PISO-DIO

Software Manual

[For Windows 95/98/NT/2000]

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 maybe registered trademarks of their respective companies.

License

The user can use, modify and backup this software **on a single machine.** The user may not reproduce, transfer or distribute this software, or any copy, in whole or in part.

Table of Contents

1.Introduction

The software is a collection of digital I/O subroutines for PISO-DIO series add-on cards for Windows 95/98, NT 4.0 and Windows 2000 Applications. These subroutines are written with C language and perform a variety of digital I/O operations.

The subroutines in PISODIO.DLL are easy understanding as its name standing for. It provides powerful, easy-to-use subroutine 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:

- **PnPInstall.pdf:**
  Install the PnP (Plug and Play) driver for PCI card under Windows 95/98.

- **SoftInst.pdf:**
  Install the software package under Windows 95/98/NT.

- **CallDll.pdf:**
  Call the DLL functions with VC++5, VB5, Delphi3 and Borland C++ Builder 3.

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

- **PISO3232.pdf:**
  The Hardware Manual for PISO-P32C32, PISO-C64 and PISO-P64.

2. Declaration Files

```
|--\Driver                  ← some device driver
    |--\BCB3                ← for Borland C++ Builder 3
    |     |--\PISODIO.H     ← Header file
    |     +--\PISODIO.LIB   ← Linkage library for BCB3 only
    |
    |--\Delphi3             ← for Delphi 3
    |     +--\PISODIO.PAS   ← Declaration file
    |
    |--\VB5                 ← for Visual Basic 5
    |     +--\PISODIO.BAS   ← Declaration file
    |
    +--\VC5                 ← for Visual C++ 5
          |--\PISODIO.H     ← Header file
          +--\PISODIO.LIB   ← Linkage library for VC5 only
```

2.1 PISODIO.H

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

// return code
#define PISODIO_NoError                 0
#define PISODIO_DriverOpenError         1
#define PISODIO_DriverNoOpen            2
#define PISODIO_GetDriverVersionError   3
#define PISODIO_InstallIrqError         4
#define PISODIO_ClearIntCountError      5
#define PISODIO_GetIntCountError        6
#define PISODIO_RegisterApcError        7
#define PISODIO_RemoveIrqError          8
#define PISODIO_FindBoardError          9
#define PISODIO_ExceedBoardNumber       10
#define PISODIO_ResetError              11

// to trigger a interrupt when high -> low
#define PISODIO_ActiveLow               0
// to trigger a interrupt when low -> high
#define PISODIO_ActiveHigh              1

// ID
#define PISO_C64                0x800800    // for PISO-C64
#define PISO_P64                0x800810    // for PISO-P64
#define PISO_P32C32             0x800820    // for PISO-P32C32
#define PISO_P8R8               0x800830    // for PISO-P8R8
#define PISO_P8SSR8AC           0x800830    // for PISO-P8SSR8AC
#define PISO_P8SSR8DC           0x800830    // for PISO-P8SSR8DC
#define PISO_730                0x800840    // for PISO-730

// Test functions
EXPORTS float       CALLBACK PISODIO_FloatSub(float fA, float fB);
EXPORTS short       CALLBACK PISODIO_ShortSub(short nA, short nB);
EXPORTS WORD        CALLBACK PISODIO_GetDllVersion(void);
```

```
// Driver functions
EXPORTS WORD      CALLBACK PISODIO_DriverInit(void);
EXPORTS void      CALLBACK PISODIO_DriverClose(void);
EXPORTS WORD      CALLBACK PISODIO_SearchCard(WORD *wBoards,
      DWORD dwPIOCardID);
EXPORTS WORD      CALLBACK PISODIO_GetDriverVersion(
      WORD *wDriverVersion);
EXPORTS WORD   CALLBACK PISODIO_GetConfigAddressSpace(
      WORD  wBoardNo, DWORD *wAddrBase, WORD *wIrqNo,
      WORD *wSubVendor, WORD *wSubDevice, WORD *wSubAux,
      WORD *wSlotBus,   WORD *wSlotDevice);
EXPORTS WORD   CALLBACK PISODIO_ActiveBoard( WORD wBoardNo );
EXPORTS WORD   CALLBACK PISODIO_WhichBoardActive(void);

