User's Guide

omega.com®
ΩOMEGA®

Shop online at

www.omega.com
e-mail: info@omega.com

ISO 9001 CERTIFIED CORPORATE QUALITY	ISO 9002 CERTIFIED CORPORATE QUALITY
STAMFORD, CT	MANCHESTER, UK

OME-TMC12
PCI-Bus Digital I/O Board

Software Manual

omega.com®
ΩOMEGA®

OMEGAnet® Online Service **www.omega.com**	**Internet e-mail** **info@omega.com**

Servicing North America:

USA:
ISO 9001 Certified

One Omega Drive, P.O. Box 4047
Stamford CT 06907-0047
TEL: (203) 359-1660 FAX: (203) 359-7700
e-mail: info@omega.com

Canada:

976 Bergar
Laval (Quebec) H7L 5A1, Canada
TEL: (514) 856-6928 FAX: (514) 856-6886
e-mail: info@omega.ca

For immediate technical or application assistance:

USA and Canada:

Sales Service: 1-800-826-6342 / 1-800-TC-OMEGA®
Customer Service: 1-800-622-2378 / 1-800-622-BEST®
Engineering Service: 1-800-872-9436 / 1-800-USA-WHEN®
TELEX: 996404 EASYLINK: 62968934 CABLE: OMEGA

Mexico:

En Español: (001) 203-359-7803 e-mail: espanol@omega.com
FAX: (001) 203-359-7807 info@omega.com.mx

Servicing Europe:

Benelux:

Postbus 8034, 1180 LA Amstelveen, The Netherlands
TEL: +31 (0)20 3472121 FAX: +31 (0)20 6434643
Toll Free in Benelux: 0800 0993344
e-mail: sales@omegaeng.nl

Czech Republic:

Frystatska 184, 733 01 Karviná, Czech Republic
TEL: +420 (0)59 6311899 FAX: +420 (0)59 6311114
Toll Free: 0800-1-66342 e-mail: info@omegashop.cz

France:

11, rue Jacques Cartier, 78280 Guyancourt, France
TEL: +33 (0)1 61 37 29 00 FAX: +33 (0)1 30 57 54 27
Toll Free in France: 0800 466 342
e-mail: sales@omega.fr

Germany/Austria:

Daimlerstrasse 26, D-75392 Deckenpfronn, Germany
TEL: +49 (0)7056 9398-0 FAX: +49 (0)7056 9398-29
Toll Free in Germany: 0800 639 7678
e-mail: info@omega.de

United Kingdom:
ISO 9002 Certified

One Omega Drive, River Bend Technology Centre
Northbank, Irlam, Manchester
M44 5BD United Kingdom
TEL: +44 (0)161 777 6611 FAX: +44 (0)161 777 6622
Toll Free in United Kingdom: 0800-488-488
e-mail: sales@omega.co.uk

It is the policy of OMEGA to comply with all worldwide safety and EMC/EMI regulations that apply. OMEGA is constantly pursuing certification of its products to the European New Approach Directives. OMEGA will add the CE mark to every appropriate device upon certification.

The information contained in this document is believed to be correct, but OMEGA Engineering, Inc. accepts no liability for any errors it contains, and reserves the right to alter specifications without notice.
WARNING: These products are not designed for use in, and should not be used for, patient-connected applications.

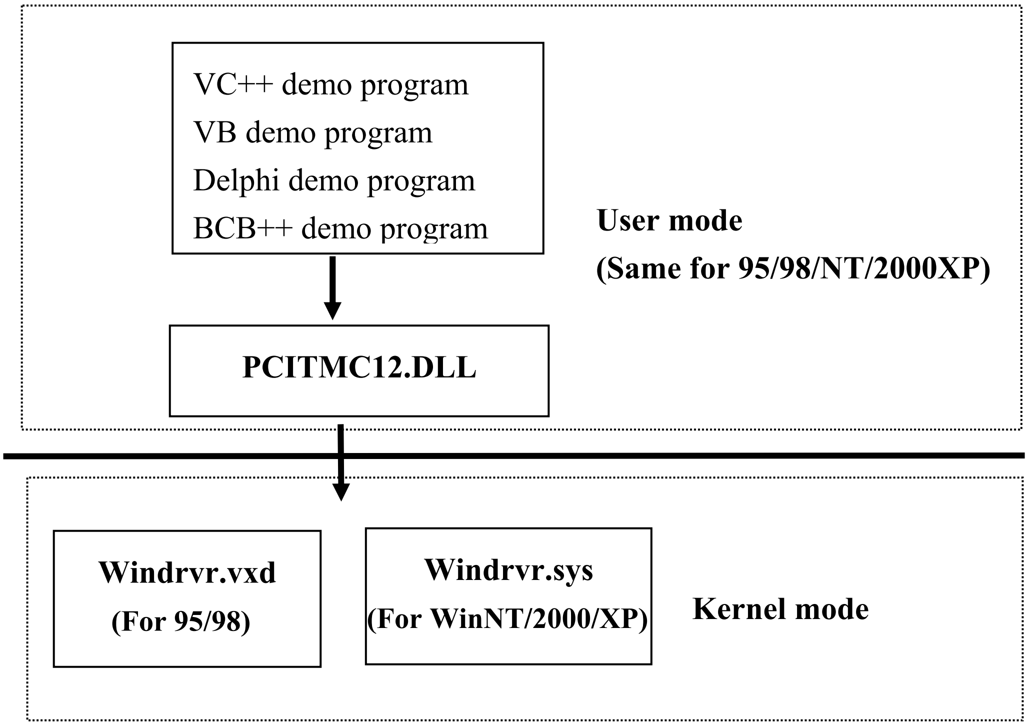
OME-PCI-TMC12(A)

Software User's Manual

Tables of Contents

1. Introduction

The demo programs, written in VC++, VB, Delphi, and BCB++, are provided in the companion CD. These demo programs will call the DLL, PCITMC12.DLL, to access the hardware of OME-PCI-TMC12. The PCITMC12.DLL will call the kernel driver, Windrvr.vxd or Windrvr.sys (For Win2000/XP as follows:



The install shield will install kernel driver, DLL driver & application demo program to system. **All demo program & DLL are same for 95/98/NT/2000/XP.** But the kernel drivers are different for different systems as follows:

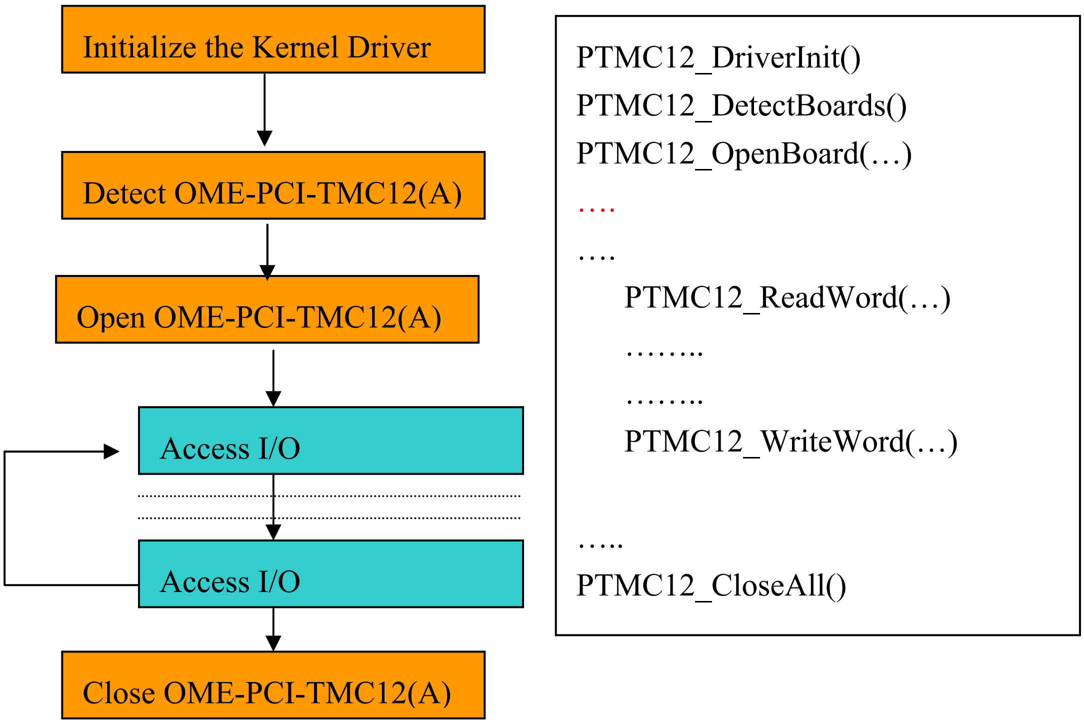
- for windows 95/98 → will copy ***WINDRVR.VXD*** to C:\WIN95\SYSTEM\VMM32
- for windows NT/2000 → will copy ***WINDRVR.SYS*** to C:\WINNT\SYSTEM32\DRIVERS
- for windows XP → will copy ***WINDRVR.SYS*** to C:\WINDOWS\SYSTEM32\DRIVERS

(Note: 2000 & XP use the same driver)

The DLLs & demo programs will not work if the kernel driver is not installed correctly. The install shield will copy the correct kernel driver to correct position if you select correct O.S.(95/98, NT, 2000, XP).

The install shield also copy all related documentations of OME-PCI-TMC12(A) to user's hard disk. Refer to **CallDll.pdf** for more information about how to call the DLL functions with VC++5, VB5, Delphi3 and Borland C++ Builder 3. Refer to **step 5 of Sec. 1.1.2** for more information.

