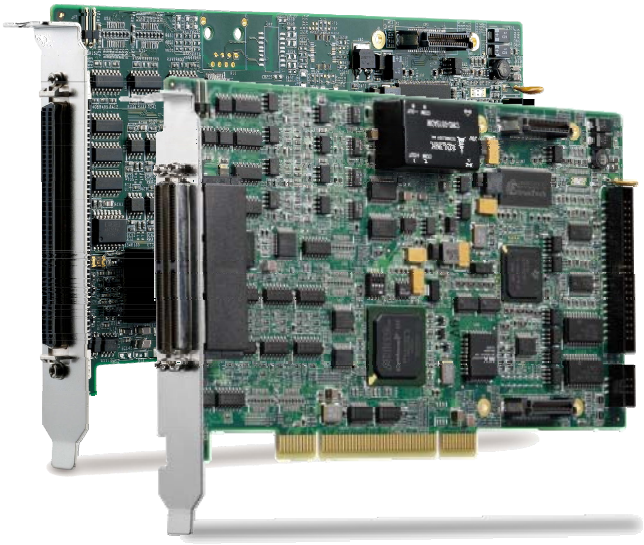
PCI-8254 / PCI-8258

DSP-Based 4/8
Advanced Motion Control Card

User manual



Version: 2.00

Updated: August 13, 2014

P/N: 50-15085-1000

Recycled Paper

Advance Technologies; Automate the World.

Revision History

Revision	Date	Description
2.00	2014-08-13	First release

Preface

Disclaimer

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

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

Environmental responsibility

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

Trademark

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

Conventions

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


NOTE

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


CAUTION

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


WARNING

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

Table of Contents

List of Figures

List of Tables

1 Introduction

The PCI-8254/PCI-8258, is a fully in-house developed DSP-based advanced motion control card from ADLINK. It supports 4/8 axis pulse type or Analog type signal commands, provides Open-loop and closed-loop circuit control options, and supports position/speed/torque commands for several different servo drivers.

The PCI-8254/PCI-8258 exchanges data with operating system through high speed PCI bus including motion control command, feedback data, parameter, etc. Used with the ADLINK exclusive Softmove kernel, it offers scores of move control functions including T/S speed profile planning, point-to-point movement, multi-dimension interpolation, and master/slave motion.

To enable absolute real time motion control, ADLINK's multi-tasks download function helps users to program with AMC code (ADLINK Motion Code), a programming language similar to C, for downloading to PCI-8254/58 embedded processors for execution. It supports up to eight different programs for concurrent download and execution.

With up to 20MHz high speed encoder feedback support and 4/8 axes independent hardware PID control plus Feed-forward gain design, it ensures precision control and reduce following errors in tandem with high speed motion feedback. The programmable servo update rate allows excellent control performance because different PID parameter adjustments can be made for individual applications.

The PCI-8254/58, see Figure 1 below for its system functions, uses one digital signal processor (DSP) from Texas Instrument (TI) as its main computing unit and integrates high speed large volume Field Programmable Gate Array (FPGA) to provide 4/8 independent PID control, high speed encoder output unit, 2/4 high speed position compare and trigger output, move & general purposed I/O and logic control. It separates isolation circuit into exclusive terminal board DIN-825-GP4 to prevent the burning out of PCI-8254/58 from incorrect wiring. Thanks to full range of flexing resistant wires from ADLINK, it connects with market available popular servo drives easily.

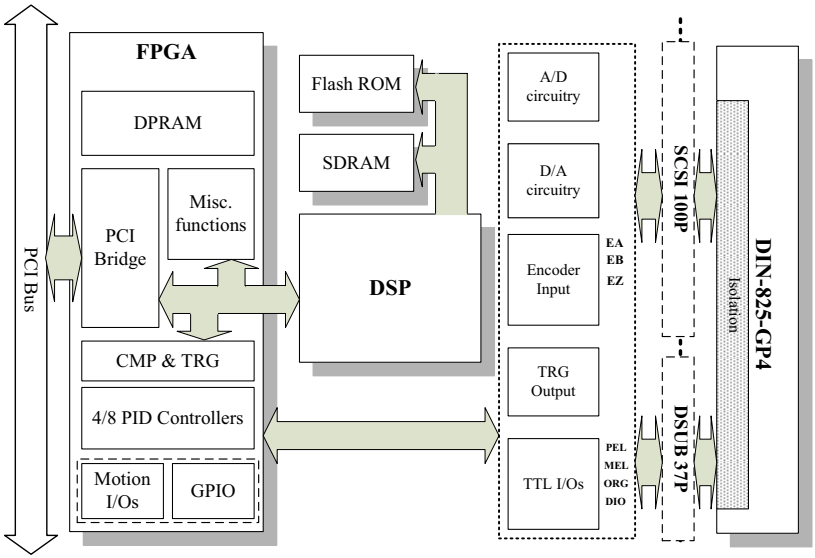


Figure 1-1: PCI-8254/58 system block diagram

Graphical motion control interface – MotionCreatorPro 2 is a Windows-based motion control software development tool for motion control and I/O status monitoring. You may employ this development tool for axis, PID, and feed-forward gain setup as well as analysis on motion variation curve and data. Embedded Setup Wizard can guide you throughout the process of hardware installation and wiring, close circuit PID parameter adjustments and single axis operation for reduced development time and costs.

The ***Windows Programming Libraries*** supports Windows coding environment including: Visual Studio C++ 6.0, Microsoft .NET framework based VB.NET and C++, and Borland's C++ Builder. There are sample programs available in the installation folders.

The flow chart below will guide you in using this manual as well as help you to locate any required information effectively.

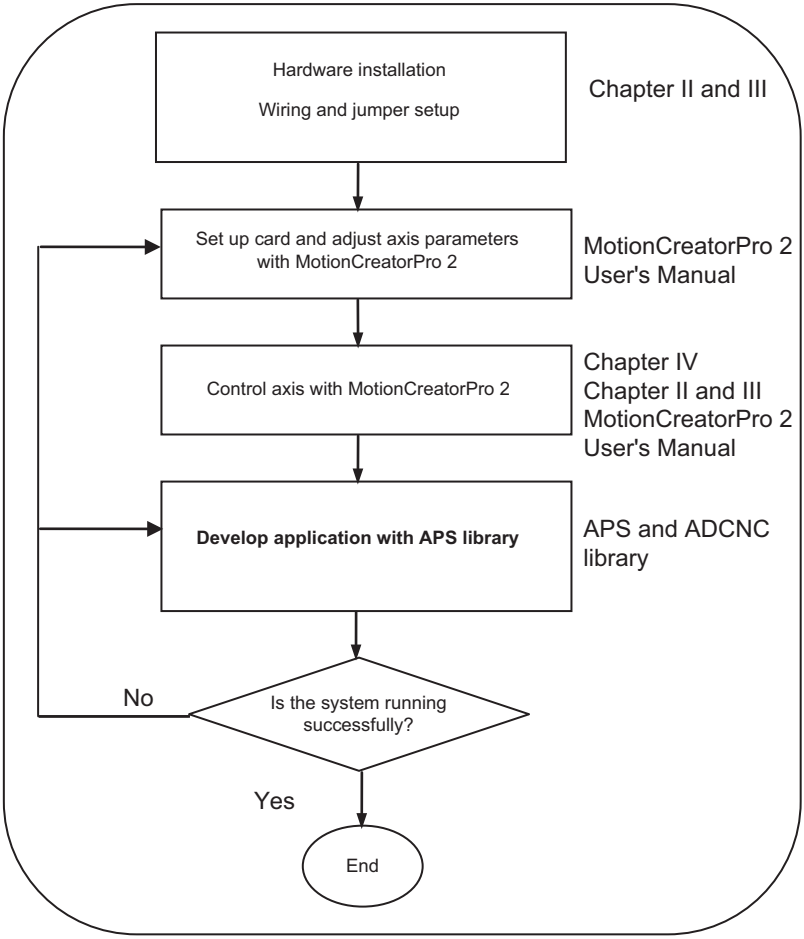


Figure 1-2: System installation flow chart

1.1 Product Specifications

	Item	Description
System	Bus information	PCI Rev. 2.2, 33MHz
	PCI bus width	32-bit
	PCI bus voltage	3.3V, 5V
	PCI bus IRQ settings	Assigned by PCI controller
DSP	Model	TI 375MHz floating DSP
	Memory (for program and data)	DDR2 SDRAM: 64Mx16bit Flash ROM: 16M-bit
Board-to-board interface	Connector	1x SCSI-II 100P for PCI-8254
		1x Dual SCSI VHDCI 100P for PCI-8258
Closed circuit control	Number of axes supported	4/8 axes for PCI-8254/8
	Analog command output resolution	±10 volts, resolution: 16 bit
	Analog command output interface	Difference / single end output
	Maximum servo update rate	50us-500us (programmable)
	PID (Kp, Ki and Kd) gain range	0 to 32,767
	Speed and acceleration feed-forward (Aff, Vff) gain range	0 to 32,767
	Position / speed command range	32 bit
	Acceleration / deceleration range	32 bit
	Encoder input frequency	20 MHz @ 4x AB
	Encoder input mode	CW/CCW, 1x/2x/4x AB Phase
	Encoder input interface	±12 volts, TTL compatible
	Filter	Biquad filter & low-pass filter

	Item	Description
I/O interface	Motion control relevant I/O	Plus/Minus end limitsignal
		Zero-position for each axis
	Drive relevant I/O	Servo ON
		In-position signal / Zero-Speed detection
		Alarm
Analog input	Max. input channel	4/8, single ended input
	Input voltage range	±10 V
	Sampling frequency	100 kHz
	Resolution	12 bits, no missing codes
	Accuracy	±1.5mV for ±10V input
	Overload voltage	±15 V
Analog output	Max. output channel	4/8, difference/single ended output
	Output voltage range	±10V
	Output current	±50mA (Typ.)
	Resolution	16 bits, no missing codes
	Accuracy	±1.2mV for ±10V output
	Protection circuit	Earth short circuit protection
	Settling Time	15us, full-scale step
General purpose digital I/O	General purpose I/O	20/24-CH input & 20/24-CH TTL output (optical isolation design for DIN-825-GP4)
Motion control function	Speed Profile Planning	Trapezoidal Curve and S-Curve
	Trajectory Planing	Jogging
		Point-to-point movement
	Linear interpolation: 2-6 axes	Online position/speed change
		3 axes arc interpolation
		3 axes spiral interpolation
		3 axes helix interpolation
	Home Return	User customizing (see zero-position, limit switch, EZ signals for reference)

	Item	Description
	Point table	Each axis supports 50 points buffer memory (BUFs)
		Supports point-to-point/line/arc and spiral interpolation
		Supports dwell function
		Supports pause/resume function
		Supports DO function
	Motion Monitoring	Motion control relevant I/O monitoring
		Motion monitoring
	Synchronous move	4/8 axes corresponding PCI-8254/PCI-8258
Industrial application	Master-client axes control	Up to 4/8 axis (including ganty control)
	Data sampling	Motion speed profile/ motion status/motion control relevant I/O
	System error diagnostics	Watchdog timer
Interrupt	Motion status event/error alarm/in position/ emergency stop	Planning in accordance with the manual
Position comparison & trigger output	Pulse output interface	Difference output
	Trigger channel	2/4 corresponding PCI-8254/PCI-8258
	Pulse logic	Programmable active-high or active-low
	Trigger output frequency	Linear compare trigger: 1MHz FIFO compare trigger: 255K ~ 1MHz
	Minimum pulse width	100ns programmable
	Position comparison mode	FIFO and linear comparison
	FIFO capacity	255 points (channel independent)

	Item	Description
PWM control	Maximum number of channels	2/4 CH correspondence PCI-8254/PCI-8258
	Control modes	● Fixed frequency, variable duty cycle ratio ● Variable frequency, fixed duty cycle ratio ● Variable frequency, variable duty cycle ratio
	Resolution	16 bit
Program download	Max. number of downloadable programs	8
	Program size	1,024 lines for each program
	Program logic	Boolean function, circuit
	Mathematical operation	Yes
	Motion event trigger	Yes

Environment Condition

	Item
Working ambient temperature	0~55°C
Storage ambient temperature	-20~75°C
Working ambient humidity	10~90%RH, without condensation
Storage ambient humidity	10~90%RH, without condensation
Noise impedance	Noise voltage 1500VPP noise frequency 25~60Hz using noise simulator
Environment condition	Minimal corrosive gas, dust
Cooling condition	Self-cooling
Power consumption	+3.3V @ 0.8A typical +5V @ 0.8A typical +/-12V @ 0.5A typical

1.2 Software Support

1.2.1 Software Support Library

PCI-8254/PCI-8258 supports Windows XP/7 32/64 bit operating system and provides a complete function library and DLL files for easy application development by users.

1.2.2 MotionCreatorPro 2

MotionCreatorPro 2 is a user interface exclusively developed for ADLINK motion control products in common Windows environment. You may easily set up card and axis parameters with the help of MotionCreatorPro 2. The Setup Wizard enables users to complete hardware installation, signal configuration, close-circuit PID gaining auto tuning and single-axis manipulation to reduce application development time. MotionCreatorPro 2 not only effectively reduces your development time but also enables you to concurrently validate the overall mechanism and electric design with all its single axis and interpolation motion operation pages.

1.3 Terminal Board

The PCI-8254/58 exclusive terminal board "DIN-825-GP4" can connect with several market available servo drives with special cables including the Mitsubishi) J3A and the Yaskawa Sigma V series or link with servo or stepper drives of other brands with single ended open cables. Brands with exclusive cables support are listed below:

Pulse command:

Cable	Supported brands
HSL-4XMO-DM	Mitsubishi J2S series
4XMO-DM-J3	Mitsubishi J3A series
HSL-4XMO-DP	Panasonic A4 and A5 series
HSL-4XMO-DY	Yaskawa Sigma V series
4XMO-DA	Delta A2 series
4XMO-OPEN	General purpose

Table 1-1: Cross-reference table of exclusive cables for pulse servo drive

Analog commands:

Cable	Supported brands
ACL-DM-J3	Mitsubishi J3A series
ACL-DY	Yaskawa Sigma V series
ACL-DP	Panasonic MINAS A5/A4 series
4XMO-OPEN	General purpose

Table 1-2: Cross-reference table of exclusive cables for analog servo drive

2 Getting Start with The Installation

This chapter teaches you how to install PCI-8254/PCI-8258 hardware and software as well as its I/O wiring.

- Package Contents
- Hardware Installation
- Software Installation
- I/O Wiring

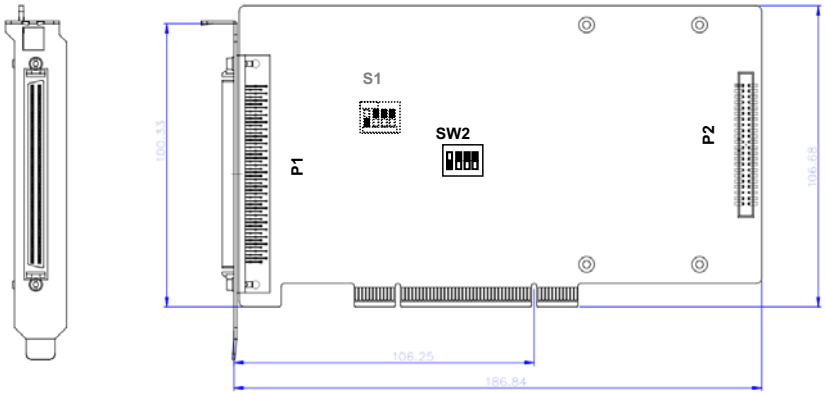
2.1 Package Contents

In addition to this manual you shall find the following item in the product package box:

- PCI-8254 or PCI-8258 card X 1
- IDE 44p – DSUB 37p flat cable x 1
- Product warranty card X 1

Should there be any item missed or damaged, please consult with your dealer immediately. Please keep the product along with items included in its package for easy replacement or repair.

2.2 PCI-8254/PCI-8258 Exterior Profile Diagram



NOTE

Dimension in unit of millimeter (mm).

Figure 2-1: Exterior of your PCI-8254

P1: for Motion control command, Position feedback, and Servo I/O feedback. (with SCSI 100-PINS connector)

P2: for 16 channel digital TTL input and 16 channel digital TTL output.

SW2: Card ID setup (0-15)

S1: Analog command mode selection (differential mode / single-ended mode)

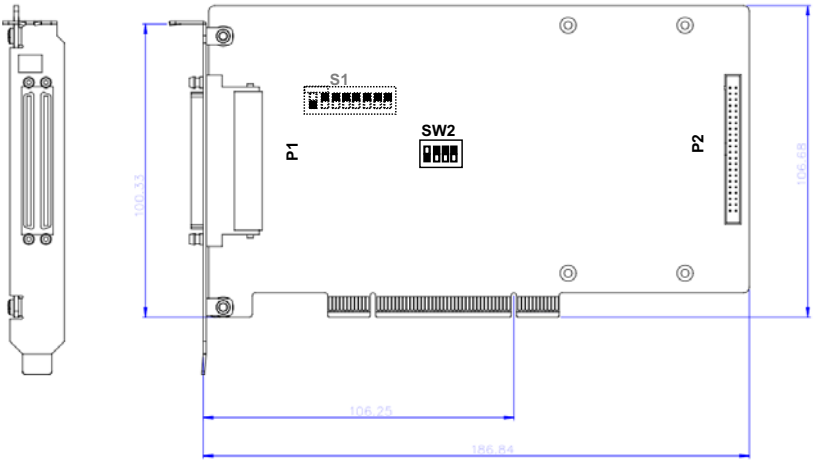


Figure 2-2: Exterior of your PCI-8258

P1: for Motion control command, Position feedback, and Servo I/O feedback. (with SCSI-VHDCI 200-PINS connector)

P2: for 16 channel digital TTL I/O. (with DSUB 37-PINS connector)

SW2: Card ID setup (0-15)

S1: Analog command mode selection (differential / single-end mode)

2.3 Hardware Installation

2.3.1 Hardware Configuration

PCI-8254/58 employs PCI Rev. 2.2 bus. System BIOS can auto configure memory and IRQ channel.
Exclusive terminal board DIN-825-GP4 provides isolation circuit and indicator lights for easy connection to varieties of servo drive and stepper drive.

2.3.2 Installation Procedures

1. Please read this manual carefully and set up signal I/O in proper mode.
2. Turn off computer power and relevant power on all terminal boards then connect PCI-8254/58 to 32-bit PCI slot in your computer. (The slot is usually in white color.) (Please make sure you have proper ESD (Electrostatic discharge) protection.)
3. Connect PCI-8254/58 and DIN-825-GP4 with SCSI 100p cable
4. Set up motion control relevant limit switch on DIN-825-GP4 board, servo signal and general purpose digital signal wiring
5. Set up servo or stepper drive connection
6. Turn on system power including computer power, terminal board relevant powers, and 24Vdc power
7. Verify all I/O signal and servo operation correctness with MotionCreatorPro 2

Please ground the shielding end of the power terminal to the earth to reduce risk of electric shock and ensure product operation of your electric appliances.

Please disconnect the motor drive from its load before using the card for the first time to protect your safety. Do not connect the motor drive to any mechanical devices before the completion of the installation and fine tuning of the control system. Connect the system only after the board is adjusted and the drive parameters can control the motor. Serious damage may be resulted in otherwise.

2.3.3 Troubleshooting

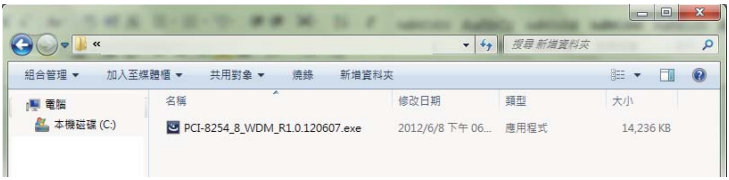
If the computer cannot power on normally or the motion control system operates abnormally after system installation, please follow steps described below for troubleshooting. If the problem persists after you have taken steps described, please consult the dealer where your product is purchased for technical services.

Abnormalities you encountered	Potential causes
The card does not show up in Windows Device Manager after its driver has been installed	Please turn off your computer, ensure the card is properly in PCI slot and the driver is properly installed by checking its proper installation in Windows Control Panel's "Add remove programs"
MotionCreatorPro2 cannot open after installing driver in computer	Ensure .NET framework v3.5 or later version has been installed in your system
The without signal indicator on MotionCreatorPro2 lights up after the motor is connected and the motor does not work.	Please ensure a 24Vdc power is connected to the terminal board
When using the MotionCreatorPro2, all the control indicators of the drive light correctly but the drive warns	Please ensure correctness of the axis parameter setup, alarm logic (ALM) and the EMG loop configuration
Value of output command differ from the feedback value from encoder	Please ensure feedback signal (CW/CCW, 1xAB, 2xAB, 4xAB) settings comply with that of the drive
If motion control, the motor moves only in one direction rather than back and forth two way movement	Please ensure setting of signal pattern (CW/CCW, OUT/DIR) comply with that of the motor drive
The motor bursts after executing Servo On command in speed control operation	Please ensure that the control direction of encoder and speed controller comply with each other
The motor makes loud noise after executing Servo On command in speed control operation	Please make sure the actuator is correctly adjusted and set up, reset all the "I" and "D" values of PID parameters, readjust PID gain

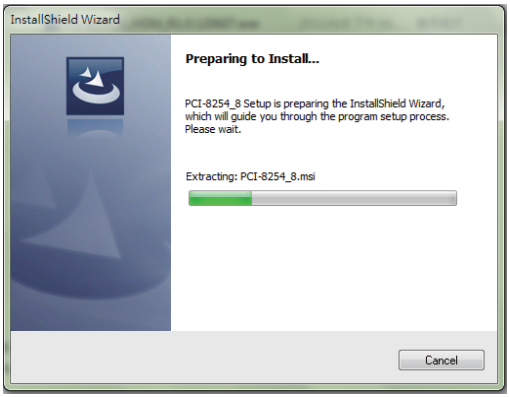
2.4 Software Installation Procedure

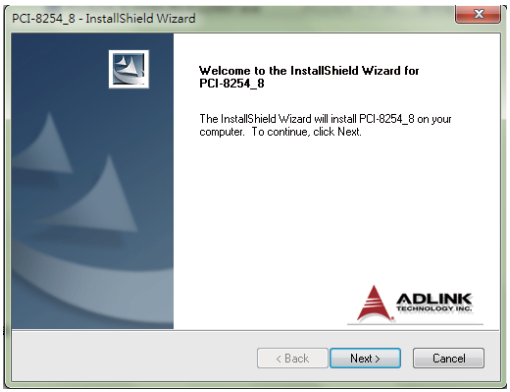
Windows driver installation procedure:

Step 1. Execute PCI-8254/PCI-8258 WDM file and run installation procedure automatically.

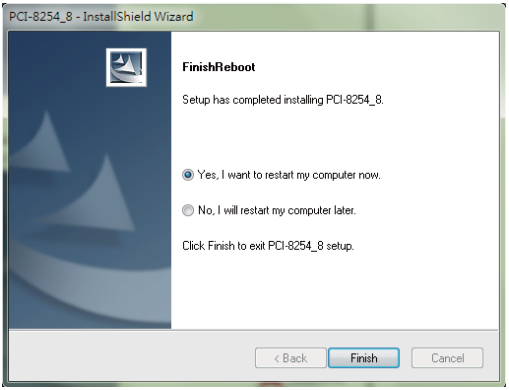


Step 2. Click "Next" as prompted to complete the installation process.

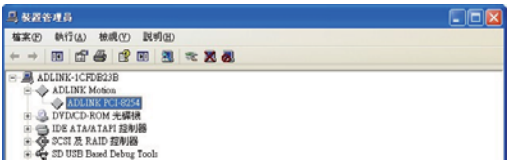




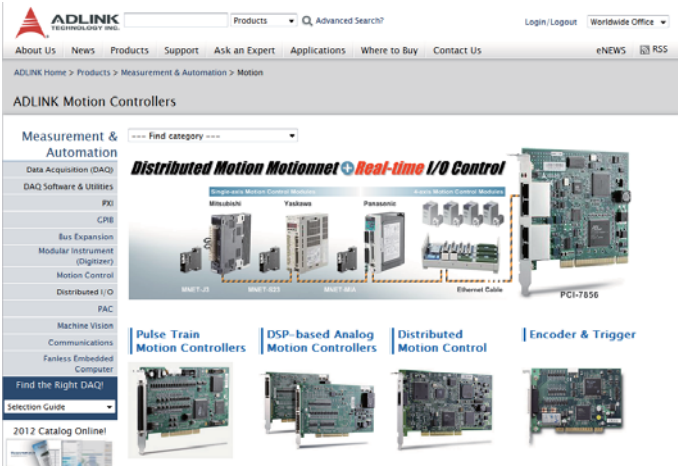
Step 3. Restart your computer after installation is completed.



Step 4. Ensure the Windows Device Manager identify your PCI-8254/PCI-8258 correctly.

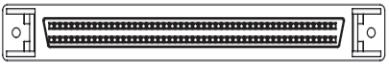


Note: Recommendations: Please download latest installation software from ADLINK official website to maintain the optimum operation environment.
(http://www.adlinktech.com/Motion-Control/index.php)



2.5 Definitions to Key Connector Signal

2.5.1 PCI-8254: Connector



- **P1**

No.	Name	I/O	Function of Axis	No.	Name	I/O	Function of Axis
1	DGND	--	Digital ground	**51**	IEMG	|	Emergency stop input
2	DGND	--	Digital ground	**52**	Rsv.	--	Reserved
3	AGND	--	Analog ground	**53**	AGND	--	Analog ground
4	AGND	--	Analog ground	**54**	AGND	--	Analog ground
5	AOUT1+	O	Analog output (+),(1)	**55**	AOUT3+	O	Analog output (+),(3)
6	AOUT1-	O	Analog output (-),(1)	**56**	AOUT3-	O	Analog output (-),(3)
7	AOUT2+	O	Analog output (+),(2)	**57**	AOUT4+	O	Analog output (+),(4)
8	AOUT2-	O	Analog output (-),(2)	**58**	AOUT4-	O	Analog output (-),(4)
9	AIN1	|	Analog input, (1)	**59**	AIN3	|	Analog input, (3)
10	AIN2	|	Analog input, (2)	**60**	AIN4	|	Analog input, (4)
11	EA5V	--	5V power	**61**	DGND	--	Digital ground
12	EA5V	--	5V power	**62**	DGND	--	Digital ground
13	OUT1+	O	Pulse output (+), (1)	**63**	OUT3+	O	Pulse output (+), (3)
14	OUT1-	O	Pulse output (-), (1)	**64**	OUT3-	O	Pulse output (-), (3)
15	DIR1+	O	Direction output (+), (1)	**65**	DIR3+	O	Direction output (+), (3)
16	DIR1-	O	Direction output (-), (1)	**66**	DIR3-	O	Direction output (-), (3)
17	OUT2+	O	Pulse output (+), (2)	**67**	OUT4+	O	Pulse output (+), (4)
18	OUT2-	O	Pulse output (-), (2)	**68**	OUT4-	O	Pulse output (-), (4)
19	DIR2+	O	Direction output (+), (2)	**69**	DIR4+	O	Direction output (+), (4)
20	DIR2-	O	Direction output (-), (2)	**70**	DIR4-	O	Direction output (-), (4)
21	TRG1+	O	Trigger output (+), (1)	**71**	TRG2+	O	Trigger output (+), (2)
22	TRG1-	O	Trigger output (-), (1)	**72**	TRG2-	O	Trigger output (-), (2)
23	EA1+	|	Encoder A-phase (+),(1)	**73**	EA3+	|	Encoder A-phase (+),(3)
24	EA1-	|	Encoder A-phase (-),(1)	**74**	EA3-	|	Encoder A-phase (-),(3)
25	EB1+	|	Encoder B-phase (+),(1)	**75**	EB3+	|	Encoder B-phase (+),(3)
26	EB1-	|	Encoder B-phase (-),(1)	**76**	EB3-	|	Encoder B-phase (-),(3)

No.	Name	I/O	Function of Axis	No.	Name	I/O	Function of Axis
27	EZ1+	I	Encoder Z-phase (+),(1)	**77**	EZ3+	I	Encoder Z-phase (+),(3)
28	EZ1-	I	Encoder Z-phase (-),(1)	**78**	EZ3-	I	Encoder Z-phase (-),(3)
29	EA2+	I	Encoder A-phase (+),(2)	**79**	EA4+	I	Encoder A-phase (+),(4)
30	EA2-	I	Encoder A-phase (-),(2)	**80**	EA4-	I	Encoder A-phase (-),(4)
31	EB2+	I	Encoder B-phase (+),(2)	**81**	EB4+	I	Encoder B-phase (+),(4)
32	EB2-	I	Encoder B-phase (-),(2)	**82**	EB4-	I	Encoder B-phase (-),(4)
33	EZ2+	I	Encoder Z-phase (+),(2)	**83**	EZ4+	I	Encoder Z-phase (+),(4)
34	EZ2-	I	Encoder Z-phase (-),(2)	**84**	EZ4-	I	Encoder Z-phase (-),(4)
35	ALM1	I	Servo alarm,(1)	**85**	ALM3	I	Servo alarm,(3)
36	ORG1	I	Origin Signal, (1)	**86**	ORG3	I	Origin Signal, (3)
37	SVON1	O	Servo-ON, (1)	**87**	SVON3	O	Servo-ON, (3)
38	PEL1	I	Positive limit, (1)	**88**	PEL3	I	Positive limit, (3)
39	ZSP1 / INP1	I	Zero Speed (1) / In-Position (1)	**89**	ZSP3 / INP3	I	Zero Speed (3) / In-Position (3)
40	MEL1	I	Negative limit, (1)	**90**	MEL3	I	Negative limit, (3)
41	ALM2	I	Servo alarm,(2)	**91**	ALM4	I	Servo alarm,(4)
42	ORG2	I	Origin Signal, (2)	**92**	ORG4	I	Origin Signal, (4)
43	SVON2	O	Servo-ON, (2)	**93**	SVON4	O	Servo-ON, (4)
44	PEL2	I	Positive limit, (2)	**94**	PEL4	I	Positive limit, (4)
45	ZSP2 / INP2	I	Zero Speed (2) / In-Position (2)	**95**	ZSP4 / INP4	I	Zero Speed (4) / In-Position (4)
46	MEL2	I	Negative limit, (2)	**96**	MEL4	I	Negative limit, (4)
47	EDO1	O	Digital Output, (1)	**97**	EDO3	O	Digital Output, (3)
48	EDI1	I	Digital Input, (1)	**98**	EDI3	I	Digital Input, (3)
49	EDO2	O	Digital Output, (2)	**99**	EDO4	O	Digital Output, (4)
50	EDI2	I	Digital Input, (2)	**100**	EDI4	I	Digital Input, (4)

2.5.2 PCI-8258: P1-A/B Connector

- **P1-A**

No.	Name	I/O	Function of Axis	No.	Name	I/O	Function of Axis
1	DGND	--	Digital ground	**51**	IEMG	|	Emergency stop input
2	DGND	--	Digital ground	**52**	Rsv.	--	Reserved
3	AGND	--	Analog ground	**53**	AGND	--	Analog ground
4	AGND	--	Analog ground	**54**	AGND	--	Analog ground
5	AOUT1+	O	Analog output (+),(1)	**55**	AOUT3+	O	Analog output (+),(3)
6	AOUT1-	O	Analog output (-),(1)	**56**	AOUT3-	O	Analog output (-),(3)
7	AOUT2+	O	Analog output (+),(2)	**57**	AOUT4+	O	Analog output (+),(4)
8	AOUT2-	O	Analog output (-),(2)	**58**	AOUT4-	O	Analog output (-),(4)
9	AIN1	|	Analog input, (1)	**59**	AIN3	|	Analog input, (3)
10	AIN2	|	Analog input, (2)	**60**	AIN4	|	Analog input, (4)
11	EA5V	--	5V power	**61**	DGND	--	Digital ground
12	EA5V	--	5V power	**62**	DGND	--	Digital ground
13	OUT1+	O	Pulse output (+), (1)	**63**	OUT3+	O	Pulse output (+), (3)
14	OUT1-	O	Pulse output (-), (1)	**64**	OUT3-	O	Pulse output (-), (3)
15	DIR1+	O	Direction output (+), (1)	**65**	DIR3+	O	Direction output (+), (3)
16	DIR1-	O	Direction output (-), (1)	**66**	DIR3-	O	Direction output (-), (3)
17	OUT2+	O	Pulse output (+), (2)	**67**	OUT4+	O	Pulse output (+), (4)
18	OUT2-	O	Pulse output (-), (2)	**68**	OUT4-	O	Pulse output (-), (4)
19	DIR2+	O	Direction output (+), (2)	**69**	DIR4+	O	Direction output (+), (4)
20	DIR2-	O	Direction output (-), (2)	**70**	DIR4-	O	Direction output (-), (4)
21	TRG1+	O	Trigger output (+), (1)	**71**	TRG2+	O	Trigger output (+), (2)
22	TRG1-	O	Trigger output (-), (1)	**72**	TRG2-	O	Trigger output (-), (2)
23	EA1+	|	Encoder A-phase (+),(1)	**73**	EA3+	|	Encoder A-phase (+),(3)
24	EA1-	|	Encoder A-phase (-),(1)	**74**	EA3-	|	Encoder A-phase (-),(3)
25	EB1+	|	Encoder B-phase (+),(1)	**75**	EB3+	|	Encoder B-phase (+),(3)
26	EB1-	|	Encoder B-phase (-),(1)	**76**	EB3-	|	Encoder B-phase (-),(3)
27	EZ1+	|	Encoder Z-phase (+),(1)	**77**	EZ3+	|	Encoder Z-phase (+),(3)
28	EZ1-	|	Encoder Z-phase (-),(1)	**78**	EZ3-	|	Encoder Z-phase (-),(3)

No.	Name	I/O	Function of Axis	No.	Name	I/O	Function of Axis
29	EA2+	I	Encoder A-phase (+),(2)	**79**	EA4+	I	Encoder A-phase (+),(4)
30	EA2-	I	Encoder A-phase (-),(2)	**80**	EA4-	I	Encoder A-phase (-),(4)
31	EB2+	I	Encoder B-phase (+),(2)	**81**	EB4+	I	Encoder B-phase (+),(4)
32	EB2-	I	Encoder B-phase (-),(2)	**82**	EB4-	I	Encoder B-phase (-),(4)
33	EZ2+	I	Encoder Z-phase (+),(2)	**83**	EZ4+	I	Encoder Z-phase (+),(4)
34	EZ2-	I	Encoder Z-phase (-),(2)	**84**	EZ4-	I	Encoder Z-phase (-),(4)
35	ALM1	I	Servo alarm,(1)	**85**	ALM3	I	Servo alarm,(3)
36	ORG1	I	Origin Signal, (1)	**86**	ORG3	I	Origin Signal, (3)
37	SVON1	O	Servo-ON, (1)	**87**	SVON3	O	Servo-ON, (3)
38	PEL1	I	Positive limit, (1)	**88**	PEL3	I	Positive limit, (3)
39	ZSP1 / INP1	I	Zero Speed (1) / In-Position (1)	**89**	ZSP3 / INP3	I	Zero Speed (3) / In-Position (3)
40	MEL1	I	Negative limit, (1)	**90**	MEL3	I	Negative limit, (3)
41	ALM2	I	Servo alarm,(2)	**91**	ALM4	I	Servo alarm,(4)
42	ORG2	I	Origin Signal, (2)	**92**	ORG4	I	Origin Signal, (4)
43	SVON2	O	Servo-ON, (2)	**93**	SVON4	O	Servo-ON, (4)
44	PEL2	I	Positive limit, (2)	**94**	PEL4	I	Positive limit, (4)
45	ZSP2 / INP2	I	Zero Speed (2) / In-Position (2)	**95**	ZSP4 / INP4	I	Zero Speed (4) / In-Position (4)
46	MEL2	I	Negative limit, (2)	**96**	MEL4	I	Negative limit, (4)
47	EDO1	O	Digital Output, (1)	**97**	EDO3	O	Digital Output, (3)
48	EDI1	I	Digital Input, (1)	**98**	EDI3	I	Digital Input, (3)
49	EDO2	O	Digital Output, (2)	**99**	EDO4	O	Digital Output, (4)
50	EDI2	I	Digital Input, (2)	**100**	EDI4	I	Digital Input, (4)

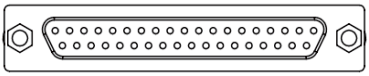
- **P1-B**

No.	Name	I/O	Function of Axis	No.	Name	I/O	Function of Axis
1	Rsv.	--	Reserved	**51**	Rsv.	--	Reserved
2	Rsv.	--	Reserved	**52**	Rsv.	--	Reserved
3	AGND	--	Analog ground	**53**	AGND	--	Analog ground
4	AGND	--	Analog ground	**54**	AGND	--	Analog ground
5	AOUT5+	O	Analog output (+),(5)	**55**	AOUT7+	O	Analog output (+),(7)

No.	Name	I/O	Function of Axis	No.	Name	I/O	Function of Axis
6	AOUT5-	O	Analog output (-),(5)	**56**	AOUT7-	O	Analog output (-),(7)
7	AOUT6+	O	Analog output (+),(6)	**57**	AOUT8+	O	Analog output (+),(8)
8	AOUT6-	O	Analog output (-),(6)	**58**	AOUT8-	O	Analog output (-),(8)
9	AIN5	I	Analog input, (5)	**59**	AIN7	I	Analog input, (7)
10	AIN6	I	Analog input, (6)	**60**	AIN8	I	Analog input, (8)
11	Rsv.	--	Reserved	**61**	DGND	--	Digital ground
12	Rsv.	--	Reserved	**62**	DGND	--	Digital ground
13	OUT5+	O	Pulse output (+), (5)	**63**	OUT7+	O	Pulse output (+), (7)
14	OUT5-	O	Pulse output (-), (5)	**64**	OUT7-	O	Pulse output (-), (7)
15	DIR5+	O	Direction output (+), (5)	**65**	DIR7+	O	Direction output (+), (7)
16	DIR5-	O	Direction output (-), (5)	**66**	DIR7-	O	Direction output (-), (7)
17	OUT6+	O	Pulse output (+), (6)	**67**	OUT8+	O	Pulse output (+), (8)
18	OUT6-	O	Pulse output (-), (6)	**68**	OUT8-	O	Pulse output (-), (8)
19	DIR6+	O	Direction output (+), (6)	**69**	DIR8+	O	Direction output (+), (8)
20	DIR6-	O	Direction output (-), (6)	**70**	DIR8-	O	Direction output (-), (8)
21	TRG3+	O	Trigger output (+), (3)	**71**	TRG4+	O	Trigger output (+), (4)
22	TRG3-	O	Trigger output (-), (3)	**72**	TRG4-	O	Trigger output (-), (4)
23	EA5+	I	Encoder A-phase (+),(5)	**73**	EA7+	I	Encoder A-phase (+),(7)
24	EA5-	I	Encoder A-phase (-),(5)	**74**	EA7-	I	Encoder A-phase (-),(7)
25	EB5+	I	Encoder B-phase (+),(5)	**75**	EB7+	I	Encoder B-phase (+),(7)
26	EB5-	I	Encoder B-phase (-),(5)	**76**	EB7-	I	Encoder B-phase (-),(7)
27	EZ5+	I	Encoder Z-phase (+),(5)	**77**	EZ7+	I	Encoder Z-phase (+),(7)
28	EZ5-	I	Encoder Z-phase (-),(5)	**78**	EZ7-	I	Encoder Z-phase (-),(7)
29	EA6+	I	Encoder A-phase (+),(6)	**79**	EA8+	I	Encoder A-phase (+),(8)
30	EA6-	I	Encoder A-phase (-),(6)	**80**	EA8-	I	Encoder A-phase (-),(8)
31	EB6+	I	Encoder B-phase (+),(6)	**81**	EB8+	I	Encoder B-phase (+),(8)
32	EB6-	I	Encoder B-phase (-),(6)	**82**	EB8-	I	Encoder B-phase (-),(8)
33	EZ6+	I	Encoder Z-phase (+),(6)	**83**	EZ8+	I	Encoder Z-phase (+),(8)
34	EZ6-	I	Encoder Z-phase (-),(6)	**84**	EZ8-	I	Encoder Z-phase (-),(8)
35	ALM5	I	Servo alarm,(5)	**85**	ALM7	I	Servo alarm,(7)
36	ORG5	I	Origin Signal, (5)	**86**	ORG7	I	Origin Signal, (7)

No.	Name	I/O	Function of Axis	No.	Name	I/O	Function of Axis
37	SVON5	O	Servo-ON, (5)	**87**	SVON7	O	Servo-ON, (7)
38	PEL5	I	Positive limit, (5)	**88**	PEL7	I	Positive limit, (7)
39	ZSP5 / INP5	I	Zero Speed (5) / In-Position (5)	**89**	ZSP7 / INP7	I	Zero Speed (7) / In-Position (7)
40	MEL5	I	Negative limit, (5)	**90**	MEL7	I	Negative limit, (7)
41	ALM6	I	Servo alarm,(6)	**91**	ALM8	I	Servo alarm,(8)
42	ORG6	I	Origin Signal, (6)	**92**	ORG8	I	Origin Signal, (8)
43	SVON6	O	Servo-ON, (6)	**93**	SVON8	O	Servo-ON, (8)
44	PEL6	I	Positive limit, (6)	**94**	PEL8	I	Positive limit, (8)
45	ZSP6 / INP6	I	Zero Speed (6) / In-Position (6)	**95**	ZSP8 / INP8	I	Zero Speed (8) / In-Position (8)
46	MEL6	I	Negative limit, (6)	**96**	MEL8	I	Negative limit, (8)
47	EDO5	O	Digital Output, (5)	**97**	EDO7	O	Digital Output, (7)
48	EDI5	I	Digital Input, (5)	**98**	EDI7	I	Digital Input, (7)
49	EDO6	O	Digital Output, (6)	**99**	EDO8	O	Digital Output, (8)
50	EDI6	I	Digital Input, (6)	**100**	EDI8	I	Digital Input, (8)

2.5.3 PCI-8254/58: P2 Connector



- **P2**

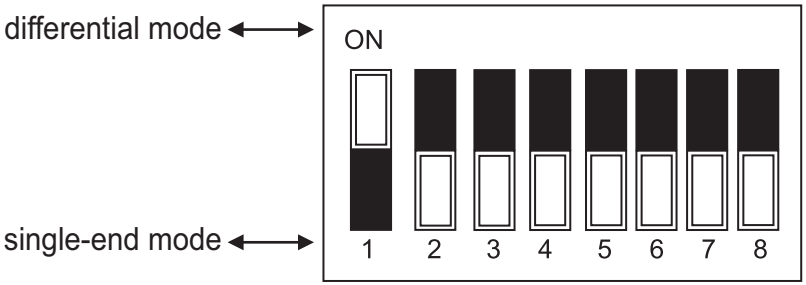
No.	Name	I/O	Function of Axis	No.	Name	I/O	Function of Axis
1	Rsv.	- -	Reserved	**20**	VDD	|	+5V power supply input
2	TDI1	|	TTL input, (1)	**21**	TDO1	O	TTL output, (1)
3	TDI2	|	TTL input, (2)	**22**	TDO2	O	TTL output, (2)
4	TDI3	|	TTL input, (3)	**23**	TDO3	O	TTL output, (3)
5	TDI4	|	TTL input, (4)	**24**	TDO4	O	TTL output, (4)
6	TDI5	|	TTL input, (5)	**25**	TDO5	O	TTL output, (5)
7	TDI6	|	TTL input, (6)	**26**	TDO6	O	TTL output, (6)
8	TDI7	|	TTL input, (7)	**27**	TDO7	O	TTL output, (7)
9	TDI8	|	TTL input, (8)	**28**	TDO8	O	TTL output, (8)
10	TDI9	|	TTL input, (9)	**29**	TDO9	O	TTL output, (9)
11	TDI10	|	TTL input, (10)	**30**	TDO10	O	TTL output, (10)

No.	Name	I/O	Function of Axis	No.	Name	I/O	Function of Axis
12	TDI11	I	TTL input, (11)	**31**	TDO11	O	TTL output, (11)
13	TDI12	I	TTL input, (12)	**32**	TDO12	O	TTL output, (12)
14	TDI13	I	TTL input, (13)	**33**	TDO13	O	TTL output, (13)
15	TDI14	I	TTL input, (14)	**34**	TDO14	O	TTL output, (14)
16	TDI15	I	TTL input, (15)	**35**	TDO15	O	TTL output, (15)
17	TDI16	I	TTL input, (16)	**36**	TDO16	O	TTL output, (16)
18	DGND	-	Digital ground	**37**	DGND	-	Digital ground
19	VDD	I	+5V power supply input	**--**	--	--	--

2.6 DIP Switch

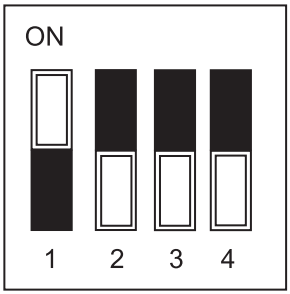
2.6.1 S1: Analog Output Mode Settings

This switch is used for switching analog command output modes with default settings at differential type output. In general, you adjust according to individual servo drive interface. The PCI-8254 model uses switches 1-4 only.



