PCI-8372+/8366+
cPCI-8312H

SSCNET Motion Control Card

User's Manual

Manual Rev. 2.04

Revision Date: June 13, 2008

Part No: 50-1H001-1020

Recycled Paper

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.

Trademarks

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

Getting Service from ADLINK

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

ADLINK TECHNOLOGY INC.

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

Please email or FAX this completed service form for prompt and satisfactory service.

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

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

Table of Contents

List of Tables

List of Figures

1 Introduction

PCI-8372+/8366+ is a PCI bus interface card designed for personal computer or industrial computer accompanied with a Mitsubishi MR-J2S-B type or SSCNET type servo amplifier. PCI-8372+ can control up to 12 servo amplifiers, where as PCI-8366+ can control up to 6 servo amplifiers.

cPCI-8312H is a CompactPCI bus interface card in 6U size. It controls up to 12 SSCNET axes and two HSL network ports in one board.

The connection between the motion control board and the amplifier is done via high-speed serial communication of the SSCNet II protocol. SSCNet II connections offer the following advantages over pulse train type connections:

- Wiring is simplified because servo amplifiers are connected by multi-drop method and the communication distance is up to 30 meters.
- Parameter management and the construction of absolute positioning system (ABS) are greatly simplified.
- Since commands are transmitted in serial data format, noise-reduction is better, thus reliability is improved. Also the control resolution is increased.
- Users can retrieve abundant information from the servo system through SSCNet II. No longer are you restricted to commands and feedback. You can now also monitor servo status, alarm status and tuning servo parameters.

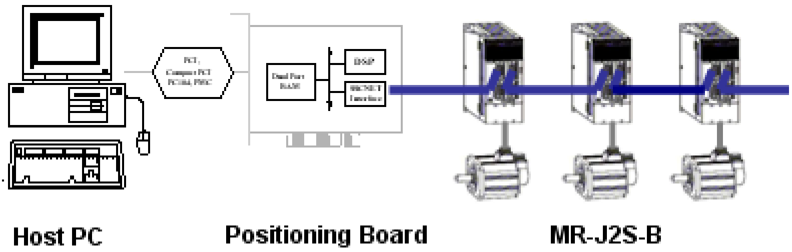


Figure 1-1: SSCNet II High-Speed Connections

- Since all axes are synchronized within the SSCNet cycle, multi-axis interpolation has better synchronicity than traditional pulse train control.

The on-board DSP controls all calculations necessary for performing various motion functions, thus, the host CPU loading is greatly reduced. These motion functions include single axis (jog, P to P move, change velocity/position on the fly, etc.), multi axes (circular, linear interpolation, etc.) and continuous motion.

Motion Creator, a Microsoft Windows based software is provided with the SSCNET board card to support in application developments. Motion Creator will be helpful in debugging a motion control system during the design phase of a project.