The software architecture is given as follows:



```
PTMC12_DriverInit()
PTMC12_DetectBoards()
PTMC12_OpenBoard(…)
….
….
    PTMC12_ReadWord(…)
    ……..
    ……..
    PTMC12_WriteWord(…)

…..
PTMC12_CloseAll()
```

Note:

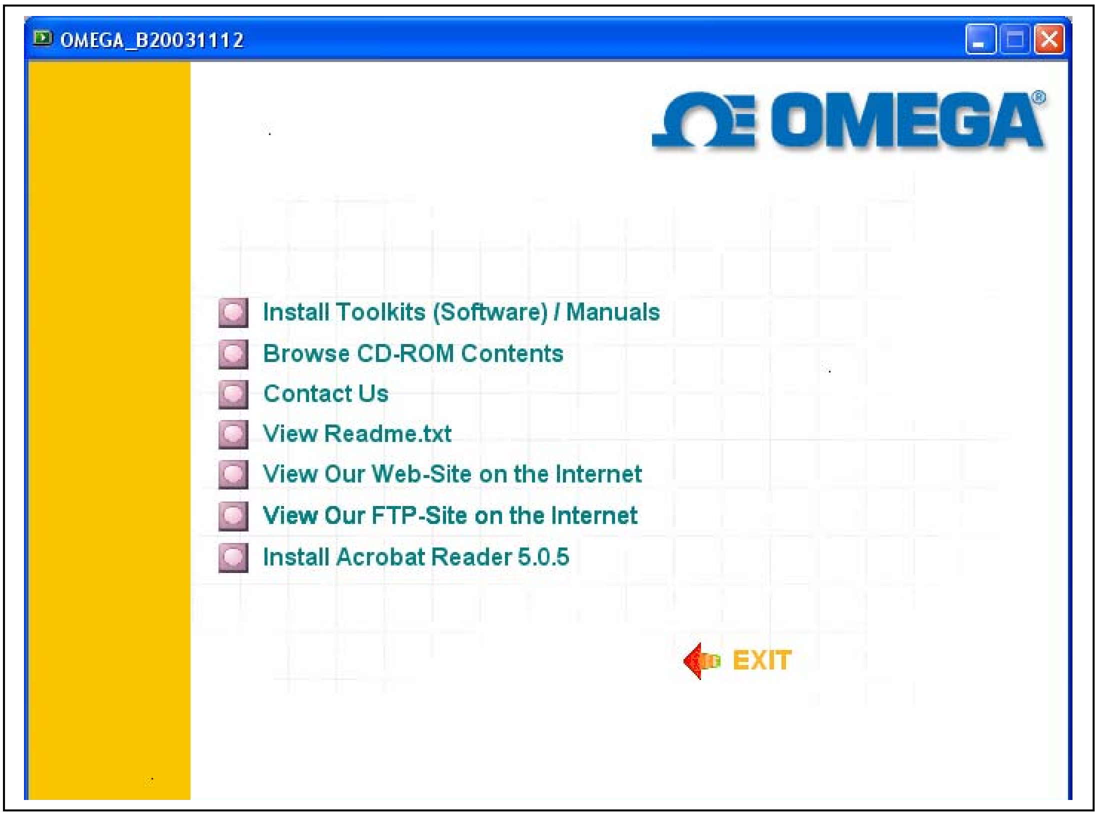
1. OME-PCI-TMC12 (A) maybe OME-PCI-TMC12 or OME-PCI-TMC12A. Refer to Sec. 3.4 of "OME-PCI-TMC12 (A) User's Manual" (not this manual) for comparison of TMC12 & TMC12A.
2. PTMC12.DLL is designed for OME-PCI-TMC12.
3. If J28 of OME-PCI-TMC12A is set to OME-PCI-**TMC12**, it can use PTMC12.DLL as same as OME-PCI-TMC12.

1.1 Installation Quick Start

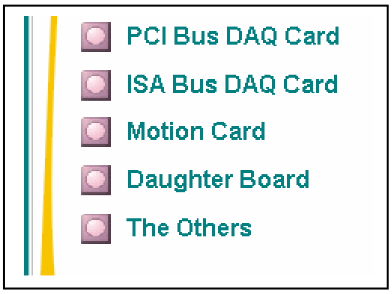
The OME-PCI-TMC12 can be used in Windows 95/98/NT/2000/XP. The recommended installation steps are given in Sec 1.1.1 ~ Sec. 1.1.4

1.1.1 Step 1: Software Installation

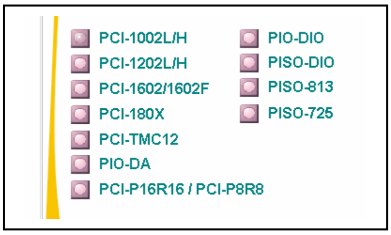
Step 1: insert the companion CD in the CD-ROM drive. Following screen should appear:


OMEGA_B20031112
OMEGA
Install Toolkits (Software) / Manuals
Browse CD-ROM Contents
Contact Us
View Readme.txt
View Our Web-Site on the Internet
View Our FTP-Site on the Internet
Install Acrobat Reader 5.0.5
EXIT

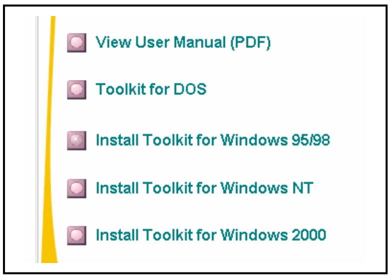
Step 2: click the first item, **Toolkits (Software)/Manuals**



Step 3: click the first item, **PCI Bus DAQ Card**



Step 4: click the last item, **OME-PCI-TMC12**


View User Manual (PDF)
Toolkit for DOS
Install Toolkit for Windows 95/98
Install Toolkit for Windows NT
Install Toolkit for Windows 2000

Step 5: click the appropriate item, example - **Install Toolkit for Windows NT**

Then the install shield will install kernel driver, DLL driver & application demo program to system. **All demo programs & DLLs are same for 95/98/NT/2000/XP.** But the kernel driver is different for different systems as follows:

- for windows 95/98 → will copy ***WINDRVR.VXD*** to C:\WIN95\SYSTEM\VMM32
- for windows NT/2000 → will copy ***WINDRVR.SYS*** to C:\WINNT\SYSTEM32\DRIVERS
- for windows XP → will copy ***WINDRVR.SYS*** to C:\WINDOWS\SYSTEM32\DRIVERS

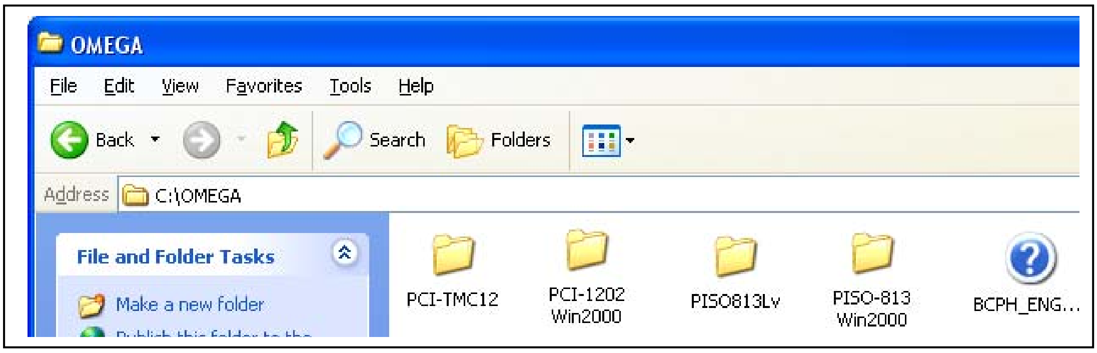
(Note: 2000 & XP use the same driver)

The DLLs & demo programs will not work if the kernel driver is not installed correctly. The install shield will copy the correct kernel driver to correct position if you select correct O.S. (95/98, NT, 2000, XP).

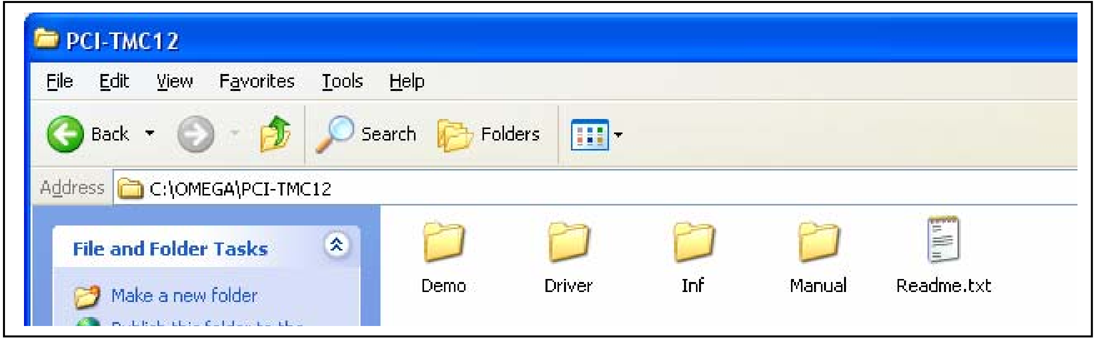
1.1.2 Step 2: Check the installed software

Assume you use the default directory & name of install shields, the software will be installed in C:\DAQPRO as follows:

Step 1: click **C:\OMEGA** as follows:



Step 2: click **PCI-TMC12** as follows:



Demo → demo program

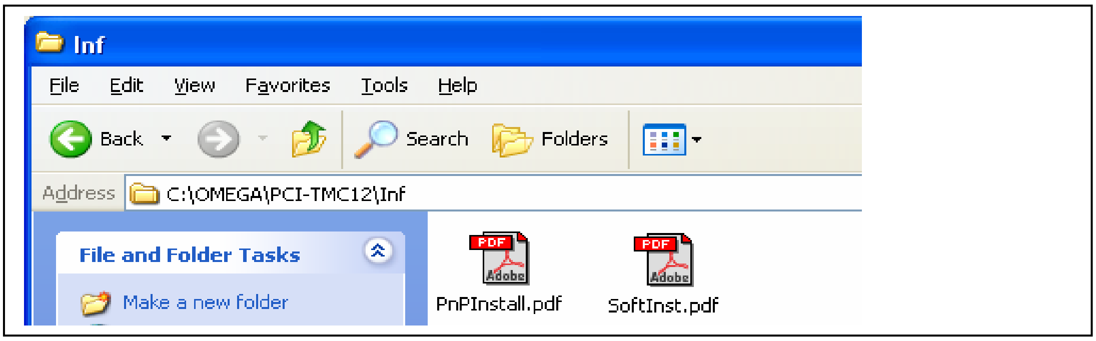
Driver → DLL

Inf → for plug & play → windows 98/2000/XP need this information

Manual → user's manual & other literature, refer to chapter 3 for more information

ReadMe → Read this file for more information

Step 3: click **Inf** as follows:



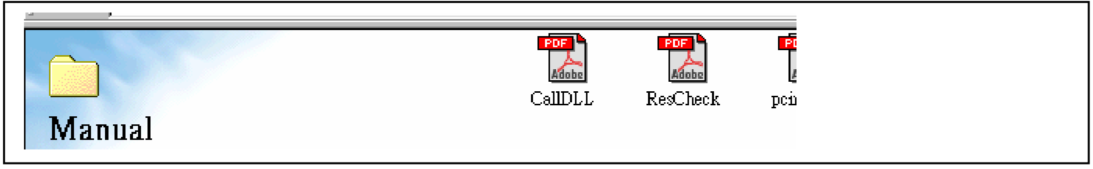
OME-PCI-TMC12.inf → inf file for windows 98/2000/XP.

PnPInstall → read this file for plug & play installation steps.

Note: 95 & NT 4.0 do not need this file.

Step 4: Click **PnPInstall** to read this PDF file for plug & play installation. For Windows 98/2000/XP, system will auto detect the OME-PCI-TMC12 & ask you to install software driver. **Read this PDF file carefully if you are not familiar with plug & play installation. You will need it in next steps.**

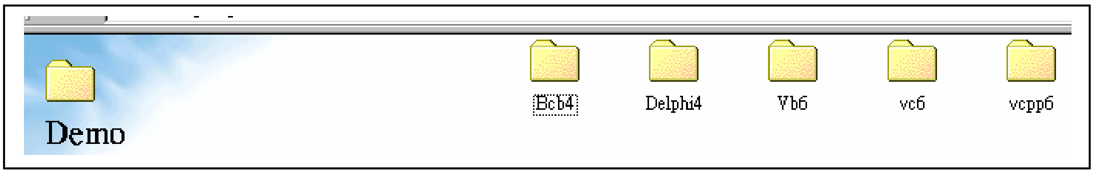
Step 5: Back to the previous directory & click **Manual** as follows:



CallDll.pdf → how to call DLL by VC, VB, Delphi & BCB++

ResCheck → how to check the allocated system resource of OME-PCI-TMC12

Step 6: Back the previous directory & click **Demo** as follows:



Bcb4 → demo program for BCB 4.0

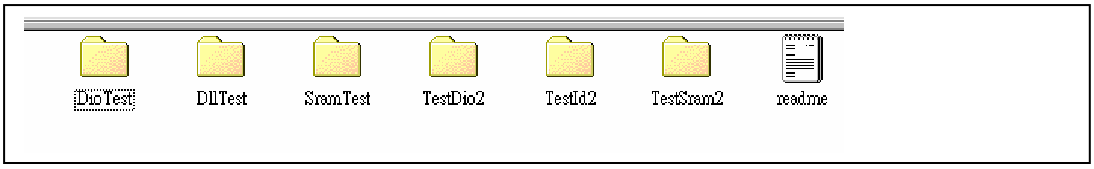
Delphi4 → demo program for Delphi 4.0

VB6 → demo program for VB 6.0

VC6 → demo program for VC 6.0, C language

VCPP → demo program for VC 6.0, C++ language

Step 7: click **VB6** as follows:



DioTest → Test D/I/O of OME-PCI-TMC12

DllTest → test DLL driver & detect OME-PCI-TMC12

TestDio2 → Write D/O then read D/I of two OME-PCI-TMC12 cards

TestId2 → Show IDs of two OME-PCI-TMC12 cards

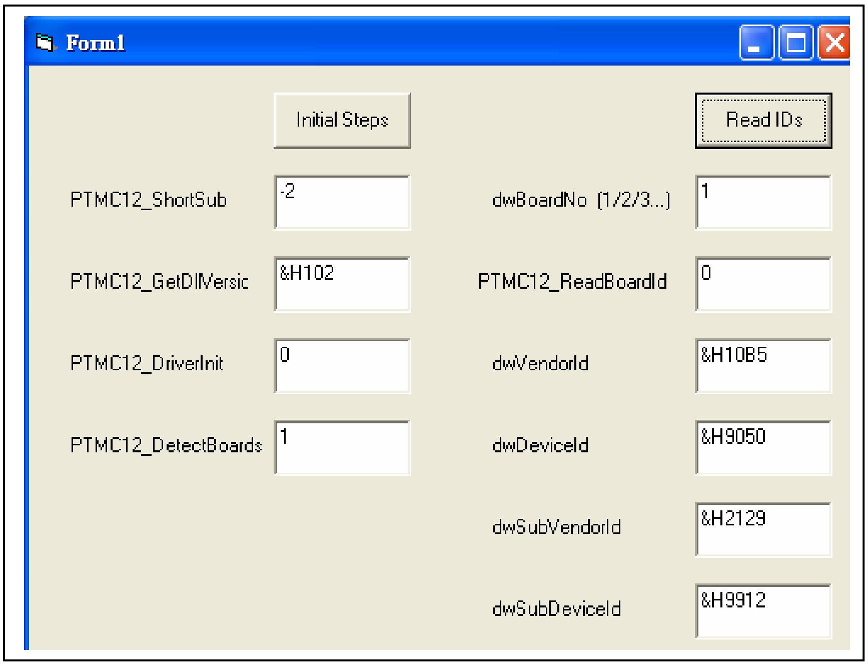
Step 8: **shutdown & power off your PC**

1.1.3 Step 3: Hardware Installation

Step 1: Install your OME-PCI-TMC12(A) in the PC
Step 2: Power on your PC
Step 3: Now 98/2000/XP will find a OME-PCI-TMC12 card & ask user to provide Plug & Play driver. Refer to **PnPInstall.pdf**, step3 of Sec. 1.1.2, for more information.

1.1.4 Step 4: Hardware Diagnostic

Step 1: run **DllTest** of VB demo program as follows: (**step7 of Sec. 1.1.2**)



PTMC12_ShortSub	-2	dwBoardNo (1/2/3...)	1
PTMC12_GetDllVersic	&H102	PTMC12_ReadBoardId	0
PTMC12_DriverInit	0	dwVendorId	&H10B5
PTMC12_DetectBoards	1	dwDeviceId	&H9050
		dwSubVendorId	&H2129
		dwSubDeviceId	&H9912

- Click ***Initial Steps*** first to check the kernel driver, DLL & PCITMC12-DetectCards()
- Check the value of ***PCITMC12_DriverInit*** must be 0
- Click ***ReadId*** to show the IDs of selected OME-PCI-TMC12 in this PC

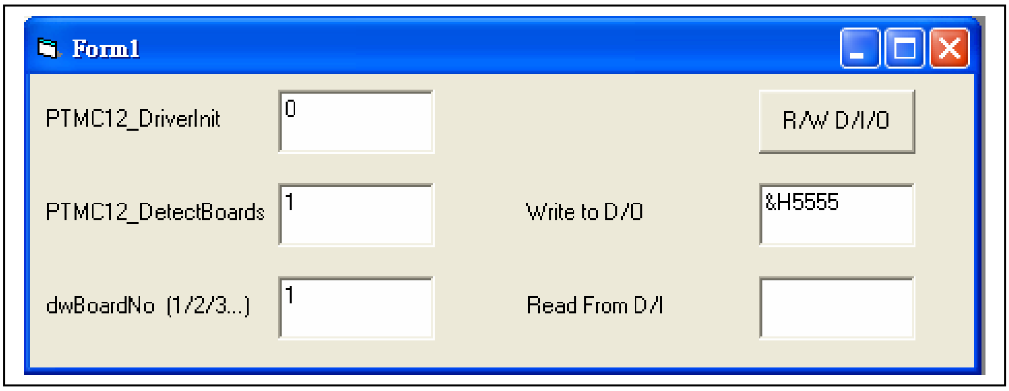
The IDs of OME-PCI-TMC12(A) cards are as follows:

- **Vendor ID = 10B5**
- **Device ID = 9050**
- **Sub-vendor ID = 2129**
- **Sub-device ID = 9912**

Note: The TMC12 & TMC12A have identical IDs.

