, SizeConst:=32)> _
	  Public Manufacture As String
	  Public ProgramVer As Integer
	  Public AddressTop As Integer
	  Public AddressBottom As Integer
	  Public CommandAddress As Integer
	  Public EncryptMode As Short
	  Public wRsv As Short
	  Public MainAddress As Integer
	End Structure

VBA	Public Type USBM_ATF_INFO
	  FormatVer As Long
	  FirmwareVer As Long
	  Description As String * 32
	  Manufacture As String * 32
	  ProgramVer As Long
	  AddressTop As Long
	  AddressBottom As Long
	  CommandAddress As Long
	  EncryptMode As Integer
	  wRsv As Integer
	  MainAddress As Long
	End Type

FormatVer　　: ATF ファイルのフォーマットに関する情報です
FirmwareVer　: ATF Maker の[要求するファームウェアバージョン]に記載した内容です
Description　: ATF Maker の[プログラムの説明]に記載した内容です
Manufacture　: ATF Maker の[作成者]に記載した内容です
ProgramVer　　: ATF Maker の[プログラムのバージョン]に記載した内容です
AddressTop　　: ATF で使用する RAM 領域の先頭アドレスです
AddressBottom　: ATF で使用する RAM 領域の最終アドレス+1 です
CommandAddress : コマンドハンドラ(ATF_Command)のアドレスです
EncryptMode　: 予約。使用されません
wRsv　　: 予約。使用されません
MainAddress　: メイン関数(ATF_Main)のアドレスです

USBM_ATFGetInfo()関数で ATF ファイルの情報を取得するための構造体です。

USBM_ATFGetInfo() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_ATFGetInfo(TW_CSTR FileName, USBM_ATF_INFO *pInfo)
VB	Function USBM_ATFGetInfo(ByVal FileName As String, ByRef pInfo As USBM_ATF_INFO) As Integer
VBA	Function USBM_ATFGetInfo(ByVal FileName As String, ByRef pInfo As USBM_ATF_INFO) As Long

FileName : [入力]ATF ファイルのパス
pInfo　　: [出力]情報を受け取る USBM_ATF_INFO 構造体へのポインタ

ATF ファイルに関する情報を取得します。

USBM_ATFDownload() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_ATFDownload(TW_HANDLE hDev, TW_CSTR FileName, LPCTSTR Reserve)
VB	Function USBM_ATFDownload(ByVal hDev As System.IntPtr, ByVal FileName As String, ByVal Reserve As String) As Integer
VBA	Function USBM_ATFDownload(ByVal hDev As Long, ByVal FileName As String, ByVal Reserve As String) As Long

hDev　　: デバイスのハンドル
FileName : [入力]ATF ファイルのパス
Reserve　: 予約。NULL にしてください

アタッチメントファーム(ATF ファイル)をデバイスの RAM にダウンロードし、コマンドハンドラの登録、メイン関数の登録を行います。

※この関数は Ver.3.0.1 以降のファームウェアが必要です。

USBM_ATFUserCommand() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_ATFUserCommand(TW_HANDLE hDev, WORD Command, DWORD Param1, DWORD Param2, void *pData, long nData)
VB	Function USBM_ATFUserCommand(ByVal hDev As System.IntPtr, ByVal Command As Short, ByVal Param1 As Integer, ByVal Param2 As Integer, ByVal pData As Object, ByVal nData As Integer) As Integer
VBA	Function USBM_ATFUserCommand(ByVal hDev As Long, ByVal Command As Integer, ByVal Param1 As Long, ByVal Param2 As Long, ByRef pData As Variant, ByVal nData As Long) As Long

hDev : デバイスのハンドル
Command : ユーザー定義のコマンド ID
Param1 : ユーザー定義のパラメータ 1
Param2 : ユーザー定義のパラメータ 2
pData : [入力]ユーザー定義の追加パラメータ
nData : ユーザー定義の追加パラメータのバイト数

