ADLINK
TECHNOLOGY INC.

PCI-8174
DSP-based
4-Axis Servo / Stepper
Motion Control Card
User's Manual

Manual Rev.	2.00
Revision Date:	April 22, 2007
Part No:	50-11148-1000

Recycled Paper

Advance Technologies; Automate the World.

Getting Service from ADLINK

Customer Satisfaction is top priority for ADLINK Technology Inc. Please contact us should you require any service or assistance.

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 this completed service form for prompt and satisfactory service.

Company Information	
Company/Organization	
Contact Person	
E-mail Address	
Address	
Country	
TEL	FAX:
Web Site	
Product Information	
Product Model	
Environment	OS: M/B: CPU: Chipset: BIOS:

Please give a detailed description of the problem(s):

Table of Contents

List of Tables

List of Figures

1 Introduction

The PCI-8174 is an advanced DSP-based 4-axis motion control card with a PCI interface. Unlike other dummy pulse motion controllers, the PCI-8174 provides DSP kernel for high level machine makers to integrate time-critical motion procedure into a motion controller and eliminates the timing issue in traditional PC-based controllers.

For high-end systems, timing is very important for high throughput demands. Using traditional polling and event-driven methods, all production procedures must be programmed on a PC. It contents many “check” and “do” procedures, wasting CPU time and missing “checks” if the CPU is busy. Even it is checked, the “do” action will be delayed when CPU is busy, which can result in serious consequences.

The PCI-8174 was designed to improve timing issues in traditional PC-based controllers. It reserves the flexibility of PC-based controller for open architecture and enhances the time-critical application capability by a built-in DSP. A real-time OS is not necessary either.

The PCI-8174 unique by different DSP kernels. One DSP kernel contains one specific feature and can be used for one specific application field or for one specific customer. Loading a different DSP kernel gives the PCI-8174 different features. The DSP kernel could also be customized to help speed up and stabilize systems.

The PCI-8174 has various terminal boards for connecting to many kinds of AC servos. The pin definitions are the same as ADLINK 4-axis stepper/servo motion controller, such as the PCI-8134 or PCI-8164. If you currently use such ADLINK product, simply transfer your old design to new one. The PCI-8174 also provides the same traditional motion functions to migrate from old 4-axis board to new PCI-8174.

Figure 1-1 shows the functional block diagram of the PCI-8174 card. A DSP as a main chip. Around the DSP are the major motion ASIC, PCL6045, and some necessary components for DSP, such as SDRAM and DPRAM.

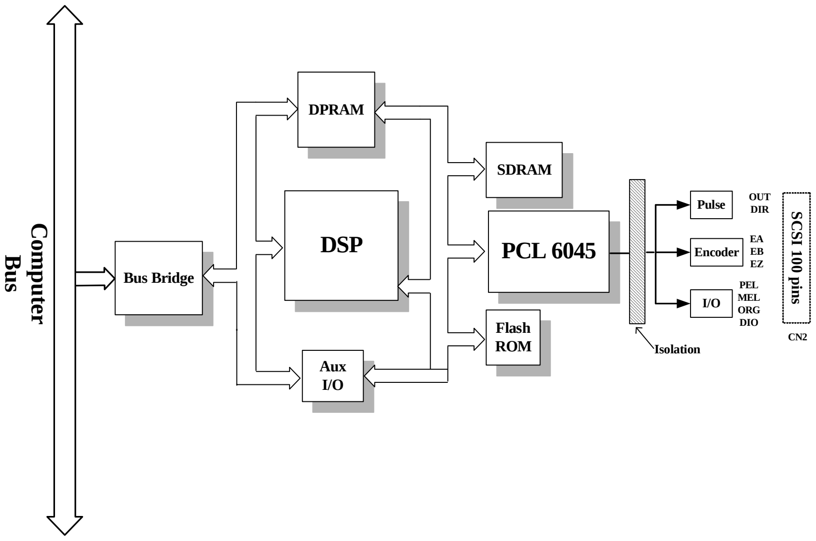


Figure 1-1: PCI-8174 Block Diagram

MotionCreatorPro is a Windows-based application development software package included with the PCI-8174. 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-8174.

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

Figure 1-2 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.

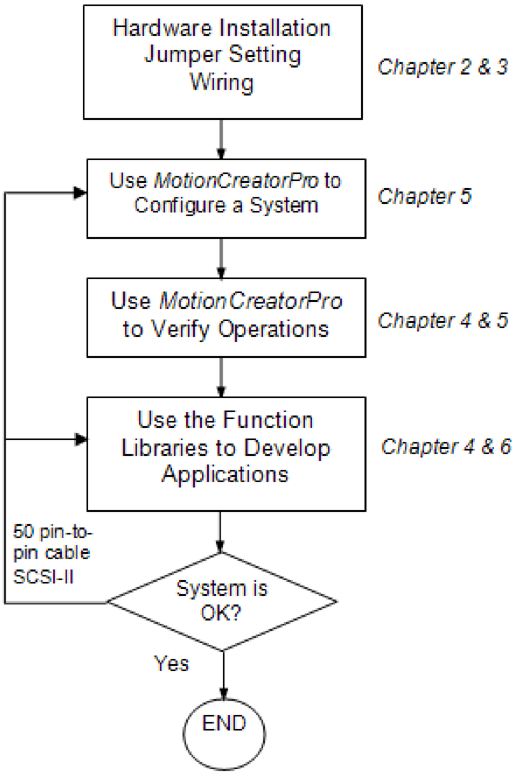

Hardware Installation
Jumper Setting
Wiring
Chapter 2 & 3
Use *MotionCreatorPro* to Configure a System
Chapter 5
Use *MotionCreatorPro* to Verify Operations
Chapter 4 & 5
Use the Function Libraries to Develop Applications
Chapter 4 & 6
50 pin-to-pin cable SCSI-II
System is OK?
Yes
END

Figure 1-2: Flow chart for building an application

1.1 Features

The following list summarizes the main features of the PCI-8174 motion control system.

- 32-bit/33Mhz PCI bus (Universal type for both 3.3V and 5V)
- On board 200Mhz DSP TI TMS320C6711D with 1200 MFLOPS
- Time-critical specific DSP kernel for options
- 4 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, AB phase
- Pulse input options: CW/CCW, AB phase x1,x2,x4
- Maximum pulse input frequency of 3.2Mhz in CW/CCW or AB phase X1 mode. (AB phase x4 could reach 6.5Mhz)
- Programmable acceleration and deceleration time for all modes
- Trapezoidal and S-curve velocity profiles for all modes
- 2 to 4 axes linear interpolation
- 2 axes circular interpolation and 3 axes helical 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
- 4-axis high speed position latch input
- 4-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-8174 cards operation in one system
- Compact PCB design
- Includes MotionCreatorPro, a Microsoft Windows-based application development software
- PCI-8174 libraries and utilities for Windows 2000/XP/Vista.

1.2 Specifications

Applicable Motors:

- Stepping motors
- AC or DC servomotors with pulse train input servo drivers

Performance:

- Number of controllable axes: 4
- Maximum pulse output frequency: 6.55MPPS, linear, trapezoidal, or S-Curve velocity profile drive
- Internal reference clock: 19.66MHz
- 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 Signals:

- 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/home signal input pins: ±EL, ORG
- Composite pins: DI / LTC (Latch) / SD (Slow-down) / PCS (Position Change Signal) / CLR (Clear) / EMG (Emergency Input)
- Servomotor interface I/O pins: INP, ALM, and ERC
- General-purposed digital output pin: SVON, DO
- General-purposed digital input pin: RDY, GDI
- Pulse signal input pin: PA and PB (with Isolation)
- Simultaneous Start/Stop signal: STA and STP

General Specifications

- Connectors: 100-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%, 300mA, max

PCI-8174 Dimensions (PCB size):

- 185mm(L) X 100 mm(W)

1.3 Supported Software

1.3.1 Programming Library

Windows 2000/XP/Vista DLLs are provided for the PCI-8174. 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

ADLINK provides the servo & steppers use terminal board for easy connection. For steppers, we provide DIN-100S which is pin-to-pin terminal board. For servo users, ADLINK offers DIN-814M, DIN-814M-J3A, DIN-814Y and DIN-814P-A4. The suitable servos are listed as follows:

Mitsubishi J2 Super	DIN-814M
Mitsubishi J3A	DIN-814M-J3A
Yaskawa Sigma II	DIN-814Y
Panasonic MINAS A4	DIN-814P-A4

2 Installation

This chapter describes how to install PCI-8174. 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-8174: advanced 4-axis Servo / Stepper Motion Control Card
- ADLINK All-in-one Compact Disc

Note: The terminal board is an optional accessory and would not be included in PCI-8174 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-8174 Outline Drawing

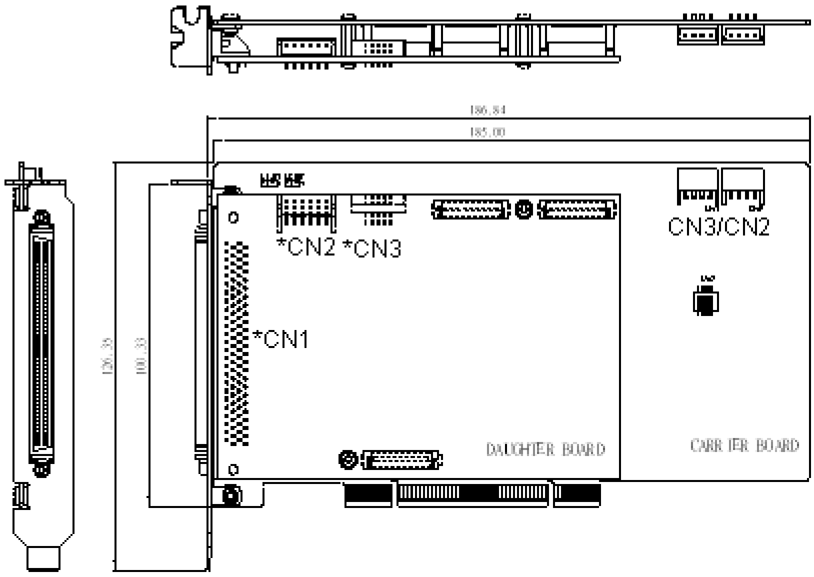


Figure 2-1: PCB Layout of the PCI-8174

- CN2/CN3: DSP Synchronous Signal Connector
- *CN1: Input / Output Signal Connector (100-pin)
- *CN2: Simultaneous start / stop signal input/output
- *CN3: Manual Pulsar
- SW1: DIP switch for card index selection (0-15)

Note: '*' means which connector on Daughter board.

2.3 PCI-8174 Hardware Installation

2.3.1 Hardware configuration

The PCI-8174 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

Some computer system may have both PCI and ISA slots. Do not force the PCI card into a PC/AT slot. The PCI-8174 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 the 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-8174, 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 Troubleshooting4:

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

1. Autorun the ADLINK All-In-One CD. Choose Driver Installation -> Motion Control -> PCI-8174.
2. Follow the procedures of the installer.
3. After setup installation is completed, restart windows.

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

2.5 *CN1 Pin Assignment: Main Connector

*CN1 is the major connector for the motion control I/O signals.

No.	Name	I/O	Function	No.	Name	I/O	Function
1	VDD	O	+5V Power Supply Output	51	VDD	O	+5V Power Supply Output
2	EXGND	--	Ext. Power Ground	52	EXGND	--	Ext. Power Ground
3	OUT0+	O	Pulse Signal (+)	53	OUT2+	O	Pulse Signal (+)
4	OUT0-	O	Pulse Signal (-)	54	OUT2-	O	Pulse Signal (-)
5	DIR0+	O	Dir. Signal (+)	55	DIR2+	O	Dir. Signal (+)
6	DIR0-	O	Dir. Signal (-)	56	DIR2-	O	Dir. Signal (-)
7	SVON0	O	Servo On/Off	57	SVON2	O	Servo On/Off
8	ERC0	O	Dev. ctr, clr. signal	58	ERC2	O	Dev. ctr, clr. signal
9	ALM0	I	Alarm signal	59	ALM2	I	Alarm signal
10	INP0	I	In-position signal	60	INP2	I	In-position signal
11	RDY0	I	Multi-purpose Input signal	61	RDY2	I	Multi-purpose Input signal
12	EXGND	--	Ext. Power Ground	62	EXGND	--	Ext. Power Ground
13	EA0+	I	Encoder A-phase (+)	63	EA2+	I	Encoder A-phase (+)
14	EA0-	I	Encoder A-phase (-)	64	EA2-	I	Encoder A-phase (-)
15	EB0+	I	Encoder B-phase (+)	65	EB2+	I	Encoder B-phase (+)
16	EB0-	I	Encoder B-phase (-)	66	EB2-	I	Encoder B-phase (-)
17	EZ0+	I	Encoder Z-phase (+)	67	EZ2+	I	Encoder Z-phase (+)
18	EZ0-	I	Encoder Z-phase (-)	68	EZ2-	I	Encoder Z-phase (-)
19	VDD	O	+5V Power Supply Output	69	VDD	O	+5V Power Supply Output
20	EXGND	--	Ext. Power Ground	70	EXGND	--	Ext. Power Ground
21	OUT1+	O	Pulse Signal (+)	71	OUT3+	O	Pulse Signal (+)
22	OUT1-	O	Pulse Signal (-)	72	OUT3-	O	Pulse Signal (-)
23	DIR1+	O	Dir. Signal (+)	73	DIR3+	O	Dir. Signal (+)
24	DIR1-	O	Dir. Signal (-)	74	DIR3-	O	Dir. Signal (-)
25	SVON1	O	Servo On/Off	75	SVON3	O	Servo On/Off
26	ERC1	O	Dev. ctr, clr. signal	76	ERC3	O	Dev. ctr, clr. signal
27	ALM1	I	Alarm signal	77	ALM3	I	Alarm signal
28	IN*CN1	I	In-position signal	78	INP3	I	In-position signal
29	RDY1	I	Multi-purpose Input signal	79	RDY3	I	Multi-purpose Input signal
30	EXGND	--	Ext. Power Ground	80	EXGND	--	Ext. Power Ground
31	EA1+	I	Encoder A-phase (+)	81	EA3+	I	Encoder A-phase (+)
32	EA1-	I	Encoder A-phase (-)	82	EA3-	I	Encoder A-phase (-)
33	EB1+	I	Encoder B-phase (+)	83	EB3+	I	Encoder B-phase (+)
34	EB1-	I	Encoder B-phase (-)	84	EB3-	I	Encoder B-phase (-)

Table 2-1: *CN1 Pin Assignment: Main Connector

No.	Name	I/O	Function	No.	Name	I/O	Function
35	EZ1+	I	Encoder Z-phase (+)	85	EZ3+	I	Encoder Z-phase (+)
36	EZ1-	I	Encoder Z-phase (-)	86	EZ3-	I	Encoder Z-phase (-)
37	PEL0	I	End limit signal (+)	87	PEL2	I	End limit signal (+)
38	MEL0	I	End limit signal (-)	88	MEL2	I	End limit signal (-)
39	GDI0	I	DI/LTC/PCS/SD/CLR0	89	GDI2	I	DI/LTC/PCS/SD/CLR2
40	DO0	O	General Output 0	90	DO2	O	General Output 2
41	ORG0	I	Origin signal	91	ORG2	I	Origin signal
42	EXGND	--	Ext. Power Ground	92	EXGND	--	Ext. Power Ground
43	PEL1	I	End limit signal (+)	93	PEL3	I	End limit signal (+)
44	MEL1	I	End limit signal (-)	94	MEL3	I	End limit signal (-)
45	GDI1	I	DI/LTC/PCS/SD/CLR1	95	GDI3	I	DI/LTC/PCS/SD/CLR3
46	DO1	O	General Output 1	96	DO3	O	General Output 2
47	ORG1	I	Origin signal	97	ORG3	I	Origin signal
48	EXGND	--	Ext. Power Ground	98	EXGND	--	Ext. Power Ground
49	EXGND	--	Ext. Power Ground	99	E_24V	I	Isolation Power Input,+24V
50	EXGND	--	Ext. Power Ground	100	E_24V	I	Isolation Power Input,+24V

Table 2-1: *CN1 Pin Assignment: Main Connector

2.6 *CN2 Pin Assignment: Simultaneous Start/Stop

*CN2 is for simultaneous start/stop signals for multiple axes or multiple cards.

No.	Name	Function
1	DGND	Digital Ground
2	STP	Simultaneous stop signal input/output
3	STA	Simultaneous start signal input/output
4	STP	Simultaneous stop signal input/output
5	STA	Simultaneous start signal input/output
6	+5V	Digital +5V Output

Table 2-2: *CN2 Pin Assignment: Simultaneous Start/Stop

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

2.7 SW1 Switch Setting for Card Index

The SW1 switch is used to set the card index. If 0 is set to ON and others are OFF, the card index as 0. Refer to the following table for values from 0-15.

Card ID	Switch Setting (ON=0)
0	1111
1	1110
2	1101
3	1100
4	1011
5	1010
6	1001
7	1000
8	0111
9	0110
10	0101
11	0100
12	0011
13	0010
14	0001
15	0000

Table 2-3: SW1 Switch Setting for Card Index

2.8 *CN3 Manual Pulsar

The signals on *CN3 are for manual pulsar input.

No.	Name	Function (Axis)
1	VDD	+5V Power Supply Output
2	PA+	Pulser A+ phase signal input
3	PA-	Pulser A- phase signal input
4	PB+	Pulser B+ phase signal input
5	PB-	Pulser B- phase signal input
6	EXGND	Ext. Power Ground
7	N/A	N/A
8	N/A	N/A
9	N/A	N/A
10	N/A	N/A

Table 2-4: *CN3 Manual Pulsar

Note: The VDD pin is directly given by the PCI-bus power conversion. Therefore, this signal is isolated.

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-8174 and any motor driver.

This chapter contains the following sections:

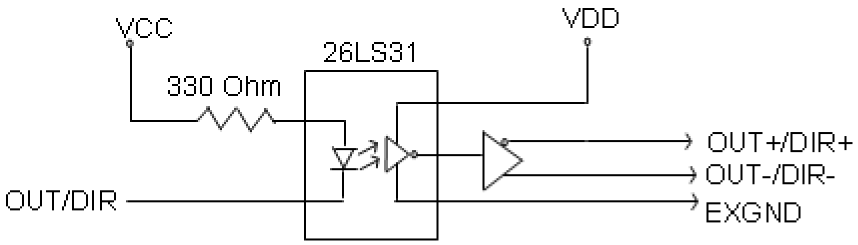
3.1 Pulse Output Signals OUT and DIR

There are 4 axes pulse output signals on the PCI-8174. 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 *CN1. The PCI-8174 has two pairs OUT/DIR pins with high speed trigger OUT. The output pin can be configured for pulse mode or high speed trigger out mode.

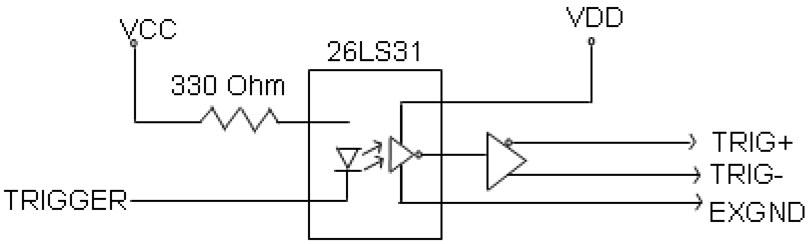
*CN1 Pin No.	Signal Name	Description	Axis #
3	OUT0+ / TRIG0+	Pulse / High Speed Trigger signal (+)	0
4	OUT0- / TRIG0-	Pulse / High Speed Trigger signal (-)	0
5	DIR0+ / TRIG1+	Direction / High Speed Trigger signal (+)	0
6	DIR0- / TRIG1-	Direction / High Speed Trigger signal (-)	0
21	OUT1+ / TRIG2+	Pulse / High Speed Trigger signal (+)	1
22	OUT1- / TRIG2-	Pulse / High Speed Trigger signal (-)	1
23	DIR1+ / TRIG3+	Direction / High Speed Trigger signal (+)	1
24	DIR1- / TRIG3-	Direction / High Speed Trigger signal (-)	1
53	OUT2+	Pulse signal (+)	2
54	OUT2-	Pulse signal (-)	2
55	DIR2+	Direction signal (+)	2
56	DIR2-	Direction signal (-)	2
71	OUT3+	Pulse signal (+)	3
72	OUT3-	Pulse signal (-)	3
73	DIR3+	Direction signal (+)	3
74	DIR3-	Direction signal (-)	3

The following wiring diagram is for OUT and DIR signals of axis.

