PCI-8102
Advanced 2-Axis Servo / Stepper Motion Control Card
User's Guide

Recycle Paper

Manual Rev. 0.1: Feb 22, 2006

Part No: 50-11136-xxx

Trademarks

Getting service from ADLINK

Customer Satisfaction is the most important priority for ADLINK Tech Inc. If you need any help or service, please contact us.

ADLINK Technology Inc.			
Web Site	http://www.adlinktech.com		
Sales & Service	Service@adlinktech.com		
TEL	+886-2-82265877	FAX	+886-2-82265717
Address	9F, No. 166, Jian Yi Road, Chungho City, Taipei, 235 Taiwan.		

Please email or FAX us your detailed information for prompt, satisfactory, and consistent service.

Detailed Company Information			
Company/Organization			
Contact Person			
E-mail Address			
Address			
Country			
TEL		FAX	
Web Site			
Questions			
Product Model			
Environment	OS: Computer Brand: M/B: Chipset: Video Card: NIC: Other:	 CPU: BIOS:	
Detail Description			
Suggestions for ADLINK			

Table of Contents

How to Use This Guide

This manual is designed to help you use the PCI-8102 and describes how to modify various settings to meet your requirements. It is divided into seven chapters:

Chapter 1	Introduction An overview of the product features, applications, and specifications.
Chapter 2	Installation Describes how to install the PCI-8102.
Chapter 3	Signal Connections Details the connector pin assignments and how to connect external signals and devices to the PCI-8102.
Chapter 4	Operation Theory Describes the operations of the PCI-8102.
Chapter 5	MotionCreatorPro Details how to use the Windows based utility program to configure and run tests with the PCI-8102.
Chapter 6	C/C++ Function Library Describes high-level programming in C/C++ to aid in programming the PCI-8102.
Chapter 7	Connection Example Illustrates typical connection examples between the PCI-8102 and servo/stepping drivers.

1

Introduction

The PCI-8102 is an advanced 2-axis motion controller card with a PCI interface. It can generate high frequency pulses (6.55MHz) to drive stepper or servomotors. As a motion controller, it can provide 2-axis linear and circular interpolation and continuous interpolation for continuous velocity. Also, changing position/speed on the fly is available with a single axis operation.

Multiple PCI-8102 cards can be used in one system. Incremental encoder interface on all four axes provide the ability to correct positioning errors generated by inaccurate mechanical transmissions. PCI-8102 features the postion compare and trigger output function which users can put the comparing points with ADLINK library and sending the triggering pulse to other device. In addition, a mechanical sensor interface, servo motor interface, and general-purposed I/O signals are provided for easy system integration.

Figure 1 shows the functional block diagram of the PCI-8102 card. The motion control functions include trapezpoidal and S-curve acceleration/deceleration, linear and circular interpolation between two axes and continuous motion positioning, and 13 home return modes. All these functions and complex computations are performed internally by the ASIC, thus it can save CPU loading.

Except this, the PCI-8102 also offers three user-friendly functions. PCI-8102 can let users assign the card index with the DIP switch setting. The value is within 0 to 15. It is useful for machine makers to recognize the card index if the whole control system is very huge. The emergency input pin can let users wire the emergency botton to trigger this board to stop sending pulse output once there is any emgergency sitiuation happened. For security protection design, users can set the 16-bit value into EEPROM. Users' interface program can uses this EEPROM to secure the software and hardware in order to prevent plagiarist.

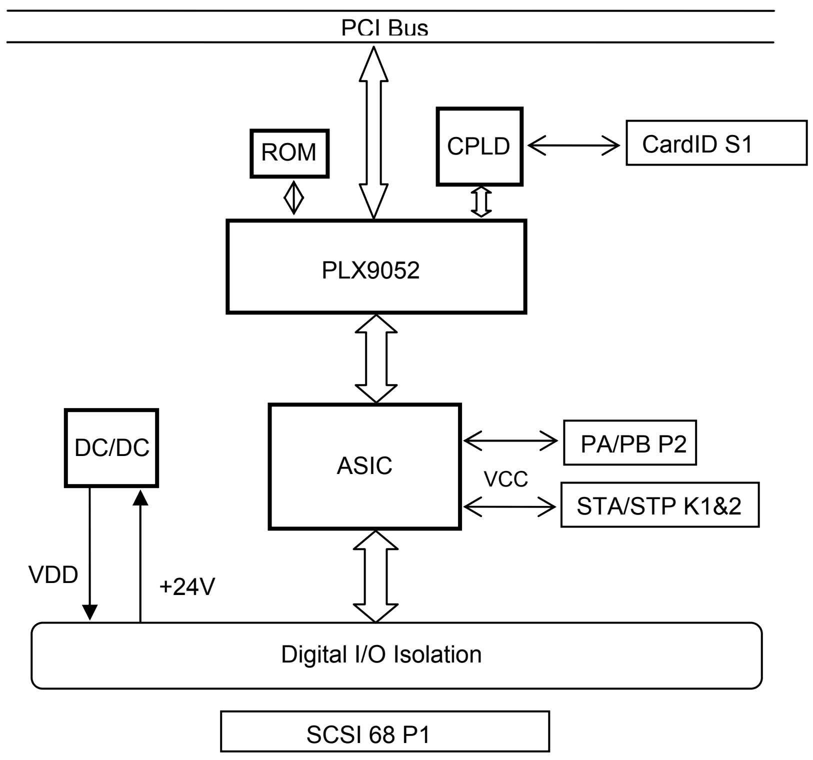


Figure 1: Block Diagram of the PCI-8102

MotionCreatorPro is a Windows-based application development software package included with the PCI-8102. *MotionCreatorPro* is useful for debugging a motion control system during the design phase of a project. An on-screen display lists all installed axes information and I/O signal status of the PCI-8102.

Windows programming libraries are also provided for C++ compiler and Visual Basic. Sample programs are provided to illustrate the operations of the functions.

Figure 2 illustrates a flow chart of the recommended process in using this manual in developing an application. Refer to the related chapters for details of each step.

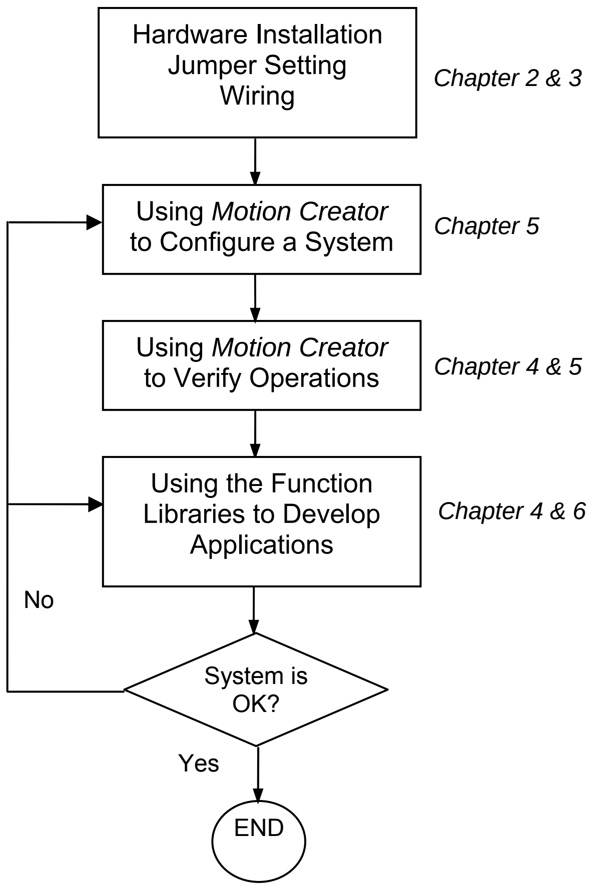

Hardware Installation
Jumper Setting
Wiring
Chapter 2 & 3
Using Motion Creator to Configure a System
Chapter 5
Using Motion Creator to Verify Operations
Chapter 4 & 5
Using the Function Libraries to Develop Applications
Chapter 4 & 6
No
System is OK?
Yes
END

Figure 2: Flow chart for building an application

1.1 Features

The following list summarizes the main features of the PCI-8102 motion control system.

- 32-bit PCI bus Plug and Play (Universal)
- 2 axes of step and direction pulse output for controlling stepping or servomotor
- Maximum output frequency of 6.55 MPPS
- Pulse output options: OUT/DIR, CW/CCW
- Programmable acceleration and deceleration time for all modes
- Trapezoidal and S-curve velocity profiles for all modes
- 2 axes linear / circular interpolation
- Continuous interpolation for contour following motion
- Change position and speed on the fly
- 13 home return modes with auto searching
- Hardware backlash compensator and vibration suppression
- 2 software end-limits for each axis
- 28-bit up/down counter for incremental encoder feedback
- Home switch, index signal (EZ), positive, and negative end limit switches interface on all axes
- 2-axis high speed position latch input
- 2-axis position compare and trigger output
- All digital input and output signals are 2500Vrms isolated
- Programmable interrupt sources
- Simultaneous start/stop motion on multiple axes
- Manual pulser input interface
- Card index selection
- Security protection on EERPOM
- Dedicated emergency input pin for wiring
- Software supports a maximum of up to 12 PCI-8102 cards operation in one system
- Compact PCB design
- Includes *MotionCreatorPro*, a Microsoft Windows-based application development software
- PCI-8102 libraries and utilities for Windows 2000/XP.

1.2 Specifications

- **Applicable Motors:**
  - Stepping motors
  - AC or DC servomotors with pulse train input servo drivers
- **Performance:**
  - Number of controllable axes: 2
  - Maximum pulse output frequency: 6.55MPPS, linear, trapezoidal, or S-Curve velocity profile drive
  - Internal reference clock: 19.66 MHz
  - 28-bit up/down counter range: 0-268,435,455 or –134,217,728 to +134,217,727
  - Position pulse setting range (28-bit): -134,217,728 to +134,217,728
  - Pulse rate setting range (Pulse Ratio = 1: 65535):

    0.1 PPS to 6553.5 PPS. (Multiplier = 0.1)

    1 PPS to 65535 PPS. (Multiplier = 1)

    100 PPS to 6553500 PPS. (Multiplier = 100)
- **I/O Signales:**
  - Input/Output signals for each axis
  - All I/O signal are optically isolated with 2500Vrms isolation voltage
  - Command pulse output pins: OUT and DIR
  - Incremental encoder signals input pins: EA and EB
  - Encoder index signal input pin: EZ
  - Mechanical limit/switch signal input pins: ±EL, SD/PCS, and ORG
  - Servomotor interface I/O pins: INP, ALM, and ERC
  - Position latch input pin: LTC (the same pin with SD/PCS)
  - Position compare TTL output pin: CMP
  - General-purposed digital output pin: SVON
  - General-purposed digital input pin: RDY
  - Pulse signal input pin: PA and PB (With Isolation)
  - Simultaneous Start/Stop signal: STA and STP
  - Emergency input signal: EMG

General-Purposed Input / Output

- 4 digital inputs / 2 digital outputs

◆ **General Specifications**

- Connectors: 68-pin SCSI-type connector
- Operating Temperature: 0°C - 50°C
- Storage Temperature: -20°C - 80°C
- Humidity: 5 - 85%, non-condensing

◆ **Power Consumption**

- Slot power supply (input): +5V DC ±5%, 900mA max
- External power supply (input): +24V DC ±5%, 500mA max
- External power supply (output): +5V DC ±5%, 500mA, max

◆ **PCI-8102 Dimension (PCB size): 120mm(L) X 100 mm(W)**

1.3 Supported Software

1.3.1 Programming Library

Windows 2000/XP DLLs are provided for the PCI-8102 users. These function libraries are shipped with the board.

1.3.2 MotionCreatorPro

This Windows-based utility is used to setup cards, motors, and systems. It can also aid in debugging hardware and software problems. It allows users to set I/O logic parameters to be loaded in their own program. This product is also bundled with the card.

Refer to Chapter 5 for more details.

1.4 Available Terminal Board

DIN-68S and DIN-68M-J3A are available for PCI-8102 wiring. DIN-68S is general purposed pin-to-pin terminal board. Users can use this to connect servos and steppers. DIN-68M-J3A is designed for Mitsubishi servo J3A amplifer. Users can user this board to connect Mitsubishi servo J3A and steppers.

2

Installation

This chapter describes how to install the PCI-8102. Please follow these steps below:

- Check what you have (section 2.1)
- Check the PCB (section 2.2)
- Install the hardware (section 2.3)
- Install the software driver (section 2.4)
- Understanding the I/O signal connections (chapter 3) and their operation (chapter 4)
- Understanding the connector pin assignments (the remaining sections) and wiring the connections

2.1 Package Contents

In addition to this *User's Guide*, the package also includes the following items:

- PCI-8102: advanced 2-Axis Servo / Stepper Motion Control Card
- ADLINK All-in-one Compact Disc

Terminal board is optional accessory. This would not be included in PCI-8102 package.

If any of these items are missing or damaged, contact the dealer from whom you purchased the product. Save the shipping materials and carton to ship or store the product in the future.

2.2 PCI-8102 Outline Drawing

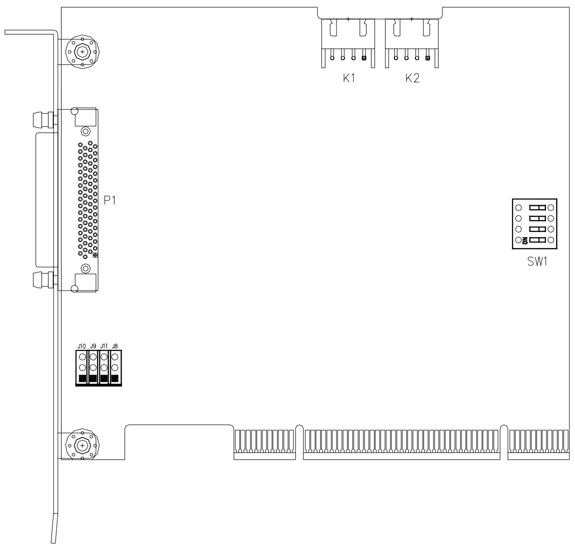


Figure 3: PCB Layout of the PCI-8102

P1: Input / Output Signal Connector (68-pin)

K1 / K2: Simultaneous Start / Stop Connector

SW1: DIP switch for card index selection (0-15)

J8~J11: Pulse output selection jumper

2.3 PCI-8102 Hardware Installation

2.3.1 Hardware configuration

The PCI-8102 is fully Plug and Play compliant. Hence memory allocation (I/O port locations) and IRQ channel of the PCI card are assigned by the system BIOS. The address assignment is done on a board-by-board basis for all PCI cards in the system.

2.3.2 PCI slot selection

Your computer system may have both PCI and ISA slots. Do not force the PCI card into a PC/AT slot. The PCI-8102 can be used in any PCI slot.

2.3.3 Installation Procedures

1. Read through this manual and setup the jumper according to your application
2. Turn off your computer. Turn off all accessories (printer, modem, monitor, etc.) connected to computer. Remove the cover from your computer.
3. Select a 32-bit PCI expansion slot. PCI slots are shorter than ISA or EISA slots and are usually white or ivory.
4. Before handling the PCI-8102, discharge any static buildup on your body by touching the metal case of the computer. Hold the edge of the card and do not touch the components.
5. Position the board into the PCI slot you have selected.
6. Secure the card in place at the rear panel of the system unit using screws removed from the slot.

2.3.4 Troubleshooting

If your system doesn’t boot or if you experience erratic operation with your PCI board in place, it’s most likely caused by an interrupt conflict (possibly an incorrect ISA setup). In general, the solution, once determined it is not a simple oversight, is to consult the BIOS documentation that comes with your system.

Check the control panel of the Windows system if the card is listed by the system. If not, check the PCI settings in the BIOS or use another PCI slot.

2.4 Software Driver Installation

PCI-8102:

Step 1: Auto run the ADLINK All-In-One CD. Choose Driver Installation -> Motion Control -> PCI-8102

Step 2: Follow the procedures of the installer.

Step 3: After setup installation is completed, restart windows.

Suggestion: Please download the latest software from ADLINK website if necessary.

2.5 P1 Pin Assignments: Main Connector

P1 is the major connector for the motion control I/O signals.

No.	Name	I/O	Function Axis 0	No.	Name	I/O	Function Axis 1
1	VPP	O	Isolated +5V Output	35	VPP	O	Isolated +5V Output
2	EXGND	-	Ext. power ground	36	EXGND	-	Ext. power ground
3	OUT0+	O	Pulse signal (+)	37	OUT1+	O	Pulse signal (+)
4	OUT0-	O	Pulse signal (-)	38	OUT1-	O	Pulse signal (-)
5	DIR0+	O	Dir. signal (+)	39	DIR1+	O	Dir. signal (+)
6	DIR0-	O	Dir. signal (-)	40	DIR1-	O	Dir. signal (-)
7	SVON0	O	Servo On/Off	41	SVON1	O	Servo On/Off
8	ERC0	O	Dev. ctr, clr. signal	42	ERC1	O	Dev. ctr, clr. Signal
9	ALM0	I	Alarm signal	43	ALM1	I	Alarm signal
10	INP0	I	In-position signal	44	INP1	I	In-position signal
11	RDY0	I	Multi-purpose input signal	45	RDY1	I	Multi-purpose input signal
12	EA0+	I	Encoder A-phase (+)	46	EA1+	I	Encoder A-phase (+)
13	EA0-	I	Encoder A-phase (-)	47	EA1-	I	Encoder A-phase (-)
14	EB0+	I	Encoder B-phase (+)	48	EB1+	I	Encoder B-phase (+)
15	EB0-	I	Encoder B-phase (-)	49	EB1-	I	Encoder B-phase (-)
16	EZ0+	I	Encoder Z-phase (+)	50	EZ1+	I	Encoder Z-phase (+)
17	EZ0-	I	Encoder Z-phase (-)	51	EZ1-	I	Encoder Z-phase (-)
18	VPP	O	Isolated +5V Output	52	VPP	O	Isolated +5V Output
19	N/C			53	EXGND	-	Ext. power ground
20	PEL0	I	End limit signal (+)	54	PEL1	I	End limit signal (+)
21	MEL0	I	End limit signal (-)	55	MEL1	I	End limit signal (-)
22	EXGND	-	Ext. power ground	56	EXGND	-	Ext. power ground
23	LTC/SD/PCS0/CLR0	I	Composite Funtion (Default: LTC)	57	LTC/SD/PCS1/CLR1	I	Composite Funtion (Default: LTC)
24	ORG0	I	Origin signal	58	ORG1	I	Origin signal
25	N/C			59	EXGND	-	Ext. power ground
26	PA+_ISO	I	Manual Pulser Input A	60	EMG	I	Emergency Input
27	PA-_ISO	I	Manual Pulser Input A	61	DIN0	I	Digital Input 0
28	PB+_ISO	I	Manual Pulser Input B	62	DIN1	I	Digital Input 1
29	PB-_ISO	I	Manual Pulser Input B	63	DIN2	I	Digital Input 2
30	CMP0	O	TTL Compare Output 0	64	DIN3	I	Digital Input 3
31	CMP1	O	TTL Compare Output 1	65	DOUT0	O	Digital Output 0,SVO RST
32	EXGND	-	Ext. power ground	66	DOUT1	O	Digital Output 1,SVO RST
33	EXGND	-	Ext. power ground	67	EXGND	-	Ext. power ground
34	EX+24V	I	+24V isolation power input	68	EX+24V	I	+24V isolation power input

2.6 K1/K2 Pin Assignments: Simultaneous Start/Stop

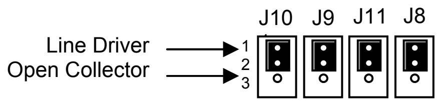
CN4 is for simultaneous start/stop signals for multiple axes or multiple cards.

No.	Name	Function (Axis)
1	+5V	PCI Bus power Output (VCC)
2	STA	Simultaneous start signal input/output
3	STP	Simultaneous stop signal input/output
4	GND	PCI Bus power ground

Note: +5V and GND pins are provided by the PCI Bus power.

2.7 Jumper Setting for Pulse Output

J8-J11 are used to set the type of pulse output signals (DIR and OUT). The output signal type can either be differential line driver or open collector output. Refer to section 3.1 for detail jumper settings. The default setting is differential line driver mode. J8 & J9 are for axis 0; J10 & J11 are for axis 1.



2.8 Switch Setting for card index

The SW1 switch is used to set the card index. For example, if you turn 1 to ON and others are OFF. It means the card index as 1. The value is from 0 to 15. Refer to the following table for details.

Card ID	Switch Setting (ON=1)
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000

9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

3

Signal Connections

Signal connections of all I/O's are described in this chapter. Refer to the contents of this chapter before wiring any cable between the PCI-8102 and any motor driver.

This chapter contains the following sections:

3.1 Pulse Output Signals OUT and DIR

There are 2 axis pulse output signals on the PCI-8102. For each axis, two pairs of OUT and DIR differential signals are used to transmit the pulse train and indicate the direction. The OUT and DIR signals can also be programmed as CW and CCW signal pairs. Refer to section 4.1.1 for details of the logical characteristics of the OUT and DIR signals. In this section, the electrical characteristics of the OUT and DIR signals are detailed. Each signal consists of a pair of differential signals. For example, OUT0 consists of OUT0+ and OUT0- signals. The following table shows all pulse output signals on P1.

P1 Pin No.	Signal Name	Description	Axis #
3	**OUT0+**	Pulse signals (+)	①
4	**OUT0-**	Pulse signals (-)	①
5	**DIR0+**	Direction signal (+)	①
6	**DIR0-**	Direction signal (-)	①
37	**OUT1+**	Pulse signals (+)	②
38	**OUT1-**	Pulse signals (-)	②
39	**DIR1+**	Direction signal (+)	②
40	**DIR1-**	Direction signal (-)	②

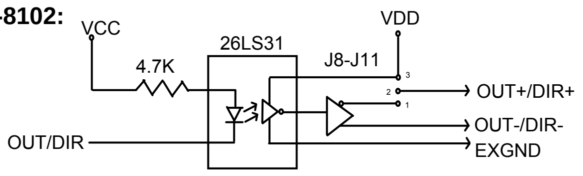
The output of the OUT or DIR signals can be configured by jumpers as either differential line drivers or open collector output. Users can select the output mode either by jumper wiring between 1 and 2 or 2 and 3 of jumpers J8-J11 as follows:

Output Signal	For differential line driver output, close breaks between 1 and 2 of:	For open collector output, close breaks between 2 and 3 of:
OUT0+	J8	J8
DIR0+	J9	J9
OUT1+	J10	J10
DIR1+	J11	J11

The **default** setting of OUT and DIR is set to differential line driver mode.

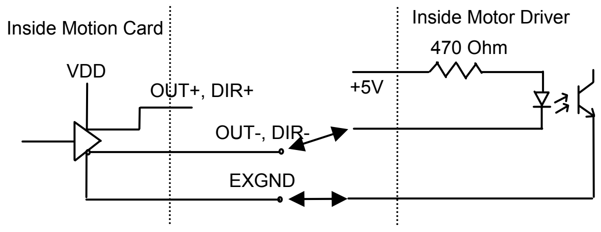
The following wiring diagram is for OUT and DIR signals on the 2 axes.

PCI-8102:



NOTE: If the pulse output is set to open collector output mode, OUT- and DIR- are used to transmit OUT and DIR signals. ***The sink current must not exceed 20mA on the OUT- and DIR- pins***. The default setting is 1-2 shorted.

Suggest Usage: Jumper 2-3 shorted and connect OUT-/DIR- to a 470 ohm pulse input interface's COM of driver. See the following figure. Choose OUT-/DIR- to connect to driver's OUT/DIR



Warning: The sink current must not exceed 20mA or the 26LS31 will be damaged!

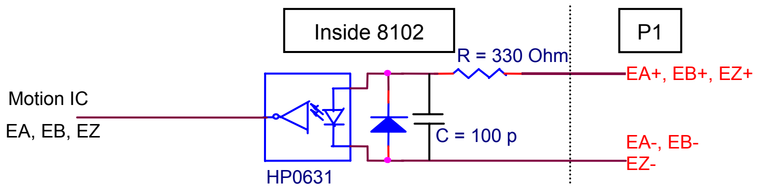
3.2 Encoder Feedback Signals EA, EB and EZ

The encoder feedback signals include EA, EB, and EZ. Every axis has six pins for three differential pairs of phase-A (EA), phase-B (EB), and index (EZ) inputs. EA and EB are used for position counting, and EZ is used for zero position indexing. Its relative signal names, pin numbers, and axis numbers are shown in the following tables:

P1 Pin No	Signal Name	Axis #	P1 Pin No	Signal Name	Axis #
12	EA0+	①	13	EA0-	①
14	EB0+	①	15	EB0-	①
46	EA1+	②	47	EA1-	②
48	EB1+	②	49	EB1-	②

P1 Pin No	Signal Name	Axis #	P1 Pin No	Signal Name	Axis #
16	EZ1+	①	17	EZ3+	①
50	EZ1-	②	51	EZ3-	②

The input circuit of the EA, EB, and EZ signals is shown as follows:

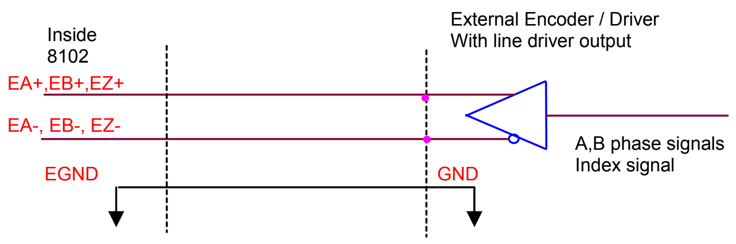


Please note that the voltage across each differential pair of encoder input signals (EA+, EA-), (EB+, EB-), and (EZ+, EZ-) should be at least 3.5V. Therefore, the output current must be observed when connecting to the encoder feedback or motor driver feedback as not to over drive the source. The differential signal pairs are converted to digital signals EA, EB, and EZ; then feed to the motion control ASIC.

Below are examples of connecting the input signals with an external circuit. The input circuit can be connected to an encoder or motor driver if it is equipped with: (1) a differential line driver or (2) an open collector output.

◆ Connection to Line Driver Output

To drive the PCI-8102 encoder input, the driver output must provide at least 3.5V across the differential pairs with at least 8mA driving capacity. The grounds of both sides must be tied together. The maximum frequency is 4Mhz or more depends on wiring distance and signal conditioning.

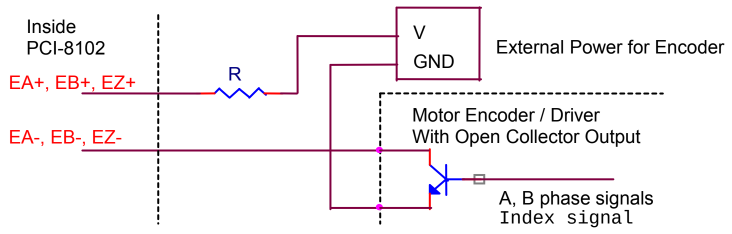


◆ Connection to Open Collector Output

To connect with an open collector output, an external power supply is necessary. Some motor drivers can provide the power source. The connection between the PCI-8102, encoder, and the power supply is shown in the diagram below. Note that an external current limiting resistor R is necessary to protect the PCI-8102 input circuit. The following table lists the suggested resistor values according to the encoder power supply.

Encoder Power (V)	External Resistor R
+5V	0Ω (None)
+12V	1.5kΩ
+24V	3.0kΩ

$I_f = 8mA$



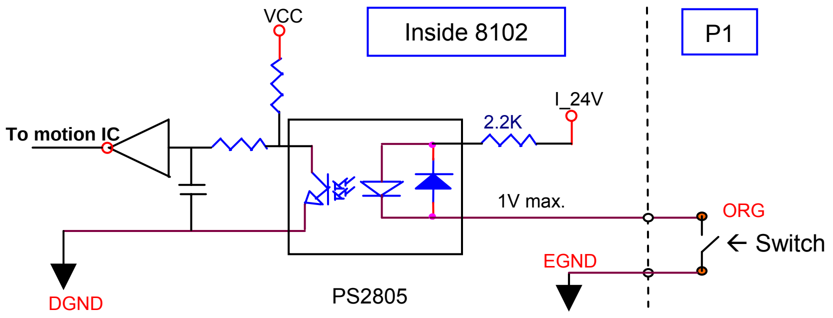
For more operation information on the encoder feedback signals, refer to section 4.4.

3.3 Origin Signal ORG

The origin signals (ORG0~ORG1) are used as input signals for the origin of the mechanism. The following table lists signal names, pin numbers, and axis numbers:

P1 Pin No	Signal Name	Axis #
24	ORG0	①
58	ORG1	②

The input circuit of the ORG signals is shown below. Usually, a limit switch is used to indicate the origin on one axis. The specifications of the limit switch should have contact capacity of +24V @ 6mA minimum. An internal filter circuit is used to filter out any high frequency spikes, which may cause errors in the operation.



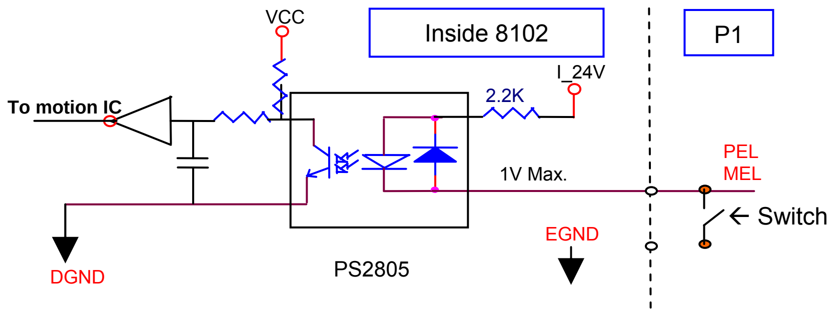
When the motion controller is operated in the home return mode, the ORG signal is used to inhibit the control output signals (OUT and DIR). For detailed operations of the ORG signal, refer to section 4.3.3.

3.4 End-Limit Signals PEL and MEL

There are two end-limit signals PEL and MEL for each axis. PEL indicates the end limit signal is in the plus direction and MEL indicates the end limit signal is in the minus direction. The signal names, pin numbers, and axis numbers are shown in the table below:

P1 Pin No	Signal Name	Axis #	P1 Pin No	Signal Name	Axis #
20	PEL0	①	21	MEL0	①
54	PEL1	②	55	MEL1	②

A circuit diagram is shown in the diagram below. The external limit switch should have a contact capacity of +24V @ 8mA minimum. Either ‘A-type’ (normal open) contact or ‘B-type’ (normal closed) contact switches can be used. To set the active logic of the external limit signal, please refer to the explanation of _8102_set_limit_logic function.

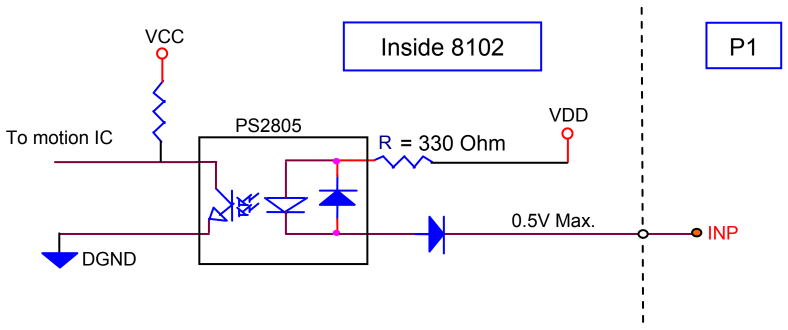


3.5 In-position Signal INP

The in-position signal INP from a servo motor driver indicates its deviation error. If there is no deviation error then the servo's position indicates zero. The signal names, pin numbers, and axis numbers are shown in the table below:

P1 Pin No	Signal Name	Axis #
10	INP0	①
44	INP1	②

The input circuit of the INP signals is shown in the diagram below:



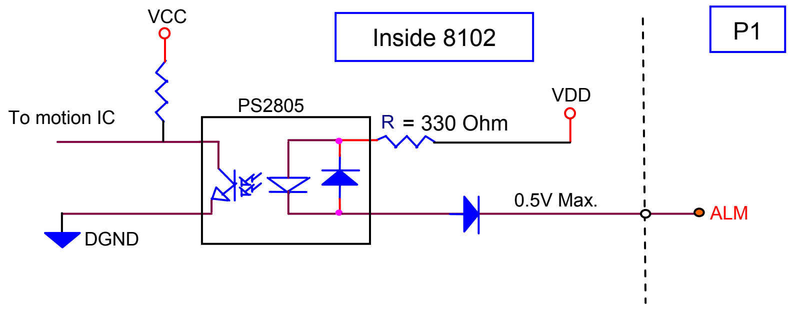
The in-position signal is usually generated by the servomotor driver and is ordinarily an open collector output signal. An external circuit must provide at least 8mA current sink capabilities to drive the INP signal.

3.6 Alarm Signal ALM

The alarm signal ALM is used to indicate the alarm status from the servo driver. The signal names, pin numbers, and axis numbers are shown in the table below:

P1 Pin No	Signal Name	Axis #
9	ALM0	①
43	ALM1	②

The input alarm circuit is shown below. The ALM signal usually is generated by the servomotor driver and is ordinarily an open collector output signal. An external circuit must provide at least 8mA current sink capabilities to drive the ALM signal.



