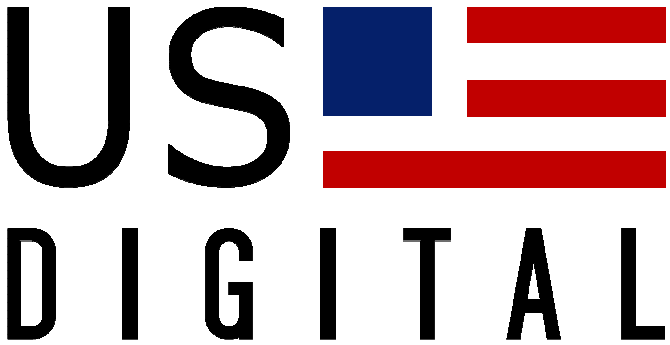


PCI-4E Manual

PCI Interface Card for Four Incremental Encoders

Revision: 2.02
9/10/2003

Table of Contents

Amendments

Date	Comment(s)	Author
1/30/03	Rev. 1.0	Steve Smith
2/24/03	Rev. 1.1 Added PCI4E_GetROM_ID function.	Steve Smith
5/28/03	- Added Chapter 5 Architecture of PCI-4E - Added “Minimum Programming for a Channel” - Grouped function calls. - Added overview for each group of function calls. - Added figures.	Sup Premvuti
5/29/03	Added “Principle of Operation for a Channel”	Kurt Liebezeit
6/9/03	Rev. 2.0	
7/23/03	Added “Using Java”	Steve Smith
9/10/03	Added Linux installation instructions	Steve Smith

1 Introduction

1.1 Purpose

The purpose of this interface specification is to provide aid in understanding how to use the features of the PCI-4E, PCI Interface Card for Four Incremental Encoders. The features of the PCI-4E card are made accessible by using the functions provided in the USD_PCI_4E.dll.

1.2 Scope

This document shall describe how to use each of the available interface methods provided by the PCI-4E card. The following chapters are included.

- Installation Instructions
- Trouble Shooting
- Running Demonstration Program
- Architecture of PCI-4E
- PCI-4E Registers
- Trigger/Capture Feature
- Typical Usage Scenario
- Function Calls
- Error Codes

2 Installation Instructions

2.1 Windows Operating Sytem

Please follow these five steps to install PCI-4E and its software.

1. Run the Setup.exe

   Note: The installation checks if an old version of windrvr.sys already exists in the ..system32\drivers directory. If an older version is found, a dialog will be displayed which presents the following three options:

   Option 1 - Install windrvr.sys version 5.2.2 and make a backup of the older version and place it in the C:\Program Files\PCI4E\Backup directory.

   Option 2 - Leave the older version installed and copy version 5.2.2 to C:\Program Files\PCI4E directory.

   Option 3 - Cancel the installation.

   The PCI-4E demos will not function properly with an older version of the windrvr.sys file.

   If you are running an application that requires a previous version of the windrvr.sys file, please contact US Digital for support.

   If you need to restore (or run with) the older version simply copy the windrvr.sys from the C:\Program Files\PCI4E\Backup directory to the ..\system32\drivers directory and then reboot.

2. Shutdown the computer and install the PCI-4E card(s).

   2.1 For additional safety, disconnect power from the computer at the main supply.

   2.2 Observe static handling procedures while working with the PCI-4E: wear an approved ground strap, and open the PCI-4E packaging only at a work surface with a grounded anti-static mat.

   2.3 Carefully insert the PCI-4E card into an available PCI slot. You can install one or more PCI-4E cards at the same time.

3. Power-On the computer. The green LED on the PCI-4E card should be flashing. If the LED is not flashing see Chapter 3, Trouble Shooting.

4. The PCI4E_rev00.inf is copied to the WINNT\inf directory and installed by the setup program so the Found New Hardware Wizard should not appear. If it does appear, follow its instructions; when asked, specify the location of the PCI4E_rev00.inf file. A copy of this file has also been placed in the C:\Program Files\PCI-4E Demo directory.

5. Launch the PCI-4E VB or MFC Demo application to test all installed PCI-4E cards. (See Chapter 4 Running Demonstration Program) After PCI-4E VB or MFC Demo is running properly with PCI-4E card, LabVIEW users may proceed to install LabVIEW VIs for PCI-4E.

If the demo application fails to load or the device manager indicates that there is a problem with the Jungo\US DIGITAL PCI-4E Firmware Revision 00 device, then ensure the windrvr.sys file is located in the correct directory, especially if installing on a multi-boot system or if the OS is installed in a non-default directory. The windrvr.sys should be located in one of the directories specified for the OS below:

Windows 98	- C:\Windows\system32\drivers
Windows 2000	- C:\WINNT\system32\drivers
Windows ME	- C:\Windows\system32\drivers
Windows XP	- C:\Windows\system32\drivers

Please contact U.S. Digital Customer Support if you have additional questions.

2.2 Linux Operating System

Please follow these steps to install PCI-4E and its software.

1. Shutdown the computer and install the PCI-4E card(s).

    2.1 For additional safety, disconnect power from the computer at the main supply.

    2.2 Observe static handling procedures while working with the PCI-4E: wear an approved ground strap, and open the PCI-4E packaging only at a work surface with a grounded anti-static mat.

    2.3 Carefully insert the PCI-4E card into an available PCI slot. You can install one or more PCI-4E cards at the same time.

2. Power-On the computer. The green LED on the PCI-4E card should be flashing. If the LED is not flashing see Chapter 3, Trouble Shooting.

3. Create local directory to copy the pci4e-1.0.tar file to, i.e., /home/steve/pci4e and change to the directory.

```
$ mkdir /home/steve/pci4e
$ cd /home/steve/pci4e
```

4. After the pci4e-1.0.tar file has been copied to the local directory. Extract the contents of pci4e-1.0.tar file.

```
$ tar xzvf pci4e-1.0.tar
```

5. Change to root or an equivalent access level.

```
$ su
     password
```

6. The kernel module driver, phob, written by Alessandro Rubini, is compiled using kernel source code. For more information on the phob driver, refer to the README file. In order to access the kernel source code, the Makefile relies on a symbolic link to the kernel source directory. If one does not exist you may create one using the following command:

Note: the version of linux installed on your system may be different than the one specified in the following command. If you are using version 2.4.20-8 of Linux, then you may skip to step 8.

```
# ln -s /usr/src/linux-2.4.20-8 /usr/src/linux
```

7. Remove previously built object files and recompile phob source files.

```
# make clean
# make
```

8. Load the phob kernel driver.

```
# ./phob_load vendor_id=0x1892 device_id=0x5747
```

9. Compile and run the pci4edemo.

```
# g++ pci4edemo.c pci4eHelper.c -o pci4edemo -Wall
# ./pci4edemo  (press any key to terminate the demo)
```

3 Trouble Shooting

Symptom:

The green led on the PCI-4E card does not come on after the computer is Powered-On.

Problem:

The board is not receiving any power.

The firmware initialization has failed.

Resolution:

Power-Off the computer and insure that the PCI-4E card is installed correctly.

Contact US Digital customer support, if all attempts fail.

Symptom:

The green led on the PCI-4E card is on but does not flash after the computer is Powered-On.

Problem:

The firmware initialization is succeeded but the PCI and PCI-4E board is not communicating.

Resolution:

Power-Off the computer and insure that the PCI-4E card is installed correctly.

Contact US Digital customer support, if all attempts fail.

4 Running Demonstration Programs

After PCI-4E hardware and driver software has been successfully installed, you should be able to run accompanied demonstration programs. The demo programs will give you an opportunity to explore features of PCI-4E.

Steps to run the demo programs (written in C, Visual Basic or LabVIEW):

(1) Connect at least one encoder to any connector of PCI-4E. (PCI-4E-D has four 10-pin connectors. PCI-4E-S has four 5-pin connectors.)
(2) Launch a demo program.
(3) The demo program will display the number of existing PCI-4E board(s) on the PCI bus and automatically assign unique device numbers for each PCI-4E board. Use an option of the demo program to choose a PCI-4E board you want to access.
(4) On the demo screen, locate a command to set “Cycles Per Revolution”. This number should be set to match the connected encoder. When using the VB Demo, select View\Configuration menu item to display Cycles Per Revolution input text box.
(5) Turn the encoder and see if the number of counts and the graph display match the movement of the encoder’s shaft.
(6) Explore available features of each demo such as changing quadrature mode.
(7) The demo programs also allow you to directly access all 32 registers of PCI-4E. The detailed explanation of architecture of PCI-4E and its registers can be found in the following chapters.

5 Architecture of PCI-4E

5.1 Overview

Memory-mapped registers based system:

PCI-4E interfaces to an application program through 32 memory-mapped 32-bit registers. An application program reads or writes these registers to set mode, select functions or get data from PCI-4E.

32 registers are divided into 4 groups (See Figure 5.1)

Channel 0 registers group:	register 0 to register 6
Channel 1 registers group:	register 8 to register 14
Channel 2 registers group:	register 16 to register 22
Channel 3 registers group:	register 24 to register 30
Miscellaneous registers group:	register 7, 15, 23, 31

All four channels are identical in number of registers and operation for each register. Each channel works independently based on its own set of registers.

Miscellaneous registers group consists of Command Register (reg.#7), Time Stamp Latch Register (reg.#15), 33 MHz Time Stamp Counter (reg.#23), and General Purpose Low Frequency Counter (reg.#31).

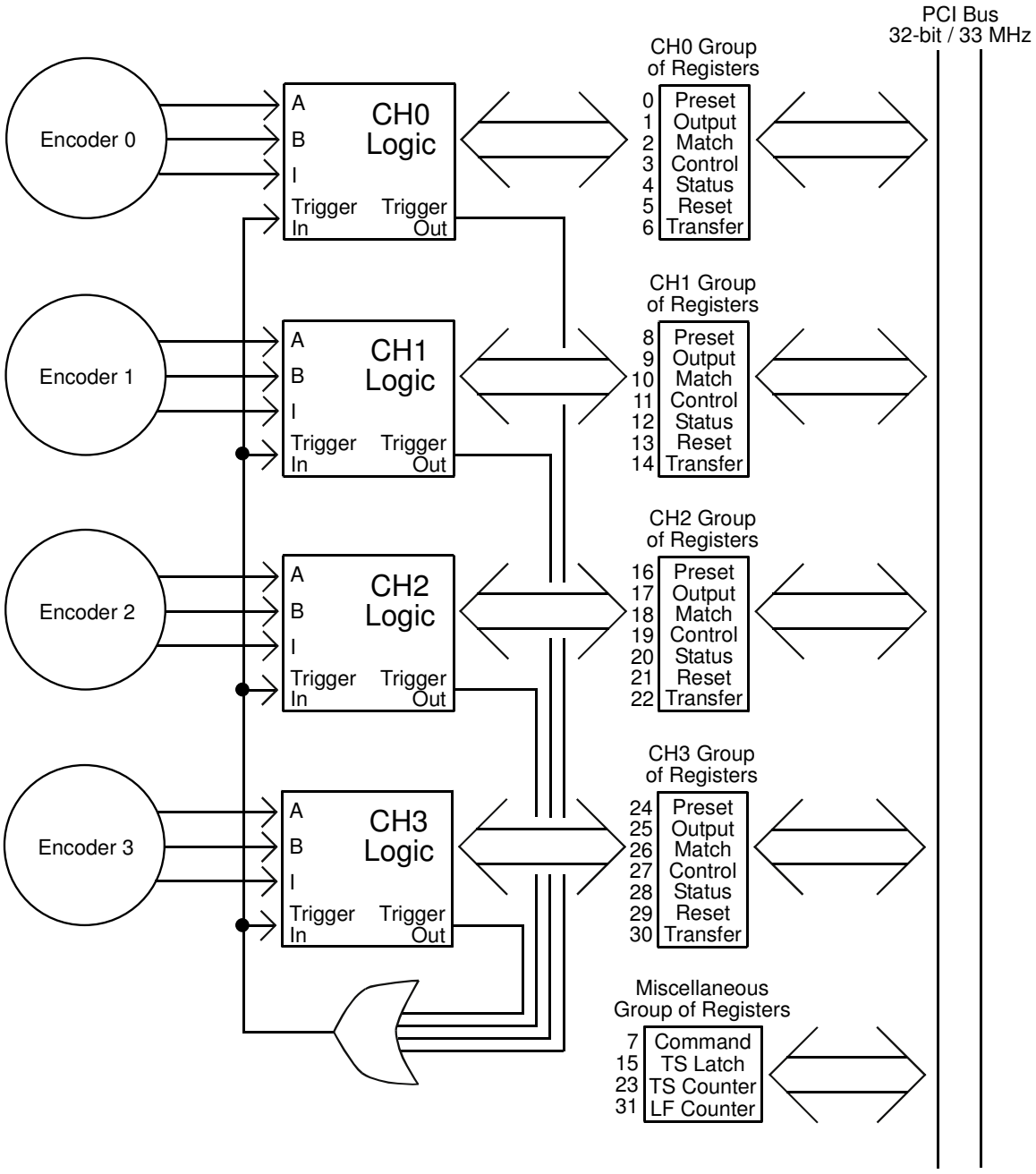


Figure 5.1 Block Diagram of PCI-4E

5.2 Principle of Operation for a Channel

The heart of each channel is a 24 bit up-down counter. (See Figure 5.2) It receives signals commanding it to count up or down from a state machine that watches the quadrature signals coming in; when the state machine sees the quadrature advance, it issues a pulse to increment the counter, and when it sees the quadrature retard (move backward) it issues a pulse to decrement the counter. The various input modes such as X1, X2, X4, and pulse/direction mode are implemented via the input state machine.

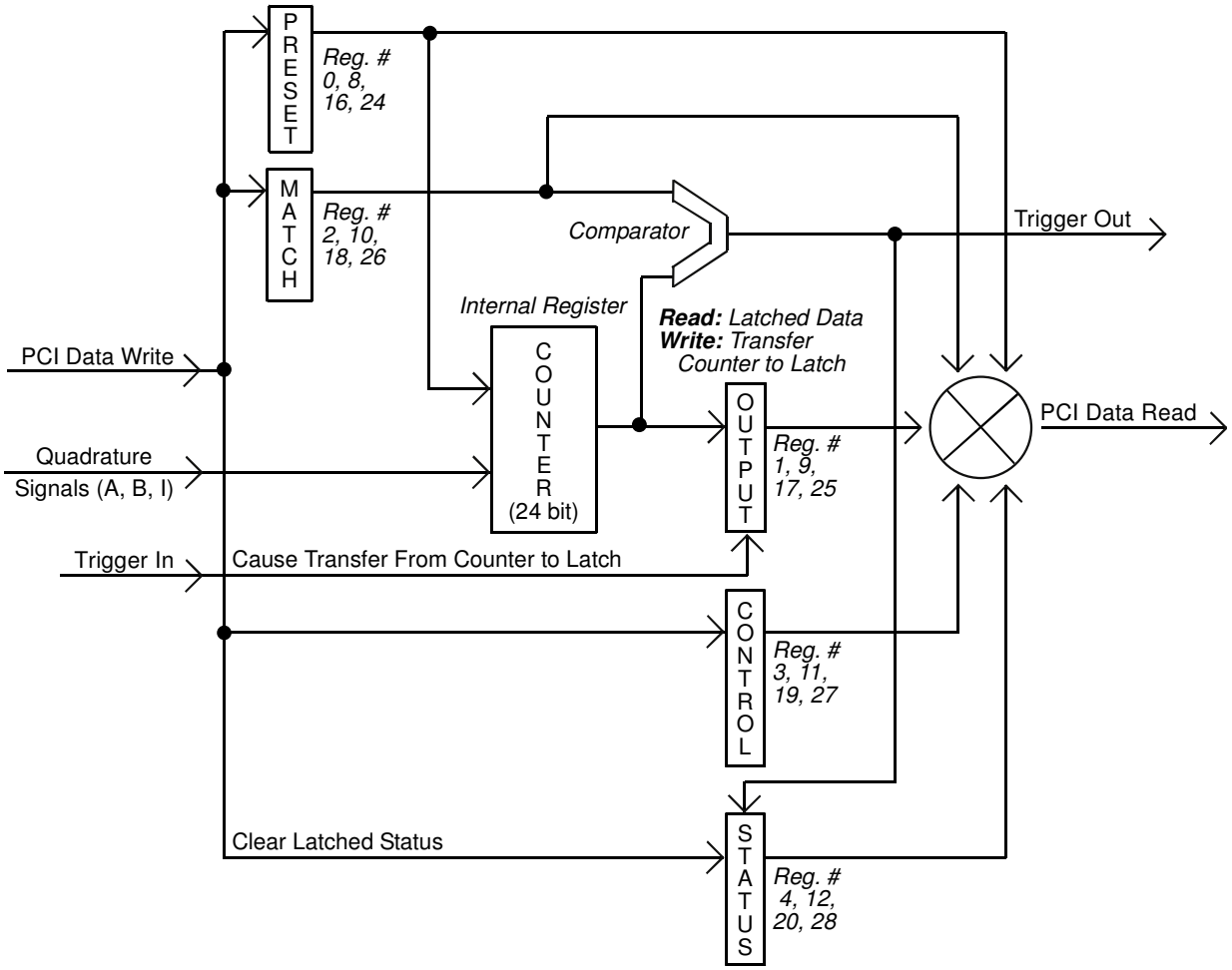
The output of the counter is made available to the host PCI bus through a latching register called the output latch. This was done so as to provide a way to capture and hold a snapshot of encoder position in hardware without requiring immediate software attention. There are two ways to transfer the counter value to the output latch: the host software can write (any value) to the output latch address, or the host can set up a triggering event that will use dedicated hardware to recognize a condition that will capture the counter value. Triggering is explained in detail in a later paragraph.

