User's Manual

TMCTL Library
Application Programming Interface

YOKOGAWA
Yokogawa Test & Measurement Corporation

IM B9852UB-01EN
9th Edition

Foreword

This user’s manual contains useful information about the precautions, functions, and API specifications of the TMCTL library, which is used to connect YOKOGAWA products to PCs for communication.
To ensure correct use, please read this manual thoroughly during operation. Keep this manual in a safe place for quick reference.
For information about the handling precautions, functions, and operating procedures of YOKOGAWA products and the handling and operating procedures of Windows, see the relevant manuals.

Notes

- The contents of this manual are subject to change without prior notice as a result of continuing improvements to the instrument's performance and functions.
  The figures given in this manual may differ from those that actually appear on your screen.
- Every effort has been made in the preparation of this manual to ensure the accuracy of its contents. However, should you have any questions or find any errors, please contact your nearest YOKOGAWA dealer.
- Copying or reproducing all or any part of the contents of this manual without the permission of YOKOGAWA is strictly prohibited.

Trademarks

- Microsoft, Windows, Excel, Visual Basic Visual C++, and Visual C# are registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.
- In this manual, the TM and ® symbols do not accompany their respective registered trademark or trademark names.
- Other company and product names are trademarks or registered trademarks of their respective companies.

Revisions

June 2017	1st Edition
July 2017	2nd Edition
February 2018	3rd Edition
August 2018	4th Edition
October 2018	5th Edition
October 2019	6th Edition
September 2020	7th Edition
March 2021	8th Edition
February 2022	9th Edition

Contents

1 Software Overview

Overview

The TMCTL library provides an application programming interface (API) for remotely controlling YOKOGAWA products.

Functions

This software can be used to perform the following functions. For details on each function, see section 6.3, Detailed API Specifications".

- Connecting and disconnecting from device
- Searching for device
- Sending and receiving commands
- Setting remote or local mode
- Clearing devices and sending device triggers
- Getting the status byte

Software Structure

This software package contains the following items.

- TMCTL Library Application Programming Interface User's Manual (this manual)
- API files (see below)

File Name	Content
TmctlAPINet.dll	Managed Application API Library
TmctlAPINet64.dll	Managed Application API Library 64-bit Version
tmctl.dll	Communication API Library
tmctl64.dll	Communication API Library 64-bit Version
YKMUSB.dll	USB Communication Library
YKMUSB64.dll	USB Communication Library 64-bit Version
tmctl.h	Function Declaration Header File (for VC++)
tmctl.lib	TMCTL API Import Library (for VC++)
tmctl64.lib	TMCTL API Import Library 64-bit Version (for VC++)
tmctl.bas	TMCTL Function Definition File (for VBA)
tmval.bas	TMCTL Constant Definition File (for VBA)

2 PC System Requirements

PC

A PC capable of running Windows 8.1, Windows 10, or Windows 11 with 1 GHz or better CPU and 1 GB or more memory (2 GB or more recommended) is required.

Development Environment

Microsoft Visual C# 20012 to Microsoft Visual C# 2022
Microsoft Visual C++ 20012 to Microsoft Visual C++ 2022
Microsoft Visual Basic 20012 to Microsoft Visual Basic 2022
Microsoft Excel 2016 (Visual Basic for Applications 7.0)

.NET Framework

Microsoft .NET Framework 3.5 to Microsoft .NET Framework 4.8

Visual C ++ library runtime

Microsoft Visual C ++ Redistributable for Visual Studio 2017 (x64)

Communication Interface

GPIB	Environment in which National Instruments GPIB interface driver runs
RS232	Environment in which a serial port runs
USB	Environment in which YOKOGAWA USB driver (YKMUSB or YTUSB) runs Environment in which National Instruments NI-VISA version 4.6.2 to 21.5 or Keysight IO library version 15.5.13009.1 to 2018 Update 1.0 is installed and USB Test and Measurement Device (IVI) runs
Ethernet	Environment in which TCP/IP runs (including VXI-11)

3 Notes on Using the Software

Disclaimers

By downloading and installing this software, the customer agrees to all of the following disclaimers.

- Yokogawa bears no liability for any problems occurring as a result of downloading or installing this software.
- Yokogawa bears no responsibility for any damage caused directly or indirectly as a result of using this software.
- This software is provided free of charge, however no unlimited warranty against software defects exists, nor is any claim made that the product is free of all defects whatsoever. Also, Yokogawa is not always able to repair defects (“bugs”) in, or respond to questions or inquiries about this software.
- Yokogawa reserves all rights to this software, including but not limited to all property rights, ownership rights, and intellectual property rights.

Usage Precautions

- This software is a library designed exclusively for remotely controlling YOKOGAWA products. It cannot be used with other products.
- Check the version of this software and the firmware version of the applicable YOKOGAWA product prior to use.

4 Installation and Uninstallation

Installation Procedure

1. Download the TMCTL library from YOKOGAWA website. Download it to an appropriate location in your PC.
2. The file that you download is a compressed file. Use an appropriate unzip tool to extract the files. A “tmctlXXXX[*1]” folder will be created.
3. You need to install the runtime of the Visual C ++ library. Execute “VC_redist.x64.exe” in the redist folder in the unzipped folder to complete the installation.
4. Place the files to be used in the folder referenced by the application.
   For details, see section 5.6, DLL Linking Method and Placement”.

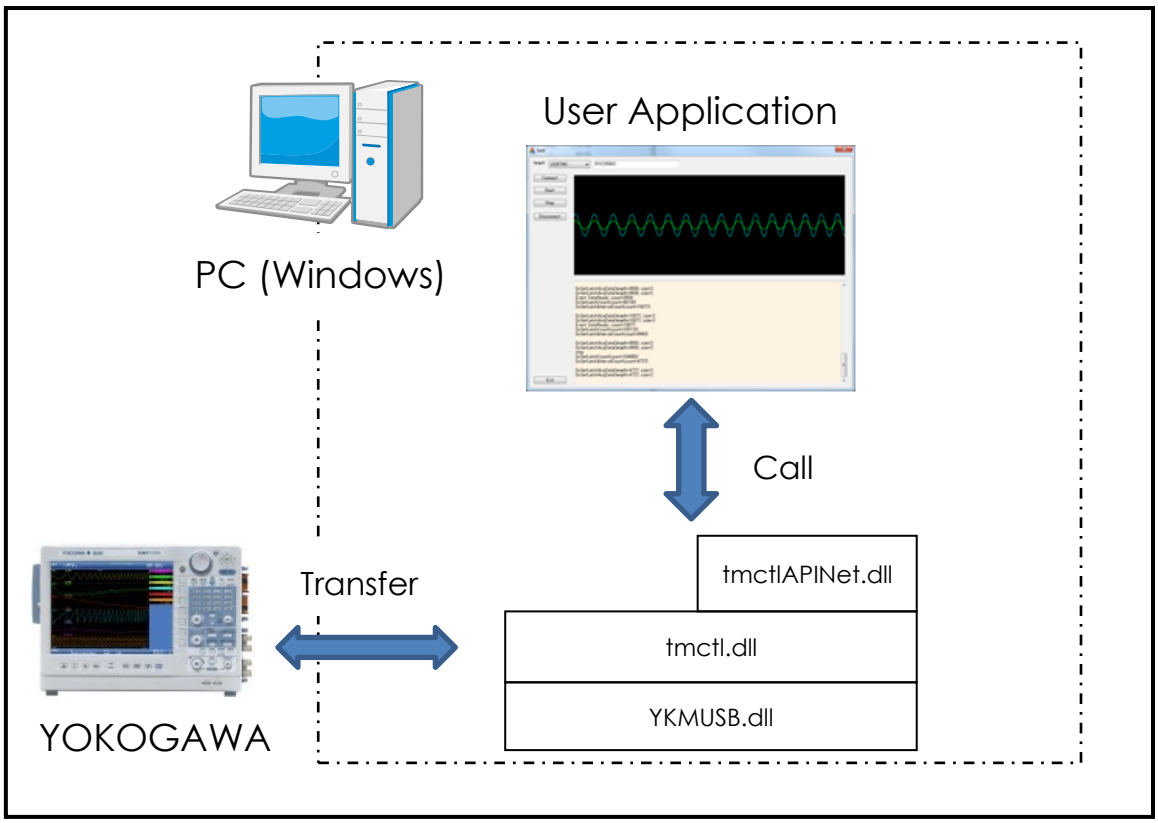
Uninstallation Procedure

1. Delete the entire “tmctlXXXX[*1]” folder.

[*1] XXXX is version number.

5 TMCTL API Overview

The API is provided as a dynamic link library (DLL). The API can be used by linking user applications with this DLL. As shown in the following figure, TMCTL API provides to the application functions for sending and receiving commands from devices.



5.1 Products with Which Communication Is Possible

5.1.1 GPIB

• YOKOGAWA products equipped with IEEE Std 488.2 compatible GPIB interface
• Communication is possible with other manufactures' products, but a portion of the functions cannot be used. (For details, see section 6.3, " Detailed API Specifications")

Applicable Series

DLM5000, 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:

- For the terminator for communicating with YOKOGAWA products, use LF and EOI in normal cases and EOI for binary data transmission.
- Use a PC's GPIB device driver that is compatible with your OS. For details on the PC's GPIB device driver, contact the manufacturer.

