PCI-1002

Software Manual
[for Windows 95/98/NT/2000/XP/2003]

Warranty

All products manufactured by ICP DAS are warranted against defective materials for a period of one year from the date of delivery to the original purchaser.

Warning

ICP DAS assumes no liability for damage consequent to the use of this product. ICP DAS reserves the right to change this manual at any time without notice. The information furnished by ICP DAS is believed to be accurate and reliable. However, ICP DAS assumes no responsibility for its use, or for any infringements of patents or other rights of third parties resulting from its use.

Copyright

Trademark

The names used for identification only may be registered trademarks of their respective companies.

Table of Contents

1. Introduction

The **PCI-1002 SDK** is a collection of DLLs and device-driver for Windows 95/98/NT 4.0 and Windows 2000/XP/2003 application. These DLLs are 32 bits and can be called by Visual C++, BC++, Visual BASIC, Delphi and LabVIEW.

The PCI-1002 SDK consists of these DLLs and device driver:

- P100X.DLL →PCI-1002 card DLL functions
- P100X.VXD →PCI-1002 Device driver for Windows 95/98/Me
- P100X.SYS →PCI-1002 Device driver for Windows NT/2K/XP

These DLLs can perform a variety of data acquisition operations as follows:

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

1.1 References

Please refer to the following user manuals:

CD:\NAPDOS\PCI\Manual\

- **Software_Install_Guide_in_Win32.pdf:**
  To install the software package under Windows 95/98/ME/2K/XP/NT.

- **Calling_DLL_functions_in_VB_VC_Delphi_BCB.pdf:**
  To call the DLL functions with VC++6, VB6, Delphi4 and Borland C++ Builder 4.

- **TroubleShooting_PCI_ISA_in_Win32_Resource_Conflict.pdf:**
  To check the resources I/O Port address, IRQ number and DMA number for add-on cards under Windows 95/98/NT/2000.

2. Declaration Files

Please refer to user manual "
Calling_DLL_functions_in_VB_VC_Delphi_BCB.pdf ".

```
|--\Demo
|       |
|       |--\VB6                     ← for Visual Basic 6.0
|       |       |--\P100X.BAS       ← Declaration file for Visual Basic
|       |
|       |
|       |--\VC6                     ← for Visual C++ 6.0
|       |       |--\P100X.H         ← Header file
|       |       |--\P100X.Lib       ← Import Library file for VC6 only
|       |
|       |
|       |--\Delphi4                 ← for Delphi 4.0
|       |       |--\P100X.PAS       ← Declaration file
|       |
|       |
|       |--\BCB4                    ← for Borland C++ Builder 4.0
|               |--\P100X.H         ← Header file
|               |--\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_InvalidateDelay           10
#define P100X_DelayTimeOut              11
#define P100X_InvalidateData            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
#define P100X_RemoveIrqError            21
#define P100X_ClearIntCountError        22
#define P100X_BoardNotFound             23
#define P100X_InstallEventError         24
#define P100X_GetDataFromDriverFailed   25



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_DelayTick(WORD wDownCount);

EXPORTS void       CALLBACK P100X_DigitOut(DWORD wOutData);
EXPORTS WORD   CALLBACK P100X_DigitIn(DWORD *wDiData);

EXPORTS WORD   CALLBACK P100X_SetChConfig
        (WORD wAdChannel, WORD wConfig);
EXPORTS WORD   CALLBACK P100X_AdPollingH (WORD *wAdVal);
EXPORTS WORD   CALLBACK P100X_AdPollingF (float *fAdVal);
EXPORTS WORD   CALLBACK P100X_AdMultiPollingF
        (float fAdVal[], WORD wNum);

EXPORTS WORD   CALLBACK P100X_AdMultiPacerF
        (float fAdVal[], WORD wNum,  WORD wSamplingDiv);


EXPORTS WORD   CALLBACK P100X_IntInstall
        (HANDLE *hEvent, DWORD dwCount);
EXPORTS WORD   CALLBACK P100X_IntGetBufferH
        (DWORD dwNum, WORD wBuf[]);
EXPORTS WORD   CALLBACK P100X_IntGetBufferF
        (DWORD dwNum, float fAdVal[]);

EXPORTS WORD   CALLBACK P100X_IntGetCount (DWORD *dwVal);
EXPORTS WORD   CALLBACK P100X_IntStart
        (WORD Ch, WORD Gain, WORD wFreqDiv);
EXPORTS WORD   CALLBACK P100X_IntStartExTrigger
(WORD Ch, WORD Gain);

EXPORTS WORD   CALLBACK P100X_IntStop ();
EXPORTS WORD   CALLBACK P100X_IntRemove();
```

2.2 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_InvalidateDelay = 10
Global Const P100X_DelayTimeOut = 11
Global Const P100X_InvalidateData = 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
Global Const P100X_RemoveIrqError = 21
Global Const P100X_ClearIntCountError = 22
Global Const P100X_BoardNotFound = 23
Global Const P100X_InstallEventError = 24

' 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_DelayTick Lib "P100X.DLL" (ByVal wDownCount
As_ Integer) As Integer

' Function of DI/DO

Declare Sub P100X_DigitOut Lib "P100X.DLL" (ByVal wOutData As Long)
Declare Function P100X_DigitIn Lib "P100X.DLL" (wDiData As Long) As_
Integer

' Function of AD

Declare Function P100X_SetChConfig Lib "P100X.DLL" _
      (ByVal wAdChannel As Integer, ByVal wConfig As Integer) As Integer
Declare Function P100X_AdPollingH Lib "P100X.DLL" (wAdVal As Integer)
_As Integer
Declare Function P100X_AdPollingF Lib "P100X.DLL" (fAdVal As Single) As_
Integer
Declare Function P100X_AdMultiPollingF Lib "P100X.DLL" _
      (fAdVal As Single, ByVal wNum As Integer) As Integer
Declare Function P100X_AdMultiPacerF Lib "P100X.DLL" (fAdVal As Single, _
      ByVal wNum As Integer, ByVal wSamplingDiv As Integer) As Integer

