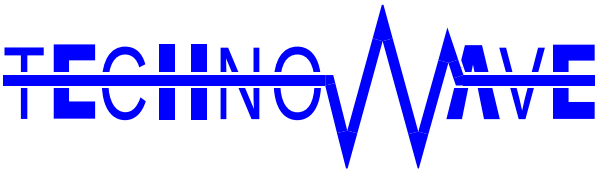
M3069 マイコンボード
プログラミング・リファレンス



テクノウェーブ株式会社

目次

1. はじめに

□ 本リファレンスについて

本リファレンスは表 1 の対応製品を Windows®搭載パソコンから制御する方法と、製品の制御用ライブラリ(「TWB ライブラリ」)の各関数についての説明を記載しています。TWB ライブラリは、弊社マイコンボード製品共通のライブラリで、提供される関数は一部を除いて、対応製品全てでご利用になれます。

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
TWB.DLL	1.0.x.x [1]	Windows 7 以降

□ 「USBM ライブラリ」との比較

TWB ライブラリは、新しい M3069 マイコンボードシリーズ制御用ライブラリです。従来の「USBM ライブラリ」(USBM3069.dll)と比較して以下のような特徴があります。

- 目的の機能に合わせた分かり易いインタフェースと関数構成
- Visual Studio®のインテリセンス(入力補助)の活用
- C# 用定義ファイルの提供
- ワイド文字列のサポート
- ユーザーファーム開発用ライブラリ(TWFA ライブラリ)と対応した関数構成

従来のライブラリファイルは TWB ライブラリと並行して提供されますので、既に USBM ライブラリで開発されたアプリケーションプログラムは従来通りご利用いただけます。また、デバイスを操作するためのハンドル値は 2 つのライブラリで共通となっていますので、両方のライブラリを同時に利用することも可能です [2]。

- TWB ライブラリのご利用にはシステムファームの Ver.5.1.1 以降が必要です。ご利用の製品のバージョンが古い場合は、予め更新してご利用ください。

Windows、Visual Studio は米国 Microsoft Corporation の米国およびその他の国における登録商標です。

[1] x にはより細かなバージョンを示す数値が入ります。

[2] 「USBM ライブラリ」の使用方法は旧版のユーザーズマニュアルと「USBM ライブラリ関数リファレンス」(USBMLibrary.pdf)を参照してください。

□ 本リファレンス内の表記について

対応製品の表記

本リファレンス内では対応製品を「製品」または「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は表 3 に従い表記します。USB インタフェースの製品に関してはフルスピード製品と、ハイスピード対応製品の区別が必要な場合、表のようにそれぞれ USB(FS)デバイス、USB(HS)デバイスと表記します。

表 3　ホストインタフェース別の製品表記方法

説明文での表記		対応製品
USB デバイス	USB(FS)デバイス	『USBM3069』 / 『USBM3069F』 / 『USBM3069-S』 / 『USBM3069-SL』
	USB(HS)デバイス	『USBM3069-HS』 / 『USBM3069-HSL』
LAN デバイス		『LANM3069』 / 『LANM3069-S』 / 『LANM3069-SL』 / 『LANM3069C』 / 『LANM3069C-S』 / 『LANM3069C-SL』 / 『LANM3069D-S』 / 『LANM3069D-SL』

電気的状態

本リファレンス内ではハードウェアの各電気的状態について下記のように表記いたします。

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。

数値

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。

関数・構造体名

本文で関数名を表記する場合、C/C++、Visual Basic®、Visual Basic for Applications の名称に従い"*TWB_Open()*"のように表記します。C#の場合、これと対応する関数は *Techw.IO* 名前空間の *TWB* クラスのスタティックメンバ関数で"*Techw.IO.TWB.Open()* "となります。構造体名についても同様です。

関数の宣言を示す場合、C/C++、Visual Basic (.NET 以後)、Visual Basic for Applications (以下 VBA)、C# の順で、それぞれの言語における関数宣言が記載されます(表 5)。C# の場合は、名前空間とクラス名は省略して記述しています。

Visual Basic は米国 Microsoft Corporation の米国およびその他の国における登録商標です。

表 5　関数宣言の表記例

言語	関数宣言
C/C++	TW_STATUS TWB_Open(TW_HANDLE *phDev, LPCTSTR pUuid, DWORD Number, long Opt)
VB	Function TWB_Open(ByRef phDev As System.IntPtr, ByVal pUuid As String, ByVal Number As Integer, ByVal Opt As TWB_OPEN_OPT) As Integer
VBA	Function TWB_Open(ByRef phDev As Long, ByVal pUuid As String, ByVal Number As Long, ByVal Opt As TWB_OPEN_OPT) As Long
C#	STATUS Open(out System.IntPtr phDev, string pUuid, int Number, OPEN_OPT Opt)

引数の入力候補

各関数の引数の中には、入力できる値が限定されていて、ある定数を入力することが適当なものがあります。そのような場合、各開発環境の入力支援機能(インテリセンス)を十分活用できるよう、言語毎に異なった定数や列挙型を定義しています。

表 6 は *TWB_Open()* 関数の *Opt* 引数の入力候補の一部です。引数の入力候補は表のように各言語別に記述方法が記載されます。

"C/C++"と書かれた行はCおよびC++で使用できる記述方法です。この値は*#define*で定義された定数です。

"C++"と書かれた行は C++で使用できる記述方法です。定数専用に宣言されたクラスのスタティックメンバになっています。Visual Studio でこの定数を入力する場合、最初に"TWB::"と入力すると画面に入力候補が表示されますので、定数を選択して入力を行ってください。

"VB/VBA"と書かれた行は Visual Basic と VBA で使用可能な記述方法です。この場合、関数の引数自体が列挙型となっており定数は列挙子です。

"C#"と書かれた行はC#で使用可能な記述方法です。この場合も Visual Basic 同様に関数の引数が列挙型となっています。名前空間は省略して記述しています。

表 6　引数の入力候補の例

言語	値	説明
C/C++	TWB_IF_USB_FS	USB(FS)デバイスに接続します。
C++	TWB::OPEN_OPT::IF_USB_FS	
VB/VBA	TWB_OPEN_OPT.IF_USB_FS	
C#	TWB.OPEN_OPT.IF_USB_FS	
C/C++	TWB_IF_USB_HS	USB(HS)デバイスに接続します。
C++	TWB::OPEN_OPT::IF_USB_HS	
VB/VBA	TWB_OPEN_OPT.IF_USB_HS	
C#	TWB.OPEN_OPT.IF_USB_HS	
C/C++	TWB_IF_LAN	LAN デバイスに接続します。
C++	TWB::OPEN_OPT::IF_LAN	
VB/VBA	TWB_OPEN_OPT.IF_LAN	
C#	TWB.OPEN_OPT.IF_LAN	

Null 値

関数の引数の中には Null 値(空値)を要求するものがあります。本文中で Null 値と表記した場合、各言語での対応する記述方法は表 7 のようになります。

表 7　Null 値

言語	記述方法
C/C++	NULL
VB	Nothing
VBA	vbNullString
C#	null

□ LabVIEW™のご利用について

付属の VI ライブラリにはプログラミング言語用の各関数と対応したアイコンが用意されていますので、製品の制御方法については本マニュアルを参照してください。各 VI の詳細や LabVIEW 固有の情報は、VI ライブラリをインストールする際に同時にインストールされるヘルプファイルに詳しい説明を記載しています。

LabVIEW は、National Instruments Corporation の商標です。

2. プログラミングの準備

プログラミングに必要なファイルは、製品付属の設定ツール（「USBMTools」、「LANMTools」）をインストールした場合、ローカルドライブにコピーが作られ、デフォルトの設定では、Windows 10 の場合[スタート]メニュー→[アプリの一覧]→[テクノウェーブ]→[ライブラリ]を、Windows 7 の場合[スタート]メニュー→[すべてのプログラム]→[テクノウェーブ]→[ライブラリ]を選択して表示することができます。

また、これらのファイルは、弊社のホームページ「https://www.techw.co.jp/SupportFrm.html?pid=USBM3069F」の「I/O ボード・I/O ユニット用ライブラリ」からダウンロードすることもできます。

□ C/C++での開発に必要なファイル

表 8 は C/C++で開発を行うために必要なファイルです。

表 8　C/C++での開発に必要なファイル

ファイル名	説明	ダウンロードファイル内の格納フォルダ
TWB.h	TWB ライブラリを使用するためのヘッダーファイル	¥DLL
TWB.lib(32bit 用)	TWB ライブラリを静的にリンクするための lib ファイル	¥DLL
TWB.lib(64bit 用)		¥DLL¥X64

「TWB.h」は、TWB ライブラリの関数や定数を使用するソースファイルでインクルードしてください。

「TWB.lib」はプロジェクトをビルドする際のリンクファイルに含める必要があります。Visual Studio では、リスト 1 のように *#pragma* を使用してソースファイル中でリンク指定することもできます。

リスト 1　インクルードとリンク指定

```
#include"TWB.h"
#pragma comment(lib, "TWB.lib")
```

これらのファイルはコンパイラがビルド時に検索できるフォルダにコピーしておく必要があります。最も簡単な方法は、ビルドするプロジェクトと同一フォルダにコピーすることです。

複数のプロジェクトを開発する場合は、これらのファイルを格納したフォルダを、開発環境の標準のインクルードパスや標準のリンクパスに追加すると便利です。追加の方法は開発環境によって異なりますので、それぞれのオンラインヘルプなどを参照してください。

- 「TWB.h」は WIN32 API 固有の型などを使用しています。「コンソール アプリケーション」や「フォーム アプリケーション」を作成する場合には、「TWB.h」より前に「Windows.h」のインクルードが必要な場合があります。

□ Visual Basic、C# での開発に必要なファイル

表 9 は Visual Basic、または、C# で開発を行うために必要なファイルです。

表 9　Visual Basic、C#での開発に必要なファイル

開発環境	ファイル名	説明	ダウンロードファイル内の格納フォルダ
Visual Basic	TWB.vb	TWB ライブラリを使用するための定義ファイル	¥DLL
Visual C#®	TWB.cs		

どちらの開発環境の場合も、Visual Studio の「ソリューション エクスプローラ」を開き、対応するファイルを開発プロジェクトの中にドラッグ･アンド･ドロップで追加することで、TWB ライブラリの呼び出しが可能になります。これらのファイルは 32 ビット、64 ビットのどちらのプログラムを作成する場合にも共通で利用可能です。

□ Visual Basic for Applications での開発に必要なファイル

表 10 は VBA で開発を行うために必要なファイルです。

表 10　Visual Basic for Applications での開発に必要なファイル

ファイル名	説明	ダウンロードファイル内の格納フォルダ
TWB.bas	TWB ライブラリを使用するための定義ファイル	¥DLL

開発を行うアプリケーションソフトで [Alt] + [F11]キーを押し、Visual Basic Editor を起動し、上記ファイルをプロジェクトウィンドウにドラッグ・アンド・ドロップで追加することで、TWB ライブラリの呼び出しが可能になります。

- プロジェクトに追加したファイルは、ドキュメントファイル内にコピーが作成されます。ファイルを更新する場合は、以前に追加したファイルを一度解放し、新しいファイルを追加してください。

□ LabVIEW での開発に必要なファイル

表 11 は LabVIEW で開発を行うために必要なファイルです。各製品のマニュアルに従って LabVIEW 用 VI ライブラリをインストールすると、対象の LabVIEW のユーザーライブラリに TWB-VI ライブラリが追加されます。

表 11　LabVIEW での開発に必要なファイル

ライブラリ名	説明
TWB-VI ライブラリ	TWB ライブラリを使用するための VI ライブラリ

Visual C# は米国 Microsoft Corporation の米国およびその他の国における登録商標です。

□ 実行時に必要なファイルについて

表 12 は開発したアプリケーションプログラムを実行する場合に必要となるファイルです。これらのファイルは、設定ツール（「USBMTools」、「LANMTools」）をインストールした場合は、自動的にシステムフォルダ（「C:¥Windows¥System32」など）にコピーされます。設定ツールをインストールしていないパソコンで製品を利用する際には表 13 の「コピー先」フォルダにファイルをコピーするようにしてください。

これらのファイルは、弊社のホームページ「https://www.techw.co.jp/SupportFrm.html?pid=USBM3069F」の「I/O ボード・I/O ユニット用ライブラリ」からダウンロードすることもできます。

表 12　プログラムの実行に必要なファイル

ファイル名	説明	ダウンロードファイル内の格納フォルダ	
TWB.dll	TWB ライブラリ本体。製品の設定ツールをインストールすることでシステムフォルダにコピーされます。	32 ビット版	¥DLL
		64 ビット版	¥DLL¥X64
USBM3069.dll	製品を制御するための低水準ライブラリ。製品の設定ツールをインストールすることでシステムフォルダにコピーされます。	32 ビット版	¥DLL
		64 ビット版	¥DLL¥X64
M3069FlashWriter.atf	ユーザープログラムモードで、フラッシュメモリへの書込みを行うために必要なファイル。製品の設定ツールをインストールすることでシステムフォルダにコピーされます。	共通	¥DLL

表 13　必要な DLL と対応するシステムフォルダ

OS	実行するプログラム	必要な DLL	コピー先
64 ビット OS	32 ビットプログラム	32 ビット版 DLL	C:¥Windows¥SysWOW64
	64 ビットプログラム	64 ビット版 DLL	C:¥Windows¥System32
32 ビット OS	32 ビットプログラム	32 ビット版 DLL	C:¥Windows¥System32

- LabVIEW や LabVIEW で作成したプログラムは、32 ビットプログラム[3]です。

[3] 64bit 版の LabVIEW には対応していません。

□ サンプルプログラム

サンプルプログラムは、弊社のホームページ「https://www.techw.co.jp/SupportFrm.html?pid=USBM3069F」の「サンプルプログラム」からダウンロードいただけます。

表 14　言語別ソリューションファイル

言語	ダウンロードファイル内の格納フォルダ	ソリューションファイル
Visual C++(MFC) [4]	¥M3069_Samples	VCSamples.sln
Visual Basic [4]		VBSamples.sln
Visual C# [4]		CSSamples.sln
Visual Basic for Application	¥M3069_Samples¥VBASamples	-
LabVIEW [5]	¥M3069_Samples¥LabVIEW_Samples	-

32 ビットプログラムか 64 ビットプログラムかを調べる方法

64 ビット OS 上で動作しているプログラムが、32 ビットプログラムか 64 ビットプログラムかは「タスク マネージャー」で調べることができます。[Ctrl] + [Alt] + [Del]キーを押して「タスクマネージャー」を起動し、[プロセス]タブを開きます。実行ファイルの右に"*32"という表示があれば 32 ビットプログラムです。

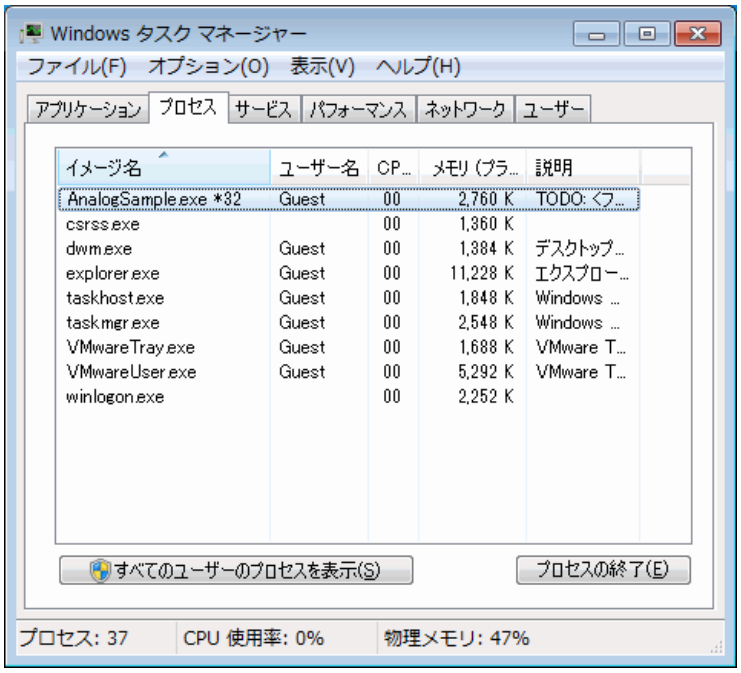


図 1　32 ビットプログラムの「タスク マネージャー」での表示

[4] Visual Studio 2005 で作成されています。ご利用のバージョンによっては変換作業が必要になります(ソリューションファイルを開くと自動的に変換ウィザードが起動します)。
[5] LabVIEW 7.1 で作成されています。

3. プログラミング

□ 接続

デバイスを制御するには、まず接続作業を行い、ハンドルを取得する必要があります。ハンドルとは接続時に決定される整数値で、接続中のデバイスを識別する ID と考えることができます(図 2)。以降の操作は取得したハンドルを使用して行いますので、ハンドルの値は操作を終了するまで記憶しておく必要があります。

また、デバイスの操作を終える場合はハンドルのクローズを行います。デバイスは 1 つのプログラムとしか接続ができませんので、ハンドルをクローズしていないプログラムが実行中の場合、他のプログラムからそのデバイスに接続することはできません。

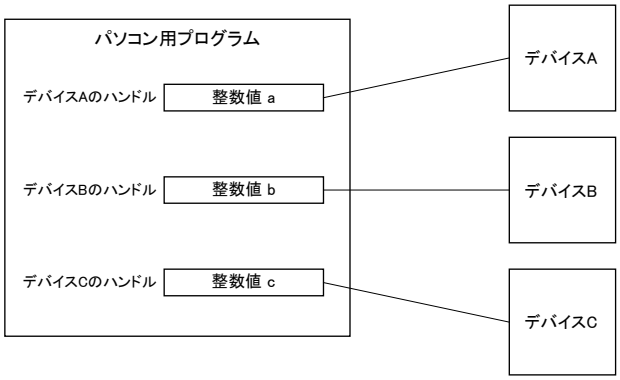


図 2　ハンドル

表 15　接続、初期化、終了に使用する関数

関数名	説明
TWB_Open()	製品情報を指定してデバイスに接続します。 LAN デバイスの場合、ローカルネットワーク上のデバイスのみ対象になります。
TWB_OpenByAddress()	LAN デバイスのアドレス、USB デバイスの USB シリアルを指定してデバイスに接続します。 LAN デバイスの場合、IP アドレスの他にドメイン名を指定することもできます。また、必要な場合にはポート番号の指定も可能です。
TWB_Listen()	LAN デバイス専用です。クライアントモードに設定されたデバイスの接続要求を受け入れるためのソケットを作成します。
TWB_Accept()	LAN デバイス専用です。クライアントモードに設定されたデバイスの接続要求があれば接続を行います。
TWB_Close()	ハンドルをクローズし、デバイスの操作を終了します。
TWB_CloseAll()	プロセスが接続している全てのデバイスの操作を終了します。
TWB_CloseListenSocket()	LAN デバイス専用です。クライアントモードのデバイスを受け入れるためのソケットをクローズします。
TWB_Initialize()	デバイスの再初期化が必要な場合呼び出します。必須ではありません。
TWB_SetPassword()	LAN デバイスと接続する際のパスワードを指定します。ネットワーク設定ツール「LANMConfig」で[パスワード]を設定した場合、接続前に本関数を呼び出します。
TWB_SetNetworkPort()	LAN デバイスと接続する際に使用するポート番号を指定します。ネットワーク設定ツール「LANMConfig」で[ポート番号]を"49152"以外に設定した場合、接続前に本関数を呼び出します。

製品情報を指定してデバイスに接続する

TWB_Open() 関数(表 16)を呼び出すことで、個々の製品を識別するための UUID や装置番号を指定してデバイスに接続することができます。

UUID や装置番号は設定ツール「M3069PIWriter」を使用して、デバイス内のフラッシュメモリに書き込むことができます(ツールの使用方法はオンラインヘルプやユーザーズマニュアルを参照してください)。

表 16　TWB_Open() の関数宣言

言語	関数宣言
C/C++	TW_STATUS TWB_Open(TW_HANDLE *phDev, LPCTSTR pUuid, DWORD Number, long Opt)
VB	Function TWB_Open(ByRef phDev As System.IntPtr, ByVal pUuid As String, ByVal Number As Integer, ByVal Opt As TWB_OPEN_OPT) As Integer
VBA	Function TWB_Open(ByRef phDev As Long, ByVal pUuid As String, ByVal Number As Long, ByVal Opt As TWB_OPEN_OPT) As Long
C#	STATUS Open(out System.IntPtr phDev, string pUuid, int Number, OPEN_OPT Opt)

pUuid 引数にはデバイスに書き込んだ UUID を文字列で指定します。指定が無い場合には Null 値とします。

Number 引数にはデバイスに書き込んだ装置番号[6]を指定します。指定が無い場合には"0"とします。

Opt 引数にはデバイスのインタフェースやモードなどをオプションとして渡します(表 17)。インタフェースに関するオプションは必須で、デバイスを限定する必要が無い場合は *TWB_IF_ANY*（相当の値）を指定してください。

表 17　TWB_Open()の Opt 引数に指定する値

言語	値	説明
C/C++	TWB_IF_USB_FS	USB(FS)デバイスに接続します。
C++	TWB::OPEN_OPT::IF_USB_FS	
VB/VBA	TWB_OPEN_OPT.IF_USB_FS	
C#	TWB.OPEN_OPT.IF_USB_FS	
C/C++	TWB_IF_USB_HS	USB(HS)デバイスに接続します。
C++	TWB::OPEN_OPT::IF_USB_HS	
VB/VBA	TWB_OPEN_OPT.IF_USB_HS	
C#	TWB.OPEN_OPT.IF_USB_HS	
C/C++	TWB_IF_LAN	LAN デバイスに接続します。
C++	TWB::OPEN_OPT::IF_LAN	
VB/VBA	TWB_OPEN_OPT.IF_LAN	
C#	TWB.OPEN_OPT.IF_LAN	
C/C++	TWB_IF_ANY	製品のインタフェースを問わずに接続します。
C++	TWB::OPEN_OPT::IF_ANY	
VB/VBA	TWB_OPEN_OPT.IF_ANY	
C#	TWB.OPEN_OPT.IF_ANY	

[6] 「M3069PIWriter」の[Number]に設定した値。

表 17　TWB_Open()の Opt 引数に指定する値(続き)

C/C++	TWB_MODE_BUS16	USB(HS)デバイス専用。デバイス内蔵のマイコンと USB インタフェース IC とのバス幅を 16 ビットとします。データ転送速度は上がりますが、P40～P47 ポートが使用できなくなります。
C++	TWB::OPEN_OPT::MODE_BUS16	
VB/VBA	TWB_OPEN_OPT.MODE_BUS16	
C#	TWB.OPEN_OPT.MODE_BUS16	
C/C++	TWB_MODE_FLASH	フラッシュ書換えモードのデバイスに接続します。
C++	TWB::OPEN_OPT::MODE_FLASH	
VB/VBA	TWB_OPEN_OPT.MODE_FLASH	
C#	TWB.OPEN_OPT.MODE_FLASH	
C/C++	TWB_MODE_LEGACY	TWB_Initialize()の説明(126 ページ)で示す「動作設定に関するオプション」を全てオフにして接続します。
C++	TWB::OPEN_OPT::MODE_LEGACY	
VB/VBA	TWB_OPEN_OPT.MODE_LEGACY	
C#	TWB.OPEN_OPT.MODE_LEGACY	
C/C++	TWB_LIST_UPDATE	LAN デバイス専用。ローカルネットワーク内の LAN デバイスを検索し、発見したデバイスをライブラリの内部テーブルに記録します。
C++	TWB::OPEN_OPT::LIST_UPDATE	
VB/VBA	TWB_OPEN_OPT.LIST_UPDATE	
C#	TWB.OPEN_OPT.LIST_UPDATE	

LAN デバイスに接続する場合で、プログラムを起動して最初に *TWB_Open()* 関数を呼び出す場合は *Opt* 引数に *TWB_LIST_UPDATE* (相当)の検索オプションが必須です。

TWB ライブラリは内部に LAN デバイスの IP アドレスを記録するためのテーブルを持っています。デバイスに接続する際にはこのテーブルから製品を検索し接続を行います。

プログラムが起動した直後はこのテーブルが構築されていないため、デバイスを探索するためのパケットをブロードキャストし、テーブルを構築する必要があります。また、ネットワーク内のデバイス構成が変わった場合にもテーブルを再構築する必要があります。

デバイスの操作を終了する

TWB_Close() 関数を呼び出します。クローズしたハンドルは無効になります。

TWB_CloseAll() による切断

デバイスのハンドルはプロセスが終了した時点で全て解放されます。多くの開発環境ではデバッグを途中で停止すると開発中のプログラムのプロセスが終了しハンドルが解放されます。この場合、デバッグ中のプログラムに接続されていたデバイスは再度接続可能な状態に戻ります。

しかし、Microsoft® Office などの一部の開発環境では開発中のプログラムが 1 つのプロセスの中で実行されるケースがあります。このような場合、プログラムのデバッグを途中で停止してもハンドルを所有していたプロセスは終了しないため、デバイスは切断されたことを認識することができません。そのため再度デバイスに接続しようとしてもデバイスは使用中とみなされ接続できない状態となります。

このような場合はプログラムの開始位置で *TWB_CloseAll()* 関数を使用すると、プロセスが接続していたデバイスが一旦全て解放されるため、デバッグを途中で停止しても再度接続することが可能になります。

リスト 2　接続／切断の例(C 言語)

```
TW_HANDLE hDev;

//接続時のポート番号を指定
//TWB_SetNetworkPort(50000);

//接続時のパスワードを指定
//TWB_SetPassword("password");

//指定 UUID 製品の装置番号 1 に接続
TWB_Open(&hDev, "c45f3213-7c3e-434f-93ba-b9ca458b5bd2", 1, TWB_IF_ANY | TWB_LIST_UPDATE);
if (hDev)
{
  //...制御の中身

  TWB_Close(hDev);  //操作を終了したらハンドルを閉じる
}
```

リスト 3　接続／切断の例(Visual Basic)

```
Dim hDev As System.IntPtr

'接続時のポート番号を指定
'TWB_SetNetworkPort(50000)

'接続時のパスワードを指定
'TWB_SetPassword("password")

'指定 UUID 製品の装置番号 1 番のデバイスに接続
TWB_Open(hDev, "c45f3213-7c3e-434f-93ba-b9ca458b5bd2", 1, _
         TWB_OPEN_OPT.IF_ANY Or TWB_OPEN_OPT.LIST_UPDATE)

If hDev <> System.IntPtr.Zero Then

  '...制御の中身

  TWB_Close(hDev)
End If
```

リスト 4　接続／切断の例(VBA)

```
Dim hDev As Long

'接続時のポート番号を指定
'TWB_SetNetworkPort(50000)

'接続時のパスワードを指定
'TWB_SetPassword("password")

'指定 UUID 製品の装置番号 1 番のデバイスに接続
TWB_Open hDev, "c45f3213-7c3e-434f-93ba-b9ca458b5bd2", 1, _
         TWB_OPEN_OPT.IF_ANY Or TWB_OPEN_OPT.LIST_UPDATE

If hDev <> 0 Then

  '...制御の中身

  TWB_Close hDev
End If
```

リスト 5　接続／切断の例(C#)

```
System.IntPtr hDev;

//接続時のポート番号を指定
//TWB.SetNetworkPort(50000);

//接続時のパスワードを指定
//TWB.SetPassword("password");

//指定 UUID 製品の装置番号 1 に接続
TWB.Open(out hDev, "c45f3213-7c3e-434f-93ba-b9ca458b5bd2", 1,
         TWB.OPEN_OPT.IF_ANY | TWB.OPEN_OPT.LIST_UPDATE);

if (hDev != System.IntPtr.Zero)
{
  //...制御の中身

  TWB.Close(hDev); //操作を終了したらハンドルを閉じる
}
```

アドレスやポート番号を指定してデバイスをオープンする(LAN デバイス)

ルーターなどを介して異なるネットワークにあるデバイスと接続する場合には、IP アドレスやドメイン名でデバイスを指定する必要があります。ポート番号は通常指定する必要はありませんが、設定ツール[LANMConfig]で変更した場合や、ルーターの設定でポート番号の指定が必要な場合があります。

アドレスまたはドメイン、ポート番号を指定してデバイスに接続するには *TWB_OpenByAddress()* 関数を使用します。

リスト 6　アドレスとポート番号を指定して接続する例(C 言語)

```
TW_HANDLE hDev;

//接続時のポート番号を指定
//TWB_SetNetworkPort(50000);

//接続時のパスワードを指定
//TWB_SetPassword("password");

//IP アドレス=192.168.0.50, ポート番号=50000 のデバイスに接続
TWB_OpenByAddress(&hDev, "192.168.0.50:50000", TWB_IF_LAN);
```

リスト 7　アドレスとポート番号を指定して接続する例(Visual Basic)

```
Dim hDev As System.IntPtr

'接続時のポート番号を指定
'TWB_SetNetworkPort(50000)

'接続時のパスワードを指定
'TWB_SetPassword("password")

'IP アドレス=192.168.0.50, ポート番号=50000 のデバイスに接続
TWB_OpenByAddress(hDev, "192.168.0.50:50000", TWB_OPEN_OPT.IF_LAN)
```

リスト 8　アドレスとポート番号を指定して接続する例(VBA)

```
Dim hDev As Long

'接続時のポート番号を指定
'TWB_SetNetworkPort(50000)

'接続時のパスワードを指定
'TWB_SetPassword("password")

'IP アドレス=192.168.0.50, ポート番号=50000 のデバイスに接続
TWB_OpenByAddress hDev, "192.168.0.50:50000", TWB_OPEN_OPT.IF_ANY
```

リスト 9　アドレスとポート番号を指定して接続する例(C#)

```
System.IntPtr hDev;

//接続時のポート番号を指定
//TWB.SetNetworkPort(50000);

//接続時のパスワードを指定
//TWB.SetPassword("password");

//IP アドレス=192.168.0.50, ポート番号=50000 のデバイスに接続
TWB.OpenByAddress(out hDev, "192.168.0.50:50000", TWB.OPEN_OPT.IF_LAN);
```

USB シリアルを指定して接続

TWB_OpenByAddress() 関数を使用すると、USB デバイスの USB シリアルという固有の番号を指定して接続することも可能です。

USB シリアルは USB デバイスの個体識別のためのユニークな番号です。デバイスの USB シリアルを調べるには「USBMTools」の「ReadID」を使用します。

クライアントモードに設定したデバイスと接続する(LAN デバイス)

デバイスをクライアントモードに設定すると、デバイス側からサーバーとなるパソコンに対してネットワーク接続を行います。クライアントモードを利用すると、インターネットなどを通じて複数のデバイスを制御したい場合に、パソコン側のポートだけを外部から接続可能な状態にすれば良いのでルーターなどの設定が容易になります。

デバイスをクライアントモードに設定する方法については製品のユーザーズマニュアル、または、設定ツール「LANMConfig」のオンラインヘルプを参照してください。

クライアントモードのデバイスと接続するためには、まず *TWB_Listen()* 関数を呼び出して、ネットワーク接続を受け入れるポートを準備します。処理が成功すると *TWB_Listen()* 関数はソケットと呼ばれる一種の識別子を返します。

次に、取得したソケットを引数として *TWB_Accept()* 関数を呼び出します。*TWB_Accept()* 関数は接続要求を行っているデバイスがあれば、そのデバイスと接続して制御用のハンドルを返します。接続が完了しても、最初に *TWB_Listen()* 関数で準備したポートとソケットは引き続き有効ですので、再度 *TWB_Accept()* 関数に渡して他のデバイスの接続要求を受け入れることができます。

TWB_Accept() 関数は接続要求が無ければ、すぐに終了し *TW_DEVICE_NOT_FOUND* を返しますので、定期的に呼び出してデバイスからの接続要求の有無をチェックするようにします。

表 18　TWB_Listen() の関数宣言

言語	関数宣言
C/C++	TW_STATUS TWB_Listen(UINT_PTR *pListenSocket, LPCTSTR pLocalIP, DWORD PortNumber)
VB	Function TWB_Listen(ByRef pListenSocket As System.IntPtr, ByVal pLocalIP As String, ByVal PortNumber As Integer) As Integer
VBA	Function TWB_Listen(ByRef pListenSocket As Long, ByVal pLocalIP As String, ByVal PortNumber As Long) As Long
C#	STATUS Listen(out System.IntPtr pListenSocket, string pLocalIP, int PortNumber)

