PIO-DIO Series

Software Manual

[ver. 1.5, April, 2013]

Supports

Board includes PIO-D24/24U/D56/D56U, PIO-D48/D48U/D48SU, PIO-D64/D64U, PIO-D96/D96U/D96SU, PIO-D144/D144U/D144LU, PIO-D168/D168U and PEX-D24/D56/D48. These cards support PIODIO.DLL driver.

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 damages 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, no responsibility is assumed by ICP DAS for its use, nor for any infringements of patents or other rights of third parties resulting from its use.

Copyright

Trademark

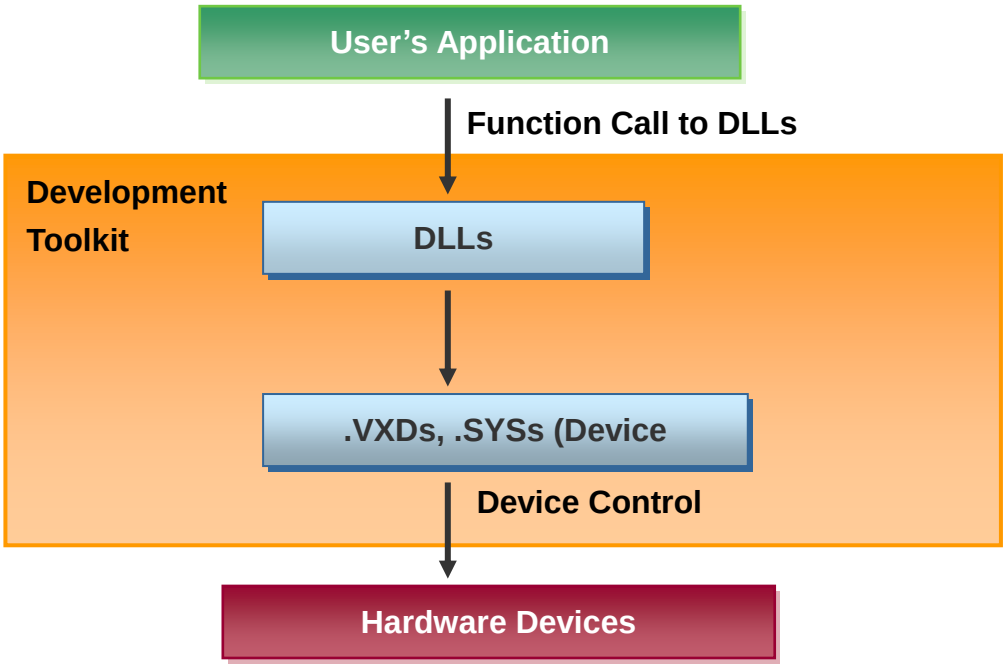
Names are used for identification only and may be registered trademarks of their respective companies.

Table of Contents

1. DLL Function Description

Introduction

The PIODIO.DLL driver is the API function library for the PIO-DIO series card that works the Windows 95/98/ME/NT/2000 and 32-bit Windows XP/2003/Vista/7 systems. The application structure is illustrated in the figure below. User's application programs that calls PIODIO.DLL library in user mode can be developed using various tools, such as VB, VC, Delphi, Borland C++ Builder, VB.NET, VC.NET and C#. The DLL driver then calls the PIO.sys file in order to access the hardware.



Support in ICP DAS Product

The following table shows ICP DAS PIO-DIO Driver DLL that are supported by ICP DAS hardware.

PIO-D24/D56 Series	**PIO-D24/24U/D56/D56U, PEX-D24/D56**
PIO-D48 Series	PIO-D48/D48U/D48SU, PEX-D48
PIO-D64 Series	PIO-D64/D64U
PIO-D96 Series	PIO-D96/D96U/D96SU
PIO-D144 Series	PIO-D144/D144U/D144LU
PIO-D168 Series	PIO-D168/D168U

1.1 Reference

Please refer to the following user manuals:

- **PnPInstall.pdf:**
  Describes how to install the PnP (Plug and Play) driver for PCI card under Windows 95/98/2000/XP/2003/Vista/7(32-bit).
  http://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/manual/

- **SoftInst.pdf:**
  Describes how to install the software package under Windows 95/98/2000/XP/2003/Vista/7(32-bit).
  http://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/manual/

- **CallDll.pdf:**
  Describes how to call the DLL functions with VC++6, VB6, Delphi4 and Borland C++ Builder 4.
  http://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/manual/

- **ResCheck.pdf:**
  Describes how to check the resources I/O Port address, IRQ number and DMA number for add-on cards under Windows 95/98/2000/XP/2003/Vista/7(32-bit).
  http://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/manual/

- **PIO-D56/D24/D48/D64/D96/D144/D168.pdf:**
  PIO-D56/PIO-D24，PIO-D48，PIO-D64，PIO-D96 and PIO-D144/PIO-D168 series hardware manual.
  http://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/pio-dio/manual/

1.2 ErrorCode and ErrorString Table

For the most errors, it is recommended to check:

1. Does the device driver installs successful?
2. Does the card have plugged?
3. Does the card conflicts with other device?
4. Close other applications to free the system resources.
5. Try to use another slot to plug the card.
6. Restart your system to try again.

Error Code	Error ID	Error String
0	PIODIO_NoError	OK (No error)
1	PIODIO_DriverOpenError	Device driver can't be opened
2	PIODIO_DriverNoOpen	The PIODIO_DriverInit() function must be called first
3	PIODIO_GetDriverVersionError	Get driver version error
4	PIODIO_InstallIrqError	Install IRQ error
5	PIODIO_ClearIntCountError	Clear counter value error
6	PIODIO_GetIntCountError	Get interrupt counter error
7	PIODIO_RegisterApcError	Get register APC error
8	PIODIO_RemoveIrqError	Remove IRQ error
9	PIODIO_FindBoardError	Cannot find board
10	PIODIO_ExceedBoardNumber	The max. number of boards is 8
11	PIODIO_ResetError	Can't reset the interrupt count
12	PIODIO_IrqMaskError	Irq-Mask is 1,2,4,8 or 1 to 0xF
13	PIODIO_ActiveModeError	Active Mode is 1,2 or 1 to 3
14	PIODIO_GetActiveFlagError	Can’t get the interrupt active flag
15	PIODIO_ActiveFlagEndOfQueue	The flag queue is empty

2. Declaration Files

After the drivers are installed, the relevant demo programs, development libraries and declaration header files for the different development environments will be available in the following locations.

For detailed PIO-DIO Windows driver installed information, please refer to Quick Start Guide (CD:\NAPDOS\PCI\PIO-DIO\Manual\QuickStart\).
http://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/pio-dio/manual/quickstart/

The demo program is contained in:
CD:\NAPDOS\PCI\PIO-DIO\DLL_OCX\Demo\
http://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/pio-dio/dll_ocx/demo/

- BCB 4 → for Borland C++ Builder 4
  PIODIO.H → Header files
  PIODIO.LIB → Linkage library for BCB only
- Delphi4 → for Delphi 4
  PIODIO.PAS → Declaration files
- VB6 → for Visual Basic 6
  PIODIO.BAS → Declaration files
- VC6 → for Visual C++ 6
  PIODIO.H → Header files
  PIODIO.LIB → Linkage library for VC only
- VB.NET2005 → for VB.NET2005
  PIODIO.vb → Visual Basic Source files
- CSharp2005 → for C#.NET2005
  PIODIO.cs → Visual C# Source files

2.1 PIODIO.H

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

// return code
#define PIODIO_NoError                          0
#define PIODIO_DriverOpenError                  1
#define PIODIO_DriverNoOpen                     2
#define PIODIO_GetDriverVersionError            3
#define PIODIO_InstallIrqError                  4
#define PIODIO_ClearIntCountError               5
#define PIODIO_GetIntCountError                 6
#define PIODIO_RegisterApcError                 7
#define PIODIO_RemoveIrqError                   8
#define PIODIO_FindBoardError                   9
#define PIODIO_ExceedBoardNumber                10
#define PIODIO_ResetError                       11
#define PIODIO_IrqMaskError                     12
#define PIODIO_ActiveModeError                  13
#define PIODIO_GetActiveFlagError               14
#define PIODIO_ActiveFlagEndOfQueue             15

// define the interrupt signal source
#define PIOD144_P2C0        0   // pin29 of CN1(37 pin D-type, pin1 to pin37)
#define PIOD144_P2C1        1   // pin28 of CN1(37 pin D-type, pin1 to pin37)
#define PIOD144_P2C2        2   // pin27 of CN1(37 pin D-type, pin1 to pin37)
#define PIOD144_P2C3        3   // pin26 of CN1(37 pin D-type, pin1 to pin37)

// Interrupt Channel for PIO-D48
#define PIOD48_INTCH0       1   // INT_CHAN_0
#define PIOD48_INTCH1       2   // INT_CHAN_1
#define PIOD48_INTCH2       4   // INT_CHAN_2
#define PIOD48_INTCH3       8   // INT_CHAN_3
// Interrupt ActiveMode for PIOD48_XXX functions
#define PIOD48_ActiveLow    1   // Active When Low
#define PIOD48_ActiveHigh   2   // Active When High
```

```
// to trigger a interrupt when high -> low
#define PIODIO_ActiveLow       0
// to trigger a interrupt when low -> high
#define PIODIO_ActiveHigh      1

// ID
#define PIO_D168               0x98800150    // 168 * D/I/O
#define PIO_D168A                0x800150    // 168A * D/I/O
#define PIO_D144                 0x800100    // 144 * D/I/O
#define PIO_D96                  0x800110    // 96 * D/I/O
#define PIO_D64                  0x800120    // 64 * D/I/O
#define PIO_D56                  0x800140    // D24 + 16I + 16O
#define PIO_D48                  0x800130    // 48 * D/I/O
#define PIO_D24                  0x800140    // 24 * D/I/O

// Test functions
EXPORTS float          CALLBACK PIODIO_FloatSub(float fA, float fB);
EXPORTS short     CALLBACK PIODIO_ShortSub(short nA, short nB);
EXPORTS WORD      CALLBACK PIODIO_GetDllVersion(void);

// Driver functions
EXPORTS WORD      CALLBACK PIODIO_DriverInit(void);
EXPORTS void           CALLBACK PIODIO_DriverClose(void);
EXPORTS WORD           CALLBACK PIODIO_SearchCard(WORD *wBoards, DWORD
                       dwPIOCardID);
EXPORTS WORD           CALLBACK PIODIO_GetDriverVersion(WORD *wDriverVersion);
EXPORTS WORD           CALLBACK PIODIO_GetConfigAddressSpace(WORD  wBoardNo,
                       DWORD *wAddrBase, WORD *wIrqNo,
                        WORD *wSubVendor, WORD *wSubDevice, WORD *wSubAux,
                       WORD *wSlotBus, WORD *wSlotDevice);
EXPORTS WORD    CALLBACK PIODIO_ActiveBoard( WORD wBoardNo );
EXPORTS WORD    CALLBACK PIODIO_WhichBoardActive(void);
```

