User's Guide

omega.com®
OMEGA®

Shop online at

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

ISO 9001 CERTIFIED CORPORATE QUALITY STAMFORD, CT

ISO 9002 CERTIFIED CORPORATE QUALITY MANCHESTER, UK

OME-PCI-1002

PCI Data Acquisition Board

Windows Software Manual

omega.com®
OMEGA®

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

Table of Contents

1. Introduction 4
   1.1 Software Installation 5
   1.2 References 7
2. Declaration Files 8
   2.1 P100X.H 9
   2.2 P100Xu.C 11
   2.3 P100X.BAS 12
   2.4 P100Xu.BAS 14
   2.5 P100X.PAS 15
   2.6 P100Xu.PAS 18
3. Demo Result 19
   3.1 Visual C++ 19
   3.2 Visual Basic 20
   3.3 Delphi 21
   3.4 Borland C++ Builder 22
4. Descriptions of Functions 23
   4.1 The Configuration Code Table 25
   4.2 The Test Functions 26
      P100X_FloatSub2 26
      P100X_ShortSub2 26
      4.2.3 P100X_GetDllVersion 27
      P100X_GetDriverVersion 27
   4.3 The DI/O Functions 28
      P100X_Di 28
      P100X_Do 28
   4.4 The A/D Fixed-mode Functions 29
      P100X_SetChannelConfig 29
      4.4.2 P100X_Polling 30
      4.4.3 P100X_AdPolling 31
      4.4.4 P100X_AdsPolling 32
      4.4.5 P100X_AdsPacer 33
   4.5 Driver Functions 34
      P100X_DriverInit 34
      P100X_DriverClose 34
      4.5.3 P100X_GetConfigAddressSpace 35
      P100X_WhichBoardActive 35
      4.5.5 P100X_ActiveBoard 36
      P100X_GetIrqNo 36
   4.6 The Interrupt Functions 37
      P100X_InstallIrq 37
      P100X_GetIntCount 37
      4.6.3 P100X_INT_AdStart 38
      P100X_INT_AdStop 38
      4.6.5 P100X_GetBuffer 39
      P100X_GetFloatBuffer 39
      4.6.7 Architecture of Interrupt mode 40
5. Program Architecture 44
6. Reporting Problems 45

1. Introduction

The **OME-PCI-1002 Toolkit** is a collection of DLLs and device-driver for Windows 95/98/NT/2000/XP applications. These DLLs are 32-bit and can be called by Visual C++, BC++, Visual BASIC, Delphi and LabVIEW.

The OME-PCI-1002 Toolkit consists of the following DLLs and device driver:

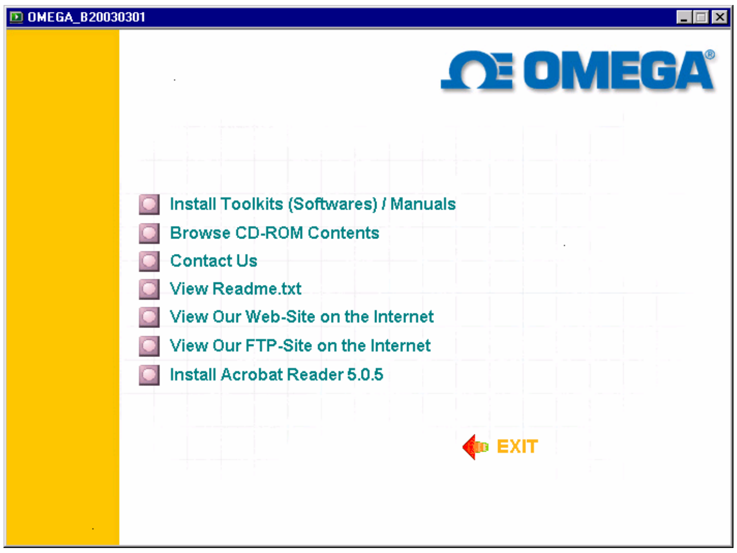
- P100X.DLL, P100X.LIB → for OME-PCI-1002 card
- P100X.VXD → OME-PCI-1002 Device driver for Windows 95/98
- P100X.SYS → OME-PCI-1002 Device driver for Windows NT/2000/XP

The DLLs perform a variety of tasks including:

- Read software version
- Initialization
- Digital Input/Output
- A/D conversion

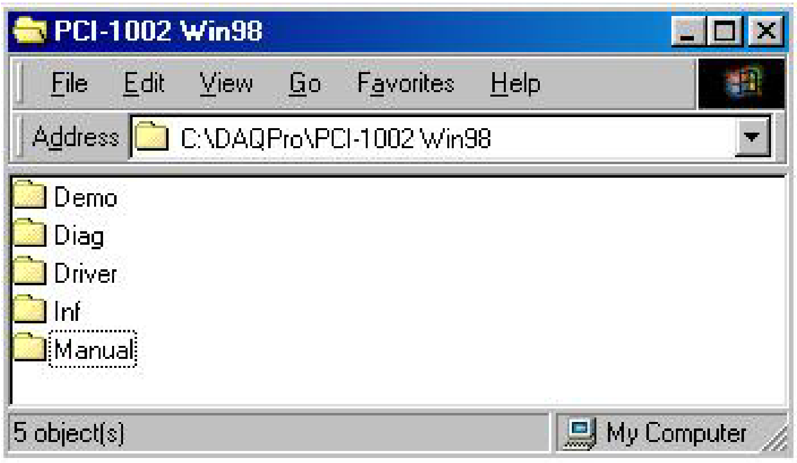
1.1 Software Installation

Insert the CD ROM included with your OME-PCI-1002 board and the following installation screen should auto-start.


OMEGA_B20030301
OMEGA®
Install Toolkits (Softwares) / 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

Follow the instructions on the screen to complete the software installation. The software is designed to support the entire OME family of data acquisition hardware, so during the installation, you will be asked to specify your particular hardware (OME-PCI-1002 board in this case). During the installation process, you will also be prompted to enter the operating system you will be using.

After installation the following folders will be created on your computer.



C:\DAQPro\PCI-1002 Win98

Demo
Diag
Driver
Inf
Manual

5 object(s)

Demo Folder

Contains all demonstration programs including their source code. Examples are provided for Visual C++, Borland C++, Visual Basic and Delphi.
Please note: The VC++ demos are developed with VC++ 4.0. After setting up the environment, use the NMAKE.EXE to compiling and linking the demo code. For Example, C:\P1002\DEMO\VC\nmake /f demo.mak

Driver Folder

Contains software drivers, include files and definition files for the programming languages.

Manual Folder

Contains hardware user manuals, software user manuals and technical notes.

Diag Folder

Contains card diagnostic programs

Inf Folder

Contains tech notes and .INF file for the plug and play installation (only available for operating systems that support plug and play).

1.2 References

Please also refer to the following user manuals:

- **SoftInst.pdf:**
  To install the software package under Windows 95/98/NT/2000/XP.

- **CallDll.pdf:**
  To call the DLL functions with Visual C++, Visual Basic, Delphi and Borland C++.

- **ResCheck.pdf:**
  To check the card resources, that is, I/O Port address, IRQ number and DMA under Windows.

2. Declaration Files

Please refer to user manual "CallDLL.pdf".

```
|--\Driver                  ← some device driver
|       |
|       |--\VB                      ← for Visual Basic
|       |       |--\P100X.BAS       ← Declaration file for Visual Basic
|       |       |--\P100Xu.BAS      ← Functions for Visual Basic
|       |
|       |--\VC                      ← for Visual C++
|       |       |--\P100X.H         ← Header file
|       |       |--\P100X.LIB       ← Import library for VC only
|       |
|       |--\Delphi                  ← for Delphi
|       |       |--\P100X.PAS       ← Declaration file
|       |       |--\P100Xu.PAS      ← Functions for Delphi
|       |
|       |--\BCB                     ← for Borland C++ Builder 3.0
|               |--\P100X.H         ← Header file
|               |--\P100Xu.C        ← Functions for BCB
|               |--\P100X.LIB       ← Import library file for BCB only
```