PCI-8174:



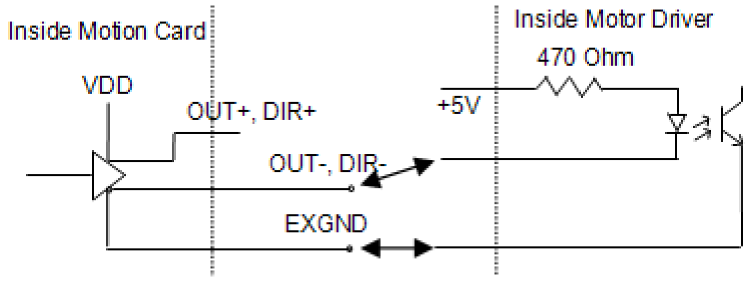
The **default** settings for OUT and DIR are pulse out mode. If high speed trigger signaling is needed, the OUT/DIR pin can be set to trigger out mode via software. The following wiring diagram is for TRIG signal.



Note that this function is only implemented on the first two axes output pins.

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!

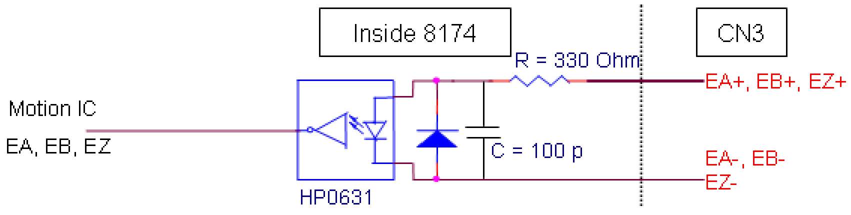
3.2 Encoder Feedback Signals EA, EB and EZ

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:

*CN1 Pin No	Signal Name	Axis #	*CN1 Pin No	Signal Name	Axis #
13	EA0+	0	14	EA0-	0
15	EB0+	0	16	EB0-	0
31	EA1+	1	32	EA1-	1
33	EB1+	1	34	EB1-	1
63	EA2+	2	64	EA2-	2
65	EB2+	2	66	EB2-	2
81	EA3+	3	82	EA3-	3
83	EB3+	3	84	EB3-	3

*CN1 Pin No	Signal Name	Axis #	*CN1 Pin No	Signal Name	Axis #
17	EZ0+	0	18	EZ0-	0
35	EZ1+	1	36	EZ1-	1
67	EZ2+	2	68	EZ2-	2
85	EZ3+	3	86	EZ3-	3

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

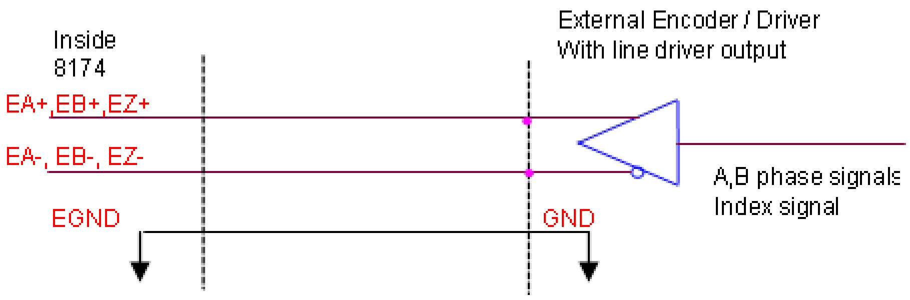


Please note that the voltage across each differential pair of encoder input signals (EA+, EA-), (EB+, EB-), and (EZ+, EZ-) should be at least 3.5V. Therefore, the output current must be observed when connecting to the encoder feedback or motor driver feedback as not to over drive the source. The differential 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-8174 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 3Mhz or more depends on wiring distance and signal conditioning.

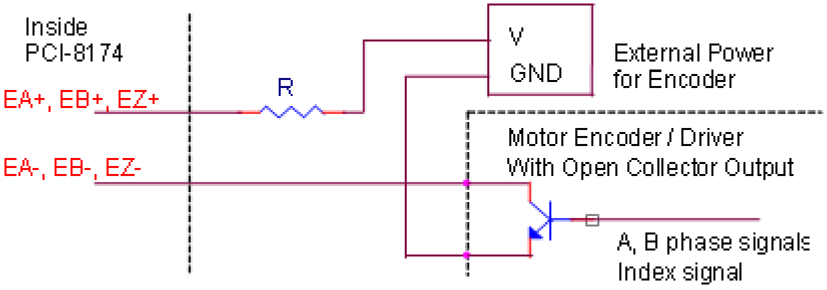


Connection to Open Collector Output

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-8174, 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-8174 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



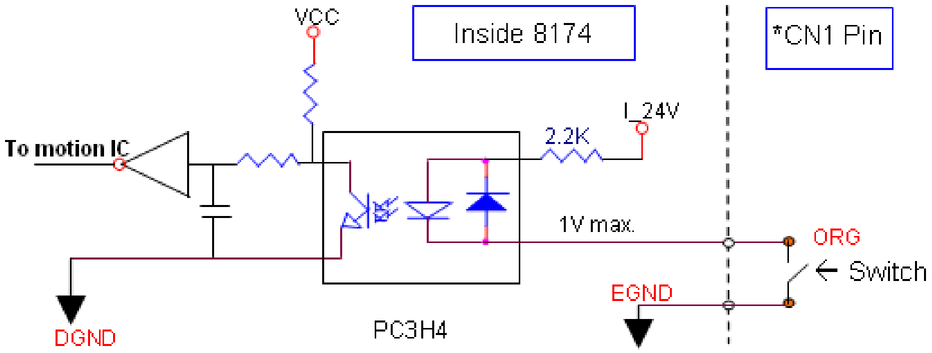
For more operation information on the encoder feedback signals, refer to section 4.4.

3.3 Origin Signal ORG

The origin signals (ORG0-ORG3) are used as input signals for the origin of the mechanism. The following table lists signal names, pin numbers, and axis numbers:

*CN1 Pin No	Signal Name	Axis #
41	ORG0	0
47	ORG1	1
91	ORG2	2
97	ORG3	3

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.



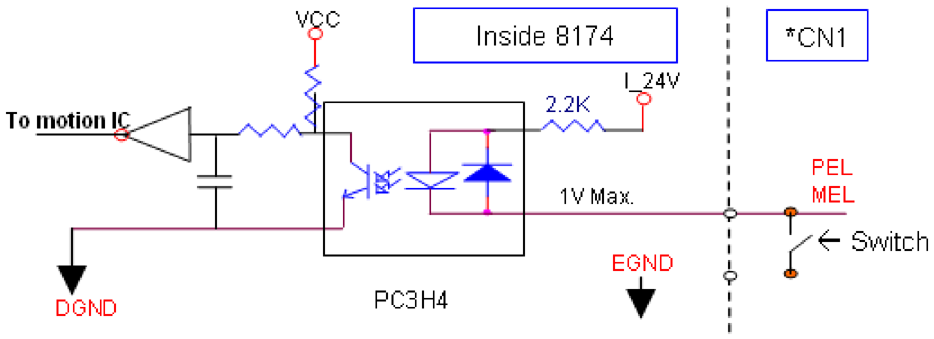
When the motion controller is operated in the home return mode, the ORG signal is used to inhibit the control output signals (OUT and DIR). For detailed operations of the ORG signal, refer to section 4.3.3.

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:

*CN1 Pin No	Signal Name	Axis #	CN3 Pin No	Signal Name	Axis #
37	PEL0	0	38	MEL0	0
43	PEL1	1	44	MEL1	1
87	PEL2	2	88	MEL2	2
93	PEL3	3	94	MEL3	3

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 _8174_set_limit_logic function.

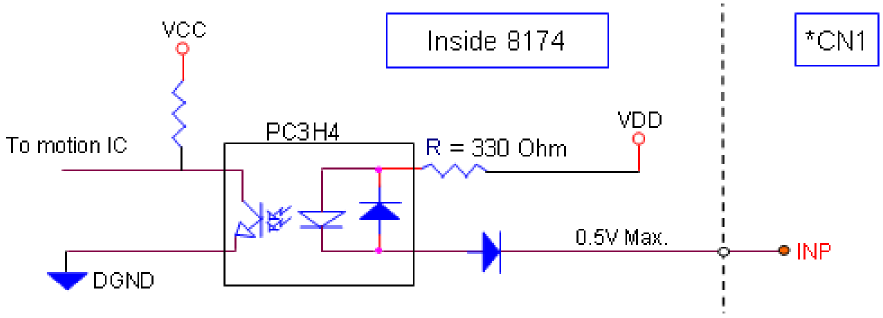


3.5 In-position Signal INP

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:

*CN1 Pin No	Signal Name	Axis #
10	INP0	0
28	INP1	1
60	INP2	2
78	INP3	3

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



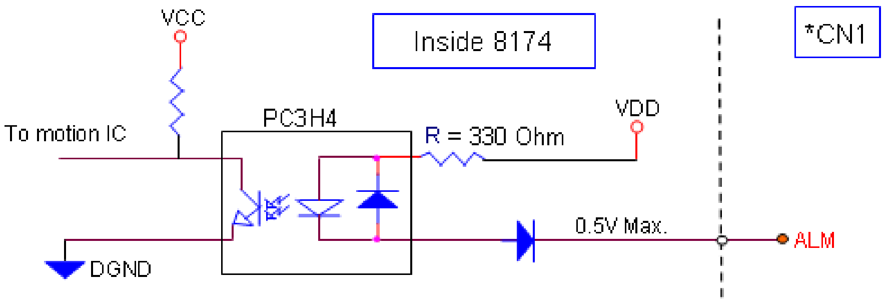
The in-position signal is usually generated by the servomotor driver and is ordinarily an open collector output signal. An external circuit must provide at least 8mA current sink capabilities to drive the INP signal.

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:

*CN1 Pin No	Signal Name	Axis #
9	ALM0	0
27	ALM1	1
59	ALM2	2
77	ALM3	3

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

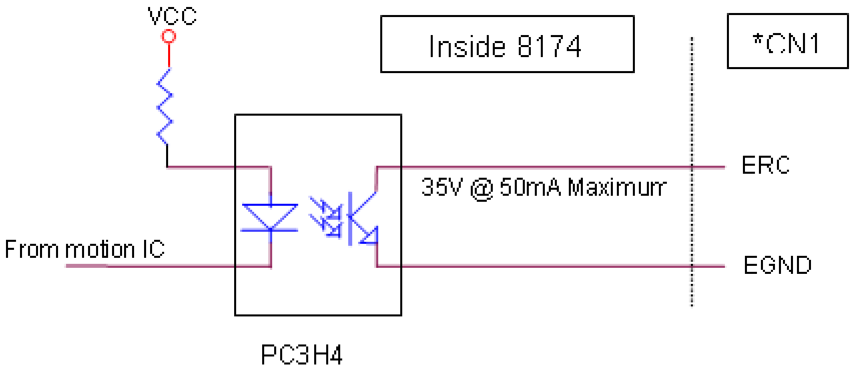
The deviation counter clear signal (ERC) is active in the following 4 situations:

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:

*CN1 Pin No	Signal Name	Axis #
8	ERC0	0
26	ERC1	1
58	ERC2	2
76	ERC3	3

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

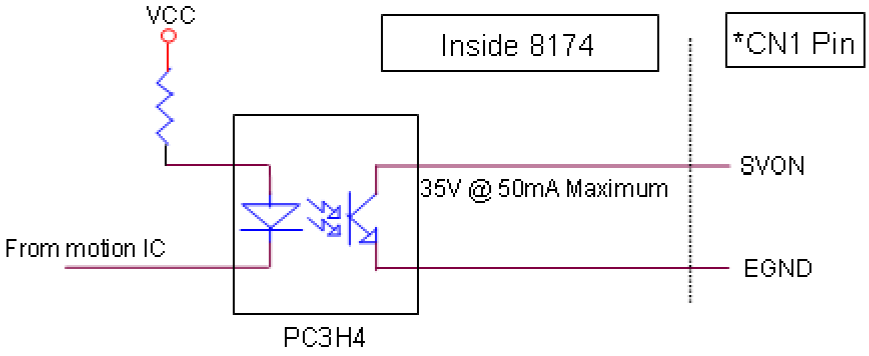


3.8 General-purpose Signal SVON

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:

*CN1 Pin No	Signal Name	Axis #
7	SVON0	0
25	SVON1	1
57	SVON2	2
75	SVON3	3

The output circuit for the SVON signal is shown below:

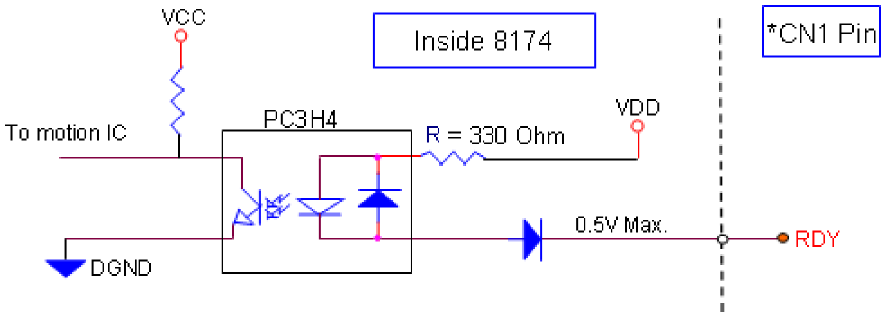


3.9 General-purpose Signal RDY

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:

*CN1 Pin No	Signal Name	Axis #
11	RDY0	0
29	RDY1	1
61	RDY2	2
79	RDY3	3

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



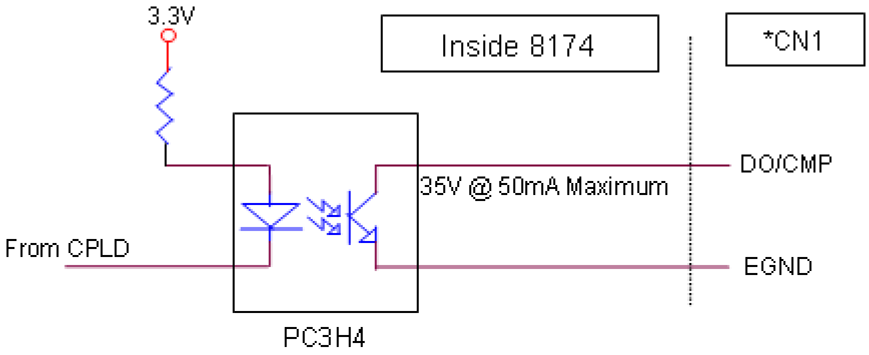
3.10 Multi-Functional output pin: DO/CMP

The PCI-8174 provides 4 multi-functional output channels: DO0/CMP0 to DO3/CMP3 corresponds to 4 axes. Each of the output pins can be configured as Digit Output (DO) or as Comparison Output (CMP) individually. When configured as a Comparison Output pin, the pin will generate a pulse signal when the encoder counter matches a pre-set value set by the user.

The multi-functional channels are located on *CN1. The signal names, pin numbers, and axis numbers are shown below:

*CN1 Pin No	Signal Name	Axis #
40	DO/CMP0	0
46	DO/CMP1	1
90	DO/CMP2	2
96	DO/CMP3	3

The following wiring diagram is of the CMP on each axis:



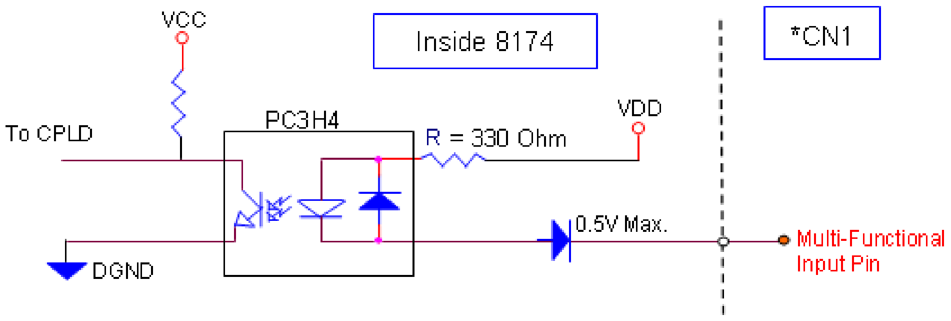
3.11 Multi-Functional input pin: DI/LTC/SD/PCS/CLR/EMG

The PCI-8174 provides 4 multi-functional input pins. Each of the 4 pins can be configured as DI (Digit Input) or LTC (Latch) or SD (Slow down) or PCS (Target position override) or CLR (Counter clear) or EMG (Emergency). To select the pin function, please refer to 6.12.

The multi-functional input pins are on *CN1. The signal names, pin numbers, and axis numbers are shown in the following table:

*CN1 Pin No	Signal Name	Axis #
39	DI/LTC/SD/PCS/CLR/EMG_0	0
45	DI/LTC/SD/PCS/CLR/EMG_1	1
89	DI/LTC/SD/PCS/CLR/EMG_2	2
95	DI/LTC/SD/PCS/CLR/EMG_3	3

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



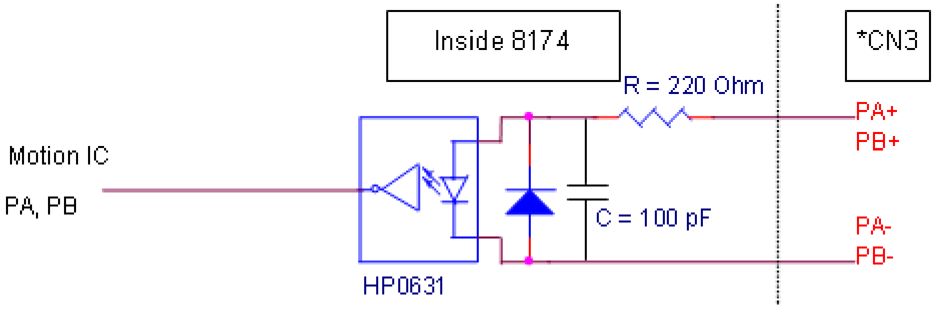
3.12 Pulser Input Signals PA and PB (PCI-8174)

The PCI-8174 can accept differential pulser input signals through the pins of *CN3 listed below. The pulser behaves like an encoder. The A-B phase signals generate the positioning information, which guides the motor.

*CN3 Pin No	Signal Name	Axis #	*CN3 Pin No	Signal Name	Axis #
2	PA+	0-3	3	PA-	0-3
4	PB+	0-3	5	PB-	0-3

The pulser signals are used for Axis 0 to Axis 3. User can decide to enable or disable each axis pulser with _8174_disable_pulser_input function.

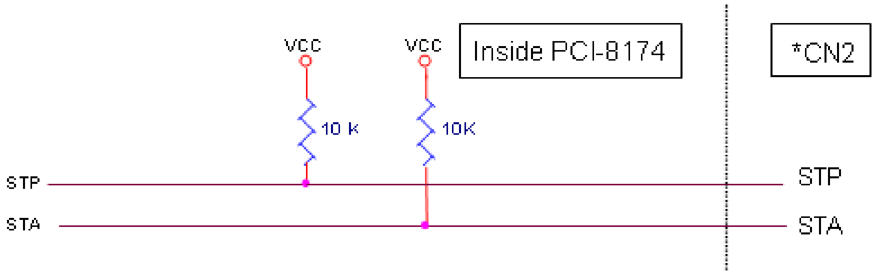
The wiring diagram of the differential pulser input pins are as followed.



3.13 Simultaneously Start/Stop Signals STA and STP

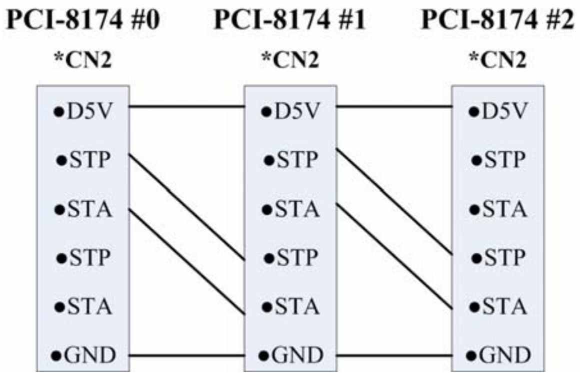
The PCI-8174 provides STA and STP signals, which enable simultaneous start/stop of motions on multiple axes. The STA and STP signals are on *CN2

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-8174. An external open collector or switch can be used to drive the STA/STP signals for simultaneous start/stop.

If there are two or more PCI-8174 cards, connect the *CN2 connector on the previous card to *CN2 connector on the following card.



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

When motor/stepper control first started, motion control was widely discussed instead of motor control. Motor control was separated into two layers: motor control and motion control. Motor control relates to PWM, power stage, closed loop, hall sensors, vector space, etc. Motion control relates to speed profile generating, trajectory following, multi-axes synchronization, and coordinating.