2.6.2 SW2: Card ID Switch

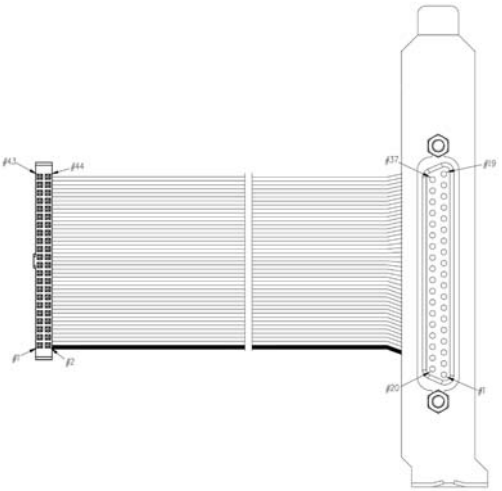
This switch is used for adjusting card ID for easy identification in user application programs. Take example. If you set card ID to ”0-0-0-1” (OFF-OFF-OFF-ON) then the card ID is ”1” and the ID table should be set up as described below:



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

2.7 IDE 44p – DSUB 37p Bus

This card include one IDE cable from IDE 44 pin to DSUB 37 pin. It is used for PCI-8254 / PCI-8258 P2 extension 16 channel digital input and 16 channel digital output.



2.8 Exclusive Board - DIN-825-GP4

The DIN-825-GP4 terminal board is designed for PCI-8254/PCI-8258 and AMP-204C/AMP-208C exclusively. It connects with market available servo drives with special cables including the Mitsubishi's J3A and the Yaskawa Sigma V series or other servo or stepper drives with single end open cables.

CAUTION

The DIN-825-GP4 board supports both PCI-8254/PCI-8258 and AMP-204C/AMP-208C. DO NOT connect it to other ADLINK's motion controller or it may be damaged.

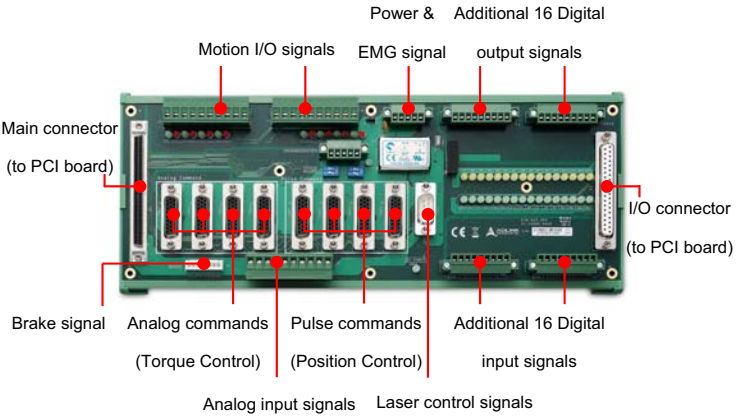


Figure 2-3: Exterior of DIN-825-GP4

2.8.1 Definitions to Connector

1. P1: This is one SCSI 100-PINS connector for motion control signals.
2. CMA1–4: These are four 26-PINS connector for connecting to servo drive to do S/T mode control and analog control commands output.
3. CMP1–4: These are four 26-PINS connectors for connecting to servo drive to do P mode control or stepper drive to output pulse control commands. It may be connected to Mitsubishi J3A series, Yaskawa Sigma II, III & V series, and Panasonic MINAS A4&A5 with exclusive cables.
4. J1–J3: These are three sets of 10-pins screw lock connectors (screwed series, Delta A2 series, or connection to other servo or stepper drives with single end open cables). It may be connected to any analog input signal, comparing trigger signal, plus/minus limit switch and homing signal '.
5. J4: This is one 8-PINS connector for connecting to Brake Signal.
6. J5: This is one 5-PINS connector for connecting to terminal board main power and emergency stop signals.

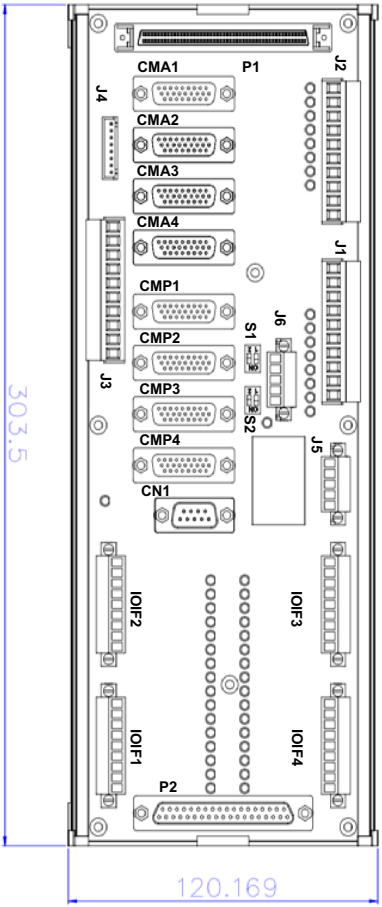
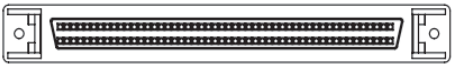


Figure 2-4: Exterior of DIN-825-GP4

7. J6: This is one 5-PINS connector for connecting to four isolation digital output channel.
8. P2: This is one DSUB 37-PINS connector for connecting to 16 channel digital input signal and 16 channel digital output signal in the controller (TTL).
9. IOIF1-IOIF4: These are four 9-PINS connectors for connecting to 16 channel digital input signal and 16 channel digital output signal for common uses.
10. Newly added CN1: This is one 9-pin connector for laser control.

2.8.2 Connector: For Connecting to PCI-8254/PCI-8258/AMP-204C/AMP-208C

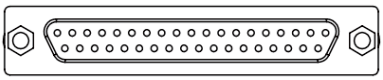
- **P1:**



No.	Name	I/O	Function of Axis	No.	Name	I/O	Function of Axis
1	DGND	--	Digital ground	**51**	IEMG	|	Emergency stop input
2	DGND	--	Digital ground	**52**	Rsv.	--	Reserved
3	AGND	--	Analog ground	**53**	AGND	--	Analog ground
4	AGND	--	Analog ground	**54**	AGND	--	Analog ground
5	AOUT1+	O	Analog output (+),(1)	**55**	AOUT3+	O	Analog output (+),(3)
6	AOUT1-	O	Analog output (-),(1)	**56**	AOUT3-	O	Analog output (-),(3)
7	AOUT2+	O	Analog output (+),(2)	**57**	AOUT4+	O	Analog output (+),(4)
8	AOUT2-	O	Analog output (-),(2)	**58**	AOUT4-	O	Analog output (-),(4)
9	AIN1	|	Analog input, (1)	**59**	AIN3	|	Analog input, (3)
10	AIN2	|	Analog input, (2)	**60**	AIN4	|	Analog input, (4)
11	Rsv.	--	Reserved	**61**	DGND	--	Digital ground
12	Rsv.	--	Reserved	**62**	DGND	--	Digital ground
13	OUT1+	O	Pulse output (+), (1)	**63**	OUT3+	O	Pulse output (+), (3)
14	OUT1-	O	Pulse output (-), (1)	**64**	OUT3-	O	Pulse output (-), (3)
15	DIR1+	O	Direction output (+), (1)	**65**	DIR3+	O	Direction output (+), (3)
16	DIR1-	O	Direction output (-), (1)	**66**	DIR3-	O	Direction output (-), (3)
17	OUT2+	O	Pulse output (+), (2)	**67**	OUT4+	O	Pulse output (+), (4)
18	OUT2-	O	Pulse output (-), (2)	**68**	OUT4-	O	Pulse output (-), (4)
19	DIR2+	O	Direction output (+), (2)	**69**	DIR4+	O	Direction output (+), (4)
20	DIR2-	O	Direction output (-), (2)	**70**	DIR4-	O	Direction output (-), (4)
21	TRG1+	O	Trigger output (+), (1)	**71**	TRG2+	O	Trigger output (+), (2)
22	TRG1-	O	Trigger output (-), (1)	**72**	TRG2-	O	Trigger output (-), (2)
23	EA1+	|	Encoder A-phase (+),(1)	**73**	EA3+	|	Encoder A-phase (+),(3)
24	EA1-	|	Encoder A-phase (-),(1)	**74**	EA3-	|	Encoder A-phase (-),(3)
25	EB1+	|	Encoder B-phase (+),(1)	**75**	EB3+	|	Encoder B-phase (+),(3)
26	EB1-	|	Encoder B-phase (-),(1)	**76**	EB3-	|	Encoder B-phase (-),(3)

No.	Name	I/O	Function of Axis	No.	Name	I/O	Function of Axis
27	EZ1+	I	Encoder Z-phase (+),(1)	**77**	EZ3+	I	Encoder Z-phase (+),(3)
28	EZ1-	I	Encoder Z-phase (-),(1)	**78**	EZ3-	I	Encoder Z-phase (-),(3)
29	EA2+	I	Encoder A-phase (+),(2)	**79**	EA4+	I	Encoder A-phase (+),(4)
30	EA2-	I	Encoder A-phase (-),(2)	**80**	EA4-	I	Encoder A-phase (-),(4)
31	EB2+	I	Encoder B-phase (+),(2)	**81**	EB4+	I	Encoder B-phase (+),(4)
32	EB2-	I	Encoder B-phase (-),(2)	**82**	EB4-	I	Encoder B-phase (-),(4)
33	EZ2+	I	Encoder Z-phase (+),(2)	**83**	EZ4+	I	Encoder Z-phase (+),(4)
34	EZ2-	I	Encoder Z-phase (-),(2)	**84**	EZ4-	I	Encoder Z-phase (-),(4)
35	ALM1	I	Servo alarm,(1)	**85**	ALM3	I	Servo alarm,(3)
36	ORG1	I	Origin Signal, (1)	**86**	ORG3	I	Origin Signal, (3)
37	SVON1	O	Servo-ON, (1)	**87**	SVON3	O	Servo-ON, (3)
38	PEL1	I	Positive limit, (1)	**88**	PEL3	I	Positive limit, (3)
39	ZSP1 / INP1	I	Zero Speed (1) / In-Position (1)	**89**	ZSP3 / INP3	I	Zero Speed (3) / In-Position (3)
40	MEL1	I	Negative limit, (1)	**90**	MEL3	I	Negative limit, (3)
41	ALM2	I	Servo alarm,(2)	**91**	ALM4	I	Servo alarm,(4)
42	ORG2	I	Origin Signal, (2)	**92**	ORG4	I	Origin Signal, (4)
43	SVON2	O	Servo-ON, (2)	**93**	SVON4	O	Servo-ON, (4)
44	PEL2	I	Positive limit, (2)	**94**	PEL4	I	Positive limit, (4)
45	ZSP2 / INP2	I	Zero Speed (2) / In-Position (2)	**95**	ZSP4 / INP4	I	Zero Speed (4) / In-Position (4)
46	MEL2	I	Negative limit, (2)	**96**	MEL4	I	Negative limit, (4)
47	EDO1	O	Digital Output, (1)	**97**	EDO3	O	Digital Output, (3)
48	EDI1	I	Digital Input, (1)	**98**	EDI3	I	Digital Input, (3)
49	EDO2	O	Digital Output, (2)	**99**	EDO4	O	Digital Output, (4)
50	EDI2	I	Digital Input, (2)	**100**	EDI4	I	Digital Input, (4)

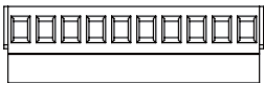
- **P2**：



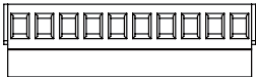
No.	Name	I/O	Function of Axis	No.	Name	I/O	Function of Axis
1	Rsv.	- -	Reserved	**20**	VDD	O	+5V power supply output
2	TDI1	I	TTL input, (1)	**21**	TDO1	O	TTL output, (1)

No.	Name	I/O	Function of Axis	No.	Name	I/O	Function of Axis
3	TDI2	I	TTL input, (2)	**22**	TDO2	O	TTL output, (2)
4	TDI3	I	TTL input, (3)	**23**	TDO3	O	TTL output, (3)
5	TDI4	I	TTL input, (4)	**24**	TDO4	O	TTL output, (4)
6	TDI5	I	TTL input, (5)	**25**	TDO5	O	TTL output, (5)
7	TDI6	I	TTL input, (6)	**26**	TDO6	O	TTL output, (6)
8	TDI7	I	TTL input, (7)	**27**	TDO7	O	TTL output, (7)
9	TDI8	I	TTL input, (8)	**28**	TDO8	O	TTL output, (8)
10	TDI9	I	TTL input, (9)	**29**	TDO9	O	TTL output, (9)
11	TDI10	I	TTL input, (10)	**30**	TDO10	O	TTL output, (10)
12	TDI11	I	TTL input, (11)	**31**	TDO11	O	TTL output, (11)
13	TDI12	I	TTL input, (12)	**32**	TDO12	O	TTL output, (12)
14	TDI13	I	TTL input, (13)	**33**	TDO13	O	TTL output, (13)
15	TDI14	I	TTL input, (14)	**34**	TDO14	O	TTL output, (14)
16	TDI15	I	TTL input, (15)	**35**	TDO15	O	TTL output, (15)
17	TDI16	I	TTL input, (16)	**36**	TDO16	O	TTL output, (16)
18	EGND	-	External power ground	**37**	EGND	-	External power ground
19	VDD	I	+5V power supply input	**--**	--	--	--

- **J1：**



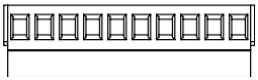
No.	Name	I/O	Function of Axis	No.	Name	I/O	Function of Axis
1	DICOM	--	Digital input common	**6**	EDI4	I	Isolated digital input, (4)
2	EDI3	I	Isolated digital input, (3)	**7**	PEL4	I	Positive limit, (4)
3	PEL3	I	Positive limit, (3)	**8**	ORG4	I	Origin Signal, (4)
4	ORG3	I	Origin Signal, (3)	**9**	MEL4	I	Negative limit, (4)
5	MEL3	I	Negative limit, (3)	**10**	DOCOM	--	Digital output common

- **J2：** 

No.	Name	I/O	Function of Axis	No.	Name	I/O	Function of Axis
1	DICOM	--	Digital input common	6	EDI2	|	Isolated digital input, (2)
2	EDI1	|	Isolated digital input, (1)	7	PEL2	|	Positive limit, (2)
3	PEL1	|	Positive limit, (1)	8	ORG2	|	Origin Signal, (2)
4	ORG1	|	Origin Signal, (1)	9	MEL2	|	Negative limit, (2)
5	MEL1	|	Negative limit, (1)	10	DOCOM	--	Digital output common

NOTE

1. Please connect DICOM to external power supply (24VDC in general) if possible.
2. Please connect DOCOM to ground (GND) of external power supply if possible.

- **J3：** 

No.	Name	I/O	Function of Axis	No.	Name	I/O	Function of Axis
1	DGND	--	Isolated digital ground	6	AGND	--	Analog ground
2	TRG2-	O	Trigger output (-), (2)	7	AI4	|	Analog input, (4)
3	TRG2+	O	Trigger output (+), (2)	8	AI3	|	Analog input, (3)
4	TRG1-	O	Trigger output (-), (1)	9	AI2	|	Analog input, (2)
5	TG1+	O	Trigger output (+), (1)	10	AI1	|	Analog input, (1)

- **J4: Brake Connector** 

No.	Name	I/O	Function of Axis	No.	Name	I/O	Function of Axis
1	BRAKE 1+	--	Brake signal (+), (1)	6	BRAKE 3+	|	Brake signal (+), (3)
2	BRAKE 1-	|	Brake signal (-), (1)	7	BRAKE 3-	|	Brake signal (-), (3)
3	BRAKE 2+	|	Brake signal (+), (2)	8	BRAKE 4+	|	Brake signal (+), (4)
4	BRAKE 2-	|	Brake signal (-), (2)	9	BRAKE 4-	|	Brake signal (-), (4)

- **J5** 

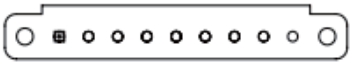
No.	Name	I/O	Function of Axis	No.	Name	I/O	Function of Axis
1	I24V	--	Ext. power supply, +24V	**4**	DOCOM	--	Digital output common
2	IGND	--	Ext. power ground	**5**	EEMG	|	Ext. Emergency signal
3	DICOM	--	Digital input common	**6**	--	--	--

NOTE

1. Please connect DICOM to external power supply (24VDC in general) if possible.
2. Please connect DOCOM to ground (GND) of external power supply if possible.

- **J6** 

No.	Name	I/O	Function of Axis	No.	Name	I/O	Function of Axis
1	EDO1	O	Digital output, (1)	**4**	EDO4	O	Digital output, (4)
2	EDO2	O	Digital output, (2)	**5**	DOCOM	O	Digital output common
3	EDO3	O	Digital output, (3)	**6**	--	O	--

- **IOIF1:** 

No.	Name	I/O	Function of Axis	No.	Name	I/O	Function of Axis
1	DI1	I	Additional isolated digital input, (1)	**6**	DI6	I	Additional isolated digital input, (6)
2	DI2	I	Additional isolated digital input, (2)	**7**	DI7	I	Additional isolated digital input, (7)
3	DI3	I	Additional isolated digital input, (3)	**8**	DI8	I	Additional isolated digital input, (8)
4	DI4	I	Additional isolated digital input, (4)	**9**	DICOM	--	Digital input common
5	DI5	I	Additional isolated digital input, (5)	--	--	--	--

- **IOIF2:**

No.	Name	I/O	Function of Axis	No.	Name	I/O	Function of Axis
1	DI9	I	Additional isolated digital input, (9)	**6**	DI14	I	Additional isolated digital input, (14)
2	DI10	I	Additional isolated digital input, (10)	**7**	DI15	I	Additional isolated digital input, (15)
3	DI11	I	Additional isolated digital input, (11)	**8**	DI1	I	Additional isolated digital input, (16)
4	DI12	I	Additional isolated digital input, (12)	**9**	DICOM	--	Digital input common
5	DI13	I	Additional isolated digital input, (13)	--	--	--	--

- **IOIF3:**

No.	Name	I/O	Function of Axis	No.	Name	I/O	Function of Axis
1	※DO1	O	Additional isolated digital output, (1)	**6**	DO6	O	Additional isolated digital output, (6)
2	※DO2	O	Additional isolated digital output, (2)	**7**	DO7	O	Additional isolated digital output, (7)
3	※DO3	O	Additional isolated digital output, (3)	**8**	DO8	O	Additional isolated digital output, (8)
4	※DO4	O	Additional isolated digital output, (4)	**9**	DOCOM	--	Digital output common
5	※DO5	O	Additional isolated digital output, (5)	--	--	--	--

NOTE

※ The digital output current may reach 250mA

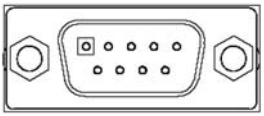
- **IOIF4:**

No.	Name	I/O	Function of Axis	No.	Name	I/O	Function of Axis
1	DO9	O	Additional isolated digital output, (9)	**6**	DO14	O	Additional isolated digital output, (14)
2	DO10	O	Additional isolated digital output, (10)	**7**	DO15	O	Additional isolated digital output, (15)
3	DO11	O	Additional isolated digital output, (11)	**8**	DO16	O	Additional isolated digital output, (16)
4	DO12	O	Additional isolated digital output, (12)	**9**	DOCOM	--	Digital output common
5	DO13	O	Additional isolated digital output, (13)	--	--	--	--

NOTE

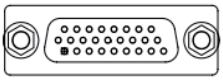
1. Please connect DICOM to external power supply (24VDC in general) if possible.
2. Please connect DOCOM to ground (GND) of external power supply if possible.

- **CN1:**



No.	Name	I/O	Function of Axis	No.	Name	I/O	Function of Axis
1	EDO4	O	Digital output (4)	**6**	EDO4-	O	Digital output (-), (4)
2	TG1+	O	Trigger output (+), (1)	**7**	TG1-	O	Trigger output (-), (1)
3	TG2+	O	Trigger output (+), (2)	**8**	TG2-	O	Trigger output (-), (2)
4	AOUT4+	O	Analog command output (+), (4)	**9**	DGND	--	Digital ground
5	AGND	--	Analog ground				

- **CMA1~CMA4:**

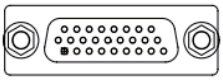


No.	Name	I/O	Function	No.	Name	I/O	Function	No.	Name	I/O	Function
				10	ALM_RST / DO	O	Reset driver signal / Digital output signal				
1	SVON	O	Servo On signal					19	EMG	I	Emergency signal
2	ZSP	I	Zero speed signal	11	ALM	I	Servo alarm signal				
				12	I24V	--	Ext. power supply, +24V	20	IGND	--	Ext. power ground
3	Rsv.	--	Reserved					21	IGND	--	Ext. power ground
4	Rsv.	--	Reserved.	13	IGND	--	Ext. power ground				
				14	BRAKE-	O	Brake signal(-)	22	IGND	--	Ext. power ground
5	AOUT-	O	Analog command output (-)					23	Rsv.	--	Reserved
				15	AGND	--	Analog ground				
6	AOUT+	O	Analog command output (+)					24	Rsv.	--	Reserved
7	EA-	I	Encoder A-phase(-)	16	EB-	I	Encoder B-phase(-)	25	EZ-	I	Encoder Z-phase(-)
				17	EB+	I	Encoder B-phase(+)				
8	EA+	I	Encoder A-phase(+)					26	EZ+	I	Encoder Z-phase(+)
				18	AGND	--	Analog ground				
9	BRAKE+	O	Brake signal(+)								

NOTE

ALM_RST / DO: This signal may be selected as general purpose digital output signal (EDO) or alarm clearance function (ALM_RST) by switch S1 or S2.

- **CMP1~CMP4:**

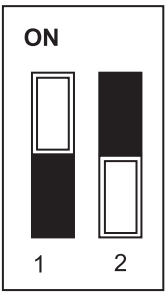


No.	Name	I/O	Function	No.	Name	I/O	Function	No.	Name	I/O	Function
				10	ALM_RST / DO	O	Reset driver signal / Digital output signal				
1	SVON	O	Servo On signal	11	ALM	I	Servo alarm signal	19	EMG	I	Emergency signal
2	INP	I	In-position signal	12	I24V	--	Ext. power supply, +24V	20	IGND	--	Ext. power ground
3	ERC	O	Dev. ctr. dir. signal	13	IGND	--	Ext. power ground	21	IGND	--	Ext. power ground
4	RDY	I	Servo ready signal	14	BRAKE-	O	Brake signal(-)	22	IGND	--	Ext. power ground
5	OUT-	O	Pulse signal (-)	15	IGND	--	Ext. power ground	23	DIR-	O	Dir. Signal(-)
6	OUT+	O	Pulse signal (+)	16	EB-	I	Encoder B-phase(-)	24	DIR+	O	Dir. Signal(+)
7	EA-	I	Encoder A-phase(-)	17	EB+	I	Encoder B-phase(+)	25	EZ-	I	Encoder Z-phase(-)
8	EA+	I	Encoder A-phase(+)	18	IGND	--	Ext. power ground	26	EZ+	I	Encoder Z-phase(+)
9	BRAKE+	O	Brake signal(+)								

NOTE

ALM_RST / DO: You may set this signal to general purpose digital output signal (EDO) or alarm clearance function (ALM_RST) by switch S1 or S2.

2.8.3 S1, S2: EDO/ALM_RST Selection Switch



ON = Reset servo driver

S1	**1**	**2**	
	ON		**Reset servo driver #1**
		ON	**Reset servo driver #2**
S2	**1**	**2**	
	ON		**Reset servo driver #3**
		ON	**Reset servo driver #4**

OFF is default

DIN-825-GP4 is equipped with 4 servo drive reset signals. You may set up CMA1~CMA4 PIN 10 and CMP1~CMP4 PIN 10 for servo drive rest or J6 connector DO.1~DO.4 by switch S1 and S2.

3 Signal Connection

PCI-8254/PCI-8258 must connect to servo or stepper motor drive with exclusive terminal board DIN-825-GP4. All optical isolation circuit of mechanical relevant I/O and servo relevant I/O are set to DIN-825-GP4 to prevent damages to primary controller PCI-8254/PCI-8258 from any invalid signal connection to it. This may effectively reduce difficulties and times required in replacing controller relevant products when doing customer service maintenance tasks. See sections below for detailed descriptions on precautions required to connect to various mechanical I/O and servo I/O signals. Contents:

3.1 Analog Control Command Signal

3.1.1 Single-ended Type Signal: AOUT+

PCI-8254/PCI-8258 provides 4/8 analog control command channels respectively. Each analog command supports 16-bit resolution and provides ±10V output range at accuracy smaller than ± 1mV. Each analog control command can be set to single ended or differential output mode by adjusting switch S1 to be used by market available Japanese/Taiwanese and American/EU servo motor drives. In general, a servo drive can be set to P/S/T (position/speed/torque) mode. When control mode is set to S/T the positioning function can be exercised by controlling motor speed or torque with analog commands. When servo drive is set to P mode, then pulse command of PCI-8254/PCI-8258 will be used for open-loop control (see Section 3.2). This analog channel is then defined as general purpose output one.

See below for corresponding pins of analog command output to DIN-825-GP4:

CMAx Pin No (x=1~4)	Signal Name	Description (n=1~8)	Axis #
6	AOUT+	Analog Out Signal, (+) (n)	1~8


NOTE

PCI-8258 need two DIN-825-GP4 for eight axes motion control functions
1 controls axes 1 ~ 4 and #2 controls axes 5 ~ 8

3.1.2 Single-ended Type Signal: AOUT+, AOUT-

4/8 analog control commands of PCI-8254/PCI-8258 can be set to single ended or differential output mode by adjusting switch S1 to be used by market available Japanese/Taiwanese and American/EU servo motor drives. See below for corresponding pins of analog command output pins against differential analog signals to DIN-825-GP4:

CMAx Pin No (x=1~4)	Signal Name	Description (n=1~8)	Axis #
6	AOUT+	Analog Out Signal, (+) (n)	1~8
5	AOUT-	Analog Out Signal, (-) (n)	1~8

NOTE

PCI-8258 need two DIN-825-GP4 for eight axes motion control functions
1 controls axes 1 ~ 4 and #2 controls axes 5 ~ 8

- **Signal connection template diagram:**

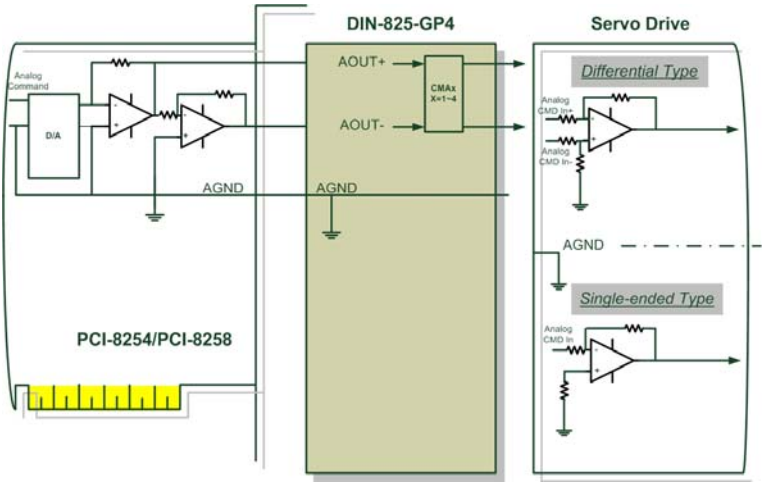


Figure 3-1: Connection example of differential analog output signal

3.2 Pulse Command

In addition to the analog command outputs described in Section 3.1, PCI-8254/PCI-8258 provides 4/8 pulse control command channel. Each pulse control command can support up to 6.5MHZ output frequency.

In general, a servo drive can be set to P/S/T (position/speed/ torque) mode. When control mode is set to P mode, then the pulse command control of PCI-8254/PCI-8258 will be used for open-loop control.

In addition to servo drive, the Stepper drive also employs pulse command interface as the primary control input commands. See below for corresponding pins of pulse command output against differential pulse signals to DIN-825-GP4:

CMAx Pin No (x=1~4)	Signal Name	Description (n=1~8)	Axis #
6	OUT+	Pulse signal, (+) (n)	1~8
5	OUT-	Pulse signal, (-) (n)	1~8
24	DIR+	Dir. Signal, (+) (n)	1~8
23	DIR-	Dir. Signal, (-) (n)	1~8

NOTE

PCI-8258 need two DIN-825-GP4 for eight axes motion control functions
1 controls axes 1 ~ 4 and #2 controls axes 5 ~ 8

Either servo motor drive or stepper motor drive employs one of the two input interfaces described below:

1. Line Driver input interface provides better anti noise-resistant and longer wiring length.

- **Signal connection diagram:**

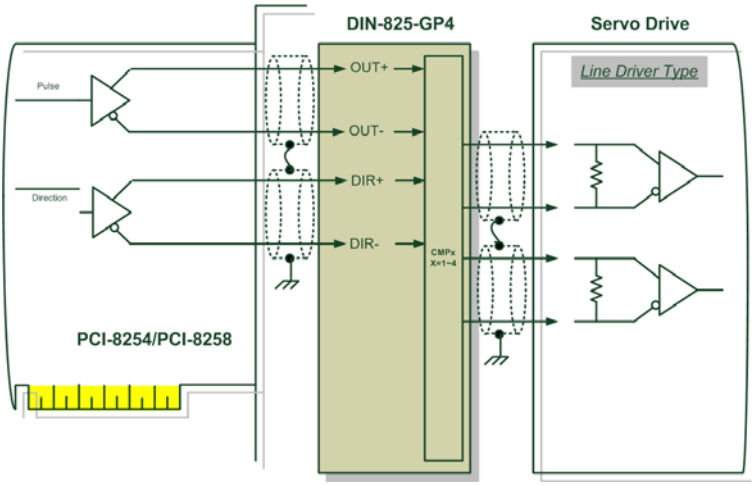


Figure 3-2: Line Driver type pulse control command signal connection example

2. Open-Collector input interface can increase passing current capacity of signal by adjusting pull-up resistance value at the shorter wiring length.

- **Signal connection diagram:**

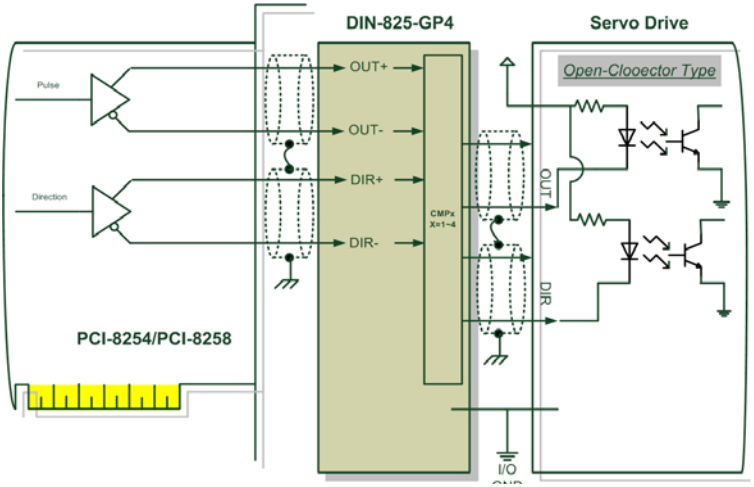


Figure 3-3: Open-Collector type pulse control command signal connection example

CAUTION

To avoid damages to Line Driver components on controller caused by invalid wiring please connect the OUT-, DIR- pins of controller to OUT, DIR pins of motor drive.

CAUTION

The controller employs Line Driver component -26LS31 with maximum Sink Current at 20mA. Do not use it at current above this value, the component may be damaged otherwise.

3.3 Encoder Input, EA & EB & EZ

PCI-8254/PCI-8258 provides 4/8 encoder input channels respectively which accept single end input frequency up to 5MHz with each channel containing EA, EB, and EZ signal. Each group of EA, EB, and EZ signal contains a pair of differential signal, e.g. the EA signal contains EA+ and EA-. See Section 4.1.1.4 for how to use the encoder. See below for corresponding pins of encoder input on DIN-825-GP4

CMAx Pin No (x=1~4)	Signal Name	Description (n=1~8)	Axis #
8	EA+	Encoder A-phase (+),(n)	1~8
7	EA-	Encoder A-phase (-),(n)	1~8
17	EB+	Encoder B-phase (+),(n)	1~8
16	EB-	Encoder B-phase (-),(n)	1~8
26	EZ+	Encoder Z-phase (+),(n)	1~8
25	EZ-	Encoder Z-phase (-),(n)	1~8


NOTE

PCI-8258 need two DIN-825-GP4 for eight axes motion control functions
1 controls axes 1 ~ 4 and #2 controls axes 5 ~ 8


CAUTION

The controller employs Line Receiver component -26LS32 with maximum Sink Current at 20mA@5V. Do not use it at current above this value, the component may be damaged otherwise.