表 19　TWB_Accept() の関数宣言

言語	関数宣言
C/C++	TW_STATUS TWB_Accept(UINT_PTR ListenSocket, TW_HANDLE *phDev, long Opt)
VB	Function TWB_Accept(ByVal ListenSocket As System.IntPtr, ByRef phDev As System.IntPtr, ByVal Opt As TWB_ACCEPT_OPT) As Integer
VBA	Function TWB_Accept(ByVal ListenSocket As Long, ByRef phDev As Long, ByVal Opt As TWB_ACCEPT_OPT) As Long
C#	STATUS Accept(System.IntPtr ListenSocket, out System.IntPtr phDev, ACCEPT_OPT Opt) STATUS Accept(System.IntPtr ListenSocket, out System.IntPtr phDev)

表 20　クライアントモードのデバイスと接続するサンプルプログラム

開発環境	プロジェクト名またはファイル名	説明
Visual C++ (MFC)	ClientModeSample	クライアントモードのデバイスと接続し、選択したデバイスのデジタルおよびアナログ入出力を制御します。
Visual Basic	ClientModeSampleVB	
Visual C#	ClientModeSampleCS	

□ デジタル入出力

デバイスが使用できるデジタル入力端子、デジタル出力端子を表 21 に示します。入力端子／出力端子は最大 8 つの端子を 1 つのグループとして、グループ単位で読出し、書込みを行います。一部の端子は他の機能と兼用となっています。

表 21　入出力端子

端子名	端子数	方向	ポート名
P10～P17	8	入力	P1
P20～P27	8	入力	P2
P40～P47	8	入出力	P4
P50～P53[7]	4	入力	P5
PA0～PA7	8	入出力	PA
POUT0#～POUT7#	8	出力	POUT

入力端子、出力端子は、それぞれ、入力ポート、出力ポートというハードウェアを通じて制御します。入力端子は入力ポートと、出力端子は出力ポートと 1 対 1 に接続されていますので、入力ポートからの読出しは入力端子の状態の読み取り、出力ポートへの書込みは出力端子の状態変更と等価です。入出力ポートの制御には、表 22 の関数を使用します。また、表 23 はデジタル入出力のサンプルとして用意されているプログラムです。

表 22　デジタル入出力で使用する関数

関数名	説明
TWB_PortWrite()	出力ポートへ書込みを行います。
TWB_PortRead()	入力ポートから読出しを行います。

表 23　デジタル入出力のサンプルプログラム

開発環境	プロジェクト名またはファイル名	説明
Visual C++ (MFC)	PortSample	入力端子の状態を表示し、出力端子状態を操作できます。
Visual Basic	PortSampleVB	
Visual C#	PortSampleCS	
VBA	PortSample1.xls	指定時刻に POUT0#～POUT7#の出力を操作する簡易プログラムタイマです。
	PortSample2.xls	入力ポートの状態が変化した日時を記録します。

[7] 『USBM3069-HS』/『USBM3069-HSL』では P53 のみ入力ポートとして使用可能です。

入力端子の状態を読み取る

TWB_PortRead() 関数で入力ポートからデータを読み出します。*Port* 引数(表 25)で読み出したいポートを指定してください。

表 24　TWB_PortRead() の関数宣言

言語	関数宣言
C/C++	TW_STATUS TWB_PortRead(TW_HANDLE hDev, DWORD Port, BYTE *pData)
VB	Function TWB_PortRead(ByVal hDev As System.IntPtr, ByVal Port As TWB_RPORT, ByRef pData As Byte) As Integer
VBA	Function TWB_PortRead(ByVal hDev As Long, ByVal Port As TWB_RPORT, ByRef pData As Byte) As Long
C#	STATUS PortRead(System.IntPtr hDev, RPORT Port, out byte pData)

表 25　TWB_PortRead()の Port 引数に指定する値

言語	値	説明
C/C++	TWB_P1	P10～P17 入力を読み取ります。
C++	TWB::RPORT::P1	
VB/VBA	TWB_RPORT.P1	
C#	TWB.RPORT.P1	
C/C++	TWB_P2	P20～P27 入力を読み取ります。
C++	TWB::RPORT::P2	
VB/VBA	TWB_RPORT.P2	
C#	TWB.RPORT.P2	
C/C++	TWB_P4	P40～P47 入力、または、出力値を読み取ります。
C++	TWB::RPORT::P4	
VB/VBA	TWB_RPORT.P4	
C#	TWB.RPORT.P4	
C/C++	TWB_P5	P50～P53 入力を読み取ります。
C++	TWB::RPORT::P5	
VB/VBA	TWB_RPORT.P5	
C#	TWB.RPORT.P5	
C/C++	TWB_PA	PA0～PA7 入力、または、出力値を読み取ります。
C++	TWB::RPORT::PA	
VB/VBA	TWB_RPORT.PA	
C#	TWB.RPORT.PA	
C/C++	TWB_POUT	POUT0#～POUT7#の出力値を読み取ります。
C++	TWB::RPORT::POUT	
VB/VBA	TWB_RPORT.POUT	
C#	TWB.RPORT.POUT	

読出しは 8 ビット単位で行い、結果は *pData* 引数に格納されます。例えば P1 ポートを読み出した場合、読み取ったデータの各ビットは下の表のように各端子の入力値と対応しています。

表 26　データビットと端子の関係

ビット	7(MSB)	6	5	4	3	2	1	0(LSB)
対応端子	P17	P16	P15	P14	P13	P12	P11	P10

対応する端子が"Lo"となっているビットは"0"に、"Hi"となっているビットは"1"として読み出されます。出力ポートから読出しを行った場合、現在の出力状態が読み出されます。

出力端子の状態を変更する

TWB_PortWrite() 関数で出力ポートに書込みを行うことで、出力端子の状態を変更できます。

表 27　TWB_PortWrite() の関数宣言

言語	関数宣言
C/C++	TW_STATUS TWB_PortWrite(TW_HANDLE hDev, DWORD Port, BYTE Data, BYTE Mask)
VB	Function TWB_PortWrite(ByVal hDev As System.IntPtr, ByVal Port As TWB_WPORT, ByVal Data As Byte, ByVal Mask As Byte) As Integer
VBA	Function TWB_PortWrite(ByVal hDev As Long, ByVal Port As TWB_WPORT, ByVal Data As Byte, ByVal Mask As Byte) As Long
C#	STATUS PortWrite(System.IntPtr hDev, WPORT Port, byte Data, byte Mask) STATUS PortWrite(System.IntPtr hDev, WPORT Port, byte Data)

表 28　TWB_PortWrite()の Port 引数に指定する値

言語	値	説明
C/C++	TWB_P4	P40～P47 の出力値を変更します。
C++	TWB::WPORT::P4	
VB/VBA	TWB_WPORT.P4	
C#	TWB.WPORT.P4	
C/C++	TWB_PA	PA0～PA7 の出力値を変更します。
C++	TWB::WPORT::PA	
VB/VBA	TWB_WPORT.PA	
C#	TWB.WPORT.PA	
C/C++	TWB_POUT	POUT0#～POUT7#の出力値を変更します。
C++	TWB::WPORT::POUT	
VB/VBA	TWB_WPORT.POUT	
C#	TWB.WPORT.POUT	

入力と同様に 8 ビット単位でデータを書き込みます。データビットと端子との関係は入力の場合と同様で、POUT0#～POUT7#に対する操作を除いて"0"を書き込んだビットと対応する端子は"Lo"となり、"1"と対応する端子は"Hi"になります。

POUT0#～POUT7#はオープンコレクタ出力です。POUT ポートに書込みを行った場合は"0"を書き込んだビットと対応する端子は"OFF"、"1"と対応する端子は"ON"になります。

TWB_PortWrite() 関数の引数 Mask により書込みの一部をマスクできます。Mask バイトのうち"0"のビットと対応する出力端子は影響を受けません。図 3 は H'55 というデータを、Mask を H'0F として出力した例です。

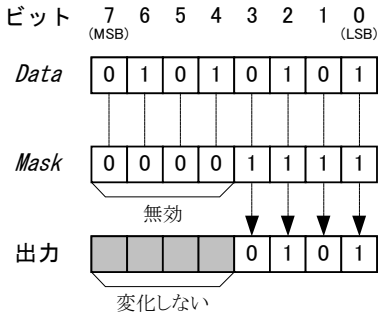


図 3　出力のマスク

リスト 10　デジタル入出力の例(C 言語)

```
BYTE bData;

//P10-P17 の読出し
TWB_PortRead(hDev, TWB_P1, &bData);

//POUT7#だけを"ON"にし、POUT6#-POUT0#は変更しない
TWB_PortWrite(hDev, TWB_POUT, 0xff, 0x80);
```

リスト 11　デジタル入出力の例(Visual Basic)

```
Dim bData As Byte

'P10-P17 の読出し
TWB_PortRead(hDev, TWB_RPORT.P1, bData)

'POUT7#だけを"ON"にし、POUT6#-POUT0#は変更しない
TWB_PortWrite(hDev, TWB_WPORT.POUT, &HFF, &H80)
```

リスト 12　デジタル入出力の例(VBA)

```
Dim bData As Byte

'P10-P17 の読出し
TWB_PortRead hDev, TWB_RPORT.P1, bData

'POUT7#だけを"ON"にし、POUT6#-POUT0#は変更しない
TWB_PortWrite hDev, TWB_WPORT.POUT, &HFF, &H80
```

リスト 13　デジタル入出力の例(C#)

```
byte bData;

//P10-P17 の読出し
TWB.PortRead(hDev, TWB.RPORT.P1, out bData);

//POUT7#だけを"ON"にし、POUT6#-POUT0#は変更しない
TWB.PortWrite(hDev, TWB.WPORT.POUT, 0xff, 0x80);
```

- 例ではデバイスへの接続やエラー処理が省略されています。接続方法については 15 ページを、エラー処理については 73 ページを参照してください。以降のページで示す例も同様です。

入出力端子の方向を変更する

P40～P47、PA0～PA7 の各端子は初期状態では入力となっていますが、出力端子としても使用可能です。これらの端子の方向を切り替えるには、*TWB_PortSetDir()* 関数(表 29)を使用します。*Dir* 引数の各ビットと端子の関係は表 26 の場合と同様で、"1"としたビットと対応する端子は出力に、"0"としたビットと対応する端子は入力になります。

表 29　TWB_PortSetDir() の関数宣言

言語	関数宣言
C/C++	TW_STATUS TWB_PortSetDir(TW_HANDLE hDev, DWORD Port, BYTE Dir)
VB	Function TWB_PortSetDir(ByVal hDev As System.IntPtr, ByVal Port As TWB_RWPORT, ByVal Dir As Byte) As Integer
VBA	Function TWB_PortSetDir(ByVal hDev As Long, ByVal Port As TWB_RWPORT, ByVal Dir As Byte) As Long
C#	STATUS PortSetDir(System.IntPtr hDev, RWPORT Port, byte Dir)

表 30　TWB_PortSetDir()の Port 引数に指定する値

言語	値	説明
C/C++	TWB_P4	P40～P47 の入出力方向を変更します。
C++	TWB::RWPORT::P4	
VB/VBA	TWB_RWPORT.P4	
C#	TWB.RWPORT.P4	
C/C++	TWB_PA	PA0～PA7 の入出力方向を変更します。
C++	TWB::RWPORT::PA	
VB/VBA	TWB_RWPORT.PA	
C#	TWB.PWPORT.PA	

□ アナログ入出力

製品はアナログ入力用に AD0～AD3、アナログ出力用に DA0～DA1 端子を備えています。

表 31 はアナログ入出力を制御するための関数です。表 32 はアナログ入出力のサンプルプログラムです。

表 31　アナログ入出力で使用する関数

関数名	説明
TWB_ADRead()	アナログ入力から変換結果を読み出します。
TWB_PortWrite()	アナログ出力値を設定します。
TWB_An16ToVolt()	アナログ入力の取得値を電圧値(ボルト単位)に変換します。
TWB_An8FromVolt()	電圧値(ボルト単位)から DA コンバータに書き込む値を計算します。

表 32　アナログ入出力のサンプルプログラム

開発環境	プロジェクト名またはファイル名	説明
Visual C++ (MFC)	AnalogSample	各アナログ入力端子の入力電圧を表示し、アナログ出力電圧を変更できるプログラムです。
Visual Basic	AnalogSampleVB	
Visual C#	AnalogSmapleCS	
VBA	AnalogSample.xls	簡易データロガーです。各アナログ入力端子の入力電圧を定期的に記録します。

アナログ入力値を読み取る

アナログ入力端子の AD 変換結果を読み出すには *TWB_ADRead()* 関数を使用します。

表 33　TWB_ADRead() の関数宣言

言語	関数宣言
C/C++	TW_STATUS TWB_ADRead(TW_HANDLE hDev, long Ch, long *pData)
VB	Function TWB_ADRead(ByVal hDev As System.IntPtr, ByVal Ch As Integer, ByRef pData As Integer) As Integer
VBA	Function TWB_ADRead(ByVal hDev As Long, ByVal Ch As Long, ByRef pData As Long) As Long
C#	STATUS ADRead(System.IntPtr hDev, int Ch, out int pData)

AD 変換結果は引数 *pData* に図 4 のように格納されます。入力電圧値と読み出される値の関係は表 34 のようになります。*pData* の値は *TWB_An16ToVolt()* 関数を使用して電圧値に変換することが可能です。

ビット	31-16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
値	常に 0	AD 変換結果										常に 0					

図 4　AD 変換結果の格納

表 34　アナログ入力電圧と変換結果の関係

入力電圧値([V])	読み出される値
5-LSB	65472
2.5	32768
0	0

・LSB = 5 / 1024 [V]
・表は理論値を示しています。

リスト 14　アナログ入力の例(C 言語)

```
long LData;
double dVolt;

//AD0 の AD 変換結果を読出し
TWB_ADRead(hDev, 0, &LData);

//取得値を電圧値に変換
dVolt = TWB_An16ToVolt(LData, 0);
```

リスト 15　アナログ入力の例(Visual Basic)

```
Dim iData As Integer
Dim dVolt As Double

'AD0 の AD 変換結果を読出し
TWB_ADRead(hDev, 0, iData)

'取得値を電圧値に変換
dVolt = TWB_An16ToVolt(iData)
```

リスト 16　アナログ入力の例(VBA)

```
Dim iData As Long
Dim dVolt As Double

'AD0 の AD 変換結果を読出し
TWB_ADRead hDev, 0, iData

'取得値を電圧値に変換
dVolt = TWB_An16ToVolt(iData)
```

リスト 17　アナログ入力の例(C#)

```
int iData;
double dVolt;

//AD0 の AD 変換結果を読出し
TWB.ADRead(hDev, 0, out iData);

//取得値を電圧値に変換
dVolt = TWB.An16ToVolt(iData);
```

アナログ出力値を変更する

アナログ出力端子の出力電圧を変更するには、デジタル出力の場合と同じ *TWB_PortWrite()* 関数(25 ページ、表 27)を使用します。*Port* 引数には DA のチャンネルを示す定数(表 35)を指定します。*Data* 引数には DA コンバータへの設定値を入力します。DA 設定値と出力電圧の関係を表 36 に示します。

TWB_An8FromVolt() 関数を使用すると、電圧値から DA コンバータへの設定値を計算することができます。

表 35　DA コンバータを示す定数値

言語	値	説明
C/C++	TWB_DA0	DA0 出力を変更します。
C++	TWB::WPORT::DA0	
VB/VBA	TWB_WPORT.DA0	
C#	TWB.WPORT.DA0	
C/C++	TWB_DA1	DA1 出力を変更します。
C++	TWB::WPORT::DA1	
VB/VBA	TWB_WPORT.DA1	
C#	TWB.WPORT.DA1	

表 36　DA 設定値とアナログ出力電圧の関係

DA 設定値	出力電圧([V])
255	5-LSB
128	2.5
0	0

・LSB = 5 / 256 [V]
・表は理論値を示しています。

リスト 18　アナログ出力の例(C 言語)

```
double dVolt;

//DA0 出力を約 3.5V に設定
dVolt = 3.5;
TWB_PortWrite(hDev, TWB_DA0, TWB_An8FromVolt(&dVolt, 0), 0xff);
```

リスト 19　アナログ出力の例(Visual Basic)

```
Dim dVolt As Double

'DA0 出力を約 3.5V に設定
dVolt = 3.5
TWB_PortWrite(hDev, TWB_WPORT.DA0, TWB_An8FromVolt(dVolt))
```

リスト 20　アナログ出力の例(VBA)

```
Dim dVolt As Double

'DAO 出力を約 3.5V に設定
dVolt = 3.5
TWB_PortWrite hDev, TWB_WPORT.DAO, TWB_An8FromVolt(dVolt)
```

リスト 21　アナログ出力の例(C#)

```
double dVolt;

//DAO 出力を約 3.5V に設定
dVolt = 3.5;
TWB.PortWrite(hDev, TWB.WPORT.DAO, TWB.An8FromVolt(ref dVolt));
```

□ パルスをカウントする

製品ではハードウェアカウンタとソフトウェアカウンタの 2 種類の方法でパルスをカウントすることができます。

ハードウェアカウンタはマイコンの 16 ビットタイマというハードウェア機能を利用したもので､名前の通り 16 ビットのカウンタレジスタで入力パルスをカウントすることができます。単相カウント、2 相カウントのどちらにも利用でき、単相パルスカウントの場合は最大 2 チャンネル、2 相パルスカウントを行う場合は 1 チャンネル利用できます。ハードウェアを利用するため高速なパルス信号に対応できる特徴があります。

ソフトウェアカウンタは外部割り込みを利用したカウンタ機能で、割り込み発生回数を 32 ビットのカウンタ変数に記録するものです。単相カウント、2 相カウントのどちらにも利用でき、単相パルスカウントの場合は最大 4 チャンネル、2 相パルスカウントの場合は最大 2 チャンネル利用可能です。また､Z 相信号でカウンタをクリアする 3 相動作も設定可能です。本マニュアルでパルスカウンタと表記した場合は､ソフトウェアカウンタのことを指します。

表 37　ハードウェアカウンタとパルスカウンタ(ソフトウェアカウンタ)の特徴

カウンタ種類	チャンネル数（最大）		カウンタビット数	特徴	備考
	単相	2 相			
ハードウェアカウンタ	2	1	16	高速、2 相カウント時の分解能が高い	マイコンのハードウェア機能(16 ビットタイマ)を利用
パルスカウンタ	4	2	32	オーバーフローしにくい	マイコンの外部割り込みをソフトウェアでカウント

表 38　パルスカウントのサンプルプログラム

開発環境	プロジェクト名またはファイル名	説明
Visual C++ (MFC)	PulseCountSample	ハードウェアカウンタとパルスカウンタのサンプルです。各カウンタの設定とカウント値の表示を行います。
Visual Basic	PulseCountSampleVB	
Visual C#	PulseCountSampleCS	
VBA	PulseCountSample.xls	簡易データロガーです。定期的に各カウンタの値と、前回の値との差分値を記録します。

- PWM 出力も 16 ビットタイマの機能を使用します。ハードウェアカウンタとして使用しているチャンネルは PWM 出力を使用できません。

ハードウェアカウンタによる単相パルスカウント

ハードウェアカウンタによる単相カウントの場合、TCLKA 入力を 16 ビットタイマのチャンネル 1 で、TCLKB 入力をチャンネル 2 でそれぞれカウントします。カウントエッジは立上り、立下り、または、両エッジから選択可能です。

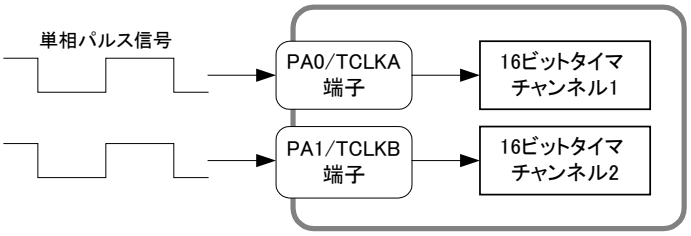


図 5　ハードウェアカウンタによる単相パルスカウント

ハードウェアカウンタによる 2 相パルスカウント

ハードウェアカウンタにより 90°位相差 2 相パルスをカウントする場合、16 ビットタイマのチャンネル 2 を使用します。

接続は TCLKA に B 相信号を TCLKB に A 相信号を入力します。インクリメンタル方式のロータリーエンコーダをこのように接続すると CW 回転でカウンタが増加、CCW 回転でカウンタが減少します。また、1 回転あたりのカウント数はロータリーエンコーダの出力パルス数の 4 倍となります(図 7)。

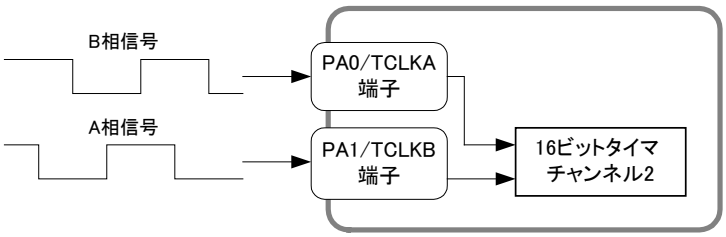


図 6　ハードウェアカウンタによる 2 相パルスカウント

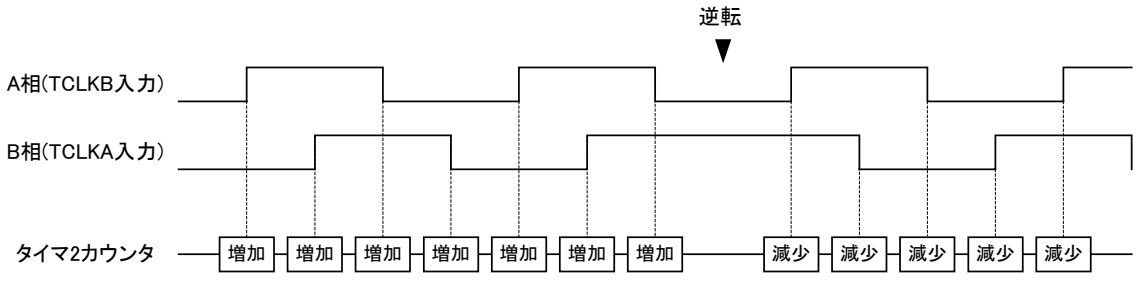


図 7　2 相パルス入力とハードウェアカウンタの増減

TCLKA 端子を B 相入力に使用するため、チャンネル 1 による単相カウントは実質的に使用できなくなります。

ハードウェアカウンタの使用方法

ハードウェアカウンタを使用するには、まず *TWB_TimerSetMode()* 関数を呼び出し、*Mode* 引数(表 41 参照)によって使用するチャンネルのカウントモードを設定します。

TWB_TimerStart() 関数でカウントを開始し、*TWB_TimerReadCnt()* 関数でカウント値を読み出します。

表 39　ハードウェアカウンタで使用する関数

関数名	説明
TWB_TimerSetMode()	カウントモードを設定します。
TWB_TimerStart()	カウントを開始します。
TWB_TimerStop()	カウントを停止します。
TWB_TimerReadCnt()	カウンタ値を読み出します。
TWB_TimerSetCnt()	カウンタ値をセットします。主にカウンタクリアに使用します。

表 40　TWB_TimerSetMode() の関数宣言

言語	関数宣言
C/C++	TW_STATUS TWB_TimerSetMode(TW_HANDLE hDev, long Ch, long Mode)
VB	Function TWB_TimerSetMode(ByVal hDev As System.IntPtr, ByVal Ch As Integer, ByVal Mode As TWB_TIMER_MODE) As Integer
VBA	Function TWB_TimerSetMode(ByVal hDev As Long, ByVal Ch As Long, ByVal Mode As TWB_TIMER_MODE) As Long
C#	STATUS TimerSetMode(System.IntPtr hDev, int Ch, TIMER_MODE Mode);

表 41　ハードウェアカウンタ使用時に Mode 引数に指定する値

言語	値	説明
C/C++	TWB_TIMER_RISE	指定チャンネルをパルスカウントモードとし、対応する入力が"Lo"から"Hi"に変化したときカウントします。1、2 チャンネルで指定可能です。
C++	TWB::TIMER_MODE::COUNT_RISE	
VB/VBA	TWB_TIMER_MODE.COUNT_RISE	
C#	TWB.TIMER_MODE.COUNT_RISE	
C/C++	TWB_TIMER_FALL	指定チャンネルをパルスカウントモードとし、対応する入力が"Hi"から"Lo"に変化したときカウントします。1、2 チャンネルで指定可能です。
C++	TWB::TIMER_MODE::COUNT_FALL	
VB/VBA	TWB_TIMER_MODE.COUNT_FALL	
C#	TWB.TIMER_MODE.COUNT_FALL	
C/C++	TWB_TIMER_BOTH	指定チャンネルをパルスカウントモードとし、極性によらず対応する入力が変化したときにカウントします。1、2 チャンネルで指定可能です。
C++	TWB::TIMER_MODE::COUNT_BOTH	
VB/VBA	TWB_TIMER_MODE.COUNT_BOTH	
C#	TWB.TIMER_MODE.COUNT_BOTH	
C/C++	TWB_TIMER_2PHASE	90° 位相差の A 相、B 相の 2 相信号をカウントします。2 チャンネルのみ指定可能です。
C++	TWB::TIMER_MODE::COUNT_2PHASE	
VB/VBA	TWB_TIMER_MODE.COUNT_2PHASE	
C#	TWB.TIMER_MODE.COUNT_2PHASE	

リスト 22　ハードウェアカウンタの使用例(C 言語)

```
WORD wCnt;

//チャンネル１で立上り時にカウント
TWB_TimerSetMode(hDev, 1, TWB_TIMER_RISE);

//タイマ１のカウントをスタート
TWB_TimerStart(hDev, TWB_TIMER_BIT1);

//タイマ１のカウンタ値を符号なし整数として読出し
TWB_TimerReadCnt(hDev, 1, (short*)&wCnt);
```

リスト 23　ハードウェアカウンタの使用例(Visual Basic)

```
Dim wCnt As System.UInt16

'チャンネル１で立上り時にカウント
TWB_TimerSetMode(hDev, 1, TWB_TIMER_MODE.COUNT_RISE)

'タイマ１のカウントをスタート
TWB_TimerStart(hDev, TWB_TIMER_BITS.TIMER1)

'タイマ１のカウンタ値を符号なし整数として読出し
TWB_TimerReadCnt(hDev, 1, wCnt)
```

リスト 24　ハードウェアカウンタの使用例(C#)

```
ushort wCnt;

//チャンネル１で立上り時にカウント
TWB.TimerSetMode(hDev, 1, TWB.TIMER_MODE.COUNT_RISE);

//タイマ１のカウントをスタート
TWB.TimerStart(hDev, TWB.TIMER_BITS.TIMER1);

//タイマ１のカウンタ値を符号なし整数として読出し
TWB.TimerReadCnt(hDev, 1, out wCnt);
```

パルスカウンタ(ソフトウェアカウンタ)による単相パルスカウント

パルスカウンタによる単相カウントの場合、PC0#～PC3#入力をそれぞれチャンネル 0～3 でカウントします。カウンタは入力が"Hi"→"Lo"に変化したタイミングでカウントアップします。

パルスカウンタ(ソフトウェアカウンタ)による 2 相パルスカウント

パルスカウンタにより 90° 位相差 2 相パルスをカウントする場合、チャンネル 0 と 1 の組み合わせ、または、チャンネル 2 と 3 の組み合わせでカウントします。後述する直接エンコーダ出力を接続する簡易な方法もありますが、外付けのインバータロジックと組み合わせて図 8 のように構成することを推奨します。図 9 に回路例を示します。

チャンネル 0 と 1 の組み合わせでカウントする場合には、PC0#に A 相の反転信号、PC1#に A 相信号を入力し、P26 端子に B 相信号を接続します。カウント値はチャンネル 0 と 1 の合計値になります。

チャンネル 2 と 3 の組み合わせでカウントする場合には、PC2#に A 相の反転信号、PC3#に A 相信号を入力し、P27 端子に B 相信号を接続します。カウント値はチャンネル 2 と 3 の合計値になります。

どちらの組み合わせを使用した場合も、インクリメンタル方式のロータリーエンコーダを接続した場合、1 回転あたりのカウント数は出力パルス数の 2 倍になります(図 10)。

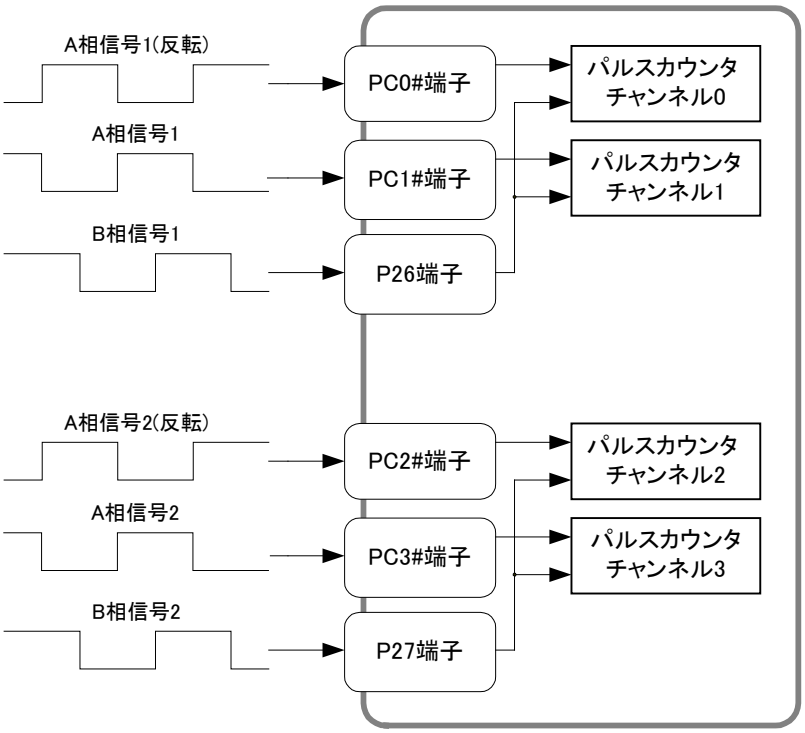


図 8 ソフトウェアカウンタによる 2 相パルスカウント(推奨接続)

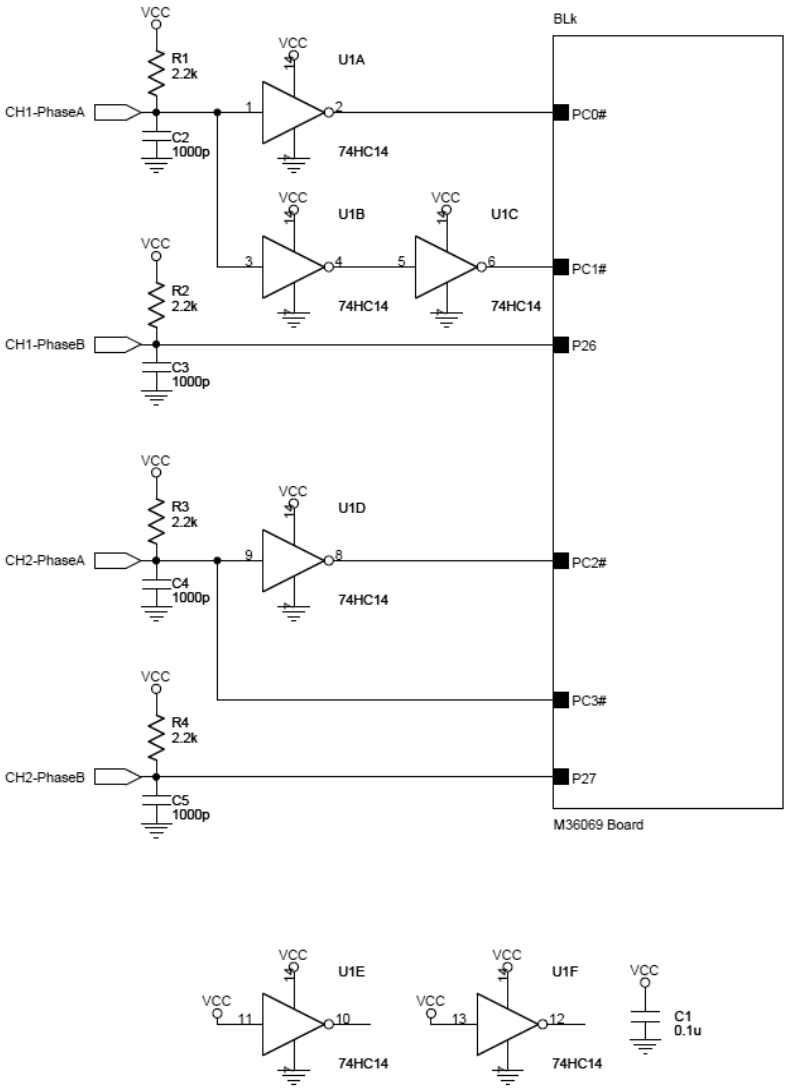


図 9　推奨接続の回路例

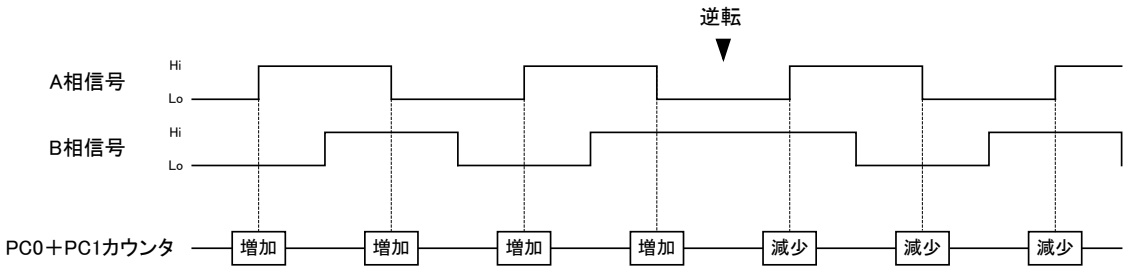


図 10　2 相パルス入力とパルスカウンタの増減(推奨接続)