4.1.1 Voltage motion control interface

The interfaces between motion and motor control are changing rapidly. Early on, a voltage signal was used as a command to the motor controller. The amplitude of the signal means how fast a motor is 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” motion controller. It is much easier to integrate into an analog circuit of motor controller; however noise is sometimes a big problem for this type of motion control. Also, to do positioning control of a motor, the analog motion controller must have a feedback signal with 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 motion control interface

The second interface of motion and motor control is a pulse train type. As a trend of digital world, pulse trains 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, a servo or stepper motor can be easier than an 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 provide for gains tuning. For analog position controllers, the control loops are built inside and users must tune the gain from the controller. For pulse type position controllers, 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 industrial 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 motion control interface

Recently, there was a new control interface was introduced--a network 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 it is digitized and packetized. Because a motion controller must be real-time, the network must have real-time capacity around a cycle time below 1 mini-second. This means that non-commercial networks cannot do this job. It must have a specific network, such as Mitsubishi SSCNET. The network may also be built with fiber optics to increase communication reliability.

4.1.4 Software real-time motion control kernel

For motion control kernel, there are three ways to accomplish it: DSP, ASIC, and software real-time.

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. A feed-back counter card can simply be used 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 real-time 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 motion control kernel

A DSP-based motion controller kernel solves real-time software problem on computer. DSP is a microprocessor 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. The software interfacing between DSP controller's vendors and users is 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. DSP based controller provide a better way than software kernel for machine makers to build they applications.

4.1.6 ASIC motion control kernel

An ASIC motion control kernel is falls between software kernel and DSP kernel in terms of difficulty. It has no real-time problem because all motion functions are done via the ASIC. Users or controller's vendors just need to set some parameters which the 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 the 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

* Including real-time OS

** DSP or real-time OS required

4.1.8 PCI-8174 motion controller type

The PCI-8174 is a DSP plus ASIC based and a pulse motion controller made into five blocks: on board DSP, DSP kernel, motion ASIC, PCI card, software motion library. The motion ASIC can be accessed via our software motion library under many kinds of Windows 2000/XP/Vista, Linux, and RTX driver. Our software motion library provides one-stop-function for controlling motors. All the speed parameter calculations are done via our library.

For example, to perform a one-axis point to point motion with a trapezoidal speed profile, only fill the target position, speed, and acceleration time in one function. Then the motor will run as the profile. It takes no CPU resources because every control cycle pulse generation is done by the ASIC. The precision of target position depends on motor drivers' closed loop control performance and mechanical 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, mechanical or electrical engineers to find out problems.

The on board DSP makes the PCI-8174 act as a standalone system. Besides accessing motion ASIC via the motion library, it can be accessed via other means. For example, time-critical motion sequences (such as position comparing with triggering output, gantry mode for two axes synchronizing with feedback monitoring, high speed pick and place sequence and other customized motion sequences) can be stored into the DSP kernel like. Contact us for custom DSP kernel support. The default DSP kernel only supports triggering and gantry functions.

4.2 Motion Control Modes

Motion control makes 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

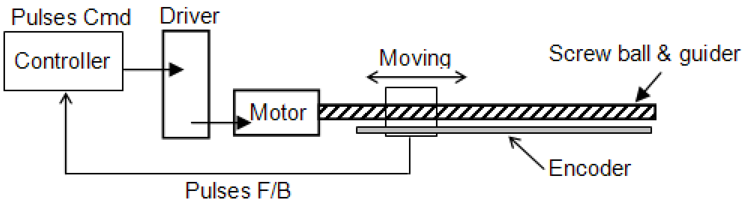
The Cartesian coordinate is used and pulses are in the unit of length. The physical length depends on mechanical parts and motor's resolution. For example, if a motor is on a screw ball, and the pitch of screw ball is 10mm and the pulses needed for a round of motor are 10,000 pulses. One pulse's physical unit is equal to 10mm/10,000p =1 mm.

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? **Simple! 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 and we want to move a motor to 9000, you 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. The result is +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).

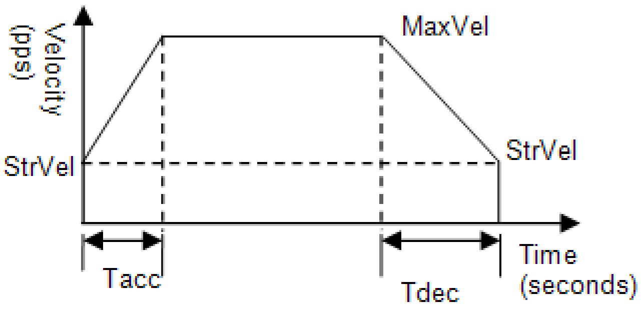


4.2.2 Absolute and relative position move

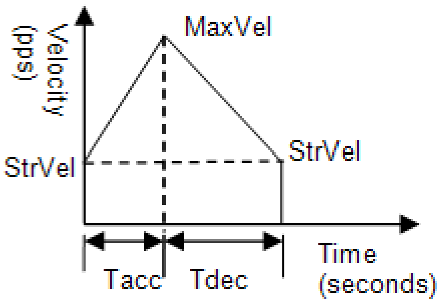
In the coordinate system, we have two kinds command for users to locate the target position. One is absolute and the other is relative. Absolute command means that user give the motion controller a position, then the motion controller will move a motor to that position from current position. Relative 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

Trapezoidal 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 distance requirement. The chart of this situation is shown as below:

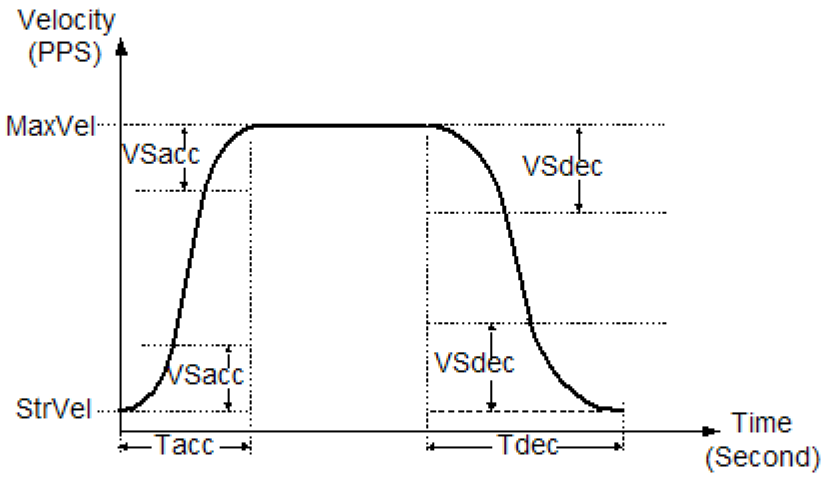


This kind of speed profile could be applied on velocity mode, position mode in one axis or multi-axes linear interpolation and two axes circular interpolation modes.

4.2.4 S-curve and Bell-curve speed profile

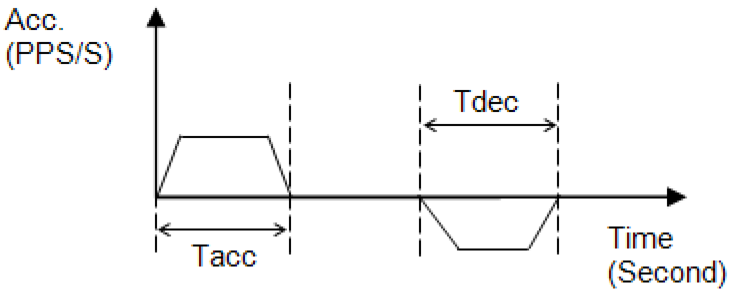
S-curve means the speed profile in accelerate/decelerate area follows a 2nd order curve. It can reduce vibration at the beginning of motor start and stop. In order to speed up the acceleration/deceleration during motion, we need to insert a linear part into these areas. We call this shape as “Bell” curve. 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 parameters 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

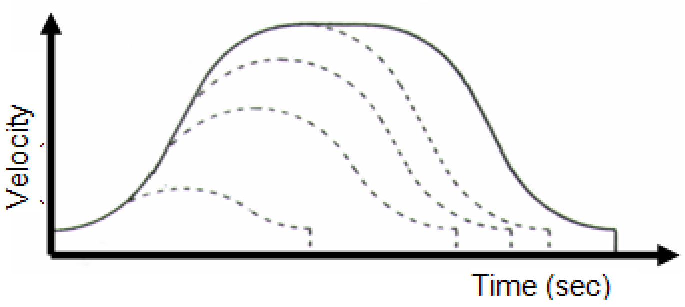
If VSacc or VSdec=0, it means acceleration or deceleration use pure S-curve without linear part. The Acceleration chart of bell curve is shown below:



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).

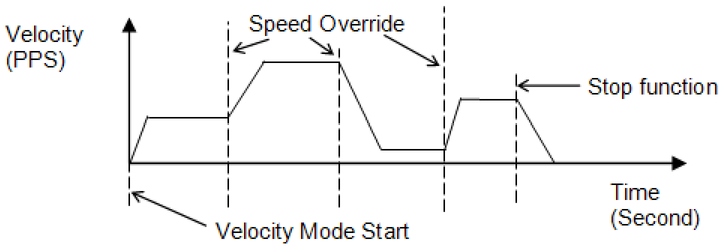
The rule is to lower the value of MaxVel and the Tacc, Tdec, VSacc, VSdec automatically, and keep StrVel, acceleration, and jerk unchanged. This is also applicable to Trapezoidal profile motion.

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.



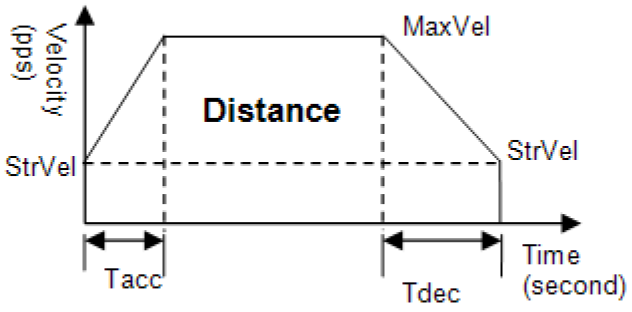
4.2.5 Velocity mode

Velocity mode means the pulse command is continuously outputting 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 overridden 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

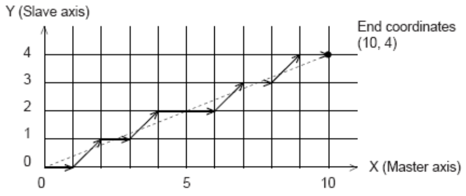
Position mode means the motion controller will output a specific amount of pulses which is equal to users' desired position or distance. The unit of distance or position is pulse internally on the motion controller. The minimum length of distance is one pulse. However, in PCI-8174, we provide a floating 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.



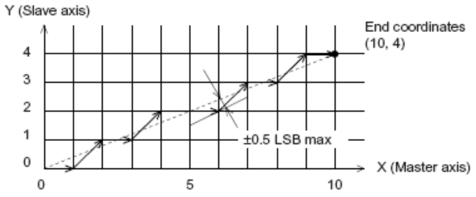
The distance is the area of the V-t diagram of this profile.

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.



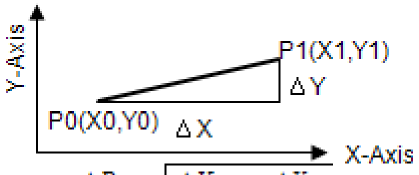
The pulses output from X or Y axis remains 1/2 pulse difference according to a perfect linear line. The precision of linear interpolation is shown as below:



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

As in the diagram below, 4-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 (ΔX: ΔY), respectively, and the vector speed is:

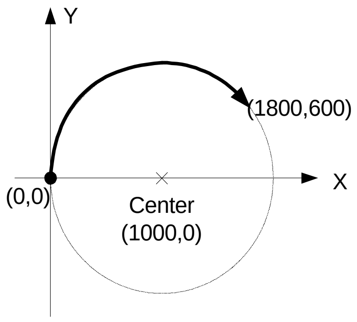


$$\frac{\Delta P}{\Delta t} = \sqrt{\left(\frac{\Delta X}{\Delta t}\right)^2 + \left(\frac{\Delta Y}{\Delta t}\right)^2}$$

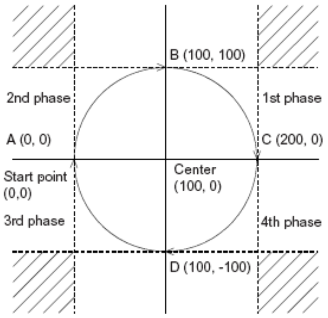
When calling 4-axis linear interpolation functions, the vector speed needs to define the start velocity, StrVel, and maximum velocity, MaxVel.

4.2.8 Two axes circular interpolation mode

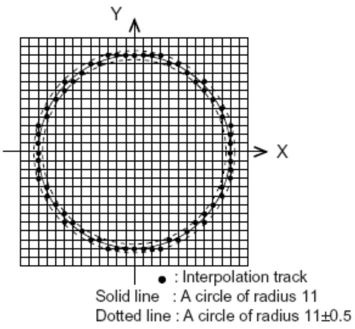
Circular interpolation means XY axes simultaneously start from initial point, (0,0) and stop at end point, (1800,600). The path between them is an arc, and the MaxVel is the tangential speed. Notice that if the end point of arc is not at a proper position, it will move circularly without stopping.



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.

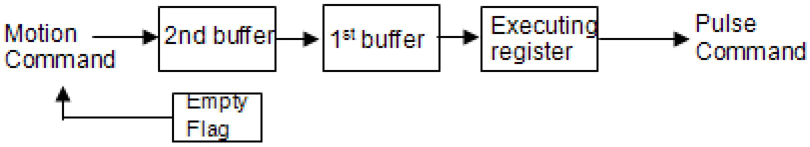


Y
X
● : Interpolation track
Solid line : A circle of radius 11
Dotted line : A circle of radius 11±0.5

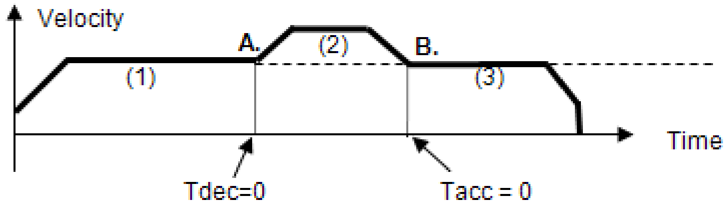
4.2.9 Continuous motion

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 architecture of continuous motion.

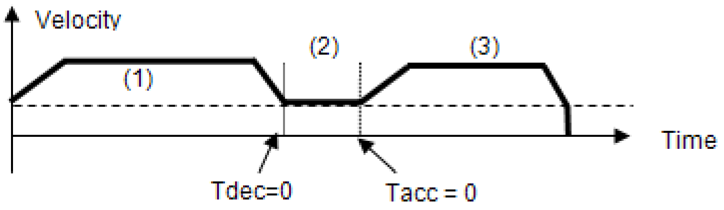


Besides position command, the speed command should be set correctly to perform a speed continuous profile. For the following example, there are three motion command of this continuous motion. The second one has high speed than the others. The interconnection of speed between these three motion functions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



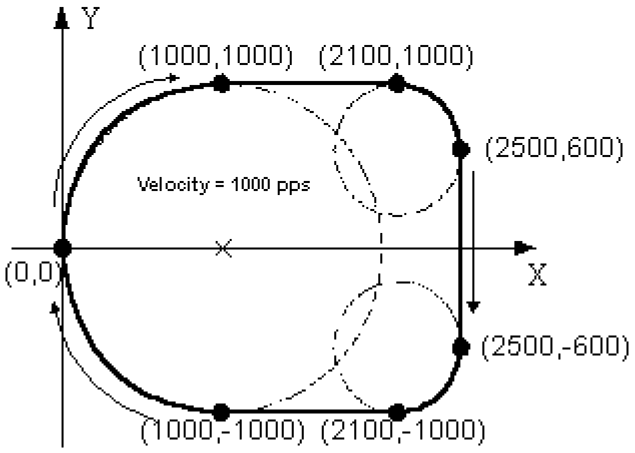
1st command's Tdec=0
2nd command's StrVel = 1st command's MaxVel
3rd command's Tacc=0
3rd command's MaxVel=2nd command's StrVel

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 4-axis continuous arc interpolation is the same concept. You can set the speed matched between two command speed settings.



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 need the smoothing, please turn INP checking off.

4.2.10 Home Return Mode

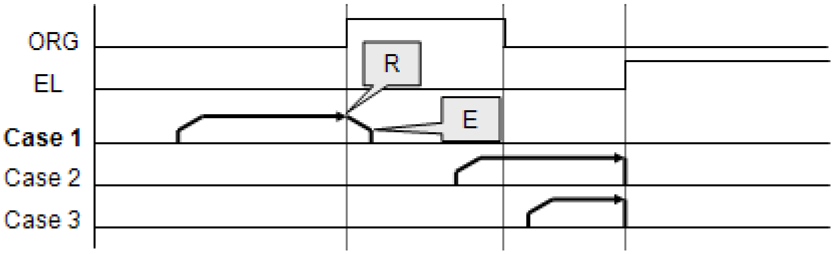
Home return means searching a zero position point on the coordinate. Sometimes, users use a ORG, EZ or EL pin as a zero position on the coordinate. At the beginning of machine power on, the program needs to find a zero point of this machine. Our motion controller provides a home return mode to make it.

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 resetting 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 machine's zero point already.

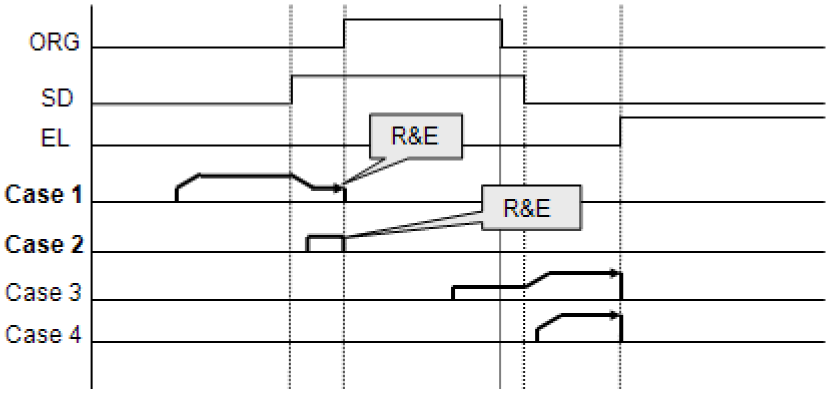
The following figures show the various home modes: R means counter reset (command and position counter). E means ERC signal output.

