User's Manual

TMCTLライブラリ
Application Programming Interface
ユーザーズマニュアル

YOKOGAWA
横河計測株式会社

IM B9852UB-01JA
6 版

はじめに

このユーザーズマニュアルは，当社製品とPCとを接続して通信を行うためのライブラリ（TMCTLライブラリ）の取り扱い上の注意/機能/API仕様などについて説明したものです。
ご使用中にこのマニュアルをよくお読みいただき，正しくお使いください。お読みになったあとは，ご使用時にすぐにご覧になれるところに，大切に保存してください。ご使用中に操作がわからなくなったときなどにきっとお役にたちます。
なお，当社製品の取り扱い上の注意/機能/操作方法，Windowsの取り扱い/操作方法などについては，それぞれのマニュアルをご覧ください。

ご注意

●本書の内容は，性能/機能の向上などにより，将来予告なしに変更することがあります。
　また，実際の画面表示内容が，本書に記載の画面表示内容と多少異なることがあります。
●本書の内容に関しては万全を期しておりますが，万一ご不審の点や誤りなどお気づきのことがありましたら，お手数ですが，当社支社/支店/営業所までご連絡ください。
●本書の内容の全部または一部を無断で転載，複製することは禁止されています。

商標

●Microsoft，Windows，Excel，Visual Basic，Visual C++および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または商標です。
●本文中の各社の登録商業または商標には，TM，®マークは表示していません。
●その他，本文中に使われている会社名/商品名は，各社の登録商標または商標です。

履歴

2017年6月	初版発行
2017年7月	2版発行
2018年2月	3版発行
2018年8月	4版発行
2018年10月	5版発行
2019年10月	6版発行

目次

1　ソフトウエア概要 1
2　動作に必要なシステム環境 2
3　ご使用にあたっての注意 3
4　インストール / アンインストール 4
5　TMCTL API 概要 5
5.1　通信可能製品 6
5.1.1　GPIB 6
5.1.2　RS232 6
5.1.3　USB (Except USBTMC) 7
5.1.4　USB (USBTMC) 7
5.1.5　USB (VISAUSB) 7
5.1.6　Ethernet (Legacy) 8
5.1.7　Ethernet (VXI-11) 8
5.1.8　Ethernet (UDP) 8
5.1.9　Ethernet (SOCKET) 8
5.2　API 機能概要 9
5.2.1　接続開始 / 終了 9
5.2.2　通信設定 9
5.2.3　送信 / 受信 9
5.2.4　情報取得 9
5.2.5　その他の機能 10
5.3　使用方法 (C#、VB.NET) 11
5.3.1　設定 11
5.3.2　実行 11
5.3.3　TMCTL クラス 11
5.4　使用方法 (Visual C++) 12
5.4.1　設定 12
5.4.2　実行 12
5.5　使用方法 (Visual Basic for Applications) 12
5.5.1　設定 12
5.5.2　実行 12
5.6　DLL リンク方式と配置 13
5.7　基本動作フローチャート 14
6　API 機能仕様 15
6.1　エラー定義 15
6.2　固定値定義 16
6.3　API 詳細仕様 17
6.3.1　Initialize 17
6.3.2　InitializeEx 24
6.3.3　Finish 26
6.3.4　SearchDevices 27
6.3.5　SearchDevicesEx 29
6.3.6　EncodeSerialNumber 31
6.3.7　DecodeSerialNumber 32
6.3.8　SetTimeout 33
6.3.9　SetTerm 34
6.3.10　Send 35
6.3.11　SendByLength 36
6.3.12　SendSetup 37

6.3.13 SendOnly 38
6.3.14 Receive 40
6.3.15 ReceiveSetup 41
6.3.16 ReceiveOnly 42
6.3.17 ReceiveBlockHeader 44
6.3.18 ReceiveBlockData 45
6.3.19 GetLastError 48
6.3.20 SetRen 49
6.3.21 CheckEnd 50
6.3.22 DeviceClear 51
6.3.23 DeviceTrigger 52
6.3.24 WaitSRQ 53
6.3.25 AbortWaitSRQ 54
6.3.26 SetCallback 55
6.3.27 ResetCallback 57
7 サンプルプログラム 58
7.1 C#環境 58
7.2 VB.NET 環境 60
7.3 Visual C++環境 62
7.4 Visual Basic for Applications 環境 64
8 付録 67
8.1 機器対応表 67

1 ソフトウエア概要

概要

TMCTL ライブラリは、当社製品をリモート制御するための API(Application Program Interface)を提供します。

機能

本ソフトウエアを利用して，以下の機能を実現できます。各機能については，6.3節 API詳細仕様」をご参考ください。

- デバイスへの接続と切断
- デバイスの検索
- コマンドの送受信
- リモート、ローカル設定
- デバイスクリア、デバイストリガ送信
- ステータスバイトの取得

ソフトウエア構成

本ソフトウエアは以下のパッケージ構成からなります。

- TMCTLライブラリ Application Programming Interface ユーザーズマニュアル(本書)
- API利用関連ファイル(下表)

ファイル名	内容
TmctlAPINet.dll	マネージアプリケーション用APIライブラリ
TmctlAPINet64.dll	マネージアプリケーション用APIライブラリ 64bit版
tmctl.dll	通信用APIライブラリ
tmctl64.dll	通信用APIライブラリ 64bit版
YKMUSB.dll	USB通信用ライブラリ
YKMUSB64.dll	USB通信用ライブラリ 64bit版
tmctl.h	関数プロトタイプ宣言のヘッダファイル（VC++用）
tmctl.lib	TMCTL APIインポートライブラリ（VC++用）
tmctl64.lib	TMCTL APIインポートライブラリ 64bit版（VC++用）
tmctl.bas	TMCTL 関数定義ファイル（VBA用）
tmval.bas	TMCTL 固定値定義ファイル（VBA用）

2 動作に必要なシステム環境

パーソナルコンピュータ本体

Windows7、Windows8.1、Windows10が動作可能なパーソナルコンピュータで，1GHz以上のCPUを搭載し，1GB以上(推奨は2GB以上)のメモリを有したものが必要です。

開発環境

Microsoft Visual C# 2008 ～ Microsoft Visual C# 2019
Microsoft Visual C++ 2008 ～ Microsoft Visual C++ 2019
Microsoft Visual Basic 2008 ～ Microsoft Visual Basic 2019
Microsoft Excel 2016 (Visual Basic for Applications 7.0)

.NET Framework

Microsoft .NET Framework 3.5 ～ Microsoft .NET Framework 4.7

Visual C++ ライブラリのランタイム

Visual Studio 2017 の Microsoft Visual C++ 再頒布可能パッケージ(x64)

通信インターフェース

GPIB	National Instruments社製GPIBインターフェースのドライバが動作する環境
RS232	シリアルポートが動作する環境
USB	当社製USBドライバ（YKMUSBまたはYTUSB）が動作する環境 National Instruments社製 NI-VISA バージョン 4.6.2～17.6または、Keysight 社製IOライブラリ バージョン 15.5.13009.1～2018 update 1.0 がインストールされており、USB Test and Measurement Device (IVI) ドライバが動作する環境
Ethernet	TCP/IPが動作する環境（VXI-11含む）

3 ご使用にあたっての注意

免責事項

当社は，お客様が本ソフトウエアをダウンロードしインストールされた時点で、下記の免責事項を許諾いただいたものとみなします。

- 本ソフトウエアをダウンロードしインストールすることによって生じるいかなる問題についても、当社はその責務を負いません。
- 本ソフトウエアの使用に関して，直接または間接に生じるいっさいの損害について，当社はその責務を負いません。
- 本ソフトウエアは無償で提供されますが、本製品になんの欠陥もないという無制限の保証をするものではありません。また、本ソフトウエアに関する不具合修正や質問についてのお問い合わせをお受けできない場合があります。
- 本ソフトウエアに関する財産権、所有権、知的財産権、その他一切の権限は、当社に帰属します。

使用上の注意

- 本ソフトウエアは，当社製品のリモート制御専用ライブラリです。他の製品には使用できません。
- 本ソフトウエアのバージョン，および使用する当社製品のファームウェアバージョンをご確認の上，ご使用ください。

4 インストール / アンインストール

インストール方法

1. TMCTLライブラリを当社ホームページからダウンロードします。ご使用のPCの適切な場所にダウンロードしてください。
2. ダウンロードしたファイルは圧縮ファイルです。適切な解凍ソフトウエアを使って解凍します。「tmctlXXXX[*1]」というフォルダが作成されます。
3. Visual C++ ライブラリのランタイムをインストールします。解凍したフォルダ内のredistフォルダの中にある「VC_redist.x64.exe」を実行しインストールを完了させてください。
4. 使用するファイルを、アプリケーションが参照するフォルダ内に配置します。
   詳しくは、5.6節「 DLLリンク方式と配置」をご参照ください。

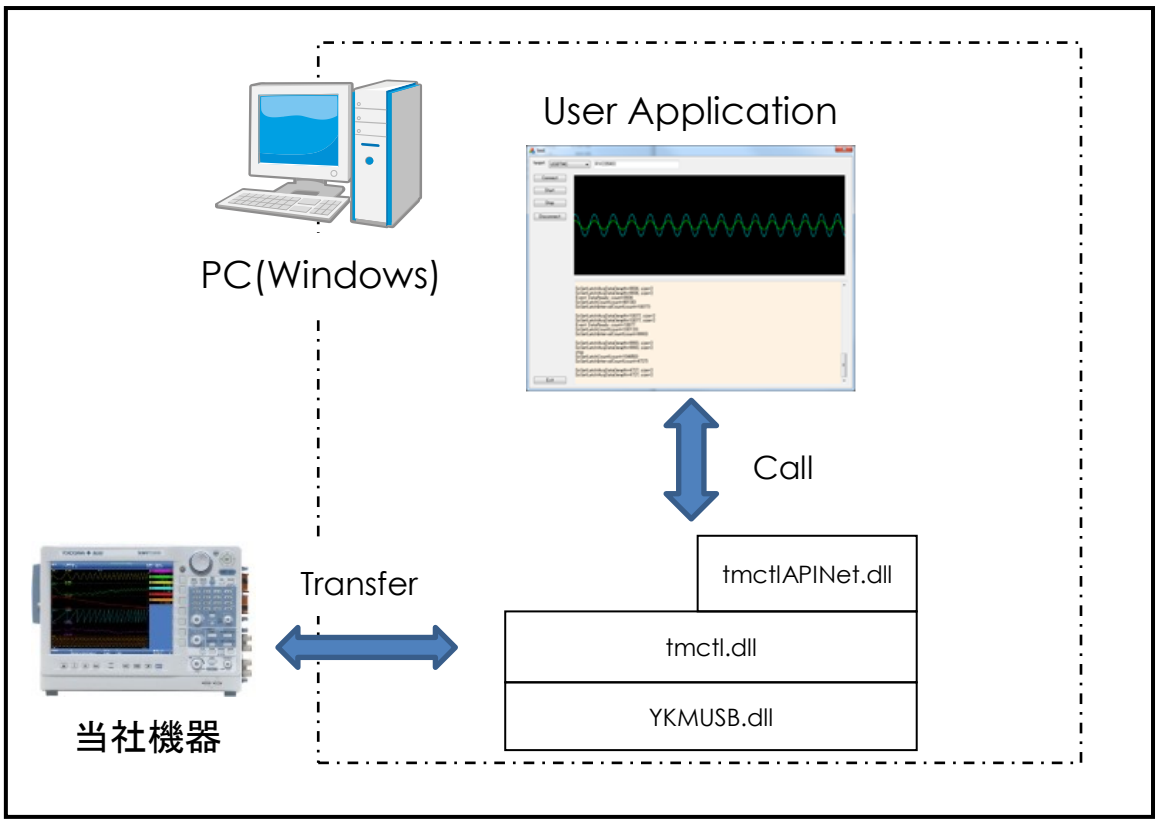
アンインストール方法

1. 「tmctlXXXX[*1]」のフォルダをフォルダごと削除してください。

[*1] XXXXはバージョンによって変わります。

5　TMCTL API 概要

API は，ダイナミックリンクライブラリ(DLL: Dynamic Link Library)として提供されます。ユーザー作成のアプリケーションと一緒に本 DLL をリンクすることで，API を利用できます。
TMCTL API は下図のように，機器に対してコマンドの送受信の機能をアプリケーションに提供します。



5.1 通信可能製品

5.1.1 GPIB

・当社製品で IEEE Std 488.2 準拠の GPIB インターフェースが搭載されているもの。
・当社以外の製品も通信は行なえますが、一部使用できない機能があります。
(詳しくは、6.3 節 API 詳細仕様」をご参照ください)

該当シリーズ

DLM4000, DLM3000, DLM2000, DL/DLM6000, DL850E/DL850EV, DL850/DL850V, DL750, DL750P, DL1600, DL1700E, DL1740, DL1720, DL9000, SB5000, DL7400, DL7200, DL7100, FG400, WT5000, WT3000E, WT3000, WT1800E, WT1800, WT1600, WT500, WT300E, WT300, PX8000, SL1400, GS820, GS610, GS200, 2553A, 2558A, 2560A, LS3300, AQ2211, AQ2212, DM7560, MT300

Note：

- 当社製品で通信を行なう場合のターミネータには、通常は LF 及び EOI、バイナリデータ転送時は EOI をご使用ください。
- PC 側 GPIB デバイスのドライバは、各 OS に対応したものを使用してください。PC 側 GPIB デバイスのドライバについてはメーカに確認してください。

5.1.2 RS232

・当社製品で RS232 搭載製品のうち、以下の設定が可能なもの
・当社以外の製品も通信は行なえますが、一部使用できない機能があります。
(詳しくは、6.3 節 API 詳細仕様」をご参照ください)

該当シリーズ

DL750, DL750P, DL1600, DL7200, DL7100, WT3000E, WT3000, WT1600, WT300E, WT300, SL1400, GS820, GS610, DM7560, MT300

設定項目

- ボーレート：
  - ・1200 , 2400 , 4800 , 9600 , 19200 , 38400 , 57600 , 115200
- データビット長とパリティ、ストップビットの組み合わせ：
  - ・8 ビット、ノーパリティ、1 ストップビット
  - ・7 ビット、イーブン、1 ストップビット
  - ・7 ビット、オッド、1 ストップビット
  - ・8 ビット、オッド、1 ストップビット
  - ・7 ビット、ノーパリティ、1.5 ストップビット
  - ・8 ビット、ノーパリティ、2 ストップビット
- ハンドシェーク：
  - ・NO-NO(ハンドシェークなし）
  - ・XON-XON(ソフトウェアハンドシェーク）
  - ・CTS-RTS(ハードウェアハンドシェーク）
- ターミネータ：
  - ・LF , CR+LF

Note :