2.1 P100X.H

```
#ifdef __cplusplus
    #define EXPORTS extern "C" __declspec (dllimport)
#else
    #define EXPORTS
#endif

// return code
#define P100X_NoError                   0
#define P100X_DriverHandleError         1
#define P100X_DriverCallError           2
#define P100X_AdControllerError         3
#define P100X_ConfigCodeError           4
#define P100X_DriverNoOpen              5
#define P100X_AdPollingTimeOut          6
#define P100X_FindBoardError            7
#define P100X_AdChannelError            8
#define P100X_DaChannelError            9
#define P100X_InvalidDelay          10
#define P100X_DelayTimeOut              11
#define P100X_InvalidData           12
#define P100X_TimeoutError              13
#define P100X_ExceedBoardNumber         14
#define P100X_NotFoundBoard             15
#define P100X_OpenError                 16
#define P100X_FindTwoBoardError         17
#define P100X_GetIntCountError          18
#define P100X_InstallIrqError           19
#define P100X_AllocateMemoryError       20

EXPORTS float       CALLBACK P100X_FloatSub(float fA, float fB);
EXPORTS short       CALLBACK P100X_ShortSub(short nA, short nB);
EXPORTS WORD    CALLBACK P100X_GetDllVersion(void);

EXPORTS WORD    CALLBACK P100X_DriverInit(WORD *wTotalBoards);
EXPORTS void        CALLBACK P100X_DriverClose(void);
EXPORTS WORD    CALLBACK P100X_GetDriverVersion
          (WORD *wDriverVersion);
EXPORTS WORD    CALLBACK P100X_GetIrqNo( WORD  *IrqNo);

EXPORTS WORD    CALLBACK P100X_GetConfigAddressSpace
          (WORD wBoardNo, WORD *wAddress0,
           WORD *wAddress1, WORD *wAddress2);

EXPORTS WORD    CALLBACK P100X_ActiveBoard( WORD wBoardNo );
EXPORTS WORD    CALLBACK P100X_WhichBoardActive(void);
EXPORTS void       CALLBACK P100X_SetupTimer
        (WORD wChannel, WORD wCoef);
EXPORTS WORD   CALLBACK P100X_Delay(WORD wDownCount);

EXPORTS void       CALLBACK P100X_Do(WORD wOutData);
EXPORTS WORD   CALLBACK P100X_Di(WORD *wDiData);

EXPORTS WORD   CALLBACK P100X_SetChannelConfig
        (WORD wAdChannel, WORD wConfig);
EXPORTS WORD   CALLBACK P100X_Polling(WORD *wAdVal);
EXPORTS WORD   CALLBACK P100X_AdPolling(float *fAdVal);
EXPORTS WORD   CALLBACK P100X_AdsPolling
        (float fAdVal[], WORD wNum);

EXPORTS WORD   CALLBACK P100X_AdsPacer
        (float fAdVal[], WORD wNum,  WORD wSamplingDiv);


EXPORTS WORD   CALLBACK P100X_InstallIrq
        (HANDLE *hEvent, DWORD dwCount);
EXPORTS WORD   CALLBACK P100X_GetBuffer
        (DWORD dwNum, WORD wBuf[]);
EXPORTS WORD   CALLBACK P100X_GetFloatBuffer
        (DWORD dwNum, float fAdVal[]);

EXPORTS WORD   CALLBACK P100X_GetIntCount(DWORD *dwVal);
EXPORTS WORD   CALLBACK P100X_INT_AdStart
        (WORD Ch, WORD Gain, WORD wFreqDiv);

EXPORTS WORD   CALLBACK P100X_INT_AdStop();
```

2.2 P100Xu.C

```
#include <math.h>


//*-------------------------------------------------*
//* Return voltage value or -100.0 if any error occurs  *
//* or parameter is out of range.                       *
//* HiLo : 1 --> High Gain , 0 --> Low Gain             *
//* Gain : 0-3                                          *
//*-------------------------------------------------*
float P100X_AD2F(Word hex, int HiLo,int Gain )
{
   float ZeroBase, VoltageRange, FullRange ;

   ZeroBase  = 2048.0 ;
   FullRange = 2048.0 ;
   VoltageRange = 10.0 ;
   Gain = Gain % 16 ;
   if ( (Gain < 0) || (Gain > 3) )
      return -100.0;

   if ( HiLo == 0 )   //Low-Gain
      return ((((hex - ZeroBase) / FullRange) * VoltageRange) / pow(  2 , Gain));
   else
      return ((((hex - ZeroBase) / FullRange) * VoltageRange) / pow( 10 ,
Gain));

}
```

2.3 P100X.BAS

```
Attribute VB_Name = "P100X"

' return code
Global Const P100X_NoError                  = 0
Global Const P100X_DriverHandleError        = 1
Global Const P100X_DriverCallError          = 2
Global Const P100X_AdControllerError        = 3
Global Const P100X_ConfigCodeError          = 4
Global Const P100X_DriverNoOpen             = 5
Global Const P100X_AdPollingTimeOut         = 6
Global Const P100X_FindBoardError           = 7
Global Const P100X_AdChannelError           = 8
Global Const P100X_DaChannelError           = 9
Global Const P100X_InvalidDelay             = 10
Global Const P100X_DelayTimeOut             = 11
Global Const P100X_InvalidData         = 12
Global Const P100X_TimeoutError             = 13
Global Const P100X_ExceedBoardNumber        = 14
Global Const P100X_NotFoundBoard            = 15
Global Const P100X_OpenError                = 16
Global Const P100X_FindTwoBoardError        = 17
Global Const P100X_GetIntCountError         = 18
Global Const P100X_InstallIrqError          = 19
Global Const P100X_AllocateMemoryError      = 20

' Function of Test
Declare Function P100X_FloatSub Lib "P100X.DLL" _
        (ByVal fA As Single, ByVal fB As Single) As Single
Declare Function P100X_ShortSub Lib "P100X.DLL" _
        (ByVal nA As Integer, ByVal nB As Integer) As Integer
Declare Function P100X_GetDllVersion Lib "P100X.DLL" () As Integer

' Function of Driver
Declare Function P100X_DriverInit Lib "P100X.DLL" _
        (wTotalBoards As Integer) As Integer
Declare Sub P100X_DriverClose Lib "P100X.DLL" ()
Declare Function P100X_GetDriverVersion Lib "P100X.DLL" _
        (wDriverVersion As Integer) As Integer
Declare Function P100X_GetIrqNo Lib "P100X.DLL" (IrqNo As Integer) _
        As Integer
Declare Function P100X_GetConfigAddressSpace Lib "P100X.DLL" _
        (ByVal wBoardNo As Integer, wAddrTimer As Integer, _
        wAddrDio As Integer, wAddrAd As Integer) As Integer
Declare Function P100X_ActiveBoard Lib "P100X.DLL" _
        (ByVal wBoardNo As Integer) As Integer
Declare Function P100X_WhichBoardActive Lib "P100X.DLL" () As Integer
Declare Sub P100X_SetupTimer Lib "P100X.DLL" _
        (ByVal wChannel As Integer, ByVal wCoef As Integer)
Declare Function P100X_Delay Lib "P100X.DLL" _
        (ByVal wDownCount As Integer) As Integer

' Function of DI/DO
Declare Sub P100X_Do Lib "P100X.DLL" (ByVal wOutData As Integer)
Declare Function P100X_Di Lib "P100X.DLL" (wDiData As Integer) As Integer

' Function of AD
Declare Function P100X_SetChannelConfig Lib "P100X.DLL" _
        (ByVal wAdChannel As Integer, ByVal wConfig As Integer) As Integer
Declare Function P100X_Polling Lib "P100X.DLL" _
        (wAdVal As Integer) As Integer
Declare Function P100X_AdPolling Lib "P100X.DLL" _
        (fAdVal As Single) As Integer
Declare Function P100X_AdsPolling Lib "P100X.DLL" _
        (fAdVal As Single, ByVal wNum As Integer) As Integer
Declare Function P100X_AdsPacer Lib "P100X.DLL" (fAdVal As Single, _
        ByVal wNum As Integer, ByVal wSamplingDiv As Integer) As Integer

' Function of Interrupt
Declare Function P100X_InstallIrq Lib "P100X.DLL" _
        (hEvent As Long, ByVal dwCount As Long) As Integer
Declare Function P100X_GetBuffer Lib "P100X.DLL" _
        (ByVal dwNum As Long, wBuf As Integer) As Integer
Declare Function P100X_GetFloatBuffer Lib "P100X.DLL" _
        (ByVal dwNum As Long, fAdVal As Single) As Integer
Declare Function P100X_INT_AdStart Lib "P100X.DLL" _
        (ByVal Ch As Integer, ByVal Gain As Integer, _
        ByVal wFreqDiv As Integer) As Integer
Declare Function P100X_INT_AdStop Lib "P100X.DLL" () As Integer
Declare Function P100X_GetIntCount Lib "P100X.DLL" (dwVal As Long) _
        As Integer
```

2.4 P100Xu.BAS

```
'*--------------------------------------------------*
'* Return voltage value or -100.0 if any error occurs  *
'* or parameter is out of range.                        *
'* HiLo : 1 --> High Gain , 0 --> Low Gain              *
'* Gain : 0-3                                    *
'*--------------------------------------------------*
Function P100X_AD2F(ByVal hex, HiLo, Gain As Integer) As Single
   Dim ZeroBase, BullRange, VoltageRange As Single

   ZeroBase = 2048#
   FullRange = 2048#
   VoltageRange = 10#
   Gain = Gain Mod 16

   If Gain < 0 Or Gain > 3 Then
      P100X_AD2F = -100#
      Exit Function
   End If

   If HiLo = 0 Then   'Low-Gain
      P100X_AD2F = ((((hex - ZeroBase) / FullRange) * VoltageRange) / (2 ^
Gain))
   Else
      P100X_AD2F = ((((hex - ZeroBase) / FullRange) * VoltageRange) / (10 ^
Gain))
   End If
End Function
```