ユーザー定義のコマンドを呼び出します。Command、Param1、Param2 はコマンドハンドラに引数として渡されます。追加のパラメータが必要な場合には pData に指定します。Windows CE の場合、または、Visual Basic 6.0 以前のバージョンでは、pData に文字列を渡すと UNICODE のデータとなります。

※この関数は Ver.3.0.1 以降のファームウェアが必要です。

USBM_ATFSetCommand() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_ATFSetCommand(TW_HANDLE hDev, DWORD Address)
VB	Function USBM_ATFSetCommand(ByVal hDev As System.IntPtr, ByVal Address As Integer) As Integer
VBA	Function USBM_ATFSetCommand(ByVal hDev As Long, ByVal Address As Long) As Long

hDev : デバイスのハンドル
Address : コマンドハンドラのアドレス

デバイスの指定アドレスをユーザー定義コマンドのハンドラとして登録します。削除するにはアドレスを 0 として呼び出します。登録されたコマンドハンドラは USBM_ATFUserCommand()関数呼び出し時に呼び出されます。通常、コマンドハンドラは USBM_ATFDownload()関数により自動で登録されます。

※この関数は Ver.3.0.1 以降のファームウェアが必要です。

USBM_ATFSetMain() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_ATFSetMain(TW_HANDLE hDev, DWORD Address)
VB	Function USBM_ATFSetMain(ByVal hDev As System.IntPtr, ByVal Address As Integer) As Integer
VBA	Function USBM_ATFSetMain(ByVal hDev As Long, ByVal Address As Long) As Long

hDev : デバイスのハンドル
Address : メイン関数のアドレス

デバイスの指定アドレスをメイン関数として登録します。削除するにはアドレスを 0 として呼び出します。メイン関数はコマンドループの先頭で呼び出される関数です。コマンド以外に常時行う処理がある場合に使用します。通常、メイン関数は USBM_ATFDownload()関数により自動で登録されます。

※この関数は Ver.3.0.1 以降のファームウェアが必要です。

□ フラッシュメモリ操作関数

これらの関数はユーザーに解放されたフラッシュメモリを操作するための関数です。使用するためにはフラッシュ書換えモードでデバイスがオープンされている必要があります。フラッシュメモリの操作についての詳細は製品のユーザーズマニュアルを参照してください。

USBM_FlashEraseBlk() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_FlashEraseBlk(TW_HANDLE hDev, long Blk)
VB	Function USBM_FlashEraseBlk(ByVal hDev As System.IntPtr, ByVal Blk As Integer) As Integer
VBA	Function USBM_FlashEraseBlk(ByVal hDev As Long, ByVal Blk As Long) As Long

hDev : デバイスのハンドル
Blk : 消去するフラッシュメモリのブロック(1-3)

フラッシュメモリの指定ブロックを消去します。

USBM_FlashRead() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_FlashRead(TW_HANDLE hDev, DWORD Address, void *pBuff, DWORD nData)
VB	Function USBM_FlashRead(ByVal hDev As System.IntPtr, ByVal Address As Integer, ByVal pBuff() As Byte[4], ByVal nData As Integer) As Integer
VBA	Function USBM_FlashRead(ByVal hDev As Long, ByVal Address As Long, ByRef pBuff() As Any, ByVal nData As Long) As Long

hDev : デバイスのハンドル
Address : 読み出すフラッシュメモリアドレス(0x1000-0x3f80)
pBuff : [出力]読み出したデータの格納先
nData : 読み出すバイト数(128 の倍数)

フラッシュメモリの指定アドレスからデータを読み出します。読み出すバイト数は 128 の倍数で指定します。

USBM_FlashWrite() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_FlashWrite(TW_HANDLE hDev, DWORD Address, void *pData, DWORD nData)
VB	Function USBM_FlashWrite(ByVal hDev As System.IntPtr, ByVal Address As Integer, ByVal pData() As Byte[5], ByVal nData As Integer) As Integer
VBA	Function USBM_FlashWrite(ByVal hDev As Long, ByVal Address As Long, ByRef pData() As Any, ByVal nData As Long) As Long

hDev : デバイスのハンドル
Address : 書込みを行うフラッシュメモリアドレス(0x1000-0x3f80)
pData : [入力]書込みデータ
nData : 書き込むバイト数(128 の倍数)