// DIO functions
EXPORTS void   CALLBACK PISODIO_OutputWord(DWORD wPortAddress,
      DWORD wOutData);
EXPORTS void   CALLBACK PISODIO_OutputByte(DWORD wPortAddr,
      WORD bOutputValue);
EXPORTS DWORD  CALLBACK PISODIO_InputWord(
      DWORD wPortAddress);
EXPORTS WORD   CALLBACK PISODIO_InputByte(DWORD wPortAddr);

// Interrupt functions
EXPORTS WORD   CALLBACK PISODIO_IntInstall(WORD wBoardNo,
      HANDLE *hEvent, WORD wInterruptSource, WORD wActiveMode);
EXPORTS WORD   CALLBACK PISODIO_IntRemove(void);
EXPORTS WORD   CALLBACK PISODIO_IntGetCount(
      DWORD *dwIntCount);
EXPORTS WORD   CALLBACK PISODIO_IntResetCount(void);
```

2.2 PISODIO.BAS

```
Attribute VB_Name = "PISODIO"

Global Const PISODIO_NoError                = 0
Global Const PISODIO_DriverOpenError        = 1
Global Const PISODIO_DriverNoOpen           = 2
Global Const PISODIO_GetDriverVersionError  = 3
Global Const PISODIO_InstallIrqError        = 4
Global Const PISODIO_ClearIntCountError     = 5
Global Const PISODIO_GetIntCountError       = 6
Global Const PISODIO_RegisterApcError       = 7
Global Const PISODIO_RemoveIrqError         = 8
Global Const PISODIO_FindBoardError         = 9
Global Const PISODIO_ExceedBoardNumber      = 10
Global Const PISODIO_ResetError             = 11

' to trigger a interrupt when high -> low
Global Const PISODIO_ActiveLow              = 0
' to trigger a interrupt when low -> high
Global Const PISODIO_ActiveHigh             = 1


' ID
Global Const PISO_C64           = &H800800     ' for PISO-C64
Global Const PISO_P64           = &H800810     ' for PISO-P64
Global Const PISO_P32C32        = &H800820     ' for PISO-P32C32
Global Const PISO_P8R8          = &H800830     ' for PISO-P8R8
Global Const PISO_P8SSR8AC      = &H800830     ' for PISO-P8SSR8AC
Global Const PISO_P8SSR8DC      = &H800830     ' for PISO-P8SSR8DC
Global Const PISO_730           = &H800840     ' for PISO-730


' The Test functions
Declare Function PISODIO_ShortSub Lib "PISODIO.dll" (ByVal a As Integer,
      ByVal b As Integer) As Integer
Declare Function PISODIO_FloatSub Lib "PISODIO.dll" (ByVal a As Single,
      ByVal b As Single) As Single
Declare Function PISODIO_GetDllVersion Lib "PISODIO.dll" () As Integer
```

```
' The Driver functions
Declare Function PISODIO_DriverInit Lib "PISODIO.dll" () As Integer
Declare Sub PISODIO_DriverClose Lib "PISODIO.dll" ()
Declare Function PISODIO_SearchCard Lib "PISODIO.dll" (
      wBoards As Integer, ByVal dwPIOPISOCardID As Long) As Integer
Declare Function PISODIO_GetDriverVersion Lib "PISODIO.dll"
      (wDriverVersion As Integer) As Integer
Declare Function PISODIO_GetConfigAddressSpace Lib "PISODIO.dll" ( _
     ByVal wBoardNo As Integer, wAddrBase As Long, wIrqNo As Integer, _
     wSubVendor As Integer, wSubDevice As Integer, wSubAux As Integer, _
     wSlotBus As Integer, wSlotDevice As Integer) As Integer

Declare Function PISODIO_ActiveBoard Lib "PISODIO.dll" (
      ByVal wBoardNo As Integer) As Integer
Declare Function PISODIO_WhichBoardActive Lib "PISODIO.dll" () As Integer