' Function of Interrupt

Declare Function P100X_IntInstall Lib "P100X.DLL" _
      (hEvent As Long, ByVal dwCount As Long) As Integer
Declare Function P100X_IntGetBufferH Lib "P100X.DLL" _
      (ByVal dwNum As Long, wBuf As Integer) As Integer
Declare Function P100X_IntGetBufferF Lib "P100X.DLL" _
      (ByVal dwNum As Long, fAdVal As Single) As Integer
Declare Function P100X_IntStart Lib "P100X.DLL" _
      (ByVal Ch As Integer, ByVal Gain As Integer, ByVal wFreqDiv As Integer)
As Integer
Declare Function P100X_IntStartExTrigger Lib "P100X.DLL" _
      (ByVal Ch As Integer, ByVal Gain As Integer) As Integer
Declare Function P100X_IntStop Lib "P100X.DLL" () As Integer
Declare Function P100X_IntGetCount Lib "P100X.DLL" (dwVal As Long) As_
Integer
Declare Function P100X_IntRemove Lib "P100X.DLL" () As Integer
```

2.2 P100X.PAS

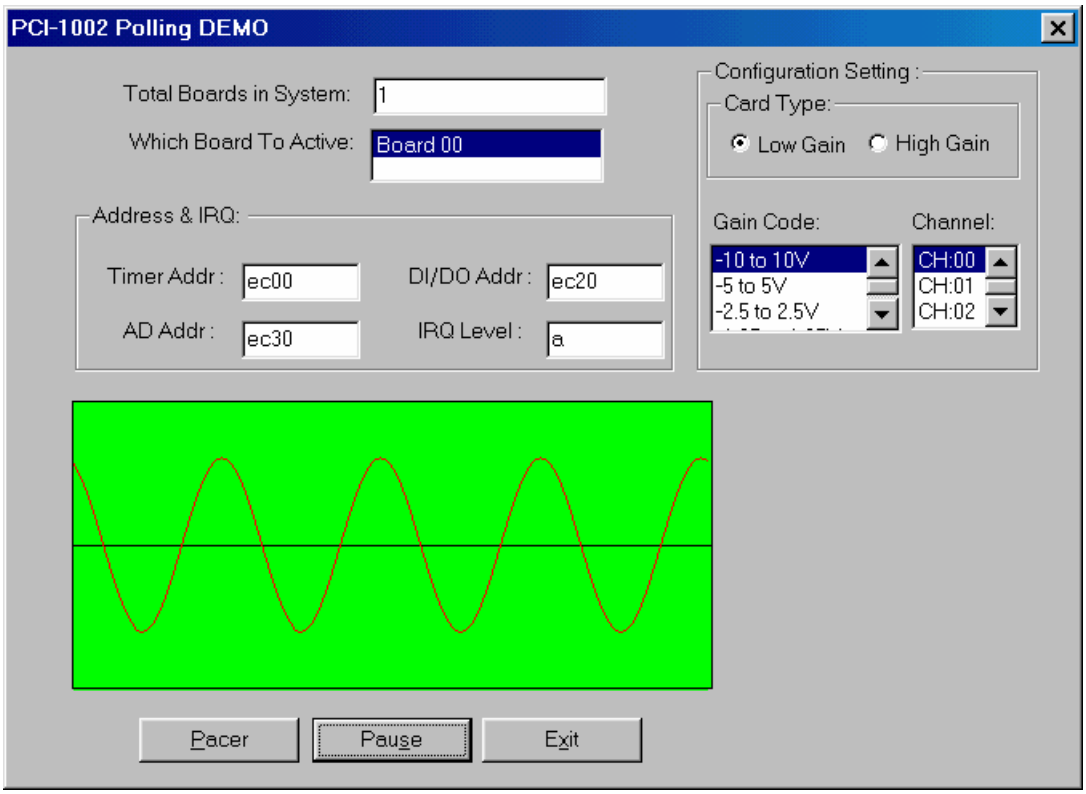
```
unit P100X;

interface

uses
   windows;

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_InvalidateDelay    =  10;
 P100X_DelayTimeOut       =  11;
 P100X_InvalidateData     =  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;
 P100X_RemoveIrqError     =  21;
 P100X_ClearIntCountError  =  22;
 P100X_BoardNotFound      =  23;
 P100X_InstallEventError   =  24;

// 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_DelayTick
        (wDownCount:Word):Word; StdCall;  // 1 tick = 0.25us
// Function of DI/DO
procedure P100X_DigitOut(wOutData:DWord); stdCall;
function  P100X_DigitIn(var wDiData:DWord):WORD ; stdCall;

// Function of AD
function  P100X_SetChConfig
        (wAdChannel:Word; wConfig:Word):WORD ; stdCall;
function  P100X_AdPollingH(var wAdVal:Word):WORD ; stdCall;
function  P100X_AdPollingF(var fAdVal:Single):WORD ; stdCall;
function  P100X_AdMultiPollingF
        (fAdVal:PSingle; wNum:Word):WORD ; stdCall;
function  P100X_AdMultiPacerF
        (fAdVal:PSingle; wNum:Word; wSamplingDiv:Word ):WORD ; stdCall;

// Function of Interrupt
function  P100X_IntInstall
        (Var hEvent:THandle; dwCount: LongInt):WORD ; stdCall;
function  P100X_IntGetBufferH(dwNum:LongInt;wBuf:PWord):WORD ; stdCall;
function  P100X_IntGetBufferF(dwNum:LongInt; fAdVal:PSingle):Word; StdCall;
function  P100X_IntStart
        (Ch:WORD; Gain:WORD; wFreqDiv:Word):WORD ; stdCall;
function  P100X_IntStartExTrigger(Ch:WORD; Gain:WORD):WORD ; stdCall;
function  P100X_IntStop:WORD ; stdCall;
function  P100X_IntGetCount(var dwVal:LongInt):WORD ; stdCall;
function  P100X_IntRemove: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';
```

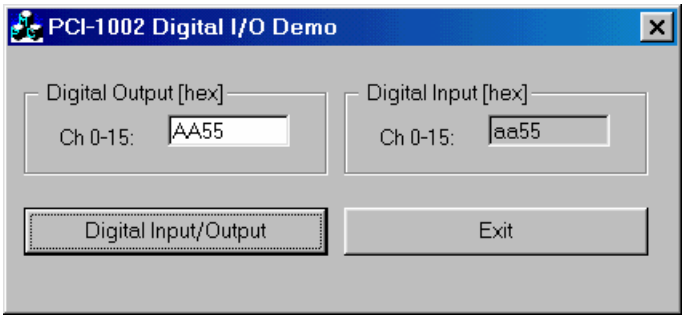
```
function        100X_DriverInit;  external 'P100X.DLL' name 'P100X_DriverInit';
procedure       100X_DriverClose;
                external 'P100X.DLL'         name 'P100X_DriverClose';
function        100X_GetIrqNo;  external 'P100X.DLL' name 'P100X_GetIrqNo';
function        100X_GetConfigAddressSpace;
                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  P100X_DelayTick;   external 'P100X.DLL' name 'P100X_DelayTick';

procedure P100X_DigitOut;    external 'P100X.DLL' name 'P100X_DigitOut';
function  P100X_DigitIn;     external 'P100X.DLL' name 'P100X_DigitIn';


function  P100X_SetChConfig;        external 'P100X.DLL' name
'P100X_SetChConfig';
function  P100X_AdPollingH;         external 'P100X.DLL' name
'P100X_AdPollingH';
function  P100X_AdPollingF;         external 'P100X.DLL' name
'P100X_AdPollingF';
function  P100X_AdMultiPollingF;    external 'P100X.DLL' name
'P100X_AdMultiPollingF';
function  P100X_AdMultiPacerF;      external 'P100X.DLL' name
'P100X_AdMultiPacerF';


