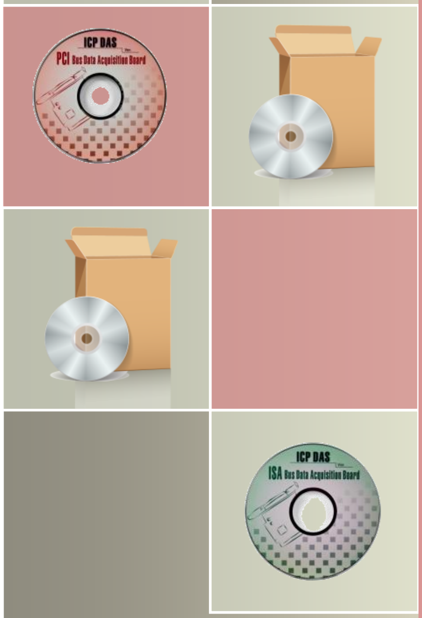
PCI-TMC12 Series Classic Driver DLL Software Manual

Version 2.2, Sep. 2015


ICP DAS
PCI Bus Data Acquisition Board
ICP DAS
ISA Bus Data Acquisition Board

SUPPORTS

Board includes PCI-TMC12 and PCI-TMC12A.

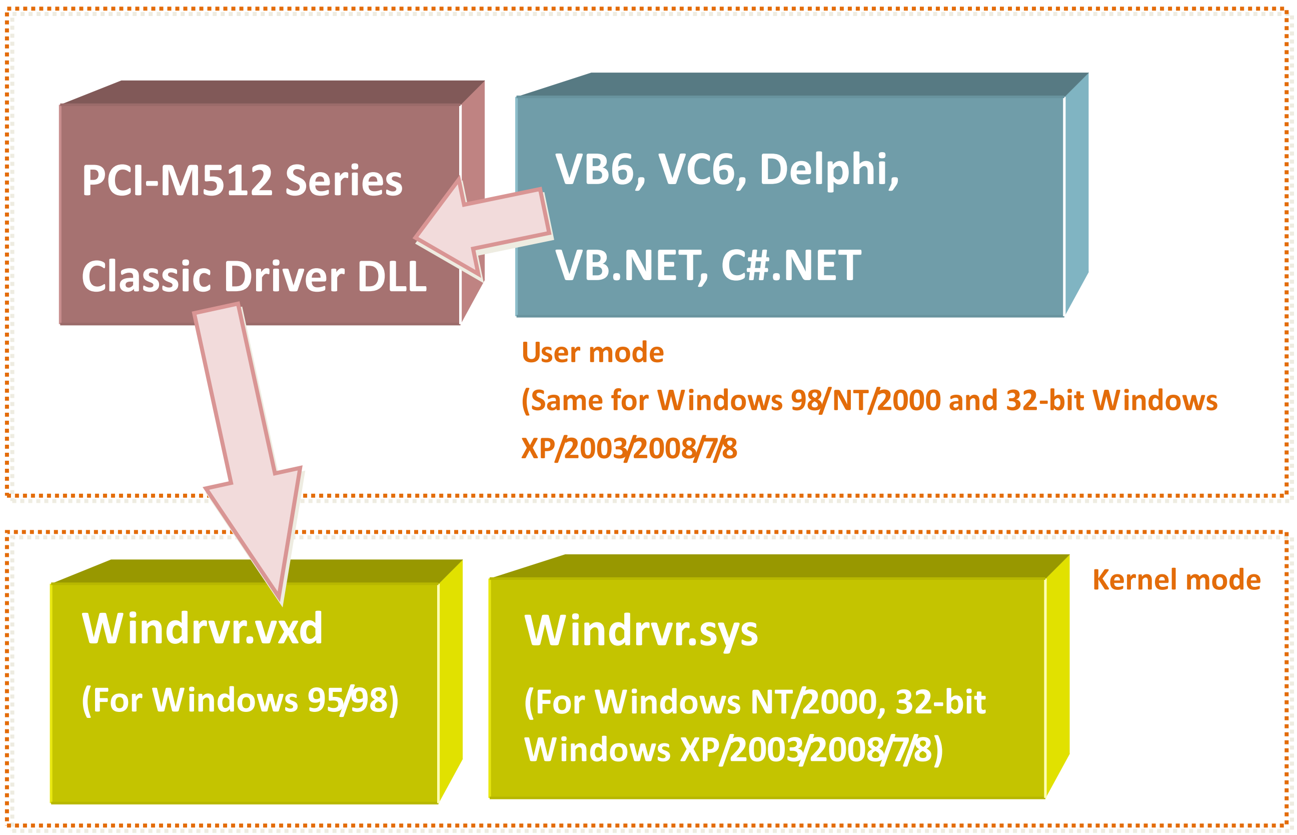
CONTACT US

If you have any question, feel to contact us by email at:
Email: service@icpdas.com or service.icpdas@gmail.com
We will respond to you within 2 working days.

Table of Contents

1. Introduction

There are many demo program, written in VB, VC, Delphi, Borland C++ Builder, VB.NET, VC.NET and C# given in the companion CD. These demo programs will call the PCITMC12.DLL to access the hardware of PCI-TMC12. The PCITMC12.DLL will call the kernel driver, Windrvr.vxd or Windrvr.sys as follows:



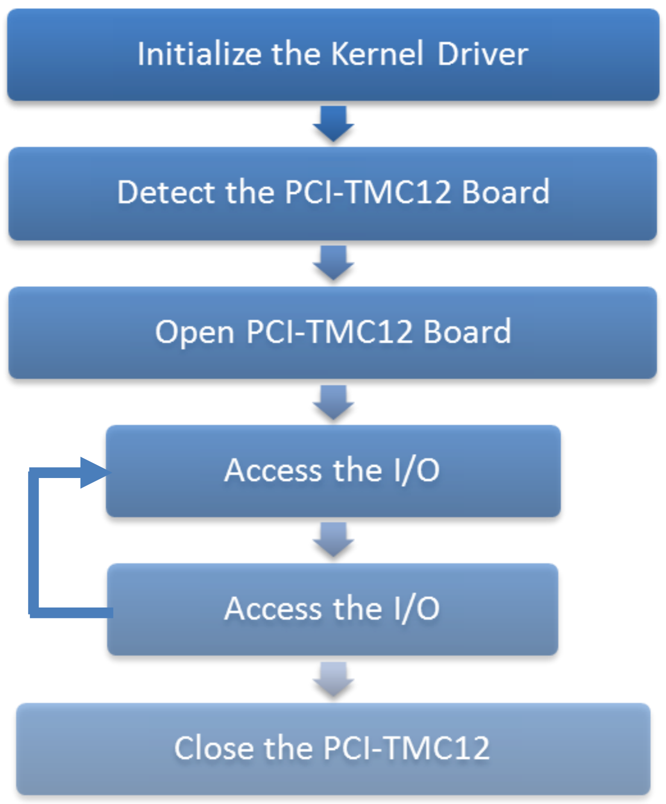
The install shields will install kernel driver, DLL driver & application demo program to system. All demo program & DLL are same for Windows 95/98/NT/2000 and 32-bit Windows XP/2003/Vista/7/8. But the kernel driver is different for different system as follows:

- For Windows 95/98 → will copy **WINDRVR.VXD** to C:\WIN95\SYSTEM\VMM32
- For Windows NT → will copy **WINDRVR.SYS** to C:\WINNT\SYSTEM32\DRIVERS
- For Windows 2000/XP/2003/Vista/2008/7 (32-bit) → will copy **WINDRVR.SYS** to C:\WINDOWS\SYSTEM32\DRIVERS

All DLL and demo program will not work if the kernel driver is not installed correctly. The install shields will copy the correct kernel driver to correct position if you select correct O.S. (Window 95/98/NT/2000/XP/2003/Vista/7/8).

The install shields also copy all related documentations of PCI-TMC12 series to user's hard disk. Refer to Calling DLL functions in VB、VC、Delphi、 BCB.pdf (http://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/manual/) for more information about how to call the DLL functions with VB, VC, Delphi, Borland C++ Builder, VB.NET, VC.NET and C#.

The software architecture is given as follows:



PTMC12_DriverInit()

PTMC12_DetectCards()

PTMC12_OpenBoard(…)
………………

PTMC12_ReadWord(…)
………………
………………
PTMC12_WriteWord(…)
……………..

