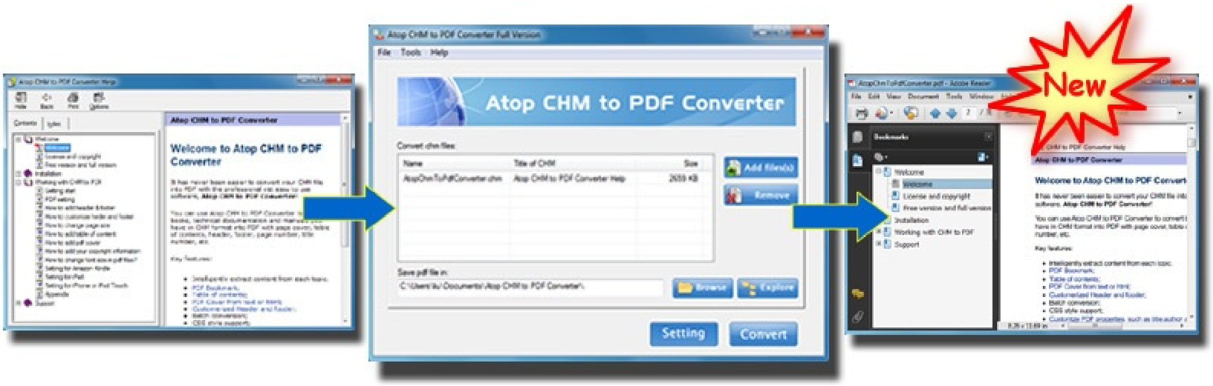
Atop CHM to PDF Converter 2.0


Atop CHM to PDF Converter Help
Hide
Back
Print
Options
Contents
Index
Welcome to Atop CHM to PDF Converter
Atop CHM to PDF Converter Full Version
File
Tools
Help
Atop CHM to PDF Converter
Add file(s)
Remove
Browse
Explore
Setting
Convert
AtopChmToPdfConverter.pdf - Adobe Reader
Bookmarks
Welcome
License and copyright
Free version and full version
Installation
Working with CHM to PDF
Support
New

Convert chm files:

Name	Title of CHM	Size
AtopChmToPdfConverter.chm	Atop CHM to PDF Converter Help	2659 KB

Save pdf file in:
C:\Users\lu\Documents\Atop CHM to PDF Converter\

http://www.chmconverter.com

NOTE: Full version does not display this page!

PCI-1220 Common Motion Driver User Manual

Ver 1.00

Web: http://www.advantech.com/eAutomation
e-mail: info@advantech.com

About this manual

This manual contains the information you need to get started with PCI-1220. This user manual is divided into the following sections:

Hardware Features

Introduction of the PCI-1220 hardware.

Introduction of Advantech Common Motion Architecture

Introduce Common Motion Architecture and it's features.

Introduction of PCI-1220 new driver (based on MS WDF framework)

Gives the user a basic idea of new driver's architecture.

Advantech Motion Utility

Introduction of the Advantech Motion Utility, and also a step by step guidance is provided for PCI-1220's Motion test.

Getting Started with PCI-1220

Provides some information on how to build an application using PCI-1220 Driver in **Visual C++.**

Besides the function description, we also provide user with a set of individual examples for VC. This gives user an easy access to PCI-1220. User can use these examples as a reference while designing their own application. They are also very useful for user to better understand PCI-1220 functions.

Function Description

Multiple functions are provided within PCI-1220. In this section, user can get the description of each function and the settings of parameters. For some of the main function groups, calling flows are provided for user's reference.

Appendix

In this manual, some abbreviations are frequently appear, so here we provide a reference for the list of abbreviations.

Note! This manual does not show you how to solve every possible programming problem. Specific questions should be directed to Advantech's application engineers.

Support

support@advantech.com.tw

To use this manual, you should already be familiar with Windows Operating System and at least one of the supported programming environments.

If you have any questions or suggestions about this manual, please contact us at yufeng.zhang@advantech.com.cn, your attention and support are highly appreciated.

Hardware Features

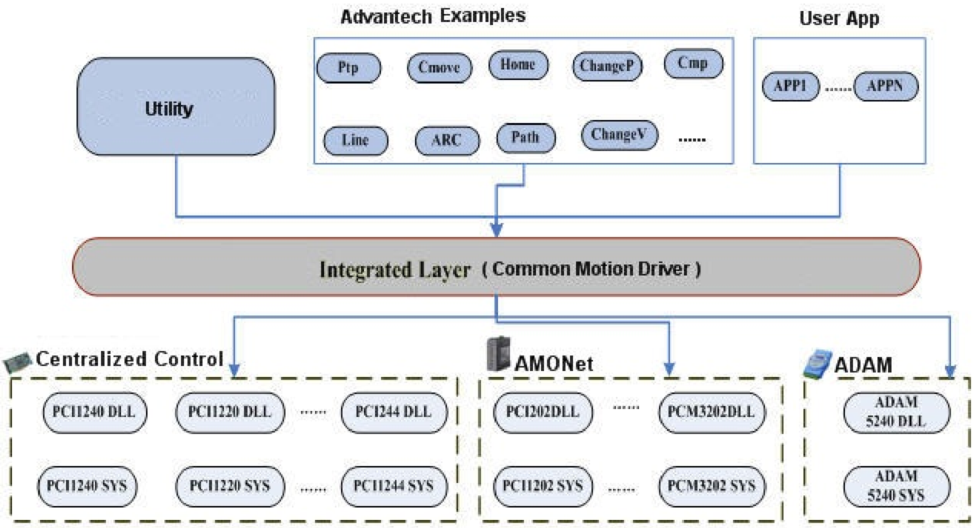
PCI-1220 2-Axis Stepping/Pulse-type Servo Motor Control Card is designed for general-purpose extreme motion applications. The high-speed 2-Axis motion control card for PCI bus simplifies stepping and pulse-type servo motor control, getting the best performance from your motors. The card's intelligent NOVA MCX312-Motion ASIC builds in a variety of motion control functions, such as 2-axis/multi-card linear interpolation, 2-axis circular interpolation, T/S-curve acceleration/deceleration rate and so on, these functions are performed without processor load during driving. For advanced applications, Windows DLL drivers and user-friendly examples are supplied to make the programming easier. Moreover, the free bundled PCI-1220 motion utility makes the configuration and diagnosis process more convenient.

Advantech PCI-1220 provides users with the most requested motor control functions as follows:

- Independent 2-axis motion control
- Hand wheel and jog function
- 2-axis/multi-card linear interpolation function
- 2-axis circular interpolation function
- Continuous interpolation function
- Programmable interrupt conditions
- Programmable T/S-curve acceleration and deceleration
- Up to 4MPPS pulse output for each axis
- Two pulse output types: Up/Down or Pulse/Direction
- Up to 1 MHz encoder input for each axis
- Two encoder pulse input types: A/B phase or Up/Down
- Position management and software limit switch function
- Board ID
- Motion Utility bundled for configuration and diagnosis
- EEPROM

Introduction of Advantech Common Motion Architecture

In order to unify user interfaces of all Advantech motion devices, a new software architecture is designed for all Advantech motion devices which is called "Common Motion Architecture". This architecture defines all user interfaces and all motion functions are implemented, including single axis and multiple axes. This unified programming platform enables us to operate devices in the same manner:



Advantech Common Motion (ACM) Architecture defines three types of operation objects: Device, Axis and Group. Each type has it's own methods, properties and states.

To start single axis motion, you have to follow the following steps:

open device->open one axis of this device->configure instance of this axis->start motion.

All operations can be done by calling corresponding ACM APIs. General calling flows of Device, Axis, Group are specified by Common Motion Architecture, refer to Calling Floww section for details.

Features of Advantech Common Motion Architecturee

Device Number

Device number is composed of 32 bits:

4th byte	3rd byte	2nd H byte	2nd L byte	1st byte
	Master/device board ID(or BaseAddr)		ringg	slave board ID

- 4 th byte:

master/device type ID, (refer to master device type ID table))

- 3 rd & 2 nd H byte:

master/device board ID (or base address)

- 2 nd L byte:

master ring number, used by remote device , use 0 as default value for local device

- 1 st byte:

slave board ID, used by remote device , use 0 as default value for local device

Therefore, the 3 types of device number are:

Local Device Number

4th byte	3rd byte	2nd H byte	2nd L byte	1st byte
master type ID	board ID / BaseAddr		0	0

For example, if one PCI1220's board ID is 1, so it's device number is 0x25001000.

Master Device Number (ex. PCI-1202)

4th byte	3rd byte	2nd H byte	2nd L byte	1st byte
device type ID	Master board ID		0xF	0

Remote Device Number (ex. PCI-1202 + AMAX slave)

4th byte	3rd byte	2nd H byte	2nd L byte	1st byte
device type ID	Master board ID		ring	slave board ID

Handle Operation

All APIs of ACM Architecture are implemented by object handle. The first handle you will get is the device handle which can be got by calling Acm_DevOpen. Through this device handle, you can open this device's all axes and get handles of these axes. If you want to start one interpolation motion, you need to create a group of handles by these axis handles.

Three Types of Property

Properties	Read/Write	Direct access HW	Description
FT_xxxx	Read	No	Feature property
CFG_xxxx	Read/Write	Yes	Configuration property, you'd better not change it after setting. But some can be updated dynamic in order to implement flexible functions.
PAR_xxxx	Read/Write	No	parameters used by software

Multiple Process and Multiple Thread

ACM Architecture supports multiple process and thread programming.

Introduction of PCI-1220 Common Motion Driver

KMDF-Based Driver

PCI-1220 new driver adopted **KMDF** driver type which is of Microsoft new driver architecture: Windows Driver Foundation (WDF). The other driver type supported by WDF is UMDF.

Common Motion Architecture

PCI-1220 supports Advantech Common Motion Architecture. The unified interfaces mainly implement three types of single axis motion: Point to Point, Continue, Home; and two types of multiple axes motion (interpolation drive): Line, Arc.

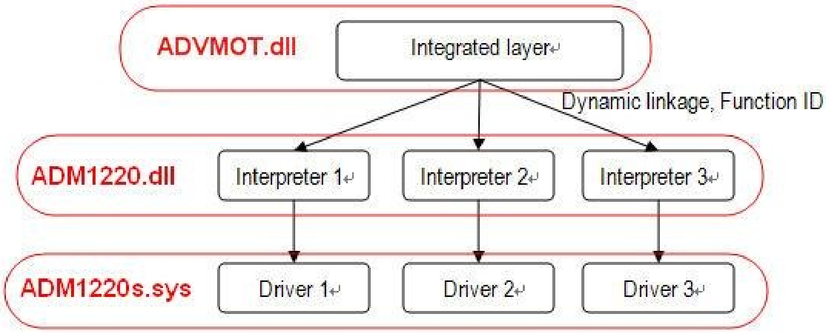
PCI-1220 can do multi-card interpolation too. If multi PCI-1220 card in system, then can set card relations by Utility or SetCardRelation example.

Supporting Platforms

Current version of Driver Supports Windows XP and Vista32.

Note! Before running examples provided by advantech on Vista32, please make sure the .manifest file is together with the .exe file.

PCI-1220's Driver Architecture



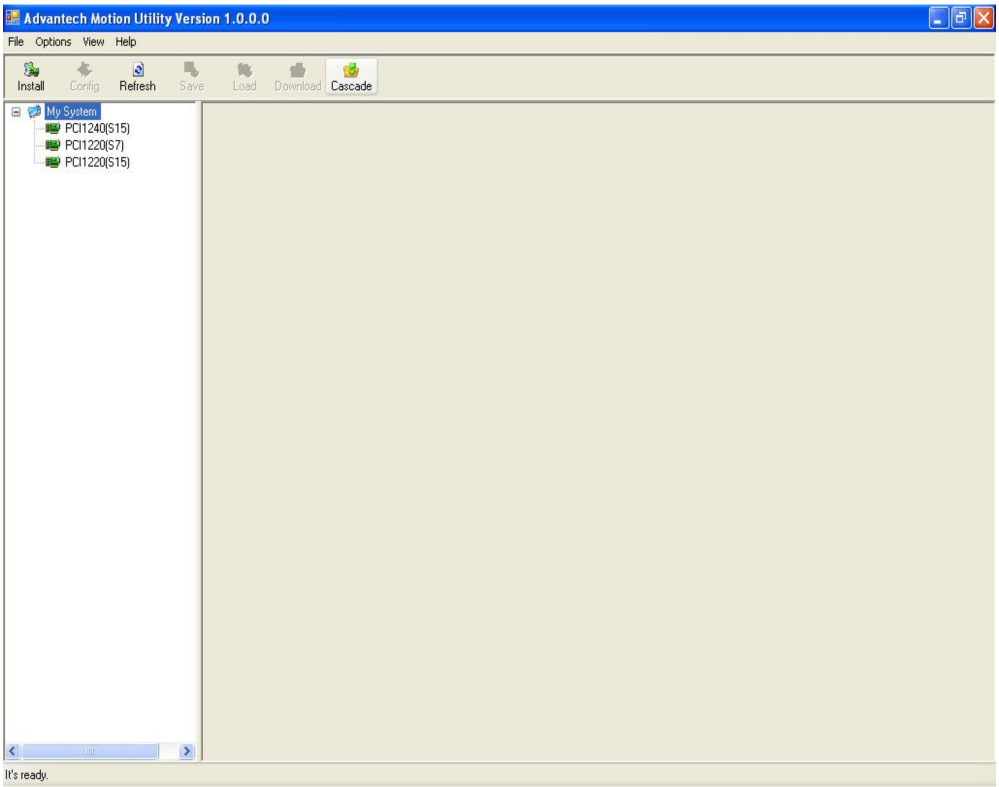
Advantech Motion Utility

Advantech Motion Utility is a user-friendly utility for testing and debugging motion devices. No coding is necessary during the system configuration and testing. It is really helpful to both hardware and software engineers.

The Motion Utility can be found at ***Start->Program Files->Advantech Automation->Motion Utility***. If a device is already plugged in with driver installed, the device will be listed in the left view when the Motion Utility is opened.

User Interface

The following picture shows the start up window of Advantech Motion Utility on Windows XP:


Advantech Motion Utility Version 1.0.0.0
File Options View Help
Install Config Refresh Save Load Download Cascade
My System
PCI1240(S15)
PCI1220(S7)
PCI1220(S15)
It's ready.

Device Format: Device Name (SBoardID), eg. PCI1220(SE).

When multiple devices of the same type are plugged in, each device should be specified with an exclusive **BoardID** which is set by the switch on the board.

Menu bar

File

Exit

Exit the Advantech Motion Utility

Options

Install Device

Re-scan hardware. If new device is found, re-install the device driver.

Configure

Configure the PCM series hardware information

Refresh

Scan the devices plugged in the PC and show them in the left tree view

Remove Device

Remove the specified device

View

Toolbar
Display or hide the tool bar.

Status Bar
Display or hide the status bar at the bottom.

Help

About
Application name, version, and copyright statement.

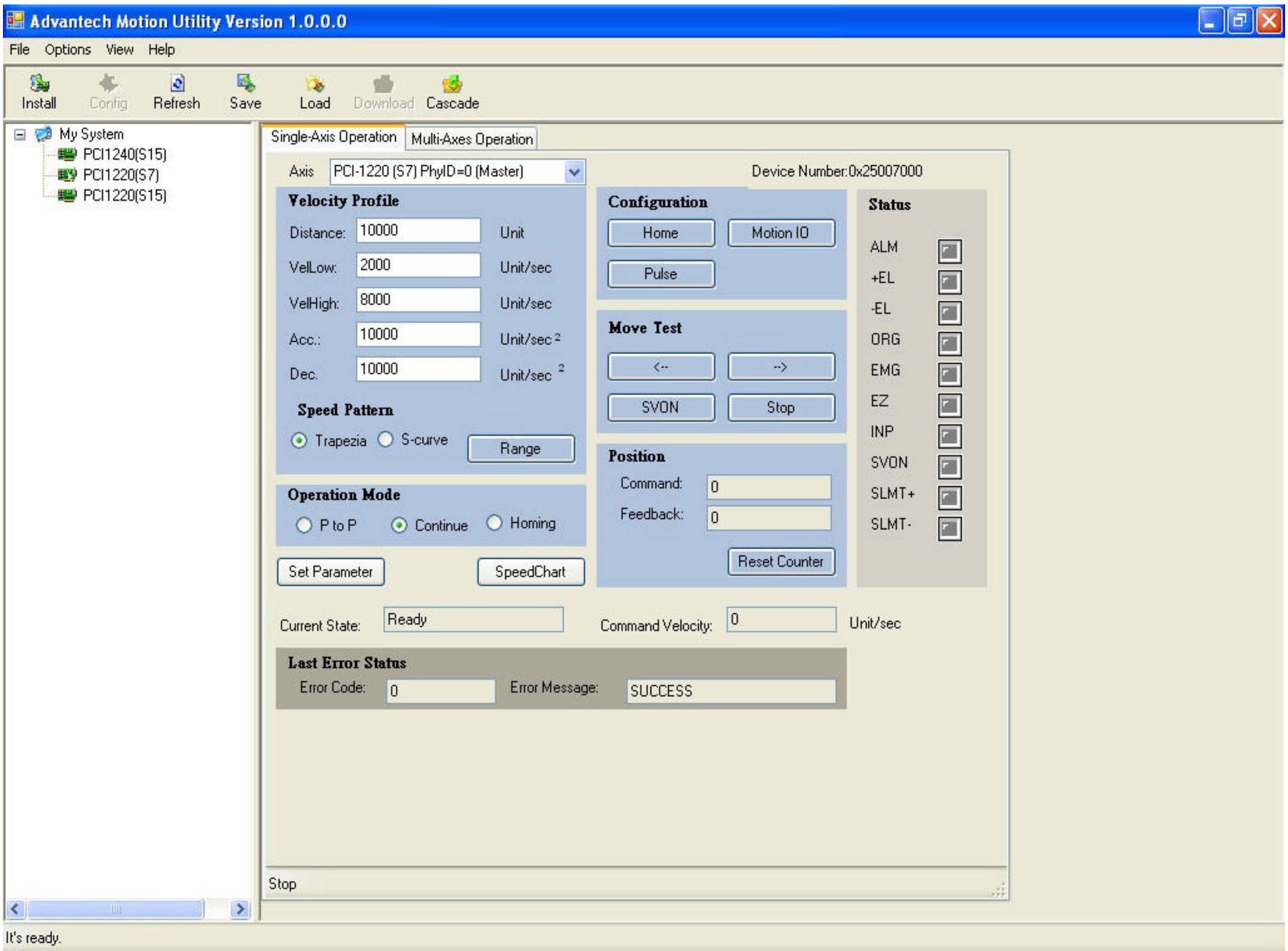
Up-ToDate on the Web
Link to: http://www.advantech.com.tw/

Left tree ICON status

	Installed device.
	Active device.

Single-Axis Operation

If PCI-1220 has been plugged in the PC, the device will be shown in the left tree view when Advantech Motion Utility is opened. Select the device, and the testing dialog will be shown in the right pane:



Axis: PCI-1220 (S7) PhyID=0 (Master)

Device Number:0x25007000

Distance: 10000 Unit

VelLow: 2000 Unit/sec

VelHigh: 8000 Unit/sec

Acc.: 10000 Unit/sec ^{2}

Dec.: 10000 Unit/sec ^{2}

● Trapezia ○ S-curve

○ P to P ● Continue ○ Homing

Command: 0

Feedback: 0

Current State: Ready

Command Velocity: 0 Unit/sec

Error Code: 0 Error Message: SUCCESS

Note:

*The **Device Number :0x2000e000** is very important for you to develop your own application, since this value must be transferred to **Acm_DeviceOpen** to get device handle. More details, please refer to "Calling Flow".*

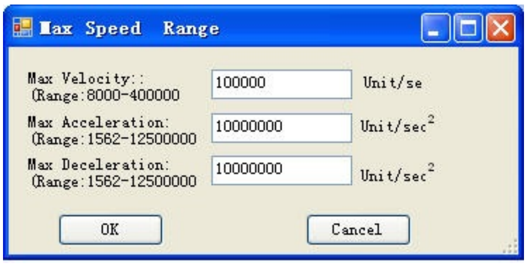
Testing Steps

Step 1

Configure the device. The **home**, **motor** signal and **pulse** input/output modes can all be configured. We can also load the configuration file to the hardware instead of configuring one by one. Please refer to configure motion for details.

Step 2

Select the Velocity profile to set velocity parameters: **Distance**, **VelLow**, **VelHigh**, **Acc**, **Dec**. These parameters' range can be configured by clicking [**View/Set Range**].



Max Velocity::
(Range:8000-400000 100000 Unit/se

Max Acceleration:
(Range:1562-12500000 10000000 Unit/sec ^{2}

Max Deceleration:
(Range:1562-12500000 10000000 Unit/sec ^{2}

After setting the velocity parameters, we have to select the speed patterns: **trapezia** or **s-curve**.

For **trapezia** , the **Acc** parameter means acceleration. For **s-curve,** the **Acc** parameter means max acceleration.
For **trapezia** , the **Dec** parameter means deceleration. For **s-curve,** the **Dec** parameter means max deceleration.

Step 3

Click [Set] to set velocity parameters and speed pattern.

Step 4

Select one kind of operation mode: P to P, Continue, Homing.

Step 5

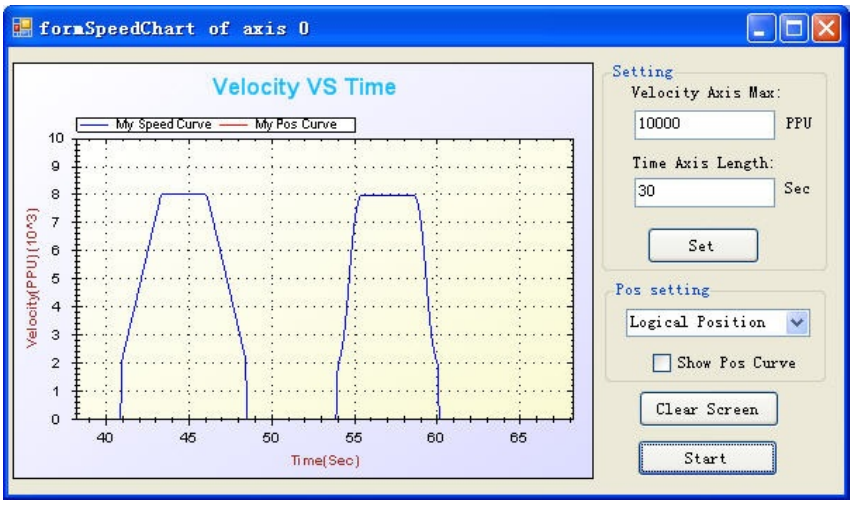
If servo motor is connected, click [**SVON**] to start the motor. If the motor is already started, click [**SVON**] to stop it.

Step 6

Start moving. click [**<--**] or [**-->**] button to move the motor backward or forward.

Step 7

View the moving curve. click [**SpeedChart**] to view speed curve or position curve.



