PISO-725

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-730 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 PISO725.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.

2. Declaration Files

```
|--\Driver                  ← some device driver
    |--\BCB3                ← for Borland C++ Builder 3
    |     |--\PISO725.H     ← Header file
    |     +--\PISO725.LIB   ← Linkage library for BCB3 only
    |
    |--\Delphi3             ← for Delphi 3
    |     +--\PISO725.PAS   ← Declaration file
    |
    |--\VB5                 ← for Visual Basic 5
    |     +--\PISO725.BAS   ← Declaration file
    |
    +--\VC5                 ← for Visual C++ 5
          |--\PISO725.H     ← Header file
          +--\PISO725.LIB   ← Linkage library for VC5 only
```

2.1 PISO725.H

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

// return code
#define PISO725_NoError                 0
#define PISO725_DriverOpenError         1
#define PISO725_DriverNoOpen            2
#define PISO725_GetDriverVersionError   3
#define PISO725_InstallIrqError         4
#define PISO725_ClearIntCountError      5
#define PISO725_GetIntCountError        6
#define PISO725_RegisterApcError        7
#define PISO725_RemoveIrqError          8
#define PISO725_FindBoardError          9
#define PISO725_ExceedBoardNumber       10
#define PISO725_ResetError              11

// ID
#define PISO_725                0x800cff    // for PISO-725

// Test functions
EXPORTS float       CALLBACK PISO725_FloatSub(float fA, float fB);
EXPORTS short       CALLBACK PISO725_ShortSub(short nA, short nB);
EXPORTS WORD        CALLBACK PISO725_GetDllVersion(void);

// Driver functions
EXPORTS WORD        CALLBACK PISO725_DriverInit(void);
EXPORTS void        CALLBACK PISO725_DriverClose(void);
EXPORTS WORD        CALLBACK PISO725_SearchCard
    (WORD *wBoards,         DWORD dwPIOCardID);
EXPORTS WORD        CALLBACK PISO725_GetDriverVersion
    (WORD *wDriverVersion);
EXPORTS WORD        CALLBACK PISO725_GetConfigAddressSpace
    (WORD  wBoardNo,        DWORD *wAddrBase,     WORD *wIrqNo,
     WORD *wSubVendor,      WORD *wSubDevice,     WORD *wSubAux,
     WORD *wSlotBus,        WORD *wSlotDevice);
EXPORTS WORD    CALLBACK PISO725_ActiveBoard( WORD wBoardNo );
EXPORTS WORD    CALLBACK PISO725_WhichBoardActive(void);
```

```
// DIO functions
EXPORTS void   CALLBACK PISO725_OutputWord
      (DWORD wPortAddress, DWORD wOutData);
EXPORTS void   CALLBACK PISO725_OutputByte
      (DWORD wPortAddr, WORD bOutputValue);
EXPORTS DWORD  CALLBACK PISO725_InputWord
      (DWORD wPortAddress);
EXPORTS WORD   CALLBACK PISO725_InputByte(DWORD wPortAddr);

// Interrupt functions
EXPORTS WORD   CALLBACK PISO725_IntInstall
      (WORD wBoardNo, HANDLE *hEvent);
EXPORTS WORD   CALLBACK PISO725_IntRemove(void);
EXPORTS WORD   CALLBACK PISO725_IntGetActiveFlag
      (WORD *bActiveHighFlag, WORD *bActiveLowFlag);
```

2.2 PISO725.BAS

```
Attribute VB_Name = "PISO725"

Global Const PISO725_NoError                = 0
Global Const PISO725_DriverOpenError        = 1
Global Const PISO725_DriverNoOpen           = 2
Global Const PISO725_GetDriverVersionError  = 3
Global Const PISO725_InstallIrqError        = 4
Global Const PISO725_ClearIntCountError     = 5
Global Const PISO725_GetIntCountError       = 6
Global Const PISO725_RegisterApcError       = 7
Global Const PISO725_RemoveIrqError         = 8
Global Const PISO725_FindBoardError         = 9
Global Const PISO725_ExceedBoardNumber      = 10
Global Const PISO725_ResetError             = 11

' ID
Global Const PISO_725                 = &H800cff      ' for PISO-725


' The Test functions
Declare Function PISO725_FloatSub Lib "PISO725.DLL" _
        (ByVal a As Single, ByVal b As Single) As Single
Declare Function PISO725_ShortSub Lib "PISO725.DLL" _
        (ByVal a As Integer, ByVal b As Integer) As Integer
Declare Function PISO725_GetDllVersion Lib "PISO725.DLL" () As Integer