function  P100X_IntInstall;         external 'P100X.DLL' name 'P100X_IntInstall';
function  P100X_IntStart;           external 'P100X.DLL' name 'P100X_IntStart';
function  P100X_IntStartExTrigger; external 'P100X.DLL' name
'P100X_IntStartExTrigger';
function  P100X_IntStop;            external 'P100X.DLL' name 'P100X_IntStop';
function  P100X_IntGetCount;        external 'P100X.DLL' name
'P100X_IntGetCount';
function  P100X_IntGetBufferH;      external 'P100X.DLL' name
'P100X_IntGetBufferH';
function  P100X_IntGetBufferF;      external 'P100X.DLL' name
'P100X_IntGetBufferF';
function  P100X_IntRemove;          external 'P100X.DLL' name
'P100X_IntRemove';

end.
```

3. Demo Result

3.1 Visual C++

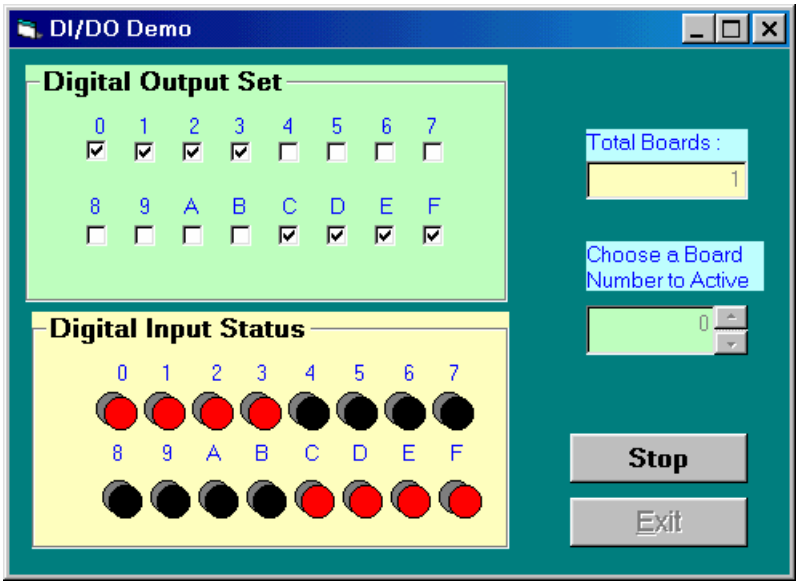


Analog Input with polling demo program

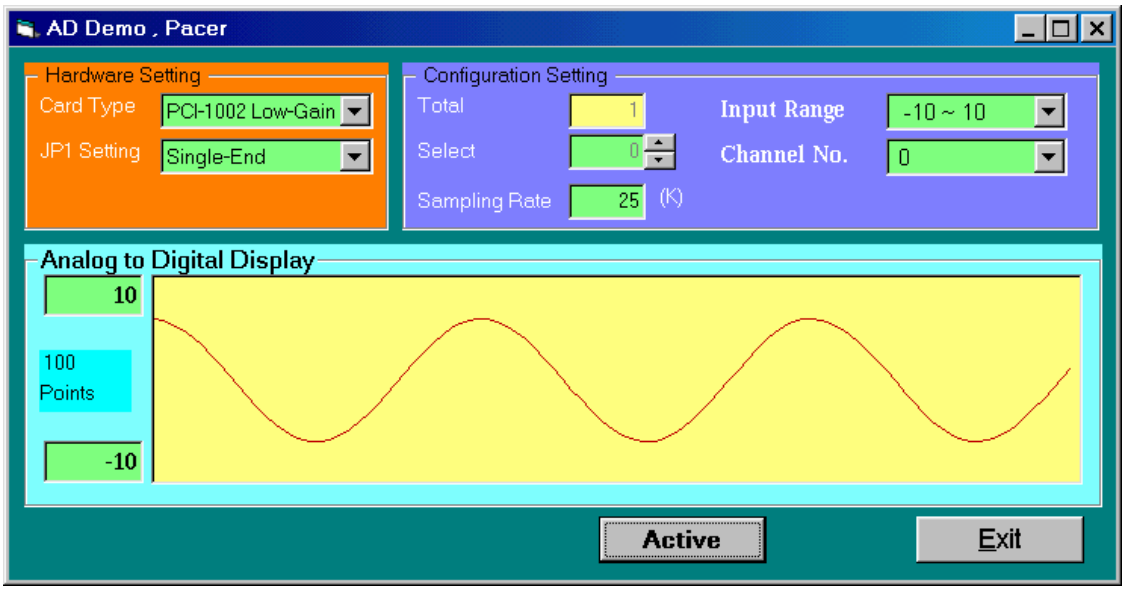


Digital I/O with MFC demo program

3.2 Visual Basic

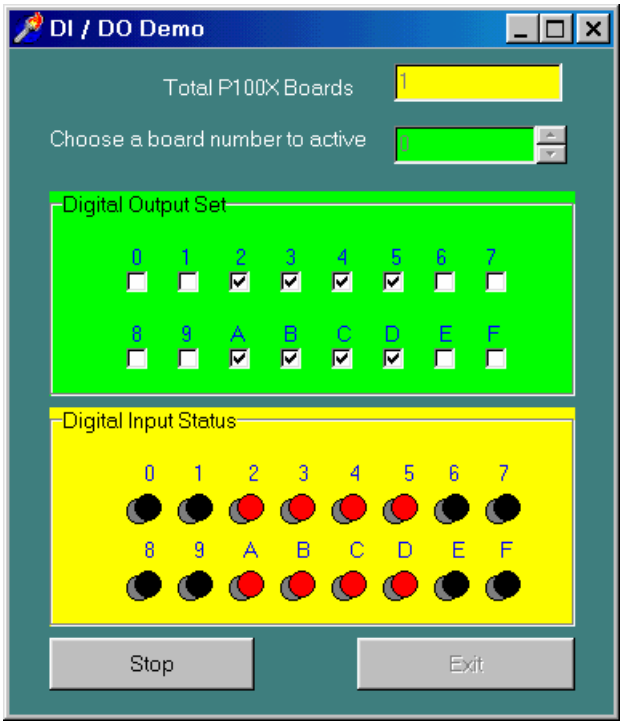


Digital I/O demo program

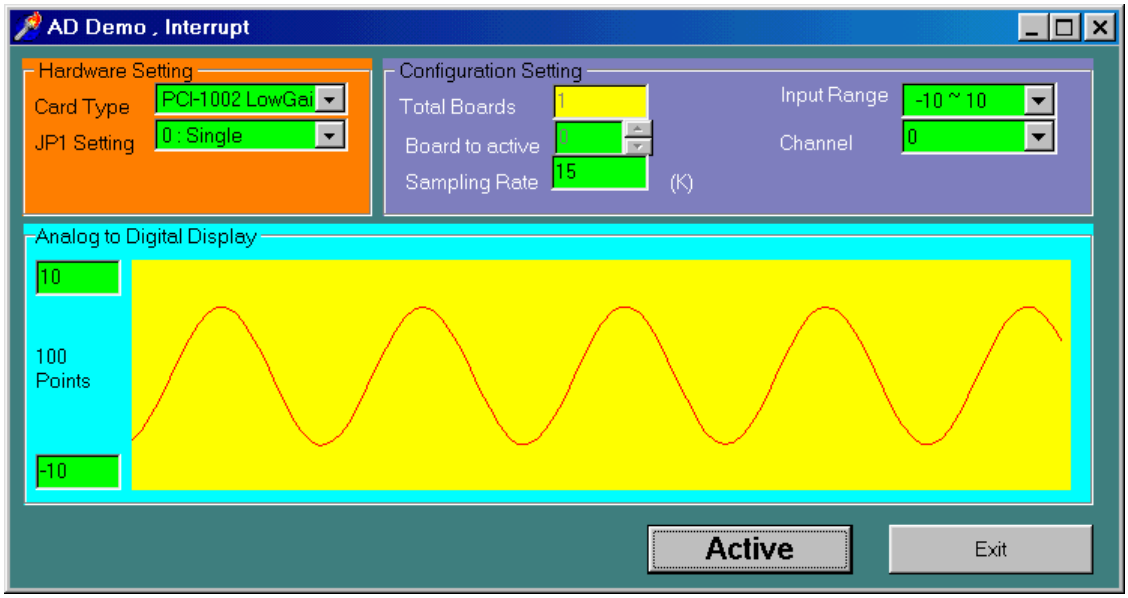


Analog Input with pacer-trigger demo program

3.3 Delphi

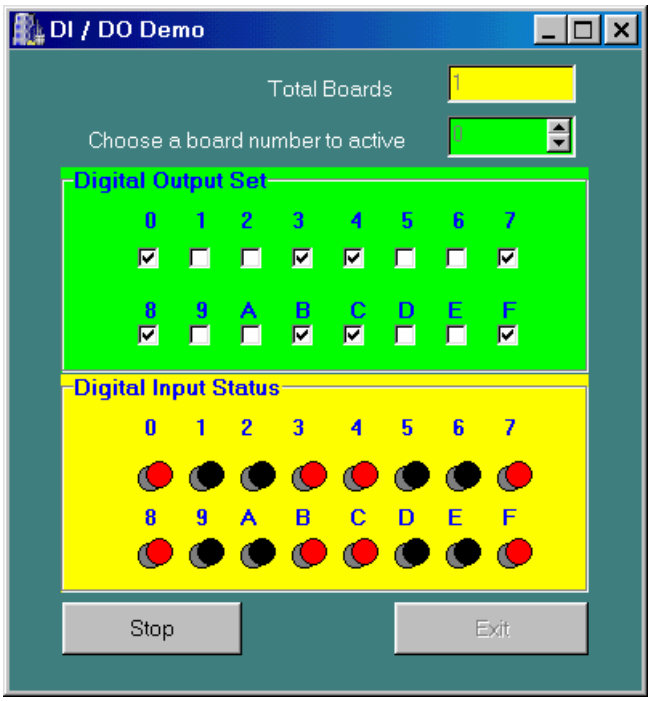


Digital I/O demo program

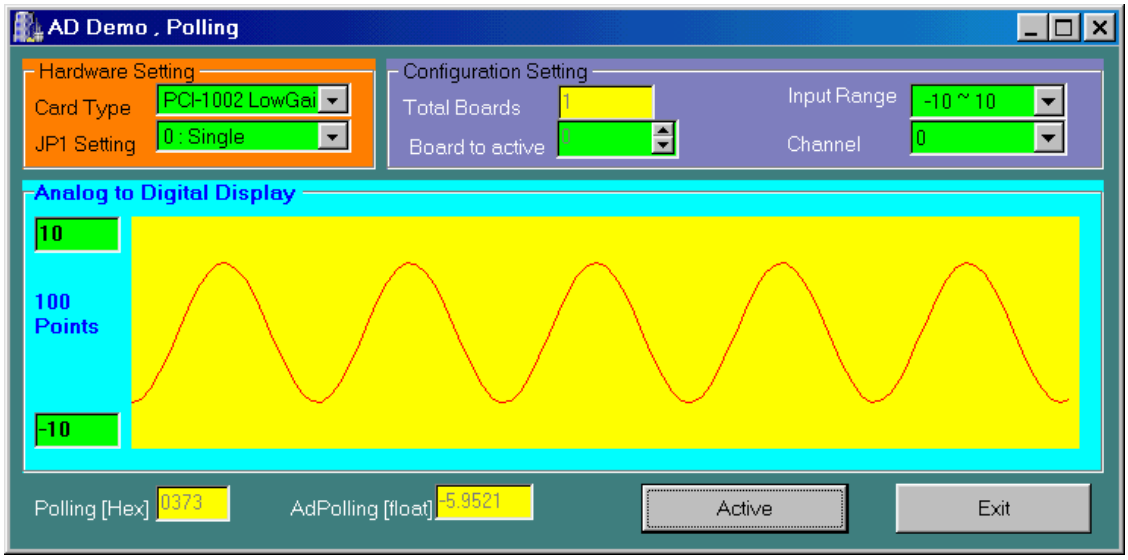


Analog Input with Interrupt demo program

3.4 Borland C++ Builder



Digital I/O demo program



Analog Input with polling demo program

4. Description of Functions

These functions in DLL are divided into several groups as following:

- The test functions
- The D/I/O functions
- The A/D fixed-mode functions
- The Driver functions
- The Interrupt functions

The functions of test listing as follows:

No.	New Functions	Reserve Functions
1	P100X_FloatSub2	No Change
2	P100X_ShortSub2	No Change
3	P100X_GetDllVersion	No Change
4	P100X_GetDriverVersion	No Change

The functions of DI/O listing as follows:

No.	New Functions	Reserve Functions
1	P100X_DigitIn	P100X_DI
2	P100X_DigitOut	P100X_DO

The functions of fixed-channel mode listing as follows:

No.	New Functions	Reserve Functions
1	P100X_SetChConfig	P100X_SetChannelConfig
2	P100X_AdPollingH	P100X_Polling
3	P100X_AdPollingF	P100X_AdPolling
4	P100X_AdMultiPollingF	P100X_AdsPolling
5	P100X_AdMultiPacerF	P100X_AdsPacer

The functions of Driver listing as follows:

No.	New Functions	Reserve Functions
1	P100X_DriverInit	No Change
2	P100X_DriverClose	No Change
3	P100X_GetConfigAddressSpace	No Change
4	P100X_WhichBoardActive	No Change
5	P100X_ActiveBoard	No Change
6	P100X_GetIrqNo	No Change
7	P100X_SetupTimer	No Change
8	P100X_DelayTick	P100X_Delay

The functions of Interrupt listing as follows:

No.	New Functions	Reserve Functions
1	P100X_IntInstall	P100X_InstallIrq
2	P100X_IntStart	P100X_INT_AdStart
3	P100X_IntStartExTrigger	P100X_ExtINT_AdStart
4	P100X_IntStop	P100X_INT_AdStop
5	P100X_IntGetCount	P100X_GetIntCount
6	P100X_IntGetBufferH	P100X_GetBuffer
7	P100X_IntGetBufferF	P100X_GetFloatBuffer
8	P100X_IntRemove	P100X_RemoveIrq

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

Keyword	Description
[Input]	The parameter must be initialized, or have a value set before calling this function.