[4] Short 及び Integer のオーバーロードが用意されています。
[5] Short 及び Integer のオーバーロードが用意されています。

フラッシュメモリの指定アドレスに書込みを行います。アドレスの指定は 128 バイト境界にそろえる必要があり、書き込むバイト数も 128 の倍数で指定します。

USBM_UPFlashAttachWriter() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_UPFlashAttachWriter(TW_HANDLE hDev)
VB	Function USBM_UPFlashAttachWriter(ByVal hDev As System.IntPtr) As Integer
VBA	Function USBM_UPFlashAttachWriter(ByVal hDev As Long) As Long

hDev ： デバイスのハンドル

フラッシュ制御用プログラムファイル(M3069FlashWriter.atf)をデバイスにダウンロードし、ユーザープログラムモードでのフラッシュメモリ操作を可能にします。プログラムはユーザーメモリにダウンロードされますので、この関数を呼び出すとユーザーメモリの内容が破壊されます。

ファイルが見つからない場合 TW_FILE_ERROR を返します。
USBM3069-S(L),USBM3069-HS(L),LANM3069-S(L),LANM3069C-S(L),LANM3069D-S(L)のみ使用可能です。

USBM_UPFlashEraseBlk() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_UPFlashEraseBlk(TW_HANDLE hDev, long Blk)
VB	Function USBM_UPFlashEraseBlk(ByVal hDev As System.IntPtr, ByVal Blk As Integer) As Integer
VBA	Function USBM_UPFlashEraseBlk(ByVal hDev As Long, ByVal Blk As Long) As Long

hDev ： デバイスのハンドル
Blk ： 消去するフラッシュメモリのブロック(1-3)

ユーザープログラムモードに設定したデバイスのフラッシュメモリの指定ブロックを消去します。
USBM3069-S(L),USBM3069-HS(L),LANM3069-S(L),LANM3069C-S(L),LANM3069D-S(L)のみ使用可能です。

USBM_UPFlashWrite() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_UPFlashWrite(TW_HANDLE hDev, DWORD Address, void *pData, DWORD nData)
VB	Function USBM_UPFlashWrite(ByVal hDev As System.IntPtr, ByVal Address As Integer, ByVal pData() As Byte[6], ByVal nData As Integer) As Integer
VBA	Function USBM_UPFlashWrite(ByVal hDev As Long, ByVal Address As Long, ByRef pData() As Any, ByVal nData As Long) As Long

hDev ： デバイスのハンドル
Address ： 書込みを行うフラッシュメモリアドレス(0x1000-0x3f80)
pData ： [入力]書込みデータ
nData ： 書き込むバイト数(128 の倍数)

ユーザープログラムモードに設定したデバイスのフラッシュメモリにデータを書き込みます。アドレスの指定は 128 バイト境界にそろえる必要があり、書き込むバイト数も 128 の倍数で指定します。
USBM3069-S(L),USBM3069-HS(L),LANM3069-S(L),LANM3069C-S(L),LANM3069D-S(L)のみ使用可能です。

[6] Short 及び Integer のオーバーロードが用意されています。

□ ハードウェアイベント監視関数

USBM_HW_EVENT 構造体

言語	宣言

C/C++	typedef struct tagHwEvent
	{
	  HWND hRecvWindow;
	  DWORD idRecvThread;
	  LPVOID lpRsv;
	  UINT Message;
	  DWORD EventBits;
	  long PCCnt[4];
	  long PCCmp[4];
	  long ADVal[4];
	  short ADCmp[4];
	} USBM_HW_EVENT;