3.7 Deviation Counter Clear Signal ERC

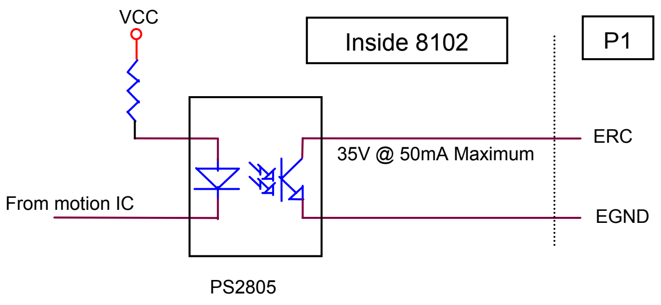
The deviation counter clear signal (ERC) is active in the following 4 situations:

1. Home return is complete
2. End-limit switch is active
3. An alarm signal stops OUT and DIR signals
4. An emergency stop command is issued by software (operator)

The signal names, pin numbers, and axis numbers are shown in the table below:

CN2 Pin No	Signal Name	Axis #
8	ERC0	①
42	ERC1	②

The ERC signal is used to clear the deviation counter of the servomotor driver. The ERC output circuit is an open collector with a maximum of 35V at 50mA driving capacity.

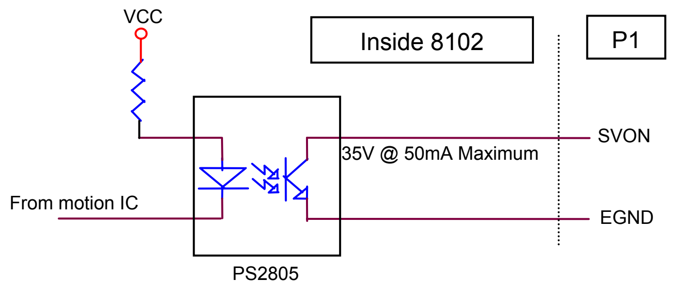


3.8 General-purpose Signal SVON

The SVON signal can be used as a servomotor-on control or general purpose output signal. The signal names, pin numbers, and its axis numbers are shown in the following table:

P1 Pin No	Signal Name	Axis #
7	SVON0	①
41	SVON1	②

The output circuit for the SVON signal is shown below:

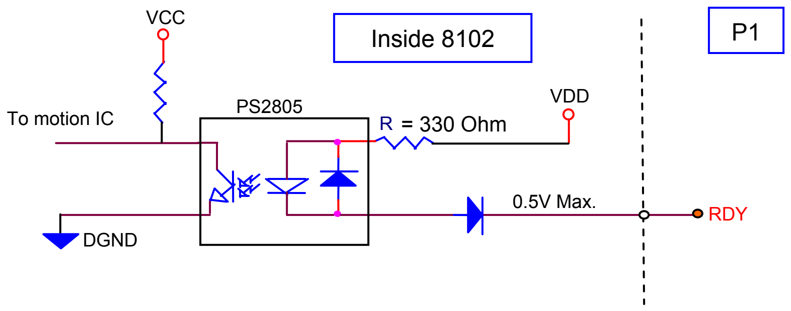


3.9 General-purpose Signal RDY

The RDY signals can be used as motor driver ready input or general purpose input signals. The signal names, pin numbers, and axis numbers are shown in the following table:

P1 Pin No	Signal Name	Axis #
11	RDY0	①
45	RDY1	②

The input circuit of RDY signal is shown in the following diagram:



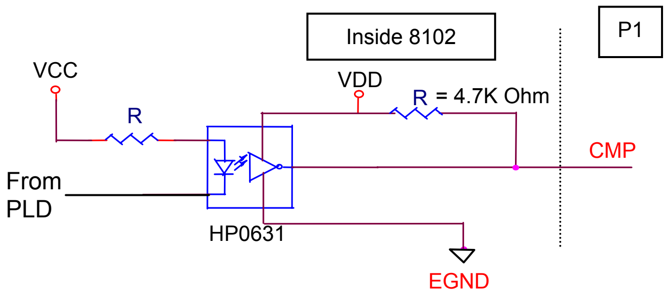
3.10 Position compare output pin: CMP

The PCI-8102 provides 2 comparison output channels, CMP0 and CMP1 used by the first 2 axes, ① & ②. The comparison output channel will generate a pulse signal when the encoder counter reaches a pre-set value set by the user.

The CMP channel is located on P1. The signal names, pin numbers, and axis numbers are shown below:

CN2 Pin No	Signal Name	Axis #
30	CMP0	①
31	CMP1	②

The following wiring diagram is of the CMP on the first 2 axes:



Note: CMP trigger type can be set as normal low (rising edge) or normal high (falling edge). Default setting is normal high. Refer to function **_8102_set_trigger_comparator** for details.

3.11 Multi-Functional input pin: LTC/SD/PCS/CLR

The PCI-8102 provides 2 multi-functional input pins. Each of the 2 pins can be configured either as LTC(Latch) or SD(Slow down) or PCS(Target position override) or CLR(Counter clear). To select the pin function, please refer to 6.12. The default value is LTC and the relavant functions are as follows:

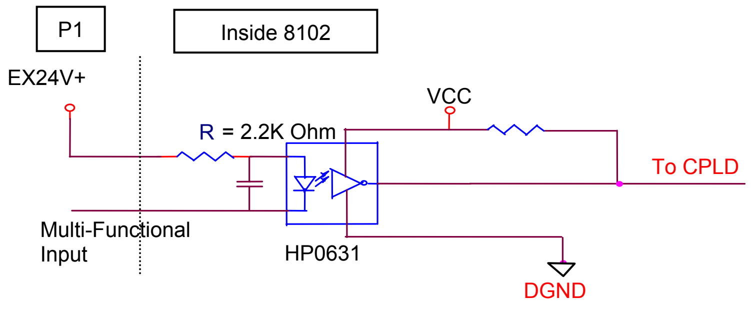
I16 _8102_select_pin23_input(I16 card_id, U16 Select);

I16 _8102_select_pin57_input(I16 card_id, U16 Select);

The multi-functional input pins are on P1. The signal names, pin numbers, and axis numbers are shown in the following table:

P1 Pin No	Signal Name	Axis #
23	LTC/SD/PCS/CLR_0	①
57	LTC/SD/PCS/CLR_1	②

The multi-functional input pin wiring diagram is as followed:



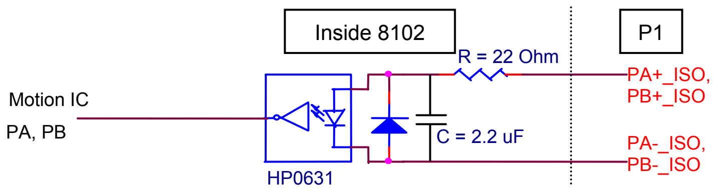
3.12 Pulser Input Signals PA and PB (PCI-8102)

The PCI-8102 can accept differential pulser input signals through the pins of PN1 listed below. The pulsesr behaves like an encoder. The A-B phase signals generate the positioning information, which guides the motor.

P1 Pin No	Signal Name	Axis #	P1 Pin No	Signal Name	Axis #
26	PA+	①②	27	PA-	①②
28	PB+	①②	29	PB-	①②

The pulser signals are used for both Axis 1 and Axis 2. User can decide to enable or disable each axis pulser with _8102_disable_pulser_input function.

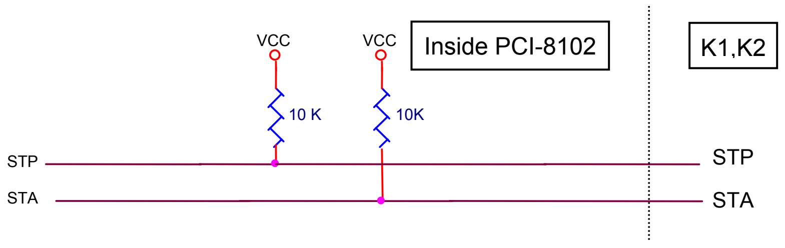
The wiring diagram of the differential pulser input pins are as followed.



3.13 Simultaneously Start/Stop Signals STA and STP

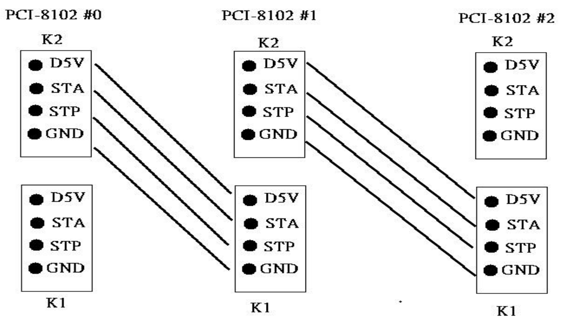
The PCI-8102 provides STA and STP signals, which enable simultaneous start/stop of motions on multiple axes. The STA and STP signals are on CN4.

The diagram below shows the onboard circuit. The STA and STP signals of the four axes are tied together respectively.

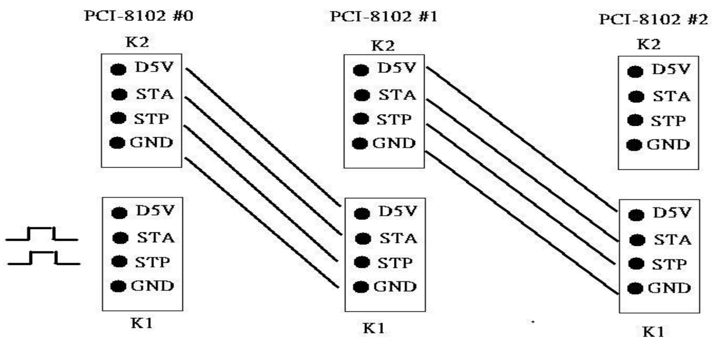


The STP and STA signals are both input and output signals. To operate the start and stop action simultaneously, both software control and external control are needed. With software control, the signals can be generated from any one of the PCI-8102. Users can also use an external open collector or switch to drive the STA/STP signals for simultaneous start/stop.

If there are two or more PCI-8102 cards, connect the K2 connector on the previous card to K1 connector on the following card. The K1 and K2 connectors on a same PCI-8102 are connected internally.



User can also use external start and stop signals to issue a cross-card simultaneous motor operation. Just connect external start and stop signals to STA and STP pins on the K1 connector of the first PCI-8102 card.

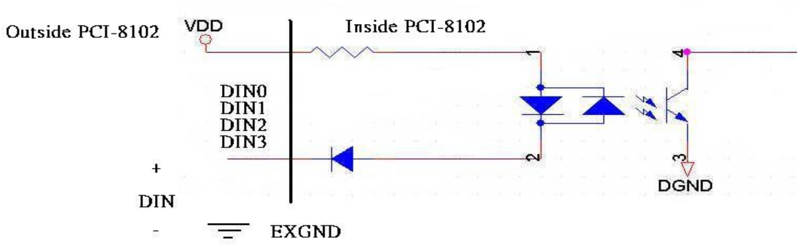


3.14 General-Purposed Digital Input / Output

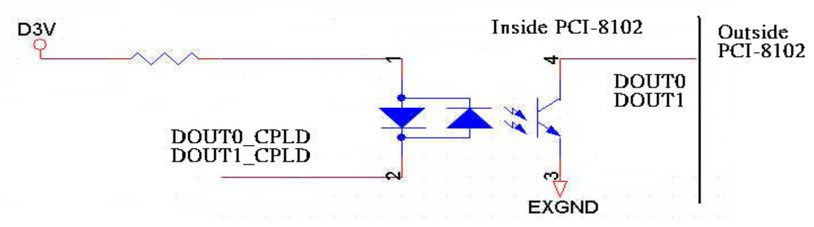
The PCI-8102 provides 2 general purpose open collector outputs and 4 inputs. The signal names, pin numbers, and axis numbers are shown in the table below:

Pin No.	Name	Function
61	DIN0	Digital IN0
62	DIN1	Digital IN1
63	DIN2	Digital IN2
64	DIN3	Digital IN3
65	DOUT0	Digital Out0
66	DOUT1	Digital Out1

The wiring diagram of the digit input signals are as followed:



The wiring diagram of the digit output signals are as followed:



3.15 Termination Board

Currently, PCI-8102 can only connected to 2 termination boards : DIN-68S and DIN-68M-J3A. DIN-68S is a general-purposed termination board. With maximum flexibility user should put more effort and care to use DIN-68S to avoid wiring errors. The other termination board can be used for PCI-8102 is DIN-68M-J3A. This termination board is used only for Mitsubishi J3A Servo Motor. For Mitsubishi J2A Servo Motor or other servo motors from different vendors, please use DIN-68S termination board.

4

Operation Theory

This chapter describes the detail operation of the motion controller card. Contents of the following sections are as follows:

4.1 Classifications of Motion Controller

At the beginning of servo/stepper driver come to the world, people start to talk about motion control widely instead of motor control. They separate motor control into two layers: one is motor control and the other is motion control. Motor control talks much about on the PWM, power stage, closed loop, hall sensors, vector space, and so on. Motion control talks much about on the speed profile generating, trajectory following, multi-axes synchronization, and coordinating.

4.1.1 Voltage type motion control Interface

The interfaces between motion and motor control are changing rapidly. From the early years, people use voltage singal as a command to motor controller. The amplitude of the signal means how fast a motor rotating and the time duration of the voltage changes means how fast a motor acceleration from one speed to the other speed. Voltage signal as a command to motor driver is so called “analog” type motion controller. It is much eaiser to integrate into an analog circuit of motor controller but sometimes noise is a big problem for this type of motion control. Besides, if people want to do positioning control of a motor, the analog type motion controller must have a feedback signal of position information and use a closed loop control algorithm to make it possible. This increased the complexity of motion control and not easy to use for a beginner.

4.1.2 Pulse type motion control Interface

The second interface of motion and motor control is pulses train type. As a trend of digital world, pulse trains type represent a new concept to motion control. The counts of pulses show how many steps of a motor rotates and the frequency of pulses show how fast a motor runs. The time duration of frequency changes represent the acceleration rate of a motor. Because of this interface, users can control a servo or stepper motor more easier than analog type for positioning applications. It means that motion and motor control can be separated more easily by this way.

Both of these two interfaces need to take care of gains tunning. For analog type position controller, the control loops are built inside and users must tune the gain from the controller. For pulses type position controller, the control loops are built outside on the motor drivers and users must tune the gains on drivers.

For more than one axes’ operation, motion control seems more important than motor control. In industial applications, reliable is a very important factor. Motor driver vendors make good performance products and a motion controller vendors make powerful and variety motion software. Integrated two products make our machine go into perfect.

4.1.3 Network type motion control Interface

Recently, there is a new control interface come into the world. That’s network type motion controller. The command between motor driver and motion controller is not analog or pulses signal any more. It is a network packet which contents position information and motor information. This type of controller is more reliable because of digitized and packetized. Because a motion controller must be real-time, the nerowrk must have real-time capacity around a cycle time below 1 mini-second. This means that not commercial network can do this job. It must have a specific network like Mitsubishi SSCNET. The network may have opto-fiber type to increase communication reliability.

4.1.4 Software real-time motion control kernel

For motion control kernel, there are three ways to accomplish it. They are DSP-based, ASIC based, and software real-time based.
A motion control system needs an absolutely real-time control cycle and the calculation on controller must provide a control data at the same cycle. If not, the motor will not run smoothly. Many machine makers will use PC's computing power to do this. They can use simply a feedback counter card and a voltage output or pulse output card to make it. This method is very low-end and takes much software effort. For sure their realtime performance, they will use a real-time software on the system. It increases the complexity of the system too. But this method is the most flexible way for a professional motion control designers. Most of these methods are on NC machines.

4.1.5 DSP based motion control kernel

A DSP-based motion controller kernel solves real-time software problem on computer. DSP is a micro-processer itself and all motion control calculations can be done on it. There is no real-time software problem because DSP has its own OS to arrange all the procedures. There is no interruption from other inputs or context switching problem like Windows based computer. Although it has such a perfect performance on real-time requirements, its calculation speed is not as fast as PC's CPU at this age. Besides, the software interfacing between DSP based controller's vendors and users are not easy to use. Some controller vendors provide some kind of assembly languages for users to learn and some controller vendors provide only a handshake documents for users to use. Both ways are not easy to use. No doubtly, DSP based controller provide a better way than software kernel for machine makers to build they applications.

4.1.6 ASIC based motion control kernel

An ASIC-base motion control kernel is a fair way between software kernel and DSP kernel. It has no real-time problem because all motion functions are done via ASIC. Users or controller's vendors just need to set some parameters which ASIC requires and the motion control will be done easily. This kind of motion control separates all system integration problems into 4 parts: Motor driver's performance, ASIC outputting profile, vendor's software parameters to ASIC, and users' command to vendors' software. It makes motion controller co-operated more smoothly between devices.

4.1.7 Compare Table of all motion control types

	Software	**ASIC**	**DSP**
Price	Fair	Cheap	Expensive
Functionality	Highest	Low	Normal
Maintenance	Hard	Easy	Fair

	Analog	**Pulses**	**Network**
Price	High	Low	Normal
Signal Quality	Fair	Good	Reliable
Maintenance	Hard	Easy	Easy

4.1.8 PCI-8102's motion controller type

The PCI-8102 is an ASIC based, pulse type motion controller. We make this card into three blocks: motion ASIC, PCI card, software motion library. Users can access motion ASIC via our software motion libray under Windows 2000/XP, Linux, and RTX driver. Our software motion linrary provides one-stop-function for controlling motors. All the speed parameters' calculations are done via our library.

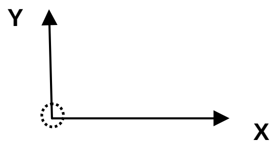
For example, if users want to perform a one-axis point to point moition with a trapezoidal speed profile, they just only fill the target position, speed, and acceleration time in one function. Then the motor will run as the profile. It takes no CPU's resource because every control cycle's pulses generation is done by ASIC. The precision of target position depends on motor drivers' closed loop control performance and mechnical parts, not on motion controller's command because the motion controller is only responsible for sending correct pulses counts via a desired speed profile. So it is much easier for programmers, mechnical or electrical engineers to find out problems.

4.2 Motion Control Modes

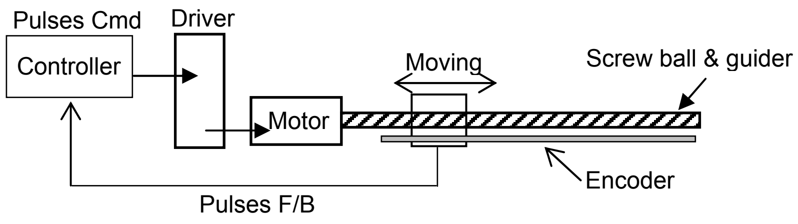
Not like motor control is only for positive or negative moving, motion control make the motors run according to a specific speed profile, path trajectory and synchronous condition with other axes. The following sections describe the motion control modes of this motion controller could be performed.

4.2.1 Coordinate system

We use Cartesian coordinate and pulses for the unit of length. The physical length depends on mechanical parts and motor's resolution. For example, if users install a motor on a screw ball. The pitch of screw ball is 10mm and the pulses needed for a round of motor are 10,000 pulses. We can say that one pulse's physical unit is equal to 10mm/10,000p =1 micro-meter.
Just set a command with 15,000 pulses for motion controller if we want to move 15mm. How about if we want to move 15.0001mm? Don't worry about that, the motion controller will keep the residue value less than 1 pulse and add it to next command.



The motion controller sends *incremental pulses* to motor drivers. It means that we can only send relative command to motor driver. But we can solve this problem by calculating the difference between current position and target position first. Then send the differences to motor driver. For example, if current position is 1000. We want to move a motor to 9000. User can use an absolute command to set a target position of 9000. Inside the motion controller, it will get current position 1000 first then calculate the difference from target position. It gets a result of +8000. So, the motion controller will send 8000 pulses to motor driver to move the position of 9000.
Sometimes, users need to install a linear scale or external encoder to check machine's position. But how do you to build this coordinate system? If the resolution of external encoder is 10,000 pulses per 1mm and the motor will move 1mm if the motion controller send 1,000 pulses, It means that when we want to move 1 mm, we need to send 1,000 pulses to motor driver then we will get the encoder feedback value of 10,000 pulses. If we want to use an absolute command to move a motor to 10,000 pulses position and current position read from encoder is 3500 pulses, how many pulses will it send to motor driver? The answer is (10000 – 3500) / (10,000 / 1,000)=650 pulses. The motion controller will calculate it automatically if users set "move ratio" already. The "move ratio" means the (feedback resolution/command resolution)

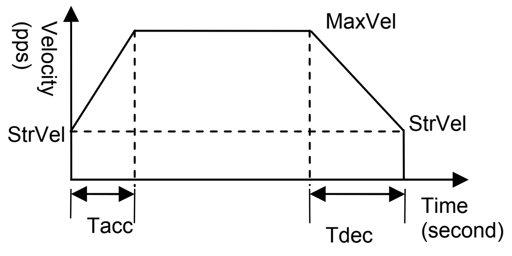


4.2.2 Absolute and relative position move

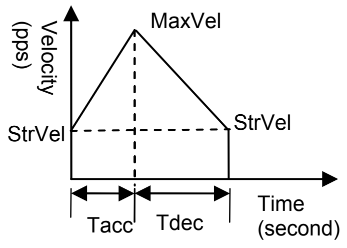
In the coordinate system, we have two kinds command for users to locate the target position. One is absolute and the other is relative. Absolute command means that user give the motion controller a position, then the motion controller will move a motor to that position from current position. Relative command means that user give the motion controller a distance, then the motion controller will move motor by the distance from current position. During the movement, users can specify the speed profile. It means user can define how fast and at what speed to reach the position.

4.2.3 Trapezoidal speed profile

Trapezodial speed profile means the acceleration/deceleration area follows a 1st order linear velocity profile (constant acceleration rate). The profile chart is shown as below:



The area of the velocity profile represents the distance of this motion. Sometimes, the profile looks like a triangle because the desired distance from user is smaller than the area of given speed parameters. When this situation happens, the motion controller will lower the maximum velocity but keep the acceleration rate to meet user's distane requirement. The chart of this situation is shown as below:

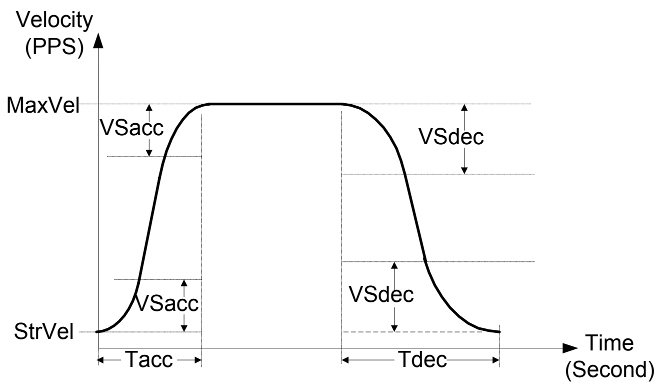


This kind of speed profile could be applied on velocity mode, position mode in one axis or multi-axes linear interpolation and two axes circular interpolation modes.

4.2.4 S-curve and Bell-curve speed profile

S-curve means the speed profile in accelerate/decelerate area follows a 2nd order curve. It can reduce vibration at the beginning of motor start and stop. In order to speed up the acceleration/deceleration during motion, we need to insert a linear part into these areas. We call this shape as "Bell" curve. It adds a linear curve between the upper side of s-curve and lower side of s-curve. This shape improves the speed of acceleration and also reduces the vibration of acceleration.

For a bell curve, we define its shape's parameter as below:



Tacc: Acceleration time in second

Tdec: Deceleration time in second

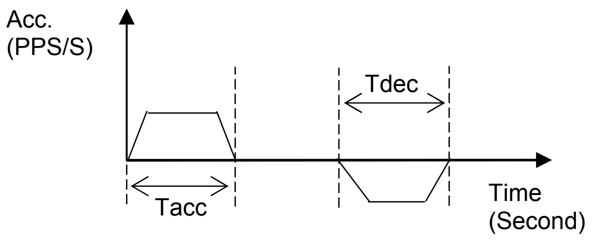
StrVel: Starting velocity in PPS

MaxVel: Maximum velocity in PPS

VSacc: S-curve part of a bell curve in deceleration in PPS

VSdec: S-curve part of a bell curve in deceleration in PPS

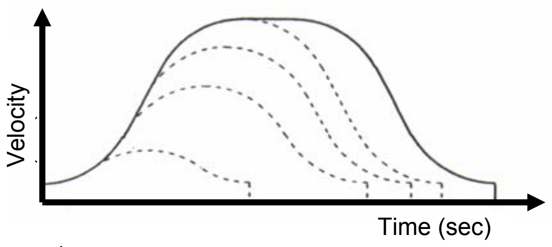
If VSacc or VSdec=0, it means acceleration or deceleration use pure S-curve without linear part. The Acceleration chart of bell curve is shown below:



The S-curve profile motion functions are designed to always produce smooth motion. If the time for acceleration parameters combined with the final position don't allow an axis to reach the maximum velocity (i.e. the moving distance is too small to reach MaxVel), then the maximum velocity is automatically lowered (see the following Figure).

The rule is to lower the value of MaxVel and the Tacc, Tdec, VSacc, VSdec automatically, and keep StrVel, acceleration, and jerk unchanged. This is also applicable to Trapezoidal profile motion.

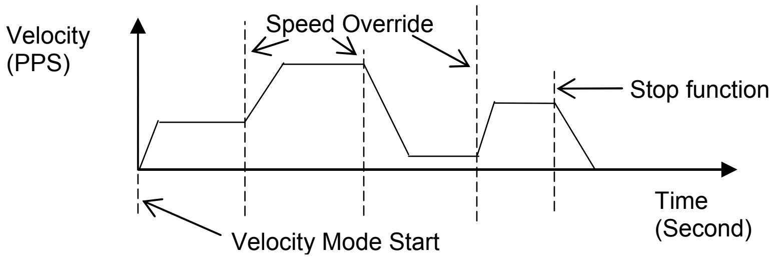
This kind of speed profile could be applied on velocity mode, position mode in one axis or multi-axes linear interpolation and two axes circular



interpolation modes.

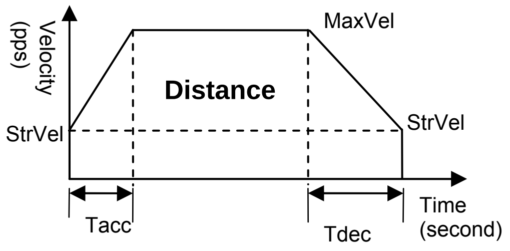
4.2.5 Velocity mode

Veloctiy mode means the pulse command is continuously outputing until a stop command is issued. The motor will run without a target position or desired distance unless it is stopped by other reasons. The output pulse accelerates from a starting velocity to a specified maximum velocity. It can be follow a linear or S-curve acceleration shape. The pulse output rate is kept at maximum velocity until another velocity command is set or a stop command is issued. The velocity could be overrided by a new speed setting. Notice that the new speed could not be a reversed speed of original running speed. The speed profile of this kind of motion is shown as below:



4.2.6 One axis position mode

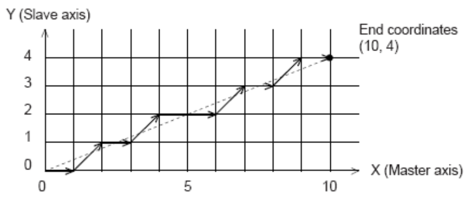
Position mode means the motion controller will output a specific amount of pulses which is equal to users' desired position or distance. The unit of distance or position is pulse internally on the motion controller. The minimum length of distance is one pulse. But in PCI-8102, we provide a floating point function for users to transform a physical length to pulses. Inside our software library, we will keep those distance less than one pulse in register and apply them to the next motion function. Besides positioning via pulse counts, our motion controller provides three types of speed profile to accomplish positioning. There are 1st order trapezoidal, 2nd order S-curve, and mixed bell curve. Users can call respective functions to perform that. The following char shows the relationship between distance and speed profile. We use trapezoidal shape to show it.



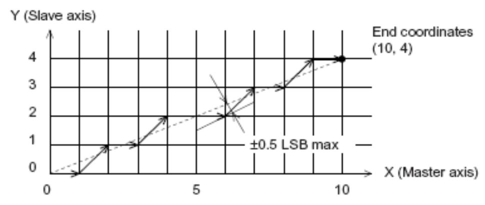
The distance is the area of the V-t diagram of this profile.

4.2.7 Two axes linear interpolation position mode

"Interpolation between multi-axes" means these axes start simultaneously, and reach their ending points at the same time. Linear means the ratio of speed of every axis is a constant value. Assume that we run a motion from (0,0) to (10,4). The linear interpolation results are shown as below.



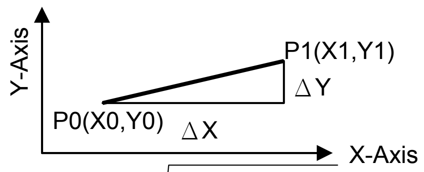
The pulses output from X or Y axis remains 1/2 pulse difference according to a perfect linear line. The precision of linear interpolaition is shown as below:



If users want to stop an interpolation group, just call a stop function on first axis of the group.

As in the diagram below, 2-axis linear interpolation means to move the XY position from P0 to P1. The 2 axes start and stop simultaneously, and the path is a straight line.

The speed ratio along X-axis and Y-axis is (△X: ΔY), respectively, and the vector speed is:

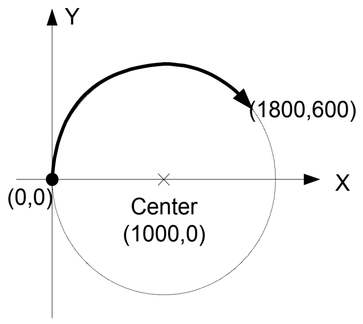


$$\frac{\Delta P}{\Delta t} = \sqrt{(\frac{\Delta X}{\Delta t})^2 + (\frac{\Delta Y}{\Delta t})^2}$$

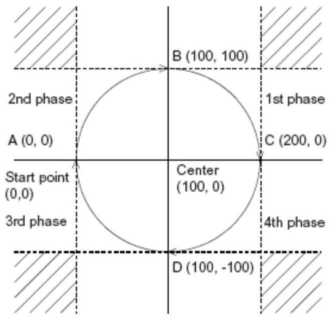
When calling 2-axis linear interpolation functions, the **vector speed** needs to define the start velocity, **StrVel**, and maximum velocity, **MaxVel**.

4.2.8 Two axes circular interpolation mode

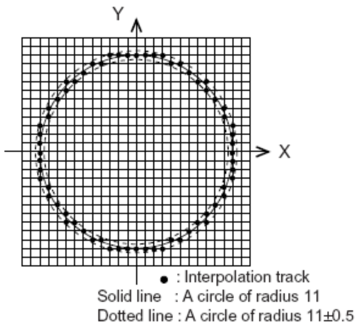
Circular interpolation means XY axes simultaneously start from initial point, (0,0) and stop at end point,(1800,600). The path between them is an arc, and the MaxVel is the tangential speed. Notice that if the end point of arc is not at a proper position, it will move circularly without stopping.



The motion controller will move to the final point user desired even this point is not on the path of arc. But if the final point is not at the location of the shadow area of the following graph, it will run circularly without stopping.



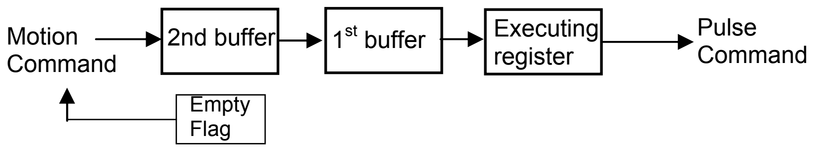
The command precision of circular interpolation is shown below. The precision range is at radius ±1/2 pulse.


Y
X
● : Interpolation track
Solid line : A circle of radius 11
Dotted line : A circle of radius 11±0.5

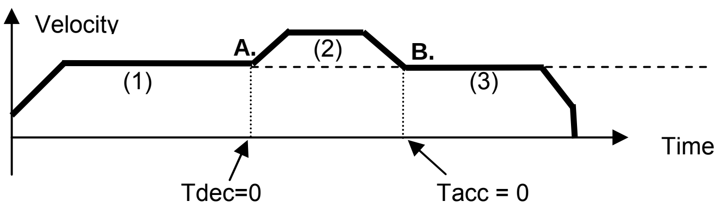
4.2.9 Continuous motion

Continuous motion means a series of motion command or position can be run continuously. Users can set a new command right after previous one without interrupting it. The motion controller can make it possible because there are three command buffers (pre-registers) inside.

When first command is executing, users can set second command into first buffer and third command into second buffer. Once the first command is finished, the motion controller will push the second command to the executing register and the third command to first buffer. Now, the second buffer is empty and user can set the 4th command into 2nd buffer. Normally, if users have enough time to set a new command into 2nd buffer before executing register is finished, the motion can run endlessly. The following diagram shows this architechture of continuous motion.