PTMC12_CloseAll()

Note:

1. *PCI-TMC12 series maybe PCI-TMC12 or PCI-TMC12A. Refer to Section 2.4.1 of "PCI-TMC12(A) User's Manual" (CD: \NAPDOS\PCI\PCI-TMC12A\ Manual\) for comparison of TMC12 and TMC12A.*
2. *If **J28** jumper of PCI-TMC12A is set to TMC12, this PCI-TMC12A can use PTMC12.DLL as same as PCI-TMC12.*
3. *PTMC12.DLL is designed for PCI-TMC12.*

1.1 Reference

- **PnP Driver Installation.pdf:**

Describes how to install the PnP (Plug and Play) driver for PCI card under Windows 95/98/2000/XP/2003/Vista/7/8(32-bit).
http://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/manual/

- **Software Installation Guide.pdf:**

Describes how to install the software package under Windows 95/98/ 2000/XP/2003/Vista/2008/7/8(32-bit).
http://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/manual/

- **Calling DLL Functions.pdf:**

Describes how to call the DLL functions with VC++6, VB6, Delphi4 and Borland C++ Builder 4.
http://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/manual/

- **Resource Checking .pdf:**

Describes how to check the resources I/O Port address, IRQ number and DMA number for add-on cards under Windows 95/98/2000/XP/2003/ Vista/2008/7(32-bit).
http://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/manual/

- **PCI-TMC12 Hardware manual.pdf:**

PCI-TMC12 series hardware manual.
http://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/pci-tmc12a/manual/

2. Software Installation

2.1 Obtaining the Driver Installer Package

PCI-TMC12 series card can be used on Linux and Windows 98/NT/2000 and 32-bit XP/2003/Vista/7/8 based systems, and the drivers are fully Plug and Play (PnP) compliant for easy installation.

The driver installer package for the PCI-TMC12 series can be found on the supplied CD-ROM, or can be obtained from the ICP DAS FTP web site. The location and addresses are indicated in the table below:

CD:\\NAPDOS\PCI\PCI-TMC12A\DLL\Driver\
ftp://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/pci-tmc12a/dll_ocx/driver/
http://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/pci-tmc12a/dll_ocx/driver/

Install the appropriate driver for your operating system, as follows:

Name	OS
PCI-TMC12_Win_Setup_xxx.exe	For Windows 95, Windows 98, Windows NT, Windows 2000, 32-bit Windows XP, 32-bit Windows 2003, 32-bit Windows Vista, 32-bit Windows 7 and 32-bit Windows 8 .

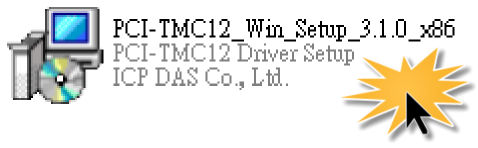
2.2 Driver Installing Procedure

Before the driver installation, you must complete the hardware installation. For detailed information about the hardware installation, please refer to appropriate hardware user manual for your PCI-TMC12 series card. The hardware user manual is contained in:

CD:\NAPDOS\PCI\PCI-TMC12A \Manual\

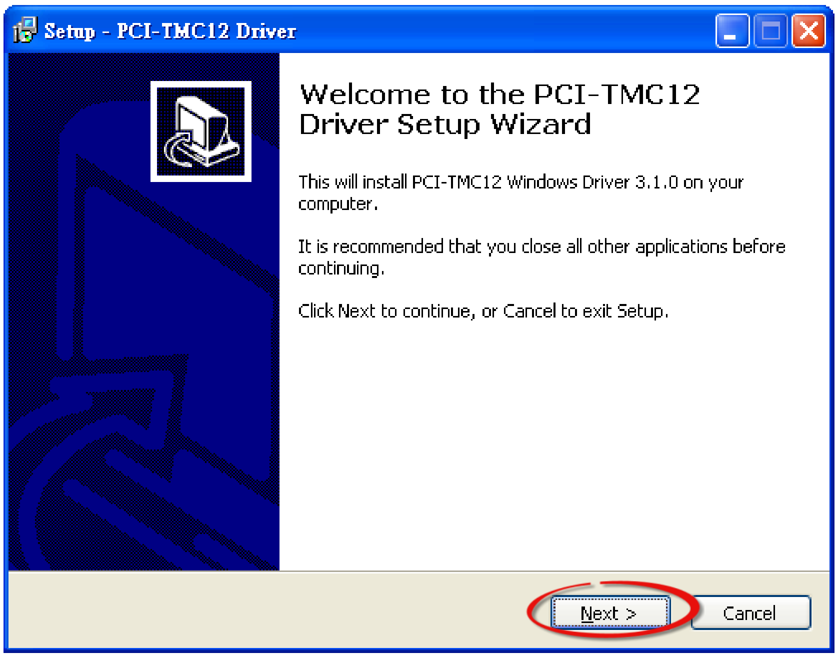
http://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/pci-tmc12a/manual/

To install the PCI-TMC12 series classic drivers, follow the procedure described below:



Step 1: Double-Click **"PCI-TMC12_Win_Setup_xxxx.exe"** to install driver.

Step 2: Click the **"Next>"** button to start the installation on the **"Setup – PCI-TMC12 Driver"** window.



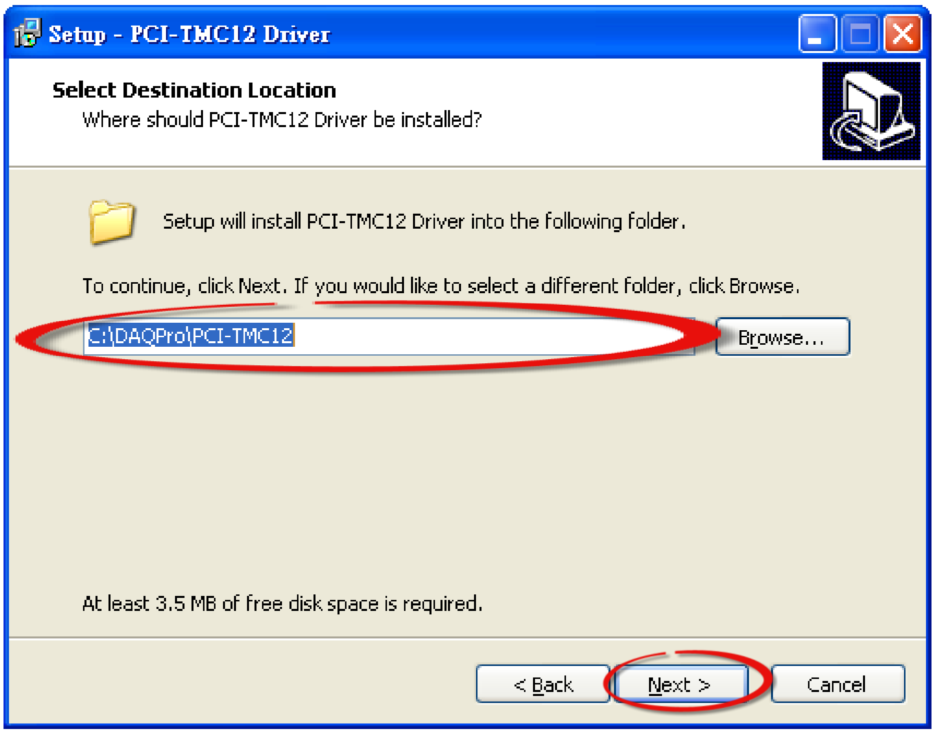
Welcome to the PCI-TMC12 Driver Setup Wizard

This will install PCI-TMC12 Windows Driver 3.1.0 on your computer.

It is recommended that you close all other applications before continuing.

Click Next to continue, or Cancel to exit Setup.

Step 3: Click the **"Next>"** button to install the driver into the **default** folder.



Select Destination Location

Where should PCI-TMC12 Driver be installed?