```
// DIO functions
EXPORTS void   CALLBACK PIODIO_OutputWord(DWORD wPortAddress, DWORD
                    wOutData);
EXPORTS void   CALLBACK PIODIO_OutputByte(DWORD wPortAddr, WORD
                    bOutputValue);
EXPORTS DWORD  CALLBACK PIODIO_InputWord(DWORD wPortAddress);
EXPORTS WORD   CALLBACK PIODIO_InputByte(DWORD wPortAddr);

// Interrupt functions
EXPORTS WORD   CALLBACK PIODIO_IntInstall(WORD wBoardNo, HANDLE *hEvent,
                       WORD wInterruptSource, WORD wActiveMode);
EXPORTS WORD   CALLBACK PIODIO_IntRemove(void);
EXPORTS WORD   CALLBACK PIODIO_IntResetCount(void);
EXPORTS WORD   CALLBACK PIODIO_IntGetCount(DWORD *dwIntCount);

// PIOD48 Counter functions
EXPORTS void   CALLBACK PIOD48_SetCounter(DWORD dwBase, WORD wCounterNo,
                    WORD bCounterMode, DWORD wCounterValue);
EXPORTS DWORD  CALLBACK PIOD48_ReadCounter(DWORD dwBase, WORD
                       wCounterNo, WORD bCounterMode);
EXPORTS void   CALLBACK PIOD48_SetCounterA(WORD wCounterNo, WORD
                    bCounterMode, DWORD wCounterValue);
EXPORTS DWORD  CALLBACK PIOD48_ReadCounterA(WORD wCounterNo, WORD
                       bCounterMode);

// PIOD48 Interrupt functions
EXPORTS WORD   CALLBACK PIOD48_IntInstall(WORD wBoardNo, HANDLE *hEvent,
                       WORD wIrqMask, WORD wActiveMode);
EXPORTS WORD   CALLBACK PIOD48_IntRemove();
EXPORTS WORD   CALLBACK PIOD48_IntGetActiveFlag(WORD *bActiveHighFlag, WORD
                       *bActiveLowFlag);
EXPORTS WORD   CALLBACK PIOD48_IntGetCount(DWORD *dwIntCount);
```

```
// PIOD64 Counter functions
EXPORTS void   CALLBACK PIOD64_SetCounter
       (DWORD dwBase, WORD wCounterNo, WORD bCounterMode, DWORD
wCounterValue);
EXPORTS DWORD  CALLBACK PIOD64_ReadCounter
       (DWORD dwBase, WORD wCounterNo, WORD bCounterMode);
EXPORTS void   CALLBACK PIOD64_SetCounterA
       (WORD wCounterNo, WORD bCounterMode, DWORD wCounterValue);
EXPORTS DWORD  CALLBACK PIOD64_ReadCounterA(WORD wCounterNo, WORD
bCounterMode);

// PIOD48 Frequency Measurement functions
EXPORTS DWORD  CALLBACK PIOD48_Freq(DWORD dwBase);
EXPORTS DWORD  CALLBACK PIOD48_FreqA();
```

2.2 PIOD48u.CPP

```
// ************************************************************
// Initialize the INT2(COut0) Interrupt to High
// this will uses the Counter0 to trigger the interrupt.
//
// wAddrBase       : The base address of PIO-D48,
//                   please refer to function 'PIODIO_GetConfigAddressSpace()'.
// wClockIntConfig : The "Clock/Int Control Register" configuration code,
//                   refer to section "Read/Write Clock/Int Control Register"
//                   in the hardware's manaul.
// wCounter0Config : The configuration code of Counter0
// wCounter0Value  : 0 to &hFFFF, the value is used to set the Counter0
//                   Only the low WORD (16-bits) is valid.
// ************************************************************

void PIOD48u_INT2InitialHigh(DWORD wAddrBase, WORD wClockIntConfig, WORD
        wCounter0Config, DWORD wCounter0Value)
{

   PIODIO_OutputByte(wAddrBase+0xf0, wClockIntConfig);

   //--- program the trigger freq as P2C0 div wCounter0Value   ---
   //--- For example: if freq of P2C0 is 100Hz, then the           ---
   //--- Freq for COut0 as P2C0/wCounter0Value                     ---

  // counter mode ?
  wCounter0Config = (WORD)( (wCounter0Config >> 1) & 0x07 );
  // Counter 0
  PIOD48_SetCounter(wAddrBase, 0, wCounter0Config, wCounter0Value);

}
```

```
// ************************************************************
// Initialize the INT3(COut2) Interrupt to High,
// this will uses the Counter1 and Counter2 to trigger the interrupt.
//
// wAddrBase       : The base address of PIO-D48,
//                   please refer to function 'PIODIO_GetConfigAddressSpace()'.
// wClockIntConfig : The "Clock/Int Control Register" configuration code,
//                   refer to section "Read/Write Clock/Int Control Register"
//                   in the hardware's manaul.
// wCounter1Config : The configuration code of Counter1
// wCounter1Value  : 0 to &hFFFF, the value is used to set the Counter1
//                   Only the low WORD (16-bits) is valid.
// wCounter2Config : The configuration code of Counter2
// wCounter2Value  : 0 to &hFFFF, the value is used to set the Counter2
//                   Only the low WORD (16-bits) is valid.
// ************************************************************

void PIOD48u_INT3InitialHigh(DWORD wAddrBase, WORD wClockIntConfig, WORD
      wCounter1Config, DWORD wCounter1Value, WORD wCounter2Config, DWORD
      wCounter2Value )
{

  PIODIO_OutputByte(wAddrBase+0xf0, wClockIntConfig);

  // Cout2 as ?hz/( wCounter1Value * wCounter2Value)

  // counter mode
  wCounter1Config =(WORD)( (wCounter1Config >> 1) & 0x07 );
 // counter mode
  wCounter2Config =(WORD)( (wCounter2Config >> 1) & 0x07 );
 // Counter 1
  PIOD48_SetCounter(wAddrBase, 1, wCounter1Config, wCounter1Value);
 // Counter 2
  PIOD48_SetCounter(wAddrBase, 2, wCounter2Config, wCounter2Value);

// wait for Cout2 to high
  for( ; ; )
  {
    if( (PIODIO_InputByte(wAddrBase+0x07)&0x08) != 0 )
       break;
  }
}
```

2.3 PIODIO.BAS

```
Attribute VB_Name = "PIODIO"

Global Const PIODIO_NoError                 = 0
Global Const PIODIO_DriverOpenError         = 1
Global Const PIODIO_DriverNoOpen                = 2
Global Const PIODIO_GetDriverVersionError   = 3
Global Const PIODIO_InstallIrqError             = 4
Global Const PIODIO_ClearIntCountError          = 5
Global Const PIODIO_GetIntCountError        = 6
Global Const PIODIO_RegisterApcError        = 7
Global Const PIODIO_RemoveIrqError          = 8
Global Const PIODIO_FindBoardError              = 9
Global Const PIODIO_ExceedBoardNumber   = 10
Global Const PIODIO_ResetError                  = 11
Global Const PIODIO_IrqMaskError            = 12
Global Const PIODIO_ActiveModeError         = 13
Global Const PIODIO_GetActiveFlagError          = 14
Global Const PIODIO_ActiveFlagEndOfQueue  = 15

' define the interrupt signal source
Global Const PIOD144_P2C0 = 0 ' pin29 of CN1(37 pin D-type, pin1 to pin37)
Global Const PIOD144_P2C1 = 1 ' pin28 of CN1(37 pin D-type, pin1 to pin37)
Global Const PIOD144_P2C2 = 2 ' pin27 of CN1(37 pin D-type, pin1 to pin37)
Global Const PIOD144_P2C3 = 3 ' pin26 of CN1(37 pin D-type, pin1 to pin37)

' Interrupt Channel for PIO-D48
Global Const PIOD48_INTCH0 = 1   ' INT_CHAN_0
Global Const PIOD48_INTCH1 = 2   ' INT_CHAN_1
Global Const PIOD48_INTCH2 = 4   ' INT_CHAN_2
Global Const PIOD48_INTCH3 = 8   ' INT_CHAN_3
' Interrupt ActiveMode for PIOD48_XXX functions
Global Const PIOD48_ActiveLow = 1     ' Active When Low
Global Const PIOD48_ActiveHigh = 2    ' Active When High

' to trigger a interrupt when high -> low
Global Const PIODIO_ActiveLow = 0
' to trigger a interrupt when low -> high
Global Const PIODIO_ActiveHigh = 1
```

```
' ID
Global Const PIO_D168 = &H98800150        ' 168 * D/I/O
Global Const PIO_D168A = &H800150         ' 168A * D/I/O
Global Const PIO_D144 = &H800100          ' 144 * D/I/O
Global Const PIO_D96 = &H800110           ' 96 * D/I/O
Global Const PIO_D64 = &H800120           ' 64 * D/I/O
Global Const PIO_D56 = &H800140           ' D24 + 16I + 16O
Global Const PIO_D48 = &H800130           ' 48 * D/I/O
Global Const PIO_D24 = &H800140           ' 24 * D/I/O


' The Test functions
Declare Function PIODIO_ShortSub Lib "PIODIO.dll" (ByVal a As Integer, ByVal b As
                    Integer) As Integer

Declare Function PIODIO_FloatSub Lib "PIODIO.dll"(ByVal a As Single, ByVal b As Single)
                    As Single

Declare Function PIODIO_GetDllVersion Lib "PIODIO.dll" () As Integer

' The Driver functions
Declare Function PIODIO_DriverInit Lib "PIODIO.dll" () As Integer

Declare Sub PIODIO_DriverClose Lib "PIODIO.dll" ()

Declare Function PIODIO_SearchCard Lib "PIODIO.dll"(wBoards As Integer, ByVal
                    dwPIOPISOCardID As Long) As Integer

Declare Function PIODIO_GetDriverVersion Lib "PIODIO.dll(wDriverVersion As Integer)
                   As Integer

Declare Function PIODIO_GetConfigAddressSpace Lib "PIODIO.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 PIODIO_ActiveBoard Lib "PIODIO.dll"(ByVal wBoardNo As Integer) As
                    Integer

Declare Function PIODIO_WhichBoardActive Lib "PIODIO.dll" () As Integer
```

```
' DIO functions
Declare Sub PIODIO_OutputByte Lib "PIODIO.dll" (ByVal address As Long, ByVal dataout
        As Integer)
Declare Sub PIODIO_OutputWord Lib "PIODIO.dll" (ByVal address As Long, ByVal
        dataout As Long)
Declare Function PIODIO_InputByte Lib "PIODIO.dll" (ByVal address As Long) As Integer
Declare Function PIODIO_InputWord Lib "PIODIO.dll" (ByVal address As Long) As Long

' Interrupt functions
Declare Function PIODIO_IntInstall Lib "PIODIO.dll" (ByVal wBoard As Integer, hEvent As
        Long, ByVal wInterruptSource As Integer, ByVal wActiveMode As
        Integer) As Integer
Declare Function PIODIO_IntRemove Lib "PIODIO.dll" () As Integer
Declare Function PIODIO_IntGetCount Lib "PIODIO.dll" (dwIntCount As Long) As Integer
Declare Function PIODIO_IntResetCount Lib "PIODIO.dll" () As Integer