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

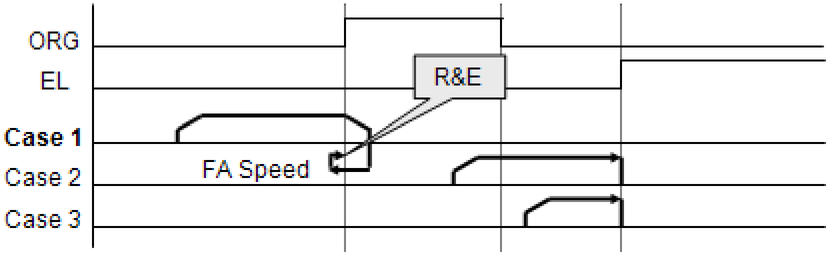
- ▶ When SD is not installed



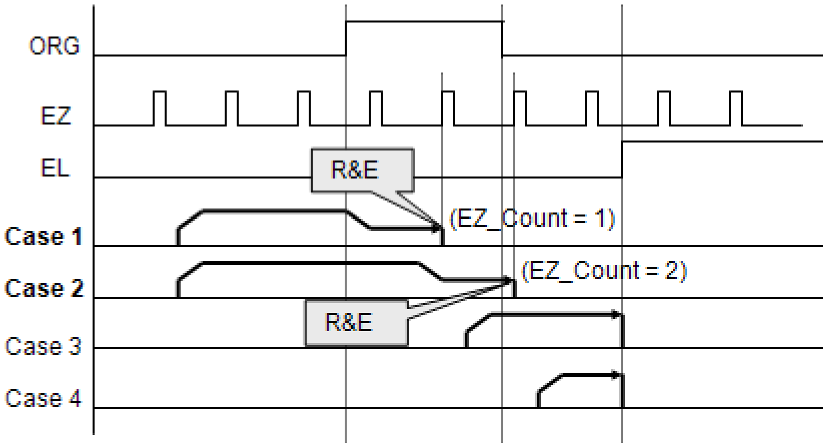
- ▶ When SD is installed and SD is not latched



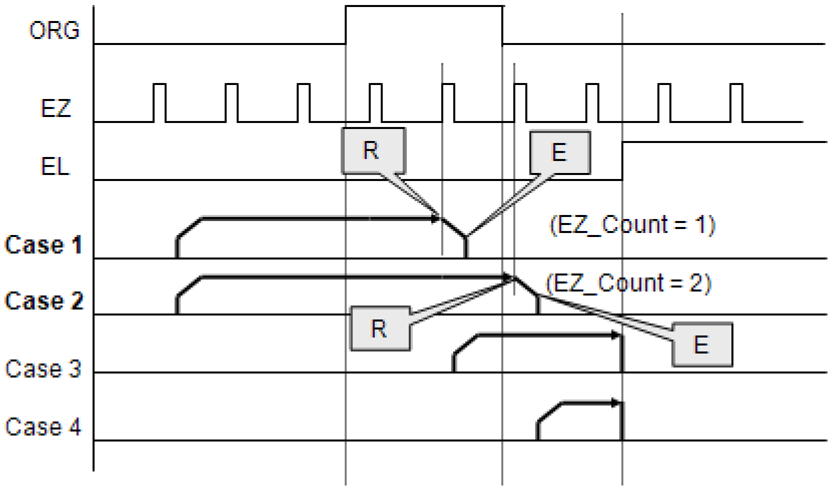
Home mode=1: (Twice ORG turn ON then reset counter)



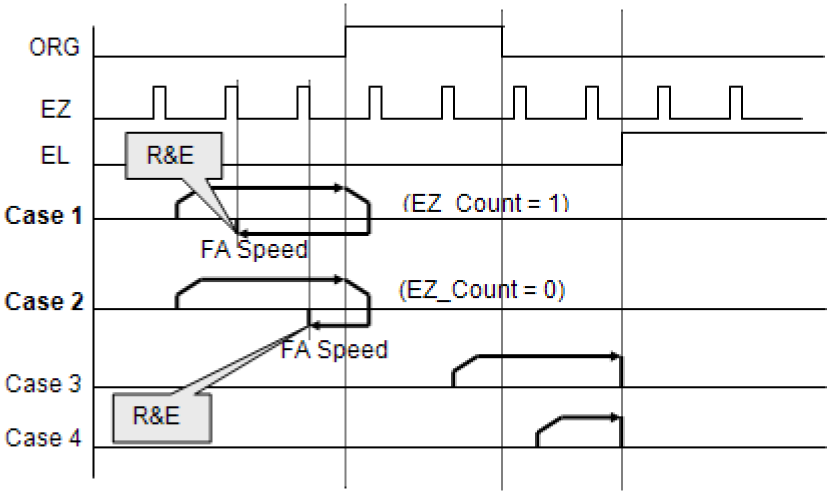
Home mode=2: (ORG ON then Slow down to count EZ numbers and reset counter)



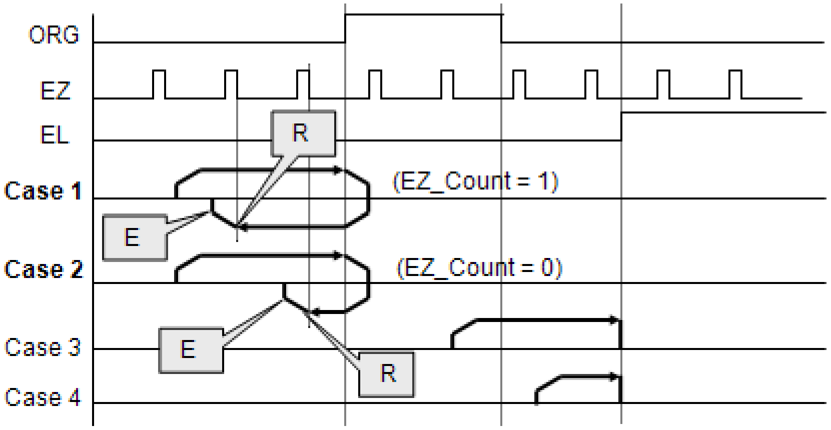
Home mode=3: (ORG ON then count EZ numbers and reset counter)



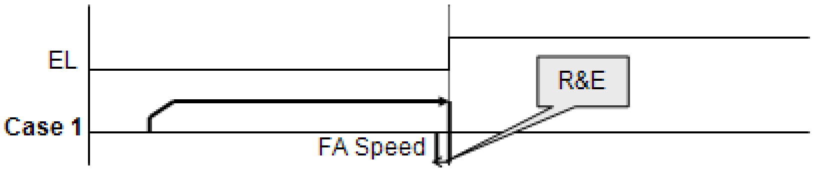
Home mode=4: (ORG On then reverse to count EZ number and reset counter)



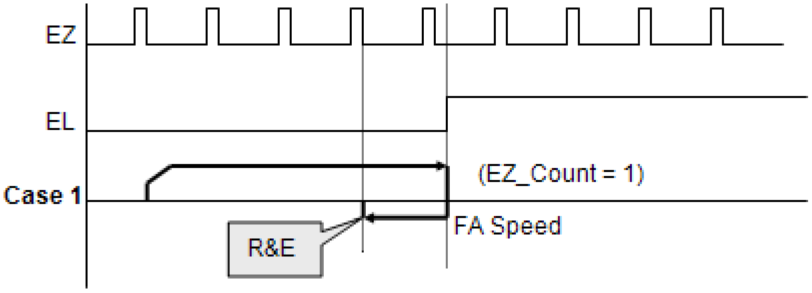
Home mode=5: (ORG On then reverse to count EZ number and reset counter, not using FA Speed)



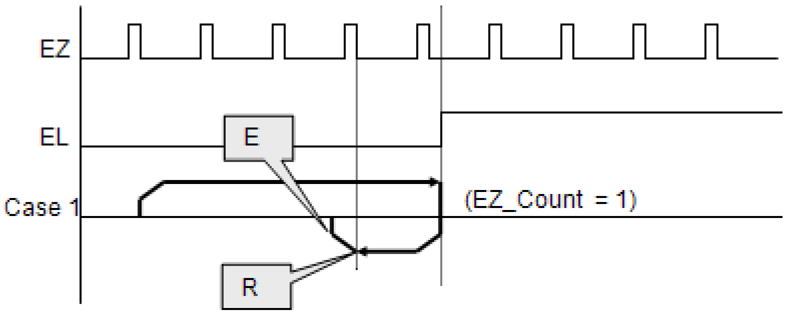
Home mode=6: (EL On then reverse to leave EL and reset counter)



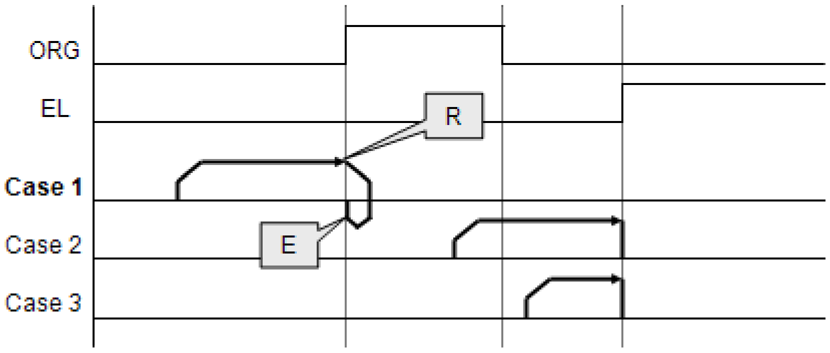
Home mode=7: (EL On then reverse to count EZ number and reset counter)



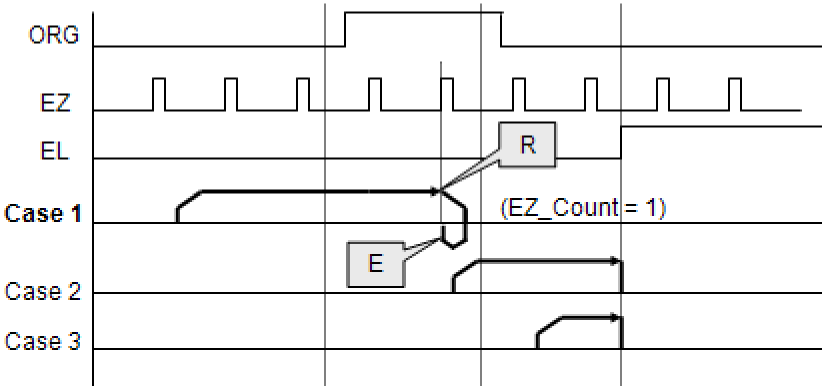
Home mode=8: (EL On then reverse to count EZ number and reset counter, not using FA Speed)



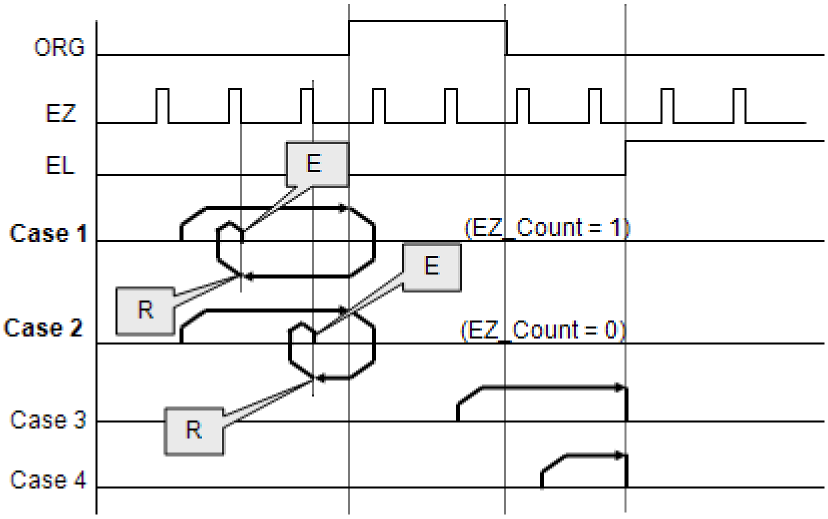
Home mode=9: (ORG On then reverse to zero position, an extension from mode 0)



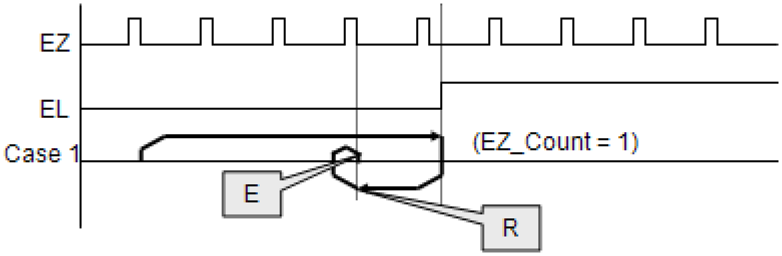
Home mode=10: (ORG On then counter EZ and reverse to zero position, an extension from mode 3)



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

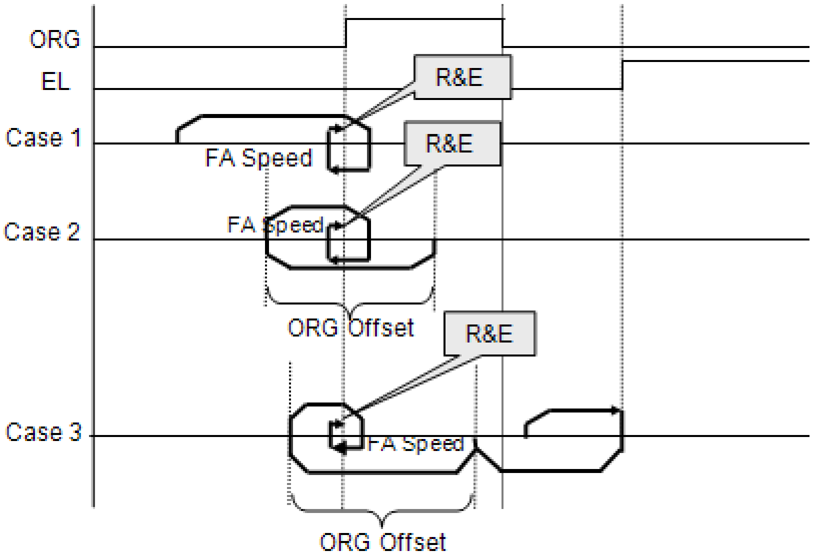


Home mode=12: (EL On then reverse to count EZ number and reverse to zero position, an extension from mode 8)



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.

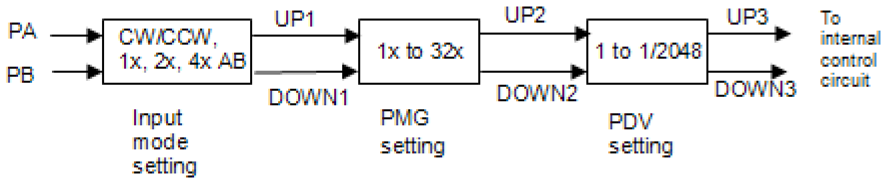


4.2.12 Manual Pulser Function

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.

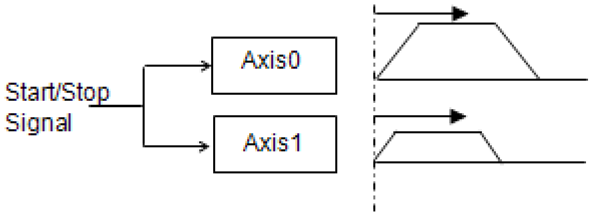
The motion controller receives two kinds of pulse trains from manual pulser device: CW/CCW and AB phase. If the AB phase input mode is selected, the multiplier has additional selection of 1, 2, or 4.

The following figure shows pulser ratio block diagram.



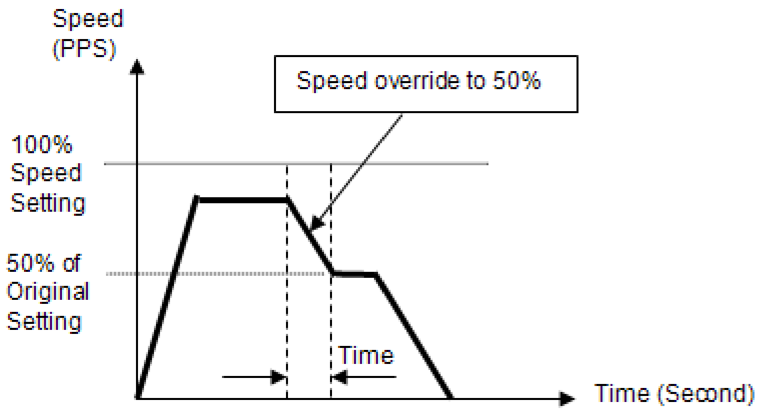
4.2.13 Simultaneous Start Function

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 external 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.



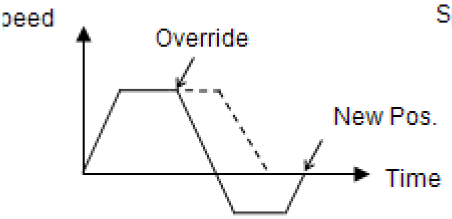
4.2.14 Speed Override Function

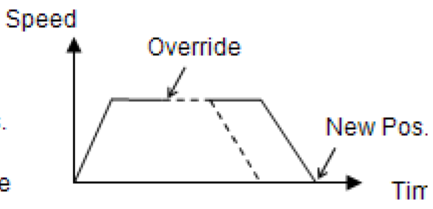
Speed override means that users can change command's speed during the operation of motion. The change parameter is a percentage of original defined speed. Users can define a 100% speed value then change the speed by percentage of original speed when motion is running. If users didn't define the 100% 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.

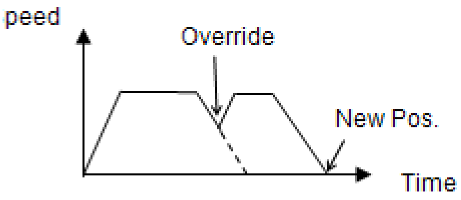


4.2.15 Position Override Function

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

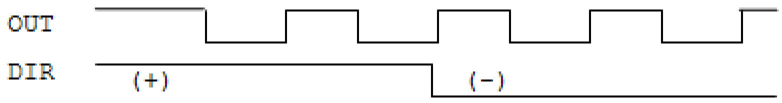
The motion controller uses pulse command to control servo/stepper motors via motor drivers. Please set the drivers to position mode which can accept pulse trains as position command. The pulse command consists of two signal pairs. It is defined as OUT and DIR pins on connector. Each signal has two pins 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)

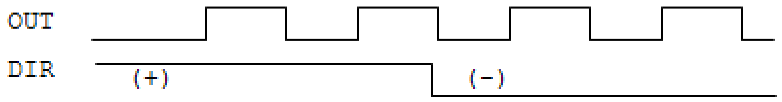
Single Pulse Output Mode (OUT/DIR Mode)

In this mode, the OUT pin is for outputting 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.

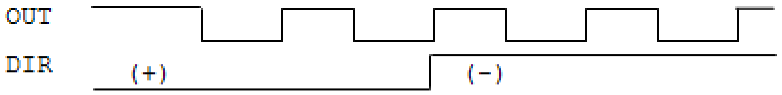
Pulse mode = 0: (OUT pin normally high)



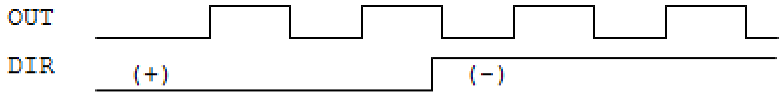
Pulse mode = 1: (OUT pin normally low)



Pulse mode = 2: (OUT pin normally high)

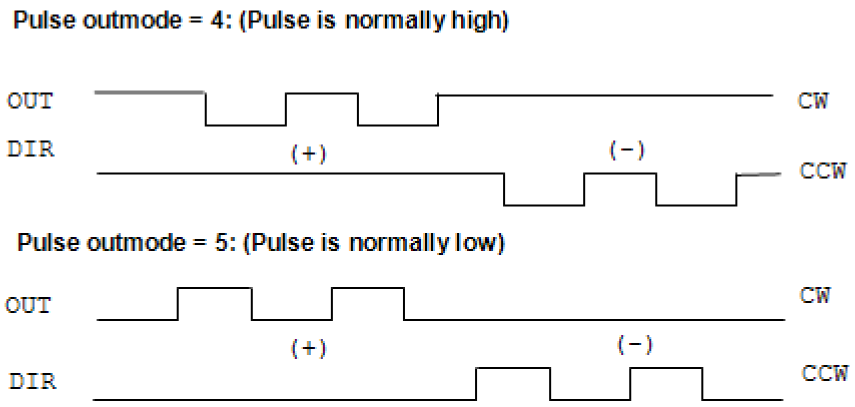


Pulse mode = 3: (OUT pin normally low)



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)
OUT
DIR
(+)
(-)
CW
CCW
Pulse outmode = 5: (Pulse is normally low)
OUT
DIR
(+)
(-)
CW
CCW

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

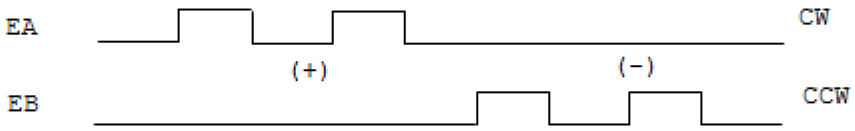
Our motion controller provides one 28-bit up/down counter of each axis for pulse feedback counting. This counter is called position counter. The position counter counts pulses from the EA and EB signal which have plus and minus pins on connector for differential signal inputs. It accepts two kinds of 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 pulse 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)



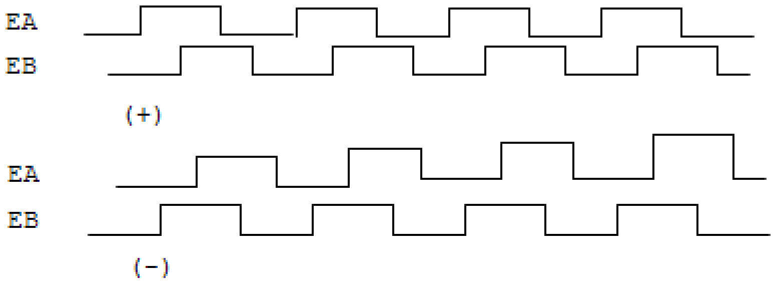
The pattern of pulses in this mode is the same as the **Dual Pulse Output Mode** in the Pulse Command Output section except that the input pins are EA and EB.

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.