[Output]	The parameter will return a value or other data after this function is call.
[Input, Output]	The parameter must be initialized, or have a value set before calling this function. And it will return a value or other data after this function is call.

Note: All of the parameters need to be allocated spaces by the user.

4.1 The Configuration Code Table

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

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

4.2.3 P100X_GetDllVersion

- **Description :**
  Read the DLL version number of the **P100X.DLL**.

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

- **Parameter:**
  None

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

4.2.4 P100X_GetDriverVersion

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

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

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

- **Return:**
  P100X_NoError : OK
  P100X_DriverHandleError : the P100X.VxD open error for Windows 95
  the 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_DigitIn

- **Description :**
  This subroutine will read the 16 bits data from DI port. This function will refer to the current active PCI-100X. Use the P100X_ActiveBoard(….) to select the active board.

- **Syntax :**
  WORD P100X_DigitIn(DWORD *wDi);

- **Parameter:**
  wDi : [Output] address of **wDi**,
  which will store the 16 bits data of Digital-Input.

- **Return:**
  P100X_NoError : OK
  P100X_FindBoardError : cannot find the PCI-100X board
  P100X_ExceedBoardNumber : invalidate board number

4.3.2 P100X_DigitOut

- **Description :**
  This subroutine will send the 16 bits data to DO port. This function will refer to the current active PCI-1002 board. Use the P100X_ActiveBoard(….) to select the active board.

- **Syntax :**
  WORD P100X_DigitOut(DWORD wDo);

- **Parameter:**
  wDo : [Input] the 16 bit data sent to Digital-Output port

- **Return:**
  P100X_NoError : OK
  P100X_ExceedBoardNumber : invalidate board number
  P100X_FindBoardError : cannot find the PCI-1002 board

4.4 The AD Fixed-mode Functions

4.4.1 P100X_SetChConfig

- **Description :**
  This subroutine will set the AD channel's configuration code. This subroutine will set the active AD channel for **P100X_ AdPollingF**, **P100X_ AdMultiPollingF** and **P100X_ AdMultiPacerF**. This function will refer to the current active PCI-1002 board. Use the P100X_ActiveBoard(….) to select the active board.

- **Syntax :**
  WORD P100X_SetChConfig(WORD wChannel, WORD wConfig);

- **Parameter:**
  wChannel : [Input] AD channel number
  wConfig : [Input] Configuration code. Refer to Sec. 3.1 for details.

- **Return:**
  P100X_NoError : OK
  P100X_ExceedBoardNumber : invalidate board number
  P100X_FindBoardError : cannot find the PCI-1002 board
  P100X_AdControllerError : MagicScan controller hardware handshake error

4.4.2 P100X_ AdPollingH

- **Description :**