・当社製品で通信を行なう場合は、通常は以下の設定にしてください。

・8 ビット、ノーパリティ、1 ストップビット

・CTS-RTS(ハードウェアハンドシェーク）

・ターミネータ LF

・DM7560 の USB は仮想 COM ポートドライバにより、RS232 として扱うことができます。

5.1.3 USB (Except USBTMC)

・当社製品で USB インターフェースが搭載されて、かつ USBTMC プロトコルに対応していないもの。

該当シリーズ

DL750, DL750P, DL1600, DL1700E, DL1740(ファームウェアバージョン 1.10 以降), DL1720, DL7400, WT3000E, WT3000(ファームウェアバージョン 2.01 以降), SL1400, AQ7270, AQ7260

Note :

・ターミネータは、LF 及び EOI、または EOI を設定してください。

・回線接続中は、PC および該当製品の電源を OFF にしないようにしてください。

5.1.4 USB (USBTMC)

・当社製品で USB インターフェースが搭載されており、かつ USBTMC プロトコルに対応しているもの。

該当シリーズ

DLM4000, DLM3000, DLM2000, DL/DLM6000, DL350, DL850E/DL850EV, DL850/DL850V, DL9000, SB5000, WT5000, WT1800E, WT1800, WT500, WT300E, WT300, PX8000, SL1000, GS820, GS610, GS200, 2553A, 2558A, 2560A, LS3300, AQ7280, AQ2211/AQ2212, AQ1300, AQ1210, AQ1200, AQ1100, AQ1000, MT300

Note :

・ターミネータは、LF 及び EOI、または EOI を設定してください。

5.1.5 USB (VISAUSB)

・当社製品で USB インターフェースが搭載されて、かつ IVI USB ドライバに対応しているもの。

該当シリーズ

DLM4000, DLM3000, DLM2000, DL350, DL850E/DL850EV, DL850/DL850V, FG400, WT5000, WT1800E, WT1800, WT500, WT300E, WT300, PX8000, SL1000, GS820, GS610, GS200, 2553A, 2558A, 2560A, LS3300, AQ7280, AQ1300, AQ1210, AQ1200, AQ1100, AQ1000, MT300

5.1.6 Ethernet (Legacy)

・当社製品で Ethernet インターフェースが搭載されており、かつ Server プロトコルに対応しているもの。

該当シリーズ

DL/DLM6000, DL750, DL750P, DL1600, DL1700E, DL1740(ファームウェアバージョン 1.30 以降), DL1720(ファームウェアバージョン 1.30 以降), DL9000, SB5000, DL7400, DL7200(ファームウェアバージョン 3.02 以降) , DL7100(ファームウェアバージョン 3.02 以降), WT3000E, WT3000(ファームウェアバージョン 2.01 以降), WT1600(ファームウェアバージョン 2.01 以降), SL1400, AQ7280, AQ7270, AQ2211/AQ2212, AQ1300, AQ1200, AQ1100

Note :

・接続時にユーザー認証があります。
(詳しくは、6.3 節 API 詳細仕様」をご参照ください)

5.1.7 Ethernet (VXI-11)

・当社製品で Ethernet インターフェースが搭載されており、かつ VXI-11 プロトコルに対応しているもの。

該当シリーズ

DLM4000, DLM3000, DLM2000, DL350, DL850E/DL850EV, DL850/DL850V, SB5000, WT5000, WT1800E, WT1800, WT500, WT300E, WT300, PX8000, SL1000, 2553A, 2558A, 2560A, LS3300, GS820, GS200, MT300,

5.1.8 Ethernet (UDP)

・当社製品で Ethernet インターフェースが搭載されており、かつ UDP プロトコルに対応しているもの。

該当シリーズ

ISDB-T 電波モニタ 732050

5.1.9 Ethernet (SOCKET)

・当社製品で Ethernet インターフェースが搭載されており、かつソケットコマンドプロトコルに対応しているもの。

該当シリーズ

GS610, GS820, GS200, DM7560, DLM3000

5.2 API 機能概要

API の機能概要について説明します。

5.2.1 接続開始 / 終了

接続に関する API を以下に示します。

API Name	Function	Page
Initialize	指定されたデバイスと接続を開始します	17
InitializeEx	指定されたデバイスと接続を開始します	24
Finish	デバイスとの接続を終了します	26
SearchDevices	回線につながっているデバイスを検索します	27
SearchDevicesEx	回線につながっているデバイスを検索します	29
EncodeSerialNumber	製品の銘板のシリアル番号を USB 内部シリアル番号に変換します	31
DecodeSerialNumber	USB 内部シリアル番号を製品の銘板のシリアル番号に変換します	32

5.2.2 通信設定

通信設定に関する API を以下に示します。

API Name	Function	Page
SetTimeout	通信のタイムアウト時間を設定します	33
SetTerm	メッセージの送受信における、ターミネータを設定します	34

5.2.3 送信 / 受信

デバイスへの送信および受信に関する API を以下に示します。

API Name	Function	Page
Send	デバイスへメッセージを送信します	35
SendByLength SendByLengthB	デバイスへメッセージを指定されたバイト数送信します	36
SendSetup	デバイスへメッセージを送信する準備をします	37
SendOnly SendOnlyB	デバイスへメッセージを指定されたバイト数送信します	38
Receive	デバイスから、メッセージを受信します	40
ReceiveSetup	デバイスから、メッセージを受信する準備をします	41
ReceiveOnly	デバイスから、(受信準備後に) メッセージを受信します	42
ReceiveBlockHeader	デバイスから、ブロックデータのバイト数を受信します	44
ReceiveBlockData ReceiveBlockB	デバイスから、(バイト数受信後に) ブロックデータを受信します	45
CheckEnd	デバイスからのメッセージが終了したかどうかを返します	50

5.2.4 情報取得

情報取得に関する API を以下に示します。

API Name	Function	Page
GetLastError	最後に発生したエラーのエラー番号を返します	48

5.2.5　その他の機能

動作に関する API を以下に示します。

API Name	Function	Page
SetRen	デバイスをリモート／ローカル状態にします	49
DeviceClear	選択されたデバイスのクリア(SDC)を実行します	51
DeviceTrigger	デバイスにトリガメッセージを送信します	52
WaitSRQ	指定されたデバイスの SRQ を受け付けます	53
AbortWaitSRQ	指定されたデバイスの SRQ 待ち関数の待ち状態を解除します	54
SetCallback	SRQ 発生時コールバックルーチンを登録します	55
ResetCallback	SRQ 発生時コールバックルーチンを削除します	57

5.3 使用方法（C#、VB.NET）

5.3.1 設定

使用ファイル名：
TmctlAPINet.dll (64bit 環境では TmctlAPINet64.dll)

・開発するプロジェクトに"TmctlAPINet.dll"(64bit 環境では"TmctlAPINet64.dll")を参照設定してください。
・アプリケーションが参照するフォルダに"tmctl.dll"と"YKMUSB.dll"を入れてください。
(64bit 環境では"tmctl64.dll"と"YKMUSB64.dll")

5.3.2 実行

本ライブラリは、ＰＣに接続されている制御対象のデバイスについて、初期化関数で回線をつなぎます。そして、その引数として返ってくる ID 値をそのデバイスの識別 ID とし、その他の送受信関数等は、その ID 値を使用して、制御します。

5.3.3 TMCTL クラス

TMCTL クラスは、tmctl ライブラリ（tmctl.dll/tmctl64.dll）に実装している関数をマネージアプリケーションに提供するためのクラスライブラリです。

名前空間名： TmctlAPINet
クラス名： TMCTL

5.4 使用方法（Visual C++）

5.4.1 設定

使用ファイル名：
tmctl.h (関数定義ヘッダファイル)
tmctl.lib (インポートライブラリ)　　(64bit 環境では"tmctl64.lib")

・ご使用になるソースファイルにインクルードファイルとして"tmctl.h"を追加してください。
#include "tmctl.h"
・リンクするライブラリファイルに、"tmctl.lib" (64bit 環境では"tmctl64.lib")を追加してください。
・アプリケーションが参照するディレクトリに"tmctl.dll"と"YKMUSB.dll"を入れてください。
(64bit 環境では"tmctl64.dll"と"YKMUSB64.dll")

5.4.2 実行

本ライブラリは、ＰＣに接続されている制御対象のデバイスについて、初期化関数で回線をつなぎます。そして、その引数として返ってくる ID 値をそのデバイスの識別 ID とし、その他の送受信関数等は、その ID 値を使用して、制御を行ないます。

5.5 使用方法（Visual Basic for Applications）

5.5.1 設定

使用ファイル名：
tmctl.bas (関数定義ファイル)
tmval.bas (定数定義ファイル)

・ご使用になるプロジェクトの標準モジュールに"tmctl.bas","tmval.bas"を追加してください。
・アプリケーションが参照するディレクトリに、"tmctl.dll"と"YKMUSB.dll"も入れてください。
(64bit Excel 環境では"tmctl64.dll"と”YKMUSB64.dll”)

5.5.2 実行

本ライブラリは、ＰＣに接続されている制御対象のデバイスについて、初期化関数で回線をつなぎます。そして、その引数として返ってくる ID 値をそのデバイスの識別 ID とし、その他の送受信関数等は、その ID 値を使用して、制御を行ないます。

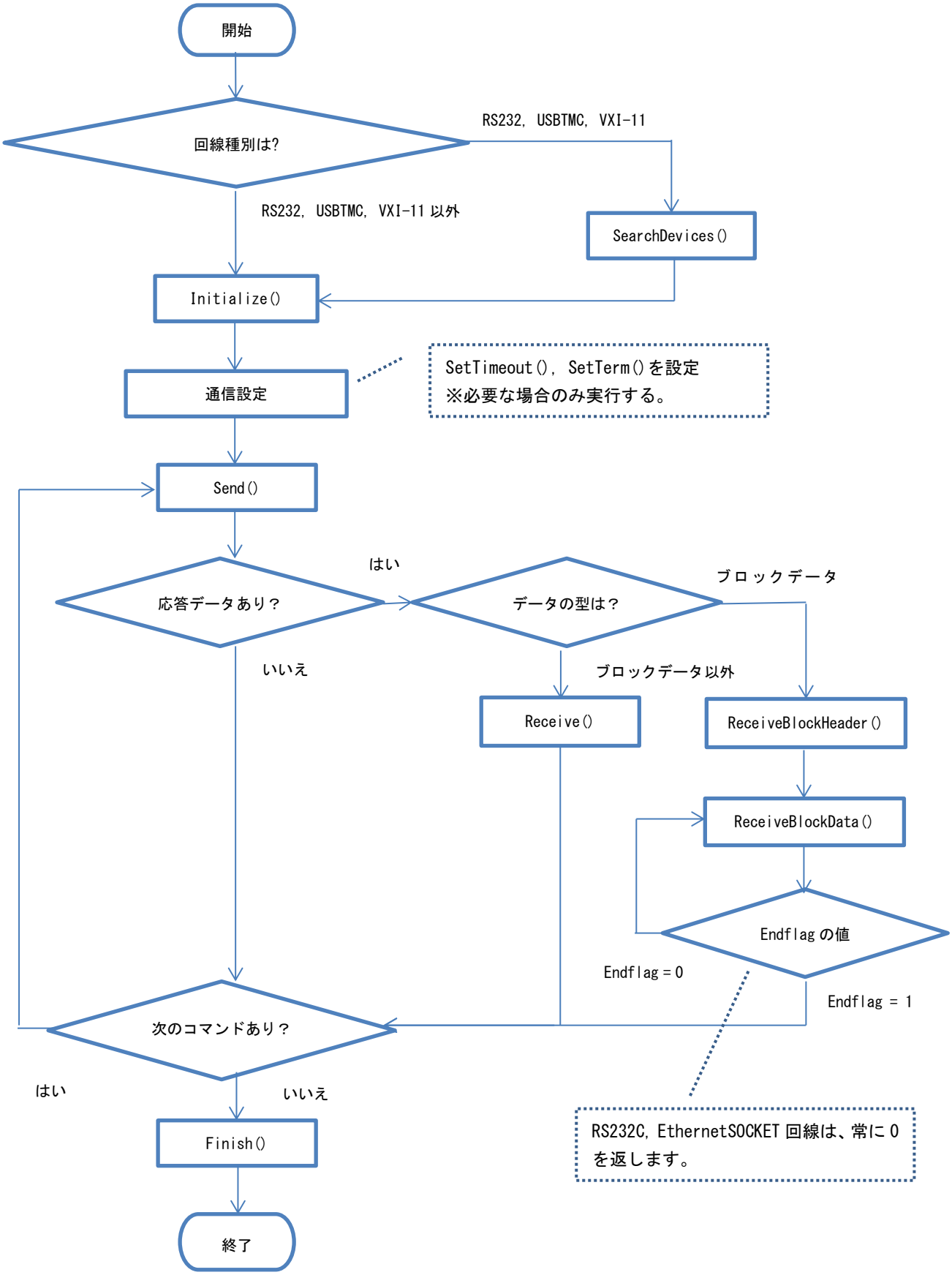
5.6 DLL リンク方式と配置

アプリケーションが参照するフォルダに以下の DLL を配置してください。

プロジェクト	VC++ / VBA		C# / VB.NET ※	
アーキテクチャ	32bit	64bit	32bit	64bit
tmctlAPINet.dll	－	－	○	－
tmctlAPINet64.dll	－	－	－	○
tmctl.dll	○	－	○	－
tmctl64.dll	－	○	－	○
YKMUSB.dll	○	－	○	－
YKMUSB64.dll	－	○	－	○

※C# / VB.NET の「Any CPU」アーキテクチャは現在、非対応です。
x86 または x64 を指定してください。

5.7　基本動作フローチャート



6　API 機能仕様

API の機能仕様について説明します。

6.1　エラー定義

GetLastError 関数が返すエラーコード一覧

コード	定義名	内容
0	TMCTL_NO_ERROR	エラーなし
1	TMCTL_TIMEOUT	タイムアウト
2	TMCTL_NO_DEVICE	対象デバイスが見つかりません
4	TMCTL_FAIL_OPEN	デバイスとの接続に失敗しました
8	TMCTL_NOT_OPEN	デバイスとの接続がされていません
16	TMCTL_DEVICE_ALREADY_OPEN	デバイスはすでに接続されています
32	TMCTL_NOT_CONTROL	パソコンが対応していません
64	TMCTL_ILLEGAL_PARAMETER	関数の引数不正です
256	TMCTL_SEND_ERROR	送信エラーです
512	TMCTL_RECV_ERROR	受信エラーです
1024	TMCTL_NOT_BLOCK	受信データはブロックデータではありません
4096	TMCTL_SYSTEM_ERROR	システムエラーです
8192	TMCTL_ILLEGAL_ID	デバイス ID 値が不正です
16384	TMCTL_NOT_SUPPORTED	サポートされていない機能です
32768	TMCTL_INSUFFICIENT_BUFFER	十分なバッファがありません
65536	TMCTL_LIBRARY_ERROR	ライブラリがありません