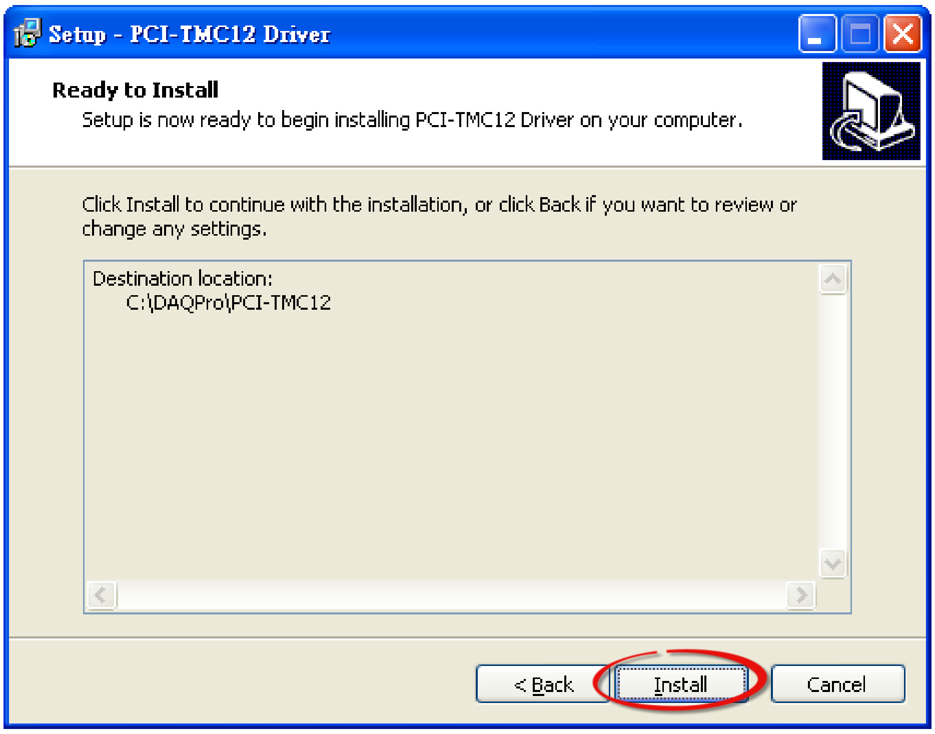
Setup will install PCI-TMC12 Driver into the following folder.

To continue, click Next. If you would like to select a different folder, click Browse.

C:\DAQPro\PCI-TMC12

At least 3.5 MB of free disk space is required.

Step 4: Click the **"Install"** button to continue.



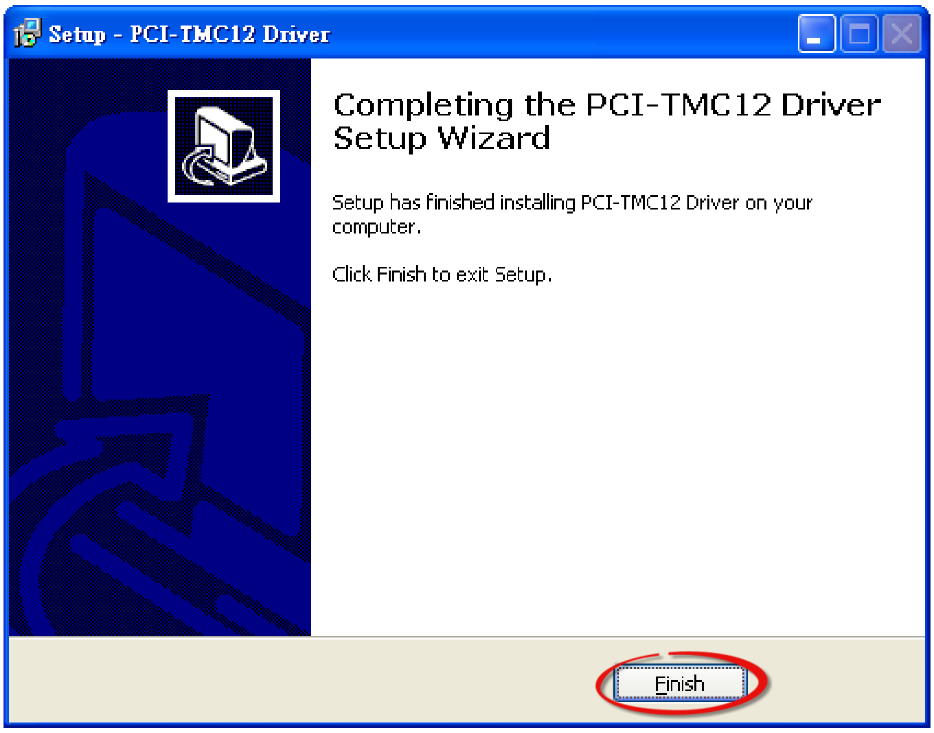
Ready to Install

Setup is now ready to begin installing PCI-TMC12 Driver on your computer.

Click Install to continue with the installation, or click Back if you want to review or change any settings.

Destination location:
    C:\DAQPro\PCI-TMC12

Step 5: Click the **"Finish"** button.



Completing the PCI-TMC12 Driver Setup Wizard

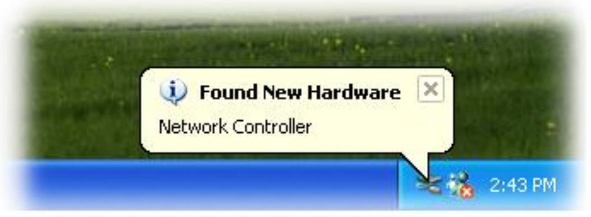
Setup has finished installing PCI-TMC12 Driver on your computer.

Click Finish to exit Setup.

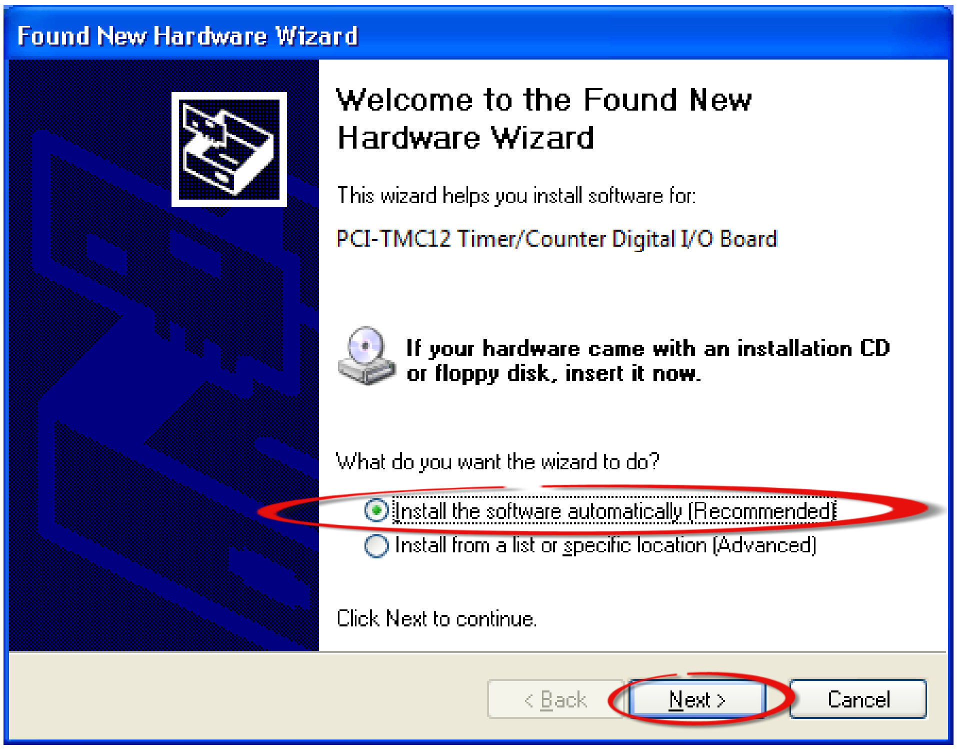
2.3 PnP Driver Installation

Step 1: The system should find the new card and then continue to finish the Plug&Play steps.

Note: More recent operating systems, such as Windows 7/8 will automatically detect the new hardware and install the necessary drivers etc., so Steps 2 to 4 can be skipped.


Found New Hardware
Network Controller

Step 2: Select **"Install the software automatically [Recommended]"** and click the **"Next>"** button.