2.5 P100X.PAS

```
unit P100X;

interface

type PSingle=^Single;
type PWord=^Word;

const
// return code
 P100X_NoError                    =   0;
 P100X_DriverHandleError          =   1;
 P100X_DriverCallError            =   2;
 P100X_AdControllerError          =   3;
 P100X_ConfigCodeError            =   4;
 P100X_DriverNoOpen               =   5;
 P100X_AdPollingTimeOut           =   6;
 P100X_FindBoardError             =   7;
 P100X_AdChannelError             =   8;
 P100X_DaChannelError             =   9;
 P100X_InvalidDelay               =   10;
 P100X_DelayTimeOut               =   11;
 P100X_InvalidData                =   12;
 P100X_TimeoutError               =   13;
 P100X_ExceedBoardNumber          =   14;
 P100X_NotFoundBoard              =   15;
 P100X_OpenError                  =   16;
 P100X_FindTwoBoardError          =   17;
 P100X_GetIntCountError           =   18;
 P100X_InstallIrqError            =   19;
 P100X_AllocateMemoryError        =  20;

// Function of Test
function  P100X_FloatSub(fA:Single; fB:Single):Single ; stdCall;
function  P100X_ShortSub(nA:SmallInt; nB:SmallInt):SmallInt ; stdCall;
function  P100X_GetDllVersion:WORD  ; stdCall;
```

```
// Function of Driver
function  P100X_DriverInit(Var wTotalBoards:Word):WORD ; stdCall;
procedure P100X_DriverClose; stdCall;
function  P100X_GetDriverVersion(var wDriverVersion:Word):WORD ; stdCall;
function  P100X_GetIrqNo(Var IrqNo:WORD):WORD; StdCall;
function  P100X_GetConfigAddressSpace(wBoardNo:Word;
          var wAddrTimer:Word; var wAddrDio:Word;
          var wAddrAd:Word) :WORD ; stdCall;
function  P100X_ActiveBoard(wBoardNo:Word):WORD ; stdCall;
function  P100X_WhichBoardActive:WORD ; stdCall;
procedure P100X_SetupTimer(wChannel:Word; wCoef:Word); stdCall;
function  P100X_Delay(wDownCount:Word):Word; StdCall;

// Function of DI/DO
procedure P100X_Do(wOutData:Word); stdCall;
function  P100X_Di(var wDiData:Word):WORD ; stdCall;

// Function of AD
function  P100X_SetChannelConfig
          (wAdChannel:Word; wConfig:Word):WORD ; stdCall;
function  P100X_Polling(var wAdVal:Word):WORD ; stdCall;
function  P100X_AdPolling(var fAdVal:Single):WORD ; stdCall;
function  P100X_AdsPolling(fAdVal:PSingle; wNum:Word):WORD ; stdCall;
function  P100X_AdsPacer(fAdVal:PSingle; wNum:Word;
           wSamplingDiv:Word ):WORD ; stdCall;

// Function of Interrupt
function  P100X_InstallIrq
          (Var hEvent:LongInt; dwCount: LongInt):WORD ; stdCall;
function  P100X_GetBuffer(dwNum:LongInt;wBuf:PWord):WORD ; stdCall;
function  P100X_GetFloatBuffer
          (dwNum:LongInt; fAdVal:PSingle):Word; StdCall;
function  P100X_INT_AdStart
          (Ch:WORD; Gain:WORD; wFreqDiv:Word):WORD ; stdCall;
function  P100X_INT_AdStop:WORD ; stdCall;
function  P100X_GetIntCount(var dwVal:LongInt):WORD ; stdCall;


implementation

function        100X_FloatSub;    external 'P100X.DLL' name 'P100X_FloatSub';
function        100X_ShortSub;   external 'P100X.DLL' name 'P100X_ShortSub';
function        100X_GetDllVersion;
                external 'P100X.DLL' name 'P100X_GetDllVersion';
function        100X_GetDriverVersion;
                external 'P100X.DLL' name 'P100X_GetDriverVersion';
```

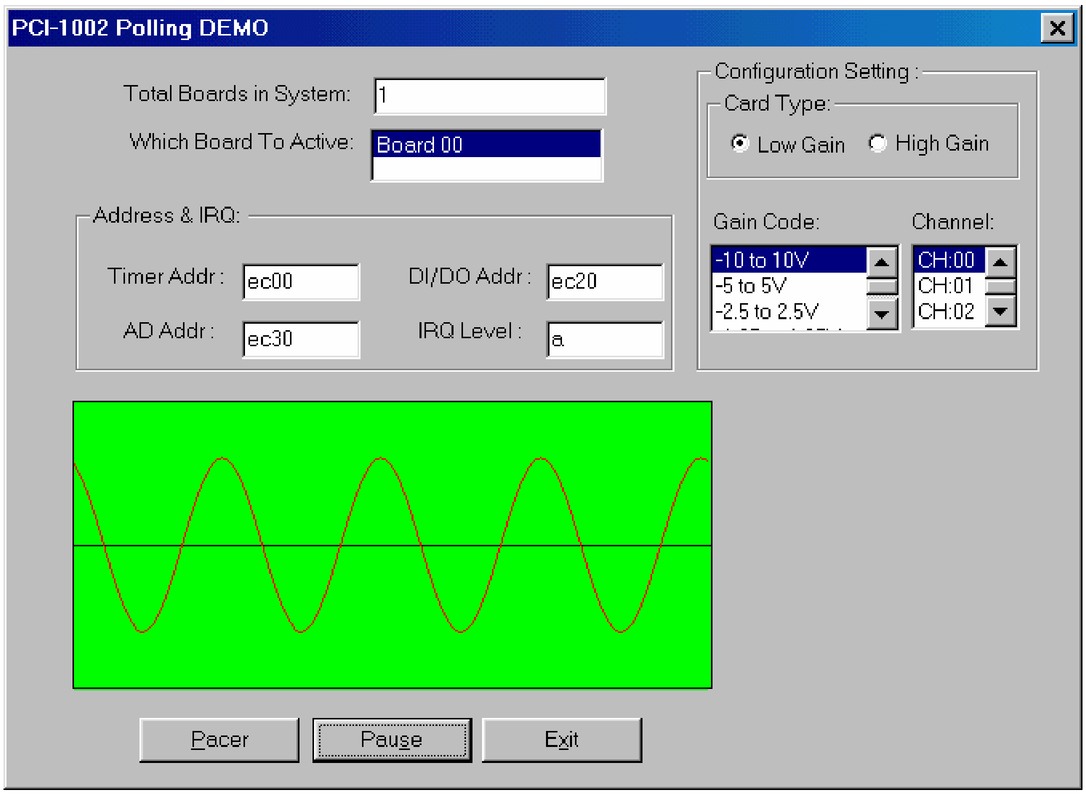
```
function     100X_DriverInit;  external 'P100X.DLL' name 'P100X_DriverInit';
procedure    100X_DriverClose;
             external 'P100X.DLL'      name 'P100X_DriverClose';
function     100X_GetIrqNo;  external 'P100X.DLL' name 'P100X_GetIrqNo';
function     100X_GetConfigAddressSpace;
             external 'P100X.DLL' name 'P100X_GetConfigAddressSpace';
function     100X_ActiveBoard;
             external 'P100X.DLL'      name 'P100X_ActiveBoard';
function     100X_WhichBoardActive;
             external 'P100X.DLL'      name 'P100X_WhichBoardActive';
procedure    100X_SetupTimer;
             external 'P100X.DLL'      name 'P100X_SetupTimer';
function     100X_Delay;     external 'P100X.DLL' name 'P100X_Delay';


procedure    P100X_Do;       external 'P100X.DLL' name 'P100X_Do';
function     P100X_Di;       external 'P100X.DLL' name 'P100X_Di';


function     P100X_SetChannelConfig;
             external 'P100X.DLL'      name 'P100X_SetChannelConfig';
function     P100X_Polling;  external 'P100X.DLL' name 'P100X_Polling';
function     P100X_AdPolling;
             external 'P100X.DLL'      name 'P100X_AdPolling';
function     P100X_AdsPolling;
             external 'P100X.DLL'      name 'P100X_AdsPolling';

function     P100X_AdsPacer;
             external 'P100X.DLL'      name 'P100X_AdsPacer';

function     P100X_InstallIrq;
             external 'P100X.DLL'      name 'P100X_InstallIrq';
function     P100X_INT_AdStart;
             external 'P100X.DLL'      name 'P100X_INT_AdStart';
function     P100X_INT_AdStop;
             external 'P100X.DLL'      name 'P100X_INT_AdStop';
function     P100X_GetIntCount;
             external 'P100X.DLL'      name 'P100X_GetIntCount';
function     P100X_GetBuffer;
             external 'P100X.DLL'      name 'P100X_GetBuffer';
function     P100X_GetFloatBuffer;
             external 'P100X.DLL'      name 'P100X_GetFloatBuffer';

end.
```

2.6 P100Xu.PAS

```
unit P100Xu;

interface


Function  P100X_AD2F(hex, HiLo, Gain :Word): Single ; StdCall;


implementation

uses math;


//*---------------------------------------------------*
//* Return voltage value or -100.0 if any error occurs  *
//* or parameter is out of range.                      *
//* HiLo : 1 --> High Gain , 0 --> Low Gain             *
//* Gain : 0-3                                          *
//*---------------------------------------------------*
Function  P100X_AD2F(hex, HiLo, Gain :Word): Single ;
Var
   ZeroBase, VoltageRange, FullRange : Single ;

Begin

   ZeroBase  := 2048;
   FullRange := 2048;
   VoltageRange := 10;
   Gain := Gain mod 16;
   If (Gain < 0) Or (Gain > 3) Then
   begin
      P100X_AD2F := -100;
      exit;
   end ;

   If HiLo = 0 Then  //Low-Gain
      Result := ((((hex - ZeroBase) / FullRange) * VoltageRange) / Power(2,
Gain))
   Else
      Result := ((((hex - ZeroBase) / FullRange) * VoltageRange) /
Power(10,Gain));

End;

end.
```