The counter is capable of counting in special modes that roll over from value N to value 0, or stop counting when a limit is reached. It does this with the help of a preset register, which defines the upper limit of the counting process. The value of the counter is continuously compared to the preset register, and in these special counting modes the counter is either disabled or reset or preset.

A separate match register is provided, to allow for comparisons against an arbitrary value even while the preset register is being used to implement a limited-range counting mode. The result of a match can be used to generate a trigger that will cause transfer of the counter value to the output latch on this channel and/or other channels simultaneously.

A control register is provided, to allow the various counting modes and input modes to be specified. A status register is also available, to report on various conditions existing within the channel; some conditions are latched, and persist until cleared explicity with writing 0xFFFFFFFF to the status register.

The triggering capability allows the host to specify conditions that will cause a capture of counter values on multiple channels. The conditions include advance of quadrature, retard of quadrature, passing through zero, encountering an index, reaching a value that corresponds to the match register, carry condition, or borrow condition. The specified condition may be sensed on any channel, and sent out of the channel to a higher level logic block, where it is OR'ed with the triggers from other channels. (See Figure 5.1) The resulting "super-trigger" then re-enters all of the channels; a channel may be enabled to respond to this event by transferring the counter contents to the output latch.



Abbreviation of register names

Symbol	Name
COUNTER	Internal counter register
CONTROL	Control register
STATUS	Status register
MATCH	Match register
PRESET	Preset register
OUTPUT	Output latch register

Figure 5.2 Block Diagram of Channel

5.3 Minimum Programming for a Channel

Once the installation has been done successfully, all PCI-4E boards in a PC are ready to be accessed through provided function calls. After understanding features of 6 registers in a Channel Group, advanced users can operate PCI-4E with just five basic function calls.

PCI Initializing

```
9.1.29      PCI4E_Initialize
9.1.2       PCI4E_CardCount
9.1.56      PCI4E_Shutdown
```

Read/Write Registers

```
9.1.32      PCI4E_ReadRegister
9.1.57      PCI4E_WriteRegister
```

In addition to 5 basic function calls, "User friendly functions" are also provided in order to facilitate writing application program with better readability. Names of user friendly functions refer directly to their functions or features. (See Chapter 9 "Function Calls" for details.) Each function call will be translated into reading, writing or combinations of reading and writing one or more of 32 registers of PCI-4E.

User friendly functions are used in the following example of a minimum program. Register numbers accessed by function calls are also provided as references.

A minimum program in C consists of four sections.
(Register numbers shown in this section are based on Channel 0. For accessing other channels, please refer to Chapter 6 PCI-4E Registers.)

1. Initialize PCI-4E cards
2. Configure counter
    - 2.1 Select value of Preset register (reg.#0)
    - 2.2 Select value of Control register (reg.#3)
        - quadrature mode
        - count mode
        - direction of count (up/down)
        - master enable
3. Get count from Output Latch register (reg.#1)
4. Close PCI-4E

Description:

(1) Initialize PCI-4E cards and get number of total PCI-4E cards on PCI bus.

Use this function:

```
PCI4E_Initialize(short *piDeviceCount );
```

(2) Select value of Preset Register (reg.#0)

If you plan to select the following counter modes; Range-limit mode, Non-recycle mode, or Modulo-N mode (See 6.1 Control Registers); the preset register must be set to your desired value.

Use this function:

```
PCI4E_SetPresetValue(short iDeviceNo, short iEncoder, long lVal);
```

(3) Select quadrature mode in Control Register (reg.#3)

Bit 15 and 14 determine how the accumulator increments: These bits may be referred to as either quadrature mode or multiplier.

Mode	bit15, bit14	Description
0	00	A quadrature input = Clock B quadrature input = Direction Each rising edge of A input causes a counter increment or decrement, depending on the level of B input.
1	01	Accumulator increments once for every for four quadrature states (X1).
2	10	Accumulator increments once for every for two quadrature states (X2)
3	11	Accumulator increments once for every for quadrature state (X4)

Use this function:

```
PCI4E_SetMultiplier(short iDeviceNo, short iEncoder, short iMode);
```

(4) Select count mode in Control Register (reg.#3)

Bit 17 and 16 determine mode of internal counter.

Mode	bit17, bit16	Description
0	00	Simple 24-bit counter mode
1	01	Range-limit mode
2	10	Non-recycle mode
3	11	Modulo-N mode

Use this function:

```
PCI4E_SetCounterMode(short iDeviceNo, short iEncoder, short iMode);
```

(5) Set direction bit (swap quadrature A/B bit) in Control Register (reg.#3)

Bit 19 of Control Register controls the direction of count (up/down)
"0" Quadrature signals A and B are treated normally in a channel's internal logic.
"1" Quadrature signals A and B are swapped in a channel's internal logic.
As the result, the direction of count (up/down) will be reversed when bit 19 changes value.

Use this function:

```
PCI4E_SetForward(short iDeviceNo, short iEncoder, BOOL bVal);
```

Note that PCI4E_SetForward function sets bit 19 of Control register when its parameter, bVal, is '1'.

(6) Set master enable bit in Control Register (reg.#3)

Set bit 18 to '1' to enable accumulator.

Use this function:

```
PCI4E_SetEnableAccumulator(short iDeviceNo, short iEncoder, BOOL bVal);
```

(7) Get count data from Output Latch Register (reg.#1)

The Output Latch Register is used to latch the count value from the internal counter register for reading by an application program. It is important to understand that the Output Latch Register will be updated ONLY after a WRITE action to the Output Latch Register (data is irrelevant). This means an application can read the Output Latch Register at any time. But its value will be updated to current count value only after it has been written.

To accommodate users who want to write a simple program that retrieves encoder counts, PCI4E_GetCount function is provided. When using this function, please be aware that write to and read from Output Latch Register are performed consecutively in one call of PCI4E_GetCount.

Use this function:

```
PCI4E_GetCount(short iDeviceNo, short iEncoder, long *plVal);
```

(8) Close PCI-4E before exiting application

The PCI4E_Shutdown function must be call in order to disconnect from the PCI4E driver.

Use this function:

```
PCI4E_Shutdown();
```

Example of a minimum program in C

```
// ConsoleApp.cpp : Defines the entry point for the console application.
//

#include "stdio.h"
#include "windows.h"
//#include "wincon.h"
#include "conio.h"
#include "..\PCI_4e.h"

// input of command from user
static char line[256];

/* maximum number of input queue events to peek at */
#define INPUT_RECS 256

/* array of records to store peeked events from the input queue */
INPUT_RECORD aInputBuffer[INPUT_RECS];

BOOL DoesErrorExist( int iErr )
{
	BOOL bRetVal = FALSE;
	switch( iErr )
	{
		case NO_CARDS_DETECTED:
			printf("NO_CARDS_DETECTED\n");
			bRetVal = TRUE;
			break;
		case INVALID_DEVICE_NUMBER:
			printf("INVALID_DEVICE_NUMBER\n");
			bRetVal = TRUE;
			break;
		case DEVICE_NOT_OPEN:
			printf("DEVICE_NOT_OPEN\n");
			bRetVal = TRUE;
			break;
		case INVALID_ENCODER_NUMBER:
			printf("INVALID_ENCODER_NUMBER\n");
			bRetVal = TRUE;
			break;
		case INVALID_REGISTER_NUMBER:
			printf("INVALID_REGISTER_NUMBER\n");
			bRetVal = TRUE;
			break;
		case INVALID_SLOT_NUMBER:
			printf("INVALID_SLOT_NUMBER\n");
			bRetVal = TRUE;
			break;
		case INVALID_QUADRATURE_MODE:
			printf("INVALID_QUADRATURE_MODE\n");
			bRetVal = TRUE;
			break;
		case FEATURE_NOT_SUPPORTED:
			printf("FEATURE_NOT_SUPPORTED\n");
			bRetVal = TRUE;
			break;
		case INVALID_COUNTER_MODE:
			printf("INVALID_COUNTER_MODE\n");
			bRetVal = TRUE;
			break;
		case FAILED_TO_OPEN_DRIVER:
			printf("FAILED_TO_OPEN_DRIVER\n");
			bRetVal = TRUE;
			break;
		case CONTROL_MODE_IS_ZERO:
			printf("CONTROL_MODE_IS_ZERO\n");
			bRetVal = TRUE;
			break;
		case INCORRECT_WINDRIVER_VERSION:
			printf("INCORRECT_WINDRIVER_VERSION\n");
			bRetVal = TRUE;
			break;
	}
```

```
        return bRetVal;
}

int main(int argc, char* argv[])
{
        int i, cmd = 0;
        short iBoardCnt = 0;
        short iDevice = 0;
        short iEncoder = 0;
        short iCPR = 0;
        short iVal = 0;
        long lVal = 0;
        long lPrevVal = 0;
        long lCtrlMode = 0;

        printf ("\n");
        printf("PCI-4E Console Application Demo!\n");
        printf("-------------------------------\n");

        // Initialize the PCI4E driver and get the number of cards detected...
        if( !DoesErrorExist(PCI4E_Initialize(&iBoardCnt)) )
        {
                // Check if any boards were detected...
                if(iBoardCnt < 1)
                        printf("No boards detected!\n");
                else
                {
                        // Get the appropriate board from user...
                        if( iBoardCnt == 1 )
                                printf("Enter the board number (only board automatically selected)\n");
                        else
                        {
                                do
                                {
                                        printf("Enter the board number (0-%d):     --> ", iBoardCnt-1);
                                        fgets(line, sizeof(line), stdin);
                                        sscanf(line, "%d", &iDevice);
                                } while (line[0] < 48 || line[0] > (48 + iBoardCnt-1));
                        }

                        // Get the encoder channel from user...
                        do
                        {
                                printf("Enter the encoder channel (0-3):  --> ");
                                fgets(line, sizeof(line), stdin);
                                sscanf(line, "%d", &iEncoder);
                        } while (line[0] < 48 || line[0] > (51));

                        // Get the encoder Counts Per Revolutions from user...
                        do
                        {
                                printf("Enter Counts Per Revolution:      --> ");
                                fgets(line, sizeof(line), stdin);
                                sscanf(line, "%d", &iCPR);
                        } while (line[0] == 10);

                        // Set the preset register value...
                        DoesErrorExist(PCI4E_SetPresetValue(iDevice, iEncoder, iCPR - 1));

                        // Get the quadrature mode (Multiplier) from user...
                        // Example 1: If the encoder CPR is 500, quadrature mode is X1, and counter
                        // mode is modulo-N, the preset register will be set to 499.
                        // Example 2: If the encoder CPR is 500, quadrature mode is X2, and counter
                        // mode is modulo-N, the preset register will be set to 999.
                        // Example 3: If the encoder CPR is 500, quadrature mode is X4, and counter
                        // mode is modulo-N, the preset register will be set to 1999.
                        printf("Enter the quadrature mode:\n");
                        printf("    0 - clock and direction\n");
                        printf("    1 - X1 multiplier\n");
                        printf("    2 - X2 multiplier\n");
                        printf("    3 - X4 multiplier\n");
                        do{
                                printf("                                    --> ");
                                fgets(line, sizeof(line), stdin);
                                sscanf(line, "%d", &iVal);
```

```
                }while (iVal < 0 || iVal > 3);

                // Set the quadrature mode ...
                PCI4E_SetMultiplier(iDevice, iEncoder, iVal);

                // Get the counter mode from user...
                printf("Enter the counter mode:\n");
                printf("        0 - simple 24-bit counter\n");
                printf("        1 - range-limit mode\n");
                printf("        2 - non-recycle mode\n");
                printf("        3 - modulo-N mode\n");
                do{
                        printf("                                --> ");
                        fgets(line, sizeof(line), stdin);
                        sscanf(line, "%d", &iVal);
                }while (iVal < 0 || iVal > 3);

                // Set the counter mode...
                DoesErrorExist(PCI4E_SetCounterMode(iDevice, iEncoder, iVal));

                //  Set the direction bit forward...
                DoesErrorExist(PCI4E_SetForward(iDevice, iEncoder, TRUE));

                // Set enable accumulator...
                DoesErrorExist(PCI4E_SetEnableAccumulator(iDevice, iEncoder, TRUE));


                printf("Encoder %d registers values:\n",iEncoder);
                printf("    Register    Hex         Decimal\n");
                printf("    ----------  ----------  ----------\n");
                for( i = 0; i<8; i++ )
                {
                        PCI4E_ReadRegister(iDevice, (short)(iEncoder * 8 + i), &lVal);
                        printf("%14d\t%10x%12d\n", i, lVal, lVal);
                }

                printf("\n\n*** Press any key to exit! ***\n\n");
                do
                {
                        // Get the encoder count value.
                DoesErrorExist(PCI4E_GetCount(iDevice, iEncoder, &lVal));

                        // Update display when value changes
                        if (lPrevVal != lVal)
                                printf("%d               \r", lVal);
                        lPrevVal = lVal;

                } while( !_kbhit() );  // Keep reading until a key is pressed...
        }
        // Close all open connections to the devices.
        PCI4E_Shutdown();
    }
    else
    {
        printf("*** Press any key to Exit! ***\n");
        do
        {
        } while( !_kbhit() ); // Keep reading until a key is pressed...
    }

    return 0;
}
```

6 PCI-4E Registers

The PCI-4E board has 32 32-bit registers that are used to provide access to four encoder channels. Each encoder channel uses 7 registers to communicate and control how data is acquired and manipulated. Note that these 7 registers utilize only 24 bits out of 32 bits. Bit 24 to bit 31 are not used and always return ‘0’ when being read. Miscellaneous Group of registers utilize all 32 bits.

Register name	Description	Register number			
		Channel 0 Group	Channel 1 Group	Channel 2 Group	Channel 3 Group
Preset	Sets roll-over value for Modulo-N mode, and upper limit for non-recycle and range limit modes	0	8	16	24
Output Latch	Counter contents are captured here by command from control register or by trigger capture. Writes cause capture of counter, reads return contents of output latch.	1	9	17	25
Match	Used as a reference to capture trigger when counter equals match register.	2	10	18	26
Control (See 6.1)	Contains bits that control the counting mode, quadrature mode, and other aspects of channel's operation.	3	11	19	27
Status (See 6.2)	Contains bits that describe the state of the counter, trigger system when read; writing 0xFFFFFFFF to status register clears all status bits.	4	12	20	28
Reset	Writing any value to this address causes the channel's counter to be reset to zero. (Write only register)	5	13	21	29
Transfer Preset	Writing any value to this address causes the counter to be set to the contents of the channel's preset register. (Write only register)	6	14	22	30

	Miscellaneous Group (See 6.3)			
Register number	7	15	23	31
Register name	Command (Command)	Time Stamp Latch (TS Latch)	33 MHz Time Stamp Counter (TS Counter)	General Purpose Low Frequency Counter (LF Counter)

6.1 Control Registers (reg.#3, reg.#11, reg.#19, reg.#27)

32-bit register:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0																	0	0	0	0	0	0	0	

'0' indicates unused or reserved bit which always returns '0' when being read.

Bit	Group name	Bit name	Default value after reset
31-24	**UNUSED**	31: unused31	
		30: unused30	
		29: unused29	
		28: unused28	
		27: unused27	
		26: unused26	
		25: unused25	
		24: unused24	
23	**CAPTURE**	23: ctrl_enable_capture	0
22-20	**INDEX**	22: ctrl_index_preset_not_reset	0
		21: ctrl_invert_index	0
		20: ctrl_enable_index	0
19	**SWAP QUADRATURE A/B**	19: ctrl_swap_quad	0
18	**MASTER ENABLE**	18: ctrl_enable	0
17-16	**COUNT MODE**	17: count_mode1	0
		16: count_mode0	0
15-14	**QUAD MODE**	15: quad_mode1	0
		14: quad mode0	0
13-7	**TRIGGER**	13: ctrl_trigger_retard	0
		12: ctrl_trigger_advance	0
		11: ctrl_trigger_index	0
		10: ctrl_trigger_borrow	0
		9: ctrl_trigger_carry	0
		8: ctrl_trigger_match	0
		7: ctrl_trigger_zero	0
6-0	**RESERVED**	6: reserved6	
		5: reserved5	
		4: reserved4	
		3: reserved3	
		2: reserved2	
		1: reserved1	
		0: reserved0	

Description

Registers 3, 11, 19, 27 are used to hold a 24-bit value that determines the operation of their respective channel. The following table defines what each of 24 bits control.

Bit	Description of Control
31-24	Unused.
23	Allow trigger_in to cause transfer from accumulator to output latch register.
22	When set and index is enabled, causes preset; otherwise if this bit is low a reset will occur when index is true.
21	Set for active low index, (invert index); leave reset for active high index.
20	When set, an index event will either reset or preset accumulator.
19	Swap A input and B input of quadrature signals. Count direction will be changed.
18	Master enable for accumulator.
17, 16	Governs the behavior of the internal counter at limits: • '00' accumulator acts like a simple 24 bit counter – counts from 0 up to 16,777,215 and then rolls over to 0 again and resumes counting upward; counting downwards, the counter goes from 0 to 16,777,215 and continues downwards. • '01' accumulator uses preset register in range-limit mode – when count reaches 0 or the preset value the counter freezes until the inputs cause a change in direction that keeps the counter within the bounds of 0 and preset value. • '10' accumulator uses preset register in non-recycle mode – when count reaches 0 going down or the preset value going upwards, the counter is frozen until a channel reset is performed. • '11' accumulator uses preset register in modulo-N mode – the counter counts upward until it matches the preset value, then rolls over to 0, and resumes counting upwards; when counting downward the counter rolls under from 0 to the preset value, and counts downward from there.
15, 14	Determines how the accumulator increments: This bit may be referred to as either quadrature mode or multiplier. • '00' A input = clock and B input = direction. Each rising edge of A input causes a counter increment or decrement, depending on the level of B input. • '01' accumulator increments once for every for four quadrature states (X1) • '10' accumulator increments once for every for two quadrature states (X2) • '11' accumulator increments once for every for quadrature state (X4)
13	Trigger occurs when accumulator decreases or retards.
12	Trigger occurs when accumulator increases or advances.
11	Trigger occurs when edge of index occurs.
10	Trigger occurs when rolling under 0 to N-1 in modulo-N mode.
9	Trigger occurs when rolling over N-1 to 0 in modulo-N mode.
8	Trigger occurs when accumulator equals match register.
7	Trigger occurs when accumulator equals zero.
6 - 0	Reserved for future use.