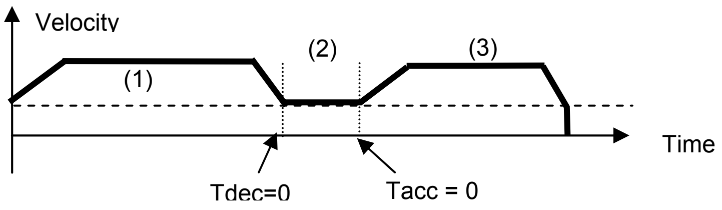


Besides position command, the speed command should be set correctly to perform a speed continuous profile. For the following example, there are three motion command of this continuous motion. The second one has high speed than the others. The interconnection of speed between these three motion functions should be set as the following diagram:



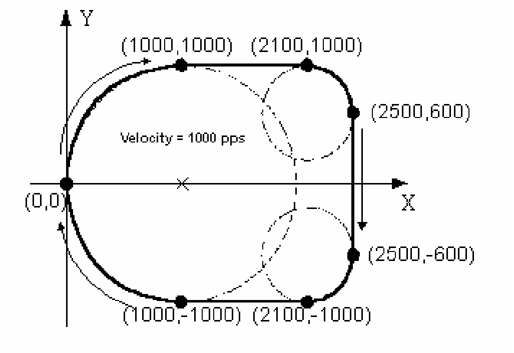
1st command's Tdec=0
2nd command's StrVel = 1st command's MaxVel
3rd command's Tacc=0
3rd command's MaxVel=2nd command's StrVel

If the 2nd command's speed value is lower than the others, the settings would be like as following diagram:



2nd command's Taccc=0
2nd command's Tdec=0
2nd command's MaxVel = 1st command's StrVel
2nd command's MaxVel = 2nd command's StrVel

For 2-axis continuous arc interpolation is the same concept. User can set the speed matched between two command's speed setting.



If the INP checking is enabled, the motion will have some delayed between each command in buffers. INP check enabled make the desired point be reached but reduce the smoothing between each command. If users don't need this delay and meed the smoothing, please turn INP checking off.

4.2.10 Home Return Mode

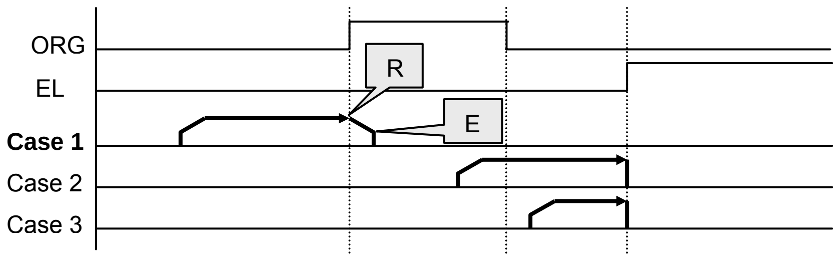
Home return means searching a zero position point on the coordinate. Sometimes, users use a ORG, EZ or EL pin as a zero position on the coordinate. At the beginning of machine power on, the program needs to find a zero point of this machine. Our motion controller provides a home return mode to make it.

We have many home modes and each mode contents many control phases. All of these phases are done by ASIC. No software efforts or CPU loading will be taken. After home return is finished, the target counter will be reset to zero at the desired condition of home mode. For example, a raising edge when ORG input. Sometimes, the motion controller will still output pulses to make machine show down after reseting the counter. When the motor stops, the counter may not be at zero point but the home return procedure is finished. The counter value you see is a reference position from mahcine's zero point already.

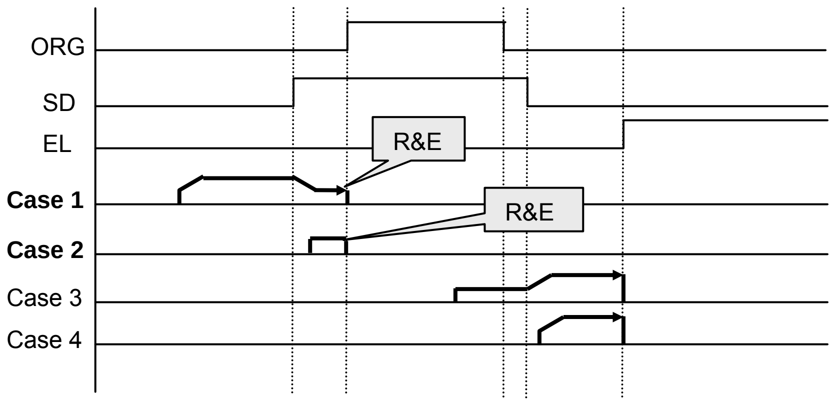
The following figures show the various home modes: R means counter reset (command and position counter). E means ERC signal output.

Home mode=0: (ORG Turn ON then reset counter)

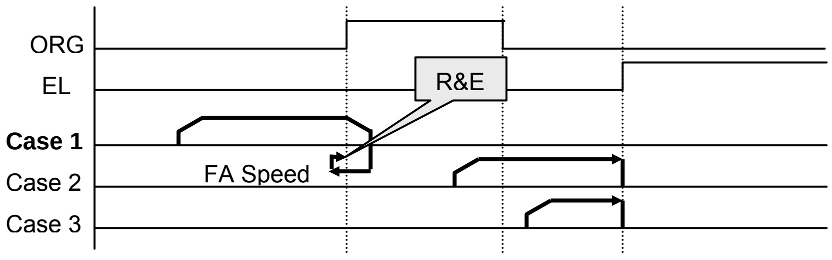
- When SD is not installed



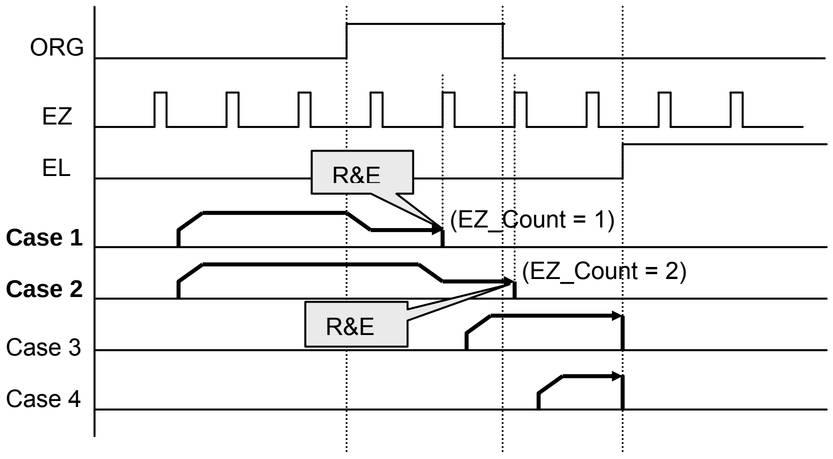
- When SD is installed and SD is not latched



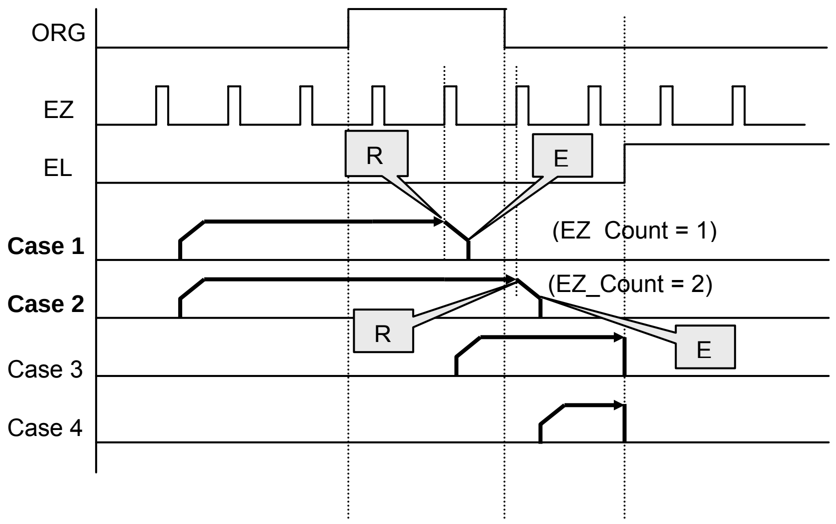
Home mode=1: (Twice ORG turn ON then reset counter)



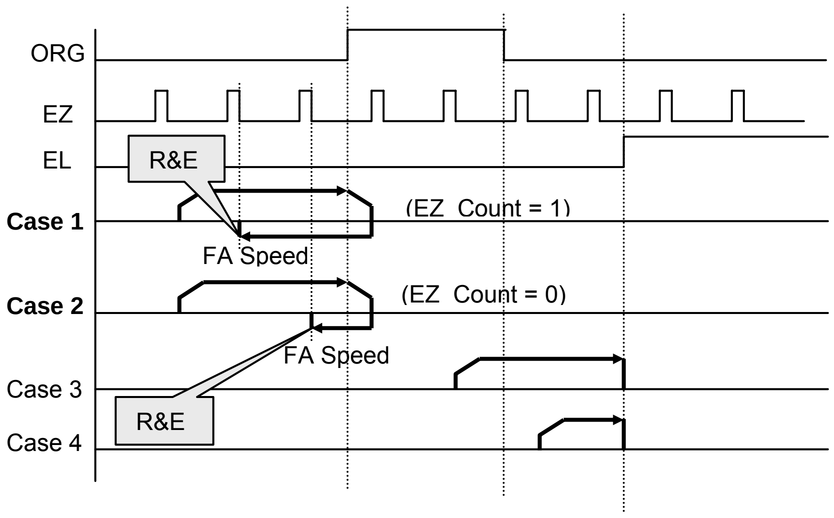
Home mode=2: (ORG ON then Slow down to count EZ numbers and reset counter)



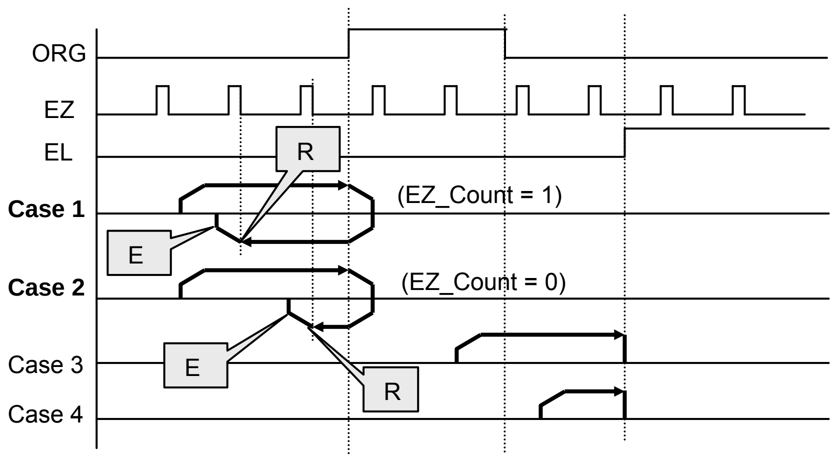
Home mode=3: (ORG ON then count EZ numbers and reset counter)



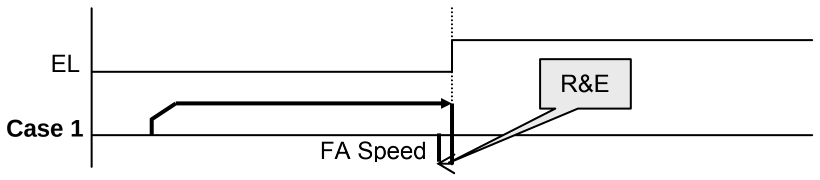
Home mode=4: (ORG On then reverse to count EZ number and reset counter)



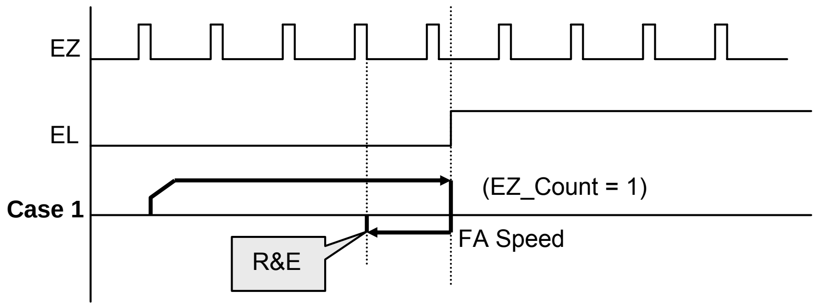
Home mode=5: (ORG On then reverse to count EZ number and reset counter, not using FA Speed)



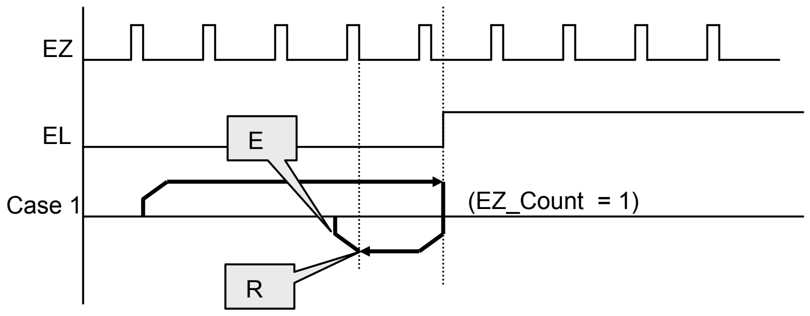
Home mode=6: (EL On then reverse to leave EL and reset counter)



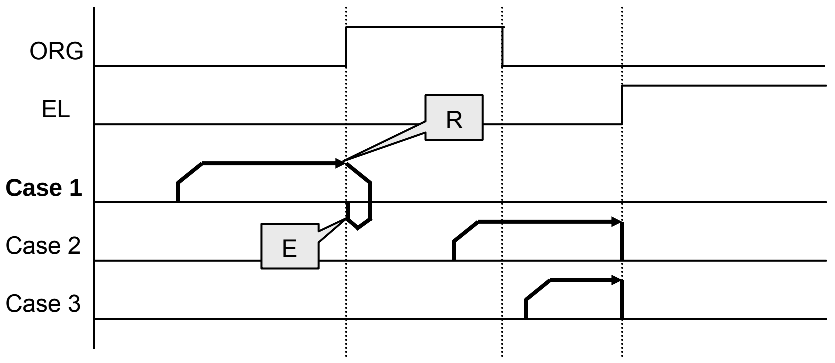
Home mode=7: (EL On then reverse to count EZ number and reset counter)



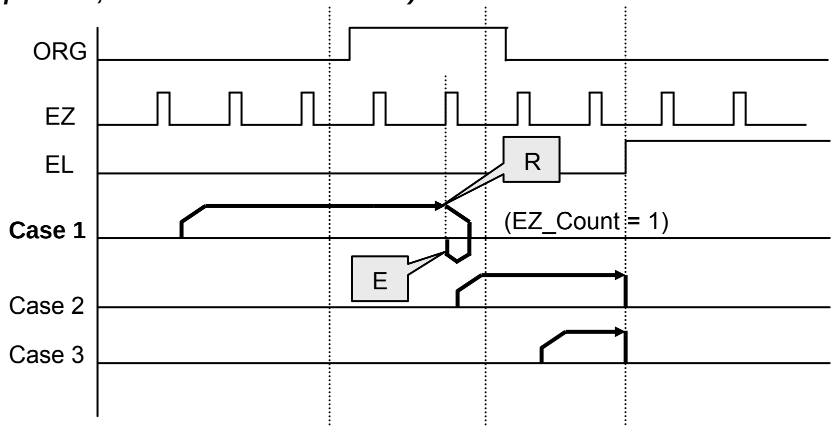
Home mode=8: (EL On then reverse to count EZ number and reset counter, not using FA Speed)



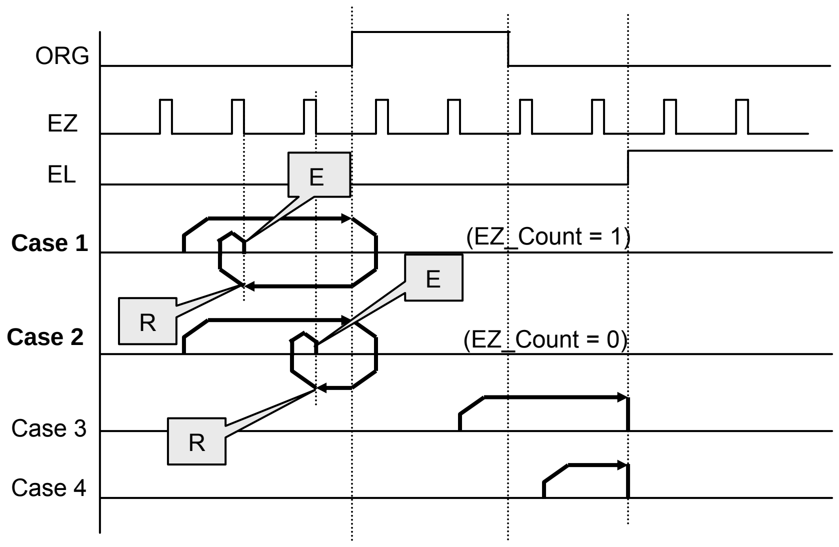
Home mode=9: (ORG On then reverse to zero position, an extension from mode 0)



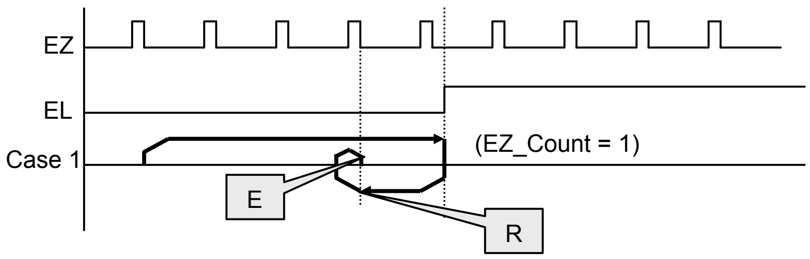
Home mode=10: (ORG On then counter EZ and reverse to zero position, an extension from mode 3)



Home mode=11: (ORG On then reverse to counter EZ and reverse to zero position, an extension from mode 5)

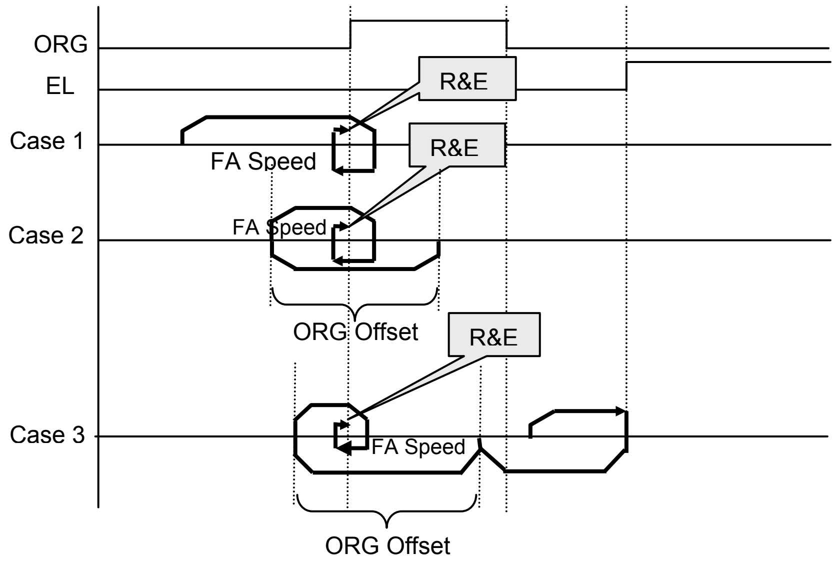


Home mode=12: (EL On then reverse to count EZ number and reverse to zero position, an extension from mode 8)



4.2.11 Home Search Function

This mode is used to add auto searching function on normal home return mode described in previous section no matter which position the axis is. The following diagram is shown the example for home mode 2 via home search function. The ORG offset can't be zero. Suggested value is the double length of ORG area.

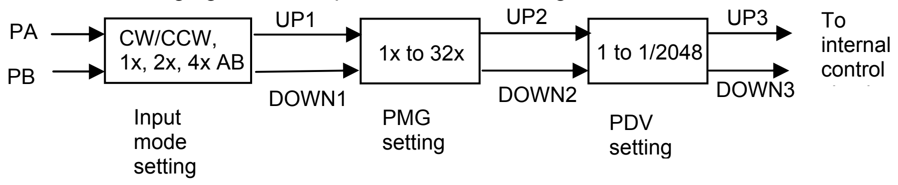


4.2.12 Manual Pulser Function

Manual pulser is a device to generate pulse trains by hand. The pulses are sent to motion controller and re-directed to pulse output pins. The input pulses could be multiplied or divided before sending out.

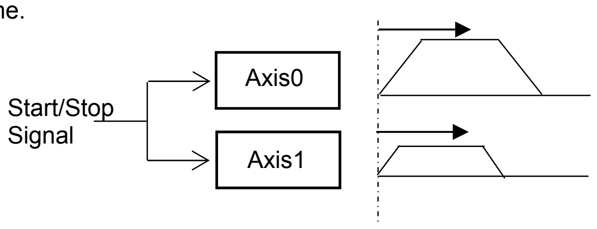
The motion controller receives two kinds of pulse trains from manual pulser device: CW/CCW and AB phase. If the AB phase input mode is selected, the multiplier has additional selection of 1, 2, or 4.

The following figure shows pulser ratio block diagram.



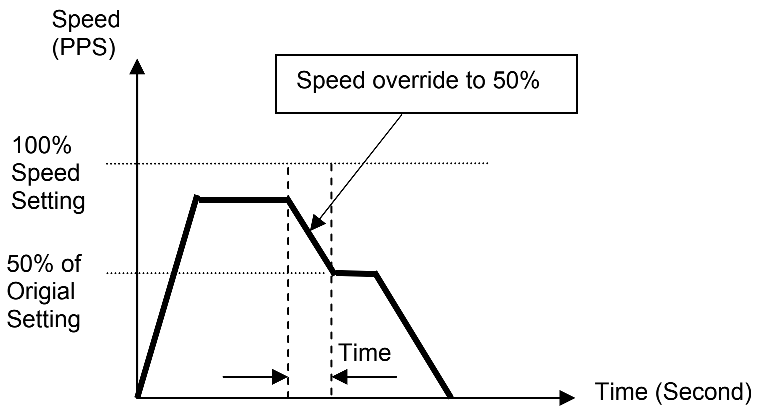
4.2.13 Simultaneous Start Function

Simultaneous motion means more than one axis can be started by a Simultaneous signal which could be external or internal signals. For external signal, users must set move parameters first for all axes then these axes will wait an extern start/stop command to start or stop. For internal signal, the start command could be from a software start function. Once it is issued, all axes which are in waiting synchronous mode will start at the same time.



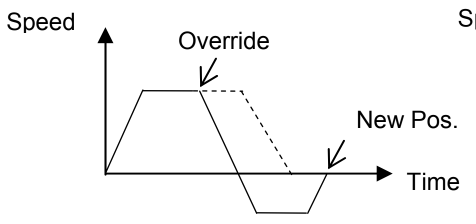
4.2.14 Speed Override Function

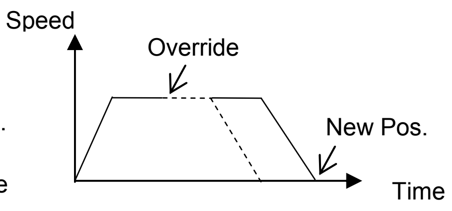
Speed override means that users can change command's speed during the operation of motion. The change parameter is a percentage of original defined speed. Users can define a 100% speed value then change the speed by percentage of original speed when motion is running. If users didn't define the 100% speed value. The default 100% speed is the latest motion command's maximum speed. This function can be applied on any motion function. If the running motion is S-curve or bell curve, the speed override will be a pure s-curve. If the running motion is t-curve, the speed override will be a t-curve.

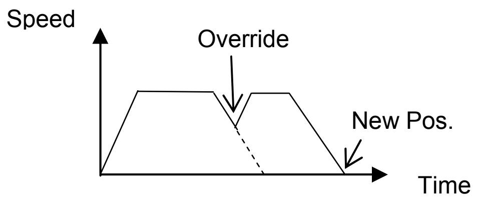


4.2.15 Position Override Function

Position override means that when users issue a positioning command and want to change its target position during this operation. If the new target position is behind current position when override command is issued, the motor will slow down then reverse to new target position. If the new target position is far away from current position on the same direction, the motion will remain its speed and run to new target position. If the override timing is on the deceleration of current motion and the target position is far away from current position on the same direction, it will accelerate to original speed and run to new target position. The operation examples are shown as below. Notice that if the new target position's relative pulses are smaller than original slow down pulses, this function can't work properly.







4.3 The motor driver interface

We provide several dedicated I/Os which can be connected to motor driver directly and have their own functions. Motor drivers have many kinds of I/O pins for external motion controller to use. We classify them to two groups. One is pulse I/O signals including pulse command and encoder interface. The other is digital I/O signals including servo ON, alarm, INP, servo ready, alarm reset and emergency stop inputs. The following sections will describe the functions these I/O pins.

4.3.1 Pulse Command Output Interface

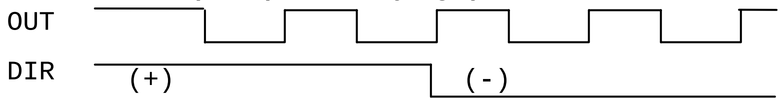
The motion controller uses pulse command to control servo/stepper motors via motor drivers. Please set the drivers to position mode which can accept pulse trains as position command. The pulse command consists of two signal pairs. It is defined as OUT and DIR pins on connector. Each signal has two pins as a pair for differential output. There are two signal modes for pulse output command: (1) single pulse output mode (OUT/DIR), and (2) dual pulse output mode (CW/CCW type pulse output). The mode must be the same as motor driver. The modes vs. signal type of OUT and DIR pins are listed in the table below:

Mode	Output of OUT pin	Output of DIR pin
Dual pulse output (CW/CCW)	Pulse signal in plus (or CW) direction	Pulse signal in minus (or CCW) direction
Single pulse output (OUT/DIR)	Pulse signal	Direction signal (level)

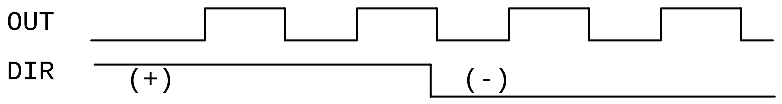
Single Pulse Output Mode (OUT/DIR Mode)

In this mode, the OUT pin is for outputing command pulse chain. The numbers of OUT pulse represent distance in pulse. The frequency of the OUT pulse represents speed in pulse per second. The DIR signal represents command direction of positive (+) or negative (-). The diagrams below show the output waveform. It is possible to set the polarity of the pulse chain.

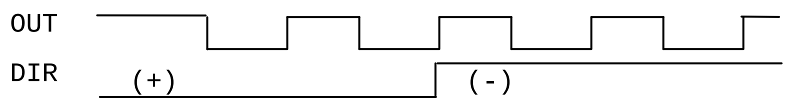
Pulse mode = 0: (OUT pin normally high)



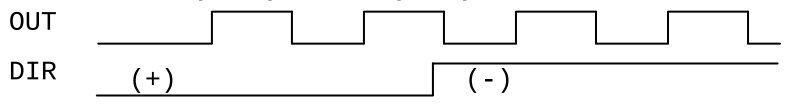
Pulse mode = 1: (OUT pin normally low)



Pulse mode = 2: (OUT pin normally high)



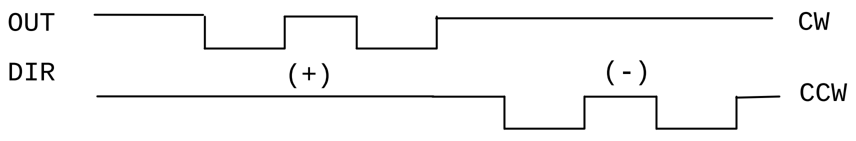
Pulse mode = 3: (OUT pin normally low)



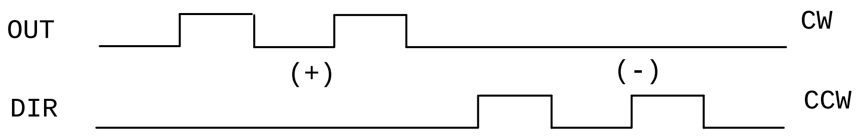
Dual Pulse Output Mode (CW/CCW Mode)

In this mode, the waveform of the OUT and DIR pins represent CW (clockwise) and CCW (counter clockwise) pulse output respectively. The numbers of pulse represent distance in pulse. The frequency of the pulse represents speed in pulse per second. Pulses output from the CW pin makes the motor move in positive direction, whereas pulse output from the CCW pin makes the motor move in negative direction. The following diagram shows the output waveform of positive (+) commands and negative (-) commands.

Pulse outmode = 4: (Pulse is normally high)



Pulse outmode = 5: (Pulse is normally low)



The command pulses are counted by a 28-bit command counter. The command counter can store a value of total pulses outputting from controller.

4.3.2 Pulse feedback input interface

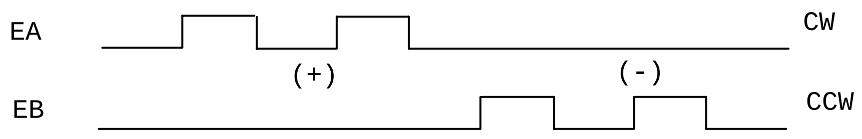
Our motion controller provides one 28-bit up/down counter of each axis for pulse feedback counting. This counter is called position counter. The position counter counts pulses from the EA and EB signal which have plus and minus pins on connector for differential signal inputs. It accepts two kinds of pulse types. One is dual pulses input (CW/CCW mode) and the other is AB phase input. The AB phase input can be multiplied by 1, 2 or 4. Multiply by 4 AB phase mode is the most commonly used in incremental encoder inputs.

For example, if a rotary encoder has 2000 pulses per rotation, then the counter value read from the position counter will be 8000 pulses per rotation when the AB phase is multiplied by four.

If users don't use encoder for motion controller, the feedback source for this counter must be set as pulse command output or the counter value will always be zero. If it is set as puse command output, users can get the position counter value from pulse command output counter because the feedback pulses are internal counted from command output pulses.

The following diagrams show these two types of pulse feedback signal.

Plus and Minus Pulses Input Mode (CW/CCW Mode)



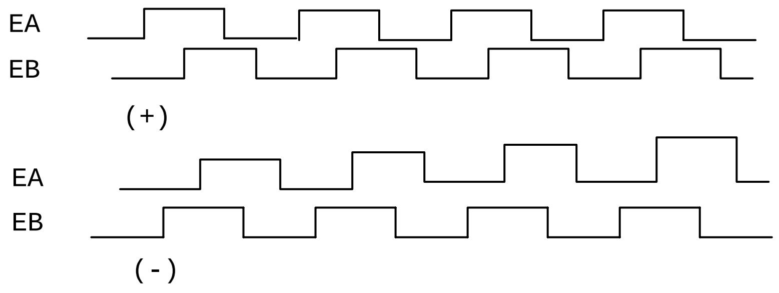
The pattern of pulses in this mode is the same as the ***Dual Pulse Output Mode*** in the Pulse Command Output section except that the input pins are EA and EB.

In this mode, pulses from EA pin cause the counter to count up, whereas EB pin caused the counter to count down.

90° phase difference signals Input Mode (AB phase Mode)

In this mode, the EA signal is a 90° phase leading or lagging in comparison with the EB signal. "Lead" or "lag" of phase difference between two signals is caused by the turning direction of the motor. The up/down counter counts up when the phase of EA signal leads the phase of EB signal.

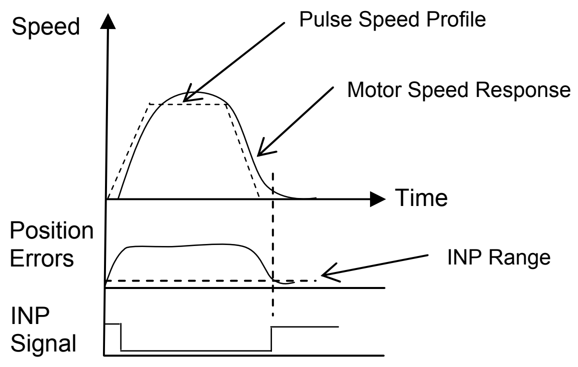
The following diagram shows the waveform.



The index input (EZ) signal is as the zero reference in linear or rotary encoder. The EZ can be used to define the mechanical zero position of the mechanism. The logic of signal must also be set correctly to get corret result.

4.3.3 In position signal

The in-position signal is an output signal from motor driver. It tells motion controllers a motor has been reached a position within a predefined error. The predefined error value is in-position value. Most motor drivers call it as INP value. After motion controller issues a positioning command, the motion busy status will keep true until the INP signal is ON. Users can disable INP check for motion busy flag. If it is disabled, the motion busy wll be FALSE when the pulses command is all sent.



4.3.4 Servo alarm signal

The alarm signal is an output signal from motor driver. It tells motion controller that there has something error inside servo motor or driver. Once the motion controller receives this signal, the pulses command will stop sending and the status of ALM signal will be ON. The reasons of alarm could be servo motor's over speed, over current, over loaded and so on. Please check motor driver's manual about the details.