VB	Public Structure USBM_HW_EVENT
	  Public hRecvWindow As System.IntPtr
	  Public idRecvThread As Integer
	  Public lpRsv As System.IntPtr
	  Public Message As Integer
	  Public EventBits As Integer
	  <MarshalAs(UnmanagedType.ByValArray, SizeConst:=4)> _
	  Public PCCnt() As Integer
	  <MarshalAs(UnmanagedType.ByValArray, SizeConst:=4)> _
	  Public PCCmp() As Integer
	  <MarshalAs(UnmanagedType.ByValArray, SizeConst:=4)> _
	  Public ADVal() As Integer
	  <MarshalAs(UnmanagedType.ByValArray, SizeConst:=4)> _
	  Public ADCmp() As Short
	
	  Public Sub Initialize()
	  ReDim PCCnt(3)
	  ReDim PCCmp(3)
	  ReDim ADVal(3)
	  ReDim ADCmp(3)
	  End Sub
	End Structure

VBA	Public Type USBM_HW_EVENT
	  hRecvWindow As Long
	  idRecvThread As Long
	  lpRsv As Long
	  Message As Long
	  EventBits As Long
	  PCCnt(3) As Long
	  PCCmp(3) As Long
	  ADVal(3) As Long
	  ADCmp(3) As Integer
	End Type

hRecvWindow ： イベント発生時にメッセージを受け取るウィンドウのハンドルを指定
idRecvThread ： イベント発生時にメッセージを受け取るスレッドの ID を指定
lpRsv ： 予約
Message ： メッセージ番号。アプリケーション独自のメッセージでは通常 WM_APP(0x8000)以上の値
EventBits ： 監視するイベントに対応するビットを 1 とします
ビット 0 ： PC0 を監視する
ビット 1 ： PC1 を監視する
ビット 2 ： PC2 を監視する
ビット 3 ： PC3 を監視する
ビット 4 ： AD0 を監視する
ビット 5 ： AD1 を監視する
ビット 6 ： AD2 を監視する
ビット 7 ： AD3 を監視する
ビット 31 ： ユーザーファームから独自のイベントを通知する
上記以外 ： 予約。0 としてください
PCCnt[4] ： パルスカウンタ各チャンネルの閾値
PCCmp[4] ： パルスカウンタ各チャンネルの閾値との比較方法。下記のいずれか
TWB_CMP_NO ： カウンタに変化があれば毎回イベントを発生
TWB_CMP_GE ： カウンタの値が PCCnt 値以上になった場合イベントを発生
TWB_CMP_LE ： カウンタの値が PCCnt 値以下になった場合イベントを発生
ADVal[4] ： AD 各チャンネルの閾値。上位 16 ビットは 0、下位 16 ビットの MSB から 10 ビット使用
ADCmp[4] ： AD 各チャンネルの閾値との比較方法。大きさはヒステリシス、極性により以下の動作
0 以上の場合 ： アナログ入力値が ADVal 以上でイベント発生
負の場合 ： アナログ入力値が ADVal 以下でイベント発生

ハードウェアイベントを監視するためのパラメータを設定します。USBM_SetHwEvent() 関数の呼び出しに使用します。
Visual Basic では構造体を使用する前に Initialize()メソッドを呼び出して初期化を行ってください。

USBM_SetHwEvent() HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_SetHwEvent(TW_HANDLE hDev, USBM_HW_EVENT *pHwEvent)
VB	Function USBM_SetHwEvent(ByVal hDev As System.IntPtr, ByRef pHwEvent As USBM_HW_EVENT) As Integer
VBA	Function USBM_SetHwEvent(ByVal hDev As Long, ByRef pHwEvent As USBM_HW_EVENT) As Long

hDev ： デバイスのハンドル
pHwEvent ： [入力]イベント監視のパラメータを USBM_HW_EVENT 構造体で渡します

パルスカウンタ 0-3、AD コンバータ 0-3 チャンネルの入力を監視し、指定値となった場合に Windows のメッセージで通知します。監視を終了する場合には pHwEvent に NULL を指定するか、EventBits メンバを 0 として呼び出します。
通知先のウィンドウやスレッド、メッセージ番号等を変更する場合は、一旦監視を終了した後、変更後のパラメータを設定してください。詳しい使用方法は製品マニュアルを参照してください。