6.2 固定値定義

TMCTL クラス内の定数定義一覧

定義名	概要
TM_CTL_GPIB	回線指定（GPIB）
TM_CTL_RS232	回線指定（RS232C）
TM_CTL_USB	回線指定（USBTMC 以外の USB）
TM_CTL_ETHER	回線指定（Ethernet）
TM_CTL_USBTMC	回線指定（DL9000 専用 USBTMC）
TM_CTL_ETHERUDP	回線指定（Ethernet UDP）
TM_CTL_USBTMC2	回線指定（DL9000 以外の USBTMC）
TM_CTL_VXI11	回線指定（VXI-11）
TM_CTL_VISAUSB	回線指定（VISAUSB）
TM_CTL_SOCKET	回線指定（Ethernet SOCKET）
TM_CTL_USBTMC3	回線指定（USBTMC で YTUSB ドライバ使用）
TM_RS_1200	RS232 回線ボーレート（1200bps）
TM_RS_2400	RS232 回線ボーレート（2400bps）
TM_RS_4800	RS232 回線ボーレート（4800bps）
TM_RS_9600	RS232 回線ボーレート（9600bps）
TM_RS_19200	RS232 回線ボーレート（19200bps）
TM_RS_38400	RS232 回線ボーレート（38400bps）
TM_RS_57600	RS232 回線ボーレート（57600bps）
TM_RS_115200	RS232 回線ボーレート（115200bps）
TM_RS_8N	RS232 回線ビット仕様（8Bit, NoParity, 1StopBit）
TM_RS_7E	RS232 回線ビット仕様（7Bit, EvenParity, 1StopBit）
TM_RS_7O	RS232 回線ビット仕様（7Bit, OddParity, 1StopBit）
TM_RS_8O	RS232 回線ビット仕様（8Bit, OddParity, 1StopBit）
TM_RS_7N5	RS232 回線ビット仕様（7Bit, NoParity, 1.5StopBit）
TM_RS_8N2	RS232 回線ビット仕様（8Bit, NoParity, 2StopBit）
TM_RS_NO	RS232 回線ハンドシェーク番号（NO-NO）
TM_RS_XON	RS232 回線ハンドシェーク番号（XON-XON）
TM_RS_HARD	RS232 回線ハンドシェーク番号（CTS-RTS）
ADRMAXLEN	DEVICELIST 構造体で使用されるメンバ adr の文字列長（64）

6.3 API 詳細仕様

API の詳細仕様について説明します。

6.3.1 Initialize

概要: 回線を初期化し、指定されたデバイスとの回線をつなぎます。
書式: [C#] int Initialize(int wire, string adr, ref int id)
[VC++] int TmcInitialize(int wire, char* adr, int* id)
[VBA] TmInitialize(ByVal wire As Long, ByVal adr As String, ByRef id As Long) As Long
引数:

[IN] wire 回線種別

Class/VC++定義	VBA 定義		
TM_CTL_GPIB	CTL_GPIB	(1)	GPIB
TM_CTL_RS232	CTL_RS232	(2)	RS232
TM_CTL_USB	CTL_USB	(3)	USB
TM_CTL_ETHER	CTL_ETHER	(4)	Ethernet
TM_CTL_USBTMC	CTL_USBTMC	(5)	USBTMC (DL9000)
TM_CTL_ETHERUDP	CTL_ETHERUDP	(6)	Ethernet (UDP)
TM_CTL_USBTMC2	CTL_USBTMC2	(7)	USBTMC (DL9000 以外)
TM_CTL_VXI11	CTL_VXI11	(8)	VXI-11
TM_CTL_VISAUSB	CTL_VISAUSB	(10)	VISAUSB
TM_CTL_SOCKET	CTL_SOCKET	(11)	Ethernet (SOCKET)
TM_CTL_USBTMC3	CTL_USBTMC3	(12)	USBTMC (YTUSB ドライバ)

[IN] adr 接続先アドレス

対象回線	設定概要
GPIB	“<GPIB アドレス>” もしくは “<インターフェース ID>,<GPIB アドレス>” ※GPIB アドレス範囲は”0”～”30”です ※インターフェース ID 範囲は”0”～”99”です ※インターフェース ID は省略可能です。
RS232	“<com port number>,<baud rate>,<bit>,<handshaking number>” com port number : 1～255 baud rate : 0 = 1200 1 = 2400 2 = 4800 3 = 9600 4 = 19200 5 = 38400 6 = 57600 7 = 115200 bit : 0 = 8Bit,NoParity,1StopBit 1 = 7Bit,EvenParity,1StopBit 2 = 7Bit,OddParity,1StopBit 3 = 8Bit,OddParity,1StopBit 4 = 7Bit,NoParity,1.5StopBit 5 = 8Bit,NoParity,2StopBit handshaking number : 0 = NO-NO 1 = XON-XON 2 = CTS-RTS
USB	“<USB 識別 ID 値>”
Ethernet	“<Server name>,<username>,<password>” もしくは “<Server name>,<port>,<username>,<password>” ※username が “anonymous” の場合、password は必要ありません ※区切りのための、”,”カンマは必要です ※port は省略可能です

Ethernet(UDP)	“<Server name>, <Port number>”
USBTMC(DL9000)	“<serial number>”
USBTMC(GS200,GS820,FG400)	“<serial number>”
USBTMC(GS610)	“<serial number>” + “C”
USBTMC(DL90000,GS,FG400 以外)	“<serial number>” ※EncodeSerialNumber でエンコードした番号
VXI-11	“<IP address>”
VISAUSB(GS200,GS820,FG400)	“<serial number>”
VISAUSB(GS610)	“<serial number>” + “C”
VISAUSB(GS,FG400 以外)	“<serial number>” ※EncodeSerialNumber でエンコードした番号
Ethernet(SOCKET)	“<Server name>,<Port number>” ※区切りのための、″,″カンマは必要です
USBTMC(YTUSB ドライバ)	“<serial number>” ※EncodeSerialNumber でエンコードした番号

[OUT] id　　他の関数等で使用するそのデバイス専用の ID 値

戻り値:

0　　接続成功

1　　接続エラー

詳細:

なし

注意:

複数のデバイスを使用する場合は、使用するデバイスごとに本関数を実行してください。

最大 127 デバイスを同時に使用できます。

使用例:[C#]

```
TMCTL cTmctl = new TMCTL();
int id = 0;
[GPIB] address = 1
   ret = cTmctl.Initialize(TMCTL.TM_CTL_GPIB, "1", ref id );
[RS232] COM1,57600,8-NO-1,CTS-RTS
   ret = cTmctl.Initialize(TMCTL.TM_CTL_RS232, "1,6,0,2", ref id );
[USB] ID = 1
   ret = cTmctl.Initialize(TMCTL.TM_CTL_USB, "1", ref id );
[Ethernet] IP = 11.22.33.44, User name = anonymous"
   ret = cTmctl.Initialize(TMCTL.TM_CTL_ETHER, "11.22.33.44,anonymous,", ref id );
[Ethernet] IP = 11.22.33.44, User name = yokogawa, Password = abcdefgh
   ret = cTmctl.Initialize(TMCTL.TM_CTL_ETHER, "11.22.33.44,yokogawa,abcdefgh", ref id );
[USBTMC(DL9000)] Serial number = 27E000001
   ret = cTmctl.Initialize(TMCTL.TM_CTL_USBTMC, "27E000001", ref id );
[USBTMC(GS200,GS820,FG400)] Serial number = 27E000001
   ret = cTmctl.Initialize(TMCTL.TM_CTL_USBTMC2, "27E000001", ref id );
[USBTMC(GS610)] Serial number = 27E000001
   ret = cTmctl.Initialize(TMCTL.TM_CTL_USBTMC2, "27E000001C", ref id );
[USBTMC(DL9000,GS シリーズ以外)] Serial number = 27E000001
   StringBuilder encode = new StringBuilder(100); // インスタンスを生成する
   ret = cTmctl.EncodeSerialNumber(encode,encode.Capacity,"27E000001");
   ret = cTmctl.Initialize(TMCTL.TM_CTL_USBTMC2, encode.ToString(), ref id );
[VXI-11] IP = 11.22.33.44
   ret = cTmctl.Initialize(TMCTL.TM_CTL_VXI11, "11.22.33.44", ref id );
[VISAUSB(GS シリーズ, FG400 以外)] serial number = 27E000001
   StringBuilder encode = new StringBuilder(100); // インスタンスを生成する
   ret = cTmctl.EncodeSerialNumber(encode, encode.Capacity,"27E000001")
   ret = cTmctl.Initialize(TMCTL.TM_CTL_VISAUSB, encode.ToString(), ref id )
[Ethernet(SOCKET)] IP = 11.22.33.44, Port number = 5198
   ret = cTmctl.Initialize(TMCTL.TM_CTL_SOCKET, "11.22.33.44,5198", ref id );
[USBTMC(YTUSB ドライバ)] Serial number = 27E000001
   StringBuilder encode = new StringBuilder(100); // インスタンスを生成する
   ret = cTmctl.EncodeSerialNumber(encode,encode.Capacity,"27E000001");
   ret = cTmctl.Initialize(TMCTL.TM_CTL_USBTMC3, encode.ToString(), ref id );
```

使用例:[VC++]

```
[GPIB] address = 1
   int id;
   int ret = TmcInitialize( TM_CTL_GPIB, "1", &id );
[RS232] COM1,57600,8-NO-1,CTS-RTS
   int id;
   int ret = TmcInitialize( TM_CTL_RS232, "1,6,0,2", &id );
[USB] ID = 1
   int id;
   int ret = TmcInitialize( TM_CTL_USB, "1", &id );
[Ethernet] IP = 11.22.33.44, User name = anonymous
   int id;
   int ret = TmcInitialize( TM_CTL_ETHER, "11.22.33.44,anonymous,", &id );
[Ethernet] IP = 11.22.33.44, User name = yokogawa, Password = abcdefgh
   int id;
   int ret = TmcInitialize( TM_CTL_ETHER, "11.22.33.44,yokogawa,abcdefgh", &id );
[USBTMC(DL9000)] Serial number = 27E000001
   int id;
   int ret = TmcInitialize( TM_CTL_USBTMC, "27E000001", &id );
[USBTMC(GS200,GS820,FG400)] Serial number = 27E000001
   int id;
   int ret = TmcInitialize( TM_CTL_USBTMC2, "27E000001", &id );
[USBTMC(GS610)] Serial number = 27E000001
   int id;
   int ret = TmcInitialize( TM_CTL_USBTMC2, "27E000001C", &id );
[USBTMC(DL9000,GS シリーズ以外)] Serial number = 27E000001
' SearchDevices を使用するとき
   int id;
   int ret;
   DEVICELIST list[127];
   int num;
   ret = TmcSearchDevices(TM_CTL_USBTMC2,list,127,&num,NULL);
   ret = TmcInitialize( TM_CTL_USBTMC2, list[0].adr, &id );
'直接シリアル番号を指定するとき
   int id;
   char encode[256];
   int ret;
   ret = TmcEncodeSerialNumber(encode,256,"27E000001");
   ret = TmcInitialize( TM_CTL_USBTMC2, encode, &id );
[VXI-11] IP = 11.22.33.44
   int id;
   int ret = TmcInitialize( TM_CTL_VXI11, "11.22.33.44", &id );
[VISAUSB(GS シリーズ, FG400 以外)] Serial number = 27E000001
'SearchDevices を使用するとき
   int id;
   int ret;
   DEVICELIST list[127];
   int num;
   ret = TmcSearchDevices(TM_CTL_VISAUSB,list,127,&num,NULL);
   ret = TmcInitialize( TM_CTL_VISAUSB, list[0].adr, &id );
'直接シリアル番号を指定するとき
   int id;
   char encode[256];
   int ret;
   ret = TmcEncodeSerialNumber(encode,256,"27E000001") ;
   ret = TmcInitialize( TM_CTL_VISAUSB, encode, &id );
[Ethernet(SOCKET)] IP = 11.22.33.44, Port number = 7655
   int id;
   int ret = TmcInitialize( TM_CTL_ETHER, "11.22.33.44,7655", &id );
[USBTMC(YTUSB ドライバ)] Serial number = 27E000001
' SearchDevices を使用するとき
```

```
    int id;
    int ret;
    DEVICELIST list[127];
    int num;
    ret = TmcSearchDevices(TM_CTL_USBTMC3,list,127,&num,NULL);
    ret = TmcInitialize( TM_CTL_USBTMC3, list[0].adr, &id );
'直接シリアル番号を指定するとき
    int id;
    char encode[256];
    int ret;
    ret = TmcEncodeSerialNumber(encode,256,"27E000001");
    ret = TmcInitialize( TM_CTL_USBTMC3, encode, &id );
```

使用例:[VBA]

```
[GPIB] address = 1
   Dim id As Long
   Dim ret As Long
   Dim adr As String
   adr = "1"
   ret = TmInitialize( 1, adr, id )

[RS232] COM1,57600,8-NO-1,CTS-RTS
   Dim id As Long
   Dim ret As Long
   Dim adr As String
   adr = "1,6,0,2"
   ret = TmInitialize( 2, adr, id )

[USB] ID = 1
   Dim id As Long
   Dim ret As Long
   Dim adr As String
   adr = "1"
   ret = TmcInitialize( 3, adr, id )

[Ethernet] IP = 11.22.33.44, User name = anonymous
   Dim id As Long
   Dim ret As Long
   Dim adr As String
   adr = "11.22.33.44,anonymous,"
   ret = TmInitialize( 4, adr, id )

[Ethernet] IP = 11.22.33.44, User name = yokogawa, Password = abcdefgh
   Dim id As Long
   Dim ret As Long
   Dim adr As String
   adr = "11.22.33.44,yokogawa,abcdefgh"
   ret = TmInitialize( 4, adr, id )

[USBTMC(DL9000)] Serial number = 27E000001
   Dim id As Long
   Dim ret As Long
   Dim adr As String
   adr = "27E000001"
   ret = TmcInitialize( 5, adr, id )

[USBTMC(GS200,GS820)] Serial number = 27E000001
   Dim id As Long
   Dim ret As Long
   Dim adr As String
   adr = "27E000001"
   ret = TmcInitialize( 7, adr, id )

[USBTMC(GS610)] Serial number = 27E000001
   Dim id As Long
   Dim ret As Long
   Dim adr As String
   adr = "27E000001" & "C"
   ret = TmcInitialize( 7, adr, id )
```

```
[USBTMC(DL9000,GS シリーズ以外)] Serial number = 27E000001
' SearchDevices を使用するとき
   Dim id As Long
   Dim ret As Long
   Dim list As DeviceListArray
   Dim num As Long
   ret = TmSearchDevices(7, list, 128, num, 0)
   ret = TmInitialize(7, list.list(0).adr, id)

'直接シリアル番号を指定するとき
   Dim id As Long
   Dim ret As Long
   Dim encode As String * 128
   ret = TmEncodeSerialNumber(encode, 128, "27E000001")
   ret = TmInitialize(7, encode, id)

[VXI-11] IP = 11.22.33.44
   Dim id As Long
   Dim ret As Long
   ret = TmInitialize( 8, "11.22.33.44", id )

[VISAUSB(GS シリーズ,FG400 以外)] Serial number = 27E000001
' SearchDevices を使用するとき
   Dim id As Long
   Dim ret As Long
   Dim list As DeviceListArray
   Dim num As Long
   ret = TmSearchDevices(10, list, 128, num, 0)
   ret = TmInitialize(10, list.list(0).adr, id)

'直接シリアル番号を指定するとき
   Dim id As Long
   Dim ret As Long
   Dim encode As String * 128
   ret = TmEncodeSerialNumber(encode, 128, "27E000001")
   ret = TmInitialize(10, encode, id)

[Ethernet(SOCKET)] IP = 11.22.33.44, Port number = 7655
   Dim id As Long
   Dim ret As Long
   Dim adr As String
   adr = "11.22.33.44,7655"
   ret = TmInitialize( 11, adr, id )

[USBTMC(YTUSB ドライバ)] Serial number = 27E000001
' SearchDevices を使用するとき
   Dim id As Long
   Dim ret As Long
   Dim list As DeviceListArray
   Dim num As Long
   ret = TmSearchDevices(12, list, 128, num, 0)
   ret = TmInitialize(12, list.list(0).adr, id)

'直接シリアル番号を指定するとき
   Dim id As Long
   Dim ret As Long
   Dim encode As String * 128
   ret = TmEncodeSerialNumber(encode, 128, "27E000001")
   ret = TmInitialize(12, encode, id)
```