' PIOD48 Counter functions
Declare Sub PIOD48_SetCounter Lib "PIODIO.dll"(ByVal dwBase As Long, ByVal
        wCounterNo As Integer, ByVal bCounterMode As Integer, ByVal
        wCounterValue As Long)
Declare Function PIOD48_ReadCounter Lib "PIODIO.dll"(ByVal dwBase As Long, ByVal
        wCounterNo As Integer, ByVal bCounterMode As Integer) As Long
Declare Sub PIOD48_SetCounterA Lib "PIODIO.dll"(ByVal wCounterNo As Integer, ByVal
        bCounterMode As Integer, ByVal wCounterValue As Long)
Declare Function PIOD48_ReadCounterA Lib "PIODIO.dll"(ByVal wCounterNo As Integer,
        ByVal bCounterMode As Integer) As Long
```

```
' PIOD48 Interrupt functions
Declare Function PIOD48_IntInstall Lib "PIODIO.dll"(ByVal wBoardNo As Integer, hEvent
                    As Long, ByVal wIrqMask As Integer, ByVal wActiveMode As Integer)
                    As Integer

Declare Function PIOD48_IntRemove Lib "PIODIO.dll" () As Integer

Declare Function PIOD48_IntGetActiveFlag Lib "PIODIO.dll" (bActiveHighFlag As Integer,
                    bActiveLowFlag As Integer) As Integer

Declare Function PIOD48_IntGetCount Lib "PIODIO.dll"(dwIntCount As Long) As Integer


' PIOD64 Counter functions

Declare Sub PIOD64_SetCounter Lib "PIODIO.dll" (ByVal dwBase As Long, ByVal
                wCounterNo As Integer, ByVal bCounterMode As Integer, ByVal
                wCounterValue As Long)

Declare Function PIOD64_ReadCounter Lib "PIODIO.dll" (ByVal dwBase As Long, ByVal
                    wCounterNo As Integer, ByVal bCounterMode As Integer) As Long

Declare Sub PIOD64_SetCounterA Lib "PIODIO.dll" (ByVal wCounterNo As Integer, ByVal
                bCounterMode As Integer, ByVal wCounterValue As Long)

Declare Function PIOD64_ReadCounterA Lib "PIODIO.dll"(ByVal wCounterNo As Integer,
                    ByVal bCounterMode As Integer) As Long


' PIOD48 Frequency Measurement Functons

Declare Function PIOD48_Freq  Lib "PIODIO.dll" (ByVal dwBase As Long) As Long

Declare Function PIOD48_FreqA Lib "PIODIO.dll" () As Long
```

2.4 PIOD48u.BAS

```
Attribute VB_Name = "PIOD48u"
Option Explicit

'//************************************************************
'// Initialize the INT2(COut0) Interrupt to High
'// this will uses the Counter0 to trigger the interrupt.
'//
'// wAddrBase       : The base address of PIO-D48,
'//                   please refer to function 'PIODIO_GetConfigAddressSpace()'.
'// wClockIntConfig : The "Clock/Int Control Register" configuration code,
'//                   refer to section "Read/Write Clock/Int Control Register"
'//                   in the hardware's manaul.
'// wCounter0Config : The configuration code of Counter0
'// wCounter0Value  : 0 to &hFFFF, the value is used to set the Counter0
'//                   Only the low WORD (16-bits) is valid.
'//************************************************************

Sub PIOD48u_INT2InitialHigh(ByVal wAddrBase As Long, ByVal wClockIntConfig As
    Integer, ByVal wCounter0Config As Integer, ByVal wCounter0Value As Long)

    PIODIO_OutputByte (wAddrBase + &HF0), wClockIntConfig

   '//--- program the trigger freq as P2C0 div wCounter0Value   ---
   '//--- For example: if freq of P2C0 is 100Hz, then the       ---
   '//--- Freq for COut0 as P2C0/wCounter0Value                 ---

    ' Counter mode
   wCounter0Config = (wCounter0Config \ 2) And &H7
    'Counter 0
   PIOD48_SetCounter wAddrBase, 0, wCounter0Config, wCounter0Value

End Sub
```

```
'//*************************************************************
'// Initialize the INT3(COut2) Interrupt to High,
'// this will uses the Counter1 and Counter2 to trigger the interrupt.
'//
'// wAddrBase       : The base address of PIO-D48,
'//                   please refer to function 'PIODIO_GetConfigAddressSpace()'.
'// wClockIntConfig : The "Clock/Int Control Register" configuration code,
'//                   refer to section "Read/Write Clock/Int Control Register"
'//                   in the hardware's manaul.
'// wCounter1Config : The configuration code of Counter1
'// wCounter1Value  : 0 to &hFFFF, the value is used to set the Counter1
'//                   Only the low WORD (16-bits) is valid.
'// wCounter2Config : The configuration code of Counter2
'// wCounter2Value  : 0 to &hFFFF, the value is used to set the Counter2
'//                   Only the low WORD (16-bits) is valid.
'//*************************************************************

Sub PIOD48u_INT3InitialHigh(ByVal wAddrBase As Long, ByVal wClockIntConfig As
    Integer, ByVal wCounter1Config As Integer, ByVal wCounter1Value As Long, ByVal
    wCounter2Config As Integer, ByVal wCounter2Value As Long)

   PIODIO_OutputByte (wAddrBase + &HF0), wClockIntConfig

   '// Cout2 as ?hz/( wCounter1Value * wCounter2Value)
   wCounter1Config = (wCounter1Config \ 2) And &H7       ' Counter mode
   wCounter2Config = (wCounter2Config \ 2) And &H7       ' Counter mode
   'Counter 1
   PIOD48_SetCounter wAddrBase, 1, wCounter1Config, wCounter1Value
   'Counter 2
   PIOD48_SetCounter wAddrBase, 2, wCounter2Config, wCounter2Value

   '// wait for Cout2 to high
   While (True)
      If ((PIODIO_InputByte(wAddrBase + &H7) And &H8) <> 0) Then
         Exit Sub
      End If
   Wend
End Sub
```

2.5 PIODIO.PAS

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

interface

const
  PIODIO_NoError                =0;
  PIODIO_DriverOpenError            =1;
  PIODIO_DriverNoOpen           =2;
  PIODIO_GetDriverVersionError  =3;
  PIODIO_InstallIrqError        =4;
  PIODIO_ClearIntCountError     =5;
  PIODIO_GetIntCountError       =6;
  PIODIO_RegisterApcError       =7;
  PIODIO_RemoveIrqError             =8;
  PIODIO_FindBoardError             =9;
  PIODIO_ExceedBoardNumber      =10;
  PIODIO_ResetError             =11;
  PIODIO_IrqMaskError           =12;
  PIODIO_ActiveModeError        =13;
  PIODIO_GetActiveFlagError     =14;
  PIODIO_ActiveFlagEndOfQueue   =15;

  // define the interrupt signal source
  PIOD144_P2C0      =0;   // pin29 of CN1(37 pin D-type, pin1 to pin37)
  PIOD144_P2C1      =1;   // pin28 of CN1(37 pin D-type, pin1 to pin37)
  PIOD144_P2C2      =2;   // pin27 of CN1(37 pin D-type, pin1 to pin37)
  PIOD144_P2C3      =3;   // pin26 of CN1(37 pin D-type, pin1 to pin37)

  // Interrupt Channel for PIO-D48
  PIOD48_INTCH0       =1;   // INT_CHAN_0
  PIOD48_INTCH1       =2;   // INT_CHAN_1
  PIOD48_INTCH2       =4;   // INT_CHAN_2
  PIOD48_INTCH3       =8;   // INT_CHAN_3
  // Interrupt ActiveMode for PIOD48_XXX functions
  PIOD48_ActiveLow    =1;   // Active When Low
  PIOD48_ActiveHigh   =2;   // Active When High
```

```
 // to trigger a interrupt when high -> low
 PIODIO_ActiveLow              =0;
 // to trigger a interrupt when low -> high
 PIODIO_ActiveHigh             =1;

 // ID
 PIO_D168             = $98800150;   // 168 * D/I/O
 PIO_D168A              = $800150;   // 168A * D/I/O
 PIO_D144               = $800100;   // 144 * D/I/O
 PIO_D96                = $800110;   // 96 * D/I/O
 PIO_D64                = $800120;   // 64 * D/I/O
 PIO_D56                = $800140;   // D24 + 16I + 16O
 PIO_D48                = $800130;   // 48 * D/I/O
 PIO_D24                = $800140;   // 24 * D/I/O

// Test functions
function  PIODIO_ShortSub(nA : smallint; nB : smallint) : smallint; StdCall;
function  PIODIO_FloatSub(fA : single; fB : single): single; StdCall;
function  PIODIO_GetDllVersion : word; StdCall;

// Driver functions
function  PIODIO_DriverInit: word; StdCall;
procedure PIODIO_DriverClose; StdCall;
function  PIODIO_SearchCard(var wBoards:WORD; dwPIOPISOCardID:LongInt): WORD;
          StdCall;
function  PIODIO_GetDriverVersion(var wDriverVer: word): word; StdCall;
function  PIODIO_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  PIODIO_ActiveBoard(wBoardNo:Word): WORD; StdCall;
function  PIODIO_WhichBoardActive: WORD; StdCall;

// DIO functions
procedure PIODIO_OutputByte(wPortAddr : LongInt; bOutputVal : Word);       StdCall;
procedure PIODIO_OutputWord(wPortAddr : LongInt; wOutputVal : LongInt);
           StdCall;
function  PIODIO_InputByte(wPortAddr  : LongInt ) : word; StdCall;
function  PIODIO_InputWord(wPortAddr  : LongInt ) : LongInt; StdCall;
```

```
// Interrupt functions
function  PIODIO_IntInstall(wBoard:Word; var hEvent:LongInt; wInterruptSource:Word;
            wActiveMode:Word) : Word; StdCall;
function  PIODIO_IntRemove: WORD; StdCall;
function  PIODIO_IntGetCount(var dwIntCount:LongInt): WORD; StdCall;
function  PIODIO_IntResetCount : WORD; StdCall;

// PIOD48 Counter functions
procedure PIOD48_SetCounter( dwBase:LongInt; wCounterNo:WORD;
          bCounterMode:WORD; wCounterValue:LongInt); StdCall;
function PIOD48_ReadCounter( dwBase:LongInt;    wCounterNo:WORD;
         bCounterMode:WORD):LongInt; StdCall;
procedure PIOD48_SetCounterA(wCounterNo:WORD; bCounterMode:WORD;
            wCounterValue:LongInt); StdCall;
function PIOD48_ReadCounterA(wCounterNo:WORD; bCounterMode:WORD):LongInt;
          StdCall;

// PIOD48 Interrupt functions
function PIOD48_IntInstall( wBoardNo:WORD; var hEvent:LongInt; wIrqMask:WORD;
          wActiveMode:WORD): WORD; StdCall;
function PIOD48_IntRemove: WORD; StdCall;
function PIOD48_IntGetActiveFlag(var bActiveHighFlag:WORD; var
          bActiveLowFlag:WORD): WORD; StdCall;
function PIOD48_IntGetCount(var dwIntCount:LongInt): WORD; StdCall;

// PIOD64 Counter functions
procedure PIOD64_SetCounter( dwBase:LongInt; wCounterNo:WORD;
                    bCounterMode:WORD; wCounterValue:LongInt); StdCall;
function PIOD64_ReadCounter( dwBase:LongInt;  wCounterNo:WORD;
                    bCounterMode:WORD):LongInt; StdCall;
procedure PIOD64_SetCounterA(wCounterNo:WORD; bCounterMode:WORD;
            wCounterValue:LongInt); StdCall;
function PIOD64_ReadCounterA(wCounterNo:WORD; bCounterMode: WORD) :LongInt;
          StdCall;
```

```
// PIOD48 Frequency Measurement functions
function  PIOD48_Freq(dwBase:LongInt) :LongInt; StdCall;
function  PIOD48_FreqA :LongInt; StdCall;

implementation

// Test functions
function  PIODIO_ShortSub; external 'PIODIO.DLL' name 'PIODIO_ShortSub';
function  PIODIO_FloatSub;  external 'PIODIO.DLL' name 'PIODIO_FloatSub';
function  PIODIO_GetDllVersion; external 'PIODIO.DLL' name 'PIODIO_GetDllVersion';

// Driver functions
function  PIODIO_DriverInit;    external 'PIODIO.DLL' name 'PIODIO_DriverInit';
procedure PIODIO_DriverClose; external 'PIODIO.DLL' name 'PIODIO_DriverClose';
function  PIODIO_SearchCard; external 'PIODIO.DLL' name 'PIODIO_SearchCard';
function  PIODIO_GetDriverVersion; external 'PIODIO.DLL' name
                                              'PIODIO_GetDriverVersion';
function  PIODIO_GetConfigAddressSpace;  external 'PIODIO.DLL' name
                                                  'PIODIO_GetConfigAddressSpace';
function  PIODIO_ActiveBoard; external 'PIODIO.DLL' name 'PIODIO_ActiveBoard';
function  PIODIO_WhichBoardActive; external 'PIODIO.DLL' name
                                               'PIODIO_WhichBoardActive';

// DIO functions
procedure PIODIO_OutputByte; external 'PIODIO.DLL' name 'PIODIO_OutputByte';
procedure PIODIO_OutputWord; external 'PIODIO.DLL' name 'PIODIO_OutputWord';
function  PIODIO_InputByte; external 'PIODIO.DLL' name 'PIODIO_InputByte';
function  PIODIO_InputWord; external 'PIODIO.DLL' name 'PIODIO_InputWord';

// Interrupt functions
function  PIODIO_IntInstall; external 'PIODIO.DLL' name 'PIODIO_IntInstall';
function  PIODIO_IntRemove; external 'PIODIO.DLL' name 'PIODIO_IntRemove';
function  PIODIO_IntGetCount; external 'PIODIO.DLL' name 'PIODIO_IntGetCount';
function  PIODIO_IntResetCount; external 'PIODIO.DLL' name 'PIODIO_IntResetCount';
```

```
// PIOD48 Counter functions
procedure PIOD48_SetCounter; external 'PIODIO.DLL' name 'PIOD48_SetCounter';
function  PIOD48_ReadCounter; external 'PIODIO.DLL' name 'PIOD48_ReadCounter';
procedure PIOD48_SetCounterA; external 'PIODIO.DLL' name 'PIOD48_SetCounterA';
function  PIOD48_ReadCounterA; external 'PIODIO.DLL' name 'PIOD48_ReadCounterA';

// PIOD48 Interrupt functions
function PIOD48_IntInstall;  external 'PIODIO.DLL' name 'PIOD48_IntInstall';
function PIOD48_IntRemove; external 'PIODIO.DLL' name 'PIOD48_IntRemove';
function PIOD48_IntGetActiveFlag; external 'PIODIO.DLL' name
                                              'PIOD48_IntGetActiveFlag';
function PIOD48_IntGetCount; external 'PIODIO.DLL' name 'PIOD48_IntGetCount';

// PIOD64 Counter functions
procedure PIOD64_SetCounter; external 'PIODIO.DLL' name 'PIOD64_SetCounter';
function  PIOD64_ReadCounter; external 'PIODIO.DLL' name 'PIOD64_ReadCounter';
procedure PIOD64_SetCounterA; external 'PIODIO.DLL' name 'PIOD64_SetCounterA';
function  PIOD64_ReadCounterA; external 'PIODIO.DLL' name 'PIOD64_ReadCounterA';

// PIOD48 Frequency Measurement functions
function  PIOD48_Freq; external 'PIODIO.DLL' name 'PIOD48_Freq';
function  PIOD48_FreqA; external 'PIODIO.DLL' name 'PIOD48_FreqA';

end.
```

