EOS-1300 Series

DI/O Function Library Reference

Manual Rev.: 2.00

Revision Date: Mar. 16, 2017

Part No: 50-1Z225-2000

Advance Technologies; Automate the World.

Revision History

Revision	Release Date	Description of Change(s)
2.00	Mar. 16, 2017	Initial Release

Preface

Disclaimer

The information in this document is subject to change without prior notice in order to improve reliability, design, and function and does not represent a commitment on the part of the manufacturer.

In no event will the manufacturer be liable for direct, indirect, special, incidental, or consequential damages arising out of the use or inability to use the product or documentation, even if advised of the possibility of such damages.

Environmental Responsibility

ADLINK is committed to fulfill its social responsibility to global environmental preservation through compliance with the European Union's Restriction of Hazardous Substances (RoHS) directive and Waste Electrical and Electronic Equipment (WEEE) directive. Environmental protection is a top priority for ADLINK. We have enforced measures to ensure that our products, manufacturing processes, components, and raw materials have as little impact on the environment as possible. When products are at their end of life, our customers are encouraged to dispose of them in accordance with the product disposal and/or recovery programs prescribed by their nation or company.

Trademarks

Product names mentioned herein are used for identification purposes only and may be trademarks and/or registered trademarks of their respective companies.

Conventions

Take note of the following conventions used throughout this manual to make sure that users perform certain tasks and instructions properly.

Additional information, aids, and tips that help users perform tasks.

Information to prevent ***minor*** physical injury, component damage, data loss, and/or program corruption when trying to complete a task.

Information to prevent ***serious*** physical injury, component damage, data loss, and/or program corruption when trying to complete a specific task.

Table of Contents

This page intentionally left blank.

List of Tables

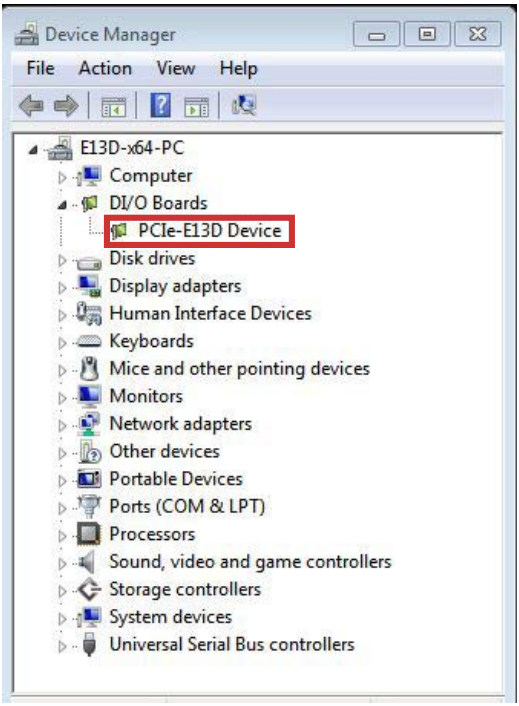
This page intentionally left blank.

List of Figures

This page intentionally left blank.

1 Introduction

This Function Library Reference provides necessary details of APIs for the PCIe-E13D, the I/O device employed by the EOS-1300 4CH GigE Vision system. Once the driver is installed, the PCIe-E13D is listed in in the Device Manager, as shown. The functions can be used to develop normal I/O control and on-the-fly (real-time) trigger and encoder applications in Visual C++, C#, Visual Basic.Net, and Borland C++ Builder.



1.1 Features and General Description

The PCIe-E13D provides the EOS-1300 with 12 DI, 16 DO, and 2 encoders with some I/O pins configurable for real-time trigger in/out. The encoder latch and DI also support digital de-bounce filtering to easily reduce noise effects, as shown.

Function	DI0 to DI3	DI4 to DI7	DI8 to DI9	DI10 to DI11
Trigger In	Yes	No	No	No
Encoder Latch	No	No	CH8:Encoder 0 CH9:Encoder 1	No
General D/I	Yes	Yes	Yes	Yes
De-bounce filter	Yes	Yes	Yes	Yes
Start Enable	No	Yes	No	No

Table 1-1: DI Channels and Support Functions

Function	DO0 to DO3	DO4 to DO15
Trigger Out	Yes	No
General DO	Yes	Yes

Table 1-2: DO Channels and Support Functions

Trigger Out can be SW (software) or HW (Hardware) triggered (i.e. trigger in and encoder), or a SW/HW hybrid.

One Trigger In can drive multiple Trigger Outs, each of which can have individual pulse and delay, as shown.

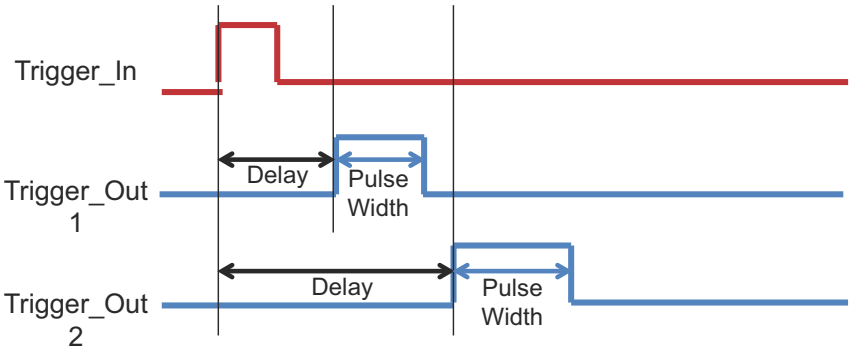


Figure 1-1: Single Trigger In Signal Driving Multiple Trigger Outs

With HW/SW combinations, flexible Trigger Out applications can be created, as shown.

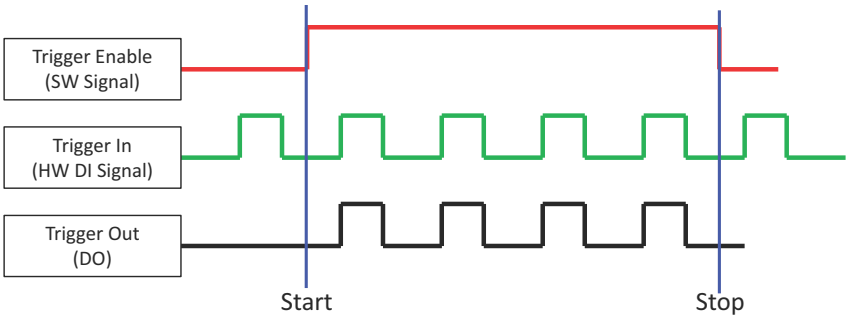


Figure 1-2: SW/HW Combination Trigger Out

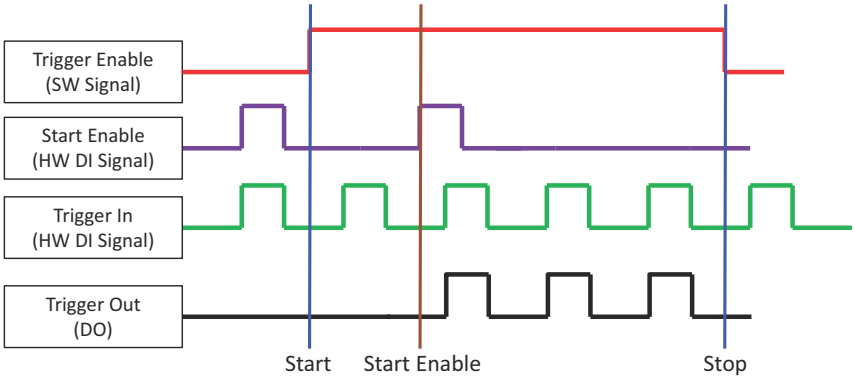


Figure 1-3: SW/HW Combination Trigger Out w/ HW Start Enabled

Referring to the Encoder, the user can send the Trigger Out pulse periodically, according to Encoder counts (Linear mode) or dynamically, according to CPU result and encoder counts (FIFO Compare Mode).

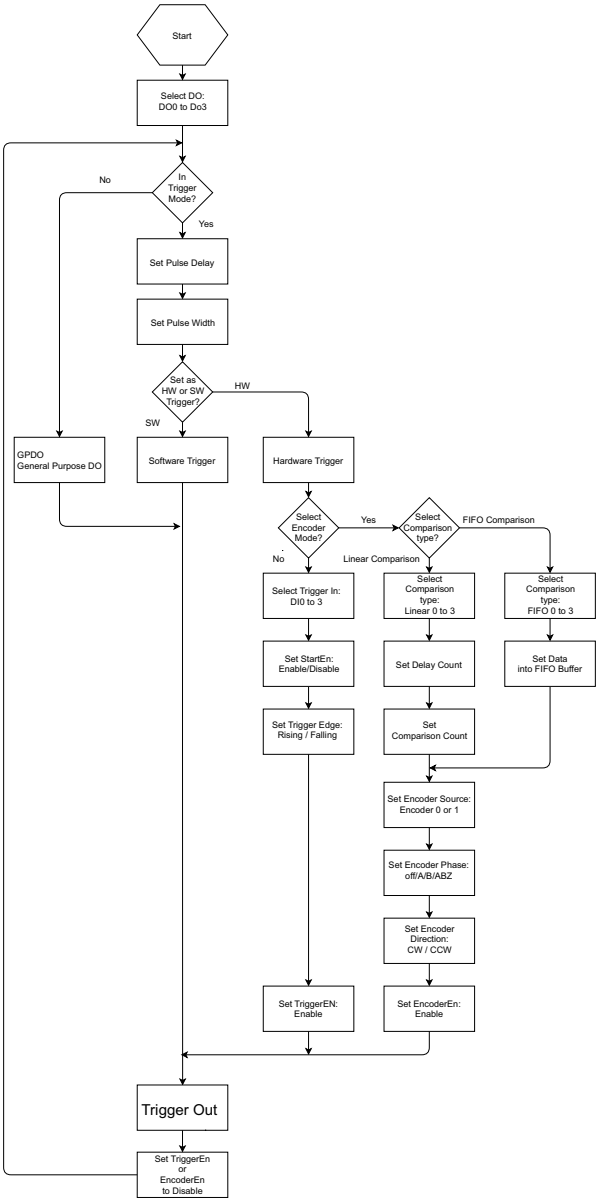


Figure 1-4: DI/O Flowchart

2 DI/O Function Library

2.1 List of Functions

Function Group	
Initialization ()See “Initialization” on page 7.	Register_Card
	Release_Card
DI/O (see “DI/O” on page 9)	DI_ReadLine
	DI_ReadPort
	DO_ReadLine
	DO_WriteLine
	DO_ReadPort
	DO_WritePort
Debounce (see “Debounce” on page 15)	SetDIFilter
	GetDIFilter
COS Interrupt (see “COS Interrupt” on page 18)	DIO_INT1_EventMessage
	DIO_INT2_EventMessage
	DIO_SetDualInterrupt
	DIO_SetCOSInterrupt32
	DIO_GetCOSLatchData32
Trigger (see “Trigger” on page 25)	GetTriggerInPolarity
	SetTriggerInPolarity
	GetTriggerOutPulseWidth
	SetTriggerOutPulseWidth
	GetTriggerOutDelay
	SetTriggerOutDelay
	GetDOMode
	SetDOMode
	SetSWTrigger
	GetHWTrigger
	SetHWTrigger
	GetTriggerStart
	SetTriggerStart
	GetTriggerEn