6.3.2 InitializeEx

概要: 回線を初期化し、指定されたデバイスとの回線をつなぎます（タイムアウトあり)。
書式: [C#] int InitializeEx(int wire, string adr, ref int id, int tmo)
[VC++] int TmcInitializeEx(int wire, char* adr, int* id, int tmo)
[VBA] TmInitializeEx(ByVal wire As Long, ByVal adr As String, ByRef id As Long, _
ByVal tmo As Long) As Long

引数:

[IN]	wire	回線種別	※Initialize と同様の定義 (6.3.1 参照)
[IN]	adr	接続先アドレス	※Initialize と同様の定義 (6.3.1 参照)
[OUT]	id	他の関数等で使用するそのデバイス専用の ID 値	
[IN]	tmo	タイムアウト(100 ミリ秒単位) Ethernet・VXI-11・SOCKET のとき有効。それ以外は無視されます。	

戻り値:

0	接続成功
1	接続エラー

詳細:
なし

注意:
複数のデバイスを使用する場合は、使用するデバイスごとに本関数を実行してください。
最大 127 デバイスを同時に使用できます。

使用例:[C#]

```
TMCTL cTmctl = new TMCTL();
int id = 0;
int tmo = 100;

[Ethernet] IP = 10.0.20.30, User name = anonymous"
   ret = cTmctl.InitializeEx(TMCTL.TM_CTL_ETHER, "10.0.20.30,anonymous,", ref id, tmo);
[Ethernet] IP = 10.0.20.30, User name = user, Password = abcd
   ret = cTmctl.InitializeEx(TMCTL.TM_CTL_ETHER, "10.0.20.30,user,abcd", ref id, tmo);
[VXI-11] IP = 10.0.20.30
   ret = cTmctl.InitializeEx(TMCTL.TM_CTL_VXI11, "10.0.20.30", ref id, tmo);
[Ethernet(SOCKET)] IP = 10.0.20.30, Port number = 10002
   ret = cTmctl.InitializeEx(TMCTL.TM_CTL_SOCKET, "10.0.20.30,10002", ref id, tmo);
```

使用例:[VC++]

```
int id;
int tmo = 100;

[Ethernet] IP = 10.0.20.30, User name = anonymous
   int ret = TmcInitializeEx( TM_CTL_ETHER, "10.0.20.30,anonymous,", &id, tmo);
[Ethernet] IP = 10.0.20.30, User name = user, Password = abcd
   int ret = TmcInitializeEx( TM_CTL_ETHER, "10.0.20.30,user,abcd", &id, tmo);
[VXI-11] IP = 10.0.20.30
   int ret = TmcInitializeEx( TM_CTL_VXI11, "10.0.20.30", &id, tmo);
[Ethernet(SOCKET)] IP = 10.0.20.30, Port number = 10002
   int ret = TmcInitializeEx( TM_CTL_ETHER, "10.0.20.30,10002", &id, tmo );
```

使用例:[VBA]

```
Dim id As Long
Dim ret As Long
Dim adr As String
Dim tmo As Long
tmo = 100

[Ethernet] IP = 10.0.20.30, User name = anonymous
   adr = "10.0.20.30,anonymous,"
   ret = TmInitializeEx( 4, adr, id, tmo )
[Ethernet] IP = 10.0.20.30, User name = user, Password = abcd
   adr = "10.0.20.30,user,abcd"
   ret = TmInitializeEx( 4, adr, id, tmo )
[VXI-11] IP = 10.0.20.30
   ret = TmInitializeEx( 8, "10.0.20.30", id, tmo )
[Ethernet(SOCKET)] IP = 10.0.20.30, Port number = 10002
   adr = "10.0.20.30,10002"
   ret = TmInitialize( 11, adr, id, tmp )
```

6.3.3 Finish

概要: デバイスとつなげている回線を閉じます。
書式: [C#] int Finish(int id)
[VC++] int TmcFinish(int id)
[VBA] TmFinish(ByVal id As Long) As Long
引数:

[IN]	id	デバイス ID 値

戻り値:

0	成功
1	エラー

詳細:
"Initialize"(初期化関数)で開いた回線を閉じます。
通信を終了する時には、必ずこの関数を実行してください。

注意:
なし

使用例:[C#]
```
int ret = cTmctl.Finish( id );
```

使用例:[VC++]
```
int ret = TmcFinish( id );
```

使用例:[VBA]
```
Dim ret As Long
ret = TmFinish( id )
```

6.3.4 SearchDevices

概要: 指定された回線につながっているデバイスのリストを返します。

書式: [C#] int SearchDevices(int wire, DEVICELIST[] list, int max,
ref int num, string option)

[VC++] int TmcSearchDevices(int wire, DEVICELIST* list, int max, int* num, char* option)

[VBA] TmSearchDevices(ByVal wire As Long, list As DeviceListArray, ByVal max As Long, _
ByRef num As Long, ByVal option1 As String) As Long

引数:

[IN] wire 回線種別

回線種別	対応
GPIB	サポートなし
RS232	TM_CTL_RS232
USB	サポートなし
Ethernet	サポートなし
USBTMC(DL9000)	サポートなし
Ethernet(UDP)	サポートなし
USBTMC(DL9000 以外)	TM_CTL_USBTMC2
VXI-11	TM_CTL_VXI11
VISAUSB	TM_CTL_VISAUSB
Ethernet(SOCKET)	サポートなし
USBTMC(YTUSB ドライバ)	TM_CTL_USBTMC3

[OUT] list 見つかったデバイスを示す文字列が入る配列へのポインタ

GPIB・RS232・USB・Ethernet・USBTMC(DL9000)は未サポート。

RS232 のとき、ポート番号が返る。

USBTMC のとき、エンコードしたシリアル番号が返る。

VXI-11 のとき、ip アドレスが返る。

VISAUSB のとき、エンコードしたシリアル番号が返る。

USBTMC(YTUSB ドライバ)のとき、エンコードしたシリアル番号が返る。

```
DEVICELIST[]       list
public    struct DEVICELIST
{
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = TMCTL.ADRMAXLEN)]
    public string   adr;
}
```

[IN] max 見つかったデバイスを示す文字列の配列数

[OUT] num 見つかったデバイスの数

[IN] option 回線ごとに必要な引数

回線種別	対応
RS232	不要
USBTMC	不要
VXI-11	ブロードキャストアドレスを文字列にしたもの
VISAUSB	不要
USBTMC(YTUSB ドライバ)	不要
その他	未定

戻り値:

0 成功

1 エラー

詳細:

なし

注意:

なし

使用例:[C#]

```
int ret ;
DEVICELIST[] list = new DEVICELIST[127];
int num = 0;

// USBTMC(DL9000 以外)のとき
ret = cTmctl.SearchDevices(TMCTL.TM_CTL_USBTMC2, list, 127, ref num, "");
ret = cTmctl.Initialize(TMCTL.TM_CTL_USBTMC2, list[0].adr.ToString(), id);

// VXI-11 のとき
ret = cTmctl.SearchDevices(TMCTL.TM_CTL_VXI11, list, 127, ref num, "192.168.255.255");
ret = cTmctl.Initialize(TMCTL.TM_CTL_VXI11, list[0].adr.ToString(), id);
```

使用例:[VB.NET]

```
Dim ret As long
Dim list As DEVICELIST()
Dim num As Integer

ReDim Preserve list(127)
num = 0

‘ USBTMC(DL9000 以外)のとき
ret = cTmctl.SearchDevices(TMCTL.TM_CTL_USBTMC2, list, 127, num, "")
ret = cTmctl.Initialize(TMCTL.TM_CTL_USBTMC2, list(0).adr, id)

‘ VXI-11 のとき
ret = cTmctl.SearchDevices(TMCTL.TM_CTL_VXI11, list, 127, num, "192.168.255.255");
ret = cTmctl.Initialize(TMCTL.TM_CTL_VXI11, list(0).adr, id);
```

使用例:[VC++]

```
int               ret ;
DEVICELIST        list[127] ;
int               num ;

// USBTMC(DL9000 以外)のとき
ret = TmcSearchDevices(TM_CTL_USBTMC2, list, 127, &num, NULL);
ret = TmcInitialize(TM_CTL_USBTMC2, list[0].adr, id);

// VXI-11 のとき
ret = TmcSearchDevices(TM_CTL_VXI11, list, 127, num, "192.168.255.255");
ret = TmcInitialize(TM_CTL_VXI11, list[0].adr, id);
```

使用例:[VBA]

```
Dim ret As long
Static list As DeviceListArray
Dim num As long

‘  USBTMC(DL9000 以外)のとき
ret = TmSearchDevices(CTL_USBTMC2, list, MaxStationNum, num, 0)
ret = TmInitialize(CTL_USBTMC2, listarray.list(0).adr, id)

‘  VXI-11 のとき
ret = TmSearchDevices(CTL_VXI11, list, MaxStationNum, num, "192.168.255.255")
ret = TmInitialize(CTL_VXI11, listarray.list(0).adr, id)
```

6.3.5 SearchDevicesEx

概要: 指定された回線につながっているデバイスのリストを返します。
書式: [C#] int SearchDevicesEx(int wire, DEVICELISTEx[] list, int max,
ref int num, string option)
[VC++] int TmcSearchDevicesEx(int wire, DEVICELISTEX* list, int max, int* num, char* option)
[VBA] TmSearchDevicesEx(ByVal wire As Long, list As DeviceListExArray, ByVal max As Long,_
ByRef num As Long, ByVal option1 As String) As Long

引数:

[IN] wire 回線種別

回線種別	対応
GPIB	サポートなし
RS232	TM_CTL_RS232
USB	サポートなし
Ethernet	サポートなし
USBTMC(DL9000)	サポートなし
Ethernet(UDP)	サポートなし
USBTMC(DL9000 以外)	TM_CTL_USBTMC2
VXI-11	TM_CTL_VXI11
VISAUSB	TM_CTL_VISAUSB
Ethernet(SOCKET)	サポートなし
USBTMC(YTUSB ドライバ)	TM_CTL_USBTMC3

[OUT] list 見つかったデバイスを示す文字列が入る配列へのポインタ
USBTMC のとき、エンコードしたシリアル番号、プロダクト ID、ベンダーID が返る。
VISAUSB のとき、エンコードしたシリアル番号、プロダクト ID、ベンダーID が返る。
USBTMC(YTUSB ドライバ)のとき、エンコードしたシリアル番号、プロダクト ID、
ベンダーID が返る。

```
DEVICELISTEx[]     list
public    struct DEVICELISTEx
{
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = TMCTL.ADRMAXLEN)]
    public string adr;
    public string dummy;
    public ushort productID;
    public ushort vendorID;
}
```

[IN] max 見つかったデバイスを示す文字列の配列数
[OUT] num 見つかったデバイスの数
[IN] option 回線ごとに必要な引数

回線種別	対応
RS232	不要
USBTMC	不要
VXI-11	ブロードキャストアドレスを文字列にしたもの
VISAUSB	不要
USBTMC(YTUSB ドライバ)	不要
その他	未定

戻り値:

0	成功
1	エラー

詳細:
なし
注意:
なし

使用例:[C#]

```
int ret ;
DEVICELISTEx[] list = new DEVICELIST[127];
int num = 0;

// USBTMC(DL9000 以外)のとき
ret = cTmctl.SearchDevicesEx(TMCTL.TM_CTL_USBTMC2, list, 127, ref num, "");
ret = cTmctl.Initialize(TMCTL.TM_CTL_USBTMC2, list[0].adr.ToString(), id);

// VXI-11 のとき
ret = cTmctl.SearchDevicesEx(TMCTL.TM_CTL_VXI11, list, 127, ref num, "192.168.255.255");
ret = cTmctl.Initialize(TMCTL.TM_CTL_VXI11, list[0].adr.ToString(), id);
```

使用例:[VB.NET]

```
Dim ret As long
Dim list As DEVICELISTEx()
Dim num As Integer

ReDim Preserve list(127)
num = 0

‘ USBTMC(DL9000 以外)のとき
ret = cTmctl.SearchDevicesEx(TMCTL.TM_CTL_USBTMC2, list, 127, num, "")
ret = cTmctl.Initialize(TMCTL.TM_CTL_USBTMC2, list(0).adr, id)

‘ VXI-11 のとき
ret = cTmctl.SearchDevicesEx(TMCTL.TM_CTL_VXI11, list, 127, num, "192.168.255.255");
ret = cTmctl.Initialize(TMCTL.TM_CTL_VXI11, list(0).adr, id);
```

使用例:[VC++]

```
int             ret ;
DEVICELISTEX    list[127] ;
int             num ;

// USBTMC(DL9000 以外)のとき
ret = TmcSearchDevicesEx(TM_CTL_USBTMC2, list, 127, &num, NULL);
ret = TmcInitialize(TM_CTL_USBTMC2, list[0].adr, id);

// VXI-11 のとき
ret = TmcSearchDevicesEx(TM_CTL_VXI11, list, 127, num, "192.168.255.255");
ret = TmcInitialize(TM_CTL_VXI11, list[0].adr, id);
```

使用例:[VBA]

```
Dim ret As long
Static listEx As DeviceListExArray
Dim num As long

‘  USBTMC(DL9000 以外)のとき
ret = TmSearchDevicesEx(CTL_USBTMC2, list, MaxStationNum, num, 0)
ret = TmInitialize(CTL_USBTMC2, listEx.list(0).adr, id)

‘  VXI-11 のとき
ret = TmSearchDevicesEx(CTL_VXI11, list, MaxStationNum, num, "192.168.255.255")
ret = TmInitialize(CTL_VXI11, listEx.list(0).adr, id)
```