' DIO functions
Declare Sub PISODIO_OutputByte Lib "PISODIO.dll" ( _
      ByVal address As Long, ByVal dataout As Integer)
Declare Sub PISODIO_OutputWord Lib "PISODIO.dll" ( _
      ByVal address As Long, ByVal dataout As Long)
Declare Function PISODIO_InputByte Lib "PISODIO.dll" ( _
      ByVal address As Long) As Integer
Declare Function PISODIO_InputWord Lib "PISODIO.dll" ( _
      ByVal address As Long) As Long

' Interrupt functions
Declare Function PISODIO_IntInstall Lib "PISODIO.dll" (
      ByVal wBoard As Integer, hEvent As Long, _
      ByVal wInterruptSource As Integer, ByVal wActiveMode As Integer) _
      As Integer
Declare Function PISODIO_IntRemove Lib "PISODIO.dll" () As Integer
Declare Function PISODIO_IntGetCount Lib "PISODIO.dll" ( _
      dwIntCount As Long) As Integer
Declare Function PISODIO_IntResetCount Lib "PISODIO.dll" () As Integer
```

2.3 PISODIO.PAS

```
unit  PISODIO;          { PISODIO.dll interface unit }

interface

const
  PISODIO_NoError                       =0;
  PISODIO_DriverOpenError               =1;
  PISODIO_DriverNoOpen                  =2;
  PISODIO_GetDriverVersionError         =3;
  PISODIO_InstallIrqError               =4;
  PISODIO_ClearIntCountError            =5;
  PISODIO_GetIntCountError              =6;
  PISODIO_RegisterApcError              =7;
  PISODIO_RemoveIrqError                =8;
  PISODIO_FindBoardError                =9;
  PISODIO_ExceedBoardNumber             =10;
  PISODIO_ResetError                    =11;

  // to trigger a interrupt when high -> low
  PISODIO_ActiveLow                     =0;
  // to trigger a interrupt when low -> high
  PISODIO_ActiveHigh                    =1;

  // ID
  PISO_C64              = $800800;    // for PISO-C64
  PISO_P64              = $800810;    // for PISO-P64
  PISO_P32C32           = $800820;    // for PISO-P32C32
  PISO_P8R8             = $800830;    // for PISO-P8R8
  PISO_P8SSR8AC         = $800830;    // for PISO-P8SSR8AC
  PISO_P8SSR8DC         = $800830;    // for PISO-P8SSR8DC
  PISO_730              = $800840;    // for PISO-730


// Test functions
function  PISODIO_ShortSub(nA : smallint; nB : smallint) : smallint; StdCall;
function  PISODIO_FloatSub(fA :   single; fB : single)       :  single; StdCall;
function  PISODIO_GetDllVersion                              :  word; StdCall;
```

```
// Driver functions
function  PISODIO_DriverInit                                    : word; StdCall;
procedure PISODIO_DriverClose                                       ; StdCall;
function  PISODIO_SearchCard(var wBoards:WORD;
      dwPIOPISOCardID:LongInt):WORD; StdCall;
function  PISODIO_GetDriverVersion(var wDriverVer: word) : word; StdCall;
function  PISODIO_GetConfigAddressSpace(
      wBoardNo:word; var wAddrBase:LongInt;  var wIrqNo:word;
      var wSubVerdor:word; var wSubDevice:word;    var wSubAux:word;
      var wSlotBus:word;   var wSlotDevice:word )           : word; StdCall;
function  PISODIO_ActiveBoard(wBoardNo:Word)          : WORD; StdCall;
function  PISODIO_WhichBoardActive                    : WORD; StdCall;

// DIO functions
procedure PISODIO_OutputByte(wPortAddr : LongInt; bOutputVal : Word)
      ; StdCall;
procedure PISODIO_OutputWord(wPortAddr : LongInt; wOutputVal : LongInt)
      ; StdCall;
function  PISODIO_InputByte(wPortAddr  : LongInt )        : word; StdCall;
function  PISODIO_InputWord(wPortAddr  : LongInt )       : LongInt; StdCall;