Function Group	
Trigger (see “Trigger” on page 25)	SetTriggerEn
	GetTriggerInCount
	GetTriggerOutCount
	ResetTriggerInCount
	ResetTriggerOutCount
Encoder (see “Encoder” on page 43)	GetEncoderEn
	SetEncoderEn
	GetEncoderInputMode
	SetEncoderInputMode
	GetEncoderCount
	GetEncoderDelayCount
	SetEncoderDelayCount
	GetEncoderLinearCount
	SetEncoderLinearCount
	GetEncoderInputDirection
	SetEncoderInputDirection
	GetComparedTriggerSource
	SetComparedTriggerSource
	SetEncoderLatchInt
	GetEncoderLatchDirection
	SetEncoderLatchDirection
	GetEncoderLatchData
	GetFIFOBufferUsage
	GetFIFOBufferFreeSpace
	GetFIFOBufferState
	GetFIFOBufferData
	SetFIFOBufferData
	ResetEncoderCount
	ResetFIFOBuffer

2.2 Function Library

2.2.1 Initialization

Register_Card

Initializes the hardware and software states of a PCI-bus data acquisition card, then returns a numeric card ID that corresponds to the initialized card. Register_Card must be called before any other PCIS-DASK library functions can be called for a particular card. The function initializes the card and variables internal to the PCIS-DASK library. Because PCI-bus data acquisition cards meet plug-and-play specifications, the base address (pass-through address) and IRQ level are assigned directly by the system BIOS

Syntax

C/C++

```
I16 Register_Card (U16 CardType, U16 card_num)
```

Visual Basic

```
Register_Card (ByVal CardType As Integer,
ByVal card_num As Integer) As Integer
```

VB.Net

```
Register_Card (ByVal CardType As Short, ByVal
card_num As Short) As Short
```

C#

```
short Register_Card (ushort CardType, ushort
card_num)
```

Parameters

CardType:

Type of card to be initialized. ADLINK periodically upgrades PCIS-DASK to add support for new PCI-bus data acquisition cards and NuIPC CompactPCI cards. Refer to release notes of the card to know if PCIS-DASK supports that card. These are the constants defined in DASK.H that represent the PCI-bus data acquisition cards supported by PCIS-DASK: PCIe-E13D

card_num:

Sequence number of the card with the same card type (as defined in argument CardType) or that belongs to the same card type series (except PCI- 7300A_Rev. A and PCI-7300A Rev. B) plugged in the PCI slot. card_num is always equal to 0 for PCIe-E13D.

Return Code

Returns a numeric card ID for the initialized card. The card ID range is between 0 and 31. If any error occurs, this returns a negative error code. Possible error codes:

```
ErrorTooManyCardRegistered
ErrorUnknownCardType
ErrorOpenDriverFailed
ErrorOpenEventFailed
```

Release_Card

At most, 32 cards can be registered simultaneously. This function tells the PCIS-DASK library that the registered card is not currently in use and may be released. Releasing a card creates a vacancy for a new card to register. Also used at the end of a program to release all registered cards.

Syntax

C/C++

```
I16 Release_Card (U16 CardNumber)
```

Visual Basic

```
Release_Card (ByVal CardNumber As Integer) As
Integer
```

VB.Net

```
Release_Card (ByVal CardNumber As Short) As
Short
```

C#

```
short Release_Card (ushort CardNumber)
```

Parameters

CardNumber:

ID of the card for release.

Return Code

`NoError`

2.2.2 DI/O

DI_ReadLine

Reads the digital logic state of the digital line in the specified port.

Syntax

C/C++

```
I16 DI_ReadLine(U16 CardNumber, U16 Port, U16 Line, U16 *State)
```

Visual Basic

```
DI_ReadLine(ByVal CardNumber As ushort, ByVal Port As ushort, ByVal Line As ushort, State As ushort) As Short
```

VB.Net

```
DI_ReadLine(ByVal CardNumber As ushort, ByVal Port As ushort, ByVal Line As ushort, ByRef State As ushort) As Short
```

C#

```
short DI_ReadLine(ushort CardNumber, ushort Port, ushort Line, out ushort State)
```

Parameters

CardNumber

ID of the card performing the operation.

Port

Digital input port number. Valid values, where PCIe-E13D supports only port 0.

Line

Digital line to be read, with valid values for PCIe-E13D of 0 to 11.

State

Returns the digital logic state of the specified line to 0 or 1.

Return Code

```
NoError 0
ErrorInvalidCardNumber -1
ErrorCardNotRegistered -4
ErrorFuncNotSupport -5
ErrorInvalidIoChannel -6
```

DI_ReadPort

Reads the digital data from the specified digital input port.

Syntax

C/C++

```
I16 DI_ReadPort(U16 CardNumber, U16 Port, U32
*Value)
```

Visual Basic

```
DI_ReadPort(ByVal CardNumber As ushort, ByVal
Port As ushort, Value As uint) As short
```

VB.Net

```
DI_ReadPort(ByVal CardNumber As ushort, ByVal
Port As ushort, ByRef Value As uint) As short
```

C#

```
short  DI_ReadPort(ushort  CardNumber,  ushort
Port, out uint Value)
```

Parameters

CardNumber:

ID of the card performing the operation.

Port:

Digital input port number. Valid values: PCIe-E13D only supports port 0

Value:

Returns the digital data read from the specified port. Valid values: PCIe-E13D 12-bit data (for port 0)

Return Code

```
NoError 0
ErrorInvalidCardNumber -1
```

```
ErrorCardNotRegistered -4
ErrorFuncNotSupport -5
ErrorInvalidIoChannel -6
```

DO_ReadLine

Reads back the digital logic state of the specified digital output line of the specified port.

Syntax

C/C++

```
I16 DO_ReadLine(U16 CardNumber, U16 Port, U16 Line, U16 *State)
```

Visual Basic

```
DO_ReadLine(ByVal CardNumber As ushort, ByVal Port As ushort, ByVal Line As ushort, State As ushort) As short
```

VB.Net

```
DO_ReadLine(ByVal CardNumber As ushort, ByVal Port As ushort, ByVal Line As ushort, ByRef State As ushort) As short
```

C#

```
short DO_ReadLine(ushort CardNumber, ushort Port, ushort Line, out ushort State)
```

Parameters

CardNumber:

ID of the card performing the operation.

Port:

Digital input port number. Valid values: PCIe-E13D only supports port 0

State

Returns the digital logic state, 0 or 1, of the specified line.

Return Code

```
NoError 0
ErrorInvalidCardNumber -1
```

```
ErrorCardNotRegistered -4
ErrorFuncNotSupport -5
ErrorInvalidIoChannel -6
```

DO_WriteLine

Sets the specified digital output line in the specified digital port to the specified state. Only available for cards supporting digital output read-back.

Syntax

C/C++

```
I16 DO_WriteLine(U16 CardNumber, U16 Port, U16 Line, U16 State)
```

Visual Basic

```
DO_WriteLine(ByVal CardNumber As ushort, ByVal Port As ushort, ByVal Line As ushort, ByVal State As ushort) As Integer
```

VB.Net

```
DO_WriteLine(ByVal CardNumber As ushort, ByVal Port As ushort, ByVal Line As ushort, ByVal State As ushort) As Short
```

C#

```
short DO_WriteLine(ushort CardNumber, ushort Port, ushort Line, ushort State)
```

Parameters

CardNumber:

ID of the card performing the operation.

Port:

Digital input port number. Valid values: PCIe-E13D only supports port 0.

Line

Digital line to be read. Valid values: PCIe-E13D 0 to 15(for port 0).

State

New digital logic state 0 or 1.

Return Code

```
NoError 0
ErrorInvalidCardNumber -1
ErrorCardNotRegistered -4
ErrorFuncNotSupport -5
ErrorInvalidIoChannel -6
```

DO_ReadPort

Reads back the output digital data from the specified digital output port.

Syntax

C/C++

```
I16 DO_ReadPort(U16 CardNumber, U16 Port, U32 *Value)
```

Visual Basic

```
DO_ReadPort(ByVal CardNumber As ushort, ByVal Port As ushort, Value As uint) As Short
```

VB.Net

```
DO_ReadPort(ByVal CardNumber As ushort, ByVal Port As ushort, ByRef Value As uint ) As Short
```

C#

```
short DO_ReadPort(ushort CardNumber, ushort Port, out uint Value)
```

Parameters

CardNumber:

ID of the card performing the operation.

Port:

Digital input port number. Valid values: PCIe-E13D only supports port 0

Value

Digital data written to the specified port. Valid values: PCIe-E13D 16-bit data (for port 0)

Return Code

```
ErrorInvalidCardNumber -1
ErrorCardNotRegistered -4
```

```
ErrorFuncNotSupport -5
ErrorInvalidIoChannel -6
```

DO_WritePort

Writes digital data to the specified digital output port.

Syntax

C/C++

```
I16 DO_WritePort(U16 CardNumber, U16 Port, U32
Value)
```

Visual Basic

```
DO_WritePort(ByVal CardNumber As ushort, ByVal
Port As ushort, ByVal Value As uint) As Short
```

VB.Net

```
DO_WritePort(ByVal CardNumber As ushort, ByVal
Port As ushort, ByVal Value As uint) As Short
```

C#

```
short DO_WritePort(ushort CardNumber, ushort
Port, uint Value)
```

Parameters

CardNumber:

ID of the card performing the operation.

Port:

Digital output port number. Valid values: PCIe-E13D only supports port 0

Value

Digital data written to the specified port. Valid values: PCIe-E13D 16-bit data (for port 0)

Return Code

```
ErrorInvalidCardNumber -1
ErrorCardNotRegistered -4
ErrorFuncNotSupport -5
ErrorInvalidIoChannel -6
```

2.2.3 Debounce

SetDIFilter

Sets time for all digital input filters.

Syntax

C/C++

```
I16 SetDIFilter(U16 CardNumber, U16 Line, U16 Value)
```

VB.Net

```
SetDIFilter(ByVal CardNumber As ushort, ByRef Line As ushort, ByVal Value As ushort) As Short
```

C#

```
short SetDIFilter(ushort CardNumber, ushort Line, ushort Value)
```

Parameters

CardNumber:

ID of the card performing the operation.

Line:

Digital line to be read.Valid values: PCIe-E13D 0 to 11

Value

0 disabling (off), and others digital filter time (1 to 9; 409.6* 2^(setting value) us)

Due to characteristics of HW components and external circuits, filter time may change dynamically.

Filter Time (50% duty cycle)	Value	Frequency (Hz)
0 (bypass)	0	Disabled
819.2 us	1	609.76
71.64 ms	2	304.88
3.28 ms	3	152.67
6.55 ms	4	76.22
13.10 ms	5	38.11
26.20 ms	6	19.06
52.43 ms	7	9.53
104.85 ms	8	4.76
209.72 ms	9	2.38

Return Code

```
NoError 0
ErrorInvalidCardNumber -1
ErrorCardNotRegistered -4
ErrorFuncNotSupport -5
ErrorInvalidIoChannel -6
```

GetDIFilter

Acquires time for all digital input filters.

Syntax

C/C++

```
I16 GetDIFilter(U16 CardNumber, U16 Line, U16
*Value)
```

VB.Net

```
GetDIFilter(ByVal CardNumber As ushort, ByRef
Line As ushort, ByRef Value As ushort) As Short
```

C#

```
short GetDIFilter(ushort CardNumber, out ush-
ort Line, out ushort Value)
```

Parameters

CardNumber:

ID of the card performing the operation.