Welcome to the Found New Hardware Wizard

This wizard helps you install software for:

PCI-TMC12 Timer/Counter Digital I/O Board

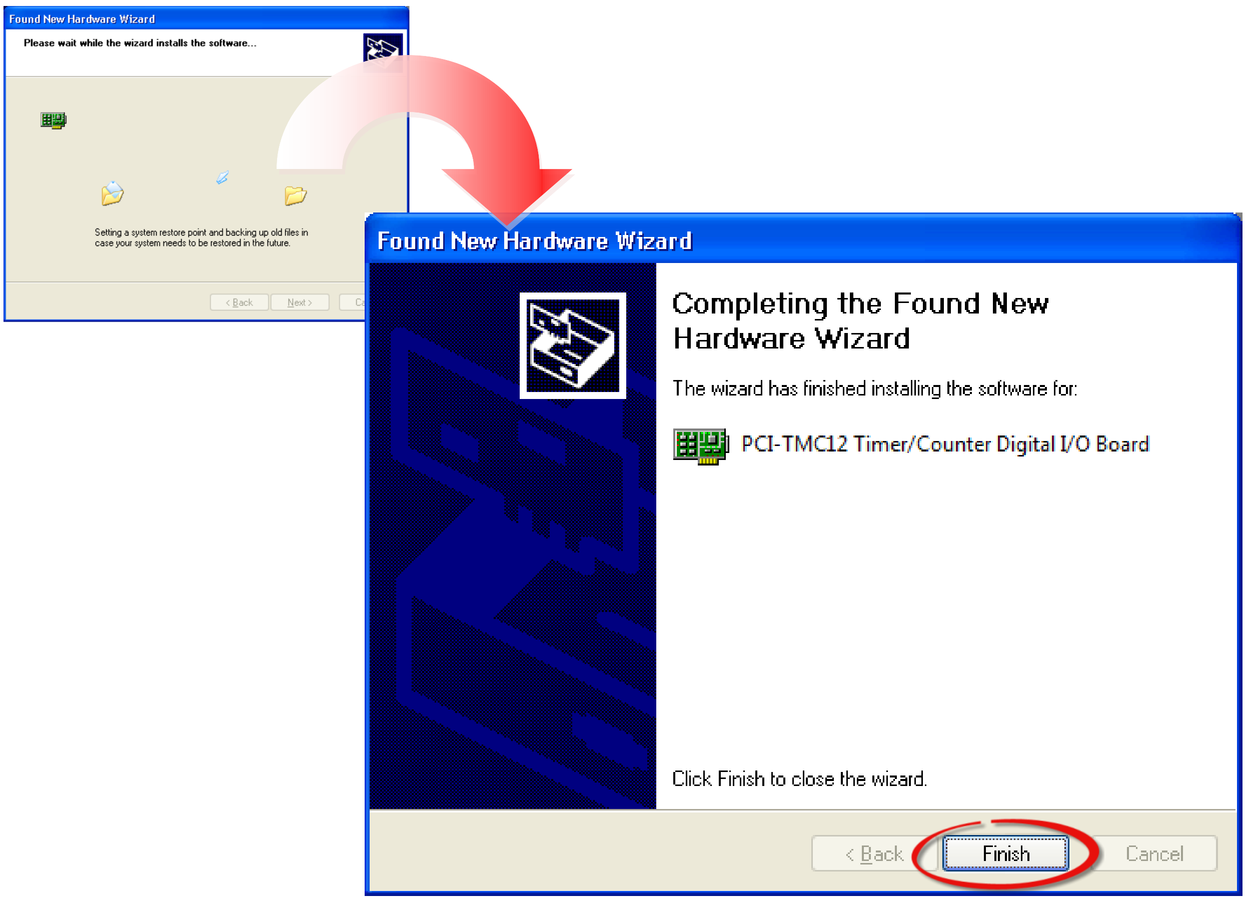
If your hardware came with an installation CD or floppy disk, insert it now.

What do you want the wizard to do?

[✓ Install the software automatically (Recommended)]
Install from a list or specific location (Advanced)

Click Next to continue.

Step 3: Click the **"Finish"** button.


Found New Hardware Wizard

Please wait while the wizard installs the software...

Setting a system restore point and backing up old files in case your system needs to be restored in the future.

Found New Hardware Wizard

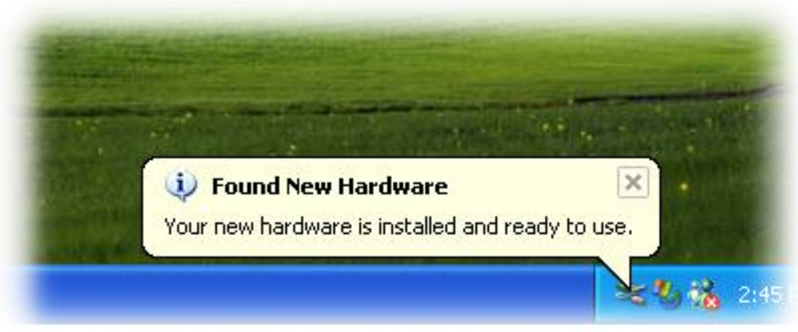
Completing the Found New Hardware Wizard

The wizard has finished installing the software for:

PCI-TMC12 Timer/Counter Digital I/O Board

Click Finish to close the wizard.

Step 4: Windows pops up **"Found New Hardware"** dialog box again.

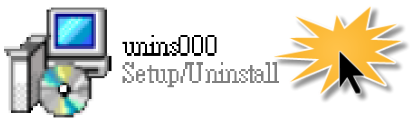

Found New Hardware

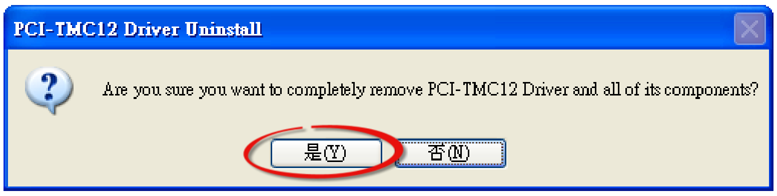
Your new hardware is installed and ready to use.

2.4 Uninstalling the PCI-TMC12 Series Classic Driver

The ICP DAS PCI-TMC12 series classic driver includes an uninstallation utility that allows you remove the software from your computer. To uninstall the software, follow the procedure described below:

Step 1: Double clock the **unins000.exe** uninstaller application, which can be found in the following folder: **C:\DAQPro\PCI-Memory**.

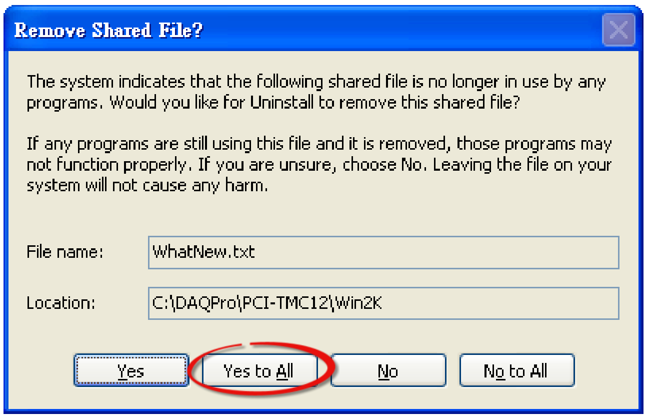



PCI-TMC12 Driver Uninstall

Are you sure you want to completely remove PCI-TMC12 Driver and all of its components?

Step 2: A dialog box will be displayed asking you to confirm that you want to remove the utility program. Click the "**Yes**" button to continue.

Step 3: The **"Remove Shared File?"** dialog box will then be displayed to confirm whether you want to remove the share files. Click the **"Yes to All"** button to continue.

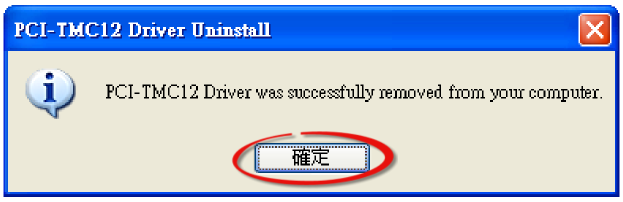

Remove Shared File?

The system indicates that the following shared file is no longer in use by any programs. Would you like for Uninstall to remove this shared file?

If any programs are still using this file and it is removed, those programs may not function properly. If you are unsure, choose No. Leaving the file on your system will not cause any harm.

File name: WhatNew.txt

Location: C:\DAQPro\PCI-TMC12\Win2K


PCI-TMC12 Driver Uninstall

PCI-TMC12 Driver was successfully removed from your computer.

Step 4: After the uninstallation process is complete, a dialog box will be displayed to you that the driver was successfully removed. Click the **"OK"** button to finish the uninstallation process.

3. DLL Function Descriptions

This list of functions is expanded on in the text that follows. However, in order to make a clear and simplified description of the functions, the attributes of the input and output parameters for every function is indicated as [input] and [output] respectively, as shown in following table. Furthermore, the error code of all functions supported by PCI-TMC12 series card is also listed in Section 3.1.

Keyword	The parameter must be set by the user **before** calling the function	The data/value returned by the parameter **after** calling the function
[Input]	**Yes**	No
[Output]	No	**Yes**
[Input, Output]	**Yes**	**Yes**

Note: The memory space required by the parameters must first be allocated by the application.