Setting

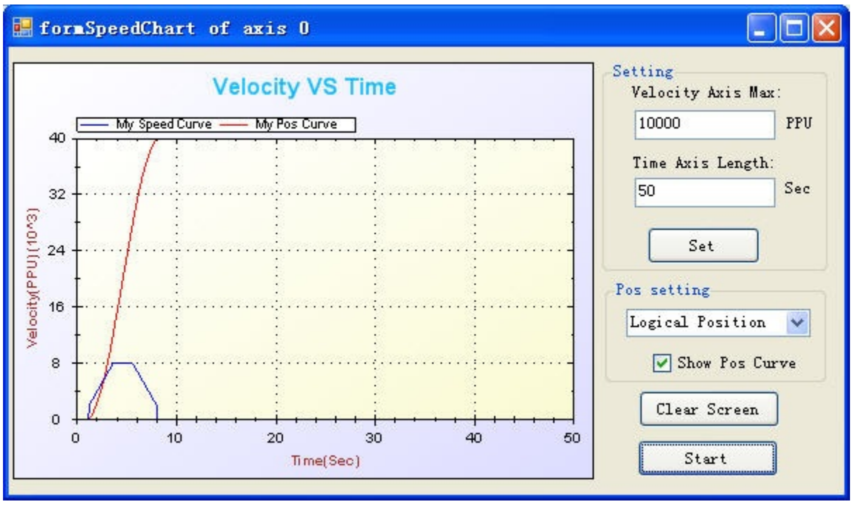
Velocity Axis Max: 10000 PPU

Time Axis Length: 30 Sec

Logical Position

□ Show Pos Curve

If [**Show pos Curve**] is checked, position curve will also be shown in the window:



Setting

Velocity Axis Max: 10000 PPU

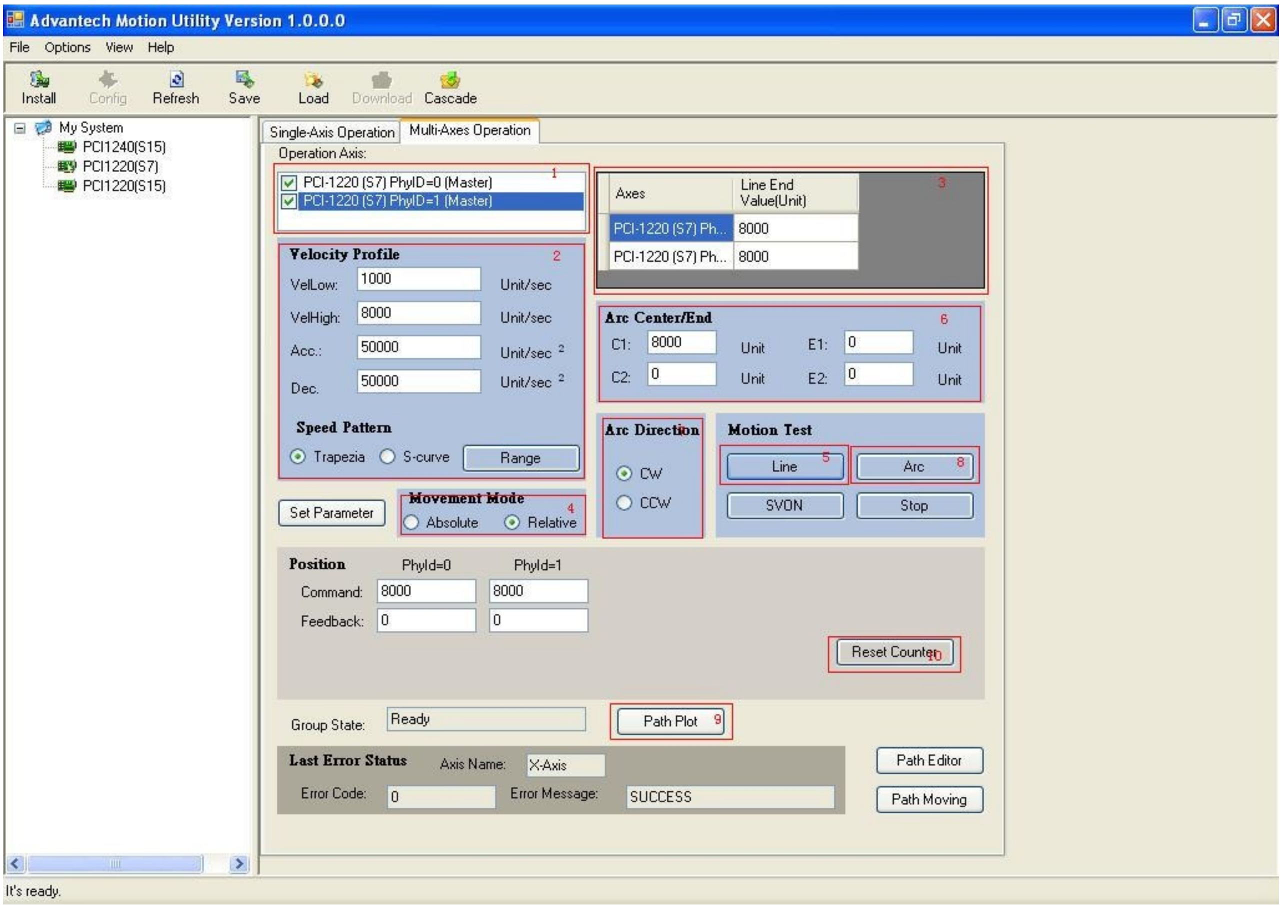
Time Axis Length: 50 Sec

Logical Position

☑ Show Pos Curve

Multi-Axes Operation

Click "**Multi-Axes Operation**" tab to view the Multi-Axes operation dialog.



Operation Axis:

☑ PCI-1220 (S7) PhyID=0 (Master)
☑ PCI-1220 (S7) PhyID=1 (Master)

Axes	Line End Value(Unit)
PCI-1220 (S7) Ph...	8000
PCI-1220 (S7) Ph...	8000

Velocity Profile

VelLow: 1000 Unit/sec
VelHigh: 8000 Unit/sec
Acc.: 50000 Unit/sec 2
Dec. 50000 Unit/sec 2

Speed Pattern: ● Trapezia ○ S-curve

Movement Mode: ○ Absolute ● Relative

Arc Center/End

C1: 8000 Unit E1: 0 Unit
C2: 0 Unit E2: 0 Unit

Arc Direction: ● CW ○ CCW

Position	PhyId=0	PhyId=1
Command:	8000	8000
Feedback:	0	0

Group State: Ready

Last Error Status

Axis Name: X-Axis
Error Code: 0 Error Message: SUCCESS

Testing Steps

The numbers marked in the highlighted areas above is correspond to the orders of each step below.

Preset

Configure the device. The **home**, **motor** signal and **pulse** input/output modes can all be configured. We can also load the configuration file to the hardware instead of configuring devices one by one. Please refer to configure motion for details.

Step 1

Select the operation axis to operate. For line motion, support any 2 or Multi axes, for arc motion, only support 2 axes.

Step 2

Select the velocity profile. You must select the velocity profile: **Trapezia** or **S-curve**; **VelLow**, **VelHigh**, **Acc**, **Dec**. Then click [**Set Parameter]** button**.**

Step 3

Set the line motion distance. Input positive value, moving forward. Input negative value, moving backward.

Step 4

Select movement mode: **Absolute** or **Relative**.

Step 5

Start line motion. Axis will move the motor backward or forward according to **Line End** settings.

Step 6

Set the arc motion center and end position.

Step 7

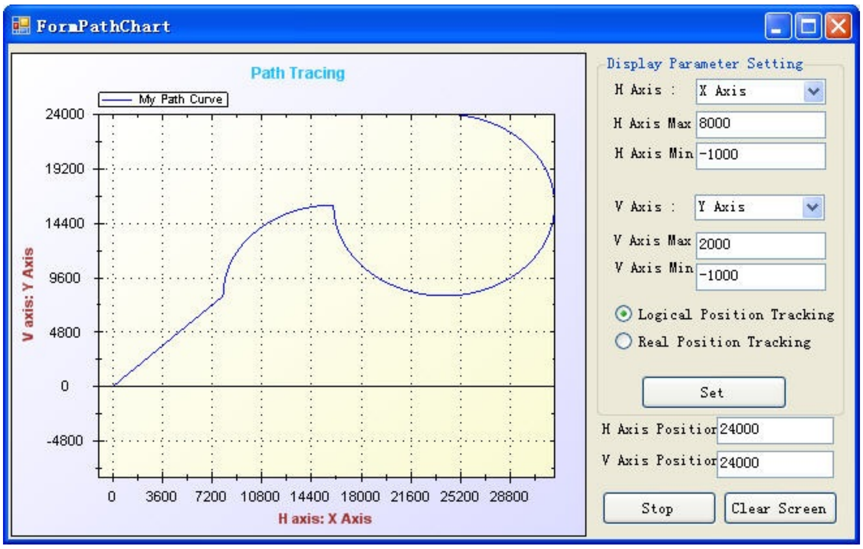
Select arc direction: **CW**(clockwise) or **CCW**(Counterclockwise).

Step 8

Start arc motion. Axis will move the motor backward or forward according to **Arc Center/End** settings. **Movement Mode** also will affect arc motion which can be seen through [**Path Plot**].

Step 9

View the motion path. Click [**Path Plot**] button to view the motion path:



Display Parameter Setting

H Axis : X Axis

H Axis Max 8000

H Axis Min -1000

V Axis : Y Axis

V Axis Max 2000

V Axis Min -1000

● Logical Position Tracking

○ Real Position Tracking

H Axis Positior 24000

V Axis Positior 24000

Step 10

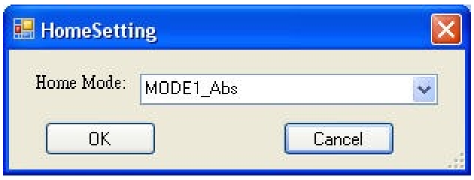
Reset the coordinate to zero.

Note:

S-Curve acceleration or deceleration is not supported by Arc and Path (continuous interpolation).

Home Configuration

In the single axis motion test dialog click [**Home**] button and the Home Configuration Dialog will popup:



Home Mode: MODE1_Abs

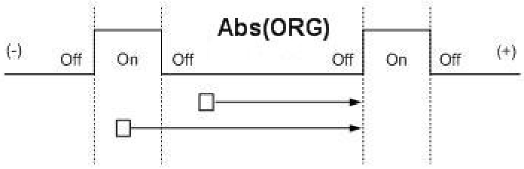
11 typical home return modes are provided by 1 axis motion mode:

- Home Abs
- Home Lmt
- Home Ref
- Home Abs Ref
- Home Abs NegRef
- Home Lmt Ref
- Home AbsSearch
- Home LmtSearch
- Home AbsSearch Ref
- Home AbsSearch NegRef
- Home LmtSearch Ref

<

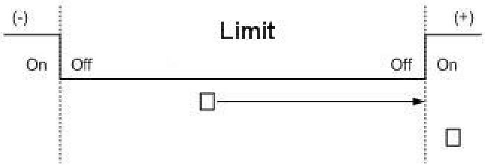
Home Abs **:ORG only, Move (Dir)->touch ORG->Stop**
e.g. Dir: Positive Direction; Abs Logic: Active High

Back to list



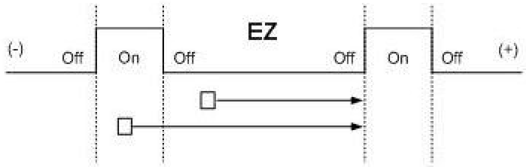
Home Lmt **:EL only, Move (Dir)->touch EL->Stop**
e.g. Dir: Positive Direction; Lmt Logic: Active High

Back to list



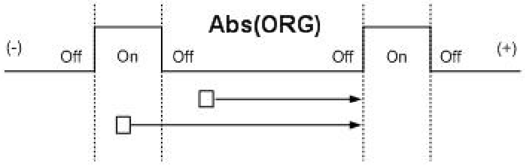
Home Ref **:EZ Only, Move (Dir)->touch EZ->Stop**
e.g. Dir: Positive Direction; EZ Logic: Active High

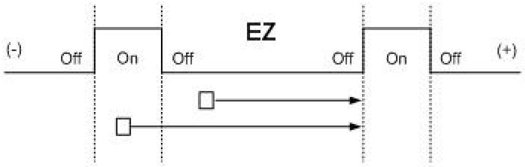
Back to list



Home Abs Ref **:ORG + EZ, Move (Dir)->touch ORG->Stop->Move (Dir)->touch EZ->Stop**
e.g. Dir: Positive Direction; Abs Logic: Active High; EZ Logic: Active High

Back to list

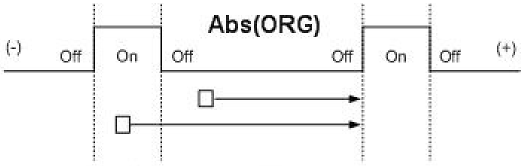


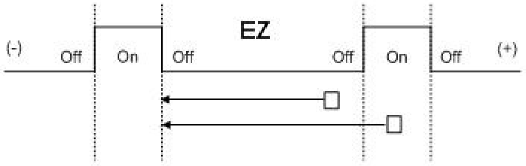


Home Abs NegRef **:ORG + NegEZ, Move (Dir)->touch ORG->Stop->Move (-Dir)->touch EZ->Stop**

e.g. Dir: Positive Direction; Abs Logic: Active High; EZ Logic: Active High

Back to list

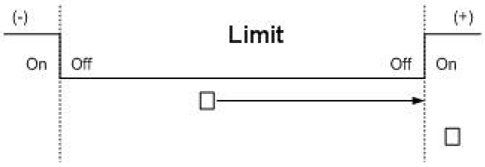


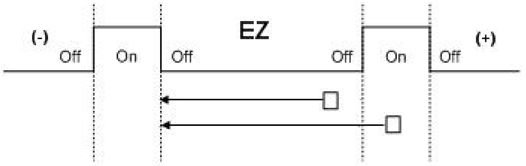


Home Lmt Ref **:EL + NegEZ, Move (Dir)->touch EL->Stop->Move (-Dir)->touch EZ->Stop**

e.g. Dir: Positive Direction; Lmt Logic: Active High; EZ Logic: Active High

Back to list

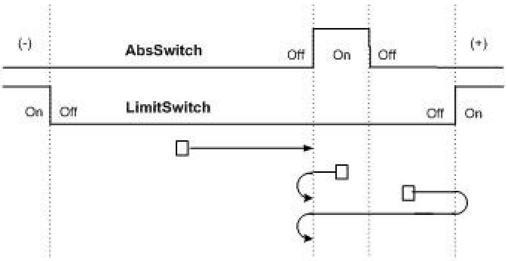




Home AbsSearch **:Search ORG only, Move (Dir)->Search ORG->Stop**

e.g. Dir: Positive Direction; Abs Logic: Active High; Lmt Logic: Active High

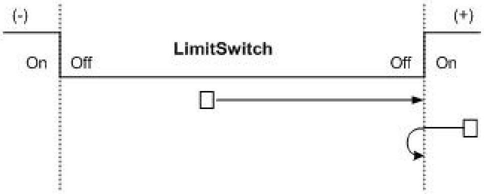
Back to list



Home LmtSearch **:Search EL only, Move (Dir)->Search EL->Stop**

e.g. Dir: Positive Direction; Lmt Logic: Active High

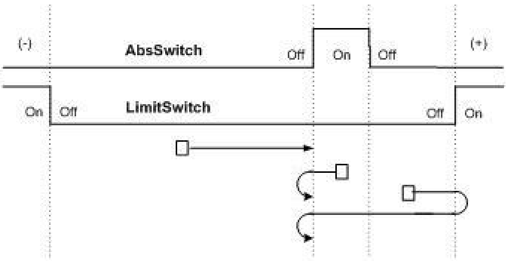
Back to list

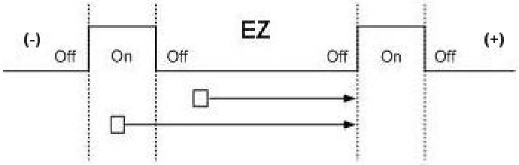


Home AbsSearch Ref **:Search ORG + EZ, Move (Dir)->Search ORG->Stop->Move (Dir)->touch EZ->Stop**

e.g. Dir: Positive Direction; Abs Logic: Active High; Lmt Logic: Active High

Back to list

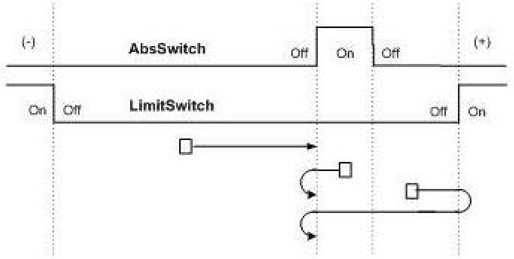


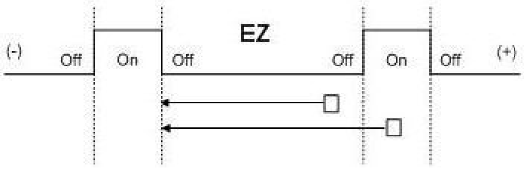


Home AbsSearch NegRef **:Search ORG + NegEZ, Move (Dir)->Search ORG->Stop->Move (-Dir)->touch EZ->Stop**

e.g. Dir: Positive Direction; Abs Logic: Active High; Lmt Logic: Active High; EZ Logic: Active High

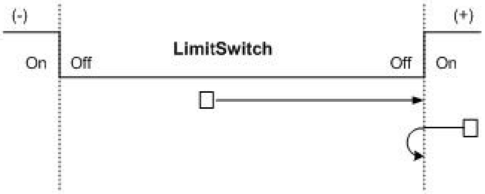
Back to list

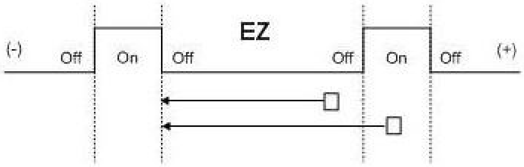




Home LmtSearch Ref **:Search EL + NegEZ, Move (Dir)->Search EL->Stop->Move (-Dir)->touch EZ->Stop** ***Back to list***

e.g. Dir: Positive Direction; Lmt Logic: Active High; EZ Logic: Active High





Notes:

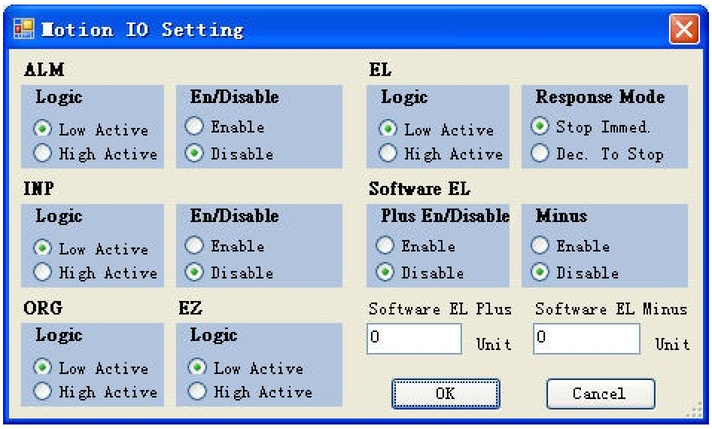
1. **AbsSwitch(ORG)** signal is IN3 signal;

2. **EZ** signal is IN0 signal;

3. If **HomeMode** is set to be Home LmtSearch or Home LmtSearch Ref, EL+ signal needs to be connected to IN1 and EL- signal needs to be connected to IN2.

Motion IO Setting

In this configuration dialog, user can configure ALM, INP, ORG, EZ, EL, and SEL. Click the radio buttons to configure signal settings and then click the **OK**.



ALM
Logic: ● Low Active ○ High Active
En/Disable: ○ Enable ● Disable

EL
Logic: ● Low Active ○ High Active
Response Mode: ● Stop Immed. ○ Dec. To Stop

INP
Logic: ● Low Active ○ High Active
En/Disable: ○ Enable ● Disable

Software EL
Plus En/Disable: ○ Enable ● Disable
Minus: ○ Enable ● Disable

ORG
Logic: ● Low Active ○ High Active

EZ
Logic: ● Low Active ○ High Active

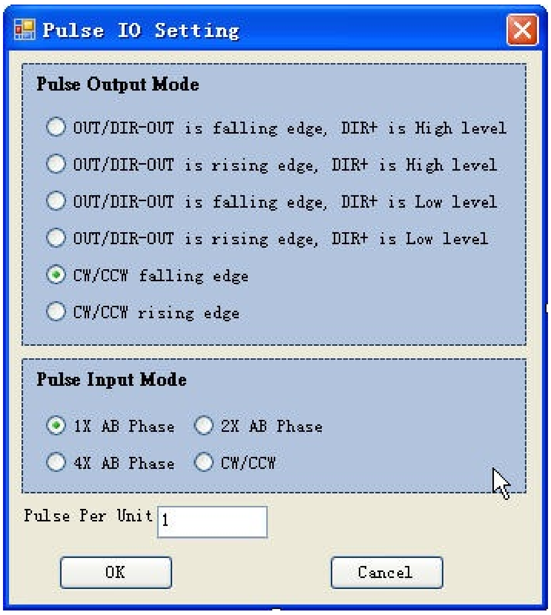
Software EL Plus: 0 Unit
Software EL Minus: 0 Unit

Abbreviations in the above dialog:

EL	End Limit, indicating whether the limit of motion is in plus direction or minus direction
ORG	Home signal input, indicating the origin of the system
ALM	Servo Alarm Signal
INP	Servo In Position Signal
EZ	Encode Z phase
SEL	Software limit.

Pulse Configuration

This dialog is used to configure pulse output mode and pulse input mode. PCI1220 supports six types of output mode and four types of input mode.



Pulse Output Mode

○ OUT/DIR-OUT is falling edge, DIR+ is High level
○ OUT/DIR-OUT is rising edge, DIR+ is High level
○ OUT/DIR-OUT is falling edge, DIR+ is Low level
○ OUT/DIR-OUT is rising edge, DIR+ is Low level
● CW/CCW falling edge
○ CW/CCW rising edge

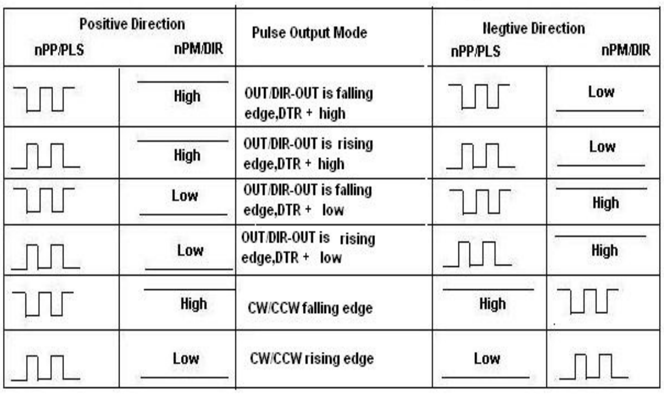
Pulse Input Mode

● 1X AB Phase ○ 2X AB Phase
○ 4X AB Phase ○ CW/CCW

Pulse Per Unit 1

Pulse Output Mode

For the above six output modes, refer to the following chapter:



Positive Direction		Pulse Output Mode	Negtive Direction	
nPP/PLS	nPM/DIR		nPP/PLS	nPM/DIR
⊔⊔	High	OUT/DIR-OUT is falling edge,DTR + high	⊔⊔	Low
⊓⊓	High	OUT/DIR-OUT is rising edge,DTR + high	⊓⊓	Low
⊔⊔	Low	OUT/DIR-OUT is falling edge,DTR + low	⊔⊔	High
⊓⊓	Low	OUT/DIR-OUT is rising edge,DTR + low	⊓⊓	High
⊔⊔	High	CW/CCW falling edge	High	⊔⊔
⊓⊓	Low	CW/CCW rising edge	Low	⊓⊓

Pulse Input Mode

1x AB phase 1 pulse of feedback

2x AB phase 2 pulses of feedback.

4x AB phase 4 pulses of feedback.

CW/CCW nECA/PPIN is count up input and nECB/PMIN is count down input. Counting starts on positive pulse rising edge.

Pulse Per Unit (PPU)

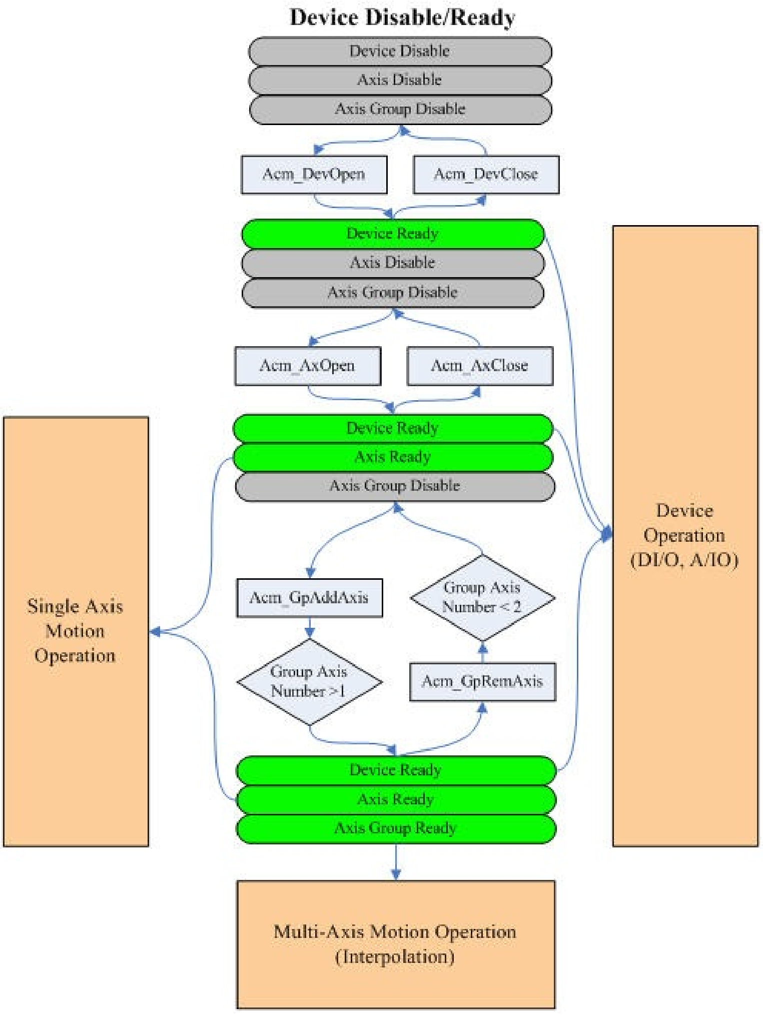
You can set different PPU for different axes if the axes are connected with different motors. This can mask the different precision of different motors. For example:

X axis's motor: one pulse equals 1 millimeter; Y axis's motor: one pulse equals 0.5 millimeter. Then you can set X axis' PPU to be 1 and set Y axis' PPU to be 2. By doing that, we get the same physical units for X axis and Y axis.