' The Driver functions
Declare Function PISO725_DriverInit Lib "PISO725.DLL" () As Integer
Declare Sub PISO725_DriverClose Lib "PISO725.DLL" ()
Declare Function PISO725_SearchCard Lib "PISO725.DLL" _
        ( wBoards As Integer, ByVal dwPIOPISOCardID As Long) As Integer
Declare Function PISO725_GetDriverVersion Lib "PISO725.DLL" _
        ( wDriverVersion As Integer) As Integer
Declare Function PISO725_GetConfigAddressSpace Lib "PISO725.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 PISO725_ActiveBoard Lib "PISO725.DLL" _
        (ByVal wBoardNo As Integer) As Integer
Declare Function PISO725_WhichBoardActive Lib "PISO725.DLL" () As
Integer
```

```
' DIO functions
Declare Sub PISO725_OutputByte Lib "PISO725.DLL" _
        (ByVal address As Long, ByVal dataout As Integer)
Declare Sub PISO725_OutputWord Lib "PISO725.DLL" _
        (ByVal address As Long, ByVal dataout As Long)
Declare Function PISO725_InputByte Lib "PISO725.DLL" _
       (ByVal address As Long) As Integer
Declare Function PISO725_InputWord Lib "PISO725.DLL" _
       (ByVal address As Long) As Long

' Interrupt functions
Declare Function PISO725_IntInstall Lib "PISO725.DLL" _
        (ByVal wBoard As Integer, hEvent As Long ) As Integer
Declare Function PISO725_IntRemove Lib "PISO725.DLL" () As Integer
Declare Function PISO725_IntGetActiveFlag Lib "PISO725.DLL" _
        (bActiveHighFlag as Integer, bActiveLowFlag as Integer) As Integer
```

2.3 PISO725.PAS

```
unit  PISO725;        { PISO725.dll interface unit }

interface

const
  PISO725_NoError                 =0;
  PISO725_DriverOpenError         =1;
  PISO725_DriverNoOpen            =2;
  PISO725_GetDriverVersionError   =3;
  PISO725_InstallIrqError         =4;
  PISO725_ClearIntCountError      =5;
  PISO725_GetIntCountError        =6;
  PISO725_RegisterApcError        =7;
  PISO725_RemoveIrqError          =8;
  PISO725_FindBoardError          =9;
  PISO725_ExceedBoardNumber       =10;
  PISO725_ResetError              =11;

  // ID
  PISO_725                  = $800cff;    // for PISO-725


// Test functions
function  PISO725_FloatSub(fA :   single; fB : single)        : single; StdCall;
function  PISO725_ShortSub(nA : smallint; nB : smallint)   : smallint; StdCall;
function  PISO725_GetDllVersion                                : word; StdCall;

// Driver functions
function  PISO725_DriverInit                                   : word; StdCall;
procedure PISO725_DriverClose                                  ; StdCall;
function  PISO725_SearchCard
        (var wBoards:WORD; dwPIOPISOCardID:LongInt)  : WORD; StdCall;
function  PISO725_GetDriverVersion(var wDriverVer: word) : word; StdCall;
function  PISO725_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  PISO725_ActiveBoard(wBoardNo:Word)               : WORD; StdCall;
function  PISO725_WhichBoardActive                             : WORD; StdCall;
```

```
// DIO functions
procedure PISO725_OutputByte
      (wPortAddr : LongInt; bOutputVal : Word); StdCall;
procedure PISO725_OutputWord
      (wPortAddr : LongInt; wOutputVal : LongInt); StdCall;
function  PISO725_InputByte(wPortAddr  : LongInt )          : Word; StdCall;
function  PISO725_InputWord(wPortAddr  : LongInt )          : LongInt; StdCall;

// Interrupt functions
function  PISO725_IntInstall
      (wBoard:Word; var hEvent:LongInt): Word; StdCall;
function  PISO725_IntRemove                                 : Word; StdCall;
function  PISO725_IntGetActiveFlag
        (var bActiveHighFlag:WORD; var bActiveLowFlag:WORD): Word;
StdCall;


implementation

// Test functions
function  PISO725_FloatSub;
        external 'PISO725.DLL' name 'PISO725_FloatSub';
function  PISO725_ShortSub;
        external 'PISO725.DLL' name 'PISO725_ShortSub';
function  PISO725_GetDllVersion;
        external 'PISO725.DLL' name 'PISO725_GetDllVersion';