Step 2:

- Install a 20-pin flat cable between CON2 & Con3 of OME-PCI-TMC12(A)
- run **DioTest** of VB demo program as follows: (**step7 of Sec. 1.1.2**)



PTMC12_DriverInit 0

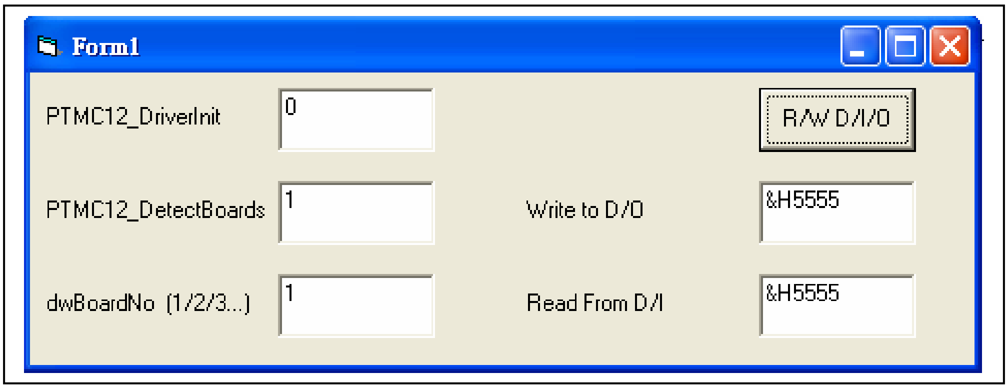
PTMC12_DetectBoards 1

dwBoardNo (1/2/3...) 1

Write to D/O &H5555

Read From D/I

- Click ***R/W D/I/O*** to write to D/O & Read D/I as follows: (write-data is given in ***Write to D/O***)



PTMC12_DriverInit 0

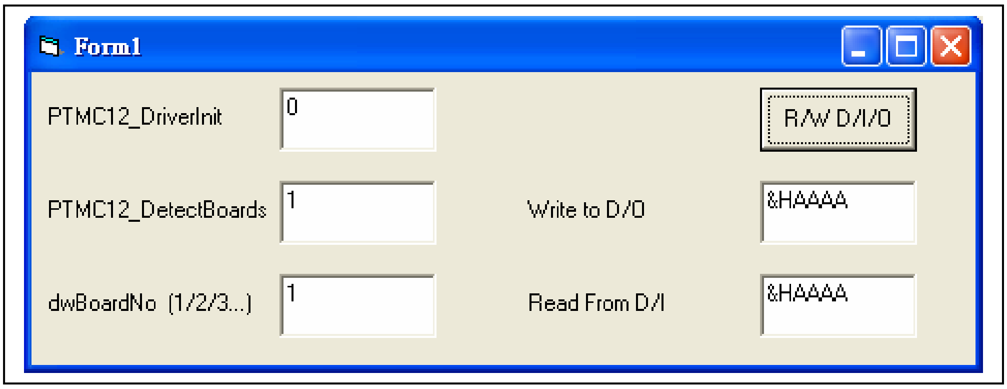
PTMC12_DetectBoards 1

dwBoardNo (1/2/3...) 1

Write to D/O &H5555

Read From D/I &H5555

- Change &H5555 to &HAAAA & Click ***R/W D/I/O*** as follows:



PTMC12_DriverInit 0

PTMC12_DetectBoards 1

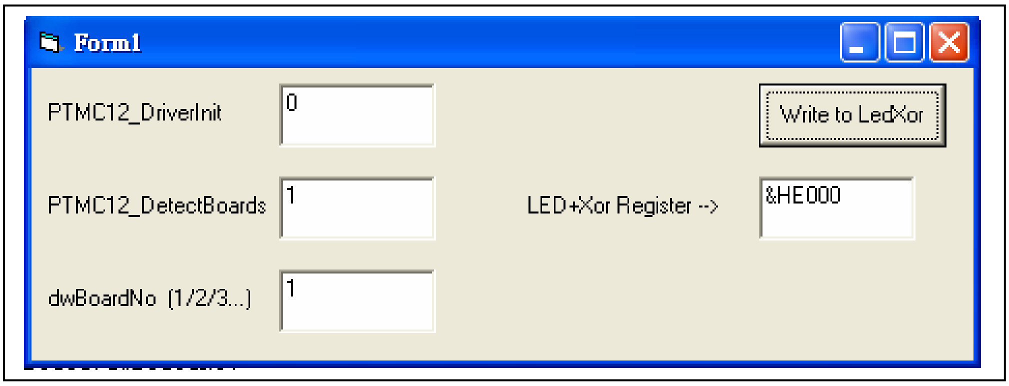
dwBoardNo (1/2/3...) 1

Write to D/O &HAAAA

Read From D/I &HAAAA

Note: if the 20-pin flat cable is not installed, the D/I will be &HFFFF

Step 3: run **LedTest** of VB demo program as follows: (**step7 of Sec. 1.1.2**)



PTMC12_DriverInit 0

PTMC12_DetectBoards 1

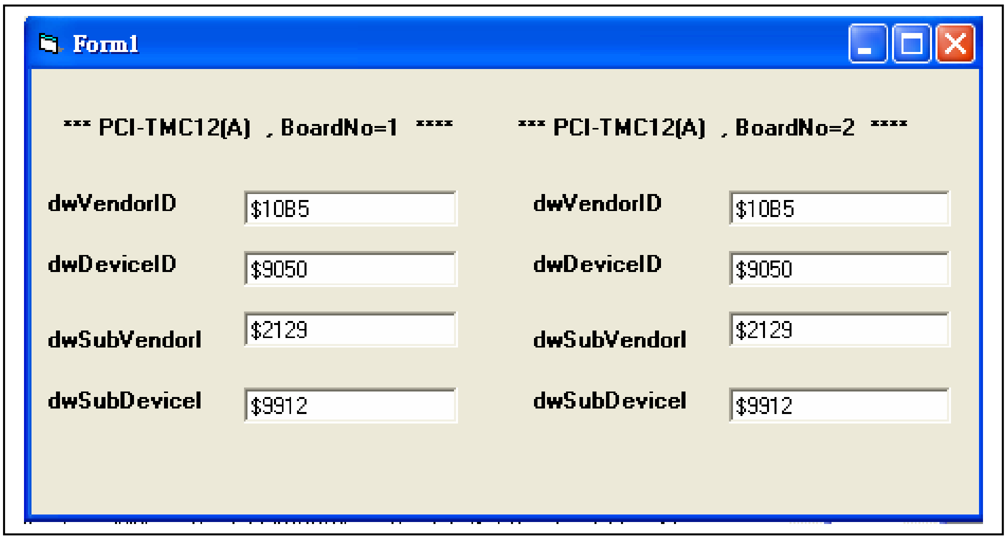
dwBoardNo (1/2/3...) 1

LED+Xor Register --> &HE000

- Click ***Write to LedXor*** & check LED1=OFF, LED2=OFF & LED3= OFF
- Change &HE000 to &H6000
- Click ***Write to LedXor*** & check LED1=OFF, LED2=OFF & LED3= ON
- Change &H6000 to &HA000
- Click ***Write to LedXor*** & check LED1=OFF, LED2=ON & LED3= OFF
- Change &H6000 to &HA000
- Click ***Write to LedXor*** & check LED1=ON, LED2=OFF & LED3= OFF
- Change &HA000 to &HE000
- Click ***Write to LedXor*** & check LED1=OFF, LED2=OFF & LED3= OFF

1.1.5 Step 5: Muti-Board Diagnostic

Step 1: run **TestId2** of VB demo program to read & show IDs of two OME-PCI-TMC12(A) as follows: (**step7 of Sec. 1.1.2**)



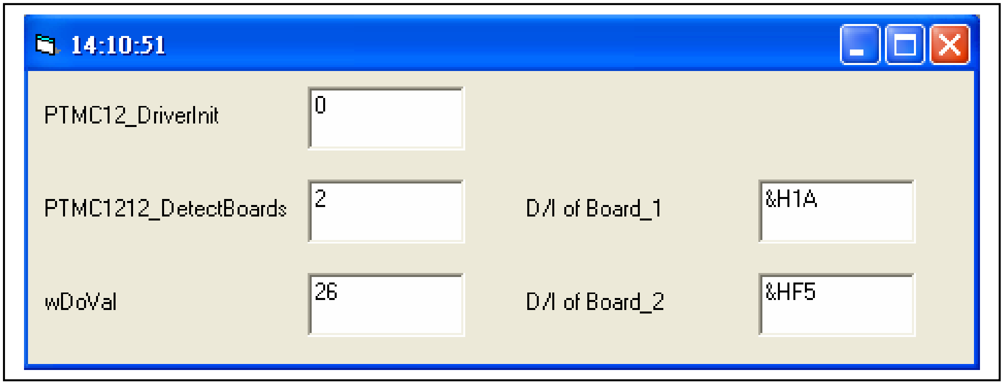
*** PCI-TMC12(A) , BoardNo=1 ****

dwVendorID $10B5
dwDeviceID $9050
dwSubVendorI $2129
dwSubDeviceI $9912

*** PCI-TMC12(A) , BoardNo=2 ****

dwVendorID $10B5
dwDeviceID $9050
dwSubVendorI $2129
dwSubDeviceI $9912

Step 2: run **TestDio2** of VB demo program to read/write D/I/O of two OME-PCI-TMC12(A) as follows: (**step7 of Sec. 1.1.2**)



PTMC12_DriverInit 0
PTMC1212_DetectBoards 2
wDoVal 26
D/I of Board_1 &H1A
D/I of Board_2 &HF5

2. DLL Driver

The included software is a collection of subroutines for OME-PCI-TMC12(A) cards for Windows 95/98/NT/2000/XP applications. These subroutines are written with C language and perform a variety of digital I/O operations.

The names of the subroutines in PCITMC12.DLL have logical names to simplify its usage. It provides powerful, easy-to-use subroutines for developing your data acquisition application. Your program can call these DLL functions by VC++, VB, Delphi, and BORLAND C++ Builder easily. To speed-up your developing process, some demonstration source program are provided.

Please refer to the following user manuals: (refer to step2 of Sec. 1.1.2 for more information)

- **PnPInstall.pdf: (Step 3 of Sec. 1.1.2)**

  Install the PnP (Plug and Play) driver for PCI card under Windows 95/98.

- **SoftInst.pdf: (Step 3 of Sec. 1.1.2)**

  Install the software package under Windows 95/98/NT/2000/XP.