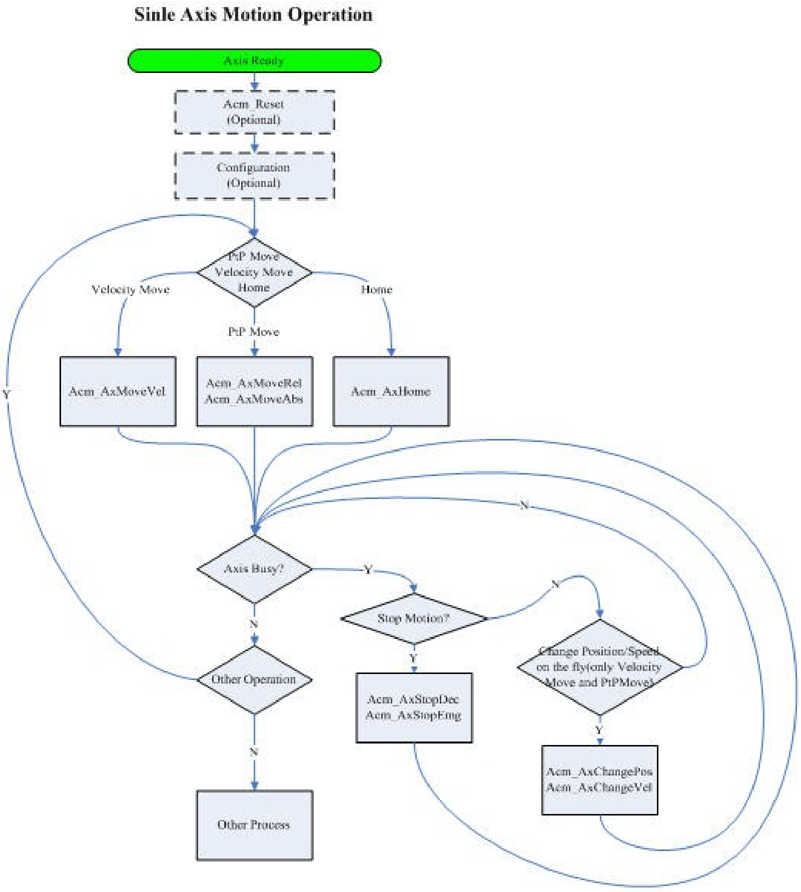
PPU value can only be integer greater than or equals to 1.

Calling Flow

Basic Flow

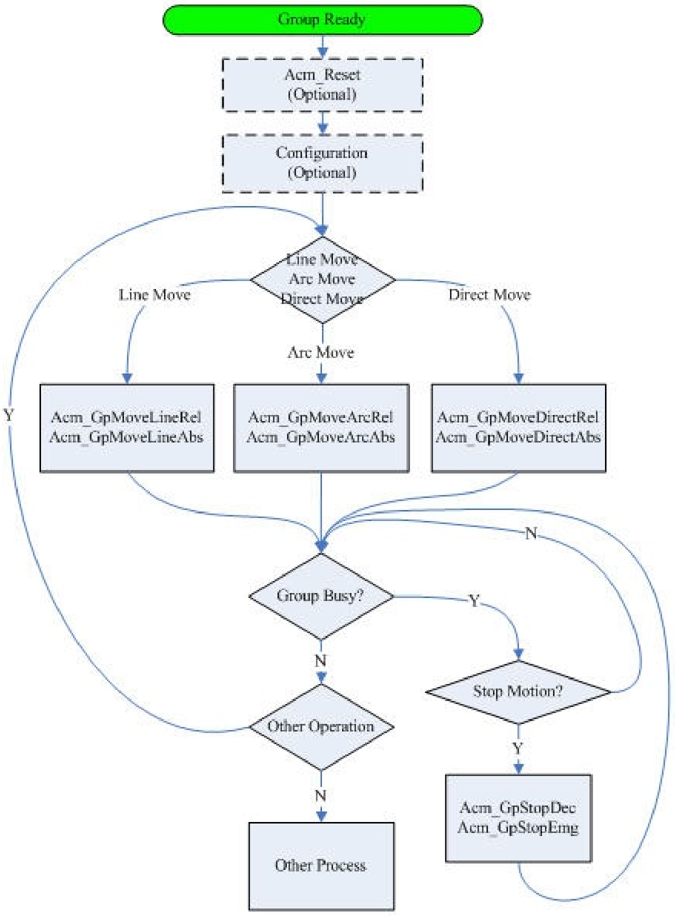


Single Axis' FLow


Sinle Axis Motion Operation
Axis Ready
Acm_Reset (Optional)
Configuration (Optional)
PtP Move Velocity Move Home
Velocity Move
Home
PtP Move
Acm_AxMoveVel
Acm_AxMoveRel Acm_AxMoveAbs
Acm_AxHome
Y
N
Axis Busy?
Y
N
Stop Motion?
N
Change Position/Speed on the fly(only Velocity Move and PtPMove)
Y
Other Operation
Acm_AxStopDec Acm_AxStopEmg
Y
Acm_AxChangePos Acm_AxChangeVel
N
Other Process

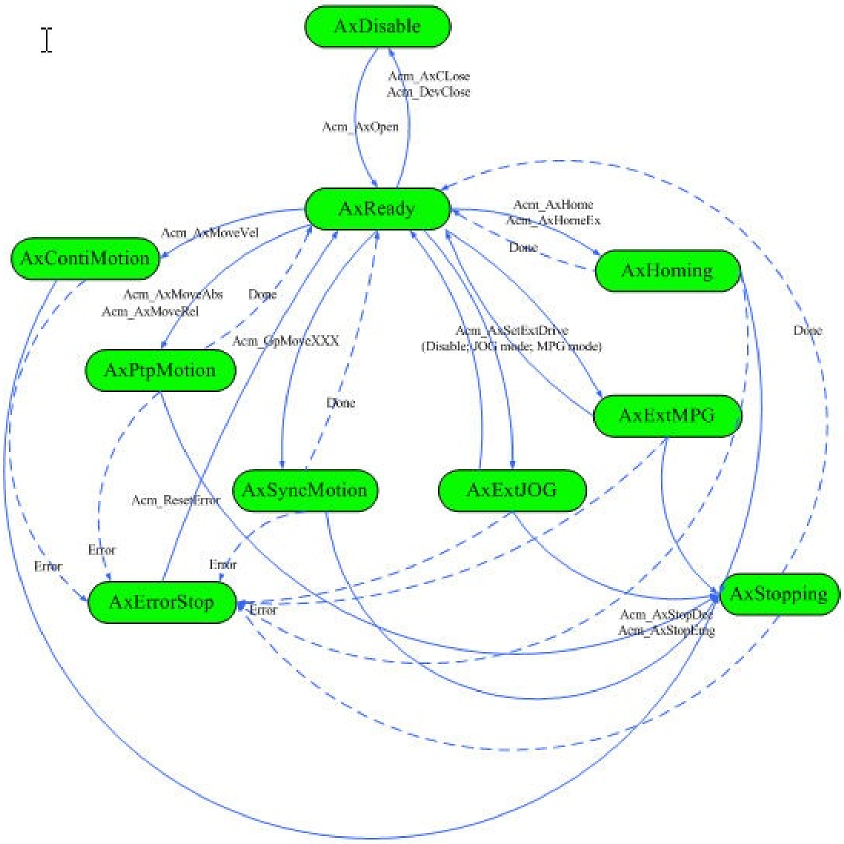
Multiple Axes' General Flow

Multi-Axes Motion Operation

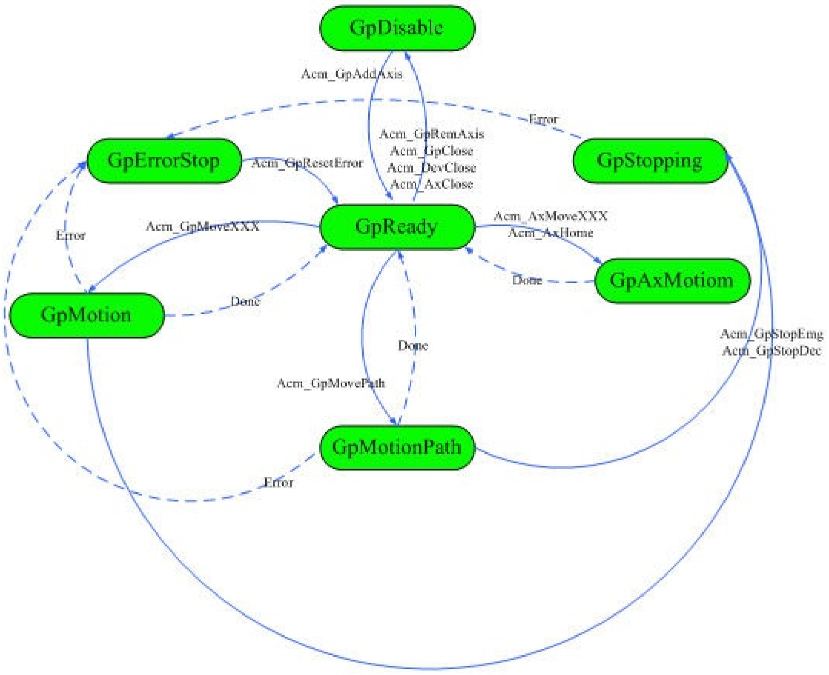


State Machine

Single Axis' State Machine



Group's State Machine

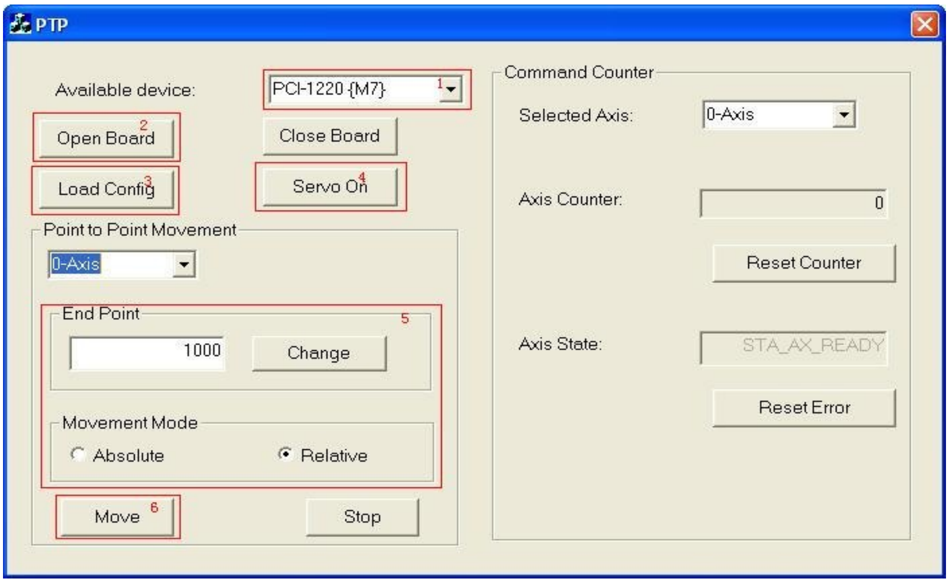


See Also:

Acm_AxGetState, Acm_GpGetState

Point to Point Demo

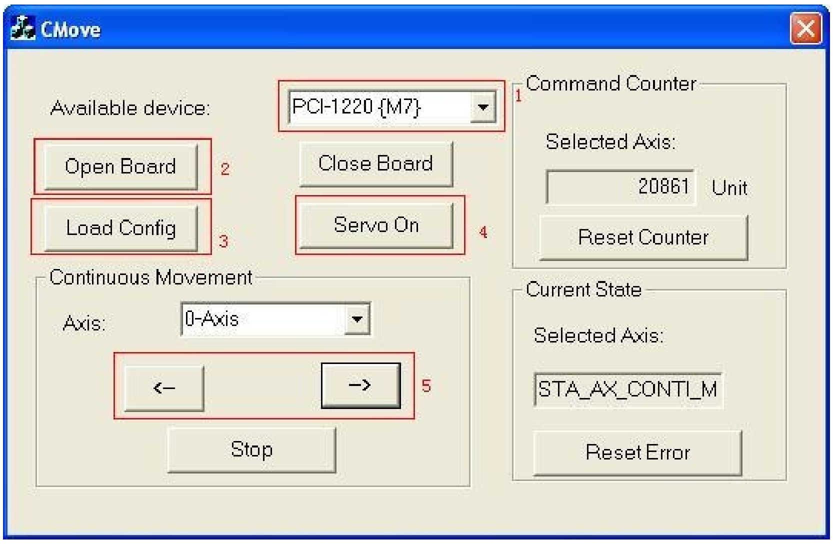
This example demonstrates how to use the ACM API to control one axis point to point motion.



1. Select a Device Number from the available device list;
2. Open device and its axes;
3. If configuration file is available, you can load this file to configure this device;
4. Click [**Servo On**] button to open servo motor after it is connected;
5. Set the end point for every axis and select **Movement Mode**: absolute or relative;
6. Click [**Move**] button to start motion.

Continuous Motion Demo

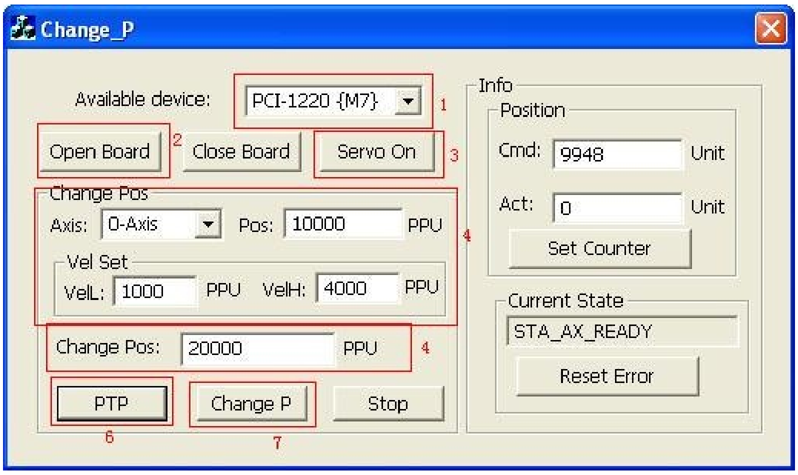
This example demonstrates how to use the ACM API to control one axis continuous motion.



1. Select a Device Number from the available device list;
2. Open device and its axes;
3. If configuration file is available, you can load this file to configure this device;
4. Click [**Servo On**] button to open servo motor after it is connected;
5. Click [**<--**] or [**-->**] below each Axis to to start continuous motion.

Change Position on the Fly Demo

This example demonstrates how to change one axis motion position on the fly.



Available device: PCI-1220 {M7}

Change Pos

Axis: 0-Axis Pos: 10000 PPU

Vel Set

VelL: 1000 PPU VelH: 4000 PPU

Change Pos: 20000 PPU

Info

Position

Cmd: 9948 Unit

Act: 0 Unit

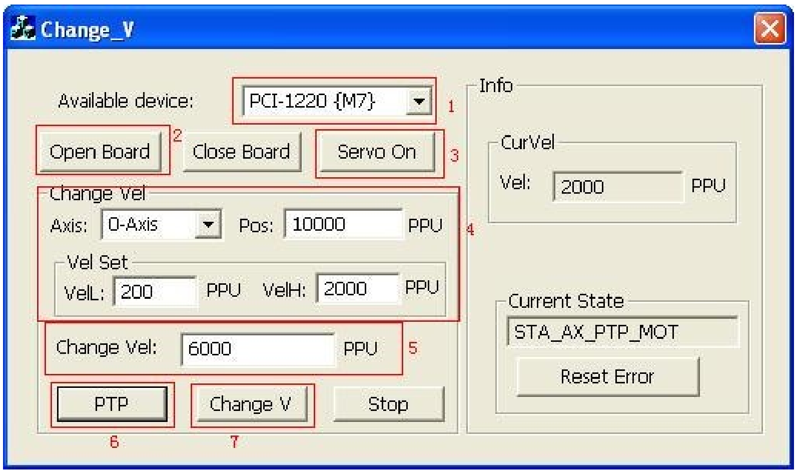
Current State

STA_AX_READY

1. Select a Device Number from the available device list;
2. Open device and its axes;
3. Click [**Servo On**] button to open servo motor after the servo motor is connected;
4. Select the axis you want to control motion. Enter position, low velocity, high velocity parameters.
5. Click [**PTP**] button to start motion.
6. Before the end of this motion, you can set a new position value;
7. Click [**Chang P**] button to change the position to the new one.

Change Velocity on the Fly Demo

This example demonstrates how to change one axis motion velocity on the fly.



Available device: PCI-1220 {M7}

Axis: 0-Axis Pos: 10000 PPU

VelL: 200 PPU VelH: 2000 PPU

Change Vel: 6000 PPU

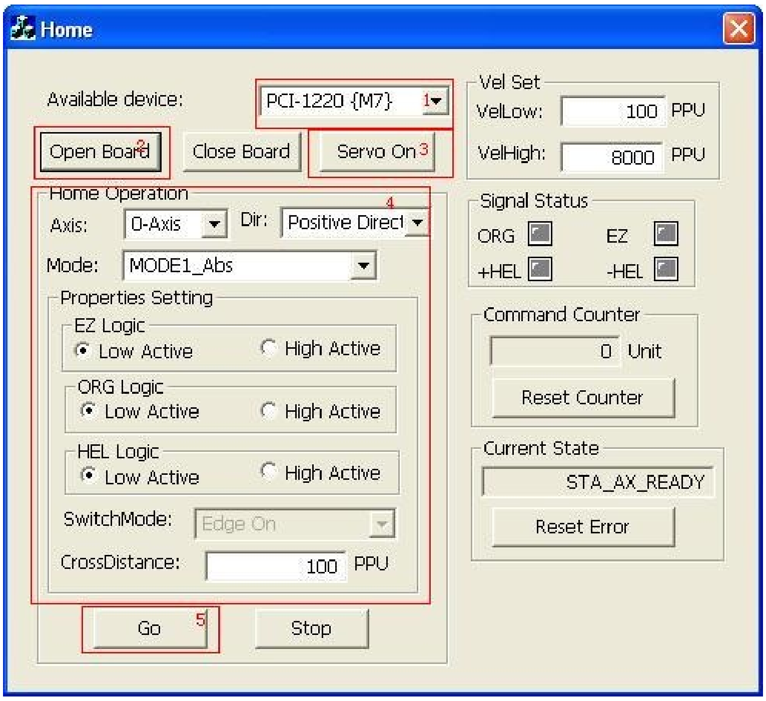
Vel: 2000 PPU

Current State: STA_AX_PTP_MOT

1. Select a Device Number from the available device list;
2. Open device and its axes;
3. Click [**Servo On**] button to open servo motor after servo motor is connected;
4. Select the axis you want to control motion. Enter position, low velocity, high velocity parameters.
5. Click [**PTP**] button to start motion.
6. Before the end of this motion, you can set a new velocity value;
7. Click [**Chang V**] button to change the velocity to the new one.

Home Demo

This example demonstrates how to use the home function. 11 typical home return modes are supported. Call Acm_AxHomeEx API to develop more flexible home function.



Available device: PCI-1220 {M7}

Vel Set
VelLow: 100 PPU
VelHigh: 8000 PPU

Home Operation
Axis: 0-Axis
Dir: Positive Direct
Mode: MODE1_Abs

Properties Setting
EZ Logic: ● Low Active ○ High Active
ORG Logic: ● Low Active ○ High Active
HEL Logic: ● Low Active ○ High Active
SwitchMode: Edge On
CrossDistance: 100 PPU

Signal Status
ORG □ EZ □
+HEL □ -HEL □

Command Counter
0 Unit

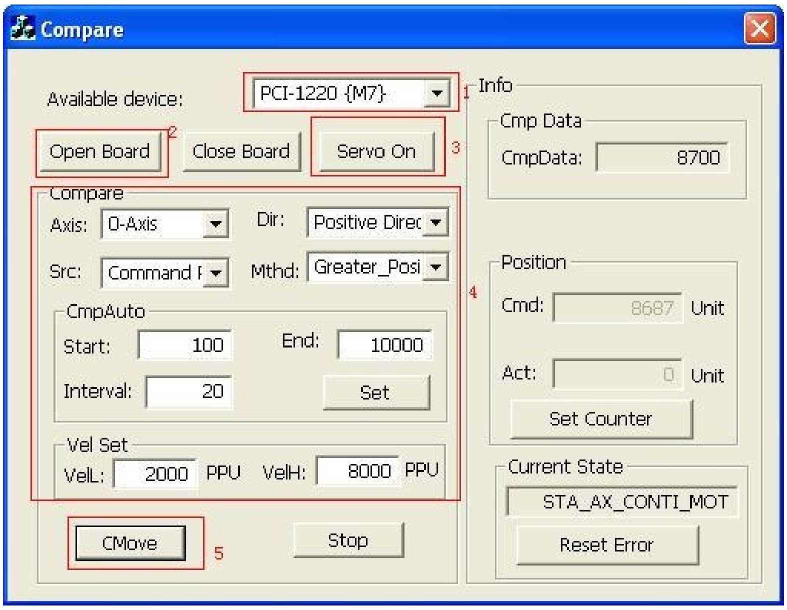
Current State
STA_AX_READY

1. Select a Device Number from the available device list;
2. Open device and its axes;
3. Click [**Servo On**] button to open servo motor after it is connected;
4. Select the axis you want to control home motion. Then set the direction, home mode, EZ logic, ORG logic, HEL Logic and CrossDistance value.
5. Click [**Go**] button to start home motion.

Refer to Home Config for more details about home mode.

Compare Demo

This example demonstrates how to use the compare function



Available device: PCI-1220 {M7}

Axis: 0-Axis   Dir: Positive Direc

Src: Command F   Mthd: Greater_Posi

Start: 100   End: 10000

Interval: 20

VelL: 2000 PPU   VelH: 8000 PPU

CmpData: 8700

Cmd: 8687 Unit

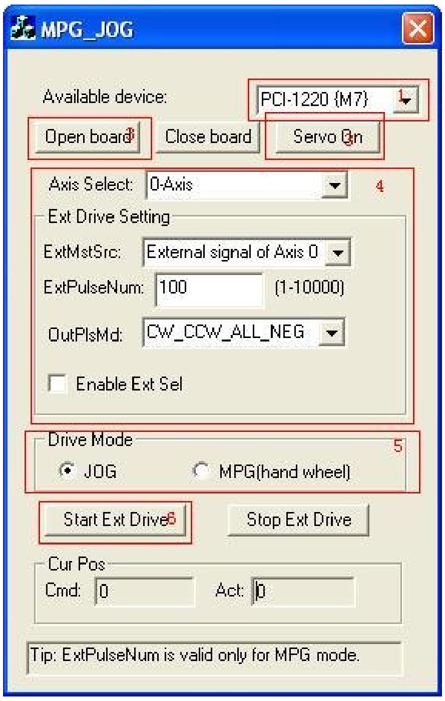
Act: 0 Unit

STA_AX_CONTI_MOT

1. Select a Device Number from the available device list;
2. Open device and its axes;
3. Click [**Servo On**] button to open servo motor after it is connected;
4. Select one operation axis. Set direction, compare source, compare method and compare data.
   In **CmpAuto** Group box you can set the *Start* compare data, *compare interval* and the *End* compare data. For example, if **Start** is set to 100, **End** is set to 10000, **Interval** is set to 20, then the compare data will be 100, 120, 140......10000. Before you click [**Set**] button, you'd better make sure these compare data are valid. Refer to Acm_AxSetCmpAuto, Acm_AxSetCmpData, Acm_AxSetCmpTable to verify the data.
5. Click [**CMove**] button to start continuous motion with compare function.

External Drive Demo

This example demonstrates how to start external drive operation on the specified device and axis. PCI1220 supports 2 external control mode, one is JOG mode and the other is MPG(Hand Wheel) mode.



Available device: PCI-1220 (M7)

Axis Select: 0-Axis

Ext Drive Setting

ExtMstSrc: External signal of Axis 0

ExtPulseNum: 100 (1-10000)

OutPlsMd: CW_CCW_ALL_NEG

□ Enable Ext Sel

Drive Mode

● JOG ○ MPG(hand wheel)

Cur Pos

Cmd: 0 Act: 0

Tip: ExtPulseNum is valid only for MPG mode.