// Driver functions
function  PISO725_DriverInit;
        external 'PISO725.DLL' name 'PISO725_DriverInit';
procedure PISO725_DriverClose;
        external 'PISO725.DLL' name 'PISO725_DriverClose';
function  PISO725_SearchCard;
        external 'PISO725.DLL' name 'PISO725_SearchCard';
function  PISO725_GetDriverVersion;
        external 'PISO725.DLL' name 'PISO725_GetDriverVersion';
function  PISO725_GetConfigAddressSpace;
        external 'PISO725.DLL' name 'PISO725_GetConfigAddressSpace';

function  PISO725_ActiveBoard;
        external 'PISO725.DLL' name 'PISO725_ActiveBoard';
function  PISO725_WhichBoardActive;
        external 'PISO725.DLL' name 'PISO725_WhichBoardActive';
```

```
// DIO functions
procedure PISO725_OutputByte;
        external 'PISO725.DLL' name 'PISO725_OutputByte';
procedure PISO725_OutputWord;
        external 'PISO725.DLL' name 'PISO725_OutputWord';
function  PISO725_InputByte;
        external 'PISO725.DLL' name 'PISO725_InputByte';
function  PISO725_InputWord;
        external 'PISO725.DLL' name 'PISO725_InputWord';

// Interrupt functions
function  PISO725_IntInstall;
        external 'PISO725.DLL' name 'PISO725_IntInstall';
function  PISO725_IntRemove;
        external 'PISO725.DLL' name 'PISO725_IntRemove';
function  PISO725_IntGetActiveFlag;
        external 'PISO725.DLL' name 'PISO725_IntGetActiveFlag';


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 PISO725_GetDllVersion

- **Description:**
  To get the version number of PISO725.DLL

- **Syntax:**
  WORD PISO725_GetDllVersion(void)

- **Parameter:**
  None

- **Return:**
  Return the DLL's version number.
  For example: 200(hex) for version 2.00

3.1.2 PISO725_ShortSub

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

- **Syntax:**
  short PISO725_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 PISO725_FloatSub

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

- **Syntax:**
  float PISO725_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 PISO725_OutputByte

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

- **Syntax :**
  void PISO725_OutputByte(DWORD wPortAddr, WORD bOutputVal);

- **Parameter :**
  wPortAddr : [Input] I/O port addresses, please refer to function PISO725_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 PISO725_InputByte

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

- **Syntax :**
  WORD PISO725_InputByte(DWORD wPortAddr);

- **Parameter :**
  wPortAddr : [Input] I/O port addresses, please refer to function PISO725_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 PISO725_OutputWord

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

- **Syntax :**
  void PISO725_OutputWord(DWORD wPortAddr, DWORD wOutputVal);

- **Parameter :**
  wPortAddr : [Input] I/O port addresses, please refer to function PISO725_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 PISO725_InputWord

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

- **Syntax :**
  DWORD PISO725_InputWord(DWORD wPortAddr);

- **Parameter :**
  wPortAddr : [Input] I/O port addresses, please refer to function PISO725_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 PISO725_GetDriverVersion

- **Description :**
  This subroutine will read the version number of PISO725 driver.

- **Syntax :**
  WORD PISO725_GetDriverVersion(WORD *wDriverVersion);

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

- **Return:**
  PISO725_NoError : OK
  PISO725_DriverNoOpen : The PISO725 driver no open
  PISO725_GetDriverVersionError : Read driver version error

3.3.2 PISO725_DriverInit

- **Description :**
  This subroutine will open the PISO725 driver and allocate the resource for the device. This function must be called once before calling other PISO725 functions.

- **Syntax :**
  WORD PISO725_DriverInit();

- **Parameter :**
  None

- **Return:**
  PISO725_NoError : OK
  PISO725_DriverOpenError : open PISO725 Driver error

3.3.3 PISO725_DriverClose

- **Description :**
  This subroutine will close the PISO725 Driver and release the resource from the device. This function must be called once before exit the user's application.

- **Syntax :**
  void PISO725_DriverClose();

- **Parameter :**
  None

- **Return:**
  None

3.3.4 PISO725_GetConfigAddressSpace

- **Description :**
  Get the I/O address of PISO725 board n.