  This subroutine will perform a single A/D conversion on the active channel by software polling. The **P100X_SetChConfig** subroutine can be used to change channel or configuration code. Use the P100X_ActiveBoard(….) to select the active board.

- **Syntax :**
  WORD P100X_ AdPollingH (word *wAdVal);

- **Parameter:**
  wAdVal : [Output] address of **wAdVal**, which store the AD data
  Data is returned as an integer value in the range 0-4095.

- **Return:**
  P100X_NoError : OK
  P100X_ExceedBoardNumber : invalidate board number
  P100X_FindBoardError : cannot find the PCI-1002 board
  P100X_AdPollingTimeOut : hardware timeout error

4.4.3 P100X_ AdPollingF

- **Description :**

  This subroutine will perform the AD conversion by polling one time. The **P100X_SetChConfig** subroutine can be used to change channel or configuration code and the **P100X_AdPollingF** will refer to that condition in later operation. This function will refer to the current active PCI-1002 board. Use the P100X_ActiveBoard(….) to select the active board.

- **Syntax :**

  WORD P100X_AdPollingF(float *fAdVal);

- **Parameter:**

  fAdVal : [output] address of **fAdVal**, which will store the AD data. this data is automatically computed based on the setting of **P100X_SetChConfig()**.

- **Return:**

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

4.4.4 P100X_ AdMultiPollingF

- **Description :**

  This subroutine will perform multiple A/D conversions on a single channel by polling. The **P100X_SetChConfig** subroutine can be used to change the channel or configuration code. This function addresses the current active PCI-1002 board. Use the P100X_ActiveBoard(….) to select the active board.

  Since software polling can be interrupted by the operating system, the P100X_ AdMultiPacerF function is recommended when precisely reconstructing the waveform is desired.

- **Syntax :**

  WORD P100X_ AdMultiPollingF (float fAdVal[], WORD wNum);

- **Parameter:**

  fAdVal : [Output] starting address of AD data buffers(Array of float), these data will be automatically computed based on the setting of **P100X_SetChConfig()**.

  The user must allocate spaces for this buffer and send the address into the function. This function will fill the data into this buffer. The user cans analyze these data from the buffer after calling this function.

  wNum : [Input] number of AD conversions will be performed.

- **Return:**

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

4.4.5 P100X_ AdMultiPacerF

- **Description :**

  This subroutine will perform multiple AD conversions by pacer trigger. The **P100X_SetChConfig** subroutine can be used to change channel or configuration code and the **P100X_ AdMultiPacerF** will refer to that condition in later operation. The hardware pacer will generate trigger signal to AD converter periodically. So these AD data can be used to reconstruct the waveform of analog input. Software polling controls the P100X_AdsPolling , so the AD conversion operation will be interrupted by system OS. **It is recommended to use *P100X_ AdMultiPacerF* if the input wave form reconstruction is needed.** This function will refer to the current active PCI-1002 board. Use the P100X_ActiveBoard(….) to select the active board.