パルスカウンタ(ソフトウェアカウンタ)による3相パルスカウント

基本的にはパルスカウンタ 2、3 チャンネルを使用した 2 相パルスカウントと同様の動作をしますが、パルスカウンタ 0 への入力で、パルスカウンタ 2、3 のカウンタ値がクリアされます。PC0#に Z 相信号を接続すると 1 回転毎にカウンタ値がクリアされ、パルスカウンタ 0 の値が増加します。

パルスカウンタ(ソフトウェアカウンタ)の使用方法

ソフトウェアカウンタを使用するには、まず *TWB_PCSetMode()* 関数(表 43)を呼び出し、使用するチャンネルの動作モードを設定します。*ChBits* 引数には表 44 に示すチャンネルを示す定数を指定します。*Mode* 引数には表 45 に示すカウントモードを指定します。

TWB_PCStart() 関数でカウントを開始します。カウント値の読出しには *TWB_PCReadCnt()* 関数(表 46)を使用します。1 チャンネルずつ読み出すこともできますが、*ChBits* 引数に *TWB_PC_ALL*(相当の値)を指定すると 0～3 チャンネルまで全てのカウンタ値を読み出すことができます。その場合は、*pCnt* 引数として 4 チャンネル分(16 バイト)の領域を確保するようにしてください。

表 42　ソフトウェアカウンタで使用する関数

関数名	説明
TWB_PCSetMode()	カウントモードを設定します。
TWB_PCStart()	カウントを開始します。
TWB_PCStop()	カウントを停止します。
TWB_PCReadCnt()	カウンタ値を読み出します。
TWB_PCSetCnt()	カウンタ値をセットします。主にカウンタクリアに使用します。

表 43　TWB_PCSetMode() の関数宣言

言語	関数宣言
C/C++	TW_STATUS TWB_PCSetMode(TW_HANDLE hDev, long ChBits, long Mode)
VB	Function TWB_PCSetMode(ByVal hDev As System.IntPtr, ByVal ChBits As TWB_PC_BITS, ByVal Mode As TWB_PC_MODE) As Integer
VBA	Function TWB_PCSetMode(ByVal hDev As Long, ByVal ChBits As TWB_PC_BITS, ByVal Mode As TWB_PC_MODE) As Long
C#	STATUS PCSetMode(System.IntPtr hDev, PC_BITS ChBits, PC_MODE Mode);

表 44　ソフトウェアカウンタ操作関数の ChBits 引数に指定する値

言語	値	説明
C/C++	TWB_PC0	パルスカウンタ 0 の設定や読出しなどで指定します。
C++	TWB::PC_BITS::PC0	
VB/VBA	TWB_PC_BITS.PC0	
C#	TWB.PC_BITS.PC0	
C/C++	TWB_PC1	パルスカウンタ 1 の設定や読出しなどで指定します。
C++	TWB::PC_BITS::PC1	
VB/VBA	TWB_PC_BITS.PC1	
C#	TWB.PC_BITS.PC1	
C/C++	TWB_PC2	パルスカウンタ 2 の設定や読出しなどで指定します。
C++	TWB::PC_BITS::PC2	
VB/VBA	TWB_PC_BITS.PC2	
C#	TWB.PC_BITS.PC2	
C/C++	TWB_PC3	パルスカウンタ 3 の設定や読出しなどで指定します。
C++	TWB::PC_BITS::PC3	
VB/VBA	TWB_PC_BITS.PC3	
C#	TWB.PC_BITS.PC3	
C/C++	TWB_PC0_PC1	パルスカウンタ 0 と 1 の設定や読出しなどで指定します。読出しの場合はカウンタ 0 と 1 の合計値が返ります。
C++	TWB::PC_BITS::PC0_PC1	
VB/VBA	TWB_PC_BITS.PC0_PC1	
C#	TWB.PC_BITS.PC0_PC1	
C/C++	TWB_PC2_PC3	パルスカウンタ 2 と 3 の設定や読出しなどで指定します。読出しの場合はカウンタ 2 と 3 の合計値が返ります。
C++	TWB::PC_BITS::PC2_PC3	
VB/VBA	TWB_PC_BITS.PC2_PC3	
C#	TWB.PC_BITS.PC2_PC3	
C/C++	TWB_PC_ALL	全てのチャンネルを同じ動作設定にする場合や、全てのカウンタ値を読み出す場合に指定します。
C++	TWB::PC_BITS::PC_ALL	
VB/VBA	TWB_PC_BITS.PC_ALL	
C#	TWB.PC_BITS.PC_ALL	

表 45　TWB_PCSetMode() の Mode 引数に指定する値

言語	値	説明
C/C++	TWB_PC_SINGLE	単相カウントを行う場合に指定します。入力の立下りでカウントします。
C++	TWB::PC_MODE::COUNT_SINGLE	
VB/VBA	TWB_PC_MODE.COUNT_SINGLE	
C#	TWB.PC_MODE.COUNT_SINGLE	
C/C++	TWB_PC_2PHASE_E	90° 位相差の A 相、B 相の 2 相信号をカウントするモードです。図 8 に従って信号を入力する必要があります。
C++	TWB::PC_MODE::COUNT_2PHASE_E	
VB/VBA	TWB_PC_MODE.COUNT_2PHASE_E	
C#	TWB.PC_MODE.COUNT_2PHASE_E	
C/C++	TWB_PC_3PHASE_E	PC2#と PC3#で 2 相信号のカウントを行い、PC0#でカウンタをクリアします。PC2#、PC3#は図 8 に従って信号を入力する必要があります。ChBits の値は無視されます。
C++	TWB::PC_MODE::COUNT_3PHASE_E	
VB/VBA	TWB_PC_MODE.COUNT_3PHASE_E	
C#	TWB.PC_MODE.COUNT_3PHASE_E	
C/C++	TWB_PC_2PHASE	90° 位相差の A 相、B 相の 2 相信号をカウントするモードです。簡易接続(図 11)の場合に指定します。
C++	TWB::PC_MODE::COUNT_2PHASE	
VB/VBA	TWB_PC_MODE.COUNT_2PHASE	
C#	TWB.PC_MODE.COUNT_2PHASE	
C/C++	TWB_PC_3PHASE	PC2#と PC3#で 2 相信号のカウントを行い、PC0#でカウンタをクリアします。PC2#と PC3#を簡易接続(図 11)とした場合に指定します。ChBits の値は無視されます。
C++	TWB::PC_MODE::COUNT_3PHASE	
VB/VBA	TWB_PC_MODE.COUNT_3PHASE	
C#	TWB.PC_MODE.COUNT_3PHASE	

表 46　TWB_PCReadCnt() の関数宣言

言語	関数宣言
C/C++	TW_STATUS TWB_PCReadCnt(TW_HANDLE hDev, long ChBits, long *pCnt)
VB	Function TWB_PCReadCnt(ByVal hDev As System.IntPtr, ByVal ChBits As TWB_PC_BITS, ByRef pCnt As Integer) As Integer Function TWB_PCReadCnt(ByVal hDev As System.IntPtr, ByVal ChBits As TWB_PC_BITS, ByVal pCnt() As Integer) As Integer
VBA	Function TWB_PCReadCnt(ByVal hDev As Long, ByVal ChBits As TWB_PC_BITS, ByRef pCnt As Long) As Long
C#	STATUS PCReadCnt(System.IntPtr hDev, PC_BITS ChBits, out int pCnt) STATUS PCReadCnt(System.IntPtr hDev, PC_BITS ChBits, out uint pCnt) STATUS PCReadCnt(System.IntPtr hDev, PC_BITS ChBits, int []pCnt) STATUS PCReadCnt(System.IntPtr hDev, PC_BITS ChBits, uint []pCnt)

リスト 25　ソフトウェアカウンタによるパルスカウントの例(C 言語)

```
long LCnt[4];
long L2Phase;

//パルスカウンタ 0,1 を単相カウントに設定
TWB_PCSetMode(hDev, TWB_PC0_PC1, TWB_PC_SINGLE);

//パルスカウンタ 2,3 を 2 相カウントに設定
TWB_PCSetMode(hDev, TWB_PC2_PC3, TWB_PC_2PHASE_E);

//全てのチャンネルのカウントを開始
TWB_PCStart(hDev, TWB_PC_ALL);

//全てのチャンネルのカウンタ値を読出し
TWB_PCReadCnt(hDev, TWB_PC_ALL, LCnt);

//2 相カウントの結果は 2 チャンネルのカウンタ値の和
L2Phase = LCnt[2] + LCnt[3];
```

リスト 26　ソフトウェアカウンタによるパルスカウントの例(Visual Basic)

```
Dim iCnt(3) As Integer
Dim i2Phase As Integer

'パルスカウンタ 0,1 を単相カウントに設定
TWB_PCSetMode(hDev, TWB_PC_BITS.PC0_PC1, TWB_PC_MODE.COUNT_SINGLE)

'パルスカウンタ 2,3 を 2 相カウントに設定
TWB_PCSetMode(hDev, TWB_PC_BITS.PC2_PC3, TWB_PC_MODE.COUNT_2PHASE_E)

'全てのチャンネルのカウントを開始
TWB_PCStart(hDev, TWB_PC_BITS.PC_ALL)

'全てのチャンネルのカウンタ値を読出し
TWB_PCReadCnt(hDev, TWB_PC_BITS.PC_ALL, iCnt)

'2 相カウントの結果は 2 チャンネルのカウンタ値の和
i2Phase = iCnt(2) + iCnt(3)
```

リスト 27　ソフトウェアカウンタによるパルスカウントの例(C#)

```
int []iCnt = new int[4];
int i2Phase;

//パルスカウンタ 0,1 を単相カウントに設定
TWB.PCSetMode(hDev, TWB.PC_BITS.PC0_PC1, TWB.PC_MODE.COUNT_SINGLE);

//パルスカウンタ 2,3 を 2 相カウントに設定
TWB.PCSetMode(hDev, TWB.PC_BITS.PC2_PC3, TWB.PC_MODE.COUNT_2PHASE_E);

//全てのチャンネルのカウントを開始
TWB.PCStart(hDev, TWB.PC_BITS.PC_ALL);

//全てのチャンネルのカウンタ値を読出し
TWB.PCReadCnt(hDev, TWB.PC_BITS.PC_ALL, iCnt);

//2 相カウントの結果は 2 チャンネルのカウンタ値の和
i2Phase = iCnt[2] + iCnt[3];
```

簡易な接続での 2 相パルスカウント

外部インバータ回路を使用しない簡易な接続例を以下に示します。この場合も推奨接続の場合と同様、使用する 2 つのチャンネルの合計がカウント値となり、エンコーダの出力パルス数の 2 倍がカウントされます。

ただし、信号の立上りをカウントできないため、ロータリーエンコーダを接続した場合に「カウントが等間隔で発生しない」、「カウント発生位置で正転、逆転を繰り返すと片方向にカウンタが進んでしまう」といった欠点があります。

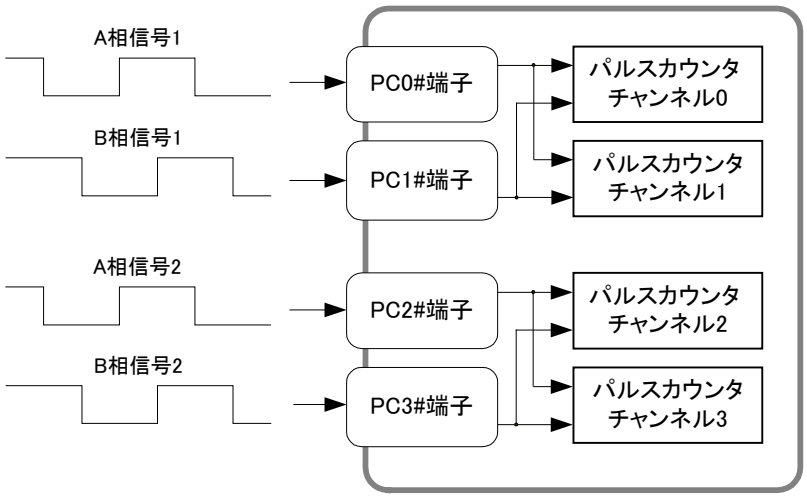


図 11 ソフトウェアカウンタによる 2 相パルスカウント(簡易接続)

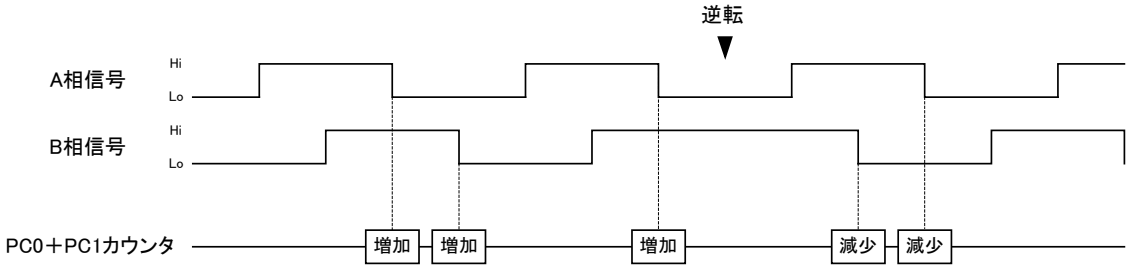


図 12　2 相パルス入力とパルスカウンタの増減(簡易接続)

□ PWM 出力

製品では最大 3 チャンネルの PWM 出力が可能です。チャンネル 0～2 で生成された PWM 信号はそれぞれ TIOCA0～TIOCA2 端子から出力されます。通常は、PWM のパルスタイミングは 25MHz の内部クロックをプリスケーラ[8]と 16 ビットタイマという内蔵カウンタで分周することで生成されます。出力できる周波数範囲は約 48Hz～１2.5MHz までの範囲です。

PWM 信号生成のための基準クロックは、外部から TCLKA 端子に入力することも可能です。より周期の長い信号が必要な場合や、外部クロックとの同期が必要な場合に利用します。外部クロックとして別チャンネルの PWM 出力を利用することもできます(図 13)。

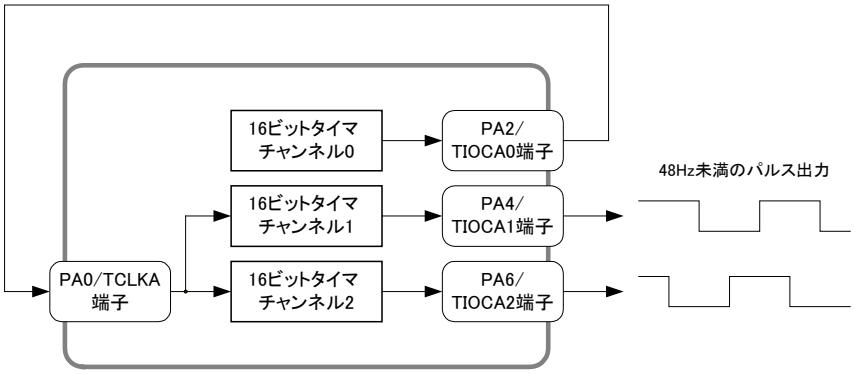


図 13　他チャンネルの出力を基準クロックとして利用

- ハードウェアカウンタも 16 ビットタイマの機能を使用します。PWM 出力に設定したチャンネルをハードウェアカウンタとして使用することはできません。

表 47　PWM 出力で使用する関数

関数名	説明
TWB_TimerSetMode()	16 ビットタイマを PWM モードへの設定、PWM モードの解除を行います。
TWB_TimerSetPwm()	出力パルスの周波数、デューティ、初期位相の設定を行います。
TWB_TimerSetPwmExt()	外部クロックを用いた場合のパルス設定を行います。
TWB_TimerStart()	パルス出力動作を開始します。
TWB_TimerStop()	パルス出力動作を停止します。
TWB_SetNumOfPulse()	出力パルス数を設定します。
TWB_ReadNumOfPulse()	残りの出力パルス数を読み出します。
TWB_TimerReadStatus()	パルス出力中かどうか調べます。
TWB_TimerSetLevel()	停止中に PWM 出力の状態を設定します。

表 48　PWM 出力のサンプルプログラム

開発環境	プロジェクト名またはファイル名	説明
Visual C++ (MFC)	PwmSample	3 チャンネルの周波数、デューティ、初期位相を設定し、指定のパルス数で出力を行います。
Visual Basic	PwmSampleVB	
Visual C#	PwmSampleCS	
VBA	PwmSample.xls	予めテーブルに入力した周波数とパルス数の設定を順次出力します。

[8] プリスケーラは 25MHz のクロックを 1/2、1/4、または、1/8 に分周します。

パルスの設定方法

内部クロックを使用したパルスの設定には *TWB_TimerSetPwm()* 関数(表 49)、外部クロックを使用したパルスの設定には *TWB_TimerSetPwmExt()* 関数(表 50)を使用します。

表 49　TWB_TimerSetPwm() の関数宣言

言語	関数宣言
C/C++	TW_STATUS TWB_TimerSetPwm(TW_HANDLE hDev, long Ch, double *pFrequency, double *pDuty, double *pPhase)
VB	Function TWB_TimerSetPwm(ByVal hDev As System.IntPtr, ByVal Ch As Integer, ByRef pFrequency As Double, ByRef pDuty As Double, ByRef pPhase As Double) As Integer
VBA	Function TWB_TimerSetPwm(ByVal hDev As Long, ByVal Ch As Long, ByRef pFrequency As Double, ByRef pDuty As Double, ByRef pPhase As Double) As Long
C#	STATUS TimerSetPwm(System.IntPtr hDev, int Ch, ref double pFrequency, ref double pDuty, ref double pPhase)

表 50　TWB_TimerSetPwmExt() の関数宣言

言語	関数宣言
C/C++	TW_STATUS TWB_TimerSetPwmExt(TW_HANDLE hDev, long Ch, double dClkFreq, double *pFrequency, double *pDuty, double *pPhase)
VB	Function TWB_TimerSetPwmExt(ByVal hDev As System.IntPtr, ByVal Ch As Integer, ByVal dClkFreq As Double, ByRef pFrequency As Double, ByRef pDuty As Double, ByRef pPhase As Double) As Integer
VBA	Function TWB_TimerSetPwmExt(ByVal hDev As Long, ByVal Ch As Long, ByVal dClkFreq As Double, ByRef pFrequency As Double, ByRef pDuty As Double, ByRef pPhase As Double) As Long
C#	STATUS TimerSetPwmExt(System.IntPtr hDev, int Ch, double dClkFreq, ref double pFrequency, ref double pDuty, ref double pPhase)

pFrequency 引数はパルスの繰り返し周波数を Hz 単位で入力します。*pDuty* 引数は ON デューティを 0～1.0 の範囲で入力します。*pPhase* 引数は出力開始時の位相を 0～1.0 の範囲で入力します。*TWB_TimerSetPwmExt()* 関数の *dClkFreq* 引数は TCLKA に入力する外部クロックの周波数を Hz 単位で設定してください。

各引数と出力パルスの関係を図 14 に示します。

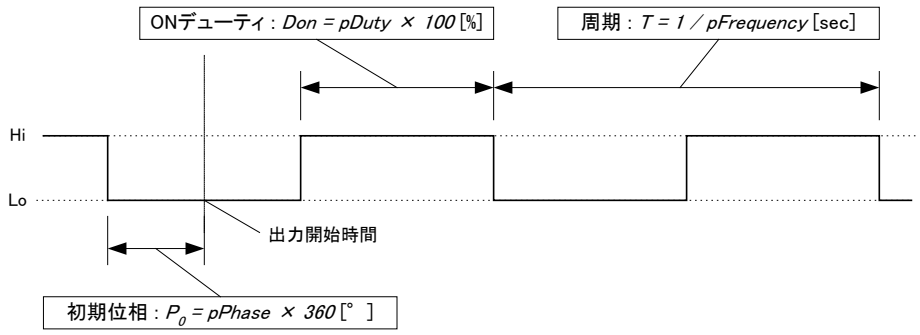


図 14　パラメータと出力パルスの関係

パルスのタイミングは基準クロックを分周して生成されるため、実際に設定できる周波数、デューティ、初期位相の各値は離散的になります。*TWB_TimerSetPwm()* 、*TWB_TimerSetPwmExt()* 関数は各パラメータを引数の入力値と近い値に調整し、*pFrequency*、*pDuty*、*pPhase* の各引数に実際に設定できた値を出力して返ります。

PWM 出力の手順

1. *TWB_TimerSetMode()* 関数(34 ページ、表 40)を呼び出し、使用するタイマチャンネルを PWM モードに設定します。*Mode* 引数の値は表 51 を参照してください。

表 51　PWM 出力で Mode 引数に指定する値

言語	値	説明
C/C++	TWB_TIMER_PWM	指定チャンネルを PWM モードに設定します。対応する端子は PWM 出力用となります。
C++	TWB::TIMER_MODE::PWM	
VB/VBA	TWB_TIMER_MODE.PWM	
C#	TWB.TIMER_MODE.PWM	
C/C++	TWB_TIMER_DISABLE	PWM モードを解除する場合に指定します。PWM 端子がデジタル入出力端子として使用可能になります。
C++	TWB::TIMER_MODE::DISABLE	
VB/VBA	TWB_TIMER_MODE.DISABLE	
C#	TWB.TIMER_MODE.DISABLE	

2. *TWB_TimerSetPwm()* または *TWB_TimerSetPwmExt()* 関数を使用し、出力パルスの設定を行います。
3. PWM 出力端子の初期状態を変更する必要がある場合は *TWB_TimerSetLevel()* 関数を使用します(16 ビットタイマチャンネルを PWM モードに設定すると、対応する PWM 出力端子はデジタル出力としてコントロールすることができなくなります)。
4. 必要であれば *TWB_TimerSetNumOfPulse()* 関数で出力パルス数を設定します。
5. *TWB_TimerStart()* 関数でパルス出力を開始します。
6. パルス出力中も *TWB_TimerSetPwm()* 関数等で周波数とデューティを変更することが可能です。
7. *TWB_TimerSetNumOfPulse()* 関数で出力パルス数を設定した場合は、指定のパルス数を出力するとタイマが自動的に停止します。残りの出力パルス数を調べたい場合には、*TWB_TimerReadNumOfPulse()* 関数を使用します。タイマが動作中か停止中かを調べるには *TWB_TimerReadStatus()* 関数を使用します。
8. パルス出力を停止する場合は *TWB_TimerStop()* 関数を使用します。

TWB_TimerStop() 関数でタイマの動作と非同期に停止を行うと、パルス出力が"Hi"状態で停止する場合があります。これを避けたい場合には以下の手順で停止を行ってください。

1. *TWB_PortWrite()* 関数で使用する PWM 端子と対応するポートビット(PA2、PA4、または、PA6)に 0 を書き込み、デジタル出力時に"Lo"となるように設定します。また、それぞれの端子は *TWB_PortSetDir()* 関数で出力端子となるように設定を行ってください。
2. *TWB_TimerSetMode()* 関数で PWM モードを解除します。この時点で端子の機能が PWM からデジタル出力に切り替わり、出力が"Lo"になります。また、タイマの動作も停止します。
3. *TWB_TimerSetLevel()* 関数で停止したタイマ出力を"Lo"にします。これを行わないと次回の PWM 出力時に意図しないパルスが出力される場合があります。

- 内部クロックによる動作中に 400Hz 以下の範囲の出力周波数変更を行うと、パルスタイミングの誤差を生じやすくなります。
- 出力周波数が 100kHz 以上の範囲でのデューティや周波数の変更を行うと、パルスタイミングの誤差や、パルスの抜け[9]を生じやすくなります。
- *TWB_TimerSetNumOfPulse()* 関数によってパルス数を指定する場合は、"Lo"期間が 50μsec 以上になるようにしてください。"Lo"期間が短すぎると、停止処理にかかる時間により指定のパルス数を越えてしまったり、"Hi"期間に停止したりする場合があります。

リスト 28　PWM 出力の例(C 言語)

```
double dFreq;
double dDuty;
TCHAR c[256];

dFreq = 9500; //周波数 = 9.5kHz
dDuty = 0.6;  //デューティ = 60%

//タイマ 0 を PWM に設定
TWB_TimerSetMode(hDev, 0, TWB_TIMER_PWM);

//パルス設定
TWB_TimerSetPwm(hDev, 0, &dFreq, &dDuty);

//実際の設定値を表示
_stprintf_s(c, 256, _T("周波数 : %.2f Hz  "), dFreq);
OutputDebugString(c);

_stprintf_s(c, 256, _T("デューティ : %.2f %% に設定しました。¥n"), dDuty * 100);
OutputDebugString(c);

//出力パルス数を 100 に設定
TWB_TimerSetNumOfPulse(hDev, 0, 100);

//出力開始
TWB_TimerStart(hDev, TWB_TIMER_BIT0);
```

[9] 出力パルスに抜けが生じると、1 サイクル以上"Lo"状態または"Hi"状態となります。

リスト 29　PWM 出力の例(Visual Basic)

```
Dim dFreq As Double
Dim dDuty As Double

dFreq = 9500 '周波数 = 9.5kHz
dDuty = 0.6  'デューティ = 60%

'タイマ 0 を PWM に設定
TWB_TimerSetMode(hDev, 0, TWB_TIMER_MODE.PWM)

'パルス設定
TWB_TimerSetPwm(hDev, 0, dFreq, dDuty)

'実際の設定値を表示
Debug.WriteLine(String.Format("周波数 : {0:#.#0} Hz", dFreq))
Debug.WriteLine(String.Format("デューティ : {0:##0.#0} % に設定しました。", dDuty * 100))

'出力パルス数を 100 に設定
TWB_TimerSetNumOfPulse(hDev, 0, 100)

'出力開始
TWB_TimerStart(hDev, TWB_TIMER_BITS.TIMER0)
```

リスト 30　PWM 出力の例(C#)

```
double dFreq;
double dDuty;
double dPhase;

dFreq = 9500; //周波数 = 9.5kHz
dDuty = 0.6;  //デューティ = 60%
dPhase = 0;

//タイマ 0 を PWM に設定
TWB.TimerSetMode(hDev, 0, TWB.TIMER_MODE.PWM);

//パルス設定
TWB.TimerSetPwm(hDev, 0, ref dFreq, ref dDuty, ref dPhase);

//実際の設定値を表示
Debug.WriteLine(string.Format("周波数 : {0:#.#0} Hz", dFreq));
Debug.WriteLine(string.Format("デューティ : {0:##0.#0} % に設定しました。", dDuty * 100));

//出力パルス数を 100 に設定
TWB.TimerSetNumOfPulse(hDev, 0, 100);

//出力開始
TWB.TimerStart(hDev, TWB.TIMER_BITS.TIMER0);
```

□ シリアルポート

シリアルポートは最大 2 チャンネル使用可能です。シリアル 0 は自由に使用することができます。

シリアル 1 はデフォルトの状態ではユーザーファームのデバッグ用ポート、または、標準入出力ポートとして機能します。ユーザーファームを利用しない場合は、*TWB_SCISetMode()* をシリアル 1 に対して呼び出すことで、使用可能な状態となります。

通信方式は調歩同期のみです。通信速度は 300bps～38400bps でフロー制御はありません。受信バッファは 127 バイトでオーバーフローするとステータスレジスタにエラーを記録し、オーバーフローしたデータは捨てられます。

また、受信データを改行コードなどで分割して読み出したい場合には、デリミタコードを設定しておくことができます。デリミタコードを設定しておくと、*TWB_SCIRead()* 呼び出し時に受信データがチェックされ、デリミタコード(1 バイトまたは 2 バイト)が現れると、シリアルポートからの読み取りを一旦中止し、デリミタコードより後には指定バイトまで 0 をコピーしてデータを返します。

表 52 にシリアルポート制御で使用する関数をあげます。

表 52　シリアルポート制御で使用する関数

関数名	説明
TWB_SCISetMode()	通信条件の設定を行います。
TWB_SCIReadStatus()	シリアルポートのエラー、受信バイト数を読み出します。
TWB_SCIRead()	シリアルポートから指定バイト数のデータを読み出します。
TWB_SCIWrite()	シリアルポートからデータを送信します。
TWB_SCISetDelimiter()	デリミタ文字を指定します。

表 53　シリアルポート制御のサンプルプログラム

開発環境	プロジェクト名またはファイル名	説明
Visual C++ (MFC)	SerialSample	文字の送受信が可能な簡易なターミナルソフト。
Visual Basic	SerialSampleVB	
Visual C#	SerialSampleCS	

シリアルポートの設定

表 54 は *TWB_SCISetMode()* 関数の宣言です。*Mode* 引数には表 55 に示す値を OR で結合して指定します。その際、データ長、パリティ、ストップビットの設定から 1 つずつオプションを選択して結合するようにしてください。指定がない設定項目はデフォルトと書かれたオプションが選択されます。

表 54　TWB_SCISetMode() の関数宣言

言語	関数宣言
C/C++	TW_STATUS TWB_SCISetMode(TW_HANDLE hDev, long Ch, long Mode, long Baud)
VB	Function TWB_SCISetMode(ByVal hDev As System.IntPtr, ByVal Ch As Integer, ByVal Mode As TWB_SCI_MODE, ByVal Baud As TWB_SCI_BAUD) As Integer
VBA	Function TWB_SCISetMode(ByVal hDev As Long, ByVal Ch As Long, ByVal Mode As TWB_SCI_MODE, ByVal Baud As TWB_SCI_BAUD) As Long
C#	STATUS SCISetMode(System.IntPtr hDev, int Ch, SCI_MODE Mode, SCI_BAUD Baud)

表 55　TWB_SCISetMode() の Mode 引数に指定する値

設定項目	言語	値	説明
データ長	C/C++	TWB_SCI_DATA8	データ長を 8 ビットにします。 （デフォルト）
	C++	TWB::SCI_MODE::DATA8	
	VB/VBA	TWB_SCI_MODE.DATA8	
	C#	TWB.SCI_MODE.DATA8	
	C/C++	TWB_SCI_DATA7	データ長を 7 ビットにします。
	C++	TWB::SCI_MODE::DATA7	
	VB/VBA	TWB_SCI_MODE.DATA7	
	C#	TWB.SCI_MODE.DATA7	
パリティ	C/C++	TWB_SCI_NOPARITY	パリティビットを使用しません。 （デフォルト）
	C++	TWB::SCI_MODE::NO_PARITY	
	VB/VBA	TWB_SCI_MODE.NO_PARITY	
	C#	TWB.SCI_MODE.NO_PARITY	
	C/C++	TWB_SCI_EVEN	偶数パリティを使用します。
	C++	TWB::SCI_MODE::EVEN	
	VB/VBA	TWB_SCI_MODE.EVEN	
	C#	TWB.SCI_MODE.EVEN	
	C/C++	TWB_SCI_ODD	奇数パリティを使用します。
	C++	TWB::SCI_MODE::ODD	
	VB/VBA	TWB_SCI_MODE.ODD	
	C#	TWB.SCI_MODE.ODD	
ストップビット	C/C++	TWB_SCI_STOP1	ストップビットを 1 ビットとします。 （デフォルト）
	C++	TWB::SCI_MODE::STOP1	
	VB/VBA	TWB_SCI_MODE.STOP1	
	C#	TWB.SCI_MODE.STOP1	
	C/C++	TWB_SCI_STOP2	ストップビットを 2 ビットとします。
	C++	TWB::SCI_MODE::STOP2	
	VB/VBA	TWB_SCI_MODE.STOP2	
	C#	TWB.SCI_MODE.STOP2	