The logic of alarm signal must be set correctly. If the alarm logic's setting is not the same as motor driver's setting, the ALM status will be always ON and the pulse command can never be outputted.

4.3.5 Error clear signal

The ERC signal is an output from the motion controller. It tells motor driver to clear the the error counter. The error counter is counted from the difference of command pulses and feedback pulses. The feedback position will always have a delay from the command position. It results in pulse differences between these two positions at any moment. The differences are shown in error counter. Motor driver uses the error counter as a basic control index. The large the error counter value is, the faster the motor speed command will be set. If the error counter is zero, it means that zero motor speed command will be set.

At following four situations, the ERC signal will be outputted automatically from motion controller to motor driver in order to clear error counter at the same time.

1. Home return is complete
2. The end-limit switch is touched
3. An alarm signal is active
4. An emergency stop command is issued

4.3.6 Servo ON/OFF switch

The servo on/off switch is a general digital output signal on motion controller. We define it as SVON pin on the connector. It can be used for switching motor driver's controlling state. Once it is turned on, the motor will be holded because the control loop of driver is active. Be careful that when the axis is vertically installed and the servo signal is turned off, the axis will be in uncontrolled state. It could be falled down on the ground. Some situations like servo alarm and emergeny signal ON will result in the same trouble.

4.3.7 Servo Ready Signal

The servo ready signal is a general digital input on motion controller. It has no relative purpose to motion controller. Users can connect this signal to motor driver's RDY signal to check if the motor driver is in ready state. It lets users to check something like the motor driver's power has been inputed or not. Or users can connect this pin as a general input for other purpose. It doesn’t affect motion control.

4.3.8 Servo alarm reset switch

The servo driver will raise an alarm signal if there is something wrong inside the servo driver. Some alarm situations like servo motor over current, over speed, over loading and so on. Power reset can clear the alarm status but users usually don’t want to power off the servo motor when operating. There is one pin from servo driver for users to reset the alarm status.Our motion controller provides one general output pin for each axis. Users can use this pin for reseting servo alarm status.

4.4 Mechanical switch interface

We provide some dedicated input pins for mechanical switches like original switch (ORG), plus and minus end-limit switch (±EL), slow down switch (SD), positioning start switch (PCS), counter latch switch (LTC), emergency stop input (EMG) and counter clear switch (CLR). These switches’ response time is very short, only a few ASIC clock times. There is no real-time problem when using these signals. All functions are done by motion ASIC. The software can just do nothing and only need to wait the results.

4.4.1 Original or home signal

Our controller provides one original or home signal for each axis. This signal is used for defining zero position of this axis. The logic of this signal must be set properly before doing home procedure. Please refer to home mode section for details.

4.4.2 End-Limit switch signal

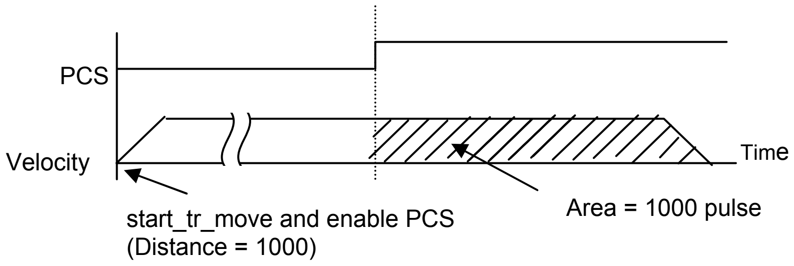
The end-limit switches are usually installed on both ending sides of one axis. We must install plus EL at the positive position of the axis and minus EL at the negative position of the axis. These two signals are for safty reason. If they are installed reversely, the protection will be invalid. Once the motor’s moving part touches one of the end-limit signal, the motion controller will stop sending pulses and output an ERC signal. It can prevent machine crash when miss operation.

4.4.3 Slow down switch

The slow down signals are used to force the command pulse to decelerate to the starting velocity when it is active. This signal is used to protect a mechanical moving part under high speed movement toward the mechanism's limit. The SD signal is effective for both plus and minus directions.

4.4.4 Positioning Start switch

The positioning start switch is used to move a specific position when it is turned on. The function is shown as below.



4.4.5 Counter Clear switch

The counter clear switch is an input signal which makes the counters of motion controller to reset. If users need to reset a counter according to external command, use this pin as controlling source.

4.4.6 Counter Latch switch

The counter latch switch is an input singal which makes counter value to be kept into a register when this input active. If users need to know counter value at the active moment of one input, they can connect this pin to catch that.

4.4.7 Emergency stop input

Our motion controller provides a global digital input for emergeny situation. Once the input is turned on, our motion controller will stop all axes' motion immediately to prevent machine's damage. Ususally, users can connect an emergency stop button to this input on their machine. We suggest this input as normal closed type for safty.

4.5 The Counters

There are four counters for each axis of this motion controller. They are described in this section.

Command position counter: counts the number of output pulses

Feedback position counter: counts the number of input pulses

Position error counter: counts the error between command and feedback pulse numbers.

General purpose counter: The source can be configured as command position, feedback position, manual pulser, or half of ASIC clock.

Target position recorder: A software-maintained target position value of latest motion command.

4.5.1 Command position counter

The command position counter is a 28-bit binary up/down counter. Its input source is the output pulses from the motion controller. It provides the information of the current command position. It is useful for debugging the motion system.

Our motion system is an open loop type. The motor driver receives pulses from motion controller and drive the motor to move. When the driver is not moving, we can check this command counter and see if there is an update value on it. If it is, it means that the pulses have seen sent and the problem could be on the motor driver. Try to check motor driver's pulse receiving counter when this situation is happened.

The unit of command counter is in pulse. The counter value could be reset by a counter clear signal or home function completion. Users can also use a software command counter setting function to reset it.

4.5.2 Feedback position counter

The feedback position counter is a 28-bit binary up/down counter. Its input source is the input pulses from the EA/EB pins. It counts the motor position from motor's encoder output. This counter could be set from a source of command position for an option when no external encoder inputs.

The command output pulses and feedback input pulses will not always be the same ratio in mini-meters. Users must set the ratio if these two pulses are not 1:1.

Because our motion controller is not a closed-loop type, the feedback position counter is just for reference after motion is moving. The position closed-loop is done by servo motor driver. If the servo driver is well tuned and the mechnical parts are well assembled, the total position error will remain in acceptable range after motion command is finished.

4.5.3 Command and Feedback error counter

The command and feedback error counter is used to calculate the error between the command position and the feedback position. The value is calculated from command subtracting feedback position.

If the ratio between command and feedback is not 1:1, the error counter is meaningless.

This counter is a 16-bit binary up/down counter.

4.5.4 General purpose counter

The source of general purpose counter could be any of the following:

1. Command position output – the same as a command position counter
2. Feedback position input – the same as a feedback position counter
3. Manual Pulser input – Default settig
4. Clock Ticks – Counter from a timer about 9.8 MHz

4.5.5 Target position recorder

The target position recorder is used for providing target position information. It is used in continuous motion because motion controller need to know the previous motion command's target position and current motion command's target position in order to calculate relative pulses of current command then send results into pre-register. Please check if the target position is the same with current command position before continuous motion. Especially after the home function and stop function.

4.6 The Comparaters

There are 5 counter comparators of each axis. Each comparater has dedicated functions. They are:

1. Positive soft end-limit comparator to command counter
2. Negative soft end-limit comparator to command counter
3. Command and feedback error counter comparator
4. General comparator for all counters
5. Trigger comparator for all command and feedback counters

4.6.1 Soft end-limit comparators

There are two comparators for end-limit function of each axis. We call them for the soft end-limit comparators. One is for plus or positive end-limit and the other is for minus or negative end-limit. The end-limit is to prevent machine crash when over traveling. We can use the soft limit instead of a real end-limit switch. Notice that these two comparators only compare the command position counter. Once the command position is over the limited set inside the positive or negative comparators, it will stop moving as it touches the end-limit switch.

4.6.2 Command and feedback error counter comparators

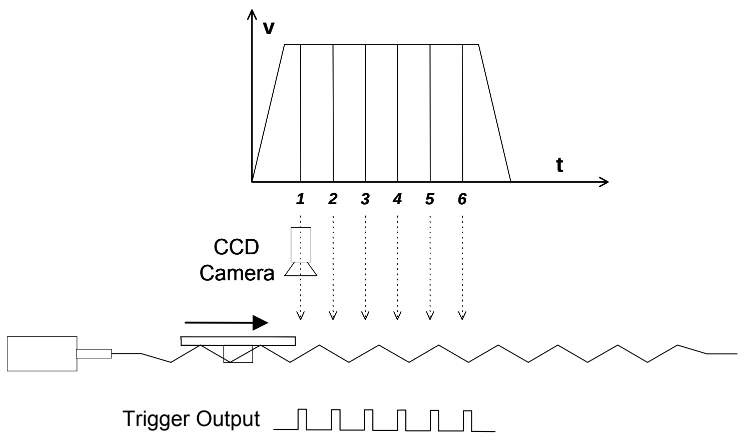
This comparator is only for command and feedback counter error. Users can use this comparator to check if the error is too big. It can be set a action when this condition is met. The actions include generating interrupt, immediately stop, and deceleration to stop.

4.6.3 General comparator

The general comparator let users to choose the source to compare. It could be chosen from command, feedback position counter, error counter or general counter. The compare methods could be chosen by equal, greater than or less than with directional or directionless. Also the action when condition is met can be chosen from generating interrupt, stop motion or others.

4.6.4 Trigger comparator

The trigger comparator is much like general comparator. It has an additional function, generating a trigger pulse when condition is met. Once the condition is met, the CMP pin on the connector will output a pulse for specific purpose like triggering a camera to catch picture. Not all of axes have this function. It depends on the existence of CMP pin of the axis. The following diagram shows the application of triggering.



In this application, the table is controlled by the motion command, and the CCD Camera is controlled by CMP pin. When the comparing position is reached, the pulse will be outputted and the image is captured. This is an on-the-fly image capture. If users want to get more images during the motion path, try to set a new comparing point right after previous image is captured. It can achieve continuous image capturing by this method.

4.7 Other Motion Functions

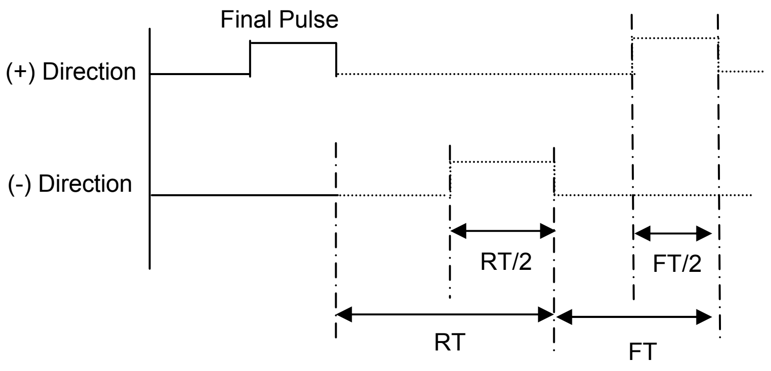
We provide many other functions on the motion controller. Such as backlash compensation, slip correction, vibration restriction, speed profile calculation and so on. The following sections will describe these functions.

4.7.1 Backlash compensation and slip corrections

The motion controller has backlash and slip correction functions. These functions output the number of command pulses in FA speed. The backlash compensation is performed each time when the direction changes on operation. The slip correction function is performed before a motion command, regardless of the direction. The correction amount of pulses can be set by function library.

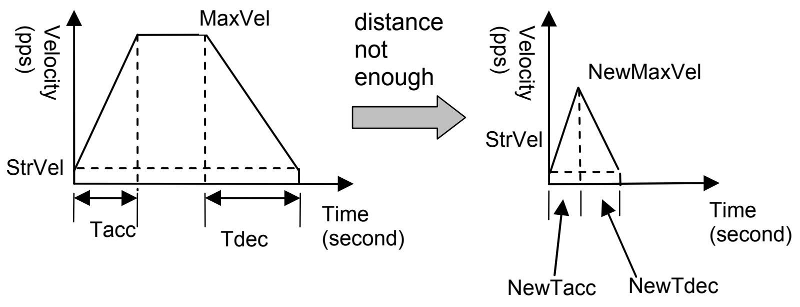
4.7.2 Vibration restriction function

The method of vibration restriction of the motion controller is by adding one pulse of reverse direction and then one pulse of forward direction shortly after completing a motion command. The timing of these two dummy pulses are shown below: (RT is reverse time and FT is forward time)



4.7.3 Speed profile calculation function

Our motion function needs several speed parameters from users. Some parameters are conflict in speed profile. For example, if users input a very fast speed profile and a very short distance to motion function, the speed profile is not exist for these parameters. At this situation, motion library will keep the acceleration and deceleration rate. It tries to lower the maximum speed from users automatically to reform a speed profile feasible. The following diagram shows this concept.



Our motion library has a series of functions to know the actual speed profile of the command from users.

4.8 Interrupt Control

The motion controller can generate an interrupt signal to the host PC. It is much useful for event-driven software application. Users can use this function **_8102_int_control()** to enable ir disable the interrupt service.

There are three kinds of interrupt sources on PCI-8102. One is motion interrupt source and the other is error interrupt source and another is GPIO interrupt sources. Motion and GPIO interrupt sources can be maskable but error interrupt sources can't. Motion interrupt sources can be maskable by **_8102_set_motion_int_factor()**. Its mask bits are shown as following table:

Motion Interrupt Source Bit Settings	
Bit	**Description**
0	Normally Stop
1	Next command in buffer starts
2	Command pre-register 2 is empty and allow new command to write
3	0
4	Acceleration Start
5	Acceleration End
6	Deceleration Start
7	Deceleration End
8	+Soft limit or comparator 1 is ON
9	-Soft limit or comparator 2 is ON
10	Error comparator or comparator 3 is ON
11	General comparator or comparator 4 is ON
12	Trigger comparator or comparator 5 is ON
13	Counter is reset by CLR input
14	Counter is latched by LTC input
15	Counter is latched by ORG Input
16	SD input turns on
17	0
18	0
19	CSTA input or _8102_start_move_all() turns on
20~31	0

The error interrupt sources are non-maskable but the error number of situation could be get from **_8102_wait_error_interrupt()**'s return code if it is not timeout.

Error Interrupt return codes	
Value	Description
0	+Soft Limit is ON and axis is stopped
1	-Soft Limit is ON and axis is stopped
2	Comparator 3 is ON and axis is stopped
3	General Comparator or comparaor 4 is ON and axis is stopped
4	Trigger Comparator or comparator 5 is ON and axis is stopped
5	+End Limit is on and axis is stopped
6	-End Limit is on and axis is stopped

7	ALM is happened and axis is stop
8	CSTP is ON or _8102_stop_move_all is on and axis is stopped
9	CEMG is on and axis is stopped
10	SD input is on and axis is slowed down to stop
11	0
12	Interpolation operation error and stop
13	axis is stopped from other axis's error stop
14	Pulse input buffer overflow and stop
15	Interpolation counter overflow
16	Encoder input signal error but axis is not stopped
17	Pulse input signal error but axis is not stopped
11~31	0

The GPIO interrupt sources are maskable. The mask bits table is shown below:

GPIO Interrupt Source Bit Settings (1=Enable,0=Disable)	
Bit	Description
0	DI0 falling edge
1	DI1 falling edge
2	DI2 falling edge
3	DI3 falling edge
4	DI0 raising edge
5	DI1 raising edge
6	DI2 raising edge
7	DI3 raising edge
8	Pin23 input interrupt
9	Pin57 input interrupt
10	Pin23/57 interrupt mode selection (0=falling, 1=raising)
11~14	0
15	GPIO interrupt switch (Always=1)

The steps for using interrupts:

1. Use _8102_int_control(CARD_ID, Enable=1/Disable=0);
2. Set interrupt sources for Event or GPIO interrupts.
3. _8102_set_motion_int_facor(AXIS0, 0x01); // Axis0 normally stop
4. _8102_set_gpio_int_factor(CARD0, 0x01); // DI0 falling edge
5. _8102_wait_motion_interrupt(AXIS0, 0x01, 1000) // Wait 1000ms for normally stop interrupt
6. _8102_wait_gpio_interrupt(CARD0, 0x01, 1000) // Wait 1000ms for DI0 falling edge interrupt
7. I16 ErrNo=_8102_wait_error_interrupt(AXIS0, 2000); // Wait 2000ms for error interrupts

4.9 Multiple Card Operation

The motion controller allows more than one card in one system. Since the motion controller is plug-and-play compatible, the base address and IRQ setting of the card are automatically assigned by the PCI BIOS at the beginning of system booting. Users don’t need and can’t change the resource settings.

When multiple cards are applied to a system, the number of card must be noted. The card number depends on the card ID switch setting on the the board. The axis number is depends on the card ID. For example, if three motion controller cards are plugged in to PCI slots, and the corresponding card ID is set, then the axis number on each card will be:

Card ID \ Axis No.	X	Y
0	0	1
2	4	5
3	6	7

Notice that if there has the same card ID on multiple cards, the function will not work correctly.

5

MotionCreatorPro

After installing the hardware (Chapters 2 and 3), it is necessary to correctly configure all cards and double check the system before running. This chapter gives guidelines for establishing a control system and manually testing the PCI-8102 cards to verify correct operation. The MotionCreatorPro software provides a simple yet powerful means to setup, configure, test, and debug a motion control system that uses PCI-8102 cards.

Note that MotionCreatorPro is only available for Windows 2000/XP with a screen resolution higher than 1024x768. It does not run under a DOS environment.

5.1 Execute MotionCreatorPro

After installing the software drivers for the 8102 in Windows 2000/XP, the MotionCreatorPro program can be located at <chosen path >\PCI-Motion\MotionCreatorPro. To execute the program, double click on the executable file or use **Start→Program Files→PCI-Motion→ MotionCreatorPro**.

5.2 About MotionCreatorPro

Before Running MotionCreatorPro, the following issues should be kept in mind.

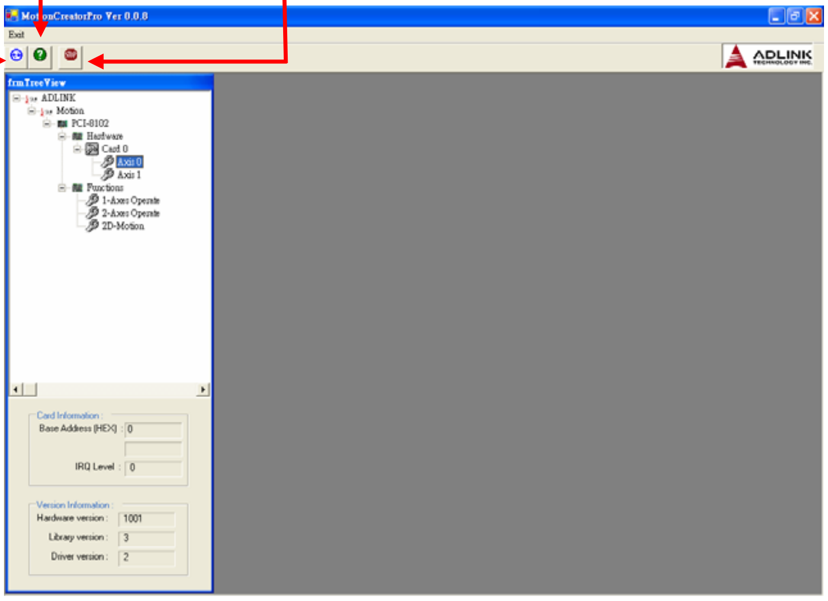
1. MotionCreatorPro is a program written in VB.NET 2003, and is available only for Windows 2000/XP with a screen resolution higher than 1024x768. It cannot be run under DOS.
2. MotionCreatorPro allows users to save settings and configurations for PCI-8102 cards. Saved configurations will be automatically loaded the next time MotionCreatorPro is executed. Two files, ***8102.ini*** and ***8102MC.ini,*** in the ***windows root directory*** are used to save all settings and configurations.
3. To duplicate configurations from one system to another, copy 8102.ini and 8102MC.ini into the windows root directory.
4. If multiple PCI-8102 cards use the same MotionCreatorPro saved configuration files, the DLL function call **_8102_config_from_file()** can be invoked within a user developed program. This function is available in a DOS environment as well.

5.3 MotionCreatorPro Form Introducing

5.3.1 Main Menu

The main menu appears after running MotionCreatorPro. It is used to:

- Reload all Menu
- Open Help menus (refer to section 5.3.8)
- Exit MotionCreatorPro



Card Information :

Base Address (HEX) : 0

IRQ Level : 0

Version Information :

Hardware version : 1001

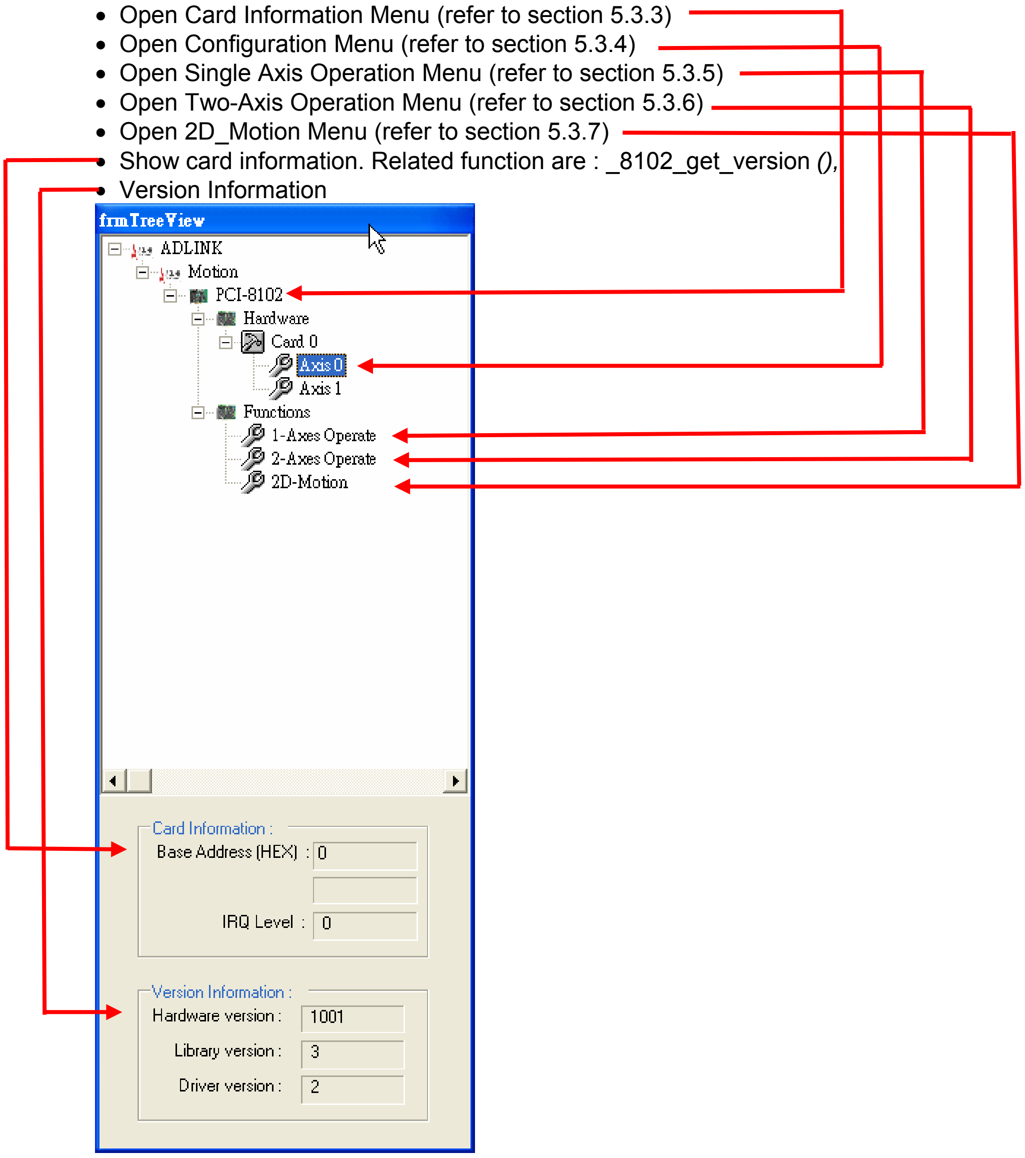
Library version : 3

Driver version : 2

5.3.2 Select Menu

The select menu appears after running MotionCreatorPro. It is used to:

- Select operating card and axis
- Open Card Information Menu (refer to section 5.3.3)
- Open Configuration Menu (refer to section 5.3.4)
- Open Single Axis Operation Menu (refer to section 5.3.5)
- Open Two-Axis Operation Menu (refer to section 5.3.6)
- Open 2D_Motion Menu (refer to section 5.3.7)
- Show card information. Related function are : _8102_get_version *(),*
- Version Information

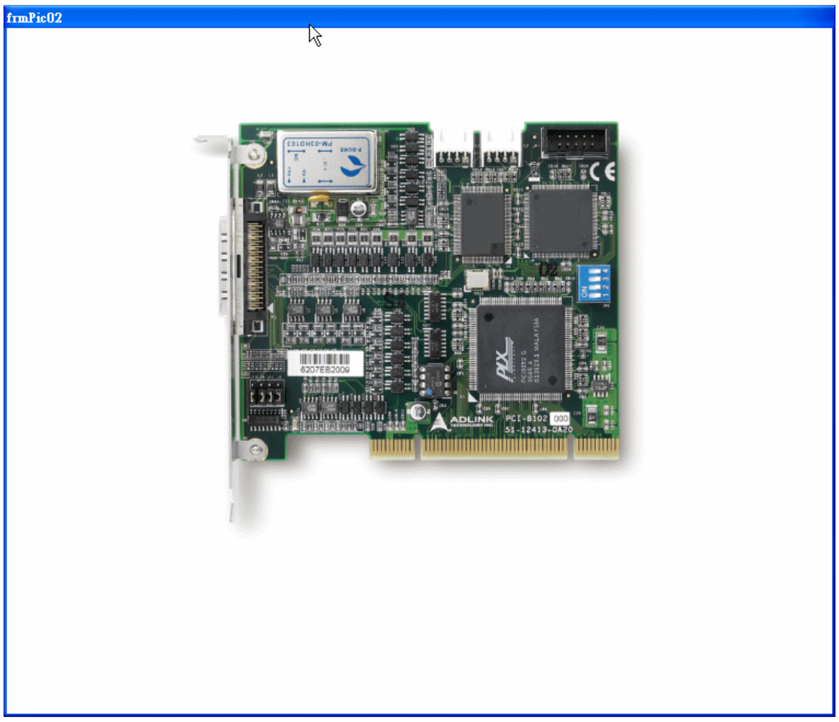


Card Information :
Base Address (HEX) : 0
IRQ Level : 0

Version Information :
Hardware version : 1001
Library version : 3
Driver version : 2

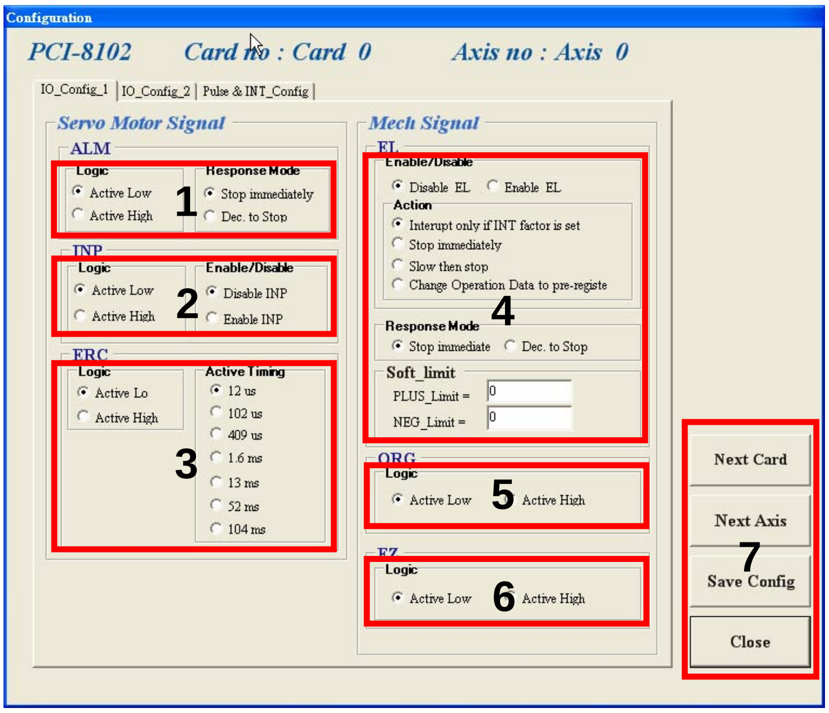
5.3.3 Card Information Menu

In this menu, it show some Information about this Card



5.3.4 Configuration Menu

In this menu, users can configure ALM, INP, ERC, EL, ORG, and EZ.



ALM
Logic: [✓ Active Low] Active High
Response Mode: [✓ Stop immediately] Dec. to Stop

INP
Logic: [✓ Active Low] Active High
Enable/Disable: [✓ Disable INP] Enable INP

ERC
Logic: [✓ Active Lo] Active High
Active Timing: [✓ 12 us] 102 us 409 us 1.6 ms 13 ms 52 ms 104 ms

EL
Enable/Disable: [✓ Disable EL] Enable EL
Action: [✓ Interrupt only if INT factor is set] Stop immediately Slow then stop Change Operation Data to pre-registe
Response Mode: [✓ Stop immediate] Dec. to Stop
Soft_limit
PLUS_Limit = 0
NEG_Limit = 0

ORG
Logic: [✓ Active Low] Active High

EZ
Logic: [✓ Active Low] Active High

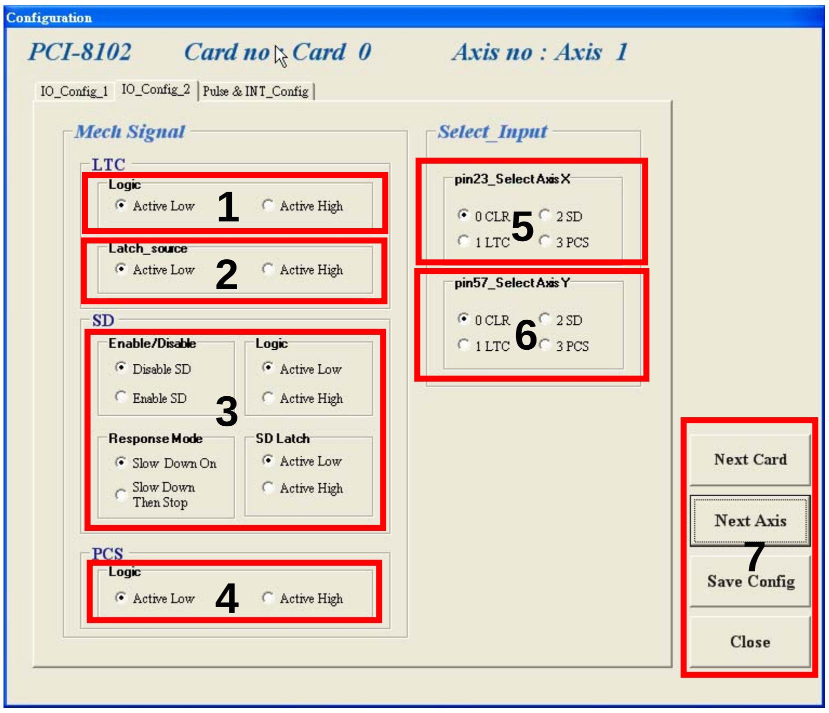
1. **ALM Logic and Response mode:** Select logic and response modes of ALM signal. The related function call is *_8102_set_alm().*
2. **INP Logic and Enable/Disable selection:** Select logic, and Enable/ Disable the INP signal. The related function call is *_8102_set_inp()*
3. **ERC Logic and Active timing:** Select the Logic and Active timing of the ERC signal. The related function call is *_8102_set_erc()*.
4. **EL Response mode:** Select the response mode of the EL signal. The related function call is *_8102_set_limit_logic ()*.
5. **ORG Logic:** Select the logic of the ORG signal. The related function call is *_8102_set_home_config().*

6. **EZ Logic:** Select the logic of the EZ signal. The related function call is *_8102_set_home_config().*

7. **Buttons:**

   - **Next Card:** Change operating card.
   - **Next Axis:** Change operating axis.
   - **Save Config:** Save current configuration to 8102.ini And 8102MC.ini.
   - **Close:** Close the menu.

In this menu, users can configure LTC, SD, PCS, and Select_Input.