- **Signal connection diagram:**

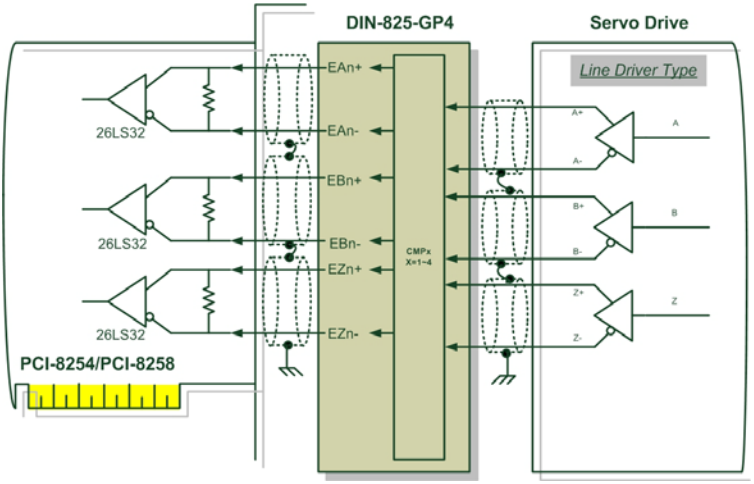


Figure 3-4: Line driver type encoder input signal connection example

3.4 Emergency Stop Input

PCI-8254/PCI-8258 provides one hardware input emergency stop signal (EMG). If the external emergency stop signal is triggered, all motion control commands will be stopped immediately. In addition, the DIN-825-GP4 is designed to transmit external emergency stop signal to servo/stepper motor drive to stop operation of every motor immediately. See below for corresponding pins of emergency stop signal input on DIN-825-GP4:

J5 Pin No	Signal Name	Description	Axis #
5	EEMG	External emergency stop input (external input)	-

CMPx / CMAx Pin No (x=1~4)	Signal Name (n=1~8)	Description	Axis #
19	EMG(n)	Emergency stop (output to drive)	1~8

NOTE

PCI-8258 need two DIN-825-GP4 for eight axes motion control functions
1 controls axes 1 ~ 4 and #2 controls axes 5 ~ 8

- **Signal connection diagram:**

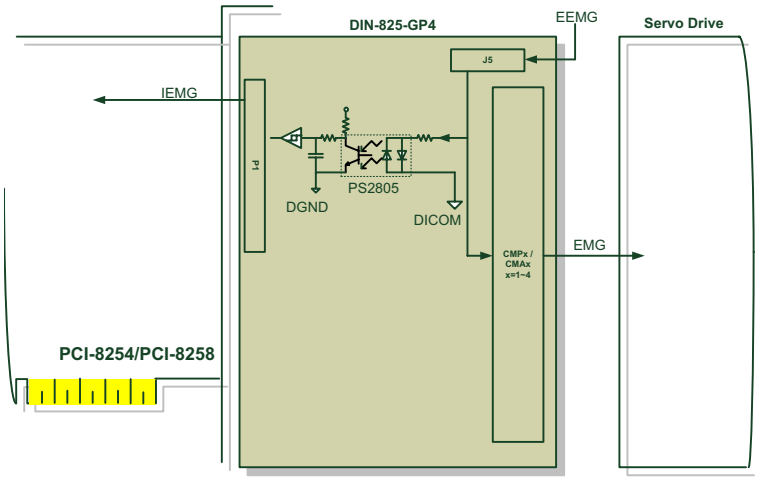


Figure 3-5: Emergency stop signal connection example

3.5 PEL/MEL Input

PCI-8254/PCI-8258 provides 4/8 End-limited switch input channels. The Plus Limited Switch (PEL) is used as the mechanical protection switch for movement in the positive direction. When this switch is triggered, the PCI-8254/PCI-8258 stops its positive direction movement immediately. The Minus Limited Switch (MEL) is used as the mechanical protection switch for movement in the negative direction. When this switch is triggered, the PCI-8254/PCI-8258 stops its negative direction movement immediately. See below for corresponding pins of mechanical limit switch signal input on DIN-825-GP4:

J1/J2 Pin No	Signal Name	Description	Axis #
3	PEL(3) / PEL(1)	Plus limit switch input (3) / (1)	3 / 1
7	PEL(4) / PEL(2)	Plus limit switch input (4) / (2)	4 / 2
5	MEL(3) / MEL(1)	Minus limit switch input (3) / (1)	3 / 1
9	MEL(4) / MEL(2)	Minus limit switch input (4) / (2)	4 / 2


NOTE

PCI-8258 need two DIN-825-GP4 for eight axes motion control functions
1 controls axes 1 ~ 4 and #2 controls axes 5 ~ 8

- **Signal connection diagram:**

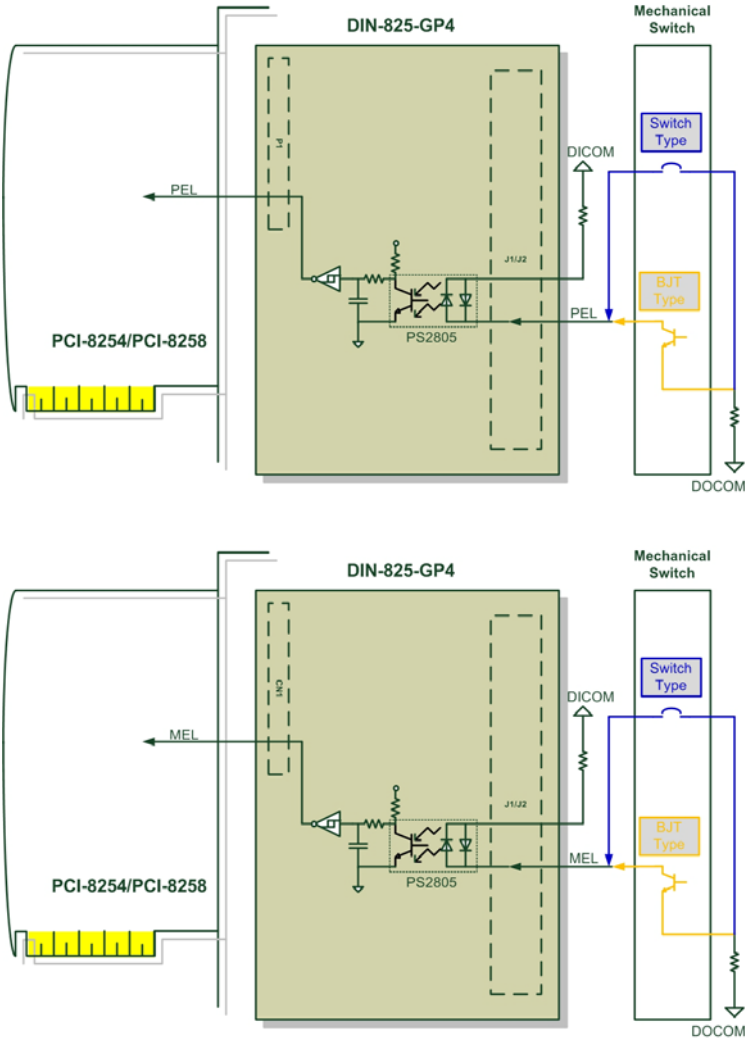


Figure 3-6: Mechanical limit switch signal connection example

3.6 ORG Input

PCI-8254/PCI-8258 provides 4/8 original position switch input channels. Working together with the home movement described in Section 4.3, this function returns the body to its original position (also known as the zero position). See below for corresponding pins of original position switch signal input on DIN-825-GP4:

J1/J2 Pin No	Signal Name	Description	Axis #
4	ORG(3) / ORG(1)	Original position switch input (3) / (1)	3 / 1
8	ORG(4) / ORG(2)	Original position switch input (4) / (2)	4 / 2

NOTE

PCI-8258 need two DIN-825-GP4 for eight axes motion control functions
1 controls axes 1 ~ 4 and #2 controls axes 5 ~ 8

- **Signal connection diagram:**

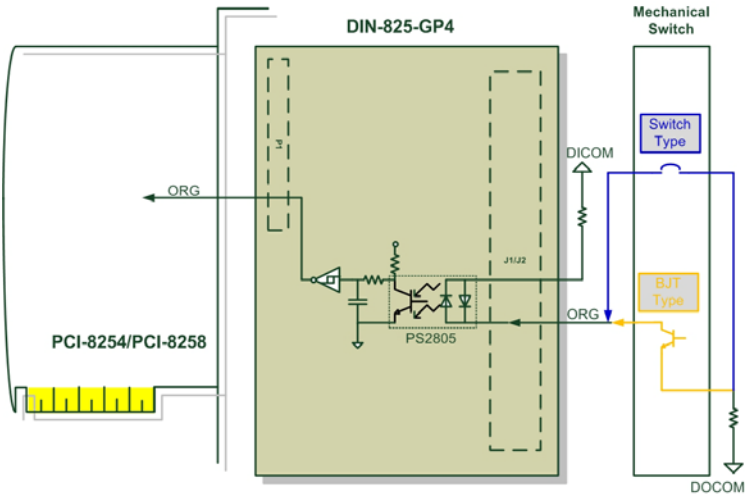


Figure 3-7: Original position switch signal connection example

3.7 INP / ZSP Input

PCI-8254/PCI-8258 provides 4/8 In-position (INP) or zero speed detection (Zero-speed (ZSP)) input channel. Working with function described in Section 4.8, it can be used as the trigger source for in-position events of individual motion. In general, when servo drive is set to position mode (P mode), the servo issues a (INP) pulse signal to controller when movement get into position. When servo drive is set at speed/torque (S/T mode), the actuator sends a (ZSP) pulse signal to controller when motor speed is zero. See below for corresponding pins of in-position or zero speed detection signal input on DIN-825-GP4:

CMAx Pin No (x=1~4)	Signal Name (n=1~4)	Description	Axis #
2	ZSP(n)	Zero speed detection signal	1~4
2	INP(n)	In-position Input (for pulse output mode only)	1~4

NOTE

PCI-8258 need two DIN-825-GP4 for eight axes motion control functions
1 controls axes 1 ~ 4 and #2 controls axes 5 ~ 8

- **Signal connection diagram:**

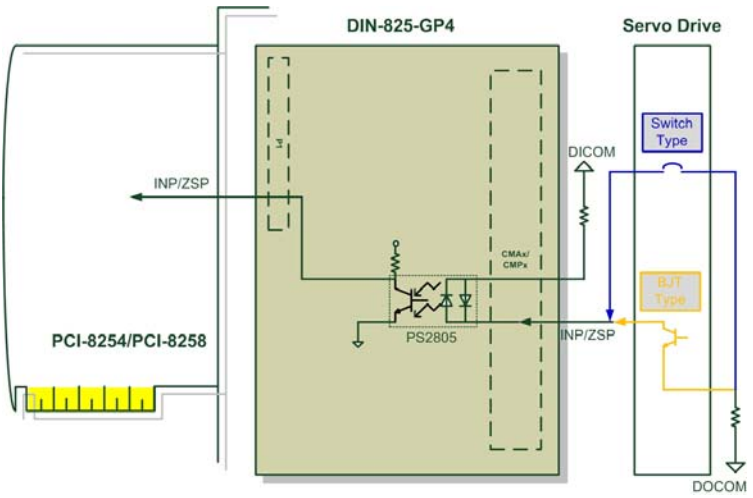


Figure 3-8: Place / zero speed detection signal connection example

3.8 ALM Input

PCI-8254/PCI-8258 provides 4/8 servo alarm input channels. Working with function described in Section 4.11 it can be used as the trigger source for motion interrupt event. In general, when abnormality is encountered during servo drive movement, it issues an (ALM) pulse signal to controller for abnormality occurrence. See below for corresponding pins of servo alarm signal input on DIN-825-GP4:

CMAx / CMPx Pin No (x=1~4)	Signal Name (n=1~4)	Description	Axis #
11	ALM(n)	Servo alarm input	1~4

NOTE

PCI-8258 need two DIN-825-GP4 for eight axes motion control functions
1 controls axes 1 ~ 4 and #2 controls axes 5 ~ 8

- **Signal connection diagram:**

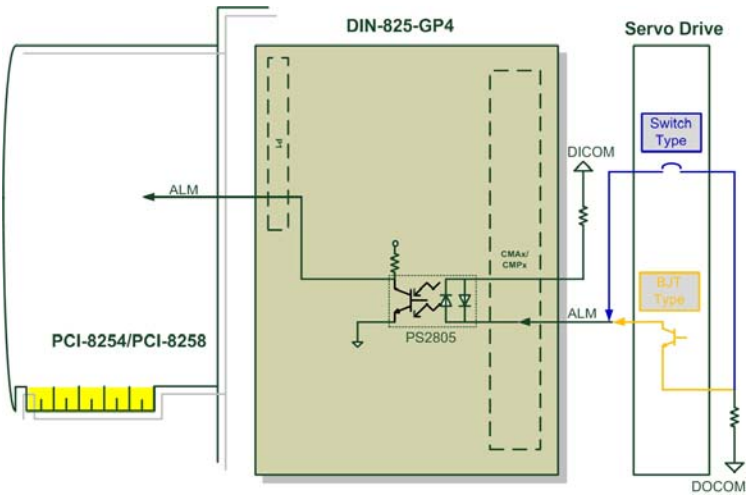


Figure 3-9: Servo alarm signal connection example

3.9 SVON Output

PCI-8254/PCI-8258 provides 4/8 servo-on output channels and utilize the servo-on signal to enable servo drive for pulse or analog commands input. If there is abnormality encountered during movement, the PCI-8254/PCI-8258 turns off this signal automatically and stops all motion control commands. See below for corresponding pins of servo-on signal output on DIN-825-GP4:

CMAx / CMPx Pin No (x=1~4)	Signal Name (n=1~4)	Description	Axis #
1	SVON(n)	Servo-on output	1~4

NOTE

PCI-8258 need two DIN-825-GP4 for eight axes motion control functions
1 controls axes 1 ~ 4 and #2 controls axes 5 ~ 8

- **Signal connection diagram:**

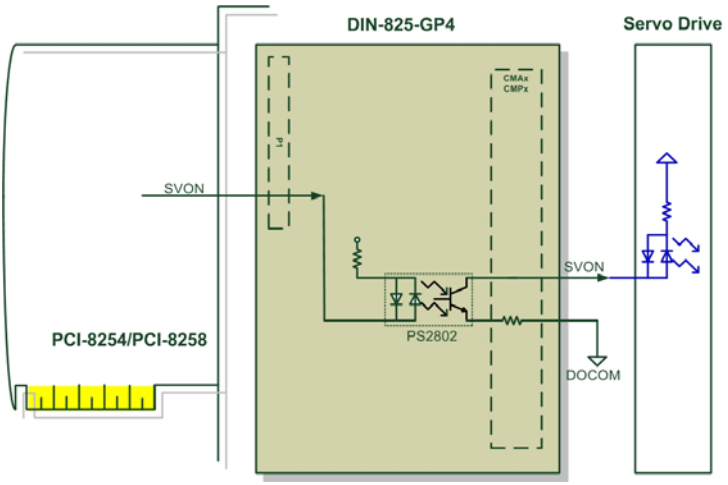


Figure 3-10: Servo-on signal connection example

3.10 Analog Input Signals

PCI-8254/PCI-8258 provides 4/8 analog input channels with 12-bit resolution and 100KHz sampling rate. You may use these analog input channels to get values from voltage sensor. See below for corresponding pins of analog input signals on DIN-825-GP4:

J3 Pin No	Signal Name	Description
10	AIN1	Analog input
9	AIN2	Analog input
8	AIN3	Analog input
7	AIN4	Analog input

NOTE

Your PCI-8258 must come with two DIN-825-GP4 to get eight analog input signals where the first gets AIN1~AIN4 and the second ANI5~AIN8.

- **Signal connection diagram:**

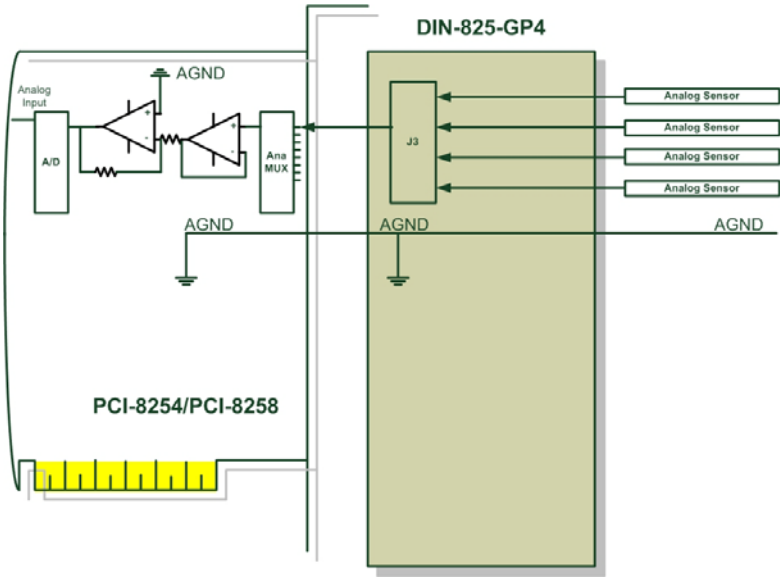


Figure 3-11: Analog input signal connection example

3.11 Compare & Trigger Output

PCI-8254/PCI-8258 provides 2/4 comparing trigger pulse output channels. Each comparing trigger channel can output pulse commands up to 1 MHZ. See Section 4.9.2 for its detail and how to use it. See below for corresponding pins of pulse command output against differential pulse signals to DIN-825-GP4:

J3 Pin No	Signal Name	Description
2	TRG2-/TRG4-	Trigger output (-), (2)/(4)
3	TRG2+ / TGR4+	Trigger output (+), (2)/(4)
4	TRG1-/TRG3-	Trigger output (-), (1)/(3)
5	TRG1+/TRG3+	Trigger output (+), (1)/(3)

NOTE

The compare trigger pulse output channel of PCI-8254/PCI-8258 employs line driver output interface for better noise signal resistance and wiring length where trigger output (3) & (4) require #2 DIN-825-GP4 for wiring.

- **Signal connection diagram:**

1. Line Driver interface:

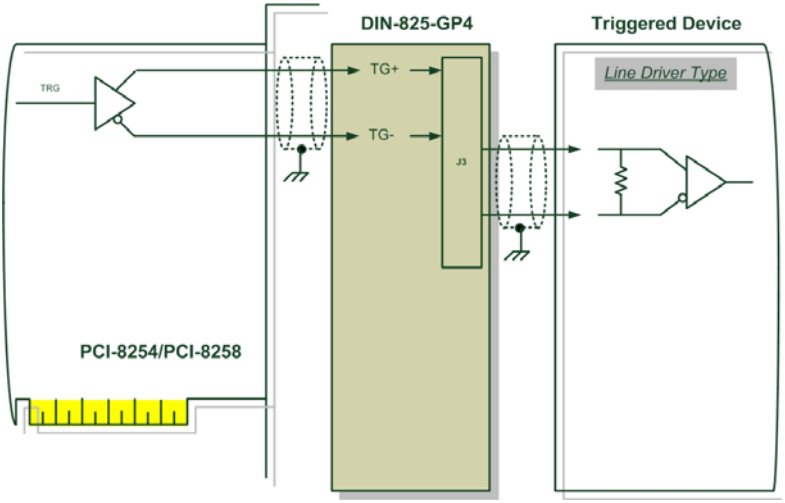


Figure 3-12: Line Driver type compare trigger signal connection example

2. Open-Collector interface:

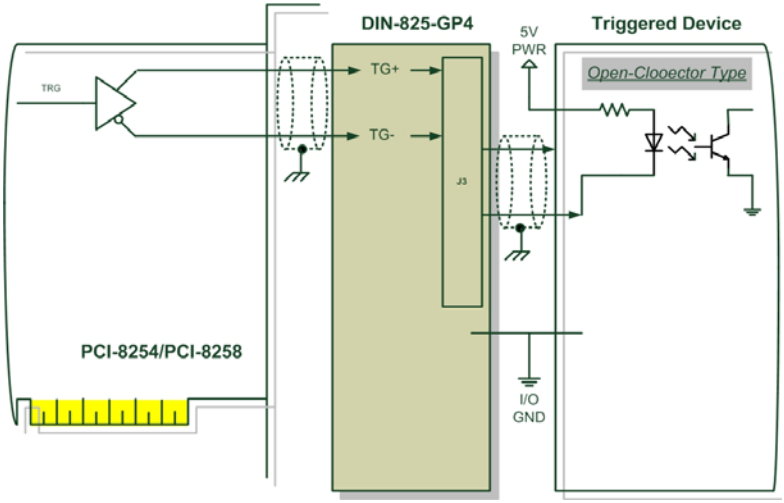


Figure 3-13: Open-Collector type compare trigger signal connection example

3.12 Digital Output/Input

PCI-8254/PCI-8258 provides 20/24 digital output/input channels. See below for corresponding pins of general purpose digital input and output signals on DIN-825-GP4:

J1/J2 Pin No.	Signal Name	Description
2	EDI(3) / EDI (1)	General purpose digital input signal (3), (1)
6	EDI(4) / EDI (2)	General purpose digital input signal (4), (2)

J6 Pin No.	Signal Name	Description
1	EDO(1)	General purpose digital output signal (1)
2	EDO(2)	General purpose digital output signal (2)
3	EDO(3)	General purpose digital output signal (3)
4	EDO(4)	General purpose digital output signal (4)

NOTE

PCI-8258 need two DIN-825-GP4 for eight axes motion control functions
1 controls axes 1 ~ 4 and #2 controls axes 5 ~ 8

NOTE

1. Please connect DICOM to external power supply (24VDC in general) if possible.
2. Please connect DOCOM to ground (GND) of external power supply if possible.

- **Signal connection diagram:**

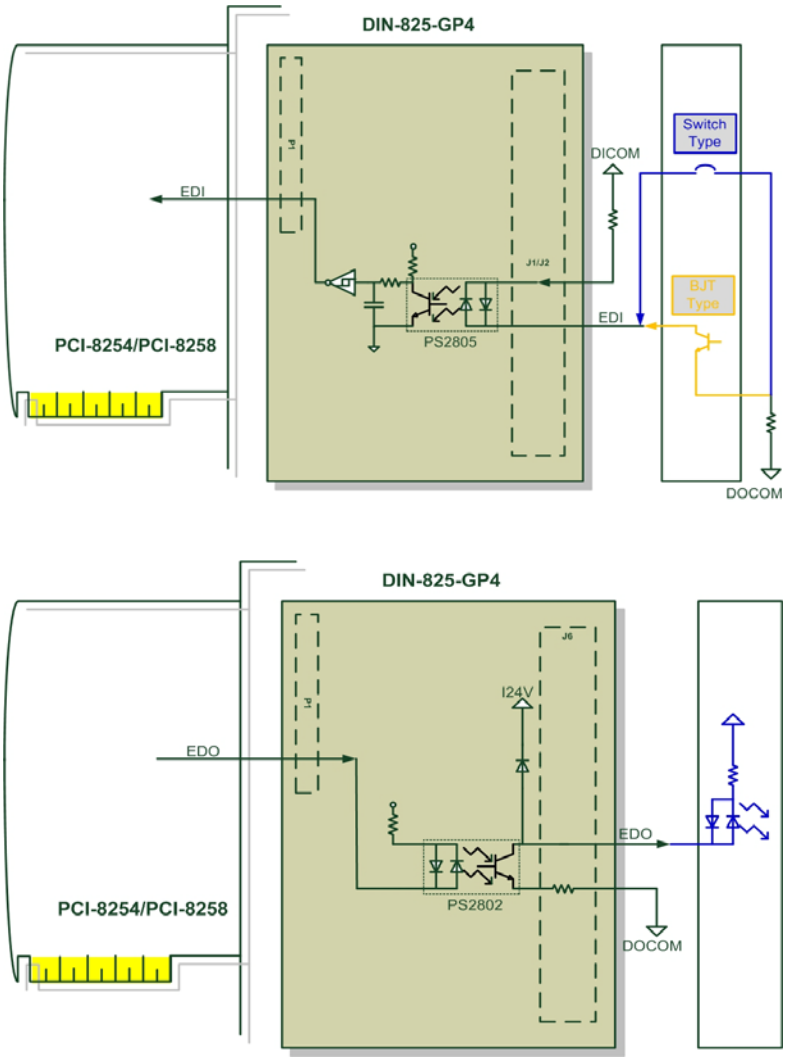


Figure 3-14: General purpose digital I/O signal connection example

IOIF1 Pin No.	Signal Name	Description
1~8	DI(1)~(8)	General purpose IOIF2 digital input signal (1)~(8)

IOIF2 Pin No.	Signal Name	Description
1~8	DI(9)~(16)	General purpose digital input signal (9)~(16)

IOIF3 Pin No.	Signal Name	Description	Axis #
※1~5	DO(1)~(5)	General purpose digital output signal (1)~(5)	-


NOTE

※The digital output current may reach 250mA

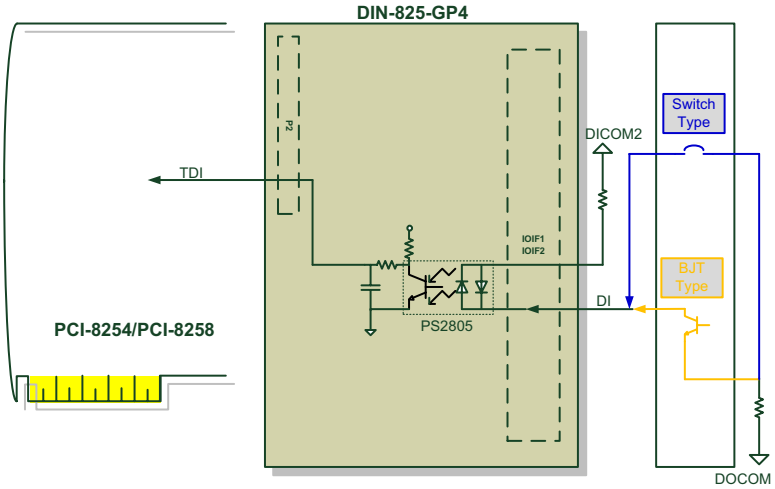
IOIF3 Pin No.	Signal Name	Description	Axis #
6~8	DO(6)~(8)	General purpose digital output signal (6)~(8)	-

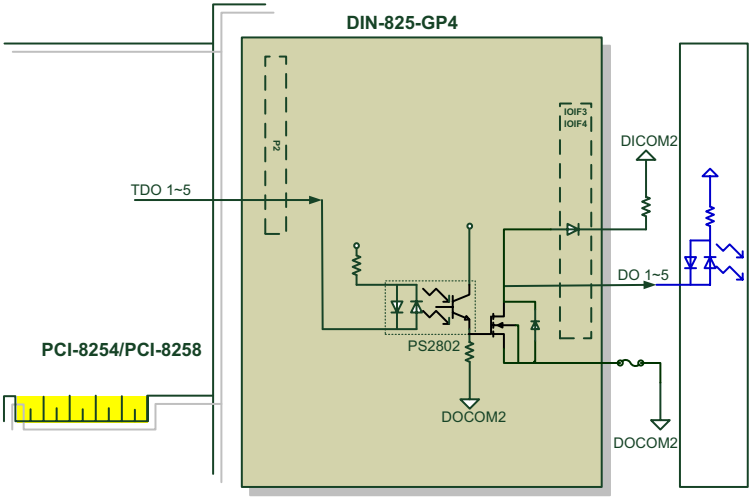
IOIF4 Pin No.	Signal Name	Description	Axis #
1~8	DO(9)~(16)	General purpose digital output signal (9)~(16)	-


NOTE

1. Please connect DICOM to external power supply (24VDC in general) if possible.
2. Please connect DOCOM to ground (GND) of external power supply if possible.

- **Signal connection diagram:**





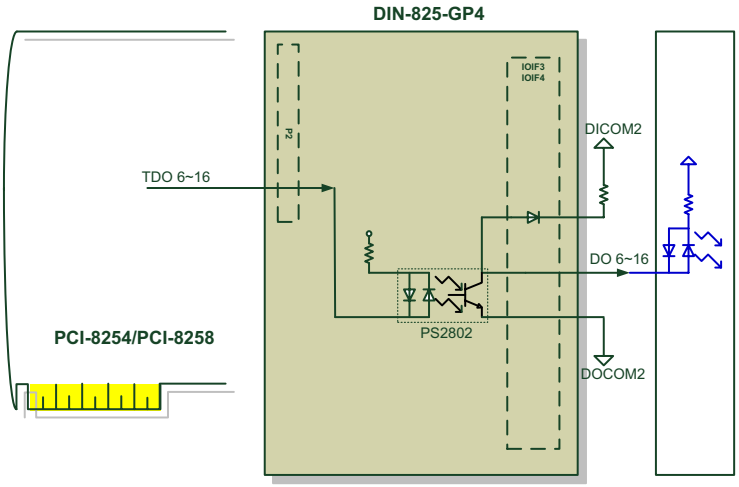


Figure 3-15: General purpose digital I/O signal connection example

4 Motion Control Theory

This chapter introduces you the motion control function of PCI-8254 / PCI-8258 as well as relevant precautions in using them. Contents:

4.1 Motion Control Mode and Interface Overview

This section describes basic setups of "PCI-8254" and "PCI-8258" before doing motion control and fundamental concepts of its core operations.

4.1.1 Motion Control Interface

4.1.1.1 Control Mode and Output Interface

You may use the **MotionCreatorPro2** application program to set up these two output interface and save your desired setting in non-volatile memory, the so-called ROM, of the controller for automatic loading when the controller power on. You may use API listed below to retrieve current settings to ensure their correctness.

APS_get_eep_curr_drv_ctrl_mode ()

4.1.1.2 Pulse Type

You can use this control mode to control stepper motor or set it to P control mode to control servo motor with pulse format signal input. The output interface of controller is OUT / DIR [1..8] pins. (Please refer Chapter 3 for detail.)

This is the so called open-loop or semi closed-loop control model where the upper controller output position command in digital pulse format signal to lower stepper motor or servo motor to form a close-loop control in servo drive. In this mode, number of pulses indicates actual distance traveled by the machine (vary with the relation between mechanical shift and pulse) while the pulse output frequency indicates speed of the machine traveling at (in unit of pulse per second, PPS).

In this mode users must pay special attention to pulse signal format acceptable to the motor to be driven. The motor works normally only when being driven by correct pulse format signal, otherwise the motor may fail to work in erroneous direction or with abnormal shaking. Users must correctly set up the controller before any motion control after the software is initialized. This controller provides two pulse signal output format:

- OUT / DIR signal format: At this mode, the OUT signal indicates frequency and amount of output pulses where DIR indicates direction of machine movement.
- CW / CCW signal format: At this mode the CW/CCW signal indicates both machine movement direction and pulse output frequency and amount

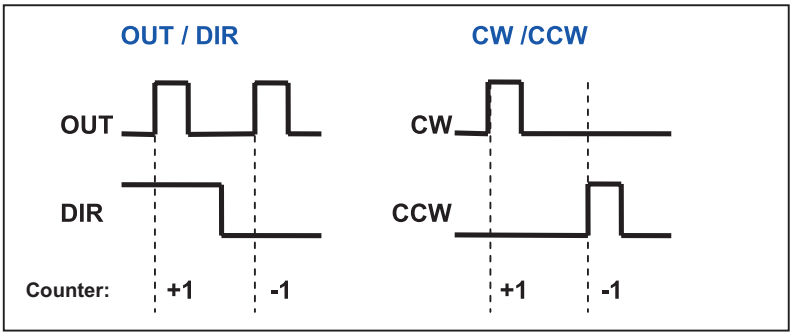


Figure 4-1: Format of pulse signal

The signal format can be set up in axis parameters:

Param. No.	Define symbol	Description
81h (129)	PRA_PULSE_OUT_MODE	Pulse output format setup

4.1.1.3 Analog Type

The analog control mode is used to control servo motor in velocity mode or torque mode that accepts signals in analog voltage.

In this mode the closed-loop function will be initiated automatically. See figure below for closed-loop illustration. See Chapter 2: Close loop control for relevant PID close loop function. Adjustments

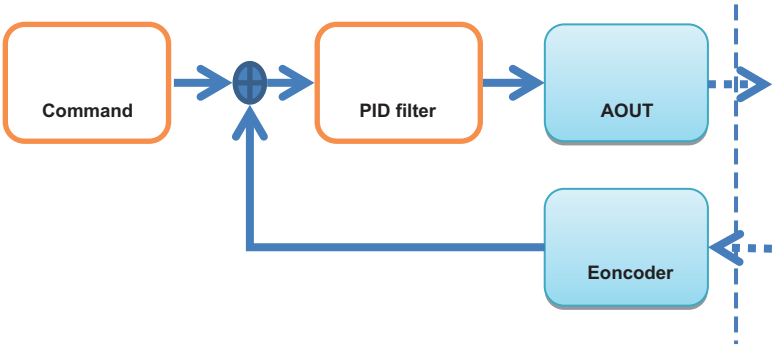


Figure 4-2: Illustration of analog command output

Set up input/output interface correctly before controlling with close loop PID controller:

1. Make sure the AOUT voltage control level and logic ailing with the receiving end of servo motor.
   See Section 4.2 Close loop control for its setup.
2. How to set up Decoding method used by the encoder.
   Please refer to the Encoder setup in next Section.

NOTE

For axis with motion control interface set to analog output its analog output channle will be controlled by the controller itself rather than by operators. For axis with motion control interface set to pulse control interface, its output control is released by the controller to the users for controlling the analog output channel by programs.

4.1.1.4 Encoder

The position encoder of this controller supports 9 kinds of digital signal input formats as described below.

CAUTION

Please set up the position encoder before doing motion control. This is especially true for analog output type closed-loop control as invalid setup may lead to motor burst.

NOTE

You may set up and test your controller with MotionCreatoPro 2 software. You can check this by manually spinning the motor (or move the workbench) and verify the encoder signal from motor or linear scale to the controller.

No	Decode Mode	Positive direction		Negative direction	
		EA	EB	EA	EB
	OUT/DIR (1)		High		Low
	CW/CCW (1)		Low	Low	
	1X AB	EA / EB		EA / EB	
	2x AB	EA / EB		EA / EB	
	4x AB	EA / EB		EA / EB	
	OUT/DIR (2)		High		Low
	OUT/DIR (3)		Low		High
	OUT/DIR (4)		Low		High
	CW/CCW (2)		High	High	

Table 4-1: Encoder input format

- **Axis parameter setup:**

Param. No.	Define symbol	Description
80h (128)	PRA_ENCODER_MODE	Encoder input signal format
85h (133)	PRA_ENCODER_DIR	Encoder counting direction setup

Table 4-2: Encoder input format

- **Axis parameter setup API:**

  APS_set_axis_param (); // write in axis parameter

  APS_set_axis_param (); // read out axis parameter

4.1.1.5 Motion Control I/O

Some motion control I/O signal of this controller definition are summarized in table below:

Param.	Defined Symbol	Type	Description
0	ALM	Input	Servo alarm
1	PEL	Input	Plus end limit
2	MEL	Input	Minus end limit
3	ORG	Input	Home input
4	EMG	Input	Emergency stop input
5	EZ	Input	Servo index input
6	INP	Input	In-Position input
7	SVON	Output	Servo ON output status

Here ALM, EZ and INP are signals sent by servo drive and SVON (Servo on) signal is the receiving signal of servo drive for driving the servo motor.

And PEL, MEL, ORG and EMG are mechanical I/O signals. Safety relevant signals, e.g. EMG, PEL and MEL are used to work together with motion control. Take example, the home movement will use ORG, PEL, MEL, EZ and other signals.

You may use following API functions to get I/O status with each bit of the parameter representing status of the axis motion control I/O.

I32 APS_set_servo_on (I32 Axis_ID, I32 Servo_on);

I32 APS_motion_status (I32 Axis_ID);

- **Signal direction**

  These signal logic may be inverted by software. Relevant axis parameters are listed below:

- **Board parameter**

Param. No.	Define symbol	Description
00h (0)	PRA_EL_LOGIC	PEL/MEL input logic
01h (1)	PRA_ORG_LOGIC	ORG input logic
04h (4)	PRA_ALM_LOGIC	Set ALM logic
05h (5)	PRA_ZSP_LOGIC / PRA_INP_LOGIC	Set INP logic
06h (6)	PRA_EZ_LOGIC	Set EZ logic

- **Board parameter**

Param. No.	Define symbol	Description	Value	Default
00h (0)	PRS_EMG_LOGIC	EMG input logic	0 : Not inverse 1 : Inverse	

- **Filter**

  This controller features filters to screening out High-Frequency Noise against motion control I/O to prevent abnormal motion control action caused by external noise. The filter is defaulted at ON status.

4.1.2 Control Cycle

The controller features three control cycle for different works. They are:

1. Servo control cycle
2. Motion control cycle
3. Host control cycle

4.1.2.1 Servo Control Cycle

The servo control cycle is the time required to complete one close loop control. Servo control cycle of this controller can be up to 20KHz, that is 50 microsecond for each cycle. In each control cycle, the controller finish various servo control relevant jobs including PID compensation and filter compensation.

4.1.2.2 Motion Control Cycle

Default motion control cycle is set at 1KHz, i.e. 1 millisecond for each cycle. Varieties of peripheral hardware components controls, including host communication, trajectory calculation, AMC program execution and data sampling, are finished in it.

4.1.2.3 Host Control Cycle

Default host control cycle is 0.5 KHz. That is, it takes 2 millisecond to finish jobs in one control cycle including communications between hosts, watch dog, kernel update, parameter management and other non-realtime jobs.

The servo control cycle runs independently while the motion control and host control cycle are done in the same processor. The controller completes scheduled jobs automatically with the motion control ones has higher priority than the host control one.

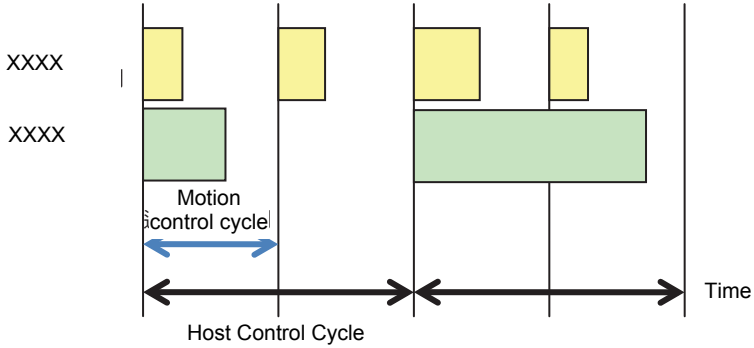


Figure 4-3: Control cycle

The motion program is executed in motion control cycle to control jobs to be executed in each motion control cycle directly for more precise completion of realtime jobs. Please pay attention to DSP loading when doing this.

Loading of CPU in controller is hard to predict as the controller is affected by many factors, e.g. external signals, user operations, and algorithm process during its operations. In most cases, please try to keep CPU utilization rate to below 70% and reserve 30% of CPU capacity to the processing of system jobs and momentary work loads.

Overloading (work loads exceed control cycle) may lead to unpredictable results. This controller provides you with some functions and tools to monitor processor utilization rate and adjust control procedures. In case of any processor overloading, the controller logs and warns (interrupt, please refer to section of interrupt) that you may take for proper responses in your program.

How to use API:

get_motion_control_timing () // get usage amount of current motion control cycle
get_max_ motion_control_timing ()// get maximum usage amount of motion control cycle
get_motion_control_timing () // get usage amount of current host control cycle
get_motion_control_timing () // get maximum usage amount of host control cycle
reset_max_motion_control_timing()
reset_max_host_control_timing()
get_over_cycle_event()
get_over_cycle_count()
reset_over_cycle_count ()

4.2 Closed-loop Control

4.2.1 Close-loop Control Overview

The close-loop control system works like this: after a command is sent, a group of sensors get system output signals during motion process and returned to the controller, a error signal then can derived by comparing the original command against the feedback signal which is then returned to controller. As the controller structure is designed on the basis of pre-defined System Dynamic, the deviation signal received by the controller will be combined into a brake signal and feed into the brake to reduce deviations and external interruptions and noises such that system output may comply with given commands gradually.

In general, commonly adopted controllers by the manufacturing industries are of PID controller on account of its simple structure and its capabilities in meeting most industrial control requirements. Thanks to its PID+ velocity and acceleration feed forward design, this controller provides much improved overall control performance with servo update rate up to 20KHz.

Below we first discuss the PID controller, a continuous-time standard PID controller mathematical form

$$u(t) = K_c \left[e(t) + \frac{1}{T_i} \int_0^t e(\tau) d\tau + T_d \frac{de(t)}{dt} \right]$$

where $u(t)$ is the combined control amount of Proportional control, Integral control, and Derivative control, K_c the gain, T_i the integral time constant, and T_d the derivative time constant. Impact of each control item on system performance is described below.

a Proportional Control

As Time-domain is concerned, more Proportional gain may speed up responses as the system bandwidth increases with that of Frequency-domain at the expense of Stability. This makes the system prone to vibration as Gain margin decreases. Another important function of proportional control is to reduce Steady-state error.

b Integral Control

Integral control can reduce Steady-state error and suppress noise at the expense of system Response time.

c Derivative Control

The derivative control improves Temporary response time and relative steadiness but helps little in reducing noise and steady state error.

d PI-Control

In terms of frequency domain, the PI controller increases system low frequency range for reduced steady state error at the expense of poorer system response speed caused by phase lag.

e PD-Control

In addition, phases of increased high frequency range may speed up system responses of PD controller. However, this may lead to more Noise impact as a result of high frequency gain.

f PID-Control

A PID controller is a PI and PD combined controller. The PID controller combines advantages of both PI and PD controllers to improve steady state error without sacrificing system response speed.

See Figure 4-4 of the close loop control system structure of this controller, see Table 1 for descriptions on all the close loop control relevant axis parameters. For actual application, the controller must realize the transformation, reference to above descriptions, from continuous time standard pattern to discrete time pattern as described in mathematical equation illustrated below

$$U(k) = \text{PRA\_KVFF\_G AIN} \cdot CmdVel(k)$$

$$+ \text{PRA\_KAFF\_G AIN} \cdot CmdAcc(k)$$

$$+ \text{PRA\_KP\_GAIN} \cdot E(k)$$

$$+ \text{PRA\_KD\_GAIN} \cdot \big(E(k) - E(k-1)\big)$$

$$+ \text{PRA\_KI\_GAIN} \cdot \sum_{i=0}^{k} E(i)$$

$$E(k) = CmdPos(k) - FbkPos(k)$$

$$CmdVel(k) = CmdPos(k) - CmdPos(k-1)$$

$$CmdAcc(k) = CmdVel(k) - CmdVel(k-1)$$

Here $CmdPos(k)$ represents a position command, $FbkPos(k)$ represents position feedback, $CmdVel(k)$ represents speed command and $CmdAcc(k)$ represents acceleration command. The PID control plus velocity / acceleration command feedforward control are added here to reduce tracking error of position command and improve control performance. The Plant in control shall receive voltage signal after processed by controller. Its physical meanings may be velocity command or torque command determined by settings of actuator.

On the other hand, after servo update speed of PID controller is changed, user must regulate controller gain at the same. You may change servo update speed in the Setup Wizard page in range of 50us~2000us. The DSP reboots after setup. The default update speed is 250us. As the controller's gaining value is limited, the working range of gaining value can be adjusted with the gain shift function as shown in the gain-shift figure below. The gain shift function double enlarges or reduces the gaining value. Take example. You can set KP gain to 100, KP shift to 2, and PID shift to 3 to get a equivalent KP of 100×22×23=3200. In general, the gain shift value should be changed only when its range is short of required control power. The auto servo parameter fine-tuning program described in next chapter may can help you in selecting proper initial gain shift value.

In addition to changes in all gain values of controller, the Integral situation, output voltage limit, output voltage offset, and output voltage inverse can be set up by users. To suppress mechanical vibration effectively, two Biquad filters are connected in series to the rear of the controller for Low pass filter and Notch filter. Users may design their own biquad filters for their own signal processing purposes. This will be discussed in detail in following chapters.

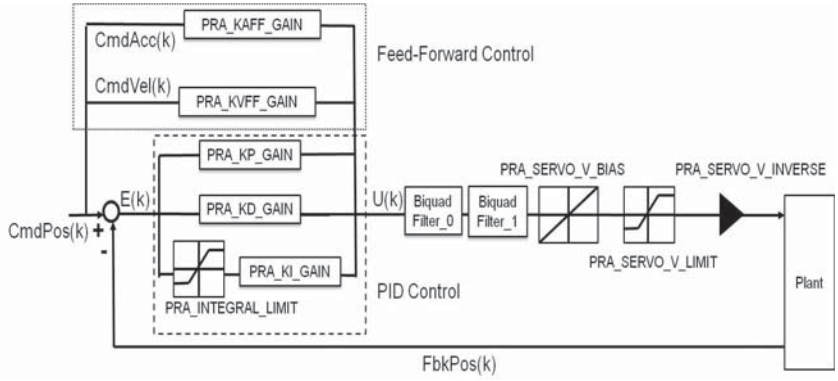


Figure 4-4: PCI-8254/PCI-8258 close loop control structure diagram

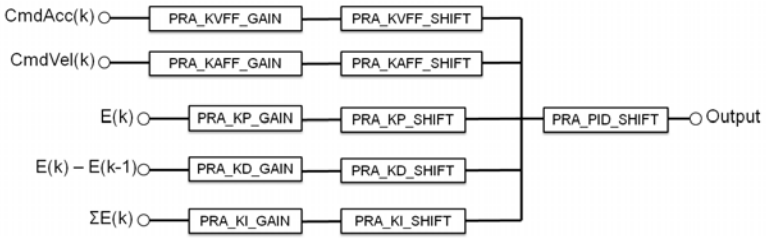


Figure 4-5: Gain and Gain shiftrelationship diagram

Control circuit relevant axis parameter table:

Param. No.	Axis parameter name	Descriptions	Setup range	Default
90h	PRA_KP_GAIN	PID controller proportional gain	0~65535	0
91h	PRA_KI_GAIN	PID controller integral gain	0~65535	0
92h	PRA_KD_GAIN	PID controller derivative gain	0~65535	0
93h	PRA_KVFF_GAIN	Velocity feedforward gain	0~65535	0
9Ah	PRA_KAFF_GAIN	Acceleration feedforward gain	0~65535	0
9Bh	PRA_KP_SHIFT	Proportional gain shift; Enlarge or contract 2^n folds	31 ~ -31	-5
9Ch	PRA_KI_SHIFT	Integral gain shift; Enlarge or contract 2^n folds	31 ~ -31	-15
9Dh	PRA_KD_SHIFT	Derivative gain shift; Enlarge or contract 2^n folds	31 ~ -31	0
9Eh	PRA_KVFF_SHIFT	Velocity feedforward gain shift; Enlarge or contract 2^n folds	31 ~ -31	0
9Fh	PRA_KAFF_SHIFT	Acceleration feedforward gain shit; Enlarge or contract 2^n folds	31 ~ -31	0
A0h	PRA_PID_SHIFT	PID control shift; Enlarge or contract 2^n folds	31 ~ -31	-5
120h	PRA_SERVO_V_BIAS	Output voltage offset (UOM: Volt)	0~10	0
123h	PRA_SERVO_V_LIMIT	Output voltage limit (UOM: volt)	0~10	10
125h	PRA_SERVO_V_INVERSE	Output voltage inverse ; 1 : Yes 0 : No	0~1	0
12Bh	PRA_INTEGRAL_LIMIT	Integral limits	0~2147483647	2147483647

4.2.2 Auto Servo Tuning

The auto tuning function is aimed at designing stable and of good performance PID controller in fast pace to provide for common users. For advanced users, extra Manual tuning can be applied by referencing to the auto tuning results to come out controllers more specifically meeting special needs.