6.2 Status Registers (reg.#4, reg.#12, reg.#20, reg.#28)

32-bit register:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0														0	0	0	0	0	0	0	0

'0' indicates unused or reserved bit which always returns '0' when being read.

Bit	Group name	Bit name
31-24	**UNUSED**	31: unused31
		30: unused30
		29: unused29
		28: unused28
		27: unused27
		26: unused26
		25: unused25
		24: unused24
23-21	**RESERVED**	23: reserved23
		22: reserved22
		21: reserved21
20-14	**EVENT DETECTED**	20: retard_detected
		19: advance_detected
		18: index_detected
		17: borrow_detected
		16: carry_detected
		15: match_detected
		14: zero_detected
13-7	**LATCHED EVENT DETECTED**	13: latched_retard_detected
		12: latched_advance_detected
		11: latched_index_detected
		10: latched_borrow_detected
		9: latched_carry_detected
		8: latched_match_detected
		7: latched_zero_detected
6-0	**RESERVED**	6: reserved6
		5: reserved5
		4: reserved4
		3: reserved3
		2: reserved2
		1: reserved1
		0: reserved0

Description

The following defines bits of the status registers:

Bit	Status
31-24	Unused
23-21	Reserved for future use
20	Retard of quadrature is detected
19	Advance of quadrature is detected
18	Index is detected
17	Borrow is detected
16	Carry is detected
15	Internal counter reaching a value that corresponds to the Match register.
14	Internal counter passing through zero.
13	Latched value of bit 20
12	Latched value of bit 19
11	Latched value of bit 18
10	Latched value of bit 17
9	Latched value of bit 16
8	Latched value of bit 15
7	Latched value of bit 14
6-0	Reserved for future use.

Writing 0xFFFFFFFF to a Status register will clear all of its status bits to 0.

6.3 Miscellaneous Registers (reg.#7, reg.#15, reg.#23, reg.#31)

This section describes the features associated with each of the following command registers:

- **Command (Register #7) : Command Register**

Bit	Description
31-24	ROM identification (ROM_ID). These bits are read only and their values may vary between boards.
23-7	Reserved.
6	This bit is used to run or stop 33 MHz Time Stamp Counter (TS Counter). 0: Time Stamp counter is running. 1: Time Stamp counter stops at count 0.
5	Bit 5 is used to cause the TS Counter to transfer to the Time Stamp Latch Register (TS Latch). Transition from 0 to 1 of this bit (write '0' then write '1') will generate a one-shot pulse that causes the transfer.
4	Bit 4 is used to initiate a software-capture-all which causes the TS Counter to transfer to the TS Latch and then causes all accumulators with captured enabled to transfer to the Output Latch. Transition from 0 to 1 of this bit (write '0' then write '1') will generate a one-shot pulse that causes all the described transfers.
3-0	Bits 0-3 of Command Register are used to initiate an acquisition of the current absolute position for each channel respectively. One or more of these bits must be brought low (0) for a period of at least 65 milliseconds and then brought high (1) for the same period. This feature is only supported by encoders that provide the ability to receive remote commands.

- **TS Latch (Register #15) : Time Stamp Latch Register**
  This register is used to hold the Time Stamp Output Latch value. When a trigger event occurs, the value of TS Counter will be transferred to TS Latch.

- **TS Counter (Register #23) : 33 MHz Time Stamp Counter**
  This register is used to hold a 32-bit counter running at 33 MHz.

- **LF Counter (Register #31) : General Purpose Low Frequency Counter**
  This register is used to hold a general purpose 32-bit free running counter.

7 Trigger/Capture Feature

The PCI-4E is organized into encoder channels. Each encoder channel may be programmed to recognize a variety of conditions (such as the counter passing through zero, or the counter being equal to the match register). This recognition of conditions is achieved by setting trigger bits in the control register, (bit 7 to bit 13 of Control register). When a trigger bit for a condition is enabled within a given channel, that triggering channel generates a hardware signal that goes out of the channel; the triggers from all the channels are OR'ed together to form a capture signal. This single capture signal goes into all the channels; each channel may be programmed, (bit 23 of Control register). to accept this signal (the channel is said to have "capture enabled"). When enabled, the capture signal causes a channel to transfer the contents of the accumulator (counter) to the output latch, without any software intervention. Recall that in normal operation the host software must issue a write to the "transfer output" register to cause the accumulator contents to be copied to the output latch; all that the triggering system provides is a way for this transfer to be done by hardware when a condition is recognized on a triggering channel.

In order to facilitate analyzing or processing of encoder position data that require precise timing information, 33 MHz Time Stamp Counter (TS Counter) and Time Stamp Latch register (TS Latch) are incorporated into triggering scheme. When a trigger signal is generated, the value of TS Counter is automatically latched to TS Latch register.

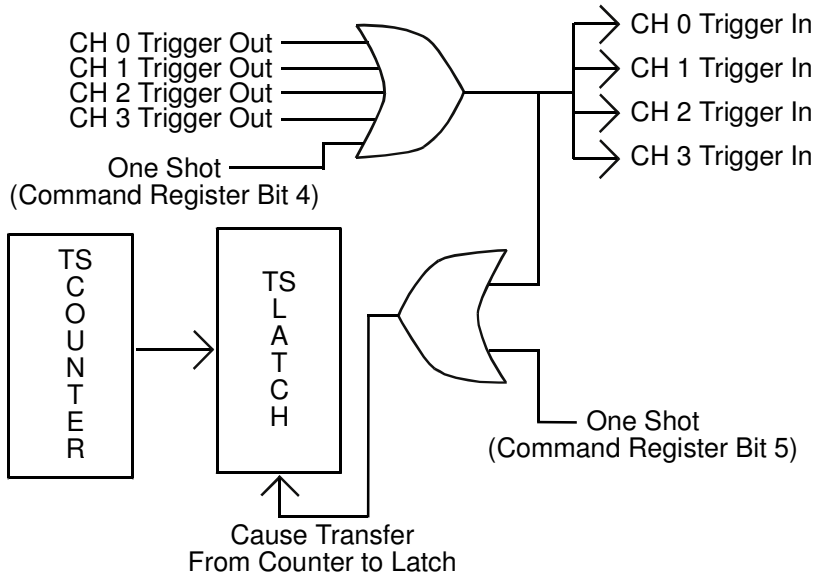


Figure 7.1 Trigger and Time Stamp Circuit

This triggering capability is most useful in the context of a data recording function. In a data recording function the user may desire to have an ordered sequence of samples stored to a disk file. Most often this sequence is generated by storing samples according to a periodic time function, but the triggering function described above allows a sample to be captured as a function of the data itself – when one channel's accumulator reaches zero, for instance, all channels may be programmed to capture the contents of their accumulators. Successive triggers would then capture successive samples.

So, the host software must be used to record a sequence of samples. In the case of periodic sampling, the host would set up a software timer; upon expiration of the timer the host would issue a write to the "transfer output" register to cause the accumulator contents to be copied to the output latch of all channels, then read the output latch of each of the channels in order, and write the data to a disk file.

In the case of a triggered acquisition sequence (actually, a better term would be storage qualified sequence) the host must set up one or more channels to generate triggering conditions, and then set up one or more channels to respond to the triggers (capture enabled). The host software is responsible for reading the output latch at the right moment, just after a capture has occurred.

The way to accomplish a storage qualified acquisition is for the host software to poll the status register of that channel, looking for a capture detected bit to be set. When a capture detected bit is found to be set, the host knows that a trigger has been generated, and data is waiting to be read in the output latches of the capture-enabled channels. This data should be read and stored to disk file, and then the status should be cleared on the triggering channel by writing 0xFFFFFFFF to the status register. Then the host may go back to polling the status register for captured event activity. Note that when collecting storage-qualified samples the host does not ever issue a write to the "transfer output" register; the hardware performs that function.

8 Typical Usage Scenario

This section describes a typical usage scenario.

```
#define PCI4E_CHANNEL_COUNT 4
int iRetVal = S_OK;
short  iCardCnt = 0;
long lDefCtrlMode = 0xF4000;      // Using the Table from section 5.1 Control Bits, identify the appropriate
                                  // control bits.  In this example the following bits are set:
                                  // Bit 14 - use X1 multiplier for quadrature mode.
                                  // Bit 16, 17 combined for a modulo-N counter mode.
                                  // Bit 18 – enable counter.
                                  // Bit 19 – swap a and b quadrature (direction is forward).
long lPresetVal = 999;      // Assume we have a 1000 line encoder, (CPR = 1000).
// PCI4E_Initialize must be called before making any other PCI4E dll calls.
iRetVal = PCI4E_Initialize(&iCardCnt);
if(iRetVal!=S_OK)
{
        // Handle error condition.
}
else
{
        // Iterate through each card…
        for(int i =0; i < iCardCnt: i++)
        {
                // Iterate through each encoder channel…
                for(int j=0; j < PCI4E_CHANNEL_COUNT; j++)
        {
                        // Set the default control mode.
                        iRetVal = PCI4E_SetControlMode(i, j, lDefCtrlMode);
                        if(iRetVal != S_OK)
{       // Handle error condition.   }
                        // Set the preset register for each encoder channel to the encoder's CPR -1.
                        iRetVal = PCI4E_SetPresetValue(i, j, lPresetVal);
                        if(iRetVal != S_OK)
{       // Handle error condition.  }
        }
        }
}
…
// Read each encoder's count.  In this scenario, the OnStatusTimer function
// has been passed as a callback function to the Win32 API SetTimer(…) function.
void OnStatusTimer()
{
        long lCntVal = 0;
        long lStatusVal = 0;
        // Iterate through each card…
        for(int i =0; i < iCardCnt: i++)
        {
                // Iterate through each encoder channel…
                for(int j=0; j < PCI4E_CHANNEL_COUNT; j++)
                {
                        // Get the encoder's count value.
                        lResult = PCI4E_GetCount(i, j, &lCntVal)
```

```
                                        if(iRetVal != S_OK){
                                                // Handle error condition.
                                        }else{
                                                // Do something with the count value…
                                        }
                                        // Get the channel's status and determine if a captured event occurred.
                                        iRetVal = PCI4E_GetStatus(i, j, &lStatusVal);
                                        if(iRetVal != S_OK){
                                                // Handle error condition.
                                        }else{
                                                // Do something with the status value…
                                        }
                                }
                        }
                }
        …
        // Cleanup before exiting…
PCI4E_Shutdown();
```

9 Function Calls

User applications may utilize the PCI-4E by calling provided functions in the PCI-4E's Dynamic Link Library (DLL). Function calls are categorized into 3 groups as follows.

- Basic functions
- PCI-4E card information functions
- User friendly functions

9.1 Basic functions (5 functions)

After understanding the meanings and functions of the 6 registers in each Channel Group, advanced users can set up and access PCI-4E with just five basic function calls.

PCI Initalizing (important)

```
9.4.29 PCI4E_Initialize
9.4.2  PCI4E_CardCount
9.4.56 PCI4E_Shutdown
```

Registers Read/Write (important) (Access by device number and register number)

```
9.4.32 PCI4E_ReadRegister
9.4.57 PCI4E_WriteRegister
```

9.2 PCI-4E card information functions (3 functions)

Three functions are provided for acquiring information related to PCI-4E card.

Functions to get PCI-4E card information (optional).

9.4.28 PCI4E_GetVersion
9.4.17 PCI4E_GetROM_ID
9.4.18 PCI4E_GetSlotNumber

9.3 User friendly functions (49 functions)

To facilitate programming with high readability, user friendly functions named with their features have been provided. Please note that advanced users can substitute all user friendly functions by reading/writing specific registers. A user friendly function that changes only a specific bit or bits of a register preserves value of other bits by writing back with the same value.

Registers Read/Write Group : (Access by device number and encoder number)

Register Name	Write functions	Read functions
Preset	9.4.48 PCI4E_SetPresetValue	9.4.16 PCI4E_GetPresetValue
Output Latch	9.4.7 PCI4E_GetCount (*Write & Read)	9.4.31 PCI4E_ReadOutputLatch(**)
Match	9.4.45 PCI4E_SetMatch	9.4.13 PCI4E_GetMatch
Control	9.4.38 PCI4E_SetControlMode	9.4.6 PCI4E_GetControlMode
Status	9.4.3 PCI4E_ClearCapturedStatus	9.4.19 PCI4E_GetStatus
Reset	9.4.35 PCI4E_ResetCount	N/A
Transfer Preset	9.4.30 PCI4E_PresetCount	N/A

Overview
Functions in this group read or write specific registers of a selected channel. Functions under "Write functions" are equivalent to PCI4E_WriteRegister but using device number and encoder number as parameters for accessing registers. Also, functions under "Read functions" are equivalent to PCI4E_ReadRegister but using device number and encoder number for accessing registers. Encoder number is equivalent to channel number. Also note the following:

* Write & Read
PCI4E_GetCount, first, writes to Output Latch register to transfer the value from the internal counter to the Output Latch register. Then, it immediately reads the Output Latch register to acquire the just transferred value. Use this function as a convenient way to get updated count of encoders when not using trigger/capture feature.

**
When using the trigger/capture feature to transfer the internal counter value to the Ouput Latch register, use the PCI4E_ReadOutputLatch function to simply read the last latched counter value.

Counter Set-up Group

Write functions	Read functions
9.4.40 PCI4E_SetCounterMode	9.4.8 PCI4E_GetCounterMode
9.4.46 PCI4E_SetMultiplier	9.4.14 PCI4E_GetMultiplier
9.4.43 PCI4E_SetForward	9.4.11 PCI4E_GetForward
9.4.48 PCI4E_SetPresetValue (***)	9.4.16 PCI4E_GetPresetValue (***)
9.4.41 PCI4E_SetEnableAccumulator	9.4.9 PCI4E_GetEnableAccumulator

Overview
Functions in this group will help you set up the counter mode to match your system. Normal step involves calling PCI4E_SetCounterMode, PCI4E_SetMultiplier and PCI4E_SetForward. If the counter mode other than 'simple 24 bit counter' is selected, PCI4E_SetPresetValue must be called to specify preset value. Call PCI4E_SetEnableAccumulator to start the internal counter.

Function PCI4E_Get... under "Read functions" may be used to verify their PCI4E_Set... counterparts.

Note: *** These functions also belong to Registers Read/Write Group.

Index Set-up Group

Write functions	Read functions
9.4.47 PCI4E_SetPresetOnIndex	9.4.15 PCI4E_GetPresetOnIndex
9.4.44 PCI4E_SetInvertIndex	9.4.12 PCI4E_GetInvertIndex
9.4.42 PCI4E_SetEnableIndex	9.4.10 PCI4E_GetEnableIndex

Overview
If index is required for resetting or presetting counter value, functions in this group should be called. PCI4E_SetPresetOnIndex will determine the action when index signal is detected, either resetting counter to 0 or presetting counter value equal to the value in preset register. PCI4E_SetInvertIndex facilitates polarity changing of index signal. Call PCI4E_SetEnableIndex to start watching for index signal.

Function PCI4E_Get... under "Read functions" may be used to verify their PCI4E_Set... counterparts.

Count Data Handling Group

Write functions	Read functions
9.4.7 PCI4E_GetCount (***)	9.4.31 PCI4E_ReadOutputLatch (***)
9.4.35 PCI4E_ResetCount (***)	
9.4.30 PCI4E_PresetCount(***)	
9.4.39 PCI4E_SetCount	
	9.4.1 PCI4E_CaptureTimeAndCounts
	9.4.33 PCI4E_ReadTimeAndCounts

Overview

After PCI4E_SetEnableAccumulator is called, the internal counter will be updated continuously based on signals input into A, B and Index pins. The internal counter cannot be read directly. The Output Latch register is used to relay the value of internal counter to external interface. To get the count value, two steps are needed. First, the Output Latch register must be written in order to transfer value from the internal counter to the Output Latch register. Second, the Output Latch register is read to retrieve the transferred value. These two steps are combined in PCI4E_GetCount function. This function is recommended when not using trigger/capture feature. PCI4E_ReadOutputLatch is normally called when trigger/latch feature is in use. It's because a trigger event will automatically transfer the count value from the internal counter to the Output Latch register. For a detailed explanation, please refer to section 'Trigger/Capture'.

PCI4E_ResetCount or PCI4E_PresetCount forces internal counter's value to zero or the same as Preset register's respectively.

PCI4E_SetCount forces internal counter's value to a specified value without permanently changing the Preset register. In fact, PCI4E_SetCount utilizes Preset register for transferring data to the internal counter, but the original value of Preset register is restored at the end of function call. When writing an application that always watches for changing of value of Preset register, the programmer must be aware of this temporary change of value.

PCI4E_ReadTimeAndCounts simply reads the TimeStamp Latch and each of the encoder's Output Latch while PCI4E_CaptureTimeAndCounts causes a synchronized capture of the TimeStamp counter and all channel accumulators that have captured enabled set true.

Note: *** These functions also belong to Registers Read/Write Group.

Time Stamp Group

Write functions	Read functions
9.4.34 PCI4E_ReadTimeStamp	
9.4.36 PCI4E_ResetTimeStamp	
	9.4.20 PCI4E_GetTimeStamp

Overview

PCI4E_ReadTimeStamp simply reads the Time Stamp Latch without causing the Time Stamp Counter to be transferred to the Time Stamp Latch. PCI4E_ResetTimeStamp sets the Time Stamp Counter value to zero. PCI4E_GetTimeStamp writes to the Command Register which causes the Time Stamp Counter to be latched to the Time Stamp Latch and then reads the Time Stamp Latch.