5.1.2 RS232

- YOKOGAWA product equipped with RS232 and that allows the following settings
- Communication is possible with other manufactures' products, but a portion of the functions cannot be used.
  (For details, see section 6.3, " Detailed API Specifications.")

Applicable Series

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

Settings

- Baud rate:
  - 1200 , 2400 , 4800 , 9600 , 19200 , 38400 , 57600 , 115200
- Data bit length, parity and stop bit combinations:
  - 8 bits, no parity, 1 stop bit
  - 7 bits, even, 1 stop bit
  - 7 bits, odd, 1 stop bit
  - 8 bits, odd, 1 stop bit
  - 7 bits, no parity, 1.5 stop bit
  - 8 bits, no parity, 2 stop bit
- Handshaking:
  - NO-NO (no handshaking)
  - XON-XON (software handshaking)
  - CTS-RTS (hardware handshaking)
- Terminator:
  - LF, CR+LF

Note:

- When communication with a YOKOGAWA product, normally use the following settings.
  - 8 bits, no parity, 1 stop bit
  - CTS-RTS (hardware handshaking)
  - Terminator LF
- The DM7560's USB can be handled as RS232 by using a COM port driver.

5.1.3 USB (Except USBTMC)

• YOKOGAWA product equipped with a USB interface but without USBTMC protocol support

Applicable Series

DL750, DL750P, DL1600, DL1700E, DL1740 (firmware version 1.10 and later), DL1720, DL7400, WT3000E, WT3000 (firmware version 2.01 and later), SL1400, AQ7270, AQ7260

Note:

- For the terminator, use LF and EOI, or EOI.
- While the interface is opened, do not turn off the PC or the product.

5.1.4 USB (USBTMC)

• YOKOGAWA product equipped with a USB interface and with USBTMC protocol support

Applicable Series

DLM5000, DLM4000, DLM3000, DLM2000, DL/DLM6000, DL950, 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:

- For the terminator, use LF and EOI, or EOI.

5.1.5 USB (VISAUSB)

• YOKOGAWA product equipped with a USB interface and with IVI USB driver support

Applicable Series

DLM5000, DLM4000, DLM3000, DLM2000, DL950, 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)

• YOKOGAWA product equipped with a an Ethernet interface and with Server protocol support

Applicable Series

DL/DLM6000, DL750, DL750P, DL1600, DL1700E, DL1740 (firmware version 1.30 and later), DL1720 (firmware version 1.30 and later), DL9000, SB5000, DL7400, DL7200 (firmware version 3.02 and later) , DL7100 (firmware version 3.02 and later), WT3000E, WT3000 (firmware version 2.01 and later), WT1600 (firmware version 2.01 and later), SL1400, AQ7280, AQ7270, AQ2211/AQ2212, AQ1300, AQ1200, AQ1100

Note:

• User authentication takes place when establishing a connection.
(For details, see section 6.3, “ Detailed API Specifications.”)

5.1.7 Ethernet (VXI-11)

• YOKOGAWA product equipped with an Ethernet interface and with VXI-11 protocol support

Applicable Series

DLM5000, DLM4000, DLM3000, DLM2000, DL950, 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)

• YOKOGAWA product equipped with an Ethernet interface and with UDP protocol support

Applicable Series

732050 ISDB-T Radio Wave Monitor

5.1.9 Ethernet (SOCKET)

• YOKOGAWA product equipped with an Ethernet interface and with socket command protocol support

Applicable Series

GS610, GS820, GS200, DM7560, DLM5000, DLM3000, DL950

5.1.10 Ethernet (HiSLIP)

• YOKOGAWA product equipped with an Ethernet interface and with HiSLIP support

Applicable Series

DL950

5.2 API Function Overview

This section provides an overview of the API functions.

5.2.1 Start/End Connection

API functions for connection are as follows:

API Name	Function	Page
Initialize	Starts connecting to the specified device	18
InitializeEx	Starts connecting to the specified device	26
Finish	Ends the device connection	28
SearchDevices	Searches for devices connected	29
SearchDevicesEx	Searches for devices connected	32
EncodeSerialNumber	Converts the serial number on the product's nameplate to an internal USB serial number	35
DecodeSerialNumber	Converts the internal USB serial number to serial number on the product's nameplate	36

5.2.2 Set Up Communication

API functions for setting up communication are as follows:

API Name	Function	Page
SetTimeout	Sets the communication timeout value	37
SetTerm	Sets the terminator for sending and receiving messages	38

5.2.3 Send and Receive

API functions for sending and receiving messages from device are as follows:

API Name	Function	Page
Send	Sends a message to the device	39
SendByLength SendByLengthB	Sends the specified number of bytes of a message to the device	40
SendSetup	Prepares to send a message to the device	41
SendOnly SendOnlyB	Sends the specified number of bytes of a message to the device.	42
Receive	Receives a message from the device	44
ReceiveSetup	Prepares to receive a message from the device	45
ReceiveOnly	Receives a message from the device (after it is ready to receive the message)	46
ReceiveBlockHeader	Receives the byte size of the block data from the device	48
ReceiveBlockData ReceiveBlockB	Receives the block data from the device (after receiving the byte size)	49
CheckEnd	Returns whether the message from the device has ended	54

5.2.4 Get Information

The API function for getting information is as follows:

API Name	Function	Page
GetLastError	Returns the error number of the error that occurred last.	52

5.2.5 Other Functions

API functions related to operation are as follows:

API Name	Function	Page
SetRen	Sets the device in remote or local mode	53
DeviceClear	Clears the selected device (SDC)	55
DeviceTrigger	Sends a trigger message to the device	56
WaitSRQ	Receives an SRQ from the specified device	57
AbortWaitSRQ	Releases the wait state of the SRQ wait function for the specified device	58
SetCallback	Registers the callback routine for SRQs	59
ResetCallback	Deletes the callback routine for SRQs	61

5.3 How to Use (C#, VB.NET)

5.3.1 Setup

Name of the used file:
TmctlAPINet.dll (TmctlAPINet64.dll in a 64-bit environment)

- Add a reference to "TmctlAPINet.dll"("TmctlAPINet64.dll" in a 64-bit environment) in the project you are developing.
- Place "tmctl.dll" and "YKMUSB.dll" in the folder referenced by the application.
("tmctl64.dll" and "YKMUSB64.dll" in a 64-bit environment)

5.3.2 Execution

With the initialization function, this library opens the interface to the control target device that is connected to the PC. The ID value that is returned as a parameter is used in other send and receive functions to control the device.

5.3.3 TMCTL Class

The TMCTL class provides functions implemented in the tmctl library (tmctl.dll/tmctl64.dll) to managed applications.

Namespace:	TmctlAPINet
Class name:	TMCTL

5.4 How to Use (Visual C++)

5.4.1 Setup

Name of the used file:
tmctl.h (function definition header file)
tmctl.lib (import library) ("tmctl64.lib" in a 64-bit environment)

• Add “tmctl.h” as an include file in the source file that you are using.
#include "tmctl.h"
• Add “tmctl.lib” ("tmctl64.lib” in a 64-bit environment) to the library file to be linked.
• Place "tmctl.dll” and "YKMUSB.dll” in the directory referenced by the application.
("tmctl64.dll" and "YKMUSB64.dll” in a 64-bit environment)

5.4.2 Execution

With the initialization function, this library opens the interface to to the control target device that is connected to the PC. The ID value that is returned as a parameter is used in other send and receive functions to control the device.

5.5 How to Use (Visual Basic for Applications)

5.5.1 Setup

Name of the used file:
tmctl.bas (function definition file)
tmval.bas (constant definition file)

• Add "tmctl.bas" and "tmval.bas” to your project's standard module.
• Also place "tmctl.dll” and "YKMUSB.dll” in the directory referenced by the application.
("tmctl64.dll" and "YKMUSB64.dll” in a 64-bit Excel environment)

5.5.2 Execution

With the initialization function, this library opens the interface to to the control target device that is connected to the PC. The ID value that is returned as a parameter is used in other send and receive functions to control the device.