※この関数は Ver.4.0.1 以降のファームウェアが必要です。

□ その他の関数

USBM_Abort() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_Abort(TW_HANDLE hDev)
VB	Function USBM_Abort(ByVal hDev As System.IntPtr) As Integer
VBA	Function USBM_Abort(ByVal hDev As Long) As Long

hDev ： デバイスのハンドル

USBM_ADBRead(),USBM_SCIRead()関数がタイムアウトした場合に、マイコン側の処理を中止させる場合に使用します。また、USBM_ADStart()関数での AD 変換を中止する場合にも使用します。

USBM_GetFTHandle() USB

言語	関数宣言
C/C++	FT_HANDLE USBM_GetFTHandle(TW_HANDLE hDev)
VB	Function USBM_GetFTHandle(ByVal hDev As System.IntPtr) As Integer
VBA	Function USBM_GetFTHandle(ByVal hDev As Long) As Long

hDev ： デバイスのハンドル

戻り値 ： D2XX API 呼び出し用ハンドル

戻り値として FTDI 社の D2XX API を呼び出すためのハンドルを返します。

USBM_GetQueueStatus() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_GetQueueStatus(TW_HANDLE hDev, DWORD *pnQueue)
VB	Function USBM_GetQueueStatus(ByVal hDev As System.IntPtr, ByRef pnQueue As Integer) As Integer
VBA	Function USBM_GetQueueStatus(ByVal hDev As Long, ByRef pnQueue As Long) As Long

hDev ： デバイスのハンドル
pnQueue ： [出力]受信バイト数(受信ワード数)の格納先

デバイスから受信したデータ数を取得します。USBM_ADStart()使用時に受信した変換結果のバイト数を取得する場合や、ユーザーファームから送信されたデータを受け取る場合に使用します。
返される値は USBM_Read()で読み出せるバイト数を示しますが、HS デバイスの場合で、USBM_Read16()を使用する場合に限りワード数を示します。

USBM_GetSocket() LAN

言語	関数宣言
C/C++	UINT_PTR USBM_GetSocket(TW_HANDLE hDev)
VB	Function USBM_GetSocket(ByVal hDev As System.IntPtr) As System.IntPtr
VBA	Function USBM_GetSocket(ByVal hDev As Long) As Long

hDev　：デバイスのハンドル

戻り値：ハンドルに結びついたソケット

戻り値としてハンドルに結びついた SOCKET を返します。LAN デバイスでは無い場合や無効なハンドルが渡された場合は TW_INVALID_SOCKET が返ります。

USBM_Purge() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_Purge(TW_HANDLE hDev, DWORD dwMask)
VB	Function USBM_Purge(ByVal hDev As System.IntPtr, ByVal dwMask As Integer) As Integer
VBA	Function USBM_Purge(ByVal hDev As Long, ByVal dwMask As Long) As Long

hDev　：デバイスのハンドル
dwMask：クリアするバッファを指定
　　USBM_PURGE_RX 受信バッファをクリア
　　USBM_PURGE_TX 送信バッファをクリア

デバイスとの通信に使用するバッファをクリアします。特にデバイスからデータを読み出す関数がタイムアウトした場合、関数から復帰した後にデバイスから送られたデータが受信バッファに溜まっている場合があるので、それらをクリアするために必要となります。
LAN デバイスでは USBM_PURGE_TX は無効です。