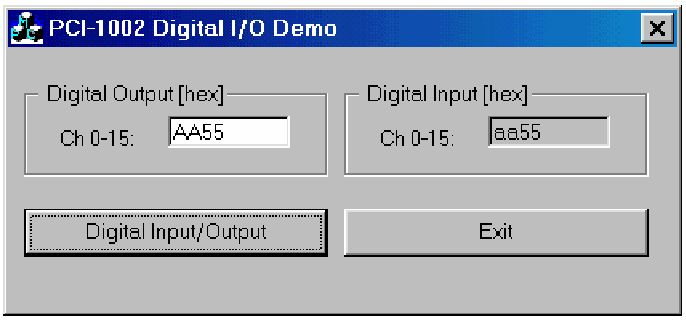
3. Demo Result

3.1 Visual C++



Total Boards in System: 1
Which Board To Active: Board 00
Address & IRQ:
Timer Addr : ec00
DI/DO Addr : ec20
AD Addr : ec30
IRQ Level : a
Configuration Setting :
Card Type: [✓ Low Gain] High Gain
Gain Code: -10 to 10V, -5 to 5V, -2.5 to 2.5V
Channel: CH:00, CH:01, CH:02

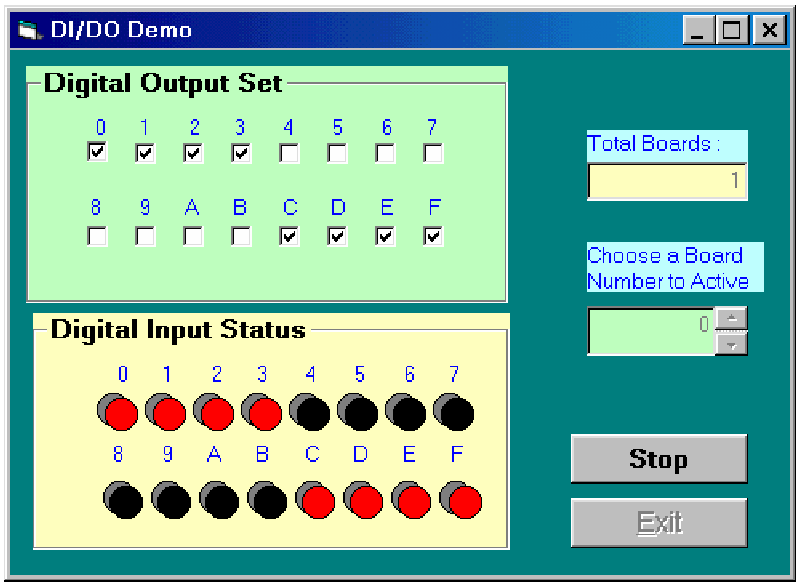
Analog Input with polling demo program



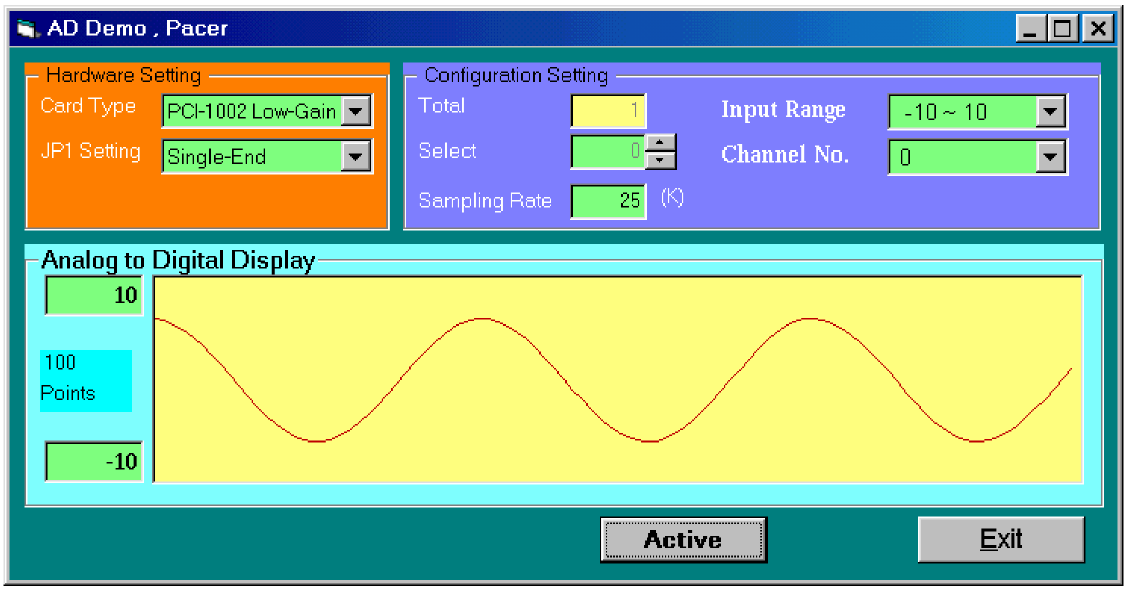
Digital Output [hex]
Ch 0-15: AA55
Digital Input [hex]
Ch 0-15: aa55