The following is an overview of the defined DLL functions:

➢ **Test Functions (Refer to Section 3.2 for more details)**

float	**PTMC12_FloatSub**(float fA, float fB);
short	**PTMC12_ShortSub**(short nA, short nB);
int	**PTMC12_IntSub**(int iA,int iB);
DWORD	**PTMC12_GetDllVersion**(void);

➢ **Driver Initialization Functions (Refer to Section 3.3 for more details)**

DWORD	**PTMC12_DriverInit**(void);
DWORD	**PTMC12_DetectBoards**(void);
DWORD	**PTMC12_OpenBoard**(DWORD dwBoardNo, DWORD dwInEnable);
DWORD	**PTMC12_ReadBoardStatus**(DWORD dwBoardNo);
DWORD	**PTMC12_ReadId**(DWORD dwBoardNo, DWORD *dwVendorId, DWORD *dwDeviceId, DWORD *dwSubVendorId, DWORD *dwSubDeviceId);
DWORD	**PTMC12_CloseBoard**(dwBoardNo);
void	**PTMC12_CloseAll**(void);

➢ Read/Write to PCI-MTC12 Functions (Refer to Section 3.4 for more details)

WORD	**PTMC12_WriteByte**(WORD dwBoardNo, DWORD dwOffse, BYTE Data);
DWORD	**PTMC12_WriteWord**(DWORD dwBoardNo, DWORD dwOffse, WORD Data);
DWORD	**PTMC12_ReadByte**(DWORD dwBoardNo, DWORD dwOffset, BYTE *Data);
DWORD	**PTMC12_ReadWord**(DWORD dwBoardNo, DWORD dwOffse, WORD *Data);

➢ Interrupt Functions (Refer to Section 3.5 for more details)

DWORD	**PTMC12_InstallCallbackFunc**(DWORD dwBoardNo, DWORD dwInitialState, void (* addrCallBackFunc)());
DWORD	**PTMC12_RemoveAllCallBackFunc**(void);
DWORD	**PTMC12_EnableInt**(DWORD dwBoardNo);
DWORD	**PTMC12_DisableInt**(DWORD dwBoardNo);
WORD	**PTMC12_WriteCounter**(DWORD dwBoardNo, BYTE Counter, BYTE Mode, DWORD Data);
WORD	**PTMC12_ReadCounter**(DWORD dwBoardNo, BYTE Counter, DWORD *Data);

➢ Read/Write to PCI Controller Functions (Refer to Section 3.6 for more details)

DWORD	**PTMC12_WritePciDWord**(DWORD dwBoardNo, WORD Data);
DWORD	**PTMC12_ReadPciDword**(DWORD dwBoardNo, WORD *Data);

3.1 Error Code Table

For the most errors, it is recommended to check:

1. Does the device driver installs successful?
2. Does the card have plugged?
3. Does the card conflicts with other device?
4. Close other applications to free the system resources.
5. Try to use another slot to plug the card.
6. Restart your system to try again.

Error Code	Error ID	Error Code	Error ID
0	PCI_NoError	16	PCI_BoardIsNotOpen
1	PCI_DriverOpenError	17	PCI_BoardOpenError
2	PCI_DriverNoOpen	18	PCI_BoardNoIsZero
3	PCI_GetDriverVersionError	19	PCI_BoardNoExceedFindBoards
4	PCI_InstallIrqError	20	PCI_InputParameterError
5	PCI_ClearIntCountError	21	PCI_IntInitialStateError
6	PCI_GetIntCountError	22	PCI_IntInitialValueError
7	PCI_RegisterApcError	23	PCI_TimeOut
8	PCI_RemoveIrqError		
9	PCI_FindBoardError		
10	PCI_ExceedBoardNumber		
11	PCI_ResetError		
12	PCI_IrqMaskError		
13	PCI_ActiveModeError		
14	PCI_GetActiveFlagError		
15	PCI_ActiveFlagEndOfQueue		

3.2 Test Functions

As part of the software installation provided for the PCI-TMC12 series card, ICPDAS also includes a range of functions that can be used to test the functionality of the board. The following sections provide a detailed description of these functions.

PTMC12_FloatSub

This function is used to perform the subtraction **fA - fB** as a float data type, and is provided for DLL linkage testing purposes.

- **Syntax:**
  float **PTMC12_FloatSub**(float **fA**, float **fB**);

- **Parameters:**
  fA
  [Input] A 4 byte floating point value

  fB
  [Input] 4 byte floating point value

- **Returns:**
  The value of fA – fB

PTMC12_ShortSub

This function is used to perform the subtraction **nA - nB** as a short data type, and is provided for DLL linkage testing purposess.

- **Syntax:**
  short **PTMC12_ShortSub**(short **nA**, short **nB**);

- **Parameters:**
  nA
  [Input] A 2 byte short data type value

  nB
  [Input] A 2 byte short data type value

- **Returns:**
  The value of nA - nB

PTMC12_IntSub

This function is used to perform the subtraction iA - iB as an int data type, and is provided for DLL linkage testing purposes.

- **Syntax:**
  Int **PTMC12_IntSub**(int **iA**, int **iB**);

- **Parameters:**
  iA
  [Input] A 4 byte int data type value

  iB
  [Input] A 4 byte int data type value

- **Returns:**
  The value of iA - iB

PTMC12_GetDllVersion

This function is used to read the version number information for the PTMC12.DLL.

- **Syntax:**
  DWORD **PTMC12_GetDllVersion**(void);

- **Parameters:**
  This function does not require any parameters

- **Returns:**
  The version umber information for the PTMC12.DLL
  For example, 102(in hexadecimal format) denotes Version 1.02

3.3 Driver Initialization Functions

PTMC12_DriverInit

This function is used to allocate resources for the Windows Driver, and must be called before using any of the DLL functions described in Sections 3.4 to 3.6.

- **Syntax:**
  DWORD **PTMC12_DriverInit**();
- **Parameters:**
  This function does not require any parameters
- **Returns:**
  PCI_NoError: OK. The command was successful.
  PCI_DriverOpenError: The Windows Driver kernel was not found.

PTMC12_DetectBoards

This subroutine will detect all installed PCI-TMC12 series boards.

- **Syntax:**
  DWORD **PTMC12_DetectBoards**();
- **Parameters:**
  This function does not require any parameters
- **Returns:**
  0: There are no PCI-TMC12 series boards is installed in this system
  1: There is only one PCI-TMC12 series board installed in this system (board number = 1)
  N: The number of PCI-TMC12 series boards installed in this system
- **Note:**
  Call **PTMC12_DriverInit()** before calling this function

PTMC12_OpenBoard

This function is used to lock the PCI-TMC12 series. Then the locked PCI-TMC12 series is dedicated to this program until PTMC12_CloseBoard is called. This function must be called first before calling these DLL functions given in Sections 3.4 to 3.6.

- **Syntax:**
  DWORD **PTMC12_OpenBoard**(DWORD **dwBoardNo**, DWORD **dwIntEnable**);

- **Parameters:**
  dwBoardNo
  [Input] The board number for the detected PCI-TMC12 series board in the range from 1 to N.

  dwIntEnable
  [Input] Enable or disable the interrupt on the PCI-TMC12 series board.

dwIntEnable	Description
0	Disabled
1	Enabled

- **Returns:**
  PCI_NoError: OK. The command was successful.
  PCI_DriverOpenError: An error occurred when attempting to initialize the kernel driver for the Board.
  PCI_BoardNoExceedFindBoards: The Board could not be found.

- **Note:**
  1. Call **PTMC12_DriverInit()** before calling this function
  2. Call **PTMC12_DetectCards()** to detect all PCI-TMC12 series board.

PTMC12_ReadBoardStatus

This function is used to show the lock-status of the PCI-TMC12 series board.

- **Syntax:**
  DWORD **PTMC12_ReadBoardStatus**(DWORD **dwBoardNo**);

- **Parameters:**
  dwBoardNo
  [Input] The board number for the detected PCI-TMC12 board in the range from 1 to N. A value of 1 indicates the first board.

- **Returns:**
  0: This PCI-TMC12 series is not locked by other program.
  1: This PCI-TMC12 series is locked by other program.

- **Note:**
  1. Call **PTMC12_DriverInit()** before calling this function
  2. Call **PTMC12_DetectBoards()** to detect all PCI-TMC12.
  3. Call **PTMC12_OpenBoard()** to lock the target PCI-TMC12 . Then the locked PCI-TMC12 is dedicated to this program.
  4. Call **PTMC12_CloseBoard()** to un-lock the target PCI-TMC12. Then other program can call PTMC12_Openboard() to lock this PCI_TMC12 boards.

PTMC12_ReadId

This function is used to show the Ids of detected PCI-TMC12 series board.

- **Syntax:**

  DWORD **PTMC12_ReadId** (DWORD **dwBoardNo**, DWORD ***dwVendorId**, DWORD ***dwDeviceId**, DWORD ***dwSubVendorId**, DWORD ***dwSubDeviceId**);

- **Parameters:**

  dwBoardNo

  [Input] The board number for the detected PCI-TMC12 board in the range from 1 to N.