Configuration

PCI-8102 Card no Card 0 Axis no : Axis 1

IO_Config_1 | IO_Config_2 | Pulse & INT_Config

Mech Signal

LTC
- Logic: [✓ Active Low] Active High
- Latch_source: [✓ Active Low] Active High

SD
- Enable/Disable: [✓ Disable SD] Enable SD
- Logic: [✓ Active Low] Active High
- Response Mode: [✓ Slow Down On] Slow Down Then Stop
- SD Latch: [✓ Active Low] Active High

PCS
- Logic: [✓ Active Low] Active High

Select_Input
- pin23_Select Axis X: [✓ 0 CLR] 1 LTC 2 SD 3 PCS
- pin57_Select Axis Y: [✓ 0 CLR] 1 LTC 2 SD 3 PCS

1. **LTC Logic:** Select the logic of the LTC signal. The related function call is *_8102_set_ltc_logic().*

2. **LTC latch_source:** Select the logic of the latch_source signal. The related function call is *_8102_set_latch_source ().*

3. **SD Configuration:** Configure the SD signal. The related function call is *_8102_set_sd().*

4. **PCS Logic:** Select the logic of the SelectNo signal. The related function call is *_8102_set_pcs_logic().*

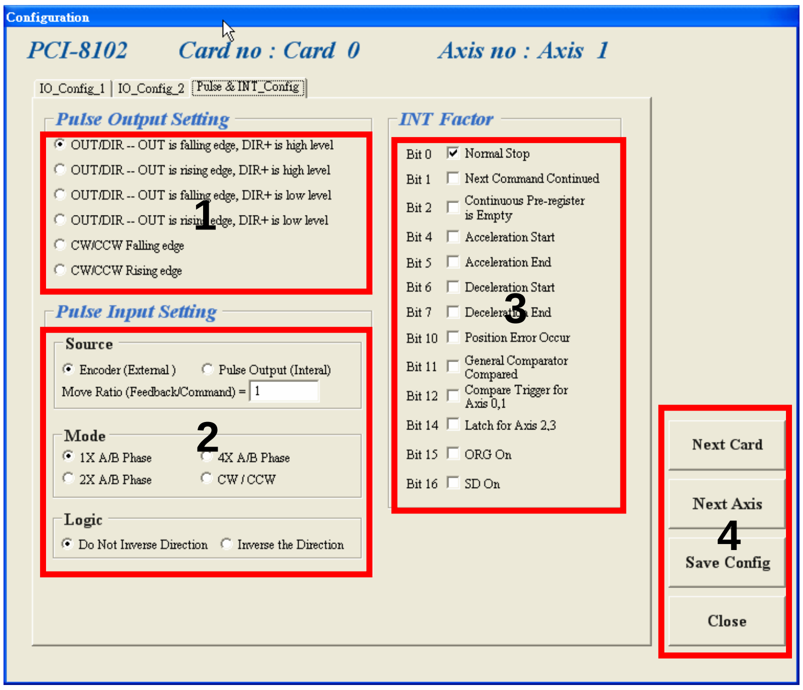
5. **pin23_Select Axis X:** Select the configurations of the Axis X. The related function call is *_8102_select_pin23_input.*

6. **pin57_Select Axis Y:** Select the configurations of the Axis Y. The related function call is *_8102_select_pin57_input.*

7. **Buttons:**

- **Next Card:** Change operating card.
- **Next Axis:** Change operating axis.
- **Save Config:** Save current configuration to 8102.ini And 8102MC.ini.
- **Close:** Close the menu.

In this menu, users can configure pulse input/output and move ratio and INT factor.


Configuration

PCI-8102 Card no : Card 0 Axis no : Axis 1

IO_Config_1 | IO_Config_2 | Pulse & INT_Config

Pulse Output Setting

- [✓ 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 Setting

Source: [✓ Encoder (External)] Pulse Output (Interal)

Move Ratio (Feedback/Command) = 1

Mode: [✓ 1X A/B Phase] 4X A/B Phase 2X A/B Phase CW / CCW

Logic: [✓ Do Not Inverse Direction] Inverse the Direction

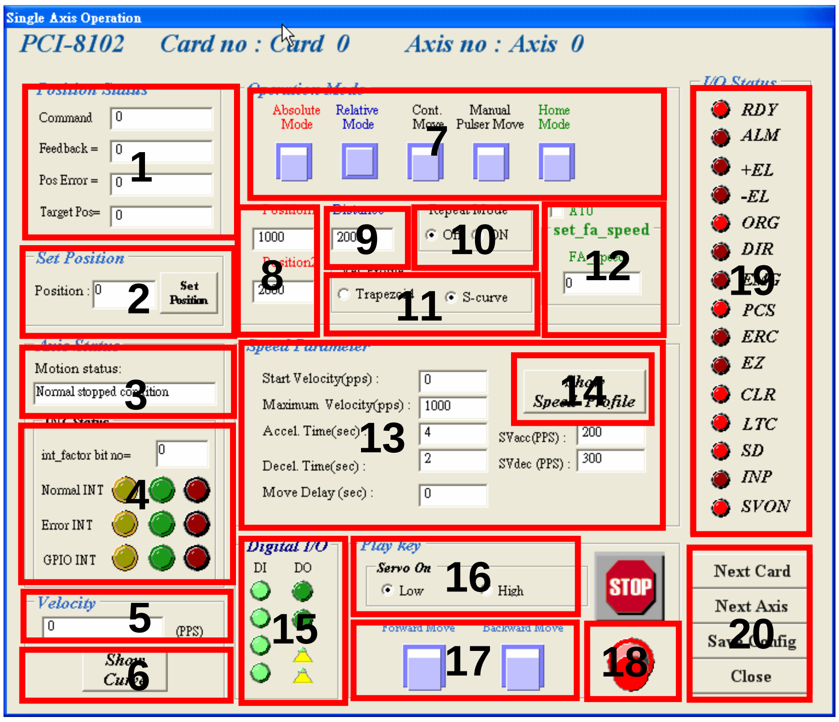
INT Factor

Bit 0	✓	Normal Stop
Bit 1		Next Command Continued
Bit 2		Continuous Pre-register is Empty
Bit 4		Acceleration Start
Bit 5		Acceleration End
Bit 6		Deceleration Start
Bit 7		Deceleration End
Bit 10		Position Error Occur
Bit 11		General Comparator Compared
Bit 12		Compare Trigger for Axis 0,1
Bit 14		Latch for Axis 2,3
Bit 15		ORG On
Bit 16		SD On

1. **Pulse Output Mode:** Select the output mode of the pulse signal (OUT/DIR). The related function call is *_8102_set_pls_outmode()*.
2. **Pulse Input:** Sets the configurations of the Pulse input signal(EA/EB). The related function calls are *_8102_set_pls_iptmode(), _8102_set_feedback_src().*
3. **INT Factor:** Select factors to initiate the event int. The related function call is *_8102_set_int_factor()*.
4. **Buttons:**
   - **Next Card:** Change operating card.
   - **Next Axis:** Change operating axis.
   - **Save Config:** Save current configuration to 8102.ini And 8102MC.ini.
   - **Close:** Close the menu.

5.3.5 Single Axis Operation Menu

In this menu, users can change the settings a selected axis, including velocity mode motion, preset relative/absolute motion, manual pulse move, and home return.


Single Axis Operation

PCI-8102 Card no : Card 0 Axis no : Axis 0

Position Status
Command 0
Feedback = 0
Pos Error = 0
Target Pos= 0

Set Position
Position : 0

Axis Status
Motion status: Normal stopped condition

int_factor bit no= 0

Velocity
0 (PPS)

Operation Mode: Absolute Mode, Relative Mode, Cont. Move, Manual Pulser Move, Home Mode

Position 1000
Position2 2000
Distance 2000
Repeat Mode: [✓ Off] ON
Trapezoidal [✓ S-curve]
set_fa_speed
FA_speed 0

Speed Parameter
Start Velocity(pps) : 0
Maximum Velocity(pps) : 1000
Accel. Time(sec) : 4
Decel. Time(sec) : 2
Move Delay (sec) : 0
SVacc(PPS) : 200
SVdec (PPS) : 300

Play key
Servo On: [✓ Low] High

1. **Position**:

- Command: displays the value of the command counter. The related function is *_8102_get_command().*
- Feedback: displays the value of the feedback position counter. The related function is *_8102_get_position()*
- Pos Error: displays the value of the position error counter. The related function is *_8102_get_error_counter().*
- Target Pos: displays the value of the target position recorder. The related function is *_8102_get_target_pos().*

2. **Position Reset:** clicking this button will set all positioning counters to a specified value. The related functions are:

   _8102_set_position()

   _8102_set_command()

   _8102_reset_error_counter()

   _8102_reset_target_pos()

3. **Motion Status:** Displays the returned value of the _8102_motion_done function. The related function is *_8102_motion_done().*

4. **INT Status:**

   int_factor bit no: Set int_factor bit.

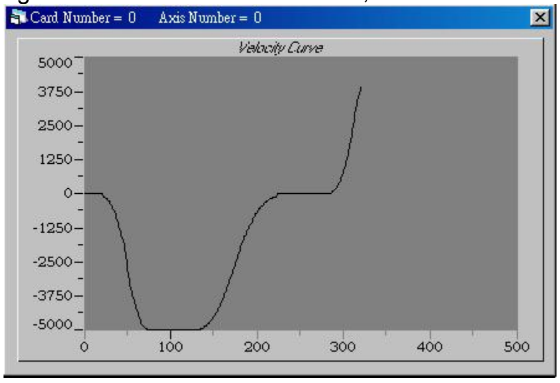
   Normal INT: display of Normal INT status. The related function is *_8102_wait_motion_interrupt ().*

   Error INT: display of Error INT status. The related function is *_8102_wait_error_interrupt ().*

   GPIO INT: display of GPIO INT status. The related function is *_8102_wait_gpio_interrupt ().*

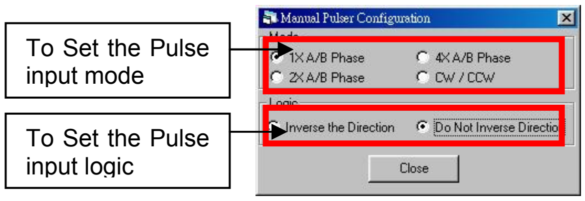
5. **Velocity:** The absolute value of velocity in units of PPS. The related function is *_8102_get_current_speed().*

6. **Show Velocity Curve Button:** Clicking this button will open a window showing a velocity vs. time curve. In this curve, every 100ms, a new velocity data point will be added. To close it, click the same button again. To clear data, click on the curve.



7. **Operation Mode:** Select operation mode.

- **Absolute Mode:** "Position1" and "position2" will be used as absolution target positions for motion. The related functions are *_8102_start_ta_move()*, *_8102_start_sa_move()*.
- **Relative Mode:** "Distance" will be used as relative displacement for motion. The related function is *_8102_start_tr_move(), _8102_start_sr_move()*.
- **Cont. Move:** Velocity motion mode. The related function is *_8102_tv_move(), _8102_start_sv_move()*.
- **Manual Pulser Move:** Manual Pulse motion. Clicking this button will invoke the manual pulse configuration window.


To Set the Pulse input mode
To Set the Pulse input logic
Manual Pulser Configuration
Close

Mode: [✓ 1X A/B Phase] 4X A/B Phase 2X A/B Phase CW / CCW

Logic: Inverse the Direction [✓ Do Not Inverse Directio]

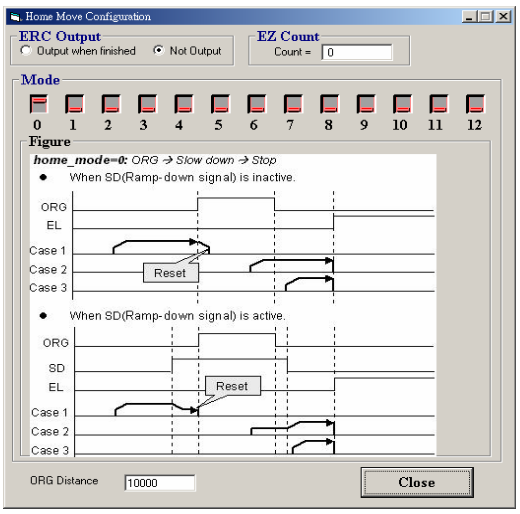
- **Home Mode:** Home return motion. Clicking this button will invoke the home move configuration window. The related function is *_8102_set_home_config().If the check box "ATU" is checked, it will execute auto homing when motion starts.*

ERC Output: Select if the ERC signal will be sent when home move completes.

EZ Count: Set the EZ count number, which is effective on certain home return modes.

Mode: Select the home return mode. There are 13 modes available.

Home Mode figure: The figure shown explains the actions of the individual home modes.



ERC Output: Output when finished [✓ Not Output]

EZ Count: Count = 0

home_mode=0: ORG → Slow down → Stop

- When SD(Ramp-down signal) is inactive.
- When SD(Ramp-down signal) is active.

ORG Distance 10000

Close: Click this button close this window.

8. **Position:** Set the absolute position for “Absolute Mode.” It is only effective when “Absolute Mode” is selected.

9. **Distance:** Set the relative distance for “Relative Mode.” It is only effective when “Relative Mode” is selected.

10. **Repeat Mode:** When “On” is selected, the motion will become repeat mode (**forward←→backward** or **position1←→position2**). It is only effective when “Relative Mode” or “Absolute Mode” is selected.

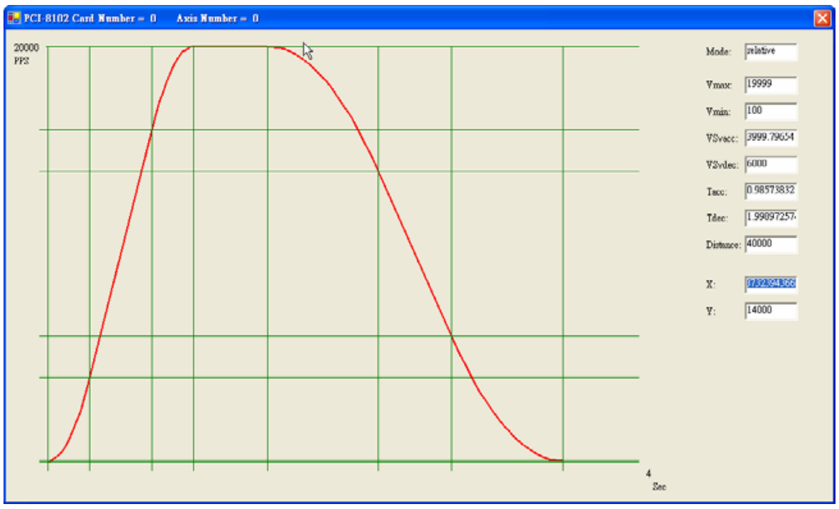
11. **Vel. Profile:** Select the velocity profile. Both Trapezoidal and S-Curve are available for “Absolute Mode,” “Relative Mode,” and “Cont. Move.”

12. **FA Speed/ATU:** Sets the configurations of the FA Speed. The related function calls are *_8102_set_fa_speed().*If the check box “ATU” is checked, it will execute auto homing when motion starts*.*

13. **Motion Parameters:** Set the parameters for single axis motion. This parameter is meaningless if “Manual Pulser Move” is selected, since the velocity and moving distance is decided by pulse input.

    - **Start Velocity:** Set the start velocity of motion in units of PPS. In “Absolute Mode” or “Relative Mode,” only the value is effective. For example, -100.0 is the same as 100.0. In “Cont. Move,” both the value and sign are effective. –100.0 means 100.0 in the minus direction.
    - **Maximum Velocity:** Set the maximum velocity of motion in units of PPS. In “Absolute Mode” or “Relative Mode,” only the value is effective. For example, -5000.0 is the same as 5000.0. In “Cont. Move,” both the value and sing is effective. –5000.0 means 5000.0 in the minus direction.
    - **Accel. Time:** Set the acceleration time in units of second.
    - **Decel. Time**: Set the deceleration time in units of second.
    - **SVacc**: Set the S-curve range during acceleration in units of PPS.
    - **SVdec**: Set the S-curve range during deceleration in units of PPS.
    - **Move Delay**: This setting is effective only when repeat mode is set “On.” It will cause the 8102 to delay for a specified time before it continues to the next motion.

14. **Speed_Profile:** Clicking this button will show the Speed Profile.



PCI-8102 Card Number = 0 Axis Number = 0

Mode: relative
Vmax: 19999
Vmin: 100
VSvacc: 3999.79654
VSvdec: 6000
Tacc: 0.98573832
Tdec: 1.99097257
Distance: 40000
X: 0732394366
Y: 14000

15. **Digital I/O:** Display and set Digital I/O. The related function is *_8102_get_gpio_output(),_8102_get_gpio_input(), _8102_set_gpio_output().*

16. **Servo On:** Set the SVON signal output status. The related function is *_8102_set_servo()*.

17. **Play Key:**

    Left play button: Clicking this button will cause the 8102 start to outlet pulses according to previous setting.

    - In "*Absolute Mode,*" it causes the axis to move to position1.
    - In "*Relative Mode,*" it causes the axis to move forward.
    - In "*Cont. Move,*" it causes the axis to start to move according to the velocity setting.
    - In "*Manual Pulser Move,*" it causes the axis to go into pulse move. The speed limit is the value set by "Maximum Velocity."

    Right play button: Clicking this button will cause the 8102 start to outlet pulses according to previous setting.

    - In "*Absolute Mode,*" it causes the axis to move to position.
    - In "*Relative Mode,*" it causes the axis to move backwards.
    - In "*Cont. Move,*" it causes the axis to start to move according to the velocity setting, but in the opposite direction.
    - In "*Manual Pulser Move,*" it causes the axis to go into pulse move. The speed limit is the value set by "Maximum Velocity."

18. **Stop Button:** Clicking this button will cause the 8102 to decelerate and stop. The deceleration time is defined in "Decel. Time." The related function is *_8102_sd_stop().*

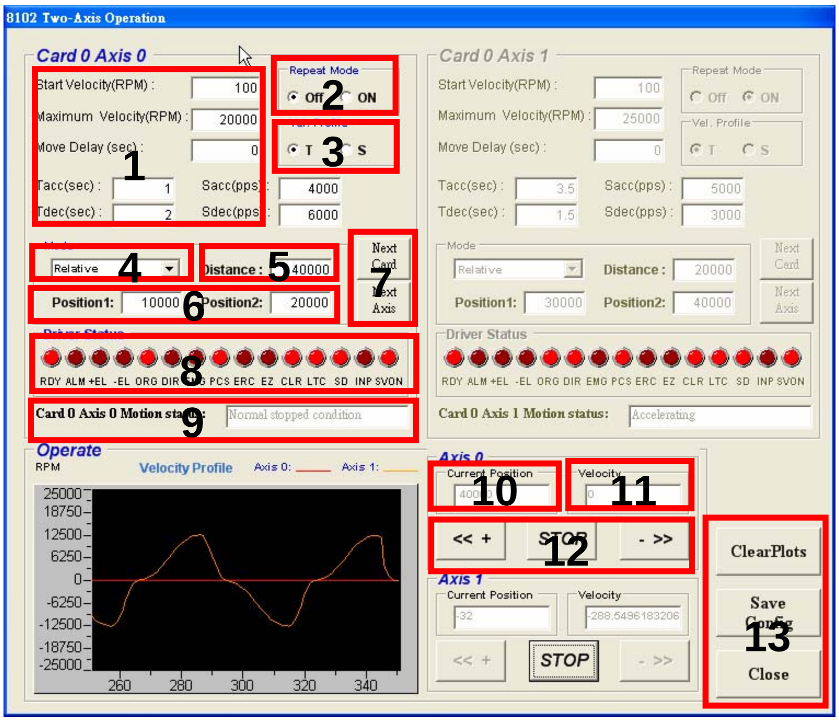
19. **I/O Status:** The status of motion I/O. Light-On means Active, while Light-Off indicates inactive. The related function is *_8102_get_io_status().*

20. **Buttons**:

- **Next Card:** Change operating card.
- **Next Axis:** Change operating axis.
- **Save Config:** Save current configuration to 8102.ini And 8102MC.ini.
- **Close:** Close the menu.

5.3.6 Two-Axis Operation Menu

In this menu, users can change the settings two selected axis, including velocity mode motion, preset relative/absolute motion.


8102 Two-Axis Operation

Card 0 Axis 0
Start Velocity(RPM) : 100
Maximum Velocity(RPM) : 20000
Move Delay (sec) : 0
Tacc(sec) : 1
Tdec(sec) : 2
Sacc(pps) : 4000
Sdec(pps) : 6000
Repeat Mode: [✓ Off] ON
Vel. Profile: [✓ T] S
Mode: Relative
Distance : 40000
Position1: 10000
Position2: 20000
Card 0 Axis 0 Motion status: Normal stopped condition

Card 0 Axis 1
Start Velocity(RPM) : 100
Maximum Velocity(RPM) : 25000
Move Delay (sec) : 0
Tacc(sec) : 3.5
Tdec(sec) : 1.5
Sacc(pps) : 5000
Sdec(pps) : 3000
Repeat Mode: Off [✓ ON]
Vel. Profile: [✓ T] S
Mode: Relative
Distance : 20000
Position1: 30000
Position2: 40000
Card 0 Axis 1 Motion status: Accelerating

Axis 0
Current Position: 400
Velocity: 0

Axis 1
Current Position: -32
Velocity: -288.5496183206

1. **Motion Parameters:** Set the parameters for single axis motion. This parameter is meaningless if “Manual Pulser Move” is selected, since the velocity and moving distance is decided by pulse input.

   - **Start Velocity:** Set the start velocity of motion in units of PPS. In “Absolute Mode” or “Relative Mode,” only the value is effective. For example, -100.0 is the same as 100.0.

   - **Maximum Velocity:** Set the maximum velocity of motion in units of PPS. In “Absolute Mode” or “Relative Mode,” only the value is effective. For example, -5000.0 is the same as 5000.0.

- **Tacc:** Set the acceleration time in units of second.
- **Tdec**: Set the deceleration time in units of second.
- **Sacc**: Set the S-curve range during acceleration in units of PPS.
- **Sdec**: Set the S-curve range during deceleration in units of PPS.

2. **Repeat Mode:** When "On" is selected, the motion will become repeat mode (**forward←→backward** or **position1←→position2**). It is only effective when "Relative Mode" or "Absolute Mode" is selected.
3. **Vel. Profile:** Select the velocity profile. Both Trapezoidal and S-Curve are available for "Absolute Mode," "Relative Mode," and "Cont. Move."
4. **Operation Mode:** Select operation mode.
   - **Absolute Mode:** "Position1" and "position2" will be used as absolution target positions for motion. The related functions are *_8102_start_ta_move()*, *_8102_start_sa_move()*.
   - **Relative Mode:** "Distance" will be used as relative displacement for motion. The related function is *_8102_start_tr_move(), _8102_start_sr_move()*.
5. **Distance:** Set the relative distance for "Relative Mode." It is only effective when "Relative Mode" is selected.
6. **Position:** Set the absolute position for "Absolute Mode." It is only effective when "Absolute Mode" is selected.
7. **Buttons**:
   - **Next Card:** Change operating card.
   - **Next Axis:** Change operating axis.
8. **I/O Status:** The status of motion I/O. Light-On means Active, while Light-Off indicates inactive. The related function is *_8102_get_io_status().*
9. **Motion status:** Displays the returned value of the _8102_motion_done function. The related function is *_8102_motion_done().*
10. **Current Position:**
    - Command: displays the value of the command counter. The related function is *_8102_get_position().*

11. **Velocity:** The absolute value of velocity in units of PPS. The related function is *_8102_get_current_speed().*

12. **Play Key:**

    Left play button: Clicking this button will cause the 8102 start to outlet pulses according to previous setting.

    - In “*Absolute Mode,*” it causes the axis to move to position1.
    - In “*Relative Mode,*” it causes the axis to move forward.

    Right play button: Clicking this button will cause the 8102 start to outlet pulses according to previous setting.

    - In “*Absolute Mode,*” it causes the axis to move to position2.
    - In “*Relative Mode,*” it causes the axis to move backwards.

    Stop Button: Clicking this button will cause the 8102 to decelerate and stop. The deceleration time is defined in “Decel. Time.” The related function is *_8102_sd_stop().*

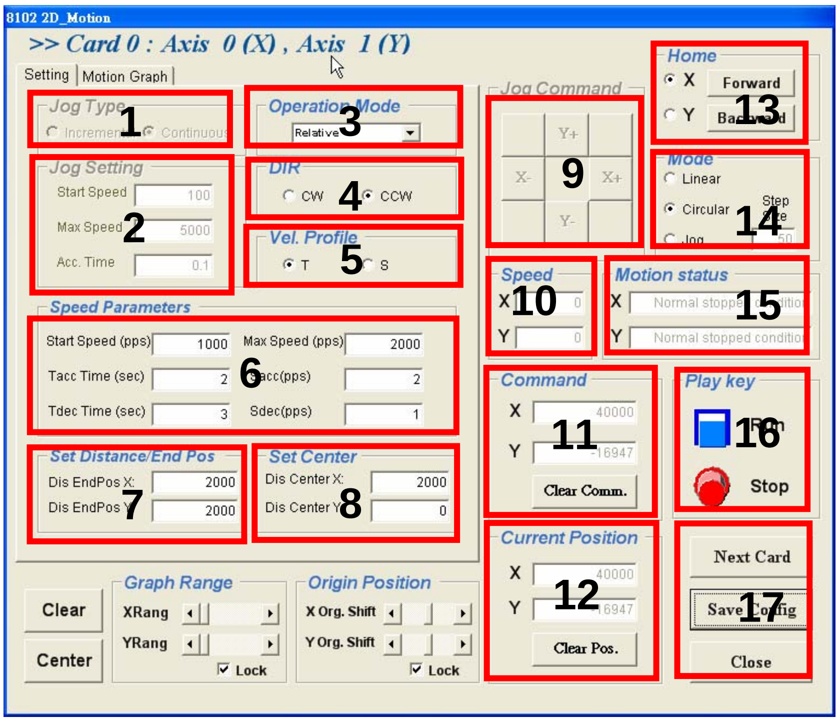
13. **Buttons**:

    - **ClearPlots:** Clear the Motion Graph.
    - **Save Config:** Save current configuration to 8102.ini And 8102MC.ini.
    - **Close:** Close the menu.

5.3.7 2D_Motion Menu

Press 2-D button in operating window will enter this window. This is for 2-D motion test. It includes the following topics:

- •Linear Interpolation
- •Circular Interpolation
- •Incremental Jog
- •Continuous Jog
- •Other Control Objects



8102 2D_Motion

>> Card 0 : Axis 0 (X) , Axis 1 (Y)

Setting | Motion Graph

Jog Type: Incremental [✓ Continuous]

Operation Mode: Relative

Jog Setting: Start Speed 100; Max Speed 5000; Acc. Time 0.1

DIR: CW [✓ CCW]

Vel. Profile: [✓ T] S

Speed Parameters: Start Speed (pps) 1000; Max Speed (pps) 2000; Tacc Time (sec) 2; Sacc(pps) 2; Tdec Time (sec) 3; Sdec(pps) 1

Set Distance/End Pos: Dis EndPos X: 2000; Dis EndPos Y: 2000

Set Center: Dis Center X: 2000; Dis Center Y: 0

Home: [✓ X] Y

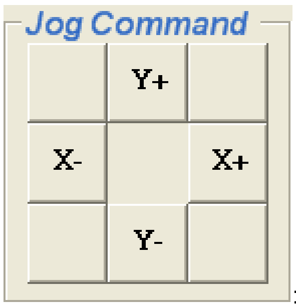
Mode: Linear [✓ Circular] Jog; Step Size 50

Speed: X 0; Y 0

Motion status: X Normal stopped condition; Y Normal stopped condition

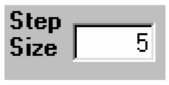
Command: X 40000; Y -16947

Current Position: X 40000; Y -16947

1. **Jog Type**:
   Continuous Jog
   

   : Continuous Jog means that when you press one directional button, the axis will continuously move with an increasing speed. The longer you press, the faster it runs. When you un-press the button, the axis will stop immediately.

Incremental Jog

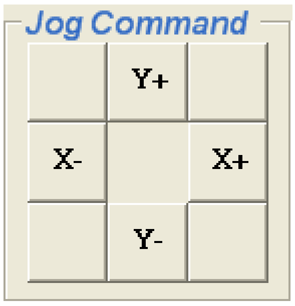

Step Size 5

: Incremental jog means that when you click one directional button, the axis will step a distance according to the Step-Size's setting.

2. **Jog Setting:** Set the parameters for single axis motion. This parameter is meaningless if "Jog Mode" is selected, since the velocity and moving distance is decided by pulse input.
   - **Start Velocity:** Set the start velocity of motion in units of PPS.
   - **Maximum Velocity:** Set the maximum velocity of motion in units of PPS.
   - **Tacc**: Set the acceleration time in units of second.
3. **Operation Mode:** Select operation mode.
   - **Absolute Mode:** "Position" will be used as absolution target positions for motion when "Linear Interpolation Mode" is selected. "ABS EndPos" and "ABS Center" will be used as absolution target positions for motion when "Circular Interpolation Mode" is selected. The related functions are *_8102_start_ta_move()*, *_8102_start_sa_move()*.
   - **Relative Mode:** "Distance" will be used as absolution target positions for motion when "Linear Interpolation Mode" is selected. "Dis EndPos" and "Dis Center" will be used as absolution target positions for motion when "Circular Interpolation Mode" is selected. The related function is *_8102_start_tr_move(), _8102_start_sr_move()*.
4. **DIR:** Specified direction of arc, CW/CCW, It is only effective when "Circular Interpolation Mode" is selected.
5. **Vel. Profile:** Select the velocity profile. Both Trapezoidal and S-Curve are available for "Linear Interpolation Mode" and "Circular Interpolation Mode".
6. **Speed Parameters:** Set the parameters for single axis motion. This parameter is meaningless if "Linear Interpolation Mode" or "Circular Interpolation Mode" is selected, since the velocity and moving distance is decided by pulse input.
   - **Start Velocity:** Set the start velocity of motion in units of PPS.

- **Maximum Velocity:** Set the maximum velocity of motion in units of PPS.
- **Accel. Time:** Set the acceleration time in units of second.
- **Decel. Time**: Set the deceleration time in units of second.
- **SVacc**: Set the S-curve range during acceleration in units of PPS.
- **SVdec**: Set the S-curve range during deceleration in units of PPS.

7. **Set Distance/End Pos:** Set the absolution target positions or relative distance for “Linear Interpolation Mode” . Set the position end of arc for “Circular Interpolation Mode”. It is available for “Linear Interpolation Mode” and “Circular Interpolation Mode”.
8. **Set Center:** Set the position of center for “Circular Interpolation Mode”. It is only effective when “Circular Interpolation Mode” is selected.
9. **Jog Command:** Press one directional button to move.



10. **Velocity:** The absolute value of velocity in units of PPS. The related function is *_8102_get_current_speed().*
11. **Interpolation Command**:
    - Command: displays the value of the command counter. The related function is *_8102_get_command().*
12. **Current Position**:
    - Feedback: displays the value of the feedback position counter. The related function is *_8102_get_position().*
13. **Home Mode:** Home return motion. Clicking this button will invoke the home move configuration window. The related function is *_8102_set_home_config().*There are two home return buttons at the

left-down corner of this window. It is useful when user need to return to the origin.

14. Mode:

Linear Interpolation

: After setting motion parameters correctly in “Motion Parameters Setting Frame”, you can enter the destination in this frame. Then click Run button to start linear interpolation motion.

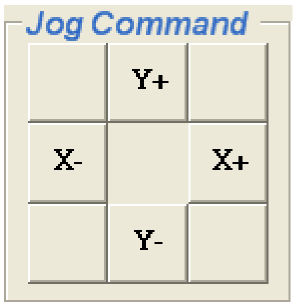
Circular Interpolation

: The setting for circular interpolation mode has three additional parameters in “Motion Parameters Setting Frame”. They are arc degree, division axis and optimize option. Please refer to section 6.7 ,6.8 to set them.

After setting these parameters, you can enter the arc center and degree in “Play Key Frame”. Click Run button to start circular interpolation motion.

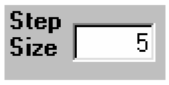
Jog Type:

Continuous Jog



: Continuous Jog means that when you press one directional button, the axis will continuously move with an increasing speed. The longer you press, the faster it runs. When you un-press the button, the axis will stop immediately.

Incremental Jog



: Incremental jog means that when you click one directional button, the axis will step a distance according to the Step-Size's setting.

15. **Motion status:** Displays the returned value of the _8102_motion_done function. The related function is *_8102_motion_done().*

16. **Play Key:**

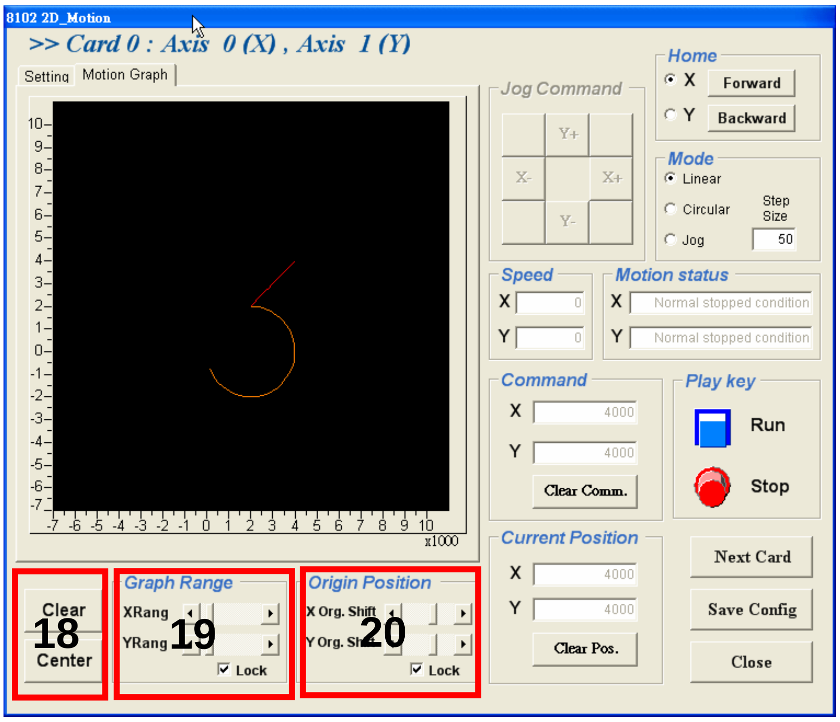
    Play button: Clicking this button will cause the 8102 start to outlet pulses according to previous setting.