Please set up proper position encoder and aligned output command and move direction especially for control mode in closed-loop control of analog output. Improper setup may lead to servo motor bursting.

Fine tuning steps:

Step 1: Set up offset limit values

This is a safety mechanism where there will be error message displayed in status bar at bottom of page and the fine tuning process stops and the motor turns off in case the deviation amount of position command against position feedback exceed given limit. In general, when running the fine tuning process for the first time the Deviation limit value should be set at low level in the beginning process.

Step 2: Select axis number

Select the axis to be fine tuned in selection menu. You can fine tune one axis at a time.

Step 3: Set up vibration amplitude

The fine tuning process calculate gain value of proportional control, integral control, and derivative control through the back and forth motion of motor. The settings of Amplitude determines the final result. When running for the first time, set up vibration amplitude at lower level at the beginning of fine tuning process. Please note that too small a value of vibration amplitude may fail the calculation of valid control gain.

Step 4: Advanced setup

Steps described above are basic setting. Most common users may skip to Step 5 directly.

Certain advanced setting are provided to improve success rate including: Data length, Hysteresis and sampling time.

(1) Data length :You may set up data length to determine number of PID controller's gain value to be calculated by the program. Gain values come out of PID controllers differ from each other as a result of varied measurements and errors cause by other factors despite they are of the same system and algorithm. A quantified index is provided to validate the calculation results. Here the average of PID controller gain values is displayed in the final value window of MCP2 while its standard deviation is shown in MCP2's Fluctuation window as the criteria for determining successful calculation. Take example. If data length is set at 200, then the program will calculate average and standard deviation (or change rate) of 200 PID gain values. The calculation is set to be completed with relative correctness in case the standard deviation (or change rate) is smaller than 10%.

(2) Hysteresis :This setup is aimed at reducing the standard deviation described earlier to unify outcome from each calculation as consistent as possible. Take example. As shown in table below the value of standard deviation declines with increasing Hysteresis range. On the contrary, a PID controller with larger Hysteresis ranges may lead to smaller system band and slower system response as shown in table below. In either case it is ensured that the controller can keep the system in specific performance range as shown in table below where phase margin is equal to 60 degree.

Hysteresis	KP gains Standard deviation (%)	KI gains Standard deviation (%)	KD gains Standard deviation (%)	Band width (Hz)	Phase margin (degree)
150	5.123	7.887	2.378	22	60
250	4.087	6.383	1.845	18.8	60
350	2.718	4.313	1.183	17.7	60
450	3.056	4.880	1.277	16.5	60

(3) Sampling time : Set up sampling time of the encoder and may affect the calculation outcome. Results of different sampling time are given in table below where system bands are roughly the same while the standard deviations are reduced and Phase margin of 60 degrees remains complaint with system specification.

Sampling time	KP gains standard deviation (%)	KI gains Standard deviation (%)	KD gains Standard deviation (%)	Band width (Hz)	Phase margin (degree)
500	4.918	7.634	2.298	22	60
300	3.562	5.580	1.574	23.8	60
200	2.552	3.970	1.154	23.8	60
100	1.087	1.707	0.478	23.8	60

In case the auto regulation failed we may try to increase Hysteresis range for better success rate at the expense of overall system band. System band and phase margin can be measured with the Bode plot provided by this control card. See later chapters for detail.

Step 5: Start up the auto fine tuning procedure

Press the Start tuning bottom to begin the auto fine tuning procedure and the motor starts vibrating. You may press the Tuning stop bottom to interrupt the process in case any problem is encountered in the tuning process.

Step 6: end of the auto fin tuning procedure

Once the auto fin tuning procedure is completed, you may find the final values and their corresponding fluctuations of proportion, integral and differential gains calculated by the said procedure in the Result field. Ideally the smaller the fluctuation is the better the final convergence will be. You may find out the best combination of vibration amplitude and fluctuation by trial and error. Press the Set final bottom to set PID control gain to the control card. Please note the feedforward control gain and filter stop working after you do this.

Step 7: PID controller performance test

To be on the safer side, please adjust PRA_ERR_POS_LEVEL to lower level before testing the results derived from auto fine tuning to prevent damages to the machine caused by position error.

- **Why the auto fine tuning process stops**

  a The process is stopped as the Tuning stop bottom is pressed by user.

  b The limit signal (PEL/MEL), warning (ALM) or emergency stop signal (EMG) is triggered.

  c The position deviation amount exceed offset limit settings. You may avoid this by set the limit to greater value.

  d The program stops automatically in case the fluctuation value (standard deviation) is greater than 10%. You may increase the Hysteresis range for better success rate.

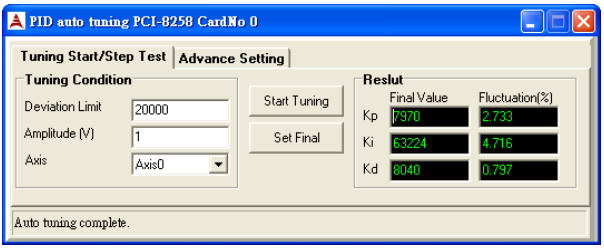


Figure 4-6: The auto fine tuning setup page in MCP2

Table 4-3: PCI-8254/8 Auto-Tuning setup

Setup	Descriptions	Setup range	Default
Offset limit Deviation Limit	The maximum difference between position command and feedback. The program stops when this value is exceeded and ignored it if it is set to zero. (UOM: pulse)	>=0	2000
Amplitude	Set up vibration amplitude of output signal (UOM: Volt)	0~10	1
Axis	Designate axis for running auto fine tuning procedure	0~7	0
Data length Data Length	Calculate required hysteresis range	>0	200
Hysteresis	Hysteresis range (UOM: pulse)	>0	50
Sampling time (Cycle Time)	Encoder sampling time (UOM: micro-second)	100~1000	500

4.2.3 Manual Servo Tuning

Manual fine tuning is still necessary for satisfying different needs. You may change three controller parameters manually: Proportional gain, KP, Integral gain, KI and Derivative gain, KD. You may change velocity or acceleration feedforward gain as required. The manual fine tuning procedure can be executed in the PID setup page of signal sampling function in MotionCreatorPro 2. The manual fine tuning procedure can be executed in steps described below:

Step 1:

Set up KI to zero and KP and KD with initial values, e.g. set value of KP to 1 and KD to 100. Start motor motion, adjust KD to greater value and observe the error position signal, in case of vibration decrease KD value until vibration situation disappears.

Step 2:

Increase KP value gradually to bring down the position error signal. Similarly vibration situation may appear when KP value is getting too high. If this happens, bring down the KP value until vibration situation disappears. You may encounter the overshoot phenomenon. Increase KD value until overshoot situation disappears then you can increase KP value. In general, for actuator in torque control mode the KP value is one fourth of that of the KD one.

Step 3:

Before adjusting KI value, please note watch carefully the integral limit settings as this will affect the effect of integral control. Increase KI value gradually to being down the ready state error to zero step by step. When there is vibration encountered, decrease KI value until there is no vibration any more.

Step 4:

Eventually you may increase velocity /acceleration feedforward gain step by step to higher up response speed furthermore. Please note that this may lead to overshoot phenomenon. After manual final tuning you may use filters to improve high frequency noise encountered when turning off the motor. For design and use of filter, please refer to Section 4.2.4.

4.2.4 Filter

A filter is designed to pass signal of certain band and attenuate all signals outside of this band. This controller supports two general purpose Biquad filter for each axis as shown in Figure 4-6 while the table below indicates corresponding axis parameters against biquad filter. Mathematical formula for biquad filter is shown below

$$R(k) = (1/\text{PRA\_BIQUAD0\_DIV}) \cdot [\text{PRA\_BIQUAD0\_B0} \cdot X(k)$$
$$+ \text{PRA\_BIQUAD0\_B1} \cdot X(k-1)$$
$$+ \text{PRA\_BIQUAD0\_B2} \cdot X(k-2)$$
$$+ \text{PRA\_BIQUAD0\_A1} \cdot R(k-1)$$
$$+ \text{PRA\_BIQUAD0\_A2} \cdot R(k-2)]$$

$$Y(k) = (1/\text{PRA\_BIQUAD1\_DIV}) \cdot [\text{PRA\_BIQUAD1\_B0} \cdot R(k)$$
$$+ \text{PRA\_BIQUAD1\_B1} \cdot R(k-1)$$
$$+ \text{PRA\_BIQUAD1\_B2} \cdot R(k-2)$$
$$+ \text{PRA\_BIQUAD1\_A1} \cdot Y(k-1)$$
$$+ \text{PRA\_BIQUAD1\_A2} \cdot Y(k-2)]$$

Here $R(k)$ and $X(k)$ is the input and output of biquad filter 0, and $Y(k)$ and $R(k)$ is the input and output of biquad filter 1. Commonly available filters are low pass filter and notch filter and will be reviewed in later sections. You may design filters of combinations of various coefficients to meet requirements of different signal processings.

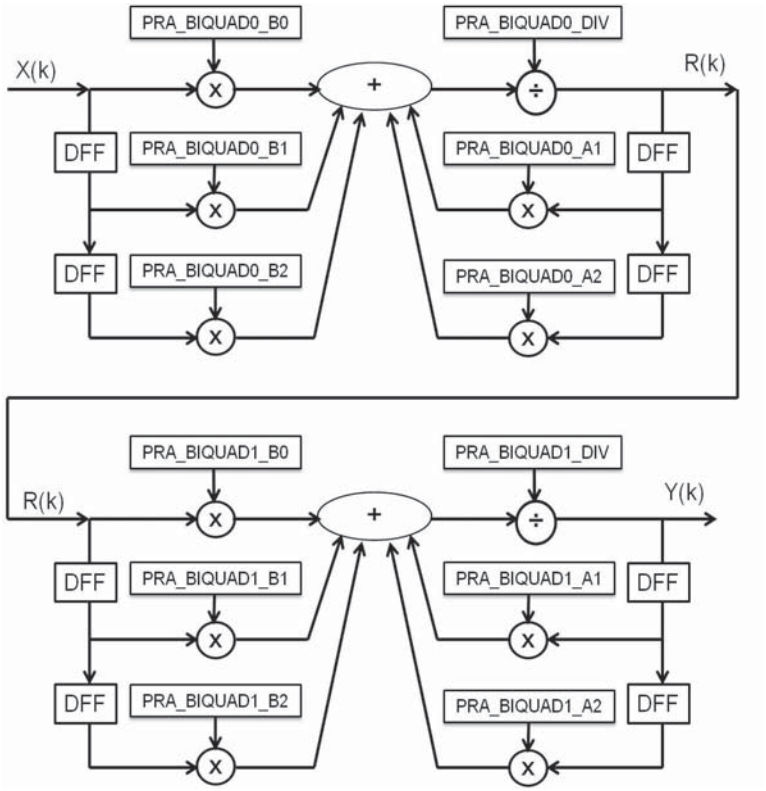


Figure 4-7: Structure of PCI-8254/8 biquad filters in serial connection

Relevant axis parameters can be found in table below:

Param. No.	Define symbol	Description	Value	Default
132h	PRA_BIQUAD0_A1	Biquad filter 0 coefficient	-32768~32767	0
133h	PRA_BIQUAD0_A2	Biquad filter 0 coefficient	-32768~32767	0
134h	PRA_BIQUAD0_B0	Biquad filter 0 coefficient	-32768~32767	1
135h	PRA_BIQUAD0_B1	Biquad filter 0 coefficient	-32768~32767	0
136h	PRA_BIQUAD0_B2	Biquad filter 0 coefficient	-32768~32767	0
137h	PRA_BIQUAD0_DIV	Biquad filter 0 coefficient	-32768~32767	1
138h	PRA_BIQUAD1_A1	Biquad filter 1 coefficient	-32768~32767	0
139h	PRA_BIQUAD1_A2	Biquad filter 1 coefficient	-32768~32767	0
13Ah	PRA_BIQUAD1_B0	Biquad filter 1 coefficient	-32768~32767	1

Param. No.	Define symbol	Description	Value	Default
13Bh	PRA_BIQUAD1_B1	Biquad filter 1 coefficient	-32768~32767	0
13Ch	PRA_BIQUAD1_B2	Biquad filter 1 coefficient	-32768~32767	0
13Dh	PRA_BIQUAD1_DIV	Biquad filter 1 coefficient	-32768~32767	1

4.2.4.1 Low Pass Filter

Increase control gains (KP and KD) is commonly adopted approach in improving response speed and accuracy. However, control gains come with high frequency vibration noise especially when mechanic (motor) stops operations. Low pass filter can be used here to eliminate high frequency vibration noise in case like this.

See Figure below for an ideal low pass filter. Here the Fc is the Cutoff frequency, the frequency range that can be passed is called the Pass band, and the frequency range to be attenuated is the Stop band. Signal frequency lower than the cutoff frequency can pass freely yet signal with frequency greater than cutoff frequency will be attenuated.

Let's explain this effect with one example. Assume a low pass filter with 1000 Hz cutoff frequency, 100 microseconds sampling time, and derived coefficients A1 at 32767, A2 at -11896, B0 at 1794, B1 at 3588, B2 at 1794, and DIV at 28046. See Figure below for simulation results from sine curves at different frequency. This figure indicates that when input signal is a 30Hz sine curve it passes the filter with amplitude and phase remains intact. for input of 1200Hz sine curve the amplitude is attenuated and the phase delayed. The high frequency noise is then illuminated.

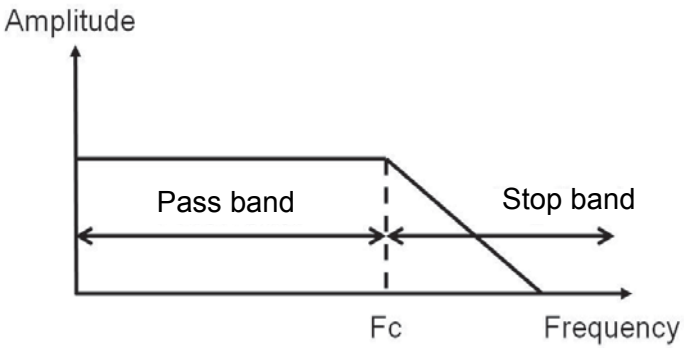
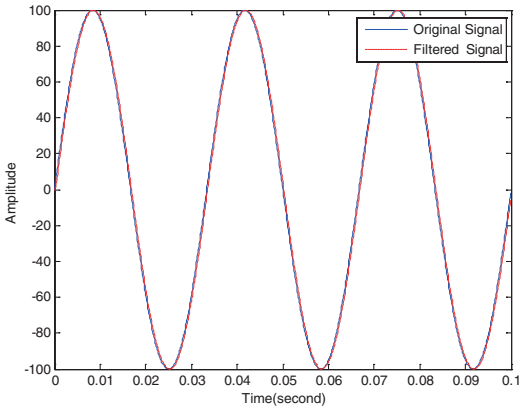
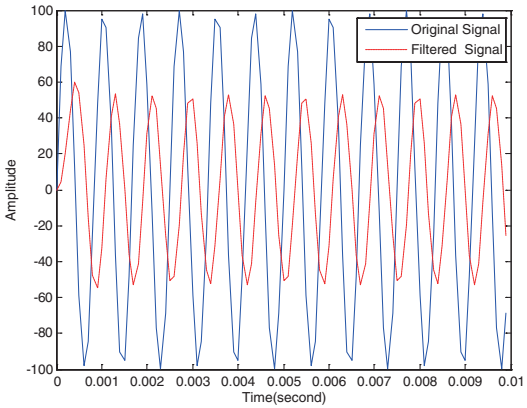


Figure 4-8: Ideal low pass filter



(a) 30Hz sine wave



(b) 1200Hz sine wave

Figure 4-9: Simulation results of low pass filter with 1000Hz cutoff frequency

Adjustment can be made through the MotionCreatorPro 2 function as shown in figure below. Enter the cutoff frequency and the MotionCreatorPro 2 starts calculation filter parameters automatically for you. You can adjust cutoff frequency from high to low and stop the adjustment process once high frequency noise disappears. Usually you can start adjusting from 2000Hz downwards. Note: Too low a cutoff frequency may lead to poor response performance and even unstable situations.

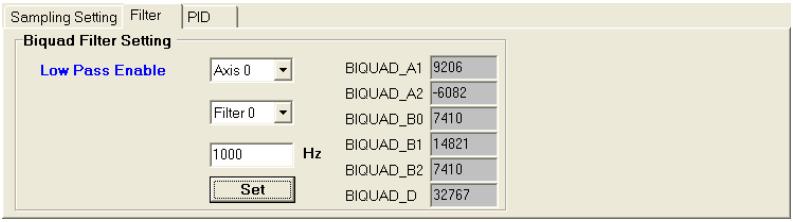


Figure 4-10: MCP2 low pass filter setup page

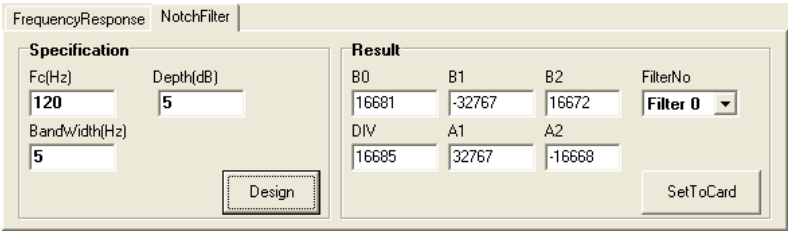


Figure 4-11: MCP2 notch filter setup page

4.2.4.2 Notch Filter

Notch filter is aimed at eliminating specific resonance frequency. The later occurs when frequency of servo response is close to the Natural frequency of the mechanic system. Take example. Resonance is usually found in servo motor mechanism with two or more coupling motors. When running with one servo motor, there shall be no vibration. When both axes servo excite operation at the same time then vibration (resonance) come up.

See Figure below for an ideal notch filter. Where Fc represents the Cutoff frequency, Fl the Low pass cutoff frequency, and Fh the High pass cutoff frequency. The passable frequency band is the Pass band and the attenuated frequency band the Stop band.

Let's explain this effect with one example. Assume a notch filter with cutoff frequency at 100Hz, low pass cutoff frequency at 95Hz, high pass cutoff frequency at 105Hz, sampling time at 100 microseconds, and derived coefficients A1 at 32767, A2 at -16398, B0 at 16415, B1 at -32767, B2 at 16415 and DIV at 16433. See Figure 7 for simulation results from sine curves at different frequency. This figure indicates that when input signal is a 50Hz and 200Hz sine curve it passes the filter with amplitude and phase remains intact. But for input of 100Hz sine curve the amplitude is attenuated and the phase delayed significantly.

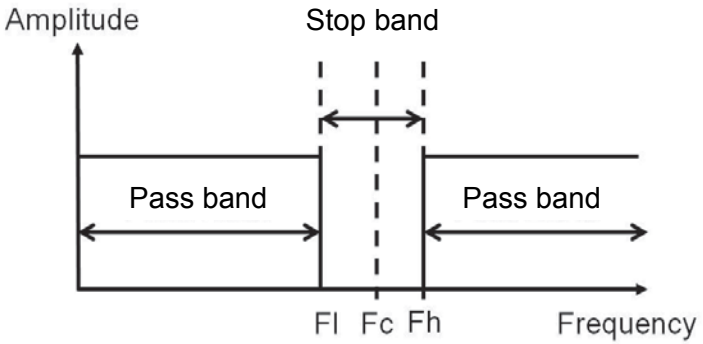
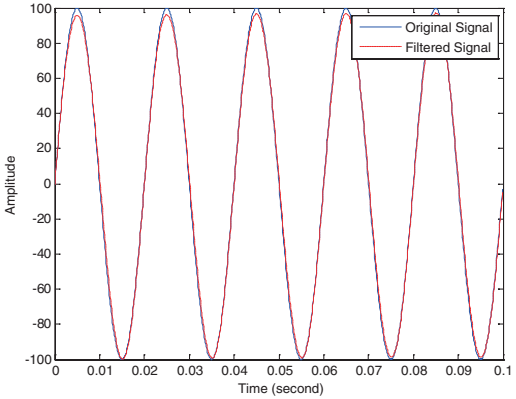
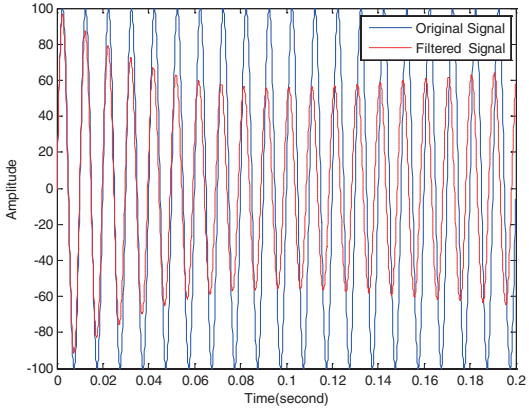


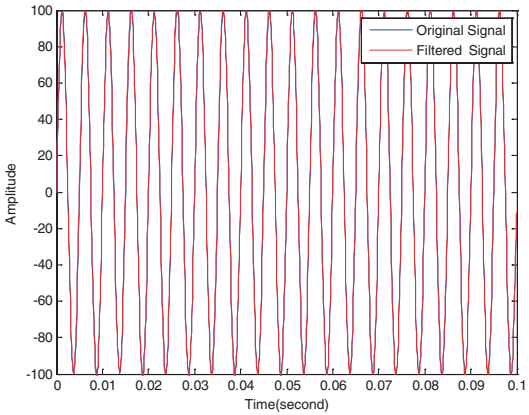
Figure 4-12: An ideal notch filter (a) 50 Hz



(a) 50Hz sine wave



(b) 100Hz sine wave



(c) 200 Hz sine wave

Figure 4-13: Simulation results of notch filter with 100Hz cutoff frequency

See figure below for MCP2's notch filter setup page where you can workout relevant filter coefficients according to your specifications. Note: You can set cutoff frequency to zero to close the notch filter.

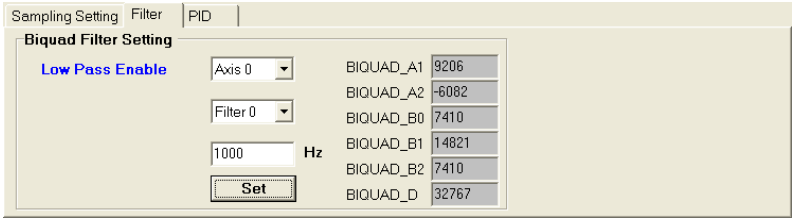


Figure 4-14: MCP2 low pass filter setup page

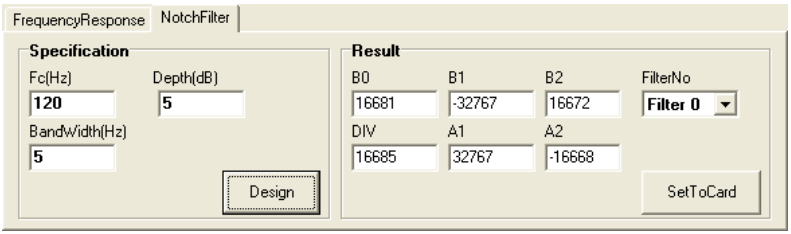


Figure 4-15: MCP2 notch filter setup page

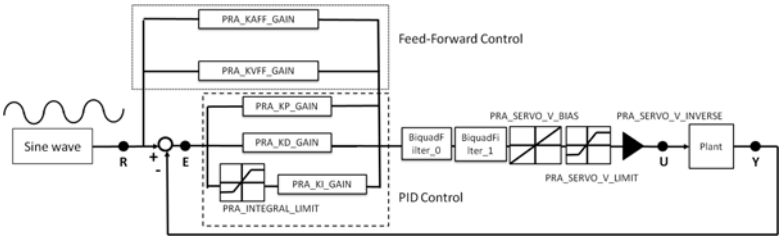
4.2.5 Bode Plot

4.2.5.1 Structure Overview

The Bode Plot tool provided by your PCI-8254/8258 may display transfer function's frequency responses of Linear Time-Invariant (LTI)system. In general the X-axis of a Bode plot indicates frequency in Log scale while its Y-axis Gain and Phase. Expressed in unit of (dB), gaining is the level of enlargement and contraction between input and output signals. Expressed in unit of (Degree), phase is the leading or trailing extent between input and output signals. You may learn about system features including stability and resonance point with the Bode plot's frequency response function. This may help users in determining system's compliance with given requirements and provide advanced users with important system features including system transfer function's order, pole, and zero points. This offers great assistance in system identification and advanced controller design.

The PCI-8254/8258 system's fully-closed circuit feature can estimate frequency response in varieties format safely and more accurately. See figure below for overall structure where continuous sine wave input signals are generated with fixed amplitude and frequency. You may retrieve servo motor's position feedback signal (encoder) to analyze close-loop system's frequency response performance, including PID controller, feedforward control, filter, digital to analog conversion, driver, and servo motor. This is a sine wave with changing amplitudes and phases that you can use to get a group of closed-loop gain and phase at specific frequency. This system's input signal is a motion command and output signal the motor's feedback position. With similar process we can get system frequency response in given frequency range by increasing sine wave frequency. In addition, the frequency response of Open-loop system and Plant is provided. The plant's frequency response covers digital-analog conversion, drive, and servo motor while the close-loop circuit frequency response is the outcome of disabled feedforward control, bypassed filter, and PID controller with plant. The signal description table below describes meanings of signals captured at different position of the structure diagram while the frequency response type table outlines I/O signals under different system frequency response.

Frequency response structure diagram



Signal description table

Name	Symbol	Unit	Description
Sine wave	R	pulse	The input of closed-loop system.
Error position	E	pulse	The input of open-loop system.
Controller output	U	pulse	The input of plant.
Encoder	Y	pulse	The output of closed-loop system, open-loop system, and plant.

Frequency response type table

Type	Transfer function T	Description
Plant	T = Y/U	It includes servo motor system
Open-loop	T = Y/E	It includes PID controller and servo motor system (Assume feed-forward controller is disable and bypass Biquad filters)
Closed-loop	T = Y/R	Total system

4.2.5.2 How to Use

Usage and precautions are illustrated below. See Bode plot relevant parameter table for some of the settings:

CAUTION

Set up your controller by manual or auto tuning before calculating the system frequency response as invalid setup in a fully-closed circuit may lead to jerking servo motor.

a **Set up initial and closing frequency and data points of Bode plot:**
The frequency range of a Bode plot is determined by its initial and closing frequency. As too small a initial value may boost calculation time, please try to start at around 10Hz. The closing frequency should not be greater than 500Hz as it is limited by driver and internal sampling rate. Each Bode plot contains frequency, gain, and phase data. Number of data points determines the resolution of frequency response. Please note that the better the resolution is the more calculation time is required.

b **Set up size of sine wave amplitude:**
The amplitude should be set according to the relation between motor resolution versus actual distance unit of measure as oversized amplitude may lead to incorrect outcome as the calculation error may be too big while undersized one may affect machine safety adversely. Take example. A motor resolution at 10000 pulse/rev may need to set the amplitude to around 300 pulse.

c **Set up update cycle of each data points of the Bode plot**:
Length of update cycle time may affect number of sampling points required for calculating Bode plot's data point. Take example. In case of update cycle at 0.1 second, sampling speed at 10KHz, and frequency range at 10Hz~100Hz the required sampling points for each Bode plot's data point is 1000. This may affect the accuracy in calculating frequency response. In most cases the longer the better.

d **Determine frequency response type**:
The system now supports three types of frequency response: closed circuit system, open circuit system and plant. See earlier descriptions for detail.

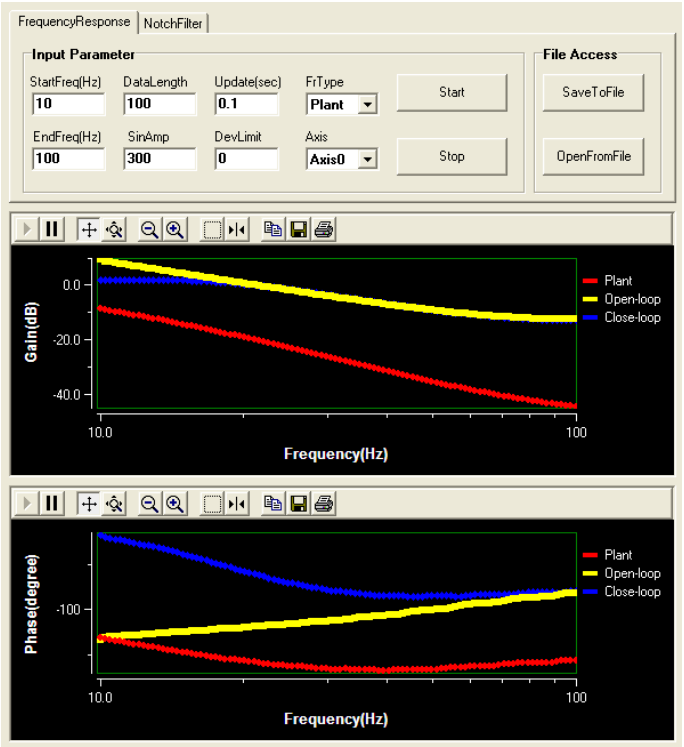
e **Set up protection mechanism:**
Set up a protection mechanism to power off the motor automatically when the error position is exceeding this deviation limit. Set the limit to zero to close this mechanism. In addition, the motor automatically turns off when a warning or emergency signal is encountered.

f **Start calculation and display outcome**:
You can initiate the process once all the settings are made. The frequency response outcome displays in the MCP2 Bode plot page as shown in figure below, where the motor has null load and resolution at 10000 pulse/rev, the sine wave amplitude is set at 300 pulse with 100 data points, each point's update cycle is 0.1 sec and frequency range is 10Hz~100Hz.

Table of the Bode plot's parameters

Name	Unit	Range	Description
Start Frequency	Hz	>0	Frequency of sine wave
End Frequency	Hz	>0	Frequency of sine wave
Sine amplitude	Pulse	>0	Amplitude of sine wave
Data Length	Value	>1	Total data length of Bode plot
Update	Second	>0	Update time of each Bode plot data
Deviation limit	Pulse	>=0; This function will be ignored if it is set to 0.	If error position exceeds deviation limit, the servo will be disable immediately.

MCP2 Bode plot page



4.3 Motion Control Operations

This section describes motion control modes provided by the controller and their operation principle. The objective is to help users make most of the motion control capacity of your controller to accomplish desired applications.

4.3.1 Coordinated System

This controller employs Cartesian coordinate system where one or more axes motion can be executed by one-to-one mapping each axis to a motor. There exists a conversion relation between axis of the Cartesian coordinate system and the motor being controlled. This conversion relation enables users to set up their own coordinate system without restrictions. Figure below indicates a coordinate system relation. The unit conversion factor will be reviewed in next section.

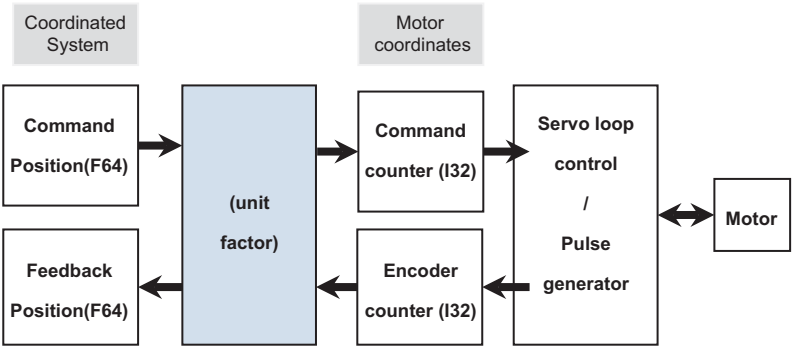


Figure 4-16: Controller coordinates system block

You may read out or set up coordinate command location or actual coordinate location

I32 APS_get_command_f (I32 Axis_ID, F64 *Command);
//command location reading
I32 APS_get_command_f (I32 Axis_ID, F64 *Command);
//command location setup
I32 APS_get_command_f (I32 Axis_ID, F64 *Command);
//actual location reading
I32 APS_get_command_f (I32 Axis_ID, F64 *Command);
//actual location setup

I32 coordinate format compliant API functions the same as API described above

I32 APS_get_command(I32 Axis_ID, I32 *Command);
I32 APS_set_command(I32 Axis_ID, I32 Command);
I32 APS_get_position(I32 Axis_ID, I32 *Position);
I32 APS_set_position (I32 Axis_ID, I32 Position);

API listed below can read motor coordinates
I32 APS_get_encoder(I32 Axis_ID, I32 *Encoder);
I32 APS_get_command_counter (I32 Axis_ID, I32 *Counter);

NOTE

In close loop control procedure you cannot set up command counter and encoder counter and so relevant setup API are not provided.

4.3.2 Unit Factor

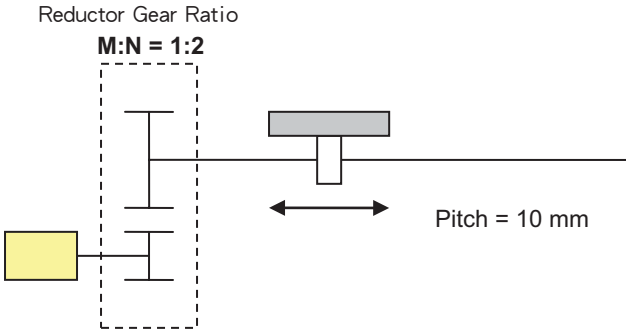
Location unit (or motion mechanism) of motor can have actual mapping against physical distance unit of coordinate system by setting up proper unit factor. The calculation formula is described below

$$\text{Unit Factor} = \frac{\text{Encoder Resolution}}{\text{Pitch (User define unit)}} \times \frac{\text{N (Deductor gea}}{\text{M (Driver gear)}}$$

We use three examples to explain the way how Unit factor is calculated:

Example 1: Ball screw carrier

Assume encoder counts (resolution) generated by one spin of the motor is 10000 and the screw pitch is 10mm and the user desired distance unit of measure is micrometer,

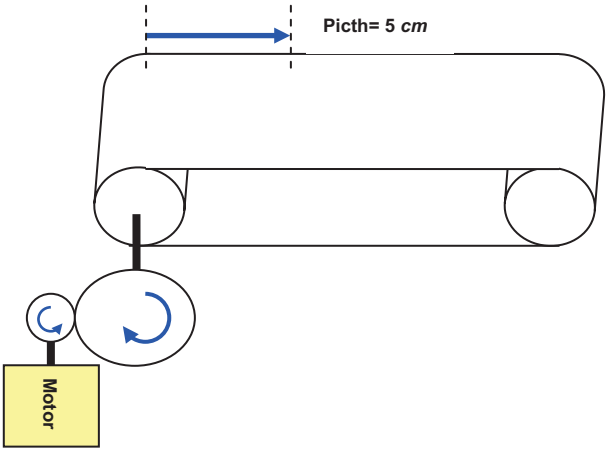


Unit factor can be calculated as described below:

$$Unit\ factor = \frac{10000}{10\text{mm} \times 1000\ \mu m} \times \frac{2}{1} = 2$$

Example 2: Conveyor system

Assume number of pulses generated by one spin of the motor is 8192, the conveyor belt shift 5cm by one spin of the belt pulley, the gear ratio is 1:2, and the user desired distance unit of measure is Millimeter then the Unit factor can be calculated as:



Unit factor can be calculated as described below:

$$Unit\ factor = \frac{8192}{5cm \times 100\ mm} \times \frac{2}{1} = 32.768$$

Example 3: Linear Motor system with Linear Scale

Take optical resolution at 1 micrometer and distance at millimeter then the unit factor should be:

$$Unit\ factor = \frac{1}{1\mu\text{m} \times 1000\ mm} = 0.001$$

Unit factor can be set up in axis parameter:

Param. No.	Define symbol	Description	Value	Default
86h (134)		Unit factor	F64 value	1

In general, you should define unit of measure at first and set up other position relevant parameter before designing any motion control application.


NOTE

If unit factor setting are changed during operation, other parameters related with distance unit of measure (e.g. position, velocity, and acceleration unit of measure) will be affected and you are required to change and adjust relevant setting on your own.

Relevant axis parameters can be found in table below:

Param. No.	Define symbol
07h (7)	PRA_SD_DEC
0Ah (10)	PRA_SPEL_POS0
0Bh (11)	PRA_SMEL_POS1
13h (19)	PRA_HOME_ACC
15h (21)	PRA_HOME_VM
17h (26)	PRA_HOME_SHIFT
19h (25)	PRA_HOME_VO
1Bh (27)	PRA_HOME_POS
21h (33)	PRA_ACC
22h (34)	PRA_DEC
23h (35)	PRA_VS
24h (36)	PRA_VM
25h (37)	PRA_VE
2Ah (42)	PRA_PRE_EVENT_DIST
2Bh (43)	PRA_POST_EVENT_DIST
43h (67)	PRA_JG_ACC
44h (68)	PRA_JG_DEC
45h (69)	PRA_JG_VM
46h (70)	PRA_JG_OFFSET


NOTE

Some API may have encoder signal of position related input parameter.

4.3.3 Acc/Deceleration Profile

Basic motion command usually contains: 1. distance; 2. velocity; and 3. acceleration data. This controller plans and calculates Acceleration & deceleration profile based on these motion command parameters to make motion operation completed as desired by users. This controller provides following acceleration profiles

1. Trapezoidal speed profile, T-curve
2. S-curve

4.3.3.1 Trapezoidal Speed Profile, T-curve

Trapezoidal speed profile (the so called T-curve) is a curve where the acceleration zone and deceleration zone matches first-order linear speed profile (equivalent acceleration). As shown in the velocity-time chart (V-T):

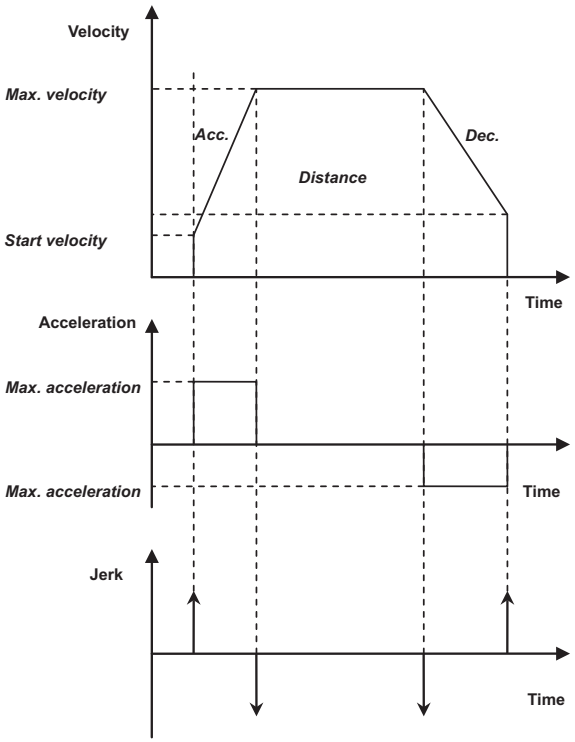


Figure 4-17: Relation of trapezoidal speed profile's speed/acceleration/jerk VS time

In a V-T chart the area under the trapezoidal curve equals motion distance. If the user does not set up sufficient motion distance, the controller shall increase (decrease) the maximum speed while maintain the acceleration, as shown in figure below:

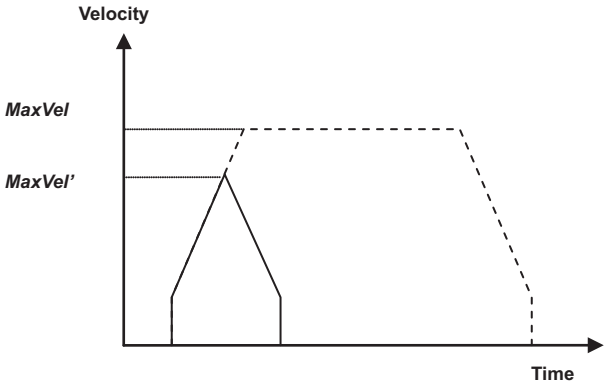


Figure 4-18: Maximum speed by auto-planning

MaxVel is the maximum velocity set up by user, dotted line indicate speed profile with sufficient distance. As the movement distance is insufficient, the controller adjust the maximum velocity to *MaxVel'* automatically. The acceleration and deceleration rate remain intact to maintain the best (shortest) motion time.