Digital I/O with MFC demo program

3.2 Visual Basic

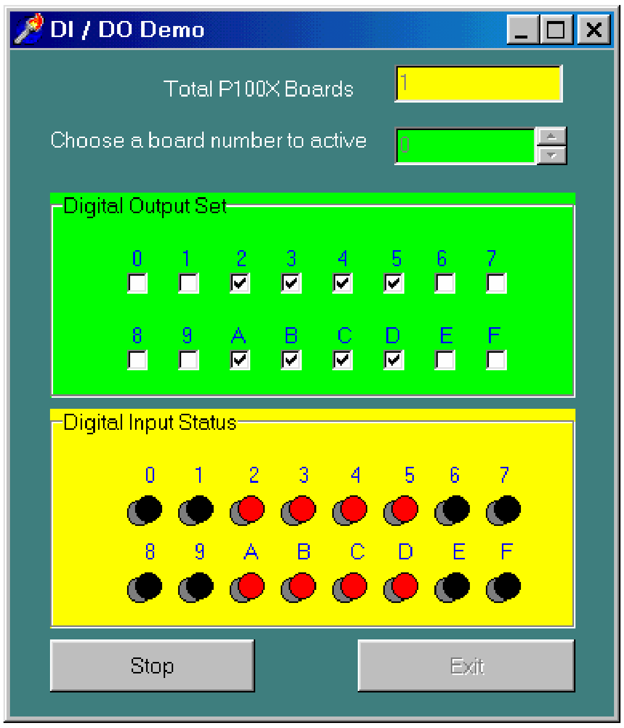


Digital I/O demo program



Analog Input with pacer-trigger demo program

3.3 Delphi

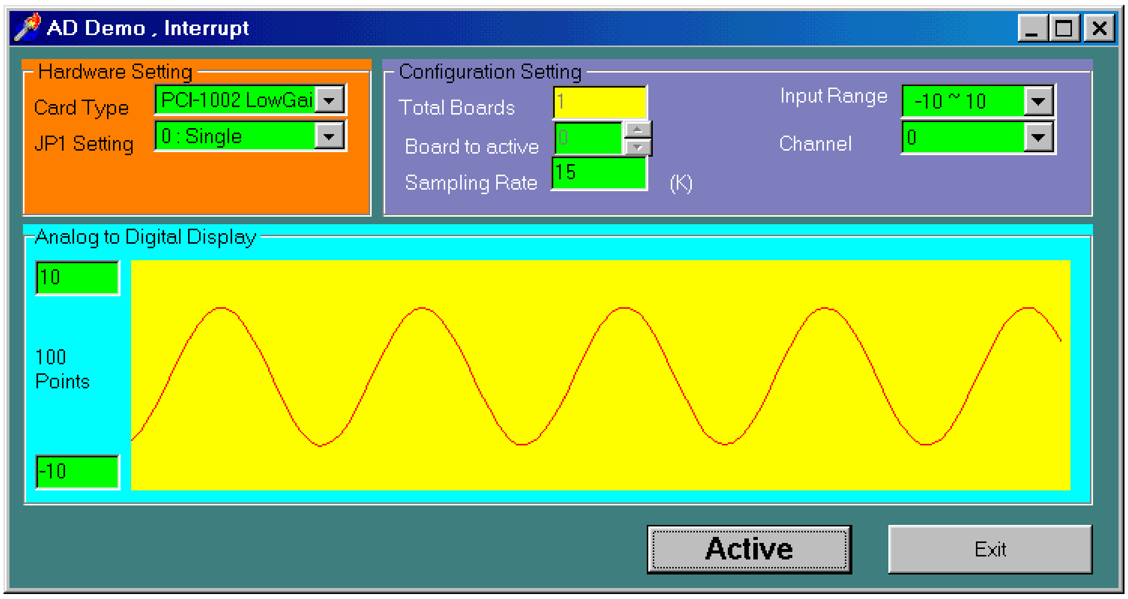


DI / DO Demo

Total P100X Boards: 1

Choose a board number to active: 0

Digital I/O demo program



AD Demo , Interrupt

Hardware Setting

Card Type: PCI-1002 LowGai

JP1 Setting: 0 : Single

Configuration Setting

Total Boards: 1

Board to active: 0

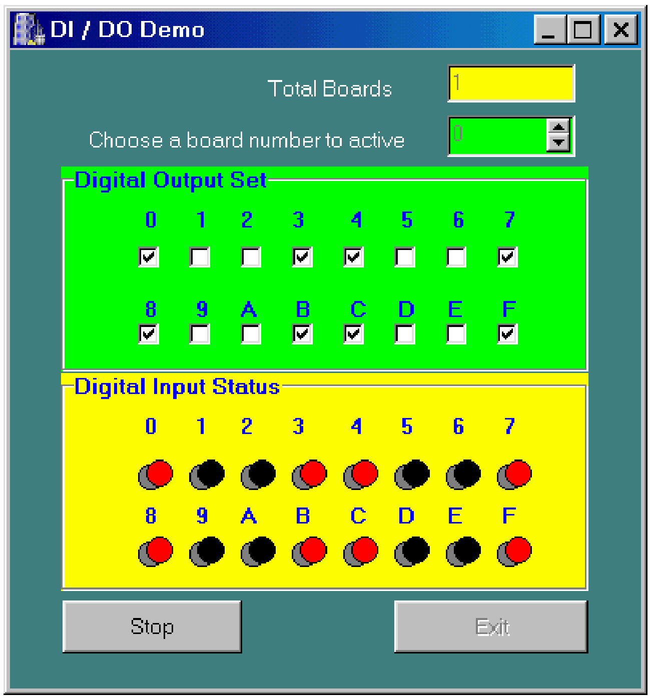
Sampling Rate: 15 (K)

Input Range: -10 ~ 10

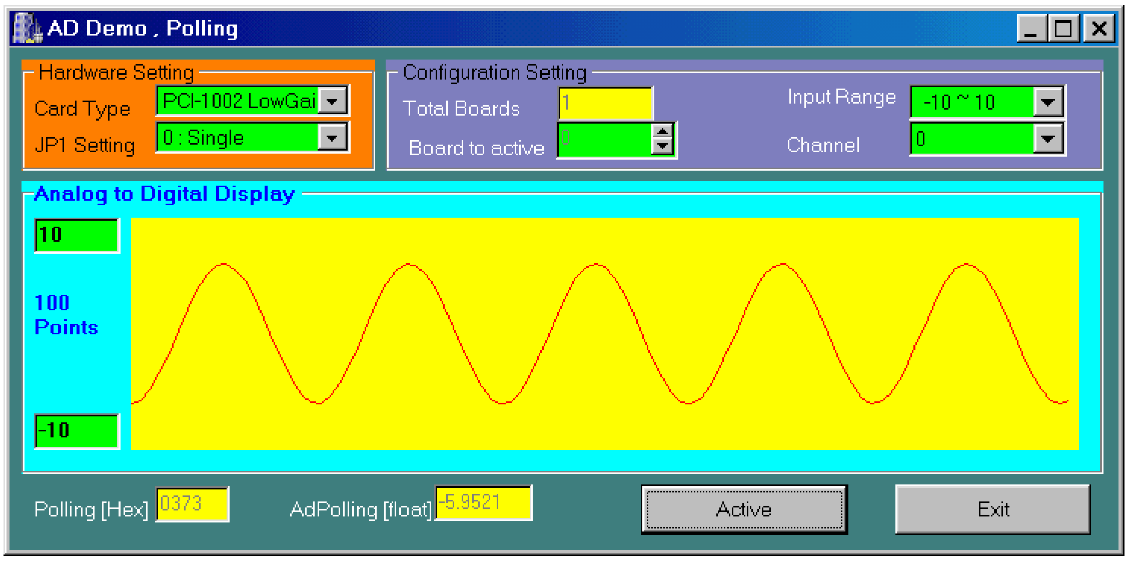
Channel: 0

Analog Input with Interrupt demo program

3.4 Borland C++ Builder



Digital I/O demo program



Analog input with polling demo program

4. Descriptions of Functions

The DLL functions are divided into the following groups:

- Test Functions
- D/I/O Functions
- A/D Fixed-mode Functions
- Driver Functions
- Interrupt Functions

Test Functions:

1. P100X_FloatSub2
2. P100X_ShortSub2
3. P100X_GetDllVersion
4. P100X_GetDriverVersion

D/I/O Functions

1. P100X_DI
2. P100X_DO

A/D Fixed-mode Functions:

1. P100X_SetChannelConfig
2. P100X_Polling
3. P100X_AdPolling
4. P100X_AdsPolling
5. P100X_AdsPacer

Driver Functions:

1. P100X_DriverInit
2. P100X_DriverClose
3. P100X_GetConfigAddressSpace
4. P100X_WhichBoardActive
5. P100X_ActiveBoard
6. P100X_GetIrqNo

Interrupt Functions:

1. P100X_InstallIrq
2. P100X_INT_AdStart
3. P100X_INT_AdStop
4. P100X_GetIntCount
5. P100X_GetBuffer
6. P100X_GetFloatBuffer

The following keywords are used to describe the attributes of function parameters.

Keyword	**Parameter set by user before calling function?**	**Data/value available from this parameter after calling function?**
[Input]	Yes	No
[Output]	No	Yes
[Input, Output]	Yes	Yes

4.1 The Configuration Code Table

OME-PCI-1002L Configuration Code Table

Gain	Bipolar	Max. Switching Frequency	Configuration Code
1	+/- 10V	110 K/S	0x00
2	+/- 5.0V	110 K/S	0x01
4	+/- 2.5V	110 K/S	0x02
8	+/- 1.25V	110 K/S	0x03

OME-PCI-1002H Configuration Code Table

Gain	Bipolar	Max. Switching Frequency	Configuration Code
1	+/- 10V	44 K/S	0x10
10	+/- 1.0V	36 K/S	0x11
100	+/- 0.1V	7 K/S	0x12
1000	+/- 0.01V	0.8 K/S	0x13

4.2 The Test Functions

4.2.1 P100X_FloatSub2

- **Description:**
  Calculates C = fA - fB in **float** format, **float=4 bytes floating point number.** This function is provided to test DLL linkage.

- **Syntax:**
  float P100X_FloatSub2(float fA, float fB);

- **Parameter:**

fA	: [Input] 4 bytes floating point value
fB	: [Input] 4 bytes floating point value

- **Return:**
  Returns the result value (= fA - fB).

4.2.2 P100X_ShortSub2

- **Description :**
  Calculates C = nA - nB in SHORT formats, **SHORT=16 bits signed number.** This function is provided to test DLL linkage.

- **Syntax :**
  short P100X_ShortSub2(Short nA, Short nB);

- **Parameter:**

nA	: [Input] 16-bit value
nB	: [Input] 16-bit value

- **Return:**
  Returns the result value (= nA - nB).

4.2.3 P100X_GetDllVersion

- **Description :**
  Reads the DLL version of the **P100X.DLL**.

- **Syntax :**
  WORD P100X_GetDllVersion(void);

- **Parameter:**
  None

- **Return:**
  Returns the version of the DLL for Device-Driver.
  return=0x200 → Version 2.0

4.2.4 P100X_GetDriverVersion

- **Description :**
  This subroutine will read the software version of the P100X.VxD for Windows 95 or P100X.SYS of Windows NT/2000/XP.

- **Syntax :**
  WORD P100X_GetDriverVersion(WORD *wDriverVersion);

- **Parameter:**

wDriverVersion	: [Output] address of **wDriverVersion,** which contains the version of Device-Driver. wDriverVersion=0x200 → Version 2.0

- **Return:**

P100X_NoError	: OK
P100X_DriverHandleError	: P100X.VxD open error for Windows 95 P100X.SYS open error for Windows NT/2000/XP
P100X_DriverCallError	: call P100X.VxD return error call P100X.SYS return error

4.3 The DI/O Functions

4.3.1 P100X_Di

- **Description :**
  This subroutine will read the 16 bit data from the DI(digital input) port. This function addresses the current active OME-PCI-1002 board. Use the P100X_ActiveBoard(….) to select the active board.

- **Syntax :**
  WORD P100X_Di(WORD *wDi);

- **Parameter:**

wDi	: [Output] address of **wDi**, which contains the 16 bits of digital input data .

- **Return:**

P100X_NoError	: OK
P100X_FindBoardError	: cannot find the OME-PCI-100X board
P100X_ExceedBoardNumber	: invalid board number

4.3.2 P100X_Do

- **Description :**
  This subroutine will write the 16 bit data to the DO(digital output) port. This function addresses the current active OME-PCI-1002 board. Use the P100X_ActiveBoard(….) to select the active board.