USBM_SetTimeouts() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_SetTimeouts(TW_HANDLE hDev, DWORD dwReadTimeout, DWORD dwWriteTimeout)
VB	Function USBM_SetTimeouts(ByVal hDev As System.IntPtr, ByVal dwReadTimeout As Integer, ByVal dwWriteTimeout As Integer) As Integer
VBA	Function USBM_SetTimeouts(ByVal hDev As Long, ByVal dwReadTimeout As Long, ByVal dwWriteTimeout As Long) As Long

hDev　　：デバイスのハンドル
dwReadTimeout　：読み出しのタイムアウト(msec 単位)
dwWriteTimeout：書き込みのタイムアウト(msec 単位)

デバイスとの送信及び受信の際に適用されるタイムアウト時間を設定します。デフォルトの設定では、送信、受信とも 5 秒です。

USBM_Read() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_Read(TW_HANDLE hDev, void *pData, DWORD nData, DWORD *pRead)
VB	Function USBM_Read(ByVal hDev As System.IntPtr, ByVal Data() As Byte[7], ByVal nData As Integer, ByRef pRead As Integer) As Integer
VBA	Function USBM_Read(ByVal hDev As Long, ByRef pData() As Any, ByVal nData As Long, ByRef pRead As Long) As Long

hDev ： デバイスのハンドル
pData ： [出力]読み出したデータの格納先へのポインタ
nData ： 読み出すバイト数(0～65536)
pRead ： [出力]読み出したデータのバイト数の格納先へのポインタ

デバイスから送られたデータを読み出します。指定バイト数のデータを読み出すかタイムアウトするまで関数から戻りません。

USBM_Write() USB HS LAN

言語	関数宣言
C/C++	TW_STATUS USBM_Write(TW_HANDLE hDev, void *pData, DWORD nData, DWORD *pWritten)
VB	Function USBM_Write(ByVal hDev As System.IntPtr, ByVal Data() As Byte[8], ByVal nData As Integer, ByRef pWritten As Integer) As Integer
VBA	Function USBM_Write(ByVal hDev As Long, ByRef pData() As Any, ByVal nData As Long, ByRef pWritten As Long) As Long

hDev ： デバイスのハンドル
pData ： [入力]送信するデータへのポインタ
nData ： 送信するバイト数(0～65536)
pWritten ： [出力]実際に送信したバイト数の格納先

デバイスにデータを送信します。

USBM_Read16() HS

言語	関数宣言
C/C++	TW_STATUS USBM_Read16(TW_HANDLE hDev, void *pData, DWORD nWords, DWORD *pnRead)
VB	Function USBM_Read16(ByVal hDev As System.IntPtr, ByVal pData() As Short[9], ByVal nWords As Integer, ByRef pnRead As Integer) As Integer
VBA	Function USBM_Read16(ByVal hDev As Long, ByRef pData() As Any, ByVal nWords As Long, ByRef pnRead As Long) As Long

hDev ： デバイスのハンドル
pData ： [出力]読み出したデータの格納先へのポインタ
nWords ： 読み出すワード数(0～65536)
pnRead ： [出力]読み出したデータのワード数の格納先

HS デバイスから 16 ビット単位でデータを受信します。ユーザーファームから SRV_Transmit16()、

[7] Short および Integer のオーバーロードが用意されています。
[8] Short および Integer のオーバーロードが用意されています。
[9] Integer のオーバーロードが用意されています。

SRV_DmaTransmit16()で送信されたデータを受信するのに使用します。この関数を呼び出すためには USBM_MODE_BUS16 オプションを指定してデバイスをオープンする必要があります。

USBM_Write16() HS

言語	関数宣言
C/C++	TW_STATUS USBM_Write16(TW_HANDLE hDev, void *pData, DWORD nWords, DWORD *pnWritten)
VB	Function USBM_Write16(ByVal hDev As System.IntPtr, ByVal pData() As Short[9], ByVal nWords As Integer, ByRef pnWritten As Integer) As Integer
VBA	Function USBM_Write16(ByVal hDev As Long, ByRef pData() As Any, ByVal nWords As Long, ByRef pnWritten As Long) As Long

hDev : デバイスのハンドル
pData : [入力]送信するデータへのポインタ
nWords : 送信するワード数(0～65536)
pnWritten : [出力]実際に送信したワード数の格納先