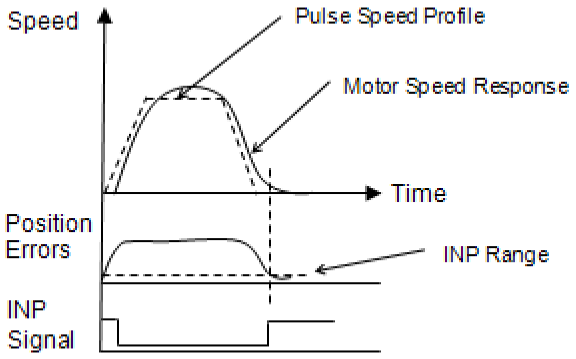
The following diagram shows the waveform.



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 correct 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 will 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 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 held 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 fall on the ground. Some situations like servo alarm and emergency 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 inputted 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 resetting 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

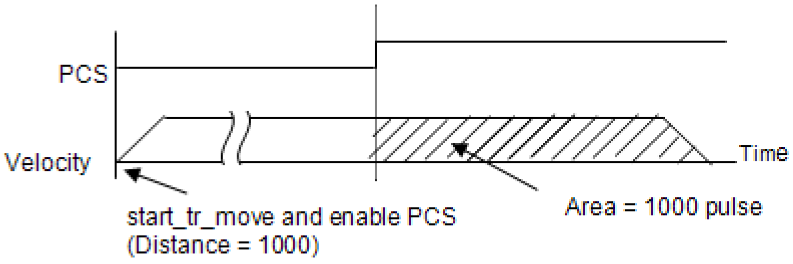
The end-limit switches are usually installed on both ending sides of one axis. We must install plus EL at the positive position of the axis and minus EL at the negative position of the axis. These two signals are for safety reason. If they are installed reversely, the protection will be invalid. Once 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 signal 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 emergency situation. Once the input is turned on, our motion controller will stop all axes' motion immediately to prevent machine's damage. Usually, users can connect an emergency stop button to this input on their machine. We suggest this input as normal closed type for safety.

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 mechanical 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 setting
4. Clock Ticks – Counter from a timer about 9.8MHz

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 and stop functions.

4.6 The Comparators

There are 5 counter comparators of each axis. Each comparator 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

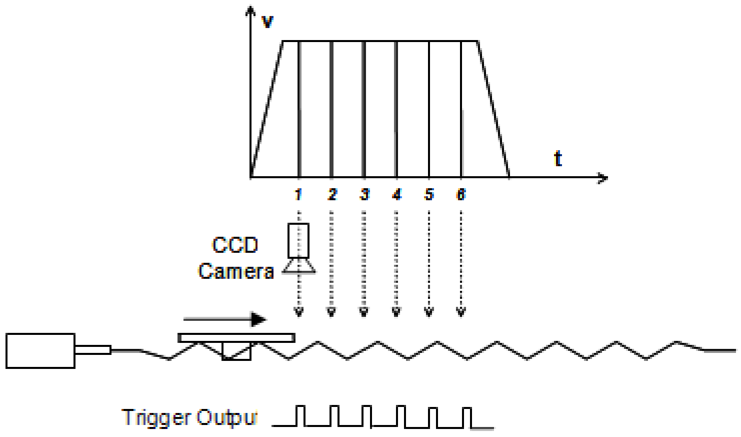
This comparator is only for command and feedback counter error. Users can use this comparator to check if the error is too big. It can be set a action when this condition is met. The actions include generating interrupt, immediately stop, and deceleration to stop.

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 direction-less. 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

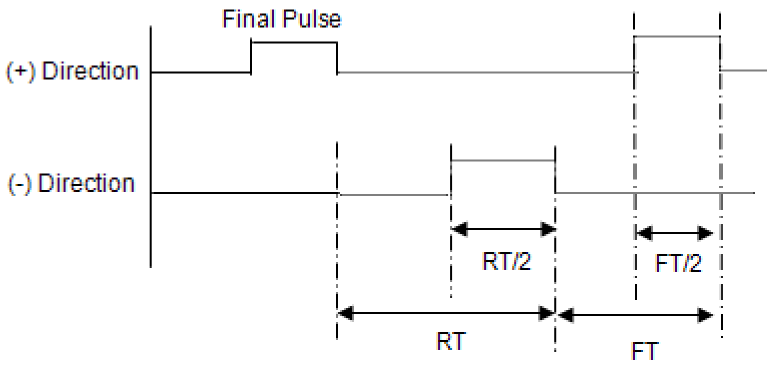
We provide many other functions on the motion controller. Such as backlash compensation, slip correction, vibration restriction, speed profile calculation and so on. The following sections will describe these functions.

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.

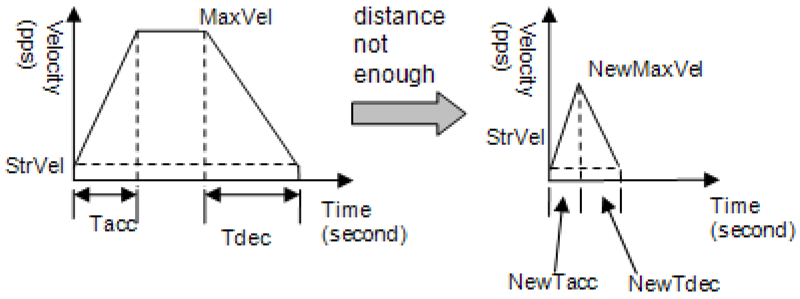
4.7.2 Vibration restriction function

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 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 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:

	Axis No.	**X**	**Y**	**Z**	**U**
Card ID					
0		0	1	2	3
2		8	9	10	11
3		12	13	14	15

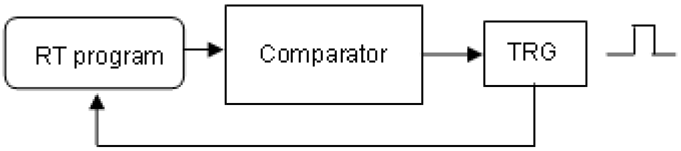
Notice that if there has the same card ID on multiple cards, the function will not work correctly.

4.9 DSP-based Triggering functions

Some applications, such as line scan camera capturing, need a large number of position comparison points for continuously triggering. Normally, there is only one hardware position comparator for position compare and triggering, so the continuous triggering is accomplished by software reloading next trigger position to comparator. Because the triggering frequency is high, it is difficult to maintain in real-time. The PCI-8174 uses a motion ASIC as a comparator and DSP kernel as a hard real-time reloading software to achieve 100Khz high frequency triggering. The following sections introduce the mechanism of this design.

4.9.1 Function block diagram

There are three major components for this function. One is comparator, the other is triggering circuit and another is real-time program inside DSP. Please see the following diagram.



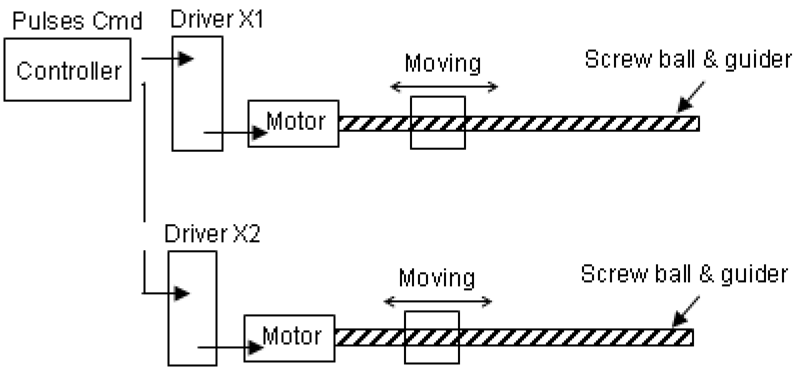
All these three blocks are on the PCI-8174 and will not be affected by programs on PC side. The RT program will load next points into comparator when last trigger is fired.

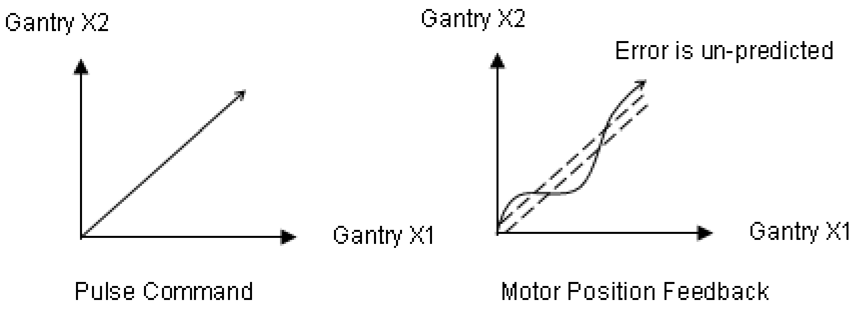
4.9.2 Compared points

The compared points are from a pre-defined linear function with fixed interval or a pre-loaded random points array on PCI-8174 SDRAM. Both of these two methods must be initialized from the PC via API functions. The linear function type is for fixed interval comparing application. It needs only three informations: start point, interval, end point or times of reloading. The random point method is used on non-fixed interval applications. A large point table inside the PCI-8174’s on board RAM must be defined via API functions.

4.10 DSP-based Gantry functions

Gantry functions are defined from mechanical structure and normally two motors controlling two parallel linear stages are used. These two motors must move simultaneously. The motion controller must control these two motors individually but synchronously. It looks like 1:1 electric gearing in master-slave motion and also looks like 1:1 interpolation motion of two axes. With the PCI-8174, use the linear motion functions of two axes to achieve this function from the PC side. It is only for command pulse synchronization. Some applications, especially on unbalanced heavy loading conveyor system, it doesn't work perfectly because of open loop architecture. Please see the following diagram:

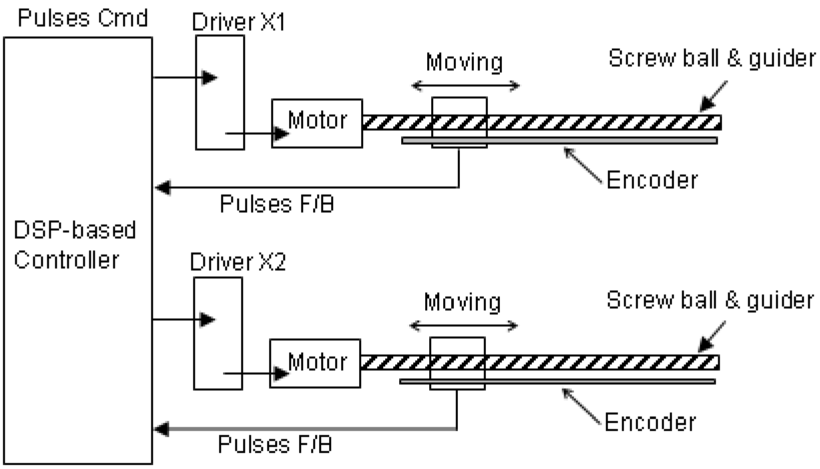


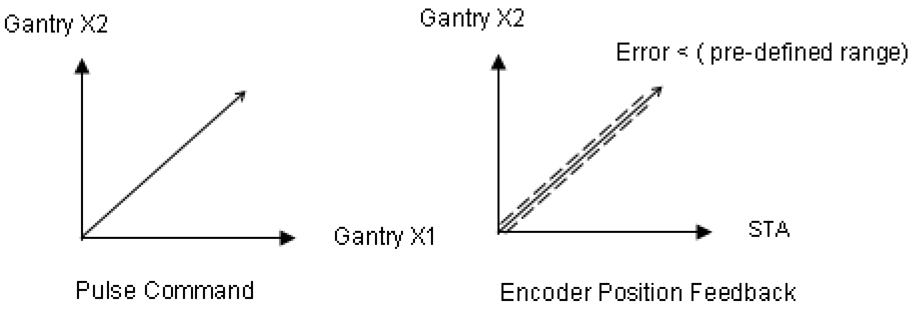


If the gantry X1 and X2 has different loading and servo parameters, these two axes will not move simultaneously.

4.10.1 Closed-loop gantry mode

In order to solve open loop gantry problem, use the PCI-8174 to make a closed loop system to control gantry stage. The control architecture is shown as below:





4.10.2 Error handling of gantry mode

During closed loop gantry motion, there might be an error condition to stop one of the gantry axes. Once this error has occurred, the other axis will stop immediately. The error could be from predefined error tolerant overrun, emergency stop occurs, alarm from motor driver, end-limit stop, etc.

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 8174 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 8174 cards.

Note that MotionCreatorPro is only available for Windows 2000/XP/Vista with a screen resolution higher than 1024x768. Recommended screen resolution is 1024x768. It cannot be executed under the DOS environment.

5.1 Execute MotionCreatorPro

After installing the software drivers for the 8174 in Windows 2000/XP/Vista, 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.

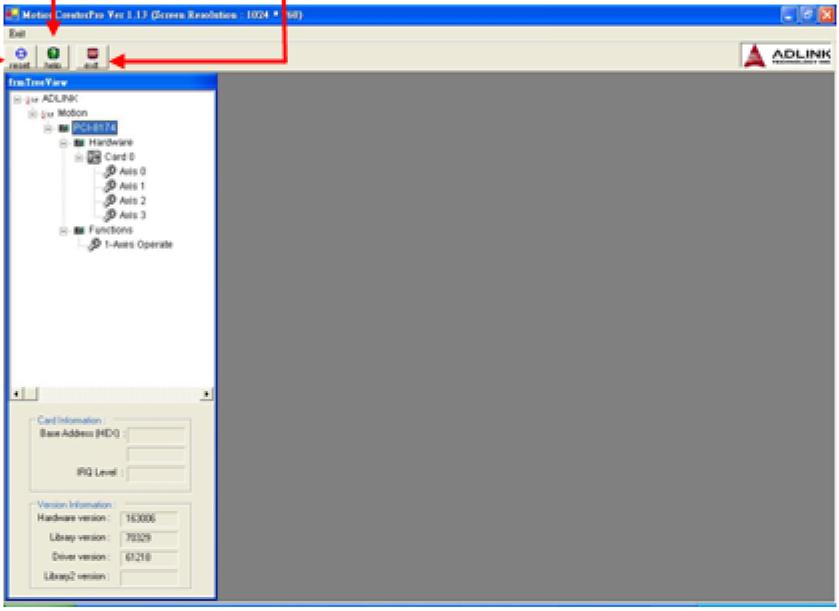
1. MotionCreatorPro is a program written in VB.NET 2003, and is available only for Windows 2000/XP with a screen resolution higher than 1024x768. It cannot be run under DOS.
2. MotionCreatorPro allows users to save settings and configurations for 8174 cards. Saved configurations will be automatically loaded the next time MotionCreatorPro is executed. Two files, **8174.ini** and **8174MC.ini**, in the **windows root directory** are used to save all settings and configurations.
3. To duplicate configurations from one system to another, copy 8174.ini and 8174MC.ini into the windows root directory.
4. If multiple 8174 cards use the same MotionCreatorPro saved configuration files, the DLL function call **_8174_config_from_file**() can be invoked within a user developed program. This function is available in a DOS environment as well.

5.3 MotionCreatorPro Introduction

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 5.3.6)
- Exit MotionCreatorPro

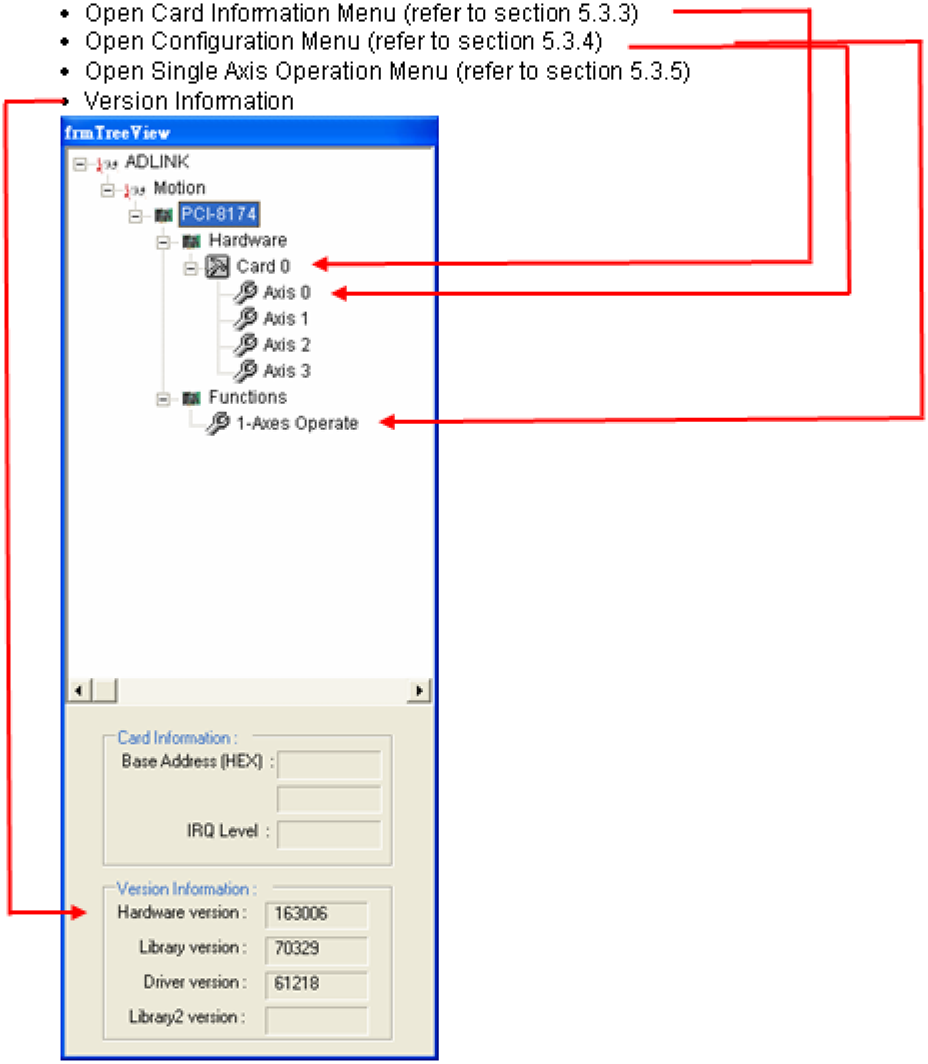


Hardware version : 163006
Library version : 70329
Driver version : 61218
Library2 version :

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)
- Version Information



Base Address (HEX) :

IRQ Level :

Version Information :

Hardware version : 163006

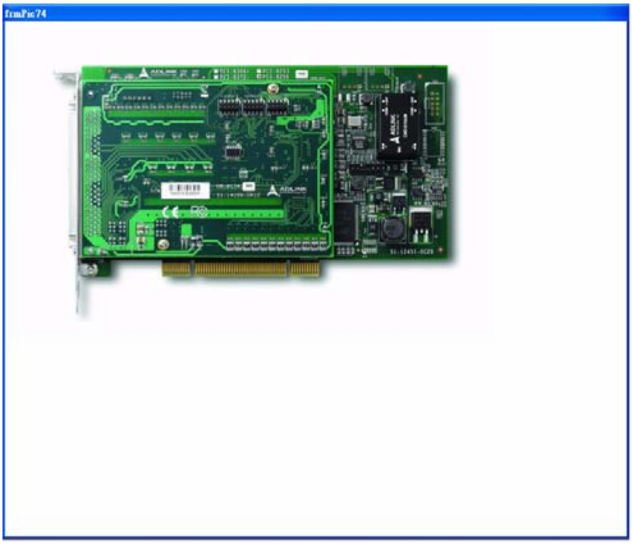
Library version : 70329

Driver version : 61218

Library2 version :

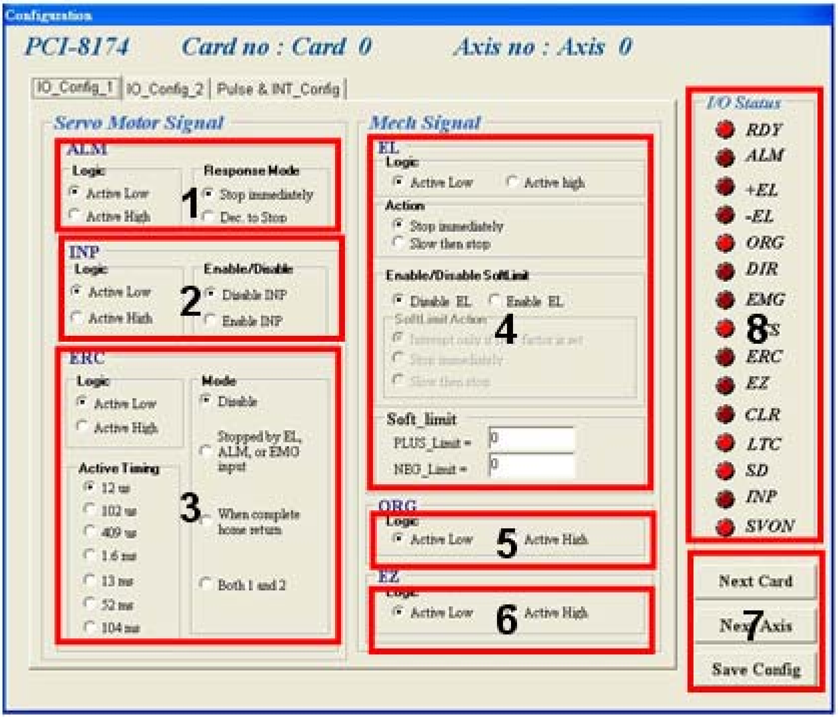
5.3.3 Card Information Menu

In this menu, it shows some Information about this card:



5.3.4 Configuration Menu

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



Configuration

PCI-8174 Card no : Card 0 Axis no : Axis 0

IO_Config_1 | IO_Config_2 | Pulse & INT_Config

Servo Motor Signal

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

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

3 ERC
- Logic: [✓ Active Low] Active High
- Active Timing: [✓ 12 us] 102 us 409 us 1.6 ms 13 ms 52 ms 104 ms
- Mode: [✓ Disable] Stopped by EL, ALM, or EMG input When complete home return Both 1 and 2

Mech Signal

4 EL
- Logic: [✓ Active Low] Active high
- Action: [✓ Stop immediately] Slow then stop
- Enable/Disable SoftLimit: [✓ Disable EL] Enable EL
- SoftLimit Action: [✓ Interrupt only if ... factor is set] Stop immediately Slow then stop
- Soft_limit: PLUS_Limit = 0, NEG_Limit = 0

5 ORG
- Logic: [✓ Active Low] Active High

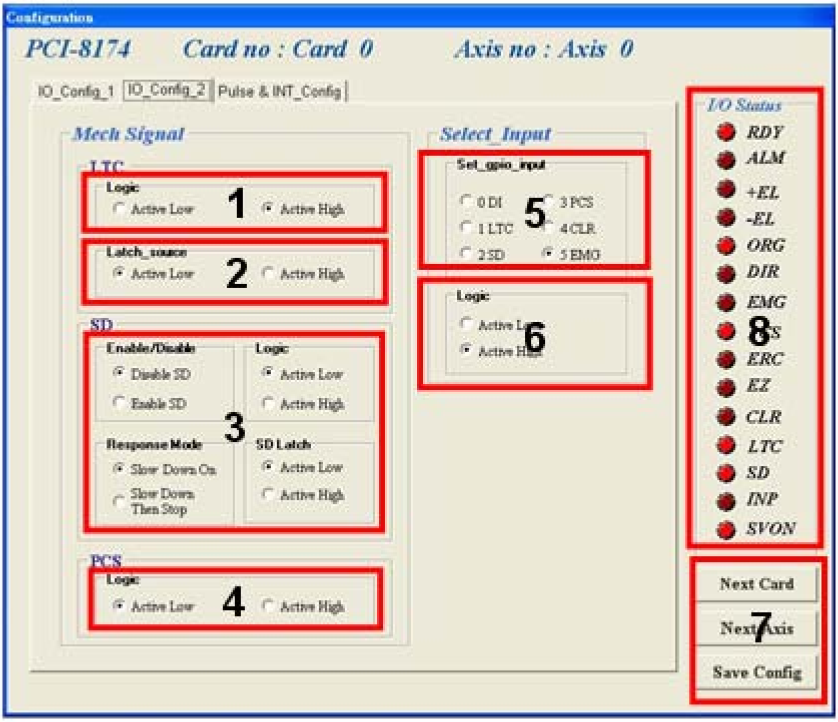
6 EZ
- Logic: [✓ Active Low] Active High

8 I/O Status: ● RDY ● ALM ● +EL ● -EL ● ORG ● DIR ● EMG ● PCS ● ERC ● EZ ● CLR ● LTC ● SD ● INP ● SVON

7 Next Card | Next Axis | Save Config

1. **ALM Logic and Response mode**: Select logic and response modes of ALM signal. The related function call is *_8174_set_alm*().
2. **INP Logic and Enable/Disable selection**: Select logic, and Enable/ Disable the INP signal. The related function call is *_8174_set_inp*()
3. **ERC Logic, Active timing and ERC mode**: Select the Logic, Active timing and mode of the ERC signal. The related function call is *_8174_set_erc*().
4. **EL Response mode**: Select the response mode of the EL signal. The related function call is *_8174_set_limit_logic*().
5. **ORG Logic**: Select the logic of the ORG signal. The related function call is *_8174_set_home_config*().
6. **EZ Logic**: Select the logic of the EZ signal. The related function call is *_8174_set_home_config*().
7. **Buttons**:
   - ▷ **Next Card**: Change operating card.
   - ▷ **Next Axis**: Change operating axis.
   - ▷ **Save Config**: Save current configuration to 8174.ini and 8174MC.ini.
8. **I/O Status**: The status of motion I/O. Light-On means Active, while Light-Off indicates inactive. The related function is *_8174_get_io_status*

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


Configuration

PCI-8174 Card no : Card 0 Axis no : Axis 0

IO_Config_1 | IO_Config_2 | Pulse & INT_Config

Mech Signal

LTC
1 Logic: Active Low [✓ Active High]
2 Latch_source: [✓ Active Low] Active High

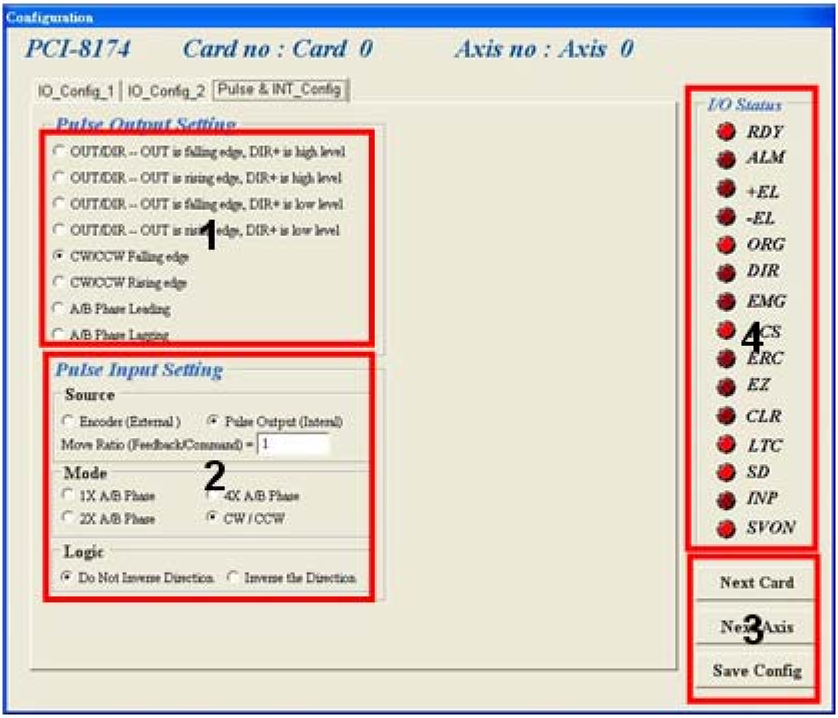
SD
3 Enable/Disable: [✓ Disable SD] Enable SD
Logic: [✓ Active Low] Active High
Response Mode: [✓ Slow Down On] Slow Down Then Stop
SD Latch: [✓ Active Low] Active High

PCS
4 Logic: [✓ Active Low] Active High

Select_Input
5 Sel_gpio_input: 0 DI, 1 LTC, 2 SD, 3 PCS, 4 CLR, [✓ 5 EMG]
6 Logic: Active Low [✓ Active High]

1. **LTC Logic**: Select the logic of the LTC signal. The related function call is *_8174_set_ltc_logic*().
2. **LTC latch_source**: Select the logic of the latch_source signal. The related function call is *_8174_set_latch_source*().
3. **SD Configuration**: Configure the SD signal. The related function call is *_8174_set_sd*().
4. **PCS Logic**: Select the logic of the SelectNo signal. The related function call is *_8174_set_pcs_logic*().
5. **Set gpio input**: Select the configurations of the gpio input. The related function call is *_8174_set_gpio_input_function*.
6. **Gpio Logic**: Select the logic of the gpio. The related function call is *_8174_set_gpio_input_function*.
7. **Buttons**:
   - ▷ **Next Card**: Change operating card.
   - ▷ **Next Axis**: Change operating axis.
   - ▷ **Save Config**: Save current configuration to 8174.ini And 8174MC.ini.
8. **I/O Status**: The status of motion I/O. Light-On means Active, while Light-Off indicates inactive. The related function is *_8174_get_io_status*

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



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
- A/B Phase Leading
- A/B Phase Lagging

Pulse Input Setting

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

Move Ratio (Feedback/Command) = 1

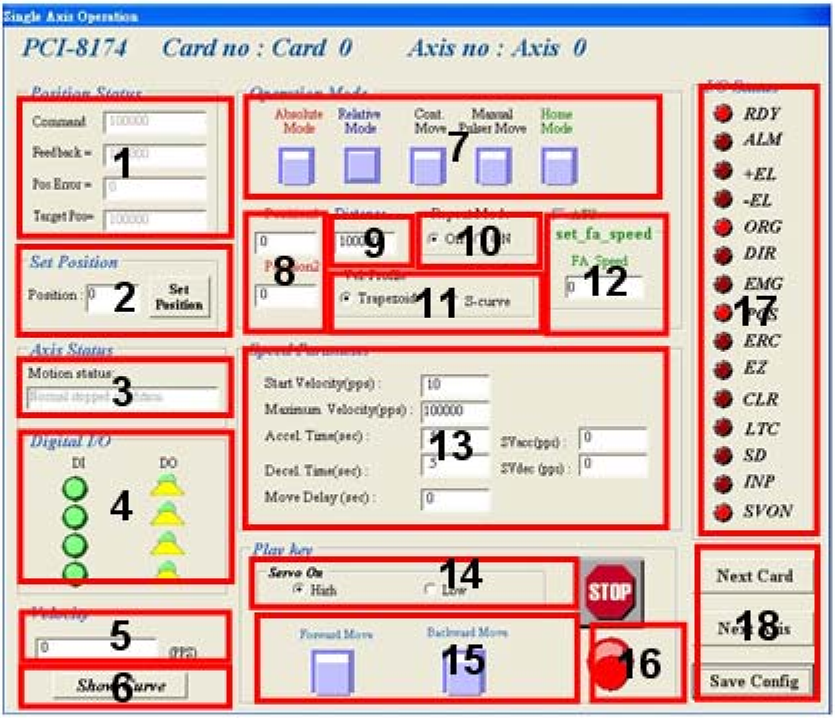
Mode: 1X A/B Phase; 4X A/B Phase; 2X A/B Phase; [✓ CW / CCW]

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

1. **Pulse Output Mode**: Select the output mode of the pulse signal (OUT/ DIR). The related function call is *_8174_set_pls_outmode*().
2. **Pulse Input**: Sets the configurations of the Pulse input signal(EA/EB). The related function calls are *_8174_set_pls_iptmode*(), *_8174_set_feedback_src*().
3. **Buttons**:
   - ▷ **Next Card**: Change operating card.
   - ▷ **Next Axis**: Change operating axis.
   - ▷ **Save Config**: Save current configuration to 8174.ini And 8174MC.ini.
4. **I/O Status**: The status of motion I/O. Light-On means Active, while Light-Off indicates inactive. The related function is *_8174_get_io_status.*

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-8174 Card no : Card 0 Axis no : Axis 0

Position Status

Command: 100000

Feedback = 100000

Pos Error = 0

Target Pos= 100000

Set Position

Position : 0

Motion status: Normal stopped condition

Start Velocity(pps) : 10

Maximum Velocity(pps) : 100000

Accel. Time(sec) :

Decel. Time(sec) : 3

Move Delay (sec) : 0

SVacc(pps) : 0

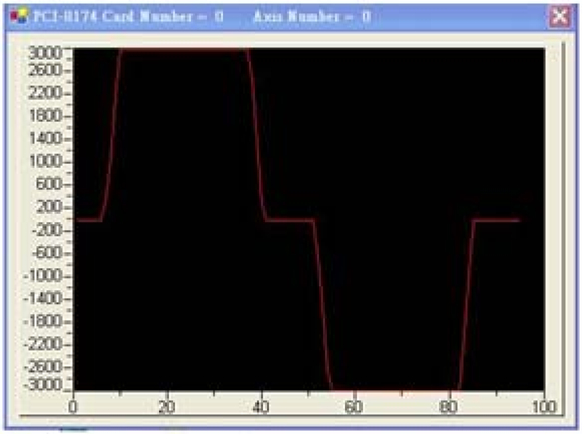
SVdec (pps) : 0

Servo On: [✓ High] Low

Vel. Profile: [✓ Trapezoid] S-curve

1. **Position**:
   - ▷ **Command**: displays the value of the command counter. The related function is *_8174_get_command*().
   - ▷ **Feedback**: displays the value of the feedback position counter. The related function is *_8174_get_position*()
   - ▷ **Pos Error**: displays the value of the position error counter. The related function is *_8174_get_error_counter*().
   - ▷ **Target Pos**: displays the value of the target position recorder. The related function is *_8174_get_target_pos*().
2. **Position Reset**: clicking this button will set all positioning counters to a specified value. The related functions are:
   - ▷ *_8174_set_position*(),
   - ▷ *_8174_set_command*(),
   - ▷ *_8174_reset_error_counter*()
   - ▷ *_8174_reset_target_pos*()
3. **Motion Status**: Displays the returned value of the *_8174_motion_done* function. The related function is *_8174_motion_done*().
4. **INT Status**:
   - ▷ **int_factor bit no**: Set int_factor bit.
   - ▷ **Normal INT:** display of Normal INT status. The related function is *_8174_wait_motion_interrupt* ().
   - ▷ **Error INT**: display of Error INT status. The related function is *_8174_wait_error_interrupt* ().
   - ▷ **GPIO INT**: display of GPIO INT status. The related function is *_8174_wait_gpio_interrupt* ().
5. **Velocity**: The absolute value of velocity in units of PPS. The related function is *_8174_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.



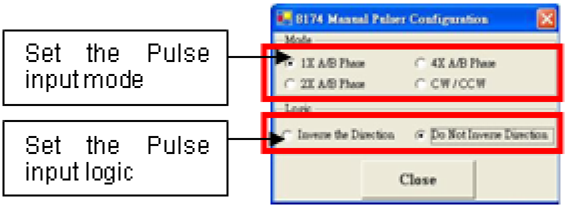
7. **Operation Mode**: Select operation mode.

  ▷ **Absolute Mode**: "Position1" and "position2" will be used as absolution target positions for motion. The related functions are *_8174_start_ta_move*(), *_8174_start_sa_move*().

  ▷ **Relative Mode**: "Distance" will be used as relative displacement for motion. The related function is *_8174_start_tr_move*(), *_8174_start_sr_move*().

  ▷ **Cont. Move**: Velocity motion mode. The related function is *_8174_tv_move*(), *_8174_start_sv_move*().

  ▷ **Manual Pulser Move**: Manual Pulse motion. Clicking this button will invoke the manual pulse configuration window.



- ▷ **Home Mode**: Home return motion. Clicking this button will invoke the home move configuration window. The related function is *_8174_set_home_config*(). *If the check box "ATU" is checked, it will execute auto homing when motion starts.*

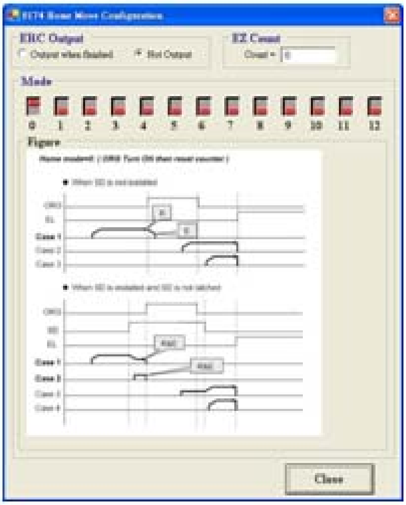
  ERC Output: Select if the ERC signal will be sent when home move completes.

  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.

  Close: Click this button close this window.


8174 Home Move Configuration

ERC Output: Output when finished [✓ Not Output]

EZ Count: Count = 0

Mode: 0 1 2 3 4 5 6 7 8 9 10 11 12

Figure

Home mode=0 (ORG Turn ON then reset counter)
When EZ is not enabled
ORG
EL
Case 1
Case 2
Case 3
When EZ is enabled and EZ is not latched
ORG
EZ
EL
Case 1
Case 2
Case 3
Case 4
Close

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 *_8174_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 8174 to delay for a specified time before it continues to the next motion.

14. **Servo On**: Set the SVON signal output status. The related function is _8174_set_servo().

15. **Play Key**:

Left play button: Clicking this button will cause the 8174 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 8174 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."

16. **Stop Button**: Clicking this button will cause the 8174 to decelerate and stop. The deceleration time is defined in "Decel. Time." The related function is _8174_sd_stop().

17. **I/O Status**: The status of motion I/O. Light-On means Active, while Light-Off indicates inactive. The related function is _8174_get_io_status().

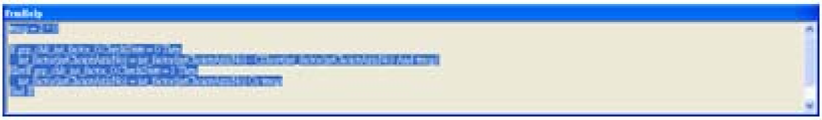
18. **Buttons**:

- ▷ **Next Card**: Change operating card.
- ▷ **Next Axis**: Change operating axis.

- ▷ **Save Config**: Save current configuration to 8174.ini And 8174MC.ini.
- ▷ **Close**: Close the menu.

5.3.6 Help Menu

In this menu, click the right mouse button to show Help Information.


[illegible]

6 Function Library

This chapter describes the supporting software for the PCI-8174 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_8174.h manually.

6.1 List of Functions

System & Initialization, Section 6.3

Function Name	Description
_8174_initial	Card initialization
_8174_close	Card Close
_8174_get_version	Check the hardware and software version

Pulse Input/Output Configuration, Section 6.4

Function Name	Description
_8174_set_pls_outmode	Set pulse command output mode
_8174_set_pls_iptmode	Set encoder input mode
_8174_set_feedback_src	Set counter input source

Velocity mode motion, Section 6.5

Function Name	Description
_8174_tv_move	Accelerate an axis to a constant velocity with trapezoidal profile
_8174_sv_move	Accelerate an axis to a constant velocity with S-curve profile
_8174_sd_stop	Decelerate to stop
_8174_emg_stop	Immediately stop
_8174_get_current_speed	Get current speed(pulse/sec)

Single Axis Position Mode, Section 6.6

Function Name	Description
_8174_start_tr_move	Begin a relative trapezoidal profile move
_8174_start_ta_move	Begin an absolute trapezoidal profile move
_8174_start_sr_move	Begin a relative S-curve profile move
_8174_start_sa_move	Begin an absolute S-curve profile move
_8174_set_move_ratio	Set the ratio of command pulse and feedback pulse.