    - In "*Linear Mode,*" it causes the axis to move to Distance. The related function is *_8102_start_tr_move_xy, _8102_start_sr_move_xy.*
    - In "*Circular Mode,*" it causes the axis to move to Distance(By Pos/Dist(pulse)).The related function is *_8102_start_tr_arc_xy, _8102_start_sr_arc_xy.*

    Stop Button: Clicking this button will cause the 8102 to decelerate and stop. The deceleration time is defined in "Decel. Time." The related function is *_8102_sd_stop().*

17. **Buttons**:

    - **Next Card:** Change operating card.
    - **Save Config:** Save current configuration to 8102.ini And 8102MC.ini.
    - **Close:** Close the menu.



18. **Graph Range Frame:**

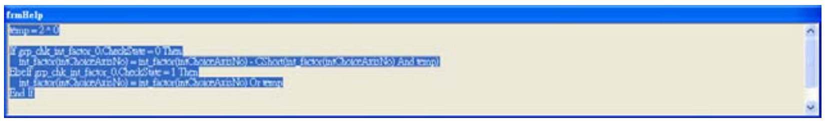
    - **Clear:** Clear the Motion Graph.
    - **Center:** Display the Motion Graph in center position.

19. **Graph Range:** controls X or Y axis's display range.

20. **Origin Position:** let user to pan the display location.

5.3.8 Help Menu

In this menu, users can Click Mouse Right Key to show Help Information.



```
temp = 2 ^ 0

If grp_chk_int_factor_0.CheckState = 0 Then
   int_factor(intChoiceAxisNo) = int_factor(intChoiceAxisNo) - CShort(int_factor(intChoiceAxisNo) And temp)
ElseIf grp_chk_int_factor_0.CheckState = 1 Then
   int_factor(intChoiceAxisNo) = int_factor(intChoiceAxisNo) Or temp
End If
```

6

Function Library

This chapter describes the supporting software for the PCI-8102 card. User can use these functions to develop programs in C, C++, or Visual Basic. If Delphi is used as the programming environment, it is necessary to transform the header files, pci_8102.h manually.

6.1 List of Functions

Sysetm & Initialization Section 6.3

Function Name	Description
_8102_initial	Card initialization
_8102_close	Card Close
_8102_get_version	Check the hardware and software version
_8102_set_user_code	Set codes into EEPROM
_8102_get_user_code	Get codes from EEPROM
_8102_config_from_file	Config PCI-8102 setting from file

Pulse Input/Output Configuration Section 6.4

Function Name	Description
_8102_set_pls_outmode	Set pulse command output mode
_8102_set_pls_iptmode	Set encoder input mode
_8102_set_feedback_src	Set counter input source

Velocity mode motion Section 6.5

Function Name	Description
_8102_tv_move	Accelerate an axis to a constant velocity with trapezoidal profile

_8102_sv_move	Accelerate an axis to a constant velocity with S-curve profile
_8102_sd_stop	Decelerate to stop
_8102_emg_stop	Immediately stop
_8102_get_current_speed	Get current speed(pulse/sec)
_8102_speed_override	Change speed on the fly
_8102_set_max_override_speed	Set the maximum orerride speed

Single Axis Position Mode Section 6.6

Function Name	Description
_8102_start_tr_move	Begin a relative trapezoidal profile move
_8102_start_ta_move	Begin an absolute trapezoidal profile move
_8102_start_sr_move	Begin a relative S-curve profile move
_8102_start_sa_move	Begin an absolute S-curve profile move
_8102_set_move_ratio	Set the ratio of command pulse and feedback pulse.
_8102_position_override	Change position on the fly

Linear Interpolated Motion Section 6.7

Function Name	Description
_8102_start_tr_move_xy	Begin a relative 2-axis linear interpolation for X & Y, with trapezoidal profile
_8102_start_ta_move_xy	Begin an absolute 2-axis linear interpolation for X & Y, with trapezoidal profile
_8102_start_sr_move_xy	Begin a relative 2-axis linear interpolation for X & Y, with S-curve profile
_8102_start_sa_move_xy	Begin an absolute 2-axis linear interpolation for X & Y, with S-curve profile

Circular Interpolation Motion Section 6.8

Function Name	Description
_8102_start_tr_arc_xy	Begin a t-curve relative circular interpolation for X & Y
_8102_start_ta_arc_xy	Begin a t-curve absolute circular interpolation for X & Y
_8102_start_sr_arc_xy	Begin a s-curve relative circular interpolation for X & Y
_8102_start_sa_arc_xy	Begin a s-curve absolute circular interpolation for X & Y

Home Return Mode Section 6.9

Function Name	Description
_8102_set_home_config	Set the home/index logic configuration
_8102_home_move	Begin a home return action
_8102_home_search	Auto-Search Home Switch

Manual Pulser Motion Section 6.10

Function Name	Description
_8102_set_pulser_iptmode	Set pulser input mode
_8102_disable_pulser_input	Disable the pulser input
_8102_pulser_vmove	Start pulser v move
_8102_pulser_pmove	Start pulser p move
_8102_set_pulser_ratio	Set manual pulser ratio for actual output pulse rate

Motion Status Section 6.11

Function Name	Description
_8102_motion_done	Return the motion status

Motion Interface I/O Section 6.12

Function Name	Description
_8102_set_servo	Set On-Off state of SVON signal
_8102_set_pcs_logic	Set PCS signal's logic
_8102_set_clr_mode	Set CLR signal's mode
_8102_set_inp	Set INP signal's logic and operating mode
_8102_set_alm	Set ALM signal's logic and operating mode
_8102_set_erc	Set ERC signal's logic and timing
_8102_set_sd	Set SD signal's logic and operating mode
_8102_enable_sd	Enable SD signal
_8102_set_limit_logic	Set EL signal's logic
_8102_set_limit_mode	Set EL operating mode
_8102_get_io_status	Get all the motion I/O status of 8102

Interrupt Control Section 6.13

Function Name	Description
_8102_int_control	Enable/Disable INT service
_8102_wait_error_interrupt	Wait error related interrupts
_8102_wait_motion_interrupt	Wait motion related interrupts
_8102_set_motion_int_factor	Set the factors of motion related interrupts
_8102_wait_gpio_interrupt	Waiting GPIO interrupts
_8102_set_gpio_int_factor	Set the factors of general purpose IO related interrupt

Position Control and Counters Section 6.14

Function Name	Description
_8102_get_position	Get the value of the feedback position counter
_8102_set_position	Set the feedback position counter
_8102_get_command	Get the value of the command position counter
_8102_set_command	Set the command position counter
_8102_get_error_counter	Get the value of the position error counter
_8102_reset_error_counter	Reset the position error counter
_8102_get_general_counter	Get the value of the general counter
_8102_set_general_counter	Set the general counter
_8102_get_target_pos	Get the value of the target position recorder
_8102_reset_target_pos	Reset target position recorder

_8102_get_res_distance	Get remaining pulses accumulated from motions
_8102_set_res_distance	Set remaining pulses record

Position Compare and Latch Section 6.15

Function Name	Description
_8102_set_trigger_logic	Set CMP signal logic
_8102_set_error_comparator	Set the error comparator
_8102_set_general_comparator	Set the general comparator
_8102_set_trigger_comparator	Set the trigger comparator
_8102_set_latch_source	Set the latch timing for a counter
_8102_set_ltc_logic	Set theLTC signal's logic
_8102_get_latch_data	Get the latch data

Continuous Motion Section 6.16

Function Name	Description
_8102_set_continuous_move	Enable continuous motion for absolute motion
_8102_check_continuous_buffer	Check if the buffer is empty
_8102_dwell_move	

Multiple Axes Simultaneous Operation Section 6.17

Function Name	Description
_8102_set_tr_move_all	Multi-axis simultaneous operation setup
_8102_set_ta_move_all	Multi-axis simultaneous operation setup
_8102_set_sr_move_all	Multi-axis simultaneous operation setup
_8102_set_sa_move_all	Multi-axis simultaneous operation setup
_8102_start_move_all	Begin a multi-axis trapezoidal profile motion
_8102_stop_move_all	Simultaneously stop multi-axis motion

General-purposed Input/Output Section 6.18

Function Name	Description
_8102_set_gpio_output	Set digital output
_8102_get_gpio_output	Get digital output
_8102_get_gpio_input	Get digital input

Soft Limit Section 6.19

Function Name	Description
_8102_disable_soft_limit	Disable soft limit function
_8102_enable_soft_limit	Enable soft limit function
_8102_set_soft_limit	Set the soft limits

Backlas Compensation / Vibration Suppression Section 6.20

Function Name	Description
_8102_backlash_comp	Set backlash corrective pulse for compensation
_8102_suppress_vibration	Set suppress vibration idle pulse counts
_8102_set_fa_speed	Set FA speed for home mode

Speed Profile Calculation Section 6.21

Function Name	**Description**
_8102_get_tr_move_profile	Get relative trapezoidal speed profile
_8102_get_ta_move_profile	Get absolute trapezoidal speed profile
_8102_get_sr_move_profile	Get relative S-curve speed profile
_8102_get_sa_move_profile	Get absoulte S-curve speed profile

6.2 C/C++ Programming Library

This section details all the functions. The function prototypes and some common data types are declared in **pci_8102.h**. We suggest you use these data types in your application programs. The following table shows the data type names and their range.

Type Name	Description	Range
U8	8-bit ASCII character	0 to 255
I16	16-bit signed integer	-32768 to 32767
U16	16-bit unsigned integer	0 to 65535
I32	32-bit signed long integer	-2147483648 to 2147483647
U32	32-bit unsigned long integer	0 to 4294967295
F32	32-bit single-precision floating-point	-3.402823E38 to 3.402823E38
F64	64-bit double-precision floating-point	-1.797683134862315E308 to 1.797683134862315E309
Boolean	Boolean logic value	TRUE, FALSE

The functions of the PCI-8102's software drivers use full-names to represent the functions real meaning. The naming convention rules are:

In a 'C' programming environment:

_{hardware_model}_{action_name}. e.g. **`_8102_initial()`**.

In order to recognize the difference between a C library and a VB library, a capital "B" is placed at the beginning of each function name e.g. **`B_8102_initial()`**.

6.3 System & Initialization

@ **Name**

_8102_initial – Card initialization
_8102_close – Card close
_8102_get_version – Check hardware and software version information
_8102_set_user_code – Set codes into EEPROM
_8102_get_user_code – Get codes into EEPROM
_8102_config_from_file Config – PCI-8102 setting from file

@ **Description**

_8102_initial:

This function is used to initialze an 8102 card without assigning the hardware resources. All 8102 cards must be initialized by this function before calling other functions in your applications. By setting the parameter “**Manual_ID**”, user can choose the type that the card’s ID is assigned manually or automaticly.

_8102_close:

This function is used to close 8102 card and release its resources, which should be called at the end of your applications.

_8102_get_version:

Lets users read back the firmware’s, driver’s and DLL’s version information.

_8102_set_user_code:

Set your own codes into EEPROM. It can secure users’ applcaition to avoid plagiarism.

_8102_get_user_code:

Get codes that you set by the function “_8102_set_user_code” from EEPROM.

_8102_config_from_file:

This function is used to load the configeration of the PCI-8102 according to specified file. By using Motion Creater, user could test and configure the 8102 correctly. After saving the configuration, the file would be existed in user’s system directory as 8102.ini.
When this function is executed, all 8102 cards in the system will be configured as the following functions were called according to parameters

recorded in 8102.ini.

_8102_set_limit_logic
_8102_set_pcs_logic
_8102_set_ltc_logic
_8102_set_inp
_8102_set_erc
_8102_set_alm
_8102_set_pls_iptmode
_8102_set_pls_outmode
_8102_set_move_ratio
_8102_set_latch_source
_8102_set_feedback_src
_8102_set_home_config
_8102_set_soft_limit
_8102_set_fa_speed
_8102_set_sd

@ Syntax

C/C++(Windows 2000/XP)

I16 _8102_initial(U16 *CardID_InBit, I16 Manual_ID);

I16 _8102_close(void);

I16 _8102_get_version(I16 card_id, I16 *firmware_ver, I32 *driver_ver, I32 *dll_ver);

I16 _8102_set_user_code(I16 card_id, I16 Length, U16 *sec_code);

I16 _8102_get_user_code(I16 card_id, I16 Length, U16 *sec_code);

I16 _8102_config_from_file();

Visual Basic 6(Windows 2000/XP)

B_8102_initial(CardID_InBit As Integer, ByVal Manual_ID As Integer) As Integer

B_8102_close() As Integer

B_8102_get_version(ByVal card_id As Integer, firmware_ver As Integer, driver_ver As Long, dll_ver As Long) As Integer

B_8102_set_user_code(ByVal card_id As Integer, ByVal Length As Integer, sec_code As Integer) As Integer

B_8102_get_user_code(ByVal card_id As Integer, ByVal Length As Integer, sec_code As Integer) As Integer

B_8102_config_from_file() As Integer

@ **Argument**

CardID_InBit: Use Hex number to show ID occupation status in the controller. For example, if user has two boards and one is set to 1(DIP swtich) and the other one is set to 3(DIP switch), you will read back the value as 0x000A because the bit 1 and bit 3 are 1 (Card ID exists) and other bits are OFF.

Manual_ID: Enable the on-board dip switch (SW1) to decide the Card ID

Value meaning:

The CardID could be decided by :

0: the sequence of PCI slot.

1: on board DIP switch (SW1).

card_id: Specify the PCI-8102 card index. The card_id could be decided by DIP switch (SW1) or depend on slot sequence.Please refer to **_8102_initial()**.

firmware_ver: The current firmware version.

driver_ver: The current device driver version.

dll_ver: The current DLL library version.

Length: Array size. Length = 1~12

***sec_code**: A numerical array, the array size would be set between 1 and 12.

6.4 Pulse Input/Output Configuration

@ Name
_8102_set_pls_iptmode – Set the configuration for feedback pulse input.
_8102_set_pls_outmode – Set the configuration for pulse command output.
_8102_set_feedback_src – Enable/Disable the external feedback pulse input

@ Description
_8102_set_pls_iptmode:

Configure the input modes of external feedback pulses. There are 4 types for feedback pulse input. Note that this function makes sense only when the **Src** parameter in **_8102_set_feedback_src()** function is enabled.

_8102_set_pls_outmode:

Configure the output modes of command pulses. There are 6 modes for command pulse output.

_8102_set_feedback_src:

If external encoder feedback is available in the system, set the **Src** parameter in this function to an **Enabled** state. Then, the internal 28-bit up/down counter will count according to the configuration of the **_8102_set_pls_iptmode()** function. Else, the counter will count the command pulse output.

@ Syntax

C/C++(Windows 2000/XP)

```
I16 _8102_set_pls_iptmode(I16 AxisNo, I16 pls_iptmode, I16 pls_logic);
I16 _8102_set_pls_outmode(I16 AxisNo, I16 pls_outmode);
I16 _8102_set_feedback_src(I16 AxisNo, I16 Src);
```

Visual Basic6 (Windows 2000/XP)

```
B_8102_set_pls_iptmode(ByVal AxisNo As Integer, ByVal pls_iptmode
                        As Integer, ByVal pls_logic As Integer) As
                        Integer
B_8102_set_pls_outmode(ByVal AxisNo As Integer, ByVal pls_outmode
                        As Integer) As Integer
B_8102_set_feedback_src(ByVal AxisNo As Integer, ByVal Src As
                        Integer) As Integer
```

@ **Argument**

AxisNo: Axis number designated to configure the pulse input/output. It varied according to users' ID setting. The following is an example:

card_id	Physical axis	AxisNo
0	0	**0**
	1	**1**
1	0	**2**
	1	**3**
2	0	**4**
	1	**…**

pls_iptmode: Encoder feedback pulse input mode setting (EA/EB signals).

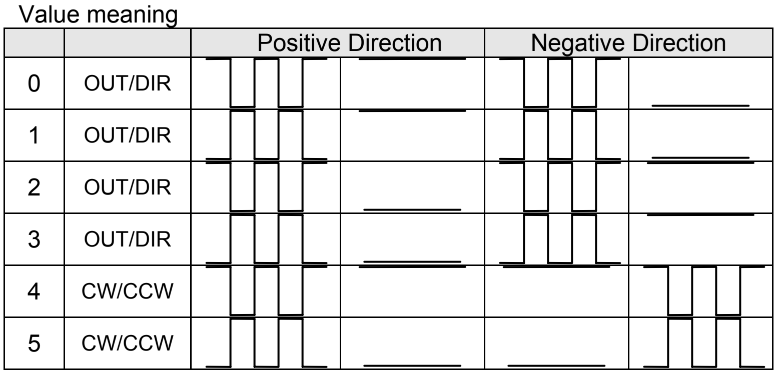
Value	meaning
0	1X A/B
1	2X A/B
2	4X A/B
3	CW/CCW

pls_logic: Logic of encoder feedback pulse.

Value	meaning
0	Not inverse direction
1	inverse direction

pls_outmode: Setting of command pulse output mode.

Value meaning



Src: Counter source

Value	meaning
0	External signal feedback
1	Command pulse

6.5 Velocity mode motion

@ Name

_8102_tv_move – Accelerate an axis to a constant velocity with trapezoidal profile
_8102_sv_move – Accelerate an axis to a constant velocity with S-curve profile
_8102_emg_stop – Immediately stop
_8102_sd_stop – Decelerate to stop
_8102_get_current_speed – Get current speed
_8102_speed_override – Change speed on the fly
_8102_set_max_override_speed – Set the maximum orerride speed

@ Description

_8102_tv_move:

This function is to accelerate an axis to the specified constant velocity with a trapezoidal profile. The axis will continue to travel at a constant velocity until the velocity is changed or the axis is commanded to stop. The direction is determined by the sign of the velocity parameter.

_8102_sv_move:

This function is to accelerate an axis to the specified constant velocity with a S-curve profile. The axis will continue to travel at a constant velocity until the velocity is changed or the axis is commanded to stop. The direction is determined by the sign of velocity parameter.

_8102_emg_stop:

This function is used to **immediately stop** an axis. This function is also useful when a preset move (both trapezoidal and S-curve motion), manual move, or home return function is performed.

_8102_sd_stop:

This function is used to **decelerate an axis to stop** with a trapezoidal or S-curve profile. This function is also useful when a preset move (both trapezoidal and S-curve motion), manual move, or home return function is performed. Note: The velocity profile is decided by original motion profile.

_8102_get_current_speed:

This function is used to read the current pulse output rate (pulse/sec) of a specified axis. It is applicable in any time in any operation mode.

_8102_speed_override:

This function is used to change motion speed on the fly. The overrided speed cannot higher than maximum motion speed. On the other hand, Users also can use the function "**_8102_set_max_override_speed**" to set the maximum override speed which may higher or lower than maximum motion speed before motion.

_8102_set_max_override_speed:

This function is used to set the max orerrided speed. This function is used before velocity motion.

@ Syntax

C/C++(Windows 2000/XP)

I16 _8102_tv_move(I16 AxisNo, F64 StrVel, F64 MaxVel, F64 Tacc);

I16 _8102_sv_move(I16 AxisNo, F64 StrVel, F64 MaxVel, F64 Tacc, F64 SVacc);

I16 _8102_emg_stop(I16 AxisNo);

I16 _8102_sd_stop(I16 AxisNo, F64 Tdec);

I16 _8102_get_current_speed(I16 AxisNo, F64 *speed)

I16 _8102_speed_override(I16 CAxisNo, F64 NewVelPercent, F64 Time);

I16 _8102_set_max_override_speed(I16 AxisNo, F64 OvrdSpeed, I16 Enable);

Visual Basic6 (Windows 2000/XP)

B_8102_tv_move(ByVal AxisNo As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double) As Integer

B_8102_sv_move(ByVal AxisNo As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal SVacc As Double) As Integer

B_8102_emg_stop(ByVal AxisNo As Integer) As Integer

B_8102_sd_stop(ByVal AxisNo As Integer, ByVal Tdec As Double) As Integer

B_8102_get_current_speed(ByVal AxisNo As Integer, ByRef Speed As Double) As Integer

B_8102_speed_override(ByVal CaxisNo As Integer , ByVal NewVelPercent as Double, ByVal Time As Interger);

B_8102_set_max_override_speed(ByVal AxisNo As Integer, ByVal OvrdSpeed As Double, ByVal Enable As Integer);

@ **Argument**

AxisNo: Axis number designated to move or stop.

card_id	Physical axis	AxisNo
0	0	**0**
	1	**1**
1	0	**2**
	1	**3**
2	0	**4**
	1	**…**

StrVel: Starting velocity in units of pulse per second
MaxVel: Maximum velocity in units of pulse per second
Tacc: Specified acceleration time in units of second
SVacc: Specified velocity interval in which S-curve acceleration is performed.
Note: SVacc = 0, for pure S-Curve
Tdec: specified deceleration time in units of second
***Speed**: Variable to get current speed (pulse/sec).

6.6 Single Axis Position Mode

@ Name

_8102_start_tr_move – Begin a relative trapezoidal profile move
_8102_start_ta_move – Begin an absolute trapezoidal profile move
_8102_start_sr_move – Begin a relative S-curve profile move
_8102_start_sa_move – Begin an absolute S-curve profile move
_8102_set_move_ratio – Set the ration of command pulse and feedback pulse
_8102_position_override – Change position on the fly

@ Description

General:

The moving direction is determined by the sign of the **Pos** or **Dist** parameter. If the moving distance is too short to reach the specified velocity, the controller will automatically lower the **MaxVel**, and the **Tacc**, **Tdec**, **VSacc**, and **VSdec** will also become shorter while dV/dt(acceleration / deceleration) and d(dV/dt)/dt (jerk) are keep unchanged.

_8102_start_tr_move:

This function causes the axis to accelerate form a starting velocity (StrVel), rotate at constant velocity (MaxVel), and decelerate to stop at the **relative distance** with **trapezoida**l profile. The acceleration (Tacc) and deceleration (Tdec) time is specified independently–it does not let the program wait for motion completion but immediately returns control to the program.

_8102_start_ta_move:

This function causes the axis to accelerate from a starting velocity (StrVel), rotate at constant velocity (MaxVel), and decelerates to stop at the specified **absolute position** with **trapezoidal** profile. The acceleration (Tacc) and deceleration (Tdec) time is specified independently. This command does not let the program wait for motion completion, but immediately returns control to the program.

_8102_start_sr_move:

This function causes the axis to accelerate from a starting velocity (StrVel), rotate at constant velocity (MaxVel), and decelerates to stop at the **relative distance** with **S-curve** profile. The acceleration (Tacc) and deceleration (Tdec) time is specified independently. This command does not let the program wait for motion completion, but immediately returns control to the program.

_8102_start_sa_move:

This function causes the axis to accelerate from a starting velocity (StrVel), rotate at constant velocity, and decelerates to stop at the specified absolute position with S-curve profile. The acceleration and deceleration time is specified independently.This command does not let the program wait for motion completion but immediately returns control to the program.

_8102_set_move_ratio:
This function configures scale factors for the specified axis. Usually, the axes only need scale factors if their mechanical resolutions are different. For example, if the resolution of feedback sensors is two times resolution of command pulse, then the parameter “**move_ratio**” could be set as 2.

_8102_position_override:

This function is used to change target position on the fly. There are some limitations on this function. Please refer to section 4.2.15 before use it.

@ Syntax

C/C++(Windows 2000/XP)

```
I16 _8102_start_tr_move(I16 AxisNo, F64 Dist, F64 StrVel, F64 MaxVel,
        F64 Tacc, F64 Tdec);
I16 _8102_start_ta_move(I16 AxisNo, F64 Pos, F64 StrVel, F64 MaxVel,
        F64 Tacc, F64 Tdec);
I16 _8102_start_sr_move(I16 AxisNo, F64 Dist, F64 StrVel, F64 MaxVel,
        F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);
I16 _8102_start_sa_move(I16 AxisNo, F64 Pos, F64 StrVel, F64 MaxVel,
        F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);
I16 _8102_set_move_ratio(I16 AxisNo, F64 move_ratio);
I16 _8102_position_override(I16 AxisNo, F64 NewPos);
```

Visual Basic6 (Windows 2000/XP)

```
B_8102_start_tr_move(ByVal AxisNo As Integer, ByVal Dist As Double,
      ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As
      Double, ByVal Tdec As Double) As Integer
B_8102_start_ta_move(ByVal AxisNo As Integer, ByVal Pos As Double,
      ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As
      Double, ByVal Tdec As Double) As Integer
B_8102_start_sr_move(ByVal AxisNo As Integer, ByVal Dist As Double,
      ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As
      Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal
      SVdec As Double) As Integer
```

B_8102_start_sa_move(ByVal AxisNo As Integer, ByVal Pos As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

B_8102_set_move_ratio(ByVal AxisNo As Integer, ByVal move_ratio As Double) As Integer

B_8102_position_override(ByVal AxisNo As Integer, ByVal NewPos As Double) As Integer

@ **Argument**

AxisNo: Axis number designated to move or change position.

card_id	Physical axis	AxisNo
0	0	**0**
	1	**1**
1	0	**2**
	1	**3**
2	0	**4**
	1	**…**

Dist: Specified relative distance to move (unit: pulse)
Pos: Specified absolute position to move (unit: pulse)
StrVel: Starting velocity of a velocity profile in units of pulse per second
MaxVel: Maximum velocity in units of pulse per second
Tacc: Specified acceleration time in units of seconds
Tdec: Specified deceleration time in units of seconds
SVacc: Specified velocity interval in which S-curve acceleration is performed.
Note: SVacc = 0, for pure S-Curve. For more details, see section 4.2.4
SVdec: specified velocity interval in which S-curve deceleration is performed.
Note: SVdec = 0, for pure S-Curve. For more details, see section 4.2.4
Move_ratio: ratio of (feedback resolution)/(command resolution) , should not be 0
NewPos: specified new absolute position to move

6.7 Linear Interpolated Motion

@ Name

_8102_start_tr_move_xy – Begin a relative 2-axis linear interpolation with trapezoidal profile
_8102_start_ta_move_xy – Begin an absolute 2-axis linear interpolation for with trapezoidal profile
_8102_start_sr_move_xy –Begin a relative 2-axis linear interpolation for with S-curve profile
_8102_start_sa_move_xy –Begin an absolute 2-axis linear interpolation for with S-curve profile

@ Description

These functions perform linear interpolation motion with different profile. Detail Comparsions of those functions are described by follow table.

Function	Total axes	Velocity Profile	Relative / Absolute	Target Axes
_8102_start_tr_move_xy	2	T	R	Axes 0 & 1
_8102_start_ta_move_xy	2	T	A	Axes 0 & 1
_8102_start_sr_move_xy	2	S	R	Axes 0 & 1
_8102_start_sa_move_xy	2	S	A	Axes 0 & 1

Velocity profile :
T : trapezoidal profile
S : s curve profile
Relative / Absolute:
R: Relative distance
A: Absoulte position

@ Syntax

C/C++(Windows 2000/XP)

```
I16 _8102_start_tr_move_xy(I16 Card_id, F64 DistX, F64 DistY, F64 StrVel, F64
                           MaxVel, F64 Tacc, F64 Tdec);

I16 _8102_start_ta_move_xy(I16 Card_id, F64 PosX, F64 PosY, F64 StrVel, F64
                           MaxVel, F64 Tacc, F64 Tdec);

I16 _8102_start_sr_move_xy(I16 Card_id, F64 DistX, F64 DistY, F64 StrVel, F64
                           MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);

I16 _8102_start_sa_move_xy(I16 Card_id, F64 PosX, F64 PosY, F64 StrVel, F64
                           MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);
```

Visual Basic6 (Windows 2000/XP)

B_8102_start_tr_move_xy(ByVal CardNo As Integer, ByVal DistX As Double, ByVal DistY As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

B_8102_start_ta_move_xy(ByVal CardNo As Integer, ByVal PosX As Double, ByVal PosY As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

B_8102_start_sr_move_xy(ByVal CardNo As Integer, ByVal DistX As Double, ByVal DistY As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

B_8102_start_sa_move_xy(ByVal CardNo As Integer, ByVal PosX As Double, ByVal PosY As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

@ **Argument**

AxisNo: Axis number designated to move or change position.

card_id	Physical axis	AxisNo
0	0	**0**
	1	**1**
1	0	**2**
	1	**3**
2	0	**4**
	1	**…**

DistX: specified relative distance of **axis 0** to move (unit: pulse).
DistY: specified relative distance of **axis 1** to move (unit: pulse).
PosX: specified absolute position of **axis 0** to move (unit: pulse).
PosY: specified absolute position of **axis 1** to move (unit: pulse).
StrVel: Starting velocity of a velocity profile in units of pulse per second.
MaxVel: Maximum velocity in units of pulse per second.
Tacc: Specified acceleration time in units of seconds.
Tdec: Specified deceleration time in units of seconds.
SVacc: Specified velocity interval in which S-curve acceleration is performed.
Note: SVacc = 0, for pure S-Curve. For more details, see section 4.2.4
SVdec: specified velocity interval in which S-curve deceleration is performed.
Note: SVdec = 0, for pure S-Curve. For more details, see section 4.2.4

6.8 Circular Interpolation Motion

@ Name

_8102_start_tr_arc_xy – Begin a T-curve relative circular interpolation
_8102_start_ta_arc_xy – Begin a T-curve absolute circular interpolation
_8102_start_sr_arc_xy – Begin a S-curve relative circular interpolation
_8102_start_sa_arc_xy –Begin a S-curve absolute circular interpolation

@ Description

Those functions perform Circular interpolation motion with different profile. Detail Comparsions of those functions are described by follow table.

Function	Total axes	Velocity Profile	Relative / Absolute	Target Axes
_8102_start_tr_arc_xy	2	trapezoidal	R	Axes 0 & 1
_8102_start_ta_arc_xy	2	trapezoidal	A	Axes 0 & 1
_8102_start_sr_arc_xy	2	S-curve	R	Axes 0 & 1
_8102_start_sa_arc_xy	2	S-curve	A	Axes 0 & 1

@ Syntax

C/C++(Windows 2000/XP)

```
I16 _8102_start_tr_arc_xy(I16 Card_id, F64 OffsetCx, F64 OffsetCy, F64 OffsetEx,
        F64 OffsetEy, I16 DIR, F64 StrVel,F64 MaxVel,F64 Tacc,F64 Tdec);
```

```
I16 _8102_start_ta_arc_xy(I16 Card_id, F64 Cx, F64 Cy, F64 Ex, F64 Ey, I16 DIR,
        F64 StrVel,F64 MaxVel,F64 Tacc,F64 Tdec);
```

```
I16 _8102_start_sr_arc_xy(I16 Card_id, F64 OffsetCx, F64 OffsetCy, F64 OffsetEx,
        F64 OffsetEy, I16 DIR, F64 StrVel,F64 MaxVel,F64 Tacc,F64 Tdec,F64
        SVacc,F64 SVdec);
```

```
I16 _8102_start_sa_arc_xy(I16 Card_id, F64 Cx, F64 Cy, F64 Ex, F64 Ey, I16 DIR,
        F64 StrVel,F64 MaxVel,F64 Tacc,F64 Tdec,F64 SVacc,F64 SVdec);
```

Visual Basic6 (Windows 2000/XP)

```
B_8102_start_tr_arc_xy(ByVal CardNo As Integer, ByVal OffsetCx As Double, ByVal
        OffsetCy As Double, ByVal OffsetEx As Double, ByVal OffsetEy As Double,
        ByVal CW_CCW As Integer, ByVal StrVel As Double, ByVal MaxVel As
        Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer
```

```
B_8102_start_ta_arc_xy(ByVal CardNo As Integer, ByVal Cx As Double, ByVal Cy
        As Double, ByVal Ex As Double, ByVal Ey As Double, ByVal CW_CCW As
        Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As
        Double, ByVal Tdec As Double) As Integer
```

```
B_8102_start_sr_arc_xy(ByVal CardNo As Integer, ByVal OffsetCx As Double, ByVal
        OffsetCy As Double, ByVal OffsetEx As Double, ByVal OffsetEy As Double,
```

ByVal CW_CCW As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

B_8102_start_sa_arc_xy(ByVal CardNo As Integer, ByVal Cx As Double, ByVal Cy As Double, ByVal Ex As Double, ByVal Ey As Double, ByVal CW_CCW As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

@ **Argument**

AxisNo: Axis number designated to move or change position.

card_id	Physical axis	AxisNo
0	0	**0**
	1	**1**
1	0	**2**
	1	**3**
2	0	**4**
	1	**…**

OffsetCx: X-axis offset to center
OffsetCy: Y-axis offset to center
OffsetEx: X-axis offset to end of arc
OffsetEy: Y-axis offset to end of arc
DIR: Specified direction of arc

Value	meaning
0	Clockwise(cw)
1	Counterclockwise(ccw)

StrVel: Starting velocity of a velocity profile in units of pulse per second.
MaxVel: Maximum velocity in units of pulse per second.
Tacc: Specified acceleration time in units of seconds.
Tdec: Specified deceleration time in units of seconds.
SVacc: Specified velocity interval in which S-curve acceleration is performed.
Note: SVacc = 0, for pure S-Curve. For more details, see section 4.2.4
SVdec: specified velocity interval in which S-curve deceleration is performed.
Note: SVdec = 0, for pure S-Curve. For more details, see section 4.2.4

6.9 Home Return Mode

@ Name

_8102_set_home_config – Set the configuration for home return move motion
_8102_home_move – Perform a home return move.
_8102_home_search – Auto-Search

@ Description

_8102_set_home_config

Configures the home return mode, origin(ORG) and index signal(EZ) logic, EZ count, and ERC output options for the home_move() function. Refer to Section 4.2.10 for the setting of home_mode control.