シリアルポートの使用手順

1. *TWB_SCISetMode()* 関数で通信設定を行います。
2. 必要があれば *TWB_SCISetDelimiter()* 関数でデリミタコードを設定します。
3. データ送信には *TWB_SCIWrite()* 関数を使用します。
4. 受信データ数やエラーを調べるには *TWB_SCIReadStatus()* 関数を使用します。
5. データを受信するには *TWB_SCIRead()* 関数を使用します。

リスト 31　シリアルポートの使用例(C 言語)

```
char cRecv[255];
char cSend[] = "Hello¥r¥nWorld¥r¥n"; //送信文字列
long L;

//シリアル 0 とシリアル 1 を設定(Mode はデフォルト設定)
TWB_SCISetMode(hDev, 0, 0, TWB_SCI_BAUD9600);
TWB_SCISetMode(hDev, 1, 0, TWB_SCI_BAUD9600);

//シリアル 1 のデリミタを CR+LF に設定
TWB_SCISetDelimiter(hDev, 1, "¥r¥n", 2);

//0 チャンネルから文字列を送信
TWB_SCIWrite(hDev, 0, cSend, (long)strlen(cSend));

while(1){
  //受信数を調べる
  TWB_SCIReadStatus(hDev, 1, NULL, &L);
  if(L == 0) break;

  //受信データを読み出す
  TWB_SCIRead(hDev, 1, cRecv, L, NULL);
  OutputDebugStringA(cRecv);
}
```

リスト 32　シリアルポートの使用例(Visual Basic)

```
Dim str As String
Dim bBuff(254) As Byte
Dim i As Integer

'送信文字列
str = "Hello" & vbCrLf & "World" & vbCrLf

'シリアル 0 とシリアル 1 を設定
TWB_SCISetMode(hDev, 0, 0, TWB_SCI_BAUD.BAUD9600)
TWB_SCISetMode(hDev, 1, 0, TWB_SCI_BAUD.BAUD9600)

'シリアル 1 のデリミタを CR+LF に設定
TWB_SCISetDelimiter(hDev, 1, vbCrLf, 2)

'0 チャンネルから文字列を送信
TWB_SCIWrite(hDev, 0, str, str.Length)

Do
  '受信数を調べる
  TWB_SCIReadStatus(hDev, 1, Nothing, i)
  If i = 0 Then Exit Do

  '受信データを読み出して文字列に変換
  TWB_SCIRead(hDev, 1, bBuff, i, i)
  str = System.Text.Encoding.GetEncoding(932).GetString(bBuff, 0, i)
  Debug.WriteLine(str)
Loop
```

リスト 33　シリアルポートの使用例(C#)

```
byte[] bBuff = new byte[255];
string str = "Hello¥r¥nWorld¥r¥n"; //送信文字列
int i;
byte b;

//シリアル 0 とシリアル 1 を設定(Mode はデフォルト設定)
TWB.SCISetMode(hDev, 0, 0, TWB.SCI_BAUD.BAUD9600);
TWB.SCISetMode(hDev, 1, 0, TWB.SCI_BAUD.BAUD9600);

//シリアル 1 のデリミタを CR+LF に設定
TWB.SCISetDelimiter(hDev, 1, "¥r¥n", 2);

//0 チャンネルから文字列を送信
TWB.SCIWrite(hDev, 0, str, str.Length);

while (true)
{
  //受信数を調べる
  TWB.SCIReadStatus(hDev, 1, out b, out i);
  if (i == 0) break;

  //受信データを読み出す
  TWB.SCIRead(hDev, 1, bBuff, i, out i);
  str = System.Text.Encoding.GetEncoding(932).GetString(bBuff, 0, i);
  Debug.WriteLine(str);
}
```

□ ハードウェアイベントの監視

『USBM3069F』、『USBM3069-S』、『USBM3069-SL』を除く製品ではパルスカウンタ（ソフトウェアカウンタ）のカウント値や AD 変換結果を閾値と比較し、指定の値になったときに、アプリケーションに通知することができます。また、ユーザーファームを作成した場合は独自のイベントをアプリケーションに通知することもできます。この通知機能をハードウェアイベントと呼びます。

ハードウェアイベントは通常のアプリケーションプログラムには Windows のメッセージとして、LabVIEW を用いたプログラムにはユーザーイベントとして通知されます。

表 56 はハードウェアイベントのサンプルプログラムです。

表 56　ハードウェアイベントのサンプルプログラム

開発環境	プロジェクト名またはファイル名	説明
Visual C++ (MFC)	EventSample	パルスカウンタの値が変化した場合、アナログ入力が閾値を超えた場合に画面に表示します。
Visual Basic	EventSampleVB	
Visual C#	EventSampleCS	

- Windows のメッセージによる通知は、割り込みのように瞬時に行われるものではありませんので、リアルタイム制御には利用できません。

ハードウェアイベントの監視を開始するには、表 57 の *TWB_SetHwEvent()* 関数を使用します。この関数には引数として *TWB_HW_EVENT* 構造体(表 58)を渡します。

表 57　TWB_SetHwEvent() の関数宣言

言語	関数宣言
C/C++	TW_STATUS TWB_SetHwEvent(TW_HANDLE hDev, TWB_HW_EVENT *pHwEvent)
VB	Function TWB_SetHwEvent(ByVal hDev As System.IntPtr, ByRef pHwEvent As TWB_HW_EVENT) As Integer
VBA	Function TWB_SetHwEvent(ByVal hDev As Long, ByRef pHwEvent As TWB_HW_EVENT) As Long
C#	STATUS SetHwEvent(System.IntPtr hDev, ref HW_EVENT pHwEvent)

表 58　TWB_HW_EVENT 構造体の宣言

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];
	} TWB_HW_EVENT;

言語	宣言

VB	Public Structure TWB_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 TWB_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

C#	public struct HW_EVENT
	{
	  public System.IntPtr hRecvWindow;
	  public uint idRecvThread;
	  public System.IntPtr lpRsv;
	  public int Message;
	  public uint EventBits;
	  [MarshalAs(UnmanagedType.ByValArray, SizeConst = 4)]
	  public int[] PCCnt;
	  [MarshalAs(UnmanagedType.ByValArray, SizeConst = 4)]
	  public int[] PCCmp;
	  [MarshalAs(UnmanagedType.ByValArray, SizeConst = 4)]
	  public int[] ADVal;。
	  [MarshalAs(UnmanagedType.ByValArray, SizeConst = 4)]
	  public short[] ADCmp;
	
	  public void Initialize()
	  {
	  PCCnt = new int[4];
	  PCCmp = new int[4];
	  ADVal = new int[4];
	  ADCmp = new short[4];
	  }
	}

hRecvWindow
ウィンドウでメッセージを受け取る場合は、ウィンドウハンドルを入力します。必要が無い場合は"0"にしてください。

idRecvThread
スレッドでメッセージを受け取る場合は、スレッド ID を指定します。必要が無い場合は 0 にしてください。

Message
指定のイベントが発生したときに通知されるメッセージ番号です。通常は H'8000～H'BFFF の範囲の任意の値を指定します。パルスカウンタやアナログ入力について条件が成立すると、ここで設定した番号のメッセージが、ウィンドウ、または、スレッドにポストされます。

EventBits
通知を受けたいイベントをビットで指定します(表 59)。複数のビットを指定することもできます。

PCCnt
パルスカウンタの値と比較する閾値を指定します。配列のインデックスがパルスカウンタのチャンネルと対応しています。

PCCmp
パルスカウンタの値と閾値の比較方法を指定します(表 60)。配列のインデックスがパルスカウンタのチャンネルと対応しています。

ADVal
アナログ入力の値と比較する閾値を指定します。配列のインデックスがアナログ入力のチャンネルと対応しています。

ADCmp
アナログ入力の値と閾値の比較方法、ヒステリシスの大きさを指定します。配列のインデックスがアナログ入力のチャンネルと対応しています。

Initialize()
Visual Basic と C#で配列の領域確保用に最初に呼び出します。

表 59　EventBits の指定

定数(C#での記述)	値	監視するイベント
TWB_EVENT_PC0(TWB.EVENT_PC0)	H'00000001	パルスカウンタ 0 を監視
TWB_EVENT_PC1(TWB.EVENT_PC1)	H'00000002	パルスカウンタ 1 を監視
TWB_EVENT_PC2(TWB.EVENT_PC2)	H'00000004	パルスカウンタ 2 を監視
TWB_EVENT_PC3(TWB.EVENT_PC3)	H'00000008	パルスカウンタ 3 を監視
TWB_EVENT_AD0(TWB.EVENT_AD0)	H'00000010	アナログ入力 0 を監視
TWB_EVENT_AD1(TWB.EVENT_AD1)	H'00000020	アナログ入力 1 を監視
TWB_EVENT_AD2(TWB.EVENT_AD2)	H'00000040	アナログ入力 2 を監視
TWB_EVENT_AD3(TWB.EVENT_AD3)	H'00000080	アナログ入力 3 を監視
TWB_EVENT_USER(TWB.EVENT_USER)	H'80000000	ユーザー定義 (ユーザーファームから通信を行う場合)
TWB_EVENT_PC_ALL(TWB.EVENT_PC_ALL)	H'0000000F	パルスカウンタ全チャンネルを監視
TWB_EVENT_AD_ALL(TWB.EVENT_AD_ALL)	H'000000F0	アナログ入力全チャンネルを監視

パルスカウンタ入力を監視する

パルスカウンタによるイベントは、カウンタ値が予め設定した閾値以上となった場合、閾値以下となった場合、または、カウンタ値が変化した場合に発生させることができます。

1. Visual Basic、C# を利用する場合、*TWB_HW_EVENT* 構造体の *Initialize()* メソッドを呼びます。
2. *TWB_HW_EVENT* 構造体の *hRecvWindow* にウィンドウのハンドルを指定します。ウィンドウを持たないアプリケーションの場合、*idRecvThread* にスレッド ID を指定します。また、イベント発生時に受け取るメッセージ番号を *Message* に指定します。
3. *TWB_HW_EVENT* 構造体の *EventBits* に監視するパルスカウンタチャンネルを指定します(表 59 参照)。複数のチャンネルを指定する場合は OR で結合することができます。
4. *TWB_HW_EVENT* 構造体の *PCCnt* にカウンタ値と比較する閾値を指定します。配列のインデックスはチャンネルを示します。例えばパルスカウンタ 2 を監視する場合は、*PCCmp[2]* に閾値を設定します。
5. *TWB_HW_EVENT* 構造体の *PCCmp* に比較方法を指定します。*PCCmp* に指定する値と、イベント発生条件を 表 60 に示します。

表 60 PCCmp の設定値とハードウェアイベントの発生条件

定数(C#での記述)	値	ハードウェアイベントの発生条件
TWB_CMP_GE(TWB.CMP_GE)	H'7FFFFFFF	指定チャンネル(x)のカウンタ値が PCCnt[x]以上になった場合
TWB_CMP_NO(TWB.CMP_NO)	0	指定チャンネル(x)のカウンタ値が変化した場合
TWB_CMP_LE(TWB.CMP_LE)	H'80000000	指定チャンネル(x)のカウンタ値が PCCnt[x]以下となった場合

6. パラメータを設定した構造体を引数として *TWB_SetHwEvent()* 関数を呼び出します。
7. 使用するパルスカウンタの設定を行い、カウント動作を開始します(32 ページ参照)。
8. 設定した条件が成立すると、指定したウィンドウ(または、スレッド)にメッセージがポストされます。メッセージの各パラメータは以下の値となります。

表 61 パルスカウンタイベントによるメッセージのパラメータ

項目	説明
Msg	TWB_HW_EVENT 構造体の Message に指定した値
wParam(WParam)	イベントを発生したカウンタチャンネルを示すビット(表 59)
lParam(LParam)	イベント発生時のカウンタの値

9. イベントの監視を終了する場合、VBA 以外の言語では *pHwEvent* を Null 値として *TWB_SetHwEvent()* 関数を呼び出します。VBA では *pHwEvent* の *EventBits* を 0 として呼び出してください。

アナログ入力を監視する

アナログ入力によるイベントは、アナログ入力値が予め設定した閾値以上となった場合、または、閾値以下となった場合に発生させることができます。

また、アナログ入力値が閾値付近にあるとき、不要なイベントが何度も発生するのを防ぐために適当なヒステリシスを持たせることができます。例えば、アナログ入力値がある閾値電圧 V_{TH} 以上になった場合にイベントを発生するように設定した場合、アナログ入力電圧が V_{TH} 以上になることでハードウェアイベントが検出されますが、この時点で該当チャンネルの次のイベント検出は一旦禁止されます。この禁止状態は入力電圧が(V_{TH} 以下ではなく) V_{TH} - V_{HYST} 以下となったときに解除されます(図 15)。ここでは V_{HYST} をヒステリシス、または、ヒステリシス電圧と呼びます。

ヒステリシス電圧が適切な大きさに設定されていないと、入力電圧が V_{TH} 付近のとき、ノイズなどによる微小な電圧変化でもハードウェアイベントが検出されてしまい、不要なメッセージが何度も送信される場合があります。

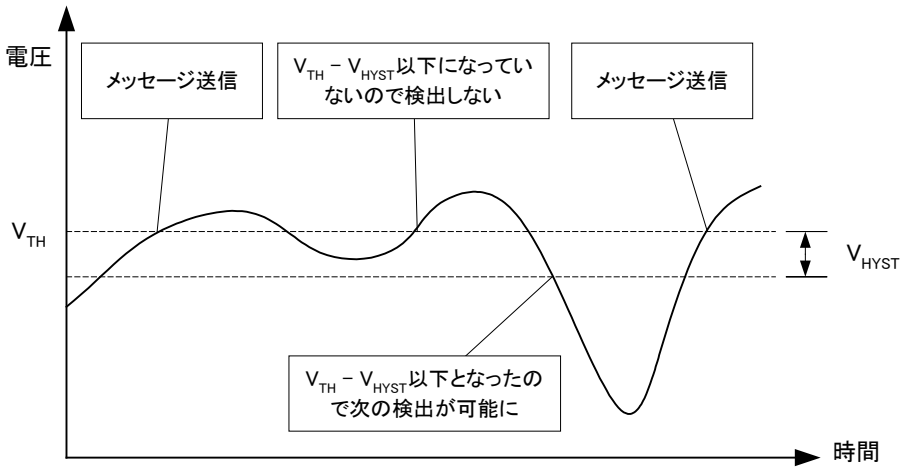


図 15　ヒステリシスが設定されている場合の動作

1. Visual Basic、C# を利用する場合、*TWB_HW_EVENT* 構造体の *Initialize()* メソッドを呼びます。
2. *TWB_HW_EVENT* 構造体の *hRecvWindow* にウィンドウのハンドルを指定します。ウィンドウを持たないアプリケーションの場合、*idRecvThread* にスレッド ID を指定します。また、イベント発生時に受け取るメッセージ番号を *Message* に指定します。
3. *TWB_HW_EVENT* 構造体の *EventBits* に監視するアナログ入力チャンネルを指定します(表 59 参照)。
4. *TWB_HW_EVENT* 構造体の *ADVal* にアナログ入力値と比較する閾値を指定します。*ADVal* の値は 28 ページ 図 4 の形式の変換値と比較されます。配列のインデックスはチャンネルを示します。例えばアナログ入力 2 を監視する場合は、*ADVal[2]* に閾値を設定します。閾値電圧 V_{TH} から *ADVal* への設定値 C_{TH} を求めるには下の式を使用します。

$$C_{TH} \fallingdotseq (V_{TH}\ [V] / 5\ [V]) \times 65536 \quad \cdots \text{式 1}$$

5. *TWB_HW_EVENT* 構造体の *ADCmp* に比較方法とヒステリシス電圧を指定します。表 62 に *ADCmp* に指定する値と、イベント発生条件、再度イベント発生が可能になる条件を示します。

表 62　ADCmp の設定値とハードウェアイベントの発生条件

ADCmp[x]の設定	ハードウェアイベントの発生条件	再度イベント発生可能となる条件
0 以上の場合	指定チャンネル(x)の AD 変換値が ADVal[x]以上になった場合	指定チャンネル(x)の AD 変換値が ADVal[x] - ADCmp[x]以下になった場合
負の場合	指定チャンネル(x)の AD 変換値が ADVal[x]以下になった場合	指定チャンネル(x)の AD 変換値が ADVal[x] - ADCmp[x]以上になった場合

6. パラメータを設定した構造体を引数として *TWB_SetHwEvent()* 関数を呼び出すと、指定のアナログ入力チャンネルの監視が開始されます。

7. 設定した条件が成立すると、指定したウィンドウ(または、スレッド)にメッセージがポストされます。メッセージの各パラメータは以下の値となります。

表 63　アナログ入力イベントによるメッセージのパラメータ

項目	説明
Msg	TWB_HW_EVENT 構造体の Message に指定した値
wParam(WParam)	イベントを発生したアナログ入力を示すビット(表 59)
lParam(LParam)	イベントを発生したアナログ入力チャンネルの AD 変換結果(28 ページ、図 4 の形式)

8. イベントの監視を終了する場合、VBA 以外の言語では *pHwEvent* を Null 値として *TWB_SetHwEvent()* 関数を呼び出します。VBA では *pHwEvent* の *EventBits* を 0 として呼び出してください。

- 0 ＜ *ADVal[x]* － *ADCmp[x]* ＜ 65535 となるようにしてください。

□ 外部バス

デバイスのメモリ空間は図 16 のようになっています。このうち白い四角の中はユーザーが利用できる外部バス空間です。プログラム中からこの領域のアドレスへアクセスした場合、外部バスへのアクセスとなります。

図のようにデバイスが扱うメモリ空間はそれぞれ 8 つのエリアに分割されており、それぞれに対して別々のチップセレクト信号が出力されるようになっています。各エリアは 2M バイトの領域を持っていますが、製品では上位 4 ビットのアドレスを出力しないため下位 20 ビットのアドレスで表現できる 1M バイトの領域だけを扱うことができます。そのため、合計で最大 4M バイトの外部バス空間が利用可能になっています。

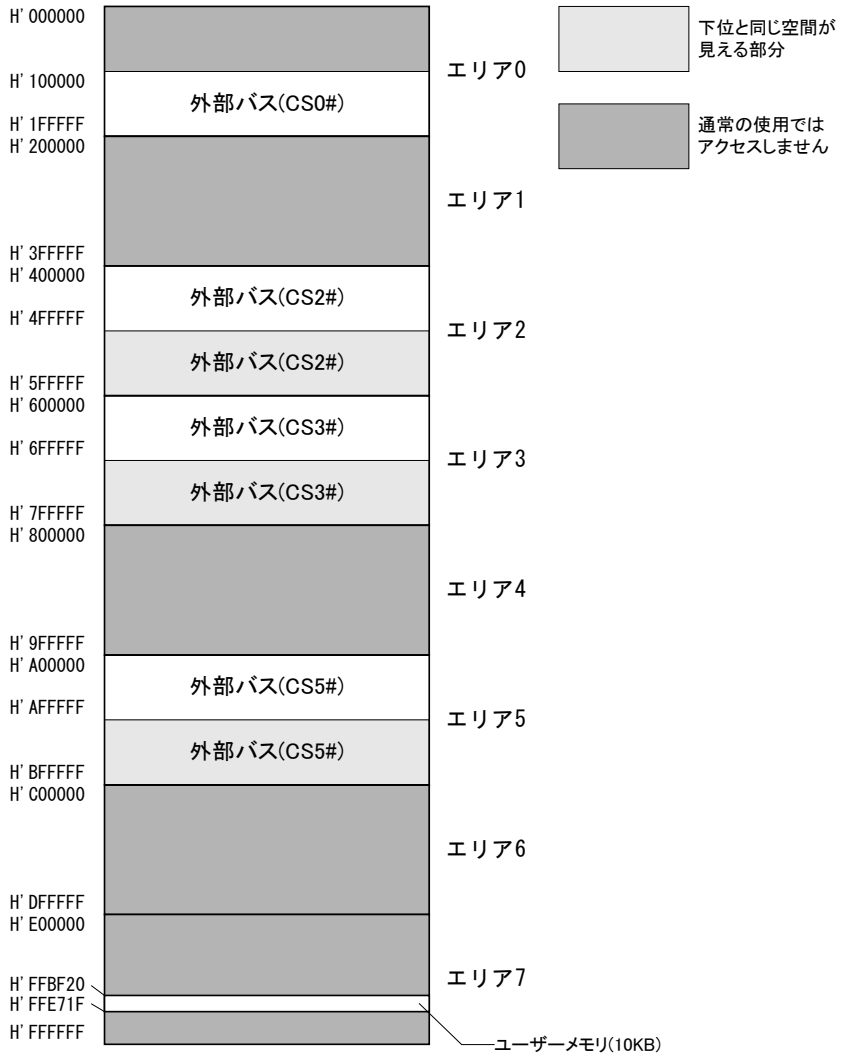


図 16　メモリ空間

表 64　外部バスを使用するための関数

関数名	説明
TWB_BusEnableAddress()	アドレスバスの出力ビット数を指定します。
TWB_BusEnableCS()	CS2#,CS3#信号の出力許可/禁止を設定します。
TWB_BusSetWidth16()	指定のエリアを 16 ビットアクセス空間にします。
TWB_BusSetWait()	指定のエリアのアクセスウェイトを設定します。
TWB_PortWrite()	1 バイトのデータを指定アドレスに書き込みます。
TWB_PortRead()	1 バイトのデータを指定アドレスから読み出します。
TWB_PortWrite16()	1 ワード(16 ビット)のデータを指定アドレスに書き込みます。書込み対象がワードアクセス可能であれば、ワードアクセスします。
TWB_PortRead16()	1 ワード(16 ビット)のデータを指定アドレスから読み出します。読出し対象がワードアクセス可能であれば、ワードアクセスします。
TWB_PortBWrite()	複数バイトのデータを指定アドレスに書き込みます。
TWB_PortBRead()	複数バイトのデータを指定アドレスから読み出します。

アドレスの出力

アドレス信号 A0～A19 は、入力ポート P10～P17、P20～P27、P50～P53 と端子が共通になっており、デフォルトの状態では出力されません。

アドレスの出力には *TWB_BusEnableAddress()* 関数(表 65)を使用します。A0 から順番に *nBits* 引数で指定したビット数のアドレスが出力されます。アドレス出力とした端子はデジタル入力端子としては使用できません。

表 65　TWB_BusEnableAddress() の関数宣言

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

チップセレクトの出力

CS0#、CS5#信号は常に出力可能です。CS2#と CS3#信号はそれぞれ PC3#、PC2#と端子が共通になっており、デフォルトの状態では出力されません。

CS2#、CS3#の出力には *TWB_BusEnableCS()* 関数を使用します。

バス幅の設定

バスはエリア毎に 8 ビットアクセスとするか 16 ビットアクセスとするかを選択できます。バス幅を設定するには *TWB_BusSetWidth16()* 関数を使用します。

8 ビット空間へのアクセスはデータバスの上位 8 ビット(D8～D15)を使用して行います(図 17)。

図 17　8 ビット空間へのアクセス

16 ビット空間へのワードアクセスではデータバスの全てのビット(D0～D15)が有効になります（図 18)。

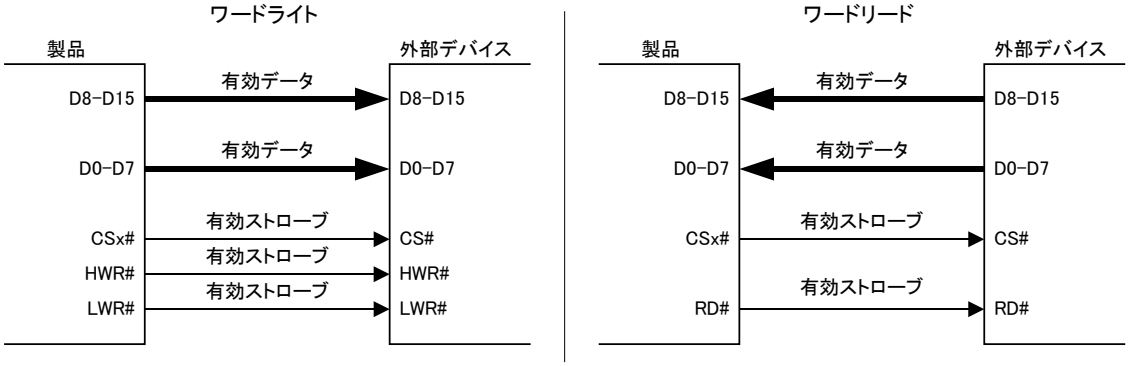


図 18　16 ビット空間へのワードアクセス

16 ビット空間へバイトアクセスする場合は、偶数アドレスへのアクセスではデータバスの上位 8 ビット(D8～D15)、奇数アドレスへのアクセスでは下位 8 ビット(D0～D7)が使用されます。また、ライトアクセスでは HWR#、LWR#によってデータバスの上位バイトと下位バイトのどちらが有効かを示します(図 19)。

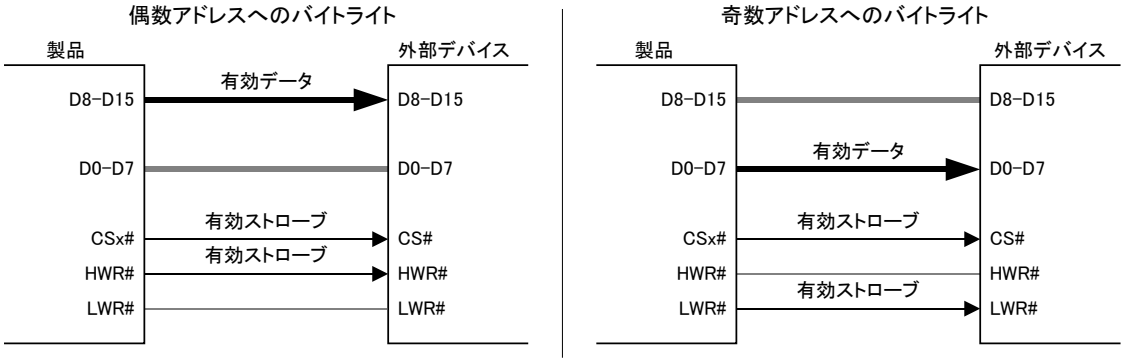


図 19　16 ビット空間へのバイトアクセス

- D0～D7 信号は P40～P47 信号と端子が共用になっています。16 ビット幅に選択したエリアが 1 つでもある場合、端子はデータバスとして機能し P40～P47 のデジタル入出力機能は使用できなくなります。

バスへのアクセス

外部バスに対して複数バイトのデータを読み書きする場合は、書込みには *TWB_PortBWrite()*、読出しには *TWB_PortBRead()* 関数を使用します(表 66、表 67)。

表 66 TWB_PortBWrite() の関数宣言

言語	関数宣言
C/C++	TW_STATUS TWB_PortBWrite(TW_HANDLE hDev, DWORD Port, void *pData, long nData, long Inc, long Dma)
VB	Function TWB_PortBWrite(ByVal hDev As System.IntPtr, ByVal Port As Integer, ByVal pData As Object, ByVal nData As Integer, ByVal Inc As Integer, ByVal Dma As Integer) As Integer
VBA	Function TWB_PortBWrite(ByVal hDev As Long, ByVal Port As Long, ByRef pData As Any, ByVal nData As Long, ByVal Inc As Long, ByVal Dma As Long) As Long
C#	STATUS PortBWrite(System.IntPtr hDev, uint Port, object pData, int nData) STATUS PortBWrite(System.IntPtr hDev, uint Port, object pData, int nData, int Inc, int Dma)

表 67 TWB_PortBRead() の関数宣言

言語	関数宣言
C/C++	TW_STATUS TWB_PortBRead(TW_HANDLE hDev, DWORD Port, void *pData, long nData, long Inc, long Dma)
VB	Function TWB_PortBRead(ByVal hDev As System.IntPtr, ByVal Port As Integer, ByVal pData As Object, ByVal nData As Integer, ByVal Inc As Integer, ByVal Dma As Integer) As Integer
VBA	Function TWB_PortBRead(ByVal hDev As Long, ByVal Port As Long, ByRef pData As Any, ByVal nData As Long, ByVal Inc As Long, ByVal Dma As Long) As Long
C#	STATUS PortBRead(System.IntPtr hDev, uint Port, object pData, int nData) STATUS PortBRead(System.IntPtr hDev, uint Port, object pData, int nData, int Inc, int Dma)

Port 引数に書込み、読出しの対象となるアドレスを指定します。

Inc 引数は 0 とするとアクセス時にアドレスのインクリメントが行われません。FIFO のような固定アドレスのデバイスにアクセスする場合に 0 とします。

Dma 引数は 0 以外の値にすると外部バスへのアクセスに、デバイスに搭載されたマイコン内蔵の DMA[10]が使用されます。DMA を使用すると、サイズの大きなデータの転送が高速になります。

- USB(FS)デバイスは、DMA を使用すると「外部リクエストモード」という転送モードが使用されます。このモードでは *TWB_PortWrite()* 関数終了時に TEND0#(PA0/TCLKA と共用)、*TWB_PortRead()* 関数終了時に TEND1#(PA1/TCLKB と共用)という転送終了を示す"Lo"レベルの信号が出力されます。この出力は端子の設定に無関係に行われますので、これらの端子を使用している場合には DMA による転送を行わないでください。
- USB(HS)デバイス、および、LAN デバイスでは TENDx#信号は出力されません。

[10] *TWB_PortBWrite()* 関数には DMA0、*TWB_PortBRead()* 関数には DMA1 が使用されます。

USB(HS)デバイスを除いて *TWB_PortBWrite()* 、*TWB_PortBRead()* 関数による外部バスへのアクセスはバイトアクセスで行われます。16 ビットアクセスを行いたい場合は *TWB_PortWrite16()*、*TWB_PortRead16()* 関数により 1 ワードずつアクセスする必要があります。

USB(HS)デバイスは、デバイス内蔵のマイコンと USB インタフェース IC の転送バス幅と、アクセスする外部アドレス空間の両方が 16 ビットに設定されている場合のみ *TWB_PortBWrite()*、*TWB_PortBRead()* 関数による外部バスへのアクセスがワードアクセスになります。

マイコンと USB インタフェース IC 間の転送バス幅を 16 ビットとするには、*TWB_Open()* 関数の *Opt* 引数に表 68 の値を指定します。

表 68　16 ビット転送用の Opt 引数の値

言語	値	説明
C/C++	TWB_MODE_BUS16	USB(HS)デバイス専用。マイコンと USB インタフェース IC 間のデータ転送を 16 ビット幅で行います。データ転送は高速になりますが、P40～P47 は入出力ポートとして使用できなくなります。
C++	TWB::OPEN_OPT::MODE_BUS16	
VB/VBA	TWB_OPEN_OPT.MODE_BUS16	
C#	TWB.OPEN_OPT.MODE_BUS16	

リスト 34　USB(HS)デバイスのワード転送例(C 言語)

```
TW_HANDLE hDev;
BYTE bBuff[256];

TWB_Open(&hDev, NULL, 1, TWB_IF_USB_HS | TWB_MODE_BUS16); //16 ビットインタフェースで接続
if(hDev){
  TWB_BusEnableAddress(hDev, 16); //A0～A15 を出力
  TWB_BusSetWidth16(hDev, TWB_BUS_AREA5); //エリア 5 を 16 ビットバス空間に設定
  TWB_PortBRead(hDev, 0xa00000, bBuff, 256, 1, 0); //256 バイトを読出し
  TWB_Close(hDev);
}
```

リスト 35　USB(HS)デバイスのワード転送例(Visual Basic)

```
Dim hDev As System.IntPtr
Dim bBuff(255) As Byte

'16 ビットインタフェースで接続
TWB_Open(hDev, Nothing, 1, TWB_OPEN_OPT.IF_USB_HS Or TWB_OPEN_OPT.MODE_BUS16)
If hDev <> IntPtr.Zero Then
  TWB_BusEnableAddress(hDev, 16) 'A0～A15 を出力
  TWB_BusSetWidth16(hDev, TWB_BUS_AREA.AREA5) 'エリア 5 を 16 ビットバス空間に設定
  TWB_PortBRead(hDev, &HA00000, bBuff, 256, 1, 0) '256 バイトを読出し
  TWB_Close(hDev)
End If
```

リスト 36　USB(HS)デバイスのワード転送例(C#)

```
System.IntPtr hDev;
byte[] bBuff = new byte[256];

//16 ビットインタフェースで接続
TWB.Open(out hDev, null, 1, TWB.OPEN_OPT.IF_USB_HS | TWB.OPEN_OPT.MODE_BUS16);
if (hDev != IntPtr.Zero)
{
  TWB.BusEnableAddress(hDev, 16); //A0～A15 を出力
  TWB.BusSetWidth16(hDev, TWB.BUS_AREA.AREA5); //エリア 5 を 16 ビットバス空間に設定
  TWB.PortBRead(hDev, 0xa00000, bBuff, 256, 1, 0); //256 バイトを読出し
  TWB.Close(hDev);
}
```