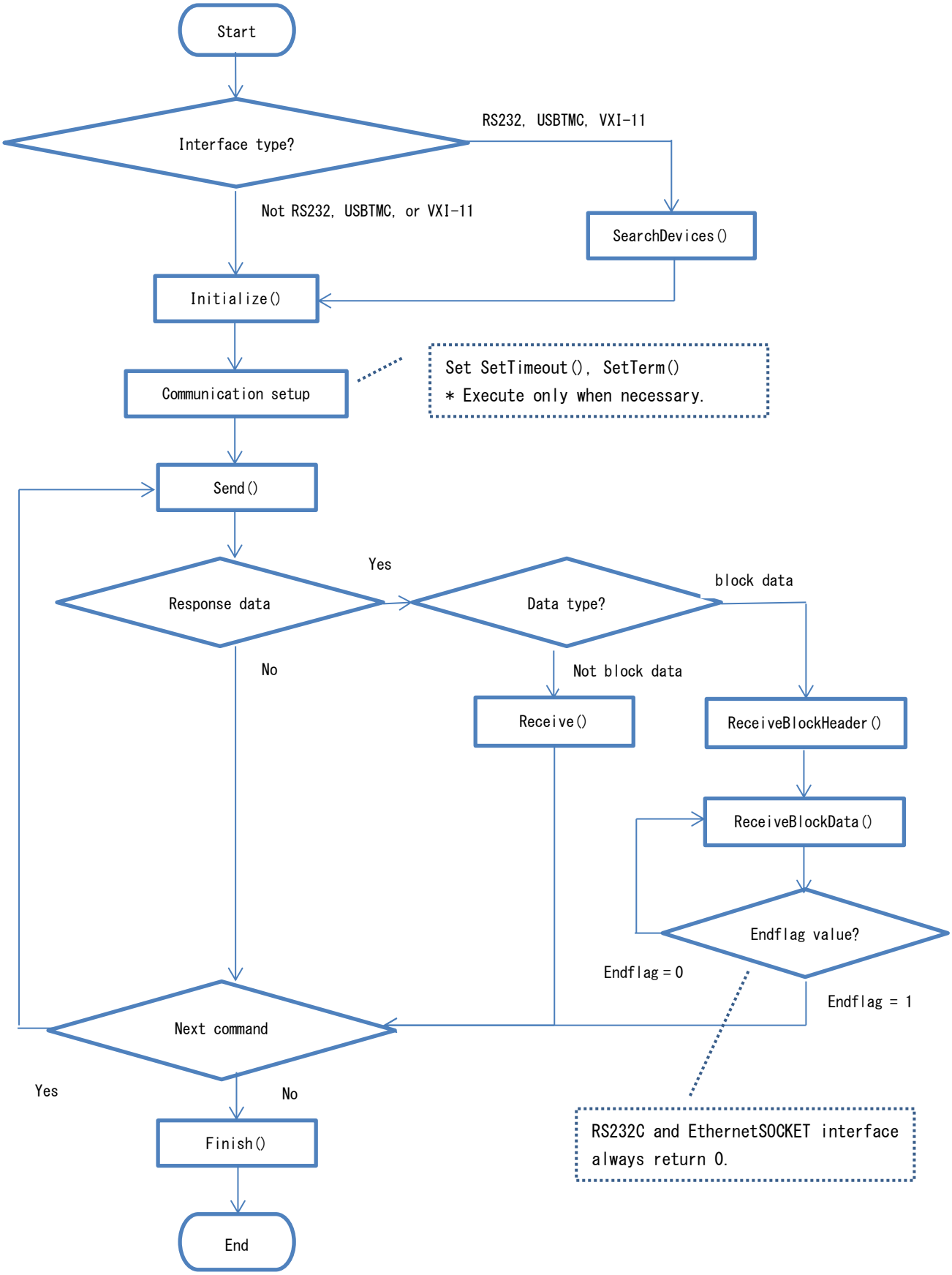
5.6 DLL Linking Method and Placement

Place the following DLLs in the same folder as the application (exe) that you create.

Project	VC++ / VBA		C# / VB.NET *	
Architecture	32 bit	64 bit	32 bit	64 bit
tmctlAPINet.dll	-	-	✓	-
tmctlAPINet64.dll	-	-	-	✓
tmctl.dll	✓	-	✓	-
tmctl64.dll	-	✓	-	✓
YKMUSB.dll	✓	-	✓	-
YKMUSB64.dll	-	✓	-	✓

* The “Any CPU” architecture of C# and VB.NET is currently not supported.
Specify either “x86” or “x64”.

5.7 Basic Operation Flowchart



6 API Functional Specifications

This chapter explains the API functional specifications.

6.1 Error Definitions

Error codes returned by the GetLastError function

Code	Definition name	Description
0	TMCTL_NO_ERROR	No error
1	TMCTL_TIMEOUT	Timeout
2	TMCTL_NO_DEVICE	Target device not found
4	TMCTL_FAIL_OPEN	Connection with the device failed.
8	TMCTL_NOT_OPEN	Not connected to the device
16	TMCTL_DEVICE_ALREADY_OPEN	Already connected to the device
32	TMCTL_NOT_CONTROL	The PC is not compatible.
64	TMCTL_ILLEGAL_PARAMETER	Illegal function parameter
256	TMCTL_SEND_ERROR	Send error
512	TMCTL_RECV_ERROR	Receive error
1024	TMCTL_NOT_BLOCK	Received data is not block data.
4096	TMCTL_SYSTEM_ERROR	System error
8192	TMCTL_ILLEGAL_ID	Illegal device ID
16384	TMCTL_NOT_SUPPORTED	Unsupported function
32768	TMCTL_INSUFFICIENT_BUFFER	Not enough buffer
65536	TMCTL_LIBRARY_ERROR	Library missing

6.2 Definition of Constants

Constant definitions in the TMCTL class

Definition name	Overview
TM_CTL_GPIB	Interface designation (GPIB)
TM_CTL_RS232	Interface designation (RS232C)
TM_CTL_USB	Interface designation (USB other than USBTMC)
TM_CTL_ETHER	Interface designation (Ethernet)
TM_CTL_USBTMC	Interface designation (DL9000 dedicated USBTMC)
TM_CTL_ETHERUDP	Interface designation (Ethernet UDP)
TM_CTL_USBTMC2	Interface designation (USBTMC on instruments other than the DL9000)
TM_CTL_VXI11	Interface designation (VXI-11)
TM_CTL_VISAUSB	Interface designation (VISAUSB)
TM_CTL_SOCKET	Interface designation (Ethernet SOCKET)
TM_CTL_USBTMC3	Interface designation (USBTMC with YTUSB driver)
TM_CTL_HISLIP	Interface designation (HiSLIP)
TM_RS_1200	RS232 interface baudrate (1200 bps)
TM_RS_2400	RS232 interface baudrate (2400 bps)
TM_RS_4800	RS232 interface baudrate (4800 bps)
TM_RS_9600	RS232 interface baudrate (9600 bps)
TM_RS_19200	RS232 interface baudrate (19200 bps)
TM_RS_38400	RS232 interface baudrate (38400 bps)
TM_RS_57600	RS232 interface baudrate (57600 bps)
TM_RS_115200	RS232 interface baudrate (115200 bps)
TM_RS_8N	RS232 interface bit specifications (8 bits, no parity, 1 stop bit)
TM_RS_7E	RS232 interface bit specifications (7 bits, even parity, 1 stop bit)
TM_RS_7O	RS232 interface bit specifications (7 bits, odd parity, 1 stop bit)
TM_RS_8O	RS232 interface bit specifications (8 bits, odd parity, 1 stop bit)
TM_RS_7N5	RS232 interface bit specifications (7 bits, no parity, 1.5 stop bits)
TM_RS_8N2	RS232 interface bit specifications (8 bits, no parity, 2 stop bits)
TM_RS_NO	RS232 interface handshaking number (NO-NO)
TM_RS_XON	RS232 interface handshaking number (XON-XON)
TM_RS_HARD	RS232 interface handshaking number (CTS-RTS)
ADRMAXLEN	String length of member adr used in the DEVICELIST structure (64)

6.3 Detailed API Specifications

This section provides the details of the API.

6.3.1 Initialize

Description: Initializes the interface and opens the interface to the specified device

Syntax: [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

Parameters:

[IN] wire Interface type

Class/VC++ definition	VBA definition	
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 (on instruments other than the DL9000)
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 (with YTUSB driver)
TM_CTL_HISLIP	CTL_HISLIP	(14) HiSLIP

[IN] adr Connection destination address

Target interface	Description of settings
GPIB	“<GPIB address>” or “<Interface ID>,<GPIB address>” *GPIB address “0” to “30” *Interface ID “0” to “99” *Interface ID is omissible.
RS232	“<com port number>,<baud rate>,<bit>,<handshaking number>” com port number : 1 to 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 ID value>”
Ethernet	“<Server name>,<username>,<password>” or “<Server name>,<port>,<username>,<password>”

	* If the user name is anonymous, you do not need a password. * The comma is necessary for separation. * The port is omissible.
Ethernet(UDP)	“<Server name>, <Port number>”
USBTMC(DL9000)	“<serial number>”
USBTMC(GS200,GS820,FG400)	“<serial number>”
USBTMC(GS610)	“<serial number>” + “C”
USBTMC (on instruments other than the DL90000, GS, or FG400)	“<serial number>” * Number encoded with EncodeSerialNumber
VXI-11	“<IP address>”
VISAUSB(GS200,GS820,FG400)	“<serial number>”
VISAUSB(GS610)	“<serial number>” + “C”
VISAUSB (on instruments other than GS or FG400)	“<serial number>” * Number encoded with EncodeSerialNumber
Ethernet(SOCKET)	“<Server name>, <Port number>” * The comma is necessary for separation.
USBTMC (with YTUSB driver)	“<serial number>” * Number encoded with EncodeSerialNumber
HiSLIP	“<IP address>:<Port number>” or “<IP address>” * If you specify “<IP address>”, the default port number(4880) is used.