2.6 PIOD48u.PAS

```
unit  PIOD48u;

interface

procedure PIOD48u_INT2InitialHigh(wAddrBase:LongInt;
                wClockIntConfig:WORD;wCounter0Config:WORD;
                wCounter0Value:LongInt);
procedure PIOD48u_INT3InitialHigh(wAddrBase:LongInt;
                wClockIntConfig:WORD;wCounter1Config:WORD;
                wCounter1Value:LongInt;wCounter2Config:WORD;
                wCounter2Value:LongInt);

implementation

uses PIODIO;

//*********************************************************
// Initialize the INT2(COut0) Interrupt to High
// this will uses the Counter0 to trigger the interrupt.
//
// wAddrBase       : The base address of PIO-D48,
//                   please refer to function 'PIODIO_GetConfigAddressSpace()'.
// wClockIntConfig : The "Clock/Int Control Register" configuration code,
//                   refer to section "Read/Write Clock/Int Control Register"
//                   in the hardware's manaul.
// wCounter0Config : The configuration code of Counter0
// wCounter0Value  : 0 to &hFFFF, the value is used to set the Counter0
//                   Only the low WORD (16-bits) is valid.
//*********************************************************

procedure PIOD48u_INT2InitialHigh(wAddrBase:LongInt;
                wClockIntConfig:WORD;wCounter0Config:WORD;
                wCounter0Value:LongInt);
begin
   PIODIO_OutputByte(wAddrBase+$f0, wClockIntConfig);

   //--- program the trigger freq as P2C0 div wCounter0Value   ---
   //--- For example: if freq of P2C0 is 100Hz, then the       ---
   //--- Freq for COut0 as P2C0/wCounter0Value                 ---

// Counter mode
   wCounter0Config := ( wCounter0Config shr 1 ) and $7;
  // Counter 0
   PIOD48_SetCounter(wAddrBase, 0, wCounter0Config, wCounter0Value);

 end;
```

```
//*********************************************************
// Initialize the INT3(COut2) Interrupt to High,
// this will uses the Counter1 and Counter2 to trigger the interrupt.
//
// wAddrBase       : The base address of PIO-D48,
//                   please refer to function 'PIODIO_GetConfigAddressSpace()'.
// wClockIntConfig : The "Clock/Int Control Register" configuration code,
//                   refer to section "Read/Write Clock/Int Control Register"
//                   in the hardware's manaul.
// wCounter1Config : The configuration code of Counter1
// wCounter1Value  : 0 to &hFFFF, the value is used to set the Counter1
//                   Only the low WORD (16-bits) is valid.
// wCounter2Config : The configuration code of Counter2
// wCounter2Value  : 0 to &hFFFF, the value is used to set the Counter2
//                    Only the low WORD (16-bits) is valid.
// *********************************************************

procedure PIOD48u_INT3InitialHigh(wAddrBase:LongInt;
                wClockIntConfig:WORD;wCounter1Config:WORD;
                wCounter1Value:LongInt;wCounter2Config:WORD;
                wCounter2Value:LongInt);
begin

  PIODIO_OutputByte(wAddrBase+$f0, BYTE(wClockIntConfig) );

  // Cout2 as ?hz/( wCounter1Value * wCounter2Value)
  wCounter1Config := ( wCounter1Config shr 1 ) and $7; // Counter mode
  wCounter2Config := ( wCounter2Config shr 1 ) and $7; // Counter mode
 // Counter 1
  PIOD48_SetCounter(wAddrBase, 1, wCounter1Config, wCounter1Value);
 // Counter 2
  PIOD48_SetCounter(wAddrBase, 2, wCounter2Config, wCounter2Value);

 // wait for Cout2 to high
  while ( true ) do
  begin
    if( (PIODIO_InputByte(wAddrBase+$07) and $08) <> 0 ) then
       exit;
  end;
end;


end.
```