// Interrupt functions
function  PISODIO_IntInstall(wBoard:Word; var hEvent:LongInt;
      wInterruptSource:Word; wActiveMode:Word)          : WORD; StdCall;
function  PISODIO_IntRemove                             : WORD; StdCall;
function  PISODIO_IntGetCount(var dwIntCount:LongInt)  : WORD; StdCall;
function  PISODIO_IntResetCount                         : WORD; StdCall;


implementation

// Test functions
function  PISODIO_ShortSub;                   external 'PISODIO.dll' name
      'PISODIO_ShortSub';
function  PISODIO_FloatSub;                   external 'PISODIO.dll' name
      'PISODIO_FloatSub';
function  PISODIO_GetDllVersion;              external 'PISODIO.dll' name
      'PISODIO_GetDllVersion';
```

```
// Driver functions
function  PISODIO_DriverInit;                external 'PISODIO.dll' name
      'PISODIO_DriverInit';
procedure PISODIO_DriverClose;               external 'PISODIO.dll' name
      'PISODIO_DriverClose';
function  PISODIO_SearchCard;                external 'PISODIO.dll' name
      'PISODIO_SearchCard';
function  PISODIO_GetDriverVersion;          external 'PISODIO.dll' name
      'PISODIO_GetDriverVersion';
function  PISODIO_GetConfigAddressSpace;     external 'PISODIO.dll' name
      'PISODIO_GetConfigAddressSpace';

function  PISODIO_ActiveBoard;               external 'PISODIO.dll' name
      'PISODIO_ActiveBoard';
function  PISODIO_WhichBoardActive;          external 'PISODIO.dll' name
      'PISODIO_WhichBoardActive';

// DIO functions
procedure PISODIO_OutputByte;                external 'PISODIO.dll' name
      'PISODIO_OutputByte';
procedure PISODIO_OutputWord;                external 'PISODIO.dll' name
      'PISODIO_OutputWord';
function  PISODIO_InputByte;                 external 'PISODIO.dll' name
      'PISODIO_InputByte';
function  PISODIO_InputWord;                 external 'PISODIO.dll' name
      'PISODIO_InputWord';

// Interrupt functions
function  PISODIO_IntInstall;                external 'PISODIO.dll' name
      'PISODIO_IntInstall';
function  PISODIO_IntRemove;                 external 'PISODIO.dll' name
      'PISODIO_IntRemove';
function  PISODIO_IntGetCount;               external 'PISODIO.dll' name
      'PISODIO_IntGetCount';
function  PISODIO_IntResetCount;             external 'PISODIO.dll' name
      'PISODIO_IntResetCount';