4.3.3.2 S-curve

An S-curve is a curve where the speed profile in the jerk area can be represented by second-order profile. This helps to reduce motor vibration at start up and stop time as indicated by points (t1, t3, t5, t7) in figure below.

To shorten acceleration and deceleration time the linear section (t2, t6) is inserted in these area to maintain the maximum acceleration and so get an acceleration-time (A-T) chart in Trapezoidal.

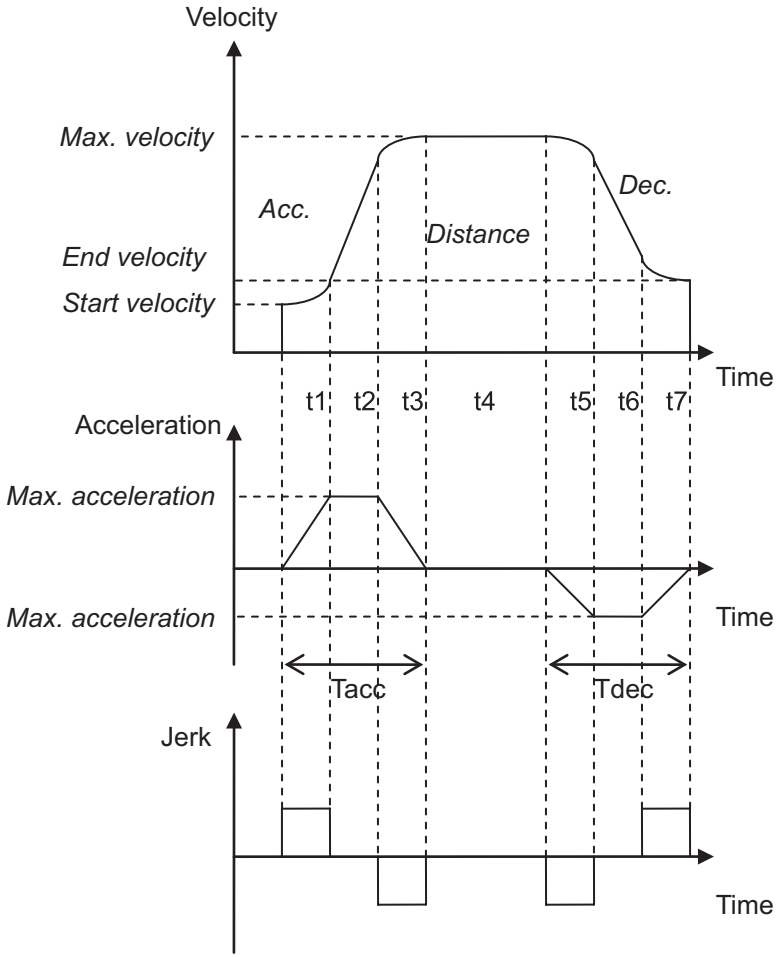


Figure 4-19: S-curve's velocity, acceleration, and jerk versus time

This controller employs S-factor (S) to control jerk ratio. Its equation is described below

$$S = \frac{t1}{t1+t2}; and \;\; t1 = t2$$

Value of S is between 0 and 1, when

S = 0, the speed profile becomes a T-curve

S >0 and S<=1: S - curve

When S = 1, the profile comes to a Pure S– curve with its A-T chart become a triangle.

The equation above indicate that the greater the value of S is the more smooth the speed profile and the smaller jerk value will become. This helps in reducing motor vibration. However, the motion process takes more time to complete. On the contrary, the smaller the S value is the greater the jerk value become and the motion time reduces to the shortest.

As with a T-curve, when motion distance is insufficient, the controller adjust the maximum velocity automatically to maintain smooth movement. Acceleration ACC and DEC and S-factor remain consistent to maintain acceleration rate and the jerk rate will be changed, as shown in figure below

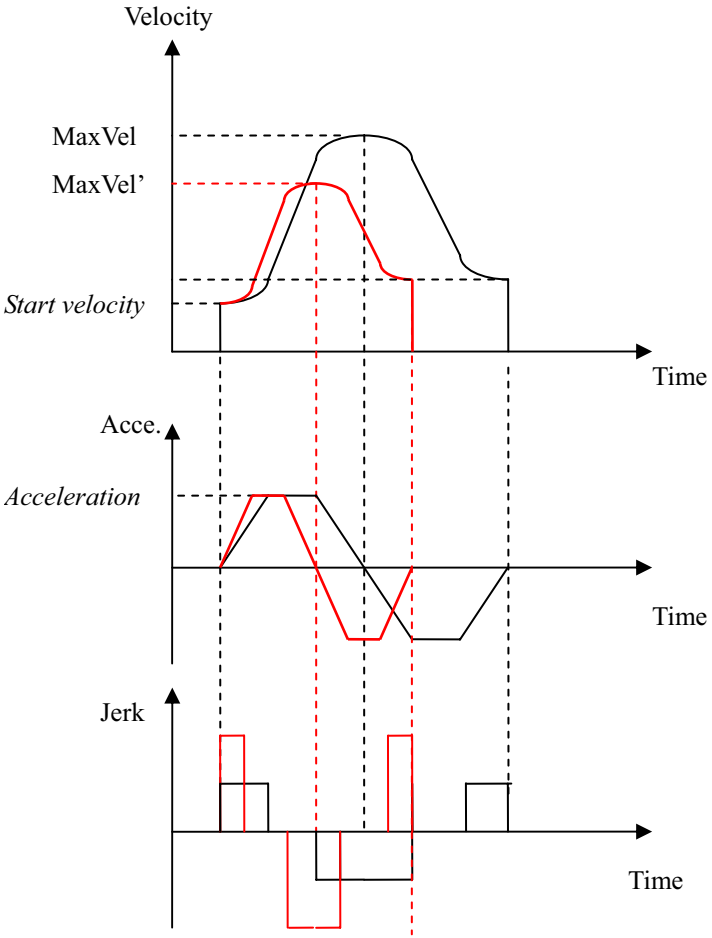


Figure 4-20: Maximum speed by auto-velocity

Acceleration profile and its rule described above applies with single axis point-to-point movement (PTP), velocity movement, home movement, and interpolation among multiple axis.

- **Relevant axis parameters**

Param. No.	Define symbol	Description
12h (18)	PRA_HOME_CURVE	Home move S-factor
20h (32)	PRA_SF	Move S-factor
42h (66)	PRA_JG_SF	Jog S-factor

NOTE

You may set up S-factor directly in some API, please refer to Function library manual for detail.

4.4 Home Move

After power on and before executing any motion control, a motion control system executes home movement to set up the zero position of the coordinate system.

Commonly available stepper motor, servo drive or linear motor mechanism accompanied by optical scale employs incremental type encoder which requires some mechanical signal to set up the original position during home operation. These mechanical signals are ORG, EZ, PEL, and MEL. Some servo drives are featured with absolute type encoder, e.g. J3-B type (SSCNET 3) of Mitsubishi, that require at least one home movement after system initialization. No more home operation is required after the absolute coordinate system is established.

After the homing command is received, the controller starts searching for original position / zero position with the help of some external signals. After the home movement is completed the controller axis stops at the original postion while command position and feedback position reset to zero. All the home movement related operations are executed by the controller automatically. No user interaction is required. Just wait it to complete automatically.

You can finish home movement operation by steps described below:

- Set up home mode and relevant parameters
- Start up home move
- Wait for home move to complete. (You may check the status by polling or interrupt.)
- If home move does not complete successfully you may troubleshooting by steps described below

Relevant APS API described below:

I32 APS_motion_status (I32 Axis_ID);

Relevant axis parameters:

Param. No.	Define symbol	Description
10h (16)	PRA_HOME_MODE	Home mode settings
11h (17)	PRA_HOME_DIR	Homing direction settings
12h (18)	PRA_HOME_CURVE	Home move S-factor
13h (19)	PRA_HOME_ACC	Homing acceleration/deceleration settings
15h (21)	PRA_HOME_VM	Homing maximum velocity

Param. No.	Define symbol	Description
17h (23)	PRA_HOME_SHIFT	Home position and shift distance of positioning signal
18h (24)	PRA_HOME_EZA	EZ alignment enable
19h (25)	PRA_HOME_VO	Homing velocity away from ORG signal
1Bh (27)	PRA_HOME_POS	Position command setup after homing completion

- **Example:**

```
#include "APS168.h"
#include "APS_define.h"
#include "ErrorCodeDef.h"

void home_move_example()
{
  //This example shows how home move operates
  I32 axis_id = 0;
  I32 return_code;
  I32 msts;

  // 1. Select home mode and config home parameters
  APS_set_axis_param( axis_id, PRA_HOME_MODE,     0 ); //Set home mode
  APS_set_axis_param( axis_id, PRA_HOME_DIR,      1 ); //Set home direction
  APS_set_axis_param( axis_id, PRA_HOME_CURVE,    0 );    // Set acceleration pattern (T-curve)
  APS_set_axis_param( axis_id, PRA_HOME_ACC, 1000000 ); // Set homing acceleration rate
  APS_set_axis_param( axis_id, PRA_HOME_VM,   100000 );  // Set homing maximum velocity.
  APS_set_axis_param( axis_id, PRA_HOME_VO,    50000 );   // Set homing
  APS_set_axis_param( axis_id, PRA_HOME_EZA,      0 );   // Set homing
  APS_set_axis_param( axis_id, PRA_HOME_SHIFT,    0 );   // Set homing
  APS_set_axis_param( axis_id, PRA_HOME_POS,      0 );   // Set homing
```

```
// 2. Start home move
return_code = APS_home_move( axis_id ); //Start homing
if( return_code != ERR_NoError )
{ /* Error handling */ }

// 3. Wait for home move done,
do{
  Sleep( 100 );
msts = APS_motion_status( axis_id );// Get motion status
msts = ( msts >> MTS_NSTP ) & 1;    // Get motion done bit
}while( msts == 1 );

// 4. Check home move success or not
msts = APS_motion_status( axis_id ); // Get motion status
msts = ( msts >> MTS_ASTP ) & 1;     // Get abnormal stop bit
if( msts == 1 )
{ // Error handling ...
I32 stop_code;
APS_get_stop_code( axis_id, &stop_code );

}else
{ // Homing success.
}
}
```

This controller provides multiple auto-home searching process for different hardware platform which may refer to three mechanical signals: ORG, EL, and EZ. You may define three homing mode with these reference signal. User may design required homing process by any combination of these three signals. Each mode can have multiple parameters to meet various positioning requirements. These three homing mode are designed in accordance with relevant system configurations and have included commonly available hardware configurations. These three modes are:

1. **ORG signal homing (Home mode = 0) (Home return by ORG signal)**
2. **EL signal homing (Home mode = 1)**
3. **Single EZ signal homing (Home mode = 2)**

Homing process and relevant parameters of these three modes are described below.

4.4.1 OGR Signal Homing - Home Mode = 0

There are three cases for this mode according to its initial position:

Condition A: The initial position is located between MEL and ORG signals or at MEL signal.

Condition B: Ths initial position is located at ORG signal

Condition C: The initial position is located between PEL and ORG signals or at PEL signal.

Table and figure below represent homing steps of these three situations with their speed and position. The three gray area in figure below represent the ON region of MEL, ORG, and PEL from left to right respectively. Move forward at the highest speed VM to search for ORG, decelerate to fully stop when ORG signal is detected. Then move away from ORG signal in VM speed. Search ORG again with low speed VO to complete the Home procedure.

- **Relevant axis parameters setup**

Axis parameters	Axis parameter values	Description to axis parameter value
PRA_HOME_MODE	0	Employing home mode 0 (homing by ORG signal)
PRA_HOME_DIR	0	Homing by moving forward in positive direction
PRA_HOME_EZA	0	Further align with signal EZ, 0: No, 1: Yes
PRA_HOME_S	0	S-curve factor
PRA_HOME_ACC	ACC	Acceleration and deceleration in unit of (distance unit of measure/sec.^{2})
PRA_HOME_VS	VS	Initial speed in unit of (distance unit of measure/sec.)
PRA_HOME_VM	VM	Speed of original position searching in unit of (distance unit of measure/sec.)
PRA_HOME_VO	VO	Homing speed in unit of (distance unit of measure/sec.)
PRA_HOME_SHIFT	0	Shift amount of final homing position against alignment signal (distance unit of measure / pulse)

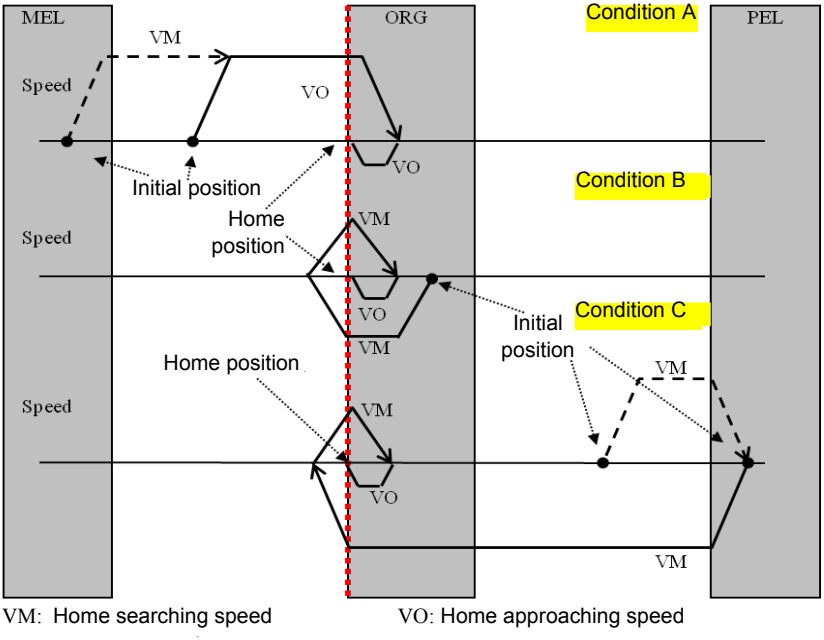


Figure 4-21: Home mode 0 (Case: ORG)

ORG signal of most mechanical device has two directional edges (the two ends of signal fender). Figure above indicates that when the homing direction parameter in axis parameters is set to positive direction (PRA_HOME_DIR), the control axis starts searching from positive direction (the ascending direction of position command). And stops at the left edge of ORG signal (close to MEL mechanical signal).

On the contrary, if the homing direction parameter in axis parameters is set to negative direction (PRA_HOME_DIR), the control axis starts searching from negative direction (the descending direction of position command). And stops at the right edge of ORG signal (close to PEL mechanical signal). Figure below indicates home movement when ”PRA_DIR” is set to negative direction

- **Relevant axis parameters setup**

Axis parameters	Axis parameter values	Description to axis parameter value
PRA_HOME_MODE	**0**	Employing home mode 0 (homing by ORG signal)
PRA_HOME_DIR	**1**	By negative direction forward homing
PRA_HOME_EZA	**0**	Further align with signal EZ, 0: No, 1: Yes
PRA_HOME_S	0	S-curve factor
PRA_HOME_ACC	ACC	Acceleration and deceleration in unit of (distance unit of measure/sec.^{2})
PRA_HOME_VS	VS	Initial speed in unit of (distance unit of measure/sec.)
PRA_HOME_VM	VM	Speed of original position searching in unit of (distance unit of measure/sec.)
PRA_HOME_VO	VO	Homing speed in unit of (distance unit of measure/sec.)
PRA_HOME_SHIFT	0	Shift amount of homing position (distance unit of measure / pulse)

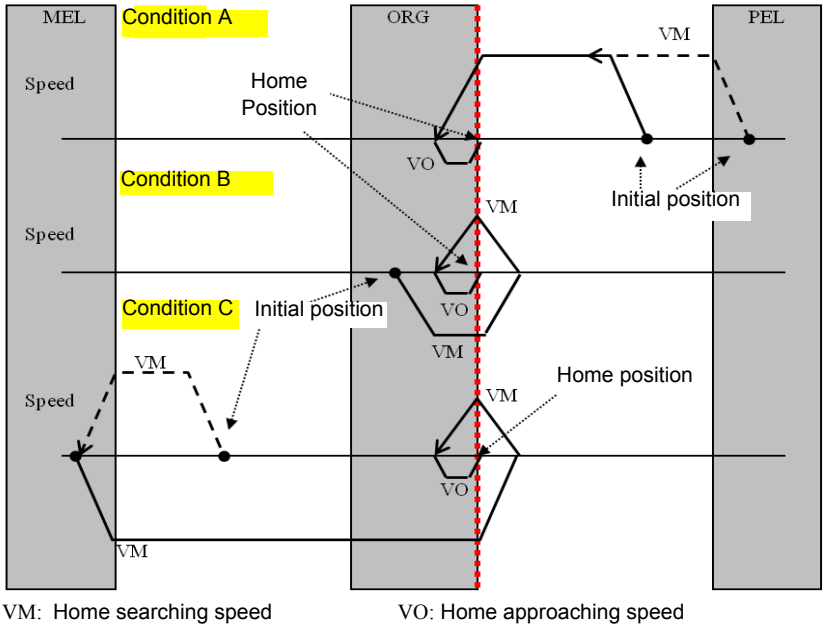


Figure 4-22: Home mode 0 (Case: ORG)

When axis parameter PRA_HOME_EZA is set to 1 it means to align with EZ, move forward to homing direction, until the first EZ is detected and place control axis at the edge of EZ, then the home movement is completed.

The motions are described below:

- **Relevant axis parameters setup**

Axis parameters	Axis parameter values	Description to axis parameter value
PRA_HOME_MODE	0	Home mode 0
PRA_HOME_DIR	0	Homing by moving forward in positive direction
PRA_HOME_EZA	1	Further align with signal EZ, 0: No, 1: Yes
PRA_HOME_S	0	S-curve factor
PRA_HOME_ACC	ACC	Acceleration and deceleration in unit of (distance unit of measure/sec.^{2})
PRA_HOME_VS	VS	Initial speed in unit of (distance unit of measure/sec.)

Axis parameters	Axis parameter values	Description to axis parameter value
PRA_HOME_VM	VM	Speed of original position searching in unit of (distance unit of measure/sec.)
PRA_HOME_VO	VO	Homing speed in unit of (distance unit of measure/sec.)
PRA_HOME_SHIFT	0	Shift amount of homing position (distance unit of measure)

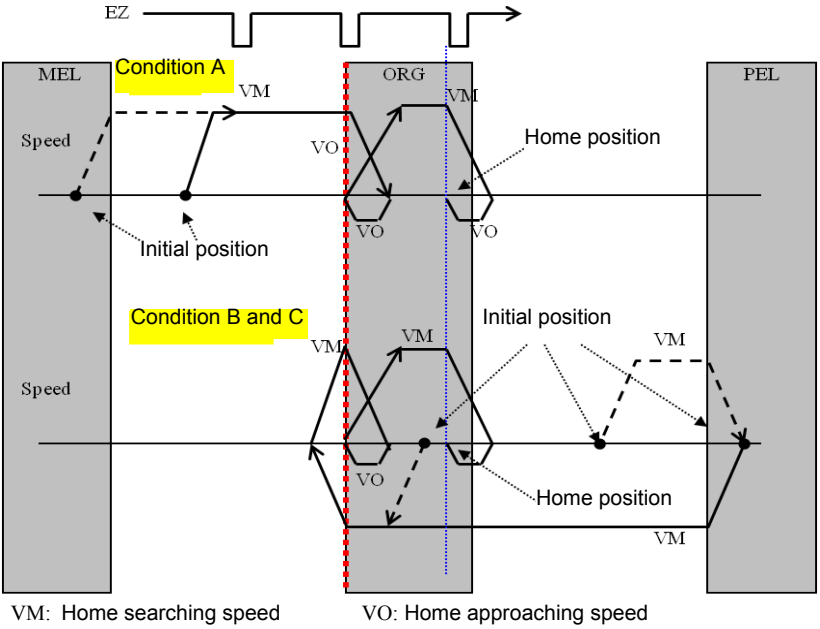


Figure 4-23: Home mode 0 (Case: ORG+EZ)

Figure below indicates a negative direction example:

- **Relevant axis parameters setup**

Axis parameters	Axis parameter values	Description to axis parameter value
PRA_HOME_MODE	**0**	Employing home mode 0 (homing by ORG signal)
PRA_HOME_DIR	**1**	By negative direction forward homing
PRA_HOME_EZA	**1**	Further align with signal EZ, 0: No, 1: Yes
PRA_HOME_S	0	S-curve factor

Axis parameters	Axis parameter values	Description to axis parameter value
PRA_HOME_ACC	ACC	Acceleration and deceleration in unit of (distance unit of measure/sec.^{2})
PRA_HOME_VS	VS	Initial speed in unit of (distance unit of measure/sec.)
PRA_HOME_VM	VM	Speed of original position searching in unit of (distance unit of measure/sec.)
PRA_HOME_VO	VO	Homing speed in unit of (distance unit of measure/sec.)
PRA_HOME_SHIFT	0	Shift amount of homing position (distance unit of measure)

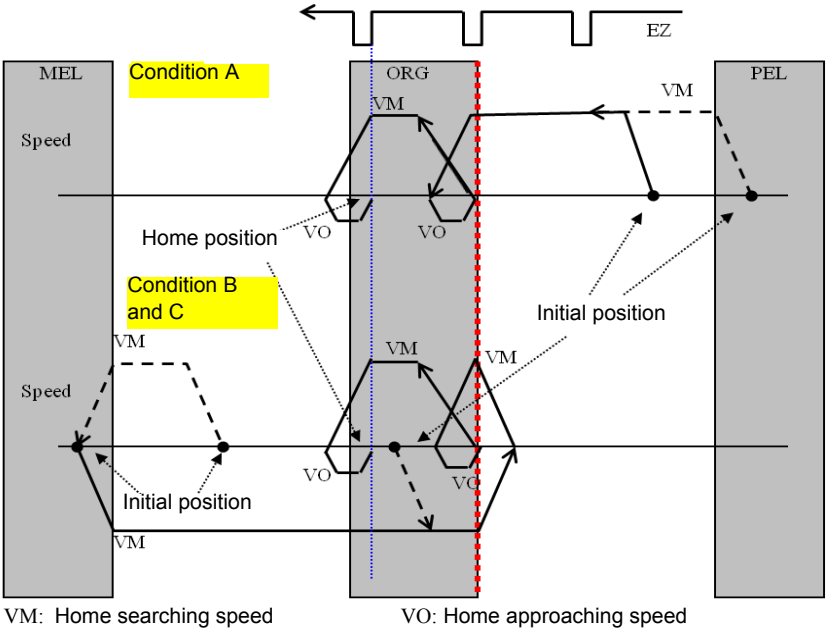
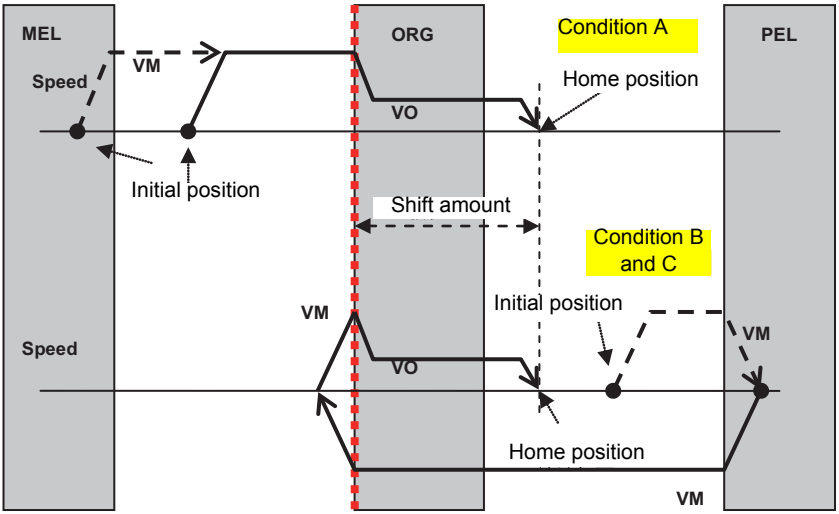


Figure 4-24: Home mode 0 adverse (Case: ORG+EZ)

You may set up homing position shift amount to fine tune the final position. Figure and table below indicate an example of setup and motion diagram.

- **Relevant axis parameters setup**

Axis parameters	Axis parameter values	Description to axis parameter value
PRA_HOME_MODE	**0**	Employing home mode 0 (homing by ORG signal)
PRA_HOME_DIR	**0**	Employing positive direction forward homing
PRA_HOME_EZA	**0**	Further align with signal EZ, 0: No, 1: Yes
PRA_HOME_S	0	S-curve factor
PRA_HOME_ACC	ACC	Acceleration and deceleration in unit of (distance unit of measure/sec.^{2})
PRA_HOME_VS	VS	Initial speed in unit of (distance unit of measure/sec.)
PRA_HOME_VM	VM	Speed of original position searching in unit of (distance unit of measure/sec.)
PRA_HOME_VO	VO	Homing speed in unit of (distance unit of measure/sec.)
PRA_HOME_SHIFT	**Shift amount**	Shift amount of homing position (distance unit of measure)



VM: Home searching speed VO: Home approaching speed

Figure 4-25: Home mode 0 decelerate to stop (Case: ORG)

4.4.2 EL Signal Homing - Home Mode 1

This is a home movement based on PEL or MEL mechanical signal. After the homing command is received, the control axis searches PEL or MEL signal position and stops at edge of the signal. You may set up to align with EZ signal and to set up shift amount.

Figure 1 below illustrates how to set up Home mode 1 (EZ signal) with postive direction homing and without EZ alignment. After home movement is completed the control axis stops at the edge of PEL signal.

- **Relevant axis parameters setup**

Axis parameters	Axis parameter values	Description to axis parameter value
PRA_HOME_MODE	**1**	Employing home mode 1 (homing by EL signal)
PRA_HOME_DIR	**0**	Employing positive direction forward homing
PRA_HOME_EZA	**0**	Further align with signal EZ, 0: No, 1: Yes
PRA_HOME_S	0	S-curve factor
PRA_HOME_ACC	ACC	Acceleration and deceleration in unit of (distance unit of measure/sec.^{2})
PRA_HOME_VS	VS	Initial speed in unit of (distance unit of measure/sec.)
PRA_HOME_VM	VM	Speed of original position searching in unit of (distance unit of measure/sec.)
PRA_HOME_VO	VO	Homing speed in unit of (distance unit of measure/sec.)
PRA_HOME_SHIFT	0	Shift amount of homing position (distance unit of measure)

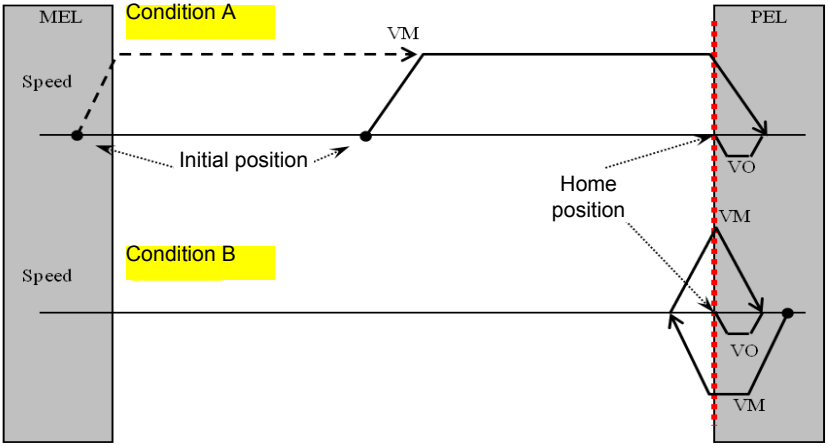


Figure 4-26: Home mode 1 (Case: EL)

EL homing mode: Positive direction home movement with control axis stops at edge of PEL signal

For EL signal homing mode with negative direction homing and without EZ alignment, the control axis stops at MEL signal edge after home movement is completed as shown in figure below.

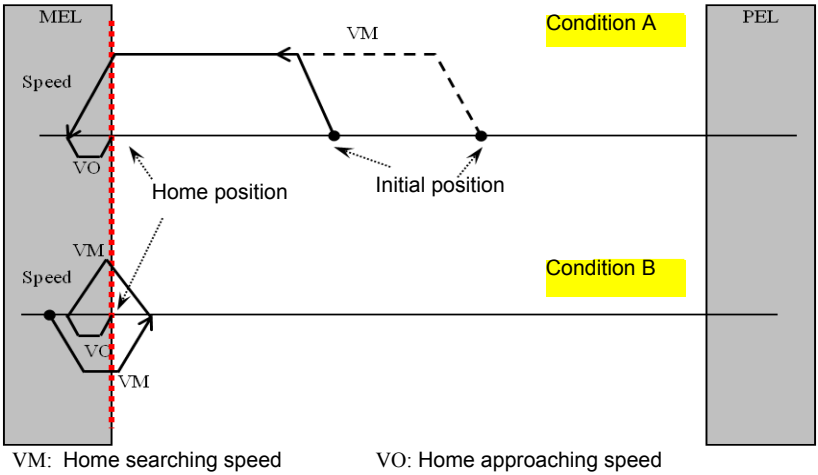


Figure below illustrates how to set up Home mode 1 (EZ signal) with positive direction homing and EZ alignment. After home movement is completed the control axis stops at the edge of EZ signal.

- **Relevant axis parameters setup**

Axis parameters	Axis parameter values	Description to axis parameter value
PRA_HOME_MODE	**1**	Employing home mode 1 (EZ signal) homing
PRA_HOME_DIR	**0**	Employing positive direction forward homing
PRA_HOME_EZA	**1**	Further align with signal EZ, 0: No, 1: Yes
PRA_HOME_S	0	S-curve factor
PRA_HOME_ACC	ACC	Acceleration and deceleration in unit of (distance unit of measure/sec.^{2})
PRA_HOME_VS	VS	Initial speed in unit of (distance unit of measure/sec.)
PRA_HOME_VM	VM	Speed of original position searching in unit of (distance unit of measure/sec.)
PRA_HOME_VO	VO	Homing speed in unit of (distance unit of measure/sec.)
PRA_HOME_SHIFT	0	Shift amount of homing position (distance unit of measure)

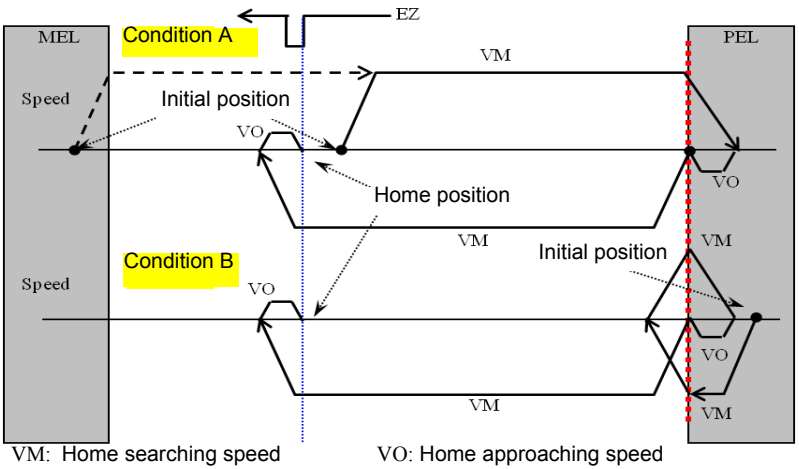
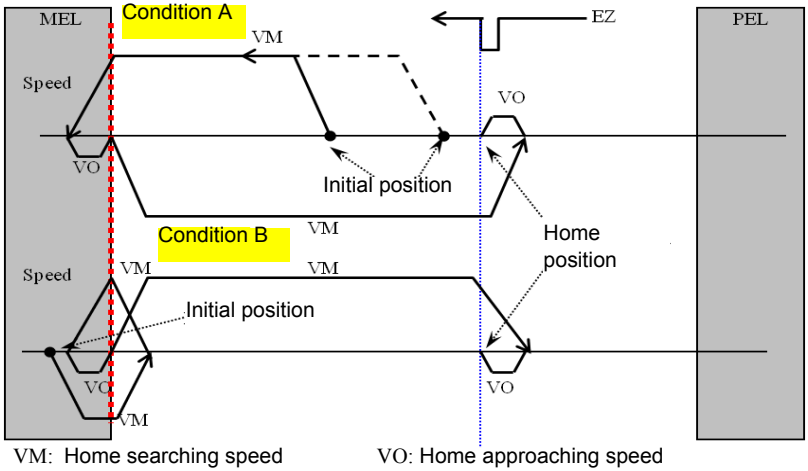


Figure 4-27: Home mode 1 (Case: EL+EZ)

For EL signal homing mode with negative direction homing and EZ alignment, the control axis stops at EZ signal edge after home movement is completed as shown in figure below:



4.4.3 Single EZ Signal Homing

Most linear motor mechanism set up only one position mark signal. This mode is used in the said mechanism.

Figure below illustrates how to set up Home mode 2 (single EZ signal) with positive direction homing. After home movement is completed the control axis stops at the edge of EZ signal.

- **Relevant axis parameters setup**

Axis parameters	Axis parameter values	Description to axis parameter value
PRA_HOME_MODE	**2**	Employing home mode 2 (single EZ signal) homing
PRA_HOME_DIR	**0**	Employing positive direction forward homing
PRA_HOME_EZA	0	Further align with signal EZ, 0: No, 1: Yes
PRA_HOME_S	0	S-curve factor
PRA_HOME_ACC	ACC	Acceleration and deceleration in unit of (distance unit of measure/sec.^{2})
PRA_HOME_VS	VS	Initial speed in unit of (distance unit of measure/sec.)
PRA_HOME_VM	VM	Speed of original position searching in unit of (distance unit of measure/sec.)
PRA_HOME_VO	VO	Homing speed in unit of (distance unit of measure/sec.)
PRA_HOME_SHIFT	0	Shift amount of homing position (distance unit of measure)

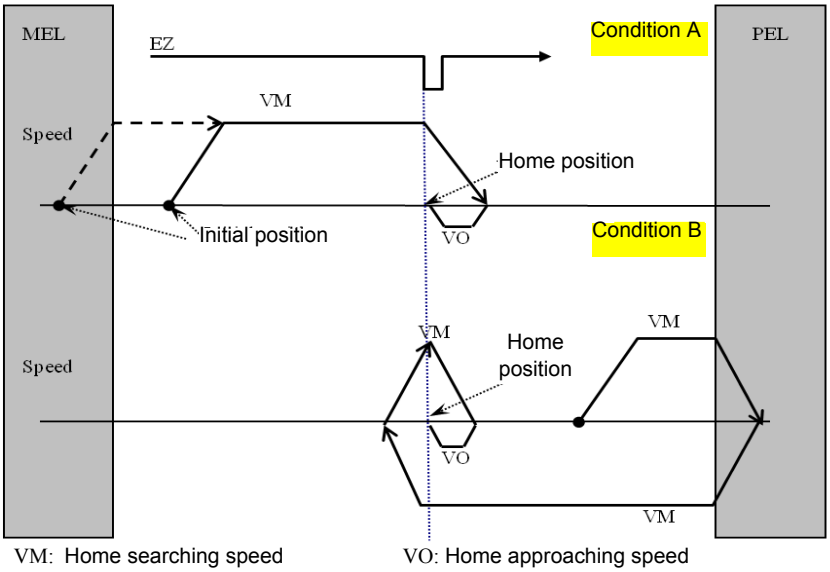


Figure 4-28: Home mode 2 (Case: EZ)

Figure below set up "Home mode 2 (single EZ signal)" with negative direction homing. After home movement is completed the control axis stops at the edge of EZ signal.

- **Relevant axis parameters setup**

Axis parameters	Axis parameter values	Description to axis parameter value
PRA_HOME_MODE	**2**	Employing home mode 2 (single EZ signal) homing
PRA_HOME_DIR	**1**	By negative direction forward homing
PRA_HOME_EZA	0	Further align with signal EZ, 0: No, 1: Yes
PRA_HOME_S	0	S-curve factor
PRA_HOME_ACC	ACC	Acceleration and deceleration in unit of (distance unit of measure/sec.^{2})
PRA_HOME_VS	VS	Initial speed in unit of (distance unit of measure/sec.)
PRA_HOME_VM	VM	Speed of original position searching in unit of (distance unit of measure/sec.)
PRA_HOME_VO	VO	Homing speed in unit of (distance unit of measure/sec.)
PRA_HOME_SHIFT	0	Shift amount of homing position (distance unit of measure)

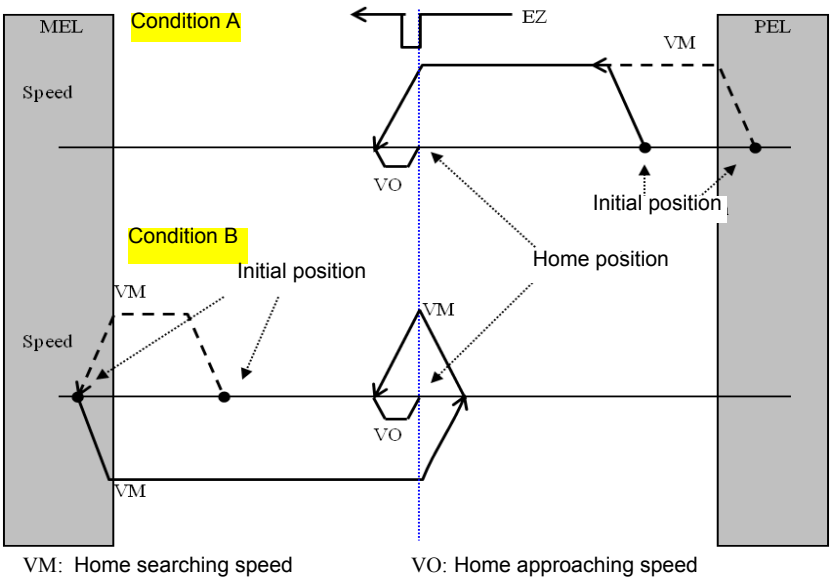


Figure 4-29: Home mode 2 adverse (Case: EZ)

NOTE

In this mode parameter PRA_HOME_EZA is functionless

4.5 Velocity Move

In this motion mode, the motion axis move along specified speed profile after proper command is received. Movement continues untill a stop movement command is received. In velocity movement mode functions listed below are supported:

- Dynamic changing maximum speed: You may change to any maximum speed during movement.
- Dynamic giving point-to-point (PTP) command: Switch velocity movement to PTP movement and then move to given position.
- Synchronized trigger: This movement can set to be enabled by trigger. When proper command is received, the axis enters a waiting-for-trigger-signal status and starts moving after the triggering signal is received. When multiple axes are in waiting-for-trigger-signal status you may send triggering signal at the same time for synchronized enabling. Please note that movement of each axis is independent from each other and so the end time varies with setup values of given parameters.

Relevant APS API described below:

I32 APS_vel (…); // give velocity movement (with F64 data format)

I32 APS_vel_all (…); // give velocity movement and all velocity parameters

I32 APS_stop_move (…); // stops by deceleration

I32 APS_emg_stop (…); // stops immediately

I32 APS_stop_move_multi (…); // give stop commands to multiple axis concurrently

I32 APS_emg_stop_multi (…); // give immediate stop commands to multiple axis concurrently

I32 APS_vel (…); // give velocity movement (with I32 data format)

I32 APS_move_trigger (…); // give synchronized startup command

- **Relevant axis parameters**

Param. No.	Define symbol	Description
20h (32)	PRA_CURVE	S-curve factor
21h (33)	PRA_ ACC	Acceleration in unit of (distance unit of measure/sec.^{2})
23h (34)	PRA_ VS	Initial speed in unit of (distance unit of measure/sec.)
24h (35)	PRA_VM	Maximum speed in unit of (distance unit of measure/sec.)

- **Example 1:**

  Set up parameters and start up velocity movement. See below for example process:

  1. Change maximum speed after 2 seconds
  2. Change maximum speed after 2 seconds
  3. Stop by deceleration after 2 seconds

```
#include "APS168.h"
#include "APS_define.h"
#include "ErrorCodeDef.h"
void velocity_move_example()
{
  I32 axis_id = 0;
  F64 speed_1 =  500.0;
  F64 speed_2 = 1000.0;
  F64 speed_3 =  600.0;

  APS_set_axis_param_f( axis_id, PRA_STP_DEC, 10000.0 );
  APS_set_axis_param_f( axis_id, PRA_CURVE,      0.5 ); //Set acceleration rate
  APS_set_axis_param_f( axis_id, PRA_ACC,    10000.0 ); //Set acceleration rate
  APS_set_axis_param_f( axis_id, PRA_DEC,    10000.0 ); //Set deceleration rate

  APS_vel( axis_id, 0, speed_1, 0 );  // Start a velocity move
  Sleep( 2000 );                      // Wait 2 second
  APS_vel( axis_id, 0, speed_2, 0 );  // Change speed on the fly
  Sleep( 2000 );                      // Wait 2 second
   APS_vel( axis_id, 0, speed_3, 0 );  // Change speed on the fly
  Sleep( 2000 );                      // Wait 2 second

  APS_stop_move( axis_id );           // Stop
}
```

Motion control input signal EMG, ALM, PEL, and MEL may lead to termination of movement, please refer to sections about safety protection

In velocity movement mode the target position may be updated from time to time as the command position does.

4.6 Jog Move

Jog operation is commonly available at control panel of machine. Its main function is to manually control the movement of motion axis or function together with mechanical switch with digital input to use DI signal as the jog movement startup signal. You may use switch on control panel to operate jog movement by setting up relevant parameters instead of coding control program.