2.7 PIODIO.VB

```
Module PIODIO

 'Return Code
Public Const PIODIO_NoError = 0
Public Const PIODIO_DriverOpenError = 1
Public Const PIODIO_DriverNoOpen= 2
Public Const PIODIO_GetDriverVersionError = 3
Public Const PIODIO_InstallIrqError = 4
Public Const PIODIO_ClearIntCountError = 5
Public Const PIODIO_GetIntCountError = 6
Public Const PIODIO_RegisterApcError = 7
Public Const PIODIO_RemoveIrqError = 8
Public Const PIODIO_FindBoardError = 9
Public Const PIODIO_ExceedBoardNumber = 10
Public Const PIODIO_ResetError = 11
Public Const PIODIO_IrqMaskError = 12
Public Const PIODIO_ActiveModeError = 13
Public Const PIODIO_GetActiveFlagError = 14
Public Const PIODIO_ActiveFlagEndOfQueue = 15

'Define the Interrupt Signal Source
Public Const PIOD144_P2C0 = 0   'pin29 of CN1(37 pin D-type, pin1 to pin37)
Public Const PIOD144_P2C1 = 1   'pin28 of CN1(37 pin D-type, pin1 to pin37)
Public Const PIOD144_P2C2 = 2   'pin27 of CN1(37 pin D-type, pin1 to pin37)
Public Const PIOD144_P2C3 = 3   'pin26 of CN1(37 pin D-type, pin1 to pin37)

' Interrupt Channel for PIO-D48
Public Const PIOD48_INTCH0 = 1   ' INT_CHAN_0
Public Const PIOD48_INTCH1 = 2   ' INT_CHAN_1
Public Const PIOD48_INTCH2 = 4   ' INT_CHAN_2
Public Const PIOD48_INTCH3 = 8   ' INT_CHAN_3

'Interrupt ActiveMode for PIOD48_XXX functions
Public Const PIOD48_ActiveLow = 1    ' Active When Low
Public Const PIOD48_ActiveHigh = 2    ' Active When High

'to trigger a interrupt when high -> low
Public Const ActiveLow = 0
'to trigger a interrupt when low -> high
Public Const ActiveHigh = 1
```

```
'Card ID
Public Const PIO_D24 = &H800140
Public Const PIO_D48 = &H800130
Public Const PIO_D56 = &H800140
Public Const PIO_D64 = &H800120
Public Const PIO_D96 = &H800110
Public Const PIO_D144 = &H800100
Public Const PIO_D168 = &H98800150
Public Const PIO_D168A = &H800150

'Test Function
<DllImport("Piodio.dll")> _
Public Function PIODIO_FloatSub(ByVal fA As Single, ByVal fB As Single) As Single

End Function
-------------------------------------------------------------------------------
<DllImport("Piodio.dll")> _
Public Function PIODIO_ShortSub(ByVal nA As Short, ByVal nB As Short) As Short

End Function
-------------------------------------------------------------------------------
<DllImport("Piodio.dll")> _
Public Function PIODIO_GetDllVersion() As Short

End Function
-------------------------------------------------------------------------------
'Driver Function
<DllImport("Piodio.dll")> _
Public Function PIODIO_DriverInit() As Short

End Function
-------------------------------------------------------------------------------
<DllImport("Piodio.dll")> _
Public Sub PIODIO_DriverClose()

End Sub
-------------------------------------------------------------------------------
<DllImport("Piodio.dll")> _
Public Function PIODIO_SearchCard(ByRef wBoards As Short, ByVal dwPIOCardID As
                    Integer) As Short

End Function
-------------------------------------------------------------------------------
<DllImport("Piodio.dll")> _
Public Function PIODIO_GetDriverVersion(ByRef wDriverVersion As Short) As Short

End Function
-------------------------------------------------------------------------------
```

```
<DllImport("Piodio.dll")> _
Public Function PIODIO_GetConfigAddressSpace(ByVal wBoards As Short, ByRef
                    wAddrBase As Integer, ByRef wIrqNo As Short, ByRef wSubVendor As
                    Short, ByRef wSubDevice As Short, ByRef wSubAux As Short, ByRef
                    wSlotBus As Short, ByRef wSlotDevice As Short) As Short

End Function
-------------------------------------------------------------------------------
<DllImport("Piodio.dll")> _
Public Function PIODIO_ActiveBoard(ByVal wBoardNo As Short) As Short

End Function
-------------------------------------------------------------------------------
<DllImport("Piodio.dll")> _
Public Function PIODIO_WhichBoardActive() As Short

End Function
-------------------------------------------------------------------------------
'**************************'
'DIO Function                '
'**************************'
<DllImport("Piodio.dll")> _
Public Sub PIODIO_OutputByte(ByVal wBaseAddr As Integer, ByVal bOutputValue As
Short)

End Sub
-------------------------------------------------------------------------------
<DllImport("Piodio.dll")> _
Public Function PIODIO_InputByte(ByVal wBaseAddr As Integer) As Short

End Function
-------------------------------------------------------------------------------
'**************************'
'Interrupt Function          '
'**************************'
<DllImport("piodio.dll")> _
Public Function PIODIO_IntInstall(ByVal wboards As Short, ByRef hEvent As Integer,
                    ByVal wInterruptSource As Short, ByVal wActiveMode As Short) As
                    Short

End Function
-------------------------------------------------------------------------------
<DllImport("piodio.dll")> _
Public Function PIODIO_IntRemove() As Short

End Function
-------------------------------------------------------------------------------
<DllImport("piodio.dll")> _
Public Function PIODIO_IntGetCount(ByRef intIntCount As Integer) As Short

End Function
-------------------------------------------------------------------------------
```

```
<DllImport("Piodio.dll")> _
Public Function PIODIO_IntResetCount() As Short

End Function
---------------------------------------------------------------------------------
'**************************************'
'PIOD48 Counter Function                '
'**************************************'
<DllImport("Piodio.dll")> _
Public Sub PIOD48_SetCounter(ByVal dwBase As Integer, ByVal wCounterNo As Short,
ByVal bCounterMode As Short, ByVal wCounterValue As Integer)

End Sub
---------------------------------------------------------------------------------
<DllImport("Piodio.dll")> _
Public Function PIOD48_ReadCounter(ByVal dwBase As Integer, ByVal wCounterNo As
Short, ByVal bCounterMode As Short) As Integer

End Function
---------------------------------------------------------------------------------
<DllImport("Piodio.dll")> _
Public Sub PIOD48_SetCounterA(ByVal wCounterNo As Short, ByVal bCounterMode As
Short, ByVal wCounterValue As Integer)

End Sub
---------------------------------------------------------------------------------
<DllImport("Piodio.dll")> _
Public Function PIOD48_ReadCounterA(ByVal wCounterNo As Short, ByVal
bCounterMode As Short) As Integer

End Function
---------------------------------------------------------------------------------
'**************************************'
'PIOD48 Interrupt Function              '
'**************************************'
<DllImport("Piodio.dll")> _
Public Function PIOD48_IntInstall(ByVal wBoardNo As Short, ByRef hEvent As Integer,
                    ByVal wIrqMask As Short, ByVal wActiveMode As Short) As Short

End Function
---------------------------------------------------------------------------------
<DllImport("Piodio.dll")> _
Public Function PIOD48_IntRemove() As Short

End Function
---------------------------------------------------------------------------------
<DllImport("Piodio.dll")> _
Public Function PIOD48_IntGetActiveFlag(ByRef bActiveHighFlag As Short, ByRef
                    bActiveLowFlag As Short) As Short

End Function
---------------------------------------------------------------------------------
<DllImport("Piodio.dll")> _
Public Function PIOD48_IntGetCount(ByRef dwIntCount As Integer) As Short

End Function
---------------------------------------------------------------------------------
```

```
'*************************************'
'PIOD64 Counter Function              '
'*************************************'
<DllImport("Piodio.dll")> _
Public Sub PIOD64_SetCounter(ByVal dwBase As Integer, ByVal wCounterNo As Short,
ByVal bCounterMode As Short, ByVal wCounterValue As Integer)

End Sub
---------------------------------------------------------------------------------
<DllImport("Piodio.dll")> _
Public Function PIOD64_ReadCounter(ByVal dwBase As Integer, ByVal wCounterNo As
Short, ByVal bCounterMode As Short) As Integer

End Function
---------------------------------------------------------------------------------
<DllImport("Piodio.dll")> _
Public Sub PIOD64_SetCounterA(ByVal wCounterNo As Short, ByVal bCounterMode As
Short, ByVal wCounterValue As Integer)

End Sub
---------------------------------------------------------------------------------
<DllImport("Piodio.dll")> _
Public Function PIOD64_ReadCounterA(ByVal wCounterNo As Short, ByVal
bCounterMode As Short) As Integer

End Function
---------------------------------------------------------------------------------
'*************************************'
'PIOD48 frequence MeasurementFunction  '
'*************************************'
<DllImport("Piodio.dll")> _
Public Function PIOD48_Freq(ByVal wAddrBase As Integer) As Integer

End Function
End Module
---------------------------------------------------------------------------------
```

2.8 PIODIO.cs

```
namespace PIODIO_Ns
{
   public class PIODIO
   {

      //****************
      //PIODIO CARD ID
      //****************
      public const uint PIOD_24=0x800140;
      public const uint PIOD_48=0x800130;
      public const uint PIOD_56=0x800140;
      public const uint PIOD_64=0x800120;
      public const uint PIOD_96=0x800110;
      public const uint PIOD_144=0x800100;
      public const uint PIOD_168=0x98800150;
      public const uint PIOD_168A=0x800150;

      //****************
      //Error Code
      //****************
      public const uint NoError = 0;
      public const uint DriverOpenError = 1;
      public const uint DriverNoOpen = 2;
      public const uint GetDriverVersionError = 3;
      public const uint InstallIrqError = 4;
      public const uint ClearIntCountError = 5;
      public const uint GetIntCountError = 6;
      public const uint RegisterApcError = 7;
      public const uint RemoveIrqError = 8;
      public const uint FindBoardError = 9;
      public const uint ExceedBoardNumber = 10;
      public const uint ResetError = 11;
      public const uint IrqMaskError = 12;
      public const uint ActiveModeError = 13;
      public const uint GetActiveFlagError = 14;
      public const uint ActiveFlagEndOfQueue = 15;

      //******************
      //PIODIO ActiveMode
      //******************

      // to trigger a interrupt when low -> high
      public const uint ActiveHigh =1;

      // to trigger a interrupt when high -> low
      public const uint  ActiveLow=0;
```

```
//**********************************
//define the interrupt signal source
//**********************************
public const uint PIOD144_P2C0 = 0; //pin29 of CN1(37 pin D-type,
                                      pin1 to pin37)
public const uint PIOD144_P2C1 = 1; //pin28 of CN1(37 pin D-type,
                                      pin1 to pin37)
public const uint PIOD144_P2C2 = 2; //pin27 of CN1(37 pin D-type,
                                      pin1 to pin37)
public const uint PIOD144_P2C3 = 3; //pin26 of CN1(37 pin D-type,
                                      pin1 to pin37)

//**********************************
// Interrupt Channel for PIO-D48
//**********************************
public const uint PIOD48_INTCH0 = 1;  // INT_CHAN_0
public const uint PIOD48_INTCH1 = 2;  // INT_CHAN_1
public const uint PIOD48_INTCH2 = 4;  // INT_CHAN_2
public const uint PIOD48_INTCH3 = 8;  // INT_CHAN_3

 //**********************************
//Test functions
//**********************************

[DllImport ("Piodio.dll",EntryPoint ="PIODIO_FloatSub")]
 public static extern float FloatSub(float fA,float fB);
[DllImport ("Piodio.dll",EntryPoint ="PIODIO_ShortSub")]
 public static extern short ShortSub(short nA,short nB);

[DllImport ("Piodio.dll",EntryPoint ="PIODIO_GetDllVersion")]
 public static extern ushort GetDllVersion();

//**************
// PIODIO Driver
//**************
[DllImport("Piodio.dll",EntryPoint="PIODIO_DriverInit")]
 public static extern ushort DriverInit();

 [DllImport("Piodio.dll",EntryPoint="PIODIO_DriverClose")]
 public static extern void DriverClose();
 [DllImport("Piodio.dll",EntryPoint="PIODIO_SearchCard")]
 public static extern ushort SearchCard(out ushort wBoards, uint dwPIOCardID);
 [DllImport ("Piodio.dll",EntryPoint ="PIODIO_GetDriverVision")]
 public static extern ushort GetDriverVersion(out ushort wDriverVersion);


 [DllImport("Piodio.dll",EntryPoint="PIODIO_GetConfigAddressSpace")]
 public static extern ushort GetConfigAddressSpace(
     ushort wBoardNo, out uint wAddrBase, out ushort wIrqNo,
     out ushort wSubVendor, out ushort wSubDevice, out ushort wSubAux,
     out ushort wSlotBus, out ushort wSlotDevice);
 [DllImport("Piodio.dll",EntryPoint="PIODIO_ActiveBoard")]
 public static extern ushort ActiveBoard(ushort wBoardNo);
 [DllImport("Piodio.dll",EntryPoint="PIODIO_WhichBoardActive")]
 public static extern ushort WhichBoardActive();
```

```
// ****************************************
[DllImport("Piodio.dll",EntryPoint="PIODIO_OutputByte")]
public static extern void OutputByte(uint wBaseAddr, ushort bOutputValue);
[DllImport("Piodio.dll",EntryPoint="PIODIO_InputByte")]
public static extern ushort InputByte(uint wBaseAddr);

 //*******************
//PIODIO Interrupt
//*******************

[DllImport("Piodio.dll", EntryPoint = "PIODIO_IntInstall")]
public static extern ushort IntInstall(ushort wBoardNo, out uint hEvent, ushort
                                wInterruptSource, ushort wActiveMode);
[DllImport("Piodio.dll", EntryPoint = "PIODIO_IntRemove")]
public static extern ushort IntRemove();

[DllImport("Piodio.dll", EntryPoint = "PIODIO_IntGetCount")]
public static extern ushort IntGetCount(out uint dwIntCount);

[DllImport("Piodio.dll", EntryPoint = "PIODIO_IntResetCount")]
public static extern ushort IntResetCount();

//*******************
//PIODIO_48 Frequency
//*******************
[DllImport("Piodio.dll")]
public static extern uint PIOD48_Freq(uint wBaseAddr);

 //**********************
//PIODIO_48 Counter
//**********************
[DllImport("Piodio.dll")]
public static extern void PIOD48_SetCounter(uint dwBase,ushort
          wCounterNo,ushort bCounterMode,uint wCounterValue );
[DllImport("Piodio.dll")]
public static extern uint PIOD48_ReadCounter(uint dwBase,ushort
          wCounterNo,ushort bCounterMode);
[DllImport ("Piodio.dll")]
public static extern void PIOD48_SetCounterA(ushort wCounterNo, ushort
           bCounterMode,uint wCounterValue);
[DllImport ("Piodio.dll")]
public static extern uint PIOD48_ReadCounterA(ushort wCounterNo,ushort
           bCounterMode);

//**********************
//PIODIO_48 Interrupt
//**********************
[DllImport ("Piodio.dll")]
public static extern ushort PIOD48_IntInstall(ushort wBoardNo, out uint hEvent,
                              ushort  wIrqMask, ushort  wActiveMode);
[DllImport ("Piodio.dll")]
public static extern ushort PIOD48_IntRemove();
[DllImport ("Piodio.dll")]
public static extern ushort PIOD48_IntGetActiveFlag(out ushort bActiveHighFlag,
                         out ushort bActiveLowFlag);
[DllImport ("Piodio.dll")]
public static extern ushort PIOD48_IntGetCount(out uint dwIntCount);
```

```
        //*********************
       //PIODIO_64 Counter
       //*********************
       [DllImport("Piodio.dll")]
       public static extern void  PIOD64_SetCounter(uint dwBase,ushort
                    wCounterNo,ushort bCounterMode,uint wCounterValue);
       [DllImport("Piodio.dll")]
       public static extern uint PIOD64_ReadCounter(uint dwBase,ushort
                                          wCounterNo,ushort bCounterMode);
       [DllImport("Piodio.dll")]
       public static extern void PIOD64_SetCounterA(ushort wCounterNo, ushort
                                    bCounterMode,uint wCounterValue);
       [DllImport("Piodio.dll")]
       public static extern uint PIOD64_ReadCounterA(ushort wCounterNo, ushort
                                   bCounterMode);

       // ****************************************
          private int DriverOpened = 0;

       // ****************************************
       //public void OutputByte(ushort wBaseAddr, ushort bValue)
       //{
        //   OutputByte(wBaseAddr, bValue);
       //}
       //public ushort InputByte(ushort wBaseAddr)
       //{
        //   return InputByte(wBaseAddr);
       //}

       public PIODIO()//constroctor
       {
          DriverOpened = 0;
       }
       ~PIODIO()
       {
          if (DriverOpened != 0)
          {
             DriverOpened = 0;
             DriverClose();
          }
       }
    }
}
```