end.
```

3. FUNCTION DESCRIPTINOS

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

Keyword	Setting parameter by user before calling this function ?	Get the data/value from this parameter after calling this function ?
[Input]	Yes	No
[Output]	No	Yes
[Input, Output]	Yes	Yes

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

3.1 FUNCTIONS OF TEST

3.1.1 PISODIO_GetDllVersion

- **Description:**
  To get the version number of PISODIO.DLL

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

- **Parameter:**
  None

- **Return:**
  200(hex) for version 2.00

3.1.2 PISODIO_ShortSub

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

- **Syntax:**
  short PISODIO_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

3.1.3 PISODIO_FloatSub

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

- **Syntax:**
  float PISODIO_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

3.2 FUNCTIONS OF I/O

3.2.1 PISODIO_OutputByte

- **Description :**
  This subroutine will send the 8 bits data to the desired I/O port.

- **Syntax :**
  void PISODIO_OutputByte(DWORD wPortAddr, WORD bOutputVal);

- **Parameter :**
  wPortAddr : [Input] I/O port addresses, please refer to function
  PISODIO_GetConfigAddressSpace.
  Only the low WORD is valid.
  bOutputVal : [Input] 8 bit data send to I/O port.
  Only the low BYTE is valid.

- **Return:**
  None

3.2.2 PISODIO_InputByte

- **Description :**
  This subroutine will input the 8 bit data from the desired I/O port.

- **Syntax :**
  WORD PISODIO_InputByte(DWORD wPortAddr);

- **Parameter :**
  wPortAddr : [Input] I/O port addresses, please refer to function
  PISODIO_GetConfigAddressSpace().
  Only the low WORD is valid.

- **Return:**
  16 bits data with the leading 8 bits are all 0.
  (Only the low BYTE is valid.)

3.2.3 PISODIO_OutputWord

- **Description :**
  This subroutine will send the 16 bits data to the desired I/O port.

- **Syntax :**
  void PISODIO_OutputWord(DWORD wPortAddr, DWORD wOutputVal);

- **Parameter :**
  wPortAddr : [Input] I/O port addresses, please refer to function PISODIO_GetConfigAddressSpace(). Only the low WORD is valid.
  wOutputVal : [Input] 16 bit data send to I/O port. Only the low WORD is valid.

- **Return:**
  None

3.2.4 PISODIO_InputWord

- **Description :**
  This subroutine will input the 16 bit data from the desired I/O port.

- **Syntax :**
  DWORD PISODIO_InputWord(DWORD wPortAddr);

- **Parameter :**
  wPortAddr : [Input] I/O port addresses, please refer to function PISODIO_GetConfigAddressSpace(). Only the low WORD is valid.

- **Return:**
  16 bit data. Only the low WORD is valid.

3.3 FUNCTIONS OF DRIVER

3.3.1 PISODIO_GetDriverVersion

- **Description :**
  This subroutine will read the version number of PISODIO driver.

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

- **Parameter :**
  wDriverVersion : [Output] address of wDriverVersion

- **Return:**
  PISODIO_NoError : OK
  PISODIO_DriverNoOpen : The PISODIO driver no open
  PISODIO_GetDriverVersionError : Read driver version error

3.3.2 PISODIO_DriverInit

- **Description :**
  This subroutine will open the PISODIO driver and allocate the resource for the device. This function must be called once before calling other PISODIO functions.

- **Syntax :**
  WORD PISODIO_DriverInit();

- **Parameter :**
  None

- **Return:**
  PISODIO_NoError : OK
  PISODIO_DriverOpenError : open PISODIO Driver error

3.3.3 PISODIO_DriverClose

- **Description :**
  This subroutine will close the PISODIO Driver and release the resource from the device. This function must be called once before exit the user's application.

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

- **Parameter :**
  None

- **Return:**
  None

3.3.4 PISODIO_GetConfigAddressSpace

- **Description :**
  Get the I/O address of PISODIO board n.

- **Syntax :**