Home Return Mode, Section 6.7

Function Name	Description
_8174_set_home_config	Set the home/index logic configuration
_8174_home_move	Begin a home return action
_8174_home_search	Perform an auto search home
_8174_set_fa_speed	Set the FA speed

Manual Pulser Motion, Section 6.8

Function Name	Description
_8174_set_pulser_iptmode	Set pulser input mode
_8174_disable_pulser_input	Disable the pulser input
_8174_pulser_vmove	Start pulser v move

Motion Status, Section 6.12

Function Name	Description
_8174_motion_done	Return the motion status

Motion Interface I/O, Section 6.10

Function Name	Description
_8174_set_servo	Set On-Off state of SVON signal
_8174_set_pcs_logic	Set PCS(Position Change Signal) signal's logic
_8174_set_pcs	Enable PCS for position override
_8174_set_inp	Set INP signal's logic and operating mode
_8174_set_alm	Set ALM signal's logic and operating mode
_8174_set_erc	Set ERC signal's logic and timing
_8174_set_erc_out	Output an ERC signal
_8174_set_sd	Set SD signal's logic and operating mode
_8174_enable_sd	Enable SD signal
_8174_set_limit_logic	Set EL signal's logic
_8174_set_limit_mode	Set EL operating mode
_8174_get_io_status	Get all the motion I/O status of 8174

Position Control and Counters, Section 6.11

Function Name	Description
_8174_get_position	Get the value of the feedback position counter
_8174_set_position	Set the feedback position counter
_8174_get_command	Get the value of the command position counter
_8174_set_command	Set the command position counter
_8174_get_error_counter	Get the value of the position error counter
_8174_reset_error_counter	Reset the position error counter
_8174_get_target_pos	Get the value of the target position recorder
_8174_reset_target_pos	Reset target position recorder

Position Compare and Latch, Section 6.12

Function Name	Description
_8174_set_latch_source	Set the latch timing for a counter
_8174_set_ltc_logic	Set the LTC signal's logic
_8174_get_latch_data	Get the latch data

General-Purpose Input/Output, Section 6.13

Function Name	Description
_8174_set_gpio_output	Set digital output
_8174_get_gpio_output	Get digital output
_8174_get_gpio_input	Get digital input
_8174_set_gpio_input_function	Set the signal types to any digital inputs

Soft Limit, Section 6.14

Function Name	Description
_8174_disable_soft_limit	Disable soft limit function
_8174_enable_soft_limit	Enable soft limit function
_8174_set_soft_limit	Set the soft limits

DSP Access Functions 001 for Trigger Applications,

Section 6.15

Function Name	Description
_8174_get_dsp_version	Get DSP Version & CPLD on DB board version
_8174_get_dsp_class_id	Get the Class ID
_8174_get_encoder	Get Encoder Value
_8174_clear_encoder	Reset Encoder Counter
_8174_set_encoder_input_mode	Set Encoder Pulse Input Mode
_8174_choose_trigger_type	Set trigger type for choosing linear or table trigger
_8174_choose_trigger_form	Set trigger form
_8174_set_pulse_cmp_output_selection	Set pin “OUT/DIR 1~2” to be as pulse output or CMP output
_8174_set_trigger_logic	Set the CMP signal’s logic
_8174_set_comparator_method	Set Trigger Comparator Method
_8174_set_comparator_value	Set Single Trigger Comparator Value
_8174_get_comparator_value	Get Single Trigger Comparator Value
_8174_set_linear_trigger_interval	Set Linear Trigger Interval
_8174_get_linear_trigger_interval	Get Linear Trigger Interval
_8174_set_linear_trigger_max_count	Set Linear Trigger Maximum Count
_8174_get_linear_trigger_max_count	Get Linear Trigger Maximum Count
_8174_get_triggered_count	Get Current Triggered Count
_8174_clear_triggered_count	Clear Current Triggered Count
_8174_build_compare_table	Set data array for table trigger
_8174_get_table_size	Get table size

6.2 C/C++ Programming Library

This section details all the functions. The function prototypes and some common data types are declared in **pci_8174.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

Table 6-1: Data type definitions

The functions of the PCI-8174 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. **`_8174_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_8174_initial()`**.

6.3 System and Initialization

@ Name

_8174_initial – Card initialization

_8174_close – Card close

_8174_get_version – Check hardware and software version information

@ Description

_8174_initial:

This function is used to initialize an 8174 card without assigning the hardware resources. All 8174 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 automatically.

_8174_close:

This function is used to close 8174 card and release its resources, which should be called at the end of your applications.

_8174_get_version:

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

@ Syntax

C/C++ (Windows 2000/XP)

```
I16 _8174_initial(U16 *CardID_InBit, I16
     Manual_ID, I16 Class_ID);
I16 _8174_close(void);
I16 _8174_get_version(I16 card_id, I16
     *firmware_ver, I32 *driver_ver, I32
     *dll_ver);
```

Visual Basic 6 (Windows 2000/XP)

```
B_8174_initial(CardID_InBit As Integer, ByVal
     Manual_ID As Integer, ByVal Class_ID As
     Integer) As Integer
B_8174_close() As Integer
B_8174_get_version(ByVal card_id As Integer,
     firmware_ver As Integer, driver_ver As Long,
     dll_ver As Long) As Integer
```

@ Argument

`CardID_InBit`:

`Class_ID`: represent different applications.

0x1999: for trigger function

`Manual_ID`: Enable the On board dip switch (SW1) to decide the Card ID

The CardID could be decided by:

0: the sequence of PCI slot.

1: on board DIP switch (SW1).

`card_id`: Specify the PCI-8174 card index. The card_id could be decided by DIP switch (SW1) or depend on slot sequence. Please refer to **`_8174_initial`**().

`firmware_ver`: The current firmware version.

`driver_ver`: The current device driver version.

`dll_ver`: The current DLL library version.

`old_secu_code`: Old security code.

`new_secu_code`: New security code.

`secu_code`: security code.

6.4 Pulse Input/Output Configuration

@ Name

_8174_set_pls_iptmode – Set the configuration for feedback pulse input.

_8174_set_pls_outmode – Set the configuration for pulse command output.

_8174_set_feedback_src – Enable/Disable the external feedback pulse input

@ Description

_8174_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 **_8174_set_feedback_src**() function is enabled.

_8174_set_pls_outmode:

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

_8174_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 **_8174_set_pls_iptmode**() function. Else, the counter will count the command pulse output.

@ Syntax

C/C++ (Windows 2000/XP)

```
I16 _8174_set_pls_iptmode(I16 AxisNo, I16
     pls_iptmode, I16 pls_logic);
I16 _8174_set_pls_outmode(I16 AxisNo, I16
     pls_outmode);
I16 _8174_set_feedback_src(I16 AxisNo, I16 Src);
```

Visual Basic6 (Windows 2000/XP)

```
B_8174_set_pls_iptmode(ByVal AxisNo As Integer,
     ByVal pls_iptmode As Integer, ByVal
     pls_logic As Integer) As Integer
B_8174_set_pls_outmode(ByVal AxisNo As Integer,
     ByVal pls_outmode As Integer) As Integer
B_8174_set_feedback_src(ByVal AxisNo As Integer,
     ByVal Src As Integer) As Integer
```

@ Argument

`AxisNo`: Axis number designated to configure the pulse input/output.

card_id	Physical axis	AxisNo
0	0	0
	1	1
	2	2
	3	3
1	0	4
	1	5
	…	…

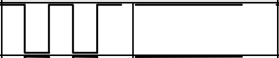
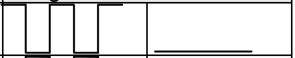
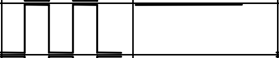
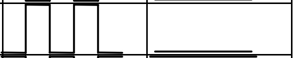
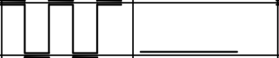
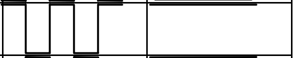
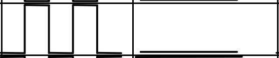
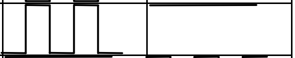
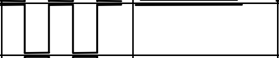
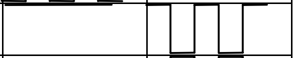
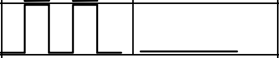
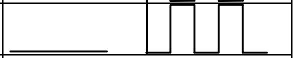
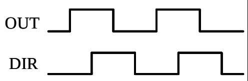
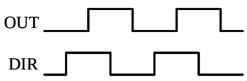
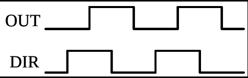
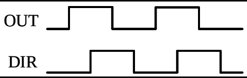
`pls_iptmode`: Encoder feedback pulse input mode setting (EA/EB signals).

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	Type	Positive Direction	Negative Direction
0	OUT/DIR		
1	OUT/DIR		
2	OUT/DIR		
3	OUT/DIR		
4	CW / CCW		
5	CW / CCW		
6	AB		
7	AB		

`Src`: Counter source

Value	Meaning
0	External signal feedback
1	Command pulse

6.5 Velocity mode motion

@ Name

_8174_tv_move – Accelerate an axis to a constant velocity with trapezoidal profile

_8174_sv_move – Accelerate an axis to a constant velocity with S-curve profile

_8174_emg_stop – Immediately stop

_8174_sd_stop – Decelerate to stop

_8174_get_current_speed – Get current speed

@ Description

_8174_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.

_8174_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.

_8174_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.

_8174_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.

`_8174_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.

@ Syntax

C/C++ (Windows 2000/XP)

```
I16 _8174_tv_move(I16 AxisNo, F64 StrVel, F64
     MaxVel, F64 Tacc);
I16 _8174_sv_move(I16 AxisNo, F64 StrVel, F64
     MaxVel, F64 Tacc, F64 SVacc);
I16 _8174_emg_stop(I16 AxisNo);
I16 _8174_sd_stop(I16 AxisNo, F64 Tdec);
I16 _8174_get_current_speed(I16 AxisNo, F64
     *speed)
```

Visual Basic6 (Windows 2000/XP)

```
B_8174_tv_move(ByVal AxisNo As Integer, ByVal
     StrVel As Double, ByVal MaxVel As Double,
     ByVal Tacc As Double) As Integer
B_8174_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_8174_emg_stop(ByVal AxisNo As Integer) As
     Integer
B_8174_sd_stop(ByVal AxisNo As Integer, ByVal
     Tdec As Double) As Integer
B_8174_get_current_speed(ByVal AxisNo As Integer,
     ByRef Speed As Double) As Integer
```

@ Argument

AxisNo: Axis number designated to move or stop.

card_id	Physical axis	AxisNo
0	0	0
	1	1
	2	2
	3	3
1	0	4
	1	5
	…	…

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

`_8174_start_tr_move` – Begin a relative trapezoidal profile move

`_8174_start_ta_move` – Begin an absolute trapezoidal profile move

`_8174_start_sr_move` – Begin a relative S-curve profile move

`_8174_start_sa_move` – Begin an absolute S-curve profile move

`_8174_set_move_ratio` – Set the ration of command pulse and feedback pulse

@ 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.

`_8174_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 **trapezoidal** 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.

_8174_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.

_8174_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.

_8174_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.

_8174_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.

@ Syntax

C/C++ (Windows 2000/XP)

```
I16 _8174_start_tr_move(I16 AxisNo, F64 Dist, F64
      StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);
I16 _8174_start_ta_move(I16 AxisNo, F64 Pos, F64
      StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);
```

```
I16 _8174_start_sr_move(I16 AxisNo, F64 Dist, F64
     StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64
     SVacc, F64 SVdec);
I16 _8174_start_sa_move(I16 AxisNo, F64 Pos, F64
     StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64
     SVacc, F64 SVdec);
I16 _8174_set_move_ratio(I16 AxisNo, F64
     move_ratio);
```

Visual Basic6 (Windows 2000/XP)

```
B_8174_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_8174_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_8174_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_8174_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_8174_set_move_ratio(ByVal AxisNo As Integer,
     ByVal move_ratio As Double) As Integer
```

@ Argument

AxisNo: Axis number designated to move or stop.

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

card_id	Physical axis	AxisNo
1	0	4
	1	5
	...	...

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 2.4.4

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 Home Return Mode

@ Name

_8174_set_home_config – Set the configuration for home return move motion

_8174_home_move – Perform a home return move.

_8174_home_search – Perform an auto search home

@ Description

_8174_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.

_8174_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 description, see section 4.2.10

_8174_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 description, see section 4.2.11

@ Syntax

C/C++ (Windows 2000/XP)

```
I16 _8174_set_home_config(I16 AxisNo, I16
        home_mode, I16 org_logic, I16 ez_logic, I16
        ez_count, I16 erc_out);
I16 _8174_home_move(I16 AxisNo, F64 StrVel, F64
        MaxVel, F64 Tacc);
I16 _8174_home_search(I16 AxisNo, F64 StrVel, F64
        MaxVel, F64 Tacc, F64 ORGOffset);
```

Visual Basic (Windows 2000/XP)

```
B_8174_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_8174_home_move(ByVal AxisNo As Integer, ByVal
        StrVel As Double, ByVal MaxVel As Double,
        ByVal Tacc As Double) As Integer
B_8174_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 stop.

card_id	Physical axis	AxisNo
0	0	0
	1	1
	2	2
	3	3
1	0	4
	1	5
	…	…

`home_mode`: Stopping modes for home return, This value is between 0 to 12. Please refer to the operation theory 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 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 signal (Unit: pulse)

6.8 Manual Pulser Motion

@ Name

_8174_disable_pulser_input – Disable the pulser input

_8174_pulser_vmove – Manual pulser v_move

_8174_set_pulser_iptmode – Set the input signal modes of pulser

@ Description

_8174_disable_pulser_input

This function is used to set the pulser input disable or enable.

_8174_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 **_8174_disable_pulser_input** function disables the pulser.

_8174_set_pulser_iptmode

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

@ Syntax

C/C++ (Windows 2000/XP)

```
I16 _8174_disable_pulser_input(I16 AxisNo, U16
     Disable);
I16 _8174_pulser_vmove(I16 AxisNo, F64
     SpeedLimit);
I16 _8174_set_pulser_iptmode(I16 AxisNo, I16
     InputMode, I16 Inverse);
```

Visual Basic (Windows 2000/XP)

```
B_8174_disable_pulser_input(ByVal AxisNo As
     Integer, ByVal Disable As Integer) As
     Integer
```

```
B_8174_pulser_vmove(ByVal AxisNo As Integer,
     ByVal SpeedLimit As Double) As Integer
B_8174_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 stop.

card_id	Physical axis	AxisNo
0	0	0
	1	1
	2	2
	3	3
1	0	4
	1	5
	…	…

`Disable`: Disable pulser input.

Disable = 1, disable pulser

Disable = 0, enable pulser

`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

6.9 Motion Status

@ Name

`_8174_motion_done` – Return the motion status

@ Description

`_8174_motion_done`:

Return the motion status of the 8174. 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)

@ Syntax

C/C++ (Windows 2000/XP)

```
I16 _8174_motion_done(I16 AxisNo)
```

Visual Basic (Windows 2000/XP)

```
B_8174_motion_done(ByVal AxisNo As Integer) As
     Integer
```

@ Argument

AxisNo: Axis number designated to move or stop.

card_id	Physical axis	AxisNo
0	0	0
	1	1
	2	2
	3	3
1	0	4
	1	5
	…	…

6.10 Motion Interface I/O

@ Name

_8174_set_servo – Set the ON-OFF state of the SVON signal

_8174_set_pcs_logic – Set the logic of PCS signal

_8174_set_pcs – Enable the PCS for position override

_8174_set_inp – Set the logic of INP signal and operating mode

_8174_set_alm – Set the logic of ALM signal and operating mode

_8174_set_erc – Set the logic of ERC signal and operating mode

_8174_set_erc_out – Output an ERC signal

_8174_set_sd – Set the logic SD signal and operating mode

_8174_enable_sd – Enable SD signal

_8174_set_limit_logic – Set the logic of PEL/MEL signal

_8174_set_limit_mode – Set PEL/MEL operating mode

_8174_get_io_status –Get all the motion I/O statuses of each 8174

@ Description

_8174_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.

_8174_set_pcs_logic:

Set the active logic of the PCS signal input

_8174_set_pcs:

Enable the position override when input signal PCS is turn ON. The PCS terminal status can be monitored by the "**_8174_get_io_status**" function.

_8174_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.

_8174_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.

_8174_set_erc:

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

_8174_set_erc_out:

This function is used to output the ERC signal manually.

_8174_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 **_8174_enable_sd**. It is disabled by default

_8174_enable_sd:

Enable the SD signal input. Default setting is default.

_8174_set_limit_logic:

Set the EL logic, normal open or normal closed.

_8174_set_limit_mode:

Set the reacting modes of the EL signal.

_8174_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 _8174_set_servo(I16 AxisNo, I16 on_off);
I16 _8174_set_pcs_logic(I16 AxisNo, I16
     pcs_logic);
I16 _8174_set_pcs(I16 AxisNo, I16 enable);
I16 _8174_set_inp(I16 AxisNo, I16 inp_enable, I16
     inp_logic);
I16 _8174_set_alm(I16 AxisNo, I16 alm_logic, I16
     alm_mode);
I16 _8174_set_erc(I16 AxisNo, I16 erc_logic, I16
     erc_pulse_width, I16 erc_mode);
I16 _8174_set_erc_out(I16 AxisNo);
I16 _8174_set_sd(I16 AxisNo, I16 sd_logic, I16
     sd_latch, I16 sd_mode);
I16 _8174_enable_sd(I16 AxisNo, I16 enable);
I16 _8174_set_limit_logic(I16 AxisNo, U16 Logic
     );
```

```
I16 _8174_set_limit_mode(I16 AxisNo, I16
     limit_mode);
I16 _8174_get_io_status(I16 AxisNo, U16 *io_sts);
```

Visual Basic (Windows 2000/XP)

```
B_8174_set_servo(ByVal AxisNo As Integer, ByVal
     on_off As Integer) As Integer
B_8174_set_pcs_logic(ByVal AxisNo As Integer,
     ByVal pcs_logic As Integer) As Integer
B_8174_set_pcs(ByVal AxisNo As Integer, ByVal
     enable As Integer)As Integer
B_8174_set_inp(ByVal AxisNo As Integer, ByVal
     inp_enable As Integer, ByVal inp_logic As
     Integer) As Integer
B_8174_set_alm(ByVal AxisNo As Integer, ByVal
     alm_logic As Integer, ByVal alm_mode As
     Integer) As Integer
B_8174_set_erc(ByVal AxisNo As Integer, ByVal
     erc_logic As Integer, ByVal erc_pulse_width
     As Integer, ByVal erc_mode As Integer) As
     Integer
B_8174_set_erc_out(ByVal AxisNo As Integer) As
     Integer
B_8174_set_sd(ByVal AxisNo As Integer, ByVal
     sd_logic As Integer, ByVal sd_latch As
     Integer, ByVal sd_mode As Integer) As
     Integer
B_8174_enable_sd(ByVal AxisNo As Integer, ByVal
     Enable As Integer) As Integer
B_8174_set_limit_logic(ByVal AxisNo As Integer,
     ByVal Logic As Integer) As Integer
B_8174_set_limit_mode(ByVal AxisNo As Integer,
     ByVal limit_mode As Integer) As Integer
I16 _8174_get_io_status(ByVal AxisNo As Integer,
     io_sts As Integer) As Integer
```

@ Argument

`AxisNo`: Axis number designated to move or stop.

card_id	Physical axis	AxisNo
0	0	0
	1	1
	2	2
	3	3
1	0	4
	1	5
	…	…

`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

`enable`: enable or disable

Value	Meaning
0	Disable
1	Enable

`targetCounterInBit`: Enable/Disable clear target counter in bit

Value	Meaning
Bit	Description
0	Reset command counter when CLR input turns ON
1	Reset position counter when CLR input turns ON
2	Reset error counter when CLR input turns ON
3	Reset general purpose counter when CLR input turns ON

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

erc_mode:

Value	Meaning
0	Disable
1	Output ERC when stopped by EL, ALM, or EMG input
2	Output ERC when complete home return
3	Both 1 and 2

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 immediately
1	Slow down then stop

`*io_sts`: I/O status. Please refer to 6.11 function description.

6.11 Position Control and Counters

@ Name

_8174_get_position – Get the value of feedback position counter

_8174_set_position – Set the feedback position counter

_8174_get_command – Get the value of command position counter

_8174_set_command – Set the command position counter

_8174_get_error_counter – Get the value of position error counter

_8174_reset_error_counter – Reset the position error counter

_8174_get_target_pos – Get the value of target position recorder

_8174_reset_target_pos – Reset target position recorder

@ Description

_8174_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 **_8174_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 **_8174_set_feedback_src**() to be external EA/EB or internal pulse output of 8174.

_8174_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.

_8174_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 8174.

_8174_set_command:

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

_8174_get_error_counter:

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

_8174_reset_error_counter:

This function is used to clear the position error counter.

_8174_get_target_pos:

This function is used to read the value of the target position recorder. The target position recorder is maintained by the 8174 software driver. It records the position to settle down for current running motion.

_8174_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 **_8174_set_position**().

@ Syntax

C/C++ (Windows 2000/XP)

```
I16 _8174_get_position(I16 AxisNo, F64 *Pos);
I16 _8174_set_position(I16 AxisNo, F64 Pos);
I16 _8174_get_command(I16 AxisNo, I32 *Command);
I16 _8174_set_command(I16 AxisNo, I32 Command);
I16 _8174_get_error_counter(I16 AxisNo, I16
     *error);
I16 _8174_reset_error_counter(I16 AxisNo);
I16 _8174_get_target_pos(I16 AxisNo, F64 *T_pos);
I16 _8174_reset_target_pos(I16 AxisNo, F64
     T_pos);
```

Visual Basic (Windows 2000/XP)

```
B_8174_get_position(ByVal AxisNo As Integer, Pos
     As Double) As Integer
B_8174_set_position(ByVal AxisNo As Integer,
     ByVal Pos As Double) As Integer
B_8174_get_command(ByVal AxisNo As Integer, Cmd
     As Long) As Integer
B_8174_set_command(ByVal AxisNo As Integer, ByVal
     Cmd As Long) As Integer
B_8174_get_error_counter(ByVal AxisNo As Integer,
     ByRef error As Integer) As Integer
B_8174_reset_error_counter(ByVal AxisNo As
     Integer) As Integer
B_8174_reset_target_pos(ByVal AxisNo As Integer,
     ByVal Pos As Double) As Integer
B_8174_get_target_pos(ByVal AxisNo As Integer,
     ByRef Pos As Double) As Integer
```

@ Argument

AxisNo: Axis number designated to move or stop.

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

card_id	Physical axis	AxisNo
1	0	4
	1	5
	…	…

Pos, ***Pos**: Feedback position counter value, (_8174_get/set_position)

range: -134217728 to 134217727

Cmd, ***Cmd**: Command position counter value,

range: -134217728 to 134217727

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

range: -32768 to 32767

TargetPos, ***TargetPos**: Target position recorder value,

range: -134217728 to 134217727

6.12 Position Compare and Latch

@ Name

`_8174_set_latch_source` – Set the latch timing for a counter

`_8174_set_ltc_logic` – Set the logic of LTC signal

`_8174_get_latch_data` – Get the latch data from counter

@ Description

`_8174_set_latch_source`:

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

`_8174_set_ltc_logic`:

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

`_8174_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 _8174_set_latch_source(I16 AxisNo, I16
     LtcSrc);
16 _8174_set_ltc_logic(I16 AxisNo, I16 LtcLogic);
I16 _8174_get_latch_data(I16 AxisNo, I16
     CounterNo, F64 *Pos);
```

Visual Basic (Windows 2000/XP)

```
B_8174_set_latch_source(ByVal AxisNo As Integer,
     ByVal LtcSrc As Integer) As Integer
B_8174_set_ltc_logic(ByVal AxisNo As Integer,
     ByVal StcLogic As Integer) As Integer
B_8174_get_latch_data(ByVal AxisNo As Integer,
     ByVal CounterNo As Integer, Pos As Double)
     As Integer
```

@ Argument

`AxisNo`: Axis number designated to move or stop.

card_id	Physical axis	AxisNo
0	0	0
	1	1
	2	2
	3	3
1	0	4
	1	5
	…	…

`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
0	No Compare (Disable)
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 (Position)

`CmpAction`:

Value	Meaning
0	No action
1	Stop immediately
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 (Position)

6.13 General-Purpose DIO

@ Name

`_8174_set_gpio_output` – Set digital output

`_8174_get_gpio_output` – Get digital output

`_8174_get_gpio_input` – Get digital input

`_8174_set_gpio_input_function` – Set the signal types for any digital inputs

@ Description

`_8174_set_gpio_output`:

The PCI-8174 has 4 digital output channels. By this function, user could control the digital outputs.