3. Function Descriptions

In order to simplify and clarify the description, the attribute of the input and output parameters of the function is indicated as [input] and [output], respectively, as shown in the following table.

Keyword	Parameter must be set by the user before calling the function	Data/value from this parameter is retrieved after calling the function
[Input]	**Yes**	No
[Output]	No	**Yes**
[Input, Output]	**Yes**	**Yes**

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

Reference	Function Definition
Functions of Test	
Sec. 3.1.1	WORD PIODIO_GetDllVersion(void);
Sec. 3.1.2	short PIODIO_ShortSub(short nA, short nB);
Sec. 3.1.3	float PIODIO_FloatSub(float fA, float fB);
Digital I/O Functions	
Sec. 3.2.1	void PIODIO_OutputByte(DWORD wPortAddr, WORD bOutputValue);
Sec. 3.2.2	WORD PIODIO_InputByte(DWORD wPortAddr);
Sec. 3.2.3	void PIODIO_OutputWord(DWORD wPortAddress, DWORD wOutData);
Sec. 3.2.4	DWORD PIODIO_InputWord(DWORD wPortAddress);
Driver Relative Interrupt Functions	
Sec. 3.3.1	WORD PIODIO_GetDriverVersion(WORD *wDriverVersion);
Sec. 3.3.2	WORD PIODIO_DriverInit(void);
Sec. 3.3.3	WORD PIODIO_SearchCard(WORD *wBoards, DWORD dwPIOCardID);
Sec. 3.3.4	WORD PIODIO_GetConfigAddressSpace(WORD wBoardNo, DWORD *wAddrBase, WORD *wIrqNo, WORD *wSubVendor, WORD *wSubDevice,WORD *wSubAux, WORD *wSlotBus, WORD *wSlotDevice);
Sec. 3.3.5	void PIODIO_DriverClose(void);
Sec. 3.3.6	WORD PIODIO_ActiveBoard(WORD wBoardNo);
Sec. 3.3.7	WORD PIODIO_WhichBoardActive(void);

Reference	Function Definition
Interrupt Functions	
Sec. 3.4.1	WORD PIODIO_IntResetCount(void);
Sec. 3.4.2	WORD PIODIO_IntGetCount(DWORD *dwIntCount);
Sec. 3.4.3	WORD PIODIO_IntInstall(WORD wBoardNo, HANDLE *hEvent, WORD wInterruptSource, WORD wActiveMode);
Sec. 3.4.4	WORD PIODIO_IntRemove(void);
PIO-D48 Interrupt Functions	
Sec. 3.5.1	WORD PIOD48_IntGetCount(DWORD *dwIntCount);
Sec. 3.5.2	WORD PIOD48_IntInstall(WORD wBoardNo, HANDLE *hEvent, WORD wIrqMask, WORD wActiveMode);
Sec. 3.5.3	WORD PIOD48_IntGetActiveFlag(WORD *bActiveHighFlag, WORD *bActiveLowFlag);
Sec. 3.5.4	WORD PIOD48_IntRemove(void);
PIO-D48 Counter Functions	
Sec. 3.6.1	void PIOD48_SetCounter(DWORD dwBase, WORD wCounterNo, WORD bCounterMode, DWORD wCounterValue);
Sec. 3.6.2	DWORD PIOD48_ReadCounter(DWORD dwBase, WORD wCounterNo, WORD bCounterMode);
Sec. 3.6.3	void PIOD48_SetCounterA(WORD wCounterNo, WORD bCounterMode, DWORD wCounterValue);
Sec. 3.6.4	DWORD PIOD48_ReadCounterA(WORD wCounterNo, WORD bCounterMode);
PIO-D64 Counter Functions	
Sec. 3.7.1	void PIOD64_SetCounter(DWORD dwBase, WORD wCounterNo, WORD bCounterMode, DWORD wCounterValue);
Sec. 3.7.2	DWORD PIOD64_ReadCounter(DWORD dwBase, WORD wCounterNo, WORD bCounterMode);
Sec. 3.7.3	void PIOD64_SetCounterA(WORD wCounterNo, WORD bCounterMode, DWORD wCounterValue);
Sec. 3.7.4	DWORD PIOD64_ReadCounterA(WORD wCounterNo, WORD bCounterMode);
PIO-D48 Frequency Functions	
Sec. 3.8.1	DWORD PIOD48_Freq(DWORD dwBase);
Sec. 3.8.2	DWORD PIOD48_FreqA();

3.1 Functions of Test

3.1.1 PIODIO_GetDllVersion

- **Description:**
  This functions is used to retrieve the version number of the PIODIO.DLL driver

- **Syntax:**
  WORD **PIODIO_GetDllVersion**(Void)

- **Parameters:**
  None

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

3.1.2 PIODIO_ShortSub

- **Description:**
  This function is used to perform the subtraction (as nA - nB in short data type), and is provided for testing DLL linkage purposes.

- **Syntax:**
  short **PIODIO_ShortSub**(short **nA**, short **nB**)

- **Parameters:**

fA	[Input]	2 bytes short data type value
fB	[Input]	2 bytes short data type value

- **Returns:**
  The value of nA – nB

3.1.3 PIODIO_FloatSub

- **Description:**
  This function is used to perform the subtraction (as fA - fB in float data type), and is provided for testing DLL linkage purposes.

- **Syntax:**
  float **PIODIO_FloatSub**(float **fA**, float **fB**)

- **Parameters:**

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

- **Returns:**
  The value of fA - fB

3.2 Digital I/O Functions

3.2.1 PIODIO_OutputByte

- **Description:**
  This function is used to send 8 bits of data to the specified I/O port.

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

- **Parameters:**

WPortAddr	[Input]	I/O port address. Refer to the PIODIO_GetConfigAddressSpace() function (Sec. 3.3.4).Only the low WORD is valid.
WOutputVal	[Input]	8 bits of data sent to the I/O port. Only the low BYTE is valid.

- **Returns:**
  None

3.2.2 PIODIO_InputByte

- **Description:**
  This function is used to read 8 bits of data from the specified I/O port.

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

- **Parameters :**

WPortAddr	[Input]	I/O port address. Refer to the PIODIO_GetConfigAddressSpace() function (Sec. 3.3.4).Only the low WORD is valid.

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

3.2.3 PIODIO_OutputWord

- **Description:**
  This function is used to send 16 bits of data to the specified I/O port.

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

- **Parameters:**

WPortAddr	[Input]	I/O port address. Refer to the PIODIO_GetConfigAddressSpace() function (Sec. 3.3.4).Only the low WORD is valid.
WOutputVal	[Input]	16 bits of data sent to the I/O port. Only the low WORD is valid.

- **Returns:**
  None

3.2.4 PIODIO_InputWord

- **Description:**
  This function is used to read 16 bits of data from the specified I/O port.

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

- **Parameters:**

WPortAddr	[Input]	I/O port address. Refer to the PIODIO_GetConfigAddressSpace() function (Sec. 3.3.4).Only the low WORD is valid.

- **Returns:**
  16 bits of data. Only the low WORD is valid.

3.3 Driver Relative Functions

3.3.1 PIODIO_GetDriverVersion

- **Description:**
  This function is used to read the version number information from the PIODIO driver.

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

- **Parameters:**

wDriverVersion	[Output]	wDriverVersion address

- **Returns:**
  Refer to "Section 1.2 ErrorCode and ErrorString Table".

3.3.2 PIODIO_DriverInit

- **Description:**
  This function is used to open the PIODIO driver and allocates the computer resource for the device. This function must be called once before applying other PIODIO functions.

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

- **Parameters:**
  None

- **Returns:**
  Refer to "Section 1.2 ErrorCode and ErrorString Table".

3.3.3 PIODIO_SearchCard

Description:

This function can be used to search for the installed card and determine total number of boards. This function must be called once before applying other PIODIO functions.

Syntax:

WORD **PIODIO_SearchCard**(WORD ***wBoards**,
DWORDn **dwPIOCardID**);

Parameters:

wBoards	[Output]	Determine the total number of boards.
DwPIOCardID	[Input]	Sub ID of the PIODIO card. Refer to Table3.1 for more details.