6.3.6 EncodeSerialNumber

概要: 銘板のシリアル番号を USB 内部シリアル番号に変換します。
書式: [C#] int EncodeSerialNumber(StringBuilder encode, int len, string src)
[VC++] int TmcEncodeSerialNumber(char* encode, size_t len, char* src)
[VBA] TmEncodeSerialNumber(ByVal encode As String, ByVal encodelen As Long, _
ByVal src As String) As Long
引数:

[OUT]	encode	変換した文字列を格納するバッファ（USB 内部シリアル番号）
[IN]	len	encode のバッファの大きさ（バイト数）
[IN]	src	銘板に書かれているシリアル番号文字列

戻り値:

0	成功
0 以外	エラー番号

詳細:
なし
注意:
なし

使用例:[C#]

```
StringBuilder encode = New StringBuilder(100);
// 銘板のシリアル番号を USB 内部シリアル番号に変換する。
ret = cTmctl.EncodeSerialNumber(encode, encode.Capacity, "12W929658");
ret = cTmctl.Initialize(TMCTL.TM_CTL_USBTMC2, encode.ToString(), ref id);
```

使用例:[VB.NET]

```
Dim encode As StringBuilder

encode = New StringBuilder(100)
' 銘板のシリアル番号を USB 内部シリアル番号に変換する。
ret = cTmctl.EncodeSerialNumber(encode, encode.Capacity, "12W929658")
ret = cTmctl.Initialize(TMCTL.TM_CTL_USBTMC2, encode.ToString(), ref id)
```

使用例:[VC++]

```
char     encode[256] ;
int ret ;
// 銘板のシリアル番号を USB 内部シリアル番号に変換する。
ret = TmcEncodeSerialNumber(encode,256,"12W929658") ;

ret = TmcInitialize(TM_CTL_USBTMC2, encode, &id) ;
```

使用例:[VBA]

```
Dim ret As Long
Dim encode As String * 128
' 銘板のシリアル番号を USB 内部シリアル番号に変換する。
ret = TmEncodeSerialNumber(encode, 128, "TEMP01")
ret = TmInitialize(7, encode, id)
```

6.3.7 DecodeSerialNumber

概要: USB 内部シリアル番号を銘板のシリアル番号に変換します。
書式: [C#] int DecodeSerialNumber(StringBuilder decode, int len, string src)
[VC++] int TmcDecodeSerialNumber(char* decode, size_t len, char* src)
[VBA] TmDecodeSerialNumber(ByVal decode As String, ByVal decodelen As Long, _
ByVal src As String) As Long
引数:

[OUT]	decode	変換した文字列を格納するバッファ(銘板に書かれている機器のシリアル番号)
[IN]	len	decode のバッファの大きさ（バイト数）
[IN]	src	USB 内部シリアル番号文字列

戻り値:

0	成功
0 以外	エラー番号

詳細:
なし
注意:
なし

使用例:[C#]

```
DEVICELIST[] list = new DEVICELIST[127];
StringBuilder encode = New StringBuilder(100);
ret = cTmctl.SearchDevices(TMCTL.TM_CTL_USBTMC2, list, 127, ref listnum, "");
ret = cTmctl.DecodeSerialNumber(encode, encode.Capacity, list[0].adr.ToString());
ret = cTmctl.Initialize(TMCTL.TM_CTL_USBTMC2, encode.ToString(), ref id);
```

使用例:[VB.NET]

```
Dim encode As StringBuilder
Dim decode As StringBuilder
‘ Encode/Decode 関数の動作確認
‘ Encode は、USBTMC(TM_CTL_USBTMC2)のシリアル番号の設定に必要
encode = New StringBuilder(100)
decode = New StringBuilder(100)
Console.WriteLine("EncodeSerialNumber:len={0}", encode.Length)
ret = cTmctl.EncodeSerialNumber(encode, encode.Capacity, "12W929658")
Console.WriteLine("EncodeSerialNumber:ret={0} encode={1}", ret, encode)
ret = cTmctl.DecodeSerialNumber(decode, decode.Capacity, encode.ToString())
Console.WriteLine("DecodeSerialNumber:ret={0} decode={1}", ret, decode)
```

使用例:[VC++]

```
char decode[256] ;
int ret ;
ret = TmcDecodeSerialNumber(decode,256,"313257393239363538") ;// decode = “12W929658”
```

使用例:[VBA]

```
Dim ret As Long
Dim decode As String * 128
Dim encode As String * 128
ret = TmEncodeSerialNumber(encode, 128, "TEMP01")
ret = TmDecodeSerialNumber(decode, 128, encode)
’ decode = “TEMP01”
```

6.3.8 SetTimeout

概要: 通信のタイムアウト時間を設定します。
書式: [C#] int SetTimeout(int id, int tmo)
[VC++] int TmcSetTimeout(int id, int tmo)
[VBA] TmSetTimeout(ByVal id As Long, ByVal tmo As Long) As Long
引数:
[IN] id デバイス ID 値

[IN] tmo タイムアウト時間 (100 ミリ秒単位) (0～65536)
tmo = 0 の場合は、
GPIB,RS232,ETHER : タイムアウト時間無限
その他 : タイムアウト時間なし
戻り値:
0 成功
1 エラー
詳細:
通信のタイムアウト時間を設定します。Initialize 後の初期値は 30 秒です。
当社製品の場合、30 秒以上を推奨します。

注意:
GPIB の場合、タイムアウトは以下の決められた値になります。
(100msec,300msec,1sec,3sec,10sec,30sec,100sec,300sec,1000sec)

使用例:[C#]
```
int ret = cTmctl.SetTimeout( id, 100 );/* 10sec */
```

使用例:[VC++]
```
int ret = TmcSetTimeout( id, 100 );   /* 10sec */
```

使用例:[VBA]
```
Dim ret As Long
ret = TmSetTimeout( id, 100 )        ' 10sec
```

6.3.9 SetTerm

概要: メッセージの送受信における、ターミネータを設定します。
書式: [C#] int SetTerm(int id, int eos, int eot)
[VC++] int TmcSetTerm(int id, int eos, int eot)
[VBA] TmSetTerm(ByVal id As Long, ByVal eos As Long, ByVal eot As Long) As Long
引数:
[IN] id デバイス ID 値

[IN] eos ターミネータ

設定値	説明
0	CR+LF
1	CR
2	LF
3	EOI(GPIB)またはなし(RS232、USB、Ethernet)

※回線が GPIB で、eos が 3 のときに EOI を使用するかどうかは、eot で設定します。

[IN] eot EOI の使用 (GPIB 専用)

設定値	説明
0	使用しない
1	使用する

戻り値:
0 成功
1 エラー
詳細:
ターミネータを設定します。Initialize 後の初期値は「eos = 2, eot = 1」です。
当社製品の場合は、初期値のまま使うことを推奨します。
eos = 2(LF)の設定のままだと、LF コードが含まれている binary データを受信すると、
そこで終了と判断してしまいます。ただし、当社製品で、"ReceiveBlockHeader"または
"ReceivceBlockData"を使ってブロックデータを受信する場合、ターミネータを切り替える
必要はありません。

注意:
USBTMC(DL9000 以外)、VXI-11、VISAUSB では、この関数を実行しても何もしません。

使用例:[C#]
```
int ret = cTmctl.SetTerm( id, 0, 1 );/* CR+LF, TRUE */
```

使用例:[VC++]
```
int ret = TmcSetTerm( id, 0, 1 );    /* CR+LF, TRUE */
```

使用例:[VBA]
```
Dim ret As Long
ret = TmSetTerm( id, 0, 1 )          ' CR+LF, TRUE
```

6.3.10 Send

概要: デバイスへメッセージを送信します。
書式: [C#] int Send(int id, string msg)
int Send(int id, StringBuilder msg)
[VC++] int TmcSend(int id, char* msg)
[VBA] TmSend(ByVal id As Long, ByVal msg As String) As Long
引数:
[IN] id デバイス ID 値
[IN] msg メッセージ文字列
戻り値:
0 成功
1 エラー
詳細:
ID 値で指定されたデバイスへ ASCII 文字列を送信します。
Binary データを送る時は、"SendByLength"を使用してください。
また、１つの送信メッセージを分割して送信する場合は、"SendSetup","SendOnly"を使用してください。

注意:
なし

使用例:[C#]
```
int ret = cTmctl.Send( id, “*IDN?” ); /* メッセージ送信 */
```

使用例:[VC++]
```
int ret = TmcSend( id, “*IDN?” );     /* メッセージ送信 */
```

使用例:[VBA]
```
Dim ret As Long
ret = TmSend( id, “*IDN?” )           ‘ メッセージ送信
```

6.3.11 SendByLength

概要: 指定されたバイト数のメッセージをデバイスへ送信します。
書式: [C#] int SendByLength(int id, string msg, int len)
int SendByLength(int id, StringBuilder msg, int len)
[VC++] int TmcSendByLength(int id, char* msg, int len)
[VBA] TmSendByLength(ByVal id As Long, ByVal msg As String, ByVal blen As Long) As Long
TmSendByLengthB(ByVal id As Long, ByRef buf() As Byte, ByVal blen As Long) As Long
引数:

[IN]	id	デバイス ID 値
[IN]	msg	メッセージ文字列
[IN]	len	送信するバイト数

戻り値:

0	成功
1	エラー

詳細:
ID 値で指定されたデバイスへメッセージを送信します。
メッセージに、Binary データを含む場合でも、送信できます。
また、１つの送信メッセージを分割して送信する場合は、"SendSetup","SendOnly"を使用してください。

注意:
なし

使用例:[C#]

```
int ret = cTmctl.SendByLength( id, “*IDN?”, 5 );        /* メッセージ送信 */
```

使用例:[VC++]

```
int ret = TmcSendByLength( id, “*IDN?”, 5 );   /* メッセージ送信 */
```

使用例:[VBA]

```
Dim ret As Long
ret = TmSendByLength( id, “*IDN?”, 5 )          ‘ メッセージ送信
```

6.3.12 SendSetup

概要: デバイスへメッセージを送信する準備をします。
書式: [C#]　int SendSetup(int id)
　　[VC++] int TmcSendSetup(int id)
　　[VBA]　TmSendSetup(ByVal id As Long) As Long
引数:
　[IN]　id　　デバイス ID 値
戻り値:
　0　　成功
　1　　エラー
詳細:
　ID 値で指定されたデバイスへメッセージを送信する準備をします。
　１メッセージを数回に分けて送信する時に送信前に１回実行します。
　実際のメッセージの送信は、"SendOnly"を使用します。ID 値で指定されたデバイスへメッセージを送信します。
　メッセージに、Binary データを含む場合でも、送信できます。

注意:
　なし

使用例:[C#]
```
int ret = cTmctl.SendSetup( id );
```

使用例:[VC++]
```
int ret = TmcSendSetup( id );
```

使用例:[VBA]
```
Dim ret As Long
ret = TmSendSetup( id )
```

6.3.13 SendOnly

概要: 指定されたバイト数のメッセージをデバイスへ送信します。

書式: [C#] int SendOnly(int id, string msg, int len, int end)
int SendOnly(int id, StringBuilder msg, int len, int end)
int SendOnly(int id, ref sbyte data, int len, int end)
int SendOnly(int id, ref byte data, int len, int end)
int SendOnly(int id, ref short data, int len, int end)
int SendOnly(int id, ref ushort data, int len, int end)
int SendOnly(int id, ref int data, int len, int end)
int SendOnly(int id, ref uint data, int len, int end)
int SendOnly(int id, ref long data, int len, int end)
int SendOnly(int id, ref ulong data, int len, int end)
int SendOnly(int id, ref float data, int len, int end)
int SendOnly(int id, ref double data, int len, int end)
[VC++] int TmcSendOnly(int id, char* msg, int len, int end)
[VBA] TmSendOnly(ByVal id As Long, ByVal msg As String, _
ByVal len As Long, ByVal end As Long) As Long
TmSendOnlyB(ByVal id As Long, ByRef buf() As Byte, _
ByVal blen As Long, ByVal ed As Long) As Long

引数:

[IN]	id	デバイス ID 値
[IN]	msg	メッセージ文字列
[IN]	data	送信データ（バイナリ）
[IN]	len	送信するバイト数
[IN]	end	送信終了フラグ

設定値	説明
0	送信中
1	送信終了

戻り値:

0	成功
1	エラー

詳細:

ID 値で指定されたデバイスへメッセージを送信します。
メッセージに、Binary データを含む場合でも、送信できます。
送信終了フラグを 1 に設定して送信したときのみ、ターミネータをメッセージの最後に送信します。
そのため、送信終了フラグが 0 のうちは、デバイス側は一連のメッセージと判断します。

注意:

なし

使用例:[C#]

```
int ret;
ret = cTmctl.SendSetup( id );
ret = cTmctl.SendOnly( id, "*ID", 3, 0 );
ret = cTmctl.SendOnly( id, "N?", 2, 1 );       /* メッセージ送信完了 */
```

使用例:[VC++]

```
int ret;
ret = TmcSendSetup ( id );
ret = TmcSendOnly( id, "*ID", 3, 0 );
ret = TmcSendOnly( id, "N?", 2, 1 );           /* メッセージ送信完了 */
```

使用例:[VBA]

```
Dim ret As Long
ret = TmSendSetup( id )
ret = TmSendOnly( id, "*ID", 3, 0 )
ret = TmSendOnly( id, "N?", 2, 1 )   ‘ メッセージ送信完了
```

6.3.14 Receive

概要: デバイスから、メッセージを受信します。
書式: [C#] int Receive(int id, [Out] StringBuilder buff, int blen, ref int rlen)
[VC++] int TmcReceive(int id, char* buff, int blen, int* rlen)
[VBA] TmReceive(ByVal id As Long, ByRef buf As String, _
ByVal blen As Long, ByRef rlen As Long) As Long
引数:

[IN]	id	デバイス ID 値
[IN]	buff	受信データ用バッファ
[IN]	blen	受信サイズ（バイト単位）
[OUT]	rlen	実受信バイト数

戻り値:

0	成功
1	エラー

詳細:
ID 値で指定されたデバイスからメッセージを受信します。ターミネータを検出した場合は、
そこまでのデータを、検出しなかった場合は、blen で指定されたバイト数までのデータを受信します。
当社ディジタルオシロスコープとの通信で、"WAVeform:SEND?"、"IMAGe:SEND?"等のメッセージの
データを受信する場合は、"ReceiveBlockHeader","ReceiveBlockData"を使用してください。

注意:
なし

使用例:[C#]

```
StringBuilder buff = new StringBuilder(1000000);
int recv_len = 0;
int ret = cTmctl.Receive( id, buff, buff.Capacity, ref recv_len );
```

使用例:[VC++]

```
char* buff[10000];
int recv_len;
int ret = TmcReceive( id, buff, sizeof(buff), &recv_len );
```