- **CallDll.pdf: (Step 5 of Sec. 1.1.2)**

  Call the DLL functions with VC++5, VB5, Delphi3 and Borland C++ Builder 3.

- **ResCheck.pdf: (Step 5 of Sec. 1.1.2)**

  Check the resources I/O Port address, IRQ number and DMA number for add-on cards under Windows 95/98/NT.

The install shield will install kernel driver, DLL driver & application demo programs. **All DLL driver & demo programs are same for all windows operating systems** (95/98/NT/2000/XP). But the kernel drivers are different for different system as follows:

- for windows 95/98 → will copy ***WINDRVR.VXD*** to C:\WIN95\SYSTEM\VMM32
- for windows NT/2000 → will copy ***WINDRVR.SYS*** to C:\WINNT\SYSTEM32\DRIVERS
- for windows XP → will copy ***WINDRVR.SYS*** to C:\WINDOWS\SYSTEM32\DRIVERS

The DLL & demo programs will not work if the kernel driver is not installed correctly. The install shield will copy the correct kernel driver to correct position if you select correct O.S.(95/98, NT, 2000, XP).

After the software driver is installed, the related header file & declaration files are given as follows:

```
|--\Demo                  ← demo program
   |--\BCB4               ← for Borland C++ Builder 4
   |   |--\PCITMC12.H     ← Header file
   |   +--\PCITMC12.LIB   ← Linkage library for BCB4 only
   |
   |--\Delphi4            ← for Delphi 4
   |   +--\PCITMC12.PAS   ← Declaration file
   |
   |--\VB6                ← for Visual Basic 6
   |   +--\PCITMC12.BAS   ← Declaration file
   |
   +--\VCPP6              ← for Visual C++ 6
       |--\PCITMC12.H     ← Header file
       +--\PCITMC12.LIB   ← Linkage library for VC++6 only
```

In this chapter, we use some keywords to indicate the attribute of Parameters.

Keyword	Is Parameter Set by user before calling this function ?	Does user Get Data/Value from this parameter after calling this function ?
[Input]	Yes	No
[Output]	No	Yes
[Input, Output]	Yes	Yes

Note: Spaces need to be allocated for all of the parameters by the user.

The return codes of DLLs are defined as follows:

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

The defined DLL are given as follows:

Functions of test, Refer to Sec. 2.2

- 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);

Functions of Driver Initialization, Refer to Sec. 2.3

- 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);

Functions of Read/Write to OME-PCI-TMC12(A), Refer to Sec. 2.4

- DWORD PTMC12_Writeyte(DWORD dwBoardNo, DWORD dwOffset, BYTE Data);
- DWORD PTMC12_WriteWord(DWORD dwBoardNo, DWORD dwOffset, WORD Data);
- DWORD PTMC12_ReadByte(DWORD dwBoardNo, DWORD dwOffset, BYTE *Data);
- DWORD PTMC12_ReadWord(DWORD dwBoardNo, DWORD dwOffset, WORD *Data);

Functions of Interrupt, Refer to Sec. 2.5

- DWORD PTMC12_InstallCallBackFunct(DWORD dwBoardNo, DWORD dwInitialState, void(* addrCallBackFunc)());
- DWORD PTMC12_RemoveAllCallBackFunc(void);
- DWORD PTMC12_EnableInt(DWORD dwBoardNo);
- DWORD PTMC12_DisableInt(DWORD dwBoardNo);

Functions of Read/Write to PCI Controller, Refer to Sec. 2.6

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

2.1 Find the Board Number

The plug & play BIOS will assign the proper base address to OME-PCI-TMC12(A). If there is only one OME-PCI-TMC12(A), user can identify this board as board_1. If there are two OME-PCI-TMC12(A) boards in the system, users will have to identify which board is board_1. Our software driver can support 32 boards max. Therefore user can install 32 boards of OME-PCI-TMC12(A) in one PC system.

The simplest way to find the board number is to use DioTest in VB demo program. This demo program will send to D/O & Read from D/I. If user installs one 20-pin flat-cable between CON2 & CON3, the D/I read back will be as same as D/O: So user can find the board number as follows:

1.Install all OME-PCI-TMC12(A) cards into this PC system

2. Install one 20-pin flat cable between CON2 & CON3 of one OME-PCI-TMC12(A)

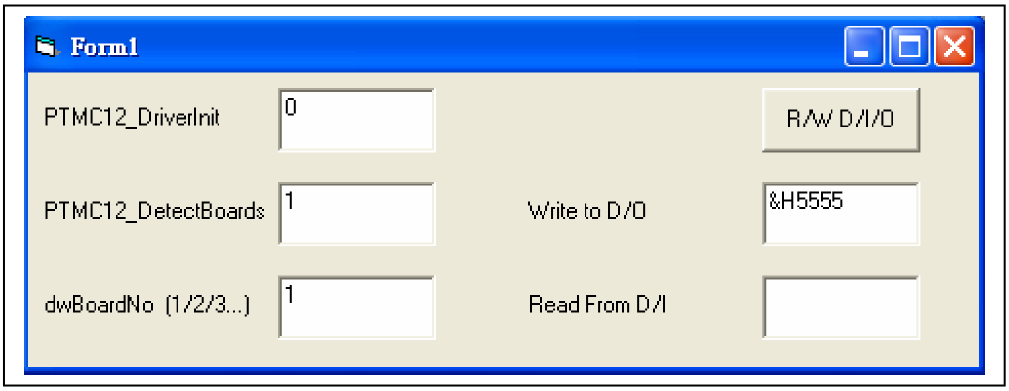
3. Power on the PC

4. Run **DioTest of VB demo program(Sec. 1.1.2)**

5. Key-in ***board number*** to 1

6. Click ***R/W D/I/O***

7. Check the value in ***Read From D/I***, if it is equal to ***Write to D/O,*** the board number of the target OME-PCI-TMC12(A) is 1. Otherwise you can go to step 5 for next **board number**.



PTMC12_DriverInit 0

PTMC12_DetectBoards 1

dwBoardNo (1/2/3...) 1

Write to D/O &H5555

Read From D/I

Note: If only one OME-PCI-TMC12(A) exists, the board number will be 1.

2.2 Functions of Test

2.2.1 PTMC12_FloatSub

- **Description:**

  To perform the subtraction as fA - fB in float data type. This function is provided for testing DLL linkage.

- **Syntax:**

  float PTMC12_FloatSub(float fA, float fB)

- **Parameter:**

  fA : [Input] 4 bytes floating point value

  fB : [Input] 4 bytes floating point value

- **Return:**

  The value of fA - fB

2.2.2 PTMC12_ShortSub

- **Description:**

  To perform the subtraction as nA - nB in short data type. This function is provided for testing DLL linkage.

- **Syntax:**

  short PTMC12_ShortSub(short nA, short nB)

- **Parameter:**

  nA :[Input] 2 bytes short data type value

  nB :[Input] 2 bytes short data type value

- **Return:**

  The value of nA - nB

2.2.3 PTMC12_IntSub

- **Description:**

  To perform the subtraction as iA - iB in int data type. This function is provided for testing DLL linkage.

- **Syntax:**

  short PTMC12_InttSub(int iA, int iB)

- **Parameter:**

nA	:[Input] 4 bytes int data type value
nB	:[Input] 4 bytes int data type value

- **Return:**

  The value of iA – iB

2.2.4 PTMC12_GetDllVersion

- **Description:**

  To get the version number of PTMC12.DLL

- **Syntax:**

  DWORD PTMC12_GetDllVersion(void)

- **Parameter:**

  None

- **Return:**

  Return the DLL's version number.

  For example: 102(hex) for version 1.02

2.3 Functions of Driver Initialization

2.3.1 PTMC12_DriverInit

- **Description :**
  This subroutine will initialize the kernel driver. This function must be called first before calling these DLL functions given in Sec 2.3 ~ Sec. 2.5.

- **Syntax :**
  DWORD PTMC12_DriverInit();

- **Parameter :** void

- **Return:**
  PCI_NoError : OK
  PCI_DriverOpenError: open kernel driver error

2.3.2 PTMC12_DetectBoards

- **Description :**
  This subroutine will detect all installed OME-PCI-TMC12(A) boards.

- **Syntax :**
  DWORD PTMC12_DetectBoards();

- **Parameter :** void

- **Return:**
  0: No OME-PCI-TMC12(A) is installed in this PC
  1: only one OME-PCI-TMC12(A) is installed in this PC(board no.=1)
  N: Number of OME-PCI-TMC12(A) cards installed in this PC(board no.=1, 2, …., N)

- **Note:**
  Call **PTMC12_DriverInit()** before calling this function

2.3.3 PTMC12_OpenBoard

- **Description :**

  This subroutine will lock/engage the OME-PCI-TMC12(A). Then the engaged OME-PCI-TMC12(A) is dedicated to this program until PTMC12_CloseBoard is called. This function must be called first before calling these DLL functions given in Sec 2.3 ~ Sec. 2.5.

- **Syntax :**

  DWORD PTMC12_OpenBoard(dwBoardNo, dwIntEnable);

- **Parameter :**

  dwBoardNo : [Input]board number, 1,2,…
  dwIntEnable : [Input]0=no interrupt, 1=using interrupt

- **Return:**

  PCI_NoError :OK
  PCI_BoardOpenError:open OME-PCI-TMC12(A) error(may be locked by others)
  PCI_BoardNoExceedFindBoards: dwBoardNo > N

- **Note:**

  1. call **PTMC12_DriverInit()** before calling this function
  2. call **PTMC12_DetectCards()** to detect all OME-PCI-TMC12(A) boards

2.3.4 PTMC12_ReadBoardStatus

- **Description :**

  This subroutine will show the lock-status of the OME-PCI-TMC12(A) board.