Line:

Digital line to be read.Valid values: PCIe-E13D 0 to 11

Value

0 disabling (off), and others digital filter time (1 to 9; 409.6* 2^(setting value) us)


NOTE:

Due to characteristics of HW components and external circuits, filter time may change dynamically.

Filter Time (50% duty cycle)	Value	Frequency (Hz)
0 (bypass)	0	Disabled
819.2 us	1	609.76
71.64 ms	2	304.88
3.28 ms	3	152.67
6.55 ms	4	76.22
13.10 ms	5	38.11
26.20 ms	6	19.06
52.43 ms	7	9.53
104.85 ms	8	4.76
209.72 ms	9	2.38

Return Code

```
NoError 0
ErrorInvalidCardNumber -1
ErrorCardNotRegistered -4
ErrorFuncNotSupport -5
ErrorInvalidIoChannel -6
```

2.2.4 COS Interrupt

DIO_INT1_EventMessage

Controls INT1 interrupt sources for a dual-interrupt system and notifies the host application when an interrupt event occurs. Notification is issued via user-specified callback or the Windows PostMessage API.

Syntax

C/C++

```
I16 DIO_INT1_EventMessage(U16 CardNumber, I16 Int1Mode, HANDLE windowHandle, U32 message, void *callbackAddr())
```

Visual Basic

```
DIO_INT1_EventMessage(ByVal CardNumber As ushort, ByVal Int1Mode As short, ByVal windowHandle As Long, ByVal message As Long, ByVal callbackAddr As Long) As Short
```

VB.Net

```
DIO_INT1_EventMessage(ByVal CardNumber As ushort, ByVal Int1Mode As Short, ByVal windowHandle As Integer, ByVal message As Integer, ByVal callbackAddr As CallbackDelegate) As Short
```

C#

x64

```
short DIO_INT1_EventMessage(ushort CardNumber, short Int1Mode, IntPtr windowHandle, ulong message, MulticastDelegate callbackAddr)
```

x86

```
short DIO_INT1_EventMessage(ushort CardNumber, short Int1Mode, IntPtr windowHandle, uint message, MulticastDelegate callbackAddr)
```

Parameters

CardNumber:

ID of the card performing the operation.

Int1Mode:

Interrupt mode of INT1 by COS of Line0 of Port 0. Valid values: INT1_DISABLE or 0, and INT1_EXT_SIGNAL or 1

windowHandle:

Handle of the window to receive a Windows message when the specified INT event happens. If 0, no Windows messages will be sent.

message:

User-defined message sent when a specified INT event occurs. The message can be of any value. In Windows, the message can be set to a value including any Windows predefined messages, such as WM_PAINT. To define custom messages, any value ranging from WM_USER (0x400) to 0x7fff can be used. This range is reserved by Windows for user-defined messages.

callbackAddr:

Address of the user callback function. The PCIS-DASK calls this function when the specified INT event occurs. If no callback function is required, set callbackAddr to 0.

Return Code(s)

```
NoError 0
ErrorInvalidCardNumber -1
ErrorCardNotRegistered -4
ErrorFuncNotSupport -5
```

DIO_INT2_EventMessage

Controls INT2 interrupt sources for a dual-interrupt system and notifies the host application when an interrupt event occurs. Notification is issued via user-specified callback or the Windows PostMessage API.

Syntax

C/C++

```
I16 DIO_INT2_EventMessage(U16 CardNumber, I16
Int2Mode, HANDLE windowHandle, U32 message,
void *callbackAddr())
```

Visual Basic

```
DIO_INT2_EventMessage(ByVal CardNumber As ush-
ort, ByVal Int2Mode As short, ByVal windowHan-
dle As Long, ByVal message As Long, ByVal
callbackAddr As Long) As Short
```

VB.Net

```
DIO_INT2_EventMessage(ByVal CardNumber As
Short, ByVal Int2Mode As Short, ByVal window-
Handle As Integer, ByVal message As Integer,
ByVal callbackAddr As CallbackDelegate) As
Short
```

C#

x64

```
short DIO_INT2_EventMessage(ushort CardNum-
ber, short Int2Mode, IntPtr windowHandle,
ulong message, MulticastDelegate callbackAddr)
```

x86

```
short DIO_INT2_EventMessage(ushort CardNum-
ber, short Int2Mode, IntPtr windowHandle, uint
message, MulticastDelegate callbackAddr)
```

Parameters

CardNumber:

ID of the card performing the operation.

Int2Mode:

Interrupt mode of INT2 by COS of Line 1 of Port 0. Valid values: INT2_DISABLE or 0, and INT2_EXT_SIGNAL or 1

windowHandle:

Handle of the window to receive a Windows message when the specified INT event happens. If 0, no Windows messages will be sent.

message:

User-defined message sent when a specified INT event occurs. The message can be of any value. In Windows, the message can be set to a value including any Windows pre-defined messages, such as WM_PAINT. To define custom messages, any value ranging from WM_USER (0x400) to 0x7fff can be used. This range is reserved by Windows for user-defined messages.

callbackAddr:

Address of the user callback function. The PCIS-DASK calls this function when the specified INT event occurs. If no call-back function is required, set callbackAddr to 0.

Return Code(s)

```
NoError 0
ErrorInvalidCardNumber -1
ErrorCardNotRegistered -4
ErrorFuncNotSupport -5
```

DIO_SetDualInterrupt

Informs the PCIS-DASK library of the interrupt mode of sources of a dual-interrupt system and returns dual interrupt events. If an interrupt is generated, the corresponding inter-rupt events are signaled. The application uses Win32 wait functions, such as WaitForSingleObject or WaitForMulti-pleObjects to check the interrupt event status.

Syntax

C/C++

```
I16 DIO_SetDualInterrupt (U16 CardNumber, I16
Int1Mode, I16 Int2Mode, HANDLE *hEvent)
```

Visual Basic

```
DIO_SetDualInterrupt (ByVal CardNumber As ush-
ort, ByVal Int1Mode As short, ByVal Int2Mode
As short, hEvent As Long) As Short
```

VB.Net

```
DIO_SetDualInterrupt (ByVal CardNumber As ush-
ort, ByVal Int1Mode As short, ByVal Int2Mode
As Short, ByRef hEvent() As IntPtr) As Short
```

C#

```
short DIO_SetDualInterrupt (ushort CardNumber,
short Int1Mode, short Int2Mode, IntPtr hEvent)
```

Parameters

CardNumber:

ID of the card performing the operation.

Int1Mode:

Interrupt mode of INT1 by COS of Line0 of Port 0, with valid values:

INT1_DISABLE or 0

INT1_EXT_SIGNAL or 1

Int2Mode:

Interrupt mode of INT2 by COS of Line1 of Port 0, with valid values:

INT2_DISABLE or 0

INT2_EXT_SIGNAL INT2 or 1

hEvent (Windows only):

Returned dual-interrupt event handles. The status of a dual-interrupt event indicates that an interrupt is generated or not for cards comprising dual-interrupt system.

Return Code(s)

```
NoError 0
ErrorInvalidCardNumber -1
ErrorCardNotRegistered -4
ErrorFuncNotSupport -5
```

DIO_SetCOSInterrupt32

Enables or disables COS (Change Of State) interrupt detection for the specified ports with 32-bit data width.

Syntax

C/C++

```
I16 DIO_SetCOSInterrupt32 (U16 CardNumber, U8
Port, U32 ctl, HANDLE *hEvent, BOOLEAN Manual-
Reset)
```

VB.Net

```
DIO_SetCOSInterrupt32 (ByVal CardNumber As
ushort, ByVal Port As Byte, ByVal ctl As uint,
ByRef hEvent As Integer, ByVal ManualReset As
Byte) As Short
```

C#

```
short DIO_SetCOSInterrupt32 (ushort CardNum-
ber, byte Port, uint ctl, out SafeWaitHandle
hEvent, bool ManualReset)
```

Parameters

CardNumber:

ID of the card performing the operation.

Port:

Channel number where COS detection capability is to be enabled/disabled, with valid port numbers: PCIe-E13D 0

ctl:

Control value for the port defined by argument port. Valid values:

Each bit of the ctrl value controls one DI channel. The '0' value of the bit value disables COS function of the corresponding line, and the '1' value of the bit value enables the COS function of the corresponding line. Valid values for ctrl are 0 to 4095 (0xFFF)

hEvent :

Returned COS interrupt event handle.

ManualReset:

Specifies whether the event is (1) manual-reset by function ResetEvent in user application or (0) autoreset Address of the user callback function. PCIS-DASK calls this function when the specified COS event occurs. If no callback function is to be used, setting should be 0.

Return Code(s)

```
NoError 0
ErrorInvalidCardNumber -1
ErrorCardNotRegistered -4
ErrorFuncNotSupport -5
```

DIO_GetCOSLatchData32

Retrieves 32-bit width DI data latched in the COS Latch register when a Change-of-State (COS) interrupt occurs.

Syntax

C/C++

```
I16 DIO_GetCOSLatchData32(U16 CardNumber, U8
Port, U32 *CosLData)
```

Visual Basic

```
DIO_GetCOSLatchData32 (ByVal CardNumber As
ushort, ByVal Port As Byte, CosLData As uint)
As Short
```

VB.Net

```
DIO_GetCOSLatchData32 (ByVal CardNumber As
ushort, ByVal Port As Byte, ByRef CosLData As
uint) As Short
```

C#

```
short DIO_GetCOSLatchData32(ushort CardNum-
ber, byte Port, out uint CosLData)
```

Parameters

CardNumber:

ID of the card performing the operation.

Port:

Channel number on which COS detection capability is to be enabled/disabled. Valid port numbers: PCIe-E13D 0

CosLData:

Retrieves DI data latched in the COS Latch register when a Change-of-State (COS) interrupt occurs. PCIe-E13D 16-bit data.

Return Code(s)

```
NoError 0
ErrorInvalidCardNumber -1
ErrorCardNotRegistered -4
ErrorFuncNotSupport -5
```

2.2.5 Trigger

GetTriggerInPolarity

Acquires polarity of the specified Trigger In for the channel.

Syntax

C/C++

```
I16 GetTriggerInPolarity(U16 CardNumber, U16 Channel, U16 *Polarity)
```

VB.Net

```
GetTriggerInPolarity(ByVal CardNumber As ushort, ByVal Channel As ushort, ByRef Polarity As ushort) As Short
```

C#

```
short GetTriggerInPolarity(ushort CardNumber, ushort Channel, out of ushort Polarity)
```

Parameters

CardNumber:

ID of the card performing the operation.

Channel:

Indicates channel of trigger output, value can be from 0 to 3.

Polarity:

0: High active (Rising), 1: Low active (Falling)

Return Code(s)

```
NoError 0
```

```
ErrorInvalidCardNumber -1
ErrorInvalidIoChannel -6
ERRORInvalidParameter -218
ERROROnTheFly -219
```

SetTriggerInPolarity

Sets polarity of the specified Trigger In for the channel.

Syntax

C/C++

```
I16 SetTriggerInPolarity(U16 CardNumber, U16 Channel, U16 Polarity)
```

VB.Net

```
SetTriggerInPolarity(ByVal CardNumber As ushort, ByVal Channel As ushort, ByVal Polarity As ushort) As Short
```

C#

```
short SetTriggerInPolarity(ushort CardNumber, ushort Channel, ushort Polarity)
```

Parameters

CardNumber:

ID of the card performing the operation.

Channel:

Indicates channel of trigger output, value can be from 0 to 3.