使用例:[VBA]

```
Dim ret As Long
Dim buf As String
Dim length As Long
buf = Space$(1000)
ret = TmReceive( id, buf, 1000, length )
```

6.3.15 ReceiveSetup

概要: デバイスから、メッセージを受信する準備をします。
書式: [C#]　int ReceiveSetup(int id)
　　[VC++] int TmcReceiveSetup(int id)
　　[VBA]　TmReceiveSetup(ByVal id As Long) As Long
引数:
　[IN]　id　　デバイス ID 値
戻り値:
　0　　成功
　1　　エラー
詳細:
　デバイスから、大量データを分割して受信する場合に、受信準備をするために実行します。
　実際のデータは、"ReceiveOnly"を使用して受信します。

注意:
　なし

使用例:[C#]
```
int ret = cTmctl.ReceiveSetup( id );
```

使用例:[VC++]
```
int ret = TmcReceiveSetup( id );
```

使用例:[VBA]
```
Dim ret As Long
ret = TmReceiveSetup( id )
```

6.3.16 ReceiveOnly

概要: デバイスから、(受信準備後に) メッセージを受信します。
書式: [C#] int ReceiveOnly(int id, StringBuilder buff, int blen, ref int rlen)
[VC++] int TmcReceiveOnly(int id, char* buff, int blen, int* rlen)
[VBA] TmReceiveOnly(ByVal id As Long, ByRef buf As String, _
ByVal blen As Long, ByRef rlen As Long) As Long
引数:

[IN]	id	デバイス ID 値
[IN]	buff	受信データ用バッファ
[IN]	blen	受信サイズ (バイト単位)
[OUT]	rlen	実受信バイト数

戻り値:

0	成功
1	エラー

詳細:
大量データを分割して受信する場合に使用します。
"ReceiveSetup"で受信準備後に、ID 値で指定されたデバイスからメッセージを受信します。
ターミネータを検出した場合は、そこまでのデータを、検出しなかった場合は、blen で指定されたバイト数までのデータを受信します。

注意:
なし

使用例:[C#]

```
int ret = 0;
string buff;
stringBuilder buff1;
int rlen = 0;

// テキストデータの送受信 CheckEnd の返り値が 1 になるまで受信しないと
//  全てのデータを受信したことにならない
buff = ":ACQuire?;:ACQuire?;:ACQuire?;*IDN?";
buff1 = new StringBuilder(1000000);

ret = cTmctl.Send(id, buff);
ret = cTmctl.ReceiveSetup(id);
ret = 1;
while(ret = 1)
{
    ret = cTmctl.ReceiveOnly(id, buff1, buff1.Capacity, ref rlen);
    ret = cTmctl.CheckEnd(id);
    buff1.Remove(0, buff1.Length());
}
```

使用例:[VB.NET]

```
Dim buff As String
Dim buff1 As StringBuilder
Dim msg As String
Dim rlen As Integer

' テキストデータの送受信 CheckEnd の返り値が 1 になるまで受信しないと
‘ 全てのデータを受信したことにならない
buff = ":ACQuire?;:ACQuire?;:ACQuire?;*IDN?"
buff1 = New StringBuilder(1000000)
rlen = 0

ret = cTmctl.Send(id, buff)
ret = cTmctl.ReceiveSetup(id)
ret = 1
While ret = 1
    ret = cTmctl.ReceiveOnly(id, buff1, buff1.Capacity, rlen)
    ret = cTmctl.CheckEnd(id)
    buff1.Remove(0, buff1.Length())
End While
```

使用例:[VC++]

```
int ret;
char buff[1000];
int  length;

ret = TmcReceiveSetup( id );
ret = TmcReceiveOnly( id, buff, 1000, &length );
ret = TmcReceiveOnly( id, buff, 1000, &length );
ret = TmcReceiveOnly( id, buff, 1000, &length );
```

使用例:[VBA]

```
Dim ret As Long
Dim buf As String
Dim length As Long

ret = TmReceiveSetup( id )
buf = Space$(1000)
ret = TmReceiveOnly( id, buf, 1000, length )
buf = Space$(1000)
ret = TmReceiveOnly( id, buf, 1000, length )
buf = Space$(1000)
ret = TmReceiveOnly( id, buf, 1000, length )
```

6.3.17 ReceiveBlockHeader

概要: デバイスから、ブロックデータのバイト数を受信します。
書式: [C#]　int ReceiveBlockHeader(int id, ref int length)
[VC++] int TmcReceiveBlockHeader(int id, int* length)
[VBA]　TmReceiveBlockHeader(ByVal id As Long, ByRef len As Long) As Long
引数:
[IN]　id　　デバイス ID 値
[OUT] length　Block Data のデータバイト数
戻り値:
0　　成功
1　　エラー
詳細:
ブロックデータ(#～で始まるメッセージ)のサイズを受信する場合に使用します。
length に、あとに続くデータバイト数が返ってきますので、そのバイト数＋１（ターミネータ）分を"ReceiveBlockData"を使用してデータを受信します。

注意:
DM7560 ではターミネータ分は含まれていません。

使用例:[C#]
```
int ret;
int length = 0;
ret = cTmctl.ReceiveBlockHeader( id, ref length );
```

使用例:[VC++]
```
int length;
int ret = TmcReceiveBlockHeader( id, &length );
```

使用例:[VBA]
```
Dim ret As Long
Dim buf As String
Dim length As Long
Dim data(999) As Integer
Dim rlen As Long
Dim ed As Long

ret = TmSend(id, ":Waveform:Send?")
Debug.Print ("TmSend:Ret=" & ret)

ret = TmReceiveBlockHeader(id, length)
Debug.Print ("TmReceiveBlockHeader:Ret=" & ret & " length=" & length)

ed = 0
While ed = 0
    ret = TmReceiveBlock(id, data(0), length, rlen, ed)
    Debug.Print ("TmReceiveBlockData:Ret=" & ret & " rlen=" & rlen & " ed=" & ed)
Wend
```

6.3.18 ReceiveBlockData

概要: デバイスから、(バイト数受信後に)ブロックデータを受信します。

書式: [C#] int ReceiveBlockData(int id, ref sbyte buff, int blen, ref int rlen, ref int end)
int ReceiveBlockData(int id, ref byte buff, int blen, ref int rlen, ref int end)
int ReceiveBlockData(int id, ref short buff, int blen, ref int rlen, ref int end)
int ReceiveBlockData(int id, ref ushort buff, int blen, ref int rlen, ref int end)
int ReceiveBlockData(int id, ref int buff, int blen, ref int rlen, ref int end)
int ReceiveBlockData(int id, ref uint buff, int blen, ref int rlen, ref int end)
int ReceiveBlockData(int id, ref long buff, int blen, ref int rlen, ref int end)
int ReceiveBlockData(int id, ref ulong buff, int blen, ref int rlen, ref int end)
int ReceiveBlockData(int id, ref float buff, int blen, ref int rlen, ref int end)
int ReceiveBlockData(int id, ref double buff, int blen, ref int rlen, ref int end)

[VC++] int TmcReceiveBlockData(int id, char* buff, int blen, int* rlen, int* end);

[VBA] TmReceiveBlock(ByVal id As Long, buf() As Integer, ByVal blen As Long, _
ByRef rlen As Long, ByRef end As Long) As Long
TmReceiveBlockB(ByVal id As Long, buf() As Byte, ByVal blen As Long, _
ByRef rlen As Long, ByRef end As Long) As Long

引数:

[IN] id	デバイス ID 値	
[OUT] buff	受信データ用バッファ	
[IN] blen	受信サイズ（バイト単位）	
[OUT] rlen	実受信バイト数	
[OUT] end	受信終了フラグ	
	設定値	**説明**
	0	受信中（残データあり）
	1	受信終了

戻り値:

0	成功
1	エラー

詳細:

ブロックデータ(#～で始まるメッセージ)のデータ部を受信する場合使用します。
"ReceiveBlockHeader"で受信準備後に、ID 値で指定されたデバイスからメッセージを受信します。
ターミネータを検出した場合は、そこまでのデータを、検出しなかった場合は、blen で指定されたバイト数までのデータを受信します。

注意:

DM7560 では受信終了フラグは使用できません（常に 0)。また、ReceiveBlockHeader の length を blen へ設定してください。
SOCKET 回線では指定したサイズを受信すると受信終了フラグが設定されます。

使用例:[C#]

```
int ret;
string buff;
int rlen = 0;
sbyte[] data;
int datasize = 0;
int totalsize = 0;
int end1 = 0;

//  バイナリデータ受信
//  あらかじめデバイスが、データを取っていないと 1000 点のデータは返ってこない
// buff = ":DATA:RAW? 1,1,1,1000";    // SL1000 用
buff = ":WAVedata:SEND:BINary?";      // AQ7270 用

ret = cTmctl.Send(m_ID, buff);
ret = cTmctl.ReceiveBlockHeader(m_ID, rlen);
data = new sbyte[rlen];
while (end1 == 0)  // End フラグがたつまで受信を続ける
{
    ret = cTmctl.ReceiveBlockData(m_ID, data[totalsize], rlen, ref datasize, ref end1);
    if (ret != 0) break;
    totalsize += datasize;
}
```

使用例:[VB.NET]

```
Dim buff As String
Dim rlen As Integer
Dim data(999) As Short
Dim datasize As Long
Dim totalsize As Long
Dim end1 As Integer

‘  バイナリデータ受信
‘  あらかじめデバイスが、データを取っていないと 1000 点のデータは返ってこない
’ buff = ":DATA:RAW? 1,1,1,1000"      ’ SL1000 用
buff = ":WAVedata:SEND:BINary?"       ’ AQ7270 用

rlen = 0
end1 = 0
ret = cTmctl.Send(m_ID, buff)
ret = cTmctl.ReceiveBlockHeader(m_ID, rlen)
While (end1 <> 1)  ’ End フラグがたつまで受信を続ける
    ret = cTmctl.ReceiveBlockData(m_ID, data(0), rlen, datasize, end1)
End While
```

使用例:[VC++]

```
int ret;
int  length;
int  len;
char buf[1000];
int  flag;

ret = TmcReceiveBlockHeader( id, &length );
if( length < 1 ) {
    return;
}
length += 1;
flag = 0;
while( flag == 0 ) {
    ret = TmcReceiveBlockData( id, buf, length, &len, &flag );
}
```

使用例:[VBA]

```
Dim ret As Long
Dim buf As String
Dim length As Long
Dim data() As Integer
Dim rlen As Long
Dim ed As Long

ret = TmSend(id, ":Waveform:Send?")
Debug.Print ("TmSend:Ret=" & ret)

ret = TmReceiveBlockHeader(id, length)
Debug.Print ("TmReceiveBlockHeader:Ret=" & ret & " length=" & length)

ed = 0
ReDim data(length + 1)
While ed = 0
    ret = TmReceiveBlock(id, data(0), length, rlen, ed)
    Debug.Print ("TmReceiveBlockData:Ret=" & ret & " rlen=" & rlen & " ed=" & ed)
Wend

Erase data
```

6.3.19 GetLastError

概要: 最後に発生したエラーのエラー番号を返します。
書式: [C#]　int GetLastError(int id)
[VC++] int TmcGetLastError(int id)
[VBA]　TmGetLastError(ByVal id As Long) As Long
引数:
[IN]　id　　デバイス ID 値
戻り値:
エラー番号
詳細:
そのデバイスの最後に発生したエラー番号を返します。
初期化関数を含め、関数の戻り値 0(= OK)以外の場合に、この関数を使用して、実際のエラー番号を取得します。

注意:
なし

使用例:[C#]
```
int err;
ret = cTmctl.Send( id, "START" )
if( ret != 0 ) {
    err = cTmctl.GetLastError( id );
}
```

使用例:[VC++]
```
int ret = TmcSend( id, "START" );
if( ret != 0 ) {
    int err = TmcGetLastError( id );
}
```

使用例:[VBA]
```
Dim ret As Long
Dim err As Long

ret = TmSend(id, "START")
If (ret <> 0) Then
    err = TmGetLastError(id)
End If
```

6.3.20 SetRen

概要: デバイスをリモート／ローカル状態にします。
書式: [C#]　int SetRen(int id, int flag)
[VC++] int TmcSetRen(int id, int flag)
[VBA]　TmSetRen(ByVal id As Long, ByVal flg As Long) As Long
引数:
[IN]　id　　デバイス ID 値
[IN]　flag　　リモート／ローカル指定

設定値	説
0	ローカル
1	リモート

戻り値:
0　　成功
1　　エラー
詳細:
回線の種類によって、動作が若干異なります。
GPIB の場合は、REN ラインを TRUE/FALSE にします。実際にリモートにする場合は、この関数を送信後、そのデバイスに対してなんらかのメッセージを送信します。（デバイス個別のリモート／ローカル操作は行ないません。）
RS232、USB、Ethernet の場合は、当社 488.2 準拠品で、通信メッセージに COMMunicate グループを
サポートしているものに限ります。この場合は、デバイス個別に操作できます。
USBTMC、VISAUSB では、コントロール転送により、リモート／ローカルの切り替えを行います。

注意:
GPIB 以外での使用は、当社製品に限定されます。

使用例:[C#]
```
int ret = cTmctl.SetRen( id, 1 );
```

使用例:[VC++]
```
int ret = TmcSetRen( id, 1 );
```

使用例:[VBA]
```
Dim ret As Long
ret = TmSetRen( id, 1 )
```

6.3.21 CheckEnd

概要: デバイスからのメッセージが終了したかどうかを返します。
書式: [C#] int CheckEnd(int id)
[VC++] int TmcCheckEnd(int id)
[VBA] TmCheckEnd(ByVal id As Long) As Long
引数:
[IN] id デバイス ID 値
戻り値:
0 メッセージ終了
1 エラー or メッセージあり
詳細:
GPIB、USB、Ethernet、USBTMC、VXI-11、VISAUSB 回線で使用できます。
一連の受信メッセージを分割して受信したとき、"ReceiveOnly"ですべて受信し終わったかどうかを返します。
(RS232、SOCKET では常に 0 を返します。)

注意:
なし

使用例:[C#]
```
int ret = cTmctl.CheckEnd( id )
if( ret == 0 ) {   /* 受信終了 */
}
else {             /* 受信継続 */
}
```
使用例:[VC++]
```
int ret = TmcCheckEnd( id );
if( ret == 0 ) {   /* 受信終了 */
}
else {             /* 受信継続 */
}
```
使用例:[VBA]
```
Dim ret As Long
ret = TmCheckEnd( id )
If( ret == 0 ) Then
    '受信終了
Else
    '受信継続
Endif
```

6.3.22 DeviceClear

概要: 選択されたデバイスのクリア(SDC)を実行します。
書式: [C#] int DeviceClear(int id)
[VC++] int TmcDeviceClear(int id)
[VBA] TmDeviceClear(ByVal id As Long) As Long
引数:
[IN] id デバイス ID 値
戻り値:
0 成功
1 エラー
詳細:
なし