Trigger/Capture Feature Group

:Capture Functions

Write functions	Read functions
9.4.37 PCI4E_SetCaptureEnabled	9.4.5 PCI4E_GetCaptureEnabled

:Trigger Functions

Write functions	Read functions
9.4.45 PCI4E_SetMatch (***)	9.4.13 PCI4E_GetMatch (***)
9.4.50 PCI4E_SetTriggerOnIncrease	9.4.22 PCI4E_GetTriggerOnIncrease
9.4.51 PCI4E_SetTriggerOnIndex	9.4.23 PCI4E_GetTriggerOnIndex
9.4.52 PCI4E_SetTriggerOnMatch	9.4.24 PCI4E_GetTriggerOnMatch
9.4.49 PCI4E_SetTriggerOnDecrease	9.4.21 PCI4E_GetTriggerOnDecrease
9.4.53 PCI4E_SetTriggerOnRollover	9.4.25 PCI4E_GetTriggerOnRollover
9.4.54 PCI4E_SetTriggerOnRollunder	9.4.26 PCI4E_GetTriggerOnRollunder
9.4.55 PCI4E_SetTriggerOnZero	9.4.55 PCI4E_SetTriggerOnZero
9.4.3 PCI4E_ClearCapturedStatus (***)	9.4.19 PCI4E_GetStatus (***)

Overview

An encoder channel may be configured to generate a trigger signal when various conditions are met. This trigger signal is forwarded to all encoder channels where if the channel has capture enabled will transfer the internal counter value to the Output Latch register. The trigger signal will also transfer the Time Stamp Counter to the Time Stamp Latch regardless of any channel having capture enabled.

Function PCI4E_Get... under "Read functions" may be used to verify their PCI4E_Set... counterparts. Please refer to section "Trigger/Capture Feature" and definitions of each function in this group.

Note: *** These functions also belong to Registers Read/Write Group.

U2 Group (For U2 Absolute Encoders)

Write functions	Read functions
	9.4.4 PCI4E_GetABSPosition

Overview

Some U.S. Digital encoders are hybrid incremental/absolute encoders. This function sends the appropriate sequence of commands to a hybrid encoder causing the encoder to report its absolute position by issuing a sequence of quadrature state changes. This feature is only supported by encoders that provide the ability to receive remote commands.

9.4 Function Definitions

9.4.1 PCI4E_CaptureTimeAndCounts

Description:

This function causes a synchronized capture of the TimeStamp counter and all channel accumulators which have captured enabled set true. Note: the `lValues` parameter of this function is pointer to an array of 4 longs. Each item in the array will contain a channel's output latch count value.

C Language Function Prototype:

```
int _stdcall PCI4E_CaptureTimeAndCounts(short iDeviceNo, long *plValues,  long
*plTimeStamp);
```

Returns:

Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:

`iDeviceNo`: identifies the PCI-4E card (zero based).

`lValues`: see description above.

`lTimeStamp`: in out parameter containing the TimeStamp value.

Example C Usage:

```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
long lValues[4] = {0};
long lTimeStamp = 0;
iResult = PCI4E_CaptureTimeAndCounts(iDeviceNo, &lValues, &lTimeStamp);
if ( iResult != S_OK ){ // Handle error...}
```

VB Language Function Declaration:

```
Public Declare Function PCI4E_CaptureTimeAndCounts Lib "USD_PCI_4E.dll" (ByVal
iDeviceNo As Integer, ByRef lValues As Long, ByRef lTimeStamp As Long) As Long
```

Example VB Usage:

```
Dim errCode       as Long
Dim iDevice       as Integer
Dim lValues(3)    as Long
Dim lTimeStamp    as long

iDevice = 0       ‘ Note: choose your value here.
iEncoder = 0      ‘ Note: choose your value here.
iMode = 1         ‘ Note: choose your value here.
                  ‘ See parameter description above.

errCode = PCI4E_CaptureTimeAndCounts(iDeviceNo, lValues(0), lTimeStamp);
If errCode <> 0 then
      ‘ Handle error..
End If
```

9.4.2 PCI4E_CardCount

Description:
This function returns the number of PCI-4E cards detected on the PCI bus. The value returned should be the same value as returned in the in-out `piDeviceCount` parameter of the `PCI4E_Initialize` function.

C Language Function Prototype:
```
int _stdcall PCI4E_CardCount();
```

Returns:
See description above.

Parameters:
None

Example C Usage:
```
short iCards = 0;
iCards = PCI4E_CardCount();
```

VB Language Function Declaration:
```
Public Declare Function PCI4E_CardCount Lib "USD_PCI_4E.dll" () As Long
```

Example VB Usage:
```
Dim iCards as Integer
iCards = PCI4E_CardCount();
```

9.4.3 PCI4E_ClearCapturedStatus

Description:
This function clears the captured event status by writing 0xFFFFFFFF into the status register.

Note: Refer to section 6.2 Status Registers.

C Language Function Prototype:
```
int _stdcall PCI4E_ClearCapturedStatus(short iDeviceNo, short iEncoder)
```

Returns:
Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:
`iDeviceNo`: identifies the PCI-4E card (zero based).
`iEncoder`: identifies the encoder channel (zero based).

Example C Usage:
```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
short iEncoder = 0;     // Note: choose your value here.

iResult  = PCI4E_ClearCapturedStatus(iDeviceNo, iEncoder);
if ( iResult != S_OK ){ // Handle error...}
```

VB Language Function Declaration:
```
Public Declare Function PCI4E_ClearCapturedStatus Lib "USD_PCI_4E.dll" (ByVal
iDeviceNo As Integer, ByVal iEncoder As Integer) As Long
```

Example VB Usage:
```
Dim errCode as Long
Dim iDevice as Integer
Dim iEncoder as Integer

iDevice = 0       ‘ Note: choose your value here.
iEncoder = 0      ‘ Note: choose your value here.

errCode = PCI4E_ClearCapturedStatus(iDeviceNo, iEncoder)
If errCode <> 0 then
      ‘ Handle error..
End If
```

9.4.4 PCI4E_GetABSPosition

Description:
Some U.S. Digital encoders are hybrid incremental/absolute encoders. This function sends the appropriate sequence of commands to a hybrid encoder causing the encoder to report its absolute position by issuing a sequence of quadrature state changes. This feature is only supported by encoders that provide the ability to receive remote commands.

C Language Function Prototype:
```
int _stdcall PCI4E_GetABSPosition(short iDeviceNo, short iEncoder, long
*plVal);
```

Returns:
Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:
`iDeviceNo`: identifies the PCI-4E card (zero based).
`iEncoder`: identifies the encoder channel (zero based).
`lVal`: in out parameter containing the absolute encoder position (count).

Example C Usage:
```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
short iEncoder = 0;     // Note: choose your value here.
long lVal = 0;

iResult  = PCI4E_GetABSPosition (iDeviceNo, iEncoder, &lVal );
if ( iResult != S_OK ){ // Handle error...}
```

VB Language Function Declaration:
```
Public Declare Function PCI4E_GetABSPosition Lib "USD_PCI_4E.dll" (ByVal
iDeviceNo As Integer, ByVal iEncoder As Integer, ByRef lVal As Long) As Long
```

Example VB Usage:
```
Dim errCode as Long
Dim iDevice as Integer
Dim iEncoder as Integer
Dim lVal as Long

iDevice = 0       ‘ Note: choose your value here.
iEncoder = 0      ‘ Note: choose your value here.

errCode = PCI4E_GetABSPosition(iDeviceNo, iEncoder, lVal)
If errCode <> 0 then
      ‘ Handle error..
End If
```

9.4.5 PCI4E_GetCaptureEnabled

Description:
This function retrieves a Boolean value that identifies if trigger_in causes a transfer from accumulator (counter) to output.

C Language Function Prototype:
```
int _stdcall PCI4E_GetCaptureEnabled(short iDeviceNo, short iEncoder, BOOL
*pbVal);
```

Returns:
Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:
`iDeviceNo`: identifies the PCI-4E card (zero based).
`iEncoder`: identifies the encoder channel (zero based).
`bVal`: inout parameter. A return value of zero is False otherwise its True.

Example C Usage:
```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
short iEncoder = 0;     // Note: choose your value here.
BOOL bVal = 0;

iResult  = PCI4E_GetCaptureEnabled(iDeviceNo, iEncoder, &bVal);
if ( iResult != S_OK ){ // Handle error...}
```

VB Language Function Declaration:
```
Public Declare Function PCI4E_GetCaptureEnabled Lib "USD_PCI_4E.dll" (ByVal
iDeviceNo As Integer, ByVal iEncoder As Integer, ByRef lVal As Long) As Long
```

Example VB Usage:
```
Dim errCode as Long
Dim iDevice as Integer
Dim iEncoder as Integer
Dim lVal as Long

iDevice = 0       ‘ Note: choose your value here.
iEncoder = 0      ‘ Note: choose your value here.

errCode = PCI4E_GetCaptureEnabled (iDeviceNo, iEncoder, lVal)
If errCode <> 0 then
      ‘ Handle error..
End If
```

9.4.6 PCI4E_GetControlMode

Description:
This function retrieves a 32 bit value from the control register. This value is used to control the operation of a channel. *See section 6.1 Control Registers*

C Language Function Prototype:
```
int _stdcall PCI4E_GetControlMode(short iDeviceNo, short iEncoder, long
*plMode);
```

Returns:
Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:
`iDeviceNo`: identifies the PCI-4E card (zero based).
`iEncoder`: identifies the encoder channel (zero based).
`lMode`: 32 bit in-out parameter containing the control mode.

Example C Usage:
```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
short iEncoder = 0;     // Note: choose your value here.
long lMode = 0;

iResult  = PCI4E_GetControlMode(iDeviceNo, iEncoder, &lMode);
if ( iResult != S_OK ){ // Handle error...}
```

VB Language Function Declaration:
```
Public Declare Function PCI4E_GetControlMode Lib "USD_PCI_4E.dll" (ByVal
iDeviceNo As Integer, ByVal iEncoder As Integer, ByRef lMode As Long) As Long
```

Example VB Usage:
```
Dim errCode as Long
Dim iDevice as Integer
Dim iEncoder as Integer
Dim lMode as Long

iDevice = 0       ‘ Note: choose your value here.
iEncoder = 0      ‘ Note: choose your value here.

errCode = PCI4E_GetControlMode(iDeviceNo, iEncoder, lMode)
If errCode <> 0 then
      ‘ Handle error..
End If
```

9.4.7 PCI4E_GetCount

Description:

This function retrieves the count value.
Notes: This function performs the following two steps.
(1) Write to Output Latch registers (reg.#1, reg.#9, reg.#17, or reg.#25 based on channel selected). This action will transfer value from internal counter register to Output Latch registers.
(2) Read from Output Latch registers (reg.#1, reg.#9, reg.#17, or reg.#25 based on channel selected). The result of this read is the updated value from Output Latch register which is passed to pVal.

Caveats: This PCI4E_GetCount is a convenient function to easily get encoder counts from PCI-4E. However, if you want to use triggering features of PCI-4E to transfer data from internal counter to Output Latch register, you should use PCI4E_ReadRegister instead of PCI4E_GetCount. In this case, using PCI4E_GetCount to read data will result in overwriting on the count value when the trigger event occurred.

C Language Function Prototype:

```
int _stdcall PCI4E_GetCount(short iDeviceNo, short iEncoder, long *plVal);
```

Returns:

Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:

`iDeviceNo`: identifies the PCI-4E card (zero based).
`iEncoder`: identifies the encoder channel (zero based).
`plVal`: in/out parameter that will receive the encoder count value.

Example C Usage:

```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
short iEncoder = 0;     // Note: choose your value here.
long lCount = 0;

iResult = PCI4E_GetCount(iDeviceNo, iEncoder, &lCount);
if ( iResult != S_OK ){ // Handle error...}
```

VB Language Function Declaration:

```
Public Declare Function PCI4E_GetCount Lib "USD_PCI_4E.dll" (ByVal iDeviceNo As
Integer, ByVal iEncoder As Integer, ByRef lCount) As Long
```

Example VB Usage:

```
Dim errCode as Long
Dim lCount as Long
Dim iDevice as Integer
Dim iEncoder as Integer
```

```
iDevice = 0       ‘ Note: choose your value here.
iEncoder = 0      ‘ Note: choose your value here.

errCode = PCI4E_GetCount(iDeviceNo, iEncoder, lCount)
If errCode <> 0 then
      ‘ Handle error..
End If
```

9.4.8 PCI4E_GetCounterMode

Description:

The function retrieves a control bit that governs the counter behavior and limits. See parameters sections for description of the possible counter modes.

C Language Function Prototype:

```
int _stdcall PCI4E_GetCounterMode(short iDeviceNo, short iEncoder, short
*piMode);
```

Returns:

Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:

`iDeviceNo`: identifies the PCI-4E card (zero based).

`iEncoder`: identifies the encoder channel (zero based).

`piMode`: 32 bit in-out parameter containing the counter mode.

0 = acc. acts like a 24 bit counter
1 = acc. uses preset register in range-limit mode
2 = acc. uses preset register in non-recycle mode
3 = acc. uses preset register in modulo-N mode.

See 6.1 Control Registers for explanation of modes.

Example C Usage:

```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
short iEncoder = 0;     // Note: choose your value here.
short iMode = 0;

iResult  = PCI4E_GetControlMode(iDeviceNo, iEncoder, &iMode );
if ( iResult != S_OK ){ // Handle error... }
```

VB Language Function Declaration:

```
Public Declare Function PCI4E_GetCounterMode Lib "USD_PCI_4E.dll" (ByVal
iDeviceNo As Integer, ByVal iEncoder As Integer, ByRef iMode As Integer) As
Long
```

Example VB Usage:

```
Dim errCode as Long
Dim iDevice as Integer
Dim iEncoder as Integer
Dim iMode as Integer

iDevice = 0       ‘ Note: choose your value here.
iEncoder = 0      ‘ Note: choose your value here.

errCode = PCI4E_GetCounterMode(iDeviceNo, iEncoder, iMode)
If errCode <> 0 then
      ‘ Handle error..
End If
```

9.4.9 PCI4E_GetEnableAccumulator

Description:

This function retrieves a Boolean value that indicates whether the master enable for accumulator is set, (must be set to true to count).

C Language Function Prototype:

```
int _stdcall PCI4E_GetEnableAccumulator(short iDeviceNo, short iEncoder, BOOL
*pbVal);
```

Returns:

Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:

`iDeviceNo`: identifies the PCI-4E card (zero based).
`iEncoder`: identifies the encoder channel (zero based).
`pbVal`: in-out Boolean parameter identifying whether the counter is enabled.

Example C Usage:

```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
short iEncoder = 0;     // Note: choose your value here.
BOOL bVal = FALSE;

iResult  = PCI4E_GetEnableAccumulator(iDeviceNo, iEncoder, &bVal);
if ( iResult != S_OK ){ // Handle error... }
```

VB Language Function Declaration:

```
Public Declare Function PCI4E_GetEnableAccumulator Lib "USD_PCI_4E.dll" (ByVal
iDeviceNo As Integer, ByVal iEncoder As Integer, ByRef lVal As Long) As Long
```

Example VB Usage:

```
Dim errCode as Long
Dim iDevice as Integer
Dim iEncoder as Integer
Dim lVal as Long

iDevice = 0       ‘ Note: choose your value here.
iEncoder = 0      ‘ Note: choose your value here.

errCode = PCI4E_GetEnableAccumulator(iDeviceNo, iEncoder, lVal)
If errCode <> 0 then
      ‘ Handle error..
End If
```

9.4.10 PCI4E_GetEnableIndex

Description:
This function retrieves a Boolean value that indicates whether index will either reset or preset accumulator (counter).

C Language Function Prototype:
```
int _stdcall PCI4E_GetEnableIndex(short iDeviceNo, short iEncoder, BOOL
*pbVal);
```

Returns:
Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:
`iDeviceNo`: identifies the PCI-4E card (zero based).
`iEncoder`: identifies the encoder channel (zero based).
`pbVal`: in-out Boolean parameter identifying whether the index is enabled.

Example C Usage:
```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
short iEncoder = 0;     // Note: choose your value here.
BOOL bVal = FALSE;

iResult  = PCI4E_GetEnableIndex(iDeviceNo, iEncoder, &bVal);
if ( iResult != S_OK ){ // Handle error... }
```

VB Language Function Declaration:
```
Public Declare Function PCI4E_GetEnableIndex Lib "USD_PCI_4E.dll" (ByVal
iDeviceNo As Integer, ByVal iEncoder As Integer, ByRef lVal As Long) As Long
```

Example VB Usage:
```
Dim errCode as Long
Dim iDevice as Integer
Dim iEncoder as Integer
Dim lVal as Long

iDevice = 0       ‘ Note: choose your value here.
iEncoder = 0      ‘ Note: choose your value here.

errCode = PCI4E_GetEnableIndex(iDeviceNo, iEncoder, lVal)
If errCode <> 0 then
      ‘ Handle error..
End If
```

9.4.11 PCI4E_GetForward

Description:
This function retrieves a Boolean value that indicates whether A input and B input of quadrature signals are swapped. True: Swapped, False: Not swapped.

C Language Function Prototype:
```
int _stdcall PCI4E_GetForward(short iDeviceNo, short iEncoder, BOOL *pbVal);
```

Returns:
Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:
`iDeviceNo`: identifies the PCI-4E card (zero based).
`iEncoder`: identifies the encoder channel (zero based).
`pbVal`: in-out Boolean parameter identifying the counting direction.