- **Syntax :**
  WORD PISO725_GetConfigAddressSpace
  (WORD wBoardNo, DWORD *wAddrBase, WORD *wIrqNo,
  WORD *wSubVendor, WORD *wSubDevice, WORD *wSubAux,
  WORD *wSlotBus, WORD *wSlotDevice);

- **Parameter :**
  wBoardNo : [Input] PISO-725 board number
  wAddrBase : [Output] The base address of PISO-725 board.
  Only the low WORD is valid.
  wIrqNo : [Output] The IRQ number that the PISO-725 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:**
  PISO725_NoError : OK
  PISO725_FindBoardError : handshake check error
  PISO725_ExceedBoardError : wBoardNo is invalidated

3.4 INTERRUPT FUNCTION

3.4.1 PISO725_IntInstall

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

- **Syntax:**
  WORD PISO725_IntInstall(WORD wBoardNo, HANDLE *hEvent);

- **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.

- **Return:**
  PISO725_NoError : OK
  PISO725_InstallIrqError : IRQ installation error

3.4.2 PISO725_IntRemove

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

- **Syntax:**
  WORD PISO725_IntRemove(void);

- **Parameter:**
  None

- **Return:**
  PISO725_NoError : OK

3.4.3 PISO725_IntGetActiveFlag

- **Description:**

  There are 8 interrupt sources in PISO-725. After the interrupt occurred, users can use this function to read the flag of "bActiveHighFlag" and "bActiveLowFlag".

  The "bActiveHighFlag" indicates the statuses of each channel are changed from low to high. And the "bActiveLowFlag" indicates the statuses of each channel are changed from high to low.

- **Syntax:**

  WORD PISO725_IntGetActiveFlag
  (WORD *bActiveHighFlag, WORD *bActiveLowFlag);

- **Parameter:**

  bActiveHighFlag : [Output] Address of ActiveHigh-Flag.
  Only the low byte is valid.

DI Status	Bit Setting
Changed from low to high	1
keep in high	0
keep in low	0

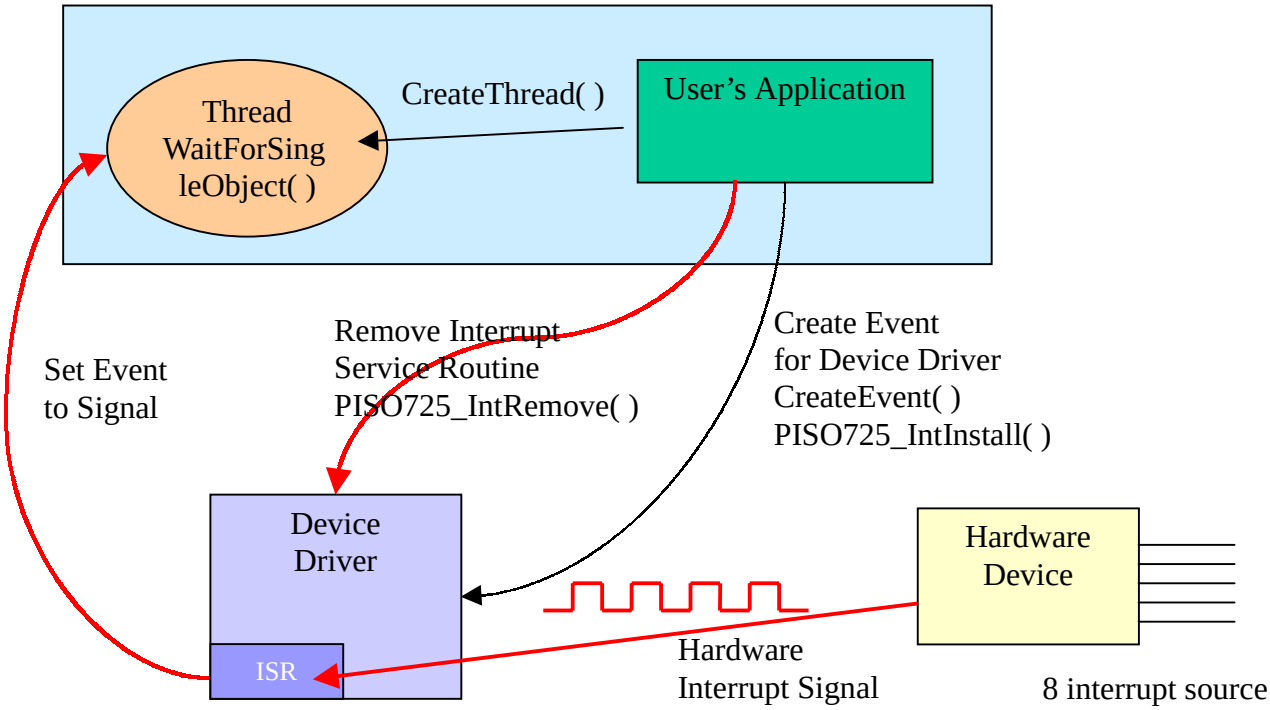
  bActiveLowFlag : [Output] Address of ActiveLow-Flag.
  Only the low byte is valid.