_8102_home_move –

This function will cause the axis to perform a home return move according to the **_8164_set_home_config()** function settings. The direction of movement is determined by the sign of velocity parameter (MaxVel). Since the stopping condition of this function is determined by the ***home_mode*** setting, users should take care in selecting the initial moving direction. Users should also take care to handle conditions when the limit switch is touched or other conditions that are possible causing the axis to stop. For more detail discribtion, see Section 4.2.10

_8102_home_search –

This function will cause the axis to perform a home-search move according to the **_8164_set_home_config()** function settings. The direction of movement is determined by the sign of velocity parameter (MaxVel). Since the stopping condition of this function is determined by the ***home_mode*** setting, users should take care in selecting the initial moving direction. Users should also take care to handle conditions when the limit switch is touched or other conditions that are possible causing the axis to stop. For more detail discribtion, see Section 4.2.10

@ Syntax

C/C++(Windows 2000/XP)

```
I16 _8102_set_home_config(I16 AxisNo, I16 home_mode, I16 org_logic, I16 ez_logic,
                          I16 ez_count, I16 erc_out);

I16 _8102_home_move(I16 AxisNo, F64 StrVel, F64 MaxVel, F64 Tacc);

I16 _8102_home_search(I16 AxisNo, F64 StrVel, F64 MaxVel, F64 Tacc, F64
                      ORGOffset);
```

Visual Basic (Windows 2000/XP)

```
B_8102_set_home_config(ByVal AxisNo As Integer, ByVal home_mode As Integer,
            ByVal org_logic As Integer, ByVal ez_logic As Integer, ByVal ez_count
            As Integer, ByVal erc_out As Integer) As Integer
B_8102_home_move(ByVal AxisNo As Integer, ByVal StrVel As Double, ByVal
            MaxVel As Double, ByVal Tacc As Double) As Integer
B_8102_home_search(ByVal AxisNo As Integer, ByVal StrVel As Double, ByVal
            MaxVel As Double, ByVal Tacc As Double, ByVal ORGOffset As Double)
            As Integer
```

@ **Argument**

AxisNo: Axis number designated to move or change position.

card_id	Physical axis	AxisNo
0	0	**0**
	1	**1**
1	0	**2**
	1	**3**
2	0	**4**
	1	**…**

home_mode: Stopping modes for home return, This value is between 0 to 12. Please see Section 4.2.10

org_logic: Action logic configuration for ORG

Value	meaning
0	Active low
1	Active high

ez_logic: Action logic configuration for EZ

Value	meaning
0	Active low
1	Active high

ez_count: 0~15 (Please refer to see Section 4.2.10)
erc_out: Set ERC output options.

Value	meaning
0	no ERC out
1	ERC signal out when home-move finishing

StrVel: Starting velocity of a velocity profile. (unit: pulse/sec)
MaxVel: Maximum velocity. (unit: pulse/sec)
Tacc: Specified acceleration time (Unit: sec)
ORGOffset: The escape pulse amounts when home search touches the ORG singal (Unit: pulse)

6.10 Manual Pulser Motion

@ Name
_8102_disable_pulser_input – Disable the pulser input
_8102_pulser_pmove – Manual pulser p_move
_8102_pulser_vmove – Manual pulser v_move
_8102_set_pulser_ratio – Set manual pulser ratio for actual output pulse rate
_8102_set_pulser_iptmode – Set the input signal modes of pulser

@ Description
_8102_disable_pulser_input
This function is used to set the pulser input disabel or enabel.

_8102_pulser_pmove
With this command, the axis begins to move according to the manual pulse input. The axis will output one pulse when it receives one manual pulse, until the **_8102_disable_pulser_input** function disables the pulser or the output pulse number reaches the distance.

_8102_pulser_vmove
With this command, the axis begins to move according to the manual pulse input. The axis will output one pulse when it receives one manual pulse, until the **_8102_disable_pulser_input** function disables the pulser.

_8102_set_pulser_ratio
Set manual pulse ratio for actual output pulse rate. The formula for manual pulse output rate is:

Output Pulse Count = Input Pulser Count $\times$ 4 (MultiF +1) $\times$(DivF +1) / 2048
The DivF = 0~2047 Divide Factor
The MultiF= 0~31 Multiplication Factor

_8102_set_pulser_iptmode
This function is used to configure the input mode of manual pulser.

@ Syntax

C/C++(Windows 2000/XP)

```
I16 _8102_disable_pulser_input(I16 AxisNo, U16 Disable );

I16 _8102_pulser_pmove(I16 AxisNo, F64 Dist, F64 SpeedLimit);

I16 _8102_pulser_vmove(I16 AxisNo, F64 SpeedLimit);

I16 _8102_set_pulser_ratio(I16 AxisNo, I16 DivF, I16 MultiF);

I16 _8102_set_pulser_iptmode(I16 AxisNo, I16 InputMode, I16 Inverse);
```

Visual Basic (Windows 2000/XP)

B_8102_disable_pulser_input(ByVal AxisNo As Integer, ByVal Disable As Integer) As Integer

B_8102_pulser_pmove(ByVal AxisNo As Integer, ByVal Dist As Double, ByVal SpeedLimit As Double) As Integer

B_8102_pulser_vmove(ByVal AxisNo As Integer, ByVal SpeedLimit As Double) As Integer

B_8102_set_pulser_ratio(ByVal AxisNo As Integer, ByVal DivF As Integer, ByVal MultiF As Integer) As Integer

B_8102_set_pulser_iptmode(ByVal AxisNo As Integer, ByVal InputMode As Integer, ByVal Inverse As Integer) As Integer

@ **Argument**

AxisNo: Axis number designated to move or change position.

card_id	Physical axis	AxisNo
0	0	**0**
	1	**1**
1	0	**2**
	1	**3**
2	0	**4**
	1	**...**

Disable: Disable pulser input.
Disable = 1, disable pulser
Disable = 0, enable pulser

Dist: Specified relative distance to move (unit: pulse)
For example, if SpeedLimit is set to be 100pps, then the axis can move at fastest 100pps , even the input pulser signal rate is more then 100pps.

DivF: Divide factor (0~2047)

MultiF: Multiplication factor (0~31)

InputMode: Setting of manual pulse input mode from the PA and PB pins

Value	meaning
0	1X AB phase type pulse input
1	2X AB phase type pulse input
2	4X AB phase type pulse input
3	CW/CCW type pulse input

Inverse: Reverse the moving direction from pulse direction

Value	meaning
0	no inverse

1	Reverse moving direction

6.11 Motion Status

@ Name
_8102_motion_done – Return the motion status

@ Description
_8102_motion_done:
Return the motion status of the 8102. The return code show as below:

0	Normal stopped condition
1	Waiting for DR
2	Waiting for CSTA input
3	Waiting for an internal synchronous signal
4	Waiting for another axis to stop
5	Waiting for a completion of ERC timer
6	Waiting for a completion of direction change timer
7	Correcting backlash
8	Wait PA/PB
9	At FA speed
10	At FL Speed
11	Accelerating
12	At FH Speed
13	Decelerating
14	Wait INP
15	Others(Controlling Start)
16	SALM
17	SPEL
18	SMEL
19	SEMG
20	SSTP
21	SERC

@ Syntax

C/C++(Windows 2000/XP)

```
I16 _8102_motion_done(I16 AxisNo)
```

Visual Basic (Windows 2000/XP)

```
B_8102_motion_done(ByVal AxisNo As Integer) As Integer
```

@ Argument
AxisNo: Axis number of Target Axis.

card_id	Physical	AxisNo

	axis	
0	0	**0**
	1	**1**
1	0	**2**
	1	**3**
2	0	**4**
	1	**...**

6.12 Motion Interface I/O

@ Name
_8102_set_servo – Set the ON-OFF state of the SVON signal
_8102_set_pcs_logic – Set the logic of PCS signal
_8102_set_lcr_mode – Set the mode of CLR signal
_8102_set_inp – Set the logic of INP signal and operating mode
_8102_set_alm – Set the logic of ALM signal and operating mode
_8102_set_erc – Set the logic of ERC signal and operating mode
_8102_set_sd – Set the logic SD signal and operating mode
_8102_enable_sd – Enable SD signal
_8102_set_limit_logic – Set the logic of PEL/MEL signal
_8102_set_limit_mode – Set PEL/MEL operating mode
_8102_select_pin23_input – set pin NO.23 signal source
_8102_select_pin57_input – set pin No.57 signal source
_8102_get_io_status –Get all the motion I/O statuses of each 8102

@ Description
_8102_set_servo:

You can set the ON-OFF state of the SVON signal with this function. The default value is 1(OFF), which means the SVON is open to GND.

_8102_set_pcs_logic:
Set the active logic of the PCS signal input

_8102_set_lcr_mode
CLR inputed signal can reset specified counters from counter 1 to 4. The reset action could be set by this function. The reset action mode has 4 types. For details refer to argument.

_8102_set_inp:

Set the active logic of the In-Position signal input from the servo driver. Users can select whether they want to enable this function. It is disabled by default.

_8102_set_alm:

Set the active logic of the ALARM signal input from the servo driver. Two reacting modes are available when the ALARM signal is active.

_8102_set_erc:

You can set the logic and on time of the ERC with this function. It also can set the pulser width of ERC signal.

_8102_set_sd:

Set the active logic, latch control, and operating mode of the SD signal input from a mechanical system. Users can select whether they want to enable this function by **_8102_enable_sd**. It is disabled by default

_8102_enable_sd:
Enable the SD signal input. Default setting is default.

_8102_set_limit_logic:

Set the EL logic, normal open or normal closed.

_8102_set_limit_mode:

Set the reacting modes of the EL signal.

_8102_select_pin23_input:
Set the pin 23 to the input signal SD1, LTC1 or PCS1.

_8102_select_pin57_input:
Set the pin 57 to the input signal SD2, LTC2 or PCS2
.
_8102_get_io_status:

Get all the I/O statuses for each axis. The definition for each bit is as follows:

Bit	Name	Description
0	RDY	RDY pin input
1	ALM	Alarm Signal
2	+EL	Positive Limit Switch
3	-EL	Negative Limit Switch
4	ORG	Origin Switch
5	DIR	DIR output
6	EMG	EMG status
7	PCS	PCS signal input
8	ERC	ERC pin output
9	EZ	Index signal
10	CLR	Clear signal
11	LTC	Latch signal input
12	SD	Slow Down signal input
13	INP	In-Position signal input
14	SVON	Servo-ON output status

@ Syntax

C/C++(Windows 2000/XP)

I16 _8102_set_servo(I16 AxisNo, I16 on_off);

I16 _8102_set_pcs_logic(I16 AxisNo, I16 pcs_logic);

I16 _8102_set_clr_mode(I16 AxisNo, I16 clr_mode);

I16 _8102_set_inp(I16 AxisNo, I16 inp_enable, I16 inp_logic);

I16 _8102_set_alm(I16 AxisNo, I16 alm_logic, I16 alm_mode);

I16 _8102_set_erc(I16 AxisNo, I16 erc_logic, I16 erc_pulse_width);

I16 _8102_set_sd(I16 AxisNo, I16 sd_logic, I16 sd_latch, I16 sd_mode);

I16 _8102_enable_sd(I16 AxisNo, I16 enable);

I16 _8102_set_limit_logic(I16 AxisNo, U16 Logic);

I16 _8102_set_limit_mode(I16 AxisNo, I16 limit_mode);

I16 _8102_select_pin23_input(I16 card_id, U16 Select);

I16 _8102_select_pin57_input(I16 card_id, U16 Select);

Visual Basic (Windows 2000/XP)

B_8102_set_servo(ByVal AxisNo As Integer, ByVal on_off As Integer) As Integer

B_8102_set_pcs_logic(ByVal AxisNo As Integer, ByVal pcs_logic As Integer) As Integer

B_8102_set_clr_mode(ByVal AxisNo As Integer, ByVal clr_mode As Integer) As Integer

B_8102_set_inp(ByVal AxisNo As Integer, ByVal inp_enable As Integer, ByVal inp_logic As Integer) As Integer

B_8102_set_alm(ByVal AxisNo As Integer, ByVal alm_logic As Integer, ByVal alm_mode As Integer) As Integer

B_8102_set_erc(ByVal AxisNo As Integer, ByVal erc_logic As Integer, ByVal erc_pulse_width As Integer) As Integer

B_8102_set_sd(ByVal AxisNo As Integer, ByVal sd_logic As Integer, ByVal sd_latch As Integer, ByVal sd_mode As Integer) As Integer

B_8102_enable_sd(ByVal AxisNo As Integer, ByVal Enable As Integer) As Integer

B_8102_set_limit_logic(ByVal AxisNo As Integer, ByVal Logic As Integer) As Integer

B_8102_set_limit_mode(ByVal AxisNo As Integer, ByVal limit_mode As Integer) As Integer

B_8102_select_pin23_input (ByVal card_id As Integer, ByVal SelectNo As Integer) As Integer

B_8102_select_pin57_input(ByVal card_id As Integer, ByVal SelectNo As Integer) As Integer

@ **Argument**

AxisNo: Axis number of Target Axis.

card_id	Physical axis	AxisNo
0	0	**0**
	1	**1**
1	0	**2**
	1	**3**
2	0	**4**
	1	**…**

on_off: ON-OFF state of SVON signal

Value	meaning
0	ON
1	OFF

pcs_logic: PCS signal input logic

Value	meaning
0	Negative logic
1	Positive logic

clr_mode: Clear action mode.

clr_mode = 0 , Clear on the falling edge (default)
clr_mode = 1 , Clear on the rising edge
clr_mode = 2 , Clear on a LOW level
clr_mode = 3 , Clear on a HIGH level

inp_enable: INP function enabled/disabled

inp_enable = 0, Disabled (default)
inp_enable = 1, Enabled

inp_logic: Set the active logic for the INP signal

Value	meaning
0	Negative logic
1	Positive logic

alm_logic: Setting of active logic for ALARM signals

Value meaning

0	Negative logic
1	Positive logic

alm_mode: Reacting modes when receiving an ALARM signal.

Value	meaning
0	motor immediately stops (default)
1	motor decelerates then stops

erc_logic: Set the active logic for the ERC signal

Value	meaning
0	Negative logic
1	Positive logic

erc_pulse_width: Set the pulse width of the ERC signal

Value	meaning
0	12 μs
1	102 μs
2	409 μs
3	1.6 ms
4	13 ms
5	52 ms
6	104 ms
7	Level output

sd_logic:

Value	meaning
0	Negative logic
1	Positive logic

sd_latch: Set the latch control for the SD signal

Value	meaning
0	Do not latch
1	latch

sd_mode: Set the reacting mode of the SD signal

Value	meaning
0	slow down only
1	slow down then stop

enable: Set the ramping-down point for high speed feed.

Value	meaning
0	Automatic setting
1	Manual setting (default)

Logic: Set the PEL/MEL logic.

Value	meaning
0	Normal low(normal

	open)
1	Normal high(normal close)

limit_mode:

Value	meaning
0	Stop immediatelly
1	Slow down then stop

Select: select the corresponding signal to specified pin

Value	meaning
0	CLR pin
1	LTC pin
2	SD pin
3	PCS pin

6.13 Interrupt Control

@ Name

_8102_int_control – Enable/Disable INT service
_8102_set_motion_int_factor – Set the factors of motion related interrupts
_8102_set_gpio_int_factor – Set the factors of general purpose IO related interrupts
_8102_wait_error_interrupt – Wait error related interrupts
_8102_wait_motion_interrupt – Wait motion related interrupts
_8102_wait_gpio_interrupt – Waiting GPIO interrupts

@ Description

_8102_int_control:

This function is used to enable the Windows interrupt event to host PC.

_8102_set_motion_int_factor:

This function allows users to select motion related factors to initiate the event int. The error can never be masked once the interrupt service is turned on by *_8102_int_control()*. Once the Interrupt function is enabled, you can use *_8102_wait_motion_interrupt()* to wait event.

_8102_set_gpio_int_factor:

This function allows users to select GPIO related factors to initiate the event int. The error can never be masked once the interrupt service is turned on by *_8102_int_control()*. Once the Interrupt function is enabled, you can use *_8102_wait_gpio_interrupt()* to wait event.

_8102_wait_error_interrupt:

When user enabled the Interrupt function by *_8102_int_control()*. He could use this function to wait the errior interrupts.

_8102_wait_motion_interrupt:

When user enabled the Interrupt function by *_8102_int_control(*) and set the interrupt factors by *_8102_set_motion_int_factor()*. User could use this function to wait the specific interrupt. When this function was running, the process would never stop until evens were triggered or the function was time out.

_8102_wait_gpio_interrupt:

When user enabled the Interrupt function by *_8102_int_control()* and set the interrupt factors by *_8102_set_gpio_int_factor()*. User could use this function to wait the specific interrupt. When this function was running, the process would never stop until evens were triggered or the function was time out.

@ Syntax

C/C++(Windows 2000/XP)

I16 _8102_int_control(I16 card_id, I16 intFlag);

I16 _8102_set_motion_int_factor(I16 AxisNo, U32 int_factor);

I16 _8102_set_gpio_int_factor(I16 card_id, U16 int_factor);

I16 _8102_wait_error_interrupt(I16 AxisNo, I32 TimeOut_ms);

I16 _8102_wait_motion_interrupt(I16 AxisNo, I16 IntFactorBitNo, I32 TimeOut_ms);

I16 _8102_wait_gpio_interrupt(I16 card_id, I16 IntFactorBitNo, I32 TimeOut_ms);

Visual Basic (Windows 2000/XP)

B_8102_int_control(ByVal card_id As Integer, ByVal intFlag As Integer) As Integer

B_8102_wait_error_interrupt(ByVal AxisNo As Integer, ByVal TimeOut_ms As Long) As Integer

B_8102_wait_motion_interrupt(ByVal AxisNo As Integer, ByVal IntFactorBitNo As Integer, ByVal TimeOut_ms As Long) As Integer

B_8102_set_motion_int_factor(ByVal AxisNo As Integer, ByVal int_factor As Long) As Integer

B_8102_wait_gpio_interrupt(ByVal card_id As Integer, ByVal IntFactorBitNo As Integer, ByVal TimeOut_ms As Long) As Integer

B_8102_set_gpio_int_factor(ByVal card_id As Integer, ByVal int_factor As Integer) As Integer

@ Argument

card_id: Specify the index of target PCI-8102 card. The card_id could be decided by DIP switch (SW1) or depend on slot sequence.Please refer to _8102_initial().

AxisNo: Axis number of Target Axis.

card_id	Physical axis	AxisNo
0	0	**0**
	1	**1**
1	0	**2**
	1	**3**
2	0	**4**

	1	**...**

int_factor: interrupt factor

motion INT factors

Value meaning (0: Disable, 1:Enable)

Bit	Description
0	Normal stop
1	Starting the next operation continuously
2	Writing to the 2nd pre-register
3	Writing to the 2nd pre-register for trigger comparator
4	Start acceleration
5	Acceleration end
6	Start deceleration
7	Deceleration end
8	When soft limit turn on (positive direction)
9	When soft limit turn on (negetive direction)
10	When error comparator conditions are met
11	When general comparator conditions are met
12	When trigger comparator conditions are met
13	When resetting the count value with a CLR signal input
14	When Latching the count value with a LTC signal input
15	When Latching the count value with an ORG signal input
16	When the SD input is ON
17	When the +/-DR input is changed
18	When the CSTA input is ON
19~31	Not define (Always set to 0)

GPIO INT factors

Value meaning (0: Disable, 1:Enable)

Bit	Description
0	Digital Input 0 falling dege
1	Digital Input 1 falling dege
2	Digital Input 2 falling dege
3	Digital Input 3 falling dege
4	Digital Input 0 rising dege
5	Digital Input 1 rising dege
6	Digital Input 2 rising dege
7	Digital Input 3 rising dege
8	CLR/LTC/SD/PCS input 0 Interrupt enable
9	CLR/LTC/SD/PCS input 1 Interrupt enable

10	CLR/LTC/SD/PCS mode (0: falling edge, 1: rising edge)
11~15	Not define (Always set to 0)

TimeOut_ms: Specifies the time-out interval, in milliseconds.

IntFactorBitNo: Specifies the bit number of the INT factor

6.14 Position Control and Counters

@ Name
_8102_get_position – Get the value of feedback position counter
_8102_set_position – Set the feedback position counter
_8102_get_command – Get the value of command position counter
_8102_set_command – Set the command position counter
_8102_get_error_counter – Get the value of position error counter
_8102_reset_error_counter – Reset the position error counter
_8102_get_general_counter – get the value of general counter
_8102_set_general_counter – Set the general counter
_8102_get_target_pos – Get the value of target position recorder
_8102_reset_target_pos – Reset target position recorder
_8102_get_res_distance – Get remaining pulses accumulated from motions
_8102_set_res_distance – Set remaining pulses record

@ Description
_8102_get_position:
This function is used to read the feedback position counter value. Note that this value has already been processed by the move ratio setting by *_8102_set_move_ratio()*. If the move ratio is 0.5, than the value of position will be twice. The source of the feedback counter is selectable by the function *_8102_set_feedback_src()* to be external EA/EB or internal pulse output of 8102 .

_8102_set_position:
This function is used to change the feedback position counter to the specified value. Note that the value to be set will be processed by the move ratio. If move ratio is 0.5, then the set value will be twice as given value.

_8102_get_command:
This function is used to read the value of the command position counter. The source of the command position counter is the pulse output of the 8102.

_8102_set_command:
This function is used to change the value of the command position counter.

_8102_get_error_counter:
This function is used to read the value of the position error counter.

_8102_reset_error_counter:
This function is used to clear the position error counter.

_8102_get_general_counter:
This function is used to read the value of the general counter.

_8102_set_general_counter:

This function is used to set the counting source of and change the value of general counter (By default, the source is pulser input).

_8102_get_target_pos:

This function is used to read the value of the target position recorder. The target position recorder is maintained by the 8102 software driver. It records the position to settle down for current running motion.

_8102_reset_target_pos:

This function is used to set new value for the target position recorder. It is necessary to call this function when home return completes, or when a new feedback counter value is set by function _8102_set_position().

_8102_get_res_distance:

This function is used to read the value of the residue distance recorder. The target position recorder is maintained by the 8102 software driver. It records the position to settle down for current running motion.

_8102_set_res_distance:

This function is used to change the value of the residue distance counter

@ **Syntax**

C/C++(Windows 2000/XP)

```
I16 _8102_get_position(I16 AxisNo, F64 *Pos);
I16 _8102_set_position(I16 AxisNo, F64 Pos);
I16 _8102_get_command(I16 AxisNo, I32 *Command);
I16 _8102_set_command(I16 AxisNo, I32 Command);
I16 _8102_get_error_counter(I16 AxisNo, I16 *error);
I16 _8102_reset_error_counter(I16 AxisNo);
I16 _8102_get_general_counter(I16 AxisNo, F64 *CntValue);
I16 _8102_set_general_counter(I16 AxisNo, I16 CntSrc, F64 CntValue);
I16 _8102_get_target_pos(I16 AxisNo, F64 *T_pos);
I16 _8102_reset_target_pos(I16 AxisNo, F64 T_pos);
I16 _8102_get_res_distance(I16 AxisNo, F64 *Res_Distance);
I16 _8102_set_res_distance(I16 AxisNo, F64 Res_Distance);
```

Visual Basic (Windows 2000/XP)

B_8102_get_position(ByVal AxisNo As Integer, Pos As Double) As Integer

B_8102_set_position(ByVal AxisNo As Integer, ByVal Pos As Double) As Integer

B_8102_get_command(ByVal AxisNo As Integer, Cmd As Long) As Integer

B_8102_set_command(ByVal AxisNo As Integer, ByVal Cmd As Long) As Integer

B_8102_get_error_counter(ByVal AxisNo As Integer, ByRef error As Integer) As Integer

B_8102_reset_error_counter(ByVal AxisNo As Integer) As Integer

B_8102_set_general_counter(ByVal AxisNo As Integer, ByVal CntSrc As Integer, ByVal CntValue As Double) As Integer

B_8102_get_general_counter(ByVal AxisNo As Integer, ByRef Pos As Double) As Integer

B_8102_reset_target_pos(ByVal AxisNo As Integer, ByVal Pos As Double) As Integer

B_8102_get_target_pos(ByVal AxisNo As Integer, ByRef Pos As Double) As Integer

B_8102_set_res_distance(ByVal AxisNo As Integer, ByVal Res_Distance As Double) As Integer

B_8102_get_res_distance(ByVal AxisNo As Integer, ByRef Res_Distance As Double) As Integer

@ **Argument**

AxisNo: Axis number of Target Axis.

card_id	Physical axis	AxisNo
0	0	**0**
	1	**1**
1	0	**2**
	1	**3**
2	0	**4**
	1	**…**

Pos, *Pos: Feedback position counter value, (_8102_get/set_position)

range: -134217728~134217727

command, *command: Command position counter value,

range: -134217728~134217727

***error**: Position error counter value,

range: -32768~32767

CntSrc: general counter source

Value	meaning
0	Command pulse
1	EA/EB
2	Pulser input
3	System clock÷2

CntValue, * CntValue: the counter value

T_pos, *T_pos: Target position recorder value,

range: -134217728~134217727

Res_Distance, * Res_Distance: residue distance

6.15 Position Compare and Latch

@ Name
_8102_set_trigger_logic – Set the CMP signal's logic
_8102_set_trigger_comparator – Set the trigger comparator
_8102_set_error_comparator – Set the error comparator
_8102_set_general_comparator – Set the general comparator
_8102_set_latch_source – Set the latch timing for a counter
_8102_set_ltc_logic – Set the logic of LTC signal
_8102_get_latch_data – Get the latch datas from counter

@ Description
_8102_set_trigger_logic:
This function is used to set the logic of CMP single.

_8102_set_error_comparator:

This function is used to set the comparing method and value for the error comparator. When the position error counter's value reaches the comparing value, the 8102 will generate an interrupt to the host PC. Also see section 6.14 "Interrupt control".

_8102_set_general_comparator:

This function is used to set the comparing source counter, comparing method and value for the general comparator. When the comparison conditions are met, there is one of the 4 reactions will be done. The detail setting, see the argument discribtion.

_8102_set_trigger_comparator:

This function is used to set the comparing source counter, comparing method and value for the trigger comparator. When the comparison source counter's value reaches the comparing value, the 8102 will generate a pulse output via CMP and an interrupt (event_int_status, bit 12) will also be sent to host PC.

_8102_set_latch_source:

There are 4 latch triggering source. By using this function, user can choose the event source to latch counters' data.

_8102_set_ltc_logic:

This function is used to set the logic of the latch input.

_8102_get_latch_data:

After the latch signal arrived, the function is used to read the latched value of counters.

@ Syntax

C/C++(Windows 2000/XP)

I16 _8102_set_trigger_logic(I16 AxisNo, I16 Logic);

I16 _8102_set_error_comparator(I16 AxisNo, I16 CmpSrc, I16 CmpMethod, I32 Data);

I16 _8102_set_general_comparator(I16 AxisNo, I16 CmpSrc, I16 CmpMethod,
I16 CmpAction, I32 Data);

I16 _8102_set_trigger_comparator(I16 AxisNo, I16 CmpSrc, I16 CmpMethod,
I32 Data);

I16 _8102_set_latch_source(I16 AxisNo, I16 ltc_src);

16 _8102_set_ltc_logic(I16 AxisNo, I16 ltc_logic);

I16 _8102_get_latch_data(I16 AxisNo, I16 CounterNo, F64 *Pos);

Visual Basic (Windows 2000/XP)

B_8102_set_trigger_logic(ByVal AxisNo As Integer, ByVal Logic As Integer) As Integer

B_8102_set_error_comparator(ByVal AxisNo As Integer, ByVal CmpSrc As Integer, ByVal CmpMethod As Integer, ByVal Data As Long) As Integer

B_8102_set_general_comparator(ByVal AxisNo As Integer, ByVal CmpSrc As Integer, ByVal CmpMethod As Integer, ByVal CmpAction As Integer, ByVal Data As Long) As Integer

B_8102_set_trigger_comparator(ByVal AxisNo As Integer, ByVal CmpSrc As Integer, ByVal CmpMethod As Integer, ByVal Data As Long) As Integer

B_8102_set_latch_source(ByVal AxisNo As Integer, ByVal ltc_src As Integer) As Integer

B_8102_set_ltc_logic(ByVal AxisNo As Integer, ByVal ltc_logic As Integer) As Integer

B_8102_get_latch_data(ByVal AxisNo As Integer, ByVal CounterNo As
Integer, Pos As Double) As Integer

@ **Argument**

AxisNo: Axis number of Target Axis.

Card_id	Physical axis	AxisNo
0	0	**0**
	1	**1**
1	0	**2**
	1	**3**
2	0	**4**
	1	**…**

Logic: logic of comparing trigger

Value	meaning
0	Negative logic
1	Positive logic

CmpSrc: The comparing source counters

Value	meaning
0	Command counter
1	Feedback counter
2	Error counter
3	General counter

CmpMethod: The comparing methods

Value	meaning
1	Data = Source counter (direction independent)
2	Data = Source counter (Count up only)
3	Data = Source counter (Count down only)
4	Data > Source counter
5	Data < Source counter

Data: comparing value

CmpAction:

Value	meaning
0	No action
1	Stop immediatlly
2	Slow down then stop

ltc_src:

Value	meaning
0	LTC pin input
1	ORG pin input

2	general comparator conditions are met
3	trigger comparator conditions are met

ltc_logic: LTC signal operation edge

Value	meaning
0	Negative logic
1	Positive logic

CounterNo: Specified the counter to latch

Value	meaning
0	Command counter
1	Feedback counter
2	Error counter
3	General counter

***Pos**: Latch data

6.16 Continuous motion

@ Name
_8102_set_continuous_move – Enable continuous motion for absolute motion
_8102_check_continuous_buffer – Check if the buffer is empty
_8102_dwell_move – Set a dwell move

@ Description
_8102_set_continuous_move:
This function is necessary before and after continuous motion command sequences

_8102_check_continuous_buffer:
This function is used to detect if the command pre-register is empty or not. Once the command pre-register is empty, users may write the next motion command into it. Otherwise, the new command will overwrite the previous command in the 2nd command pre-register. If the return code is 1 means buffer is full. Otherwise return code is 0, buffer is not full.

_8102_dwell_move:
This function is used to start a dwell move that means the move does not cause real motion for a specific time.
Following example is shown how does the user

```
_8102_set_continuous_move( 2, 1 ); // start continuous move
_8102_start_tr_move( 2, 20000.0, 10.0, 10000.0, 0.1, 0.1);
_8102_dwell_move( 2, 2000); //dwell move for 2 sec.
_8102_start_sr_move( 2, 20000.0, 10.0, 10000.0, 0.1, 0.1, 0, 0 );
_8102_set_continuous_move( 2, 0 ); //end continuous move
```

@ Syntax

C/C++(Windows 2000/XP)

```
I16 _8102_set_continuous_move(I16 AxisNo, I16 conti_logic);
I16 _8102_check_continuous_buffer(I16 AxisNo);
I16 _8102_dwell_move(I16 AxisNo, F64 miniSecond);
```

Visual Basic (Windows 2000/XP)

```
B_8102_set_continuous_move(ByVal AxisNo As Integer, ByVal conti_logic
        As Integer) As Integer
```

B_8102_check_continuous_buffer(ByVal AxisNo As Integer) As Integer

B_8102_dwell_move(ByVal AxisNo As Integer, ByVal miniSecond As Double) As Integer

@ **Argument**

AxisNo: Axis number of Target Axis.