[OUT] id Device specific ID used with other functions and the like

Return value:

0	Connection successful
1	Connection error

Detail:

None

Note:

To use multiple devices, execute this function for each device.

Up to 127 devices can be used simultaneously.

Example: [C#]

```
TMCTL cTmctl = new TMCTL();
int ret = 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 (on instruments other than the DL9000 or GS series)] Serial number = 27E000001
    StringBuilder encode = new StringBuilder(100);        // Create an instance.
    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 (on instruments other than GS series or FG400)] serial number = 27E000001
    StringBuilder encode = new StringBuilder(100);        // Create an instance.
    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(with YTUSB driver)] Serial number = 27E000001
    StringBuilder encode = new StringBuilder(100);        // Create an instance.
    ret = cTmctl.EncodeSerialNumber(encode,encode.Capacity,"27E000001");
    ret = cTmctl.Initialize(TMCTL.TM_CTL_USBTMC3, encode.ToString(), ref id );
[HiSLIP] IP = 11.22.33.44, Port = 4880
    ret = cTmctl.Initialize(TMCTL.TM_CTL_HISLIP, "11.22.33.44:4880", ref id );
```

Example: [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 (on instruments other than the DL9000 or GS series)] Serial number = 27E000001
‘ When using SearchDevices
    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 );
‘ When specifying the serial number directly
    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 (on instruments other than GS series or FG400)] Serial number = 27E000001
' When using SearchDevices
    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 );
‘ When specifying the serial number directly
    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 (with YTUSB)] Serial number = 27E000001
‘ When using SearchDevices
```

```
    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 );
‘ When specifying the serial number directly
    int id;
    char encode[256];
    int ret;
    ret = TmcEncodeSerialNumber(encode,256,"27E000001");
    ret = TmcInitialize( TM_CTL_USBTMC3, encode, &id );
[HiSLIP] IP = 11.22.33.44, Port = 4880
    int id;
    int ret = TmcInitialize( TM_CTL_HISLIP, "11.22.33.44:4880", &id );
```

Example: [VBA]

[GPIB] address = 1

```
Dim id As Long
Dim ret As Long
Dim adr As String
adr = "1"
ret = TmlInitialize( 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 = TmlInitialize( 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 = TmlInitialize( 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 = TmlInitialize( 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 (on instruments other than the DL9000 or GS series)] Serial number = 27E000001
‘ When using SearchDevices
    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)

‘ When specifying the serial number directly
    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(on instruments other than GS series or FG400)] Serial number = 27E000001
‘ When using SearchDevices
    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)

‘ When specifying the serial number directly
    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 (with YTUSB)] Serial number = 27E000001
‘ When using SearchDevices
    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)

‘ When specifying the serial number directly
    Dim id As Long
    Dim ret As Long
    Dim encode As String * 128
    ret = TmEncodeSerialNumber(encode, 128, "27E000001")
    ret = TmInitialize(12, encode, id)
```

```
[HiSLIP] IP = 11.22.33.44, Port = 4880
    Dim id As Long
    Dim ret As Long
    ret = TmInitialize( 14, "11.22.33.44:4880", id )
```

6.3.2 InitializeEx

Description: Initializes the interface and opens the interface to the specified device. (support timeout)

Syntax: [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

Parameters:

[IN]	wire	Interface type	※Same definition as Initialize(6.3.1)
[IN]	adr	Connection destination address	※Same definition as Initialize(6.3.1)
[OUT]	id	Device specific ID used with other functions and the like	
[IN]	tmo	Connection timeout (100-ms resolution) Valid when Ethernet, VXI-11, SOCKET.	

Return value:

0	Connection successful
1	Connection error

Detail:

None

Note:

To use multiple devices, execute this function for each device.

Up to 127 devices can be used simultaneously.

Example: [C#]

```
TMCTL cTmctl = new TMCTL();
int ret = 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 );
```

Example: [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 );
```

Example: [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 )
[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 = TmInitializeEx( 11, adr, id, tmo )
```

6.3.3 Finish

Description: Closes the interface to the device.

Syntax: [C#] int Finish(int id)
[VC++] int TmcFinish(int id)
[VBA] TmFinish(ByVal id As Long) As Long

Parameters:

[IN]	id	Device ID

Return value:

0	Success
1	Error

Detail:

Closes the interface opened with “Initialize” (initialization function).
To end communication, be sure to execute this function.

Note:

None

Example: [C#]

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

Example: [VC++]

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

Example: [VBA]

```
Dim ret As Long
ret = TmFinish( id )
```

6.3.4 SearchDevices

Description: Returns a list of devices connected to the specified interface

Syntax: [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

Parameters:

[IN] wire Interface type

Interface type	Support
GPIB	Not supported
RS232	TM_CTL_RS232
USB	Not supported
Ethernet	Not supported
USBTMC(DL9000)	Not supported
Ethernet(UDP)	Not supported
USBTMC (on instruments other than the DL9000)	TM_CTL_USBTMC2
VXI-11	TM_CTL_VXI11
VISAUSB	TM_CTL_VISAUSB
Ethernet(SOCKET)	Not supported
USBTMC (with YTUSB driver)	TM_CTL_USBTMC3
HiSLIP	Not supported

[OUT] list Pointer to a string array that indicates the found device

Undefined for GPIB, RS232, USB, Ethernet, USBTMC (DL9000) and HiSLIP.
A port number is returned for RS232C.
An encoded serial number is returned for USBTMC.
An IP address is returned for VXI-11.
An encoded serial number is returned for VISAUSB.
An encoded serial number is returned for USBTMC(with YTUSB driver).

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

[IN] max Number of arrays of strings that indicate the found devices

[OUT] num Number of found devices

[IN] option Parameter necessary for each interface

Interface type	Support
RS232	Not necessary
USBTMC	Not necessary
VXI-11	Broadcast address converted into a string
VISAUSB	Not necessary
USBTMC(with YTUSB driver)	Not necessary
Other	Undefined

Return value:

0 Success

1 Error

Detail:

None

Note:

None

Example: [C#]

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

// For USBTMC (on instruments other than the DL9000)
ret = cTmctl.SearchDevices(TMCTL.TM_CTL_USBTMC2, list, 127, num, "");
ret = cTmctl.Initialize(TMCTL.TM_CTL_USBTMC2, list[0].adr.ToString(), id);

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

Example: [VB.NET]

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

ReDim Preserve list(127)
num = 0

‘ For USBTMC (on instruments other than the DL9000)
ret = cTmctl.SearchDevices(TMCTL.TM_CTL_USBTMC2, list, 127, num, "")
ret = cTmctl.Initialize(TMCTL.TM_CTL_USBTMC2, list(0).adr, id)

‘ For VXI-11
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);
```

Example: [VC++]

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

// For USBTMC (on instruments other than the DL9000)
ret = TmcSearchDevices(TM_CTL_USBTMC2, list, 127, &num, NULL);
ret = TmcInitialize(TM_CTL_USBTMC2, list[0].adr, id);

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

Example: [VBA]

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

‘ USBTMC (on instruments other than the DL9000)
ret = TmSearchDevices(CTL_USBTMC2, list, MaxStationNum, num, 0)
ret = TmInitialize(CTL_USBTMC2, listarray.list(0).adr, id)

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

6.3.5 SearchDevicesEx

Description: Returns a list of devices connected to the specified interface

Syntax: [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

Parameters:

[IN] wire Interface type

Interface type	Support
GPIB	Not supported
RS232	TM_CTL_RS232
USB	Not supported
Ethernet	Not supported
USBTMC(DL9000)	Not supported
Ethernet(UDP)	Not supported
USBTMC (on instruments other than the DL9000)	TM_CTL_USBTMC2
VXI-11	TM_CTL_VXI11
VISAUSB	TM_CTL_VISAUSB
Ethernet(SOCKET)	Not supported
USBTMC (with YTUSB driver)	TM_CTL_USBTMC3

[OUT] list Pointer to a string array that indicates the found device

An encoded serial number, vendor ID and product ID are returned for USBTMC.
An encoded serial number, vendor ID and product ID are returned for VISAUSB.
An encoded serial number, vendor ID and product ID are returned for
USBTMC (with YTUSB driver).

```
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 Number of arrays of strings that indicate the found devices

[OUT] num Number of found devices

[IN] option Parameter necessary for each interface

Interface type	Support
RS232	Not necessary
USBTMC	Not necessary
VXI-11	Broadcast address converted into a string
VISAUSB	Not necessary
USBTMC(with YTUSB driver)	Not necessary
Other	Undefined

Return value:

0 Success

1 Error

Detail:

None

Note:

None

Example: [C#]

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

// For USBTMC (on instruments other than the DL9000)
ret = cTmctl.SearchDevicesEx(TMCTL.TM_CTL_USBTMC2, list, 127, num, "");
ret = cTmctl.Initialize(TMCTL.TM_CTL_USBTMC2, list[0].adr.ToString(), id);

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

Example: [VB.NET]

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

ReDim Preserve list(127)
num = 0

‘ For USBTMC (on instruments other than the DL9000)
ret = cTmctl.SearchDevicesEx(TMCTL.TM_CTL_USBTMC2, list, 127, num, "")
ret = cTmctl.Initialize(TMCTL.TM_CTL_USBTMC2, list(0).adr, id)

‘ For VXI-11
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);
```