1. Select a Device Number from the available device list;

2. Open device and open its axes;

3. Click [**Servo On**] button to open servo motor after it is connected;

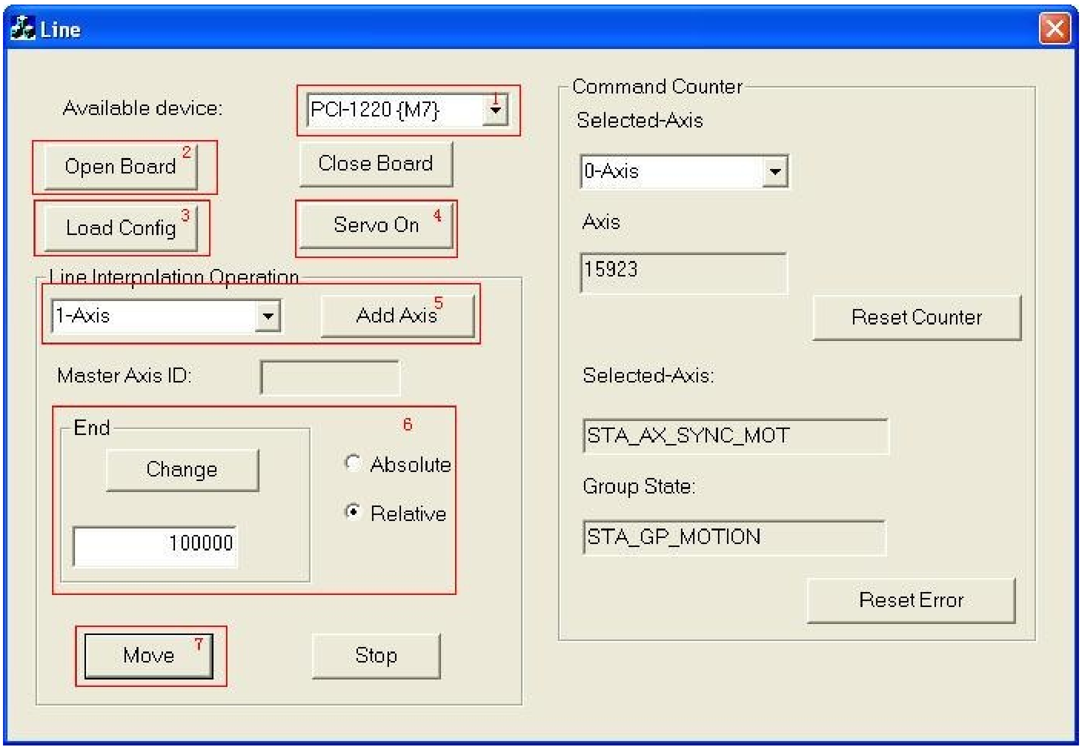
4. Select one operation axis. Set properties: CFG_AxExtMasterSrc ,CFG_AxExtSelEnable, CFG_AxExtPulseNum. If **Enable Ext Sel** is checked, the **ExtMstSrc** can only be **X Axis** or **Y Axis**.

5. Select one external drive mode: JOG or MPG. About the difference between JOG and MPG, please refer to Acm_AxSetExtDrive.

6. Click [**Enable Ext Drive**] button to enable external drive mode. After this, the motion will start after the input signal from nEXOP+ / nEXOP- is active.

Line Demo

This example demonstrates how to use the ACM API to control an interpolation group's line motion.



Line

Available device: PCI-1220 {M7}

Line Interpolation Operation: 1-Axis

Master Axis ID:

End: 100000

○ Absolute

● Relative

Command Counter

Selected-Axis: 0-Axis

Axis: 15923

Selected-Axis: STA_AX_SYNC_MOT

Group State: STA_GP_MOTION

1. Select a Device Number from the available device list;

2. Open device and its axes;

3. If configure file is available, you can load this file to configure this device;

4. Click [**Servo On**] button to open servo motor after it is connected;

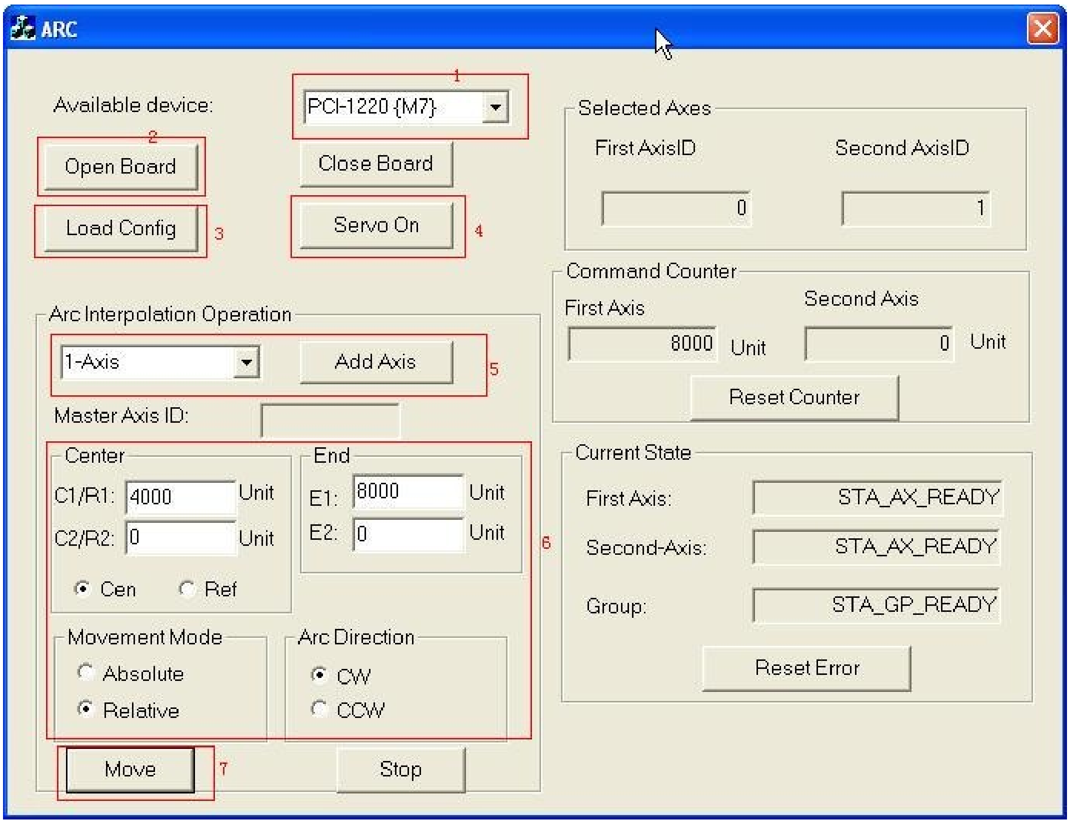
5. Add axis to a group. For line motion, if there are slave devices on, you can add two axes on the slave or master device or the first axis on the slave devices.

6. Set end points for the axes in the group. E1 is the end point for the axis with min. physical ID (CFG_AxPhyID).

7. Click [**Move**] button to start line motion.

Arc Demo

This example demonstrates how to use the ACM API to control an interpolation group's arc motion.



Available device: PCI-1220 {M7}

Selected Axes
First AxisID: 0
Second AxisID: 1

Command Counter
First Axis: 8000 Unit
Second Axis: 0 Unit

Arc Interpolation Operation
1-Axis
Master Axis ID:

Center
C1/R1: 4000 Unit
C2/R2: 0 Unit
● Cen ○ Ref

End
E1: 8000 Unit
E2: 0 Unit

Movement Mode
○ Absolute
● Relative

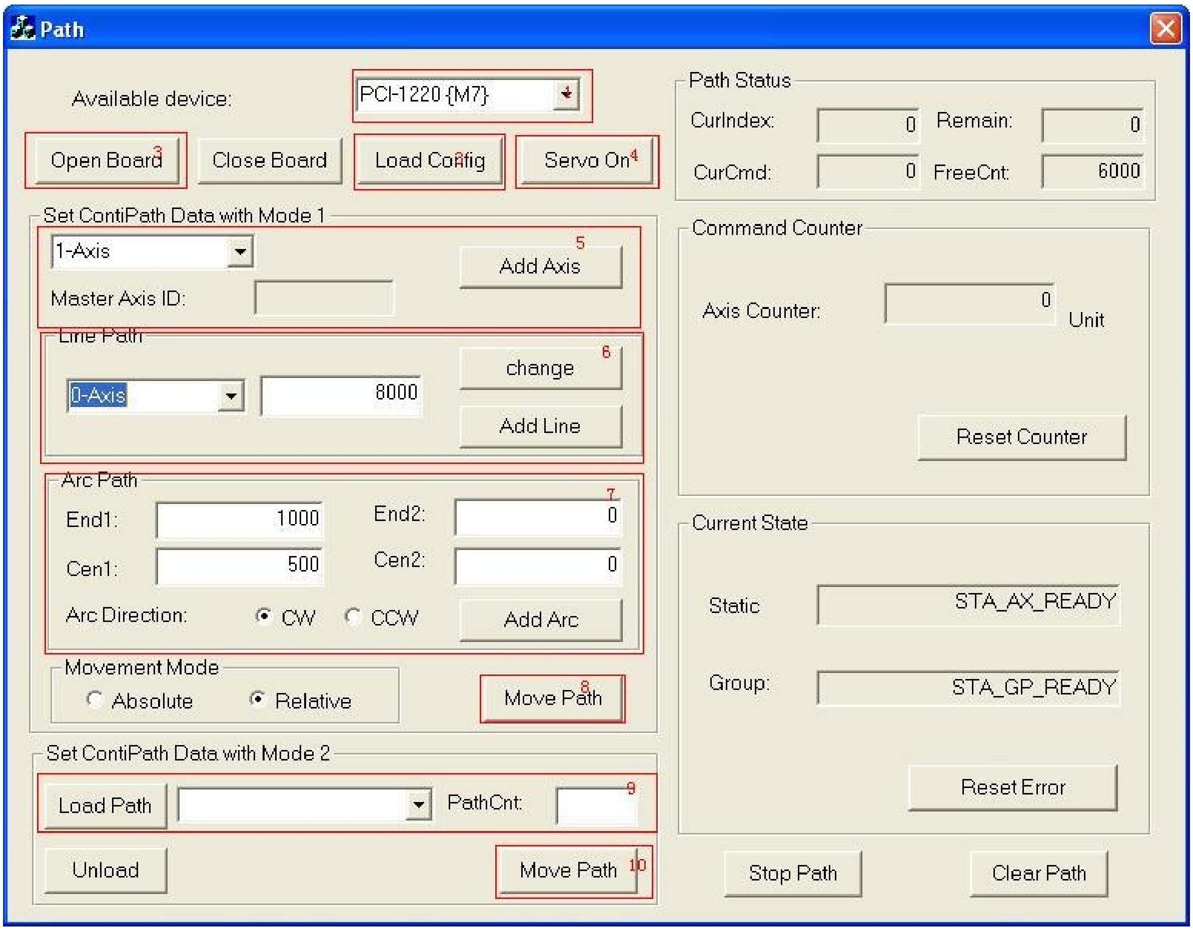
Arc Direction
● CW
○ CCW

Current State
First Axis: STA_AX_READY
Second-Axis: STA_AX_READY
Group: STA_GP_READY

1. Select a Device Number from the available device list;
2. Open device and its axes;
3. If you configuration file is available, you can load this file to configure this device;
4. Click [**Servo On**] button to open servo motor after it is connected;
5. Add axis to a group. For arc interpolation, if there are slave devices on this device, you can add two axes on the same slave device or on master device to one group.
6. Set the end points and center points for the axes in the group. **E1**(end point 1) and **C1**(center point 1) are for the master axis with min physical ID (CFG_AxPhyID). If **Cen** is checked, Acm_GpMoveCircularRel or Acm_GpMoveCircularAbs will be called, otherwise Acm_GpMoveCircularRel_3P or Acm_GpMoveCircularAbs_3P will be called.
7. Click [**Move**] button to start arc motion.

Path Demo

This example demonstrates how to use the ACM API to control an interpolation group's path (continuous interpolation) motion.



Available device: PCI-1220 {M7}

Path Status
CurIndex: 0 Remain: 0
CurCmd: 0 FreeCnt: 6000

Set ContiPath Data with Mode 1
1-Axis
Master Axis ID:

Line Path
0-Axis 8000

Arc Path
End1: 1000 End2: 0
Cen1: 500 Cen2: 0
Arc Direction: ● CW ○ CCW

Movement Mode
○ Absolute ● Relative

Set ContiPath Data with Mode 2
PathCnt:

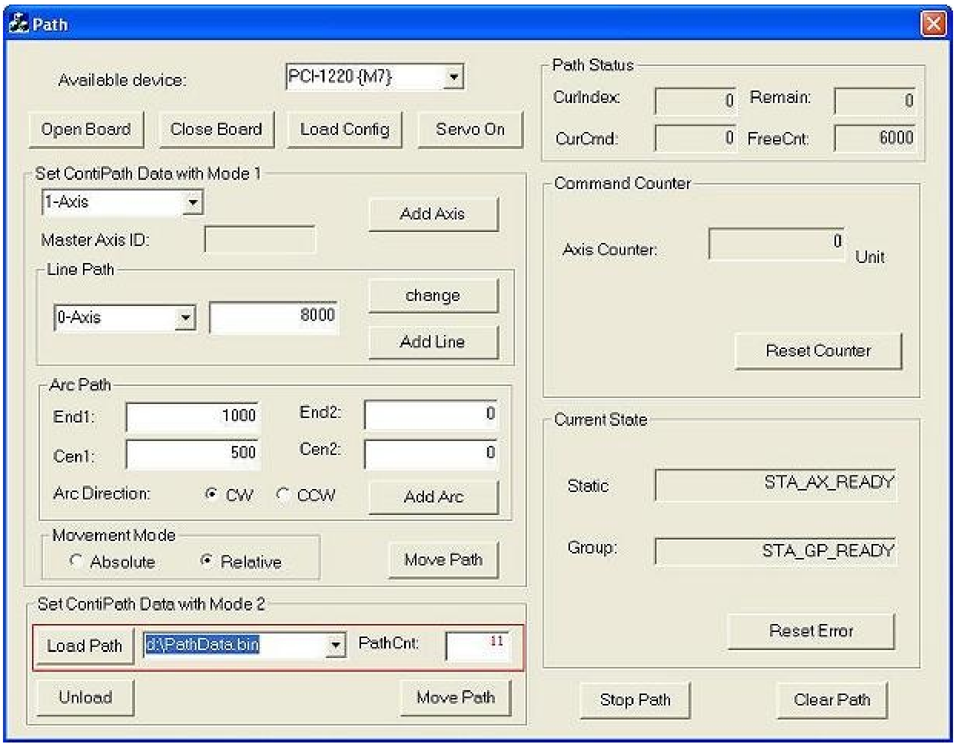
Command Counter
Axis Counter: 0 Unit

Current State
Static: STA_AX_READY
Group: STA_GP_READY

1. Select a Device Number from the available device list;
2. Open device and open its axes;
3. If configuration file is available, you can load this file to configure this device;
4. Click [**Servo On**] button to open servo motor after it is connected;
5. Add axes to one group. When the group's state is **STA_GP_READY,** you can add path data.
6. Set end points for line path data and click [**Add Line**] button to add a line path data. This step can be repeated or skipped.
7. Set end points and center points for arc path data. Click [**Add Arc**] button to add a arc path data. This step can be repeated or skipped.

Of course you can do step 7 before step 6. If there are path data in driver's path buffer, the **Remain** value on the right pane should be greater than 0.

8. Click [**Move Path]** to start path motion with the path data in driver's path buffer.
9. If path data file is available, you can click [**Load Path**] button to load it:



Available device: PCI-1220 {M7}

1-Axis

Master Axis ID:

0-Axis 8000

End1: 1000 End2: 0

Cen1: 500 Cen2: 0

Arc Direction: ● CW ○ CCW

○ Absolute ● Relative

d:\PathData.bin PathCnt: 11

CurIndex: 0 Remain: 0

CurCmd: 0 FreeCnt: 6000

Axis Counter: 0

Static: STA_AX_READY

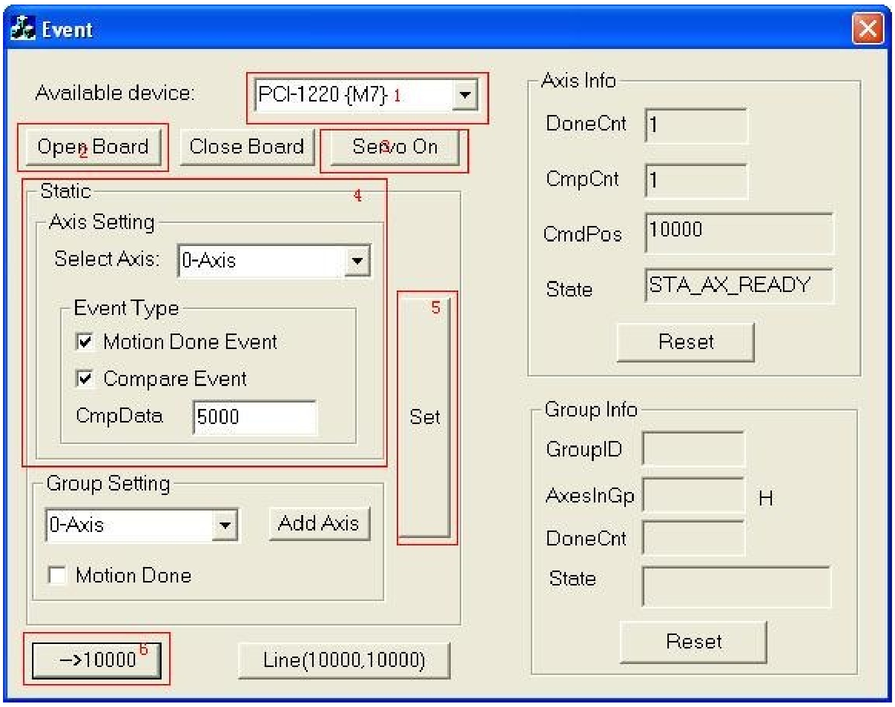
Group: STA_GP_READY

10. Click [**Move Path**] button to start path motion with this path data file.

11. After path motion ends, you can click [**Unload Path**] to unload this file and then load another path data file.

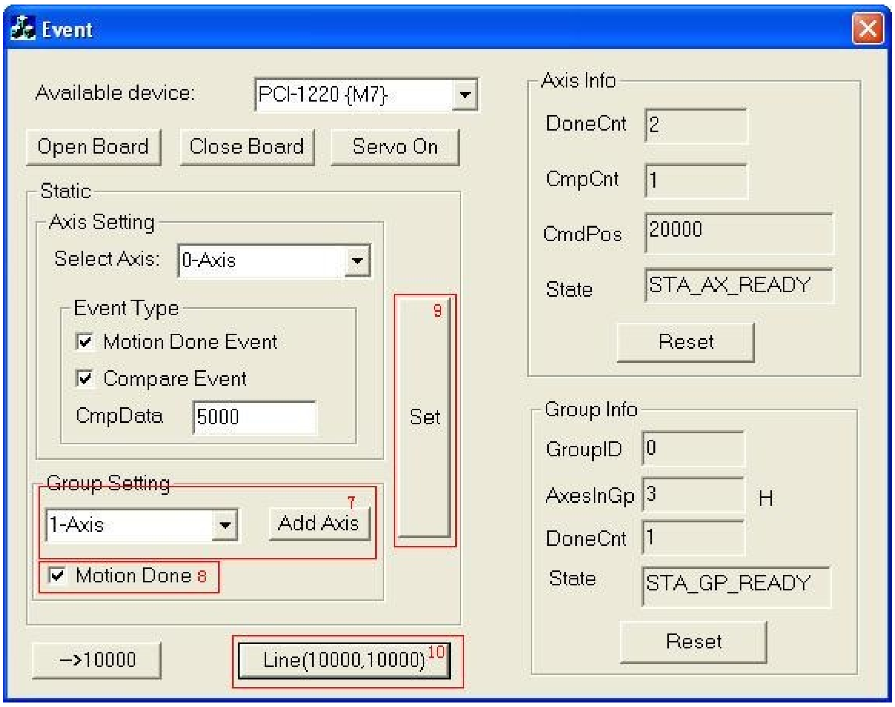
Event Demo

This example demonstrates how to check event from PCI1220 driver. PCI1220 supports three types of event: motion done event of single axis, compare event of single axis and motion done event of group.



Available device: PCI-1220 {M7}
Select Axis: 0-Axis
☑ Motion Done Event
☑ Compare Event
CmpData 5000
0-Axis
□ Motion Done
DoneCnt 1
CmpCnt 1
CmdPos 10000
State STA_AX_READY
GroupID
AxesInGp H
DoneCnt
State

1. Select a Device Number from the available device list;
2. Open device and open its axes;
3. Click [**Servo On**] button to open servo motor after it is connected;
4. Set event type for selected axis. Set valid compare data if you want to enable **Compare Event**.
5. Click [**Set**] button to set compare data and enable event(s).
6. Click [**->10000**] button to start A PTP motion. When the position matches the compare data, compare event will be checked. When the motion ends, there will be a motion done event.



Available device: PCI-1220 {M7}
Select Axis: 0-Axis
☑ Motion Done Event
☑ Compare Event
CmpData 5000
1-Axis
☑ Motion Done
DoneCnt 2
CmpCnt 1
CmdPos 20000
State STA_AX_READY
GroupID 0
AxesInGp 3 H
DoneCnt 1
State STA_GP_READY

7. If you want to check group event, click [**Add Axis**] to add axes to group.

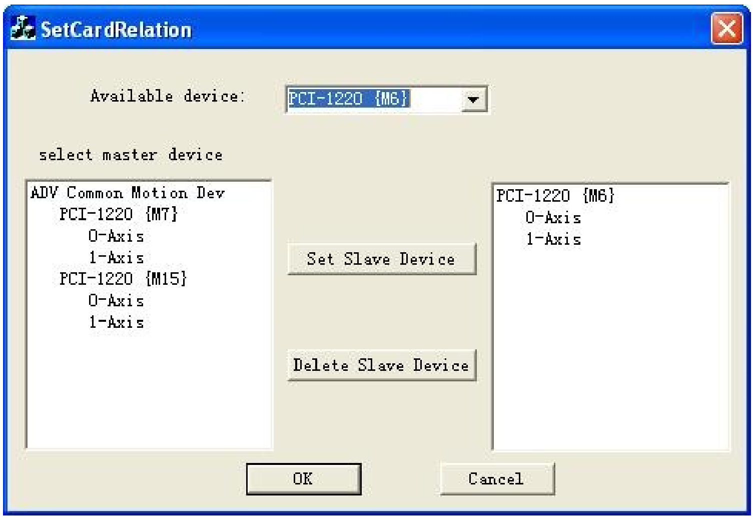
8. Check **Motion Done.**

9. Set compare data and enable events. (In the above picture, three types of event are enabled)

10. Click [**Line(10000,10000)**] to start a line motion. When the selected axis' position matches the compare data, compare event will be checked. When the motion ends, there will be a single axis' motion done event and a group's motion done event.

SetCardRelation Demo

This example demonstrates how to use the ACM API to control relations between multi PCI-1220 devices.



Available device: PCI-1220 {M6}

select master device

```
ADV Common Motion Dev
   PCI-1220 {M7}
      0-Axis
      1-Axis
   PCI-1220 {M15}
      0-Axis
      1-Axis
```

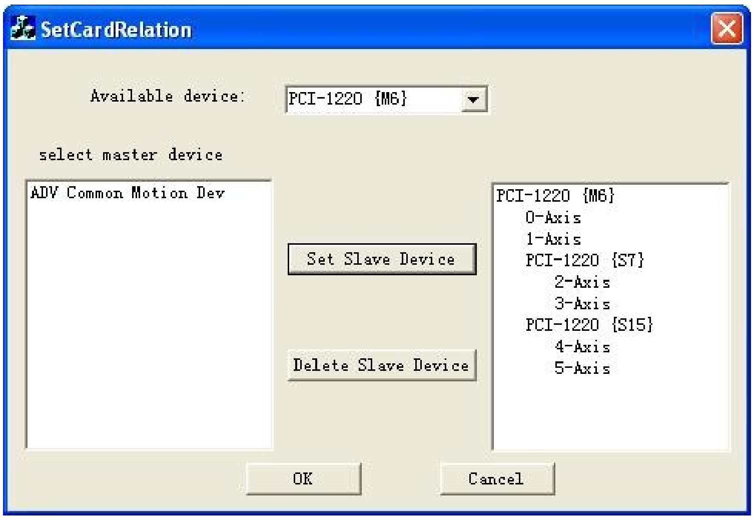
```
PCI-1220 {M6}
   0-Axis
   1-Axis
```

1. Select a Device Number from the available device list(PCI-1220{M7} for example);

2. The selected Device number will be displayed in the left panel,

3. Select a device from the right panel (PCI-1220{M6} for example);

4. Click "Set Slave Device" button, the selected device in the right panel (PCI-1220{M6}) will become a slave device of the selected device in the left panel(PCI-1220{M7}).



Available device: PCI-1220 {M6}

select master device

```
ADV Common Motion Dev
```

```
PCI-1220 {M6}
   0-Axis
   1-Axis
   PCI-1220 {S7}
      2-Axis
      3-Axis
   PCI-1220 {S15}
      4-Axis
      5-Axis
```

5. If select a slave device in the left panel and click **Delete Slave Device**, then the slave device will become master device again and will be listed in the right panel.

Acm_DevOpen function

Format:

U32 Acm_DevOpen(U32 DeviceNumber, PHAND DeviceHandle)

Purpose:

Open a specified device to get device handle.

Parameters:

Name	Type	In or Out	Description
DeviceNumber	U32	IN	Device number
DeviceHandle	PHAND	OUT	Return a point to the device handle

Return Value:

Error Code.

Acm_DevClose function

Format:

U32 Acm_DevClose(PHAND DeviceHandle)

Purpose:

Close a device.

Parameters:

Name	Type	In or Out	Description
DeviceHandle	PHAND	IN	a pointer to the device handle

Return Value:

Error Code.

Comments:

After calling this API, the device handle can not be used again.