Polarity:

0: High active (Rising), 1: Low active (Falling)

NOTE:

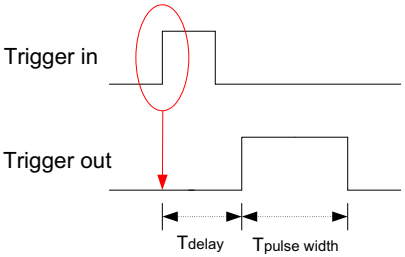
Disable SetTriggerEn()and SetEncoderEn() before configuring SetTriggerInPolarity().

Return Code(s)

```
NoError 0
```

```
ErrorInvalidCardNumber -1
ErrorInvalidIoChannel -6
ERRORInvalidParameter -218
ERROROnTheFly -219
```

Trigger by rising edge



Trigger by falling edge

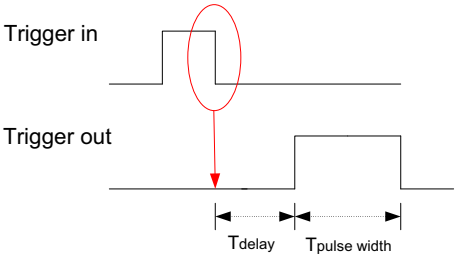


Figure 2-1: Trigger by Rising/Falling Edge

GetTriggerOutPulseWidth

Acquires pulse width of Trigger Out.

Syntax

C/C++

```
I16   GetTriggerOutPulseWidth(U16   CardNumber,
U16 Channel, U32 *Width)
```

VB.Net

```
GetTriggerOutPulseWidth(ByVal CardNumber As ushort, ByVal Channel As ushort, ByRef Width As uint) As Short
```

C#

```
short GetTriggerOutPulseWidth(ushort CardNumber, ushort Channel, out of unit Width)
```

Parameters

CardNumber:

ID of the card performing the operation.

Channel:

Indicates channel of trigger output, value can be from 0 to 3, with 0 to 3 indicating trigger in 0 to 3.

Width:

Pulse width, 0 to 2097151, indicating 1 to 2097150 us (21 bits).

Measure time = PulseWidth * 1.024
Due to the characteristics of HW components and external circuits, it is recommended that width be set above 1ms

Return Code(s)

```
NoError 0
ErrorInvalidCardNumber -1
ErrorInvalidIoChannel -6
ERRORInvalidParameter -218
ERROROnTheFly -219
```

SetTriggerOutPulseWidth

Sets pulse width of Trigger Out.

Syntax

C/C++

```
I16 SetTriggerOutPulseWidth(U16 CardNumber, U16 Channel, U32 Width)
```

VB.Net

```
SetTriggerOutPulseWidth(ByVal CardNumber As ushort, ByVal Channel As ushort, ByVal Width As uint) As Short
```

C#

```
short SetTriggerOutPulseWidth(ushort CardNumber, ushort Channel, uint Width)
```

Parameters

CardNumber:

ID of the card performing the operation.

Channel:

Indicates channel of trigger output, value can be from 0 to 3, with 0 to 3 indicating trigger in 0 to 3.

Width:

Pulse width, 0 to 2097151, indicating 1 to 2097150 us (21 bits).

Measure time = PulseWidth * 1.024
Due to the characteristics of HW components and external circuits, it is recommended that width be set above 1ms
Disable SetTriggerEn() and SetEncoderEn() before configuring SetTriggerOutPulseWidth

Return Code(s)

```
NoError 0
ErrorInvalidCardNumber -1
ErrorInvalidIoChannel -6
ERRORInvalidParameter -218
ERROROnTheFly -219
```

GetTriggerOutDelay

Acquires delay of Trigger Out.

Syntax

C/C++

```
I16 GetTriggerOutDelay(U16 CardNumber, U16
Channel, U32 *Delay)
```

VB.Net

```
GetTriggerOutDelay(ByVal CardNumber As uint,
ByVal Channel As uint, ByRef Delay As uint) As
Short
```

C#

```
short GetTriggerOutDelay(ushort CardNumber,
ushort Channel, out of uint Delay)
```

Parameters

CardNumber:

ID of the card performing the operation.

Channel:

Indicates channel of trigger output, value can be from 0 to 3, with 0 to 3 indicating trigger in 0 to 3.

Delay:

Pulse width, 0 to 2097151, indicating 1 to 2097150 us (21 bits).


NOTE:

Measure time = PulseWidth * 1.024
Due to the characteristics of HW components and external circuits, it is recommended that width be set above 1ms

Return Code(s)

```
NoError 0
ErrorInvalidCardNumber -1
ErrorInvalidIoChannel -6
ERRORInvalidParameter -218
ERROROnTheFly -219
```

SetTriggerOutDelay

Sets delay of Trigger Out.

Syntax

C/C++

```
I16 SetTriggerOutDelay(U16 CardNumber, U16 Channel, U32 Delay)
```

VB.Net

```
SetTriggerOutDelay(ByVal CardNumber As uint, ByVal Channel As uint, ByVal Delay As uint) As Short
```

C#

```
short SetTriggerOutDelay(ushort CardNumber, ushort Channel, uint Delay)
```

Parameters

CardNumber:

ID of the card performing the operation.

Channel:

Indicates channel of trigger output, value can be from 0 to 3, with 0 to 3 indicating trigger in 0 to 3.

Delay:

Pulse width, 0 to 2097151, indicating 1 to 2097150 us (21 bits).


NOTE:

Measure time = PulseWidth * 1.024
Due to the characteristics of HW components and external circuits, it is recommended that width be set above 1ms
Disable SetTriggerEn() and SetEncoderEn() efore configuring SetTriggerOutDelay()

Return Code(s)

```
NoError 0
ErrorInvalidCardNumber -1
ErrorInvalidIoChannel -6
ERRORInvalidParameter -218
ERROROnTheFly -219
```

GetDOMode

Acquires mode (GPDO, SW trigger or HW trigger).

Syntax

C/C++

```
I16 GetDOMode(U16 CardNumber, U16 Channel, U32
*Mode)
```

VB.Net

```
GetDOMode(ByVal CardNumber As ushort, ByVal
Channel As ushort, ByRef Mode As uint) As Short
```

C#

```
short GetDOMode(ushort CardNumber, ushort
Channel, out of uint Mode)
```

Parameters

CardNumber:

ID of the card performing the operation.

Channel:

Indicates channel of trigger output, value can be from 0 to 3, with 0 to 3 indicating trigger in 0 to 3.

Mode:

0: GPDO, 1: HW trigger, 2: SW trigger

Return Code(s)

```
NoError 0
ErrorInvalidCardNumber -1
ErrorInvalidIoChannel -6
ERRORInvalidParameter -218
ERROROnTheFly -219
```

SetDOMode

Sets (GPDO, SW trigger or HW trigger) of Trigger Out.

Syntax

C/C++

```
I16 SetDOMode(U16 CardNumber, U16 Channel, U32
Mode)
```

VB.Net

```
SetDOMode(ByVal CardNumber As ushort, ByVal
Channel As ushort, ByVal Mode As uint) As Short
```

C#

```
short SetDOMode(ushort CardNumber, ushort
Channel, uint Mode)
```

Parameters

CardNumber:

ID of the card performing the operation.

Channel:

Indicates channel of trigger output, value can be from 0 to 3, with 0 to 3 indicating trigger in 0 to 3.

Mode:

0: GPDO, 1: HW trigger, 2: SW trigger


NOTE:

Disable SetTriggerEn() and SetEncoderEn() before configuring SetTriggerOutDelay()

Return Code(s)

```
NoError 0
ErrorInvalidCardNumber -1
ErrorInvalidIoChannel -6
ERRORInvalidParameter -218
ERROROnTheFly -219
```

SetSWTrigger

Sets trigger of SW signal for each channel (DO0 to 3). Set function and trigger immediately.

Syntax

C/C++

```
I16 SetSWTrigger(U16 CardNumber, U16 Channel)
```

VB.Net

```
SetSWTrigger(ByVal CardNumber As ushort, ByVal
Channel As ushort) As Short
```

C#

```
short SetSWTrigger(ushort CardNumber, ushort
Channel)
```

Parameters

CardNumber:

ID of the card performing the operation.

Channel:

Indicates channel of trigger output, value can be from 0 to 3, with 0 to 3 indicating trigger in 0 to 3.

Return Code(s)

```
NoError 0
ErrorInvalidCardNumber -1
ErrorInvalidIoChannel -6
ERRORInvalidParameter -218
ERROROnTheFly -219
```

GetHWTrigger

Acquires Line In source of target DO.

Syntax

C/C++

```
I16 GetHWTrigger(U16 CardNumber, U16 Channel,
U16 *LineIn)
```

VB.Net

```
GetHWTrigger(ByVal CardNumber As ushort, ByVal
Channel As ushort, ByRef LineIn As ushort) As
Short
```

C#

```
short GetHWTrigger(ushort CardNumber, ushort
Channel, out of ushort LineIn)
```

Parameters

CardNumber:

ID of the card performing the operation.

Channel:

Indicates channel of trigger output, value can be from 0 to 3, with 0 to 3 indicating trigger out 0 to 3.

LineIn

0 to 3: Trigger DI0 to 3

4 to 7: Linear mode of compared trigger

8 to 11: FIFO mode of compared trigger

Return Code(s)

```
NoError 0
ErrorInvalidCardNumber -1
ErrorInvalidIoChannel -6
ERRORInvalidParameter -218
ERROROnTheFly -219
```

SetHWTrigger

Sets target DO as HW trigger and DI0 to 3, and Linear mode 4 to 7 or FIFO mode 8 to 11 as LineIn.

Syntax

C/C++

```
I16 SetHWTrigger(U16 CardNumber, U16 Channel,
U16 LineIn)
```

VB.Net

```
SetHWTrigger(ByVal CardNumber As ushort, ByVal
Channel As ushort, ByVal LineIn As ushort) As
Short
```

C#

```
short SetHWTrigger(ushort CardNumber, ushort
Channel, ushort LineIn)
```

Parameters

CardNumber:

ID of the card performing the operation.

Channel:

Indicates channel of trigger output, value can be from 0 to 3, with 0 to 3 indicating trigger out 0 to 3.

LineIn

0 to 3: Trigger DI0 to 3

4 to 7: Linear mode of compared trigger

8 to 11: FIFO mode of compared trigger


NOTE:

Disable SetTriggerEn() and SetEncoderEn() before configuring SetHWTrigger()

Return Code(s)

```
NoError 0
ErrorInvalidCardNumber -1
ErrorInvalidIoChannel -6
ERRORInvalidParameter -218
ERROROnTheFly -219
```

GetTriggerStart

Acquires state of trigger start mode.

Syntax

C/C++

```
I16 GetTriggerStart(U16 CardNumber, U16 Lin-
eIn, U16 *State)
```

VB.Net

```
GetTriggerStart(ByVal CardNumber As ushort,
ByVal LineIn As ushort, ByRef State As ushort)
As Short
```

C#

```
short GetTriggerStart(ushort CardNumber, ush-
ort LineIn, out of ushort State)
```

Parameters

CardNumber:

ID of the card performing the operation.

LineIn:

Indicates channel of trigger in, value can be from 0 to 3, with 0 to 3 indicating trigger in 0 to 3.

State

0: without start enable

1: with corresponding DI as HW trigger(DI4 -> DI0, …, DI 7 -> DI 3)

Line In as Trigger In	HW Signal as Trigger Start Pin
DI0	DI4
DI1	DI5
DI2	DI6
DI3	DI7

Return Code(s)

```
NoError 0
ErrorInvalidCardNumber -1
ErrorInvalidIoChannel -6
ERRORInvalidParameter -218
ERROROnTheFly -219
```

SetTriggerStart

Sets state of trigger start mode.