NOTES:

Different versions of the PIO-DIO series boards may have different Sub IDs. This function will determine the total number of PIO-DIO series boards including all versions; no matter what version Sub ID is input.

For example:

wRtn=PIODIO_SearchCard(&wBoards, 0x800100);

Will determine the total number of PIO-D144(U) boards installed in the PC regardless of version.

Returns:

Refer to "Section 1.2 ErrorCode and ErrorString Table".

3.3.4 PIODIO_GetConfigAddressSpace

- **Description:**
  This function is used to obtain the I/O address and other information for the PIODIO board.

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

- **Parameters:**

wBoardNo	[Input]	The PIODIO board number.
wAddrBase	[Output]	The base address of the PIODIO board. Only the low WORD is valid.
wIrqNo	[Output]	The IRQ number that is being used by the PIODIO 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.

- **Returns:**
  Refer to "Section 1.2 ErrorCode and ErrorString Table".

3.3.5 PIODIO_DriverClose

- **Description:**
  This function is used to close the PIODIO Driver and release the device resources from the computer. This function must be called once before exiting the user's application.

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

- **Parameters:**
  None

- **Returns:**
  None

3.3.6 PIODIO_ActiveBoard

- **Description:**
  This function is used to active one of the PIO-DIO boards installed in the system. This function must call once before the D/I/O and interrupt functions are called.

- **Syntax:**
  void **PIODIO_ActiveBoard**(WORD **wBoardNo**);

- **Parameters:**

wBoardNo	[Input]	The board numbers to active.

- **Returns:**
  Refer to "Section 1.2 ErrorCode and ErrorString Table".

3.3.7 PIODIO_WhichBoardActive

- **Description:**
  This function is used to return the board number of the active board.

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

- **Parameters:**
  None

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

3.4 Interrupt Functions

3.4.1 PIODIO_IntResetCount

- **Description:**
  This function is used to clear the counter value of the device driver for the interrupt.

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

- **Parameters:**
  None

- **Returns:**
  Refer to "Section 1.2 ErrorCode and ErrorString Table".

3.4.2 PIODIO_IntGetCount

- **Description:**
  This function is used to read the **dwIntCount** value defined in the device driver.

- **Syntax:**
  WORD **PIODIO_IntGetCount**(DWORD ***dwIntCount**);

- **Parameters:**

dwIntCount	[Output]	Address of dwIntCount, which is used of store the value of the interrupt counter.

- **Returns:**
  Refer to "Section 1.2 ErrorCode and ErrorString Table".

3.4.3 PIODIO_IntInstall

■ **Description:**
This function is used to install the IRQ service routine.

■ **Syntax:**
WORD **PIODIO_IntInstall**(WORD **wBoardNo**, HANDLE ***hEvent**,
WORD **wInterruptSource**, WORD **wActiveMode**);

■ **Parameters:**

wBoardNo	[Input]	The board to be used.
hEvent	[Input]	Address of an Event handle. The user's program must call the Windows API function "CreateEvent()" to create an event object.
wInterruptSource	[Input]	The Interrupt Source to be used. Refer to the following Table 3.1.
wActiveMode	[Input]	The mode for triggering the interrupt. 0 → PIODIO_ActiveLow 1 → PIODIO_ActiveHigh

■ **Returns:**
Refer to "Section 1.2 ErrorCode and ErrorString Table".

Table 3.1

Card No.	wInterruptSource	Description
PIO-D48 Series	**0**	**PC3/PC7 from Port-2**
	1	PC3/PC7 from Port-5
	2	Cout0
	3	Cout2
PIO-D56/D24 Series	**0**	**PC0**
	1	PC1
	2	PC2
	3	PC3
PIO-D64 Series	**0**	**EXTIRQ**
	1	EVTIRQ
	2	TMRIRQ
PIO-D96 Series	**0**	**P2C0**
	1	P5C0
	2	P8C0
	3	P11C0
PIO-D144/D168 Series	**0**	**P2C0**
	1	P2C1
	2	P2C2
	3	P2C3

3.4.4 PIODIO_IntRemove

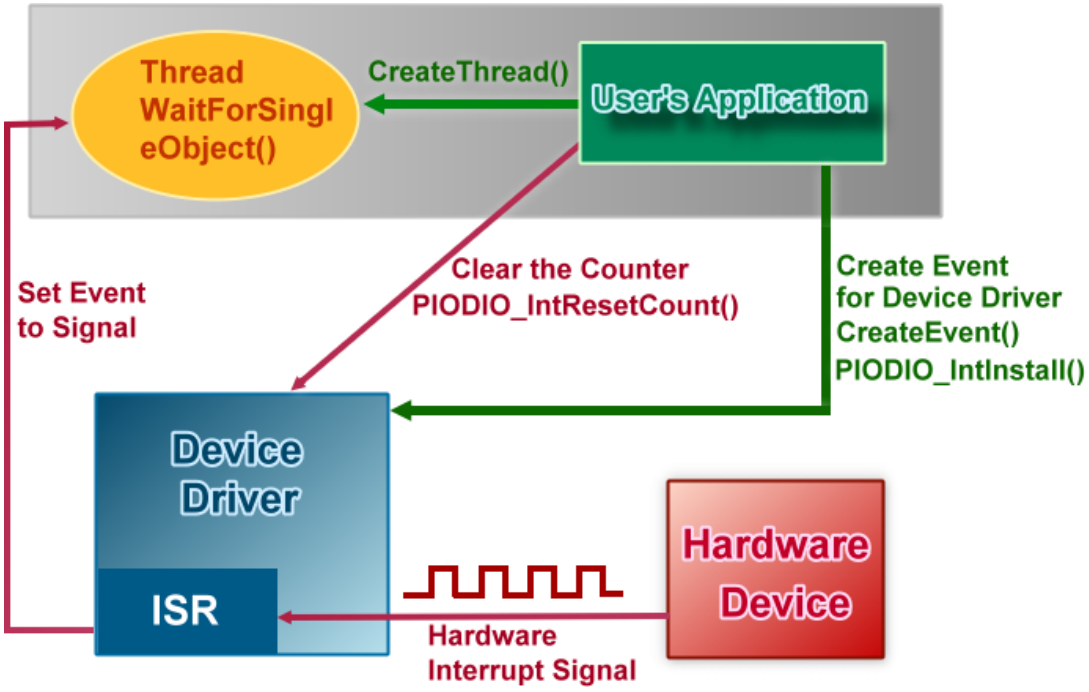
- **Description:**
  This function is used to remove the IRQ service routine.

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

- **Parameters:**
  None

- **Returns:**
  Refer to "Section 1.2 ErrorCode and ErrorString Table".