  **dwVendorId*

  [Output] Vendor ID of this board.

  **dwDeviceId*

  [Output] Device ID of this board.

  **dwSubVendorId*

  [Output] Sub-Vendor ID of this board.

  **dwSubDeviceId*

  [Output] Sub-Device ID of this board.

- **Returns:**

  PCI-NoError: OK

  PCI-BoardIsNotOpen: This PCI-TMC12 series is not locked by others

  PCI-BoardNoExceedFindBoards: dwBoardNo > available board number

- **Note:**

  Call **PTMC12_DriverInit()** before calling this function

  Call **PTMC12_DetectBoards()** to detect all PCI-TMC12.

  Call **PTMC12_OpenBoard()** to lock the target PCI-TMC12 . Then the locked PCI-TMC12 is dedicated to this program.

PTMC12_CloseBoard

This function is used to unlock the PCI-TMC12 series board, then other program can use this PCI-TMC12 series now.

- **Syntax:**
  DWORD **PTMC12_CloseBoard**(DWORD **dwBoardNo**);

- **Parameters:**
  dwBoardNo
  [Input] The board number for the detected PCI-TMC12 board in the range from 1 to N.

- **Returns:**
  PCI_NoError : OK. The command was successful.
  PCI_BoardIsNotOpen: An error occurred because this board is not open.
  PCI_BoardNoExceedFindBoards: The board could not be found.

- **Note:**
  Call **PTMC12_DriverInit()** before calling this function
  Call **PTMC12_DetectBoards()** to detect all PCI-TMC12.
  Call **PTMC12_OpenBoard()** to lock the target PCI-TMC12 . Then the locked PCI-TMC12 is dedicated to this program.

PTMC12_CloseAll

This function is used to unlock all PCI-TMC12 series board installed in this PC, then other program can use these PCI-TMC12 series now.

- **Syntax:**
  DWORD **PTMC12_CloseAll**();

- **Parameters:**
  This function does not require any parameters

- **Returns:**
  PCI_NoError : OK. The command was successful.
  PCI_BoardIsNotOpen: An error occurred because this board is not open.
  PCI_BoardNoExceedFindBoards: The board could not be found.

- **Note:**
  Call **PTMC12_DriverInit()** before calling this function
  Call **PTMC12_DetectBoards()** to detect all PCI-TMC12.
  Call **PTMC12_OpenBoard()** to lock the target PCI-TMC12 . Then the locked PCI-TMC12 is dedicated to this program.

3.4 Read/Write to PCI-TMC12 Functions

PTMC12_WriteByte

Write one byte (8-bit) of data to PCI-TMC12 series.

- **Syntax:**
  WORD **PTMC12_WriteByte**(WORD **dwBoardNo**, DWORD **dwOffset**, BYTE **Data**);

- **Parameters:**
  dwBoardNo
  [Input] The board number for the detected PCI-TMC12 board in the range from 1 to N.

  dwOffset
  [Input] Offset address

  Data
  [Input] One byte of data (8-bit).

- **Returns:**
  0: Write OK. The command was successful.
  PCI_DriverNoOpen: An error occurred because the kernel driver was not found.
  PCI_BoardNoIsZero: The **dwBoardNo** value is 0. It must be in the range of 1 to N.
  PCI_BoardNoExceedFindBoards: The board could not be found because the **dwBoardNo** value is > N.

- **Note:**
  1. Call **PTMC12_DetectCards()** before calling this function
  2. Call **PTMC12_DetectBoards()** to detect all PCI-TMC12 boards
  3. Call **PTMC12_OpenBoard()** before calling this function
  4. **PTMC12_WriteByte(dwBoardNo, 0x10, Data)** → select the active 8254, refer to Section 6.3.1 of "PCI-TMC12(A) User's Hardware Manual"
  5. **PTMC12_WriteByte(dwBoardNo, 0x14, Data)** → Write to DO0 to 7, refer to Section 6.3.4 of "PCI-TMC12(A) User's Hardware Manual"

PTMC12_WriteWord

Write one word (16-bit) of data to PCI-TMC12 series.

- **Syntax:**
  DWORD **PTMC12_WriteSramWord**(DWORD **dwBoardNo**, DWORD **dwOffset**, WORD **Data**);

- **Parameters:**
  dwBoardNo
  [Input] The board number for the detected PCI-TMC12 board in the range from 1 to N.

  dwOffset
  [Input] Offset address

  Data
  [Input] One word of data (16-bit).

- **Returns:**
  0: Write OK. The command was successful.
  PCI_DriverNoOpen: An error occurred because the kernel driver was not found.
  PCI_BoardNoIsZero: The **dwBoardNo** value is 0. It must be in the range of 1 to N.
  PCI_BoardNoExceedFindBoards: The board could not be found because the **dwBoardNo** value is > N.

- **Note:**
  1. Call **PTMC12_DetectCards()** before calling this function
  2. Call **PTMC12_DetectBoards()** to detect all PCI-TMC12 boards
  3. Call **PTMC12_OpenBoard()** before calling this function
  4. **PTMC12_WriteByte(dwBoardNo, 0x10, Data)** → select the active 8254, refer to Section 6.3.1 of "PCI-TMC12(A) User's Hardware Manual"
  5. **PTMC12_WriteByte(dwBoardNo, 0x14, Data)** → Write to DO0 to 15, refer to Section 6.3.4 of "PCI-TMC12(A) User's Hardware Manual"

PTMC12_ReadByte

Read one byte (8-bit) of data from PCI-TMC12 series.

- **Syntax:**
  DWORD **PTMC12_ReadByte**(DWORD **dwBoardNo**, DWORD **dwOffset**, BYTE ***Data**);

- **Parameters:**
  dwBoardNo
  [Input] The board number for the detected PCI-TMC12 board in the range from 1 to N.

  dwOffset
  [Input] Offset address

  Data
  [Output] One byte of data (8-bit).

- **Returns:**
  0: Write OK. The command was successful.
  PCI_DriverNoOpen: An error occurred because the kernel driver was not found.
  PCI_BoardNoIsZero: The **dwBoardNo** value is 0. It must be in the range of 1 to N.
  PCI_BoardNoExceedFindBoards: The board could not be found because the **dwBoardNo** value is > N.

- **Note:**
  1. Call **PTMC12_DetectCards()** before calling this function
  2. Call **PTMC12_DetectBoards()** to detect all PCI-TMC12 boards
  3. Call **PTMC12_OpenBoard()** before calling this function
  4. **PTMC12_ReadByte(dwBoardNo, 0x14, Data)** → Read to DO0 to 7, refer to Section 6.3.3 of "PCI-TMC12(A) User's Hardware Manual"

PTMC12_ReadWord

Read one word (16-bit) of data from PCI-TMC12 series.

- **Syntax:**

  DWORD **PTMC12_ReadWord**(DWORD **dwBoardNo**, DWORD **dwOffset**, WORD ***Data**);

- **Parameters:**

  dwBoardNo

  [Input] The board number for the detected PCI-TMC12 board in the range from 1 to N.

  dwOffset

  [Input] Offset address

  Data

  [Output] One word of data (16-bit).

- **Returns:**

  0: Write OK. The command was successful.

  PCI_DriverNoOpen: An error occurred because the kernel driver was not found.

  PCI_BoardNoIsZero: The **dwBoardNo** value is 0. It must be in the range of 1 to N.

  PCI_BoardNoExceedFindBoards: The board could not be found because the **dwBoardNo** value is > N.

- **Note:**

  1. Call **PTMC12_DetectCards()** before calling this function
  2. Call **PTMC12_DetectBoards()** to detect all PCI-TMC12 boards
  3. Call **PTMC12_OpenBoard()** before calling this function
  4. **PTMC12_ReadByte(dwBoardNo, 0x14, Data)** → Read to DO0 to 14, refer to Section 6.3.3 of “PCI-TMC12(A) User’s Hardware Manual”

3.5 Interrupt Functions

PTMC12_InstallCallBackFunc

Install user's call back function to driver. So if the interrupt signal is active, driver will call this function once.

- **Syntax:**
  DWORD **PTMC12_InstallCallBackFunc**(DWORD **dwBoardNo**, DWORD **dwIntType**, void **user_function()**);

- **Parameters:**
  dwBoardNo
  [Input] The board number for the detected PCI-TMC12 board in the range from 1 to N.

  dwIntType
  [Input] 1: Initial low and active high
  2: Initial high and active low

  User_function()
  [Input] Address of user's call back function

- **Returns:**
  0: Write OK. The command was successful.
  PCI_DriverNoOpen: An error occurred because the kernel driver was not found.
  PCI_BoardNoIsZero: The **dwBoardNo** value is 0. It must be in the range of 1 to N.
  PCI_BoardNoExceedFindBoards: The board could not be found because the **dwBoardNo** value is > N.