Acm_DevLoadConfig function

Format:

U32 Acm_DevLoadConfig(HAND DeviceHandle, PI8 ConfigPath)

Purpose:

Set all configurations for the device according to the loaded file.

Parameters:

Name	Type	In or Out	Description
DeviceHandle	HAND	IN	Device handle from Acm_DevOpen
ConfigPath	PI8	IN	pointer to a string that saves configuration file's path.

Return Value:

Error Code.

Comments:

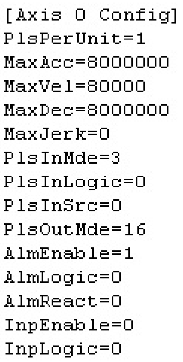
Configuration file can be binary or text file. If the file extension is .bin, driver reads the file in binary format. Otherwise, driver reads the file in .INI(text format).

The binary file format must use **MOT_DEV_CONFIG** structure:

```
typedef struct _MOT_DEV_CONFIG
{
ULONG MstrBaudRate0;
ULONG MstrBaudRate1;
ULONG CommWdgMde;
ULONG FwMemMde;
MOT_AX_CONFIG axi} MOT_DEV_CONFIG, *PMOT_DEV_CONFIG;
```

```
typedef struct _MOT_AX_CONFIG

  {

  ULONG PlsPerUnit;

  double MaxVel;

  double MaxAcc;

  double MaxDec;

  double MaxJerk;

  ULONG PlsInMde;

  ULONG PlsInLogic;

  ULONG PlsInSrc;

  ULONG PlsOutMde;

  ULONG AlmEnable;

  ULONG AlmLogic;

  ULONG AlmReact;

  ULONG InpEnable;
  ULONG InpLogic;

  ULONG ErcLogic;

  ULONG ErcOnTime;

  ULONG ErcOffTime;

  ULONG ErcEnMde;

  ULONG SdEnable;

  ULONG SdLogic;

  ULONG SdReact;

  ULONG SdLatch;

  ULONG ElEnable;

  ULONG ElLogic;

  ULONG ElReact;

  ULONG SwMelEnable;

  ULONG SwPelEnable;

  ULONG SwMelReact;

  ULONG SwPelReact;

  LONG SwMelValue;

  LONG SwPelValue;

  ULONG OrgLogic;

  ULONG EzLogic;

  ULONG PosLagEn;

  double MaxPosLag;

  } MOT_AX_CONFIG, *PMOT_AX_CONFIG;
```

The text file format looks like this:



```
[Axis 0 Config]
PlsPerUnit=1
MaxAcc=8000000
MaxVel=80000
MaxDec=8000000
MaxJerk=0
PlsInMde=3
PlsInLogic=0
PlsInSrc=0
PlsOutMde=16
AlmEnable=1
AlmLogic=0
AlmReact=0
InpEnable=0
InpLogic=0
```

Acm_EnableMotionEvent function

Format:

U32 Acm_EnableMotionEvent(HAND DeviceHandle, PU32 AxEnableEvtArray, PU32 GpEnableEvtArray, U32 AxArrayElements, U32 GpArrayElements)

Purpose:

Enable motion event.

Parameters:

<table>
<tr><th>Name</th><th>Type</th><th>In or Out</th><th>Description</th></tr>
<tr><td>DeviceHandle</td><td>HAND</td><td>IN</td><td>Device handle from Acm_DevOpen</td></tr>
<tr><td>AxEnableEvtArray</td><td>PU32</td><td>IN</td><td>Array[n], enable interrupt event for each axis., n is the axis count of motion device.
Array is of 32 bits data type, each bit represents different Event types:
<table>
<tr><th>Bit</th><th>31…2</th><th>1</th><th>0</th></tr>
<tr><td>Description</td><td>Reserved</td><td>EVT_AX_COMPARED</td><td>EVT_AX_MOTION_DONE</td></tr>
</table>
Bit n = 1 : Enable event ; Bit n = 0 : Disable event
eq. PCI-1220 has 2 axes, array[0] represent X-Axis, array[1] represent Y-Axis. When array[1]=0x2, it means the event of EVT_AX_COMPARED is enabled while EVT_AXMOTION_DONE is disabled.</td></tr>
<tr><td>GpEnableEvtArray</td><td>PU32</td><td>IN</td><td>Array[n], enable interrupt event for each group.
GpEnableEvtArray is 32 bits data type array and currently the value of n can only be 1.
<table>
<tr><th>Bit</th><th>31…n</th><th>1</th><th>0</th></tr>
<tr><td>Description</td><td>EVT_GPn_MOTION_DONE</td><td>EVT_GP2_MOTION_DONE</td><td>EVT_GP1_MOTION_DONE</td></tr>
</table>
Bit n = 1 : Enable event; Bit n = 0 : Disable event
<b><i>Note: For EVT_GPn_MOTION_DONE, n is GroupID. It can be got form PAR_GpGroupID property</i></b></td></tr>
<tr><td>AxArrayElements</td><td>U32</td><td>IN</td><td>number of AxEvtStatusArray elements</td></tr>
<tr><td>GpArrayElements</td><td>U32</td><td>IN</td><td>number of GpEvtStatusArray elements</td></tr>
</table>

Return Value:

Error Code.

See Also

Acm_CheckMotionEvent

Acm_CheckMotionEvent function

Format:

U32 Acm_CheckMotionEvent (HAND DeviceHandle, PU32 AxEvtStatusArray, PU32 GpEvtStatusArray, U32 AxArrayElements, U32 GpArrayElements, U32 Millisecond)

Purpose:

Check motion event.

Parameters:

<table>
<tr><th>Name</th><th>Type</th><th>In or Out</th><th>Description</th></tr>
<tr><td>DeviceHandle</td><td>HAND</td><td>IN</td><td>Device handle from Acm_DevOpen</td></tr>
<tr><td>AxEvtStatusArray</td><td>PU32</td><td>IN/OUT</td><td>Array[n] return interrupt event status of each axis. n is the axis count of motion device. Each array element is 32 bits data type array each bit represents different Event types:
<table>
<tr><td>Bit</td><td>31…2</td><td>1</td><td>0</td></tr>
<tr><td>Description</td><td>Reserved</td><td>EVT_AX_COMPARED</td><td>EVT_AX_MOTION_DONE</td></tr>
</table>
Bit n = 1 : Enable event ; Bit n = 0 : Disable event
e.q. PCI-1220 has 2 axes, array[0] represent X-Axis, array[1] represent Y-Axis. When array[1]=0x2, it means the event of EVT_AX_COMPARED is enabled while EVT_AXMOTION_DONE is disabled.</td></tr>
<tr><td>GpEvtStatusArray</td><td>PU32</td><td>IN/OUT</td><td>Array[n] returns Interrupt event status for each group.
GpEvtStatus is 32 bits data type array and currently the value of n can only be1.
<table>
<tr><td>Bit</td><td>31…n</td><td>1</td><td>0</td></tr>
<tr><td>Description</td><td>EVT_GPn_MOTION_DONE</td><td>EVT_GP2_MOTION_DONE</td><td>EVT_GP1_MOTION_DONE</td></tr>
</table>
Bit n = 1 : Enable event ; Bit n = 0 : Disable event
<b><i>Note: EVT_GPn_MOTION_DONE, n is GroupID. It can be get form PAR_GpGroupID property</i></b></td></tr>
<tr><td>AxArrayElements</td><td>U32</td><td>IN</td><td>number of AxEvtStatusArray elements</td></tr>
<tr><td>GpArrayElements</td><td>U32</td><td>IN</td><td>number of GpEvtStatusArray elements</td></tr>
<tr><td>Millisecond</td><td>U32</td><td>IN</td><td>Specify the time out value in millisecond for each checking</td></tr>
</table>

Return Value:

Error Code.

See Also

Acm_EnableMotionEvent

Acm_GetProperty function

Format:

U32 Acm_GetProperty(HAND Handle, U32 ProperyID, PVOID Buffer, U32 *BufferLength)

Purpose:

Get property value.

Parameters:

Name	Type	In or Out	Description
Handle	HAND	IN	Object handle. This handle may be device handle from Acm_DevOpen, or axis handle from Acm_AxOpen, or group handle from Acm_GpAddAxis
ProperyID	U32	IN	Property ID to query
Buffer	PVOID	OUT	data buffer for property
BufferLength	U32 *	IN/OUT	IN, buffer size for the property; OUT, returned data required length.

Return Value:

Error Code.

Comments:

If the buffer is too small, the return value will be error code "**InvalidInputParam**". In this case, driver will return the actual size of the property in **BufferLength**.

Acm_SetProperty function

Format:

U32 Acm_SetProperty(HAND Handle, U32 ProperyID, PVOID Buffer, U32 BufferLength)

Purpose:

Set property value.

Parameters:

Name	Type	In or Out	Description
Handle	HAND	IN	Object handle. This handle may be device handle from Acm_DevOpen, or axis handle from Acm_AxOpen, or group handle from Acm_GpAddAxis
ProperyID	U32	IN	Property ID to query
Buffer	PVOID	IN	data buffer for property
BufferLength	U32	IN	buffer size for the property

Return Value:

Error Code.

Comments:

For some properties, driver may package the value with some adjustment for precision consideration. So some properties' output value may be different from the input value.

Acm_GetLastError function

Format:

U32 Acm_GetLastError(HAND ObjectHandle)

Purpose:

Get device or axis or group 's last error. Some objects' error need to be got in this asynchronous mode.

Parameters:

Name	Type	In or Out	Description
ObjectHandle	HAND	IN	Object handle. This handle may be device handle from Acm_DevOpen, or axis handle from Acm_AxOpen, or group handle from Acm_GpAddAxis

Return Value:

Error Code.

Acm_DevReadEEPROM function

Format:

U32 Acm_DevReadEEPROM(U32 DeviceNumber, U16 EEPROMAddr, PU16 readValue)

Purpose:

Open a specified device to get device handle.

Parameters:

Name	Type	In or Out	Description
DeviceNumber	U32	IN	Device number
EEPROMAddr	U16	IN	the EEPROM address to be read
readValue	PU16	OUT	the pointer that point to the value read from the EEPROM

Return Value:

Error Code.

Acm_DevWriteEEPROM function

Format:

U32 Acm_DevWriteEEPROM(HAND DeviceHandle, U16 EEPROMAddr, U16 writeValue)

Purpose:

Close a device.

Parameters:

Name	Type	In or Out	Description
DeviceHandle	PHAND	IN	a pointer to the device handle
EEPROMAddr	U16	IN	the EEPROM address to be written
writeValue	U16	IN	the data to be written to the EEPROM

Return Value:

Error Code.

Acm_AxOpen function

Format:

U32 Acm_AxOpen(HAND DeviceHandle, U16 PhyAxis, PHAND AxisHandle)

Purpose:

Open specified axis and get this axis object's handle.

Parameters:

Name	Type	In or Out	Description
DeviceHandle	HAND	IN	Device handle from Acm_DevOpen
PhyAxis	U16	IN	Physical Axis Number (ex: PCI-1220: 0,1)
AxisHandle	PHAND	OUT	returns a pointer to the axis handle

Return Value:

Error Code.

Acm_AxClose function

Format:

U32 Acm_AxClose(PHAND AxisHandle)

Purpose:

Close axis.

Parameters:

Name	Type	In or Out	Description
AxisHandle	PHAND	IN	Pointer to the axis handle

Return Value:

Error Code.

Comments:

After calling this API, the axis handle can not be used again.

Acm_AxGetState function

Format:

U32 Acm_AxGetState(HAND AxisHandle, PU16 State)

Purpose:

Get the axis' current state.

Parameters:

Name	Type	In or Out	Description
AxisHandle	HAND	IN	Device handle from Acm_AxOpen
State	PU16	OUT	Return axis state:

Define	Value	Description
STA_AxDisable	0	Axis is disabled, you can open it to active it
STA_AxReady	1	Axis is ready and waiting for new command
STA_Stopping	2	Axis is stopping
STA_AxErrorStop	3	Axis has stopped because of error
STA_AxHoming	4	Axis is executing home motion
STA_AxPtpMotion	5	Axis is executing PTP motion
STA_AxContiMotion	6	Axis is executing continuous motion
STA_AxSyncMotion	7	Axis is in one group and the group is executing interpolation motion
STA_AX_EXT_JOG	8	Axis is controlled by external signal and will execute JOG mode motion once external signal is active
STA_AX_EXT_MPG	9	Axis is controlled by external signal and will execute MPG mode motion once external signal is active

Return Value:

Error Code.

Acm_AxResetError function

Format:

U32 Acm_AxResetError(HAND AxisHandle)

Purpose:

Reset the axis' state. If the axis is in STA_AxErrorStop state, the state will be changed to STA_AxReady after calling this function.

Parameters:

Name	Type	In or Out	Description
AxisHandle	HAND	IN	Axis handle from Acm_AxOpen

Return Value:

Error Code.

Acm_AxSetSvOn function

Format:

U32 Acm_AxSetSvOn(HAND AxisHandle, U32 OnOff)

Purpose:

Set servo Driver ON or OFF.

Parameters:

<table>
<tr><th>Name</th><th>Type</th><th>In or Out</th><th>Description</th></tr>
<tr><td>AxisHandle</td><td>HAND</td><td>IN</td><td>Axis handle from Acm_AxOpen</td></tr>
<tr><td>OnOff</td><td>U32</td><td>IN</td><td>Setting the action of SVON signal
<table>
<tr><th><em>Value</em></th><th><em>Meaning</em></th></tr>
<tr><td>0</td><td>In-active</td></tr>
<tr><td>1</td><td>Active</td></tr>
</table>
</td></tr>
</table>

Return Value:

Error Code.

Comments:

If you want to use this function, you must set property CFG_AxGenDoEnable to be **GEN_DO_EN** first.

Acm_AxGetMotionIO function

Format:

U32 Acm_AxGetMotionIO(HAND AxisHandle, PU32 Status)

Purpose:

Get the motion I/O status of the axis.

Parameters:

<table>
<tr><th>Name</th><th>Type</th><th>In or Out</th><th>Description</th></tr>
<tr><td>AxisHandle</td><td>HAND</td><td>IN</td><td>Device handle from Acm_AxOpen</td></tr>
<tr><td>Status</td><td>PU32</td><td>OUT</td><td>
<table>
<tr><th>Bit</th><th>Name</th><th>Description</th></tr>
<tr><td>0</td><td>RDY</td><td>RDY pin input(PCI1220 not support)</td></tr>
<tr><td>1</td><td>ALM</td><td>Alarm Signal</td></tr>
<tr><td>2</td><td>+EL</td><td>Positive Limit Switch</td></tr>
<tr><td>3</td><td>-EL</td><td>Negative Limit Switch</td></tr>
<tr><td>4</td><td>ORG</td><td>Origin Switch</td></tr>
<tr><td>5</td><td>DIR</td><td>DIR output(PCI1220 not support)</td></tr>
<tr><td>6</td><td>EMG</td><td>Emergency signal input</td></tr>
<tr><td>7</td><td>PCS</td><td>PCS signal input(PCI1220 not support)</td></tr>
<tr><td>8</td><td>ERC</td><td>ERC pin output(PCI1220 not support)</td></tr>
<tr><td>9</td><td>EZ</td><td>Encode Z phase</td></tr>
<tr><td>10</td><td>CLR</td><td>Clear signal (PCI1220 not support)</td></tr>
<tr><td>11</td><td>Latch</td><td>Latch signal input(PCI1220 not support)</td></tr>
<tr><td>12</td><td>SD</td><td>Slow Down signal input(PCI1220 not support)</td></tr>
<tr><td>13</td><td>INP</td><td>In-Position signal input</td></tr>
<tr><td>14</td><td>SVON</td><td>Servo-ON output status(PCI1220 not support)</td></tr>
<tr><td>15</td><td>RALM</td><td>Alarm Reset output status(PCI1220 not support)</td></tr>
<tr><td>16</td><td>SLMT+</td><td>Positive Software Limit</td></tr>
<tr><td>17</td><td>SLMT-</td><td>Negative Software Limit</td></tr>
<tr><td>…</td><td>…</td><td>…</td></tr>
<tr><td>31</td><td>…</td><td>…</td></tr>
</table>
</td></tr>
</table>

Return Value:

Error Code.

Acm_AxGetMotionStatus function

Format:

U32 Acm_AxGetMotionStatus(HAND AxisHandle, PU32 Status)

Purpose:

Get current motions status of the axis.

Parameters:

<table>
<tr><th>Name</th><th>Type</th><th>In or Out</th><th>Description</th></tr>
<tr><td>AxisHandle</td><td>HAND</td><td>IN</td><td>Device handle from Acm_AxOpen</td></tr>
<tr><td>Status</td><td>PU32</td><td>OUT</td><td>
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>0</td><td>Stop</td></tr>
<tr><td>1</td><td>Reserved</td></tr>
<tr><td>2</td><td>Wait ERC finished</td></tr>
<tr><td>3</td><td>Reserved</td></tr>
<tr><td>4</td><td>Correcting Backlash</td></tr>
<tr><td>5</td><td>Reserve</td></tr>
<tr><td>6</td><td>Feeding in return velocity = FA</td></tr>
<tr><td>7</td><td>Feeding in StrVel speed = FL</td></tr>
<tr><td>8</td><td>Accelerating</td></tr>
<tr><td>9</td><td>Feeding in MaxVel speed = FH</td></tr>
<tr><td>10</td><td>Decelerating</td></tr>
<tr><td>11</td><td>Waiting for INP input</td></tr>
<tr><td>12</td><td>Reserved</td></tr>
<tr><td>13</td><td>Reserved</td></tr>
<tr><td>14</td><td>Reserved</td></tr>
<tr><td>15</td><td>Reserved</td></tr>
<tr><td>…</td><td>…</td></tr>
<tr><td>31</td><td>…</td></tr>
</table>
</td></tr>
</table>

Return Value:

Error Code.

Acm_AxMoveRel function

Format:

U32 Acm_AxMoveRel(HAND AxisHandle, F64 Distance)

Purpose:

Start single axis' relative motion.

Parameters:

Name	Type	In or Out	Description
AxisHandle	HAND	IN	Device handle from Acm_AxOpen
Distance	F64	IN	Relative distance (unit = pulse per unit)

Return Value:

Error Code.

Comments:

The speed curve is decided by properties: PAR_AxVelLow, PAR_AxVelHigh, PAR_AxAcc, PAR_AxDec, PAR_AxJerk.

Acm_AxMoveAbs function

Format:

U32 Acm_AxMoveAbs(HAND AxisHandle, F64 Position)

Purpose:

Start single axis' absolute motion.

Parameters:

Name	Type	In or Out	Description
AxisHandle	HAND	IN	Device handle from Acm_AxOpen
Position	F64	IN	Absolute position (unit = pulse per unit)

Return Value:

Error Code.

Comments:

The speed curve is decided by properties:PAR_AxVelLow, PAR_AxVelHigh, PAR_AxAcc, PAR_AxDec, PAR_AxJerk.

Acm_AxMoveVel function

Format:

U32 Acm_AxMoveVel(HAND AxisHandle, U16 Direction)

Purpose:

To command axis to make a never ending movement with a specified velocity.

Parameters:

<table>
<tr><th>Name</th><th>Type</th><th>In or Out</th><th>Description</th></tr>
<tr><td>AxisHandle</td><td>HAND</td><td>IN</td><td>Device handle from Acm_AxOpen</td></tr>
<tr><td>Direction</td><td>U16</td><td>IN</td><td>Direction
<table>
<tr><th><i>Value</i></th><th><i>Meaning</i></th></tr>
<tr><td>0</td><td>Positive direction</td></tr>
<tr><td>1</td><td>Negative direction</td></tr>
</table></td></tr>
</table>

Return Value:

Error Code.

Comments:

The speed curve is decided by properties: PAR_AxVelLow, PAR_AxVelHigh, PAR_AxAcc, PAR_AxDec, PAR_AxJerk.

Acm_AxChangePos function

Format:

U32 Acm_AxChangePos(HAND AxisHandle, F64 NewDistance)

Purpose:

To command axis to change the distance while axis is in velocity motion.

Parameters:

Name	Type	In or Out	Description
AxisHandle	HAND	IN	Device handle from Acm_AxOpen
NewDistance	F64	IN	New relative distance (unit = pulse per unit)

Return Value:

Error Code.

Acm_AxChangeVel function

Format:

U32 Acm_AxChangeVel(HAND AxisHandle, F64 NewVelocity)

Purpose:

To command axis to change the velocity while axis is in velocity motion.

Parameters:

Name	Type	In or Out	Description
AxisHandle	HAND	IN	Device handle from Acm_AxOpen
NewVelocity	F64	IN	New velocity (unit = pulse per unit)

Return Value:

Error Code.

Comments:

The speed curve is decided by properties: PAR_AxAcc, PAR_AxDec, PAR_AxJerk.

Acm_AxStopDec function

Format:

U32 Acm_AxStopDec(HAND AxisHandle)

Purpose:

To command axis to decelerate to stop.

Parameters:

Name	Type	In or Out	Description
AxisHandle	HAND	IN	Device handle from Acm_AxOpen

Return Value:

Error Code.

Comments:

Deceleration curve is decided by proeprties:PAR_AxVelLow, PAR_AxVelHigh, PAR_AxAcc, PAR_AxDec, PAR_AxJerk.

Acm_AxStopEmg function

Format:

U32 Acm_AxStopEmg(HAND AxisHandle)

Purpose:

To command axis to stop immediately without deceleration .

Parameters:

Name	Type	In or Out	Description
AxisHandle	HAND	IN	Device handle from Acm_AxOpen

Return Value:

Error Code.

Acm_AxHome function

Format:

U32 Acm_AxHome(HAND AxisHandle, U32 HomeMode, U32 Dir)

Purpose:

To command axis to start typical home motion. The 11 types of typical home motion is composed of extended home.

Parameters:

<table>
<tr><th>Name</th><th>Type</th><th>In or Out</th><th>Description</th></tr>
<tr><td>AxisHandle</td><td>HAND</td><td>IN</td><td>Device handle from Acm_AxOpen</td></tr>
<tr><td>HomeMode</td><td>U32</td><td>IN</td><td>Home mode
<table>
<tr><th><i>Value</i></th><th><i>Definition</i></th><th><i>Meaning</i></th></tr>
<tr><td>0</td><td>MODE1_Abs</td><td>ORG only (switch off)</td></tr>
<tr><td>1</td><td>MODE2_Lmt</td><td>EL only (switch off)</td></tr>
<tr><td>2</td><td>MODE3_Ref</td><td>EZ only (switch off)</td></tr>
<tr><td>3</td><td>MODE4_Abs_Ref</td><td>ORG+EZ(switch off)</td></tr>
<tr><td>4</td><td>MODE5_Abs_NegRef</td><td>ORG+ inverse EZ (switch off)</td></tr>
<tr><td>5</td><td>MODE6_Lmt_Ref</td><td>EL+EZ (switch off)</td></tr>
<tr><td>6</td><td>MODE7_AbsSearch</td><td>ORG only (switch On)</td></tr>
<tr><td>7</td><td>MODE8_LmtSearch</td><td>EL only (switch On)</td></tr>
<tr><td>8</td><td>MODE9_AbsSearch_Ref</td><td>ORG+EZ (switch On)</td></tr>
<tr><td>9</td><td>MODE10_AbsSearch_NegRef</td><td>ORG+ inverse EZ (switch On)</td></tr>
<tr><td>10</td><td>MODE11_LmtSearch_Ref</td><td>EL+EZ (switch On)</td></tr>
</table>
Details about home mode, you can see Home Config</td></tr>
<tr><td>Dir</td><td>U32</td><td>IN</td><td>Direction
<table>
<tr><th><i>Value</i></th><th><i>Meaning</i></th></tr>
<tr><td>0</td><td>Positive direction</td></tr>
<tr><td>1</td><td>Negative</td></tr>
</table></td></tr>
</table>

Return Value:

Error Code.

Comments:

If property CFG_AxHomeResetEnable is set to be true, command position and actual position will be reset to be zero after home motion ends. This home motion only supports acceleration/deceleration of symmetrical trapezia curve or constant velocity with PAR_AxVelLow.

Acm_AxHomeEx function

Format:

U32 Acm_AxHomeEx(HAND AxisHandle, U32 DirMode)

Purpose:

To command axis to start extended home motion. The home mode is specified by property CFG_HomeModeEx.

Parameters:

<table>
<tr><th>Name</th><th>Type</th><th>In or Out</th><th>Description</th></tr>
<tr><td>AxisHandle</td><td>HAND</td><td>IN</td><td>Device handle from Acm_AxOpen</td></tr>
<tr><td>DirMode</td><td>U32</td><td>IN</td><td>Direction mode
<table>
<tr><th><i>Value</i></th><th><i>Definition</i></th><th><i>Description</i></th></tr>
<tr><td>0</td><td>PosiDir</td><td>Always in positive direction.</td></tr>
<tr><td>1</td><td>NegDir</td><td>Always in negative direction.</td></tr>
<tr><td>2</td><td>SwitchPosi</td><td>Depends on switch status. If switch is off, direction is positive. Otherwise, direction is negative.</td></tr>
<tr><td>3</td><td>SwitchNeg</td><td>Depends on switch status. If switch is off, direction is negative. Otherwise, direction is positive.</td></tr>
</table>
</td></tr>
</table>

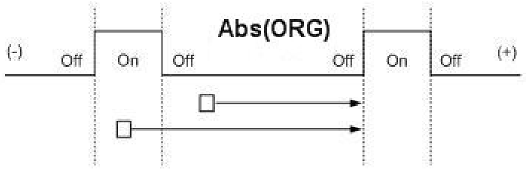
Return Value:

Error Code.