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
);
```

3.5 PIO-D48 Interrupt

The following PIOD48_XXX series function is designed for PIO-D48 series card only. They cannot be used with other cards.

The most different between the PIO-DIO and PIO-D48 interrupt functions is the PIO-DIO supports only one interrupt-source at a time and the PIO-D48 supports 4 interrupt-source at a time.

3.5.1 PIOD48_IntGetCount

■ **Description:**

This subroutine will read the **Interrupt-Counter** value in the device driver. The Interrupt-Counter will be increased (in the ISR) when the interrupt is triggered. When the interrupt setting to Active-High only or Active-Low only, some of the interrupt signal will be ignored and the Interrupt-Counter will not increase.

■ **Syntax:**

WORD **PIOD48_IntGetCount**(DWORD ***dwIntCount**);

■ **Parameters:**

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

■ **Returns:**

Refer to "Section 1.2 ErrorCode and ErrorString Table".

3.5.2 PIOD48_IntInstall

Description:

This subroutine will install the IRQ service routine. This function supports multiple interrupt-source and the Active-Mode can setting to "Active-Low only", "Active-High only" and "Active-Low or Active-High".

Syntax:

WORD **PIOD48_IntInstall**(WORD **wBoardNo**, HANDLE ***hEvent**,
WORD **wIrqMask**, WORD **wActiveMode**);

Parameters:

wBoardNo	[Input]	The board to be used.
hEvent	[Input]	Address of an Event handle. The user's program must call the Windows API function "CreateEvent()" to create an event object.
wIrqMask	[Input]	What the Interrupt-Source to be used ? Please refer to hardware's manual for the detail information. **wIrqMask** / **Description** 1 / INT_CHAN_0: PC3/PC7 from Port-2 2 / INT_CHAN_1: PC3/PC7 from Port-5 4 / INT_CHAN_2: Cout0 8 / INT_CHAN_3: Cout2 This function supports 4 interrupt-source at a time, thus users can use multiple interrupt-source like 1 +2 +4 +8.
wActiveMode	[Input]	When the ISR will service the interrupt ? **wActiveMode** / **Description** 1 / **PIOD48_ActiveLow** The interrupt is occurred when the Interrupt-Source status is low. 2 / **PIOD48_ActiveHigh** The interrupt is occurred when the Interrupt-Source status is high. This can be 1 (Active-High), 2(Active-Low) or 1 + 2 (Both of the High and Low will active the interrupt).

Returns:

Please refer to "Section 1.2 ErrorCode and ErrorString Table".

3.5.3 PIOD48_IntGetActiveFlag

■ **Description:**

This subroutine will read the Active-High and Active-Low flag from the device driver's memory queues (First-in-First-out, Buffer Size: 2000 flags for High/Low).

The Active-Flag is used to records the Active-State-change of interrupt-source when the interrupt occurred. The Active-High-Flag records which interrupt-source changed to high state and the Active-Low-Flag records which interrupt-source changed to low state. Users can uses these flags to indicate which interrupt-source has changed.

If the Active-Mode is set to Active-Low(/Active-High) only and the state for the Active-Low(/Active-High) is equal to zero, then the ISR will not increased the interrupt-counter, and the Active-Flag for High and Low will not recorded.

If users have not calling this function to retrieve the flags from device driver's memory queues, these queues will stop record the flags (lost data) while the buffer is full. But the interrupt-counter will still counting while the ISR services the interrupt.

■ **Syntax:**

WORD **PIOD48_IntGetActiveFlag**(WORD ***bActiveHighFlag**, WORD ***bActiveLowFlag**);

■ **Parameters:**

bActiveHighFlag	[Output]	Returns a flag that indicates which interrupt-source changed to High-State.
bActiveLowFlag	[Output]	Returns a flag that indicates which interrupt-source changed to Low-State.

■ **Returns:**

Refer to "Section 1.2 ErrorCode and ErrorString Table".

3.5.4 PIOD48_IntRemove

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

- **Syntax:**
  WORD **PIOD48_IntRemove**(void);

- **Parameters:**
  None

- **Returns:**
  Refer to "Section 1.2 ErrorCode and ErrorString Table".

3.6 PIO-D48 Counter

The following PIOD48_XXX series function is designed for PIO-D48 card only.

3.6.1 PIOD48_SetCounter

■ **Description:**

This subroutine is used to set the 8254 counter's mode and value.

■ **Syntax:**

void **PIOD48_SetCounter**(DWORD **dwBase**, WORD **wCounterNo**,
WORD **bCounterMode**, DWORD **wCounterValue**);

■ **Parameters:**

dwBase	[Input]	I/O port address. Refer to the PIODIO_GetConfigAddressSpace() function (Sec. 3.3.4). Only the low WORD is valid.
wCounterNo	[Input]	The 8254 Counter-Number: 0 to 2.
wCounterMode	[Input]	The 8254 Counter-Mode: 0 to 5. Please refer to the hardware manual for details.
wCounterValue	[Input]	The 16 bits value for the timer/counter to count. Only the low WORD is valid.

■ **Returns:**

None

3.6.2 PIOD48_ReadCounter

■ **Description:**
This subroutine is used to read the 8254 counter's value.

■ **Syntax:**
DWORD **PIOD48_ReadCounter**(DWORD **dwBase**, WORD **wCounterNo**,
WORD **bCounterMode**);

■ **Parameters:**

dwBase	[Input]	I/O port address. Refer to the PIODIO_GetConfigAddressSpace() function (Sec. 3.3.4). Only the low WORD is valid.
wCounterNo	[Input]	The 8254 Counter-Number: 0 to 2.
wCounterMode	[Input]	The 8254 Counter-Mode: 0 to 5. Please refer to the hardware manual for details.

■ **Returns:**
Returns the 16 bits counter's value. (Only the low WORD is valid.)

3.6.3 PIOD48_SetCounterA

■ **Description:**
This subroutine is used to set the 8254 counter's mode and value.
Users have to call the PIODIO_ActiveBoard() function before calling this function.

■ **Syntax:**
void **PIOD48_SetCounterA**(WORD **wCounterNo**, WORD **bCounterMode**,
DWORD **wCounterValue**);

■ **Parameters:**

wCounterNo	[Input]	The 8254 Counter-Number: 0 to 2.
wCounterMode	[Input]	The 8254 Counter-Mode: 0 to 5. Please refer to the hardware manual for details.
wCounterValue	[Input]	The 16 bits value for the counter to count. Only the low WORD is valid.

■ **Returns:**
None

3.6.4 PIOD48_ReadCounterA

Description:

This subroutine is used to read the 8254 counter's value.

Users have to call the PIODIO_ActiveBoard() function before calling this function.

Syntax:

DWORD **PIOD48_ReadCounterA**(WORD **wCounterNo**, WORD **bCounterMode**);

Parameters:

wCounterNo	[Input]	The 8254 Counter-Number: 0 to 2.
wCounterMode	[Input]	The 8254 Counter-Mode: 0 to 5. Please refer to the hardware manual for details.

Returns:

Returns the 16 bits counter's value. (Only the low WORD is valid.)

3.7 PIO-D64 Counter

The following PIOD64_XXX series function is designed for PIO-D64 card only.

3.7.1 PIOD64_SetCounter

■ **Description:**

This subroutine is used to set the 8254 counter's mode and value.

■ **Syntax:**

void **PIOD64_SetCounte**r(DWORD **dwBase**, WORD **wCounterNo**,
WORD **bCounterMode**, DWORD **wCounterValue**);

■ **Parameters:**

dwBase	[Input]	I/O port address. Refer to the PIODIO_GetConfigAddressSpace() function (Sec. 3.3.4). Only the low WORD is valid.
wCounterNo	[Input]	The 8254 Counter-Number: 0 to 5. (0 to 2: Chip-0, 3 to 5: Chip-1)
wCounterMode	[Input]	The 8254 Counter-Mode: 0 to 5. Please refer to the hardware manual for details.
wCounterValue	[Input]	The 16 bits value for the counter to count. Only the low WORD is valid.

■ **Returns:**

None

3.7.2 PIOD64_ReadCounter

■ **Description:**
This subroutine is used to read the 8254 counter's value.

■ **Syntax:**
DWORD **PIOD64_ReadCounter**(DWORD **dwBase**, WORD **wCounterNo**,
WORD **bCounterMode**);

■ **Parameters:**

dwBase	[Input]	I/O port address. Refer to the PIODIO_GetConfigAddressSpace() function (Sec. 3.3.4). Only the low WORD is valid.
wCounterNo	[Input]	The 8254 Counter-Number: 0 to 5. (0 to 2: Chip-0, 3 to 5: Chip-1)
wCounterMode	[Input]	The 8254 Counter-Mode: 0 to 5. Please refer to the hardware manual for details.

■ **Returns:**
Returns the 16 bits counter's value. (Only the low WORD is valid.)

3.7.3 PIOD64_SetCounterA

■ **Description:**
This subroutine is used to set the 8254 counter's mode and value.
Users have to call the PIODIO_ActiveBoard() function before calling this function.

■ **Syntax:**
void **PIOD64_SetCounterA**(WORD **wCounterNo**, WORD **bCounterMode**,
DWORD **wCounterValue**);

■ **Parameters:**

dwBase	[Input]	I/O port address. Refer to the PIODIO_GetConfigAddressSpace() function (Sec. 3.3.4). Only the low WORD is valid.
wCounterNo	[Input]	The 8254 Counter-Number: 0 to 5. (0 to 2: Chip-0, 3 to 5: Chip-1)
wCounterMode	[Input]	The 8254 Counter-Mode: 0 to 5. Please refer to the hardware manual for details.

■ **Returns:**
None

3.7.4 PIOD64_ReadCounterA

Description:

This subroutine is used to read the 8254 counter's value.

Users have to call the PIODIO_ActiveBoard() function before calling this function.

Syntax:

DWORD **PIOD64_ReadCounterA**(WORD **wCounterNo**, WORD **bCounterMode**);

Parameters:

wCounterNo	[Input]	The 8254 Counter-Number: 0 to 5. (0 to 2: Chip-0, 3 to 5: Chip-1)
wCounterMode	[Input]	The 8254 Counter-Mode: 0 to 5. Please refer to the hardware manual for details.

Returns:

Returns the 16 bits counter's value. (Only the low WORD is valid.)

3.8 PIO-D48 Frequency

The following PIOD48_XXX series function is designed for PIO-D48 card only.

3.8.1 PIOD48_Freq

■ **Descriptions:**
This subroutine is used to measurement the signal frequency. Users have to connect the signal(+) with CN1.Pin29, and connect the signal(-) with CN1.Pin19. It will uses the Counter-0 and Counter-1 to measure the frequency, thus users shouldn't use Counter-0 and Counter-1 for other purposes.

■ **Syntax:**
DWORD **PIOD48_Freq**(DWORD **dwBase**);

■ **Parameters:**

dwBase	[Input]	I/O port address. Refer to the PIODIO_GetConfigAddressSpace() function (Sec. 3.3.4). Only the low WORD is valid.

■ **Returns:**
Returns the frequency value. (Only the low WORD is valid.)

3.8.2 PIOD48_FreqA

■ **Description:**
Please refer to the description of "PIOD48_Freq()" function.
Users have to calling the "PIODIO_ActiveBoard()" function before calling this function.

■ **Syntax:**
DWORD **PIOD48_FreqA**();

■ **Parameters:**
None

■ **Returns:**
Returns the frequency value. (Only the low WORD is valid.)

4. DOS LIB Function

4.1 ErrorCode and ErrorString Table

Table 4-1 ErrorCode and ErrorString

Error Code	Error ID	Error String
0	NoError	OK ! No Error!
1	DriverHandleError	Device driver opened error
2	DriverCallError	Got the error while calling the driver functions
3	FindBoardError	Can't find the board on the system
4	TimeOut	Timeout
5	ExceedBoardNumber	Invalidate board number (Valid range: 0 to TotalBoards -1)
6	NotFoundBoard	Can't detect the board on the system

4.2 PIO_DriverInit

■ **Description:**

This function can detect all PIO/PISO series cards in the system. It is implemented based on the PCI Plug & Play mechanism-1. It will find all PIO/PISO series cards installed in this system and save all their resources in the library.

■ **Syntax:**

WORD **PIO_DriverInit**(WORD *wBoards, WORD wSubVendorID, WORD wSubDeviceID,WORD wSubAuxID)

■ **Parameters:**

wBoards	[Output]	Number of boards found in this PC
wSubVendorID	[Input]	SubVendor ID of the board
wSubDeviceID	[Input]	SubDevice ID of the board
wSubAuxID	[Input]	SubAux ID of the board

■ **Returns:**

Please refer to Sec. 4.1.

4.3 PIO_GetConfigAddressSpace

■ **Description:**

The user can use this function to save the resources found on all the PIO/PISO cards installed on the system. Then the application program can control all the functions of PIO/PISO series cards directly.

■ **Syntax:**

WORD **PIO_GetConfigAddressSpace**(wBoardNo,*wBase,*wIrq, wSubVendor, *wSubDevice,*wSubAux,*wSlotBus,*wSlotDevice)

■ **Parameters:**

wBoardNo	[Input]	Board number
wBase	[Output]	The base address of the board
wIrq	[Output]	The IRQ number that the 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

■ **Returns:**

Please refer to Sec. 4.1.

4.4 PIO_GetDriver Version

■ **Description:**

This subroutine will obtain the version number of PIODIO driver.

■ **Syntax:**

WORD **PIO_GetDriverVersion**(WORD *wDriverVersion)

■ **Parameters:**

wDriverVersion	[Output]	Address of wDriverVersion

■ **Returns:**

Please refer to Sec. 4.1.

4.5 ShowPIOPISO

■ **Description:**

This function will show a text string for a special Sub_ID. This text string is the same as that defined in PIO.H.

■ **Syntax:**

WORD **ShowPIOPISO**(wSubVendor, wSubDevice, wSubAux)

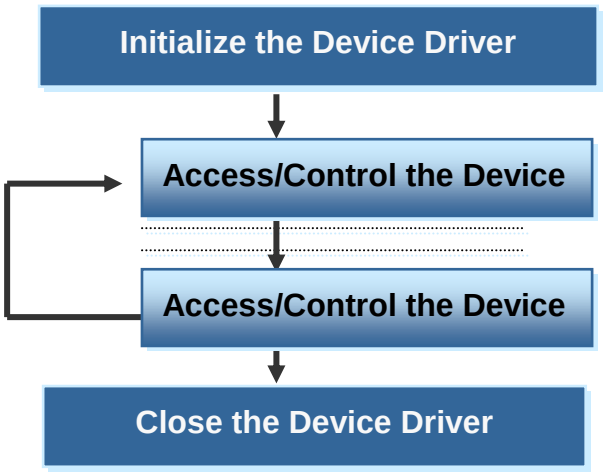
■ **Parameters:**

wSubVendor	[Output]	SubVendor ID of the board
wSubDevice	[Input]	SubDevice ID of the board
wSubAux	[Input]	SubAux ID of the board

■ **Returns:**

Please refer to Sec. 4.1.

5. Program Architecture




```
PIODIO_DriverInit()
Enable All DI/DO Ports
….
     PIODIO_InputByte( … )
     ……..
     ……..
     PIODIO_OutputByte(…)
…..

PIODIO_DriverClose()
```

5.1 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 Service.icpdas@ gmail.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 98, Windows 2000 or 32-bit Windows XP/2003/Vista/2008/7.
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 test 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 you comments? And please keeps contact with us.

E-mail: Service@icpads.com
Service.icpdas@ gmail.com

WebSite: http://www.icpdas.com
http://www.icpdas.com.tw