Example: [VC++]

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

// For USBTMC (on instruments other than the DL9000)
ret = TmcSearchDevicesEx(TM_CTL_USBTMC2, list, 127, &num, NULL);
ret = TmcInitialize(TM_CTL_USBTMC2, list[0].adr, id);

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

Example: [VBA]

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

‘ For USBTMC (on instruments other than the DL9000)
ret = TmSearchDevicesEx(CTL_USBTMC2, list, MaxStationNum, num, 0)
ret = TmInitialize(CTL_USBTMC2, listEx.list(0).adr, id)

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

6.3.6 EncodeSerialNumber

Description: Converts the serial number on the nameplate to an internal USB serial number

Syntax: [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

Parameters:

[OUT]	encode	Buffer for storing the converted string (internal USB serial number)
[IN]	len	Size of the encode buffer (bytes)
[IN]	src	Serial number string written on the nameplate

Return value:

0	Success
A value other than 0	Error number

Detail:

None

Note:

None

Example: [C#]

```
StringBuilder encode = New StringBuilder(100);
// Convert the serial number on the nameplate to an internal USB serial number.
ret = cTmctl.EncodeSerialNumber(encode, encode.Capacity, "12W929658");
ret = cTmctl.Initialize(TMCTL.TM_CTL_USBTMC2, encode.ToString(), ref id);
```

Example: [VB.NET]

```
Dim encode As StringBuilder

encode = New StringBuilder(100)
‘ Convert the serial number on the nameplate to an internal USB serial number.
ret = cTmctl.EncodeSerialNumber(encode, encode.Capacity, "12W929658")
ret = cTmctl.Initialize(TMCTL.TM_CTL_USBTMC2, encode.ToString(), ref id)
```

Example: [VC++]

```
char        encode[256] ;
int ret ;
// Convert the serial number on the nameplate to an internal USB serial number.
ret = TmcEncodeSerialNumber(encode,256,”12W929658”) ;

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

Example: [VBA]

```
Dim ret As Long
Dim encode As String * 128
‘ Convert the serial number on the nameplate to an internal USB serial number.
ret = TmEncodeSerialNumber(encode, 128, "TEMP01")
ret = TmInitialize(7, encode, id)
```

6.3.7 DecodeSerialNumber

Description: Converts the internal USB serial number to serial number on the nameplate

Syntax: [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

Parameters:

[OUT]	decode	Buffer for storing the converted string (product serial number written on the nameplate)
[IN]	len	Size of the decode buffer (bytes)
[IN]	src	Internal USB serial number string

Return value:

0	Success
A value other than 0	Error number

Detail:

None

Note:

None

Example: [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);
```

Example: [VB.NET]

```
Dim encode As StringBuilder
Dim decode As StringBuilder
' Check the operation of the Encode/Decode function.
' Encode is necessary to set the serial number of USBTMC (TM_CTL_USBTMC2).
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)
```

Example: [VC++]

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

Example: [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

Description: Sets the communication timeout value

Syntax: [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

Parameters:

[IN] id Device ID

[IN] tmo Timeout value (100-ms resolution) (0 to 65536)
For tmo = 0,
GPIB, RS232, ETHER: Infinite timeout
Other: No timeout

Return value:

0 Success
1 Error

Detail:

Sets the communication timeout value The default value after initialization is 30 s.
We recommend 30 s of longer for YOKOGAWA products.

Note:

In the case of GPIB, the timeout is the value determined as follows.
(100msec,300msec,1sec,3sec,10sec,30sec,100sec,300sec,1000sec)

Example: [C#]

```
int ret = cTmctl.SetTimeout( id, 100 );/* 10sec */
```

Example: [VC++]

```
int ret = TmcSetTimeout( id, 100 );          /* 10sec */
```

Example: [VBA]

```
Dim ret As Long
ret = TmSetTimeout( id, 100 )   ' 10sec
```

6.3.9 SetTerm

Description: Sets the terminator for sending and receiving messages

Syntax: [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

Parameters:

[IN] id Device ID

[IN] eos terminator

Value	Description
0	CR+LF
1	CR
2	LF
3	EOI (GPIB) or none (RS232, USB, Ethernet)

*If the interface type is GPIB and eos is set to 3, use eot to set whether EOI will be used.

[IN] eot EOI use (only for GPIB)

Value	Description
0	Not use
1	Use

Return value:

0 Success
1 Error

Detail:

Sets the terminator. The default value after initialization is “eos = 2, eot = 1.”
We recommend that you use the default value for YOKOGAWA products.
If eos = 2(LF) when receiving binary data, if the binary code contains an LF code, it will be assumed that the data ends there.
However, when using "ReceiveBlockHeader” or "ReceiveBlockData” to receive block data from a YOKOGAWA product, the terminator does not need to be changed.

Note:

With USBTMC (on instruments other than the DL9000), VXI-11, or VISAUSB, nothing will happen even when this function is executed.

Example: [C#]

```
int ret = cTmctl.SetTerm( id, 0, 1 );          /* CR+LF, TRUE */
```

Example: [VC++]

```
int ret = TmcSetTerm( id, 0, 1 ); /* CR+LF, TRUE */
```

Example: [VBA]

```
Dim ret As Long
ret = TmSetTerm( id, 0, 1 )                    ‘ CR+LF, TRUE
```

6.3.10 Send

Description: Sends a message to the device

Syntax: [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

Parameters:

[IN]	id	Device ID
[IN]	msg	Message string

Return value:

0	Success
1	Error

Detail:

Sends an ASCII string to the device specified by the ID value.

When sending binary data, use “SendByLength.”

To send a single transmission message in segments, use “SendSetup” and “SendOnly.”

Note:

None

Example: [C#]

```
int ret = cTmctl.Send( id, “*IDN?” );          /* Send message */
```

Example: [VC++]

```
int ret = TmcSend( id, “*IDN?” );              /* Send message */
```

Example: [VBA]

```
Dim ret As Long
ret = TmSend( id, “*IDN?” )                    ' Send message
```

6.3.11 SendByLength

Description: Send a message with the specified number of bytes to the device.

Syntax: [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

Parameters:

[IN]	id	Device ID
[IN]	msg	Message string
[IN]	len	Number of bytes to send

Return value:

0	Success
1	Error

Detail:

Sends a message to the device specified by the ID value.
Messages can be sent even if they contain binary data.
To send a single transmission message in segments, use “SendSetup” and “SendOnly.”

Note:

None

Example: [C#]

```
int ret = cTmctl.SendByLength( id, “*IDN?”, 5 );        /* Send message */
```

Example: [VC++]

```
int ret = TmcSendByLength( id, “*IDN?”, 5 );            /* Send message */
```

Example: [VBA]

```
Dim ret As Long
ret = TmSendByLength( id, “*IDN?”, 5 )     ' Send message
```

6.3.12 SendSetup

Description: Prepares to send a message to the device

Syntax: [C#] int SendSetup(int id)
[VC++] int TmcSendSetup(int id)
[VBA] TmSendSetup(ByVal id As Long) As Long

Parameters:

[IN]	id	Device ID

Return value:

0	Success
1	Error

Detail:

Prepares to send a message to the device specified by the ID value.

Execute this function once when sending a single message in several segments.

To actually send the message, use “SendOnly.” Sends a message to the device specified by the ID value.

Messages can be sent even if they contain binary data.

Note:

None

Example: [C#]

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

Example: [VC++]

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

Example: [VBA]

```
Dim ret As Long
ret = TmSendSetup( id )
```

6.3.13 SendOnly

Description: Send a message with the specified number of bytes to the device.

Syntax:

```
[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
```

Parameters:

[IN]	id	Device ID
[IN]	msg	Message string
[IN]	data	Send data (binary)
[IN]	len	Number of bytes to send
[IN]	end	Send end flag

Value	Description
0	Transmitting
1	Send end

Return value:

0	Success
1	Error

Detail:

Sends a message to the device specified by the ID value.

Messages can be sent even if they contain binary data.

Only when a message is sent with the Send end flag is set to 1, the terminator is sent at the end of the message.

Therefore, as long as the Send end flag is 0, the device side assumes that data is part of the message.