DI Status	Bit Setting
Changed from high to low	1
keep in high	0
keep in low	0

- **Return:**

  PISO725_NoError : OK

3.4.4 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( )

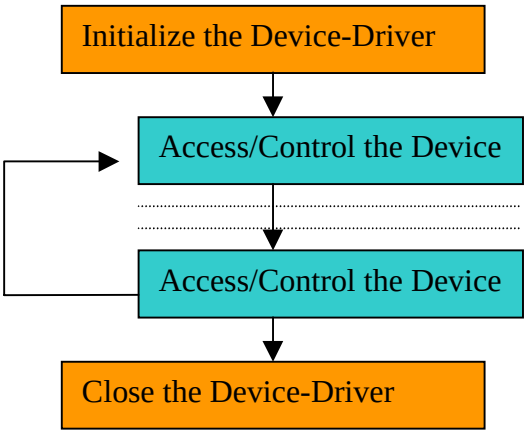
The WaitForSingleObject function returns when one of the following occurs:

- 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.

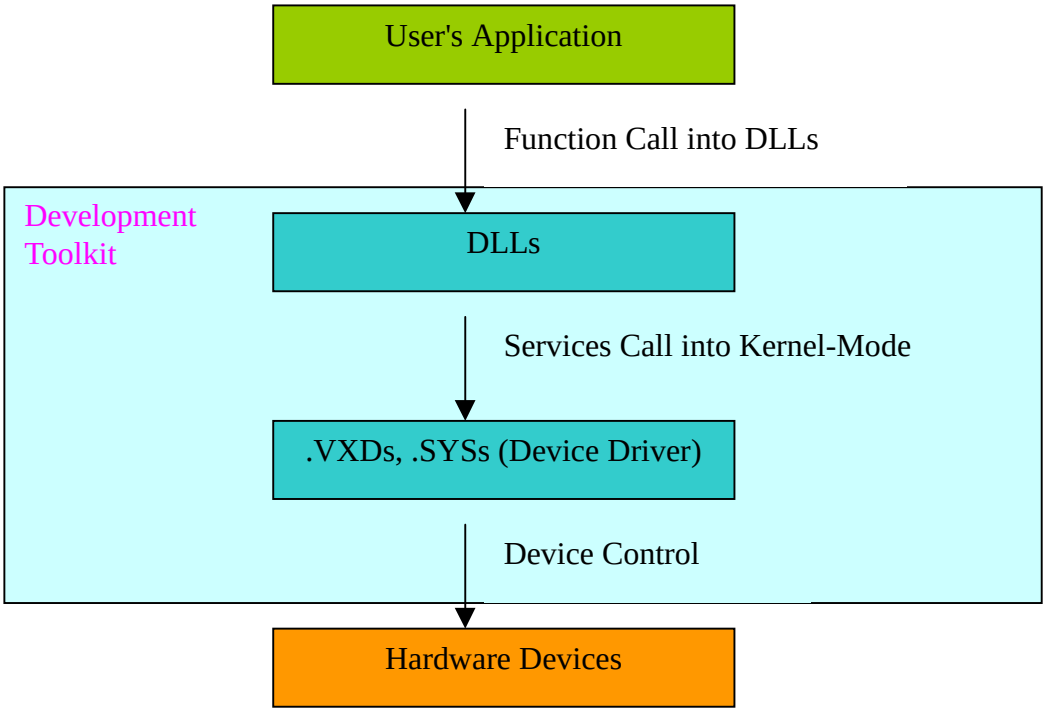
```
DWORD WaitForSingleObject(
        HANDLE hHandle,          // handle to object to wait for
        DWORD dwMilliseconds   // time-out interval in
    milliseconds
);
```

4. Program Architecture




```
PISO725_DriverInit()
// Enable All DI/DO
….
    PISO725_InputByte( … )
    ……..
    ……..
    PISO725_OutputByte(…)
…..

PISO725_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