- **Syntax :**

  WORD P100X_ AdMultiPacerF (float fAdVal[], WORD wNum, WORD wSample);

- **Parameter:**

  fAdVal : [Output] Address of AD data buffers (Array of WORD), these data will be automatically computed based on the setting of **P100X_SetChConfig()**.
  The user must allocate spaces for this buffer and send the address into the function. This function will fill the data into this buffer. The user cans analyze these data from the buffer after calling this function.

  wNum : [Input] number of AD conversions will be performed.
  wSample : [Input] **AD sampling rate = 4M/wSample.**

- **Return:**

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

4.5 The Driver Functions

4.5.1 P100X_DriverInit

- **Description :**
  This function will detect all the 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 store the number of PCI-1002 boards on the system.
  wTotalBoard=0 → Not found.
  wTotalBoard=1 → one PCI-1002 card in the system
  wTotalBoard=n → n*PCI-1002 cards in the system

- **Return:**

P100X_NoError	: OK
P100X_NoFoundBoard	: can not detect any 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 :**
  Release all resources to system. This function must be called once before program is terminated.

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

- **Parameter:**
  None

- **Return:**
  None

4.5.3 P100X_GetConfigAddressSpace

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

- **Syntax :**
  WORD P100X_GetConfigAddressSpace(WORD wBoardNo,
  WORD wAddrTimer, WORD *wAddrDio, WORD *wAddrAd);

- **Parameter:**
  wBoardNo : [Input] PCI-1002 board number
  wAddrTimer, wAddrDio, wAddrAd
  : [Output] Address of wAddrTimer, wAddrDio, wAddrAD which will store the address for access the Timer, DI/DO and AD. Please refer to Hardware manual for details.

- **Return:**
  P100X_NoError : OK
  P100X_FindBoardError : handshake check error
  P100X_ExceedBoardError : wBoardNo is invalidated

4.5.4 P100X_WhichBoardActive

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

- **Syntax:**
  WORD P100X_WhichBoardActive(void);

- **Parameter:**
  None

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

4.5.5 P100X_ActiveBoard

- **Description:**
  This function wills active one of the PCI-1002 boards installed in the system. This function must call once before the D/I/O, A/D, D/A functions are called.

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

- **Parameter:**
  wBoardNo : [Input] The board numbers to active.

- **Return:**
  P100X_NoError : OK
  P100X_ExceedBoardError : wBoardNo is invalidated

4.5.6 P100X_GetIrqNo

- **Description:**
  This function will get the IRQ number of the active PCI-1002 board installed in the system. It can let you known what IRQ number is used by PCI-1002. This function is not the necessary for your program.