Example C Usage:
```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
short iEncoder = 0;     // Note: choose your value here.
BOOL bVal = FALSE;

iResult  = PCI4E_GetForward(iDeviceNo, iEncoder, &bVal);
if ( iResult != S_OK ){ // Handle error... }
```

VB Language Function Declaration:
```
Public Declare Function PCI4E_GetForward Lib "USD_PCI_4E.dll" (ByVal iDeviceNo
As Integer, ByVal iEncoder As Integer, ByRef lVal As Long) As Long
```

Example VB Usage:
```
Dim errCode as Long
Dim iDevice as Integer
Dim iEncoder as Integer
Dim lVal as Long

iDevice = 0       ‘ Note: choose your value here.
iEncoder = 0      ‘ Note: choose your value here.

errCode = PCI4E_GetForward(iDeviceNo, iEncoder, lVal)
If errCode <> 0 then
      ‘ Handle error..
End If
```

9.4.12 PCI4E_GetInvertIndex

Description:
This function retrieves a Boolean value that determines if active low index is set.

C Language Function Prototype:
```
int _stdcall PCI4E_GetInvertIndex(short iDeviceNo, short iEncoder, BOOL
*pbVal);
```

Returns:
Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:
`iDeviceNo`: identifies the PCI-4E card (zero based).
`iEncoder`: identifies the encoder channel (zero based).
`pbVal`: in-out Boolean parameter. See description above.

Example C Usage:
```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
short iEncoder = 0;     // Note: choose your value here.
BOOL bVal = FALSE;

iResult  = PCI4E_GetInvertIndex(iDeviceNo, iEncoder, &bVal);
if ( iResult != S_OK ){ // Handle error... }
```

VB Language Function Declaration:
```
Public Declare Function PCI4E_GetInvertIndex Lib "USD_PCI_4E.dll" (ByVal
iDeviceNo As Integer, ByVal iEncoder As Integer, ByRef lVal As Long) As Long
```

Example VB Usage:
```
Dim errCode as Long
Dim iDevice as Integer
Dim iEncoder as Integer
Dim lVal as Long

iDevice = 0       ‘ Note: choose your value here.
iEncoder = 0      ‘ Note: choose your value here.

errCode = PCI4E_GetInvertIndex(iDeviceNo, iEncoder, lVal)
If errCode <> 0 then
      ‘ Handle error..
End If
```

9.4.13 PCI4E_GetMatch

Description:
This function retrieves the Match register value. The Match register is used as a reference to signal a capture when the counter equals the match register value.

C Language Function Prototype:
```
int _stdcall PCI4E_GetMatch(short iDeviceNo, short iEncoder, long *plVal);
```

Returns:
Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:
`iDeviceNo`: identifies the PCI-4E card (zero based).
`iEncoder`: identifies the encoder channel (zero based).
`plVal`: in-out parameter containing the match register value.

Example C Usage:
```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
short iEncoder = 0;     // Note: choose your value here.
long lVal = 0;

iResult  = PCI4E_GetMatch(iDeviceNo, iEncoder, &lVal);
if ( iResult != S_OK ){ // Handle error... }
```

VB Language Function Declaration:
```
Public Declare Function PCI4E_GetMatch Lib "USD_PCI_4E.dll" (ByVal iDeviceNo As
Integer, ByVal iEncoder As Integer, ByRef lVal As Long) As Long
```

Example VB Usage:
```
Dim errCode as Long
Dim iDevice as Integer
Dim iEncoder as Integer
Dim lVal as Long

iDevice = 0       ‘ Note: choose your value here.
iEncoder = 0      ‘ Note: choose your value here.

errCode = PCI4E_GetMatch(iDeviceNo, iEncoder, lVal)
If errCode <> 0 then
      ‘ Handle error..
End If
```

9.4.14 PCI4E_GetMultiplier

Description:
This function retrieves a control bit that determines when the counter is to be incremented based on the number of quadrature state transitions. See possible values in parameters description.

C Language Function Prototype:
```
int _stdcall PCI4E_GetMultiplier(short iDeviceNo, short iEncoder, short
*piMode);
```

Returns:
Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:
`iDeviceNo`: identifies the PCI-4E card (zero based).
`iEncoder`: identifies the encoder channel (zero based). 0 - 3
`piMode`: identifies when the counter increments.

Possible values are: 0 = in_a is clock, in_b is direction
1 = count increments once every four quad states, X1
2 = count increments once every two quad states, X2
3 = count increments once every quad state, X4

Example C Usage:
```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
short iEncoder = 0;     // Note: choose your value here.
long lVal = 0;

iResult  = PCI4E_GetMultiplier(iDeviceNo, iEncoder, &lVal);
if ( iResult != S_OK ){ // Handle error... }
```

VB Language Function Declaration:
```
Public Declare Function PCI4E_GetMultiplier Lib "USD_PCI_4E.dll" (ByVal
iDeviceNo As Integer, ByVal iEncoder As Integer, ByRef lVal As Long) As Integer
```

Example VB Usage:
```
Dim errCode as Long
Dim iDevice as Integer
Dim iEncoder as Integer
Dim lVal as Long

iDevice = 0       ‘ Note: choose your value here.
iEncoder = 0      ‘ Note: choose your value here.

errCode = PCI4E_GetMultiplier(iDeviceNo, iEncoder, lVal)
If errCode <> 0 then
      ‘ Handle error..
End If
```

9.4.15 PCI4E_GetPresetOnIndex

Description:
This function retrieves a Boolean value that determines whether a preset occurs when set and index is enabled.

C Language Function Prototype:
```
int _stdcall PCI4E_GetPresetOnIndex(short iDeviceNo, short iEncoder, BOOL
pbVal);
```

Returns:
Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:
`iDeviceNo`: identifies the PCI-4E card (zero based).
`iEncoder`: identifies the encoder channel (zero based).
`pbVal`: in-out parameter that will receive the preset on index enable value.

Example C Usage:
```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
short iEncoder = 0;     // Note: choose your value here.
BOOL bVal = False;

iResult = PCI4E_GetPresetOnIndex(iDeviceNo, iEncoder, &bVal);
if ( iResult != S_OK ){ // Handle error...}
```

VB Language Function Declaration:
```
Public Declare Function PCI4E_GetPresetOnIndex Lib "USD_PCI_4E.dll" (ByVal
iDeviceNo As Integer, ByVal iEncoder As Integer, ByRef lVal As Long) As Long
```

Example VB Usage:
```
Dim errCode as Long
Dim iDevice as Integer
Dim iEncoder as Integer
Dim lVal as Long

iDevice = 0       ‘ Note: choose your value here.
iEncoder = 0      ‘ Note: choose your value here.

errCode = PCI4E_GetPresetOnIndex(iDeviceNo, iEncoder, lVal)
If errCode <> 0 then
      ‘ Handle error..
End If
```

9.4.16 PCI4E_GetPresetValue

Description:
This function retrieves the Preset Register value.

C Language Function Prototype:
```
int _stdcall PCI4E_GetPresetValue(short iDeviceNo, short iEncoder, long
*plVal);
```

Returns:
Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:
`iDeviceNo`: identifies the PCI-4E card (zero based).
`iEncoder`: identifies the encoder channel (zero based).
`plVal`: the preset register value may also be referred to as upper-limit, range-limit, max count, or resolution -1.

Example C Usage:
```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
short iEncoder = 0;     // Note: choose your value here.
long lVal = 0;

iResult = PCI4E_GetPresetValue(iDeviceNo, iEncoder, &lVal );
if ( iResult != S_OK ){ // Handle error...}
```

VB Language Function Declaration:
```
Public Declare Function PCI4E_GetPresetValue Lib "USD_PCI_4E.dll" (ByVal
iDeviceNo As Integer, ByVal iEncoder As Integer, ByRef lVal As Long) As Long
```

Example VB Usage:
```
Dim errCode as Long
Dim iDevice as Integer
Dim iEncoder as Integer
Dim lVal as Long

iDevice = 0       ‘ Note: choose your value here.
iEncoder = 0      ‘ Note: choose your value here.

errCode = PCI4E_GetPresetValue(iDeviceNo, iEncoder, lVal);
If errCode <> 0 then
      ‘ Handle error..
End If
```

9.4.17 PCI4E_GetROM_ID

Description:
This function retrieves the ROM_ID which is contained bits 24 through 31 of the Command Register.

C Language Function Prototype:
```
int _stdcall PCI4E_GetROM_ID(short iDeviceNo, short *piROM_ID);
```

Returns:
Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:
`iDeviceNo`: identifies the PCI-4E card (zero based).
`piROM_ID`: an eight bit value that identifies the ROM ID.

Example C Usage:
```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
short iROMID = 0;

iResult = PCI4E_GetROM_ID(iDeviceNo, &iROMID);
if ( iResult != S_OK ){ // Handle error...}
```

VB Language Function Declaration:
```
Public Declare Function PCI4E_GetROM_ID Lib "USD_PCI_4E.dll" (ByVal iDeviceNo
As Integer, ByRef iROMID As Long) As Long
```

Example VB Usage:
```
Dim errCode as Long
Dim iDevice as Integer
Dim iROMID as Long

iDevice = 0       ‘ Note: choose your value here.
iEncoder = 0      ‘ Note: choose your value here.

errCode = PCI4E_GetROM_ID(iDeviceNo, iROMID);
If errCode <> 0 then
      ‘ Handle error..
End If
```

9.4.18 PCI4E_GetSlotNumber

Description:
This function retrieves the slot number associated with a given PCI-4E card.

C Language Function Prototype:
```
int _stdcall PCI4E_GetSlotNumber(short iDeviceNo, short *piSlotNo);
```

Returns:
Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:
`iDeviceNo`: identifies the PCI-4E card (zero based).
`piSlotNo`: identifies the associated slot number.

Example C Usage:
```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
short iSlotNo = 0;

iResult  = PCI4E_GetSlotNumber(iDeviceNo, &iSlotNo);
if ( iResult != S_OK ){ // Handle error... }
```

VB Language Function Declaration:
```
Public Declare Function PCI4E_GetSlotNumber Lib "USD_PCI_4E.dll" (ByVal
iDeviceNo As Integer, ByRef iSlotNo As Integer) As Long
```

Example VB Usage:
```
Dim errCode as Long
Dim iDevice as Integer
Dim iSlotNo as Integer

iDevice = 0       ‘ Note: choose your value here.
iEncoder = 0      ‘ Note: choose your value here.

errCode = PCI4E_GetSlotNumber(iDeviceNo, iSlotNo)
If errCode <> 0 then
      ‘ Handle error..
End If
```

9.4.19 PCI4E_GetStatus

Description:
This function retrieves the status register value for a channel.

C Language Function Prototype:
```
int _stdcall PCI4E_GetStatus(short iDeviceNo, short iEncoder, long *plVal);
```

Returns:
Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:
`iDeviceNo`: identifies the PCI-4E card (zero based).
`iEncoder`: identifies the encoder channel (zero based). 0 - 3 or 4 for all channels
`plVal`: in out parameters which contains the status register value. *Refer to section 6.2 Status Registers.*

Example C Usage:
```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
short iEncoder = 0;     // Note: choose your value here.
long lStatus = 0;

iResult  = PCI4E_GetStatus(iDeviceNo, iEncoder, &lStatus);
if ( iResult != S_OK ){ // Handle error... }
```

VB Language Function Declaration:
```
Public Declare Function PCI4E_GetStatus Lib "USD_PCI_4E.dll" (ByVal iDeviceNo
As Integer, ByVal iEncoder As Integer, ByRef lVal As Long) As Long
```

Example VB Usage:
```
Dim errCode as Long
Dim iDevice as Integer
Dim iEncoder as Integer
Dim lStatus as Long

iDevice = 0       ‘ Note: choose your value here.
iEncoder = 0      ‘ Note: choose your value here.

errCode = PCI4E_GetStatus(iDeviceNo, iEncoder, lStatus);
If errCode <> 0 then
      ‘ Handle error..
End If
```

9.4.20 PCI4E_GetTimeStamp

Description:

This function writes to the CMD_Register which causes the TimeStamp counter to be latched to the TimeStamp Latch and then reads the TimeStamp Latch. Refer to the ReadTimeStamp function to simply read the TimeStamp Latch without causing the TimeStamp counter to be transferred to the TimeStamp Latch.

C Language Function Prototype:

```
int _stdcall PCI4E_GetTimeStamp(short iDeviceNo, long *plTimeStamp);
```

Returns:

Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:

`iDeviceNo`: identifies the PCI-4E card (zero based).

`plTimeStamp`: in out parameter containing the TimeStamp Latch value.

Example C Usage:

```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
long lTimeStamp = 0;

iResult = PCI4E_GetTimeStamp(iDeviceNo, &lTimeStamp);
if ( iResult != S_OK ){ // Handle error...}
```

VB Language Function Declaration:

```
Public Declare Function PCI4E_GetTimeStamp Lib "USD_PCI_4E.dll" (ByVal
iDeviceNo As Integer,  ByRef lTimeStamp As Long) As Long
```

Example VB Usage:

```
Dim errCode as Long
Dim iDevice as Integer
Dim lTimeStamp as Long

iDevice = 0 ‘ Note: choose your value here.

errCode = PCI4E_GetTimeStamp(iDeviceNo, lTimeStamp);
If errCode <> 0 then
      ‘ Handle error…
End If
```

9.4.21 PCI4E_GetTriggerOnDecrease

Description:
This function retrieves a Boolean value that indicates whether a trigger occurs when the count decreases.

C Language Function Prototype:
```
int _stdcall PCI4E_GetTriggerOnDecrease(short iDeviceNo, short iEncoder, BOOL
*pbVal);
```

Returns:
Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:
`iDeviceNo`: identifies the PCI-4E card (zero based).
`iEncoder`: identifies the encoder channel (zero based). 0 - 3 or 4 for all channels
`pbVal`: in out parameters. See description above.

Example C Usage:
```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
short iEncoder = 0;     // Note: choose your value here.
BOOL bVal = False;

iResult  = PCI4E_GetTriggerOnDecrease(iDeviceNo, iEncoder, & bVal);
if ( iResult != S_OK ){ // Handle error... }
```

VB Language Function Declaration:
```
Public Declare Function PCI4E_GetTriggerOnDecrease Lib "USD_PCI_4E.dll" (ByVal
iDeviceNo As Integer, ByVal iEncoder As Integer, ByRef lVal As Long) As Long
```

Example VB Usage:
```
Dim errCode as Long
Dim iDevice as Integer
Dim iEncoder as Integer
Dim lVal as Long

iDevice = 0       ‘ Note: choose your value here.
iEncoder = 0      ‘ Note: choose your value here.

errCode = PCI4E_GetTriggerOnDecrease(iDeviceNo, iEncoder, lVal);
If errCode <> 0 then
      ‘ Handle error…
End If
```

9.4.22 PCI4E_GetTriggerOnIncrease

Description:
This function retrieves a Boolean value that indicates whether a trigger will be set when a count increases.

C Language Function Prototype:
```
int _stdcall PCI4E_GetTriggerOnIncrease(short iDeviceNo, short iEncoder, BOOL
*pbVal);
```

Returns:
Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:
`iDeviceNo`: identifies the PCI-4E card (zero based).
`iEncoder`: identifies the encoder channel (zero based). 0 - 3 or 4 for all channels
`pbVal`: in-out Boolean parameter. See description above.

Example C Usage:
```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
short iEncoder = 0;     // Note: choose your value here.
BOOL bVal = False;

iResult  = PCI4E_GetTriggerOnIncrease(iDeviceNo, iEncoder, &bVal);
if ( iResult != S_OK ){ // Handle error... }
```

VB Language Function Declaration:
```
Public Declare Function PCI4E_GetTriggerOnIncrease Lib "USD_PCI_4E.dll" (ByVal
iDeviceNo As Integer, ByVal iEncoder As Integer, ByRef lVal As Long) As Long
```

Example VB Usage:
```
Dim errCode as Long
Dim iDevice as Integer
Dim iEncoder as Integer
Dim lVal as Long

iDevice = 0       ‘ Note: choose your value here.
iEncoder = 0      ‘ Note: choose your value here.

errCode = PCI4E_GetTriggerOnIncrease(iDeviceNo, iEncoder, lVal);
If errCode <> 0 then
      ‘ Handle error…
End If
```

9.4.23 PCI4E_GetTriggerOnIndex

Description:
This function retrieves a Boolean value that indicates whether a trigger will occur when the edge of index is detected.

C Language Function Prototype:
```
int _stdcall PCI4E_GetTriggerOnIndex(short iDeviceNo, short iEncoder, BOOL
*pbVal);
```

Returns:
Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:
`iDeviceNo`: identifies the PCI-4E card (zero based).
`iEncoder`: identifies the encoder channel (zero based). 0 - 3 or 4 for all channels
`pbVal`: in out Boolean parameter that identifies whether a trigger occurs when the edge of index is detected.

Example C Usage:
```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
short iEncoder = 0;     // Note: choose your value here.
BOOL bVal = False;

iResult  = PCI4E_GetTriggerOnIndex(iDeviceNo, iEncoder, &bVal);
if ( iResult != S_OK ){ // Handle error... }
```

VB Language Function Declaration:
```
Public Declare Function PCI4E_GetTriggerOnIndex Lib "USD_PCI_4E.dll" (ByVal
iDeviceNo As Integer, ByVal iEncoder As Integer, ByRef lVal As Long) As Long
```

Example VB Usage:
```
Dim errCode as Long
Dim iDevice as Integer
Dim iEncoder as Integer
Dim lVal as Long

iDevice = 0       ‘ Note: choose your value here.
iEncoder = 0      ‘ Note: choose your value here.

errCode = PCI4E_GetTriggerOnIndex(iDeviceNo, iEncoder, lVal);
If errCode <> 0 then
      ‘ Handle error…
End If
```

9.4.24 PCI4E_GetTriggerOnMatch

Description:
This function retrieves a Boolean value that indicates whether a trigger will be set when count = match.

C Language Function Prototype:
```
int _stdcall PCI4E_GetTriggerOnMatch(short iDeviceNo, short iEncoder, BOOL
*pbVal);
```

Returns:
Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:
`iDeviceNo`: identifies the PCI-4E card (zero based).
`iEncoder`: identifies the encoder channel (zero based). 0 - 3 or 4 for all channels
`pbVal`: in out Boolean parameter. See description above.