- **Syntax :**

  DWORD PTMC12_ReadBoardStatus(dwBoardNo);

- **Parameter :**

  dwBoardNo : [Input] OME-PCI-TMC12(A) board number

- **Return:**

  0: this OME-PCI-TMC12(A) is not locked by other program

  1: this OME-PCI-TMC12(A) is locked by other program

- **Note:**

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

2.3.5 PTMC12_ReadId

- **Description :**

  This subroutine will show the IDs of detected OME-PCI-TMC12(A) boards.

- **Syntax :**

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

- **Parameter :**

dwBoardNo	: [Input] OME-PCI-TMC12(A) board number
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

- **Return:**

  0: this is a valid board no → all return IDs are valid

  Other: this is not a valid board no → all return IDs are invalid

- **Note:**

  1. call **PTMC12_DriverInit()** before calling this function
  2. call **PTMC12_DetectBoards()** to detect all OME-PCI-TMC12(A) boards
  3. call **PTMC12_OpenBoard()** before calling this function

2.3.6 PTMC12_CloseBoard

- **Description :**

  This subroutine will unlock the OME-PCI-TMC12(A), then other program can use this OME-PCI-TMC12(A) now.

- **Syntax :**

  DWORD PTMC12_CloseBoard(dwBoardNo);

- **Parameter :**

  dwBoardNo : [Input] OME-PCI-TMC12(A) board number

- **Return:**

  PCI_NoError : OK
  PCI_BoardIsNotOpen: This OME-PCI-TMC12(A) is not locked by others
  PCI_BoardNoExceedFindBoards: dwBoardNo > available board number

- **Note:**

1. call **PTMC12_DriverInit()** before calling this function
2. call **PTMC12_DetectBoards()** to detect all OME-PCI-TMC12(A) boards
3. call **PTMC12_OpenBoard()** before calling this function

2.3.7 PTMC12_CloseAll

- **Description :**

  This subroutine will unlock all OME-PCI-TMC12(A)s installed in this PC, then other program can use these OME-PCI-TMC12(A)s now.

- **Syntax :**

  DWORD PTMC12_CloseAll();

- **Parameter :** void

- **Return:**

  PCI_NoError : OK
  PCI_BoardIsNotOpen: This OME-PCI-TMC12(A) is not locked by others
  PCI_BoardNoExceedFindBoards: dwBoardNo > available board number

- **Note:**

1. call **PTMC12_DriverInit()** before calling this function
2. call **PTMC12_DetectBoards()** to detect all OME-PCI-TMC12(A) boards
3. call **PTMC12_OpenBoard()** before calling this function

2.4 Read/Write to OME-PCI-TMC12(A)

2.4.1 PTMC12_WriteByte

- **Description:**

  Write one byte, 8-bit, of data to OME-PCI-TMC12(A).

- **Syntax:**

  DWORD PTMC12_WriteByte(dwBoardNo, dwOffset, Data)

- **Parameter:**

dwBoardNo	: [Input] board number, from 1 to N
dwOffset	: [Input] offset address
Data	: [Input] one byte of data (8-bit)

- **Return:**

0:	Write OK
PCI_DriverNoOpen:	kernel driver not found
PCI_BoardNoIsZero:	dwBoardNo is 0, it must be in the range of 1 ~ N

  PCI_BoardNoExceedFindBoards: dwBoardNo > N

- **Note:**

  1. call **PTMC12_DetectCards()** before calling this function
  2. call **PTMC12_DetectBoards()** to detect all OME-PCI-TMC12(A) boards.
  3. call **PTMC12_OpenBoard()** before calling this function
  4. **PTMC12_WriteByte(dwBoardNo, 0x10, Data)** → select the active 8254, refer to Sec. 3.3.1 of "OME-PCI-TMC12(A) User's Manual"
  5. **PTMC12_WriteByte(dwBoardNo, 0x14, Data)** → write to D/O-0 ~ 7, refer to Sec. 3.3.4 of "OME-PCI-TMC12(A) User's Manual"

2.4.2 PCITMC12_WriteWord

- **Description:**

  Write one word; 16-bit; of data to OME-PCI-TMC12(A).

- **Syntax:**

  DWORD PTMC12_WriteWord(dwBoardNo, dwOffset, Data)

- **Parameter:**

dwBoardNo	: [Input] board number, from 1 to N
dwOffset	: [Input] offset address
Data	: [Input] one word of data (16-bit)

- **Return:**

0:	Write OK
PCI_DriverNoOpen:	kernel driver not found
PCI_BoardNoIsZero:	dwBoardNo is 0, it must be in the range of 1 ~ N

  PCI_BoardNoExceedFindBoards: dwBoardNo > N

- **Note:**

1. call **PTMC12_DetectCards()** before calling this function
2. call **PTMC12_DetectBoards()** to detect all OME-PCI-TMC12(A) boards
3. call **PTMC12_OpenBoard()** before calling this function
4. **PTMC12_WriteWord(dwBoardNo, 0x10, Data)** → select the active 8254, refer to Sec. 3.3.1 of “OME-PCI-TMC12(A) User’s Manual”
5. **PTMC12_WriteWord(dwBoardNo, 0x14, Data)** → write to D/O-0 ~ 15, refer to Sec. 3.3.4 of “OME-PCI-TMC12(A) User’s Manual”

2.4.3 PTMC12_ReadByte

- **Description:**

  Read one byte, 8-bit, of data from OME-PCI-TMC12(A).

- **Syntax:**

  DWORD PCITMC12_ReadByte(dwBoardNo, dwOffset, *Data)

- **Parameter:**

dwBoardNo	: [Input] board number, from 1 to N
dwOffset	: [Input] offset address
Data	: [output] one byte of data (8-bit)

- **Return:**

0:	Write OK
PCI_DriverNoOpen:	kernel driver not found
PCI_BoardNoIsZero:	dwBoardNo is 0, it must be in the range of 1 ~ N
PCI_BoardNoExceedFindBoards:	dwBoardNo > N

- **Note:**

1. call **PTMC12_DetectCards()** before calling this function
2. call **PTMC12_DetectBoards()** to detect all OME-PCI-TMC12(A) boards.
3. call **PTMC12_OpenBoard()** before calling this function
4. **PTMC12_ReadByte(dwBoardNo, 0x14, Data)** → Read D/I-0 ~ 7, refer to Sec. 3.3.3 of "OME-PCI-TMC12(A) User's Manual"

2.4.4 PTMC12_ReadWord

- **Description:**

Read one word; 16-bit; of data from OME-PCI-TMC12(A).

- **Syntax:**

DWORD PCITMC12_ReadWord(dwBoardNo, dwOffset, *Data)

- **Parameter:**

dwBoardNo	: [Input] board number, from 1 to N
dwOffset	: [Input] offset address
Data	: [output] one byte of data (16-bit)

- **Return:**

0:	Write OK
PCI_DriverNoOpen:	kernel driver not found
PCI_BoardNoIsZero:	dwBoardNo is 0, it must be in the range of 1 ~ N
PCI_BoardNoExceedFindBoards:	dwBoardNo > N

- **Note:**

1. call **PTMC12_DetectCards()** before calling this function
2. call **PTMC12_DetectBoards()** to detect all OME-PCI-TMC12(A) boards
3. call **PTMC12_OpenBoard()** before calling this function
4. **PTMC12_ReadWord(dwBoardNo, 0x14, Data)** → Read D/I-0 ~ 15, refer to Sec. 3.3.3 of "OME-PCI-TMC12(A) User's Manual"

2.5 Interrupt Related DLLs

2.5.1 PTMC12_InstallCallBackFunc

- **Description:**

  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(dwBoardNo, dwIntType,
  user_function)

- **Parameter:**

dwBoardNo	: [Input] board number, from 1 to N
dwIntType	: [Input] 1=initial low & active high, 2=initial high & active low

  user_function(): [Input] address of user's call back function

- **Return:**

0:	Write OK
PCI_DriverNoOpen:	kernel driver not found
PCI_BoardNoIsZero:	dwBoardNo is 0, it must be in the range of 1 ~ N

  PCI_BoardNoExceedFindBoards: dwBoardNo > N

- **Note:**

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

2.5.2 PTMC12_RemoveAllCallBackFunc

- **Description:**

  Disable interrupt & call back functions of all OME-PCI-TMC12(A) installed in this PC.

- **Syntax:**

  DWORD PTMC12_InstallCallBackFunc()

- **Parameter:** void

- **Return:**

0:	Write OK
PCI_DriverNoOpen:	kernel driver not found
PCI_BoardNoIsZero:	dwBoardNo is 0, it must be in the range of 1 ~ N
PCI_BoardNoExceedFindBoards:	dwBoardNo > N

- **Note:**

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

2.5.3 PTMC12_EnableInt

- **Description:**

  Enable interrupt of OME-PCI-TMC12(A).

- **Syntax:**

  DWORD PTMC12_Enable(dwBoardNo)

- **Parameter:**

  dwBoardNo : [Input] board number, from 1 to N

- **Return:**

0:	Write OK
PCI_DriverNoOpen:	kernel driver not found
PCI_BoardNoIsZero:	dwBoardNo is 0, it must be in the range of 1 ~ N
PCI_BoardNoExceedFindBoards:	dwBoardNo > N

- **Note:**

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

2.5.4 PTMC12_DisableInt

- **Description:**

  Disable interrupt of OME-PCI-TMC12(A).