Comments:

If property CFG_AxHomeResetEnable is set to be true, command position and actual position will be reset to be zero after home motion ends. This home motion only supports acceleration/deceleration of symmetrical trapezia curve or constant velocity with PAR_AxVelLow. If property CFG_HomeModeEx is set to be RefPulse(EZ) mode, home motion can only execute constant velocity with PAR_AxVelLow.

For example, if parameter **DirMode** is **PosiDir**, property CFG_HomeModeEx is **AbsSwitch**(ORG) and the logic of ORG is **active high**, the extended home motion is lick this:



Acm_AxGetCmdPosition function

Format:

U32 Acm_AxGetCmdPosition(HAND AxisHandle, PF64 Position)

Purpose:

Get current command position of the specified axis.

Parameters:

Name	Type	In or Out	Description
AxisHandle	HAND	IN	Device handle from Acm_AxOpen
Position	PF64	OUT	Return the command position

Return Value:

Error Code.

Acm_AxSetCmdPosition function

Format:

U32 Acm_AxSetCmdPosition(HAND AxisHandle, F64 Position)

Purpose:

Set command position for the specified axis.

Parameters:

Name	Type	In or Out	Description
AxisHandle	HAND	IN	Device handle from Acm_AxOpen
Position	F64	IN	New command position

Return Value:

Error Code.

Acm_AxGetActualPosition function

Format:

U32 Acm_AxGetActualPosition(HAND AxisHandle, PF64 Position)

Purpose:

Get current actual position of the specified axis.

Parameters:

Name	Type	In or Out	Description
AxisHandle	HAND	IN	Device handle from Acm_AxOpen
Position	PF64	OUT	Return the actual position

Return Value:

Error Code.

Acm_AxSetActualPositionfunction

Format:

U32 Acm_AxSetActualPosition(HAND AxisHandle, F64 Position)

Purpose:

Set actual position for the specified axis.

Parameters:

Name	Type	In or Out	Description
AxisHandle	HAND	IN	Device handle from Acm_AxOpen
Position	F64	IN	New actual position

Return Value:

Error Code.

Acm_AxGetCmdVelocity function

Format:

U32 Acm_AxGetCmdVelocity(HAND AxisHandle, PF64 Velocity)

Purpose:

Get current command velocity of the specified axis.

Parameters:

Name	Type	In or Out	Description
AxisHandle	HAND	IN	Device handle from Acm_AxOpen
Velocity	PF64	OUT	Return the command velocity

Return Value:

Error Code.

Acm_AxSetCmpAuto function

Format:

U32 Acm_AxSetCmpAuto(HAND AxisHandle, F64 Start, F64 End, F64 Interval)

Purpose:

Set compare data for the specified axis.

Parameters:

Name	Type	In or Out	Description
AxisHandle	HAND	IN	Device handle from Acm_AxOpen
Start	F64	IN	First compare data
End	F64	IN	Last compare data
Interval	F64	IN	Compare interval

Return Value:

Error Code.

Comments:

If property CFG_AxCmpMethod is set to **MTD_GREATER_POSITION**, the **Start** parameter should be smaller than current position (command position or actual position). The first compare data will be loaded to comparator+, and if the current position matches the comparator+, pulse will be generated, the next compare data will be loaded to comparator+ automatically. If property CFG_AxCmpMethod is set to **MTD_SMALLER_POSITION**, the **Start** parameter should be greater than current position (command position or actual position). The first compare data will be loaded to comparator-, and if the current position matches the comparator-, pulse will be generated, the next compare data will be loaded to comparator- automatically.

Before setting compare data, you need to set property CFG_AxCmpEnable to **CMP_EN** first. If you want to close compare function, you only need to set property CFG_AxCmpEnable to **CMP_DIS** and nothing is necessary to clear compare data.

Once any function of Acm_AxSetCmpData, Acm_AxSetCmpAuto , Acm_AxSetCmpTable is called, the before compare data will be clear.

See Also:

Acm_AxSetCmpData, Acm_AxSetCmpTable, Acm_AxGetCmpData

Acm_AxSetCmpData function

Format:

U32 Acm_AxSetCmpData(HAND AxisHandle, F64 CmpPosition)

Purpose:

Set compare data for the specified axis.

Parameters:

Name	Type	In or Out	Description
AxisHandle	HAND	IN	Device handle from Acm_AxOpen
CmpPosition	F64	IN	Compare data

Return Value:

Error Code.

Comments:

If property CFG_AxCmpMethod is set to **MTD_GREATER_POSITION**, the **CmpPosition** should be smaller than current position (command position or actual position). If property CFG_AxCmpMethod is set to **MTD_SMALLER_POSITION**, the **CmpPosition** should be greater than current position (command position or actual position). The new compare data will always be loaded to comparator+ .

Before setting compare data, you need to set property CFG_AxCmpEnable to **CMP_EN** first. If you want to close compare function, you only need to set property CFG_AxCmpEnable to **CMP_DIS** and nothing is necessary to clear compare data.

Once any function of Acm_AxSetCmpData, Acm_AxSetCmpAuto , Acm_AxSetCmpTable is called, the previous compared data will be cleared.

See Also:

Acm_AxSetCmpData, Acm_AxSetCmpAuto , Acm_AxGetCmpData

Acm_AxSetCmpTable function

Format:

U32 Acm_AxSetCmpTable(HAND AxisHandle, PF64 TableArray, I32 ArrayCount)

Purpose:

Set compare data for the specified axis.

Parameters:

Name	Type	In or Out	Description
AxisHandle	HAND	IN	Device handle from Acm_AxOpen
TableArray	PF64	IN	Compare data table
ArrayCount	I32	IN	Compare data count in the table

Return Value:

Error Code.

Comments:

If property CFG_AxCmpMethod is set to **MTD_GREATER_POSITION**, the first data in table should be smaller than current position (command position or actual position). The first data will be loaded to comparator+, and if the current position matches the comparator+, pulse will be generated, the next compare data will be loaded to comparator+ automatically. If property CFG_AxCmpMethod is set to **MTD_SMALLER_POSITION**, the first data in table should be greater than current position (command position or actual position). The first data will be loaded to comparator-, and if the current position matches the comparator-, pulse will be generated, the next compare data will be loaded to comparator- automatically.

Before setting compare data, please set set property CFG_AxCmpEnable to **CMP_EN** first. If you want to close compare function, you only need to set property CFG_AxCmpEnable to **CMP_DIS** and nothing is necessary to clear compare data.

As long as any of the three functions Acm_AxSetCmpData, Acm_AxSetCmpAuto , Acm_AxSetCmpTable is called, the previously compared data will be cleared.

See Also:

Acm_AxSetCmpData, Acm_AxSetCmpAuto , Acm_AxGetCmpData

Acm_AxGetCmpData function

Format:

U32 Acm_AxGetCmpData(HAND AxisHandle, PF64 CmpPosition)

Purpose:

Get current compare data in the comparator.

Parameters:

Name	Type	In or Out	Description
AxisHandle	HAND	IN	Device handle from Acm_AxOpen
CmpPosition	PF64	OUT	Return current compare data

Return Value:

Error Code.

Comments:

If no compare data is saved in compare buffer, this function will return error code **FuncError.** If there are data left not compared in compare buffer, this function will return the current compare data by **CmpPosition**. If all data in compare buffer has been compared, this function will return the last data in compare buffer.

See Also:

Acm_AxSetCmpData, Acm_AxSetCmpAuto , Acm_AxSetCmpTable

Acm_AxDoSetBit function

Format:

U32 Acm_AxDoSetBit(HAND AxisHandle, U16 DoChannel, U8 BitData)

Purpose:

Output DO value to channel.

Parameters:

Name	Type	In or Out	Description
AxisHandle	HAND	IN	Device handle from Acm_AxOpen
DoChannel	U16	IN	Digital output channel(0~3)
BitData	U8	IN	DO value: 0 or 1

Return Value:

Error Code.

Comments:

If you want to use this general DO function, you must set property CFG_AxGenDoEnable to **GEN_DO_EN** first. When CFG_AxGenDoEnable is enabled, the property CFG_AxCmpEnable will be disabled automatically and Acm_AxSetCmpData, Acm_AxSetCmpTable, Acm_AxSetCmpAuto will not be able to generate trigger pulse because these two functions use the same output pins(OUT0 ~ OUT3).

Acm_AxDoGetBit function

Format:

U32 Acm_AxDoGetBit(HAND AxisHandle, U16 DoChannel, PU8 BitData)

Purpose:

Get DO channel status.

Parameters:

Name	Type	In or Out	Description
AxisHandle	HAND	IN	Device handle from Acm_AxOpen
DoChannel	U16	IN	Digital output channel(0~3)
BitData	PU8	OUT	Return DO status: 0 or 1

Return Value:

Error Code.

Acm_AxDiGetBit function

Format:

U32 Acm_AxDiGetBit(HAND AxisHandle, U16 DiChannel, PU8 BitData)

Purpose:

Get the specified channel's DI value.

Parameters:

Name	Type	In or Out	Description
AxisHandle	HAND	IN	Device handle from Acm_AxOpen
DiChannel	U16	IN	Digital input channel(EZ,IN0,IN1,ORG)
BitData	PU8	OUT	Return DI value: 0 or 1

Return Value:

Error Code.

Acm_AxSetExtDrive function

Format:

U32 Acm_AxSetExtDrive(HAND AxisHandle, U16 ExtDrvMode)

Purpose:

Enable or disable external drive mode.

Parameters:

<table>
<tr><th>Name</th><th>Type</th><th>In or Out</th><th>Description</th></tr>
<tr><td>AxisHandle</td><td>HAND</td><td>IN</td><td>Device handle from Acm_AxOpen</td></tr>
<tr><td>ExtDrvMode</td><td>U16</td><td>IN</td><td>External drive mode
<table>
<tr><th><i>Value</i></th><th><i>Meaning</i></th></tr>
<tr><td>0</td><td>Disable</td></tr>
<tr><td>1</td><td>JOG mode</td></tr>
<tr><td>2</td><td>MPG mode(hand wheel)</td></tr>
</table>
</td></tr>
</table>

Return Value:

Error Code.

Acm_GpAddAxis function

Format:

U32 Acm_GpAddAxis(PHAND GpHandle,HAND AxHandle)

Purpose:

Add an axis to the specified group.

Parameters:

Name	Type	In or Out	Description
GpHandle	PHAND	IN/OUT	Group handle
AxHandle	HAND	IN	Axis handle to be added

Return Value:

Error Code.

Comments:

If **GpHandle** points to NULL, driver will create a new group handle and add the axis to this new group. If **GpHandle** points to a valid group handle, driver will just add the axis to the group.

Acm_GpRemAxis function

Format:

U32 Acm_GpRemAxis(HAND GpHandle,HAND AxHandle)

Purpose:

Remove an axis from the specified group.

Parameters:

Name	Type	In or Out	Description
GpHandle	HAND	IN	Group handle
AxHandle	HAND	IN	Axis handle to be removed

Return Value:

Error Code.

Comments:

After **Acm_GpRemAxis** is called and no axis is in group, the **GpHandle** can still be used. You can use this group handle to add other axis. But if you have called Acm_GpClose to close this group handle, the group handle can't be used again.

Acm_GpClose function

Format:

U32 Acm_GpClose(PHAND pGroupHandle)

Purpose:

Remove all axes in the group and close the group handle.

Parameters:

Name	Type	In or Out	Description
pGroupHandle	PHAND	IN	Point to group handle to be closed

Return Value:

Error Code.

Acm_GpGetState function

Format:

U32 Acm_GpGetState (HAND GroupHandle, PU16 pState)

Purpose:

Get the group's current state.

Parameters:

<table>
<tr><th>Name</th><th>Type</th><th>In or Out</th><th>Description</th></tr>
<tr><td>GroupHandle</td><td>HAND</td><td>IN</td><td>Group handle from Acm_GpAddAxis</td></tr>
<tr><td>pState</td><td>PU16</td><td>OUT</td><td>Return group state:
<table>
<tr><th>Define</th><th>Value</th><th>Description</th></tr>
<tr><td>STA_GP_DISABLE</td><td>0</td><td>Group is disabled, you can call Acm_GpAddAxis to add axis into it.</td></tr>
<tr><td>STA_GP_READY</td><td>1</td><td>Group is ready and waiting for new command</td></tr>
<tr><td>STA_GP_STOPPING</td><td>2</td><td>Group is being stopped</td></tr>
<tr><td>STA_GP_ERROR_STOP</td><td>3</td><td>Group stopped because of error</td></tr>
<tr><td>STA_GP_MOTION</td><td>4</td><td>Group is executing interpolation motion (Line, Arc, Direct)</td></tr>
<tr><td>STA_GP_AX_MOTION</td><td>5</td><td>Axis(Axes) in group is(are) executing single axis motion</td></tr>
<tr><td>STA_GP_MOTION_PATH</td><td>6</td><td>Group is executing continuous interpolation motion (Path)</td></tr>
</table>
</td></tr>
</table>

Return Value:

Error Code.

Acm_GpResetError function

Format:

U32 Acm_GpResetError(HAND GroupHandle)

Purpose:

Reset the group's state. If the group is in STA_GP_ERROR_STOP state, the state will be changed to STA_GP_READY after calling this function.

Parameters:

Name	Type	In or Out	Description
GroupHandle	HAND	IN	Group handle from Acm_GpAddAxis

Return Value:

Error Code.

Acm_GpMoveLinearRel function

Format:

U32 Acm_GpMoveLinearRel(HAND GroupHandle,PF64 DistanceArray,PU32 pArrayElements)

Purpose:

Command group to execute relative line interpolation.

Parameters:

Name	Type	In or Out	Description
GroupHandle	HAND	IN	Group handle from Acm_GpAddAxis
DistanceArray	PF64	IN	Distance array of axes in group, each value of array elements represent the axis move distance
pArrayElements	PU32	IN	Element count in the array(This count must equal to the axes count in this group)

Return Value:

Error Code.

Commnets:

The sequence of data in **DistanceArray** must follow the order of 0 axis, 1 axis, 2 axis(if there are slave devices) and so on. For example, if one group has two axes: 0 axis and 1 axis. The first data in **DistanceArray** means 0 axis' relative distance and the second data means 1 axis' relative distance.

The difference between line interpolation and direct interpolation: line interpolation executes even line speed, but direct interpolation's line speed is not constant.

Acm_GpMoveLinearAbs function

Format:

U32 Acm_GpMoveLinearAbs(HAND GroupHandle,PF64 PositionArray,PU32 pArrayElements)

Purpose:

Command group to execute absolute line interpolation.

Parameters:

Name	Type	In or Out	Description
GroupHandle	HAND	IN	Group handle from Acm_GpAddAxis
PositionArray	PF64	IN	Position array of axes in group, each value of array elements represent the axis absolute position
pArrayElements	PU32	IN	Element count in the array(This count must equal to the axes count in this group)

Return Value:

Error Code.

Comments:

The sequence of data in **PositionArray** must follow the order of 0 axis, 1 axis, 2 axis(if there are slave devices) and so on. For example, if one group has two axes: 0 axis and 1 axis. The first data in **PositionArray** means 0 axis' absolute position and the second data means 1 axis' absolute position.

The difference between line interpolation and direct interpolation: line interpolation executes even line speed, but direct interpolation's line speed is not constant.

Acm_GpMoveDirectRel function

Format:

U32 Acm_GpMoveDirectRel(HAND GroupHandle, PF64 DistanceArray, PU32 ArrayElements)

Purpose:

Command group to execute relative direct interpolation.

Parameters:

Name	Type	In or Out	Description
GroupHandle	HAND	IN	Group handle from Acm_GpAddAxis
DistanceArray	PF64	IN	Distance array of axes in group, each value of array elements represent the axis move distance
pArrayElements	PU32	IN	Element count in the array(This count must equal to the axes count in this group)

Return Value:

Error Code.

Commnets:

The sequence of data in **DistanceArray** must follow the order of 0 axis, 1 axis, 2 axis(if there are slave devices) and so on. For example, if one group has two axes: 0 axis and 1 axis. The first data in **DistanceArray** means 0 axis' relative distance and the second data means 1 axis' relative distance.

The difference between line interpolation and direct interpolation: line interpolation executes even line speed, but direct interpolation's line speed is not constant.

Acm_GpMoveDirectAbs function

Format:

U32 Acm_GpMoveDirectAbs(HAND GroupHandle, PF64 PositionArray, PU32 ArrayElements)

Purpose:

Command group to execute absolute line interpolation.

Parameters:

Name	Type	In or Out	Description
GroupHandle	HAND	IN	Group handle from Acm_GpAddAxis
PositionArray	PF64	IN	Position array of axes in group, each value of array elements represent the axis absolute position
pArrayElements	PU32	IN	Element count in the array(This count must equal to the axes count in this group)

Return Value:

Error Code.

Comments:

The sequence of data in **PositionArray** must follow the order of 0 axis, 1 axis, 2 axis(if there are slave devices) and so on. For example, if one group has two axes: 0 axis and 1 axis. The first data in **PositionArray** means 0 axis' absolute position and the second data means 1 axis' absolute position.

The difference between line interpolation and direct interpolation: line interpolation executes even line speed, but direct interpolation's line speed is not constant.

Acm_GpMoveCircularRel function

Format:

U32 Acm_GpMoveCircularRel(HAND GroupHandle, PF64 CenterArray, PF64 EndArray, PU32 pArrayElements, I16 Direction)

Purpose:

Command group to execute relative ARC interpolation.

Parameters:

<table>
<tr><th>Name</th><th>Type</th><th>In or Out</th><th>Description</th></tr>
<tr><td>GroupHandle</td><td>HAND</td><td>IN</td><td>Group handle from Acm_GpAddAxis</td></tr>
<tr><td>CenterArray</td><td>PF64</td><td>IN</td><td>Relative distance of center point.</td></tr>
<tr><td>EndArray</td><td>PF64</td><td>IN</td><td>Relative distance of end point.</td></tr>
<tr><td>pArrayElements</td><td>PU32</td><td>IN</td><td>Element count in the array(This count must equal to the axes count in this group)</td></tr>
<tr><td>Direction</td><td>I16</td><td>IN</td><td>Direction
<table>
<tr><th><i>Value</i></th><th><i>Meaning</i></th></tr>
<tr><td>0</td><td>DIR_CW(clockwise)</td></tr>
<tr><td>1</td><td>DIR_CCW(counterclockwise)</td></tr>
</table>
</td></tr>
</table>

Return Value:

Error Code.

Comments:

The sequence of data in **CenterArray** and **EndArray** must follow the order of 2n axis, 2n+1(n=0,1,2,...,15) axis. For example, if one group has 0 axis and 1 axis, the first data in **CenterArray** means 0 axis' center distance and the second data means 1 axis' center distance.

See Also:

Acm_GpMoveCircularAbs_3P, Acm_GpMoveCircularAbs, Acm_GpMoveCircularRel_3P

Acm_GpMoveCircularAbs function

Format:

U32 Acm_GpMoveCircularAbs(HAND GroupHandle, PF64 CenterArray, PF64 EndArray, PU32 pArrayElements, I16 Direction)

Purpose:

Command group to execute absolute ARC interpolation.

Parameters:

<table>
<tr><th>Name</th><th>Type</th><th>In or Out</th><th>Description</th></tr>
<tr><td>GroupHandle</td><td>HAND</td><td>IN</td><td>Group handle from Acm_GpAddAxis</td></tr>
<tr><td>CenterArray</td><td>PF64</td><td>IN</td><td>Absolute position of center point.</td></tr>
<tr><td>EndArray</td><td>PF64</td><td>IN</td><td>Absolute position of end point.</td></tr>
<tr><td>pArrayElements</td><td>PU32</td><td>IN</td><td>Element count in the array(This count must equal to the axes count in this group)</td></tr>
<tr><td>Direction</td><td>I16</td><td>IN</td><td>Direction
<table>
<tr><th><i>Value</i></th><th><i>Meaning</i></th></tr>
<tr><td>0</td><td>DIR_CW(clockwise)</td></tr>
<tr><td>1</td><td>DIR_CCW(counterclockwise)</td></tr>
</table>
</td></tr>
</table>

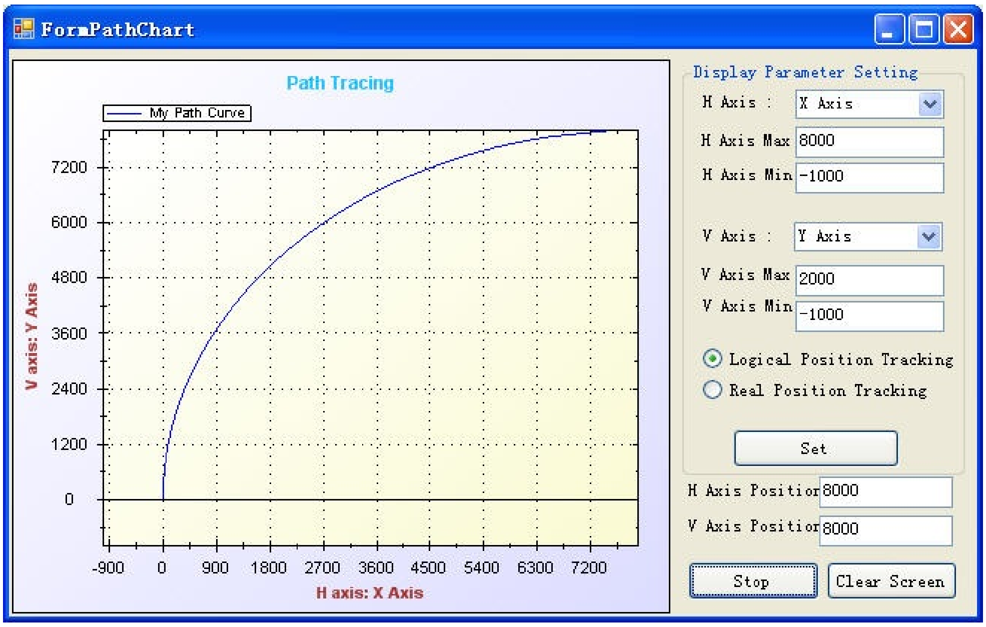
Return Value:

Error Code.

Comments:

The sequence of data in **CenterArray** and **EndArray** must follow the order of 2n axis, 2n+1(n=0,1,2,...,15) axis. For example, if one group has 0 axis and 1 axis, the first data in **CenterArray** means 0 axis' center position and the second data means 1 axis' center position .

Current point will affect absolute ARC interpolation' track. For example, if current point is (0,0),center point is (8000,0), end point is (8000,8000), direction is CW, the track is a quarter of cycle like this:



Display Parameter Setting

H Axis : X Axis

H Axis Max 8000

H Axis Min -1000

V Axis : Y Axis

V Axis Max 2000

V Axis Min -1000

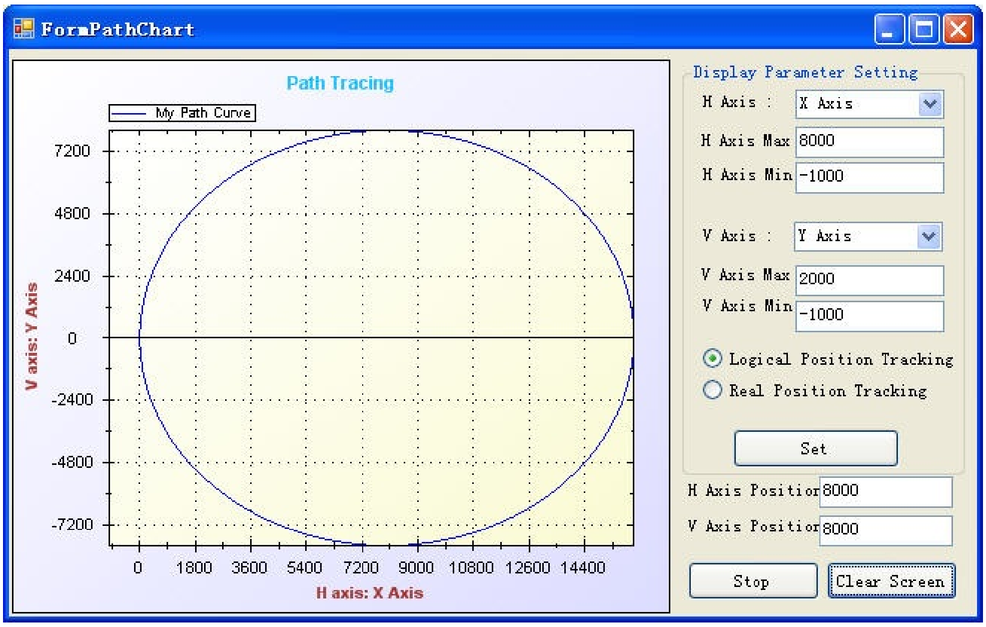
● Logical Position Tracking

○ Real Position Tracking

H Axis Positior 8000

V Axis Positior 8000

If current point is (8000,8000) and other parameters are same with above, the track is a full cycle like this:



H Axis : X Axis

H Axis Max 8000

H Axis Min -1000

V Axis : Y Axis

V Axis Max 2000

V Axis Min -1000

● Logical Position Tracking

○ Real Position Tracking

H Axis Positior 8000

V Axis Positior 8000

See Also:

Acm_GpMoveCircularRel_3P, Acm_GpMoveCircularAbs_3P, Acm_GpMoveCircularRel,

Acm_GpMoveCircularRel_3P function

Format:

U32 Acm_GpMoveCircularRel_3P(HAND GroupHandle, PF64 RefArray, PF64 EndArray, PU32 pArrayElements, I16 Direction)

Purpose:

Command group to execute relative ARC interpolation by three specified points.