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

- **Parameter:**
  IrqNo : [Output] Address of IrqNo, which will store the IRQ No that allocated by the system.

- **Return:**
  P100X_NoError : OK

4.6 The Interrupt Functions

4.6.1 P100X_IntInstall

- **Description :**
  This subroutine will install interrupt handler for a specific IRQ n. and setting the maximum number of interrupts. Refer to Section 4.6.9. for more details on using interrupts.

- **Syntax :**
  WORD P100X_IntInstall(HANDLE *hEvent, DWORD dwCount);

- **Parameter:**
  hEvent : [Input] The user must uses the CreateEvent() to create the Event object and obtain its handle and pass the handle into this function.
  dwCount : [Input] Maximum numbers of counter for interrupt transfer.

- **Return:**
  P100X_NoError : successful
  P100X_InstallIrqError : fail in install IRQ handler.

4.6.2 P100X_IntGetCount

- **Description :**

This subroutine will read the interrupt transfer count.

- **Syntax :**
  WORD P100X_IntGetCount(DWORD *dwVal)

- **Parameter:**
  dwVal : [Output] the address of dwVal, which will store the value of interrupt transferred count.

- **Return:**
  P100X_NoError : successful
  P100X_GetIntCountError : fail get interrupt count.

4.6.3 P100X_IntStart

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

- **Syntax :**
  WORD P100X_IntStart (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 4M/(wFreqDiv)

- **Return:**
  P100X_NoError : successful
  P100X_INTStartError : failure

4.6.4 P100X_IntStartExTrigger

- **Description :**
  This subroutine will start the interrupt transfer for a specific A/D channel and programming the gain code from external interrupt (Pin19).

- **Syntax :**
  WORD P100X_IntStartExTrigger (WORD Ch, WORD Gain)

- **Parameter:**
  Ch : [Input] the A/D channel.
  Gain : [Input] the Gain, refer to Section 3.1

- **Return:**
  P100X_NoError : successful
  P100X_INTStartError : failure

4.6.5 P100X_IntStop

- **Description :**
  This subroutine will stop the interrupt transfer.

- **Syntax :**
  WORD P100X_IntStop (void)

- **Parameter:**
  None

- **Return:**
  P100X_NoError : successful
  P100X_INTStopError : failure

4.6.6 P100X_IntRemove

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

- **Syntax :**
  WORD P100X_IntRemove (void)

- **Parameter:**
  None

- **Return:**
  P100X_NoError : successful
  P100X_INTStopError : failure

4.6.7 P100X_ IntGetBufferH

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

- **Syntax :**
  WORD P100X_IntGetBufferH (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 store the Hex A/D value.
  The user must allocate spaces for this buffer and send the address into the function. This function will fill the data into this buffer. The user cans analyze these data from the buffer after calling this function.

- **Return:**
  P100X_NoError : successful
  P100X_GetBufferError : fail

4.6.8 P100X_IntGetBufferF

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

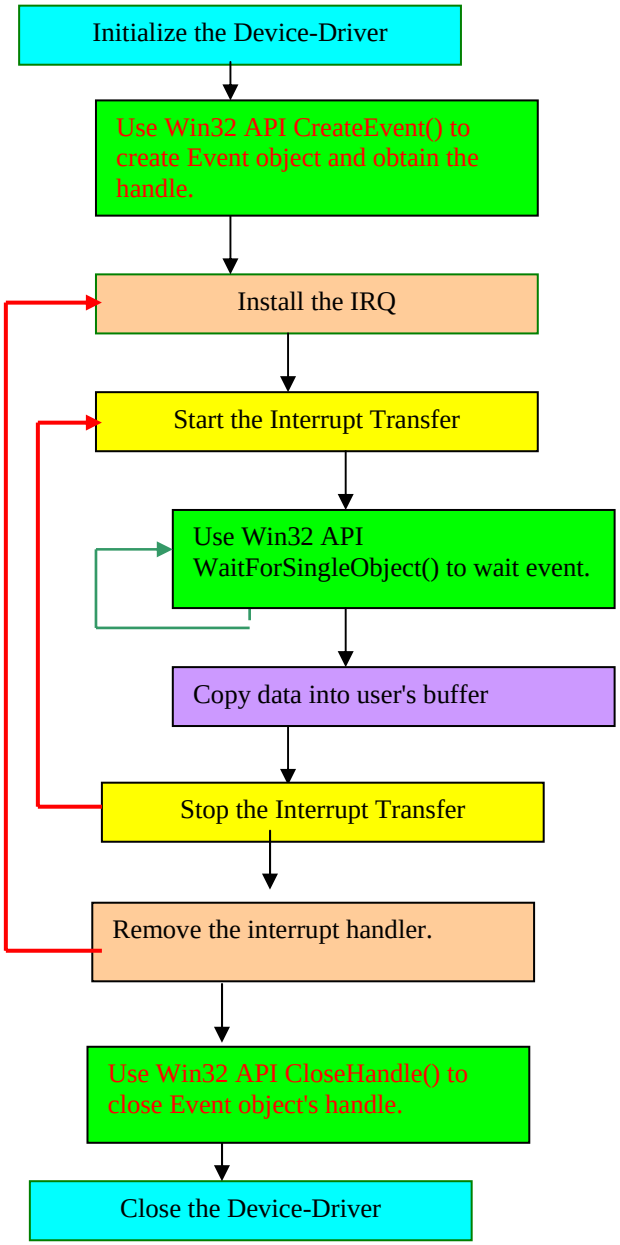