Card_id	Physical axis	AxisNo
0	0	**0**
	1	**1**
1	0	**2**
	1	**3**
2	0	**4**
	1	**…**

conti_logic: continuous motion logic

Value	meaning
0	continuous motion sequence is finished
1	continuous motion sequence is started

miniSecond: Time of dwell move, the unit is in ms

6.17 Multiple Axes Simultaneous Operation

@ Name

_8102_set_tr_move_all – Multi-axis simultaneous operation setup
_8102_set_ta_move_all – Multi-axis simultaneous operation setup
_8102_set_sr_move_all – Multi-axis simultaneous operation setup
_8102_set_sa_move_all – Multi-axis simultaneous operation setup
_8102_start_move_all – Begin a multi-axis trapezoidal profile motion
_8102_stop_move_all – Simultaneously stop Multi-axis motion

@ Description

Theses functions are related to simultaneous operations of multi-axes, even in different cards. The simultaneous multi-axis operation means to start or stop moving specified axes at the same time. The axes moved are specified by the parameter "AxisArray," and the number of axes are defined by parameter "TotalAxes" in _8102_set_tr_move_all().
When properly setup with _8102_set_xx_move_all(), the function _8102_start_move_all() will cause all specified axes to begin a trapezoidal relative movement, and _8102_stop_move_all() will stop them. Both functions guarantee that motion Starting/Stopping on all specified axes are at the same time. Note that it is necessary to make connections according to Section 3.13 if these two functions are needed.
The following code demos how to utilize these functions. This code moves axis 0 and axis 1 to distance 80000.0 and 120000.0 respectively. If we choose velocities and accelerations that are proportional to the ratio of distances, then the axes will arrive at their endpoints at the same time.

[Example]

```
I16 axes[2] = {0, 1};
F64 dist[2] = {80000.0, 120000.0},
F64 str_vel[2] = {0.0, 0.0},
F64 max_vel[2] = {4000.0, 6000.0},
F64 Tacc[2] = {0.1, 0.6},
F64 Tdec[2] = {0.1, 0.6};

_8102_set_tr_move_all(2, axes, dist, str_vel, max_vel, Tacc, Tdec);
_8102_start_move_all(axes[0]);
```

@ Syntax

C/C++(Windows 2000/XP)

```
I16 _8102_set_tr_move_all(I16 TotalAxes, I16 *AxisArray, F64 *DistA, F64
        *StrVelA, F64 *MaxVelA, F64 *TaccA, F64 *TdecA);

I16 _8102_set_ta_move_all(I16 TotalAx, I16 *AxisArray, F64 *PosA, F64
        *StrVelA, F64 *MaxVelA, F64 *TaccA, F64 *TdecA);
```

I16 _8102_set_sr_move_all(I16 TotalAx, I16 *AxisArray, F64 *DistA, F64 *StrVelA, F64 *MaxVelA, F64 *TaccA, F64 *TdecA, F64 *SVaccA, F64 *SVdecA);

I16 _8102_set_sa_move_all(I16 TotalAx, I16 *AxisArray, F64 *PosA, F64 *StrVelA, F64 *MaxVelA, F64 *TaccA, F64 *TdecA, F64 *SVaccA, F64 *SVdecA);

I16 _8102_start_move_all(I16 FirstAxisNo);

I16 _8102_stop_move_all(I16 FirstAxisNo);

Visual Basic (Windows 2000/XP)

B_8102_set_tr_move_all(ByVal TotalAxes As Integer, ByRef AxisArray As Integer, ByRef DistA As Double, ByRef StrVelA As Double, ByRef MaxVelA As Double, ByRef TaccA As Double, ByRef TdecA As Double) As Integer

B_8102_set_sa_move_all(ByVal TotalAxes As Integer, ByRef AxisArray As Integer, ByRef PosA As Double, ByRef StrVelA As Double, ByRef MaxVelA As Double, ByRef TaccA As Double, ByRef TdecA As Double, ByRef SVaccA As Double, ByRef SVdecA As Double) As Integer

B_8102_set_ta_move_all(ByVal TotalAxes As Integer, ByRef AxisArray As Integer, ByRef PosA As Double, ByRef StrVelA As Double, ByRef MaxVelA As Double, ByRef TaccA As Double, ByRef TdecA As Double) As Integer

B_8102_set_sr_move_all(ByVal TotalAxes As Integer, ByRef AxisArray As Integer, ByRef DistA As Double, ByRef StrVelA As Double, ByRef MaxVelA As Double, ByRef TaccA As Double, ByRef TdecA As Double, ByRef SVaccA As Double, ByRef SVdecA As Double) As Integer

B_8102_start_move_all(ByVal FirstAxisNo As Integer) As Integer

B_8102_stop_move_all(ByVal FirstAxisNo As Integer) As Integer

@ **Argument**

TotalAxes: Number of axes for simultaneous motion
***AxisArray**: Specified axes number array designated to move.
***DistA**: Specified distance array in units of pulse
***StrVelA**: Starting velocity array in units of pulse per second
***MaxVelA**: Maximum velocity array in units of pulse per second
***TaccA**: Acceleration time array in units of seconds
***TdecA**: Deceleration time array in units of seconds
***PosA**: Specified position array in units of pulse

***SvaccA**: Specified velocity interval array in which S-curve acceleration is performed.
***SvdecA**: Specified velocity interval array in which S-curve deceleration is performed.
FirstAxisNo: The first element in AxisArray.

6.18 General-purposed DIO

@ Name
_8102_set_gpio_output – Set digital output
_8102_get_gpio_output – Get digital output
_8102_get_gpio_input – Get digital input

@ Description
_8102_set_gpio_output:
The PCI-8102 has 2 digital output channels. By this function, user could control the digital outputs.
_8102_get_gpio_output:
This function is used to get the digitl output status.
_8102_get_gpio_input:
PCI-8102 has 4 digital input channels. By this function, user can get the digital input status.

@ Syntax

C/C++(Windows 2000/XP)

I16 _8102_set_gpio_output(I16 card_id, U16 do_value);

I16 _8102_get_gpio_output(I16 card_id, U16 *do_status);

I16 _8102_get_gpio_input(I16 card_id, U16 *di_status);

Visual Basic (Windows 2000/XP)

B_8102_set_gpio_output Lib "8102.dll" Alias "_8102_set_gpio_output" (ByVal card_id As Integer, ByVal do_value As Integer) As Integer

B_8102_get_gpio_output Lib "8102.dll" Alias "_8102_get_gpio_output" (ByVal card_id As Integer, do_status As Integer) As Integer

B_8102_get_gpio_input Lib "8102.dll" Alias "_8102_get_gpio_input" (ByVal card_id As Integer, di_status As Integer) As Integer

@ Argument
card_id: Specify the PCI-8102 card index. The card_id could be decided by DIP switch (SW1) or depend on slot sequence.Please refer to _8102_initial().
do_value: Unsigned 16 bits value. Bit 0: D_out0, Bit 1: D_out1
***do_status**: Digital output status. Bit 0: D_out0, Bit 1: D_out1
***di_status**: Digital input status, Bit0~3: D_in0~3

6.19 Soft Limit

@ Name
_8102_disable_soft_limit – Disable soft limit function
_8102_enable_soft_limit – Enable soft limit function
_8102_set_soft_limit – Set soft limit

@ Description
_8102_disable_soft_limit:

This function is used to disable the soft limit function.

_8102_enable_soft_limit:

This function is used to enable the soft limit function. Once enabled, the action of soft limit will be exactly the same as physical limit.

_8102_set_soft_limit:

This function is used to set the soft limit value.

@ Syntax

C/C++(Windows 2000/XP)

I16 _8102_disable_soft_limit(I16 AxisNo);

I16 _8102_enable_soft_limit(I16 AxisNo, I16 Action);

I16 _8102_set_soft_limit(I16 AxisNo, I32 Plus_Limit, I32 Neg_Limit);

Visual Basic (Windows 2000/XP)

B_8102_disable_soft_limit(ByVal AxisNo As Integer) As Integer

B_8102_enable_soft_limit(ByVal AxisNo As Integer, ByVal Action As Integer) As Integer

B_8102_set_soft_limit(ByVal AxisNo As Integer, ByVal Plus_Limit As Long, ByVal Neg_Limit As Long) As Integer

@ Argument
AxisNo: Axis number of Target Axis.

card_id	Physical axis	AxisNo
0	0	**0**
	1	**1**
1	0	**2**

	1	**3**
2	0	**4**
	1	**…**

Action: The reacting method of soft limit

Value	meaning
0	INT only
1	Immediately stop
2	slow down then stop

Plus_Limit: Soft limit value, positive direction
Neg_Limit: Soft limit value, negative direction

6.20 Backlash Compensation / Vibration Suppression

@ Name

_8102_backlash_comp – Set backlash corrective pulse for compensation
_8102_suppress_vibration – Set vibration suppressing timing
_8102_set_fa_speed – Set the FA speed

@ Description

_8102_backlash_comp:

Whenever direction change occurs, the 8102 outputs backlash corrective pulses before sending commands. This function is used to set the compensation pulse numbers.

_8102_suppress_vibration:

This function is used to suppress vibration of mechanical systems by outputting a single pulse for negative direction and the single pulse for positive direction right after completion of command movement.

_8102_set_fa_speed:

This function is used to specify the low speed for backlash correction or slip correction. It also used as a reverse low speed for home return operation.

@ Syntax

C/C++(Windows 2000/XP)

```
I16 _8102_backlash_comp(I16 AxisNo, I16 CompPulse, I16 Mode);

I16 _8102_suppress_vibration(I16 AxisNo, U16 ReverseTime,
          U16 ForwardTime);

I16 _8102_set_fa_speed(I16 AxisNo, F64 FA_Speed);
```

Visual Basic (Windows 2000/XP)

```
B_8102_backlash_comp Lib "8102.dll" Alias "_8102_backlash_comp"
        (ByVal AxisNo As Integer, ByVal CompPulse As Integer, ByVal
        Mode As Integer) As Integer

B_8102_suppress_vibration(ByVal AxisNo As Integer, ByVal ReverseTime
        As Integer, ByVal ForwardTime As Integer) As Integer

B_8102_set_fa_speed(ByVal AxisNo As Integer, ByVal FA_Speed As
        Double) As Integer
```

@ **Argument**

AxisNo: Axis number of Target Axis.

Card_id	Physical axis	AxisNo
0	0	**0**
	1	**1**
1	0	**2**
	1	**3**
2	0	**4**
	1	**…**

CompPulse: Specified number of corrective pulses, 12 bit

Mode:

Value	meaning
0	Turns off
1	enable backlash compensation
2	Slip correction

ReverseTime: Specified Reverse Time, 0 ~ 65535, unit 1.6 us
ForwardTime: Specified Forward Time, 0 ~ 65535, unit 1.6 us
FA_Speed: fa speed (unit: pulse/sec)

6.21 Speed Profile Calculation

@ Name

_8102_get_tr_move_profile – Get the relative trapezoidal speed profile
_8102_get_ta_move_profile – Get the absolute trapezoidal speed profile
_8102_get_sr_move_profile – Get the relative S-curve speed profile
_8102_get_sa_move_profile – Get the absolute S-curve speed profile

@ Description

_8102_get_tr_move_profile:

This function is used to get the relative trapezoidal speed profiles. By this function, user can get the actual speed profile before running.

_8102_get_ta_move_profile:

This function is used to get the absolute trapezoidal speed profiles. By this function, user can get the actual speed profile before running.

_8102_get_sr_move_profile:

This function is used to get the relative S-curve speed profiles. By this function, user can get the actual speed profile before running.

_8102_get_sa_move_profile:

This function is used to get the absolute S-curve speed profiles. By this function user can get the actual speed profile before running.

@ Syntax

C/C++(Windows 2000/XP)

```
I16 _8102_get_tr_move_profile(I16 AxisNo, F64 Dist, F64 StrVel, F64
       MaxVel, F64 Tacc, F64 Tdec, F64 *pStrVel, F64 *pMaxVel, F64
       *pTacc, F64 *pTdec, F64 *pTconst );

I16 _8102_get_ta_move_profile(I16 AxisNo, F64 Pos, F64 StrVel, F64
       MaxVel, F64 Tacc, F64 Tdec, F64 *pStrVel, F64 *pMaxVel, F64
       *pTacc, F64 *pTdec, F64 *pTconst );

I16 _8102_get_sr_move_profile(I16 AxisNo, F64 Dist, F64 StrVel, F64
       MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec,F64 *pStrVel,
       F64 *pMaxVel, F64 *pTacc, F64 *pTdec, F64 *pSVacc, F64 *pSVdec,
       F64 *pTconst);

I16 _8102_get_sa_move_profile(I16 AxisNo, F64 Pos, F64 StrVel, F64
       MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec,F64 *pStrVel,
       F64 *pMaxVel, F64 *pTacc, F64 *pTdec, F64 *pSVacc, F64 *pSVdec,
       F64 *pTconst);
```

Visual Basic (Windows 2000/XP)

```
B_8102_get_tr_move_profile(ByVal AxisNo As Integer, ByVal Dist As
        Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal
        Tacc As Double, ByVal Tdec As Double, ByRef pStrVel As Double,
        ByRef pMaxVel As Double, ByRef pTacc As Double, ByRef pTdec
        As Double, ByRef pTconst As Double) As Integer
B_8102_get_ta_move_profile(ByVal AxisNo As Integer, ByVal Pos As
        Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal
        Tacc As Double, ByVal Tdec As Double, ByRef pStrVel As Double,
        ByRef pMaxVel As Double, ByRef pTacc As Double, ByRef pTdec
        As Double, ByRef pTconst As Double) As Integer
B_8102_get_sr_move_profile(ByVal AxisNo As Integer, ByVal Dist As
        Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal
        Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double,
        ByVal SVdec As Double, ByRef pStrVel As Double, ByRef pMaxVel
        As Double, ByRef pTacc As Double, ByRef pTdec As Double,
        ByRef pSVacc As Double, ByRef pSVdec As Double, ByRef
        pTconst As Double) As Integer
B_8102_get_sa_move_profile(ByVal AxisNo As Integer, ByVal Pos As
        Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal
        Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double,
        ByVal SVdec As Double, ByRef pStrVel As Double, ByRef pMaxVel
        As Double, ByRef pTacc As Double, ByRef pTdec As Double,
        ByRef pSVacc As Double, ByRef pSVdec As Double, ByRef
        pTconst As Double) As Integer
```

@ **Argument**

AxisNo: Axis number of Target Axis.

card_id	Physical axis	AxisNo
0	0	**0**
	1	**1**
1	0	**2**
	1	**3**
2	0	**4**
	1	**…**

Dist: Specified relative distance (unit: pulse)
Pos: Specified absolute position (unit: pulse)
StrVel: Starting velocity (unit: pulse/sec)
MaxVel: Maximum velocity (unit: pulse/sec)
Tacc: time for acceleration (unit: sec)
Tdec: time for deceleration (unit: sec)

SVacc: S-curve region during acceleration (unit: pulse/sec)
Note: SVacc = 0, for pure S-Curve. For more details, section 4.2.4
SVdec: S-curve region during deceleration (unit: pulse/sec)
Note: SVdec = 0, for pure S-Curve. For more details, section 4.2.4
***pStrVel**: Starting velocity by calculation
***pMaxVel**: Maximum velocity by calculation
***pTacc**: Acceleration time by calculation
***pTdec**: Deceleration time by calculation
***pSVacc**: S-curve region during acceleration by calculation
***pSVdec**: S-curve region during deceleration by calculation
***pTconst**: constant speed time(maximum speed)

6.22 Return Code

The return error code is defined in “8102_err.h”. The meaning is discribed in following table.

Code	Meaning
0	No error
-10000	Error Card number
-10001	Error operation system version
-10002	Error card’s ID conflict
-10200	Error other process exist
-10201	Error card not found
-10202	Error Open driver failed
-10203	Error ID mapping failed
-10204	Error trigger channel
-10205	Error trigger type
-10206	Error event already enabled
-10207	Error event not enable yet
-10208	Error on board FIFO full
-10209	Error unknown command type
-10210	Error unknow chip type
-10211	Error card not initial
-10212	Error position out of range
-10213	Error motion busy
-10214	Error speed error
-10215	Error slow down point
-10216	Error axis range error
-10217	Error compare parameter error
-10218	Error compare method
-10219	Error axis already stop
-10220	Error axis INT wait failed
-10221	Error user code write failed
-10222	Error array size exceed
-10223	Error factor number
-10224	Error enable range
-10225	Error auto accelerate time
-10226	Error dwell time
-10227	Error dwell distance
-10228	Error new position

-10229	Error motion not in running
-10230	Error velocity change time
-10231	Error speed target
-10232	Error velocity percent
-10233	Error postion change backward
-10234	Error counter number

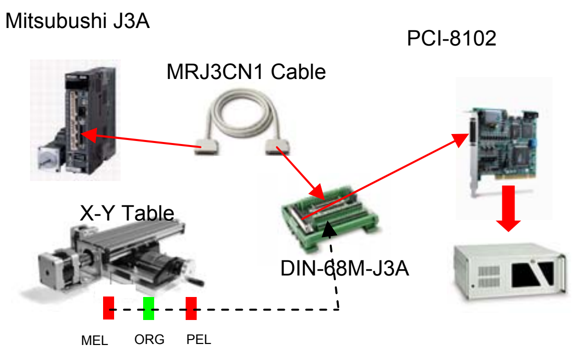
7

Connection Example

This chapter shows some connection examples between the PCI-8102 and servo drivers and stepping drivers.

7.1 General Description of Wiring

Main connection between the PCI-8102 and the pulse input servo driver or stepping driver. The following figure illustrates how to integrate the PCI-8102 and DIN-68M-J3A

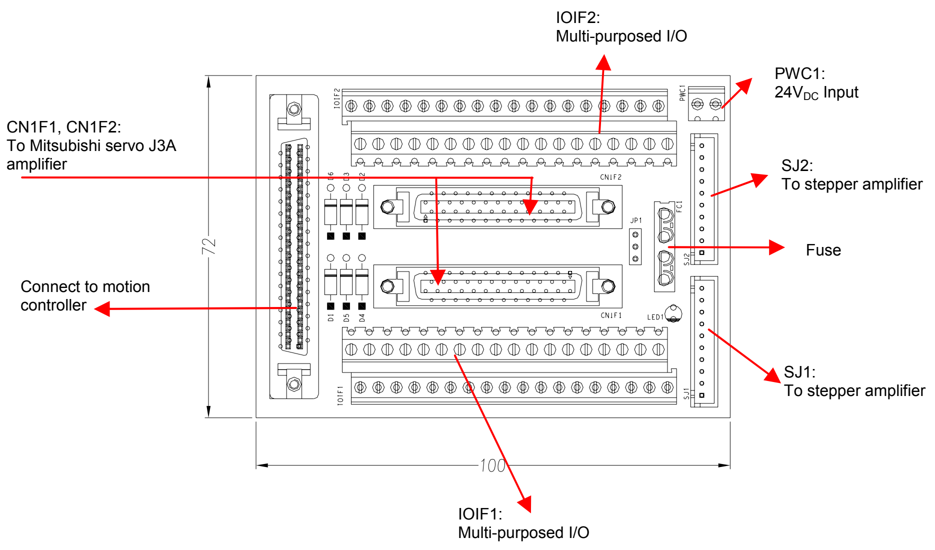


7.2 Wiring with DIN-68M-J3A

Notice

The DIN-68M-J3A is used for wiring between **Mitsubishi J3A** series servo drivers / stepper with pulse trains input driver and **ADLINK** c**PCI-8168, PCI-8102** motion controller card ONLY. Never use this terminal board with any other servo driver or cards.

P/N: 50-xxxxx-xxx



PWC1:
$24V_{DC}$ Input
SJ2:
To stepper amplifier
Fuse
SJ1:
To stepper amplifier

Note:

1. **Servo & Stepper:** The DIN-68M-J3A provides 2 connection methods for each axis: CN1F1 & CN1F2 connectors for Mitsubishi J3A series servo drivers, and a SJ connector for stepping drivers. DO NOT use both connectors at the same time.

2. **CN1F1 or CN1F2 cables:** One pin-to-pin 50-PIN cables are required for connection between the CF1F1(CN1F2) and the Mitsubishi J3A driver. It is available from ADLINK, or users may contact the local dealer or distributor to get cable information.

3. **JP1:** Fuse selection. Pin 1-2 short means the board is protected by fuse. Pin 2-3 short menas the board is not protected by fuse.

4. **Fuse:** The fuse protects from demaging internal circuit. This guse can sustain 2A current. Users can buy it in local area when the fuse is burnt out.

● PIN Assignments:

CN1F1/CN1F2 (Mitsubishi AC Servo Driver J3A)

No.	Name	I/O	Function	No.	Name	I/O	Function
1	P15R	--	15VDC power supply	2	VLA	O	Analog speed limit
3	EXGND	--	Ext. power ground	4	EA+	I	Encoder A-phase (+)
5	EA-	I	Encoder A-phase (-)	6	EB+	I	Encoder B-phase (+)
7	EB-	I	Encoder B-phase (-)	8	EZ+	I	Encoder Z-phase (+)
9	EZ-	I	Encoder Z-phase (-)	10	OUT+	O	Pulse signal (+)
11	OUT-	O	Pulse signal (-)	12	(Empty)	N.C.	
13	(Empty)	N.C.		14	(Empty)	N.C.	
15	SERVO ON	O	Servo On signal	16	SP2	O	Speed selection 2
17	ABSM	O	ABS transfer	18	ABSR	O	ABS request
19	RES	O	Reset	20	EX+24V	I	Ext. power supply, +24V
21	EX+24V	I	Ext. power supply, +24V	22	ABSB0	I	ABS transmission data bit 0
23	ZSP	I	Zero speed	24	INP	I	Servo In Position signal
25	TLC	I	Limiting torque	26	(Empty)	N.C.	
27	TC	O	Analog torque command	28	EXGND	--	Ext. power ground
29	(Empty)	N.C.		30	(Empty)	N.C.	
31	(Empty)	N.C.		32	(Empty)	N.C.	
33	(Empty)	N.C.		34	EXGND	--	Ext. power ground
35	DIR+	O	Direction Signal (+)	36	DIR-	O	Direction Signal (-)
37	(Empty)	N.C.		38	(Empty)	N.C.	
39	(Empty)	N.C.		40	(Empty)	N.C.	
41	ERC	O	Dev. ctr, clr. Signal	42	EMG	I	External EMG Signal
	SP1	O	Speed selection 1				
43	EXGND	--	Ext. power ground	44	EXGND	--	Ext. power ground
45	LOP	O	Control change	46	EXGND	--	Ext. power ground
47	EXGND	--	Ext. power ground	48	ALM	I	Servo Alarm
49	RDY	I	Servo Ready	50	(Empty)	N.C.	

IOIF1/IOIF2 (External Motion Input Signal Interface)

No.	Name	I/O	Function	No.	Name	I/O	Function
1	EXGND	--	Ext. power ground	19	EX+24V	I	Ext. power supply, +24V
2	EXGND	--	Ext. power ground	20	EX+24V	I	Ext. power supply, +24V
3	EXGND	--	Ext. power ground	21	EX+24V	I	Ext. power supply, +24V
4	EXGND	--	Ext. power ground	22	EX+24V	I	Ext. power supply, +24V
5	EXGND	--	Ext. power ground	23	EX+24V	I	Ext. power supply, +24V
6	SD	I	Slow Down signal	24	MEL	I	Negative Limit (-)
	EXGND	--	Ext. power ground				
7	CMP	O	Compare Trigger Output	25	PEL	I	Positive Limit (+)
	EXGND	--	Ext. power ground				
8	RES	O	Reset	26	ORG	I	Origin signal
9	DOUT	O	Digital Output	27	DO_COM	I	Digital Output COM
10	HSOUT	O	High Speed DO	28	HO_COM	I	Common of HSOUT
	AO		Analog Output		AGND	--	Analog Ground
11	DIN	I	Digital Input	29	DI_COM	I	Digital Input COM
12	EMG	I	External EMG Signal	30	HSIN	I	High Speed DI
					AIN		Analog Input
13	ABSB0	I	ABS transmission data bit 0	31	P15R	--	15VDC power supply

14	ABSM	O	ABS transfer	32	ERC	O	Dev. ctr, clr. Signal
					SP1	O	Speed selection 1
15	VLA	O	Analog speed limit	33	SP2	O	Speed selection 2
16	ABSR	O	ABS request	34	TC	O	Analog torque command
17	ZSP	I	Zero speed	35	TLC	I	Limiting torque
18	EXGND	--	Ext. power ground	36	LOP	O	Control change

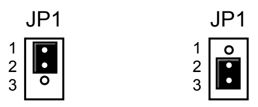
SJ1/SJ2 (Stepping Motor Control Interface)

No.	Name	I/O	Function
1	OUT+	O	Pulse Signal (+)
2	OUT-	O	Pulse Signal (-)
3	DIR+	O	Direction Signal (+)
4	DIR-	O	Direction Signal (-)
5	EZ+	I	Index Signal
6	ALM	I	Servo Alarm
7	+5V	O	Voltage output
8	Servo ON	O	Servo On
9	+5V	O	Voltage output
10	EXGND	--	Ext. power ground

PWC1 (External +24VDC Input Connector)

No.	Name	I/O	Function
1	EXGND	--	External Power Supply Ground
2	EX+24V	I	External Power Supply Input (+24V DC ± 5%)

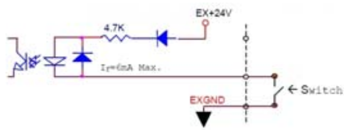
JP1 (Setting for fuse)



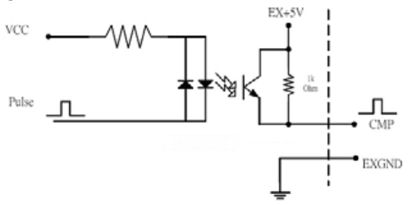
With fuse   Without fuse(default)

● Signal Connections of Interface Circuit

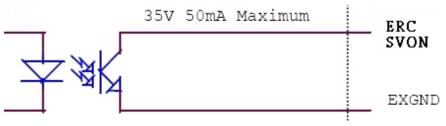
1. PEL、MEL、ORG 、INP、ALM、RDY



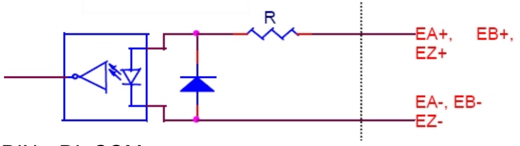
2. CMP



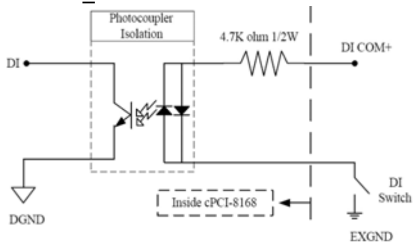
3. ERC、SVON



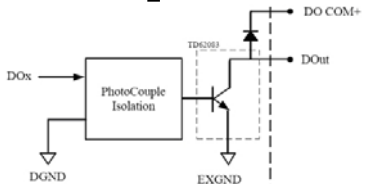
4. EA+、EB+、EZ+、EA-、EB-、EZ-



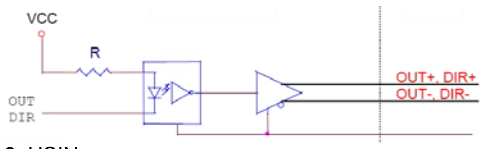
5. DIN、DI_COM



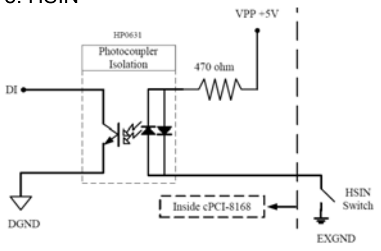
6. DOUT、DO_COM



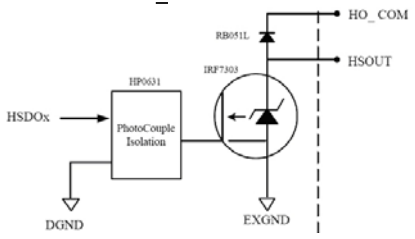
7. DIR+、OUT +、DIR-、OUT-



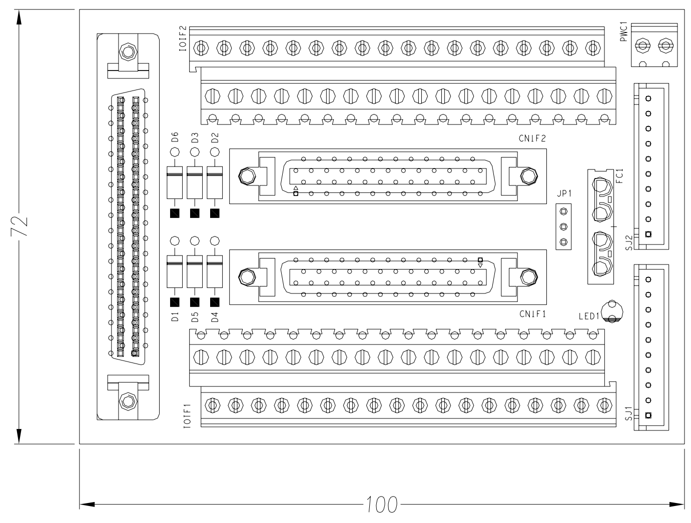
8. HSIN



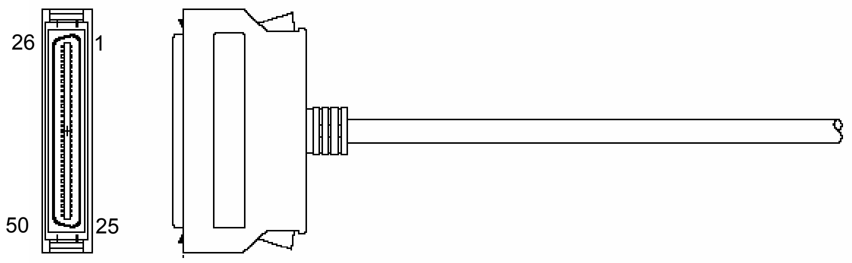
9. HSOUT、HO_COM

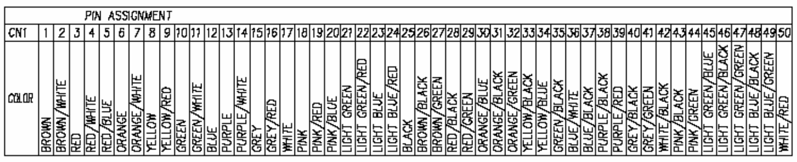


Mechanical Dimensions:



Appendix A: Color code of Mitsubishi servo J3A cable (MRJ3CN1 xM-OPEN)





PIN ASSIGNMENT										
CN1	1	2	3	4	5	6	7	8	9	10
COLOR	BROWN	BROWN/WHITE	RED	RED/WHITE	RED/BLUE	ORANGE	ORANGE/WHITE	YELLOW	YELLOW/RED	GREEN

CN1	11	12	13	14	15	16	17	18	19	20
COLOR	GREEN/WHITE	BLUE	PURPLE	PURPLE/WHITE	GREY	GREY/RED	WHITE	PINK	PINK/RED	PINK/BLUE

CN1	21	22	23	24	25	26	27	28	29	30
COLOR	LIGHT GREEN	LIGHT GREEN/RED	LIGHT BLUE	LIGHT BLUE/RED	BLACK	BROWN/BLACK	BROWN/GREEN	RED/BLACK	RED/GREEN	ORANGE/BLUE

CN1	31	32	33	34	35	36	37	38	39	40
COLOR	ORANGE/BLACK	ORANGE/GREEN	YELLOW/BLACK	YELLOW/BLUE	GREEN/BLACK	BLUE/WHITE	BLUE/BLACK	PURPLE/BLACK	PURPLE/RED	GREY/BLACK

CN1	41	42	43	44	45	46	47	48	49	50
COLOR	GREY/GREEN	WHITE/BLACK	PINK/BLACK	PINK/GREEN	LIGHT GREEN/BLUE	LIGHT GREEN/BLACK	LIGHT GREEN/GREEN	LIGHT BLUE/BLACK	LIGHT BLUE/GREEN	WHITE/RED

Warranty Policy

Thank you for choosing ADLINK. To understand your rights and enjoy all the after-sales services we offer, please read the following carefully:

1. Before using ADLINK's products please read the user manual and follow the instructions exactly.
2. When sending in damaged products for repair, please attach an RMA application form.
3. All ADLINK products come with a two-year guarantee, repaired free of charge.
   - The warranty period starts from the product's shipment date from ADLINK's factory.
   - Peripherals and third-party products not manufactured by ADLINK will be covered by the original manufacturers' warranty.
   - End users requiring maintenance services should contact their local dealers. Local warranty conditions will depend on local dealers.
4. This warranty will not cover repair costs due to:
   a. Damage caused by not following instructions.
   b. Damage caused by carelessness on the users' part during product transportation.
   c. Damage caused by fire, earthquakes, floods, lightening, pollution, other acts of God, and/or incorrect usage of voltage transformers.
   d. Damage caused by unsuitable storage environments (i.e. high temperatures, high humidity, or volatile chemicals.
   e. Damage caused by leakage of battery fluid.
   f. Damage from improper repair by unauthorized technicians.
   g. Products with altered and/or damaged serial numbers.
   h. Other categories not protected under our guarantees.
5. Customers are responsible for shipping costs to transport damaged products to our company or sales office.
6. To ensure the speed and quality of product repair, please download a RMA application form from our company website: www.adlinktech.com. Damaged products with attached RMA forms receive priority.

For further questions, please contact our FAE staff.

ADLINK: service@adlinktech.com