Example C Usage:
```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
short iEncoder = 0;     // Note: choose your value here.
BOOL bVal = FALSE;

iResult  = PCI4E_GetTriggerOnMatch(iDeviceNo, iEncoder, &bVal);
if ( iResult != S_OK ){ // Handle error... }
```

VB Language Function Declaration:
```
Public Declare Function PCI4E_GetTriggerOnMatch Lib "USD_PCI_4E.dll" (ByVal
iDeviceNo As Integer, ByVal iEncoder As Integer, ByRef lVal As Long) As Long
```

Example VB Usage:
```
Dim errCode as Long
Dim iDevice as Integer
Dim iEncoder as Integer
Dim lVal as Long

iDevice = 0       ‘ Note: choose your value here.
iEncoder = 0      ‘ Note: choose your value here.

errCode = PCI4E_GetTriggerOnMatch(iDeviceNo, iEncoder, lVal);
If errCode <> 0 then
      ‘ Handle error…
End If
```

9.4.25 PCI4E_GetTriggerOnRollover

Description:
This function retrieves a Boolean value that indicates whether a trigger will be set when rolling over N-1 to 0 in modulo-N mode.

C Language Function Prototype:
```
int _stdcall PCI4E_GetTriggerOnRollover(short iDeviceNo, short iEncoder, BOOL
*pbVal);
```

Returns:
Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:
`iDeviceNo`: identifies the PCI-4E card (zero based).
`iEncoder`: identifies the encoder channel (zero based). 0 - 3 or 4 for all channels
`pbVal`: in out parameters. See description above.

Example C Usage:
```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
short iEncoder = 0;     // Note: choose your value here.
BOOL bVal = False;

iResult  = PCI4E_GetTriggerOnRollover(iDeviceNo, iEncoder, &bVal);
if ( iResult != S_OK ){ // Handle error... }
```

VB Language Function Declaration:
```
Public Declare Function PCI4E_GetTriggerOnRollover Lib "USD_PCI_4E.dll" (ByVal
iDeviceNo As Integer, ByVal iEncoder As Integer, ByRef lVal As Long) As Long
```

Example VB Usage:
```
Dim errCode as Long
Dim iDevice as Integer
Dim iEncoder as Integer
Dim lVal as Long

iDevice = 0       ‘ Note: choose your value here.
iEncoder = 0      ‘ Note: choose your value here.

errCode = PCI4E_GetTriggerOnRollover(iDeviceNo, iEncoder, lVal);
If errCode <> 0 then
      ‘ Handle error…
End If
```

9.4.26 PCI4E_GetTriggerOnRollunder

Description:
This function retrieves a Boolean value that indicates whether a trigger will be set when rolling under 0 to N-1 in modulo-N mode.

C Language Function Prototype:
```
int _stdcall PCI4E_GetTriggerOnRollunder(short iDeviceNo, short iEncoder, BOOL
*pbVal);
```

Returns:
Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:
`iDeviceNo`: identifies the PCI-4E card (zero based).
`iEncoder`: identifies the encoder channel (zero based). 0 - 3 or 4 for all channels
`pbVal`: in-out parameters. See description above.

Example C Usage:
```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
short iEncoder = 0;     // Note: choose your value here.
BOOL bVal = False;

iResult  = PCI4E_GetTriggerOnRollunder(iDeviceNo, iEncoder, & bVal);
if ( iResult != S_OK ){ // Handle error... }
```

VB Language Function Declaration:
```
Public Declare Function PCI4E_GetTriggerOnRollunder Lib "USD_PCI_4E.dll" (ByVal
iDeviceNo As Integer, ByVal iEncoder As Integer, ByRef lVal As Long) As Long
```

Example VB Usage:
```
Dim errCode as Long
Dim iDevice as Integer
Dim iEncoder as Integer
Dim lVal as Long

iDevice = 0       ‘ Note: choose your value here.
iEncoder = 0      ‘ Note: choose your value here.

errCode = PCI4E_GetTriggerOnRollunder(iDeviceNo, iEncoder, lVal);
If errCode <> 0 then
      ‘ Handle error…
End If
```

9.4.27 PCI4E_GetTriggerOnZero

Description:
This function retrieves a Boolean value that indicates whether a trigger will be set when count = 0.

C Language Function Prototype:
```
int _stdcall PCI4E_GetTriggerOnZero(short iDeviceNo, short iEncoder, BOOL
*pbVal);
```

Returns:
Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:
`iDeviceNo`: identifies the PCI-4E card (zero based).
`iEncoder`: identifies the encoder channel (zero based). 0 - 3 or 4 for all channels
`bVal`: in-out parameters. See description above.

Example C Usage:
```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
short iEncoder = 0;     // Note: choose your value here.
BOOL bVal = False;

iResult  = PCI4E_GetTriggerOnZero(iDeviceNo, iEncoder, &bVal);
if ( iResult != S_OK ){ // Handle error... }
```

VB Language Function Declaration:
```
Public Declare Function PCI4E_GetTriggerOnZero Lib "USD_PCI_4E.dll" (ByVal
iDeviceNo As Integer, ByVal iEncoder As Integer, ByRef lVal As Long) As Long
```

Example VB Usage:
```
Dim errCode as Long
Dim iDevice as Integer
Dim iEncoder as Integer
Dim lVal as Long

iDevice = 0       ‘ Note: choose your value here.
iEncoder = 0      ‘ Note: choose your value here.

errCode = PCI4E_GetTriggerOnZero(iDeviceNo, iEncoder, lVal);
If errCode <> 0 then
      ‘ Handle error…
End If
```

9.4.28 PCI4E_GetVersion

Description:
This function retrieves the version number associated with a specified device..

C Language Function Prototype:
```
int _stdcall PCI4E_GetVersion(short iDeviceNo, short *piVersion);
```

Returns:
Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:
`iDeviceNo`: identifies the PCI-4E card (zero based).
`piVersion`: identifies the version number of the PCI-4E card.

Example C Usage:
```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
int iVersion = 0;
iResult = PCI4E_GetVersion(iDeviceNo, &iVersion);
if ( iResult != S_OK ){ // Handle error...}
```

VB Language Function Declaration:
```
Public Declare Function PCI4E_GetVersion Lib "USD_PCI_4E.dll" (ByVal iDeviceNo
As Integer, ByRef lVal As Integer) As Long
```

Example VB Usage:
```
Dim errCode as Long
Dim iDevice as Integer
Dim lVersion as Long

iDevice = 0 ‘ Note: choose your value here.

errCode = PCI4E_GetVersion(iDeviceNo, lVersion)
If errCode <> 0 then
      ‘ Handle error..
End If
```

9.4.29 PCI4E_Initialize

Description:

This function is used to open a connection with all installed and detected PCI-4E data acquisition cards. This function returns the number of cards detected in the in-out parameter pDeviceCount. This function must be called before any other function. Almost all other function calls require a device number. If there are two boards detected, then the first board will be device number 0 and the second device number 1.

C Language Function Prototype:

```
int _stdcall PCI4E_Initialize(short *piDeviceCount);
```

Returns:

Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:

`piDeviceCount`: an in/out parameter used to return the number of boards detected.

Example C Usage:

```
int iDeviceCount = 0;

if( PCI4E_Initialize(&iDeviceCount) != 0 )
{
      // Handle error condition.  See Error Codes for description of possible
      error codes.
};
```

VB Language Function Declaration:

```
Public Declare Function PCI4E_Initialize Lib "USD_PCI_4E.dll" (ByRef
iDeviceCount As Integer) As Long
```

Example VB Usage:

```
Dim errCode as Long
Dim iDeviceCount as Long

errCode = PCI4E_Initialize(iDeviceCount)
If errCode <> 0 then
      ‘ Handle error..
End If
```

9.4.30 PCI4E_PresetCount

Description:
The PresetCount function sets the channel counter to the preset register value.

C Language Function Prototype:
```
int _stdcall PCI4E_PresetCount(short iDeviceNo, short iEncoder);
```

Returns:
Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:
`iDeviceNo`: identifies the PCI-4E card (zero based).
`iEncoder`: identifies the encoder channel (zero based).

Example C Usage:
```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
short iEncoder = 0;     // Note: choose your value here.

iResult = PCI4E_PresetCount(iDeviceNo, iEncoder );
if ( iResult != S_OK ){ // Handle error...}
```

VB Language Function Declaration:
```
Public Declare Function PCI4E_PresetCount Lib "USD_PCI_4E.dll" (ByVal iDeviceNo
As Integer, ByVal iEncoder As Integer) As Long
```

Example VB Usage:
```
Dim errCode as Long
Dim iDevice as Integer
Dim iEncoder as Integer

iDevice = 0       ‘ Note: choose your value here.
iEncoder = 0      ‘ Note: choose your value here.

errCode = PCI4E_PresetCount(iDeviceNo, iEncoder);
If errCode <> 0 then
      ‘ Handle error..
End If
```

9.4.31 PCI4E_ReadOutputLatch

Description:

This function returns the value last value transferred from the accumulator (counter) register to the Output Latch Register. This function is similar to GetCount except that it does not transfer the current accumulator value to the Output Latch before returning the Output Latch value.

C Language Function Prototype:

```
int _stdcall PCI4E_ReadOutputLatch(short iDeviceNo, short iEncoder, long
*plVal);
```

Returns:

Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:

`iDeviceNo`: identifies the PCI-4E card (zero based).

`iEncoder`: identifies the encoder channel (zero based).

`plVal`: in/out parameter that will receive the encoder's last transferred count value.

Example C Usage:

```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
short iEncoder = 0;     // Note: choose your value here.
long lCount = 0;

iResult = PCI4E_ReadOutputLatch(iDeviceNo, iEncoder, &lCount);
if ( iResult != S_OK ){ // Handle error...}
```

VB Language Function Declaration:

```
Public Declare Function PCI4E_ReadOutputLatch Lib "USD_PCI_4E.dll" (ByVal
iDeviceNo As Integer, ByVal iEncoder As Integer, ByRef lVal) As Long
```

Example VB Usage:

```
Dim errCode as Long
Dim lCount as Long
Dim iDevice as Integer
Dim iEncoder as Integer

iDevice = 0       ' Note: choose your value here.
iEncoder = 0      ' Note: choose your value here.

errCode = PCI4E_ReadOutputLatch(iDeviceNo, iEncoder, lCount)
If errCode <> 0 then
      ' Handle error..
End If
```

9.4.32 PCI4E_ReadRegister

Description:
This function returns the value stored in a specific PCI-4E register.

C Language Function Prototype:
```
int _stdcall PCI4E_ReadRegister(short iDeviceNo, short iRegister, long *plVal);
```

Returns:
Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:
`iDeviceNo`: identifies the PCI-4E card (zero based).
`iRegister`: identifies the specific register to read. Valid registers are 0 - 31.
`plVal`: in-out parameter containing value read from the specified register.

Example C Usage:
```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
short iRegister = 0;    // Note: choose your value here.
long lVal = 0;

iResult = PCI4E_ReadRegister(iDeviceNo,  iRegister, &lVal);
if( iResult != S_OK ){  // Handle error...}
```

VB Language Function Declaration:
```
Public Declare Function PCI4E_ReadRegister Lib "USD_PCI_4E.dll" (ByVal
iDeviceNo As Integer, ByVal iRegister As Integer, ByRef lVal As Long) As Long
```

Example VB Usage:
```
Dim errCode as Long
Dim iDevice as Integer
Dim iRegister as Integer
Dim lVal as Long

iDevice = 0       'Note: choose your value here.
iRegister = 0     'Note: choose your value here.

errCode =  PCI4E_ReadRegister( iDevice, iRegister, lVal)
If errCode <> 0 Then
      ‘ Handle error...
End If
```

9.4.33 PCI4E_ReadTimeAndCounts

Description:
This function reads the TimeStamp Latch and each encoder's Output Latch.

C Language Function Prototype:
```
int _stdcall PCI4E_ReadTimeAndCounts(short iDeviceNo, long *plValues,  long
*plTimeStamp);
```

Returns:
Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:
`iDeviceNo`: identifies the PCI-4E card (zero based).
`plValues`: array of 4 longs containing the Output Latch value for each channel.
`plTimeStamp`: in-out parameter containing the TimeStamp Latch value.

Example C Usage:
```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
long lValues[4] = {0};
long lTimeStamp = 0;

iResult = PCI4E_ReadTimeAndCounts(iDeviceNo, &lValues, & lTimeStamp);
if ( iResult != S_OK ){ // Handle error...}
```

VB Language Function Declaration:
```
Public Declare Function PCI4E_ReadTimeAndCounts Lib "USD_PCI_4E.dll" (ByVal
iDeviceNo As Integer, ByRef lValues As Long, ByRef lTimeStamp As Long) As Long
```

Example VB Usage:
```
Dim errCode as Long
Dim iDevice as Integer
Dim lTimeStamp as Long
Dim lValues(3) as Long

iDevice = 0 ‘ Note: choose your value here.

errCode = PCI4E_ReadTimeAndCounts(iDeviceNo, lValue(0), lTimeStamp);
If errCode <> 0 then
      ‘ Handle error…
End If
```

9.4.34 PCI4E_ReadTimeStamp

Description:
This function reads the TimeStamp Latch.

C Language Function Prototype:
```
int _stdcall PCI4E_ReadTimeStamp(short iDeviceNo, long *plTimeStamp);
```

Returns:
Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:
`iDeviceNo`: identifies the PCI-4E card (zero based).
`plTimeStamp`: in-out parameter containing the TimeStamp Latch value.

Example C Usage:
```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
long lTimeStamp = 0;

iResult = PCI4E_ReadTimeStamp(iDeviceNo, &lTimeStamp);
if ( iResult != S_OK ){ // Handle error...}
```

VB Language Function Declaration:
```
Public Declare Function PCI4E_ReadTimeStamp Lib "USD_PCI_4E.dll" (ByVal
iDeviceNo As Integer, ByRef lTimeStamp As Long) As Long
```

Example VB Usage:
```
Dim errCode as Long
Dim iDevice as Integer
Dim lTimeStamp as Long

iDevice = 0 ‘ Note: choose your value here.

errCode = PCI4E_ReadTimeStamp(iDeviceNo, lTimeStamp);
If errCode <> 0 then
      ‘ Handle error…
End If
```

9.4.35 PCI4E_ResetCount

Description:
This function sets the counter value to zero.

C Language Function Prototype:
```
int _stdcall PCI4E_ResetCount(short iDeviceNo, short iEncoder);
```

Returns:
Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:
`iDeviceNo`: identifies the PCI-4E card (zero based).
`iEncoder`: identifies the encoder channel (zero based).

Example C Usage:
```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
short iEncoder = 0;     // Note: choose your value here.

iResult = PCI4E_ResetCount(iDeviceNo, iEncoder);
if ( iResult != S_OK ){ // Handle error...}
```

VB Language Function Declaration:
```
Public Declare Function PCI4E_ResetCount Lib "USD_PCI_4E.dll" (ByVal iDeviceNo
As Integer, ByVal iEncoder As Integer) As Long
```

Example VB Usage:
```
Dim errCode as Long
Dim iDevice as Integer
Dim iEncoder as Integer

iDevice = 0       ‘ Note: choose your value here.
iEncoder = 0      ‘ Note: choose your value here.

errCode = PCI4E_ResetCount(iDeviceNo, iEncoder);
If errCode <> 0 then
      ‘ Handle error…
End If
```

9.4.36 PCI4E_ResetTimeStamp

Description:
This function sets the TimeStamp counter value to zero.

C Language Function Prototype:
```
int _stdcall PCI4E_ResetTimeStamp(short iDeviceNo);
```

Returns:
Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:
`iDeviceNo`: identifies the PCI-4E card (zero based).

Example C Usage:
```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.

iResult = PCI4E_ResetTimeStamp(iDeviceNo);
if ( iResult != S_OK ){ // Handle error...}
```

VB Language Function Declaration:
```
Public Declare Function PCI4E_ResetTimeStampLib "USD_PCI_4E.dll" (ByVal
iDeviceNo) As Long
```

Example VB Usage:
```
Dim errCode as Long
Dim iDevice as Integer

iDevice = 0 ‘ Note: choose your value here.

errCode = PCI4E_ResetTimeStamp(iDeviceNo);
If errCode <> 0 then
      ‘ Handle error…
End If
```

9.4.37 PCI4E_SetCaptureEnabled

Description:
This function sets a Boolean value that determines whether a trigger_in will cause a transfer from counter (accumulator) to output.

C Language Function Prototype:
```
int _stdcall PCI4E_SetCaptureEnabled(short iDeviceNo, short iEncoder, BOOL
bVal);
```

Returns:
Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:
`iDeviceNo`: identifies the PCI-4E card (zero based).
`iEncoder`: identifies the encoder channel (zero based).
`bVal`: indicates whether a trigger_in will cause a transfer from accumulator to output.

Example C Usage:
```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
short iEncoder = 0;     // Note: choose your value here.
BOOL  bVal = TRUE;

iResult = PCI4E_SetCaptureEnabled(iDeviceNo, iEncoder, &bVal);
if ( iResult != S_OK ){ // Handle error...}
```

VB Language Function Declaration:
```
Public Declare Function PCI4E_SetCaptureEnabled Lib "USD_PCI_4E.dll" (ByVal
iDeviceNo As Integer, ByVal iEncoder As Integer, ByVal bVal As Boolean) As Long
```

Example VB Usage:
```
Dim errCode as Long
Dim iDevice as Integer
Dim iEncoder as Integer
Dim bVal as Boolean

iDevice = 0       ‘ Note: choose your value here.
iEncoder = 0      ‘ Note: choose your value here.
lVal = 0          ‘ Note: choose your value here.

errCode = PCI4E_SetCaptureEnabled(iDevice, iEncoder, lVal)
if errCode <> 0 then
      ‘ Handle error…
End If
```

9.4.38 PCI4E_SetControlMode

Description:
This function sets the control register value which controls the operation of a channel.