Parameters:

Name	Type	In or Out	Description
GroupHandle	HAND	IN	Group handle from Acm_GpAddAxis
RefArray	PF64	IN	Relative distance of reference point.
EndArray	PF64	IN	Relative distance of end point.
pArrayElements	PU32	IN	Element count in the array(This count must equal to the axes count in this group)
Direction	I16	IN	Direction ***Value*** – ***Meaning*** 0 – DIR_CW(clockwise) 1 – DIR_CCW(counterclockwise)

Return Value:

Error Code.

Comments:

The sequence of data in **RefArray** and **EndArray** must follow the order of 2n axis, 2n+1(n=0,1,2,...,15) axis. For example, if one group has 0 axis and 1 axis, the first data in **RefArray** means 0 axis' reference distance and the second data means 1 axis' reference distance. If the axes order in the group is 1 axis and 0 axis, the first data in **RefArray** means 1 axis' reference distance and the second data means 0 axis' reference distance.

See Also:

Acm_GpMoveCircularAbs_3P , Acm_GpMoveCircularRel, Acm_GpMoveCircularAbs

Acm_GpMoveCircularAbs_3P function

Format:

U32 Acm_GpMoveCircularAbs_3P(HAND GroupHandle, PF64 RefArray, PF64 EndArray, PU32 pArrayElements, I16 Direction)

Purpose:

Command group to execute absolute ARC interpolation by three specified points.

Parameters:

Name	Type	In or Out	Description
GroupHandle	HAND	IN	Group handle from Acm_GpAddAxis
RefArray	PF64	IN	Absolute position of reference point.
EndArray	PF64	IN	Absolute position of end point.
pArrayElements	PU32	IN	Element count in the array(This count must equal to the axes count in this group)
Direction	I16	IN	Direction ***Value*** / ***Meaning*** 0 / DIR_CW(clockwise) 1 / DIR_CCW(counterclockwise)

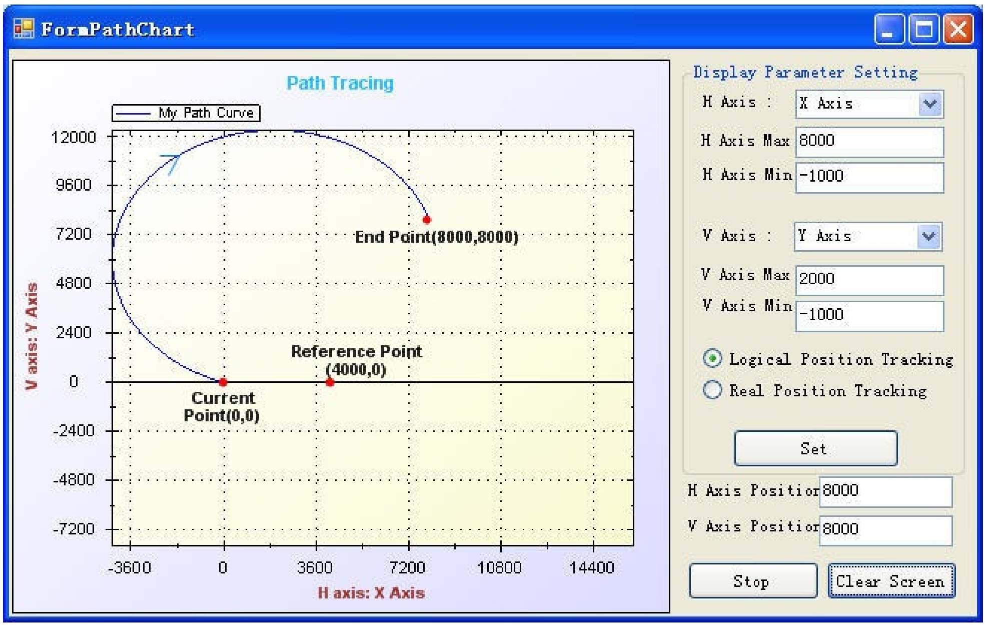
Return Value:

Error Code.

Comments:

The sequence of data in **RefArray** and **EndArray** must follow the order of 2n axis, 2n+1(n=0,1,2,...,15) axis. For example, if one group has 0 axis and 1 axis, the first data in **RefArray** means 0 axis' reference position and the second data means 1 axis' reference position.

Current point will affect absolute ARC interpolation' track. For example, if current point is (0,0), reference point is (4000,0), end point is (8000,8000), the track is like this:



Display Parameter Setting

H Axis : X Axis

H Axis Max 8000

H Axis Min -1000

V Axis : Y Axis

V Axis Max 2000

V Axis Min -1000

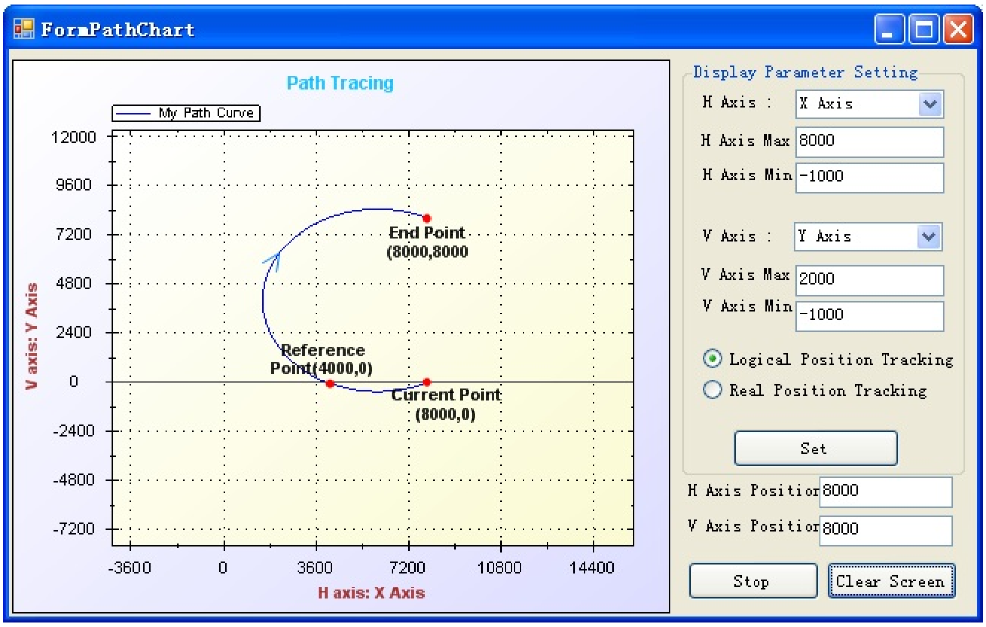
● Logical Position Tracking

○ Real Position Tracking

H Axis Positior 8000

V Axis Positior 8000

If current point is (8000,0) and other parameters are same with above, the track is like this:



H Axis : X Axis

H Axis Max 8000

H Axis Min -1000

V Axis : Y Axis

V Axis Max 2000

V Axis Min -1000

● Logical Position Tracking

○ Real Position Tracking

H Axis Positior 8000

V Axis Positior 8000

See Also:

Acm_GpMoveCircularRel_3P, Acm_GpMoveCircularRel, Acm_GpMoveCircularAbs

Acm_GpChangeVel function

Format:

U32 Acm_GpChangeVel(HAND GroupHandle, F64 NewVelocity)

Purpose:

Command group to change the velocity while group is in line-interpolation motion.

Parameters:

Name	Type	In or Out	Description
GroupHandle	HAND	IN	Group handle from Acm_GpAddAxis
NewVelocity	F64	IN	New velocity

Return Value:

Error Code.

Acm_GpLoadPath function

Format:

U32 Acm_GpLoadPath(HAND GroupHandle, PI8 FilePath, PHAND PathHandle, PU32 pTotalCount)

Purpose:

Load path data from path file. It can load up to 600 path data one time.

Parameters:

Name	Type	In or Out	Description
GroupHandle	HAND	IN	Group handle from Acm_GpAddAxis
FilePath	PI8	IN	Point to a file path name of the motion path data to be loaded.
PathHandle	PHAND	OUT	Return the pointer to path handle
pTotalCount	PU32	OUT	Return actual total count of path data in the path file

Return Value:

Error Code.

Comments:

The path data file(binary) is usually generated by Motion Utility's [**Path Editor**]. If you are familiar with Advantech motion product, you can create file by yourself.

See Also:

Acm_GpUnloadPath,Acm_GpMovePath, Acm_GpGetPathStatus, Acm_GpResetPath, Acm_GpAddPath

Acm_GpUnloadPath function

Format:

U32 Acm_GpUnloadPath(HAND GroupHandle, PHAND PathHandle)

Purpose:

Unload path data.

Parameters:

Name	Type	In or Out	Description
GroupHandle	HAND	IN	Group handle from Acm_GpAddAxis
PathHandle	PHAND	IN	Pointer to path handle from Acm_GpLoadPath

Return Value:

Error Code.

See Also:

Acm_GpLoadPath, Acm_GpMovePath, Acm_GpGetPathStatus, Acm_GpResetPath, Acm_GpAddPath

Acm_GpAddPath function

Format:

U32 Acm_GpAddPath (HAND GroupHandle,U16 MoveCmd,U16 MoveMode,F64 FH,F64 FL, PF64 EndPoint_DataArray,PF64 CenPoint_DataArray,PU32 ArrayElements)

Purpose:

Add an interpolation path to system path buffer. For PCI1220, this system buffer can save up to 6000 path data.

Parameters:

<table>
<tr><th>Name</th><th>Type</th><th>In or Out</th><th>Description</th></tr>
<tr><td>GroupHandle</td><td>HAND</td><td>IN</td><td>Group handle from Acm_GpAddAxis</td></tr>
<tr><td>MoveCmd</td><td>U16</td><td>IN</td><td>Move command
<table>
<tr><th><i>Value</i></th><th><i>Define</i></th></tr>
<tr><td>1</td><td>Abs2DLine</td></tr>
<tr><td>2</td><td>Rel2DLine</td></tr>
<tr><td>3</td><td>Abs2DArcCW</td></tr>
<tr><td>4</td><td>Abs2DArcCCW</td></tr>
<tr><td>5</td><td>Rel2DArcCW</td></tr>
<tr><td>6</td><td>Rel2DArcCCW</td></tr>
<tr><td>7</td><td>Abs3DLine(PCI-1220 not supported)</td></tr>
<tr><td>8</td><td>Rel3DLine(PCI-1220 not supported)</td></tr>
<tr><td>9</td><td>AbsMultiLine</td></tr>
<tr><td>10</td><td>RelMultiLine</td></tr>
</table></td></tr>
<tr><td>MoveMode</td><td>U16</td><td>IN</td><td>Move mode(this parameter is no sense for PCI1220, so 0 or 1 can both be used)
<table>
<tr><th><i>Value</i></th><th><i>Define</i></th></tr>
<tr><td>0</td><td>BlendingEn</td></tr>
<tr><td>1</td><td>BlendingDis</td></tr>
</table></td></tr>
<tr><td>FH</td><td>F64</td><td>IN</td><td>High velocity(driving velocity, only FH parameter of the first path which has not been executed will be used)</td></tr>
<tr><td>FL</td><td>F64</td><td>IN</td><td>Low velocity(start velocity, only FL parameter of the first path which has not been executed will be used)</td></tr>
<tr><td>EndPoint_DataArray</td><td>PF64</td><td>IN</td><td>End point</td></tr>
<tr><td>CenPoint_DataArray</td><td>PF64</td><td>IN</td><td>Center point</td></tr>
<tr><td>ArrayElements</td><td>PU32</td><td>IN</td><td>Number of array element</td></tr>
</table>

Return Value:

Error Code.

comments:

The group handle of every path in system buffer must be the same. So, if there are some unexecuted path in system buffer and you want to add new path into it by call **Acm_GpAddPath,** the parameter **GroupHandle** must be the same with the first unexecuted path's group handle. Otherwise, you have to call Acm_GpResetPath to clear buffer. The current status of system path buffer can be got by call Acm_GpGetPathStatus. Path data in buffer will be loaded to hardware execution registers sequentially after calling Acm_GpMovePath. Even group is executing path motion, you can still call **Acm_GpAddPath** to add new path data into buffer dynamically.

See Also:

Acm_GpLoadPath, Acm_GpUnloadPath,Acm_GpMovePath, Acm_GpGetPathStatus, Acm_GpResetPath

Acm_GpMovePath function

Format:

U32 Acm_GpMovePath(HAND GroupHandle, HAND PathHandle)

Purpose:

Start continuous interpolation motion(Path).

Parameters:

Name	Type	In or Out	Description
GroupHandle	HAND	IN	Group handle from Acm_GpAddAxis
PathHandle	HAND	IN	Path handle from Acm_GpLoadPath or be NULL

Return Value:

Error Code.

Comments:

If the **PathHandle** is returned by Acm_GpLoadPath, the path data will be passed to system path buffer first(the before path data in buffer are all cleared), then driver will load the first path data to start path motion. If the **PathHandle** is NULL, the path data in path buffer will be executed directly.

See Also:

Acm_GpLoadPath, Acm_GpUnloadPath, Acm_GpGetPathStatus, Acm_GpResetPath, Acm_GpAddPath

Acm_GpGetPathStatus function

Format:

U32 Acm_GpGetPathStatus (HAND GroupHandle, PU32 pCurIndex, PU32 pCurCmdFunc, PU32 pRemainCount, PU32 pFreeSpaceCount)

Purpose:

Get current status of path buffer.

Parameters:

<table>
<tr><th>Name</th><th>Type</th><th>In or Out</th><th>Description</th></tr>
<tr><td>GroupHandle</td><td>HAND</td><td>IN</td><td>Group handle from Acm_GpAddAxis</td></tr>
<tr><td>pCurIndex</td><td>PU32</td><td>OUT</td><td>Return current index of path data in path buffer</td></tr>
<tr><td>pCurCmdFunc</td><td>PU32</td><td>OUT</td><td>Return current command function in executing
<table>
<tr><th><i>Value</i></th><th><i>Define</i></th></tr>
<tr><td>0</td><td>PathEnd</td></tr>
<tr><td>1</td><td>Abs2DLine</td></tr>
<tr><td>2</td><td>Rel2DLine</td></tr>
<tr><td>3</td><td>Abs2DArcCW</td></tr>
<tr><td>4</td><td>Abs2DArcCCW</td></tr>
<tr><td>5</td><td>Rel2DArcCW</td></tr>
<tr><td>6</td><td>Rel2DArcCCW</td></tr>
<tr><td>7</td><td>Abs3DLine</td></tr>
<tr><td>8</td><td>Rel3DLine</td></tr>
<tr><td>9</td><td>AbsMultiLine</td></tr>
<tr><td>10</td><td>RelMultiLine</td></tr>
</table>
</td></tr>
<tr><td>pRemainCount</td><td>PU32</td><td>OUT</td><td>Return number of unexecuted path data in path</td></tr>
<tr><td>pFreeSpaceCount</td><td>PU32</td><td>OUT</td><td>Return number of free space in path buffer</td></tr>
</table>

Return Value:

Error Code.

Comments:

Since there is only one path buffer for all groups, no matter what group handle to be passed, the returned status values are all the same.

See Also:

Acm_GpLoadPath, Acm_GpUnloadPath,Acm_GpMovePath, Acm_GpResetPath, Acm_GpAddPath

Acm_GpResetPath function

Format:

U32 Acm_GpResetPath (PHAND GroupHandle)

Purpose:

Clear path buffer. If there is group executing path, the path motion will be stopped.

Parameters:

Name	Type	In or Out	Description
GroupHandle	PHAND	IN	Point to group handle from Acm_GpAddAxis

Return Value:

Error Code.

Comments:

See Also:

Acm_GpLoadPath, Acm_GpUnloadPath,Acm_GpMovePath, Acm_GpGetPathStatus, Acm_GpAddPath

Acm_GpStopDec function

Format:

U32 Acm_GpStopDec (HAND GroupHandle)

Purpose:

Command axes in this group to decelerate to stop.

Parameters:

Name	Type	In or Out	Description
GroupHandle	HAND	IN	Group handle from Acm_GpAddAxis

Return Value:

Error Code.

Acm_GpStopEmg function

Format:

U32 Acm_GpStopEmg(HAND GroupHandle)

Purpose:

Command axes in this group to stop immediately without deceleration.

Parameters:

Name	Type	In or Out	Description
GroupHandle	HAND	IN	Group handle from Acm_GpAddAxis

Return Value:

Error Code.

Property Introduction

Property Type

Property is classified into 3 types categorized by access permission:

Properties	Read/Write	Direct access HW	Description
FT_xxxx	Read	No	Feature property
CFG_xxxx	Read/Write	Yes	Configuration property, you'd better not change it after setting. But some can be updated dynamic in order to implement flexible functions.
PAR_xxxx	Read/Write	No	parameters used by software

If property is categorized according to its belonged object, there are also three types:

Properties	Description
XXX_Devxxx	Device' property
XXX_Axxxx	Axis' property
XXX_Gpxxx	Group's proeprty

Reference function

All Properties can be set by function Acm_SetProperty and can be got by function Acm_GetProperty. The object handle passed to these two functions is different for different property types.

FT_DevFunctionMap Property

Data type:

U32

Access:

Read only

Meaning:

Get device supported functions.1: support, 0: Not support

Bits	***Description***
0	Motion
1	DI
2	DO
3	AI
4	AO
5	Timer
6	Counter
...	Not define
31	Not define

Comments:

For PCI1220 device, this property value is 0x7.

FT_DevIpoTypeMap Property

Data type:

U32

Access:

Read only

Meaning:

Get supported interpolation types. 1: support, 0: Not support.

Bits	***Description***
0	Line interpolation, 2 axes
1	Line interpolation, Multi axes
2~7	Not define
8	Arc interpolation, 2 axes
...	Not define
31	Not define

Comments:

For PCI1220 device, this property value is 0x103.

FT_DevAxesCount Property

Data type:

U32

Access:

Read only

Meaning:

Get axis number of this device.

Comments:

For PCI1220 device, this property value is 0x4.

CFG_AxCmpEnable Property

Data type:

U32

Access:

Read only

Meaning:

Get max value of position counter.

Comments:

For PCI1220 device, this property value is 0x7FFFFFFF.

CFG_DevBoardID Property

Data type:

U32

Access:

Read only

Meaning:

Get Device ID. For PCI1220, this property value will be 0~15.

CFG_DevBaseAddress Property

Data type:

U32

Access:

Read only

Meaning:

For PCI1220 device, return IO base address.

CFG_DevInterrupt Property

Data type:

U32

Access:

Read only

Meaning:

Get Device interrupt number.

CFG_DevBusNumber Property

Data type:

U32

Access:

Read only

Meaning:

Get device bus number.

CFG_DevSlotNumber Property

Data type:

U32

Access:

Read only

Meaning:

Get device slot number.

CFG_DevDllVersion Property

Data type:

char*

Access:

Read only

Meaning:

Get DLL driver's version. The format is:1.0.0.1

CFG_DevDriverVersion Property

Data type:

char*

Access:

Read only

Meaning:

Get SYS driver's version. The format is:1.0.0.1

CFG_DevMasterDev Property

Data type:

U32

Access:

Read only

Meaning:

If the PCI-1220 is a slave device, this property returns which master device it belongs to.

CFG_DevSlaveDevs Property

Data type:

PU32

Access:

Read only

Meaning:

If this PCI-1220 is a master device, this property returns all slave devices belong to it.

CFG_DevBelongsTo Property

Data type:

U32

Access:

Write only

Meaning:

If there are multiple PCI-1220 devices in the ystem, this property can be used to specify which master device this slave PCI-1220 device belongs to. If a valid PCI-1220 device number is set, the PCI-1220 device will be set as the slave of the device with that number; if an invalid PCI-1220 device number is set, the slave PCI-1220 device will be set into master again.

FT_AxFunctionMap Property

Data type:

U32

Access:

Read only

Meaning:

Get the motion functions supported.

Bits	***Description***
0	Inp
1	Alm
2	Erc
3	Sd
4	El
5	SW EL
6	Org
7	EZ
8	Backlash corrective
9	Suppress vibration
10	Home
...	Not define
31	Not define

Comments:

For PCI1220 device, this property value is 0x4F3.

FT_AxMaxVel Property

Data type:

F64

Access:

Read only

Meaning:

Get the feature of max velocity (Unit: Pulse/S)

Comments:

For PCI1220 device, this property value is 4000000.

FT_AxMaxAcc Property

Data type:

F64

Access:

Read only

Meaning:

Get the feature of max acceleration(Unit:Pulse/S^2)

Comments:

For PCI1220 device, this property value is 500000000.

FT_AxMaxDec Property

Data type:

F64

Access:

Read only

Meaning:

Get the feature of max deceleration (Unit:Pulse/S^2)

Comments:

For PCI1220 device, this property value is 500000000.

FT_AxMaxJerk Property

Data type:

F64

Access:

Read only

Meaning:

Get the feature of max jerk (Unit:Pulse/S^3)

Comments:

For PCI1220 device, this property value is 31250000000.

FT_AxPulseInMap Property

Data type:

U32

Access:

Read only

Meaning:

Get the pulse input features supported by this motion device.

Bits	***Description***
0	Mode
1	Logic
2	Source
...	Not define
31	Not define

Comments:

For PCI1220 device, this property value is 0x1.

FT_AxPulseInModeMap Property

Data type:

U32

Access:

Read only

Meaning:

Get pulse input mode supported by PCI1140 axis.

Bits	***Description***
0	1X A/B
1	2X A/B
2	4X A/B
3	CW/CCW
…	Not define
31	Not define

Comments:

For PCI1220 device, this property value is 0xf. Refer to Pulse Config for details.

FT_AxPulseOutMap Property

Data type:

U32

Access:

Read only

Meaning:

Get the pulse output features supported by this motion device.

Bits	***Description***
0	Mode
...	Not define
31	Not define

Comments:

For PCI1220 device, this property value is 0x1.

FT_AxPulseOutModeMap Property

Data type:

F64

Access:

Read only

Meaning:

Get pulse output modes supported by PCI1140 axis.

Bits	***Description***
0	OUT/DIR
1	OUT/DIR, OUT negative logic
2	OUT/DIR, DIR negative logic
3	OUT/DIR, OUT&DIR negative logic
4	CW/CCW
5	CW/CCW, CW&CCW negative logic
6	A/B phase(PCI1220 not support)
7	B/A phase(PCI1220 not support)
...	Not define
31	Not define

Comments:

For PCI1220 device, this property value is 0x3f. Refer to Pulse Config for details.

FT_AxAlmMap Property

Data type:

U32

Access:

Read only

Meaning:

Get the alarm features supported by this motion axis.

Bits	*Description*
0	Enable
1	Logic
2	React
...	Not define
31	Not define

Comments:

For PCI1220 device's axis, this property value is 0x3.

FT_AxInpMap Property

Data type:

U32

Access:

Read only

Meaning:

Get the In-Position features supported by this motion axis.

Bits	***Description***
0	Mode
1	Logic
...	Not define
31	Not define

Comments:

For PCI1220 device's axis, this property value is 0x3.

FT_AxErcMap Property

Data type:

U32

Access:

Read only

Meaning:

Get the ERC features supported by this motion axis.

Bits	***Description***
0	Enable mode
1	Logic
2	On time
3	Off time
...	Not define
31	Not define

Comments:

PCI1220 does not support ERC, so for PCI1220, this property value is 0x0.

FT_AxSdMap Property

Data type:

U32

Access:

Read only

Meaning:

Get the Slow-Down(SD) features supported by this motion axis.

Bits	*Description*
0	Enable
1	Logic
2	React
3	Latch
...	Not define
31	Not define

Comments:

PCI1220 does not support SD, so for PCI1220, this property value is 0x0.

FT_AxElMap Property

Data type:

U32

Access:

Read only

Meaning:

Get the hardware end limit(HEL) features supported by this motion axis.

Bits	***Description***
0	Enable
1	Logic
2	React
...	Not define
31	Not define

Comments:

For PCI1220 device, the HEL function is always enabled, so this property value is 0x6.

FT_AxSwMelMap Property

Data type:

U32

Access:

Read only

Meaning:

Get the software minus limit features supported by the motion axis.

Bits	***Description***
0	Enable
1	React
2	Value
...	Not define
31	Not define

Comments:

For PCI1220 device, this property value is 0x5.