There are two jog movement modes:

1. **Continuous mode:** This is similar to velocity movement with given maximum speed and acceleration profile. When JOG-ON signal (*) produces rising edge event trigger then the specified control axis starts velocity movement. When JOG-ON signal (*) produces falling edge event trigger then the specified control axis starts deceleration stop. See figure below:

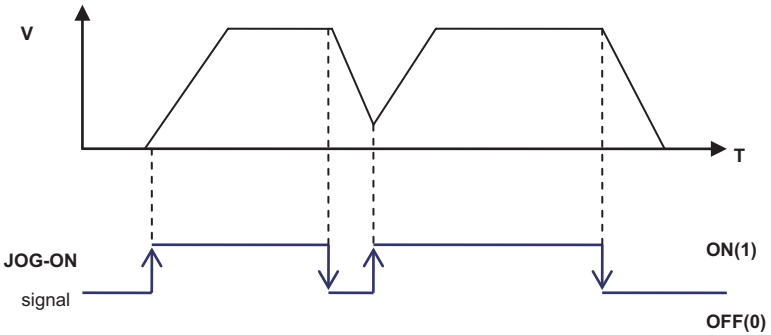


Figure 4-30: Relation between V-T chart of JOG movement and JOG-ON signal

NOTE

JOG-ON signal is a software digital signal where ON is represented by value 1 and OFF by 0

2. **Step mode:** In addition to velocity parameters this mode requires specific offset and so is easy for stop position prediction. After the JOG-ON control signal is triggered at the rising edge, the axis being controlled moves a distance of given offset then stops, pauses for a period of time (known as the delay time), if the control signal remains ON in delay time, the control axis moves in given speed profile until the signal disappears. It differs from continuous mode in that the total displacement will be increased to integer times of given offset value. This is useful in achieving more precise offset control during fine tuning.

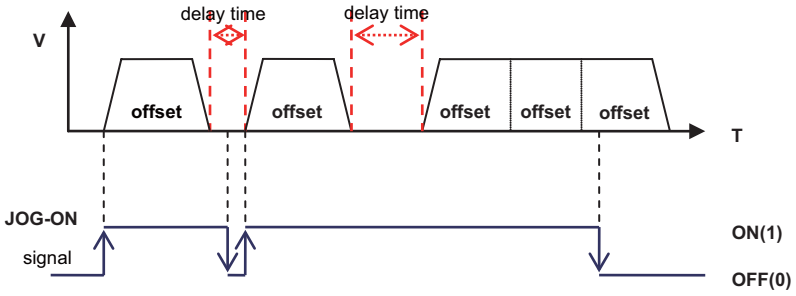


Figure 4-31: Jog step mode

- JOG-ON and digital input signal linkage:

The JOG-ON control signal not only can be given with API function but also can be used in setting digital input signal as control signal. You may set up axis parameter 48h, 49h, 4Ah, and 4Bh in two methods (distinguished by number of DI points):

1. Use two DI channel and set one of it to positive direction movement JOG-ON signal and the other to negative direction movement JOG-ON signal
2. Use one DI channel and set it to JOG-ON signal with its direction to be determined by axis parameter.

Relevant APS API described below:

I32 APS_jog_on (...); // give velocity movement command

- **Relevant axis parameters**

Parameter code	Parameter definition	Meaning of parameter value
40h ()	PRA_JG_MODE	Set up JOG mode [0: Continuous, 1: Step]
41h ()	PRA_JG_DIR	Set up JOG direction: [0: Negative, 1: Positive direction]
42h ()	PRA_JG_SF	Set up JOG S factor [0 ~ 1]
43h ()	PRA_JG_ACC	Set up JOG acceleration [Value 0]
44h ()	PRA_JG_DEC	Set up JOG deceleration [Value 0]
45h ()	PRA_JG_VM	Set up JOG Max. velocity [Value 0]
46h ()	PRA_JG_OFFSET	Set up JOG offset position. Step mode use [Value = 0]
47h ()	PRA_JG_DELAY	Set up JOG delay time, step mode use [0 ~ 10,000,000] us
48h ()	PRA_JG_MAP_DI_EN	Set up JOG signal and DI signal correlation, that is opposite direction DI signal
49h ()	PRA_JG_P_JOG_DI	Set DI signal of certain channel to positive direction JOG signal
4Ah ()	PRA_JG_N_JOG_DI	Set DI signal of certain channel to negative direction JOG signal
4Bh ()	PRA_JG_JOG_DI	Set DI signal of certain channel to JOG signal with direction settings by 0x41

- **Example:**

```
#include "APS168.h"
#include "APS_define.h"
#include "ErrorCodeDef.h"

void jog_move_example()
{
  //This example shows how jog move work
}
```


NOTE

1. Motion control input signal EMG, ALM, PEL, and MEL may lead to movement termination. Please refer to safety protection related sections.
2. In continuous mode the target position may be updated from time to time (so does the command position).
3. When control axis is in jog movement, other movement commands is disabled to prevent malfunctions.
4. When the control axis is running other movements (e.g. home movement) the jog command will be ignored.

4.7 Point-to-Point Move

4.7.1 Point-to-Point Move

Point-to-Point movement (PTP movement) is to move one axis from position A to position B at given speed. PTP movement can be relative or absolute movement based on its given position parameter.

This controller provides T-curve and S-factor adjustable S-curve. Each profile contains start velocity, maximum velocity, end velocity, and acceleration / deceleration parameters that can be adjusted individually as shown in figure below. See Acceleration / Deceleration Profile in Section 4.3.3 for detail.

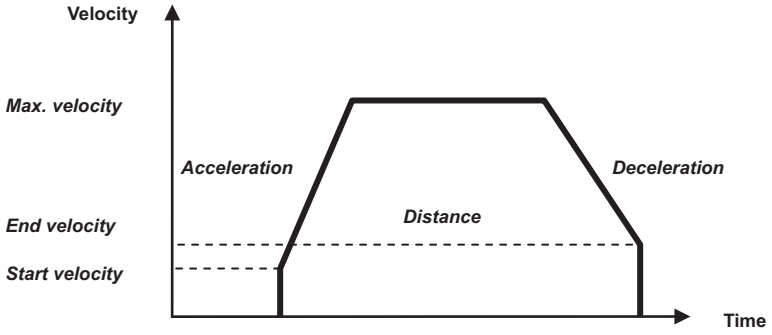


Figure 4-32: T-curve V-T chart

In addition, the APS_motion_status () function can be used to get motion status data of each axis to identify the end of PTP movement. See Section 4.9 for description on motion status. You may use APS_stop_move () or APS_emg_stop () function to abort in-progress movement.

Relevant APS API described below:

I32 APS_ptp (); // PTP move

I32 APS_ptp_v (); // PTP move with maximum speed parameter

I32 APS_ptp_all (); // PTP move with all speed parameter

I32 APS_relative_move (); // Relative PTP move in I32 data format

I32 APS_absolute_move (); // Absolute PTP move in I32 data format

I32 APS_stop_move (); // deceleration stop

I32 APS_emg_stop (); // immediately stop

- **Relevant axis parameters**

Param. No.	Define symbol	Description
07h (7)	PRA_SD_DEC	stop_move (), deceleration rate
20h (32)	PRA_SF	S-factor
21h (33)	PRA_ACC	Acceleration rate
22h (34)	PRA_DEC	Deceleration rate
23h (35)	PRA_VS	Start velocity
24h (36)	PRA_VM	Maximum velocity
25h (37)	PRA_VE	End velocity

4.7.2 Synchronous Start

Synchronous start: This movement can set to be enabled by trigger. When proper command is received, the axis enters a waiting-for-trigger-signal status and starts moving after the trigger is received. When multiple axes are in waiting-for-trigger-signal status you may send trigger at the same time for synchronized enabling. Please note that movement of each axis is independent from each other and so the end time varies with amount and acceleration profile.

4.7.3 On The Fly Change

You may dynamically change position and velocity parameter in PTP movement process by methods described below:

1. Dynamically change to new position while the velocity parameter remain intact.
2. Dynamically change the maximum velocity while target position remain intact.
3. Dynamically change to new position and speed profile. That is, give whole new PTP command.

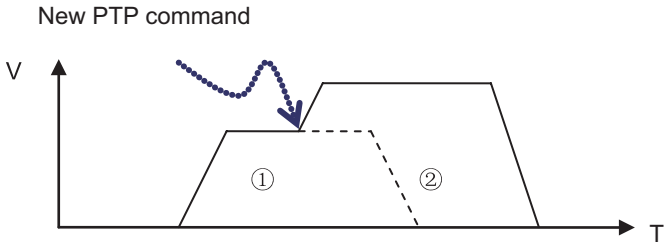


Figure 4-33: Dynamically change position and velocity

4.7.4 Continuous PTP Move

Each axis features one motion buffer that can contain 10 commands. After the first PTP command is received, the axis starts moving immediately. You may give PTP commands during movement in process. Following commands are stored in buffer queuing for execution. After the first movement arrived at given position, the controller continuous executing PTP commands in buffer until there is no new command in existence. You may set up blend mechanism for speed commands. Available speed command blending mechanism are:

1. Buffered: blending speed commands in unit of buffer
2. Blend low: Blend with the one with slower maximum speed
3. Blend high: Blend with the one with faster maximum speed
4. Blend previous: Blend in the maximum speed of the previous one
5. Blend next: Blend in the maximum speed of the next one

Take example. V-T chart with 3 continuous PTP movements and different speed blending settings:

1. **Buffered**

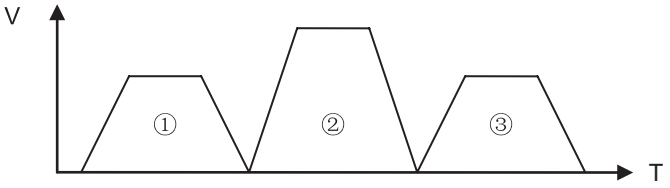


Figure 4-34: Continuous three position V-T chart

2. **Blend low: Blend with the one with slower maximum speed**

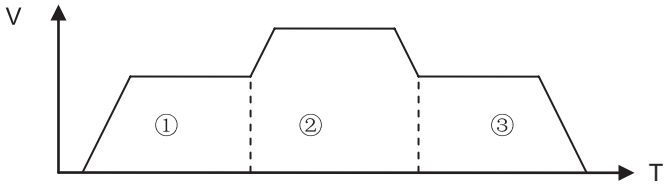


Figure 4-35: Continuous three position V-T chart (auto speed connection (1)

3. **Blend high: Blend with the one with faster maximum speed**

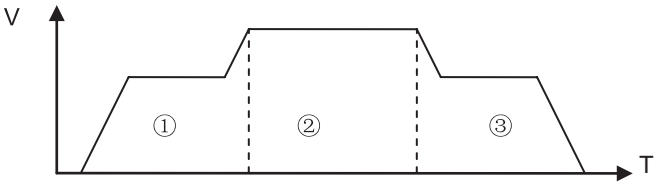


Figure 4-36: Continuous three position V-T chart (auto speed connection (2)

4. **Blend previous: Blend in the maximum speed of the previous one**

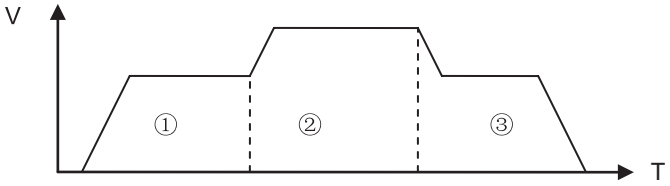


Figure 4-37: Continuous three position V-T chart (auto speed connection (3)

5. **Blend next: Blend in the maximum speed of the next one**

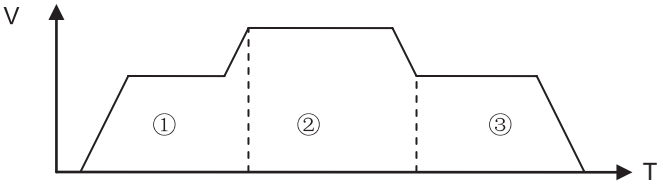


Figure 4-38: Continuous three position V-T chart (auto speed connection (4)

4.8 Interpolation

Interpolation is a multi-axes locus movement based on given locus properties, e.g. center of circle and end point, and velocity data. The controller then calculate relations between path and time. Axis involved in interpolation start up at the same time and end at the same time after operation completed.

This controller supports couple of interpolation including straight line interpolation of any 2~6 axes, arc interpolation of any 3 axes and spiral interpolation of any 3 axes.

The interpolation usually requires two or more axes to complete. Required axes are assigned by array input. Here the axes array is defined by assuming the first axis ID to be the reference axis.

Axis parameters of the reference axis is used as the basis for setting up speed profiles of interpolation while settings of speed profile are all in vector (composite). Take example. To execute a 2 axes straight line interpolation by Axis 1 and Axis 2, the axis ID array can be declared as described below. In example below, the first axis ID in axes_array is Axis 1. Then the interpolation movement required initial speed, maximum speed, ending speed, and acceleration rate are all set up relative to axis parameters of Axis 1. If Axis 2 is the first element in axes_array then Axis 2 is the reference axis.

I32 axes_array[2] = { 1, 2 };

Sections below describe straight line, arc, and spiral interpolation mechanism and operation and followed by continuous interpolation.

4.8.1 Linear Interpolation

This controller supports up to six axes straight line interpolation. After the straight line interpolation command is received, all relevant axes start up at the same time and move according to specified (relative or absolute) position, speed, and acceleration profiles, all relevant axes stop concurrently. The speed profiles are set by synthetic vector.

Assume you want to make a N (N=2~6) axes straight line interpolation with offset of each axes represented by ΔX0,ΔX1… ΔXN-1 then the synthetic shift ΔP shall be:

$$\Delta P = \sqrt{(\Delta X_0)^2 + (\Delta X_1)^2 + ... + (\Delta X_{N-1})^2}$$

If synthetic velocity is set to V, the velocity of each axes Vn should be:

$$V_n = \frac{\Delta X_n}{\Delta P} \times V$$

See figure below for a two dimension straight line interpolation with starting point at S and ending point at E:

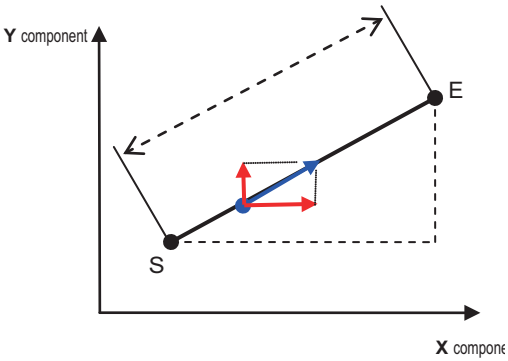


Figure 4-39: Two-dimension straight line interpolation

ΔX and ΔY is the offset at X-axis and Y-axis respectively. Interpolation distance is set according to component of each axis (e.g. relative distance ΔX and ΔY or absolute coordinates of ending point E). The composite shift ΔP can be calculated by formula described below:

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

Velocity and acceleration are set by composite vectors. Component of each axis, take composite velocity Vx and Vy as example, can be calculated by formula described below:

$$V_x = \frac{\Delta X}{\Delta P} \times V \quad V_y = \frac{\Delta Y}{\Delta P} \times V$$

Relevant APS API described below:

I32 APS_line (); // multi axes straight line interpolation

I32 APS_line_v (); // multi axes straight line interpolation with maximum speed settings

I32 APS_line_all (); // multi axes straight line interpolation with all speed settings

I32 APS_stop_move (); // deceleration stop

I32 APS_emg_stop (); // immediately stop

I32 APS_absolute_linear_move (); // straight line interpolation with given absolute position
(in I32 data format)

I32 APS_relative_linear_move (); // straight line interpolation with given relative position
(in I32 data format)

- **Relevant axis parameters**

Param. No.	Define symbol	Description
07h (7)	PRA_SD_DEC	stop_move (), deceleration rate
20h (32)	PRA_SF	S-factor
21h (33)	PRA_ACC	Acceleration rate
22h (34)	PRA_DEC	Deceleration rate
23h (35)	PRA_VS	Start velocity
24h (36)	PRA_VM	Maximum velocity
25h (37)	PRA_VE	End velocity

4.8.2 Arc Interpolation

This controller supports 2-dimension and 3-dimension arc interpolation as well as multiple input methods to deal with demands of various applications. 2D and 3D arc movement and its command composition methods are described below.

4.8.2.1 3D Arc Interpolation

An arc can be described by

method 1: given center of circle, angle and normal vector

method 2: given center of circle and end point

Relevant commands are described below:

Function name	Description
APS_arc3_ca APS_arc3_ca_v APS_arc3_ca_all	Execute 3-dimension arc interpolation with center, angle, and normal vector
APS_arc3_ce APS_arc3_ce_v APS_arc3_ce_all	Execute 3-dimension arc interpolation with center and end point **Limit:** Cannot execute half or full circle interpolation

- **method 1: given center of circle, angle and normal vector as shown in figure below:**

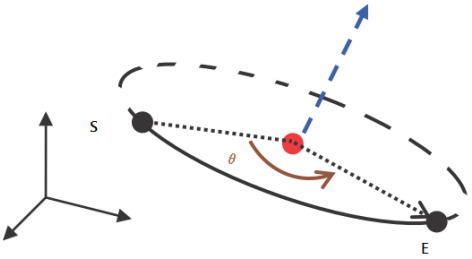


Figure 4-40: Three-dimension arc interpolation (method 1)

This entry method is easy for you to create an arc track without the restriction of half or full circle. You must ensure the correctness of normal vector. This controller is default to correct your normal vector value if necessary.

- **Auto Normal Vector Correct:**

If the normal vector entered by you is not the orthogonal vector from center of circle to starting point the controller correct your entry value automatically by Gram-Schmidt normalization,

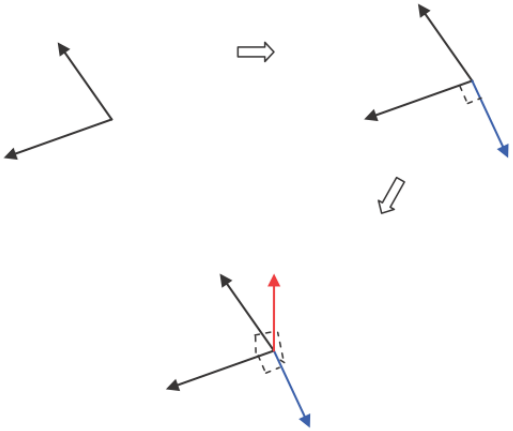


Figure 4-41: Defining spatial normal vector

- **How to determine arc direction and path of multiple laps**

Use the right-hand grip rule as shown in figure below, where the your thumb indicates normal vector direction and the other four fingers the positive rotating direction. Enter negative value for angle parameter to rotate in opposite direction.

Set up angle value directly to execute multiple laps (circles greater than 360 degrees), e.g. 2 laps = 720 degree.

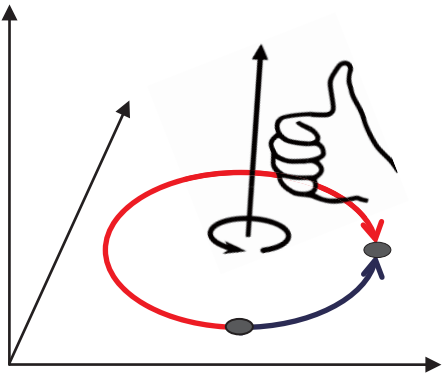


Figure 4-42: Determining arc direction in space

Coordinates of end point may have certain error caused by computing accuracy of your computer. To get precise end point position, you may use method 2 to enter exact end position accurately (as described in next section)

➢ **method 2: given center of circle and end point**

This method requires center of circle and position of end point only. Benefit of this method lies in that it does not need a normal vector and that it can have accurate ending position to meet demands from contour or applications that need accurate positioning. This method has two restrictions:

1. It cannot execute half circle (angle of 180 degree)
2. It cannot execute full circle (angle of 360 degree)

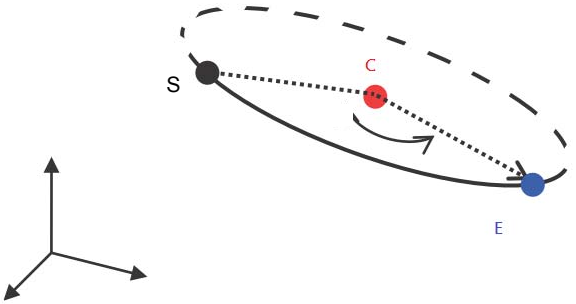


Figure 4-43: Three dimension arc interpolation (method 2)

- **How to determine direction of arc**

Use the right-hand grip rule as shown in figure below, where the your thumb indicates normal vector direction and the other four fingers the positive rotating direction.

Use parameter “I16 Dir” to determine direction, if Dir>= 0 then rotate in positive direction and negative direction if DIR <= -1

- **Path of multiple laps (arcs of angle greater than 360 degree)**

Use parameter “I16 Dir” to determine rotating angle with formula described below:

$$\text{Angle} = \theta + \text{Dir} * 2\text{PI}$$

Take example. If θ = 30 degree, then

Dir	calculation formula	Angle (Degree)
0	30 + 0 x 360	30
1	30 + 1 x 360	390
2	30 + 2 x 360	750
-1	30 + (-1) x 360	-330
-2	30 + (-2) x 360	-690

- **Example:**

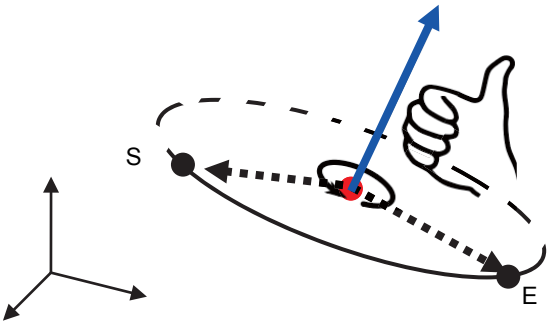


Figure 4-44: Three dimension arc interpolation example

4.8.2.2 2D Arc Interpolation

Same as 3D arc, two description methods are provided for 2D arc:

Method 1: given center of circle and angle

method 2: given center of circle and end point

2D arc has the same setup method as the 3D one does. See 3D arc interpolation for detail

Relevant APS API described below

I32 APS_arc2_ca ();
I32 APS_arc2_ca_v ();
I32 APS_arc2_ca_all ();
I32 APS_arc2_ce ();
I32 APS_arc2_ce_v ();
I32 APS_arc2_ce_all ();

4.8.2.3 Helical Interpolation

This controller supports 3-dimension helical interpolation (also known as Spiral–Helix interpolation) as well as multiple input methods to deal with demands of various applications. See below for its setup:

Method 1: Given center of circle and angle (Center-Angle)

Method 2: Given center of circle and end point (Center-End)

Both methods are described below.

➢ **Method 1: Given center of circle and angle (Center-Angle)**

See table and figure below for helical curve parameters

Parameters	Description
Center point	Center of circle (relative or absolute)
Angle	Starting point and ending point angle projected at the circle plane of starting point (as shown in figure below). Plus and minus sign indicate directions.
Normal vector	Normal vector of starting point circle plane
Height	Cone height (relative)
Final radius	Radius of circle where the ending point is

- **Example:**

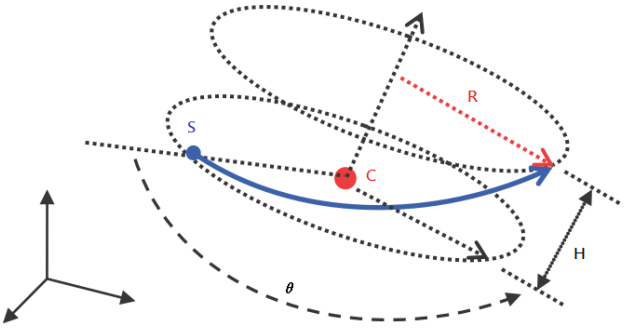


Figure 4-45: Three dimension spiral interpolation (method 1)

➢ Method 2: Given center of circle and end point (Center-End)

See table and figure below for helical curve parameters

Parameters	Description
Center point	Center of circle (relative or absolute)
Normal vector	Normal vector of starting point circle plane
End point	Ending point of cone (relative or absolute)
Direction	Rotating direction and laps

Direct parameters can be set up as the 3D arc does. See prior section for detail.

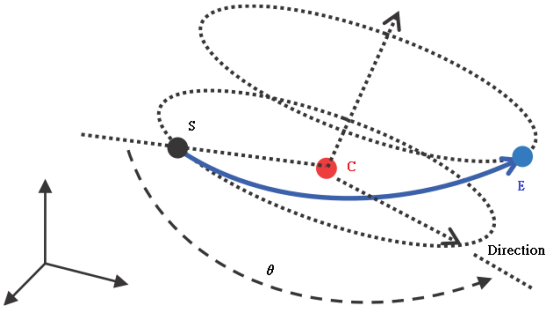


Figure 4-46: Three-dimension spiral interpolation (method 2)

All helical interpolation input methods described above requires giving normal vector. If there is error with the normal vector, the controller corrects it automatically. See Section 4.8.2 Arc interpolation for correction method.

Relevant APS API described below:

Input method	API	Description
Method 1 Center-Angle	I32 APS_sprial_ca ();	Start up 3D helical interpolation
	I32 APS_sprial_ca_v ();	Start up 3D helical interpolation + maximum velocity parameter
	I32 APS_sprial_ca_all ();	Start up 3D helical interpolation + all velocity parameter setup
Method 2 Center-End	I32 APS_sprial_ce ();	Start up 3D helical interpolation
	I32 APS_sprial_ce_v ();	Start up 3D helical interpolation + maximum velocity parameter
	I32 APS_sprial_ce_all ();	Start up 3D helical interpolation + all velocity parameter setup

Relevant axis parameters

Param. No.	Define symbol	Description
07h (7)	PRA_SD_DEC	stop_move (), deceleration rate
20h (32)	PRA_SF	S-factor
21h (33)	PRA_ACC	Acceleration rate
22h (34)	PRA_DEC	Deceleration rate
23h (35)	PRA_VS	Start velocity
24h (36)	PRA_VM	Maximum velocity
25h (37)	PRA_VE	End velocity

4.8.3 Continuous Interpolation

With continuous interpolation the controller continuously executes multiple interpolation paths including straight line, arc and helical interpolations described above. You do continuous interpolation by giving multiple interpolation commands in sequence. These commands shall be saved in buffer of the controller queueing for execution.

Figure 4-47: Illustration on continuous interpolation (Buffer) movement

For continuous interpolation the only restriction is that the dimension and axis ID must remain the same. E.g., 3D straight line must link with 3D arc but not 2D arc and axis to be used must be the same as well. This controller provides seven setup methods for speed blending between two adjacent paths.

1. **Aborting and blending**
2. **Aborting forced**
3. **Aborting stop**
4. **Buffered**
5. **Blending when deceleration start**
6. **Blending when residue-distance met**
7. **Blending when residue-distance % met**

You can set up this with input parameter "Flag". See ASP API user manual for detailed parameter description.

In essence, the first three methods, method (1), (2) and (3), stops any running interpolation when new interpolation command is received and start executing the new interpolation command immediately. Pending interpolation command in motion buffer shall be cleared now. These three methods differ in their termination approach and are commonly used in cases where immediate interpolation path change is required.

Method (4), (5), (6) and (7) executes by sequence in motion buffer. Method 4 features its exact execution per interpolation path and speed schedule without any path error.

Methods (5), (6) and (7) employs a mechanism of speed blending. Its benefits with smooth track, vibration free, and constant motion speed. These three methods differ in their beginning time of blending which may affect the actual calculation path in blending process and path errors against user's plan. You may select and adjust as required. These seven speed link methods are described below:

1. Aborting and blending

If the "aborting" is received the interpolation command take effects immediately and operation transfer to new command without slowing down the speed. The controller smooth the transition track automatically to avoid vibration and to ensure speed component of each axes translated smoothing. Take figure below. The first linear interpolation command is for moving from position S1 to E1, a "aborting" linear interpolation command is executed for moving to position E2. Figure in the left is the track diagram. The controller calculate smooth track (red line) automatically to transfer to new interpolation command. Figure at right hand side is a track combined velocity-time (V-T) chart.

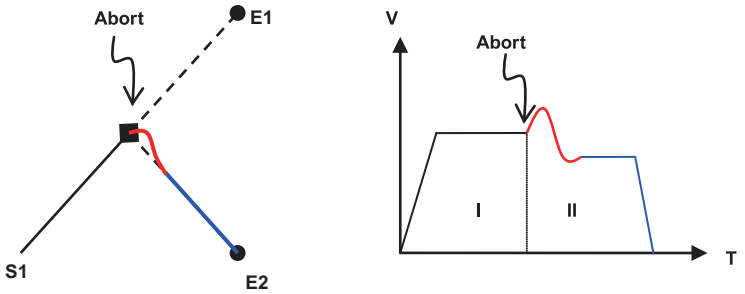


Figure 4-48: Velocity blending (method 1)

2. Aborting forced

Characteristic of this kind of command is that the track transfer to new command immediately. The controller makes no smoothing treatment and so the motion track match with the command exactly. In this mode speed component of each axes may become un-smooth. You must pay special attention to transfer speed and angle to prevent vibration from happening.

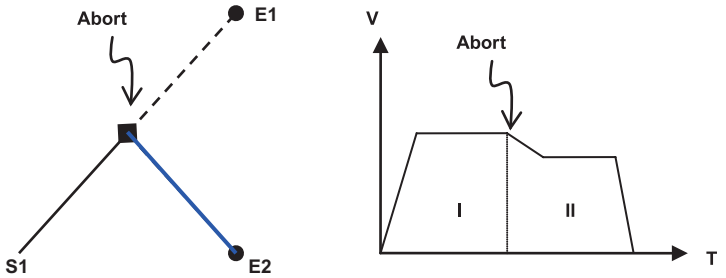


Figure 4-49: Velocity blending (method 2)

3. Aborting stop

After command is received the original interpolation command slows down to stop (deceleration rate adjustable), new interpolation command starts after movement fully stopped.

Take figure below. When executing a straight line interpolation command from S1 to E1, a "abort - decelerate" interpolation command is given at position E1'. The controller then slows down according to given deceleration rate and stops at S2, and then move to E2. Please note that if position is given by relative distance the relative starting point is E1' rather than S2.

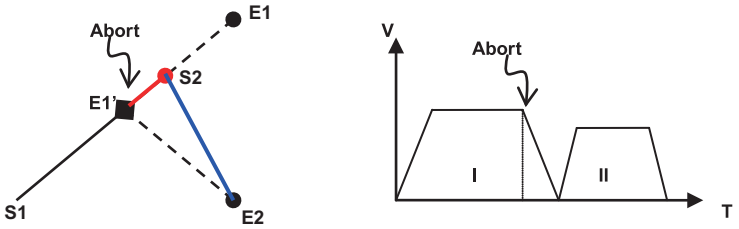


Figure 4-50: Velocity blending (method 3)

4. Buffered

When new interpolation command is received it is saved in motion buffer first. Commands in queue then continues to execute after the original interpolation command is finished. Take figure below. When executing a straight line interpolation command from S1 to E1, a "buffered" interpolation command is given during movement in progress. The controller then saves interpolation command in queue and move from E1 to E2 after interpolation command is completed. The profile follows user settings exactly. To ensure no slow down or minor slowdown between two interpolation commands you may set up ending speed of previous interpolation command and starting speed of next interpolation command properly.

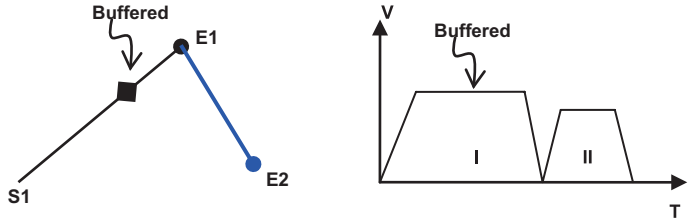


Figure 4-51: Velocity blending (method 4)

5. Blending when deceleration start

When new interpolation command is received it is saved in motion buffer first. When original interpolation command starts slowing down the new interpolation command also starts for blending as shown in figure below. You may determine blending time by adjusting deceleration rate. The higher the deceleration rate is the smaller the blending region will be and the smaller deviation form original interpolation command path will be.

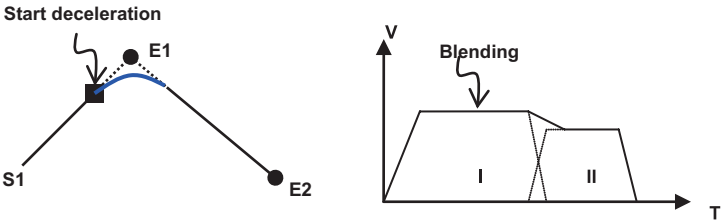


Figure 4-52: Velocity blending (method 5)

6. Blending when residue-distance met

The controller saves newly received command in motion buffer first. You may set up an offset amount, e.g. the so called residual distance as shown in figure below, and start the new interpolation command for blending after the distance of original interpolation command path from target position is smaller than the residual distance (E1) as shown in figure below.

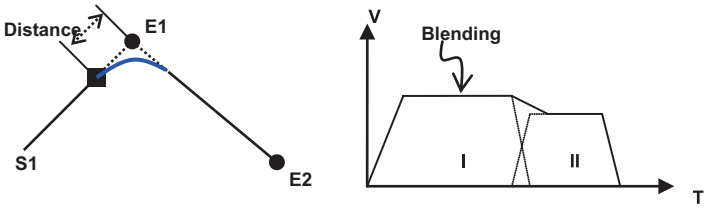


Figure 4-53: Velocity blending (method 6)

5. Blending when residue-distance % met

Similar to method 6 but with given residual distance ratio (percentage of residual distance to interpolation distance) as the P% in figure below.

Take figure below. If the residual distance ratio is set to 10% and the straight line interpolation distance from S1 to E1 is 1000, the the next interpolation command starts for blending when the motion axis move to position 900 (relative to starting point).

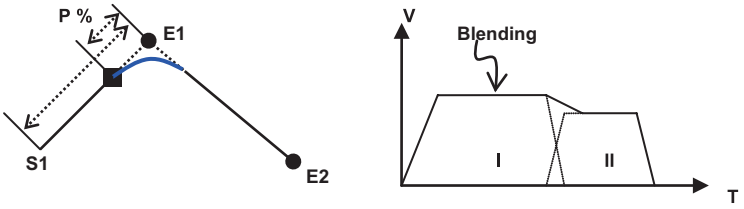


Figure 4-54: Velocity blending (method 7)

- **Example:**

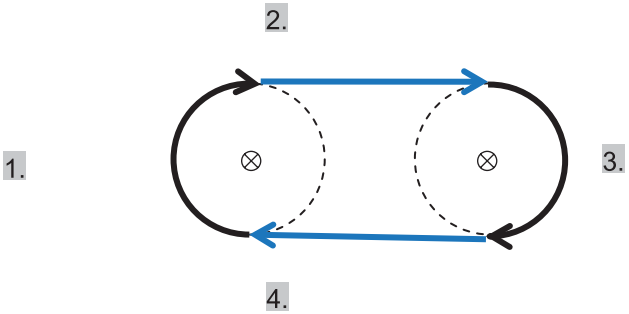


Figure 4-55: Continuous interpolation examples

4.9 Motion Status Monitoring

During the motion control process it is necessary to monitor motion status of control axis and convert to next process control at appropriate time. Take example. During system initialization the upper control program (the control program of user) execute home operation to each control axis at first. The controller starts home movement once the command is received and the control program must wait for the completion of home operation. Usually the polling method is used to determine the completion of homing process. That is, read motion status signal of controller at regular time span. Next stage control operation starts only after current movement is completed.

In addition, there may be exceptional situations occurred during motion operation. The upper control program must be able to detect abnormality and deal with exceptions accordingly. Take example. When emergency stop button is pressed during home movement or the end limit signal is triggered during movement. See figure below for basic flow chart of home movement.

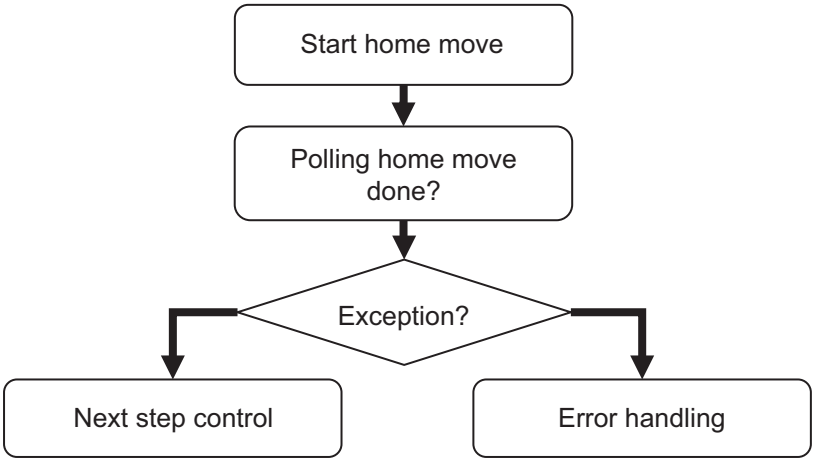


Figure 4-56: Motion status monitoring process

Motion control status and its behavior provided by this controller shall be described in Section 4.9.1.

4.9.1 Motion Status

Use following API functions to read motion status of each axes:

I32 APS_motion_status ();

Motion status data of individual axis is combined in return parameter I32 (32 bit integer). See table below for motion status and meaning represented by each bit:

Bit No.	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Status		HMV	MDN	DIR	DEC	ACC	VM	CSTP
Bit No.	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Status	JOG				PTB	WAIT		
Bit No.	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Status					POSTD	PRED	BLD	ASTP
Bit No.	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**
Status				GER				

Table below describes meaning of motion status:

Bit No.	**Define**	**Description**
0	CSTP	End of single motion command
1	VM	At maximum speed
2	ACC:	Accelerating
3	DEC:	Decelerating
4	DIR:	Motion direction; 1: Positive and 0: Negative
5	MDN	End of motion command
6	HMV	Executing home movement
7~9		(Reserved)
10	WAIT	Waiting for motion trigger
11	PTB	Executing PTB movement
12~14		(Reserved)

Bit No.	Define	Description
15	JOG	Jog movement in progress
16	ASTP	Clear this signal after next movement is executed
17	BLD	The axis is running blending movement
18	PRED	Pre-offset event, clear this signal after next movement is executed
19	POSTD	Post-offset event, clear this signal after next movement is executed
20~27		(Reserved)
28	GER	In geared, the axis is a slave one
29~31		(Reserved)

Motion status is described below in sequence of bit:

Bit0~ bit4:

CSTP (Command stop): When this signal is ON, the controller is not sending movement command.

VM: when this signal is ON, the movement reached its maximum velocity settings.

ACC: when this signal is ON, the movement is accelerating.

DEC: when this signal is ON, the movement is decelerating.

DIR: when this signal is ON, the movement is moving at positive direction. When movement stops, the DIR saves status right before movement stopped.

Relation between movement and signal shown in figure below:

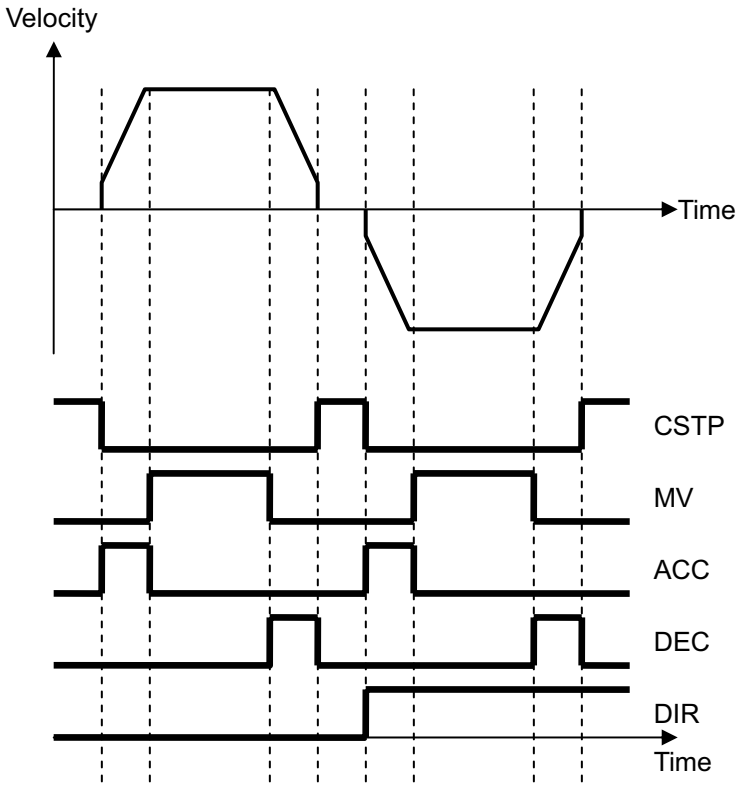


Figure 4-57: Relation of different motion signals VS motions

Bit 5: Motion Done – MDN

Single movement command or multiple movement command is completed. Single movement command is a single axis point to point movement and multiple axes point to point movement. Multiple movement is like homing movement combinated by a series of movement. With this signal you may use polling or interrupt event generation to schedule movement process.

Note: Abnormal movement stop will generate this signal as well. You may ensure abnormal movement stop by checking ASTP signal.