- **Syntax:**

  DWORD PTMC12_Disable(dwBoardNo)

- **Parameter:**

  dwBoardNo : [Input] board number, from 1 to N

- **Return:**

0:	Write OK
PCI_DriverNoOpen:	kernel driver not found
PCI_BoardNoIsZero:	dwBoardNo is 0, it must be in the range of 1 ~ N
PCI_BoardNoExceedFindBoards:	dwBoardNo > N

- **Note:**

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

2.6 Read/Write to PCI Controller

2.6.1 PTMC12_WritePciDword

- **Description:**

  Write one dword; 32-bit; of data to PCI controller

- **Syntax:**

  DWORD PCITMC12_WritePciDword(dwBoardNo, Data)

- **Parameter:**

dwBoardNo	: [Input] board number, from 1 to N
Data	: [Input] one dword of data (32-bit)

- **Return:**

0:	Write OK
PCI_DriverNoOpen:	kernel driver not found
PCI_BoardNoIsZero:	dwBoardNo is 0, it must be in the range of 1 ~ N
PCI_BoardNoExceedFindBoards: dwBoardNo > N	

- **Note:**

1. call **PTMC12_DetectCards()** before calling this function
2. call **PTMC12_DetectBoards()** to detect all OME-PCI-TMC12(A) boards.
3. call **PTMC12_OpenBoard()** before calling this function
4. **PTMC12_WritPciDword(dwBoardNo, 0x4c, Data)** → write to interrupt controller register, refer to Sec. 3.3.5 of “OME-PCI-TMC12(A) User’s Manual”

2.6.2 PTMC12_ReadPciDword

- **Description:**

  Read one dword; 32-bit; of data from PCI control register.

- **Syntax:**

  DWORD PCITMC12_ReadPciDword(dwBoardNo, *Data)

- **Parameter:**

  dwBoardNo : [Input] board number, from 1 to N

  Data: [output] one dword of data (32-bit)

- **Return:**

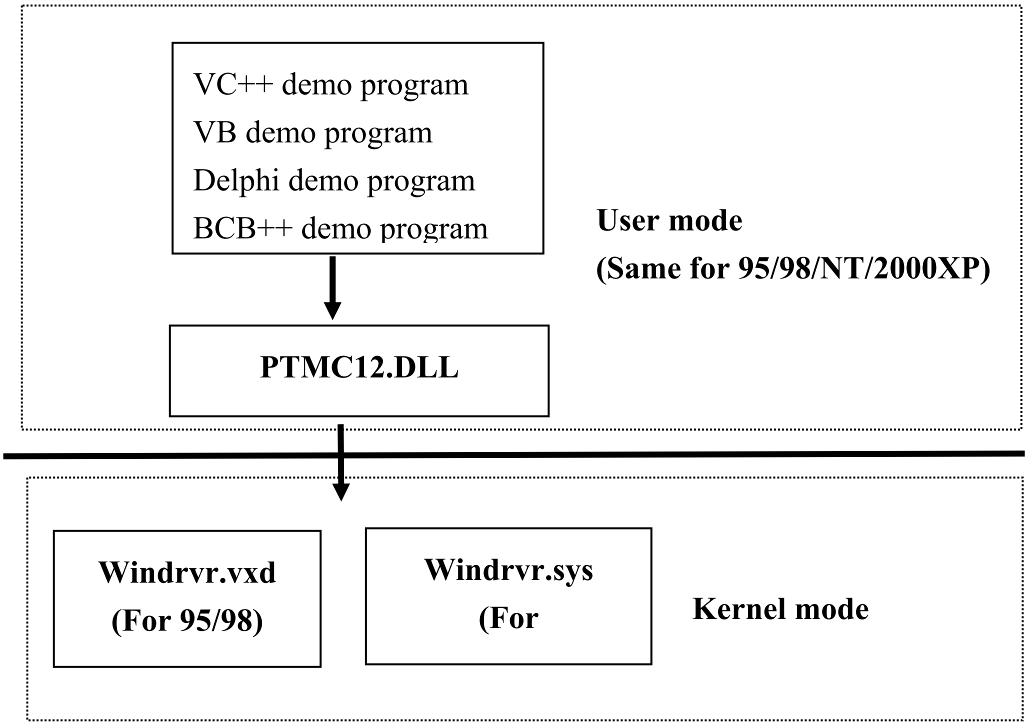
0:	Write OK
PCI_DriverNoOpen:	kernel driver not found
PCI_BoardNoIsZero:	dwBoardNo is 0, it must be in the range of 1 ~ N
PCI_BoardNoExceedFindBoards:	dwBoardNo > N

- **Note:**

1. call **PTMC12_DetectCards()** before calling this function
2. call **PTMC12_DetectBoards()** to detect all OME-PCI-TMC12(A) boards
3. call **PTMC12_OpenBoard()** before calling this function
4. **PTMC12_ReadPciDword(dwBoardNo, 0x4c, Data)** → read the interrupt status register, refer to Sec. 3.3.5 of “OME-PCI-TMC12(A) User’s Manual)

3. Demo Program

There are many demo programs; written in VC++, VB, Delphi, and BCB++; provided with the companion CD. These demo programs will call the DLL PTMC12.DLL to access the hardware of OME-PCI-TMC12(A). The PTMC12.DLL will call the kernel driver, Windrvr.vxd or Windrvr.sys as follows:



The install shield will install kernel driver, DLL driver & application demo program to system. **All demo programs & DLLs are same for 95/98/NT/2000/XP.** But the kernel drivers are different for different system as follows:

- for windows 95/98 → will copy ***WINDRVR.VXD*** to C:\WIN95\SYSTEM\VMM32
- for windows NT/2000/XP → will copy ***WINDRVR.SYS*** to C:\WINNT\SYSTEM32\DRIVERS

The DLL & demo programs will not work if the kernel driver is not installed correctly. The install shield will copy the correct kernel driver to correct position if you select correct O.S.(95/98, NT, 2000, XP).

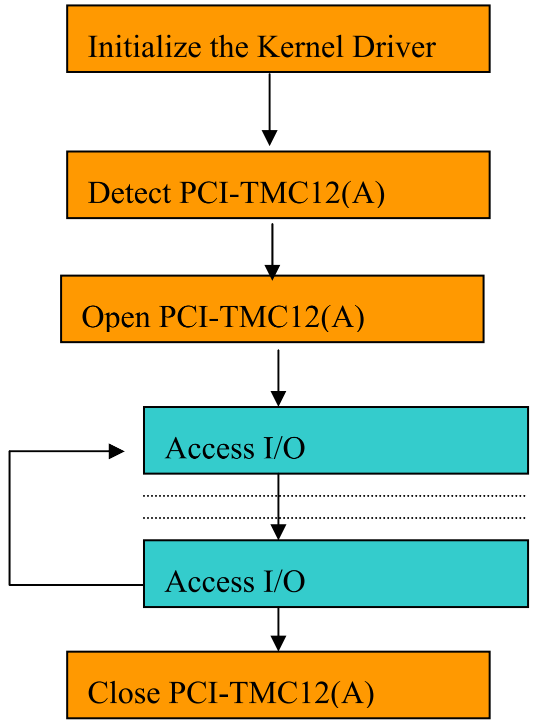
Refer to **CallDll.pdf** for more information about how to call the DLL functions with VC++5, VB5, Delphi3 and Borland C++ Builder 3. Refer to **step5 of Sec. 1.2.2** for more information.

3.1 Program Architecture

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

The PCI_OpenBoard(…) will open & lock the target OME-PCI-TMC12(A) until PCI_CloseBoard(…) or PCI_CloseAll() is called.

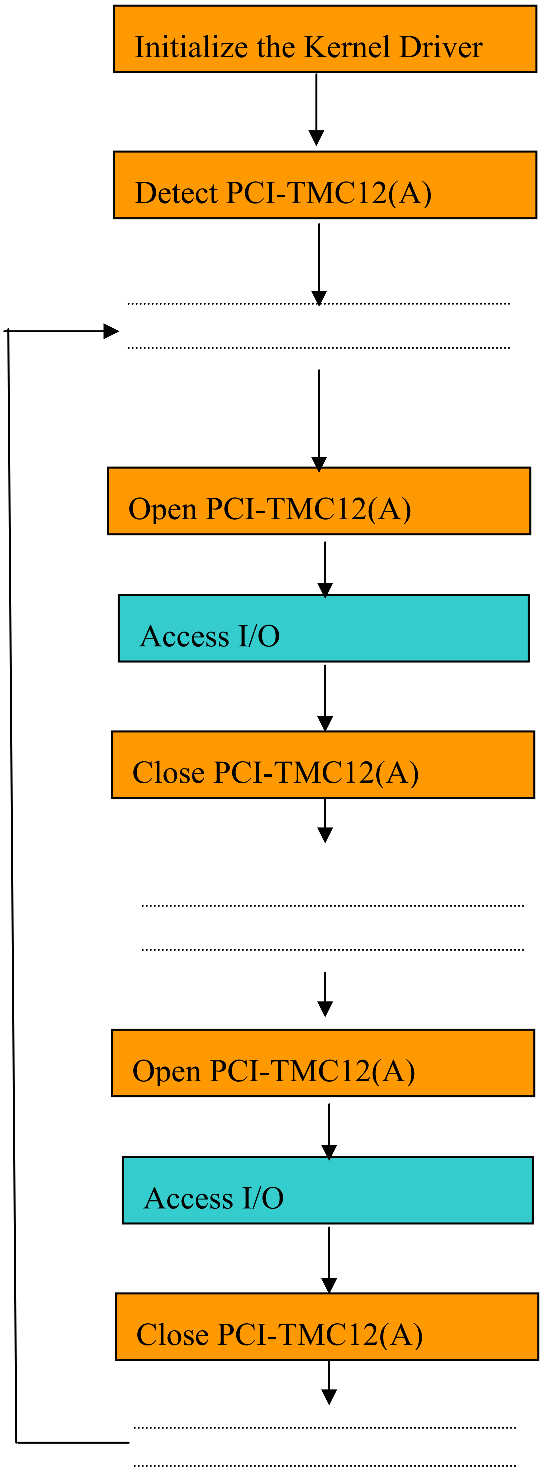
For single-task applications, only one user's program control OME-PCI-TMC12(A). So the program will open & lock OME-PCI-TMC12(A) in the program is start & un-lock OME-PCI-TMC12(A) in the program is exit as follows:



```
PTMC12_DriverInit()
PTMC12_DetectBoards()
PTMC12_OpenBoard(…)
….
….
    PTMC12_ReadWord(…)
    ……..
    ……..
    PTMC12_WriteWord(…)

…..
PTMC12_CloseAll()
```