注意:
GPIB、USBTMC(DL9000 以外)、VXI-11、VISAUSB、USBTMC(YTUSB ドライバ)のみ対応しています。他の回線では何も行ないません（常に 0 を返します)。

使用例:[C#]
```
int ret = cTmctl.DeviceClear( id );
```

使用例:[VC++]
```
int ret = TmcDeviceClear( id );
```

使用例:[VBA]
```
Dim ret As Long
ret = TmDeviceClear( id )
```

6.3.23 DeviceTrigger

概要: デバイスにトリガメッセージを送信します
書式: [C#] int DeviceTrigger(int id)
[VC++] int TmcDeviceTrigger(int id)
[VBA] TmDeviceTrigger(ByVal id As Long) As Long
引数:
[IN] id デバイス ID 値
戻り値:
0 成功
1 エラー
詳細:
なし

注意:
GPIB、USBTMC(DL9000 以外)、VXI-11、VISAUSB、USBTMC(YTUSB ドライバ)のみ対応しています。他の回線では何も行ないません（常に 0 を返します)。

使用例:[C#]
```
int ret = cTmctl.DeviceTrigger( id );
```

使用例:[VC++]
```
int ret = TmcDeviceTrigger( id );
```

使用例:[VBA]
```
Dim ret As Long
ret = TmDeviceTrigger( id )
```

6.3.24 WaitSRQ

概要: 指定されたデバイスの SRQ を受け付けます
書式: [C#] int WaitSRQ(int id, ref byte stsbyte, int tmo)
[VC++] int TmcWaitSRQ(int id, char* stsbyte, int tmo)
[VBA] -
引数:

[IN]	id	デバイス ID 値
[OUT]	stsbyte	SRQ の要因
[IN]	tmo	タイムアウト値 (100msec 単位)

戻り値:

0	成功
1	エラー

詳細:
なし
注意:
VBA では使えません。
GPIB,USBTMC(DL9000 以外)、VXI-11、VISAUSB、USBTMC(YTUSB ドライバ)のみ対応しています。他の回線では常に 0 を返します。
GPIB の場合、タイムアウトは以下の決められた値になります。
(100msec,300msec,1sec,3sec,10sec,30sec,100sec,300sec,1000sec)

使用例:[C#]

```
byte sts = 0;
// タイムアウト値 10 秒で SRQ 待ち
ret = cTmctl.WaitSRQ(id, ref sts, 100);
```

使用例:[VB.NET]

```
Dim sts As Byte
// タイムアウト値 10 秒で SRQ 待ち
ret = cTmctl.WaitSRQ(id, sts, 100);
```

使用例:[VC++]

```
int      ret
char     sts ;

// タイムアウト値 10 秒で SRQ 待ち
ret = TmcWaitSRQ(id, &sts, 100) ;
```

6.3.25 AbortWaitSRQ

概要: 指定されたデバイスの SRQ 待ち関数の待ち状態を解除します
書式: [C#]　int AbortWaitSRQ(int id)
　　[VC++] int TmcAbortWaitSRQ(int id)
　　[VBA]　-
引数:
　[IN]　id　　デバイス ID 値
戻り値:
　0　　成功
　1　　エラー
詳細:
　なし
注意:
　VBA では使えません。
　USBTMC(DL9000 以外)、VXI-11、VISAUSB、USBTMC(YTUSB ドライバ)のみ対応しています。他の回線では常に 0 を返します。

使用例:[C#]

```
int ret = cTmctl.AbortWaitSRQ( id );
```

使用例:[VC++]

```
int ret = TmcAbortWaitSRQ( id );
```

6.3.26 SetCallback

概要: SRQ 発生時コールバックルーチンを登録します
書式: [C#] int SetCallback(int id, Hndlr func, uint p1, uint p2)
[VC++] int TmcSetCallback(int id, Hndlr func, ULONG p1,ULONG p2)
[VBA] -
引数:
[IN] id デバイス ID 値
[IN] func SRQ 発生時のコールバック関数
public delegate void Hndlr(int id, byte buff, uint p1, uint p2)
SRQ 発生時呼び出されるコールバック関数へのポインタを設定します。

[IN] p1 コールバック関数の第一引数
[IN] p2 コールバック関数の第二引数
戻り値:
0 成功
1 エラー
詳細:
コールバック関数はライブラリ内部で生成されたコールバック用スレッドから呼び出されます。

注意:
VBA では使えません。
USBTMC(DL9000 以外)、VXI-11、VISAUSB 、USBTMC(YTUSB ドライバ)のみ対応しています。他の回線では常に 0 を返します。

使用例:[C#]

```
// Callback 関数　SRQ 受信を Callback 関数で取得する例
public TMCTL.Hndlr method;
public uint p1 = 1;
public uint p2 = 2;

public void Func1(int id, byte buff, uint p1, uint p2)
{
    // SRQ 取得用 Callback 関数
    Console.WriteLine("id={0} buff={1} p1={2} p2={3}", id, buff, p1, p2);
}
private void Button1_Click(object sender, EventArgs e)
{
    method = new TMCTL.Hndlr(Func1);

    // Callback 関数を設定する
    ret = cTmctl.SetCallback(m_ID, method, p1, p2);
    Console.WriteLine("SetCallback:ret={0}", ret);
}
```

使用例:[VB.NET]

```
‘ Callback 関数　SRQ 受信を Callback 関数で取得する例
Public Method As TMCTL.Hndlr
Method = New TMCTL.Hndlr(AddressOf func1)

Public Shared Sub func1(ByVal id As Integer, ByVal buff As Byte, _
                        ByVal p1 As UInteger, ByVal p2 As UInteger)
    ‘ SRQ 取得用 Callback 関数
    Console.WriteLine("id={0} buff={1} p1={2} p2={3}", id, buff, p1, p2)
End Sub

Public p1 As UInteger = 1
Public p2 As UInteger = 2

Private Sub Button15_Click(ByVal sender As System.Object, _
                           ByVal e As System.EventArgs) Handles Button15.Click
    ‘ Callback 関数を設定する
    ret = tmctl.SetCallback(m_ID, Method, p1, p2)
    Console.WriteLine("SetCallback:ret={0}", ret)
End Sub
```

使用例:[VC++]

```
void func1(int id,UCHAR stb,ULONG p1,ULONG p2)
{
    printf(“SRQ occurred id=%d stb=0x%x p1=%d p2=%d¥n”, id,stb,p1,p2) ;
}
void setCallBack()
{
    int ret = TmcSetCallback(id, func1, 1, 2) ;
}
```

6.3.27 ResetCallback

概要: SRQ 発生時コールバックルーチンを削除します
書式: [C#]　int ResetCallback(int id)
　　[VC++] int TmcResetCallback(int id)
　　[VBA]　-
引数:
　[IN]　id　　デバイス ID 値
戻り値:
　0　　成功
　1　　エラー
詳細:
　なし
注意:
　VBA では使えません。
　USBTMC(DL9000 以外)、VXI-11、VISAUSB 、USBTMC(YTUSB ドライバ)のみ対応しています。他の回線では常に 0 を返します。

使用例:[C#]
```
int ret = cTmctl.ResetCallback( id );
```

使用例:[VC++]
```
int ret = TmcResetCallback( id );
```

7　サンプルプログラム

7.1 C#環境

```
using System.Text;
using TmctlAPINet;

private int ExecuteCommunicate()
{
    TMCTL cTmctl = new TMCTL();
    int ret = 0;
    int id = 0;
    int rlen = 0;
    int endflag = 0;

    DEVICELIST[] list = new DEVICELIST[10];
    int devlist_num = 0;

    StringBuilder encode = new StringBuilder(100);
    StringBuilder buff = new StringBuilder(256);

    sbyte[] recvdata;
    int totalsize = 0;
    int datasize = 0;

    // ex1: GPIB address = 1
    ret = cTmctl.Initialize(TMCTL.TM_CTL_GPIB, "1", ref id);
    // ex2: RS232 COM1,57600,8-NO-1,CTS-RTS
    ret = cTmctl.Initialize(TMCTL.TM_CTL_RS232, "1,6,0,1", ref id);
    // ex3: USB ID = 1
    ret = cTmctl.Initialize(TMCTL.TM_CTL_USB, "1", ref id);
    // ex4: Ethernet IP = 192.168.0.100, User name = yokogawa, Password = abcdefgh
    ret = cTmctl.Initialize(TMCTL.TM_CTL_ETHER, "192.168.0.100,yokogawa,abcdefgh", ref id);
    // ex5: USBTMC(DL9000) Serial Number = 27E000001
    ret = cTmctl.Initialize(TMCTL.TM_CTL_USBTMC, "27E000001", ref id);
    // ex6: USBTMC(GS200,GS820) Serial Number = 27E000001
    ret = cTmctl.Initialize(TMCTL.TM_CTL_USBTMC2, "27E000001", ref id);
    // ex7: USBTMC(GS610) Serial Number = 27E000001
    ret = cTmctl.Initialize(TMCTL.TM_CTL_USBTMC2, "27E000001C", ref id);
    // ex8: USBTMC(DL9000,GS シリーズ以外) Serial Number = 27E000001
    ret = cTmctl.EncodeSerialNumber(encode, encode.Capacity, "27E000001");
    ret |= cTmctl.Initialize(TMCTL.TM_CTL_USBTMC2, encode.ToString(), ref id);
    // ex9: VXI-11 IP = 192.168.0.100
    ret = cTmctl.Initialize(TMCTL.TM_CTL_VXI11, " 192.168.0.100", ref id);
    // ex10: VISAUSB (GS200,GS820,FG400) Serial Number = 27E000001
    ret = cTmctl.Initialize(TMCTL.TM_CTL_VISAUSB, "27E000001", ref id);
    // ex11: VISAUSB (GS610) Serial Number = 27E000001
    ret = cTmctl.Initialize(TMCTL.TM_CTL_VISAUSB, "27E000001C", ref id);
    // ex12: USBTMC(YTUSB ドライバ) Serial Number = 27E000001
    ret = cTmctl.EncodeSerialNumber(encode, encode.Capacity, "27E000001");
```

```
        ret |= cTmctl.Initialize(TMCTL.TM_CTL_USBTMC3, encode.ToString(), ref id);

        ret = cTmctl.SetTerm(id, 2, 1);
        if (ret != 0) {
            return cTmctl.GetLastError(id);
        }
        ret = cTmctl.SetTimeout(id, 300);
        if (ret != 0) {
            return cTmctl.GetLastError(id);
        }
        ret = cTmctl.SetRen(id, 1);
        if (ret != 0) {
            return cTmctl.GetLastError(id);
        }
        // Send *RST
        ret = cTmctl.Send(id, "*RST");
        if (ret != 0) {
            return cTmctl.GetLastError(id);
        }
        // Send *IDN? and receive query
        ret = cTmctl.Send(id, "*IDN?");
        if (ret != 0) {
            return cTmctl.GetLastError(id);
        }
        ret = cTmctl.Receive(id, buff, buff.Capacity, ref rlen);
        if (ret != 0) {
            return cTmctl.GetLastError(id);
       }
       ret = cTmctl.Send(id, ":WAVEFORM:FORMAT ASCII;:WAVEFORM:SEND?");
       // Receive block data
       if (ret != 0) {
          return cTmctl.GetLastError(id);
       }
       ret = cTmctl.ReceiveBlockHeader(id, ref rlen);
       rlen += 1; // term size
       recvdata = new sbyte[rlen];
       do{
          ret = cTmctl.ReceiveBlockData(id, ref recvdata[totalsize], rlen - totalsize, ref datasize, ref
endflag);
          if (ret != 0) break;
          totalsize += datasize;
       }while(endflag == 0);

      ret = cTmctl.Finish(id);
      if (ret != 0) {
          return cTmctl.GetLastError(id);
      }

      return 0;
}
```

7.2 VB.NET 環境

```
Imports System.Text
Imports TmctlAPINet

private Function ExecuteCommunicate() As Integer
{
        Dim cTmctl As TMCTL
        Dim ret As Long
        Dim id As Long
        Dim endflag As Long

        Dim encode As StringBuilder
        Dim buff As StringBuilder
        Dim length As Long

        cTmctl = new TMCTL()
        encode = new StringBuilder(100)
        buff = new StringBuilder(1024)

        ' ex1: GPIB address = 1
        ret = cTmctl.Initialize( TMCTL.TM_CTL_GPIB, "1", id)
        ' ex2: RS232 COM1,57600,8-NO-1,CTS-RTS
        ret = cTmctl.Initialize( TMCTL.TM_CTL_RS232, "1,6,0,1", id)
        ' ex3: USB ID = 1
        ret = cTmctl.Initialize( TMCTL.TM_CTL_USB, "1", id)
        ' ex4: Ethernet IP = 11.22.33.44, User name = yokogawa, Password = abcdefgh
        ret = cTmctl.Initialize( TMCTL.TM_CTL_ETHER, "11.22.33.44,yokogawa,abcdefgh", id)
        ' ex5: USBTMC(DL9000) Serial Number = 27E000001
        ret = cTmctl.Initialize( TMCTL.TM_CTL_USBTMC, "27E000001", id)
        ' ex6: USBTMC(GS200,GS820) Serial Number = 27E000001
        ret = cTmctl.Initialize( TMCTL.TM_CTL_USBTMC2, "27E000001", id)
        ' ex7: USBTMC(GS610) Serial Number = 27E000001
        ret = cTmctl.Initialize( TMCTL.TM_CTL_USBTMC2, "27E000001C", id)
        ' ex8: USBTMC(DL9000,GS シリーズ以外) Serial Number = 27E000001
        ret = cTmctl.EncodeSerialNumber( encode, encode.Capacity, "27E000001")
        ret = cTmctl.Initialize( TMCTL.TM_CTL_USBTMC2, encode.ToString(), id)
        ' ex9: VXI-11 IP = 11.22.33.44
        ret = cTmctl.Initialize( TMCTL.TM_CTL_VXI11, "11.22.33.44", id)
        ' ex10: VISAUSB (GS200,GS820,FG400) Serial Number = 27E000001
        ret = cTmctl.Initialize( TMCTL.VISAUSB, "27E000001", id)
        ' ex11: VISAUSB (GS610) Serial Number = 27E000001
        ret = cTmctl.Initialize( TMCTL.VISAUSB, "27E000001C", id)
        ' ex12: VISAUSB (GS シリーズ,FG400 以外) Serial Number = 27E000001
        ret = cTmctl.EncodeSerialNumber( encode, encode.Capacity, "27E000001")
        ret = cTmctl.Initialize( TMCTL.VISAUSB, encode.ToString(), id)
        ' ex13: USBTMC(YTUSB ドライバ) Serial Number = 27E000001
        ret = cTmctl.EncodeSerialNumber( encode, encode.Capacity, "27E000001")
        ret = cTmctl.Initialize( TMCTL.TM_CTL_USBTMC3, encode.ToString(), id)
        If( ret <> 0 ) Then
```

```
                ExecuteCommunicate = cTmctl.GetLastError( id )
                Return ExecuteCommunicate
        End If
        ret = cTmctl.SetTerm( id, 2, 1 )
        If( ret <> 0 ) Then
                ExecuteCommunicate = cTmctl.GetLastError( id )
                Return ExecuteCommunicate
        End If
        ret = cTmctl.SetTimeout( id, 300 )
        If( ret <> 0 ) Then
                ExecuteCommunicate = cTmctl.GetLastError( id )
                Return ExecuteCommunicate
        End If
        ret = cTmctl.SetRen( id, 1 )
        If( ret <> 0 ) Then
                ExecuteCommunicate = cTmctl.GetLastError( id )
                Return ExecuteCommunicate
        End If
        ' *RST 送信
        ret = cTmctl.Send( id, "*RST" )
        If( ret <> 0 ) Then
                ExecuteCommunicate = cTmctl.GetLastError( id )
                Return ExecuteCommunicate
        End If
        ' *IDN?送信＆クエリ受信
        ret = cTmctl.Send( id, "*IDN?" )
        If( ret <> 0 ) Then
                ExecuteCommunicate = cTmctl.GetLastError( id )
                Return ExecuteCommunicate
        End If
        ret = cTmctl.Receive( id, buff, buff.Capacity, length )
        If( ret <> 0 ) Then
                ExecuteCommunicate = cTmctl.GetLastError( id )
                Return ExecuteCommunicate
        End If
        ' ブロックデータ受信
        ret = cTmctl.Send( id, ":WAVEFORM:FORMAT ASCII;:WAVEFORM:SEND?" )
        If( ret <> 0 ) Then
                return cTmctl.GetLastError( id )
        End if
        ret = cTmctl.ReceiveBlockHeader(m_ID, rlen)
        endflag = 0
        While (endflag <> 1)        ' End フラグがたつまで受信を続ける
                ret = cTmctl.ReceiveBlockData(m_ID, buf, buff.Capacity, rlen, endflag)
        End while
        ret = cTmctl.Finish( id )
        If( ret <> 0 ) Then
                ExecuteCommunicate = cTmctl.GetLastError( id )
                Return ExecuteCommunicate
        End If
        Return 0
End Function
```