- **Syntax :**
  WORD P100X_Do(WORD wDo);

- **Parameter:**

wDo	: [Input] the 16-bit data sent to the digital-output port

- **Return:**

P100X_NoError	: OK
P100X_ExceedBoardNumber	: invalid board number
P100X_FindBoardError	: cannot find OME-PCI-1002 board

4.4 The A/D Fixed-mode Functions

4.4.1 P100X_SetChannelConfig

- **Description :**

  This function will set the A/D channel configuration code. This function will also set the active A/D channel for **P100X_AdPolling**, **P100X_AdsPolling** and **P100X_AdsPacer** functions. The function addresses the current active OME-PCI-1002 board. Use the P100X_ActiveBoard(….) to select the active board.

- **Syntax :**

  WORD P100X_SetChannelConfig(WORD wChannel, WORD wConfig);

- **Parameter:**

wChannel	: [Input] A/D channel number
wConfig	: [Input] Configuration code. Refer to Sec. 3.1 for details.

- **Return:**

P100X_NoError	: OK
P100X_ExceedBoardNumber	: invalid board number
P100X_FindBoardError	: can not find the OME-PCI-1002 board
P100X_AdControllerError	: MagicScan controller hardware handshake error

4.4.2 P100X_Polling

- **Description :**
  Performs a single A/D conversion on the active channel by software polling. The **P100X_SetChannelConfig** subroutine can be used to change the channel or configuration code. Use the P100X_ActiveBoard(….) to select the active board.

- **Syntax :**
  WORD P100X_Polling(word *wAdVal);

- **Parameter:**

wAdVal	: [Output] address of **wAdVal**, which contains the A/D data Data is returned as an integer value in the range 0-4095.

- **Return:**

P100X_NoError	: OK
P100X_ExceedBoardNumber	: invalid board number
P100X_FindBoardError	: can not find the OME-PCI-1002 board
P100X_AdPollingTimeOut	: hardware timeout error

4.4.3 P100X_AdPolling

- **Description :**
  This subroutine will perform a single A/D conversion by polling The **P100X_SetChannelConfig** function can be used to change the channel or configuration code. This function addresses the current active OME-PCI-1002 board. Use the P100X_ActiveBoard(….) function to select the active board.

- **Syntax :**
  WORD P100X_AdPolling(float *fAdVal);

- **Parameter:**

fAdVal	: [Output] address of **fAdVal**, which contains the AD data. The data is automatically converted to voltage based on the settings of **P100X_SetChannelConfig()**.

- **Return:**

P100X_NoError	: OK
P100X_ExceedBoardNumber	: invalid board number
P100X_FindBoardError	: cannot find the OME-PCI-1002 board
P100X_AdPollingTimeOut	: hardware timeout error

4.4.4 P100X_AdsPolling

- **Description :**

  Performs multiple A/D conversions on a single channel by polling. The **P100X_SetChannelConfig** subroutine can be used to change the channel or configuration code. This function addresses the current active OME-PCI-1002 board. Use P100X_ActiveBoard(….) to select the active board.

  Since software polling can be interrupted by the operating system, the P100X_AdsPacer function is recommended when precisely reconstructing the waveform is desired.

- **Syntax :**

  WORD P100X_AdsPolling(float fAdVal[], WORD wNum);

- **Parameter:**

fAdVal	: [Output] starting address of the A/D data buffer(Array of float) The data is converted to voltage based on the setting of the **P100X_SetChannelConfig()** function. The user must allocate sufficient space for the buffer. The user can access the data after calling the function.
wNum	: [Input] number of A/D conversions to be performed.

- **Return:**

P100X_NoError	: OK
P100X_ExceedBoardNumber	: Invalid board number
P100X_FindBoardError	: Can not find the OME-PCI-1002 board
P100X_AdPollingTimeOut	: Hardware timeout error

4.4.5 P100X_AdsPacer

- **Description :**
  This function performs multiple A/D conversions on a single channel by pacer trigger. The **P100X_SetChannelConfig** function can be used to change the channel or configuration code. The function addresses the current active OME-PCI-1002 board. Use P100X_ActiveBoard(….) to select the active board.

- **Syntax :**
  WORD P100X_AdsPacer(float fAdVal[], WORD wNum, WORD wSample);

- **Parameter:**

fAdVal	: [Output] Address of the A/D data buffer (Array of WORD), data will be converted to voltage based on the settings of **P100X_SetChannelConfig()**. The user must allocate sufficient space for the buffer. The user cans access the data after calling the function.
wNum	: [Input] number of AD conversions to be performed.
wSample	: [Input] **AD sampling rate = 2M/wSample.**

- **Return:**

P100X_NoError	: OK
P100X_ExceedBoardNumber	: invalid board number
P100X_FindBoardError	: cannot find the OME-PCI-1002 board
P100X_AdPollingTimeOut	: hardware timeout error

4.5 Driver Functions

4.5.1 P100X_DriverInit

- **Description :**
  This function will detect all OME-PCI-1002 boards installed in the system. This function must be called once before the other functions are called.

- **Syntax :**
  WORD P100X_DriverInit(WORD *wTotalBoard);

- **Parameter:**

wTotalBoard	: [Output] Address of **wTotalBoard,** which will contain the number of OME-PCI-1002 boards in the system. wTotalBoard=0 → Not found. wTotalBoard=1 → one OME-PCI-1002 card in the system wTotalBoard=n → n OME-PCI-1002 cards in the system

- **Return:**

P100X_NoError	: OK
P100X_NoFoundBoard	: can not detect any OME-PCI-1002
P100X_FindBoardError	: handshake check error
P100X_DriverHandleError	: the P100X.VxD .open error for Windows 95 the P100X.SYS .open error for Windows NT
P100X_DriverCallError	: call P100X.VxD return error call P100X.SYS return error

4.5.2 P100X_DriverClose

- **Description :**
  Releases all system resources. This function should be called before terminating the program.

- **Syntax :**
  void P100X_DriverClose(void);

- **Parameter:**
  None

- **Return:**
  None

4.5.3 P100X_GetConfigAddressSpace

- **Description :**
  Get the I/O address of OME-PCI-1002 board n. This function is for debugging purposes. It is not normally necessary to call this function.

- **Syntax :**

```
WORD P100X_GetConfigAddressSpace(WORD wBoardNo,
            WORD *wAddrTimer, WORD *wAddrDio, WORD *wAddrAd);
```

- **Parameter:**

wBoardNo	: [Input] OME-PCI-1002 board number
wAddrTimer, wAddrDio, wAddrAd	: [Output] Address of wAddrTimer, wAddrDio, wAddrAD stores the address of the Timer, DI/DO and A/D. Please refer to Hardware manual for additional details.

- **Return:**

P100X_NoError	: OK
P100X_FindBoardError	: handshake check error
P100X_ExceedBoardError	: wBoardNo is invalidd

4.5.4 P100X_WhichBoardActive

- **Description:**
  Returns the board number of the active board.

- **Syntax:**

```
WORD P100X_WhichBoardActive(void);
```

- **Parameter:**
  None

- **Return:**
  Returns the board number of the active board.

4.5.5 P100X_ActiveBoard

- **Description:**
  This function makes a board active. This function must be called once before the D/I/O, A/D or D/A functions are called.

- **Syntax:**
  WORD P100X_ActiveBoard(WORD wBoardNo);

- **Parameter:**

wBoardNo	[Input]The board number of the board to make active

- **Return:**

P100X_NoError	: OK
P100X_ExceedBoardError	: wBoardNo is invalid

4.5.6 P100X_GetIrqNo

- **Description:**
  This function will get the IRQ number of the active OME-PCI-1002 board installed in the system. This function is not normally used by user applications.

- **Syntax:**
  WORD P100X_GetIrqNo(WORD *IrqNo);

- **Parameter:**

IrqNo	: [Output] Address of IrqNo, which contains the IRQ No allocated by the system.

- **Return:**

P100X_NoError	: OK

4.6 The Interrupt Functions

4.6.1 P100X_InstallIrq

- **Description :**
  This subroutine will install the interrupt handler for a specific IRQ n. and set the maximum number of interrupts. Refer to section 3.6.7.for more details on using interrupts

- **Syntax :**
  WORD P100X_InstallIrq(HANDLE *hEvent, DWORD dwCount);

- **Parameter:**

hEvent	: [Input] The user must use the CreateEvent() to create the Event object and obtain its handle and pass the handle to this function.
dwCount	: [Input] Maximum number of counts for interrupt transfer.

- **Return:**

P100X_NoError	: successful
P100X_InstallIrqError	: failed installing the IRQ handler.

4.6.2 P100X_GetIntCount

- **Description :**
  This subroutine will read the interrupt transfer count.

- **Syntax :**
  WORD P100X_GetIntCount(DWORD *dwVal)

- **Parameter:**

dwVal	: [Output] the address of dwVal, which contains the value of interrupt transferred count.

- **Return:**

P100X_NoError	: successful
P100X_GetIntCountError	: fail get interrupt count.

4.6.3 P100X_INT_AdStart

- **Description :**
  This subroutine will start the interrupt transfer for a specific A/D channel, set the gain code and sample rate.

- **Syntax :**
  WORD P100X_INT_AdStart(WORD Ch, WORD Gain, WORD wFreqDiv)

- **Parameter:**

Ch	: [Input] the A/D channel.
Gain	: [Input] the Gain, refer to Section 3.1
wFreqDiv	: [Input] the sampling rate is 2M/(wFreqDiv)

- **Return:**

P100X_NoError	: successful
P100X_INTStartError	: failure

4.6.4 P100X_INT_AdStop

- **Description :**
  This subroutine will stop the interrupt transfer and remove the installed interrupt handler.