Note:

None

Example: [C#]

```
int ret;
ret = cTmctl.SendSetup( id );
ret = cTmctl.SendOnly( id, "*ID", 3, 0 );
ret = cTmctl.SendOnly( id, "N?", 2, 1 );      /* End sending the message */
```

Example: [VC++]

```
int ret;
ret = TmcSendSetup ( id );
ret = TmcSendOnly( id, "*ID", 3, 0 );
ret = TmcSendOnly( id, "N?", 2, 1 );                    /* End sending the message */
```

Example: [VBA]

```
Dim ret As Long
ret = TmSendSetup( id )
ret = TmSendOnly( id, "*ID", 3, 0 )
ret = TmSendOnly( id, "N?", 2, 1 )          ' End sending the message
```

6.3.14 Receive

Description: Receives a message from the device

Syntax: [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

Parameters:

[IN]	id	Device ID
[IN]	buff	Receive data buffer
[IN]	blen	Receive size (in units of bytes)
[OUT]	rlen	Number of actually received bytes

Return value:

0	Success
1	Error

Detail:

Receives a message from the device specified by the ID value. If a terminator is detected and data up to that point is received. If it is not detected, data is received up to the number of bytes indicated with blen.

To receive data of "WAVeform:SEND?", "IMAGe:SEND?", and other similar messages from a YOKOGAWA digital oscilloscope, use "ReceiveBlockHeader" and "ReceiveBlockData".

Note:

[VC++ / VBA] The null termination of the received message is not guaranteed.

Example: [C#]

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

Example: [VC++]

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

Example: [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

Description: Prepares to receive a message from the device

Syntax: [C#] int ReceiveSetup(int id)
[VC++] int TmcReceiveSetup(int id)
[VBA] TmReceiveSetup(ByVal id As Long) As Long

Parameters:

[IN]	id	Device ID

Return value:

0	Success
1	Error

Detail:

Execute this command to prepare receiving data when receiving large amounts of data in segments from the device.

Use "ReceiveOnly" to receive the actual data.

Note:

None

Example: [C#]

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

Example: [VC++]

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

Example: [VBA]

```
Dim ret As Long
ret = TmReceiveSetup( id )
```

6.3.16 ReceiveOnly

Description: Receives a message from the device (after it is ready to receive the message)

Syntax: [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

Parameters:

[IN]	id	Device ID
[IN]	buff	Receive data buffer
[IN]	blen	Receive size (in units of bytes)
[OUT]	rlen	Number of actually received bytes

Return value:

0	Success
1	Error

Detail:

Use this command to receive large amounts of data in segments.

After preparing to receive data with “ReceiveSetup,” receive the message from the device specified by the ID number.

If a terminator is detected and data up to that point is received. If it is not detected, data is received up to the number of bytes indicated with blen.

Note:

[VC++ / VBA] The null termination of the received message is not guaranteed.

Example: [C#]

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

// Not continuing to receive data until the return value of text data send/receive CheckEnd
changes to 0
// means that not all the data has been received.
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());
}
```

Example: [VB.NET]

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

' Not continuing to receive data until the return value of text data send/receive CheckEnd
changes to 1
' means that not all the data has been received.
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
```

Example: [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 );
```

Example: [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

Description: Receives the byte size of the block data from the device

Syntax: [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

Parameters:

[IN]	id	Device ID
[OUT]	length	Byte size of block data

Return value:

0	Success
1	Error

Detail:

Use this command to receive the size of block data (messages starting with #).
The number of data bytes that will follow is returned in length. Then, receive the data with “ReceiveBlockData” for the number of bytes + 1 (terminator).

Note:

For the DM7560, the terminator is not included.

Example: [C#]

```
int ret;
int length = 0;
ret = cTmctl.ReceiveBlockHeader( id, ref length );
```

Example: [VC++]

```
int length;
int ret = TmcReceiveBlockHeader( id, &length );
```

Example: [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

Description: Receives the block data from the device (after receiving the byte size).

Syntax: [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

Parameters:

[IN] id	Device ID	
[OUT] buff	Receive data buffer	
[IN] blen	Receive size (in units of bytes)	
[OUT] rlen	Number of actually received bytes	
[OUT] end	Receive end flag	
	Value	**Description**
	0	Receiving (remaining data available)
	1	Receive end

Return value:

0	Success
1	Error

Detail:

Use this command to receive the data section of block data (messages starting with #).

After preparing to receive data with “ReceiveBlockHeader,” receive the message from the device specified by the ID number.

If a terminator is detected and data up to that point is received. If it is not detected, data is received up to the number of bytes indicated with blen.

Note:

On the DM7560, receive end flag cannot be used (always 0). In addition, set the ReceiveBlockHeader length in blen.

On the SOCKET interface, the reception end flag is set when the specified size is received.

Example: [C#]

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

// Receive binary data
// If the device has not received data in advance, 1000 points of data is not returned.
// buff = ":DATA:RAW? 1,1,1,1000";          // For the SL1000
buff = ":WAVedata:SEND:BINary?";          // For the AQ7270

ret = cTmctl.Send(m_ID, buff);
ret = cTmctl.ReceiveBlockHeader(m_ID, rlen);
data = new sbyte[rlen];
while (end1 == 0)     // Continue to receive data until the end flag is set.
{
    ret = cTmctl.ReceiveBlockData(m_ID, data[totalsize], rlen, ref datasize, ref end1);
    if (ret != 0) break;
    totalsize += datasize;
}
```

Example: [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

' Receive binary data
' If the device has not received data in advance, 1000 points of data is not returned.
' buff = ":DATA:RAW? 1,1,1,1000"            ' For the SL1000
buff = ":WAVedata:SEND:BINary?"           ' For the AQ7270

rlen = 0
end1 = 0
ret = cTmctl.Send(m_ID, buff)
ret = cTmctl.ReceiveBlockHeader(m_ID, rlen)
While (end1 <> 1)    ' Continue to receive data until the end flag is set.
    ret = cTmctl.ReceiveBlockData(m_ID, data(0), rlen, datasize, end1)
End While
```

Example: [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 );
}
```

Example: [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

Description: Returns the error number of the error that occurred last.

Syntax: [C#] int GetLastError(int id)
[VC++] int TmcGetLastError(int id)
[VBA] TmGetLastError(ByVal id As Long) As Long

Parameters:

[IN] id Device ID

Return value:

Error number

Detail:

Returns the error number of the error that occurred last in the device.

Use this function to obtain the actual error number when the return value of a function, including the initialization function, is not 0 (OK).

Note:

None

Example: [C#]

```
int err;
ret = cTmctl.Send( id, "START" )
if( ret != 0 ) {
    err = cTmctl.GetLastError( id );
}
```

Example: [VC++]

```
int ret = TmcSend( id, "START" );
if( ret != 0 ) {
    int err = TmcGetLastError( id );
}
```

Example: [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

Description: Sets the device in remote or local mode

Syntax: [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

Parameters:

[IN] id Device ID

[IN] flag Remote/local designation

Value	Description
0	Local
1	Remote control input

Return value:

0	Success
1	Error

Detail:

The behavior varies slightly depending on the interface type.

For GPIB, the REN line is set to true or false. The device is actually set to remote mode when another message is sent. (Device-specific remote/local control is not performed.)

For RS232, USB, and Ethernet, this command can be used only on 488.2-compliant YOKOGAWA products that support the COMMunicate group messages.

If this applies, each device can be controlled separately.

For USBTMC and VISAUSB, remote/local switching is performed through control transfer.

Note:

Use of this command on interface types other than GPIB is limited only to YOKOGAWA products.

Example: [C#]

```
int ret = cTmctl.SetRen( id, 1 );
```

Example: [VC++]

```
int ret = TmcSetRen( id, 1 );
```

Example: [VBA]

```
Dim ret As Long
ret = TmSetRen( id, 1 )
```

6.3.21 CheckEnd

Description: Returns whether the message from the device has ended.

Syntax: [C#] int CheckEnd(int id)
[VC++] int TmcCheckEnd(int id)
[VBA] TmCheckEnd(ByVal id As Long) As Long

Parameters:

[IN]	id	Device ID

Return value:

0	Message end
1	Error or message present

Detail:

This command can be used on GPIB, USB, Ethernet, USBTMC, VXI-11, and VISAUSB interface types.
When a series of receive messages is received in segments, “ReceiveOnly” returns whether the message has been received completely.
(This command always returns 0 for RS232 and SOCKET.)

Note:

None

Example: [C#]

```
int ret = cTmctl.CheckEnd( id )
if( ret == 0 ) {          /* Receive end */
}
else {                    /* Receive continue */
}
```

Example: [VC++]

```
int ret = TmcCheckEnd( id );
if( ret == 0 ) {          /* Receive end */
}
else {                    /* Receive continue */
}
```

Example: [VBA]

```
Dim ret As Long
ret = TmCheckEnd( id )
If( ret == 0 ) Then
    ' Receive end
Else
    ' Receive continue
Endif
```

6.3.22 DeviceClear

Description: Clears the selected device (SDC)

Syntax: [C#] int DeviceClear(int id)
[VC++] int TmcDeviceClear(int id)
[VBA] TmDeviceClear(ByVal id As Long) As Long

Parameters:

[IN]	id	Device ID

Return value:

0	Success
1	Error

Detail:

None

Note:

This function is exclusive to GPIB, USBTMC (on instruments other than the DL9000), VXI-11, VISAUSB, and USBTMC (with YTUSB driver).