Syntax

C/C++

```
I16 SetTriggerStart(U16 CardNumber, U16 Lin-
eIn, U16 State)
```

VB.Net

```
SetTriggerStart(ByVal CardNumber As ushort,
ByVal LineIn As ushort, ByVal State As ushort)
As Short
```

C#

```
short SetTriggerStart(ushort CardNumber, ush-
ort LineIn, ushort State)
```

Parameters

CardNumber:

ID of the card performing the operation.

LineIn:

Indicates channel of trigger in, value can be from 0 to 3, with 0 to 3 indicating trigger in 0 to 3.

State

0: without start enable

1: with corresponding DI as HW trigger(DI4 -> DI0, …, DI 7 -> DI 3)

Line In as Trigger In	HW Signal as Trigger Start Pin
DI0	DI4
DI1	DI5
DI2	DI6
DI3	DI7

NOTE:

Disable SetTriggerEn() and SetEncoderEn() before configuring SetTriggerStart()

Return Code(s)

```
NoError 0
ErrorInvalidCardNumber -1
ErrorInvalidIoChannel -6
ERRORInvalidParameter -218
```

```
ERROROnTheFly -219
```

GetTriggerEn

Acquires state of trigger mode.

Syntax

C/C++

```
I16 GetTriggerEn(U16 CardNumber, U16 Trig-
gerIn, U16 *State)
```

VB.Net

```
GetTriggerEn(ByVal CardNumber As ushort, ByVal
TriggerIn As ushort, ByRef State As ushort) As
Short
```

C#

```
short GetTriggerEn(ushort CardNumber, ushort
TriggerIn, out of ushort State)
```

Parameters

CardNumber:

ID of the card performing the operation.

TriggerIn:

0 to 3: Trigger in 0 to 3

State

0: Disable (Reset DO channel referring to corresponding TriggerIn as GPDO. No HW trigger signal if statue set as 0)

1: Enable

Return Code(s)

```
NoError 0
ErrorInvalidCardNumber -1
ErrorInvalidIoChannel -6
ERRORInvalidParameter -218
```

SetTriggerEn

Sets state of trigger mode.

Syntax

C/C++

```
I16 SetTriggerEn(U16 CardNumber, U16 Trig-
gerIn, U16 State)
```

VB.Net

```
SetTriggerEn(ByVal CardNumber As ushort, ByVal
TriggerIn As ushort, ByVal State As ushort) As
Short
```

C#

```
short SetTriggerEn(ushort CardNumber, ushort
TriggerIn, ushort State)
```

Parameters

CardNumber:

ID of the card performing the operation.

TriggerIn:

0 to 3: Trigger in 0 to 3

State

0: Disable (Reset DO channel referring to corresponding TriggerIn as GPDO. No HW trigger signal if statue set as 0)

1: Enable

Return Code(s)

```
NoError 0
ErrorInvalidCardNumber -1
ErrorInvalidIoChannel -6
ERRORInvalidParameter -218
```

GetTriggerInCount

Acquires count value of valid Trigger In, incremental until reset by ResetTriggerInCount call.

Syntax

C/C++

```
I16 GetTriggerInCount(U16 CardNumber, U16
Channel, U32 *Count)
```

VB.Net

```
GetTriggerInCount(ByVal CardNumber As ushort,
ByVal Channel As ushort, ByRef Count As uint)
As Short
```

C#

```
I16 GetTriggerInCount(ushortCardNumber, ush-
ortChannel, out uint Count)
```

Parameters

CardNumber:

ID of the card performing the operation.

Channel:

Indicates Trigger In channel, value from 0 to 3.

Count:

Indicates count value.

Return Code(s)

```
NoError 0
ErrorInvalidCardNumber -1
ErrorInvalidIoChannel -6
```

GetTriggerOutCount

Acquires count value of valid Trigger Out, incremental until reset by ResetTriggerOutCount call.

Syntax

C/C++

```
I16 GetTriggerOutCount(U16 CardNumber, U16
Channel, U32 *Count)
```

VB.Net

```
GetTriggerOutCount(ByVal CardNumber As ushort,
ByVal Channel As ushort, ByRef Count As uint)
As Short
```

C#

```
I16 GetTriggerOutCount(ushortCardNumber, ush-
ort Channel, out uint Count)
```

Parameters

CardNumber:

ID of the card performing the operation.

Channel:

Indicates Trigger Out channel, value from 0 to 3.

Count:

Indicates count value.

Return Code(s)

```
NoError 0
ErrorInvalidCardNumber -1
ErrorInvalidIoChannel -6
```

ResetTriggerInCount

Resets Trigger In count.

Syntax

C/C++

```
I16 ResetTriggerInCount (U16 CardNumber, U16 Channel)
```

VB.Net

```
ResetTriggerInCount (ByVal CardNumber As ushort, ByVal Channel As ushort) As Short
```

C#

```
I16 ResetTriggerInCount (ushortCardNumber, ushortChannel)
```

Parameters

CardNumber:

ID of the card performing the operation.

Channel:

Indicates the channel of trigger in, value from 0 to 3.

Return Code(s)

```
NoError 0
```

```
ErrorInvalidCardNumber -1
ErrorInvalidIoChannel -6
ERRORInvalidParameter -218
```

ResetTriggerOutCount

Resets Trigger Out count.

Syntax

C/C++

```
I16 ResetTriggerOutCount (U16 CardNumber, U16 Channel)
```

VB.Net

```
ResetTriggerOutCount (ByVal CardNumber As ushort, ByVal Channel As ushort) As Short
```

C#

```
I16 ResetTriggerOutCount (ushort CardNumber, ushortChannel)
```

Parameters

CardNumber:

ID of the card performing the operation.

Channel:

Indicates the channel of trigger output, value from 0 to 3.

Return Code(s)

```
NoError 0
ErrorInvalidCardNumber -1
ErrorInvalidIoChannel -6
ERRORInvalidParameter -218
```

2.2.6 Encoder

GetEncoderEn

Acquires state of encoder.

Syntax

C/C++

```
I16 GetEncoderEn(U16 CardNumber, U16 Channel,
U16 *State)
```

VB.Net

```
GetEncoderEn(ByVal CardNumberAs Uushort short,
ByVal Channel As ushort, ByRef State As ush-
ort) As Short
```

C#

```
short GetEncoderEn (ushort CardNumber, ushort
Channel, out of ushort State)
```

Parameters

CardNumber:

ID of the card performing the operation.

Channel:

0 to 1: Encoder 0 to 1

State:

0: Disable

1: Enable

Return Code(s)

```
NoError 0
ErrorInvalidCardNumber -1
ErrorInvalidIoChannel -6
ERROR InvalidParameter -218
```

SetEncoderEn

Sets state of encoder.

Syntax

C/C++

```
I16 SetEncoderEn(ushortCardNumber, ushortChan-
nel, ushort State)
```

VB.Net

```
SetEncoderEn(ByVal CardNumber As ushort, ByVal
Channel As ushort, ByVal State As ushort) As
Short
```

C#

```
short SetEncoderEn(ushort CardNumber, ushort
Channel, ushort State)
```

Parameters

CardNumber:

ID of the card performing the operation.

Channel:

0 to 1: Encoder 0 to 1

State:

0: Disable

1: Enable

Return Code(s)

```
NoError 0
ErrorInvalidCardNumber -1
ErrorInvalidIoChannel -6
ERRORInvalidParameter -218
```

NOTE:

If SetEncoderEn() is disabled, encoder count is retained until count is reset via ResetEncoderCount().

GetEncoderInputMode

Acquires mode of encoder input, which indicates trigger input source type. Encoder input usually connects to optics ruler or motor encoder.

Syntax

C/C++

```
I16 GetEncoderInputMode(U16 CardNumber, U16
Channel, U16 *Mode)
```

VB.Net

```
GetEncoderInputMode(ByVal CardNumber As ushort, ByVal Channel As ushort, ByRef Mode As ushort) As Short
```

C#

```
I16 GetEncoderInputMode(ushort CardNumber, ushort Channel, out ushort Mode)
```

Parameters

CardNumber:

ID of the card performing the operation.

Channel:

Indicates the channel of Encoder, value from 0 to 1.

Mode:

0: Off

1: A phase

2: A/B phase

3: A/B/Z phase

Return Code(s)

```
NoError 0
ErrorInvalidCardNumber -1
ErrorInvalidIoChannel -6
ERRORInvalidParameter -218
ERROROnTheFly -219
```

SetEncoderInputMode

Sets mode of encoder input, indicating trigger input source type. Encoder input usually connects to optics ruler or motor encoder.

Syntax

C/C++

```
I16 SetEncoderInputMode(ushortCardNumber, ushortChannel, ushort Mode)
```

VB.Net

```
SetEncoderInputMode(ByVal CardNumber As ush-
ort, ByVal Channel As ushort, ByVal Mode As
ushort) As Short
```

C#

```
I16 SetEncoderInputMode(ushort CardNumber,
ushort Channel, ushort Mode)
```

Parameters

CardNumber:

ID of the card performing the operation.

Channel:

Indicates the channel of Encoder, value from 0 to 1.

Mode:

0: Off

1: A phase

2: A/B phase

3: A/B/Z phase

Disable SetTriggerEn() and SetEncoderEn() before configuring SetEncoderInputMode()
If EZ signal of ABZ mode is to be reset, encoder input mode must be reset by calling SetEncoderInputMode(CardNumber,Channel,0) before setting SetEncoderInputMode(CardNumber,Channel,3)

Return Code(s)

```
NoError 0
ErrorInvalidCardNumber -1
ErrorInvalidIoChannel -6
ERRORInvalidParameter -218
ERROROnTheFly -219
```

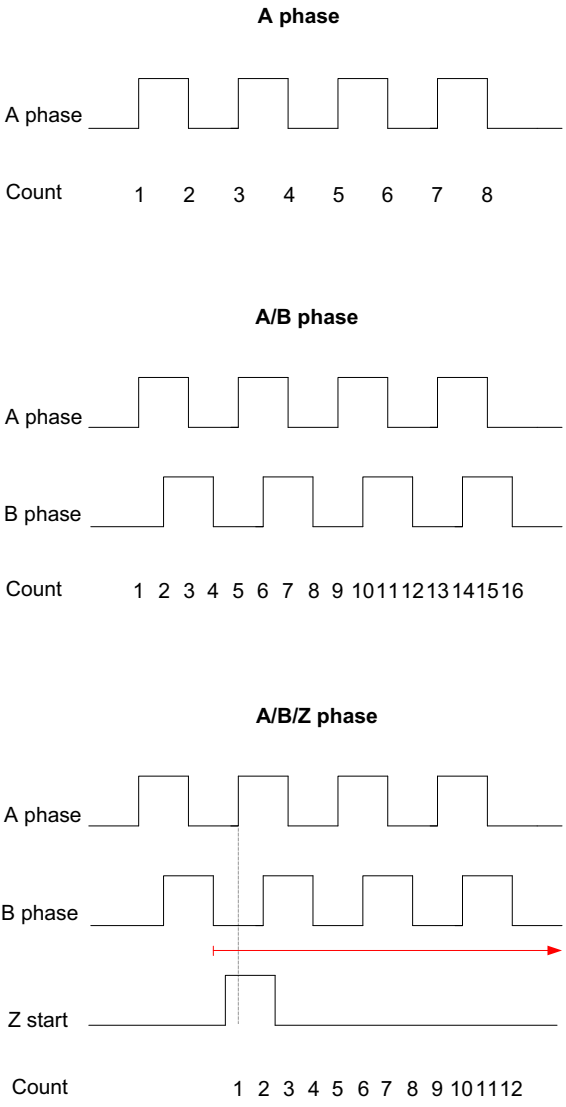


Figure 2-2: Encoder Signals

GetEncoderCount

Acquires count value of valid encoder trigger in (cnt_num), incremental or decremented until reset by ResetEncoder-Count call.