`_8174_get_gpio_output`:

This function is used to get the digital output status.

`_8174_get_gpio_input`:

PCI-8174 has 4 digital input channels. By this function, user can get the digital input status.

`_8174_set_gpio_input_function`:

PCI-8174 has 4 digital input channels. By this function, user can set one of several input signals to any specific DI channels. Those signals include LTCn, SDn, PCSn, CLRn, EMG. (The index word n mean axis index)

@ Syntax

C/C++ (Windows 2000/XP)

```
I16 _8174_set_gpio_output(I16 card_id, I16
     DoValue);
I16 _8174_get_gpio_output(I16 card_id, I16 *
     DoValue);
I16 _8174_get_gpio_input(I16 card_id, I16 *
     DiValue);
I16 _8174_set_gpio_input_function(I16 card_id,
     I16 Channel, I16Select, I16 Logic);
```

Visual Basic (Windows 2000/XP)

```
B_8174_set_gpio_output(ByVal card_id As Integer,
     ByVal DoValue As Integer) As Integer
B_8174_get_gpio_output(ByVal card_id As Integer,
     DoValue As Integer) As Integer
B_8174_get_gpio_input(ByVal card_id As Integer,
     DiValue As Integer) As Integer
B_8174_set_gpio_input_function(ByVal card_id As
     Integer, ByVal Channel As Integer, ByVal
     Select As Integer, ByVal Logic As Integer)As
     Integer
```

@ Argument

card_id: Specify the PCI-8174 card index. The card_id could be decided by DIP switch (SW1) or depend on slot sequence. Please refer to **_8174_initial**().

DoValue, ***DoValue**: Digital output value. Bit 0-3: D_out 0-3.

***DiValue**: Digital input value, Bit 0-3: D_in 0-3

Channel: Digital channel DI0 - DI3

Select: signal types select

Value	Meaning
0	General DI (default)
1	LTC (active low)
2	SD (active low)
3	PCS (active low)
4	CLR (active low)
5	EMG (active low)

Logic: input signal logic

Value	Meaning
0	Not inverse (default)
1	Inverse

6.14 Soft Limit

@ Name

`_8174_disable_soft_limit` – Disable soft limit function

`_8174_enable_soft_limit` – Enable soft limit function

`_8174_set_soft_limit` – Set soft limit

@ Description

`_8174_disable_soft_limit`:

This function is used to disable the soft limit function.

`_8174_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.

`_8174_set_soft_limit`:

This function is used to set the soft limit value.

@ Syntax

C/C++ (Windows 2000/XP)

```
I16 _8174_disable_soft_limit(I16 AxisNo);
I16 _8174_enable_soft_limit(I16 AxisNo, I16
     Action);
I16 _8174_set_soft_limit(I16 AxisNo, I32
     PlusLimit, I32 MinusLimit);
```

Visual Basic (Windows 2000/XP)

B_8174_disable_soft_limit(ByVal AxisNo As Integer) As Integer

```
B_8174_enable_soft_limit(ByVal AxisNo As Integer,
     ByVal Action As Integer) As Integer
B_8174_set_soft_limit(ByVal AxisNo As Integer,
     ByVal PlusLimit As Long, ByVal MinusLimit As
     Long) As Integer
```

@ Argument

`AxisNo`: Axis number designated to move or stop.

card_id	Physical axis	AxisNo
0	0	0
	1	1
	2	2
	3	3
1	0	4
	1	5
	…	…

`Action`: The reacting method of soft limit

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

`PlusLimit`: Soft limit value, positive direction

`MinusLimit`: Soft limit value, negative direction

6.15 Class ID=0X1999 for Channel Trigger

@ Name

`_8174_get_dsp_version` – Get DSP version & CPLD on DB board version

`_8174_get_dsp_class_id` – Get the class ID

`_8174_get_encoder` – Get encoder value

`_8174_clear_encoder` – Reset encoder counter

`_8174_set_encoder_input_mode` – Set encoder pulse input mode

`_8174_choose_trigger_type` – set trigger type for choosing linear or table trigger

`_8174_choose_trigger_form` – set trigger form

`_8174_set_pulse_cmp_output_selection` – set pin “OUT/DIR 1~2” to be as pulse output or CMP output

`_8174_set_trigger_logic` – Set the CMP signal’s logic

`_8174_set_comparator_method` – Set trigger comparator method

`_8174_set_comparator_value` – Set single trigger comparator value

`_8174_get_comparator_value` – Get single trigger comparator value

`_8174_set_linear_trigger_interval` – Set linear trigger interval

`_8174_get_linear_trigger_interval` – Get linear trigger interval

`_8174_set_linear_trigger_max_count` – Set linear trigger maximum count

`_8174_get_linear_trigger_max_count` –Get linear trigger maximum count

`_8174_get_triggered_count` – Get linear triggered count

_8174_clear_triggered_count – Clear linear triggered count

_8174_build_compare_table – Set data array and sizes for table trigger

_8174_get_table_size – Get table size

@ Description

_8174_get_dsp_version:

Reads back DSP_Version and CPLD on DB board version.

_8174_get_dsp_class_id:

Reads back the class ID from DSP.

_8174_get_encoder:

This function is used to get the feedback position counter.

_8174_clear_encoder:

This function is used to clear the feedback position counter to the 0.

_8174_set_encoder_input_mode:

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 **_8174_set_feedback_src()** function is enabled.

_8174_set_comparator_value:

This function is used to set single trigger comparator value. Before using other trigger function, to use this function to set a first point. The linear trigger formula is, (First point) + (Interval) * (Max_count) = (Last point)

`_8174_get_comparator_value`:

This function is used to get current comparing data.

`_8174_set_comparator_method`:

This function is used to set the comparing method for the trigger comparator.

`_8174_set_linear_trigger_interval`:

This function is used to set linear trigger interval. The linear trigger formula is, (First point) + (Interval) * (Max_count) = (Last point)

`_8174_get_linear_trigger_interval`:

This function is used to get linear trigger interval.

`_8174_set_linear_trigger_max_count`:

This function is used to set maximum trigger count. The linear trigger formula is, (First point) + (Interval) * (Max_count) = (Last point)

`_8174_get_linear_trigger_max_count`:

This function is used to get maximum trigger count.

`_8174_get_triggered_count`:

This function is used to get current triggered count.

`_8174_clear_triggered_count`:

This function is used to clear current triggered count to 0.

`_8174_choose_trigger_type`:

Trigger type includes linear trigger and table trigger. This function is used to choose trigger type. The default is linear trigger.

`_8174_choose_trigger_form`:

This function is used to set trigger form. Four forms are provided. Detailed explanation is described as following. Please refer to parameter “trigger_from” in argument section.

`_8174_set_pulse_cmp_output_selection`:

By this function, user can set pin “OUT/DIR 1~2” to be as pulse output or CMP output. The default is pulse output.

`_8174_set_trigger_logic`:

This function is used to set the logic of CMP single.

`_8174_build_compare_table`:

This function is used to set data array for table trigger. The function stores those data to SDRAM and set the first data of array to comparator. To setup **`_8174_choose_trigger_form()`** and **`_8174_choose_trigger_type()`** before using this function. If a lot of points are stored to SDRAM, it can spend a little time.

`_8174_get_table_size`:

This function is used to get table size on SDRAM stored for table trigger.

@ Syntax

C/C++ (Windows 2000/XP)

```
I16 _8174_get_dsp_version(I16 card_id, I32
     *dsp_ver, I32 *build_date, U16
     *CPLD_vesion);
I16 _8174_get_dsp_class_id(I16 card_id);
I16 _8174_get_encoder(I16 card_id, I32
     *Encoder_Count);
I16 _8174_clear_encoder(I16 card_id);
I16 _8174_set_encoder_input_mode(I16 card_id, I16
     Input_mode, I16 logic);
I16 _8174_choose_trigger_type(I16 card_id, U16
     Trigger_Type);
I16 _8174_choose_trigger_form (I16 card_id, U16
     Trigger_Form);
I16 _8174_set_pulse_cmp_output_selection(I16
     CardId, I16 Channel, I16
     PulseCmpOutputSelection);
I16 _8174_set_trigger_logic(I16 AxisNo, I16
     Logic);
I16 _8174_set_comparator_method(I16 card_id, I16
     channel, I16 Comp_Method);
I16 _8174_set_comparator_value(I16 card_id, I16
     channel, I32 Comp_Value);
I16 _8174_get_comparator_value(I16 card_id, I16
     channel, I32 *Comp_Value);
```

```
I16 _8174_set_linear_trigger_interval(I16
     card_id, I16 channel, I32 Comp_Interval);
I16 _8174_get_linear_trigger_interval(I16
     card_id, I16 channel, I32 *Comp_Interval);
I16 _8174_set_linear_trigger_max_count(I16
     card_id, I16 channel, I32 Max_Count);
I16 _8174_get_linear_trigger_max_count(I16
     card_id, I16 channel, I32 *Max_Count);
I16 _8174_get_triggered_count(I16 card_id, I16
     channel, I32 *Triggered_Count);
I16 _8174_clear_triggered_count(I16 card_id, I16
     channel);
I16 _8174_build_compare_table(I16 card_id, I16
     channel, I32 *TableArray, U32 Table_Size);
I16 _8174_get_table_size(I16 card_id, I32
     *table_size);
```

Visual Basic (Windows 2000/XP)

```
B_8174_get_dsp_version(ByVal card_id As Integer,
     dsp_ver As Long, build_date As Long,
     CPLD_vesion As Integer) As Integer
B_8174_get_dsp_class_id(ByVal card_id As Integer)
     As Integer
B_8174_get_encoder(ByVal card_id As Integer,
     Encoder_Count As Long) As Integer
B_8174_clear_encoder(ByVal card_id As Integer) As
     Integer
B_8174_set_encoder_input_mode(ByVal card_id As
     Integer, ByVal Input_mode As Integer, ByVal
     logic As Integer) As Integer
B_8174_choose_trigger_type(ByVal card_id As
     Integer, ByVal trigger_Type As Integer) As
     Integer
B_8174_choose_trigger_form(ByVal card_id As
     Integer, ByVal trigger_form As Integer) As
     Integer
B_8174_set_pulse_cmp_output_selection(ByVal
     card_id As Integer, ByVal Channel As
     Integer, ByVal PulseCmpOutputSelection As
     Integer) As Integer
B_8174_set_trigger_logic(ByVal AxisNo As Integer,
     ByVal Logic As Integer) As Integer
```

```
B_8174_set_comparator_method(ByVal card_id As
     Integer, ByVal channel As Integer, ByVal
     Comp_Method As Integer) As Integer
B_8174_set_comparator_value(ByVal card_id As
     Integer, ByVal channel As Integer, ByVal
     Comp_Value As Long) As Integer
B_8174_get_comparator_value(ByVal card_id As
     Integer, ByVal channel As Integer,
     Comp_Value As Long) As Integer
B_8174_set_linear_trigger_interval(ByVal card_id
     As Integer, ByVal channel As Integer, ByVal
     Comp_Interval As Long) As Integer
B_8174_get_linear_trigger_interval(ByVal card_id
     As Integer, ByVal channel As Integer,
     Comp_Interval As Long) As Integer
B_8174_set_linear_trigger_max_count(ByVal
     card_id As Integer, ByVal channel As
     Integer, ByVal Max_Count As Long) As Integer
B_8174_get_linear_trigger_max_count(ByVal
     card_id As Integer, ByVal channel As
     Integer, Max_Count As Long) As Integer
B_8174_clear_triggered_count(ByVal card_id As
     Integer, ByVal channel As Integer) As
     Integer
B_8174_get_triggered_count(ByVal card_id As
     Integer, ByVal channel As Integer,
     Triggered_Count As Long) As Integer
B_8174_build_compare_table(ByVal card_id As
     Integer, ByVal channel As Integer,
     TableArray As Long, ByVal Table_Size As
     Long) As Integer
B_8174_get_table_size(ByVal card_id As Integer,
     table_size As Long) As Integer
```

@ Argument

`card_id`: The id of the card designated to perform.

`channel`: The channel designed to perform. Generally, the valid value is between 0-3.

`dsp_ver`: The current DSP version.

`build_date`: The current DSP_Build date.

CPLD_vesion: The version of CPLD on DB board.

Encoder_Count: Feedback position counter value.

- Range: -134217728~134217727.

Input_mode: 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

Comp_Value: Single trigger comparator value

Comp_Method: The comparing methods

Value	Meaning
0	No Compare (Disable)
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

Comp_Interval: Interval for linear trigger function

Max_Count: The maximum count for linear trigger function. The linear trigger formula is, (First point) + (Interval) * (Max_count) = (Last point).

Triggered_Count: The current triggered count is showed, when the comparator is triggered.

`trigger_Type`: Two trigger types are provided.

- 0: Linear trigger (default)
- 1: Table trigger

`trigger_form`: There are four choices to use Linear/Table function. When users set "trigger_form" to 0, channel 0 only can be used for trigger. 1800000 points can be used for table trigger. The default is 3, so all channels can be used for trigger.

trigger_form	Channel	Table Size Status
0	0	Max. 1800000 points
	1	Not used
	2	Not used
	3	Not used
1	0	Max. 900000 points
	1	Max. 900000 points
	2	Not used
	3	Not used
2	0	600000 points
	1	600000 points
	2	600000 points
	3	Not used
3	0	450000 points
	1	450000 points
	2	450000 points
	3	450000 points

`PulseCmpOutputSelection`: Set to pulse output or CMP output

- Bit 0~3: Pin OUT/DIR 1~2
- Bit 0: set OUT1+/- to be as pulse output or CMP OUTPUT.
- Bit 1: set DIR1+/- to be as pulse output or CMP OUTPUT.
- Bit 2: set OUT2+/- to be as pulse output or CMP OUTPUT.
- Bit 3: set DIR2+/- to be as pulse output or CMP OUTPUT.
  - Value in bit0~3: 0: As pulse output, 1: As CMP OUTPUT

Logic: logic of comparing trigger

Value	Meaning
0	Negative logic
1	Positive logic

***TableArray**: Data array of table trigger.

Table_Size: The numbers of Table Array. If table size isn't matching with numbers of table array, It will return error.

trigger_Type: Two trigger types are provided.

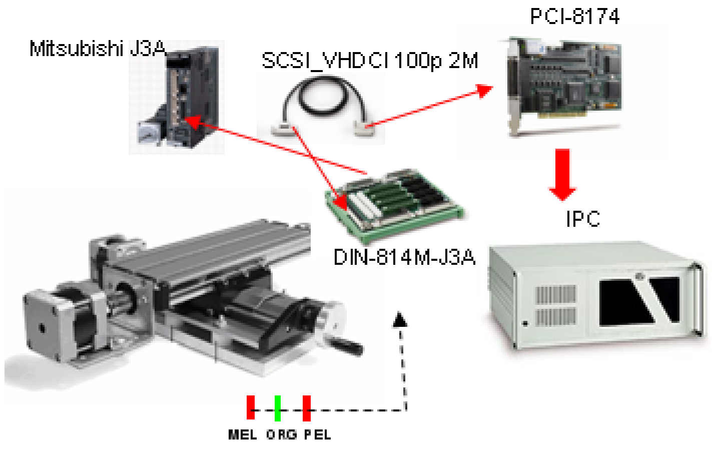
- 0: Linear trigger (default)
- 1: Table trigger

7 Connection Example

This chapter shows some connection examples between the PCI-8174 and servo drivers and stepping drivers.

7.1 General Description of Wiring

Main connection between the PCI-8174 and the pulse input servo driver or stepping driver. The following figure illustrates how to integrate the PCI-8174 and DIN-814M-J3A.



7.2 Terminal Board User Guide

Please refer the individual user guide of terminal board. The support terminal boards are as follows:

Mitsubishi J2 Super	DIN-814M
Mitsubishi J3A	DIN-814M-J3A
Yaskawa Sigma II	DIN-814Y
Panasonic MINAS A4	DIN-814P-A4

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. When sending in damaged products for repair, please attach an RMA application form which can be downloaded from: http://rma.adlinktech.com/policy/.
2. All ADLINK products come with a limited two-year warranty, one year for products bought in China:
   - ▶ The warranty period starts on the day the product is shipped from ADLINK's factory.
   - ▶ Peripherals and third-party products not manufactured by ADLINK will be covered by the original manufacturers' warranty.
   - ▶ For products containing storage devices (hard drives, flash cards, etc.), please back up your data before sending them for repair. ADLINK is not responsible for any loss of data.
   - ▶ Please ensure the use of properly licensed software with our systems. ADLINK does not condone the use of pirated software and will not service systems using such software. ADLINK will not be held legally responsible for products shipped with unlicensed software installed by the user.
   - ▶ For general repairs, please do not include peripheral accessories. If peripherals need to be included, be certain to specify which items you sent on the RMA Request & Confirmation Form. ADLINK is not responsible for items not listed on the RMA Request & Confirmation Form.

3. Our repair service is not covered by ADLINK's guarantee in the following situations:
   - Damage caused by not following instructions in the User's Manual.
   - Damage caused by carelessness on the user's part during product transportation.
   - Damage caused by fire, earthquakes, floods, lightening, pollution, other acts of God, and/or incorrect usage of voltage transformers.
   - Damage caused by unsuitable storage environments (i.e. high temperatures, high humidity, or volatile chemicals).
   - Damage caused by leakage of battery fluid during or after change of batteries by customer/user.
   - Damage from improper repair by unauthorized ADLINK technicians.
   - Products with altered and/or damaged serial numbers are not entitled to our service.
   - This warranty is not transferable or extendible.
   - Other categories not protected under our warranty.
4. Customers are responsible for shipping costs to transport damaged products to our company or sales office.
5. To ensure the speed and quality of product repair, please download an RMA application form from our company website: http://rma.adlinktech.com/policy. Damaged products with attached RMA forms receive priority.

If you have any further questions, please email our FAE staff: service@adlinktech.com.