```
WORD PISODIO_GetConfigAddressSpace
        ( WORD wBoardNo,   DWORD *wAddrBase,     WORD *wIrqNo,
        WORD *wSubVendor, WORD *wSubDevice,     WORD *wSubAux,
        WORD *wSlotBus,     WORD *wSlotDevice);
```

- **Parameter :**

wBoardNo	: [Input] PISODIO board number
wAddrBase	: [Output] The base address of PISODIO board. Only the low WORD is valid.
wIrqNo	: [Output] The IRQ number that the PISODIO board using.
wSubVendor	: [Output] Sub Vendor ID.
wSubDevice	: [Output] Sub Device ID.
wSubAux	: [Output] Sub Aux ID.
wSlotBus	: [Output] Slot Bus number.
wSlotDevice	: [Output] Slot Device ID.

- **Return:**

PISODIO_NoError	: OK
PISODIO_FindBoardError	: handshake check error
PISODIO_ExceedBoardError	: wBoardNo is invalidated

3.4 INTERRUPT FUNCTION

3.4.1 PISODIO_IntResetCount

- **Description:**
  This subroutine will reset the d**wIntCount** defined in device-driver.

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

- **Parameter:**
  None

- **Return:**

PISODIO_NoError	: OK
PISODIO_DriverNoOpen	: the device driver no open
PISODIO_ClearIntCountError	: **dwIntCount** clear error

3.4.2 PISODIO_IntGetCount

- **Description:**
  This subroutine will read the d**wIntCount** defined in device driver.

- **Syntax :**
  WORD PISODIO_IntGetCount(WDORD *dwIntCount);

- **Parameter:**

dwIntCount	: [Output] Address of dwIntCount, which will stores the counter value of interrupt.

- **Return:**

PISODIO_NoError	: OK
PISODIO_GetIntCountError	: **dwIntCount** read error

3.4.3 PISODIO_IntInstall

- **Description:**
  This subroutine will install the IRQ service routine.

- **Syntax:**
  WORD PISODIO_IntInstall(WORD wBoardNo, HANDLE *hEvent,
  WORD wInterruptSource, WORD wActiveMode);

- **Parameter:**
  wBoardNo : [Input] Which board to be used.
  hEvent : [Input] Address of an Event handle. The user's program
  must call the Windows API function "CreateEvent()" to create the event-object.
  wInterruptSource : [Input] What the Interrupt-Source to be used ? Please refer to hardware's manual for the detail information.