FT_AxSwMelMap Property

Data type:

U32

Access:

Read only

Meaning:

Get the software plus limit features supported by the motion axis.

Bits	***Description***
0	Enable
1	React
2	Value
...	Not define
31	Not define

Comments:

For PCI1220 device, this property value is 0x5.

FT_AxHomeMap Property

Data type:

U32

Access:

Read only

Meaning:

Get the home features supported by this motion axis.

Bits	***Description***
0	Home mode
1	ORG logic
2	EZ Logic
...	Not define
31	Not define

Comments:

For PCI1220 device, this property value is 0x7

CFG_AxPhyID Property

Data type:

U32

Access:

Read only

Meaning:

Get physical ID of the axis.

Value	***Meaning***
0	Master device's X Axis
1	Master device's Y Axis
If exist	
2n	Slave device n's X Axis
2n+1	Slave device n's Y Axis

n=1,2,3,...15

CFG_AxPPU Property

Data type:

U32

Access:

Read & Write

Meaning:

Pulse per unit(PPU), a virtual unit.

Comments:

For PCI1220 device, this property value must be greater than 0. The default value is 10.

This property value's change will affect CFG_AxMaxVel, CFG_AxMaxAcc, CFG_AxMaxDec, PAR_AxVelHigh, PAR_AxVelLow, PAR_AxAcc, PAR_AxDec, PAR_GpVelHigh, PAR_GpVelLow, PAR_GpAcc, PAR_GpDec.

CFG_AxMaxVel Property

Data type:

F64

Access:

Read & Write

Meaning:

Configure the max velocity for the motion axis(Unit:PPU/s).

Comments:

For PCI1220 device, this property 's max value= FT_AxMaxVel / CFG_AxPPU and min value = 1/ CFG_AxPPU.

Since **Rate**(1~500) is determined by this property , it will affect CFG_AxMaxAcc and CFG_AxMaxDec. For example: if **CFG_AxMaxVel** is set to 40,000, it means **Rate**=5, and CFG_AxMaxAcc will not be able to be set to 8,000,000 since the new CFG_AxMaxAcc value requires **Rate** to be 8. **Rate** can not be modified by user, it is just determined by **CFG_AxMaxVel**.

CFG_AxMaxAcc Property

Data type:

F64

Access:

Read & Write

Meaning:

Configure the max acceleration for the motion axis(Unit:PPU/S^2).

Comments:

For PCI1220 device, this property 's max value= FT_AxMaxAcc / **Rate** / 125/CFG_AxPPU and min value = 125/ CFG_AxPPU.

Rate is determined by CFG_AxMaxVel.

CFG_AxMaxDec Property

Data type:

F64

Access:

Read & Write

Meaning:

Configure the max deceleration for the motion axis(Unit:PPU/S^2).

Comments:

For PCI1220 device, this property 's max value= FT_AxMaxDec / **rate** / 125/CFG_AxPPU and min value = 125/ CFG_AxPPU.

Rate is determined by CFG_AxMaxVel.

CFG_AxPulseInMode Property

Data type:

U32

Access:

Read & Write

Meaning:

Setting of encoder feedback pulse input mode.

Value	*Meaning*
0	1X A/B
1	2X A/B
2	4X A/B
3	CW/CCW

Comments:

Details about different modes you can see Pulse Config.

CFG_AxPulseOutMode Property

Data type:

U32

Access:

Read & Write

Meaning:

Setting of command pulse output mode.

Value	***Meaning***
1	OUT/DIR
2	OUT/DIR, OUT negative logic
4	OUT/DIR, DIR negative logic
8	OUT/DIR, OUT&DIR negative logic
16	CW/CCW
32	CW/CCW, CW&CCW negative logic

Comments:

Details about different modes you can see Pulse Config.

CFG_AxAlmEnable Property

Data type:

U32

Access:

Read & Write

Meaning:

Motion Alarm function enable/disable. Alarm is a signal generated by motor drive when motor drive is in alarm status.

Value	***Meaning***
0	Disabled (Default)
1	Enabled

CFG_AxAlmLogic Property

Data type:

U32

Access:

Read & Write

Meaning:

Setting of active logic for Alarm signal.

Value	***Meaning***
0	Low Active
1	High Active

CFG_AxInpEnable Property

Data type:

U32

Access:

Read & Write

Meaning:

In-Position function enable/disable.

Value	*Meaning*
0	Disabled (Default)
1	Enabled

CFG_AxInpLogic Property

Data type:

U32

Access:

Read & Write

Meaning:

Setting of active logic for In-Position signal.

Value	***Meaning***
0	Low Active
1	High Active

CFG_AxElLogic Property

Data type:

U32

Access:

Read & Write

Meaning:

Setting of active logic for hardware limit signal.

Value	***Meaning***
0	Low Active
1	High Active

CFG_AxElReact Property

Data type:

U32

Access:

Read & Write

Meaning:

Setting the reacting mode of EL signal.

Value	***Meaning***
0	Motor immediately stop(Default)
1	Motor decelerate to stop

CFG_AxSwPelEnable Property

Data type:

U32

Access:

Read & Write

Meaning:

Enable/Disable the plus software limit.

Value	***Meaning***
0	Disabled (Default)
1	Enabled

Comments:

When axis is executing homing, the soft limit function is disabled automatically.

CFG_AxSwMelEnable Property

Data type:

U32

Access:

Read & Write

Meaning:

Enable/Disable the minus software limit.

Value	***Meaning***
0	Disabled (Default)
1	Enabled

Comments:

When axis is executing homing, the soft limit function is disabled automatically.

CFG_AxSwPelValue Property

Data type:

I32

Access:

Read & Write

Meaning:

Setting the value of plus software limit. The property value's range is: -2,147,483,648 ~ +2,147,483,647.

CFG_AxSwMelValue Property

Data type:

I32

Access:

Read & Write

Meaning:

Setting the value of minus software limit. The property value's range is: -2,147,483,648 ~ +2,147,483,647.

CFG_AxOrgLogic Property

Data type:

U32

Access:

Read & Write

Meaning:

Setting the active logic for ORG(IN3) signal.

Value	***Meaning***
0	Low Active
1	High Active

CFG_AxEzLogic Property

Data type:

U32

Access:

Read & Write

Meaning:

Setting the active logic for EZ(INOP,INON) signal.

Value	***Meaning***
0	Low Active
1	High Active

CFG_AxHomeResetEnable Property

Data type:

U32

Access:

Read & Write

Meaning:

Enable or Disable reset logical counter after finish Home.

Value	***Meaning***
0	Disabled (Default)
1	Enabled

CFG_AxCmpSrc Property

Data type:

U32

Access:

Read & Write

Meaning:

Configure compare source counter.

Value	***Define***	***Meaning***
0	SRC_COMMAND_POSITION	Command Position(Default)
1	SRC_ACTUAL_POSITION	Actual position

CFG_AxCmpEnable Property

Data type:

U32

Access:

Read & Write

Meaning:

Enable axis comparator function.

Value	***Define***	***Meaning***
0	CMP_DIS	Disabled (Default)
1	CMP_EN	Enabled

Comments:

If property **CFG_AxCmpEnable** is enabled, the properties CFG_AxGenDOEnable, CFG_AxSwMelEnable, CFG_AxSwPelEnable are disabled automatically. Functions Acm_AxSetSVON and Acm_AxDoSetBit will not be able to output signal.

CFG_AxCmpMethod Property

Data type:

U32

Access:

Read & Write

Meaning:

Set or get compare method.

Value	***Define***	***Meaning***
0	MTD_GREATER_POSITION	>= Position Counter(Default)
1	MTD_SMALLER_POSITION	<= Position Counter
2	MTD_DIRECTIONLESS	=Counter (Directionless)(PCI1220 not support this mode)

CFG_AxGenDoEnable Property

Data type:

U32

Meaning:

Enable PCI1220 axis general DO function.

Value	Define	Meaning
0	GEN_DO_DIS	Disabled
1	GEN_DO_EN	Enabled(Default)

Comments:

If property **CFG_AxGenDoEnable** is enabled, the property **CFG_AxCmpEnable** is disabled automatically. Functions Acm_AxSetCmpData, Acm_AxSetCmpTable, Acm_AxSetCmpAuto will not be able to output signal.

CFG_AxExtMasterSrc Property

Data type:

U32

Access:

Read & Write

Meaning:

Axis is controlled by which physical axis' external signal.

Value	***Meaning***
0	Master device's X Axis
1	Master device's Y Axis
If exist	
2n	Slave device n's X Axis
2n+1	Slave device n's Y Axis

n=1,2,3,...16

CFG_AxExtSelEnable Property

Data type:

U32

Access:

Read & Write

Meaning:

Enable external signal selection function.

Value	***Meaning***
0	Disabled (Default)
1	Enabled

Comments:

If one axis' property **CFG_AxExtSelEnable** is enabled, this axis' property CFG_AxExtMasterSrc must set to be 0 or 1. Then U_IN1 and U_IN2 will determine which axis is controlled by the master source.

U-IN2	U-IN1	Driving Axis
0	0	X-axis
0	1	Y-axis

CFG_AxExtPulseNum Property

Data type:

U32

Access:

Read & Write

Meaning:

Set command pulse number when axis' external drive mode is MPG and the A/B or B/A phase signal is triggered. The property value 's range is: 1~10000.

PAR_AxVelHigh Property

Data type:

F64

Access:

Read & Write

Meaning:

Set high velocity (driving velocity) of this axis (Unit:PPU/S). This property value must be smaller than CFG_AxMaxVel and greater than PAR_AxVelLow. The default value is 8000 PPU.

PAR_AxVelLow Property

Data type:

F64

Access:

Read & Write

Meaning:

Set low velocity (start velocity) of this axis (Unit: PPU/S). This property value must be smaller than or equal to PAR_AxVelHigh . The default value is 2000 PPU.

PAR_AxDec Property

Data type:

F64

Access:

Read & Write

Meaning:

Set deceleration of this axis (Unit: PPU/S^2). This property value must be smaller than or equal to CFG_AxMaxDec. The default value is 10000.

PAR_AxAcc Property

Data type:

F64

Access:

Read & Write

Meaning:

Set acceleration of this axis (Unit: PPU/S^2). This property value must be smaller than or equal to CFG_AxMaxAcc . The default value is 10000.

PAR_AxJerk Property

Data type:

F64

Access:

Read & Write

Meaning:

Set the type of velocity profile: **t-curve** or **s-curve**.

Value	***Meaning***
0	T-curve(Default)
1(!=0)	S-curve

Comments:

If PAR_AxJerk is set to be 1, the PAR_AxAcc not means acceleration but max acceleration and PAR_AxDec not means deceleration but max deceleration. If Par_AxVelHigh, Par_AxVelLow, PAR_AxAcc, PAR_AxDec are all same, the acceleration time of **s-curve** is two times of **t-curve** and the deceleration time of **s-curve** is also two times of **t-curve.**

PAR_AxHomeCrossDistance Property

Data type:

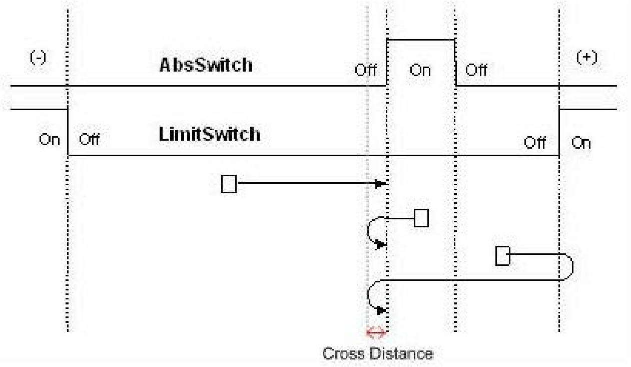
F64

Access:

Read & Write

Meaning:

Set the home cross distance (Unit: Pulse). For PCI1220, this property must be greater than 0. The default value is 100.



PAR_AxHomeExMode Property

Data type:

U32

Access:

Read & Write

Meaning:

Set the extended mode of home. This property value will affect Acm_AxHomeEx.

Value	***Define***	***Meaning***
0	AbsSwitch(Default)	Home operation searching for an absolute switch(ORG). Supported **DirMode**: PosiDir, NegDir, SwitchPosi, SwitchNeg
1	LimitSwitch	Home operation searching for hardware limit switch(EL). Supported **DirMode**: PosiDir, NegDir, SwitchPosi, SwitchNeg
2	RefPulse	Home operation searching for zero pulse in encoder(EZ). Supported **DirMode**: PosiDir, NegDir (If Inupt SwitchPosi, driver will treat it as PosiDir; If Inupt SwitchNeg, driver will treat it as NegDir)

DirMode	***Define***	***Description***
0	PosiDir	Always in positive direction
1	NegDir	Always in negative direction
2	SwitchPosi	Depends on switch status. If switch is off, direction is positive. Otherwise, direction is negative.
3	SwitchNeg	Depends on switch status. If switch is off, direction is negative. Otherwise, direction is positive.

PAR_AxHomeExSwitchMode Property

Data type:

U32

Access:

Read & Write

Meaning:

Setting the stopping condition of Acm_AxHomeEx. PCI1220 only supports **EdgeOn,** so the default value is also 2.

Value	***Define***	***Meaning***
0	LevelOn	When sensor is ON(Active)
1	LevelOff	When sensor is OFF(Non-active)
2	EdgeOn	When OFF to ON transition in sensor
3	EdgeOff	When ON to OFF transition in sensor

PAR_GpGroupID Property

Data type:

U32

Access:

Read only

Meaning:

Get Group ID by specified group handle. The ID uses decimal number starting from 0. This property value needs to be got when Acm_EnableMotionEvent or Acm_CheckMotionEvent is called.

CFG_GpAxesInGroup Property

Data type:

U32

Access:

Read only

Meaning:

Get information about which axes is(are) in this group. For example, if one group has X axis and Y axis, this property value is 0x3.

Bit	31…n	1	0
Description	AxPhyID n	AxPhyID 1	AxPhyID 0

See also:

CFG_AxPhyID

PAR_GpVelLow Property

Data type:

F64

Access:

Read & Write

Meaning:

Set low velocity (start velocity) of this group (Unit: PPU/S). This property value must be smaller than or equal to Par_GpVelHigh . The default value is the first added axis' PAR_AxVelLow.

PAR_GpVelHigh Property

Data type:

F64

Access:

Read & Write

Meaning:

Set high velocity (driving velocity) of this group (Unit:PPU/S). This property value must be smaller than master axis' CFG_AxMaxVel and greater than Par_GpVelLow. The default value is the first added axis' PAR_AxVelHigh. Master axis is the axis with the smallest CFG_AxPhyID.

PAR_GpAcc Property

Data type:

F64

Access:

Read & Write

Meaning:

Set acceleration of this group (Unit: PPU/S^2). This property value must be smaller than or equal to master axis' CFG_AxMaxAcc. The default value is the first added axis' PAR_AxAcc. Master axis is the axis with the smallest CFG_AxPhyID.

PAR_GpDec Property

Data type:

F64

Access:

Read & Write

Meaning:

Set deceleration of this group (Unit: PPU/S^2). This property value must be smaller than or equal to master axis' CFG_AxMaxDec. The default value is the first added axis' PAR_AxDec. Master axis is the axis with the smallest CFG_AxPhyID.

PAR_GpJerk Property

Data type:

F64

Access:

Read & Write

Meaning:

Set the type of velocity profile: **t-curve** or **s-curve**.

Value	***Meaning***
0	T-curve(Default)
1(!=0)	S-curve

Comments:

If PAR_GpJerk is set to 1, the PAR_GpAcc doesn't mean acceleration but max acceleration and PAR_GpDec doesn't means deceleration but max deceleration. If Par_GpVelHigh, Par_GpVelLow, PAR_GpAcc, PAR_GpDec are all the same, the acceleration time of **s-curve** is twice of **t-curve** and the deceleration time of **s-curve** is also twice of **t-curve.**

Data Types

Data types		
TYPE	**Windows Data Type**	**Description**
U8	UCHAR	8 bit unsigned integer
U16	USHORT	16 bit unsigned integer
U32	ULONG	32 bit unsigned integer
U64	ULONGLONG	64 bit unsigned integer
I8	CHAR	8 bit signed integer
I16	SHORT	16 bit signed integer
I32	LONG	32 bit signed integer
I64	LONGLONG	64 bit signed integer
F32	FLOAT	32 bit Floating point variable
F64	DOUBLE	64 bit Floating point variable
PU8	UCHAR*	Pointer to U8
PU16	USHORT*	Pointer to U16
PU32	ULONG*	Pointer to U32
PU64	ULONGLONG*	Pointer to U64
PI8	CHAR*	Pointer to I8
PI16	SHORT*	Pointer to I16
PI32	LONG*	Pointer to I32
PI64	LONGLONG*	Pointer to I64
PF32	FLOAT*	Pointer to F32
PF64	DOUBLE*	Pointer to F64
HAND	U32(U64)	Handle
PHAND	U32*(U64*)	Point to handle
ULONG_PTR	unsigned long*	Point to U32

Notes:

DataType HAND is U32 on 32-bit machine and U64 on 64-bit machine.

Error Codes

Error codes	
0x00000000	SUCCESS
0x10000000	Warning
0x80000000	FuncError
0x80001000	CommError
0x80002000	MotionError
0x80003000	DaqError
0x80004000	DevEvtError
FuncError + 0	InvalidDevNumber
FuncError + 1	DevRegDataLost
FuncError + 2	LoadDllFailed
FuncError + 3	GetProcAddrFailed
FuncError + 4	MemAllocateFailed
FuncError + 5	InvalidHandle
FuncError + 6	CreateFileFailed
FuncError + 7	OpenEventFailed
FuncError + 8	EventTimeOut
FuncError + 9	InvalidInputParam
FuncError + 10	PropertyIDNotSupport
FuncError + 11	PropertyIDReadOnly
FuncError + 12	ConnectWinIrqFailed
FuncError + 13	InvalidAxCfgVel
FuncError + 14	InvalidAxCfgAcc
FuncError + 15	InvalidAxCfgDec
FuncError + 16	InvalidAxCfgJerk
FuncError + 17	InvalidAxParVelLow
FuncError + 18	InvalidAxParVelHigh
FuncError + 19	InvalidAxParAcc
FuncError + 20	InvalidAxParDec
FuncError + 21	InvalidAxParJerk
FuncError + 22	InvalidAxPulseInMode
FuncError + 23	InvalidAxPulseOutMode
FuncError + 24	InvalidAxAlarmEn
FuncError + 25	InvalidAxAlarmLogic
FuncError + 26	InvalidAxInPEn
FuncError + 27	InvalidAxInPLogic
FuncError + 28	InvalidAxHLmtEn
FuncError + 29	InvalidAxHLmtLogic
FuncError + 30	InvalidAxHLmtReact
FuncError + 31	InvalidAxSLmtPEn

FuncError + 32	InvalidAxSLmtPReact
FuncError + 33	InvalidAxSLmtPValue
FuncError + 34	InvalidAxSLmtMEn
FuncError + 35	InvalidAxSLmtMReact
FuncError + 36	InvalidAxSLmtMValue
FuncError + 37	InvalidAxOrgLogic
FuncError + 38	InvalidAxOrgEnable
FuncError + 39	InvalidAxEzLogic
FuncError + 40	InvalidAxEzEnable
FuncError + 41	InvalidAxEzCount
FuncError + 42	InvalidAxState
FuncError + 43	InvalidAxInEnable
FuncError + 44	InvalidAxSvOnOff
FuncError + 45	InvalidAxDistance
FuncError + 46	InvalidAxPosition
FuncError + 47	InvalidAxHomeModeKw
FuncError + 48	InvalidAxCntInGp
FuncError + 49	AxInGpNotFound
FuncError + 50	AxIsInOtherGp
FuncError + 51	AxCannotIntoGp
FuncError + 52	GpInDevNotFound
FuncError + 53	InvalidGpCfgVel
FuncError + 54	InvalidGpCfgAcc
FuncError + 55	InvalidGpCfgDec
FuncError + 56	InvalidGpCfgJerk
FuncError + 57	InvalidGpParVelLow
FuncError + 58	InvalidGpParVelHigh
FuncError + 59	InvalidGpParAcc
FuncError + 60	InvalidGpParDec
FuncError + 61	InvalidGpParJerk
FuncError + 62	JerkNotSupport
FuncError + 63	ThreeAxNotSupport
FuncError + 64	DevIpoNotFinished
FuncError + 65	InvalidGpState
FuncError + 66	OpenFileFailed
FuncError + 67	InvalidPathCnt
FuncError + 68	InvalidPathHandle
FuncError + 69	InvalidPath
FuncError + 70	IoctlError
FuncError + 71	AmnetRingUsed
FuncError + 72	DeviceNotOpened

FuncError + 73	InvalidRing
FuncError + 74	InvalidSlaveIP
FuncError + 75	InvalidParameter
FuncError + 76	InvalidGpCenterPosition
FuncError + 77	InvalidGpEndPosition
FuncError + 78	InvalidAddress
FuncError + 79	DeviceDisconnect
FuncError + 80	DataOutBufExceeded
FuncError + 81	SlaveDeviceNotMatch
FuncError + 82	SlaveDeviceError
FuncError + 83	SlaveDeviceUnknow
FuncError + 84	FunctionNotSupport
FuncError + 85	InvalidPhysicalAxis
FuncError + 86	InvalidVelocity
FuncError + 87	InvalidAxPulseInLogic
FuncError + 88	InvalidAxPulseInSource
FuncError + 89	InvalidAxErcLogic
FuncError + 90	InvalidAxErcOnTime
FuncError + 91	InvalidAxErcOffTime
FuncError + 92	InvalidAxErcEnableMode
FuncError + 93	InvalidAxSdEnable
FuncError + 94	InvalidAxSdLogic
FuncError + 95	InvalidAxSdReact
FuncError + 96	InvalidAxSdLatch
FuncError + 97	InvalidAxHomeResetEnable
FuncError + 98	InvalidAxBacklashEnable
FuncError + 99	InvalidAxBacklashPulses
FuncError + 100	InvalidAxVibrationEnable
FuncError + 101	InvalidAxVibrationRevTime
FuncError + 102	InvalidAxVibrationFwdTime
FuncError + 103	InvalidAxAlarmReact
FuncError + 104	InvalidAxLatchLogic
FuncError + 105	InvalidFwMemoryMode
FuncError + 106	InvalidConfigFile
FuncError + 107	InvalidAxEnEvtArraySize
FuncError + 108	InvalidAxEnEvtArray
FuncError + 109	InvalidGpEnEvtArraySize
FuncError + 110	InvalidGpEnEvtArray
FuncError + 111	InvalidIntervalData
FuncError + 112	InvalidEndPosition
FuncError + 113	InvalidAxisSelect

FuncError + 114	InvalidTableSize
FuncError + 115	InvalidGpHandle
FuncError + 116	InvalidCmpSource
FuncError + 117	InvalidCmpMethod
FuncError + 118	InvalidCmpPulseMode
FuncError + 119	InvalidCmpPulseLogic
FuncError + 120	InvalidCmpPulseWidth
FuncError + 121	InvalidPathFunctionID
FuncError + 122	SysBufAllocateFailed
AMONet Communication error	
FuncError + 150	SlaveIOUpdateError
FuncError + 151	NoSlaveDevFound
FuncError + 152	MasterDevNotOpen
FuncError + 153	MasterRingNotOpen
DAQ function error	
FuncError + 200	InvalidDIPort
FuncError + 201	InvalidDOPort
FuncError + 202	InvalidDOValue
Event Error	
FuncError + 203	CreateEventFailed
FuncError + 204	CreateThreadFailed
FuncError + 205	InvalidHomeModeEx
FuncError + 206	InvalidDirMode
FuncError + 207	AxHomeMotionFailed
FuncError + 208	ReadFileFailed
FuncError + 209	PathBufIsFull
FuncError + 210	PathBufIsEmpty
MotionError + 0	HLmtPExceeded
MotionError + 1	HLmtNExceeded
MotionError + 2	SLmtPExceeded
MotionError + 3	SLmtNExceeded
MotionError + 4	AlarmHappened
MotionError + 5	EmgHappened
MotionError + 6	TimeLmtExceeded
MotionError + 7	DistLmtExceeded
MotionError + 8	InvalidPositionOverride
MotionError + 9	OperationErrorHappened
MotionError + 10	SimultaneousStopHappened
MotionError + 11	OverflowInPAPB
MotionError + 12	OverflowInIPO
MotionError + 13	STPHappened

MotionError + 14	SDHappened
MotionError + 15	AxsiNoCmpDataLeft

Abbreviations

PCS	Position Change Signal Input (not support at present)
EL	End limit, indicate the limit of motion in plus direction or minus direction
ORG	Home Signal Input, indicating the origin of the system
SD	Ramp-Down Signal Input
ALM	Servo Alarm Signal
INP	Servo In Position Signal
RDY	Servo Ready Signal
LTC	Position Latch Signal Input
EMG	Emergency Stop Signal
ERC	Clear Servo Error Counter Signal Output
RALM	Reset the ALM status inside the servo driver
OUT	Pulse Signal Output
EA	Encode A Phase
EB	Encode B Phase
EZ	Encode Z phase
DIR	Direction Signal Output
CMP	Position Compare Output