- **Note:**
  1. Call **PTMC12_DetectCards()** before calling this function
  2. Call **PTMC12_DetectBoards()** to detect all PCI-TMC12 boards
  3. Call **PTMC12_OpenBoard()** before calling this function

PTMC12_RemoveAllCallBackFunc

Disable interrupt and call back functions of all. PCI-TMC12 series installed in this PC.

- **Syntax:**

  DWORD **PTMC12_RemoveAllCallBackFunc**();

- **Parameters:**

  This function does not require any parameters

- **Returns:**

  0: Write OK. The command was successful.

  PCI_DriverNoOpen: An error occurred because the kernel driver was not found.

  PCI_BoardNoIsZero: The **dwBoardNo** value is 0. It must be in the range of 1 to N.

  PCI_BoardNoExceedFindBoards: The board could not be found because the **dwBoardNo** value is > N.

- **Note:**

  1. Call **PTMC12_DetectCards()** before calling this function
  2. Call **PTMC12_DetectBoards()** to detect all PCI-TMC12 boards
  3. Call **PTMC12_OpenBoard()** before calling this function

PTMC12_EnableInt

Enable interrupt of PCI-TMC12 series.

- **Syntax:**
  DWORD **PTMC12_EnableInt**(DWORD **dwBoardNo**);

- **Parameters:**
  dwBoardNo
  [Input] The board number for the detected PCI-TMC12 board in the range from 1 to N.

- **Returns:**
  0: Write OK. The command was successful.
  PCI_DriverNoOpen: An error occurred because the kernel driver was not found.
  PCI_BoardNoIsZero: The **dwBoardNo** value is 0. It must be in the range of 1 to N.
  PCI_BoardNoExceedFindBoards: The board could not be found because the **dwBoardNo** value is > N.

- **Note:**
  4. Call **PTMC12_DetectCards()** before calling this function
  5. Call **PTMC12_DetectBoards()** to detect all PCI-TMC12 boards
  6. Call **PTMC12_OpenBoard()** before calling this function

PTMC12_DisableInt

Disable interrupt of PCI-TMC12 series.

- **Syntax:**
  DWORD **PTMC12_DisableInt**(DWORD **dwBoardNo**);

- **Parameters:**
  dwBoardNo
  [Input] The board number for the detected PCI-TMC12 board in the range from 1 to N.

- **Returns:**
  0: Write OK. The command was successful.
  PCI_DriverNoOpen: An error occurred because the kernel driver was not found.
  PCI_BoardNoIsZero: The **dwBoardNo** value is 0. It must be in the range of 1 to N.
  PCI_BoardNoExceedFindBoards: The board could not be found because the **dwBoardNo** value is > N.

- **Note:**
  7. Call **PTMC12_DetectCards()** before calling this function
  8. Call **PTMC12_DetectBoards()** to detect all PCI-TMC12 boards
  9. Call **PTMC12_OpenBoard()** before calling this function

PTMC12_WriteCounter

Set the counter, mode and data of the Timer/Counter.

- **Syntax:**

  DWORD **PTMC12_WriteCounter**(DWORD **dwBoardNo**, BYTE **Counter**, BYTE **Mode**, DWORD **Data**);

- **Parameters:**

  dwBoardNo
  [Input] The board number for the detected PCI-TMC12 board in the range from 1 to N.

  Counter
  [Input] Set the counter number (From 1 to 12).

  Mode
  [Input] User set mode 0~5 of the Timer/Counter

  Data
  [Input] User set counter value of the Timer/Counter channels.

- **Returns:**

  Refer to the Section 3.1 "Error Code Table"

PTMC12_ReadCounter

This function could read Timer/Counter data.

- **Syntax:**

  DWORD **PTMC12_ReadCounter**(DWORD **dwBoardNo**, BYTE **Counter**, DWORD ***Data**);

- **Parameters:**

  dwBoardNo

  [Input] The board number for the detected PCI-TMC12 board in the range from 1 to N.

  Counter

  [Input] Set the counter number (From 1 to 12).

  Data

  [Input] Read value of the Timer/Counter.

- **Returns:**

  Refer to the Section 3.1 “Error Code Table”

3.6 Read/Write to PCI Controller Functions

PTMC12_WritePciDword

Write one DWORD (32-bit) of data to PCI controller.

➢ **Syntax:**

DWORD **PTMC12_WritePciDword** (DWORD **dwBoardNo**, WORD **Data**);

➢ **Parameters:**

dwBoardNo

[Input] The board number for the detected PCI-TMC12 board in the range from 1 to N.

Data

[Input] One DWORD of data (32-bit).

➢ **Returns:**

0: Write OK. The command was successful.

PCI_DriverNoOpen: An error occurred because the kernel driver was not found.

PCI_BoardNoIsZero: The **dwBoardNo** value is 0. It must be in the range of 1 to N.

PCI_BoardNoExceedFindBoards: The board could not be found because the **dwBoardNo** value is > N.

➢ **Note:**

1. Call **PTMC12_DetectCards()** before calling this function
2. Call **PTMC12_DetectBoards()** to detect all PCI-TMC12 boards
3. Call **PTMC12_OpenBoard()** before calling this function
4. **PTMC12_WritePciDword(dwBoardNo, 0x41)** → Write to interrupt controller register, refer to Section 6.3.5 of “PCI-TMC12(A) User’s Hardware Manual”

PTMC12_ReadPciDword

Read one DWORD (32-bit) of data to PCI controller.

- **Syntax:**
  DWORD **PTMC12_ReadPciDword** (DWORD **dwBoardNo**, WORD *__Data__);

- **Parameters:**
  dwBoardNo
  [Input] The board number for the detected PCI-TMC12 board in the range from 1 to N.

  **Data*
  [Input] One DWORD of data (32-bit).

- **Returns:**
  0: Write OK. The command was successful.
  PCI_DriverNoOpen: An error occurred because the kernel driver was not found.
  PCI_BoardNoIsZero: The **dwBoardNo** value is 0. It must be in the range of 1 to N.
  PCI_BoardNoExceedFindBoards: The board could not be found because the **dwBoardNo** value is > N.

- **Note:**
  1. Call **PTMC12_DetectCards()** before calling this function
  2. Call **PTMC12_DetectBoards()** to detect all PCI-TMC12 boards
  3. Call **PTMC12_OpenBoard()** before calling this function
  4. **PTMC12_ReadPciDword(dwBoardNo, Data)** → Read the interrupt status register, refer to Section 6.3.5 of “PCI-TMC12(A) User’s Hardware Manual”

4. Demo Programs

4.1 For Microsoft Windows

During the installation process for the DLL driver, the correct kernel driver will be registered in the operating system and the DLL driver and demo programs will be copied to the correct location based on the driver software package that was selected (Win98/ME/NT/2K and 32-bit Win XP/2003/Visa/7). After installing the driver, the related demo programs, development library and declaration header files for the different development environments will be available in the folders indicated below.

The demo programs can be found in the \NAPDOS\PCI\PCI-TMC12A\DLL_OCX\Demo\ folder on the companion CD, or can be downloaded from
http://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/pci-tmc12a/dll_ocx/demo/

- BCB 4 → for Borland C++ Builder 4
  PCITMC12.H → Header files
  PCITMC12.LIB → Linkage library for BCB only

- Delphi4 → for Delphi 4
  PCI-TMC12.PAS → Declaration files

- VB6 → for Visual Basic 6
  PCITMC12.BAS → Declaration files

- VC6 → for Visual C++ 6
  PCITMC12.H → Header files
  PCITMC12.LIB → Linkage library for VC6 only

- VB.NET2005 → for VB.NET2005
  PCITMC12.vb → Visual Basic Source files

- CSharp2005 → for C#.NET2005
  PCITMC12.cs → Visual C# Source files

Note that none of the demo programs will function correctly if the DLL driver has not been properly installed.

4.2 For DOS

Programming for the Intel 8254 chip can be very complicated, so, to help users easily solve real world problems, ICP DAS provides a range of demo programs that can be used in a DOS environment, together with the source code for the library.