C Language Function Prototype:
```
int _stdcall PCI4E_SetControlMode(short iDeviceNo, short iEncoder, long lMode);
```

Returns:
Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:
`iDeviceNo`: identifies the PCI-4E card (zero based).
`iEncoder`: identifies the encoder channel (zero based).
`lMode`: 32 bit parameter containing the control mode. *See section 6.1 Control Registers*

Example C Usage:
```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
short iEncoder = 0;     // Note: choose your value here.
long lVal = 0xF4000;    // Obtain the correct control mode.

iResult = PCI4E_SetControlMode (iDeviceNo, iEncoder, &lVal);
if ( iResult != S_OK ){ // Handle error...}
```

VB Language Function Declaration:
```
Public Declare Function PCI4E_SetControlMode Lib "USD_PCI_4E.dll" (ByVal
iDeviceNo As Integer, ByVal iEncoder As Integer, ByVal lVal As Long) As Long
```

Example VB Usage:
```
Dim errCode as Long
Dim iDevice as Integer
Dim iEncoder as Integer
Dim lVal as Long

iDevice = 0       ‘ Note: choose your value here.
iEncoder = 0      ‘ Note: choose your value here.
lVal = &hF7000    ‘ Note: choose your value here.

errCode = PCI4E_SetControlMode(iDevice, iEncoder, lVal)
if errCode <> 0 then
      ‘ Handle error…
End If
```

9.4.39 PCI4E_SetCount

Description:
This function sets the count to a specified value.

Caveats: PCI4E_SetCount forces internal counter's value to a specified value without permanently changing the Preset register. In fact, PCI4E_SetCount utilizes Preset register for transferring data to the internal counter, but the original value of Preset register is restored at the end of function call. When writing an application that always watches for changing of value of Preset register, the programmer must be aware of this temporary change of value.

C Language Function Prototype:
```
int _stdcall PCI4E_SetCount(short iDeviceNo, short iEncoder, long lVal);
```

Returns:
Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:
`iDeviceNo`: identifies the PCI-4E card (zero based).
`iEncoder`: identifies the encoder channel (zero based).
`lVal`: the new value to be written to the counter register.

Example C Usage:
```
int iResult = S_OK;
short iDeviceNo = 0;       // Note: choose your value here.
short iEncoder = 0;        // Note: choose your value here.
long lCount= 0;            //  Note: choose your value here.

iResult = SetCount(iDeviceNo, iEncoder, lCount);
if ( iResult != S_OK ){ // Handle error...}
```

VB Language Function Declaration:
```
Public Declare Function PCI4E_SetCount Lib "USD_PCI_4E.dll" (ByVal iDeviceNo As Integer,
ByVal iEncoder As Integer, ByVal lVal As Long) As Long
```

Example VB Usage:
```
Dim errCode as Long
Dim iDevice as Integer
Dim iEncoder as Integer
Dim lVal as Long

iDevice = 0          ‘ Note: choose your value here.
iEncoder = 0         ‘ Note: choose your value here.
lVal = 0             ‘ Note: choose your value here.

errCode =  PCI4E_SetCount(iDeviceNo, iEncoder, lVal);
If errCode <> 0 then
       ‘ Handle error..
End If
```

9.4.40 PCI4E_SetCounterMode

Description:
This function sets a control bit that governs the counter behavior and limits. See parameters sections for description of the possible counter modes.

C Language Function Prototype:
```
int _stdcall PCI4E_SetCounterMode(short iDeviceNo, short iEncoder, short
iMode);
```

Returns:
Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:
`iDeviceNo`: identifies the PCI-4E card (zero based).
`iEncoder`: identifies the encoder channel (zero based).
`iMode`: 32 bit in-out parameter containing the counter mode.
0 = acc. acts like a 24 bit counter
1 = acc. uses preset register in range-limit mode
2 = acc. uses preset register in non-recycle mode
3 = acc. uses preset register in modulo-N mode.
See 6.1 Control Registers

Example C Usage:
```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
short iEncoder = 0;     // Note: choose your value here.
short lMode = 0;

iResult  = PCI4E_SetCounterMode(iDeviceNo, iEncoder, iMode);
if ( iResult != S_OK ){ // Handle error... }
```

VB Language Function Declaration:
```
Public Declare Function PCI4E_SetCounterMode Lib "USD_PCI_4E.dll" (ByVal
iDeviceNo As Integer, ByVal iEncoder As Integer, ByVal iMode As Integer) As
Long
```

Example VB Usage:
```
Dim errCode as Long
Dim iDevice as Integer
Dim iEncoder as Integer
Dim iMode as Long

iDevice = 0       ‘ Note: choose your value here.
iEncoder = 0      ‘ Note: choose your value here.

errCode = PCI4E_SetCounterMode(iDeviceNo, iEncoder, iMode)
If errCode <> 0 then
      ‘ Handle error..
End If
```

9.4.41 PCI4E_SetEnableAccumulator

Description:
This function sets a Boolean value that determines whether the master enable for accumulator is set, (must be set to true to count).

C Language Function Prototype:
```
int _stdcall PCI4E_SetEnableAccumulator(short iDeviceNo, short iEncoder, BOOL
bVal);
```

Returns:
Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:
`iDeviceNo`: identifies the PCI-4E card (zero based).
`iEncoder`: identifies the encoder channel (zero based).
`bVal`: in-out Boolean parameter identifying whether the counter is enabled.

Example C Usage:
```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
short iEncoder = 0;     // Note: choose your value here.
BOOL bVal = FALSE;      // Note: choose your value here.

iResult  = PCI4E_SetEnableAccumulator(iDeviceNo, iEncoder, bVal);
if ( iResult != S_OK ){ // Handle error... }
```

VB Language Function Declaration:
```
Public Declare Function PCI4E_SetEnableAccumulator Lib "USD_PCI_4E.dll" (ByVal
iDeviceNo As Integer, ByVal iEncoder As Integer, ByVal bVal As Boolean) As Long
```

Example VB Usage:
```
Dim errCode as Long
Dim iDevice as Integer
Dim iEncoder as Integer
Dim bVal as Boolean

iDevice = 0       ‘ Note: choose your value here.
iEncoder = 0      ‘ Note: choose your value here.
bVal = True       ‘ Note: choose your value here.

errCode = PCI4E_SetEnableAccumulator(iDeviceNo, iEncoder, bVal)
If errCode <> 0 then
      ‘ Handle error..
End If
```

9.4.42 PCI4E_SetEnableIndex

Description:
This function sets a Boolean value that determines, when set, whether the index will either reset or preset accumulator.

C Language Function Prototype:
```
int _stdcall PCI4E_SetEnableIndex(short iDeviceNo, short iEncoder, BOOL bVal);
```

Returns:
Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:
`iDeviceNo`: identifies the PCI-4E card (zero based).
`iEncoder`: identifies the encoder channel (zero based).
`bVal`: in-out Boolean parameter identifying whether the index is enabled.

Example C Usage:
```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
short iEncoder = 0;     // Note: choose your value here.
BOOL bVal = FALSE;

iResult  = PCI4E_SetEnableIndex(iDeviceNo, iEncoder, &bVal);
if ( iResult != S_OK ){ // Handle error... }
```

VB Language Function Declaration:
```
Public Declare Function PCI4E_SetEnableIndex Lib "USD_PCI_4E.dll" (ByVal
iDeviceNo As Integer, ByVal iEncoder As Integer, ByVal bVal As Boolean) As Long
```

Example VB Usage:
```
Dim errCode as Long
Dim iDevice as Integer
Dim iEncoder as Integer
Dim bVal as Boolean

iDevice = 0       ‘ Note: choose your value here.
iEncoder = 0      ‘ Note: choose your value here.
bVal = True       ‘ Note: choose your value here.

errCode = PCI4E_SetEnableIndex(iDeviceNo, iEncoder, bVal)
If errCode <> 0 then
      ‘ Handle error..
End If
```

9.4.43 PCI4E_SetForward

Description:
This function takes a Boolean value that determines whether a and b quadratures are swapped. Note that PCI4E_SetForward function sets bit 19 of Control register to '1' when its parameter, bVal, is '1'.

C Language Function Prototype:
```
int _stdcall PCI4E_SetForward(short iDeviceNo, short iEncoder, BOOL bVal);
```

Returns:
Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:
`iDeviceNo`: identifies the PCI-4E card (zero based).
`iEncoder`: identifies the encoder channel (zero based).
`bVal`: in-out Boolean parameter identifying the counting direction.

Example C Usage:
```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
short iEncoder = 0;     // Note: choose your value here.
BOOL bVal = TRUE;       // Note: choose your value here.

iResult  = PCI4E_SetForward(iDeviceNo, iEncoder, bVal);
if ( iResult != S_OK ){ // Handle error... }
```

VB Language Function Declaration:
```
Public Declare Function PCI4E_SetForward Lib "USD_PCI_4E.dll" (ByVal iDeviceNo
As Integer, ByVal iEncoder As Integer, ByVal bVal As Boolean) As Long
```

Example VB Usage:
```
Dim errCode as Long
Dim iDevice as Integer
Dim iEncoder as Integer
Dim bVal as Boolean

iDevice = 0       ' Note: choose your value here.
iEncoder = 0      ' Note: choose your value here.
bVal = True       ' Note: choose your value here.

errCode = PCI4E_SetForward(iDeviceNo, iEncoder, bVal)
If errCode <> 0 then
      ' Handle error..
End If
```

9.4.44 PCI4E_SetInvertIndex

Description:

This function takes a Boolean value that determines the active level of the index.
`bVal = TRUE` when the index is active low. Default is active high.

C Language Function Prototype:

```
int _stdcall PCI4E_SetInvertIndex(short iDeviceNo, short iEncoder, BOOL bVal);
```

Returns:

Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:

`iDeviceNo`: identifies the PCI-4E card (zero based).
`iEncoder`: identifies the encoder channel (zero based).
`bVal`: in-out Boolean parameter. See description above.

Example C Usage:

```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
short iEncoder = 0;     // Note: choose your value here.
BOOL bVal = FALSE;

iResult  = PCI4E_SetInvertIndex(iDeviceNo, iEncoder, bVal);
if ( iResult != S_OK ){ // Handle error... }
```

VB Language Function Declaration:

```
Public Declare Function PCI4E_SetInvertIndex Lib "USD_PCI_4E.dll" (ByVal
iDeviceNo As Integer, ByVal iEncoder As Integer, ByVal bVal As Boolean) As Long
```

Example VB Usage:

```
Dim errCode as Long
Dim iDevice as Integer
Dim iEncoder as Integer
Dim bVal as Boolean

iDevice = 0       ‘ Note: choose your value here.
iEncoder = 0      ‘ Note: choose your value here.
bVal = False      ‘ Note: choose your value here.

errCode = PCI4E_SetInvertIndex(iDeviceNo, iEncoder, bVal)
If errCode <> 0 then
      ‘ Handle error..
End If
```

9.4.45 PCI4E_SetMatch

Description:
This function sets the match register value. It is used as a reference to signal a capture when the counter equals the match register value.

C Language Function Prototype:
```
int _stdcall PCI4E_SetMatch(short iDeviceNo, short iEncoder, long lVal);
```

Returns:
Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:
`iDeviceNo`: identifies the PCI-4E card (zero based).
`iEncoder`: identifies the encoder channel (zero based).
`lVal`: in-out parameter containing the match register value.

Example C Usage:
```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
short iEncoder = 0;     // Note: choose your value here.
long lMatchVal = 500;   //  Note: choose your value here.

iResult  = PCI4E_SetMatch(iDeviceNo, iEncoder, & lMatchVal);
if ( iResult != S_OK ){ // Handle error... }
```

VB Language Function Declaration:
```
Public Declare Function PCI4E_SetMatch Lib "USD_PCI_4E.dll" (ByVal iDeviceNo As
Integer, ByVal iEncoder As Integer, ByVal lMatchVal As Long) As Long
```

Example VB Usage:
```
Dim errCode as Long
Dim iDevice as Integer
Dim iEncoder as Integer
Dim lVal as Long

iDevice = 0       ‘ Note: choose your value here.
iEncoder = 0      ‘ Note: choose your value here.
lVal = 499        ‘ Note: choose your value here.

errCode = PCI4E_SetMatch(iDeviceNo, iEncoder, lVal)
If errCode <> 0 then
      ‘ Handle error..
End If
```

9.4.46 PCI4E_SetMultiplier

Description:
This function takes a control bit that determines when the counter is to be incremented based on the number of quadrature state transitions. See possible values in parameters description.

C Language Function Prototype:
```
int _stdcall PCI4E_SetMultiplier(short iDeviceNo, short iEncoder, short iMode);
```

Returns:
Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:
`iDeviceNo`: identifies the PCI-4E card (zero based).
`iEncoder`: identifies the encoder channel (zero based). 0 - 3 or 4 for all channels
`iMode`: identifies when the count increments.
Possible values are: 0 = in_a is clock, in_b is direction
1 = count increments once every four quad states, X1
2 = count increments once every two quad states, X2
3 = count increments once every quad state, X4

Example C Usage:
```
int iResult = S_OK;
short iDeviceNo = 0;     // Note: choose your value here.
short iEncoder = 0;      // Note: choose your value here.
short iMode = 1;         // Note: choose your value here.

iResult = PCI4E_SetMultiplier(iDeviceNo, iEncoder, iMode);
if ( iResult != S_OK ){ // Handle error...}
```

VB Language Function Declaration:
```
Public Declare Function PCI4E_SetMultiplier Lib "USD_PCI_4E.dll" (ByVal
iDeviceNo As Integer, ByVal iEncoder As Integer, ByVal iMode As Integer) As
Long
```

Example VB Usage:
```
Dim errCode as Long
Dim iDevice as Integer
Dim iEncoder as Integer
Dim iMode as Integer

iDevice = 0       ‘ Note: choose your value here.
iEncoder = 0      ‘ Note: choose your value here.
iMode = 1         ‘ Note: choose your value here.

errCode = PCI4E_SetMultiplier(iDeviceNo, iEncoder, iMode);
If errCode <> 0 then
      ‘ Handle error..
End If
```

9.4.47 PCI4E_SetPresetOnIndex

Description:
This function takes a Boolean value that determines whether a preset occurs when the index is enabled.

C Language Function Prototype:
```
int _stdcall PCI4E_SetPresetOnIndex(short iDeviceNo, short iEncoder, BOOL
bVal);
```

Returns:
Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:
`iDeviceNo`: identifies the PCI-4E card (zero based).
`iEncoder`: identifies the encoder channel (zero based).
`bVal`: in/out Boolean parameter. See description above.

Example C Usage:
```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
short iEncoder = 0;     // Note: choose your value here.
BOOL bVal = False;

iResult = PCI4E_SetPresetOnIndex(iDeviceNo, iEncoder, bVal);
if ( iResult != S_OK ){ // Handle error...}
```

VB Language Function Declaration:
```
Public Declare Function PCI4E_SetPresetOnIndex Lib "USD_PCI_4E.dll" (ByVal
iDeviceNo As Integer, ByVal iEncoder As Integer, ByVal bVal As Boolean) As Long
```

Example VB Usage:
```
Dim errCode as Long
Dim iDevice as Integer
Dim iEncoder as Integer
Dim bVal as Boolean

iDevice = 0       ‘ Note: choose your value here.
iEncoder = 0      ‘ Note: choose your value here.
bVal = True       ‘ Note: choose your value here.

errCode = PCI4E_SetPresetOnIndex(iDeviceNo, iEncoder, bVal)
If errCode <> 0 then
      ‘ Handle error..
End If
```

9.4.48 PCI4E_SetPresetValue

Description:
This function sets the Preset Register with a new value.

C Language Function Prototype:
```
int _stdcall PCI4E_SetPresetValue(short iDeviceNo, short iEncoder, long lVal);
```

Returns:
Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:
`iDeviceNo`: identifies the PCI-4E card (zero based).
`iEncoder`: identifies the encoder channel (zero based).
`lVal`: the new preset register value may also be referred to as upper-limit, range-limit, max count, or resolution -1.

Example C Usage:
```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
short iEncoder = 0;     // Note: choose your value here.
long lVal = 499;        //  Note: choose your value here.

iResult = PCI4E_SetPresetValue(iDeviceNo, iEncoder, lVal);
if ( iResult != S_OK ){ // Handle error...}
```

VB Language Function Declaration:
```
Public Declare Function PCI4E_SetPresetValue Lib "USD_PCI_4E.dll" (ByVal
iDeviceNo As Integer, ByVal iEncoder As Integer, ByVal lVal As Long) As Long
```

Example VB Usage:
```
Dim errCode as Long
Dim iDevice as Integer
Dim iEncoder as Integer
Dim lPresetVal as Long

iDevice = 0       ‘ Note: choose your value here.
iEncoder = 0      ‘ Note: choose your value here.
lPresetVal = 499  ‘ Note: choose your value here.

errCode = PCI4E_SetPresetValue(iDeviceNo, iEncoder, lPresetVal);
If errCode <> 0 then
      ‘ Handle error..
End If
```

9.4.49 PCI4E_SetTriggerOnDecrease

Description:
This function takes a Boolean value that determines whether a trigger occurs when the count decreases.

C Language Function Prototype:
```
int _stdcall PCI4E_SetTriggerOnDecrease(short iDeviceNo, short iEncoder, BOOL
bVal);
```

Returns:
Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:
`iDeviceNo`: identifies the PCI-4E card (zero based).
`iEncoder`: identifies the encoder channel (zero based). 0 - 3 or 4 for all channels
`bVal`: in out parameters. See description above.