□ ユーザーステータスレジスタ／ユーザーメモリの利用

パソコン上のアプリケーションプログラムを終了させても、デバイスがどのような状態にあるかを記憶しておき、次にアプリケーションプログラムを実行したときに、その続きから制御を行いたい場合があります。このようなときにユーザーステータスレジスタとユーザーメモリが利用できます。

ユーザーステータスレジスタはデバイス内の 1 バイトのメモリで、デバイスの起動時や再初期化のときには必ず 0 にクリアされます。ユーザーステータスレジスタを利用して、デバイスが初期化済みであるか、どのような状態にあるか、といった簡単な情報を保存しておくことができます。

利用例として、パルスカウンタを使ったアプリケーションプログラムで、「プログラム終了後もカウント動作を継続し、再度プログラムを起動した場合にはそのときのカウント値を表示したい」といった場合を考えます。このようなとき、最初にアプリケーションプログラムがパルスカウンタを設定した時点で、ユーザーステータスレジスタに初期化済みであるフラグを記録しておきます。2 回目以降のアプリケーションプログラムの実行では、フラグを調べてカウンタの初期化が必要無く、単にカウンタ値を読み出せば良いことがわかります。何らかの理由でデバイスの電源が切れた場合には、ユーザーステータスレジスタ上のフラグがクリアされるので初期化が必要なことがわかります。

ユーザーメモリはデバイスの RAM に確保された 10K バイトのメモリ空間です。ユーザーステータスレジスタでは保存できない比較的大きな設定情報などを記憶することができます。この領域の値は起動時に不定となり、自動的にクリアされることもありませんのでユーザーステータスレジスタと組み合わせて使用してください。ユーザーメモリのアドレスはデバイス上の H'FFBF20～H'FFE71F の範囲です(58 ページ、図 16 参照)。

表 69　ユーザーステータスレジスタ／ユーザーメモリの操作に使用する関数

関数名	説明
TWB_PortWrite()	ユーザーステータスレジスタにデータを書き込みます。
TWB_PortRead()	ユーザーステータスレジスタからデータを読み出します。
TWB_PortBWrite()	ユーザーメモリにデータを書き込みます。
TWB_PortBRead()	ユーザーメモリからデータを読み出します。

表 70　ユーザーステータスレジスタを利用したサンプルプログラム

開発環境	プロジェクト名またはファイル名	説明
Visual C++ (MFC)	PulseCountSample	カウントモードの記録にユーザーステータスレジスタを利用しています。
Visual Basic	PulseCountSampleVB	
Visual C#	PulseCountSampleCS	

ユーザーステータスレジスタの操作方法

入出力ポートなどと同様に *TWB_PortWrite()*、*TWB_PortRead()* 関数を使用して、書込み、読出しが行えます。*Port* 引数には表 71 の値を指定してください。

表 71　ユーザーステータスレジスタを指定する定数

言語	値	説明
C/C++	TWB_USER_STATUS	ユーザーステータスレジスタの読み書き時に指定します。
C++	TWB::WPORT::USER_STATUS TWB::RPORT::USER_STATUS	
VB/VBA	TWB_WPORT.USER_STATUS TWB_RPORT.USER_STATUS	
C#	TWB.WPORT.USER_STATUS TWB.RPORT.USER_STATUS	

ユーザーメモリの操作方法

TWB_PortBRead()、*TWB_PortBWrite()* 関数(表 66、表 67)を使用すると、大きなデータを効率良くリード／ライトできます。これらの関数では *Port* 引数にアドレス、*nData* 引数にバイト数を指定してデバイス上の任意のメモリアドレスにアクセスできます。

- ユーザーメモリ以外の領域に対して読み書きを行うと、誤動作する場合があります。
- ユーザーメモリはユーザーファームの動作にも使用します。ユーザーファーム利用時には自由に使用できる領域が変化しますので誤って操作しないように特に注意が必要です。

□ フラッシュメモリの利用

製品にはフラッシュメモリが内蔵されています。フラッシュメモリは電源を切っても記録した情報が保存される不揮発性のメモリ空間で、製品が動作するためのファームウェアもこの領域に書き込まれています。図 20 はフラッシュメモリ領域を詳しく示した図です。

フラッシュメモリは消去単位毎に EB0～EB15 の 16 ブロックに分けて管理されます。このうち、EB1～EB3 の 12K バイトの領域がユーザーに開放されています。電源を切っても内容が消えないため、アプリケーション固有の設定情報やキャリブレーションデータの保存などに利用可能です。

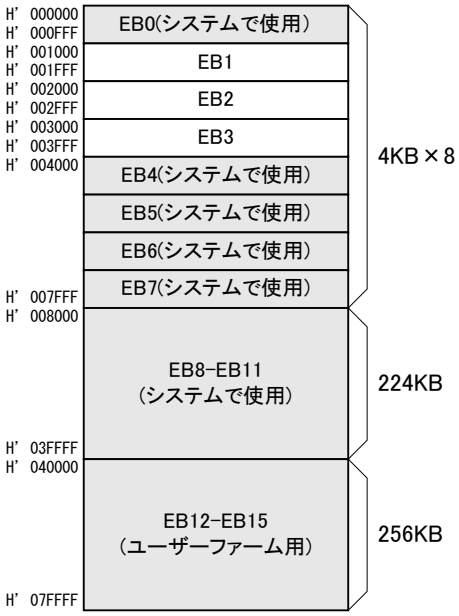


図 20　フラッシュメモリマップ

表 72　フラッシュメモリの操作に使用する関数

モード	関数名	説明
フラッシュ書換えモードで使用する関数	*TWB_FlashEraseBlk()*	フラッシュ書換えモード用。フラッシュメモリの指定ブロックを消去します。
	TWB_FlashWrite()	フラッシュ書換えモード用。フラッシュメモリに書込みを行います。
	TWB_FlashRead()	フラッシュ書換えモード用。フラッシュメモリからデータを読み出します。
ユーザープログラムモードで使用する関数	*TWB_UPFlashAttachWriter()*	ユーザープログラムモード用。フラッシュメモリの消去／書込みのためのファームウェアをデバイスにダウンロードします。
	TWB_UPFlashEraseBlk()	ユーザープログラムモード用。フラッシュメモリの指定ブロックを消去します。
	TWB_UPFlashWrite()	ユーザープログラムモード用。フラッシュメモリに書込みを行います。
	TWB_PortBRead()	ユーザープログラムモード用。フラッシュメモリからデータを読み出します。

表 73　フラッシュメモリ操作のサンプルプログラム

開発環境	プロジェクト名またはファイル名	説明
Visual C++ (MFC)	FlashSample	フラッシュ書換えモードのデバイスに接続し、フラッシュメモリの状態表示、ファイルデータのフラッシュメモリへの書込みを行います。
Visual Basic	FlashSampleVB	
Viusal C#	FlashSampleCS	
VBA	FlashSample.xls	セルを利用した簡易バイナリエディタです。編集内容をフラッシュメモリに書き込むことができます。フラッシュ書換えモード用です。
Visual C++ (MFC)	UPFlashSample	ユーザープログラムモードのデバイスに接続し、フラッシュメモリの状態表示、ファイルデータのフラッシュメモリへの書込みを行います。
Visual Basic	UPFlashSampleVB	
Viusal C#	UPFlashSampleCS	

フラッシュメモリへの書込み操作は特殊で、通常のメモリのように 1 バイト単位でデータを書き込むことはできません。書込みを行う領域には、まず消去の操作を行います。消去の単位は図 20 に示した EB 1 ～EB3 のブロック単位で、消去対象のブロックは全ビットが"1"となります。

続いて、実際に保存するデータの書込み行います。書込みは 128 バイト毎のブロック単位で行います。そのため、書込みの先頭アドレスは常に 128 バイト境界(アドレスの下位 7 ビットが 0)となります。

- フラッシュメモリの書換え可能回数の目安は 100 回、データ保持年数は 10 年です。

『USBM3069(F)』、『LANM3069』、『LANM3069C』で、フラッシュメモリを操作するには、フラッシュ書換えモードに設定する必要があります。その他のデバイスではフラッシュ書換えモードとユーザープログラムモードのどちらかでフラッシュメモリを操作します。それぞれのモードで使用できる関数が異なりますのでご注意ください。

フラッシュメモリの消去方法(フラッシュ書換えモード)

1. デバイスをフラッシュ書換えモードで起動します。
2. *Opt* 引数に *TWB_MODE_FLASH*(相当のオプション) を指定して *TWB_Open()*、 または、*TWB_OpenByAddress()* 関数でデバイスに接続します。
3. *TWB_FlashEraseBlk()* 関数を呼び出します。*Blk* 引数に消去したいブロック番号(1～3)を指定します。

表 74　TWB_FlashEraseBlk() の関数宣言

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

フラッシュメモリの読出し、および、書込み方法(フラッシュ書換えモード)

1. デバイスをフラッシュ書換えモードで起動します。
2. *Opt* 引数に *TWB_MODE_FLASH*（相当のオプション)を指定して *TWB_Open()*、 または、*TWB_OpenByAddress()* 関数でデバイスに接続します。
3. フラッシュ書換えモードのデバイスからデータを読み出すには *TWB_FlashRead()* 関数を使用する必要があります。*Address* 引数には書込み先のアドレスとして 0x1000～0x3f80 の値が指定できますが、128 バイト境界に合わせる必要がありますので、下位 7 ビットは常に 0 になります。また、*nData* 引数に指定する書込みバイト数も 128 の倍数としてください。

表 75 TWB_FlashRead() の関数宣言

言語	関数宣言
C/C++	TW_STATUS TWB_FlashRead(TW_HANDLE hDev, DWORD Address, void *pData, long nData)
VB	Function TWB_FlashRead(ByVal hDev As System.IntPtr, ByVal Address As Integer, ByVal pData As Object, ByVal nData As Integer) As Integer
VBA	Function TWB_FlashRead(ByVal hDev As Long, ByVal Address As Long, ByRef pData As Any, ByVal nData As Long) As Long
C#	SATUS FlashRead(System.IntPtr hDev, uint Address, object pData, int nData)

4. フラッシュメモリに書込みを行うには *TWB_FlashWrite()* 関数を呼び出します。*Address* 引数には書込み先のアドレスとして 0x1000～0x3f80 の値が指定できますが、128 バイト境界に合わせる必要がありますので、下位 7 ビットは常に 0 になります。また、*nData* 引数に指定する書込みバイト数も 128 の倍数としてください。

表 76 TWB_FlashWrite() の関数宣言

言語	関数宣言
C/C++	TW_STATUS TWB_FlashWrite(TW_HANDLE hDev, DWORD Address, void *pData, long nData)
VB	Function TWB_FlashWrite(ByVal hDev As System.IntPtr, ByVal Address As Integer, ByVal pData As Object, ByVal nData As Integer) As Integer
VBA	Function TWB_FlashWrite(ByVal hDev As Long, ByVal Address As Long, ByRef pData As Any, ByVal nData As Long) As Long
C#	SATUS FlashWrite(System.IntPtr hDev, uint Address, object pData, int nData)

リスト 37　フラッシュ書換えモードでのフラッシュメモリ操作例(C 言語)

```
TW_HANDLE hDev;
char cWrite[128] = "Hello World";
char cRead[128];

//フラッシュ書換えモードのデバイスに接続
TWB_Open(&hDev, NULL, 0, TWB_IF_ANY | TWB_MODE_FLASH);

if(m_hDev){
  //ブロック 1 を消去
  TWB_FlashEraseBlk(hDev, 1);

  //書込み
  TWB_FlashWrite(hDev, 0x1000, cWrite, 128);

  //読出し
  TWB_FlashRead(hDev, 0x1000, cRead, 128);
  OutputDebugStringA(cRead);

  TWB_Close(hDev);
}
```

リスト 38　フラッシュ書換えモードでのフラッシュメモリ操作例(Visual Basic)

```
Dim hDev As System.IntPtr
Dim strWrite As New System.Text.StringBuilder("Hello World")
Dim bBuff(127) As Byte

strWrite.Length = 128

'フラッシュ書換えモードのデバイスに接続
TWB_Open(hDev, Nothing, 0, TWB_OPEN_OPT.IF_ANY Or TWB_OPEN_OPT.MODE_FLASH)

If hDev <> System.IntPtr.Zero Then

  'ブロック 1 を消去
  TWB_FlashEraseBlk(hDev, 1)

  '書込み
  TWB_FlashWrite(hDev, &H1000, strWrite, 128)

  '読出し
  TWB_FlashRead(hDev, &H1000, bBuff, 128)
  Debug.WriteLine(System.Text.Encoding.GetEncoding(932).GetString(bBuff, 0, 128))

  TWB_Close(hDev)
End If
```

リスト 39　フラッシュ書換えモードでのフラッシュメモリ操作例(C#)

```
System.IntPtr hDev;
StringBuilder strWrite = new System.Text.StringBuilder("Hello World");
byte[] bBuff = new byte[128];

strWrite.Length = 128;

//フラッシュ書換えモードのデバイスに接続
TWB.Open(out hDev, null, 0, TWB.OPEN_OPT.IF_ANY | TWB.OPEN_OPT.MODE_FLASH);

if (hDev != System.IntPtr.Zero)
{
  //ブロック 1 を消去
  TWB.FlashEraseBlk(hDev, 1);

  //書込み
  TWB.FlashWrite(hDev, 0x1000, strWrite, 128);

  //読出し
  TWB.FlashRead(hDev, 0x1000, bBuff, 128);
  Debug.WriteLine(System.Text.Encoding.GetEncoding(932).GetString(bBuff, 0, 128));

  TWB.Close(hDev);
}
```

フラッシュメモリの消去方法(ユーザープログラムモード)

ユーザープログラムモードでフラッシュメモリの消去／書込みを行う際には、予めフラッシュメモリを制御するためのファームウェアをデバイスにダウンロードする必要があります。このファームウェアはユーザーメモリ(64 ページ参照)にダウンロードされますので、ユーザーメモリは一時的に使用できなくなり、内部のデータも破壊されてしまいますので注意してください。

1. デバイスをユーザープログラムモードに変更します。通常モードからユーザープログラムモードへの変更は起動後に行うこともできます。
2. 通常モードと同様の手順でデバイスに接続します。
3. *TWB_UPFlashAttachWriter()* 関数を呼び出します。
4. 消去したいブロック番号(1～3)を引数として、*TWB_UPFlashEraseBlk()* 関数を呼び出します。

フラッシュメモリへの書込み方法(ユーザープログラムモード)

1. デバイスをユーザープログラムモードに変更します。通常モードからユーザープログラムモードへの変更は起動後に行うこともできます。
2. 通常モードと同様の手順でデバイスに接続します。
3. *TWB_UPFlashAttachWriter()* 関数を呼びます。
4. *TWB_UPFlashWrite()* 関数を呼び出します。*Address* 引数には書込み先のアドレスとして 0x1000～0x3f80 の値が指定できますが、128 バイト境界に合わせる必要がありますので、下位 7 ビットは常に 0 です。また、*nData* 引数に指定する書込みバイト数も 128 の倍数としてください。

ユーザープログラムモードでのフラッシュメモリの読出しは、通常モードのメモリ読出しと同様に *TWB_PortBRead()* 関数で行うことができます。

リスト 40　ユーザープログラムモードでのフラッシュメモリ操作例(C 言語)

```
char cWrite[128] = "Hello World";
char cRead[128];

//ファームウェアのダウンロード
TWB_UPFlashAttachWriter(hDev);

//ブロック 1 を消去
TWB_UPFlashEraseBlk(hDev, 1);

//書込み
TWB_UPFlashWrite(hDev, 0x1000, cWrite, 128);

//読出し
TWB_PortBRead(hDev, 0x1000, cRead, 128);
OutputDebugStringA(cRead);
```

リスト 41　ユーザープログラムモードでのフラッシュメモリ操作例(Visual Basic)

```
Dim strWrite As New System.Text.StringBuilder("Hello World")
Dim bBuff(127) As Byte

strWrite.Length = 128

'ファームウェアのダウンロード
TWB_UPFlashAttachWriter(hDev)

'ブロック 1 を消去
TWB_UPFlashEraseBlk(hDev, 1)

'書込み
TWB_UPFlashWrite(hDev, &H1000, strWrite, 128)

'読出し
TWB_PortBRead(hDev, &H1000, bBuff, 128)
Debug.WriteLine(System.Text.Encoding.GetEncoding(932).GetString(bBuff, 0, 128))
```

リスト 42　ユーザープログラムモードでのフラッシュメモリ操作例(C#)

```
StringBuilder strWrite = new System.Text.StringBuilder("Hello World");
byte []bBuff = new byte[128];

strWrite.Length = 128;

//ファームウェアのダウンロード
TWB.UPFlashAttachWriter(hDev);

//ブロック１を消去
TWB.UPFlashEraseBlk(hDev, 1);

//書込み
TWB.UPFlashWrite(hDev, 0x1000, strWrite, 128);

//読出し
TWB.PortBRead(hDev, 0x1000, bBuff, 128);
Debug.WriteLine(System.Text.Encoding.GetEncoding(932).GetString(bBuff, 0, 128));
```

□ エラー処理

TWB ライブラリの関数のほとんどは戻り値で関数の実行結果を返します。本リファレンスのプログラム例は要点を分かりやすくするために、関数の戻り値チェックを省略していますが、実際のプログラムでは関数が正しく実行されたかどうかチェックすることを推奨します。

関数の戻り値についての詳細は「関数リファレンス」(75 ページ)を参照してください。

リスト 43　エラー処理の例(C 言語)

```
TW_STATUS ret;

ret = TWB_PortWrite(hDev, TWB_POUT, 0x00, 0xff);
if(ret){
  TWB_Close(hDev);
  hDev = 0;
  printf("エラーが発生しました。TW_STATUS = %08X(HEX)", ret);
  return ret;
}
```

リスト 44　エラー処理の例(C++、MFC)

```
CString str;
TW_STATUS ret;

ret = TWB_PortWrite(hDev, TWB::WPORT::POUT, 0x00);
if(ret){
  TWB_Close(hDev);
  hDev = 0;
  str.Format(_T("エラーが発生しました。TW_STATUS=%08X(HEX)"), ret);
  AfxMessageBox(str);
  return ret;
}
```

リスト 45　エラー処理の例(Visual Basic)

```
Dim ret As Integer

ret = TWB_PortWrite(hDev, TWB_WPORT.POUT, &H0)

If ret <> TW_STATUS.TW_OK Then
  TWB_Close(hDev)
  hDev = System.IntPtr.Zero
  MsgBox(String.Format("エラーが発生しました。TW_STATUS = {0:X8}(HEX)", ret))
  Exit Sub
End If
```

リスト 46　エラー処理の例(C#)

```
TWB.STATUS ret;

ret = TWB.PortWrite(hDev, TWB.WPORT.POUT, 0);

if (ret != 0)
{
  TWB.Close(hDev);
  hDev = System.IntPtr.Zero;
  MessageBox.Show(string.Format("エラーが発生しました。TW_STATUS = {0:X8}(HEX)", ret));
  return ret;
}
```

4. 関数リファレンス

□ 関数の説明について

一部の関数は LAN インタフェース製品専用となっています。各関数は対応するホストインタフェースを示すために、関数名を記したタイトルの後に表 77 の対応表示を行っています。

表 77　関数の対応表示

表示	デバイスのホストインタフェース
USB	USB デバイス
LAN	LAN デバイス

説明では最初に C/C++、Visual Basic (.NET 以後)、Visual Basic for Applications (以下 VBA)、C#、それぞれの言語における関数宣言が記載されます(表記方法は前章までと同様です)。次に引数の説明、戻り値の説明(特殊な戻り値を返す場合のみ)、動作説明の順になっています。

ポインタや参照渡しの引数の場合、[入力]または[出力]という記述でデータの方向を示しています。[入力]の場合、関数の入力パラメータとして呼び出し側が予め変数に値をセットする必要があることを示します。[出力]の場合、関数側が実行結果を変数にセットして戻ります。変数によっては両方向に意味があるものもあります。

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 (次項参照)。関数がそれ以外の特別な戻り値を返す場合は、引数の説明の後に戻り値についての説明が記載されます。

- VBA のインテリセンスでは入力候補として列挙子のみが表示され型が表示されませんが、複数の型で同じ列挙子名を使用している場合があるため、列挙子のみ記述した場合にエラーとなる場合があります。例えば P4 という列挙子は TWB_WPORT.P4 と TWB_RPORT.P4 が定義されているため、P4 のみの記述ではエラーとなります。

□ 関数の戻り値の意味

以下に主な戻り値と、予想される原因、対処方法を示します。括弧内の数値が関数から返される数値です。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」、「.vb」、「.bas」、「.cs」のファイル) 中で定義されています。

TW_OK (H'00000000)

正常終了。

TW_INVALID_HANDLE (H'00000001)

デバイスのハンドルが無効であることを示します。既に切断されたハンドルや、接続されていないハンドルを関数に渡した可能性があります。プログラムの内容を確認してください。

TW_DEVICE_NOT_FOUND (H'00000002)

デバイスが見つからなかった場合に返ります。デバイスの電源が入っていない、正しいネットワークに接続されていない、他のプログラムがデバイスを使用しているなどの理由が考えられます。デバイスの状態を確認してください。

TW_IO_ERROR (H'00000004)

デバイスとの通信中にエラーが発生したことを示します。ケーブルが抜かれた場合などに発生します。接続を確認してください。

TW_INSUFFICIENT_RESOURCES (H'00000005)

1 つのプロセスから接続可能なデバイス数(デフォルト値 256)を超えたことを示します。別のプロセスから制御する必要があります。

TW_INVALID_ARGS (H'00000010)

関数に与えられた引数が無効であることを示します。引数の値が誤っている可能性がありますので、プログラムの内容を確認してください。

TW_NOT_SUPPORTED (H'00000011)

サポートされない機能が呼び出されたことを示します。接続中のデバイスがその機能をサポートしない場合、デバイスのファームウェアバージョンが古い場合、動作モードが誤っている場合などに発生します。デバイスの仕様や状態を確認してください。

TW_OTHER_ERROR (H'00000012)

予期しないエラーを示します。お手数ですが製品サポートにお問い合わせください。

TW_TIMEOUT (H'FFFF0001)

送信または受信処理がタイムアウトしたことを示します。何らかの原因でデバイスからの応答がない場合に発生します。電源や接続、動作モードを確認してください。

TW_FILE_ERROR (H'FFFF0002)

ファイル操作に関するエラーが発生したことを示します。ファイルパスが間違っていないか、他のプロセスにより排他オープンされていないか確認してください。

TW_MEMORY_ERROR (H'FFFF0003)

操作に必要なメモリ確保に失敗したことを示します。システムの状態を確認してください。

TW_DATA_NOT_FOUND (H'FFFF0004)

必要な設定データが見つからないことを示します。お手数ですが製品サポートにお問い合わせください。

TW_SOCKET_ERROR (H'FFFF0005)

ネットワークライブラリのエラーです。ネットワークの制御においてエラーが発生したことを示します。多くの場合 Winsock の 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 (H'FFFF0006)

デバイスとの認証作業に失敗したことを示します。パスワードが間違っている可能性があります。デバイスに設定したパスワードとライブラリに設定したパスワードが一致していることを確認してください。

TW_NOT_SUPPORTED_MODE (H'FFFF0007)

呼び出された機能をサポートしないモードであることを示します。デバイスのモード設定を確認してください。

TW_FLASH_MODE_DEVICE (H'FFFF0008)

フラッシュ書換えモードのデバイスのため制御ができないことを示します。デバイスのモード設定を確認してください。

TW_ATF_ERR_FILE_VERSION (H'FFFF0101)

ATF ファイルのバージョンエラーです。ATF のファイルバージョンがライブラリで扱えないものであるときに発生します。新しいライブラリを入手してください。

TW_ATF_ERR_ILLEGAL_FILE (H'FFFF0102)

ATF ファイルの内容が不正であることを示します。ファイルが壊れているか、何らかの理由で正しくない ATF ファイルが作成されたことが考えられます。ファイルに異常が無いと思われる場合には、お手数ですが製品サポートにお問い合わせください。

TW_ATF_ERR_SERVICE_VERSION (H'FFFF0103)

システムファームのバージョンが ATF ファイルの要求に合わないことを示します。システムファームのバージョンが古いか、ATF ファイルの作成時に誤ったバージョン番号を記述した可能性があります。システムファームのバージョン、および、ATF ファイルの要求バージョンを確認してください。

TW_FLASH_ERR_ERASE (H'FFFF0203)

フラッシュメモリの消去に失敗したことを示します。ハードウェアトラブルの可能性があります。お手数ですが製品サポートにお問い合わせください。

TW_FLASH_ERR_WRITE (H'FFFF0204)

フラッシュメモリの書込みに失敗したことを示します。ハードウェアトラブルの可能性があります。お手数ですが製品サポートにお問い合わせください。

TW_FLASH_ERR_SIZE (H'FFFF0210)

フラッシュメモリへの書込みバイト数が正しくないことを示します。128 バイトの倍数になっていない場合、4096 バイトより大きい指定になっている場合などに発生します。書込みバイト数の指定を確認してください。

TW_FLASH_ERR_ADDRESS (H'FFFF0211)

フラッシュメモリの書込み指定アドレスが正しくないことを示します。書込みが許されない領域、または、128 バイト境界になっていないことが考えられます。アドレス指定を確認してください。

TW_FLASH_ERR_NOT_ERASED (H'FFFF0212)

未消去のフラッシュメモリ領域に書込みを行おうとしたことを示します。書込みを行う前に消去を行ってください。

□ 構造体

一部の関数では構造体を使用します。構造体は関連する関数の前に説明を記述しています。関数同様 C/C++、Visual Basic、VBA、C#、それぞれの言語における宣言が記載され、次にメンバの意味、構造体の説明の順になっています。

□ マルチスレッドプログラムからの呼び出しについて

ライブラリ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

TWB_Open() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_Open(TW_HANDLE *phDev, LPCSTR pUuid, DWORD Number, long Opt)
VB	Function TWB_Open(ByRef phDev As System.IntPtr, ByVal pUuid As String , ByVal Number As Integer, ByVal Opt As TWB_OPEN_OPT) As Integer
VBA	Function TWB_Open(ByRef phDev As Long, ByVal pUuid As String, ByVal Number As Long, ByVal Opt As TWB_OPEN_OPT) As Long
C#	STATUS Open(out System.IntPtr phDev, string pUuid, int Number, OPEN_OPT Opt)

phDev ：[出力]取得したハンドルの格納先
pUuid ：接続するデバイスの UUID 文字列
Number ：接続する製品の装置番号。0 を指定した場合は比較されません。
Opt ：接続オプション。以下を OR で結合

言語	値	説明
C/C++	TWB_IF_USB_FS	ホストインタフェースが USB(FS)のデバイスに接続します。
C++	TWB::OPEN_OPT::IF_USB_FS	
VB/VBA	TWB_OPEN_OPT.IF_USB_FS	
C#	TWB.OPEN_OPT.IF_SUB_FS	
C/C++	TWB_IF_USB_HS	ホストインタフェースが USB(HS)のデバイスに接続します。
C++	TWB::OPEN_OPT::IF_USB_HS	
VB/VBA	TWB_OPEN_OPT.IF_USB_HS	
C#	TWB.OPEN_OPT.IF_USB_HS	
C/C++	TWB_IF_LAN	ホストインタフェースが LAN のデバイスに接続します。
C++	TWB::OPEN_OPT::IF_LAN	
VB/VBA	TWB_OPEN_OPT.IF_LAN	
C#	TWB.OPEN_OPT.IF_LAN	
C/C++	TWB_IF_ANY	インタフェースの種類を問わずに接続します。
C++	TWB::OPEN_OPT::IF_ANY	
VB/VBA	TWB_OPEN_OPT.IF_ANY	
C#	TWB.OPEN_OPT.IF_ANY	
C/C++	TWB_MODE_FLASH	フラッシュ書換えモードのデバイスに接続します。
C++	TWB::OPEN_OPT::MODE_FLASH	
VB/VBA	TWB_OPEN_OPT.MODE_FLASH	
C#	TWB.OPEN_OPT.MODE_FLASH	
C/C++	TWB_MODE_BUS16	USB(HS)デバイスのみ指定可能です。デバイス内のマイコンと USB インタフェース IC 間のデータ転送を 16 ビット幅で行います。
C++	TWB::OPEN_OPT::MODE_BUS16	
VB/VBA	TWB_OPEN_OPT.MODE_BUS16	
C#	TWB.OPEN_OPT.MODE_BUS16	
C/C++	TWB_MODE_LEGACY	TWB_Initialize()の説明(126 ページ)で示す「動作設定に関するオプション」を全てオフにして接続します。
C++	TWB::OPEN_OPT::MODE_LEGACY	
VB/VBA	TWB_OPEN_OPT.MODE_LEGACY	
C#	TWB.OPEN_OPT.MODE_LEGACY	
C/C++	TWB_LIST_UPDATE	LAN デバイスのみ指定可能です。ローカルネットワーク内のデバイスを検索し、結果をライブラリ内部のテーブルに登録します。
C++	TWB::OPEN_OPT::LIST_UPDATE	
VB/VBA	TWB_OPEN_OPT. LIST_UPDATE	
C#	TWB.OPEN_OPT. LIST_UPDATE	

製品情報を指定してデバイスに接続します。成功すると phDev にデバイスへのハンドルが格納されます。
製品情報の設定方法は製品のユーザーズマニュアルを参照してください。

UUID を指定しない場合は pUUID を Null 値にします。装置番号を指定しない場合は Number を 0 とします。

Opt 引数により接続先製品のインタフェースや動作オプションを指定することができます。
Opt 引数に TWB_MODE_FLASH(相当のオプション)を指定するとフラッシュ書換えモードとなっているデバイスに接続することができます。フラッシュ書換えモードデバイスとの接続では pUUID や Number を元にデバイスを検索することはできません。最初に見つかったデバイスと接続し、pUUID と Number を比較し、一致しない場合は TW_DEVICE_NOT_FOUND を返します。また、インタフェース指定のオプション以外は無視されます。

TWB_MODE_BUS16(相当のオプション)は USB(HS)デバイスのみ指定可能なオプションです。マイコンと USB インタフェース IC とのデータ転送を 16 ビット幅で行いますので、送受信のスループットが上がります。ただし、P40～P47 端子は D0～D7 のデータバスとして使用されますので、入出力ポートとして使用できなくなります。

TWB_MODE_LEGACY(相当のオプション)を指定すると、TWB_Initialize()関数(126 ページ)の「動作設定に関するオプション」を全てオフにして接続します。

LAN デバイスへの接続では、プログラムが起動後初めて関数を呼び出す場合に、TWB_LIST_UPDATE(相当のオプション)を指定する必要があります。このオプションにより、ローカルネットワーク内のデバイスが検索され接続可能なデバイスを列挙した内部テーブルが更新されます。

TWB_Close() USB LAN

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

hDev : デバイスのハンドル

ハンドルをクローズし、デバイスの操作を終了します。

TWB_CloseAll() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_CloseAll()
VB	Function TWB_CloseAll() As Integer
VBA	Function TWB_CloseAll() As Long
C#	STATUS CloseAll()

プロセスが接続しているデバイス全てをクローズします。

TWB_OpenByAddress() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_OpenByAddress(TW_HANDLE *phDev, LPCTSTR Address, long Opt)
VB	Function TWB_OpenByAddress(ByRef phDev As System.IntPtr, ByVal Address As String, ByVal Opt As TWB_OPEN_OPT) As Integer
VBA	Function TWB_OpenByAddress(ByRef phDev As Long, ByVal Address As String, ByVal Opt As TWB_OPEN_OPT) As Long
C#	STATUS OpenByAddress(out System.IntPtr phDev, string Address, OPEN_OPT Opt)

phDev　：[出力]取得したハンドルの格納先
Address：接続する製品の IP アドレス、ドメイン名、または、USB シリアル
Opt　　：接続オプション。以下を OR で結合