Syntax

C/C++

```
I16 GetEncoderCount(U16 CardNumber, U16  Chan-
nel, I32 *Count)
```

VB.Net

```
GetEncoderCount(ByVal  CardNumber  As  ushort,
ByVal Channel As ushort, ByRef Count As int) As
Short
```

C#

```
I16 GetEncoderCount(U16 CardNumber, U16  Chan-
nel, out int Count)
```

Parameters

CardNumber:

ID of the card performing the operation.

Channel:

Indicates the channel of Encoder, value from 0 to 1.

Count:

Indicates the count value by 2's complement.

Return Code(s)

```
NoError 0
ErrorInvalidCardNumber -1
ErrorInvalidIoChannel -6
```

GetEncoderDelayCount

Acquires count value of valid encoder delay count (dly_num).

Syntax

C/C++

```
I16   GetEncoderDelayCount(U16   CardNumber,U16
ComparedTrigger, U32 *Count)
```

VB.Net

```
GetEncoderDelayCount(ByVal CardNumber As ush-
ort, ByVal ComparedTrigger As ushort, ByRef
Count As U32) As Short
```

C#

```
I16   GetEncoderDelayCount(ushort   CardNumber,
ushort ComparedTrigger, out uint Count)
```

Parameters

CardNumber:

ID of the card performing the operation.

ComparedTrigger:

0 to 3: Linear mode 0 to 3.

Count:

Indicates the delay count of encoder input signal, a 32-bit positive integer.

Return Code(s)

```
NoError 0
ErrorInvalidCardNumber -1
ErrorInvalidIoChannel -6
ERRORInvalidParameter -218
ERROROnTheFly -219
```

SetEncoderDelayCount

Sets count value of valid encoder delay count (dly_num).

Syntax

C/C++

```
I16   SetEncoderDelayCount(U16   CardNumber,U16
ComparedTrigger, U32 Count)
```

VB.Net

```
SetEncoderDelayCount(ByVal CardNumber As ush-
ort, ByVal ComparedTriggeAs ushort, ByVal
Count As U32) As Short
```

C#

```
I16 SetEncoderDelayCount(ushort CardNumber,
ushort ComparedTrigger, uint Count)
```

Parameters

CardNumber:

ID of the card performing the operation.

ComparedTrigger:

0 to 3: Linear mode 0 to 3.

Count:

Indicates the delay count of encoder input signal, a 32-bit positive integer.


NOTE:

Disable SetTriggerEn() and SetEncoderEn() before configuring SetEncoderDelayCount()

Return Code(s)

```
NoError 0
ErrorInvalidCardNumber -1
ErrorInvalidIoChannel -6
ERRORInvalidParameter -218
ERROROnTheFly -219
```

GetEncoderLinearCount

Acquires comparison count of encoder input signal, i.e. how many signals are treated as valid.

Syntax

C/C++

```
I16 GetEncoderLinearCount(U16 CardNumber,U16
ComparedTrigger, U32 *Count)
```

VB.Net

```
GetEncoderLinearCount(ByVal CardNumber As ush-
ort, ByVal ComparedTriggerAs ushort, ByRef
Count As U32) As Short
```

C#

```
I16 GetEncoderLinearCount(ushort CardNumber,
ushort ComparedTrigger, out uint Count)
```

Parameters

CardNumber:

ID of the card performing the operation.

ComparedTrigger:

0 to 3: Linear mode 0 to 3.

Count:

Indicates the linear count of encoder input signal, a 32-bit positive integer.

Return Code(s)

```
NoError 0
ErrorInvalidCardNumber -1
ErrorInvalidIoChannel -6
ERRORInvalidParameter -218
ERROROnTheFly -219
```

SetEncoderLinearCount

Sets the compared count of encoder input signal, i.e. how many signals are treated as valid.

Syntax

C/C++

```
I16 SetEncoderLinearCount(U16 CardNumber,U16
ComparedTrigger, U32 Count)
```

VB.Net

```
SetEncoderLinearCount(ByVal CardNumber As ush-
ort, ByVal ComparedTriggerAs ushort, ByVal
Count As U32) As Short
```

C#

```
I16  SetEncoderLinearCount(ushort  CardNumber,
ushort ComparedTrigger, uint Count)
```

Parameters

CardNumber:

ID of the card performing the operation.

ComparedTrigger:

0 to 3: Linear mode 0 to 3

Count:

Indicates line count of encoder input signal, a 32-bit positive integer.

Return Code(s)

```
NoError 0
ErrorInvalidCardNumber -1
ErrorInvalidIoChannel -6
ERRORInvalidParameter -218
ERROROnTheFly -219
```

A phase input valid only (**A_en** is enabled), **enc_cmp_num** = 5, **dly_num_en** is disabled, as follows.

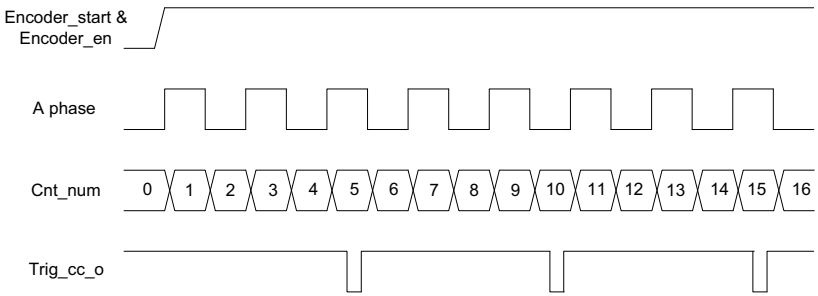


Figure 2-3: Encoder Mode A Phase without Delay

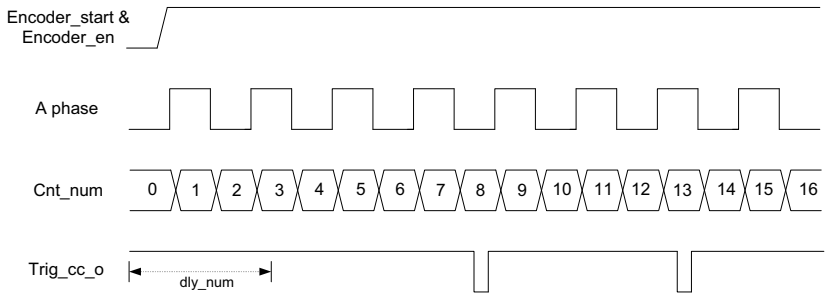


Figure 2-4: Encoder Mode A Phase with Delay

GetEncoderInputDirection

Acquires direction of encoder input signal when encoder input mode is A/B phase or A/B/Z phase.

Syntax

C/C++

```
I16    GetEncoderInputDirection(U16    CardNum-
ber,U16 Channel, U16 *Direction)
```

VB.Net

```
GetEncoderInputDirection(ByVal  CardNumber  As
ushort, ByVal Channel As ushort, ByRef Direc-
tion As ushort) As Short
```

C#

```
I16    GetEncoderInputDirection(U16    CardNum-
ber,U16 Channel, out ushort Direction)
```

Parameters

CardNumber:

ID of the card performing the operation.

Channel:

Indicates Encoder channel, value can be from 0 to 1.

Direction:

0: Clockwise rotation (CW)

1: Counter Clockwise rotation (CCW)

Return Code(s)

```
NoError 0
ErrorInvalidCardNumber -1
ErrorInvalidIoChannel -6
ERRORInvalidParameter -218
ERROROnTheFly -219
```

SetEncoderInputDirection

Sets direction of encoder input signal when encoder input mode is A/B phase or A/B/Z phase.

Syntax

C/C++

```
I16    SetEncoderInputDirection(U16    CardNum-
ber,U16 Channel, U16 Direction)
```

VB.Net

```
SetEncoderInputDirection(ByVal  CardNumber  As
ushort, ByVal Channel As ushort, ByVal Direc-
tion As ushort) As Short
```

C#

```
I16    SetEncoderInputDirection(U16    CardNum-
ber,U16 Channel, ushort Direction)
```

Parameters

CardNumber:

ID of the card performing the operation.

Channel:

Indicates Encoder channel, value can be from 0 to 1.

Direction:

0: Clockwise rotation (CW)

1: Counter Clockwise rotation (CCW)

Return Code(s)

```
NoError 0
ErrorInvalidCardNumber -1
```

```
ErrorInvalidIoChannel -6
ERROROInvalidParameter -218
ERROROnTheFly -219
```

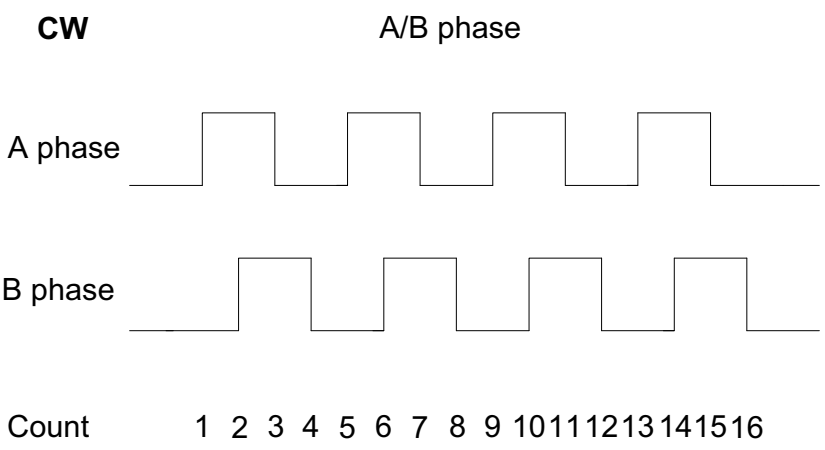


Figure 2-5: Encoder input direction signals A/B phase CW

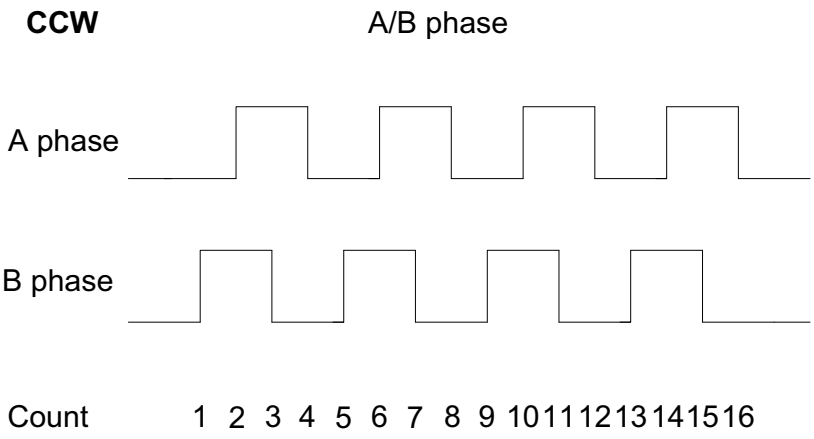


Figure 2-6: Encoder input direction signals A/B phase CCW

GetComparedTriggerSource

Acquires encoder source of each compared trigger.