Card No.	wInterruptSource	Description
PISO-730	0	DI0
	1	DI1

  wActiveMode : [Input] When to trigger the interrupt ?
  This can be PISODIO_ActiveHigh
  or PISODIO_ActiveLow.

- **Return:**
  PISODIO_NoError : OK
  PISODIO_InstallIrqError : IRQ installation error

3.4.4 PISODIO_IntRemove

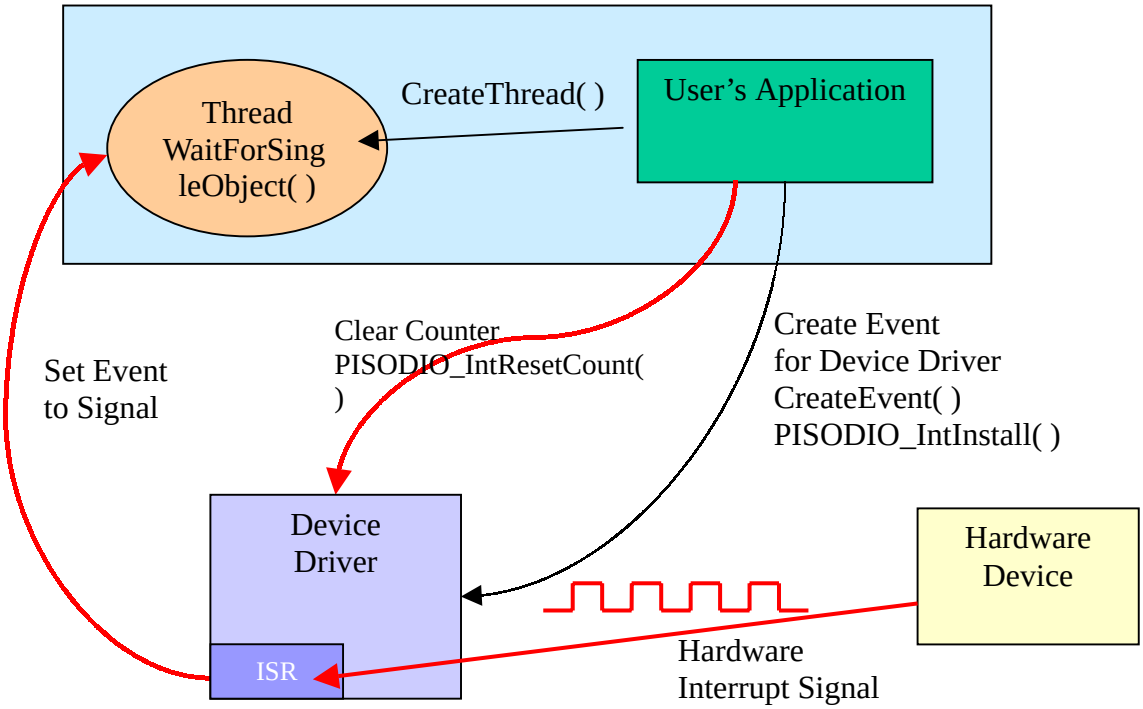
- **Description:**
  This subroutine will remove the IRQ service routine.

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

- **Parameter:**
  None

- **Return:**
  PISODIO_NoError : OK

3.4.5 Architecture of Interrupt mode



Please refer to the following Windows API functions:

The following portion description of these functions was copied from MSDN. For the detailed and completely information, please refer to MSDN.

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( )

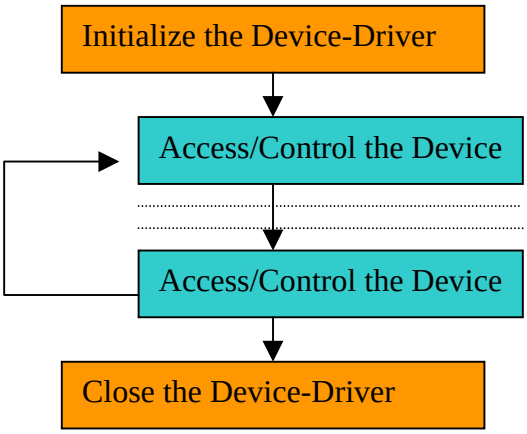
The WaitForSingleObject function returns when one of the following occurs:

- 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.

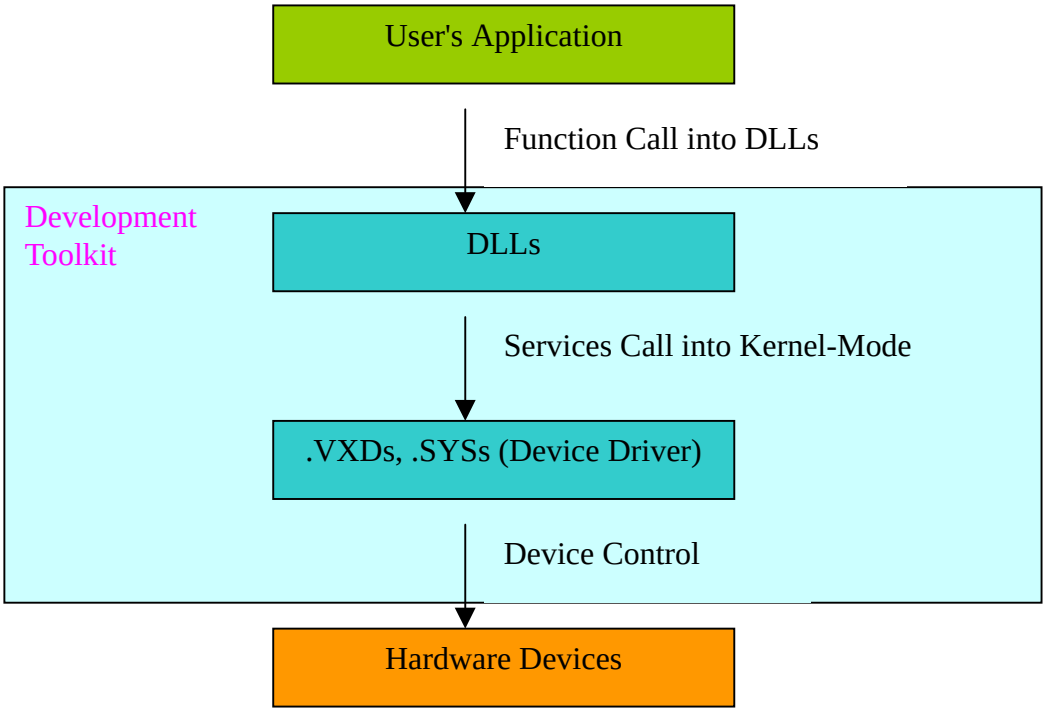
```
DWORD WaitForSingleObject(
        HANDLE hHandle,          // handle to object to wait for
        DWORD dwMilliseconds   // time-out interval in
    milliseconds
);
```

4. Program Architecture




```
PISODIO_DriverInit()
// Enable All DI/DO
….
    PISODIO_InputByte( … )
    ……..
    ……..
    PISODIO_OutputByte(…)
…..

PISODIO_DriverClose()
```



5. Problems Report

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

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

on the Internet.

When reporting problems, please include the following information:

1) Is the problem reproducible? If so, how?
2) What kind and version of platform that you using? For example, Windows 3.1, Windows 95, or 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 test the problems that you said. And then reply as soon as possible to you. Please check that if we had received your comments? And please keeps contact with us.

ICP DAS
E-mail: icpdas@ms8.hinet.net
Service@icpdas.com
Web Site: http://www.icpdas.com
http://www.icpdas.com.tw