言語	値	説明
C/C++	TWB_IF_USB_FS	ホストインタフェースが USB(FS)のデバイスに接続します。
C++	TWB::OPEN_OPT::IF_USB_FS	
VB/VBA	TWB_OPEN_OPT.IF_USB_FS	
C#	TWB.OPEN_OPT.IF_SUB_FS	
C/C++	TWB_IF_USB_HS	ホストインタフェースが USB(HS)のデバイスに接続します。
C++	TWB::OPEN_OPT::IF_USB_HS	
VB/VBA	TWB_OPEN_OPT.IF_USB_HS	
C#	TWB.OPEN_OPT.IF_USB_HS	
C/C++	TWB_IF_LAN	ホストインタフェースが LAN のデバイスに接続します。
C++	TWB::OPEN_OPT::IF_LAN	
VB/VBA	TWB_OPEN_OPT.IF_LAN	
C#	TWB.OPEN_OPT.IF_LAN	
C/C++	TWB_MODE_FLASH	フラッシュ書換えモードのデバイスに接続します。
C++	TWB::OPEN_OPT::MODE_FLASH	
VB/VBA	TWB_OPEN_OPT.MODE_FLASH	
C#	TWB.OPEN_OPT.MODE_FLASH	
C/C++	TWB_MODE_BUS16	USB(HS)デバイスのみ指定可能です。デバイス内のマイコンと USB インタフェース IC 間のデータ転送を 16 ビット幅で行います。
C++	TWB::OPEN_OPT::MODE_BUS16	
VB/VBA	TWB_OPEN_OPT.MODE_BUS16	
C#	TWB.OPEN_OPT.MODE_BUS16	
C/C++	TWB_MODE_LEGACY	TWB_Initialize()の説明(126 ページ)で示す「動作設定に関するオプション」を全てオフにして接続します。
C++	TWB::OPEN_OPT::MODE_LEGACY	
VB/VBA	TWB_OPEN_OPT.MODE_LEGACY	
C#	TWB.OPEN_OPT.MODE_LEGACY	

LAN デバイスへの接続では IP アドレスやドメインを指定してデバイスに接続します。通常、デバイスのポート番号(デフォルトで 49152)を指定する必要はありませんが､NAT などによる変換で指定が必要な場合には、"ドメイン名:ポート番号"のようにアドレス指定に続けて":"(コロン)とポート番号を指定します。

LAN デバイスに接続する場合で TWB_MODE_FLASH(相当のオプション)を指定し、フラッシュ書換えモードのデバイスに接続をする場合､Address に指定された IP アドレスをデバイスに一時的に割り当てて接続を確立します。Address が NULL 値の場合は DHCP によるアドレス割り当てを試みます。

USB デバイスへの接続では､デバイス固有の USB シリアルを指定して接続します｡USB シリアルは｢USBMTools｣の「ReadID」で調べることができます。

その他、オプションについては TWB_Open()関数を参照してください。

TWB_Listen() LAN

言語	関数宣言
C/C++	TW_STATUS TWB_Listen(UINT_PTR *pListenSocket, LPCTSTR pLocalIP, DWORD PortNumber)
VB	Function TWB_Listen(ByRef pListenSocket As System.IntPtr, ByVal pLocalIP As String, ByVal PortNumber As Integer) As Integer
VBA	Function TWB_Listen(ByRef pListenSocket As Long, ByVal pLocalIP As String, ByVal PortNumber As Long) As Long
C#	STATUS Listen(out System.IntPtr pListenSocket, string pLocalIP, int PortNumber)

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

指定番号の TCP ポートをオープンし、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 TWB_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インタフェースが複数ある場合に、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 値を渡すことができます。

TWB_Accept() LAN

言語	関数宣言
C/C++	TW_STATUS TWB_Accept(UINT_PTR ListenSocket, TW_HANDLE *phDev, long Opt)
VB	Function TWB_Accept(ByVal ListenSocket As System.IntPtr, ByRef phDev As System.IntPtr, ByVal Opt As TWB_ACCEPT_OPT) As Integer
VBA	Function TWB_Accept(ByVal ListenSocket As Long, ByRef phDev As Long, ByVal Opt As TWB_ACCEPT_OPT) As Long
C#	STATUS Accept(System.IntPtr ListenSocket, out System.IntPtr phDev, ACCEPT_OPT Opt) STATUS Accept(System.IntPtr ListenSocket, out System.IntPtr phDev)

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
Opt : 接続オプション。以下の値を指定できます。指定が無い場合 0 としてください

言語	値	説明
C/C++	TWB_MODE_LEGACY	TWB_Initialize()の説明(126 ページ)で示す「動作設定に関するオプション」を全てオフにして接続します。
C++	TWB::ACCEPT_OPT::MODE_LEGACY	
VB/VBA	TWB_ACCEPT_OPT.MODE_LEGACY	
C#	TWB.ACCEPT_OPT.MODE_LEGACY	

TWB_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。

TWB_CloseListenSocket() LAN

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

ListenSocket : 接続待ちソケット

TWB_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。

□ ポート操作／メモリ操作関数

TWB_PortWrite() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_PortWrite(TW_HANDLE hDev, DWORD Port, BYTE Data, BYTE Mask)
VB	Function TWB_PortWrite(ByVal hDev As System.IntPtr, ByVal Port As TWB_WPORT, ByVal Data As Byte, ByVal Mask As Byte) As Integer
VBA	Function TWB_PortWrite(ByVal hDev As Long, ByVal Port As TWB_WPORT, ByVal Data As Byte, ByVal Mask As Byte) As Long
C#	STATUS PortWrite(System.IntPtr hDev, WPORT Port, byte Data, byte Mask) STATUS PortWrite(System.IntPtr hDev, WPORT Port, byte Data) STATUS PortWrite(System.IntPtr hDev, uint Port, byte Data, byte Mask) STATUS PortWrite(System.IntPtr hDev, uint Port, byte Data)

hDev ： デバイスのハンドル
Port ： 書込み先の指定。以下の値またはアドレスを指定します

言語	値	説明
C/C++	TWB_P4	P40-P47 出力を変更します。
C++	TWB::WPORT::P4	
VB/VBA	TWB_WPORT.P4	
C#	TWB.WPORT.P4	
C/C++	TWB_PA	PA0-PA7 出力を変更します。
C++	TWB::WPORT::PA	
VB/VBA	TWB_WPORT.PA	
C#	TWB.WPORT.PA	
C/C++	TWB_POUT	POUT0#-POUT7#出力を変更します。
C++	TWB::WPORT::POUT	
VB/VBA	TWB_WPORT.POUT	
C#	TWB.WPORT.POUT	
C/C++	TWB_DA0	DA0 出力を変更します。
C++	TWB::WPORT::DA0	
VB/VBA	TWB_WPORT.DA0	
C#	TWB.WPORT.DA0	
C/C++	TWB_DA1	DA1 出力を変更します。
C++	TWB::WPORT::DA1	
VB/VBA	TWB_WPORT.DA1	
C#	TWB.WPORT.DA1	
C/C++	TWB_USER_STATUS	ユーザー用ステータスレジスタを変更します。
C++	TWB::WPORT::USER_STATUS	
VB/VBA	TWB_WPORT.USER_STATUS	
C#	TWB.WPORT.USER_STATUS	

Data ： 書き込むデータ
Mask ： 操作するビットを指定するマスク(1 と対応するビットのみ影響を受ける)

デジタル出力、アナログ出力の変更に使用します。また、ユーザー用ステータスレジスタやユーザーメモリなどの領域への書込みにも使用します。
Mask 引数は特定のビットだけを操作する場合に使用します。Mask 引数が 1 となっているビットのみ書込みの影響を受けます。例えば POUT7#だけを操作する場合、Mask = 0x80 とします。全てのビットを操作する場合 Mask = 0xff です。

TWB_PortRead() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_PortRead(TW_HANDLE hDev, DWORD Port, BYTE *pData)
VB	Function TWB_PortRead(ByVal hDev As System.IntPtr, ByVal Port As TWB_RPORT, ByRef pData As Byte) As Integer
VBA	Function TWB_PortRead(ByVal hDev As Long, ByVal Port As TWB_RPORT, ByRef pData As Byte) As Long
C#	STATUS PortRead(System.IntPtr hDev, RPORT Port, out byte pData) STATUS PortRead(System.IntPtr hDev, uint Port, out byte pData)

hDev ： デバイスのハンドル
Port ： 読出し対象を指定。以下の値またはアドレスを指定します

言語	値	説明
C/C++	TWB_P1	P10～P17 入力を読み取ります。
C++	TWB::RPORT::P1	
VB/VBA	TWB_RPORT.P1	
C#	TWB.RPORT.P1	
C/C++	TWB_P2	P20～P27 入力を読み取ります。
C++	TWB::RPORT::P2	
VB/VBA	TWB_RPORT.P2	
C#	TWB.RPORT.P2	
C/C++	TWB_P4	P40～P47 入力、または、出力値を読み取ります。
C++	TWB::RPORT::P4	
VB/VBA	TWB_RPORT.P4	
C#	TWB.RPORT.P4	
C/C++	TWB_P5	P50～P53 入力を読み取ります。
C++	TWB::RPORT::P5	
VB/VBA	TWB_RPORT.P5	
C#	TWB.RPORT.P5	
C/C++	TWB_PA	PA0～PA7 入力、または、出力値を読み取ります。
C++	TWB::RPORT::PA	
VB/VBA	TWB_RPORT.PA	
C#	TWB.RPORT.PA	
C/C++	TWB_POUT	POUT0#～POUT7#の出力値を読み取ります。
C++	TWB::RPORT::POUT	
VB/VBA	TWB_RPORT.POUT	
C#	TWB.RPORT.POUT	
C/C++	TWB_DA0	DA0 の出力値を読み取ります。
C++	TWB::RPORT::DA0	
VB/VBA	TWB_RPORT.DA0	
C#	TWB.RPORT.DA0	
C/C++	TWB_DA1	DA1 の出力値を読み取ります。
C++	TWB::RPORT::DA1	
VB/VBA	TWB_RPORT.DA1	
C#	TWB.RPORT.DA1	
C/C++	TWB_USER_STATUS	ユーザー用ステータスレジスタの値を読み取ります。
C++	TWB::RPORT::USER_STATUS	
VB/VBA	TWB_RPORT.USER_STATUS	
C#	TWB.RPORT.USER_STATUS	

pData ： [出力]読み出した値の格納先

デジタル入力値の読出しに使用します。また、デジタル出力、アナログ出力の出力値、ユーザー用ステータスレジスタやユーザーメモリなどの領域の読出しにも使用します。

TWB_PortSetDir() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_PortSetDir(TW_HANDLE hDev, DWORD Port, BYTE Dir)
VB	Function TWB_PortSetDir(ByVal hDev As System.IntPtr, ByVal Port As TWB_RWPORT, ByVal Dir As Byte) As Integer
VBA	Function TWB_PortSetDir(ByVal hDev As Long, ByVal Port As TWB_RWPORT, ByVal Dir As Byte) As Long
C#	STATUS PortSetDir(System.IntPtr hDev, RWPORT Port, byte Dir)

hDev : デバイスのハンドル
Port : 入出力方向を変更するポート。以下のいずれかの値を指定します

言語	値	説明
C/C++	TWB_P4	P40～P47 の入出力方向を変更します。
C++	TWB::RWPORT::P4	
VB/VBA	TWB_RWPORT.P4	
C#	TWB.RWPORT.P4	
C/C++	TWB_PA	PA0～PA7 の入出力方向を変更します。
C++	TWB::RWPORT::PA	
VB/VBA	TWB_RWPORT.PA	
C#	TWB.RWPORT.PA	

Dir : 方向の指定。0 のビットと対応する端子は入力、1 のビットと対応する端子は出力となります

入力、出力どちらにも設定できる端子の方向を設定します。

TWB_PortWrite16() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_PortWrite16(TW_HANDLE hDev, DWORD Port, WORD wData)
VB	Function TWB_PortWrite16(ByVal hDev As System.IntPtr, ByVal Port As Integer, ByVal wData As UShort) As Integer
VBA	Function TWB_PortWrite16(ByVal hDev As Long, ByVal Port As Long, ByVal wData As Integer) As Long
C#	STATUS PortWrite16(System.IntPtr hDev, uint Port, short wData) STATUS PortWrite16(System.IntPtr hDev, uint Port, ushort wData)

hDev : デバイスのハンドル
Port : 書込みを行うアドレス
wData : 書込みデータ

指定のアドレスのメモリ、レジスタ、外部バスなどに 16 ビット単位で書込みを行います。
書込み対象が 16 ビットアクセス可能であれば 16 ビットでのアクセスが行われます。

TWB_PortRead16() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_PortRead16(TW_HANDLE hDev, DWORD Port, WORD *pwData)
VB	Function TWB_PortRead16(ByVal hDev As System.IntPtr, ByVal Port As Integer, ByRef pwData As UShort) As Integer
VBA	Function TWB_PortRead16(ByVal hDev As Long, ByVal Port As Long, ByRef pwData As Integer) As Long
C#	STATUS PortRead16(System.IntPtr hDev, uint Port, out short pwData) STATUS PortRead16(System.IntPtr hDev, uint Port, out ushort pwData)

hDev　: デバイスのハンドル
Port　: 読出しを行うアドレス
pwData : [出力]読み出したデータの格納先

指定のアドレスのメモリ、レジスタ、外部バスなどから 16 ビット単位でデータを読出します。
読出し対象が 16 ビットアクセス可能であれば 16 ビットでのアクセスが行われます。

TWB_PortBWrite() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_PortBWrite(TW_HANDLE hDev, DWORD Port, void *pData, long nData, long Inc, long Dma)
VB	Function TWB_PortBWrite(ByVal hDev As System.IntPtr, ByVal Port As Integer, ByVal pData As Object, ByVal nData As Integer, ByVal Inc As Integer, ByVal Dma As Integer) As Integer
VBA	Function TWB_PortBWrite(ByVal hDev As Long, ByVal Port As Long, ByRef pData As Any, ByVal nData As Long, ByVal Inc As Long, ByVal Dma As Long) As Long
C#	STATUS PortBWrite(System.IntPtr hDev, uint Port, object pData, int nData, int Inc, int Dma) STATUS PortBWrite(System.IntPtr hDev, uint Port, object pData, int nData)

hDev　: デバイスのハンドル
Port　: 書き込むメモリのアドレス
pData : [入力]書き込むデータ
nData : 書き込むバイト数
Inc　: 書込み先のアドレスのインクリメントを指定します
　　0 : 書込み先アドレスを固定します。FIFO などにアクセスするときに使用します
　　1 : 書込み先アドレスは自動的にインクリメントされます
Dma　: 0 以外を指定するとデバイス内部のデータ転送に DMA を使用します

デバイスの内部メモリや、外部バスに対してデータを書き込みます。
DMA を使用するとデバイス内部のデータ転送速度が向上します。ただし、USB(FS)デバイスの場合、転送終了時に PA0/TCLKA 端子に TEND#信号として"Lo"が出力されるので注意してください。

TWB_PortBRead() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_PortBRead(TW_HANDLE hDev, DWORD Port, void *pData, long nData, long Inc, long Dma)
VB	Function TWB_PortBRead(ByVal hDev As System.IntPtr, ByVal Port As Integer, ByVal pData As Object, ByVal nData As Integer, ByVal Inc As Integer, ByVal Dma As Integer) As Integer
VBA	Function TWB_PortBRead(ByVal hDev As Long, ByVal Port As Long, ByRef pData As Any, ByVal nData As Long, ByVal Inc As Long, ByVal Dma As Long) As Long
C#	STATUS PortBRead(System.IntPtr hDev, uint Port, object pData, int nData, int Inc, int Dma) STATUS PortBRead(System.IntPtr hDev, uint Port, object pData, int nData)

hDev ： デバイスのハンドル
Port ： 読み出すメモリのアドレス
pData ： [出力]読み出したデータの格納先
nData ： 読み出すバイト数
Inc ： 読出しアドレスのインクリメントを指定します
0 ： 読出しアドレスを固定します。FIFO などにアクセスするときに使用します
1 ： 読出しアドレスは自動的にインクリメントされます
Dma ： 0 以外を指定するとデバイス内部のデータ転送に DMA を使用します

デバイスの内部メモリや、外部バスからデータを読み出します。
DMA を使用するとデバイス内部のデータ転送速度が向上します。ただし、USB(FS)デバイスの場合、転送終了時に PA1/TCLKB 端子に TEND#信号として"Lo"が出力されるので注意してください。

□ バス制御関数

TWB_BusEnableAddress() USB LAN

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

hDev ： デバイスのハンドル
nBits ： アドレスとして出力するビット数を指定します(0～20)

出力するアドレスのビット数を指定します。
A0 から順に指定されたビット数が出力されます。例えば 16 ビット出力する場合は A0～A15 が出力されます。
アドレス出力に指定した端子は入力ポートとしては使用できません。

TWB_BusEnableCS() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_BusEnableCS(TW_HANDLE hDev, long AreaBits);
VB	Function TWB_BusEnableCS(ByVal hDev As System.IntPtr, ByVal AreaBits As TWB_BUS_AREA) As Integer
VBA	Function TWB_BusEnableCS(ByVal hDev As Long, ByVal AreaBits As TWB_BUS_AREA) As Long
C#	STATUS BusEnableCS(System.IntPtr hDev, BUS_AREA AreaBits)

hDev ： デバイスのハンドル
AreaBits ： CS#信号を出力する外部バスエリア。下記の値を OR で結合します

言語	値	説明
C/C++	TWB_BUS_AREA2	CS2#を出力します。
C++	TWB::BUS_AREA::AREA2	
VB/VBA	TWB_BUS_AREA.AREA2	
C#	TWB.BUS_AREA.AREA2	
C/C++	TWB_BUS_AREA3	CS3#を出力します。
C++	TWB::BUS_AREA::AREA3	
VB/VBA	TWB_BUS_AREA.AREA3	
C#	TWB.BUS_AREA.AREA3	

CS2#、CS3#の信号を出力する場合に使用します。
AreaBits で指定された CS#信号は出力され、それ以外は禁止されます。CS#信号として出力した端子はパルスカウンタ入力としては使用できません。
CS0#、CS5#はこの関数の影響を受けず常に出力可能です。

TWB_BusSetWidth16() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_BusSetWidth16(TW_HANDLE hDev, long AreaBits)
VB	Function TWB_BusSetWidth16(ByVal hDev As System.IntPtr, ByVal AreaBits As TWB_BUS_AREA) As Integer
VBA	Function TWB_BusSetWidth16(ByVal hDev As Long, ByVal AreaBits As TWB_BUS_AREA) As Long
C#	STATUS BusSetWidth16(System.IntPtr hDev, BUS_AREA AreaBits)

hDev　　：デバイスのハンドル
AreaBits：アクセスバス幅を 16 ビットとする外部バスエリア。以下の値を OR で結合します

言語	値	説明
C/C++	TWB_BUS_AREA0	エリア 0 を 16 ビットアクセス空間とします。
C++	TWB::BUS_AREA::AREA0	
VB/VBA	TWB_BUS_AREA.AREA0	
C#	TWB.BUS_AREA.AREA0	
C/C++	TWB_BUS_AREA2	エリア 2 を 16 ビットアクセス空間とします。
C++	TWB::BUS_AREA::AREA2	
VB/VBA	TWB_BUS_AREA.AREA2	
C#	TWB.BUS_AREA.AREA2	
C/C++	TWB_BUS_AREA3	エリア 3 を 16 ビットアクセス空間とします。
C++	TWB::BUS_AREA::AREA3	
VB/VBA	TWB_BUS_AREA.AREA3	
C#	TWB.BUS_AREA.AREA3	
C/C++	TWB_BUS_AREA5	エリア 5 を 16 ビットアクセス空間とします。
C++	TWB::BUS_AREA::AREA5	
VB/VBA	TWB_BUS_AREA.AREA5	
C#	TWB.BUS_AREA.AREA5	

指定の外部バスエリアを 16 ビットアクセス空間とします。
16 ビットアクセス空間とすると、奇数番地のアクセスにデータバスの D0～D7 が使用されます。16 ビットアクセス空間が 1 つでも存在すると、P40～P47 の端子は入出力ポートとしては利用できなくなります。

TWB_BusSetWait() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_BusSetWait(TW_HANDLE hDev, long AreaBits, long Wait)
VB	Function TWB_BusSetWait(ByVal hDev As System.IntPtr, ByVal AreaBits As TWB_BUS_AREA, ByVal Wait As TWB_BUS_WAIT) As Integer
VBA	Function TWB_BusSetWait(ByVal hDev As Long, ByVal AreaBits As TWB_BUS_AREA, ByVal Wait As TWB_BUS_WAIT) As Long
C#	STATUS BusSetWait(System.IntPtr hDev, BUS_AREA AreaBits, BUS_WAIT Wait)

hDev　　：デバイスのハンドル
AreaBits：ウェイトを指定する外部バスエリア。以下の値を OR で結合します

言語	値	説明
C/C++	TWB_BUS_AREA0	エリア 0 のアクセスウェイトを指定します。
C++	TWB::BUS_AREA::AREA0	
VB/VBA	TWB_BUS_AREA.AREA0	
C#	TWB.BUS_AREA.AREA0	
C/C++	TWB_BUS_AREA2	エリア 2 のアクセスウェイトを指定します。
C++	TWB::BUS_AREA::AREA2	
VB/VBA	TWB_BUS_AREA.AREA2	
C#	TWB.BUS_AREA.AREA2	
C/C++	TWB_BUS_AREA3	エリア 3 のアクセスウェイトを指定します。
C++	TWB::BUS_AREA::AREA3	
VB/VBA	TWB_BUS_AREA.AREA3	
C#	TWB.BUS_AREA.AREA3	
C/C++	TWB_BUS_AREA5	エリア 5 のアクセスウェイトを指定します。
C++	TWB::BUS_AREA::AREA5	
VB/VBA	TWB_BUS_AREA.AREA5	
C#	TWB.BUS_AREA.AREA5	

Wait　：ウェイトの指定。以下のいずれかの値を指定します

言語	値	説明
C/C++	TWB_BUS_WAIT0	3 ステートアクセスのウェイト 0 に設定します。
C++	TWB::BUS_WAIT::WAIT0	
VB/VBA	TWB_BUS_WAIT.WAIT0	
C#	TWB.BUS_WAIT.WAIT0	
C/C++	TWB_BUS_WAIT1	3 ステートアクセスのウェイト 1 に設定します。
C++	TWB::BUS_WAIT::WAIT1	
VB/VBA	TWB_BUS_WAIT.WAIT1	
C#	TWB.BUS_WAIT.WAIT1	
C/C++	TWB_BUS_WAIT2	3 ステートアクセスのウェイト 2 に設定します。
C++	TWB::BUS_WAIT::WAIT2	
VB/VBA	TWB_BUS_WAIT.WAIT2	
C#	TWB.BUS_WAIT.WAIT2	
C/C++	TWB_BUS_WAIT3	3 ステートアクセスのウェイト 3 に設定します。
C++	TWB::BUS_WAIT::WAIT3	
VB/VBA	TWB_BUS_WAIT.WAIT3	
C#	TWB.BUS_WAIT.WAIT3	
C/C++	TWB_BUS_2STATE	2 ステートアクセスに設定します。最もアクセス速度が速い設定です。
C++	TWB::BUS_WAIT::STATE2	
VB/VBA	TWB_BUS_WAIT.STATE2	
C#	TWB.BUS_WAIT.STATE2	

外部バスエリアのソフトウェアウェイトを設定します。
ウェイトが0～3の範囲では該当エリアは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 に TWB_BUS_2STATE(相当の値)を指定すると、該当エリアは 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 2 ステートアクセスが最も高速で、それ以外はウェイト数が大きいほど低速です。初期状態では全てのエリアがウェイト 1 に設定されています。
バスのアクセスタイミングの詳細は、搭載マイコンのハードウェアマニュアルを参照してください。

□ アナログ入力／アナログ値変換関数

TWB_ADRead() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_ADRead(TW_HANDLE hDev, long Ch, long *pData)
VB	Function TWB_ADRead(ByVal hDev As System.IntPtr, ByVal Ch As Integer, ByRef pData As Integer) As Integer
VBA	Function TWB_ADRead(ByVal hDev As Long, ByVal Ch As Long, ByRef pData As Long) As Long
C#	STATUS ADRead(System.IntPtr hDev, int Ch, out int pData)

hDev ： デバイスのハンドル
Ch ： 入力するチャンネル(0,1,2,3)
pData ： [出力]AD 変換結果の格納先

指定チャンネルを AD 変換した結果を読み出します。変換結果は上位 16 ビットが常に 0 で、下位 16 ビットの MSB から 10 ビットに値が格納されます。値の範囲は 16 進数で 0x0000-0xffc0、10 進数で 0-65472 の範囲となります。

ビット	31-16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
値	常に 0	AD 変換結果										常に 0					

TWB_An16ToVolt() USB LAN

言語	関数宣言
C/C++	double TWB_An16ToVolt(long Data, long Opt)
VB	Function TWB_An16ToVolt (ByVal Data As Integer, ByVal Opt As Integer) As Double
VBA	Function TWB_An16ToVolt(ByVal Data As Long,ByVal Opt As Long) As Double
C#	double An16ToVolt(int Data)

Data ： AD 変換で得られた値
Opt ： 予約。0 としてください

戻り値 ： Data を電圧に換算した値(ボルト単位)

TWB_ADRead()で得られた AD 変換の結果をボルト単位に変換して返します。

TWB_An8FromVolt() USB LAN

言語	関数宣言
C/C++	BYTE TWB_An8FromVolt(double *pData, long Opt)
VB	Function TWB_An8FromVolt(ByRef pData As Double, ByVal Opt As Integer) As Byte
VBA	Function TWB_An8FromVolt(ByRef pData As Double, ByVal Opt As Long) As Byte
C#	byte An8FromVolt(ref double pData)

pData ：[入力]DA コンバータの設定値に変換したい電圧値(ボルト単位)
[出力]DA コンバータの精度で表現可能な電圧の理論値(ボルト単位)
Opt ：予約。0 としてください

戻り値 ：pData を DA への書込み値に変換した値

ボルト単位の電圧値から DA コンバータに書き込む値を計算して返します。DA コンバータの精度で表現可能な理論値を pData に返します。

□ パルスカウンタ（ソフトウェアカウンタ）操作関数

TWB_PCSetMode() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_PCSetMode(TW_HANDLE hDev, long ChBits, long Mode)
VB	Function TWB_PCSetMode(ByVal hDev As System.IntPtr, ByVal ChBits As TWB_PC_BITS, ByVal Mode As TWB_PC_MODE) As Integer
VBA	Function TWB_PCSetMode(ByVal hDev As Long, ByVal ChBits As TWB_PC_BITS, ByVal Mode As TWB_PC_MODE) As Long
C#	STATUS PCSetMode(System.IntPtr hDev, PC_BITS ChBits, PC_MODE Mode)

hDev　：デバイスのハンドル
ChBits：設定チャンネル。以下の値のいずれか

言語	値	説明
C/C++	TWB_PC0	PC0 の動作を設定します。Mode 指定が 2 相カウント以外の場合指定可能です。
C++	TWB::PC_BITS::PC0	
VB/VBA	TWB_PC_BITS.PC0	
C#	TWB.PC_BITS.PC0	
C/C++	TWB_PC1	PC1 の動作を設定します。Mode 指定が 2 相カウント以外の場合指定可能です。
C++	TWB::PC_BITS::PC1	
VB/VBA	TWB_PC_BITS.PC1	
C#	TWB.PC_BITS.PC1	
C/C++	TWB_PC2	PC2 の動作を設定します。Mode 指定が 2 相カウント以外の場合指定可能です。
C++	TWB::PC_BITS::PC2	
VB/VBA	TWB_PC_BITS.PC2	
C#	TWB.PC_BITS.PC2	
C/C++	TWB_PC3	PC3 の動作を設定します。Mode 指定が 2 相カウント以外の場合指定可能です。
C++	TWB::PC_BITS::PC3	
VB/VBA	TWB_PC_BITS.PC3	
C#	TWB.PC_BITS.PC3	
C/C++	TWB_PC0_PC1	PC0 と PC1 の動作設定をします。
C++	TWB::PC_BITS::PC0_PC1	
VB/VBA	TWB_PC_BITS.PC0_PC1	
C#	TWB.PC_BITS.PC0_PC1	
C/C++	TWB_PC2_PC3	PC2 と PC3 の動作設定をします。
C++	TWB::PC_BITS::PC2_PC3	
VB/VBA	TWB_PC_BITS.PC2_PC3	
C#	TWB.PC_BITS.PC2_PC3	
C/C++	TWB_PC_ALL	全てのチャンネルを同じ動作設定にします。Mode 指定が 2 相カウント以外の場合指定可能です。
C++	TWB::PC_BITS::PC_ALL	
VB/VBA	TWB_PC_BITS.PC_ALL	
C#	TWB.PC_BITS.PC_ALL	

Mode　: カウントモード。以下の値のいずれか

言語	値	説明
C/C++	TWB_PC_SINCLE	入力が"Hi"から"Lo"に変化したときにカウントします。
C++	TWB::PC_MODE::COUNT_SINGLE	
VB/VBA	TWB_PC_MODE.COUNT_SINGLE	
C#	TWB.PC_MODE.COUNT_SINGLE	
C/C++	TWB_PC_2PHASE_E	90° 位相差の A 相、B 相の 2 相信号をカウントするモードです。36 ページ、図 8 の推奨接続用です。
C++	TWB::PC_MODE::COUNT_2PHASE_E	
VB/VBA	TWB_PC_MODE.COUNT_2PHASE_E	
C#	TWB.PC_MODE.COUNT_2PHASE_E	
C/C++	TWB_PC_3PHASE_E	PC2#と PC3#で 2 相信号のカウントを行い、PC0#でカウンタをクリアします。PC2#と PC3#は 36 ページ、図 8 の推奨接続を使用します。ChBits の値は無視されます。
C++	TWB::PC_MODE::COUNT_3PHASE_E	
VB/VBA	TWB_PC_MODE.COUNT_3PHASE_E	
C#	TWB.PC_MODE.COUNT_3PHASE_E	
C/C++	TWB_PC_2PHASE	90° 位相差の A 相、B 相の 2 相信号をカウントするモードです。簡易接続(42 ページ、図 11)の場合に指定します。
C++	TWB::PC_MODE::COUNT_2PHASE	
VB/VBA	TWB_PC_MODE.COUNT_2PHASE	
C#	TWB_PC_MODE.COUNT_2PHASE	
C/C++	TWB_PC_3PHASE	PC2#と PC3#で 2 相信号のカウントを行い、PC0#でカウンタをクリアします。PC2#と PC3#を簡易接続(42 ページ、図 11)とした場合に指定します。ChBits の値は無視されます。
C++	TWB::PC_MODE::COUNT_3PHASE	
VB/VBA	TWB_PC_MODE.COUNT_3PHASE	
C#	TWB.PC_MODE.COUNT_3PHASE	

パルスカウンタの指定チャンネルのカウントモードを設定します。
接続方法は 36 ページを参照してください。

TWB_PCStart() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_PCStart(TW_HANDLE hDev, long ChBits)
VB	Function TWB_PCStart(ByVal hDev As System.IntPtr, ByVal ChBits As TWB_PC_BITS) As Integer
VBA	Function TWB_PCStart(ByVal hDev As Long, ByVal ChBits As TWB_PC_BITS) As Long
C#	STATUS PCStart(System.IntPtr hDev, PC_BITS ChBits)

hDev　: デバイスのハンドル
ChBits : パルスカウンタのチャンネル。以下の値を OR で結合

言語	値	説明
C/C++	TWB_PC0	PC0 のカウントを開始します。
C++	TWB::PC_BITS::PC0	
VB/VBA	TWB_PC_BITS.PC0	
C#	TWB_PC_BITS.PC0	
C/C++	TWB_PC1	PC1 のカウントを開始します。
C++	TWB::PC_BITS::PC1	
VB/VBA	TWB_PC_BITS.PC1	
C#	TWB.PC_BITS.PC1	
C/C++	TWB_PC2	PC2 のカウントを開始します。
C++	TWB::PC_BITS::PC2	
VB/VBA	TWB_PC_BITS.PC2	
C#	TWB.PC_BITS.PC2	
C/C++	TWB_PC3	PC3 のカウントを開始します。
C++	TWB::PC_BITS::PC3	
VB/VBA	TWB_PC_BITS.PC3	
C#	TWB.PC_BITS.PC3	
C/C++	TWB_PC0_PC1	PC0 と PC1 のカウントを開始します。
C++	TWB::PC_BITS::PC0_PC1	
VB/VBA	TWB_PC_BITS.PC0_PC1	
C#	TWB.PC_BITS.PC0_PC1	
C/C++	TWB_PC2_PC3	PC2 と PC3 のカウントを開始します。
C++	TWB::PC_BITS::PC2_PC3	
VB/VBA	TWB_PC_BITS.PC2_PC3	
C#	TWB.PC_BITS.PC2_PC3	
C/C++	TWB_PC_ALL	全てのチャンネルのカウントを開始します。
C++	TWB::PC_BITS::PC_ALL	
VB/VBA	TWB_PC_BITS.PC_ALL	
C#	TWB.PC_BITS.PC_ALL	