It will not do anything to devices connected to other interface types (0 is always returned).

Example: [C#]

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

Example: [VC++]

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

Example: [VBA]

```
Dim ret As Long
ret = TmDeviceClear( id )
```

6.3.23 DeviceTrigger

Description: Sends a trigger message to the device

Syntax: [C#] int DeviceTrigger(int id)
[VC++] int TmcDeviceTrigger(int id)
[VBA] TmDeviceTrigger(ByVal id As Long) As Long

Parameters:

[IN]	id	Device ID

Return value:

0	Success
1	Error

Detail:

None

Note:

This function is exclusive to GPIB, USBTMC (on instruments other than the DL9000), VXI-11, VISAUSB, and USBTMC (YTUSB driver).

It will not do anything to devices connected to other interface types (0 is always returned).

Example: [C#]

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

Example: [VC++]

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

Example: [VBA]

```
Dim ret As Long
ret = TmDeviceTrigger( id )
```

6.3.24 WaitSRQ

Description: Receives an SRQ from the specified device

Syntax: [C#] int WaitSRQ(int id, ref byte stsbyte, int tmo)
[VC++] int TmcWaitSRQ(int id, char* stsbyte, int tmo)
[VBA] -

Parameters:

[IN]	id	Device ID
[OUT]	stsbyte	Cause of SRQ
[IN]	tmo	Timeout value (100 ms resolution)

Return value:

0	Success
1	Error

Detail:

None

Note:

This cannot be used with VBA.

Only GPIB, USBTMC (on instruments other than the DL9000), VXI-11, VISAUSB, and USBTMC (with YTUSB driver) are supported. This function always returns 0 for other interface types.

In the case of GPIB, the timeout is the value determined as follows.

(100msec,300msec,1sec,3sec,10sec,30sec,100sec,300sec,1000sec)

Example: [C#]

```
byte sts = 0;
// Wait for an SRQ with a 10 s timeout
ret = cTmctl.WaitSRQ(id, ref sts, 100);
```

Example: [VB.NET]

```
Dim sts As Byte
// Wait for an SRQ with a 10 s timeout
ret = cTmctl.WaitSRQ(id, sts, 100);
```

Example: [VC++]

```
int         ret
char        sts ;

// Wait for an SRQ with a 10 s timeout
ret = TmcWaitSRQ(id, &sts, 100) ;
```

6.3.25 AbortWaitSRQ

Description: Releases the wait state of the SRQ wait function for the specified device

Syntax: [C#] int AbortWaitSRQ(int id)
[VC++] int TmcAbortWaitSRQ(int id)
[VBA] -

Parameters:

[IN]	id	Device ID

Return value:

0	Success
1	Error

Detail:

None

Note:

This cannot be used with VBA.

Only USBTMC (on instruments other than the DL9000), VXI-11, VISAUSB, and USBTMC (with YTUSB driver) are supported. This function always returns 0 for other interface types.

Example: [C#]

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

Example: [VC++]

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

6.3.26 SetCallback

Description: Registers the callback routine for SRQs

Syntax: [C#] int SetCallback(int id, Hndlr func, uint p1, uint p2)
[VC++] int TmcSetCallback(int id, Hndlr func, ULONG p1,ULONG p2)
[VBA] -

Parameters:

[IN]	id	Device ID
[IN]	func	Callback function for when SRQs occur public delegate void Hndlr(int id, byte buff, uint p1, uint p2) Set the pointer to the callback function that is called when SRQs occur.
[IN]	p1	First parameter of the callback function
[IN]	p2	Second parameter of the callback function

Return value:

0	Success
1	Error

Detail:

The callback function is called from a callback thread that is created within the library.

Note:

This cannot be used with VBA.

Only USBTMC (on instruments other than the DL9000), VXI-11, VISAUSB, and USBTMC (with YTUSB driver) are supported. This function always returns 0 for other interface types.

Example: [C#]

```
// Example in which a received SRQ is obtained with a callback function
public TMCTL.Hndlr method;
public uint p1 = 1;
public uint p2 = 2;

public void Func1(int id, byte buff, uint p1, uint p2)
{
    // Callback function for obtaining SRQs
    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);

    // Set the callback function.
    ret = cTmctl.SetCallback(m_ID, method, p1, p2);
    Console.WriteLine("SetCallback:ret={0}", ret);
}
```

Example: [VB.NET]

```
' Example in which a received SRQ is obtained with a callback function
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)
    ' Callback function for obtaining SRQs
    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
    ' Set the callback function.
    ret = tmctl.SetCallback(m_ID, Method, p1, p2)
    Console.WriteLine("SetCallback:ret={0}", ret)
End Sub
```

Example: [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

Description: Deletes the callback routine for SRQs

Syntax: [C#] int ResetCallback(int id)
[VC++] int TmcResetCallback(int id)
[VBA] -

Parameters:

[IN]	id	Device ID

Return value:

0	Success
1	Error

Detail:

None

Note:

This cannot be used with VBA.

Only USBTMC (on instruments other than the DL9000), VXI-11, VISAUSB, and USBTMC (with YTUSB driver) are supported. This function always returns 0 for other interface types.

Example: [C#]

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

Example: [VC++]

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

7 Sample Programs

7.1 C# Environment

```
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(on instruments other than the DL9000 or GS series) 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(with YTUSB driver) 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 Environment

```
Imports System.Text
Imports TmctlAPINet

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

        Dim encode As StringBuilder
        Dim buff As StringBuilder
        Dim length As Integer

        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 (on instruments other than the DL9000 or GS series) 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.TM_CTL_VISAUSB, "27E000001", id)
        ' ex11: VISAUSB (GS610) Serial Number = 27E000001
        ret = cTmctl.Initialize( TMCTL. TM_CTL_VISAUSB , "27E000001C", id)
        ' ex12: VISAUSB (on instruments other than GS series or FG400) Serial Number = 27E000001
        ret = cTmctl.EncodeSerialNumber( encode, encode.Capacity, "27E000001")
        ret = cTmctl.Initialize( TMCTL. TM_CTL_VISAUSB , encode.ToString(), id)
        ' ex13: USBTMC (with YTUSB driver) Serial Number = 27E000001
        ret = cTmctl.EncodeSerialNumber( encode, encode.Capacity, "27E000001")
        ret = cTmctl.Initialize( TMCTL.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
' Send *RST
ret = cTmctl.Send( id, "*RST" )
If( ret <> 0 ) Then
        ExecuteCommunicate = cTmctl.GetLastError( id )
        Return ExecuteCommunicate
End If
' Send *IDN? and receive query
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
' Receive block data
ret = cTmctl.Send( id, ":WAVEFORM:FORMAT ASCII;:WAVEFORM:SEND?" )
If( ret <> 0 ) Then
        ExecuteCommunicate = cTmctl.GetLastError( id )
        Return ExecuteCommunicate
End if
ret = cTmctl.ReceiveBlockHeader(id, length)
If (ret <> 0) Then
        ExecuteCommunicate = cTmctl.GetLastError(id)
        Return ExecuteCommunicate
End If
endflag = 0
While (endflag <> 1) ' Continue to receive data until the end flag is set.
        ret = cTmctl.ReceiveBlockData(id, blockData(0), blockData.Length, length, endflag)
        If (ret <> 0) Then
                Exit While
        End If
```

```
        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++ Environment

```
#include "tmctl.h"

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

	// Example 1: GPIB address = 1
	ret = TmcInitialize( TM_CTL_GPIB, "1", &id );
	// Example 2: RS232 COM1, 57600, 8-NO-1, CTS-RTS
	ret = TmcInitialize( TM_CTL_RS232, "1,6,0,1", &id );
	// Example 3: USB ID = 1
	ret = TmcInitialize( TM_CTL_USB, "1", &id );
	// Example 4: Ethernet IP = 11.22.33.44, User name = yokogawa, Password = abcdefgh
	ret = TmcInitialize( TM_CTL_ETHER, "11.22.33.44,yokogawa,abcdefgh", &id );
	// Example 5: USBTMC (DL9000) Serial Number = 27E000001
	ret = TmcInitialize( TM_CTL_USBTMC, "27E000001", &id );
	// Example 6: USBTMC (GS200, GS820) Serial Number = 27E000001
	ret = TmcInitialize( TM_CTL_USBTMC2, "27E000001", &id );
	// Example 7: USBTMC (GS610) Serial Number = 27E000001
	ret = TmcInitialize( TM_CTL_USBTMC2, "27E000001C", &id );
	// Example 8: USBTMC (on instruments other than the DL9000 or GS series) Serial Number =
27E000001
	char encode[256] ;
	ret = TmcEncodeSerialNumber(encode,256,"27E000001") ;
	ret = TmcInitialize( TM_CTL_USBTMC2, encode, &id) ;
	// Example 9: VXI-11 IP = 11.22.33.44
	ret = TmcInitialize( TM_CTL_VXI11, "11.22.33.44", &id );
	// Example 10: VISAUSB (GS200, GS820, FG400) Serial Number = 27E000001
	ret = TmcInitialize( TM_CTL_ VISAUSB , "27E000001", &id );
	// Example 11: VISAUSB (GS610) Serial Number = 27E000001
	ret = TmcInitialize( TM_CTL_ VISAUSB, "27E000001C", &id );
	// Example 12: VISAUSB (on instruments other than GS series or FG400) Serial Number = 27E000001
	char encode[256] ;
	ret = TmcEncodeSerialNumber(encode,256,"27E000001") ;
	ret = TmcInitialize( TM_CTL_VISAUSB, encode, &id) ;
	// Example 13: USBTMC (with YTUSB driver) 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 );
        }
        // Send *RST
        ret = TmcSend( id, "*RST" );
        if( ret != 0 ) {
                return TmcGetLastError( id );
        }
        // Send *IDN? and receive query
        ret = TmcSend( id, "*IDN?" );
        if( ret != 0 ) {
                return TmcGetLastError( id );
        }
        ret = TmcReceive( id, buf, 1000, &length );
        if( ret != 0 ) {
                return TmcGetLastError( id );
        }
        // Receive block data
        ret = TmcSend( id, ":WAVEFORM:FORMAT ASCII;:WAVEFORM:SEND?" );
        if( ret != 0 ) {
                return TmcGetLastError( id );
        }
        ret = TmcReceiveBlockHeader(m_ID, rlen);
        while (endflag == 0) {          // Continue to receive data until the end flag is set.
                ret = TmcReceiveBlockData(m_ID, buf, 1000, rlen, endflag);
        }
        ret = TmcFinish( id );
        if( ret != 0 ) {
                return TmcGetLastError( id );
        }
}
```