- **Syntax :**
  WORD P100X_ INT_AdStop(void)

- **Parameter:**
  None

- **Return:**

P100X_NoError	: successful
P100X_INTStopError	: failure

4.6.5 P100X_GetBuffer

- **Description :**
  This subroutine will copy the transferred interrupted data into the user's buffer (in word format).

- **Syntax :**
  WORD P100X_GetBuffer(DWORD dwNum, WORD wBuffer[])

- **Parameter:**

wNum	: [Input] The total number to transfer to User's Buffer.
wBuffer	: [Output] The address of wBuffer (Array of word) that will contain the hex A/D value. The user must allocate sufficient space for this buffer. This function will fill the buffer with the data. The user can access the data after calling this function.

- **Return:**

P100X_NoError	: successful
P100X_GetBufferError	: failure

4.6.6 P100X_GetFloatBuffer

- **Description :**
  This subroutine will copy the data into the user's buffer (in floating-point format).

- **Syntax :**
  WORD P100X_GetFloatBuffer(DWORD dwNum, float fAdVal[])

- **Parameter:**

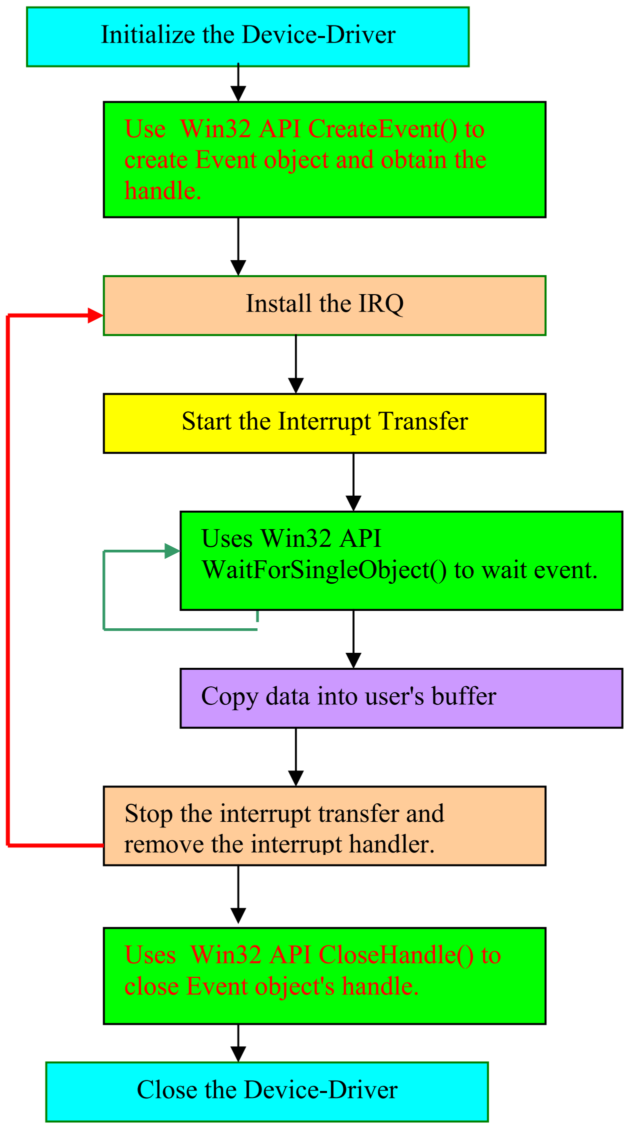
wNum	: [Input] The total number of data points to transfer to the user'sbuffer.
fAdVal	: [Output] Address of fAdVals (Array of float) that will contain the data as a voltage value(floating-point). The user must allocate sufficient space for the buffer. This function will fill the buffer with the data. The user cans access the data after calling this function.

- **Return:**

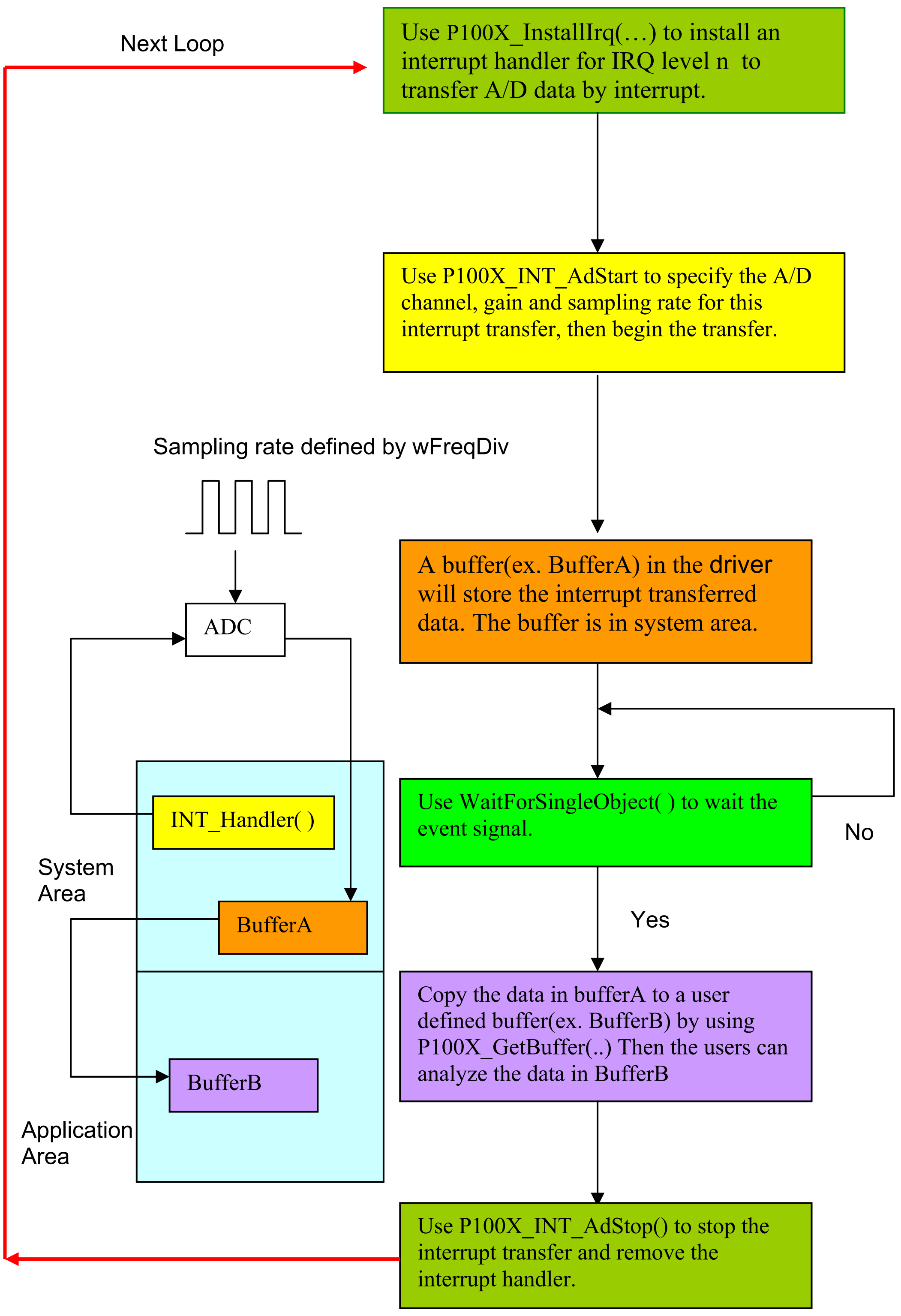
P100X_NoError	: successful
P100X_GetBufferError	: failure

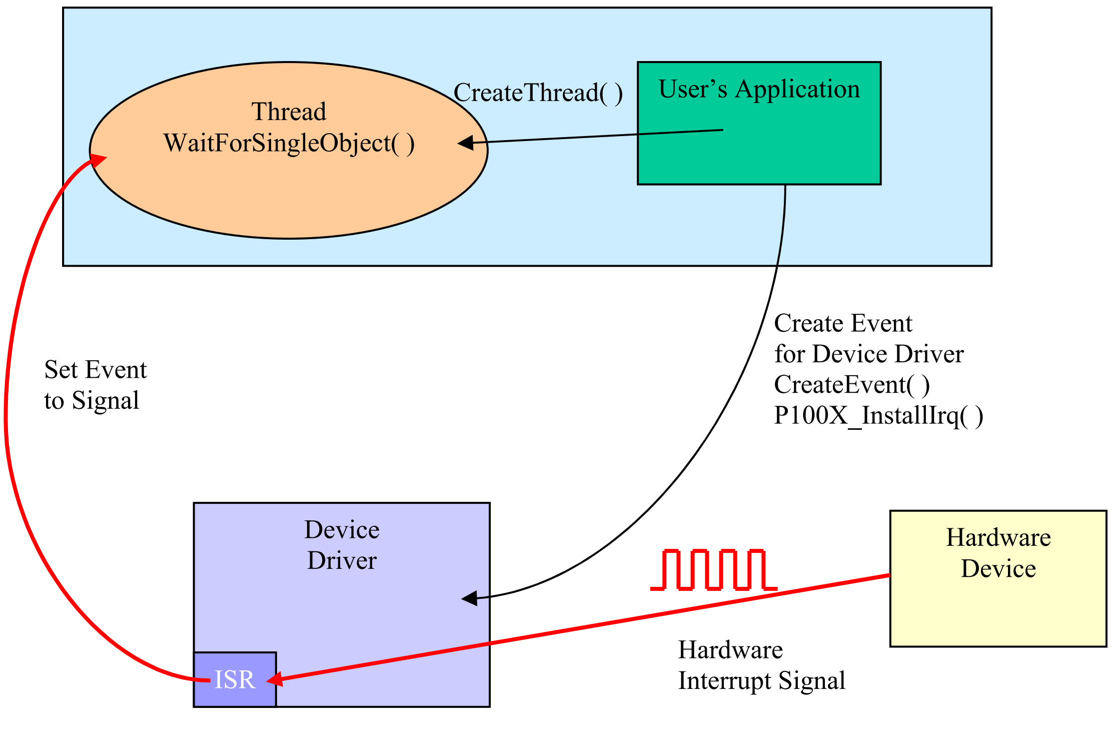
4.6.7 Architecture of Interrupt mode

The flow chart below shows the steps for programming the A/D interrupt functions:


Initialize the Device-Driver
Use Win32 API CreateEvent() to create Event object and obtain the handle.
Install the IRQ
Start the Interrupt Transfer
Uses Win32 API WaitForSingleObject() to wait event.
Copy data into user's buffer
Stop the interrupt transfer and remove the interrupt handler.
Uses Win32 API CloseHandle() to close Event object's handle.
Close the Device-Driver

```
P100X_DriverInit( )

    …….
    CreateEvent(  )

    …….

    P100X _InstallIrq( … )

    ………

    P100X _INT_AdStart( … )

    ……

          WaitForSingleObject( …. )

          ………..

          P100X _GetBuffer( …. )

    ……

    P100X _INT_AdStop

     ………

     CloseHandle( )

     ……….

P100X _DriverClose( )
```


Next Loop
Use P100X_InstallIrq(…) to install an interrupt handler for IRQ level n to transfer A/D data by interrupt.
Use P100X_INT_AdStart to specify the A/D channel, gain and sampling rate for this interrupt transfer, then begin the transfer.
Sampling rate defined by wFreqDiv
ADC
A buffer(ex. BufferA) in the driver will store the interrupt transferred data. The buffer is in system area.
INT_Handler( )
System Area
BufferA
Use WaitForSingleObject( ) to wait the event signal.
No
Yes
BufferB
Application Area
Copy the data in bufferA to a user defined buffer(ex. BufferB) by using P100X_GetBuffer(..) Then the users can analyze the data in BufferB
Use P100X_INT_AdStop() to stop the interrupt transfer and remove the interrupt handler.



Please refer to the following Windows API functions:

The following descriptions of these functions were copied from MSDN. Refer to MSDN for complete details.

CreateEvent( )

The CreateEvent function creates or opens a named or unnamed event object.

```
HANDLE CreateEvent(
    // pointer to security attributes
    LPSECURITY_ATTRIBUTES lpEventAttributes,
    BOOL bManualReset,      // flag for manual-reset event
    BOOL bInitialState,     // flag for initial state
    LPCTSTR lpName          // pointer to event-object name
);
```

CreateThread( )

The CreateThread function creates a thread to execute within the virtual address space of the calling process.

To create a thread that runs in the virtual address space of another process, use the CreateRemoteThread function.

```
HANDLE CreateThread(
        // pointer to security attributes
        LPSECURITY_ATTRIBUTES lpThreadAttributes,
        DWORD dwStackSize,              // initial thread stack size
         // pointer to thread function
        LPTHREAD_START_ROUTINE lpStartAddress,
        LPVOID lpParameter,             // argument for new thread
        DWORD dwCreationFlags,          // creation flags
        LPDWORD lpThreadId              // pointer to receive thread ID
);
```

WaitForSingleObject( )

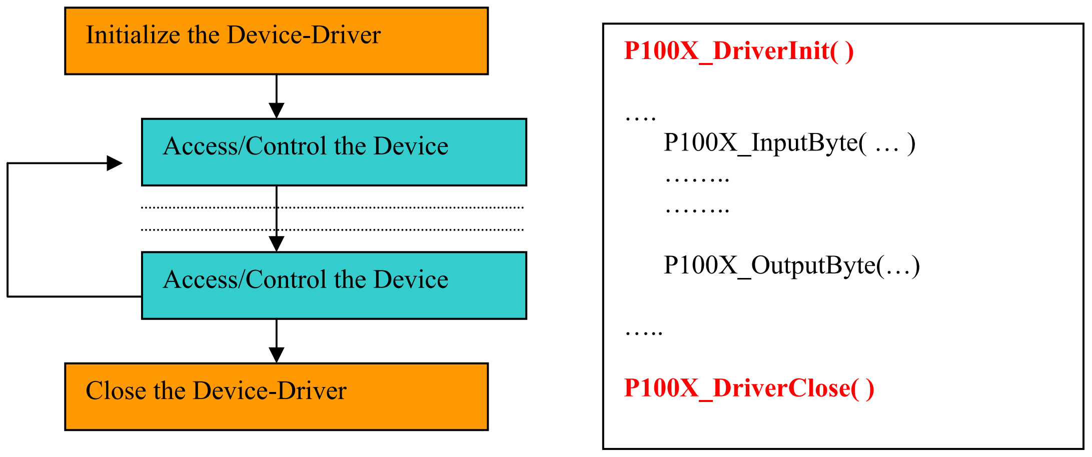
The WaitForSingleObject function returns when one of the following occurs:

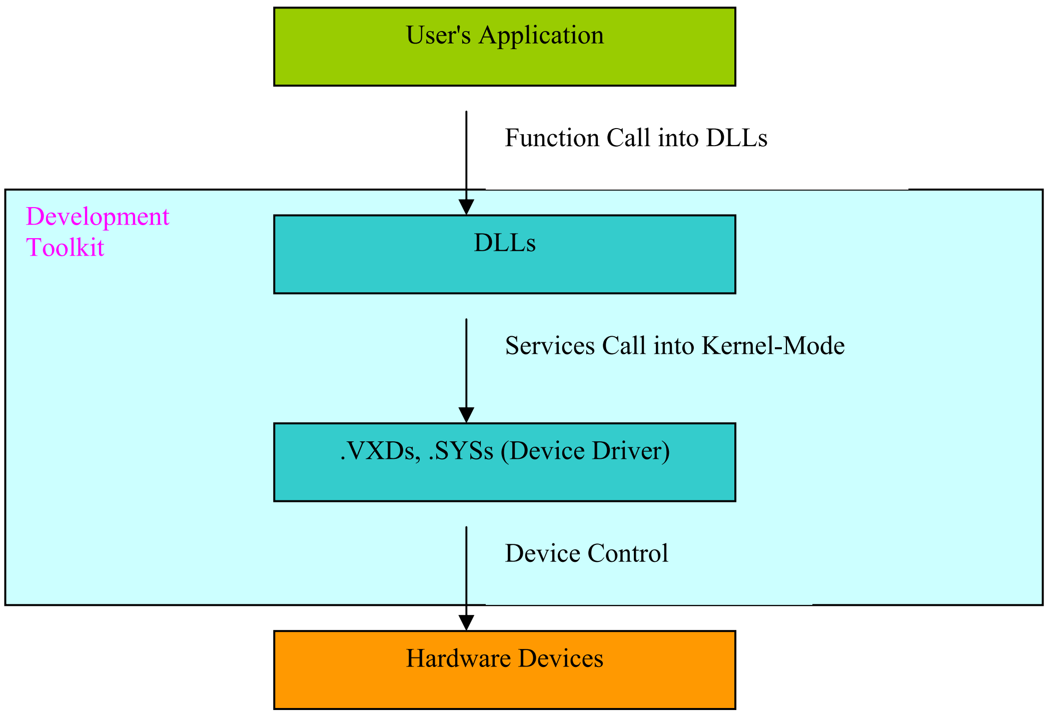
- The specified object is in the signaled state.
- The time-out interval elapses.

To enter an alert-able wait state, use the WaitForSingleObjectEx function. To wait for multiple objects, use the WaitForMultipleObjects.

```
DWORD WaitForSingleObject(
            HANDLE hHandle,             // handle to object to wait for
            DWORD dwMilliseconds   // time-out interval in
      milliseconds
);
```

5. Program Architecture





6. Reporting Problems

Technical support is provided at no charge you may contact us by telephone or email at

Telephone: 1-800-872-9436

Email: das@omega.com

When reporting problems, please include the following information:

1) Is the problem reproducible? If so, how?

2) What platform and version are you using? For example, Windows 3.1, Windows for Workgroups, Windows NT 4.0, etc.

3) Part number of the product that you are using?

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 and/or hardware devices, please provide a complete description of those items.

6) Other comments relative to this problem. Your suggestions are welcome.

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 <u>WARRANTY</u> 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 <u>NON-WARRANTY</u> 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.

M3928 / 0203