指定チャンネルのパルスカウンタのカウントをスタートさせます。

TWB_PCStop() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_PCStop(TW_HANDLE hDev, long ChBits)
VB	Function TWB_PCStop(ByVal hDev As System.IntPtr, ByVal ChBits As TWB_PC_BITS) As Integer
VBA	Function TWB_PCStop(ByVal hDev As Long, ByVal ChBits As TWB_PC_BITS) As Long
C#	STATUS PCStop(System.IntPtr hDev, PC_BITS ChBits)

hDev　: デバイスのハンドル
ChBits : パルスカウンタのチャンネル。以下の値を OR で結合

言語	値	説明
C/C++	TWB_PC0	PC0 のカウントを停止します。
C++	TWB::PC_BITS::PC0	
VB/VBA	TWB_PC_BITS.PC0	
C#	TWB.PC_BITS.PC0	
C/C++	TWB_PC1	PC1 のカウントを停止します。
C++	TWB::PC_BITS::PC1	
VB/VBA	TWB_PC_BITS.PC1	
C#	TWB.PC_BITS.PC1	
C/C++	TWB_PC2	PC2 のカウントを停止します。
C++	TWB::PC_BITS::PC2	
VB/VBA	TWB_PC_BITS.PC2	
C#	TWB.PC_BITS.PC2	
C/C++	TWB_PC3	PC3 のカウントを停止します。
C++	TWB::PC_BITS::PC3	
VB/VBA	TWB_PC_BITS.PC3	
C#	TWB.PC_BITS.PC3	
C/C++	TWB_PC0_PC1	PC0 と PC1 のカウントを停止します。
C++	TWB::PC_BITS::PC0_PC1	
VB/VBA	TWB_PC_BITS.PC0_PC1	
C#	TWB.PC_BITS.PC0_PC1	
C/C++	TWB_PC2_PC3	PC2 と PC3 のカウントを停止します。
C++	TWB::PC_BITS::PC2_PC3	
VB/VBA	TWB_PC_BITS.PC2_PC3	
C#	TWB.PC_BITS.PC2_PC3	
C/C++	TWB_PC_ALL	全てのチャンネルのカウントを停止します。
C++	TWB::PC_BITS::PC_ALL	
VB/VBA	TWB_PC_BITS.PC_ALL	
C#	TWB.PC_BITS.PC_ALL	

指定チャンネルのパルスカウンタのカウントを停止します。

TWB_PCReadCnt() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_PCReadCnt(TW_HANDLE hDev, long ChBits, long *pCnt)
VB	Function TWB_PCReadCnt(ByVal hDev As System.IntPtr, ByVal ChBits As TWB_PC_BITS, ByRef pCnt As Integer) As Integer Function TWB_PCReadCnt(ByVal hDev As System.IntPtr, ByVal ChBits As TWB_PC_BITS, ByVal pCnt() As Integer) As Integer
VBA	Function TWB_PCReadCnt(ByVal hDev As Long, ByVal ChBits As TWB_PC_BITS, ByRef pCnt As Long) As Long
C#	STATUS PCReadCnt(System.IntPtr hDev, PC_BITS ChBits, out int pCnt) STATUS PCReadCnt(System.IntPtr hDev, PC_BITS ChBits, out uint pCnt) STATUS PCReadCnt(System.IntPtr hDev, PC_BITS ChBits, int []pCnt) STATUS PCReadCnt(System.IntPtr hDev, PC_BITS ChBits, uint []pCnt)

hDev : デバイスのハンドル
ChBits : パルスカウンタのチャンネル。以下の値のいずれか

言語	値	説明
C/C++	TWB_PC0	PC0 のカウント値を読み出します。
C++	TWB::PC_BITS::PC0	
VB/VBA	TWB_PC_BITS.PC0	
C#	TWB.PC_BITS.PC0	
C/C++	TWB_PC1	PC1 のカウント値を読み出します。
C++	TWB::PC_BITS::PC1	
VB/VBA	TWB_PC_BITS.PC1	
C#	TWB.PC_BITS.PC1	
C/C++	TWB_PC2	PC2 のカウント値を読み出します。
C++	TWB::PC_BITS::PC2	
VB/VBA	TWB_PC_BITS.PC2	
C#	TWB.PC_BITS.PC2	
C/C++	TWB_PC3	PC3 のカウント値を読み出します。
C++	TWB::PC_BITS::PC3	
VB/VBA	TWB_PC_BITS.PC3	
C#	TWB.PC_BITS.PC3	
C/C++	TWB_PC0_PC1	PC0 と PC1 のカウント値の合計を読み出します。
C++	TWB::PC_BITS::PC0_PC1	
VB/VBA	TWB_PC_BITS.PC0_PC1	
C#	TWB.PC_BITS.PC0_PC1	
C/C++	TWB_PC2_PC3	PC2 と PC3 のカウント値の合計を読み出します。
C++	TWB::PC_BITS::PC2_PC3	
VB/VBA	TWB_PC_BITS.PC2_PC3	
C#	TWB.PC_BITS.PC2_PC3	
C/C++	TWB_PC_ALL	PC0 から PC3 までのカウント値を同時に読み出します。 pCnt には 4 チャンネル分の領域が必要です。
C++	TWB::PC_BITS::PC_ALL	
VB/VBA	TWB_PC_BITS.PC_ALL	
C#	TWB.PC_BITS.PC_ALL	

pCnt : [出力]カウント値の格納先

指定チャンネルのパルスカウンタの値を読み出します。
TWB_PC0_PC1、TWB_PC2_PC3(相当の値)を指定した場合、それぞれ、チャンネル 0 と 1 のカウンタ値の合計、チャンネル 2 と 3 のカウンタ値の合計を返します。

TWB_PC_ALL(相当の値)を指定した場合 pCnt にチャンネル 0～3 までの値が連続して格納されますので pCnt には 4 チャンネル分のデータを格納できる領域(4 x 4 = 16 バイト)が必要です。

TWB_PCSetCnt() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_PCSetCnt(TW_HANDLE hDev, long ChBits, long Cnt)
VB	Function TWB_PCSetCnt(ByVal hDev As System.IntPtr, ByVal ChBits As TWB_PC_BITS, ByVal Cnt As Integer) As Integer
VBA	Function TWB_PCSetCnt(ByVal hDev As Long, ByVal ChBits As TWB_PC_BITS, ByVal Cnt As Long) As Long
C#	STATUS PCSetCnt(System.IntPtr hDev, PC_BITS ChBits, int Cnt) STATUS PCSetCnt(System.IntPtr hDev, PC_BITS ChBits, uint Cnt)

hDev　: デバイスのハンドル
ChBits : パルスカウンタのチャンネル。以下値のいずれか

言語	値	説明
C/C++	TWB_PC0	PC0 のカウンタに値をセットします。
C++	TWB::PC_BITS::PC0	
VB/VBA	TWB_PC_BITS.PC0	
C#	TWB.PC_BITS.PC0	
C/C++	TWB_PC1	PC1 のカウンタに値をセットします。
C++	TWB::PC_BITS::PC1	
VB/VBA	TWB_PC_BITS.PC1	
C#	TWB.PC_BITS.PC1	
C/C++	TWB_PC2	PC2 のカウンタに値をセットします。
C++	TWB::PC_BITS::PC2	
VB/VBA	TWB_PC_BITS.PC2	
C#	TWB.PC_BITS.PC2	
C/C++	TWB_PC3	PC3 のカウンタに値をセットします。
C++	TWB::PC_BITS::PC3	
VB/VBA	TWB_PC_BITS.PC3	
C#	TWB.PC_BITS.PC3	
C/C++	TWB_PC0_PC1	PC0 と PC1 のカウンタに合計が Cnt となるように値をセットします。
C++	TWB::PC_BITS::PC0_PC1	
VB/VBA	TWB_PC_BITS.PC0_PC1	
C#	TWB.PC_BITS.PC0_PC1	
C/C++	TWB_PC2_PC3	PC2 と PC3 のカウンタに合計が Cnt となるように値をセットします。
C++	TWB::PC_BITS::PC2_PC3	
VB/VBA	TWB_PC_BITS.PC2_PC3	
C#	TWB.PC_BITS.PC2_PC3	

Cnt　: セットする値

指定チャンネルのパルスカウンタに値をセットします。カウンタをクリアする場合は Cnt を 0 として呼び出します。
ChBits が TWB_PC0_PC1 または TWB_PC2_PC3(相当の値)の場合、2 つのカウンタの合計値が Cnt となるように調整され 2 つのカウンタに値がセットされます。

16 ビットタイマ（ハードウェアカウンタ）操作関数

TWB_TimerSetMode() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_TimerSetMode(TW_HANDLE hDev, long Ch, long Mode)
VB	Function TWB_TimerSetMode(ByVal hDev As System.IntPtr, ByVal Ch As Integer, ByVal Mode As TWB_TIMER_MODE) As Integer
VBA	Function TWB_TimerSetMode(ByVal hDev As Long, ByVal Ch As Long, ByVal Mode As TWB_TIMER_MODE) As Long
C#	STATUS TimerSetMode(System.IntPtr hDev, int Ch, TIMER_MODE Mode)

hDev : デバイスのハンドル
Ch : タイマチャンネル(0～2)
Mode : 動作モード。以下のいずれか

言語	値	説明
C/C++	TWB_TIMER_DISABLE	PWM モードを解除する場合に指定します。全てのチャンネルに指定可能です。PWM 端子がデジタル出力端子として使用可能になります。
C++	TWB::TIMER_MODE::DISABLE	
VB/VBA	TWB_TIMER_MODE.DISABLE	
C#	TWB.TIMER_MODE.DISABLE	
C/C++	TWB_TIMER_PWM	指定チャンネルを PWM モードに設定します。全てのチャンネルに指定可能です。対応する端子は PWM 出力用となります。
C++	TWB::TIMER_MODE::PWM	
VB/VBA	TWB_TIMER_MODE.PWM	
C#	TWB.TIMER_MODE.PWM	
C/C++	TWB_TIMER_RISE	指定チャンネルをパルスカウントモードとし、対応する入力が"Lo"から"Hi"に変化したときカウントします。1、2 チャンネルのみ指定可能です。
C++	TWB::TIMER_MODE::COUNT_RISE	
VB/VBA	TWB_TIMER_MODE.COUNT_RISE	
C#	TWB.TIMER_MODE.COUNT_RISE	
C/C++	TWB_TIMER_FALL	指定チャンネルをパルスカウントモードとし、対応する入力が"Hi"から"Lo"に変化したときカウントします。1、2 チャンネルのみ指定可能です。
C++	TWB::TIMER_MODE::COUNT_FALL	
VB/VBA	TWB_TIMER_MODE.COUNT_FALL	
C#	TWB.TIMER_MODE.COUNT_FALL	
C/C++	TWB_TIMER_BOTH	指定チャンネルをパルスカウントモードとし、極性によらず対応する入力が変化したときにカウントします。1、2 チャンネルのみ指定可能です。
C++	TWB::TIMER_MODE::COUNT_BOTH	
VB/VBA	TWB_TIMER_MODE.COUNT_BOTH	
C#	TWB.TIMER_MODE.COUNT_BOTH	
C/C++	TWB_TIMER_2PHASE	90° 位相差の A 相、B 相の 2 相信号をカウントします。2 チャンネルのみ指定可能です。
C++	TWB::TIMER_MODE::COUNT_2PHASE	
VB/VBA	TWB_TIMER_MODE.COUNT_2PHASE	
C#	TWB.TIMER_MODE.COUNT_2PHASE	

16 ビットタイマの動作モードを変更します。
PWM モードに設定すると、チャンネルに対応する端子が PWM 出力用端子となりデジタル入出力端子としての制御ができなくなります。デジタル入出力端子に戻す場合は Mode 引数に TWB_TIMER_DISABLE(相当の値)を指定して呼び出してください。
単相のパルスカウントモードはチャンネル 1 と 2 のみ指定可能です。それぞれ TCLKA、TCLKB の入力パルスをカウントします。
2 相のパルスカウントモードはチャンネル 2 のみ指定可能です。TCLKA に B 相、TCLKB に A 相を接続して使用します。

TWB_TimerSetPwm() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_TimerSetPwm(TW_HANDLE hDev, long Ch, double *pFrequency, double *pDuty, double *pPhase)
VB	Function TWB_TimerSetPwm(ByVal hDev As System.IntPtr, ByVal Ch As Integer, ByRef pFrequency As Double, ByRef pDuty As Double, ByRef pPhase As Double) As Integer
VBA	Function TWB_TimerSetPwm(ByVal hDev As Long, ByVal Ch As Long, ByRef pFrequency As Double, ByRef pDuty As Double, ByRef pPhase As Double) As Long
C#	STATUS TimerSetPwm(System.IntPtr hDev, int Ch, ref double pFrequency, ref double pDuty, ref double pPhase)

hDev : デバイスのハンドル
Ch : 設定する 16 ビットタイマのチャンネル(0～2)
pFrequency : [入力]希望の周波数 [出力]実際に設定できた周波数(Hz 単位)
pDuty : [入力]希望のデューティ [出力]実際に設定できたデューティ(0-1.0)
pPhase : [入力]希望の初期位相 [出力]実際に設定できた初期位相(0-1.0)

PWM 出力の設定を行います。
デューティ(pDuty)は 0-1.0 の範囲で ON デューティを指定します。例えば 0.3 を指定した場合、周期の約 30%が"Hi"のパルスが出力されます。
初期位相(pPhase)は該当のタイマチャンネルが停止中のみ変更可能です。0-1.0 の範囲で指定します。1.0 は 360°、0.5 は 180°に相当します。
波形は製品の内部クロック(25MHz)を分周して作られるため、周波数、デューティ、初期位相の各パラメータは設定できる値が離散的になります。そのため、引数に与えた希望値と設定可能な値が異なる場合があります。関数は各引数に実際に設定できた値をセットして戻ります。

TWB_TimerSetPwmExt() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_TimerSetPwmExt(TW_HANDLE hDev, long Ch, double dClkFreq, double *pFrequency, double *pDuty, double *pPhase)
VB	Function TWB_TimerSetPwmExt(ByVal hDev As System.IntPtr, ByVal Ch As Integer, ByVal dClkFreq As Double, ByRef pFrequency As Double, ByRef pDuty As Double, ByRef pPhase As Double) As Integer
VBA	Function TWB_TimerSetPwmExt(ByVal hDev As Long, ByVal Ch As Long, ByVal dClkFreq As Double, ByRef pFrequency As Double, ByRef pDuty As Double, ByRef pPhase As Double) As Long
C#	STATUS TimerSetPwmExt(System.IntPtr hDev, int Ch, double dClkFreq, ref double pFrequency, ref double pDuty, ref double pPhase)

hDev : デバイスのハンドル
Ch : 設定する 16 ビットタイマのチャンネル(0～2)
dClkFreq : 外部クロックの周波数(Hz 単位)
pFrequency : [入力]希望の周波数 [出力]実際に設定できた周波数(Hz 単位)
pDuty : [入力]希望のデューティ [出力]実際に設定できたデューティ(0-1.0)
pPhase : [入力]希望の初期位相 [出力]実際に設定できた初期位相(0-1.0)

基準となるクロックとして外部入力を使用する点を除いて TWB_TimerSetPwm()関数と同様です。
内部クロックを用いた場合、出力できる周波数の下限が約 48Hz となりますので、それより低い周波数を出力する場合に使用してください。

外部クロックは TCLKA に入力します。別の機器からの出力信号を用いることもできますが、他のチャンネルの PWM 出力を入力することも可能です。

TWB_TimerStart() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_TimerStart(TW_HANDLE hDev, long ChBits)
VB	Function TWB_TimerStart(ByVal hDev As System.IntPtr, ByVal ChBits As TWB_TIMER_BITS) As Integer
VBA	Function TWB_TimerStart(ByVal hDev As Long, ByVal ChBits As TWB_TIMER_BITS) As Long
C#	STATUS TimerStart(System.IntPtr hDev, TIMER_BITS ChBits)

hDev　: デバイスのハンドル
ChBits : スタートする 16 ビットタイマのチャンネル。以下の値を OR で結合

言語	値	説明
C/C++	TWB_TIMER_BIT0	0 チャンネルの動作を開始します。
C++	TWB::TIMER_BITS::TIMER0	
VB/VBA	TWB_TIMER_BITS.TIMER0	
C#	TWB.TIMER_BITS.TIMER0	
C/C++	TWB_TIMER_BIT1	1 チャンネルの動作を開始します。
C++	TWB::TIMER_BITS::TIMER1	
VB/VBA	TWB_TIMER_BITS.TIMER1	
C#	TWB.TIMER_BITS.TIMER1	
C/C++	TWB_TIMER_BIT2	2 チャンネルの動作を開始します。
C++	TWB::TIMER_BITS::TIMER2	
VB/VBA	TWB_TIMER_BITS.TIMER2	
C#	TWB.TIMER_BITS.TIMER2	
C/C++	TWB_TIMER_BITS_ALL	全チャンネルの動作を開始します。
C++	TWB::TIMER_BITS::TIMER_ALL	
VB/VBA	TWB_TIMER_BITS.TIMER_ALL	
C#	TWB.TIMER_BITS.TIMER_ALL	

指定のタイマチャンネルの動作を開始します。
PWM 出力に設定されているチャンネルはパルス出力を開始し、パルスカウントモードに設定されているチャンネルはパルスのカウントを開始します。

TWB_TimerStop() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_TimerStop(TW_HANDLE hDev, long ChBits)
VB	Function TWB_TimerStop(ByVal hDev As System.IntPtr, ByVal ChBits As TWB_TIMER_BITS) As Integer
VBA	Function TWB_TimerStop(ByVal hDev As Long, ByVal ChBits As TWB_TIMER_BITS) As Long
C#	STATUS TimerStop(System.IntPtr hDev, TIMER_BITS ChBits)

hDev　: デバイスのハンドル
ChBits : 停止する 16 ビットタイマのチャンネル。以下の値を OR で結合

言語	値	説明
C/C++	TWB_TIMER_BIT0	0 チャンネルの動作を停止します。
C++	TWB::TIMER_BITS::TIMER0	
VB/VBA	TWB_TIMER_BITS.TIMER0	
C#	TWB.TIMER_BITS.TIMER0	
C/C++	TWB_TIMER_BIT1	1 チャンネルの動作を停止します。
C++	TWB::TIMER_BITS::TIMER1	
VB/VBA	TWB_TIMER_BITS.TIMER1	
C#	TWB.TIMER_BITS.TIMER1	
C/C++	TWB_TIMER_BIT2	2 チャンネルの動作を停止します。
C++	TWB::TIMER_BITS::TIMER2	
VB/VBA	TWB_TIMER_BITS.TIMER2	
C#	TWB.TIMER_BITS.TIMER2	
C/C++	TWB_TIMER_BITS_ALL	全チャンネルの動作を停止します。
C++	TWB::TIMER_BITS::TIMER_ALL	
VB/VBA	TWB_TIMER_BITS.TIMER_ALL	
C#	TWB.TIMER_BITS.TIMER_ALL	

指定のタイマチャンネルの動作を停止します。

TWB_TimerSetLevel() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_TimerSetLevel(TW_HANDLE hDev, long ChBits)
VB	Function TWB_TimerSetLevel(ByVal hDev As System.IntPtr, ByVal ChBits As TWB_TIMER_BITS) As Integer
VBA	Function TWB_TimerSetLevel(ByVal hDev As Long, ByVal ChBits As TWB_TIMER_BITS) As Long
C#	STATUS TimerSetLevel(System.IntPtr hDev, TIMER_BITS ChBits)

hDev　: デバイスのハンドル
ChBits : 出力を"Hi"とするビットを指定します。以下を OR で結合

言語	値	説明
C/C++	TWB_TIMER_BIT0	TIOCA0 を"Hi"出力にします。
C++	TWB::TIMER_BITS::TIMER0	
VB/VBA	TWB_TIMER_BITS.TIMER0	
C#	TWB.TIMER_BITS.TIMER0	
C/C++	TWB_TIMER_BIT1	TIOCA1 を"Hi"出力にします。
C++	TWB::TIMER_BITS::TIMER1	
VB/VBA	TWB_TIMER_BITS.TIMER1	
C#	TWB.TIMER_BITS.TIMER1	
C/C++	TWB_TIMER_BIT2	TIOCA2 を"Hi"出力にします。
C++	TWB::TIMER_BITS::TIMER2	
VB/VBA	TWB_TIMER_BITS.TIMER2	
C#	TWB.TIMER_BITS.TIMER2	
C/C++	TWB_TIMER_BITS_ALL	TIOCA0、TIOCA1、TIOCA2 を"Hi"出力にします。
C++	TWB::TIMER_BITS::TIMER_ALL	
VB/VBA	TWB_TIMER_BITS.TIMER_ALL	
C#	TWB.TIMER_BITS.TIMER_ALL	

PWM 出力となっている端子の状態を変更します。ChBits で指定した端子は"Hi"、その他の端子は"Lo"となります。
呼び出しはタイマの停止中に行ってください。

TWB_TimerReadStatus() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_TimerReadStatus(TW_HANDLE hDev, long *pStatus)
VB	Function TWB_TimerReadStatus(ByVal hDev As System.IntPtr, ByRef pStatus As Integer) As Integer
VBA	Function TWB_TimerReadStatus(ByVal hDev As Long, ByRef pStatus As Long) As Long
C#	STATUS TimerReadStatus(System.IntPtr hDev, out TIMER_BITS pStatus) STATUS TimerReadStatus(System.IntPtr hDev, out int pStatus)

hDev　: デバイスのハンドル
pStatus : [出力]タイマの状態
　ビット０ : ０チャンネルが動作状態のとき１となります
　ビット１ : １チャンネルが動作状態のとき１となります
　ビット２ : ２チャンネルが動作状態のとき１となります

タイマの動作状態を返します。動作中のチャンネルは pStatus の対応するビットが１になります。

TWB_TimerReadCnt() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_TimerReadCnt(TW_HANDLE hDev, long Ch, short *pCnt)
VB	Function TWB_TimerReadCnt(ByVal hDev As System.IntPtr, ByVal Ch As Integer, ByRef pCnt As Short) As Integer Function TWB_TimerReadCnt(ByVal hDev As System.IntPtr, ByVal Ch As Integer, ByRef pCnt As System.UInt16) As Integer
VBA	Function TWB_TimerReadCnt(ByVal hDev As Long, ByVal Ch As Long, ByRef pCnt As Integer) As Long
C#	STATUS TimerReadCnt(System.IntPtr hDev, int Ch, out short pCnt) STATUS TimerReadCnt(System.IntPtr hDev, int Ch, out ushort pCnt)

hDev : デバイスのハンドル
Ch : チャンネル(0～2)
pCnt : [出力]カウント数の格納先

指定されたチャンネルのカウンタの値を読み出します。

TWB_TimerSetCnt() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_TimerSetCnt(TW_HANDLE hDev, long Ch, short Cnt)
VB	Function TWB_TimerSetCnt(ByVal hDev As System.IntPtr, ByVal Ch As Integer, ByVal Cnt As Short) As Integer Function TWB_TimerSetCnt(ByVal hDev As System.IntPtr, ByVal Ch As Integer, ByVal Cnt As System.UInt16) As Integer
VBA	Function TWB_TimerSetCnt(ByVal hDev As Long, ByVal Ch As Long, ByVal Cnt As Integer) As Long
C#	STATUS TimerSetCnt(System.IntPtr hDev, int Ch, short Cnt) STATUS TimerSetCnt(System.IntPtr hDev, int Ch, ushort Cnt)

hDev : デバイスのハンドル
Ch : チャンネル(0～2)
Cnt : カウント数

指定されたチャンネルのカウンタに値をセットします。

TWB_TimerSetNumOfPulse() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_TimerSetNumOfPulse(TW_HANDLE hDev, long Ch, DWORD nPulse)
VB	Function TWB_TimerSetNumOfPulse(ByVal hDev As System.IntPtr, ByVal Ch As Integer, ByVal nPulse As Integer) As Integer
VBA	Function TWB_TimerSetNumOfPulse(ByVal hDev As Long, ByVal Ch As Long, ByVal nPulse As Long) As Long
C#	STATUS TimerSetNumOfPulse(System.IntPtr hDev, int Ch, uint nPulse)

hDev　: デバイスのハンドル
Ch　　: チャンネル(0～2)
nPulse : 出力するパルス数
　　0x00000000 : 設定できません。TW_INVALID_ARGS が返ります
　　0xffffffff : 停止するまでパルス出力を続けます(デフォルト動作)
　　上記以外　: 指定した数のパルスを出力して停止します

PWM 出力で出力するパルス数を指定します。

TWB_TimerReadNumOfPulse() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_TimerReadNumOfPulse(TW_HANDLE hDev, long Ch, DWORD *pnPulse)
VB	Function TWB_TimerReadNumOfPulse(ByVal hDev As System.IntPtr, ByVal Ch As Integer, ByRef pnPulse As Integer) As Integer
VBA	Function TWB_TimerReadNumOfPulse(ByVal hDev As Long, ByVal Ch As Long, ByRef pnPulse As Long) As Long
C#	STATUS TimerReadNumOfPulse(System.IntPtr hDev, int Ch, out uint pnPulse)

hDev　: デバイスのハンドル
Ch　　: チャンネル(0～2)
pnPulse : [出力]残りパルス数の格納先

PWM 出力の残りのパルス数を返します。出力パルス数が設定されていない場合、pnPulse には 0 が返ります。

□ シリアルポート操作関数

TWB_SCISetMode() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_SCISetMode(TW_HANDLE hDev, long Ch, long Mode, long Baud)
VB	Function TWB_SCISetMode(ByVal hDev As System.IntPtr, ByVal Ch As Integer, ByVal Mode As TWB_SCI_MODE, ByVal Baud As TWB_SCI_BAUD) As Integer
VBA	Function TWB_SCISetMode(ByVal hDev As Long, ByVal Ch As Long, ByVal Mode As TWB_SCI_MODE, ByVal Baud As TWB_SCI_BAUD) As Long
C#	STATUS SCISetMode(System.IntPtr hDev, int Ch, SCI_MODE Mode, SCI_BAUD Baud)

hDev ： デバイスのハンドル
Ch ： チャンネル(0,1)
Mode ： シリアルポートの動作設定。次の値からデータ長、パリティ、ストップビットに関する設定を1つずつ選択して OR で結合します。指定しない項目はデフォルトと書かれた設定になります。

言語	値	説明
C/C++	TWB_SCI_DATA8	データ長を 8 ビットにします(デフォルト)。
C++	TWB::SCI_MODE::DATA8	
VB/VBA	TWB_SCI_MODE.DATA8	
C#	TWB.SCI_MODE.DATA8	
C/C++	TWB_SCI_DATA7	データ長を 7 ビットにします。
C++	TWB::SCI_MODE::DATA7	
VB/VBA	TWB_SCI_MODE.DATA7	
C#	TWB.SCI_MODE.DATA7	
C/C++	TWB_SCI_NOPARITY	パリティビットを使用しません(デフォルト)。
C++	TWB::SCI_MODE::NO_PARITY	
VB/VBA	TWB_SCI_MODE.NO_PARITY	
C#	TWB.SCI_MODE.NO_PARITY	
C/C++	TWB_SCI_EVEN	偶数パリティを使用します。
C++	TWB::SCI_MODE::EVEN	
VB/VBA	TWB_SCI_MODE.EVEN	
C#	TWB.SCI_MODE.EVEN	
C/C++	TWB_SCI_ODD	奇数パリティを使用します。
C++	TWB::SCI_MODE::ODD	
VB/VBA	TWB_SCI_MODE.ODD	
C#	TWB.SCI_MODE.ODD	
C/C++	TWB_SCI_STOP1	ストップビットを 1 ビットとします(デフォルト)。
C++	TWB::SCI_MODE::STOP1	
VB/VBA	TWB_SCI_MODE.STOP1	
C#	TWB.SCI_MODE.STOP1	
C/C++	TWB_SCI_STOP2	ストップビットを 2 ビットとします。
C++	TWB::SCI_MODE::STOP2	
VB/VBA	TWB_SCI_MODE.STOP2	
C#	TWB.SCI_MODE.STOP2	

Baud ： ボーレート。次の値のいずれか

言語	値	説明
C/C++	TWB_SCI_BAUD300	ボーレートを 300bps にします。
C++	TWB::SCI_BAUD::BAUD300	
VB/VBA	TWB_SCI_BAUD.BAUD300	
C#	TWB.SCI_BAUD.BAUD300	
C/C++	TWB_SCI_BAUD600	ボーレートを 600bps にします。
C++	TWB::SCI_BAUD::BAUD600	
VB/VBA	TWB_SCI_BAUD.BAUD600	
C#	TWB.SCI_BAUD.BAUD600	
C/C++	TWB_SCI_BAUD1200	ボーレートを 1200bps にします。
C++	TWB::SCI_BAUD::BAUD1200	
VB/VBA	TWB_SCI_BAUD.BAUD1200	
C#	TWB.SCI_BAUD.BAUD1200	
C/C++	TWB_SCI_BAUD2400	ボーレートを 2400bps にします。
C++	TWB::SCI_BAUD::BAUD2400	
VB/VBA	TWB_SCI_BAUD.BAUD2400	
C#	TWB.SCI_BAUD.BAUD2400	
C/C++	TWB_SCI_BAUD4800	ボーレートを 4800bps にします。
C++	TWB::SCI_BAUD::BAUD4800	
VB/VBA	TWB_SCI_BAUD.BAUD4800	
C#	TWB.SCI_BAUD.BAUD4800	
C/C++	TWB_SCI_BAUD9600	ボーレートを 9600bps にします。
C++	TWB::SCI_BAUD::BAUD9600	
VB/VBA	TWB_SCI_BAUD.BAUD9600	
C#	TWB.SCI_BAUD.BAUD9600	
C/C++	TWB_SCI_BAUD14400	ボーレートを 14400bps にします。
C++	TWB::SCI_BAUD::BAUD14400	
VB/VBA	TWB_SCI_BAUD.BAUD14400	
C#	TWB.SCI_BAUD.BAUD14400	
C/C++	TWB_SCI_BAUD19200	ボーレートを 19200bps にします。
C++	TWB::SCI_BAUD::BAUD19200	
VB/VBA	TWB_SCI_BAUD.BAUD19200	
C#	TWB.SCI_BAUD.BAUD19200	
C/C++	TWB_SCI_BAUD38400	ボーレートを 38400bps にします。
C++	TWB::SCI_BAUD::BAUD38400	
VB/VBA	TWB_SCI_BAUD.BAUD38400	
C#	TWB.SCI_BAUD.BAUD38400	

シリアルポートの動作設定と速度設定を行います。チャンネル 1 に対してこの関数を呼び出すと、再起動するまでユーザーファームのデバッグ用ポートとして使用できなくなります。

TWB_SCIReadStatus() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_SCIReadStatus(TW_HANDLE hDev, long Ch, BYTE *pStatus, long *pnReceive)
VB	Function TWB_SCIReadStatus(ByVal hDev As System.IntPtr, ByVal Ch As Integer, ByRef pStatus As Byte, ByRef pnReceive As Integer) As Integer
VBA	Function TWB_SCIReadStatus(ByVal hDev As Long, ByVal Ch As Long, ByRef pStatus As Byte, ByRef pnReceive As Long) As Long
C#	STATUS SCIReadStatus(System.IntPtr hDev, int Ch, out byte pStatus, out int pnReceive)

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。各ビットの意味は以下です
0(LSB)ビット～2 ビット : 常に 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) : 常に 0 です
pnReceive : [出力]受信データバイト数の格納先

シリアルポートのステータスと、デバイス内部のシリアル用受信バッファ中のデータバイト数を取得します。
シリアル用受信バッファは、チャンネル毎に 127 バイトまでのデータを格納できます。

TWB_SCIRead() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_SCIRead(TW_HANDLE hDev, long Ch, void *pData, long nData, long *pnRead)
VB	Function TWB_SCIRead(ByVal hDev As System.IntPtr, ByVal Ch As Integer, ByVal pData As Object, ByVal nData As Integer, ByRef pnRead As Integer) As Integer
VBA	Function TWB_SCIRead(ByVal hDev As Long, ByVal Ch As Long, ByRef pData As Any, ByVal nData As Long, ByRef pnRead As Long) As Long
C#	STATUS SCIRead(System.IntPtr hDev, int Ch, object pData, int nData, out int pnRead)

hDev : デバイスのハンドル
Ch : チャンネル(0,1)
pData : [出力]読み出したデータの格納先
nData : 受信するバイト数(0～255)
pnRead : [出力]実際に受信したバイト数の格納先