7.4 Visual Basic for Applications Environment

```
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
        ' Example 1: GPIB address = 1
        adr = "1"
        ret = TmInitialize( 1, adr, id )
        ' Example 2: RS232 COM1, 57600, 8-NO-1, CTS-RTS
        adr = "1,6,0,2"
        ret = TmInitialize( 2, adr, id )
        ' Example 3: USB ID = 1
        adr = "1"
        ret = TmInitialize( 3, adr, id )
        ' Example 4: Ethernet IP = 11.22.33.44, User name = yokogawa, Password = abcdefgh
        adr = "11.22.33.44,yokogawa,abcdefgh"
        ret = TmInitialize( 4, adr, id )
        ' Example 5: USBTMC (DL9000) Serial Number = 27E000001
        adr = "27E000001"
        ret = TmInitialize( 5, adr, id )
        ' Example 6: USBTMC (GS200, GS820) Serial Number = 27E000001
        adr = "27E000001"
        ret = TmInitialize( 7, adr, id )
        ' Example 7: USBTMC (GS610) Serial Number = 27E000001
        adr = "27E000001C"
        ret = TmInitialize( 7, adr, id )
        ' Example 8: USBTMC (on instruments other than the DL9000 or GS series) Serial Number = 27E000001
        Dim encode As String * 128
        ret = TmEncodeSerialNumber(encode,128,"27E000001")
        ret = TmInitialize( 7, encode, &id)
        ' Example 9: VXI-11 IP = 11.22.33.44
        ret = TmcInitialize( 8, "11.22.33.44", &id )
        ' Example 10: VISA (GS200, GS820, FG400) Serial Number = 27E000001
        adr = "27E000001"
        ret = TmInitialize( 10, adr, id )
        ' Example 11: VISAUSB (GS610) Serial Number = 27E000001
        adr = "27E000001C"
        ret = TmInitialize(10, adr, id )
        ‘ Example 12: VISAUSB (on instruments other than GS series or FG400) Serial Number = 27E000001
        Dim encode As String * 128
        ret = TmEncodeSerialNumber(encode,128,"27E000001")
        ret = TmInitialize( 10, encode, &id)
        ‘ Example 13: USBTMC (with YTUSB driver) 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
' Send *RST
ret = TmSend( id, "*RST" )
If( ret <> 0 ) Then
        ExecuteCommunicate = TmGetLastError( id )
        Exit Function
Endif
' Send *IDN? and receive query
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
' Receive block data
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) ' Continue to receive data until the end flag is set.
        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 Appendix

8.1 Device Compatibility Table

Series \ Protocol	GPIB	RS232	USB	USBTMC (YKMUSB)	VISAUSB	Ethernet (Legacy)	VXI-11	UDP	SOCKET	HiSLIP	USBTMC (YTUSB)	Notes
DLM5000	✓	-	-	-	✓	-	✓	-	✓	-	✓	
DLM4000	✓	-	-	✓	✓	-	✓	-	-	-	-	
DLM3000	✓	-	-	-	✓	-	✓	-	✓	-	✓	
DLM2000	✓	-	-	✓	✓	-	✓	-	-	-	-	
DL/DLM6000	✓	-	-	✓	-	✓	-	-	-	-	-	
DL950	-	-	-	-	✓	-	✓	-	✓	✓	✓	
DL350	-	-	-	✓	✓	-	✓	-	-	-	-	
DL850E/DL850EV	✓	-	-	✓	✓	-	✓	-	-	-	-	
DL850/DL850V	✓	-	-	✓	✓	-	✓	-	-	-	-	
DL750	✓	✓	✓	-	-	✓	-	-	-	-	-	
DL750P	✓	✓	✓	-	-	✓	-	-	-	-	-	
DL1600	✓	✓	✓	-	-	✓	-	-	-	-	-	
DL1700E	✓	-	✓	-	-	✓	-	-	-	-	-	
DL1740	✓	-	✓ *1	-	-	✓ *2	-	-	-	-	-	*1: Version 1.10 and later *2: Version 1.30 and later
DL1720	✓	-	✓	-	-	✓ *3	-	-	-	-	-	*3: Version 1.30 and later
DL9000	✓	-	-	✓	-	✓	-	-	-	-	-	
SB5000	✓	-	-	✓	-	✓	✓	-	-	-	-	
DL7400	✓	-	✓	-	-	✓	-	-	-	-	-	
DL7200	✓	✓	-	-	-	✓ *4	-	-	-	-	-	*4: Version 3.02 and later
DL7100	✓	✓	-	-	-	✓ *5	-	-	-	-	-	*5: Version 3.02 and later
FG400	✓	-	-	-	✓	-	-	-	-	-	-	
WT5000	✓	-	-	-	✓	-	✓	-	-	-	✓	
WT3000E	✓	✓	✓	-	-	✓	-	-	-	-	-	
WT3000	✓	✓	✓ *6	-	-	✓ *6	-	-	-	-	-	*6: Version 2.01 and later
WT1800E	✓	-	-	✓	✓	-	✓	-	-	-	-	
WT1800	✓	-	-	✓	✓	-	✓	-	-	-	-	
WT1600	✓	✓	-	-	-	✓ *7	-	-	-	-	-	*7: Version 2.01 and later
WT500	✓	-	-	✓	✓	-	✓	-	-	-	-	
WT300E	✓	✓	-	✓	✓	-	✓	-	-	-	-	
WT300	✓	✓	-	✓	✓	-	✓	-	-	-	-	
PX8000	✓	-	-	✓	✓	-	✓	-	-	-	-	
SL1000	-	-	-	✓	✓	-	✓	-	-	-	-	

Series \ Protocol	GPIB	RS232	USB	USBTMC (YKMUSB)	VISAUSB	Ethernet (Legacy)	VXI-11	UDP	SOCKET	HiSLIP	USBTMC (YTUSB)	Notes
SL1400	✓	✓	✓	-	-	✓	-	-	-	-	-	
GS820	✓	✓	-	✓	✓	-	✓	-	✓	-	-	
GS610	✓	✓	-	✓	✓	-	-	-	✓	-	-	
GS200	✓	-	-	✓	✓	-	✓	-	✓	-	-	
2553A	✓	-	-	✓	✓	-	✓	-	-	-	-	
2558A	✓	-	-	✓	✓	-	✓	-	-	-	-	
2560A	✓	-	-	✓	✓	-	✓	-	-	-	-	
LS3300	✓	-	-	✓	✓	-	✓	-	-	-	-	
AQ7280	-	-	-	✓	✓	✓	-	-	-	-	-	
AQ7270	-	-	✓	-	-	✓	-	-	-	-	-	
AQ7260	-	-	✓	-	-	-	-	-	-	-	-	
AQ2211/AQ2212	✓	-	-	✓	-	✓	-	-	-	-	-	
AQ1300	-	-	-	✓	✓	✓	-	-	-	-	-	
AQ1210	-	-	-	-	✓	-	-	-	-	-	✓	
AQ1200	-	-	-	✓	✓	✓	-	-	-	-	-	
AQ1100	-	-	-	✓	✓	✓	-	-	-	-	-	
AQ1000	-	-	-	✓	✓	-	-	-	-	-	-	
732050	-	-	-	-	-	-	-	✓	-	-	-	
DM7560	✓	✓ *8	-	-	-	-	-	-	✓	-	-	*8: USB can be handled as RS232 by using a COM port driver.
MT300	✓	✓	-	-	✓	-	✓	-	-	-	✓	*8: USB can be handled as RS232 by using a COM port driver.