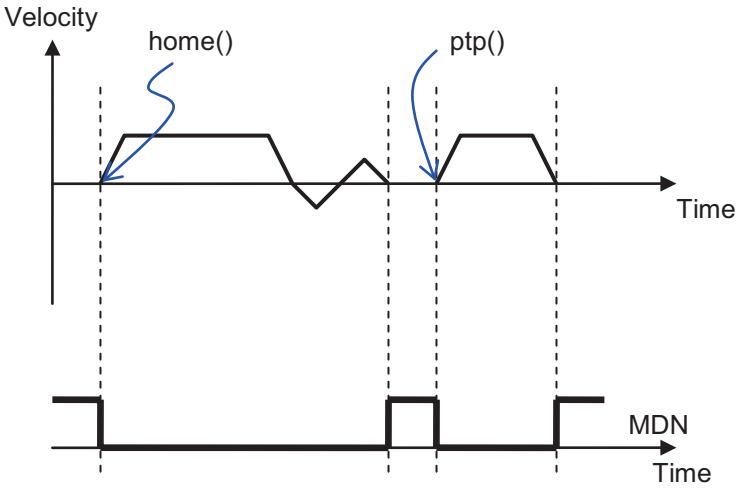


Figure 4-58: Relation of motion done (MDN) signals VS motion

Bit 6: In Homing signal - HMV

When home movement command home () is received at the controller and home movement starts being executed the HMV signal sets NO (=1). When home movement is completed or aborted, this signal is turned off (=0)

See Section 4.4 for detailed home movement.

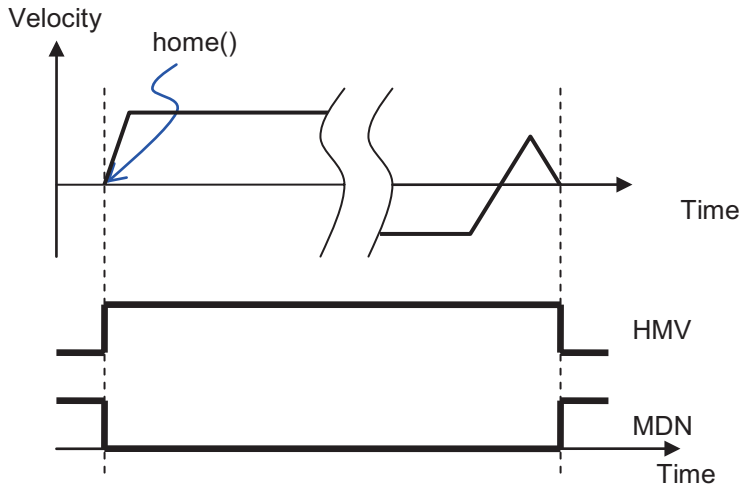


Figure 4-59: Relation of motion done (MDN), In-homing (HMV) signals VS motion

Bit10: Wait Move Trigger – WAIT

This signal is set ON when the signal is at status ready for movement triggering. When trigger is sent: Use move_trigger () function to trigger queueing axis.

When parameter Flag is set to MF_WAIT (0x00100) for motion control functions listed below, the relevant commands are set to trigger initiated. The target axis do not start up immediately, only the WAIT signal is set to ON.

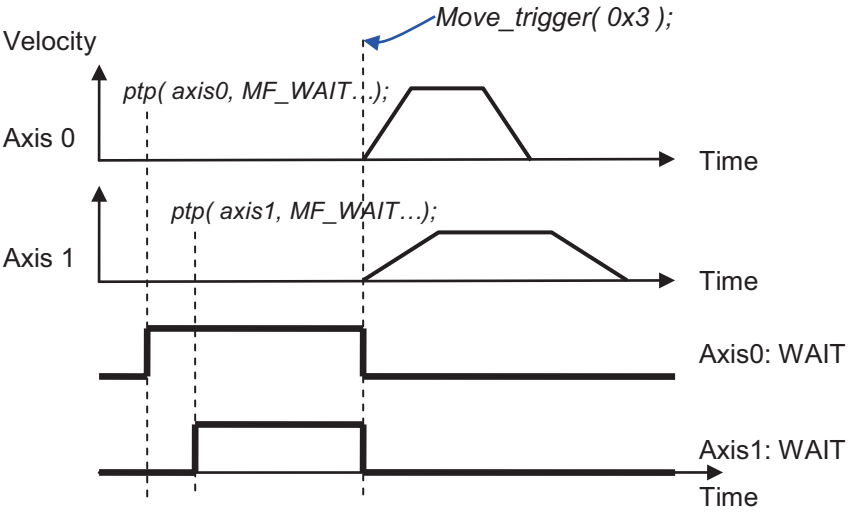


Figure 4-60: Relation of WAIT signals VS motion

Bit11: Point Buffer movement signal - PTB

When point buffer movement is started, this signal is set to ON and to OFF when movement is completed.

Bit 15: Jog movement signal - JOG

When an axis is doing jog movement, the JOG signal is set to ON and to OFF when jog movement is completed.

See Section 4.6 for detailed jog movement.

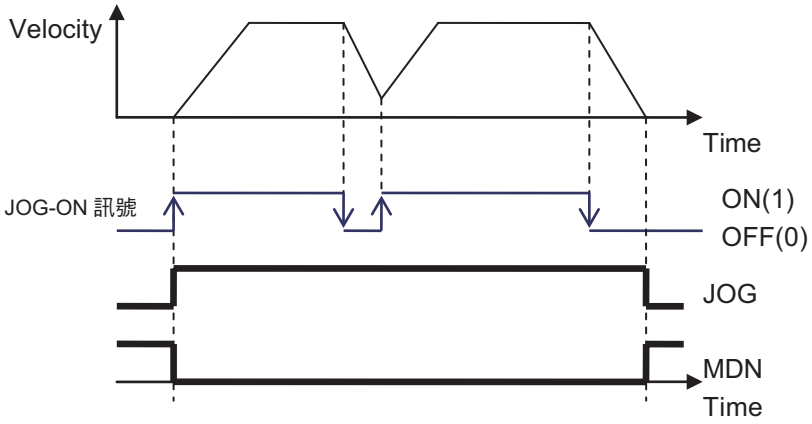


Figure 4-61: Relation of JOG and motion done(MDN) signals VS motion

Bit 16: Abnormal stop – ASTP

This signal turns on when movement is aborted by certain reasons. See table below for causes to abnormal stop. You may use get_stop_code () function to get abnormal stop code (Stop code). This code can be used in follow-up error handling procedure.

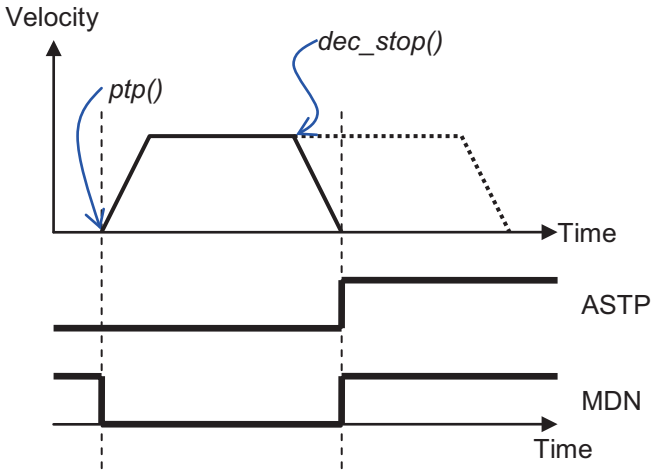


Figure 4-62: Relation of ASTP VS motion

Bit 17: Blending movement - BLD

Continuous interpolation has several speed succession method. The blending method has a transition region at the interconnection points of two paths (as shown in figure below). The BLD signal indicate that the axis is entering this area.

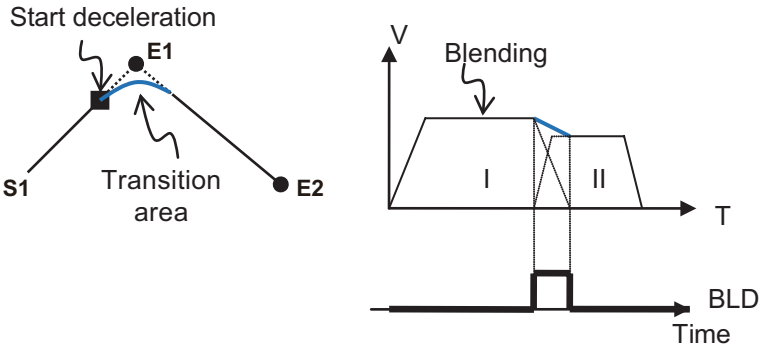


Figure 4-63: Relation of blending (BLD) signal VS motion

Bit 18 and 19: Pre- and post distance event

Every position movement command can set up pre-distance and post-distance to trigger the controller to issue signals when distance of movement meet given conditions.

The controller starts recording shift distance of movement when point to point movement is started. When distance of movement is greater than pre-distance, the pre-distance event occurs. Similarly, when remaining shift distance of the point to point movement is less than the post-distance then post-distance event occurs. Relation between movement and signal shown in figure below:

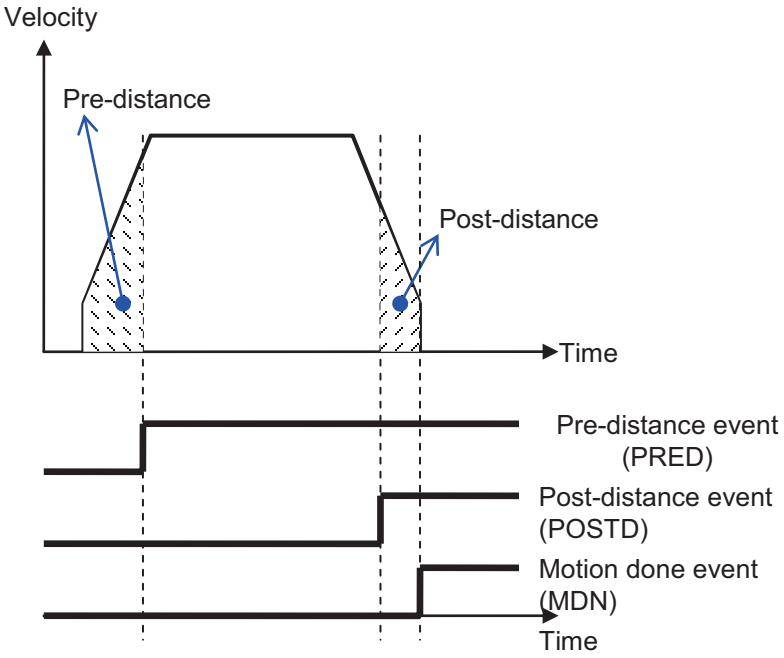


Figure 4-64: Relation between pre- and post distance event signals and movement

4.10 Application Functions

4.10.1 Electronic Gearing

Electronic gear function: You may set up movement relation of one axis (slave axis) against another axis (master axis) that is similar to a mechanical gear structure. Relation between two gears is usually expressed with gear ratio. Take example. For a pair of gears with gear ratio 1:2, then the Y (slave) axis rotate 2 turns when the X (master axis) rotate 1 turn. Similarly, you can set up electronic gear ratio so that when the master axis executes control motions, the slave axis rotate in accordance with given gear ratio.

This controller provides 2 modes: standard and gantry modes. These two modes differ in that the gantry mode is designed for dual drives gantry mechanism exclusively where two motors are used to drive one rigid connected mechanism. It features special safety and control behaviors. These two modes are described in sections below.

4.10.1.1 Standard Mode

To set up a electronic gear in standard mode: select a slave axis, all parameters and commands are set by reference to this axis. Set up axis parameters listed below before initiating electronic gear mode:

Param. No.	Define symbol	Description
60h (96)	PRA_GEAR_MASTER	Gear master
61h (97)	PRA_GEAR_ENGAGE_RATE	Engage rate
62h (98)	PRA_GEAR_RATIO	Gear ratio

Use APS_start_gear (slave axis ID) to enable electronic gear function after setup. After the electronic gear function is enabled, the slave axis moves along with the master axis by gear ratio setup values. As the master axis may not be motionless, you may set up appropriate engage rate for the slave axis to move at given velocity from zero to given gear ratio and to prevent vibration caused by very fast instant acceleration. In addition, gear ratio in this mode can be changed dynamically where the changing process is subject to engage rate.

Engage rate = Gear ratio / Engage time

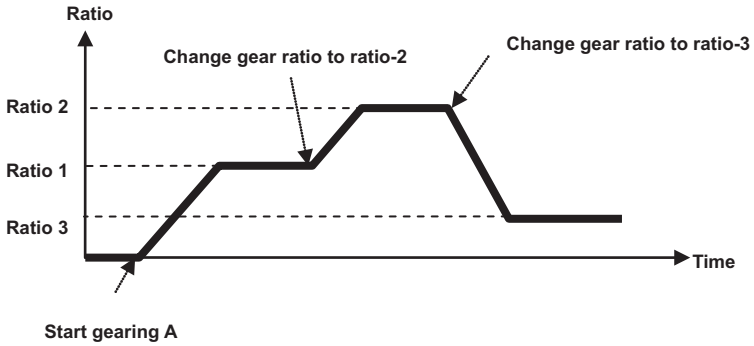


Figure 4-65: Adjust electronic gear's auto engagement speed

There are several conditions that may relieve gear relations in standard mode:

1. Relieve gear relation by APS_start_gear () manually
2. If the EMG / ALM / PEL / MEL / ALM signal of slave axis turns ON, the master axis is not affected if it is moving.
3. When slave axis received stop (), emg_stop (), and servo_off () commands

4.10.1.2 Gantry Mode

The dual drives gantry mechanism features the following:

1. Gear relation remains unless manually relieved by users.
2. Master axis stops when EMG / ALM / PEL / MEL / ALM signal of slave axis is set to ON.
3. Master and slave axes stop when stop (), emg_stop (), and servo_off () commands received by the slave axis.
4. Gear ration is fixed at 1:1 and cannot be changed.
5. Settings of engage rate are ignored.

In addition, this mode features a protection mechanism against two levels of mis-position errors. The controller checks position errors of both axes at every movement cycle for exceeding error setup. If error value is greater than position error settings of level 1, it starts the deceleration stop. If error value is greater than the

level 2 position error settings, the controller executes Servo-Off operation to both axes.

The setup value of this protection mechanism is to set axis parameter of slave axis by 1: master axis selected to follow and 2: two level of position error protection. Start up gantry mode with APS_start_gear (slave axis ID) after set up is completed. After the gantry mode is active, only the master axis need to be operated and the slave axis functions exactly the same as the master axis does.

Param. No.	Define symbol	Description
60h (96)	PRA_GEAR_MASTER	Gear master
63h (99)	PRA_GANTRY_PROTECT_1	Gantry mode protection level one
64h (100)	PRA_GANTRY_PROTECT_2	Gantry mode protection level two

4.10.2 High Speed Position Compare Trigger

This controller provides Compare trigger with structure as shown in figure below. Trigger is sent by TRG0 ~3. You can set up two trigger output formats. The one is pulse output and other is level toggle output. Length and logic of pulse signal can be adjusted by embedded PWM module. The PWM signal can be generated in two ways. The one is to generate trigger by manual trigger by calling API. The other is compare trigger that can be further divided into linear compare and table compare. Any one of these triggers is acceptable to PWM. Manual trigger and compare trigger are described in sections below.

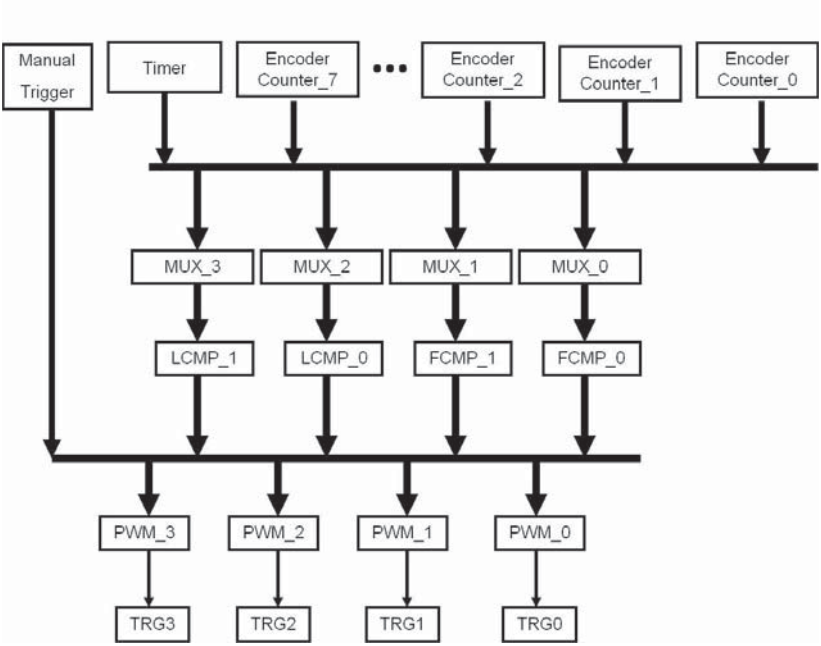


Figure 4-66: Compare trigger block diagram

TRG / PWM / Timer relevant parameter setup		
NO	**Define**	**Description**
0x06	TGR_TRG_EN	TRG0~3 output switch
0x10~0x13	TGR_TRGx_SRC	Set up TRG0~3 trigger source. You can have multiple sources to choose.
0x14~0x17	TGR_TRGx_PWD	Set up TRG0~3 pulse width
0x18~1B	TGR_TRGx_LOGIC	Set up TRG0~3 logic level
0x1C~1F	TGR_TRGx_TGL	Set up TRG0~3 output format
0x20	TIMR_ITV	Set up timer interval
0x21	TIMR_DIR	Set up timer counting direction
0x22	TIMR_RING_EN	Set up timer counter overflow activity
0x23	TIMR_EN	Start up timer

See APS Library operation manual for details on compare trigger relevant parameter list. Set up parameter APIs as described below

APS_set_trigger_param ();

APS_get_trigger_param ();

You may select either encoder counter or internal timer as the source of compare device. Relevant APIs are described below:

APS_get_timer_counter (); // read timer counter

APS_get_timer_counter (); // set up timer counter

4.10.2.1 Manual Trigger

Use APS_set_trigger_manual () API to output pulse signal. Please set up TRG with manual trigger source. See below for one operation example:

NO	Define	Description
0x10~0x13	TGR_TRGx_SRC	Set up TRG0~3 trigger source. You can have multiple sources to choose.

4.10.2.2 Compare Trigger

Compare trigger is a trigger generated when source value of comparator (CMP) matches with the value to be compared. There are two kinds of comparators, the one is the encoder counters (0~7) of each axis and the other is the timer. There are two compare methods too, linear compare trigger and table compare trigger. Operation rules and methods of both triggers are described below.

4.10.2.2.1 Linear Compare Trigger

You may need to define the subject to be compared, encoder counter or timer, before using the linear comparator. Then set up start point, repeat times and interval. The setup method, in terms of a position-time chart, is described below. Here P1 is the start point, repeat times is 4, interval is L, P1~P4 are four compare points separated by space I. When the motor move pass each compare points the TRG send pulse signals in sequence. The compare direction is determined by positive or negative value of

interval. Linear compare trigger can have compare speed up to 1MHz and integral times of 32 bit comparable points.

Use APIs below to set up start point, repeat times and interval of linear compare.

APS_set_trigger_linear ();

Parameters of linear compare trigger		
NO	**Define**	**Description**
0x00	TGR_LCMP0_SRC	Compare source of linear comparator LCMP0
0x01	TGR_LCMP1_SRC	Compare source of linear comparator LCMP1

- **Example:**

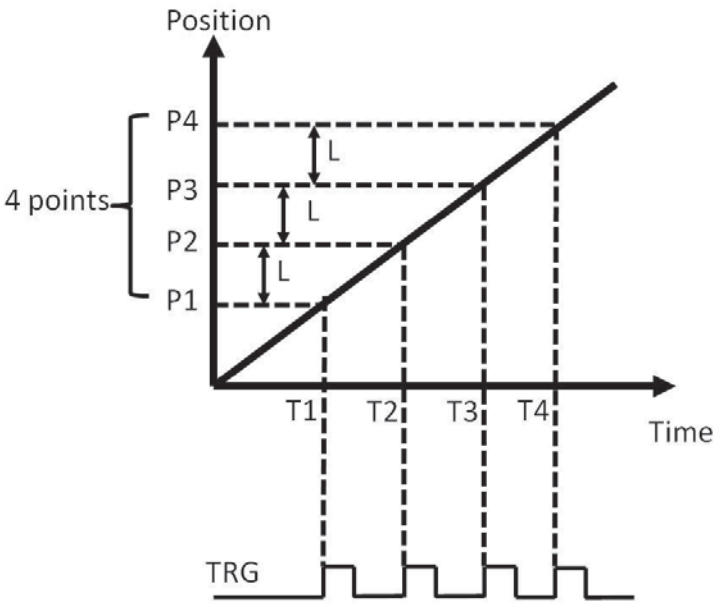


Figure 4-67: Linear compare trigger example

4.10.2.2.2 Table Compare Trigger

Table compare trigger differs from the linear compare trigger in that compare points can be determined by user. That is, intervals between compare points are variable. You may set up any four points (P1~P4) and send triggers when motor reaches each of them as shown in figure below.

Parameters of table compare trigger		
0x02	TGR_TCMP0_SRC	Set up compare subject of table comparator CH0
0x03	TGR_TCMP1_SRC	Set up compare subject of table comparator CH1
0x04	TGR_TCMP0_DIR	Set up compare direction of table comparator CH0
0x05	TGR_TCMP1_DIR	Set up compare direction of table comparator CH1

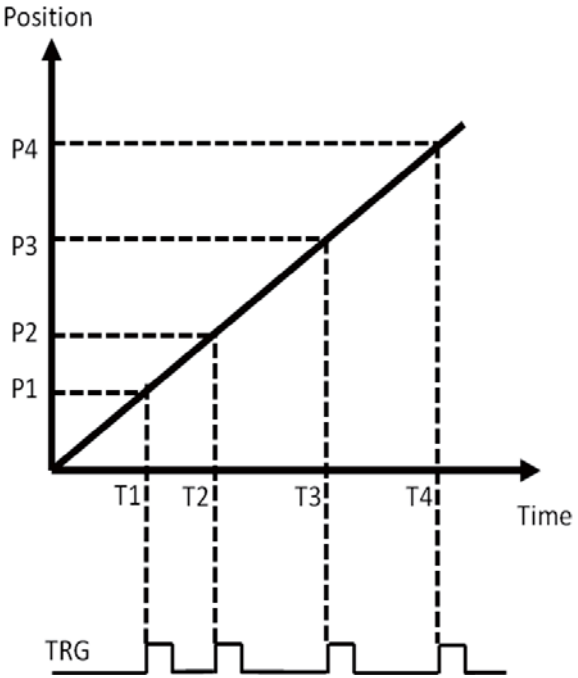


Figure 4-68: Table compare trigger example

There are two levels of FIFO buffer design contained in controller and hardware to accelerate compare speed. The hardware FIFO can have 255 records with compare speed up to 1 MHz. The controller contains 999 FIFO buffers and execute points filling in operation in every motion control cycle. You can input point array of any size in the APS function library (limited by system memory size). The APS function library shall load all compare points to the controller dynamically. No extra program coding is required for loading compare point dynamically in the controller even in case of many compare points.

APIs for loading compare table array:

APS_set_trigger_table ();

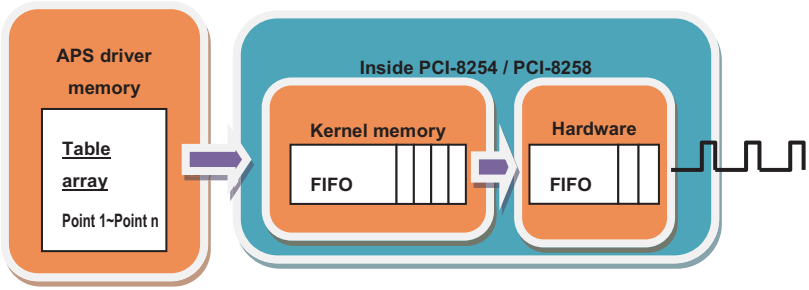


Figure 4-69: Table compare trigger block diagram

4.10.3 PWM Control (Laser Control) (VAO Table Control)

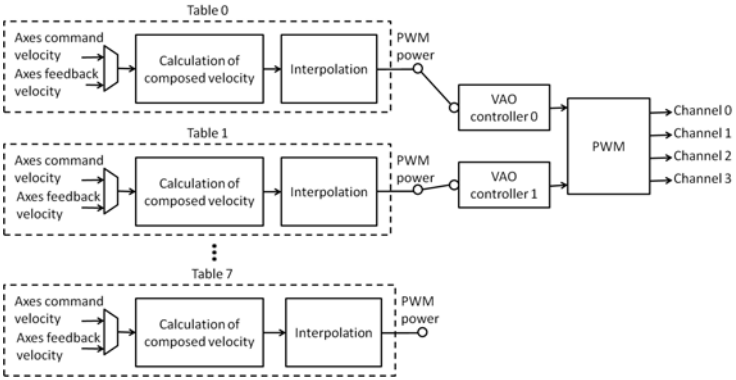
4.10.3.1 Structure Overview

Laser cutting is now commonly applied in various metal, non-metal, and composite material processing. The application is highly associated with motion control. To meet this requirements the VAO module is offered by your PCI-8254/8. The VAO module enables quality cutting by controlling laser intensity with speed information. Laser intensity is commonly controlled by pulse-width modulation (PWM). Your PCI-8254/8 features varieties of PWM control for specific applications.

See figure below for the operation of VAO module. The VAO controller monitors PWM output with VAO table at different speeds. Your PCI-8254/8 features two VAO controllers with output channels that you can set as required. This enables it to output the same or different PWMs in multiple channels concurrently. Each VAO controller may switch among different VAO table to meet the multi-level cutting requirements. Your PCI-8254/8 features eight VAO tables, Table 0~7, to come up with corresponding PWM settings, by interpolating composite speed among multiple axes, for laser output intensity control. Source of speeds may come from individual axes' command or feedback velocity where the noise may be removed by embedded filters. The refresh frequency of feedback speed is 1KHz now. The VAO module can work together with the Point table.

Please set up PWM control mode and VAO table before using the VAO module. Please set up function parameters with the VAO parameter table. See below for descriptions and setup procedures.

Structure of the VAO module



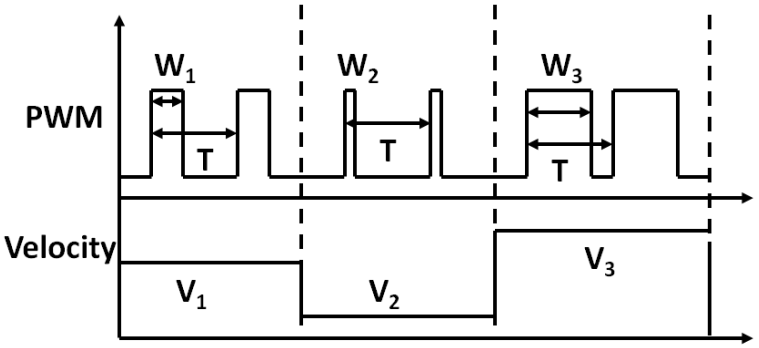
4.10.3.2 Control Modes

Your PCI-8254/8 VAO module now supports three control nodes:

a. Mode1: PWM mode

This control mode adjust PWM duty cycle according to fixed PWM frequency and variable speeds as shown in figure below. The fixed PWM frequency is 1/T and the PWM duty cycle W_1/T, W_2/T and W_3/T based on VAO table under speed V_1, V_2 and V_3 with different PWM pulse width at W_1, W_2 and W_3. See below for details on VAO table. To use this control mode to set

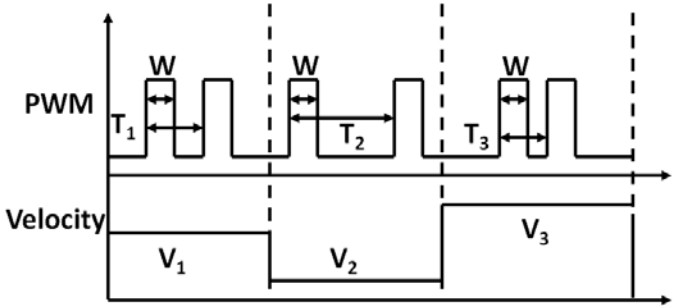
1. Set up control mode: Use APS_set_vao_param ( ) to set up value of VAO_TABLE_OUTPUT_TYPE parameter to 0x1.
2. Set up fixed PWM output frequency:
   Use APS_set_vao_param( ) to set up parameter VAO_TABLE_PWM_Config in unit of Hz. Valid frequency input range now is 3Hz ~ 50MHz.



b. Mode 2: PWM frequency mode with fixed width

This control mode changes PWM frequency according to speed at fixed PWM pulse width. Under fixed PWM pulse width W, the VAO table gives PWM frequency $1/T_1$, $1/T_2$ and $1/T_3$ at speed V_1, V_2 and V_3 as shown in figure below. To use this mode to

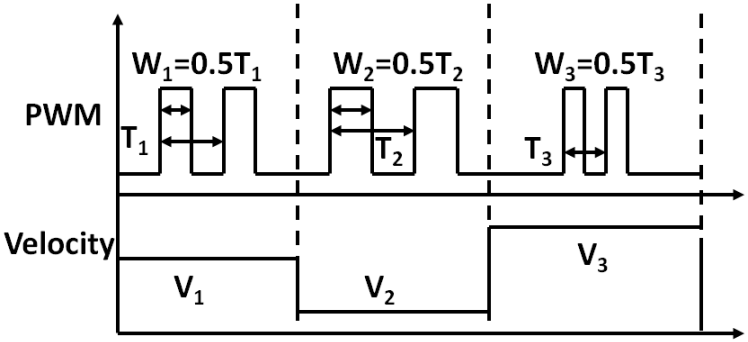
1. Set up control mode: Use APS_set_vao_param( ) to set up value of VAO_TABLE_OUTPUT_TYPE parameter to 0x2.
2. Set up fixed PWM output pulse width:
   Use APS_set_vao_param( ) to set up parameter VAO_TABLE_PWM_Config in unit of ns. Valid input range is 20ns ~ 335544320 ns。



c. Mode 3: PWM frequency mode with fixed duty cycle

This control mode changes PWM frequency according to speed at fixed PWM duty cycle. As shown in figure below, the duty cycle W_1/T_1, W_2/T_2 and W_3/T_3 are the same under varying speed while their frequency and pulse width changes according to the VAO table.

1. Set up control mode: Use APS_set_vao_param( ) to set up value of VAO_TABLE_OUTPUT_TYPE parameter to 0x3.
2. Set up fixed PWM duty cycle:
   Use APS_set_vao_param( ) to set up parameter VAO_TABLE_PWM_Config in unit of %. Valid range is 0.05% ~ 100% now.



4.10.3.3 VAO Table

The VAO table is designed to give speed-based PWM power for VAO controller to control actual PWM output signal. The VAO module features eight VAO tables for layered cutting. Each table may contain up to 32 speed-power pairs. See below for details on calculating speed-based power value. In figure below the X-axis is for composite speed of axes and the Y-axis its relevant PWM power. Please note the unit of power varies with control modes. Take example. If the control mode is set to mode 1: PWM mode the corresponding power is PWM duty cycle; if the control mode is set to mode 2: PWM frequency mode with fixed width then its power is PWM frequency. If the composite speed V_X is available, its corresponding power P_x can be interpolated as

$$P_X = (P_3 - P_2) * (V_X - V_2) / (V_3 - V_2) + P_2$$

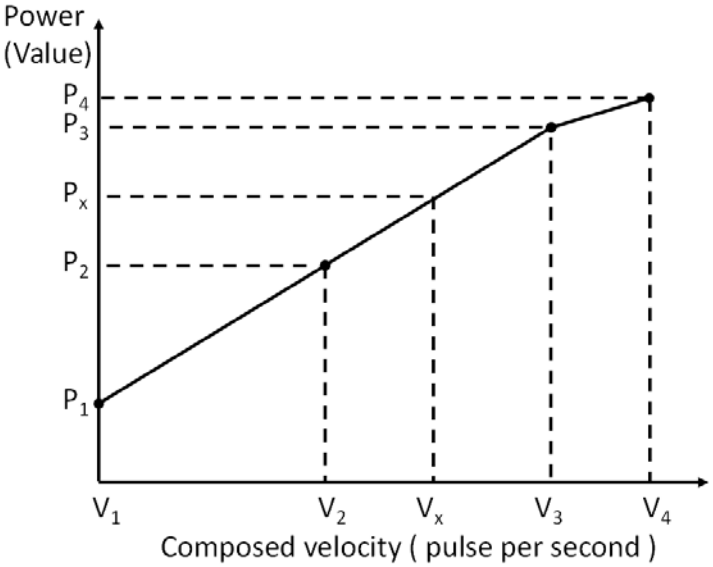


Table below suggests power range and resolution that can be set up by different control modes' VAO tables.

Mode	Power output range	Resolution
1. PWM mode	Duty cycle: 0~2000 (0.05%~100%)	0.05%
2. PWM frequency mode with fixed width	Frequency: 3Hz ~ 50MHz	1 Hz
3. PWM frequency mode with fixed duty cycle	Frequency: 3Hz ~ 50MHz	1 Hz

4.10.3.4 Output Settings

The VAO module now supports 4 PWM output channels for users' selection. You may set multiple channels to output the same control signals at the same time. Individual VAO modules are now opened with specific APS_start_vao() functions.

4.10.3.5 VAO Parameter Table

The VAO parameter table helps you in determining settings for control modes and VAO table. See table below on definitions of VAO parameters.

NO	Define	Description	Value	Default:
0x00 + (2 * N) Note: N is Table No, range is 0 ~ 7.	VAO_TABLE_OUTPUT_TYPE	Table output type	1: PWM mode 2: PWM frequency mode with fixed width 3. PWM frequency mode with fixed duty cycle	1
0x01 + (2 * N) Note: N is Table No, range is 0 ~ 7.	VAO_TABLE_INPUT _TYPE	Table input type	0: Feedback speed 1: Command speed	0
0x10 + N Note: N is Table No, range is 0 ~ 7.	VAO_TABLE_PWM_Config	Configure PWM according to output type	a. Mode 1 - set a fixed frequency (3~50M Hz) b. Mode 2 - set a fixed Pulse Width (20~335544300 ns) c. Mode 3 – set a fixed duty cycle: N * 0.05 %. (N: 1 ~ 2000)	100
0x20 + N Note: N is Table No, range is 0 ~ 7.	VAO_TABLE_SRC	Specify axisID for VAO table.	Bit0: Axis 0 On Bit1: Axis 1 On Bit2: Axis 2 On Bit3: Axis 3 On	0x01
0x30~	Reserved	Reserved	Reserved	Reserved

4.10.3.6 Digital Output and Relevant PWM Function

The VAO module features one special function to turn on or off the PWM with digital output control. It turns on or off the PWM signal output by working together with the point table's control options. Please use the board parameters to set up relations between PWM and digital outputs before using this function as shown in figure below. Here DO0~7 indicates digital output. Take example. If PWM 0 is set to DO2 and logic to 1, then PWM 0 starts output when DO2 changes from low to high and stops output vice versa. On the contrary, the PWM 0 starts output when DO2 changes from high to low when the logic is set to 0. Please note that if pairing relation is set before VAO module's PWM output opening, the PWM output may start or stop in accordance with existing digital

output and logic status. See description below for a use case outline.

1. Use APS_set_board_param() to set up PWM output channel and relevant digital output and judgment logic according to the board parameters.
2. Click Option indicated by point table to open DO_Enable, select DO_Channels and DO_ON or DO_OFF.

Pairing relation diagram

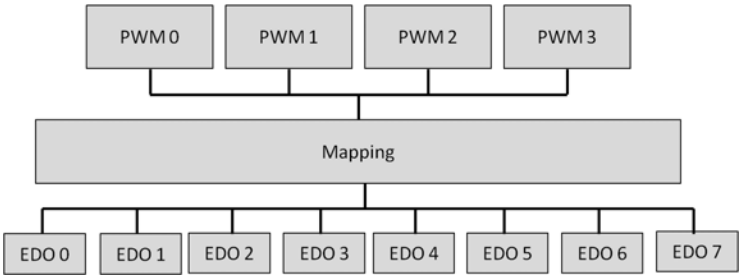


Table 4-4: Board parameter table

NO	SYMBOL	Description	Default
110h	PRB_PWM0_MAP_DO	(1) -1: Disable mapping ; > 0: Enable mapping (2) Bit0~7: Specify a Do channel. (3) Bit8: Select logic; Set to 1: Turning on Do maps enabling PWM0. Turning off Do maps disabling PWM0. Set to 0: Turning on Do maps disabling PWM0. Turning off Do maps enabling PWM0.	-1
111h	PRB_PWM1_MAP_DO	Please see description of PRB_PWM0_MAP_DO	-1
112h	PRB_PWM2_MAP_DO	Please see description of PRB_PWM0_MAP_DO	-1
113h	PRB_PWM3_MAP_DO	Please see description of PRB_PWM0_MAP_DO	-1

4.10.3.7 Operation Process Examples

Operation flow for various control modes are outlined below for your reference.

Mode	Description
1: PWM mode	**a. VAO parameter table - APS_set_vao_param ()** 0x00: set to 1 – PWM mode 0x01: set to 1 – command speed 0x10: set to 1000 – set fixed frequency to 1000 Hz 0x20: set to 3 – Axis0 and Axis1 are selected **b. "Velocity to Power" mapping lookup table - APS_set_vao_table ()** Duty cycle range: 0 ~ 2000 units (Be equal to 0 ~ 100 %) Points range: 1 ~ 32 points **c. Switch VAO table - APS_switch_vao_table ()** **d. Enable VAO output channel - APS_start_vao ()**
2: PWM frequency mode with fixed width	**a. VAO parameter table - APS_set_vao_param ()** 0x00: set to 2 – PWM frequency mode with fixed width 0x01: set to 1 – command speed 0x10: set to 1000 – set fixed pulse width to 1000 ns 0x20: set to 3 – Axis0 and Axis1 are selected **b. "Velocity to Power" mapping lookup table - APS_set_vao_table ()** Frequency range: 1 ~ 25Mhz for PCI-8253/6 Points range: 1 ~ 32 points **c. Switch VAO table - APS_switch_vao_table ()** **d. Enable VAO output channel - APS_start_vao ()**
3: PWM frequency mode with fixed duty cycle	**a. VAO parameter table - APS_set_vao_param ()** 0x00: set to 3 – PWM frequency mode with fixed duty cycle 0x01: set to 1 – command speed 0x10: set to 200 – set fixed duty cycle to 10%. (200 * 0.05%) 0x20: set to 3 – Axis0 and Axis1 are selected **b. "Velocity to Power" mapping lookup table - APS_set_vao_table ()** Frequency range: 1 ~ 25Mhz for PCI-8253/6 Points range: 1 ~ 32 points c. Switch VAO table - APS_switch_vao_table () d. Enable VAO output channel - APS_start_vao ()

4.10.4 Motion Control and I/O Sampling Function

4.10.4.1 Sampling Source

This control card supports multiple signal sampling for analysis. There are two signal sources: the one belongs to motion kernel signal and the other the close-loop control signal. In figure below, the bottom layer's motion kernel and controller's adjustable update rate is 1ms and 250us respectively, sampling rate of the sampling process is 1ms, and the sampled signals are sent by APS library to MotionCreatorPro2 or other applications to display. The close-loop control signal would be a better choice for learning the system's overall control performance. Please note that close-loop control signal sampling is invalid in pulse control mode. See table below for meanings of individual signals and Section 4 of MotionCreatorPro 2 User Manual for operation pages and steps.