For multi-task applications, many user programs will control OME-PCI-TMC12(A). So the program will open & lock OME-PCI-TMC12(A) before access the I/O. Then un-lock OME-PCI-TMC12(A) after access the I/O as follows:



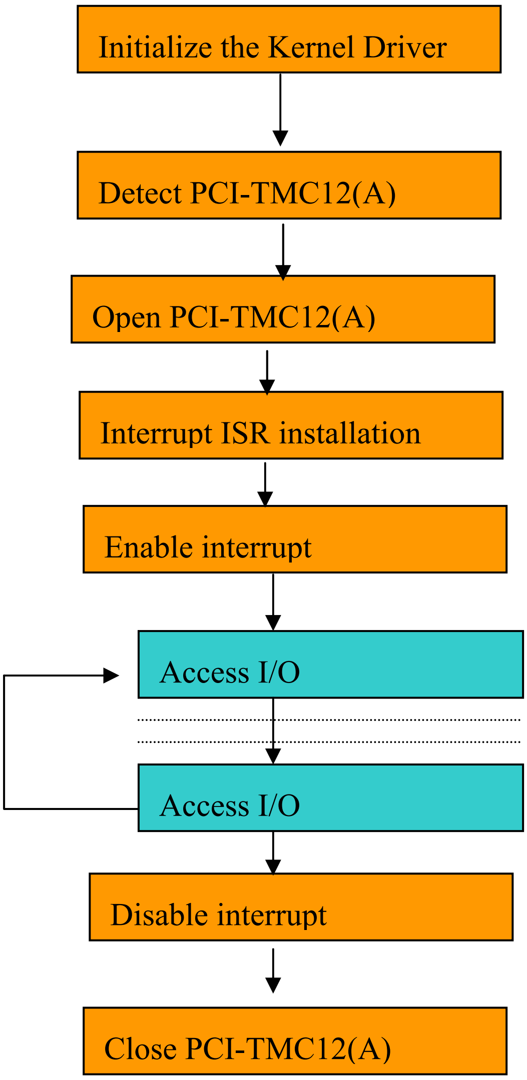
```
PTMC12_DriverInit()
PTMC12_DetectBoards()
….
….
   PTMC12_OpenBoard(…)
    PTMC12_ReadWord(…)
    PTMC12_CloseBoard(…)


     PTMC12_OpenBoard(…)
    PTMC12_ReadWord(…)
    PTMC12_CloseBoard(…)……..
     ……..



…..
PTMC12_CloseAll()
```

For interrupt applications, the program style is given as follows:



```
PTMC12_DriverInit()
PTMC12_DetectBoards()
PTMC12_OpenBoard(…)
PTMC12_InstallCallBackFunc(…)
PTMC12_EnableInt(….)
….
….
    PTMC12_ReadWord(…)
    ……..
    ……..
    PTMC12_WriteWord(…)

…..
PTMC12_DisableInt(…)
PTMC12_CloseAll()
```

When the interrupt signal is active, the kernel mode driver will call user's call back function once. Refer to demo program, WatchDog & CountLow, for more information.

3.2 Report Problems

Technical support is available at no charge as described below. The best way to report problems is send electronic mail to **das@omega.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 Operation Systems that you running? For example, Windows 3.1, Windows for Workgroups, Windows NT 4.0, etc.

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 text 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 received your comments, we will take about two business days to testing the problems that you said. And then reply as soon as possible to you. Please check that we have received your comments? And please keeping contact with us.

E-mail: das@omega.com
Web-Site: http://www.omega.com

WARRANTY/DISCLAIMER

OMEGA ENGINEERING, INC. warrants this unit to be free of defects in materials and workmanship for a period of **13 months** from date of purchase. OMEGA's WARRANTY adds an additional one (1) month grace period to the normal **one (1) year product warranty** to cover handling and shipping time. This ensures that OMEGA's customers receive maximum coverage on each product.

If the unit malfunctions, it must be returned to the factory for evaluation. OMEGA's Customer Service Department will issue an Authorized Return (AR) number immediately upon phone or written request. Upon examination by OMEGA, if the unit is found to be defective, it will be repaired or replaced at no charge. OMEGA's WARRANTY does not apply to defects resulting from any action of the purchaser, including but not limited to mishandling, improper interfacing, operation outside of design limits, improper repair, or unauthorized modification. This WARRANTY is VOID if the unit shows evidence of having been tampered with or shows evidence of having been damaged as a result of excessive corrosion; or current, heat, moisture or vibration; improper specification; misapplication; misuse or other operating conditions outside of OMEGA's control. Components which wear are not warranted, including but not limited to contact points, fuses, and triacs.

OMEGA is pleased to offer suggestions on the use of its various products. However, OMEGA neither assumes responsibility for any omissions or errors nor assumes liability for any damages that result from the use of its products in accordance with information provided by OMEGA, either verbal or written. OMEGA warrants only that the parts manufactured by it will be as specified and free of defects. OMEGA MAKES NO OTHER WARRANTIES OR REPRESENTATIONS OF ANY KIND WHATSOEVER, EXPRESS OR IMPLIED, EXCEPT THAT OF TITLE, AND ALL IMPLIED WARRANTIES INCLUDING ANY WARRANTY OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE HEREBY DISCLAIMED. LIMITATION OF LIABILITY: The remedies of purchaser set forth herein are exclusive, and the total liability of OMEGA with respect to this order, whether based on contract, warranty, negligence, indemnification, strict liability or otherwise, shall not exceed the purchase price of the component upon which liability is based. In no event shall OMEGA be liable for consequential, incidental or special damages.

CONDITIONS: Equipment sold by OMEGA is not intended to be used, nor shall it be used: (1) as a "Basic Component" under 10 CFR 21 (NRC), used in or with any nuclear installation or activity; or (2) in medical applications or used on humans. Should any Product(s) be used in or with any nuclear installation or activity, medical application, used on humans, or misused in any way, OMEGA assumes no responsibility as set forth in our basic WARRANTY/DISCLAIMER language, and, additionally, purchaser will indemnify OMEGA and hold OMEGA harmless from any liability or damage whatsoever arising out of the use of the Product(s) in such a manner.

RETURN REQUESTS/INQUIRIES

Direct all warranty and repair requests/inquiries to the OMEGA Customer Service Department. BEFORE RETURNING ANY PRODUCT(S) TO OMEGA, PURCHASER MUST OBTAIN AN AUTHORIZED RETURN (AR) NUMBER FROM OMEGA'S CUSTOMER SERVICE DEPARTMENT (IN ORDER TO AVOID PROCESSING DELAYS). The assigned AR number should then be marked on the outside of the return package and on any correspondence.

The purchaser is responsible for shipping charges, freight, insurance and proper packaging to prevent breakage in transit.

FOR **WARRANTY** RETURNS, please have the following information available BEFORE contacting OMEGA:

1. Purchase Order number under which the product was PURCHASED,
2. Model and serial number of the product under warranty, and
3. Repair instructions and/or specific problems relative to the product.

FOR **NON-WARRANTY** REPAIRS, consult OMEGA for current repair charges. Have the following information available BEFORE contacting OMEGA:

1. Purchase Order number to cover the COST of the repair,
2. Model and serial number of the product, and
3. Repair instructions and/or specific problems relative to the product.

OMEGA's policy is to make running changes, not model changes, whenever an improvement is possible. This affords our customers the latest in technology and engineering.

OMEGA is a registered trademark of OMEGA ENGINEERING, INC.

Where Do I Find Everything I Need for Process Measurement and Control? OMEGA...Of Course!

Shop online at www.omega.com

TEMPERATURE

- ☑ Thermocouple, RTD & Thermistor Probes, Connectors, Panels & Assemblies
- ☑ Wire: Thermocouple, RTD & Thermistor
- ☑ Calibrators & Ice Point References
- ☑ Recorders, Controllers & Process Monitors
- ☑ Infrared Pyrometers

PRESSURE, STRAIN AND FORCE

- ☑ Transducers & Strain Gages
- ☑ Load Cells & Pressure Gages
- ☑ Displacement Transducers
- ☑ Instrumentation & Accessories

FLOW/LEVEL

- ☑ Rotameters, Gas Mass Flowmeters & Flow Computers
- ☑ Air Velocity Indicators
- ☑ Turbine/Paddlewheel Systems
- ☑ Totalizers & Batch Controllers

pH/CONDUCTIVITY

- ☑ pH Electrodes, Testers & Accessories
- ☑ Benchtop/Laboratory Meters
- ☑ Controllers, Calibrators, Simulators & Pumps
- ☑ Industrial pH & Conductivity Equipment

DATA ACQUISITION

- ☑ Data Acquisition & Engineering Software
- ☑ Communications-Based Acquisition Systems
- ☑ Plug-in Cards for Apple, IBM & Compatibles
- ☑ Datalogging Systems
- ☑ Recorders, Printers & Plotters

HEATERS

- ☑ Heating Cable
- ☑ Cartridge & Strip Heaters
- ☑ Immersion & Band Heaters
- ☑ Flexible Heaters
- ☑ Laboratory Heaters

ENVIRONMENTAL MONITORING AND CONTROL

- ☑ Metering & Control Instrumentation
- ☑ Refractometers
- ☑ Pumps & Tubing
- ☑ Air, Soil & Water Monitors
- ☑ Industrial Water & Wastewater Treatment
- ☑ pH, Conductivity & Dissolved Oxygen Instruments