7.3 Visual C++環境

```
#include "tmctl.h"

int ExecuteCommunicate( void )
{
        char adr[100];
        int  ret;
        int  id;
        char buf[1000];
        int  length;
        int  endflag = 0;

        // 例１：GPIB アドレス = 1
        ret = TmcInitialize( TM_CTL_GPIB, "1", &id );
        // 例２：RS232 COM1,57600,8-NO-1,CTS-RTS
        ret = TmcInitialize( TM_CTL_RS232, "1,6,0,1", &id );
        // 例３：USB ID = 1
        ret = TmcInitialize( TM_CTL_USB, "1", &id );
        // 例４：Ethernet IP = 11.22.33.44, User name = yokogawa, Password = abcdefgh
        ret = TmcInitialize( TM_CTL_ETHER, "11.22.33.44,yokogawa,abcdefgh", &id );
        // 例５：USBTMC(DL9000) Serial Number = 27E000001
        ret = TmcInitialize( TM_CTL_USBTMC, "27E000001", &id );
        // 例６：USBTMC(GS200,GS820) Serial Number = 27E000001
        ret = TmcInitialize( TM_CTL_USBTMC2, "27E000001", &id );
        // 例７：USBTMC(GS610) Serial Number = 27E000001
        ret = TmcInitialize( TM_CTL_USBTMC2, "27E000001C", &id );
        // 例８：USBTMC(DL9000,GS シリーズ以外) Serial Number = 27E000001
        char encode[256] ;
        ret = TmcEncodeSerialNumber(encode,256,"27E000001") ;
        ret = TmcInitialize( TM_CTL_USBTMC2, encode, &id) ;
        // 例９：VXI-11 IP = 11.22.33.44
        ret = TmcInitialize( TM_CTL_VXI11, "11.22.33.44", &id );
        // 例１０：VISAUSB (GS200,GS820,FG400) Serial Number = 27E000001
        ret = TmcInitialize( TM_CTL_ VISAUSB , "27E000001", &id );
        // 例１１：VISAUSB (GS610) Serial Number = 27E000001
        ret = TmcInitialize( TM_CTL_ VISAUSB, "27E000001C", &id );
        // 例１２：VISAUSB(GS シリーズ,FG400 以外) Serial Number = 27E000001
        char encode[256] ;
        ret = TmcEncodeSerialNumber(encode,256,"27E000001") ;
        ret = TmcInitialize( TM_CTL_VISAUSB, encode, &id) ;
        // 例１３：USBTMC(YTUSB ドライバ) Serial Number = 27E000001
        char encode[256] ;
        ret = TmcEncodeSerialNumber(encode,256,"27E000001") ;
        ret = TmcInitialize( TM_CTL_USBTMC3, encode, &id) ;
        if( ret != 0 ) {
                return TmcGetLastError( id );
        }
        ret = TmcSetTerm( id, 2, 1 );
        if( ret != 0 ) {
```

```
                return TmcGetLastError( id );
        }
        ret = TmcSetTimeout( id, 300 );
        if( ret != 0 ) {
                return TmcGetLastError( id );
        }
        ret = TmcSetRen( id, 1 );
        if( ret != 0 ) {
                return TmcGetLastError( id );
        }
        // *RST 送信
        ret = TmcSend( id, "*RST" );
        if( ret != 0 ) {
                return TmcGetLastError( id );
        }
        // *IDN?送信＆クエリ受信
        ret = TmcSend( id, "*IDN?" );
        if( ret != 0 ) {
                return TmcGetLastError( id );
        }
        ret = TmcReceive( id, buf, 1000, &length );
        if( ret != 0 ) {
                return TmcGetLastError( id );
        }
        // ブロックデータ受信
        ret = TmcSend( id, ":WAVEFORM:FORMAT ASCII;:WAVEFORM:SEND?" );
        if( ret != 0 ) {
                return TmcGetLastError( id );
        }
        ret = TmcReceiveBlockHeader(m_ID, rlen);
        while (endflag == 0) {      // End フラグがたつまで受信を続ける
                ret = TmcReceiveBlockData(m_ID, buf, 1000, rlen, endflag);
        }
        ret = TmcFinish( id );
        if( ret != 0 ) {
                return TmcGetLastError( id );
        }
}
```

7.4 Visual Basic for Applications 環境

```
Function ExecuteCommunicate() As Integer
        Dim adr As String
        Dim ret As Long
        Dim id As Long
        Dim buf As String
        Dim length As Long
        Dim endflag As Long
        ChDrive ActiveWorkbook.Path
        ChDir ActiveWorkbook.Path
        '例１：GPIB アドレス = 1
        adr = "1"
        ret = TmInitialize( 1, adr, id )
        '例２：RS232 COM1,57600,8-NO-1,CTS-RTS
        adr = "1,6,0,2"
        ret = TmInitialize( 2, adr, id )
        '例３：USB ID = 1
        adr = "1"
        ret = TmInitialize( 3, adr, id )
        '例４：Ethernet IP = 11.22.33.44, User name = yokogawa, Password = abcdefgh
        adr = "11.22.33.44,yokogawa,abcdefgh"
        ret = TmInitialize( 4, adr, id )
        '例５：USBTMC(DL9000) Serial Number = 27E000001
        adr = "27E000001"
        ret = TmInitialize( 5, adr, id )
        '例６：USBTMC(GS200,GS820) Serial Number = 27E000001
        adr = "27E000001"
        ret = TmInitialize( 7, adr, id )
        '例７：USBTMC(GS610) Serial Number = 27E000001
        adr = "27E000001C"
        ret = TmInitialize( 7, adr, id )
        '例８：USBTMC(DL9000,GS シリーズ以外) Serial Number = 27E000001
        Dim encode As String * 128
        ret = TmEncodeSerialNumber(encode,128,"27E000001")
        ret = TmInitialize( 7, encode, &id)
        '例９：VXI-11 IP = 11.22.33.44
        ret = TmcInitialize( 8, "11.22.33.44", &id )
        '例１０：VISA(GS200,GS820,FG400) Serial Number = 27E000001
        adr = "27E000001"
        ret = TmInitialize( 10, adr, id )
        '例１１：VISAUSB(GS610) Serial Number = 27E000001
        adr = "27E000001C"
        ret = TmInitialize(10, adr, id )
        '例１２：VISAUSB(GS シリーズ,FG400 以外) Serial Number = 27E000001
        Dim encode As String * 128
        ret = TmEncodeSerialNumber(encode,128,"27E000001")
        ret = TmInitialize( 10, encode, &id)
        '例１３：USBTMC(YTUSB ドライバ) Serial Number = 27E000001
        Dim encode As String * 128
```

```
ret = TmEncodeSerialNumber(encode,128,"27E000001")
ret = TmInitialize( 12, encode, &id)
If( ret <> 0 ) Then
        ExecuteCommunicate = TmGetLastError( id )
        Exit Function
Endif
ret = TmSetTerm( id, 2, 1 )
If( ret <> 0 ) Then
        ExecuteCommunicate = TmGetLastError( id )
        Exit Function
Endif
ret = TmSetTimeout( id, 300 )
If( ret <> 0 ) Then
        ExecuteCommunicate = TmGetLastError( id )
        Exit Function
Endif
ret = TmSetRen( id, 1 )
If( ret <> 0 ) Then
        ExecuteCommunicate = TmGetLastError( id )
        Exit Function
Endif
'*RST 送信
ret = TmSend( id, "*RST" )
If( ret <> 0 ) Then
        ExecuteCommunicate = TmGetLastError( id )
        Exit Function
Endif
'*IDN?送信＆クエリ受信
ret = TmSend( id, "*IDN?" )
If( ret <> 0 ) Then
        ExecuteCommunicate = TmGetLastError( id )
        Exit Function
Endif
buf = Space$(1000)
ret = TmReceive( id, buf, 1000, &length )
If( ret <> 0 ) Then
        ExecuteCommunicate = TmGetLastError( id )
        Exit Function
Endif
' ブロックデータ受信
ret = TmSend( id, ":WAVEFORM:FORMAT ASCII;:WAVEFORM:SEND?" )
If( ret <> 0 ) Then
        ExecuteCommunicate TmGetLastError( id )
        Exit Function
End if
ret = TmReceiveBlockHeader(m_ID, rlen)
While (endflag <> 1)        ' End フラグがたつまで受信を続ける
        ret = TmReceiveBlockData(m_ID, buf, 1000, rlen, endflag)
End while
ret = TmFinish( id )
If( ret <> 0 ) Then
        ExecuteCommunicate = TmGetLastError( id )
```

```
                    Exit Function
          Endif
          ExecuteCommunicate = 0
End Function
```

8 付録

8.1 機器対応表

Series \ Protocol	GPIB	RS232	USB	USBTMC (YKMUSB)	VISAUSB	Ethernet (Legacy)	VXI-11	UDP	SOCKET	USBTMC (YTUSB)	備考
DLM4000	○	–	–	○	○	–	○	–	–	–	
DLM3000	○	–	–	–	○	–	○	–	○	○	
DLM2000	○	–	–	○	○	–	○	–	–	–	
DL/DLM6000	○	–	–	○	–	○	–	–	–	–	
DL350	–	–	–	○	○	–	○	–	–	–	
DL850E/DL850EV	○	–	–	○	○	–	○	–	–	–	
DL850/DL850V	○	–	–	○	○	–	○	–	–	–	
DL750	○	○	○	–	–	○	–	–	–	–	
DL750P	○	○	○	–	–	○	–	–	–	–	
DL1600	○	○	○	–	–	○	–	–	–	–	
DL1700E	○	–	○	–	–	○	–	–	–	–	
DL1740	○	–	○ ※1	–	–	○ ※2	–	–	–	–	※1:Version 1.10 以降 ※2:Version 1.30 以降
DL1720	○	–	○	–	–	○ ※3	–	–	–	–	※3:Version 1.30 以降
DL9000	○	–	–	○	–	○	–	–	–	–	
SB5000	○	–	–	○	–	○	○	–	–	–	
DL7400	○	–	○	–	–	○	–	–	–	–	
DL7200	○	○	–	–	–	○ ※4	–	–	–	–	※4:Versio 3.02 以降
DL7100	○	○	–	–	–	○ ※5	–	–	–	–	※5:Versio 3.02 以降
FG400	○	–	–	–	○	–	–	–	–	–	
WT5000	○	–	–	–	○	–	–	○	–	○	
WT3000E	○	○	○	–	–	○	–	–	–	–	
WT3000	○	○	○ ※6	–	–	○ ※6	–	–	–	–	※6:Version 2.01 以降
WT1800E	○	–	–	○	○	–	○	–	–	–	
WT1800	○	–	–	○	○	–	○	–	–	–	
WT1600	○	○	–	–	–	○ ※7	–	–	–	–	※7:Version 2.01 以降
WT500	○	–	–	○	○	–	○	–	–	–	
WT300E	○	○	–	○	○	–	○	–	–	–	
WT300	○	○	–	○	○	–	○	–	–	–	
PX8000	○	–	–	○	○	–	○	–	–	–	
SL1000	–	–	–	○	○	–	○	–	–	–	
SL1400	○	○	○	–	–	○	–	–	–	–	
GS820	○	○	–	○	○	–	○	–	○	–	

Series \ Protocol	GPIB	RS232	USB	USBTMC (YKMUSB)	VISAUSB	Ethernet (Legacy)	VXI-11	UDP	SOCKET	USBTMC (YTUSB)	備考
GS610	○	○	–	○	○	–	–	–	○	–	
GS200	○	–	–	○	○	–	○	–	○	–	
2553A	○	–	–	○	○	–	○	–	–	–	
2558A	○	–	–	○	○	–	○	–	–	–	
2560A	○	–	–	○	○	–	○	–	–	–	
LS3300	○	–	–	○	○	–	○	–	–	–	
AQ7280	–	–	–	○	○	○	–	–	–	–	
AQ7270	–	–	○	–	–	○	–	–	–	–	
AQ7260	–	–	○	–	–	–	–	–	–	–	
AQ2211/AQ2212	○	–	–	○	–	○	–	–	–	–	
AQ1300	–	–	–	○	○	○	–	–	–	–	
AQ1210	–	–	–	–	○	–	–	–	–	○	
AQ1200	–	–	–	○	○	○	–	–	–	–	
AQ1100	–	–	–	○	○	○	–	–	–	–	
AQ1000	–	–	–	○	○	–	–	–	–	–	
732050	–	–	–	–	–	–	–	○	–	–	
DM7560	○	○ ※8	–	–	–	–	–	–	○	–	※8:USB は仮想 COM ポートドライバにより、RS232 として扱うことができます。
MT300	○	○ ※8	–	–	○	–	○	–	–	○	※8:USB は仮想 COM ポートドライバにより、RS232 として扱うことができます。