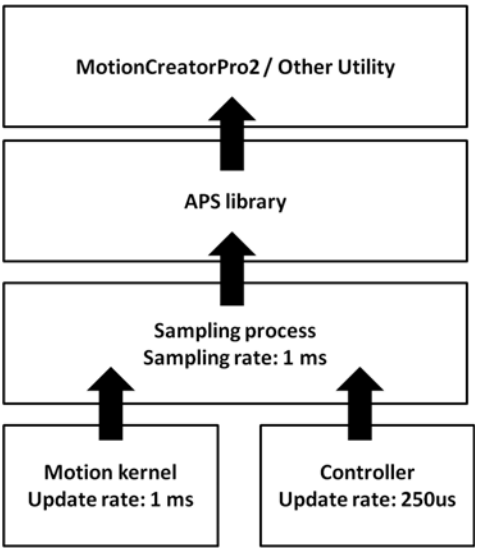


Figure 4-70: Signal sampling structure diagram

Table 4-5: Motion kernel signal table

Signal name	Range	Data type	Descriptions
SAMP_SRC_COM_POS	Axis 0~7	Integer	Position command: Unit: pulse
SAMP_SRC_FBK_POS	Axis 0~7	Integer	Feedback position Unit: pulse
SAMP_SRC_CMD_VEL	Axis 0~7	Integer	Command velocity; Unit: pulse/sec
SAMP_SRC_FBK_VEL	Axis 0~7	Integer	Feedback velocity; Unit: pulse/sec
SAMP_SRC_MIO	Axis 0~7	Integer	Motion I/O, see Note 1 for its definition.
SAMP_SRC_MSTS	Axis 0~7	Integer	Motion status, see Note 2 for its definition.
SAMP_SRC_MSTS_ACC	Axis 0~7	Integer	Motion status: Acceleration section (motion status acc); 0: Null acceleration 1: Current acceleration
SAMP_SRC_MSTS_MV	Axis 0~7	Integer	Motion status: Constant speed section (motion status at max velocity); 0: No constant speed 1: Current constant speed
SAMP_SRC_MSTS_DEC	Axis 0~7	Integer	Motion status: Deceleration section (motion status DEC); 0: Null deceleration 1: Current deceleration
SAMP_SRC_MSTS_CSTP	Axis 0~7	Integer	Motion status: The Stop motion command (motion status CSTP); 0: During movement 1: Stop the movement command
SAMP_SRC_MSTS_MDN	Axis 0~7	Integer	Motion status: Movement completed (motion status MDN); 0: During movement 1: Movement completed
SAMP_SRC_MIO_INP	Axis 0~7	Integer	Motion status: Movement in-place (motion status MDN); 0: Movement not in-place 1: Movement in-place
SAMP_SRC_MIO_ORG	Axis 0~7	Integer	Motion status: ORG signal (motion status OGR); 0: No ORG signal 1: Touches ORG signal
SAMP_SRC_CONTROL_VOL	Axis 0~7	Integer	Output voltage (Control command voltage); Unit: mV
SAMP_GTY_DEVIATION	Axis 0~7	Integer	Set up feedback offset (gantry deviation) between given and slave axes in gantry movement; Unit: pulse
SAMP_SRC_ENCODER_RAW	Axis 0~7	Integer	Drive's feedback position original signal (Encoder raw data) ; Unit: pulse
SAMP_SRC_ERR_POS	Axis 0~7	Integer	Error position; Unit: pulse
SAMP_SRC_COM_POS_F64	Axis 0~7	Double	Same as SAMP_SRC_COM_POS but presented in float point numbers

Signal name	Range	Data type	Descriptions
SAMP_SRC_FBK_POS_F64	Axis 0~7	Double	Same as SAMP_SRC_FBK_POS but presented in float point numbers
SAMP_SRC_CMD_VEL_F64	Axis 0~7	Double	Same as SAMP_SRC_CMD_VEL but presented in float point numbers
SAMP_SRC_FBK_VEL_F64	Axis 0~7	Double	Same as SAMP_SRC_FBK_VEL but presented in float point numbers
SAMP_SRC_CONTROL_VOL_F64	Axis 0~7	Double	Same as SAMP_SRC_CONTROL_VOL but presented in float point numbers
SAMP_SRC_ERR_POS_F64	Axis 0~7	Double	Same as SAMP_SRC_FBK_POS but presented in float point numbers
SAMP_PWM_FREQUENCY_F64	Channel 0~3	Double	PWM frequency; Unit: Hz
SAMP_PWM_DUTY_CYCLE_F64	Channel 0~3	Double	PWM duty cycle; Unit: %
SAMP_PWM_WIDTH_F64	Channel 0~3	Double	PWM width; Unit : ns
SAMP_VAO_COMP_VEL_F64	No. 0~1	Double	Composed velocity for Laser power control; Unit: pulse/sec
SAMP_PTBUFF_COMP_VEL_F64	Table 0~1	Double	Composed velocity of point table; Unit: pulse/sec
SAMP_PTBUFF_COMP_ACC_F64	Table 0~1	Double	Composed acceleration of point table); Unit: pulse/sec^2

Table 4-6: Closed circuit control signal table

Signal name	Range	Data type	Descriptions
PID_CMD_POS_COUNT	Axis 0~7	Integer	Position command; Unit: pulse
PID_CMD_ACC_COUNT	Axis 0~7	Integer	Acceleration command; Unit: pulse
PID_CMD_VEL_COUNT	Axis 0~7	Integer	Velocity command; Unit: pulse
PID_ERR_POS_COUNT	Axis 0~7	Integer	Position error; Unit: pulse
PID_KP_CALC_COUNT	Axis 0~7	Integer	Proportional control output; Unit: pulse
PID_KI_CALC_COUNT	Axis 0~7	Integer	Integration control output; Unit: pulse
PID_KD_CALC_COUNT	Axis 0~7	Integer	Derivative control output; Unit: pulse
PID_KAFF_CALC_COUNT	Axis 0~7	Integer	Acceleration feedforward control output; Unit: pulse
PID_KVFF_CALC_COUNT	Axis 0~7	Integer	Velocity feedforward control output; Unit: pulse

Signal name	Range	Data type	Descriptions
PID_CALC_COUNT	Axis 0~7	Integer	PID and feedforward control combined output; Unit: pulse
BIQUAD_0_CALC_COUNT	Axis 0~7	Integer	Biquad Filter 0 output; Unit: pulse
BIQUAD_1_CALC_COUNT	Axis 0~7	Integer	Biquad Filter 1 output; Unit: pulse
PID_ERR_POS_SUM_COUNT	Axis 0~7	Integer	Error position accumulation of integral control
PID_LAST_ERR_POS_COUNT	Axis 0~7	Integer	Error position of derivative control at last moment; Unit: pulse
PID_ERR_POS_DIFF_COUNT	Axis 0~7	Integer	Error position differential of derivative control; Unit: pulse

Note 1: Motion I/O definition table

7	6	5	4	3	2	1	0
SVON	INP	EZ	EMG	ORG	MEL	PEL	ALM
15	14	13	12	11	10	9	8
			SMEL	SPEL			

Bit number detail description:

Bit	Define	Description
0	ALM	Servo alarm input status
1	PEL	Positive end limit
2	MEL	Minus end limit
3	ORG	Original input (Home input)
4	EMG	Emergency stop input
5	EZ	Servo index input
6	INP	In-Position input
7	SVON	Servo ON output status
…		
11	SPEL	1: Soft-positive-end limit condition match.
12	SMEL	1: Soft-minus-end limit condition match

Note 2: Motion status definition table

7	6	5	4	3	2	1	0
	HMV	MDN	DIR	DEC	ACC	VM	CSTP
15	14	13	12	11	10	9	8
JOG				PTB	WAIT		
23	22	21	20	19	18	17	16
				POSTD	PRED	BLD	ASTP
31	30	29	28	27	26	25	24
			GER				

Bit number detail description:

Bit	Define	Description
0	CSTP	Command stopped (But it could be in motion)
1	VM	In maximum velocity
2	ACC:	In acceleration
3	DEC:	In deceleration
4	DIR:	Move direction. 1:Positive direction, 0:Negative direction
5	MDN	Motion done. 0: In motion, 1: Motion done (It could be abnormal stop)
6	HMV	In homing
…		
10	WAIT	Axis is in waiting state. (Wait move trigger)
11	PTB	Axis is in point buffer moving. (When this bit on, MDN and ASTP will be cleared)
…		
15	JOG	In jogging
16	ASTP	0: Stop normally, 1: abnormal stop, When axis in motion, this bit will be clear.
17	BLD	Axis (Axes) in blending moving
18	PRED	Pre-distance event, 1: event arrived. The event will be clear when axis start moving
19	POSTD	Post-distance event. 1: event arrived. The event will be clear when axis start moving
…		
28	GER	1: In geared (This axis as slave axis and it follow a master specified in axis parameter.)
29	--	--

4.10.5 Simultaneous Move

4.10.5.1 Simultaneous Start

Synchronized (Simultaneous) start: This movement can set to be enabled by trigger. When proper command is received, the axis enters a waiting-for-trigger-signal status and starts moving after the trigger is received. When multiple axes are in waiting-for-trigger-signal status you may send trigger signal at the same time for synchronized enabling. Please note that movement of each axis is independent from each other and so the end time varies with offset amount and acceleration profile.

Please enables simultaneous start by steps below:

a Set axis movement to triggered startup and check the axis status for waiting for trigger

b Send the trigger to run synchronized start

a Set axis movement to triggered startup and check the axis status for waiting for trigger

You may set up the startup-by-trigger mode by the Option parameter of the controller's function. The axis enters trigger waiting status once the command is received.

Take APS_ptp, the function prototype may look like

I32 APS_ptp(I32 Axis_ID, I32 Option, …);

See table below for definitions of Option. Please note that given axis is set to startup-by-trigger mode when Bit 8 is given value 1.

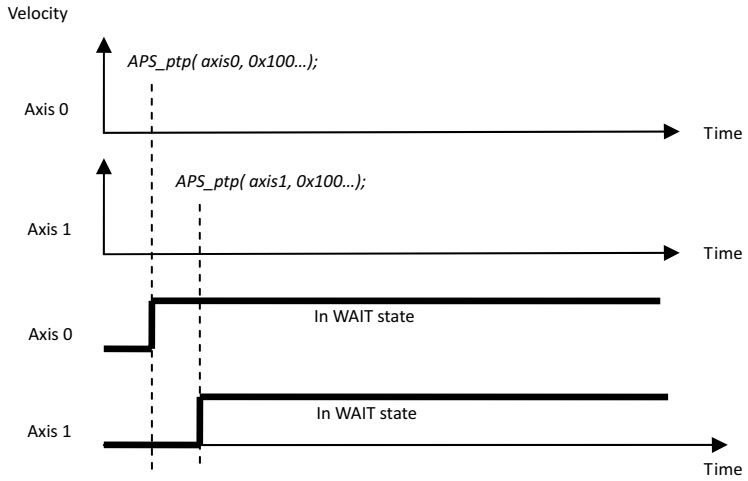
7	6	5	4	3	2	1	Bit : 0
							Absolute(0) / Relative(1)
15	14	13	12	11	10	9	8
Buffer mode							Wait trigger

After an axis movement is set to startup-by-trigger mode it enters the trigger waiting status, i.e. the WAIT signal of Bit 10 in table below is ON. You may display its signal status with function library, the motion status monitoring function is

I32 APS_motion_status ();

Motion status definition table							
7	6	5	4	3	2	1	Bit : 0
--	HMV	MDN	DIR	DEC	ACC	VM	CSTP
15	14	13	12	11	10	9	8
JOG	--	--	--	PTB	WAIT	--	--

See figure below for an illustration of motion set to trigger waiting



Relevant APS API described below

I32 APS_ptp ();I32 APS_ptp_v ();I32 APS_ptp_all ();I32 APS_line ();I32 APS_line_v ();

I32APS_line_all ();I32 APS_vel ();I32 APS_vel_all ();I32 APS_arc2_ca ();I32 APS_arc2_ca_v ();

I32 APS_arc2_ca_all ();I32 APS_arc2_ce ();I32 APS_arc2_ce_v ();I32 APS_arc2_ce_all ();

I32 APS_arc3_ca ();I32 APS_arc3_ca_v ();I32 APS_arc3_ca_all ();I32 APS_arc3_ce ();

I32 APS_arc3_ce_v ();I32 APS_arc3_ce_all ();I32 APS_arc3_ca ();I32 APS_arc3_ca_v ();

I32 APS_arc3_ca_all ();I32 APS_sprial_ca ();I32 APS_sprial_ca_v ();I32 APS_sprial_ca_all ();

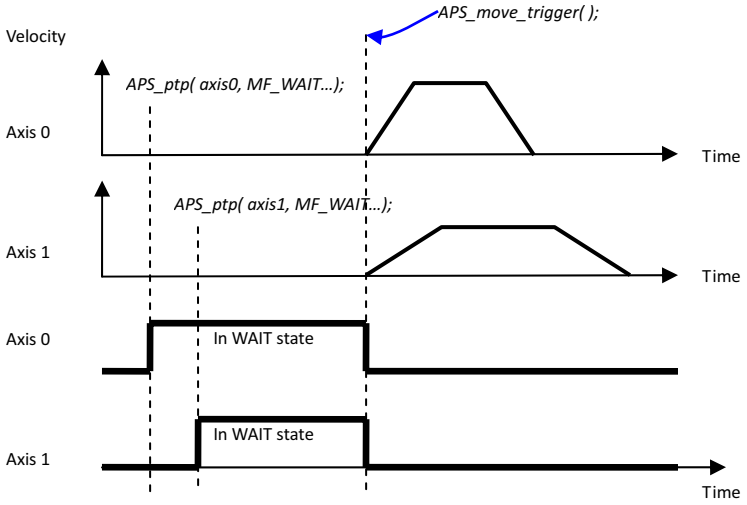
I32 APS_sprial_ce ();I32 APS_sprial_ce_v ();I32 APS_sprial_ce_all ();

b. Send the trigger signal to run synchronized start

You may enable multiple axes at the same time by sending trigger signal with function in form of:

I32 APS_move_trigger ();

See figure below for multiple axes' concurrent startup by trigger:



Relevant APS API described below:

I32 APS_move_trigger (); // trigger issued

I32 APS_stop_move (); // synchronized deceleration stopped

I32* APS_stop_move *(); // synchronized Emg stopped

- **Example:**

```
#include "APS168.h"
#include "APS_define.h"
#include "ErrorCodeDef.h"

void simultaneous_move_example()
{
   //This example shows how to execute a simultaneous move
   I32 option = 0x100; //bit 8 = 1
   I32 return_code = 0;
   I32 dimension = 2;
   I32 axis_array[2] = { 0, 1 };

   return_code = APS_ptp( 0, option, 10000, 0 ); //axis 0  設定為觸發啟動模式
   return_code = APS_ptp( 1, option, 20000, 0 ); //axis 1  設定為觸發啟動模式
   return_code = APS_move_trigger( dimension, axis_array ); //發出觸發訊號讓axis 0 & 1同時啟動
}
```

4.10.6 Point Table Movement

The controller features two point table which contains 50 buffer points respectively. You may enjoy point table functions of large amount of points and free from any practical limits by monitoring the usages status of buffer point space and reloading these 50 buffer point space repetitively.

With the controller's point table movement function you may get continuous movement in multi-sections with relevant function. Available commands in point table movements are straight line, arc, and spiral interpolation and dwell. The instruction command covers digital output and VAO table switch which can be used to program application relevant requirements.

4.10.6.1 Point Table Parameter Setup

There are three groups of point table parameters:

a Movement parameter setup

b Instruction command setup

c Set movement command to point table

a. Movement parameter setup

Please set up movement parameters before executing movement commands including absolute and relative movement, maximum, ending velocity, acceleration and deceleration, S-factor and speed blending method between adjacent paths, for speed and path planning applicable to applications. Please note that these parameter settings are kept by the program memory once being set up. Existing settings may be applied to other movement commands automatically. Repetitive setups are not required for each movement command unless you want to change parameter settings.

Movement parameter setup	Paired APS function
Absolute / relative movement	APS_pt_set_absolute / APS_pt_set_relative
Maximum speed	APS_pt_set_vm
Ending speed	APS_pt_set_ve
Acceleration	APS_pt_set_acc
Deceleration	APS_pt_set_dec
Acceleration and deceleration	APS_pt_set_acc_dec
S-factor	APS_pt_set_s
Speed blending between adjacent paths Please refer to Section 4.11.3.	APS_pt_set_trans_buffered (**buffer)** APS_pt_set_trans_inp (**buffered in-place**) APS_pt_set_trans_blend_dec (**blend - deceleration**) APS_pt_set_trans_blend_dist (**blend - residue-distance**) APS_pt_set_trans_blend_pcnt (**blend - residue-distance ratio**)

b. Instruction command setup

The instruction command is executed simultaneously with the point table's movement one. That is, it may control digital outputs concurrently at different movement section during motion execution.

Instruction command (executed along with the movement command)	Paired APS function
Digital output (DO)	APS_pt_ext_set_do_ch
VAO table switch	APS_pt_ext_set_table_no

c. Set movement command to point table

The point table offers movement commands of straight line, arc and spiral interpolation which can set movement commands in point table with the paired APS function.

Movement commands	Paired APS function
Straight line interpolation	APS_pt_line
Arc interpolation	APS_pt_arc2_ca / APS_pt_arc2_ce APS_pt_arc2_ca / APS_pt_arc2_ce
Spiral interpolation	APS_pt_sprial_ca / APS_pt_sprial_ce
Dwell	APS_pt_dwell

You may set up relevant movement parameter and required synchronous instruction command as well as save them in the point table with these three steps. Follow the same steps to save all graphic sections in point table.

4.10.6.2 Execute Point Table Movement

The controller features two point table which contains 50 buffer points respectively. You may enjoy point table functions of large amount of points and free from any practical limits by monitoring the usages status of buffer point space and reloading these 50 buffer point space repetitively.

A point table movement can be executed in following steps:

a Enable/disable point table function

b Monitor buffer space and fill in the points

c Start /stop point table movement

a. Enable/disable point table function

Please enable the point table function, set up its ID (0~1), movement dimension and axis number before using it. Please disable it after the point table function is ended.

I32 APS_pt_enable (I32 Board_ID, I32 PtbId, I32 Dimension, I32 *AxisArr);

Point table functions	Paired APS function
Enable the point table function	APS_pt_enable
Disable the point table function	APS_pt_disable

b. Monitor buffer space and fill in the points

A point table features 50 buffer points. You may monitor these buffer points and fill in the table with movement commands (see Section 4.11.1 for detail) by loading in all the graphic points dynamically.

Point table functions	Paired APS function
Monitor buffer status	APS_get_pt_status

c. Start /stop point table movement

After enabling the point table and fill in the buffers with movement commands you can then start up the point table function. The motion kernel program starts executing movement commands contained in buffer points in sequence until interrupted or each buffer point has been executed.

Point table functions	Paired APS function
Enable the point table movement	APS_pt_start
End the point table movement	APS_pt_stop

- **Example:**

```
#include "APS168.h"
#include "APS_define.h"
#include "ErrorCodeDef.h"

void pt_move_example ()

  //This example shows how pt move operation
  I32 ret;
    I32 Board_ID = 0;
    I32 PtbId = 0; //Point table 0
    I32 Dimension = 2; //2D Dimension
    I32 AxisArr[2] = { 0, 1 }; //Set Axis 0 & Axis 1 to point table 0
    PTLINE Prof;
    PTSTS Status;

    //Enable point table id 0 for 2D dimension with aixs 0 and axis 1.
    APS_pt_enable (Board_ID , PtbId, Dimension, & AxisArr); //Enable point table id 0


    //Get status of point table id 0 to monitor buffer
    APS_get_pt_status (Board_ID , PtbId, &Status);
    if ( !(Status.BitSts & 0x02 ) ) //Point buffer is not full

       //Push move into point buffer
       Prof.Dim = 2;
       Prof.Pos[0] = 10000;
       Prof.Pos[1] = 10000;
       ret = APS_pt_line (Board_ID, PtbId, &Prof, &Status);


    //Start point table move
    APS_pt_start (Board_ID, PtbId, 0);
```

4.11 Safety Protection

During equipment operation, there maybe errors or situations where emergency stops are required. In case of this, the usual method is to stop the mechanical equipment from operation. This controller provides some safety mechanism to detect predefined error situations. When these conditions are encountered, the controller take proper actions to protect personnel safety and to prevent damage to equipments. Some of these safety mechanism require external hardware signal while others do regular checks with software. These safety mechanism are described below.

4.11.1 Hardware Protection

The controller provides external hardware signal based detective protection mechanism including emergency stop (EMG), servo alarm (ALM) and mechanical plus and minus limit (PEL and MEL). Detailed operation theories are described below.

4.11.1.1 Emergency Stop (EMG)

See table below for EMG hardware input pins:

P1A Pin No	Signal Name
51	IEMG

EMG signal is a hardware input signal. When EMG signal is set to ON status the controller responses with following actions:

1. If the EMG signal is set to ON when the axis is in motion status, the controller stops the axis movement immediately. Error stop code of the axis is set to "1" (STOP_EMG) and motion status of axis is set to abnormal stop (ASTP).
2. If the axis is not in motion status and the EMG signal is ON, user's motion command shall be ignored by the controller while error stop code of the axis is set to STOP_EMG (1) and motion status of axis is set to abnormal stop (ASTP).

Relevant APIs:

APS_motion_status (); // read in motion status (ASTP)

APS_get_stop_code (); // read in error stop code

4.11.1.2 Servo Alarm (ALM)

See table below for ALM hardware input pins and corresponding axis number:

P1A Pin No	Signal Name	Axis #	P1B Pin No	Signal Name	Axis #
35	ALM1	0	35	ALM5	4
41	ALM2	1	41	ALM6	5
85	ALM3	2	85	ALM7	6
91	ALM4	3	91	ALM8	7

ALM signal is a hardware input signal where ALM signal is from servo drive to controller through ALM pin. When ALM signal is set to ON status, the controller responses with following actions:

1. If ALM signal is asserted for an axis in motion status, the controller stops motion of the axis immediately and error stop code of the axis is set to "2" (STOP_ALM) and motion status of axis is set to abnormal stop (ASTP = ON).
2. If the axis is not in motion status and the ALM signal is asserted then user's motion command shall be ignored by the controller while error stop code of the axis is set to "2" (STOP_ALM) and motion status of axis is set to abnormal stop (ASTP = ON).

4.11.1.3 Plus and Minus Limit Signal (PEL/MEL)

See table below for EL hardware input pins and corresponding axis number:

P1A Pin No	Signal Name	Axis #	P1B Pin No	Signal Name	Axis #
38	PEL1	0	40	MEL1	0
44	PEL2	1	46	MEL2	1
88	PEL3	2	90	MEL3	2
94	PEL4	3	96	MEL4	3
38	PEL5	4	40	MEL5	4
44	PEL6	5	46	MEL6	5
88	PEL7	6	90	MEL7	6
94	PEL8	7	96	MEL8	7

EL signal is a hardware input signal including PEL and MEL. PEL is the limit signal in positive direction and MEL the negative direction one. An asserted EL signal causes the controller responses with following actions:

1. If PEL signal is asserted for an axis in positive motion status, the controller stops motion of the axis immediately and error stop code of the axis is set to "4" (STOP_PEL) and motion status of axis is set to abnormal stop (ASTP = ON).
2. If MEL signal is asserted for an axis in negative motion status, the controller stops motion of the axis immediately and error stop code of the axis is set to "5" (STOP_MEL) and motion status of axis is set to abnormal stop (ASTP = ON).
3. If the axis is not in motion status and the PEL signal is asserted then user's positive direction motion command shall be ignored by the controller while error stop code of the axis is set to "4" (STOP_PEL) and motion status of axis is set to abnormal stop (ASTP = ON).
4. If the axis is not in motion status and the MEL signal is asserted then user's negative direction motion command shall be ignored by the controller while error stop code of the axis is set to "5" (STOP_MEL) and motion status of axis is set to abnormal stop (ASTP = ON).
5. There are two stop mode options available: decelerating stop and immediate stop. The axis parameter code is PRA_EL_MODE (0x02).

4.11.2 Software Protection

The controller provides software protection mechanism of software limit and position error protection.

4.11.2.1 Soft-limit Signal

Software limit functions almost the same as that of the hardware limit with the exception that limit signal is generated by checking location of each axis with the software limit function. There are the same plus limit (PEL) and minus limit (MEL) signals. Steps of using the software limit are described below:

1. Set up position of software limit with axis parameters PRA_SPEL_POS and PRA_SMEL_POS shown in table below.
2. Set up stop mode in response to limit signal. You can select decelerating stop or immediate stop with axis parameter PRA_EL_MODE (0x02) and PRA_SD_DEC (0x07).
3. Start up software limit function with axis parameter PRA_SPEL_EN (0x08) and PRA_SMEL_EN (0x09) shown in table below.

Please run home operation to ensure limit position of the coordinate system before the software limit function can be started.

NO	Define	Description
02h	PRA_EL_MODE	EL signal stop mode See deceleration rate reference parameter, PRA_SD_DEC, for deceleration stop mode:
07h	PRA_SD_DEC	Set up rate for deceleration stop
08h	PRA_SPEL_EN	Soft PEL switch
09h	PRA_SMEL_EN	Soft MEL switch
0Ah	PRA_SPEL_POS	Soft PEL position
0Bh	PRA_SMEL_POS	Soft MEL position

After the software limit function is initiated, you may use the function library provided by the controller to display signal status. This IO monitoring function is described below:

APS_motion_io_status ();

When software limit signal is set to ON status the controller responses with following actions:

1. If SPEL signal is asserted for an axis in positive direction motion status, the controller stops motion of the axis immediately and error stop code of the axis is set to "6" (STOP_SPEL) and motion status of axis is set to abnormal stop (ASTP).
2. If SMEL signal is asserted for an axis in negative direction motion status, the controller stops motion of the axis immediately and error stop code of the axis is set to "7" (STOP_SMEL) and motion status of axis is set to abnormal stop (ASTP).
3. If the axis is not in motion status and the SPEL signal is asserted then user's positive direction motion command shall be ignored by the controller while error stop code of the axis is set to STOP_SPEL (6) and motion status of axis is set to abnormal stop (ASTP).
4. If the axis is not in motion status and the SMEL signal is asserted then user's negative direction motion command shall be ignored by the controller while error stop code of the axis is set to STOP_SPEL (7) and motion status of axis is set to abnormal stop (ASTP).

4.11.2.2 Position Error Protection

Position error protection is a software protection mechanism by monitoring difference between command counter and feedback counter. This difference is defined as position error. When position error is too big, the controller sends a Servo off signal which can be set up for use before servo fine tuning.

This function can be set up with axis parameters as shown in table below. You can disable the position error protection by setting position error parameter (PRA_ERR_POS_LEVEL) to value "0". The position error protection is enabled when position error parameter is set to non-zero value.

NO	Define	Description
124h	PRA_ERR_POS_LEVEL	Position error protection setup

Position error protection causes the controller responses with following actions:

If the position error value is greater than setting given by users, the controller run Servo off command directly and error Stop code of the axis is set to STOP_ERROR_LEVEL (6) and motion status of axis is set to abnormal stop (ASTP).

4.11.2.3 Watch Dog

The watchdog protection mechanism is a timer inside the controller. Timeout of the timer will enable predefined response actions including Servo off, turning off digital output and turning off PWM output. After the watchdog protection mechanism is enabled, the user program must be in responsible status. Before timeout of the timer, the watchdog mechanism should be reset continuously to restart timing of the timer. As long as the user program remain in responsible status relevant events shall not be triggered. In another words, the watchdog protection mechanism is used to detect stagnation (failure) status of the upper control program. If stagnation situation is encountered, the controller triggers a protection mechanism to close signal output.

You may use interrupt function in Windows environment as described below:

1. Set up trigger event of timer timeout
2. Enable watchdog protection mechanism
3. Reset timer within cycle time

See below for relevant APS APIs:

APS_wdt_set_action_event ();

APS_wdt_set_action_event ();

APS_wdt_start ();

APS_wdt_get_timeout_period ();

APS_wdt_reset_counter ();

Detailed operation methods are described below:

1. Set up trigger event of timer timeout:
   Use ***APS_wdt_set_action_event ()*** function to set up trigger event.
   Use ***APS_wdt_get_action_event ()*** function to get trigger event.
2. Enable watchdog protection mechanism:
   Use ***APS_wdt_start ()*** to set up a timeout period and enable the watchdog function then the internal timer starts clicking. Set timeout period to "0" to disable watchdog function.
   Use ***APS_wdt_get_timeout_period ()*** to read in the timeout settings.

3. Reset timer continuously

   After the watchdog protection mechanism is enabled, the watchdog mechanism should be reset within timeout period to rest the timer and retiming from beginning. In case of timer timeout relevant events are triggered per setting given by step 1.

   Use ***APS_wdt_reset_counter ()*** to reset watchdog.

- **Example:**

```
void watchdog_example()
{
  // This example shows how interrupt functions work.
  I32 board_id = 0;
  I32 timer_no = 0;   // Only timer 0 to be used
  I32 time_out = 10;  // Time out is 10*100 ms = 1 sec
  I32 EventByBit = 0x01; // Action event is defined by bit.
                  // Bit0: Motor servo off
                  // Bit1: Digital output off
                  // Bit2: PWM off
  I32 ret = 0; // return code
  //Step 1: 設定計時器觸發後的事件
  ret = APS_wdt_set_action_event( board_id, timer_no, EventByBit );
  //Step 2: 啟動看門狗保護機制
  ret = APS_wdt_start( board_id, timer_no, time_out );
   //使用者的timer，while loop，或者是可回應點
  timer(500ms)//每500ms去重置一次計時器
   {
    //Step 3: 不間斷的重置計時器
   ret = APS_wdt_reset_counter( board_id, timer_no );
   …Do Something
   }
}
```

4.12 Host Interrupt

An interrupt is a process starting when specified event is encountered, the device (this controller) issue hardware interrupt signal to the operating system, the operating system enable the driver to execute corresponding interrupt service routine. See figure below for illustration to this flow.

Either interrupt or polling mechanism is employed to detect the certain event. The polling mechanism consumes CPU time repetitively and lead to CPU over utilization. The interrupt mechanism alert the CPU of event after it is encountered. This process consumes much less CPU time and so can reduce CPU utilization rate. It also frees up the CPU to process other tasks for multi tasks and effective CPU resource utilization when waiting for interrupt signal.

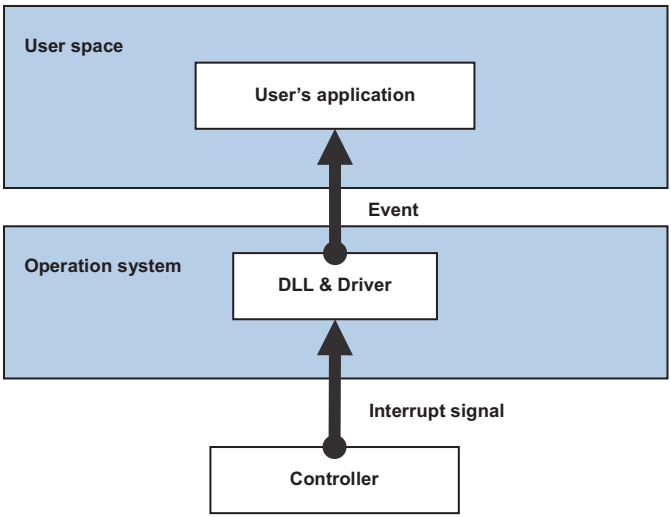


Figure 4-71: Interruption flow chart

Types of interrupt events provided by this controller are described below:

1. Axis interrupt
2. System interrupt
3. Digital input interrupt

Axis interrupt type contains all control axis relevant events. The digital input interrupt contains rising edge interrupt and falling edge interrupt. And other events are contained in system interrupt type.

See table below for all interrupt event types contained in this controller. Here items 0~7 are interrupt relevant to each control axis, item 8 is system relevant interrupt and item 9 and 10 are digital input interrupt. (Note: For PCI-8254 items 0~3 and 4~7 are reserved.)

- **Interrupt Item overview:I**

Item	Item type description
0-7	Axis 0~7 interrupt (Item 4 ~ 7 of PCI-8254 are reserved)
8	System interrupt
9	Digital input rising edge interrupt
10	Digital input falling edge interrupt

Each item can have up to 32 kinds of interrupt events (32 bit). See tables below for detailed definitions.

- **Item = 0~7: Axis interrupt events overview**

Bit No.	7	6	5	4	3	2	1	0
Factor	--	IEMG	IINP	IEZ	IORG	IMEL	IPEL	IALM
Bit No.	15	14	13	12	11	10	9	8
Factor	ISPEL	--	IASTP	IMDN	IDEC	IACC	IVM	ICSTP
Bit No.	23	22	21	20	19	18	17	16
Factor	--	--	--	--	IPOSTD	IPRED	--	ISMEL
Bit No.	31	30	29	28	27	26	25	24
Factor	--	--	--	--	--	--	--	--

- **Axis interrupt events description:**

bit.	Symbol	Interrupt event description
0	IALM	ALM signal occurrence
1	IPEL	PEL signal occurrence
2	IMEL	MEL signal occurrence
3	IORG	ORG signal occurrence
4	IEZ	Motor Z phase signal (EZ) occurrence
5	IINP	Drive in-place (INP) signal occurrence
6	IEMG	Emergency stop signal EMG occurrence (same as system IEMG)
7	--	Reserved, set to "0"
8	ICSTP	CSTP signal occurrence
9	IVM	Maximum velocity
10	IACC	Start acceleration
11	IDEC	Start deceleration
12	IMDN	Motion done
13	IASTP	Abnormal stop
14	--	Reserved, set to "0"
15	ISPEL	Soft PEL occurrence
16	ISMEL	Soft MEL occurrence
17	--	Reserved, set to "0"
18	IPRED	Pre-distance event occurrence
19	IPOSTD	Post-distance event occurrence
20~	--	Reserved, set to "0"

- **Item = 8: System interruption events overview**

Bit No.	7	6	5	4	3	2	1	0
Factor	--	IHOV	IMOV	IFCF1	IFCF0	ILCF1	ILCF0	IEMG
Bit No.	15	14	13	12	11	10	9	8
Factor	--	--	--	--	--	--	--	--
Bit No.	23	22	21	20	19	18	17	16
Factor	--	--	--	--	--	--	--	--
Bit No.	31	30	29	28	27	26	25	24
Factor	--	--	--	--	--	--	--	--

- **System interrupt events description**

bit.	Symbol	Interrupt event description
0	IEMG	Emergency stop signal (EMG) signal occurrence
1	ILCF0	Linear comparator 0 comparing end
2	ILCF1	Linear comparator 1 comparing end
3	IFCF0	FIFO comparator 0 comparing end
4	IFCF1	FIFO comparator 1 comparing end
5	IMOV	Motion control loop overload
6	IHOV	System job loop overload
7	--	Reserved, set to "0"

- **Item = 9: Digital input rising edge interrupt**

Bit No.	7	6	5	4	3	2	1	0
Factor	IDIR7	IDIR6	IDIR5	IDIR4	IDIR3	IDIR2	IDIR1	IDIR0
Bit No.	15	14	13	12	11	10	9	8
Factor	IDIR15 (TTL7)	IDIR14 (TTL6)	IDIR13 (TTL5)	IDIR12 (TTL4)	IDIR11 (TTL3)	IDIR10 (TTL2)	IDIR9 (TTL1)	IDIR8 (TTL0)
Bit No.	23	22	21	20	19	18	17	16
Factor	IDIR23 (TTL15)	IDIR22 (TTL14)	IDIR21 (TTL13)	IDIR20 (TTL12)	IDIR19 (TTL11)	IDIR18 (TTL10)	IDIR17 (TTL9)	IDIR16 (TTL8)
Bit No.	31	30	29	28	27	26	25	24
Factor	--	--	--	--	--	--	--	--

- **Item = 10: Digital input falling edge interrupt:**

Bit No.	7	6	5	4	3	2	1	0
Factor	IDIF7	IDIF6	IDIF5	IDIF4	IDIF3	IDIF2	IDIF1	IDIF0
Bit No.	15	14	13	12	11	10	9	8
Factor	IDIF15 (TTL7)	IDIF14 (TTL6)	IDIF13 (TTL5)	IDIF12 (TTL4)	IDIF11 (TTL3)	IDIF10 (TTL2)	IDIF9 (TTL1)	IDIF8 (TTL0)
Bit No.	23	22	21	20	19	18	17	16
Factor	IDIF23 (TTL15)	IDIF22 (TTL14)	IDIF21 (TTL13)	IDIF20 (TTL12)	IDIF19 (TTL11)	IDIF18 (TTL10)	IDIF17 (TTL9)	IDIF16 (TTL8)
Bit No.	31	30	29	28	27	26	25	24
Factor	--	--	--	--	--	--	--	--

Digital input signal (DI) status changes are detected by controller in every motion cycle. Interrupt can be generated only when the period of external input signal change cycle is greater than that of motion cycle.

You may use interrupt function in Windows environment as described below:

1. Set up interrupt events
2. Enable main interrupt switch
3. Waiting for interrupt trigger
4. Reset interrupt to non-signaled state
5. Close interrupt event and main interrupt switch

See table below for relevant APS APIs:

I32 APS_int_enable (I32 Board_ID, I32 Enable);

I32 APS_set_int_factor (I32 Board_ID, I32 Item_No, I32 Factor_No, Enable);

I32 APS_get_int_factor (I32 Board_ID, I32 Item_No, I32 Factor_No, *Enable);

HANDLE APS_int_no_to_handle (I32 Int_No);

I32 APS_wait_single_int (I32 Int_No, I32 Time_Out);

I32 APS_wait_multiple_int (I32 Int_Count, I32 *Int_No_Array, I32 Wait_All, I32 Time_Out);

I32 APS_reset_int (I32 Int_No);

I32 APS_set_int (I32 Int_No);

Detailed operation methods are described below:

1. Set up interrupt events:

   Use ***APS_set_int_factor( )*** to set up interrupt event for waiting. The function returns interrupt event number if setup is successful. You shall store event number in a parameter to be used by later Wait functions.

   The ***APS_set_int_factor( )*** function can be used to close opened interrupt event as required by application.

2. Enable main interrupt switch:

   Interrupt signal of hardware device can be received only when the main interrupt switch of controller is opened. Open with ***APS_int_enable( )***.

3. Waiting for interrupt trigger

   Use ***APS_wait_single_int( )*** to wait for single interrupt event, or ***APS_wait_multiple_int( )*** to wait for multiple interrupt events concurrently.

   The program enters sleep mode after entering this function. That is, the program (or thread) consumes no CPU resources until there is interrupt event or timeout occurred. You may use the returned value from the "wait" function to ensure the occurrence of event in waiting and execute followed application steps.

4. Reset interrupt to non-signaled state

   When there is event triggered and the program left the "wait" function, the interrupt event is set in signaled state. To wait for the same event's occurrence again, please reset the interrupt status to non-signaled state manually. Call the "Wait" function before reset will cause the Wait function to return directly. Function for reset: ***APS_reset_int( )***

5. Close interrupt event and main interrupt switch

   Finally, use ***APS_set_int_factor( )*** function to close individual interrupt event, use ***APS_int_enable( )*** function to close main interrupt switch to release all interrupt relevant resources.

- **Example:**

```
void interrupt_example()
{
  // This example shows how interrupt functions work.
  I32 board_id = 0;
  I32 int_no;      // Interrupt number
  I32 return_code; // function return code
  I32 item = 0;    // Axis #0 interrupt
  I32 factor = ( 1 << 12 ); // bit 12 IMDN interrupt

  //Step 1: 設定要等待的中斷事件, factor = IMDN
  int_no = APS_set_int_factor( board_id, item, factor, 1 );
  //Step 2: 設定中斷總開關
  APS_int_enable( board_id, 1 ); // Enable the interrupt main switch
  //Step 3: 等待中斷觸發
  return_code = APS_wait_single_int( int_no, -1 ); //Wait interrupt
  if( return_code == ERR_NoError )
  { //Interrupt occurred
    //Step 4: 重置中斷為觸發狀態
APS_reset_int( int_no );
  }
  // Step 5: 關閉中斷事件和中斷總開關
  APS_set_int_factor( board_id, item, factor, 0 );
  APS_int_enable( board_id, 0 );
}
```

In addition, you may use Event handle of win32 by using APS_int_no_to_handle() after step 1 to convert Event number into format of win32 Event handle.

- **Example:**

```
#include <windows.h>   // Using event handle
#include "APS168.h"
#include "ErrorCodeDef.h"

void interrupt_with_win32_example()
{
  // This example shows how interrupt functions work.
  I32 board_id = 0;
  I32 int_no;       // Interrupt number
  DWORD return_code; // function return code
  I32 item = 0;      // Axis #0 interrupt
  I32 factor = ( 1 << 12 ); // bit 12 IMDN interrupt
  HANDLE handle;

  //Step 1: 設定要等待的中斷事件, factor = IMDN
  int_no = APS_set_int_factor( board_id, item, factor, 1 );
  handle   = APS_int_no_to_handle( int_no );
  //Step 2: 設定中斷總開關
  APS_int_enable( board_id, 1 ); // Enable the interrupt main switch
  //Step 3: 等待中斷觸發
  return_code = WaitForSingleObject( handle, INFINITE );

  if( return_code == ERR_NoError )
  { //Interrupt occurred
    //Step 4: 重置中斷為為觸發狀態
ResetEvent( handle );
  }
  // Step 5: 關閉中斷事件和中斷總開關
  APS_set_int_factor( board_id, item, factor, 0 );
  APS_int_enable( board_id, 0 );
}
```

Important Safety Instructions

For user safety, please read and follow all instructions, WARNINGS, CAUTIONS, and NOTES marked in this manual and on the associated equipment before handling/operating the equipment.

- Read these safety instructions carefully.
- Keep this user's manual for future reference.
- Read the specifications section of this manual for detailed information on the operating environment of this equipment.
- When installing/mounting or uninstalling/removing equipment:
  - Turn off power and unplug any power cords/cables.
- To avoid electrical shock and/or damage to equipment:
  - Keep equipment away from water or liquid sources;
  - Keep equipment away from high heat or high humidity;
  - Keep equipment properly ventilated (do not block or cover ventilation openings);
  - Make sure to use recommended voltage and power source settings;
  - Always install and operate equipment near an easily accessible electrical socket-outlet;
  - Secure the power cord (do not place any object on/over the power cord);
  - Only install/attach and operate equipment on stable surfaces and/or recommended mountings; and,
  - If the equipment will not be used for long periods of time, turn off and unplug the equipment from its power source.

- Never attempt to fix the equipment. Equipment should only be serviced by qualified personnel.

A Lithium-type battery may be provided for uninterrupted, backup or emergency power.

Risk of explosion if battery is replaced with one of an incorrect type. Dispose of used batteries appropriately.

- Equipment must be serviced by authorized technicians when:
  - The power cord or plug is damaged;
  - Liquid has penetrated the equipment;
  - It has been exposed to high humidity/moisture;
  - It is not functioning or does not function according to the user's manual;
  - It has been dropped and/or damaged; and/or,
  - It has an obvious sign of breakage.

Getting Service

Contact us should you require any service or assistance.

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

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

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

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

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

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

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

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

ADLINK Technology, Inc. (Korean Liaison Office)
Address: 137-881 서울시 서초구 서초대로 326, 802 (서초동, 모인터빌딩)
802, Mointer B/D, 326 Seocho-daero, Seocho-Gu,
Seoul 137-881, Korea
Tel: +82-2-2057-0565
Fax: +82-2-2057-0563
Email: korea@adlinktech.com

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

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

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

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