Syntax

C/C++

```
I16  SetComparedTriggerSource(U16  CardNumber,
U16 ComparedTrigger, U16 Source)
```

VB.Net

```
GetComparedTriggerSource(ByVal  CardNumber  As
ushort, ByVal ComparedTrigger As ushort, ByRef
Source As ushort) As Short
```

C#

```
I16    GetComparedTriggerSource(U16    CardNum-
ber,U16 ComparedTrigger, out ushort Source)
```

Parameters

CardNumber:

ID of the card performing the operation.

ComparedTrigger:

0 to 3: Linear mode 0 to 3

4 to 7: FIFO mode 0 to 3

Source:

0: Encoder 0

1: Encoder 1

Return Code(s)

```
NoError 0
ErrorInvalidCardNumber -1
ErrorInvalidIoChannel -6
ERRORInvalidParameter -218
ERROROnTheFly -219
```

SetComparedTriggerSource

Sets encoder source of each compared trigger.

Syntax

C/C++

```
I16  SetComparedTriggerSource(U16  CardNumber,
U16 ComparedTrigger, U16 Source)
```

VB.Net

```
SetComparedTriggerSource(ByVal  CardNumber  As
ushort, ByVal ComparedTrigger As ushort, ByVal
Source As ushort) As Short
```

C#

```
I16    SetComparedTriggerSource(U16    CardNum-
ber,U16 ComparedTrigger, ushort Source)
```

Parameters

CardNumber:

ID of the card performing the operation.

ComparedTrigger:

0 to 3: Linear mode 0 to 3

4 to 7: FIFO mode 0 to 3

Source:

0: Encoder 0

1: Encoder 1

NOTE:

Disable SetTriggerEn() and SetEncoderEn() before configuring SetComparedTriggerSource()

Return Code(s)

```
NoError 0
ErrorInvalidCardNumber -1
ErrorInvalidIoChannel -6
ERRORInvalidParameter -218
ERROROnTheFly -219
```

SetEncoderLatchInt

Interrupt is generated when DI8 or DI9 receive pull high or pull low signal, with corresponding interrupt events sig-

naled. The application uses Win32 wait functions, such as WaitForSingleObject or WaitForMultipleObjects to determine interrupt event status.

Syntax

C/C++

```
I16 SetEncoderLatchInt(U16 CardNumber, I16
Int1Mode, I16 Int2Mode, HANDLE *hEvent)
```

Linux C/C++

```
I16 SetEncoderLatchInt(U16 CardNumber, I16
Int1Mode,       I16       Int2Mode,       void
(*event1_handler)(int),                   void
(*event2_handler)(int))
```

Visual Basic

```
SetEncoderLatchInt(ByVal CardNumber As Inte-
ger, ByVal Int1Mode As Integer, ByVal Int2Mode
As Integer, hEvent As Long) As Integer
```

VB.Net

```
SetEncoderLatchInt(ByVal CardNumber As Short,
ByVal Int1Mode As Short, ByVal Int2Mode As
Short, ByRef hEvent() As IntPtr) As Short
```

C#

```
short SetEncoderLatchInt(ushort CardNumber,
short Int1Mode, short Int2Mode, IntPtr hEvent)
```

Parameters

CardNumber:

ID of the card performing the operation.

Int1Mode:

Interrupt mode of INT1 by COS of Line8 of Port 0, with valid values:

INT1_DISABLE or 0

INT1_EXT_SIGNAL or 1

Int2Mode:

Interrupt mode of INT2 by COS of Line9 of Port 0, with valid values:

INT2_DISABLE or 0

INT2_EXT_SIGNAL or 1

hEvent (Windows only):

Returns dual-interrupt event handles, where status of a dual-interrupt event indicates whether an interrupt is generated for cards comprising dual-interrupt system.

Return Code(s)

```
NoError 0
ErrorInvalidCardNumber -1
ErrorCardNotRegistered -4
ErrorFuncNotSupport -5
```

GetEncoderLatchDirection

Acquires direction of encoder latch signal, DI8 and DI9 corresponding to encoder 0 and encoder 1.

Syntax

C/C++

```
I16    GetEncoderLatchDirection(U16    CardNum-
ber,U16 Channel, U16 *Direction)
```

VB.Net

```
GetEncoderLatchDirection(ByVal  CardNumber  As
ushort, ByVal Channel As ushort, ByRef Direc-
tion As ushort) As Short
```

C#

```
I16    GetEncoderLatchDirection(U16    CardNum-
ber,U16 Channel, out ushort Direction)
```

Parameters

CardNumber:

ID of the card performing the operation.

Channel:

Indicates Encoder channel, value can be from 0 to 1.

Direction:

Based on the direction of the specific DI signal

0: rising, 1: falling

Return Code(s)

```
NoError 0
ErrorInvalidCardNumber -1
ErrorInvalidIoChannel -6
ERRORInvalidParameter -218
ERROROnTheFly -219
```

SetEncoderLatchDirection

Sets direction of encoder latch signal, with DI8 and DI9 corresponding to encoder 0 and encoder 1.

Syntax

C/C++

```
I16    SetEncoderLatchDirection(U16    CardNum-
ber,U16 Channel, U16 Direction)
```

VB.Net

```
SetEncoderLatchDirection(ByVal  CardNumber  As
ushort, ByVal Channel As ushort, ByVal Direc-
tion As ushort) As Short
```

C#

```
I16    SetEncoderLatchDirection(U16    CardNum-
ber,U16 Channel, ushort Direction)
```

Parameters

CardNumber:

ID of the card performing the operation.

Channel:

Indicates Encoder channel, value can be from 0 to 1.

Direction:

Based on the direction of the specific DI signal

0: rising, 1: falling

Return Code(s)

```
NoError 0
ErrorInvalidCardNumber -1
ErrorInvalidIoChannel -6
ERRORInvalidParameter -218
ERROROnTheFly -219
```

GetEncoderLatchData

Acquires encoder0 count when DI8 is rising or falling (set by SetEncoderLatchDirection) and acquires encoder1 count when DI9 is rising or falling.

Syntax

C/C++

```
I16   GetEncoderLatchData(U16   CardNumber,U16
Channel, I32 *Count)
```

VB.Net

```
GetEncoderLatchData(ByVal CardNumber As ush-
ort, ByVal Channel As ushort, ByRef Count As
int) As Short
```

C#

```
I16   GetEncoderLatchData(ushort   CardNumber,
ushort Channel, out int Count
```

Parameters

CardNumber:

ID of the card performing the operation.

Channel:

Indicates Encoder channel, value can be from 0 to 1.

Count:

Indicates the count value by 2’s complement.

Return Code(s)

```
NoError 0
ErrorInvalidCardNumber -1
ErrorInvalidIoChannel -6
ERRORInvalidParameter -218
```

ERROROnTheFly -219

When Encoder Latch Switch (Encoder0: DI8; Encoder1: DI9) changes value, latch the encoder counter number.

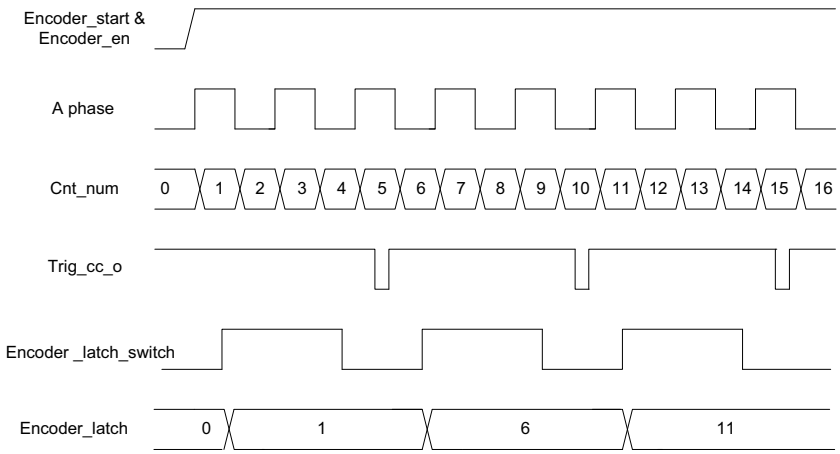


Figure 2-7: Encoder Mode A Phase with Encoder Latch (rising edge)

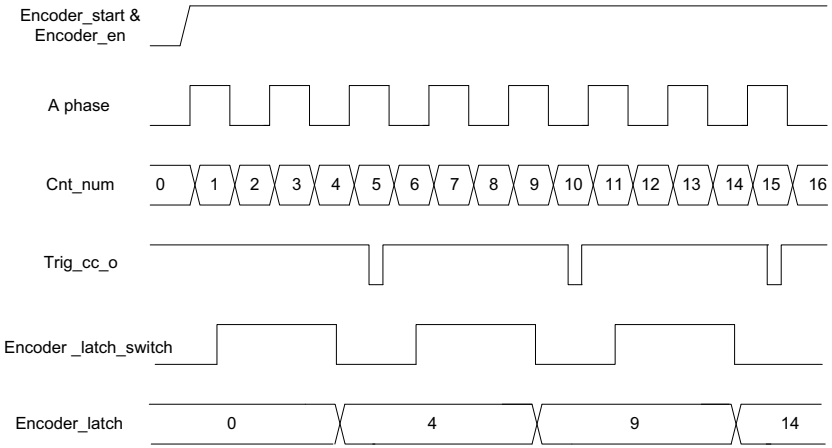


Figure 2-8: Encoder Mode A Phase with Encoder Latch (falling edge)

GetFIFOBufferUsage

Acquires the number of comparison values in FIFO buffer.

Syntax

C/C++

```
I16 GetFIFOBufferUsage(U16 CardNumber,U16 Com-
paredTrigger, U16 *Count)
```

VB.Net

```
GetFIFOBufferUsage(ByVal CardNumber As ushort,
ByVal ComparedTriggerAs ushort, ByRef Count As
ushort) As Short
```

C#

```
I16 GetFIFOBufferUsage(ushort CardNumber, ush-
ort ComparedTrigger, out ushort Count)
```

Parameters

CardNumber:

ID of the card performing the operation.

ComparedTrigger:

0 to 3: FIFO mode 0 to 3.

Count:

0 to 1023, where

0: Buffer empty

1 to 1022: Buffer populated

1023: Buffer full

Return Code(s)

```
NoError 0
ErrorInvalidCardNumber -1
ErrorInvalidIoChannel -6
ERRORInvalidParameter -218
ERROROnTheFly -219
```

GetFIFOBufferFreeSpace

Acquires the amount of space remaining in FIFO buffer.

Syntax

C/C++

```
I16 GetFIFOBufferFreeSpace(U16 CardNumber,U16
ComparedTrigger, U16 *Count)
```

VB.Net

```
GetFIFOBufferFreeSpace(ByVal CardNumber As
ushort, ByVal ComparedTriggerAs ushort, ByRef
Count As ushort) As Short
```

C#

```
I16 GetFIFOBufferFreeSpace(ushort CardNumber,
ushort ComparedTrigger, out ushort Count)
```

Parameters

CardNumber:

ID of the card performing the operation.

ComparedTrigger:

0 to 3: FIFO mode 0 to 3.

Count:

0 to 1023, where

0: Buffer full

1 to 1022: Buffer populated

1023: Buffer empty

Return Code(s)

```
NoError 0
ErrorInvalidCardNumber -1
ErrorInvalidIoChannel -6
ERRORInvalidParameter -218
ERROROnTheFly -219
```

GetFIFOBufferState

Acquires whether the FIFO buffer is populated.