デバイスのシリアルポートからデータを読み出します。
デバイスの受信バッファ中のデータ数よりも大きなデータを要求すると、関数と同時にデバイスもデータ待ち状態(ブロッキング)となります。
ブロッキングを起こさないようにするには、この関数を呼び出す前に TWB_SCIReadStatus()を使用し、受信バッファ中のデータバイト数を確認した上で、受信バイト数以下のデータを要求するようにしてください。

TWB_SCIWrite() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_SCIWrite(TW_HANDLE hDev, long Ch, void *pData, long nData)
VB	Function TWB_SCIWrite(ByVal hDev As System.IntPtr, ByVal Ch As Integer, ByVal pData As Object, ByVal nData As Integer) As Integer
VBA	Function TWB_SCIWrite(ByVal hDev As Long, ByVal Ch As Long, ByRef pData As Any, ByVal nData As Long) As Long
C#	STATUS SCIWrite(System.IntPtr hDev, int Ch, object pData, int nData)

hDev ： デバイスのハンドル
Ch ： チャンネル(0,1)
pData ： [入力]送信データ
nData ： 送信するバイト数(0～255)

デバイスのシリアルポートからデータを送信します。

TWB_SCISetDelimiter() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_SCISetDelimiter(TW_HANDLE hDev, long Ch, void *pDelimiter, long nDelimiter)
VB	Function TWB_SCISetDelimiter(ByVal hDev As System.IntPtr, ByVal Ch As Integer, ByVal pDelimiter() As Byte, ByVal nDelimiter As Integer) As Integer Function TWB_SCISetDelimiter(ByVal hDev As System.IntPtr, ByVal Ch As Integer, ByVal pDelimiter() As String, ByVal nDelimiter As Integer) As Integer
VBA	Function TWB_SCISetDelimiter(ByVal hDev As Long, ByVal Ch As Long, ByVal pDelimiter As Byte, ByVal nDelimiter As Long) As Long
C#	STATUS SCISetDelimiter(System.IntPtr hDev, int Ch, byte[] pDelimiter, int nDelimiter) STATUS SCISetDelimiter(System.IntPtr hDev, int Ch, string pDelimiter, int nDelimiter)

hDev ： デバイスのハンドル
Ch ： チャンネル(0,1)
pDelimiter ： [入力]デリミタコード
nDelimiter ： デリミタコードのバイト数(0～2)

シリアルポートにデリミタコードを設定します。デリミタの設定は TWB_SCIRead()の動作に影響します。
TWB_SCIRead()関数はデリミタコード(1 バイトまたは 2 バイト)が現れると、シリアルポートからの読み取りを一旦中止し、デリミタコードより後には指定バイトまで 0 をコピーしてデータを返します。

□ ユーザーファームサポート関数

TWB_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;
	} TWB_ATF_INFO;

VB	Public Structure TWB_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 TWB_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

言語	宣言

C#	public struct ATF_INFO
	{
	  public uint FormatVer;
	  public uint FirmwareVer;
	  [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 32)]
	  public string Description;
	  [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 32)]
	  public string Manufacture;
	  public uint ProgramVer;
	  public uint AddressTop;
	  public uint AddressBottom;
	  public uint CommandAddress;
	  public short EncryptMode;
	  public short wRsv;
	  public uint MainAddress;
	}

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

TWB_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。

TWB_ATFGetInfo() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_ATFGetInfo(LPCTSTR FileName, TWB_ATF_INFO *pInfo)
VB	Function TWB_ATFGetInfo(ByVal FileName As String, ByRef pInfo As TWB_ATF_INFO) As Integer
VBA	Function TWB_ATFGetInfo(ByVal FileName As String, ByRef pInfo As TWB_ATF_INFO) As Long
C#	STATUS ATFGetInfo(string FileName, out ATF_INFO pInfo)

FileName : ATF ファイルのパス
pInfo : [出力]ATF ファイルの情報

ATF ファイルの情報を返します。内容は TWB_ATF_INFO 構造体の説明を参照してください。

TWB_ATFUserCommand() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_ATFUserCommand(TW_HANDLE hDev, WORD Command, DWORD Param1, DWORD Param2, void *pData, long nData)
VB	Function TWB_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 TWB_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
C#	STATUS ATFUserCommand(System.IntPtr hDev, ushort Command, uint Param1, uint Param2, object pData, int nData) STATUS ATFUserCommand(System.IntPtr hDev, ushort Command, uint Param1, uint Param2)

hDev : デバイスのハンドル
Command : コマンドを識別する番号
Param1 : コマンドパラメータ 1
Param2 : コマンドパラメータ 2
pData : [入力]付加するデータ
nData : 付加するデータのバイト数

ユーザーファームにコマンドを送信します。Command, Param1, Param2 の各パラメータを引数としてユーザーファームの ATF_Command()関数が呼び出されます。
追加の送信データがある場合 pData を指定すると送信することができます。追加したデータはユーザーファーム側で TWFA_Receive()関数を呼び出して全て読み取る必要があります。

TWB_ATFDownload() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_ATFDownload(TW_HANDLE hDev, LPCTSTR FileName, LPCTSTR Reserve)
VB	Function TWB_ATFDownload(ByVal hDev As System.IntPtr, ByVal FileName As String, ByVal Reserve As String) As Integer
VBA	Function TWB_ATFDownload(ByVal hDev As Long, ByVal FileName As String, ByVal Reserve As String) As Long
C#	STATUS ATFDownload(System.IntPtr hDev, string FileName)

hDev : デバイスのハンドル
FileName : ATF ファイルのパス
Reserve : 予約。Null 値としてください。

ATF ファイルをデバイスにダウンロードし使用可能にします。

TWB_Write() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_Write(TW_HANDLE hDev, void *pData, DWORD nData, DWORD *pWritten)
VB	Function TWB_Write(ByVal hDev As System.IntPtr, ByVal pData As Object, ByVal nData As Integer, ByRef pWritten As Integer) As Integer
VBA	Function TWB_Write(ByVal hDev As Long, ByRef pData As Any, ByVal nData As Long, ByRef pWritten As Long) As Long
C#	STATUS Write(System.IntPtr hDev, object pData, int nData, out int pWritten) STATUS Write(System.IntPtr hDev, object pData, int nData)

hDev : デバイスのハンドル
pData : [入力]送信するデータの格納先
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

ユーザーファームにデータを送信する場合に使用します。通常は TWB_ATFUserCommand()で送信しきれないデータがある場合に補助的に呼び出します。
この関数によるデータはコマンドとしてではなく、そのままデバイスに送信されます。システムファームのコマンドループは受信したデータを全てコマンドとして処理しようとしますので、この関数の呼び出しタイミングには注意が必要です。この関数で送信したデータがデバイスの受信バッファに残ったままシステムファームに処理が移ると、システムファームはデータをコマンドとして解釈し誤った動作を行います。
送信したデータはユーザーファーム内の ATF_Command()や ATF_Main()関数内で確実に読み出してください。

TWB_Read() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_Read(TW_HANDLE hDev, void *pData, DWORD nData, DWORD *pRead)
VB	Function TWB_Read(ByVal hDev As System.IntPtr, ByVal pData As Object, ByVal nData As Integer, ByRef pRead As Integer) As Integer
VBA	Function TWB_Read(ByVal hDev As Long, ByRef pData As Any, ByVal nData As Long, ByRef pRead As Long) As Long
C#	STATUS Read(System.IntPtr hDev, object pData, int nData, out int pRead) STATUS Read(System.IntPtr hDev, object pData, int nData)

hDev : デバイスのハンドル
pData : [出力]受信したデータの格納先
nData : 受信するバイト数(0～65536)
pRead : [出力]実際に受信したバイト数の格納先

デバイスから送信されたデータを読み出します。通常は TWB_ATFUserCommand()の応答としてユーザーファームから送信したデータを受信するために呼び出します。

TWB_Write16() USB

言語	関数宣言
C/C++	TW_STATUS TWB_Write16(TW_HANDLE hDev, void *pData, DWORD nWord, DWORD *pnWritten)
VB	Function TWB_Write16(ByVal hDev As System.IntPtr, ByVal pData As Object, ByVal nWord As Integer, ByRef pnWritten As Integer) As Integer
VBA	Function TWB_Write16(ByVal hDev As Long, ByRef pData As Any, ByVal nWord As Long, ByRef pnWritten As Long) As Long
C#	STATUS Write16(System.IntPtr hDev, object pData, int nWord, out int pnWritten) STATUS Write16(System.IntPtr hDev, object pData, int nWord)

hDev : デバイスのハンドル
pData : [入力]送信するデータの格納先
nWord : 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

ユーザーファームに 16 ビット単位でデータを送信する場合に使用します。この関数は USB(HS)デバイス専用で、接続時にオプションとして TWB_MODE_BUS16(相当のオプション)を指定した場合のみ呼び出し可能です。

データは格納されたバイト順で送信されます。2 バイト以上の数値データを送信する場合はバイトオーダーの変換が必要です。

この関数の送信データはユーザーファーム側では TWFA_Receive16()または TWFA_DmaReceive16()関数で受信する必要があります。

TWB_Read16() USB

言語	関数宣言
C/C++	TW_STATUS TWB_Read16(TW_HANDLE hDev, void *pData, DWORD nWord, DWORD *pnRead);
VB	Function TWB_Read16(ByVal hDev As System.IntPtr, ByVal pData As Object, ByVal nWord As Integer, ByRef pnRead As Integer) As Integer
VBA	Function TWB_Read16(ByVal hDev As Long, ByRef pData As Any, ByVal nWord As Long, ByRef pnRead As Long) As Long
C#	STATUS Read16(System.IntPtr hDev, object pData, int nWord, out int pnRead); STATUS Read16(System.IntPtr hDev, object pData, int nWord);

hDev : デバイスのハンドル
pData : [出力]受信したデータの格納先
nWord : 受信するワード数(0～65536)
pnRead : [出力]実際に受信したワード数の格納先

ユーザーファームから 16 ビット単位で送信されたデータを読み出す場合に使用します。この関数は USB(HS)デバイス専用で、デバイスとの接続時にオプションとして TWB_MODE_BUS16(相当のオプション)を指定した場合のみ呼び出し可能です。

データは受信したバイト順に格納されます。2 バイト以上の数値データを送信する場合はバイトオーダーの変換が必要です。

この関数で受け取るデータはユーザーファーム側では TWFA_Transmit16()または TWFA_DmaTransmit16()関数で送信する必要があります。

TWB_GetQueueStatus() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_GetQueueStatus(TW_HANDLE hDev, DWORD *pnQueue)
VB	Function TWB_GetQueueStatus(ByVal hDev As System.IntPtr, ByRef pnQueue As Integer) As Integer
VBA	Function TWB_GetQueueStatus(ByVal hDev As Long, ByRef pnQueue As Long) As Long
C#	STATUS GetQueueStatus(System.IntPtr hDev, out int pnQueue)

hDev　　: デバイスのハンドル
pnQueue : [出力]受信バッファ中のバイト数

デバイスから送信されたデータを蓄積するパソコン内のバッファに何バイト受信しているかを調べます。
pnQueue に返される値は通常バイト数ですが、USB(HS) デバイスで TWB_Read16()を使用する場合に限りワード数を示します。

TWB_Purge() USB LAN

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

hDev　: デバイスのハンドル

デバイスから送信されたデータを蓄積するパソコン内の受信バッファをクリアします。

□ フラッシュメモリ操作関数

TWB_FlashEraseBlk() USB LAN

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

hDev ： デバイスのハンドル
Blk ： ブロック番号(1～3)

フラッシュ書換えモード用のフラッシュメモリ消去関数です。フラッシュメモリの指定ブロックを消去します。

TWB_FlashWrite() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_FlashWrite(TW_HANDLE hDev, DWORD Address, void *pData, long nData)
VB	Function TWB_FlashWrite(ByVal hDev As System.IntPtr, ByVal Address As Integer, ByVal pData As Object, ByVal nData As Integer) As Integer
VBA	Function TWB_FlashWrite(ByVal hDev As Long, ByVal Address As Long, ByRef pData As Any, ByVal nData As Long) As Long
C#	STATUS FlashWrite(System.IntPtr hDev, uint Address, object pData, int nData)

hDev ： デバイスのハンドル
Address ： 書き込むアドレス(0x1000～0x3f80,128 バイト境界を指定)
pData ： [入力]書き込むデータ
nData ： データのバイト数(0～4096, 128 バイト単位)

フラッシュ書換えモード用のフラッシュメモリ書込み関数です。書き込むアドレスの指定は 128 バイト境界(下位 7 ビットが 0)を指定する必要があります。また書き込むデータのバイト数は 128 の倍数を指定してください。

TWB_FlashRead() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_FlashRead(TW_HANDLE hDev, DWORD Address, void *pData, long nData)
VB	Function TWB_FlashRead(ByVal hDev As System.IntPtr, ByVal Address As Integer, ByVal pData As Object, ByVal nData As Integer) As Integer
VBA	Function TWB_FlashRead(ByVal hDev As Long, ByVal Address As Long, ByRef pData As Any, ByVal nData As Long) As Long
C#	STATUS FlashRead(System.IntPtr hDev, uint Address, object pData, int nData)

hDev　: デバイスのハンドル
Address : 読み出すアドレス(0x1000～0x3f80,128 バイト境界を指定)
pData　: [出力]読み出したデータの格納先
nData　: データのバイト数(0～4096, 128 バイト単位)

フラッシュ書換えモード用のフラッシュメモリ読出し関数です。読み出すアドレスの指定は 128 バイト境界(下位 7 ビットが 0)を指定する必要があります。また読み出すデータのバイト数は 128 の倍数を指定してください。

TWB_UPFlashAttachWriter() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_UPFlashAttachWriter(TW_HANDLE hDev)
VB	Function TWB_UPFlashAttachWriter(ByVal hDev As System.IntPtr) As Integer
VBA	Function TWB_UPFlashAttachWriter(ByVal hDev As Long) As Long
C#	STATUS UPFlashAttachWriter(System.IntPtr hDev);

hDev　: デバイスのハンドル

フラッシュメモリ制御用のファームウェア(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フラッシュメモリが操作できるようにします。ファイルが見つからない場合 TW_FILE_ERROR を返します。
フラッシュメモリ制御用のファームウェアはユーザーメモリにダウンロードされますので、この関数の呼び出しによりユーザーメモリの内容は破壊されます。また、フラッシュメモリの操作が必要な間はユーザーメモリを利用することができません。

TWB_UPFlashEraseBlk() USB LAN

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

hDev ： デバイスのハンドル
Blk ： ブロック番号(1～3)

ユーザープログラムモード用のフラッシュメモリ消去関数です。フラッシュメモリの指定ブロックを消去します。この関数を使用するには予め TWB_UPFlashAttachWriter()が正常に終了している必要があります。

TWB_UPFlashWrite() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_UPFlashWrite(TW_HANDLE hDev, DWORD Address, void *pData, long nData)
VB	Function TWB_UPFlashWrite(ByVal hDev As System.IntPtr, ByVal Address As Integer, ByVal pData As Object, ByVal nData As Integer) As Integer
VBA	Function TWB_UPFlashWrite(ByVal hDev As Long, ByVal Address As Long, ByRef pData As Any, ByVal nData As Long) As Long
C#	STATUS UPFlashWrite(System.IntPtr hDev, uint Address, object pData, int nData)

hDev ： デバイスのハンドル
Address ： 書き込むアドレス(0x1000～0x3f80,128 バイト境界を指定)
pData ： [入力]書き込むデータ
nData ： データのバイト数(0～4096, 128 バイト単位)

ユーザープログラムモード用のフラッシュメモリ書込み関数です。書き込むアドレスの指定は 128 バイト境界(下位 7 ビットが 0)を指定する必要があります。また書き込むデータのバイト数は 128 の倍数を指定してください。
この関数を使用するには予め TWB_UPFlashAttachWriter()が正常に終了している必要があります。

□ ハードウェアイベント操作関数

- 『USBM3069F』、『USBM3069-S』、『USBM3069-SL』では使用できません。

TWB_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];
	} TWB_HW_EVENT;

VB	Public Structure TWB_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 TWB_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

言語	宣言

C#	public struct HW_EVENT
	{
	  public System.IntPtr hRecvWindow;
	  public uint idRecvThread;
	  public System.IntPtr lpRsv;
	  public int Message;
	  public uint EventBits;
	  [MarshalAs(UnmanagedType.ByValArray, SizeConst = 4)]
	  public int[] PCCnt;
	  [MarshalAs(UnmanagedType.ByValArray, SizeConst = 4)]
	  public int[] PCCmp;
	  [MarshalAs(UnmanagedType.ByValArray, SizeConst = 4)]
	  public int[] ADVal;。
	  [MarshalAs(UnmanagedType.ByValArray, SizeConst = 4)]
	  public short[] ADCmp;
	
	  public void Initialize()
	  {
	  PCCnt = new int[4];
	  PCCmp = new int[4];
	  ADVal = new int[4];
	  ADCmp = new short[4];
	  }
	}

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。TWB_SetHwEvent() 関数の呼び出しに使用します。
Visual Basic と C#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。

TWB_SetHwEvent() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_SetHwEvent(TW_HANDLE hDev, TWB_HW_EVENT *pHwEvent)
VB	Function TWB_SetHwEvent(ByVal hDev As System.IntPtr, ByRef pHwEvent As TWB_HW_EVENT) As Integer
VBA	Function TWB_SetHwEvent(ByVal hDev As Long, ByRef pHwEvent As TWB_HW_EVENT) As Long
C#	STATUS SetHwEvent(System.IntPtr hDev, ref HW_EVENT pHwEvent)

hDev　　：デバイスのハンドル
pHwEvent：[入力]ハードウェアイベントの通知設定

デバイスでパルスカウンタとアナログ入力の状態を監視し、予め設定した状態になったときWindowsのアプリケーションに対してメッセージによる通知を行います。メッセージの送信先はウィンドウかスレッドを指定でき、両方に通知することもできます(TWB_HW_EVENT構造体参照)。

VBA以外の言語ではpHwEventをNull値としてTWB_SetHwEvent() 関数を呼び出します。VBAではpHwEventのEventBitsを0として呼び出してください。

ユーザーファームで SRV_TransmitEvent()関数を利用することで、独自のメッセージをアプリケーションに通知することが可能です。

□ その他の関数

TWB_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];
	} TWB_PRODUCT_INFO;

VB	Public Structure TWB_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 TWB_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

C#	public struct PRODUCT_INFO
	{
	  public short Empty;
	  public short wRsv;
	  public UUID ID;
	  public int Number;
	  [MarshalAs(UnmanagedType.ByValTStr, SizeConst=32)]
	  public string Description;
	  [MarshalAs(UnmanagedType.ByValTStr, SizeConst=32)]
	  public string Manufacture;
	  [MarshalAs(UnmanagedType.ByValArray, SizeConst=40)]
	  public byte []Reserve;
	}

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

TWB_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。

TWB_UUID 構造体

言語	宣言

VB	Public Structure TWB_UUID
	  Public Data1 As Integer
	  Public Data2 As Short
	  Public Data3 As Short
	  <MarshalAs(UnmanagedType.ByValArray, SizeConst:=8)> _
	  Public Data4() As Byte
	End Structure

VBA	Public Type TWB_UUID
	  Data1 As Long
	  Data2 As Integer
	  Data3 As Integer
	  Data4(7) As Byte
	End Type

C#	public struct UUID
	{
	  public int Data1;
	  public short Data2;
	  public short Data3;
	  [MarshalAs(UnmanagedType.ByValArray, SizeConst=8)]
	  public byte []Data4;
	}

TWB_PRODUCT_INFO 構造体の一部として使用します。UUID を数値情報として格納するための構造体です。

TWB_ReadPI() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_ReadPI(TW_HANDLE hDev, TWB_PRODUCT_INFO *pPI)
VB	Function TWB_ReadPI(ByVal hDev As System.IntPtr, ByRef pPI As TWB_PRODUCT_INFO) As Integer
VBA	Function TWB_ReadPI(ByVal hDev As Long, ByRef pPI As TWB_PRODUCT_INFO) As Long
C#	STATUS ReadPI(System.IntPtr hDev, out PRODUCT_INFO pPI)

hDev : デバイスのハンドル
pPI　: [出力]製品情報の格納先

接続中のデバイスの製品情報を取得します。

TWB_Initialize() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_Initialize(TW_HANDLE hDev, long Opt)
VB	Function TWB_Initialize(ByVal hDev As System.IntPtr, ByVal Opt As TWB_INIT_OPT) As Integer
VBA	Function TWB_Initialize(ByVal hDev As Long, ByVal Opt As TWB_INIT_OPT) As Long
C#	STATUS Initialize(System.IntPtr hDev, INIT_OPT Opt)

hDev ： デバイスのハンドル
Opt ： 初期化する機能、動作オプションを OR で結合します

初期化に関するオプション

言語	値	説明
C/C++	TWB_INIT_ALL	全機能を初期化し、次の「動作設定に関するオプション」を全て有効にします。
C++	TWB::INIT_OPT::ALL	
VB/VBA	TWB_INIT_OPT.ALL	
C#	TWB.INIT_OPT.ALL	
C/C++	TWB_INIT_PORT_DIR	入出力ポートの方向を初期化します。
C++	TWB::INIT_OPT::PORT_DIR	
VB/VBA	TWB_INIT_OPT.PORT_DIR	
C#	TWB.INIT_OPT.PORT_DIR	
C/C++	TWB_INIT_PORT_DATA	デジタル出力端子の状態を初期化します。
C++	TWB::INIT_OPT::PORT_DATA	
VB/VBA	TWB_INIT_OPT.PORT_DATA	
C#	TWB.INIT_OPT.PORT_DATA	
C/C++	TWB_INIT_BUS	外部バスの設定を初期化します。
C++	TWB::INIT_OPT::BUS	
VB/VBA	TWB_INIT_OPT.BUS	
C#	TWB.INIT_OPT.BUS	
C/C++	TWB_INIT_TIMER	16 ビットタイマ(ハードウェアカウンタ)を初期化します。
C++	TWB::INIT_OPT::TIMER	
VB/VBA	TWB_INIT_OPT.TIMER	
C#	TWB.INIT_OPT.TIMER	
C/C++	TWB_INIT_SCI	シリアルポートを初期化します。
C++	TWB::INIT_OPT::SCI	
VB/VBA	TWB_INIT_OPT.SCI	
C#	TWB.INIT_OPT.SCI	
C/C++	TWB_INIT_PC	パルスカウンタ(ソフトウェアカウンタ)を初期化します。
C++	TWB::INIT_OPT::PC	
VB/VBA	TWB_INIT_OPT.PC	
C#	TWB.INIT_OPT.PC	
C/C++	TWB_INIT_DA	アナログ出力端子を初期化します。
C++	TWB::INIT_OPT::DA	
VB/VBA	TWB_INIT_OPT.DA	
C#	TWB.INIT_OPT.DA	

動作設定に関するオプション

言語	値	説明
C/C++	TWB_INIT_EI_IN_SCI0	マイコンがシリアル 0 の受信割り込み処理中、他の割り込みを許可します。
C++	TWB::INIT_OPT:: EI_IN_SCI0	
VB/VBA	TWB_INIT_OPT. EI_IN_SCI0	
C#	TWB.INIT_OPT. EI_IN_SCI0	
C/C++	TWB_INIT_EI_IN_SCI1	マイコンがシリアル 1 の受信割り込み処理中、他の割り込みを許可します。
C++	TWB::INIT_OPT::EI_IN_SCI1	
VB/VBA	TWB_INIT_OPT.EI_IN_SCI1	
C#	TWB.INIT_OPT.EI_IN_SCI1	
C/C++	TWB_INIT_EI1_IN_PC	マイコンがパルスカウンタの割り込み処理中、優先度の高い割り込みを許可します。 PWM の回数指定を行う場合に、PWM 出力停止を優先的に行うことができるようになります。
C++	TWB::INIT_OPT::EI1_IN_PC	
VB/VBA	TWB_INIT_OPT.EI1_IN_PC	
C#	TWB.INIT_OPT.EI1_IN_PC	
C/C++	TWB_INIT_SYNC_TIMER	PWM 動作中の周波数やデューティ変更の際、タイマ動作への同期を優先します。同期できるパルス周期が約 2 秒まで延長されます(指定しない場合、20msec 以上のパルスには同期できません)。
C++	TWB::INIT_OPT::SYNC_TIMER	
VB/VBA	TWB_INIT_OPT.SYNC_TIMER	
C#	TWB.INIT_OPT.SYNC_TIMER	
C/C++	TWB_INIT_TIMER_PRI1	マイコン内部での 16 ビットタイマによる割り込み優先度を上げます。 TWB_INIT_EI1_IN_PC(相当の)オプションと組み合わせることで PWM 出力の停止動作を優先的に行います。
C++	TWB::INIT_OPT::TIMER_PRI1	
VB/VBA	TWB_INIT_OPT.TIMER_PRI1	
C#	TWB.INIT_OPT.TIMER_PRI1	

USBM ライブラリの機能を初期化するオプション

言語	値	説明
C/C++	TWB_INIT_DMA	デバイス搭載マイコンの DMA 機能を初期化します。
C++	TWB::INIT_OPT::DMA	
VB/VBA	TWB_INIT_OPT.DMA	
C#	TWB.INIT_OPT.DMA	
C/C++	TWB_INIT_TCPY	タイマコピーの機能を初期化します。
C++	TWB::INIT_OPT::TCPY	
VB/VBA	TWB_INIT_OPT.TCPY	
C#	TWB_INIT_OPT.TCPY	
C/C++	TWB_INIT_AD	アナログ入力に関する機能を初期化します。
C++	TWB::INIT_OPT::AD	
VB/VBA	TWB_INIT_OPT.AD	
C#	TWB.INIT_OPT.AD	

デバイスの初期化を行います。デバイスは起動時に初期化されますので呼び出しは必須ではありません。

Opt 引数で初期化する機能と動作オプションを指定できます。「USBM ライブラリの機能を初期化するオプション」は、TWB ライブラリより低位に位置する USBM ライブラリでのみ提供される機能を初期化するためのオプションです。

動作設定に関するオプションは、通常の接続では全てオンになります。デバイスへの接続オプションとして TWB_MODE_LEGACY(相当の値)を指定した場合は、これらの設定は全てオフに設定されます。

ユーザーステータスレジスタは Opt によらず 0 に初期化されます。

TWB_ReadVersion() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_ReadVersion(TW_HANDLE hDev, DWORD *pVersion)
VB	Function TWB_ReadVersion(ByVal hDev As System.IntPtr, ByRef pVersion As Integer) As Integer
VBA	Function TWB_ReadVersion(ByVal hDev As Long, ByRef pVersion As Long) As Long
C#	STATUS ReadVersion(System.IntPtr hDev, out uint pVersion) STATUS ReadVersion(System.IntPtr hDev, out int pVersion)

hDev : デバイスのハンドル
pVersion : [出力]バージョン情報の格納先

ファームウェアのバージョン情報を読み出します。pVersion に読み出された値は、最上位バイトが予約、次いでメジャーバージョン、マイナーバージョン、最下位バイトがリビジョンを示します。

ビット	31-24	23-16	15-8	7-0
意味	予約	メジャーバージョン	マイナーバージョン	リビジョン

ファームウェアのバージョンが 5.1.1 の場合、格納される値は 0x00050101 となります。

TWB_SetTimeouts() USB LAN

言語	関数宣言
C/C++	TW_STATUS TWB_SetTimeouts(TW_HANDLE hDev, DWORD dwReadTimeout, DWORD dwWriteTimeout)
VB	Function TWB_SetTimeouts(ByVal hDev As System.IntPtr, ByVal dwReadTimeout As Integer, ByVal dwWriteTimeout As Integer) As Integer
VBA	Function TWB_SetTimeouts(ByVal hDev As Long, ByVal dwReadTimeout As Long, ByVal dwWriteTimeout As Long) As Long
C#	STATUS SetTimeouts(System.IntPtr hDev, uint dwReadTimeout, uint dwWriteTimeout) STATUS SetTimeouts(System.IntPtr hDev, int dwReadTimeout, int dwWriteTimeout)

hDev : デバイスのハンドル
dwReadTimeout : 読出しのタイムアウト時間(msec 単位)
dwWriteTimeout : 書込みのタイムアウト時間(msec 単位)

デバイスからデータの読出し、デバイスへの書込みの際のタイムアウト時間を設定します。デフォルトではどちらも 5 秒に設定されています。

TWB_SetPassword() LAN

言語	関数宣言
C/C++	TW_STATUS TWB_SetPassword(LPCTSTR pPass)
VB	Function TWB_SetPassword(ByVal pPass As String) As Integer
VBA	Function TWB_SetPassword(ByVal pPass As String) As Long
C#	STATUS SetPassword(string pPass)

pPass ： パスワード文字列

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(LANMConfig)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 値 または ""(空文字列)の場合、デフォルトのパスワードが使用されます。

TWB_SetNetworkPort() LAN

言語	関数宣言
C/C++	TW_STATUS TWB_SetNetworkPort(DWORD PortNumber)
VB	Function TWB_SetNetworkPort(ByVal PortNumber As Integer) As Integer
VBA	Function TWB_SetNetworkPort(ByVal PortNumber As Long) As Long
C#	STATUS SetNetworkPort(int PortNumber)

PortNumber ： デバイスとの接続に使用するポート番号

サーバーモードの LAN デバイスと接続する際に使用するポート番号を変更します。この関数でポート番号を変更した場合、以後のそのプロセスからの接続は全て設定したポート番号に対して行われます。
ポート番号のデフォルト値は 49152 です。

TWB_GetSocket() LAN

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

hDev ： デバイスのハンドル

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。

□ VBA 用ヘルパー関数

以下の関数は VBA 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。16 ビットタイマのカウンタ値を符号無し整数として扱いたい場合などに使用します。

TWB_ToUINT16() USB LAN

```
Function TWB_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。

TWB_ToINT32() USB LAN

```
Function TWB_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。

5. 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]
MaxDevice=512

[LANM3069]
PortNumber=49152
LocalNet= 192.168.1.0
```

図 21　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 78　[COMMON]セクションのキー

キー	意味
MaxDevice	プロセスが接続できる最大デバイス数を指定します。デフォルト値は 256 です。

表 79　[LANM3069]セクションのキー

キー	値	意味
PortNumber	49152～65535 (初期値 49152)	サーバーモードの LAN デバイスに接続する際の、接続先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)	サーバーモードの 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)	クライアントモードの LAN デバイスと接続する際の、TWB_Accept()の待ち時間を msec 単位で指定します。
LocalNet	Ipv4 アドレスを示す文字列	サーバーモードの LAN デバイスと接続する場合の、ネットワークアドレスを指定します。デバイスとの通信に使用するネットワークを限定したい場合に使用します。

6. トラブルシューティング

□ USB デバイスと接続できない場合

USB デバイスと通信ができない場合、下記の事項をお確かめください。

- ドライバが正しくインストールされているか確認してください。確認方法は製品のユーザーズマニュアルを参照してください。
- 『USBM3069-S/SL』、『USBM3069-HS/HSL』では外部回路が必要です。USB のみ接続しても電源が入りませんのでご注意ください。外部回路の構成方法は製品のユーザーズマニュアルを参照してください。
- 『USBM3069F』、『USBM3069-S/SL』では、「USB-シリアルポート変換デバイス」をご利用の場合に、それぞれのドライババージョンが競合し、正しく動作しない場合があります。お手数ですが弊社サポート窓口にお問い合わせください。

□ LAN デバイスと接続できない場合

LAN デバイスと通信ができない場合、下記の事項をお確かめください。

- ご利用のパソコンで、インターネット通信などを監視するセキュリティソフトを使用されている場合は、セキュリティソフトを一時的に停止し、再度接続してみてください。セキュリティソフトの設定によっては、製品との通信がブロックされる場合があります。
- 「Windows ファイアウォール」が有効になっている場合、一時的に無効に設定し、再度接続してみてください。「Windows ファイアウォール」によって通信がブロックされてしまう場合は、「コントロールパネル→セキュリティセンター→Windows ファイアウォール」を開き、「例外」にアプリケーションを登録することで通信可能になります。
- 製品の IP アドレスを自動取得（DHCP）に設定されている場合は、固定の IP アドレスを設定してみてください。IP アドレスのリース数の制限や、DHCP サーバーとの通信が上手くいかないことにより、ネットワーク設定ができない場合があります。
- ネットワーク内に、製品、および、ホストパソコンと同一の IP アドレスを使用しているノードが無いこと確認してください。また、制御に使用するポート番号（TCP、および、UDP の 49152 番）を利用しているアプリケーションが無いことを確認してください。

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
2012 年 6 月	初	
2013 年 3 月	2	・誤記を修正
2013 年 5 月	3	・誤記を修正
2019 年 4 月	4	・対応製品を追加
2021 年 3 月	5	・対応製品を追加