HS デバイスに 16 ビット単位でデータを送信します。送信したデータはユーザーファームの SRV_Receive16()、SRV_DmaReceive16()で受信します。この関数を呼び出すためには USBM_MODE_BUS16 オプションを指定してデバイスをオープンする必要があります。

USBM_GetIFType() USB HS LAN

言語	関数宣言
C/C++	DWORD USBM_GetIFType(TW_HANDLE hDev)
VB	Function USBM_GetIFType(ByVal hDev As System.IntPtr) As Integer
VBA	Function USBM_GetIFType(ByVal hDev As Long) As Long

hDev : デバイスのハンドル

戻り値 : デバイスのインタフェース種別とデバイスのモード
インタフェース種別として以下のビットのいずれかが 1 になります
USBM_IF_USB(0x80000000) : USB デバイス
USBM_IF_LAN(0x40000000) : LAN デバイス
USBM_IF_HS(0x20000000) : HS デバイス

HS デバイスの場合、USB インタフェース IC との接続バス幅を下のビットで示します
USBM_MODE_BUS16(0x00400000) : USB インタフェース IC との 16 ビット接続を示すビット

フラッシュ書換えモードの場合上記に加え以下のビットが 1 になります
USBM_MODE_FLASH(0x00800000) : フラッシュ書換えモードを示すビット

接続しているデバイスのインタフェースタイプを取得します。ハンドルが無効な場合 0 が返ります。

USBM_SetNetworkPort() LAN

言語	関数宣言
C/C++	TW_STATUS USBM_SetNetworkPort(DWORD PortNumber)
VB	Function USBM_SetNetworkPort(ByVal PortNumber As Integer) As Integer
VBA	Function USBM_SetNetworkPort(ByVal PortNumber Long) As Long

PortNumber : ポート番号(0～65535)

LAN デバイスと通信する場合のポート番号を指定します。ここで指定するポート番号はデバイス側のものですので設定ツールでデバイスに書き込んだものと同じものを指定してください。一般的にポート番号として使用できるのは 49152～65535 の値です。0 を指定するとデフォルト値(49152)に設定されます。

USBM_SetPassword() LAN

言語	関数宣言
C/C++	TW_STATUS USBM_SetPassword(TW_CSTR *pPass)
VB	Function USBM_SetPassword(ByVal pPass As String) As Integer
VBA	Function USBM_SetPassword(ByVal pPass As String) As Long

pPass : パスワード(NULL 終端文字列)

LAN デバイスと接続する際のパスワードを入力します。パスワードは設定ツールでデバイスに書き込んだものと同じものを指定してください。pPass が NULL または "" の場合、デフォルトのパスワードが使用されます。

□ Visual Basic 用ヘルパー関数

以下の関数は Visual Basic のプリミティブな変数と、16 ビットの符号無し整数を変換するためのヘルパー関数です。内容は、VisualBasic.NET、VisualBasic におけるプロトタイプ、変数の説明、動作説明の順になっています。

USBM_ToUINT16() USB HS LAN

言語	関数宣言
VB	Function USBM_ToUINT16(ByVal data As Integer) As Short
VBA	Function USBM_ToUINT16(ByVal data As Long) As Integer

data : 16 ビットコードに変換したい値

0～65535 の値を 16 ビットコード(0000h～FFFFh)に変換します。
data が負の場合は 0 として、また 65535 を超える場合は 65535 として変換します。

USBM_ToINT32() USB HS LAN

言語	関数宣言
VB	Function USBM_ToINT32(ByVal data As Short) As Integer
VBA	Function USBM_ToINT32(ByVal data As Integer) As Long

data : 32 ビット値に変換したい値

data を符号無し 16 ビットコード(0000h～FFFFh)とみなし、32 ビット値(0～65535)に変換します。

設定ファイル

設定ファイルを使ってライブラリ動作の一部を変更することができます。設定ファイル名は「USBM3069.ini」とし、アプリケーションプログラムと同一フォルダに置いてください。
以下に設定ファイルの例を示します。

```
[COMMON]
SerchLANM=1

[LANM3069]
PortNumber=49152
```