Syntax

C/C++

```
I16 GetFIFOBufferState(U16 CardNumber,U16 Com-
paredTrigger, U16 *State)
```

VB.Net

```
GetFIFOBufferState(ByVal CardNumber As ushort,
ByVal ComparedTriggerAs ushort, ByRef State As
ushort) As Short
```

C#

```
I16 GetFIFOBufferState(ushort CardNumber, ush-
ort ComparedTrigger, out ushort State)
```

Parameters

CardNumber:

ID of the card performing the operation.

ComparedTrigger:

0 to 3: FIFO mode 0 to 3.

State:

0: Buffer empty

1: Buffer populated

2: Buffer full

Return Code(s)

```
NoError 0
ErrorInvalidCardNumber -1
ErrorInvalidIoChannel -6
ERRORInvalidParameter -218
ERROROnTheFly -219
```

GetFIFOBufferData

Acquires the latest comparison value from FIFO buffer.

Syntax

C/C++

```
I16 GetFIFOBufferData(U16 CardNumber,U16 Com-
paredTrigger,I32 *Value)
```

VB.Net

```
GetFIFOBufferData(ByVal CardNumber As ushort,
ByVal ComparedTrigger As ushort, ByRef Value
As I32) As Short
```

C#

```
I16 GetFIFOBufferData(ushort CardNumber, ush-
ort ComparedTrigger,out int Value)
```

Parameters

CardNumber:

ID of the card performing the operation.

ComparedTrigger:

0 to 3: FIFO mode 0 to 3.

Value:

The latest comparison value in FIFO buffer (2's complement).

Return Code(s)

```
NoError 0
ErrorInvalidCardNumber -1
ErrorInvalidIoChannel -6
ERRORInvalidParameter -218
ERROROnTheFly -219
```

SetFIFOBufferData

Sets comparison value in FIFO buffer.

Syntax

C/C++

```
I16 SetFIFOBufferData(U16 CardNumber,U16 Com-
paredTrigger, I32 Value)
```

VB.Net

```
SetFIFOBufferData(ByVal CardNumber As ushort,
ByVal ComparedTriggerAs ushort, ByVal Value As
I32) As Short
```

C#

```
I16 SetFIFOBufferData(ushort CardNumber, ush-
ort ComparedTrigger, int Value)
```

Parameters

CardNumber:

ID of the card performing the operation.

ComparedTrigger:

0 to 3: FIFO mode 0 to 3.

Value:

The comparison value in FIFO buffer (2's complement).

Return Code(s)

```
NoError 0
ErrorInvalidCardNumber -1
ErrorInvalidIoChannel -6
ERRORInvalidParameter -218
ERROROnTheFly -219
ERRORFIFOFull -220
```

ResetEncoderCount

Resets encoder count.

Syntax

C/C++

```
I16 ResetEncoderCount(U16 CardNumber, U16
Channel)
```

VB.Net

```
ResetEncoderCount(ByVal CardNumber As ushort,
ByVal Channel As ushort) As Short
```

C#

```
I16 ResetEncoderCount(U16 CardNumber, U16
Channel)
```

Parameters

CardNumber:

ID of the card performing the operation.

Channel:

Indicates the channel of Encoder, value can be from 0 to 1.

Return Code(s)

```
NoError 0
ErrorInvalidCardNumber -1
ErrorInvalidIoChannel -6
ERRORInvalidParameter -218
```

ResetFIFOBuffer

Clears all data from the FIFO buffer.

Syntax

C/C++

```
I16 ResetFIFOBuffer(U16 CardNumber, U16 Com-
paredTrigger)
```

VB.Net

```
ResetFIFOBuffer(ByVal CardNumber As ushort,
ByVal ComparedTrigger As ushort) As Short
```

C#

```
I16 ResetFIFOBuffer(U16 CardNumber, U16 Com-
paredTrigger)
```

Parameters

CardNumber:

ID of the card performing the operation.

ComparedTrigger:

0 to 3: FIFO mode 0 to 3.

Return Code(s)

```
NoError 0
ErrorInvalidCardNumber -1
ErrorInvalidIoChannel -6
ERRORInvalidParameter -218
```

This page intentionally left blank.

Important Safety Instructions

For user safety, please read and follow all instructions, Warnings, Cautions, and Notes marked in this manual and on the associated device before handling/operating the device, to avoid injury or damage.

S'il vous plaît prêter attention stricte à tous les avertissements et mises en garde figurant sur l'appareil , pour éviter des blessures ou des dommages.

- Read these safety instructions carefully
- Keep the User's Manual for future reference
- Read the Specifications section of this manual for detailed information on the recommended operating environment
- The device can be operated at an ambient temperature of 55ºC (with DC supply) and 50ºC (with adapter supply);
- When installing/mounting or uninstalling/removing device; or when removal of a chassis cover is required for user servicing (See "DI/O Function Library" on page 5.):
  - Turn off power and unplug any power cords/cables
  - Reinstall all chassis covers before restoring power
- To avoid electrical shock and/or damage to device:
  - Keep device away from water or liquid sources
  - Keep device away from high heat or humidity
  - Keep device properly ventilated (do not block or cover ventilation openings)
  - Always use recommended voltage and power source settings
  - Always install and operate device near an easily accessible electrical outlet
  - Secure the power cord (do not place any object on/over the power cord)
  - Only install/attach and operate device on stable surfaces and/or recommended mountings
- If the device will not be used for long periods of time, turn off and unplug from its power source

- Never attempt to repair the device, which should only be serviced by qualified technical personnel using suitable tools
- A Lithium-type battery may be provided for uninterrupted backup or emergency power.

Risk of explosion if battery is replaced with one of an incorrect type; please dispose of used batteries appropriately.
Risque d'explosion si la pile est remplacée par une autre de type incorrect. Veuillez jeter les piles usagées de façon appropriée.

- The device must be serviced by authorized technicians when:
  - The power cord or plug is damaged
  - Liquid has entered the device interior
  - The device has been exposed to high humidity and/or moisture
  - The device is not functioning or does not function according to the User's Manual
  - The device has been dropped and/or damaged and/or shows obvious signs of breakage
- Disconnect the power supply cord before loosening the thumbscrews and always fasten the thumbscrews with a screwdriver before starting the system up
- It is recommended that the device be installed only in a server room or computer room where access is:
  - Restricted to qualified service personnel or users familiar with restrictions applied to the location, reasons therefor, and any precautions required
  - Only afforded by the use of a tool or lock and key, or other means of security, and controlled by the authority responsible for the location
- If PoE (Power over Ethernet) is enabled for the device, the system can ONLY be deployed indoors. Unless otherwise noted, the PoE system is NOT designed to withstand the rigors of outdoor use.

	**BURN HAZARD** Touching this surface could result in bodily injury. To reduce risk, allow the surface to cool before touching. ***RISQUE DE BRÛLURES*** *Ne touchez pas cette surface, cela pourrait entraîner des blessures.* *Pour éviter tout danger, laissez la surface refroidir avant de la toucher.*

This page intentionally left blank.

Getting Service

Ask an Expert: http://askanexpert.adlinktech.com

ADLINK Technology, Inc.
Address: 9F, No.166 Jian Yi Road, Zhonghe District
New Taipei City 235, Taiwan
新北市中和區建一路 166 號 9 樓
Tel: +886-2-8226-5877
Fax: +886-2-8226-5717
Email: service@adlinktech.com

Ampro ADLINK Technology, Inc.
Address: 5215 Hellyer Avenue, #110
San Jose, CA 95138, USA
Tel: +1-408-360-0200
Toll Free: +1-800-966-5200 (USA only)
Fax: +1-408-360-0222
Email: info@adlinktech.com

ADLINK Technology (China) Co., Ltd.
Address: 上海市浦东新区张江高科技园区芳春路 300 号 (201203)
300 Fang Chun Rd., Zhangjiang Hi-Tech Park
Pudong New Area, Shanghai, 201203 China
Tel: +86-21-5132-8988
Fax: +86-21-5132-3588
Email: market@adlinktech.com

ADLINK Technology Beijing
Address: 北京市海淀区上地东路 1 号盈创动力大厦 E 座 801 室(100085)
Rm. 801, Power Creative E, No. 1 Shang Di East Rd.
Beijing, 100085 China
Tel: +86-10-5885-8666
Fax: +86-10-5885-8626
Email: market@adlinktech.com

ADLINK Technology Shenzhen
Address: 深圳市南山区科技园南区高新南七道 数字技术园
A1 栋 2 楼 C 区 (518057)
2F, C Block, Bldg. A1, Cyber-Tech Zone, Gao Xin Ave. Sec. 7
High-Tech Industrial Park S., Shenzhen, 518054 China
Tel: +86-755-2643-4858
Fax: +86-755-2664-6353
Email: market@adlinktech.com

LiPPERT ADLINK Technology GmbH
Address: Hans-Thoma-Strasse 11
D-68163 Mannheim, Germany
Tel: +49-621-43214-0
Fax: +49-621 43214-30
Email: emea@adlinktech.com

PENTA ADLINK Technology GmbH
Ulrichsbergerstrasse 17
94469 Deggendorf, Germany
Tel: +49 (0) 991 290 94 – 10
Fax: +49 (0) 991 290 94 - 29
Email: emea@adlinktech.com

ADLINK Technology, Inc. (French Liaison Office)
Address: 6 allée de Londres, Immeuble Ceylan
91940 Les Ulis, France
Tel: +33 (0) 1 60 12 35 66
Fax: +33 (0) 1 60 12 35 66
Email: france@adlinktech.com

ADLINK Technology Japan Corporation
Address: 〒101-0045 東京都千代田区神田鍛冶町 3-7-4
神田 374 ビル 4F
KANDA374 Bldg. 4F, 3-7-4 Kanda Kajicho,
Chiyoda-ku, Tokyo 101-0045, Japan
Tel: +81-3-4455-3722
Fax: +81-3-5209-6013
Email: japan@adlinktech.com

ADLINK Technology, Inc. (Korean Liaison Office)
Address: 경기도 성남시 분당구 수내로 46 번길 4 경동빌딩 2 층
(수내동 4-4 번지) (우) 463-825
2F, Kyungdong B/D, 4 Sunae-ro 46 beon-gil
Bundang-gu, Seongnam-si, Gyeonggi-do, Korea, 463-825
Toll Free +82-80-800-0585
Tel +82-31-786-0585
Fax +82-31-786-0583
Email: korea@adlinktech.com

ADLINK Technology Singapore Pte. Ltd.
Address: 84 Genting Lane #07-02A, Cityneon Design Centre
Singapore 349584
Tel: +65-6844-2261
Fax: +65-6844-2263
Email: singapore@adlinktech.com

ADLINK Technology Singapore Pte. Ltd. (Indian Liaison Office)
Address: #50-56, First Floor, Spearhead Towers
Margosa Main Road (between 16th/17th Cross)
Malleswaram, Bangalore - 560 055, India
Tel: +91-80-65605817, +91-80-42246107
Fax: +91-80-23464606
Email: india@adlinktech.com

ADLINK Technology, Inc. (Israeli Liaison Office)
Address: 27 Maskit St., Corex Building
PO Box 12777
Herzliya 4673300, Israel
Tel: +972-54-632-5251
Fax: +972-77-208-0230
Email: israel@adlinktech.com

ADLINK Technology, Inc. (UK Liaison Office)
Tel: +44 774 010 59 65
Email: UK@adlinktech.com