The relevant DOS software and demo programs can be found in the:
CD:\NAPDOS\PCI\PCI-TMC12A\DOS\ folder on the companion CD, or can be downloaded from
http://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/pci-tmc12a/dos/

An overview of the demo programs and library files for use with Turbo C 2.xx or above is provided below:

```
\TC\*.*                        → for Turbo C 2.xx or above
\TC\LARGE\*.*                  → for large model
\TC\LARGE\LIB\*.*              → for library source code
\TC\LARGE\DEMO?\*.*            → demo program source code
```

```
\TC\LARGE\LIB\PCITMC12.H       → library header file
\TC\LARGE\LIB\PCITMC12.C       → library source file
\TC\LARGE\LIB\A.BAT            → compiler file
\TC\LARGE\LIB\B.BAT            → link file
\TC\LARGE\LIB\PCITMC12.lib     → library file
```

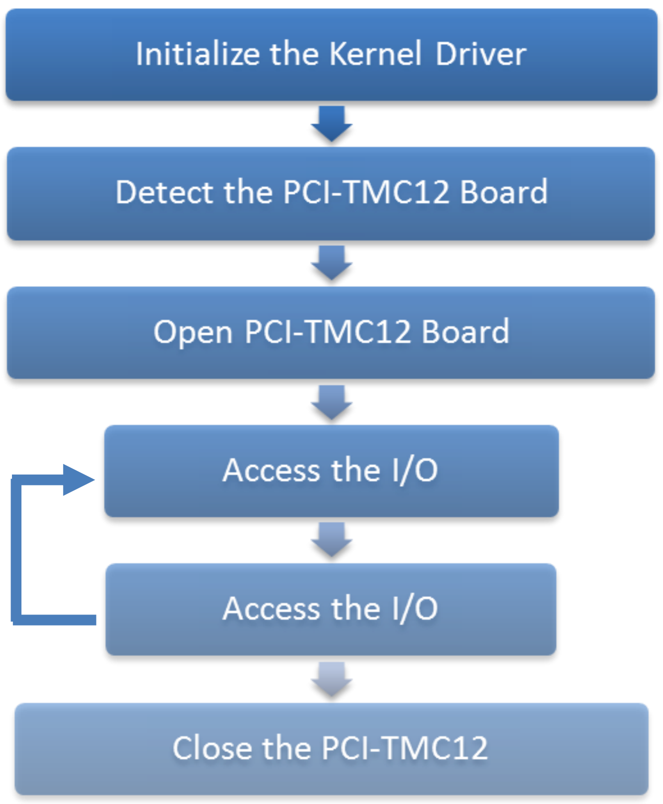
```
\TC\LARGE\DEMO1\PCITMC12.H   → library header file
\TC\LARGE\DEMO1\DEMO1.C      → demo1 source file
\TC\LARGE\DEMO1\DEMO1.PRJ    → TC project file
\TC\LARGE\DEMO1\IOPORTL.LIB → I/O port library file
\TC\LARGE\DEMO1\PCITMC12.LIB→ library file
\TC\LARGE\DEMO1\DEMO1.EXE    → demo1 execution file
```

5. Program Architecture

In general, the first DLL called must be PTMC12_DriverInit(), it will initiate the kernel mode driver. The second DLL called must be PTMC12_DetectBoards(), it will find all PCI-TMC12 series in this PC.

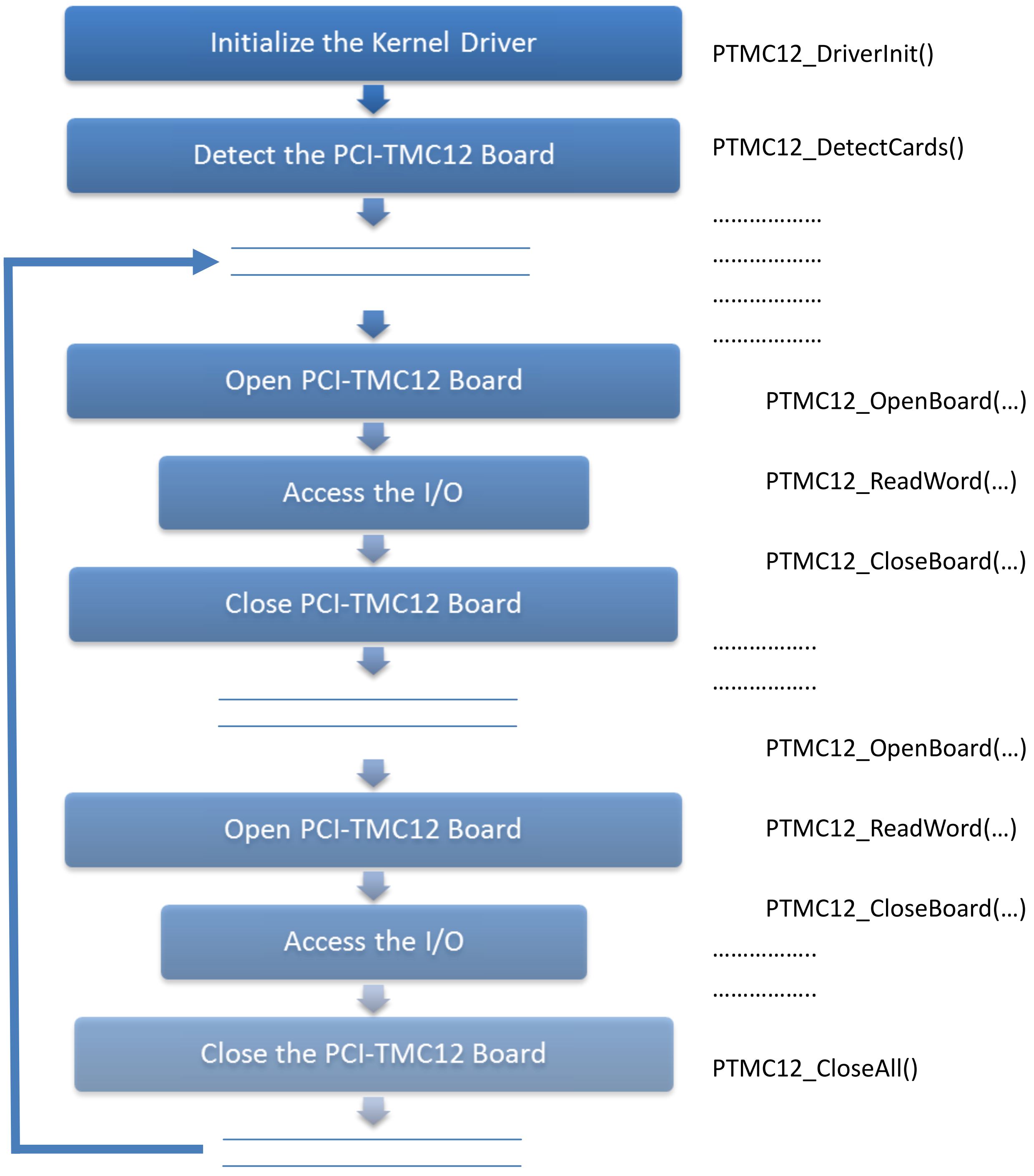
The PCI_OpenBoard(...) will open and Lock the target PCI-TMC12 series until PCI_Close Board(...) or PCI_CloseAll() is called.

For single-task applications, only one user's program control PCI-TMC12 series. So the program will open and lock PCI-TMC12 series in the program is start and un-lock PCI-TMC12 series in the program is exit as follows:



```
PTMC12_DriverInit()

PTMC12_DetectCards()

PTMC12_OpenBoard(…)
………………

     PTMC12_ReadWord(…)
     ………………
     ………………
     PTMC12_WriteWord(…)
………………

PTMC12_CloseAll()
```

For multi-task applications, many user's programs will control PCI-TMC12 series. So the program will open and lock PCI-TMC12 series before access the I/O. Then un-lock PCI-TMC12 series after access the I/O as follows:



Problems Report

Technical support is available at no charge as described below. The best way to report problems is to send electronic mail to Service@icpdas.com or Service.icpdas@ gmail.com on the Internet.

When reporting problems, please include the following information:

1. Is the problem reproducible? If so, how?
2. What kind and version of **platform** that you using? For example, Windows 98, Windows 2000 or 32-bit Windows XP/2003/Vista/7/8.
3. What kinds of our **products** that you using? Please see the product's manual.
4. If a dialog box with an **error message** was displayed, please include the full test of the dialog box, including the text in the title bar.
5. If the problem involves **other programs** or **hardware devices**, what devices or version of the failing programs that you using?
6. **Other comments** relative to this problem or **any suggestions** will be welcomed.

After we had received your comments, we will take about two business days to test the problems that you said. And then reply as soon as possible to you. Please check that if we had received you comments? And please keeps contact with us.