- **Syntax :**
  WORD P100X_IntGetBufferF(DWORD dwNum, float fAdVal[])

- **Parameter:**
  wNum : [Input] The total number to transfer to User's Buffer.
  fAdVal : [Output] Address of fAdVals (Array of float) that store the voltage value(floating-point).
  The user must allocate spaces for this buffer and send the address into the function. This function will fill the data into this buffer. The user cans analyze these data from the buffer after calling this function.

- **Return:**
  P100X_NoError : successful
  P100X_GetBufferError : fail

4.6.9 Architecture of Interrupt mode

From 4.6.1 to 4.6.8 are these functions to perform the A/D conversion with interrupt transfer. The flow chart about steps of programming these functions are given as follows:



P100X_DriverInit( )

…….
CreateEvent( )

…….

P100X _IntInstall(…)

……

P100X _IntStart(…)

……

WaitForSingleObject(….)

……..

P100X _IntGetBufferF(….)

……

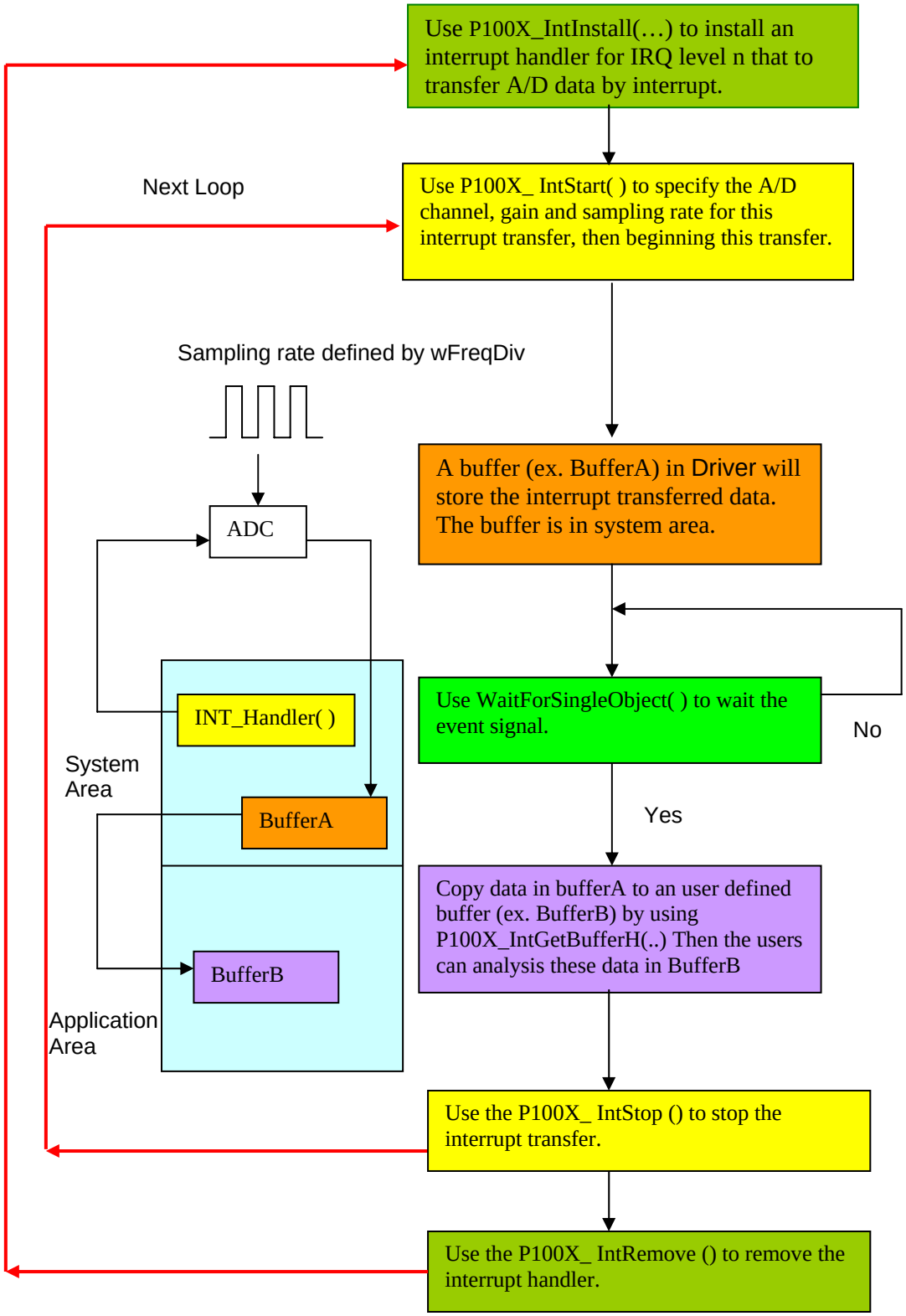
P100X _IntStop( )

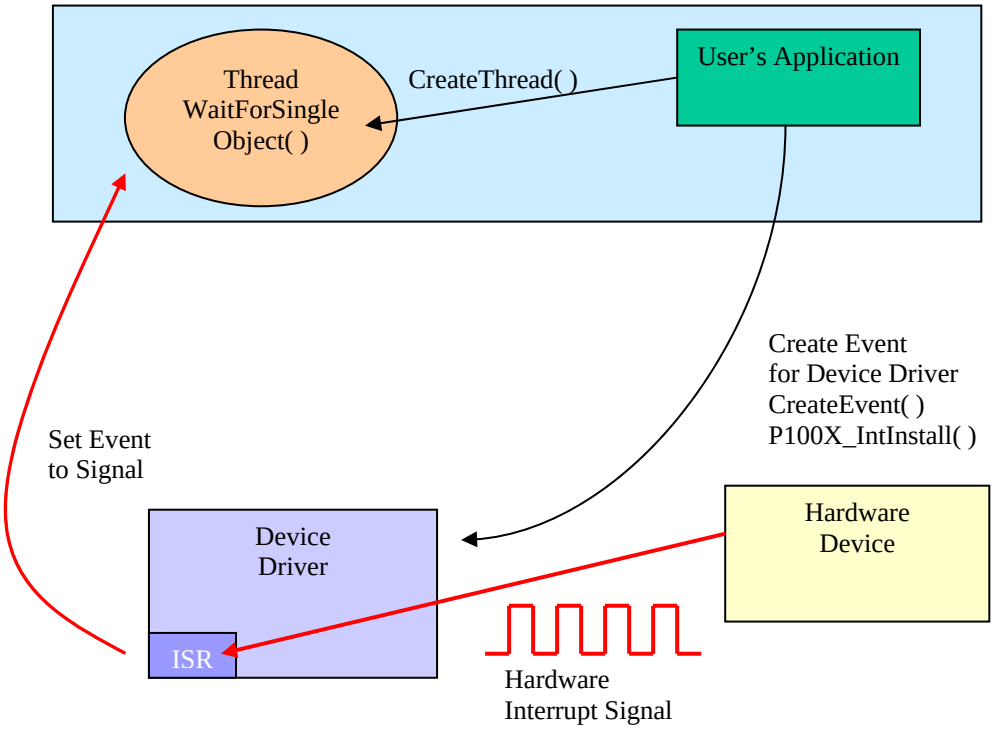
……

P100X_IntRemove( )
CloseHandle( )

…….

P100X _DriverClose( )


Use P100X_IntInstall(…) to install an interrupt handler for IRQ level n that to transfer A/D data by interrupt.
Next Loop
Use P100X_ IntStart( ) to specify the A/D channel, gain and sampling rate for this interrupt transfer, then beginning this transfer.
Sampling rate defined by wFreqDiv
A buffer (ex. BufferA) in Driver will store the interrupt transferred data. The buffer is in system area.
ADC
INT_Handler( )
Use WaitForSingleObject( ) to wait the event signal.
No
System Area
BufferA
Yes
Copy data in bufferA to an user defined buffer (ex. BufferB) by using P100X_IntGetBufferH(..) Then the users can analysis these data in BufferB
BufferB
Application Area
Use the P100X_ IntStop () to stop the interrupt transfer.
Use the P100X_ IntRemove () to remove the interrupt handler.



Please refer to the following Windows API functions:

The following description of these functions was 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( )

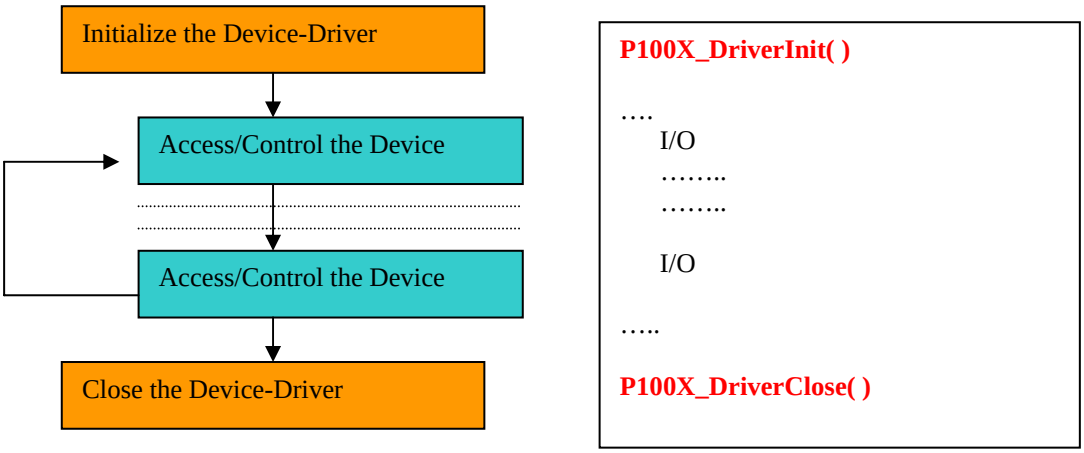
The WaitForSingleObject function returns when one of the following occurs:

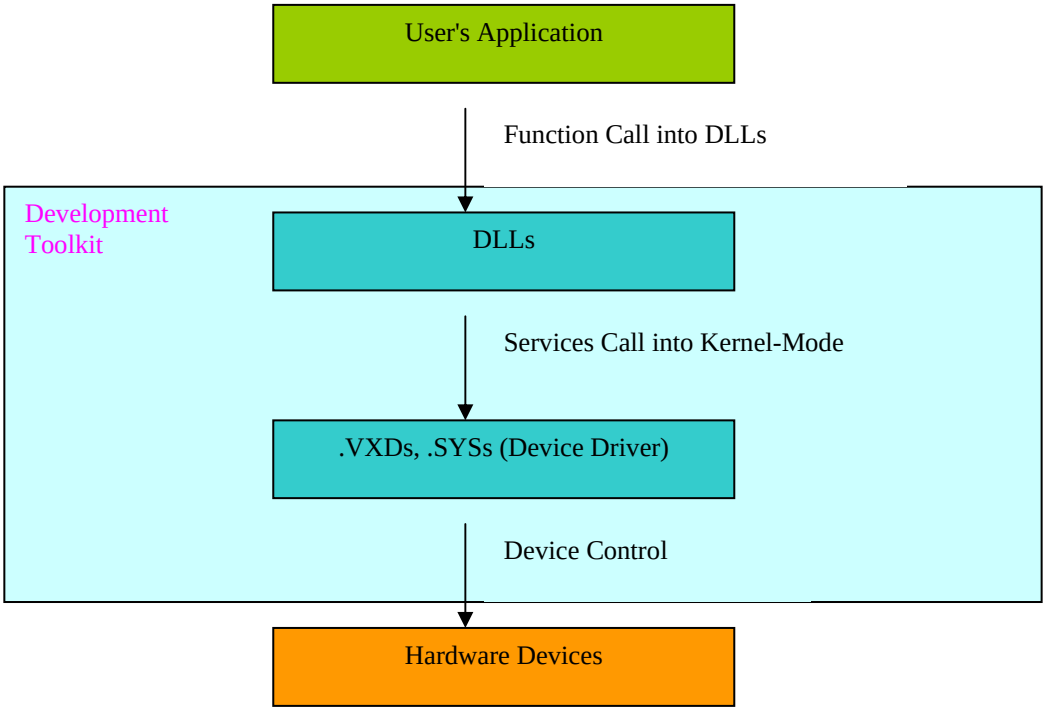
- The specified object is in the signaled state.
- The time-out interval elapses.

To enter an alertable 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. Problems Report

Technical support is available at no charge as described below. The best way to report problems is to send electronic mail to

Service@icpdas.com
or icpdas@ms8.hinet.net

on the Internet.

When reporting problems, please include the following information:

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

2) What kind and version of Platform that you using? For example, Windows 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 had 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 had received your comments? And please keeps contact with us.

E-mail: Service@icpdas.com
Web-Site: http://www.icpdas.com