Example C Usage:
```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
short iEncoder = 0;     // Note: choose your value here.
BOOL bVal = FALSE;      // Note: choose your value here.

iResult  = PCI4E_SetTriggerOnDecrease(iDeviceNo, iEncoder, bVal);
if ( iResult != S_OK ){ // Handle error... }
```

VB Language Function Declaration:
```
Public Declare Function PCI4E_SetTriggerOnDecrease Lib "USD_PCI_4E.dll" (ByVal
iDeviceNo As Integer, ByVal iEncoder As Integer, ByVal bVal As Boolean) As Long
```

Example VB Usage:
```
Dim errCode as Long
Dim iDevice as Integer
Dim iEncoder as Integer
Dim  bVal as Boolean

iDevice = 0       ‘ Note: choose your value here.
iEncoder = 0      ‘ Note: choose your value here.
bVal = False      ‘ Note: choose your value here.

errCode = PCI4E_SetTriggerOnDecrease(iDeviceNo, iEncoder, bVal);
If errCode <> 0 then
      ‘ Handle error…
End If
```

9.4.50 PCI4E_SetTriggerOnIncrease

Description:
This function takes a Boolean value that determines whether a trigger will be set when a count increases.

C Language Function Prototype:
```
int _stdcall PCI4E_SetTriggerOnIncrease(short iDeviceNo, short iEncoder, BOOL
bVal);
```

Returns:
Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:
`iDeviceNo`: identifies the PCI-4E card (zero based).
`iEncoder`: identifies the encoder channel (zero based). 0 - 3 or 4 for all channels
`bVal`: in out Boolean parameter. See description above.

Example C Usage:
```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
short iEncoder = 0;     // Note: choose your value here.
BOOL bVal = False;

iResult  = PCI4E_SetTriggerOnIncrease(iDeviceNo, iEncoder, bVal);
if ( iResult != S_OK ){ // Handle error... }
```

VB Language Function Declaration:
```
Public Declare Function PCI4E_SetTriggerOnIncrease Lib "USD_PCI_4E.dll" (ByVal
iDeviceNo As Integer, ByVal iEncoder As Integer, ByVal bVal As Boolean) As Long
```

Example VB Usage:
```
Dim errCode as Long
Dim iDevice as Integer
Dim iEncoder as Integer
Dim bVal as Boolean

iDevice = 0       ‘ Note: choose your value here.
iEncoder = 0      ‘ Note: choose your value here.
bVal = False      ‘ Note: choose your value here.

errCode = PCI4E_SetTriggerOnIncrease(iDeviceNo, iEncoder, bVal);
If errCode <> 0 then
      ‘ Handle error…
End If
```

9.4.51 PCI4E_SetTriggerOnIndex

Description:
This function takes a Boolean value that determines whether a trigger occurs when the edge of index is detected.

C Language Function Prototype:
```
int _stdcall PCI4E_SetTriggerOnIndex(short iDeviceNo, short iEncoder, BOOL
bVal);
```

Returns:
Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:
`iDeviceNo`: identifies the PCI-4E card (zero based).
`iEncoder`: identifies the encoder channel (zero based). 0 - 3 or 4 for all channels
`bVal`: in out Boolean parameter that determines whether a trigger occurs when the edge of index is detected.

Example C Usage:
```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
short iEncoder = 0;     // Note: choose your value here.
BOOL bVal = False;

iResult  = PCI4E_SetTriggerOnIndex(iDeviceNo, iEncoder, bVal);
if ( iResult != S_OK ){ // Handle error... }
```

VB Language Function Declaration:
```
Public Declare Function PCI4E_SetTriggerOnIndex Lib "USD_PCI_4E.dll" (ByVal
iDeviceNo As Integer, ByVal iEncoder As Integer, ByVal bVal As Boolean) As Long
```

Example VB Usage:
```
Dim errCode as Long
Dim iDevice as Integer
Dim iEncoder as Integer
Dim bVal as Boolean

iDevice = 0       ‘ Note: choose your value here.
iEncoder = 0      ‘ Note: choose your value here.
bVal = True       ‘ Note: choose your value here.

errCode = PCI4E_SetTriggerOnIndex(iDeviceNo, iEncoder, bVal);
If errCode <> 0 then
      ‘ Handle error…
End If
```

9.4.52 PCI4E_SetTriggerOnMatch

Description:
This function takes a Boolean value that determines whether a trigger will be set when count = match.

C Language Function Prototype:
```
int _stdcall PCI4E_SetTriggerOnMatch(short iDeviceNo, short iEncoder, BOOL
bVal);
```

Returns:
Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:
`iDeviceNo`: identifies the PCI-4E card (zero based).
`iEncoder`: identifies the encoder channel (zero based). 0 - 3 or 4 for all channels
`bVal`: in out Boolean parameter. See description above.

Example C Usage:
```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
short iEncoder = 0;     // Note: choose your value here.
BOOL bVal = FALSE;      // Note: choose your value here.

iResult  = PCI4E_SetTriggerOnMatch(iDeviceNo, iEncoder, bVal);
if ( iResult != S_OK ){ // Handle error... }
```

VB Language Function Declaration:
```
Public Declare Function PCI4E_SetTriggerOnMatch Lib "USD_PCI_4E.dll" (ByVal
iDeviceNo As Integer, ByVal iEncoder As Integer, ByVal bVal As Boolean) As Long
```

Example VB Usage:
```
Dim errCode as Long
Dim iDevice as Integer
Dim iEncoder as Integer
Dim bVal as Boolean

iDevice = 0       ‘ Note: choose your value here.
iEncoder = 0      ‘ Note: choose your value here.
bVal = False      ‘ Note: choose your value here.
errCode = PCI4E_SetTriggerOnMatch(iDeviceNo, iEncoder, lVal);
If errCode <> 0 then
      ‘ Handle error…
End If
```

9.4.53 PCI4E_SetTriggerOnRollover

Description:
This function takes a Boolean value that determines whether a trigger will be set when rolling over N-1 to 0 in modulo-N mode.

C Language Function Prototype:
```
int _stdcall PCI4E_SetTriggerOnRollover(short iDeviceNo, short iEncoder, BOOL
bVal);
```

Returns:
Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:
`iDeviceNo`: identifies the PCI-4E card (zero based).
`iEncoder`: identifies the encoder channel (zero based). 0 - 3 or 4 for all channels
`bVal`: in out Boolean parameters. See description above.

Example C Usage:
```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
short iEncoder = 0;     // Note: choose your value here.
BOOL bVal = False;

iResult  = PCI4E_SetTriggerOnRollover(iDeviceNo, iEncoder, bVal);
if ( iResult != S_OK ){ // Handle error... }
```

VB Language Function Declaration:
```
Public Declare Function PCI4E_SetTriggerOnRollover Lib "USD_PCI_4E.dll" (ByVal
iDeviceNo As Integer, ByVal iEncoder As Integer, ByVal bVal As Boolean) As Long
```

Example VB Usage:
```
Dim errCode as Long
Dim iDevice as Integer
Dim iEncoder as Integer
Dim bVal as Boolean

iDevice = 0       ‘ Note: choose your value here.
iEncoder = 0      ‘ Note: choose your value here.
bVal = False      ‘ Note: choose your value here.

errCode = PCI4E_SetTriggerOnRollover(iDeviceNo, iEncoder, bVal);
If errCode <> 0 then
      ‘ Handle error…
End If
```

9.4.54 PCI4E_SetTriggerOnRollunder

Description:
This function takes a Boolean value that determines whether a trigger will be set when rolling under 0 to N-1 in modulo-N mode.

C Language Function Prototype:
```
int _stdcall PCI4E_SetTriggerOnRollunder(short iDeviceNo, short iEncoder, BOOL
bVal);
```

Returns:
Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:
`iDeviceNo`: identifies the PCI-4E card (zero based).
`iEncoder`: identifies the encoder channel (zero based). 0 - 3 or 4 for all channels
`bVal`: in out parameters. See description above.

Example C Usage:
```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
short iEncoder = 0;     // Note: choose your value here.
BOOL bVal = FALSE;      // Note: choose your value here.

iResult  = PCI4E_SetTriggerOnRollunder(iDeviceNo, iEncoder, bVal);
if ( iResult != S_OK ){ // Handle error... }
```

VB Language Function Declaration:
```
Public Declare Function PCI4E_SetTriggerOnRollunder Lib "USD_PCI_4E.dll" (ByVal
iDeviceNo As Integer, ByVal iEncoder As Integer, ByVal bVal As Boolean) As Long
```

Example VB Usage:
```
Dim errCode as Long
Dim iDevice as Integer
Dim iEncoder as Integer
Dim bVal as Boolean

iDevice = 0       ‘ Note: choose your value here.
iEncoder = 0      ‘ Note: choose your value here.
bVal = False      ‘ Note: choose your value here.

errCode = PCI4E_GetTriggerOnRollunder(iDeviceNo, iEncoder, bVal);
If errCode <> 0 then
      ‘ Handle error…
End If
```

9.4.55 PCI4E_SetTriggerOnZero

Description:
This function takes a Boolean value that determines whether a trigger will be set when count = 0.

C Language Function Prototype:
```
int _stdcall PCI4E_SetTriggerOnZero(short iDeviceNo, short iEncoder, BOOL
bVal);
```

Returns:
Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:
`iDeviceNo`: identifies the PCI-4E card (zero based).
`iEncoder`: identifies the encoder channel (zero based). 0 - 3 or 4 for all channels
`bVal`: in out parameter. See description above.

Example C Usage:
```
int iResult = S_OK;
short iDeviceNo = 0;     // Note: choose your value here.
short iEncoder = 0;      // Note: choose your value here.
BOOL bVal = FALSE;       // TODO: Set this parameter to TRUE or FALSE;

iResult  = PCI4E_SetTriggerOnZero(iDeviceNo, iEncoder, bVal);
if ( iResult != S_OK ){ // Handle error... }
```

VB Language Function Declaration:
```
Public Declare Function PCI4E_SetTriggerOnZero Lib "USD_PCI_4E.dll" (ByVal
iDeviceNo As Integer, ByVal iEncoder As Integer, ByVal bVal As Boolean) As Long
```

Example VB Usage:
```
Dim errCode as Long
Dim iDevice as Integer
Dim iEncoder as Integer
Dim bVal as Boolean

iDevice = 0        ‘ Note: choose your value here.
iEncoder = 0       ‘ Note: choose your value here.
bVal = False       ‘ Note: choose your value here.

errCode = PCI4E_SetTriggerOnZero(iDeviceNo, iEncoder, bVal);
If errCode <> 0 then
      ‘ Handle error…
End If
```

9.4.56 PCI4E_Shutdown

Description:
This function must be call in order to disconnect from PCI4E driver.

C Language Function Prototype:
```
void _stdcall PCI4E_Shutdown();
```

Returns:
None

Parameters:
None

Example C Usage:
```
PCI4E_Shutdown();
```

VB Language Function Declaration:
```
Public Declare Sub PCI4E_Shutdown Lib "USD_PCI_4E.dll"
```

Example VB Usage:
```
PCI4E_Shutdown
```

9.4.57 PCI4E_WriteRegister

Description:
This function allows a value to be written into a specific PCI-4E register.

C Language Function Prototype:
```
int _stdcall PCI4E_WriteRegister(short iDeviceNo, short iRegister, long lVal);
```

Returns:
Result code as an integer: See error code section for values other than zero. Zero implies function call is successful.

Parameters:
`iDeviceNo`: identifies the PCI-4E card (zero based).
`iRegister`: identifies the specific register to read. Valid registers are 0 - 31.
`lVal` : the value to be written to the specified register.

Example C Usage:
```
int iResult = S_OK;
short iDeviceNo = 0;    // Note: choose your value here.
short iRegister = 0;    // Note: choose your value here.
long lVal = 0;          // Note: choose your value here.

iResult = PCI4E_WriteRegister(iDeviceNo,  iRegister, lVal);
if( iResult != S_OK ){  // Handle error...}
```

VB Language Function Declaration:
```
Public Declare Function PCI4E_WriteRegister Lib "USD_PCI_4E.dll" (ByVal
iDeviceNo As Integer, ByVal iRegister As Integer, ByVal lVal As Long) As Long
```

Example VB Usage:
```
Dim errCode as Long
Dim iDevice as Integer
Dim iRegister as Integer
Dim lVal as Long

iDevice = 0       ‘ Note: choose your value here.
iRegister = 0     ‘ Note: choose your value here.
lVal = 0          ‘ Note: choose your value here.

errCode =  PCI4E_WriteRegister( iDevice, iRegister, lVal )
If errCode <> 0 Then
      ‘ Handle error...
End If
```

10 Using Java

This section describes a method for calling the C language PCI-4E functions from a Java language application. This example was derived from the following two URLs:
http://java.sun.com/docs/books/tutorial/native1.1/stepbystep/step1.html
http://developer.java.sun.com/developer/onlineTraining/Programming/JDCBook/jniexamp.html

Step 1. Write the Java Code

- Declare native methods that will be used to wrap each C function you wish to call.
- Load the library that implements the functions which wrap the actual C functions to be called.
- Write the main method that calls the native methods.

Note: All parameters that are passed by reference in C must be replaced with a Java reference type parameter such as an object or array.

Note: In this example, the PCI-4E parameters that are passed by reference were made member variables of the class making the C function calls. For example: the single parameter, a short, passed by reference in PCI4E_Intialize function may be declared in your Java class as static member variable. Then add a function to set the value of this member variable.

```
public class pci4e {
    static short mBoardCount;
    static int mCounterValue;

    // Callback functions used by the pci4e_java_wrapper
    // library to set member variables.
    public void BoardCount(short boardCnt) { mBoardCount = boardCnt; }
    public void CountValue(int iValue){ mCountValue = iValue;        }

    // A small subset of the PCI4E native C wrapper functions…
    public native int Initialize();
    public native void Shutdown();
    public native int SetRegister(short iDevice, short iRegister, int iVal);
    public native int SetPresetValue(short iDeviceNo, short iEncoder, int iVal);
    public native int GetCount(short iDevice, short iEncoder);

    // Used to call a win32 api function that check the console for input…
    public native int kbhit();

    static {
        System.loadLibrary("pci4e_java_wrapper");
    }

    public static void main(String[] args) {
        int iResult = 0;       // return code
        short iDevice = 0;     // device number (0-number to cards-1)
        short iEncoder = 0;    // encoder channel number (0-3)
        int iCPR = 500;        // counts per revolution

        // Create class instance...
        pci4e PCI4E_Instance = new pci4e();

        // Initialize pci4e driver...
```

```
        // Note: the BoardCount method should be called.
        iResult = PCI4E_Instance.Initialize();  called.
        System.out.println("Found " + mBoardCount +  " board(s)");


        // Set the counts per revolution…
        iResult = PCI4E_Instance.SetPresetValue(iDevice, iEncoder, iCPR);

        System.out.println("\n*** press any key to quit! ***\n");
        int iPrevVal = 0;
        do
        {
                PCI4E_Instance.GetCount(iDevice, iEncoder);
                if(iPrevVal != mCountValue){
                        System.out.print(mCountValue + "                    \r");
                        iPrevVal = mCountValue;
                }
        }while(PCI4E_Instance.kbhit() == 0 );

        PCI4E_Instance.Shutdown();
        System.out.println("PCI4E Shutdown!");
    }
}
```

Step 2. Compile the Java Code
javac pci4e.java

Step 3. Create the .h File
Javah –jni pci4e
The header file provides a C function signature for the implementation of the native methods defined in the pci4e class.
Here's an example of the Initialize method signature found in the pci4e.h file:

```
* Class:     pci4e
 * Method:    Initialize
 * Signature: ()I
 */
JNIEXPORT jint JNICALL Java_pci4e_Initialize
  (JNIEnv *, jobject);
```

Step 4. Write the Native Method Implemenation
Using the same method signature for each of the methods in the .h file implement the wrapper functions in a pci4eimp.c file.
Here's an example of the implementation for the Initialize method found in the pci4eimp.c file:

```
#include <jni.h>
#include "stdio.h"
#include "windows.h"
#include "conio.h"
#include "PCI_4e.h"

#define TRANSLATION_ERROR -99

/*
```

```
 * Class:     pci4e
 * Method:    Initialize
 * Signature: ()J
 */
JNIEXPORT jshort JNICALL Java_pci4e_Initialize
  (JNIEnv *env, jobject obj)
{
      short iCnt = 0;
      short iRc = 0;

      jclass cls = (*env)->GetObjectClass(env, obj);
      jmethodID mid = (*env)->GetMethodID(env, cls, "BoardCount", "(S)V");
    if (mid == 0) {
        return TRANSLATION_ERROR; // Java to C translation error...
    }

      // Get the board count...
      iRc = PCI4E_Initialize( &iCnt );

      // Call back on the java wrapper and set the board count...
    //printf("In C, Initialize, about to enter Java\n");
    (*env)->CallVoidMethod(env, obj, mid, iCnt);
    //printf("In C, back from Java\n");

      return iRc;
}
```

Step 5. Create a Shared Library
Using Visual Studio 6, create a new Win32 Dynamic-Link Library project.
Add the pci4eimp.c to the project.
From the Projects Settings menu for the file and check no pre-compiled headers.
From the Projects Settings menu for the project enter the output file location and name, ie ..\pci4e_java_wrapper.dll.
Build the library.

Step 6. Run the Program
java pci4e

A simple example Java application that utilizes some of the PCI4E card's functionality is available for download from the **http://usdigital.com/** website.

11 Error Codes

- NO_CARDS_DETECTED -1
- INVALID_DEVICE_NUMBER -2
- DEVICE_NOT_OPEN -3
- INVALID_ENCODER_NUMBER -4
- INVALID_REGISTER_NUMBER -5
- INVALID_SLOT_NUMBER -6
- INVALID_QUADRATURE_MODE -7
- FEATURE_NOT_SUPPORTED -8
- INVALID_COUNTER_MODE -9
- FAILED_TO_OPEN_DRIVER -10
- CONTROL_MODE_IS_ZERO -11
- INCORRECT_WINDRIVER_VERSION -12
- OVERRUN_ERROR_DETECTED -13
- FRAMING_ERROR_DETECTED -14
- RX_BUFFER_FULL -15
- INVALID_PARAMETER -16
- FATAL_ERROR -17