図 1　設定ファイルの例

[COMMON] セクションには USB デバイスと LAN デバイス共通のオプションが含まれます。
[LANM3069] セクションには LAN デバイス専用のオプションが含まれます。
各セクションに現在設定可能なキーと意味を以下に示します。

表 7　[COMMON]セクションのキー

キー	値	意味
SerchLANM	0	USBM_Open() 関数で LAN デバイスを検索しません。
	1(初期値)	USBM_Open() 関数で LAN デバイスを検索します。

表 8　[LANM3069]セクションのキー

キー	値	意味
PortNumber	49152～65535 (初期値 49152)	LAN デバイスとのアクセスに使用するネットワークプロトコルのポート番号を指定します。デバイス側の設定と一致している必要があります。
ConnectTimeout	32bit 値 (初期値 1500)	LAN デバイスとの接続時のタイムアウト時間を msec 単位で指定します。
AcceptTimeout	32bit 値 (初期値 0)	USBM_Accept()の待ち時間を msec 単位で指定します。

Appendix

定数、変数型の名称変更について

以前のバージョンで「FT_～」と表記していた定数、変数の型名の一部は「TW_～」または「USBM_～」という名称に改められました。ただし、意味は同様となっていますので以前の名称でも使用可能です。

D2XX 関数の呼び出しについて

FTDI 社標準の D2XX API 関数（FT_Purge()など）の使用について、以前のマニュアルでは使用方法の一部としてご案内していましたが、本リファレンスに記載のライブラリバージョンでは、デバイスのハンドルに互換が無く、これらの関数を直接呼び出すことはできなくなっています。D2XX API 関数の呼び出しが必要な場合は、下記のようにハンドル変換用の関数を使用してください。

```
FT_Purge(USBM_GetFTHandle(hDev), USBM_PURGE_RX)
```

引数の初期値変更について

USBM_PortBRead() 、*USBM_PortBWrite()* 関数の *Dma* 引数の初期値が以前のバージョン（2.2.x.x 以前）では TRUE（DMA を使用する）でしたが、FALSE（DMA を使用しない）に変更になっています。引数を省略して呼び出していた場合、動作が変わってしまいますのでご注意ください。

サポート情報

製品に関する情報、最新のファームウェア、ユーティリティなどは弊社ホームページにてご案内しております。また、お問い合わせ、ご質問などは下記までご連絡ください。

テクノウェーブ(株)

URL ： https://www.techw.co.jp

E-mail ： support@techw.co.jp

(1) 本書、および本製品のホームページに掲載されている応用回路、プログラム、使用方法などは、製品の代表的動作・応用例を説明するための参考資料です。これらに起因する第三者の権利（工業所有権を含む）侵害、損害に対し、弊社はいかなる責任も負いません。

(2) 本書の内容の一部または全部を無断転載することをお断りします。

(3) 本書の内容については、将来予告なしに変更することがあります。

(4) 本書の内容については、万全を期して作成いたしましたが、万一ご不審な点や誤り、記載もれなど、お気づきの点がございましたらご連絡ください。

改訂記録

年月	版	改訂内容
2007年2月	初	
2007年3月	2	・Appendixに内容追加 ・誤記の修正
2007年9月	3	・USBM3069.DLL Ver.3.3.x.xに対応
2008年6月	4	・USBM3069.DLL Ver.3.4.x.xに対応
2009年4月	5	・USBM3069.DLL Ver.3.5.x.xに対応 ・誤記の修正
2010年6月	6	・USBM3069.DLL Ver.3.6.x.xに対応 ・誤記の修正
2011年9月	7	・USBM3069.DLL Ver.4.0.x.xに対応 ・誤記の修正
2019年4月	8	・対応製品を追加
2021年3月	9	・対応製品を追加