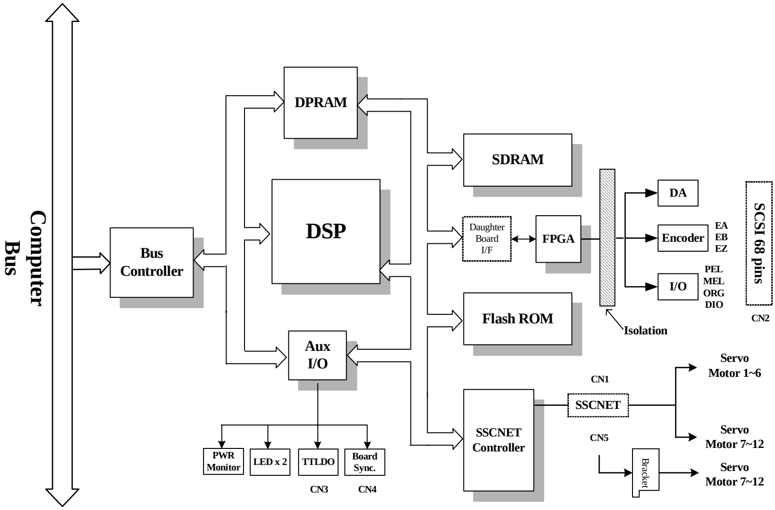


Figure 1-2: Block Diagram

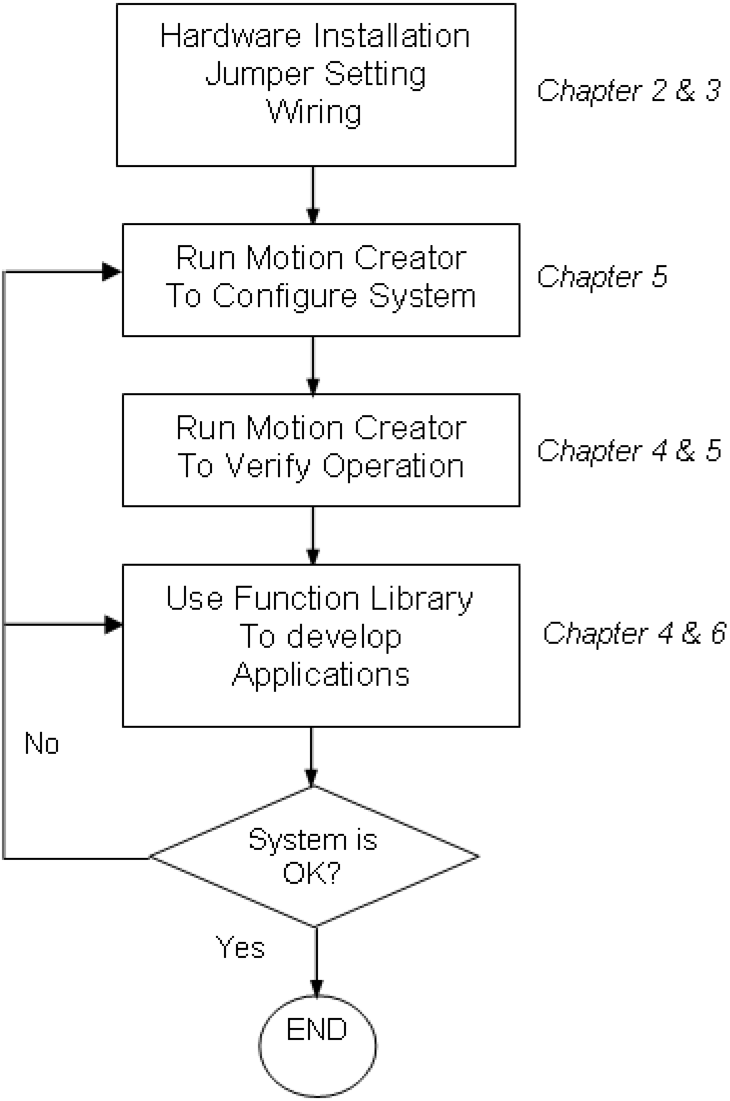


Figure 1-3: Flowchart for Building an Application

1.1 Specifications

	Item	Description
System	Bus Type for PCI board	PCI Rev. 2.2, 33MHz
	Bus width for PCI	32-bit
	Bus Voltage	5V
	Memory usage	16KByte
	IRQ on PCI board	Assigned by PCI controller
General Specifications	Operating temperature	0°C - 60°C
	Storage temperature	-20°C -80°C
	Humidity	5 - 95%, non-condensing
	Power Consumption	PCI-8372+/8366+: +5V @ 1A typical
DSP	Type	TI TMS320C6711
	Clock	200 MHz
	DSP performance	1200 MFLOPS
Board Interface		
	I/O Connector	68-pin VHDIC
	SSCNet Connector	3M 10220-52A2JL
Driver Communication	Protocol	SSCNET II
	Bit Rate	5.625Mhz
	Physical layer	RS-485
	Maximum working length	30m for each 6 axes
	Error detection	CRC
Servo Loop	Max. No of controllable axes	8372: 12; 8366: 6
	Servo update rate	0.888ms
	Servo Data Monitors	Current position
		Droop (deviation)
		Velocity Command
		Velocity feedback
		Torque command
		Servo alarm number …etc
	Servo parameter tuning	Parameter read/write

Table 1-1: Specifications

	Item	Description
Motion Function	Motion Velocity Profile	Trapezoidal & S-Curve
	Single motion	Jog move
		Single axis P to P motion
		Change P/V on the fly
		Linear interpolation: up to 4 axes
		2-axis Circular interpolation
	Home move	1 home mode
	Continuous motion	Start / End motion list
		Add linear trajectory
		Add arc trajectory: 2 axes
		Add Dwell
		Smooth Trajectory
		Start/Sop command
		Motion IO status read/configure
		Motion status
Application Functions	Move Ratio	In unit of Pulse per mm
	Software Limit	Each axis has 2 soft limits
	Position Compare	Each axis has 2 comparators
	Interlock	2 axes interlock system
	System error check	Watchdog timer
Interrupt	During operation stop	Possible to select conditions where interrupt occurs
	During alarms, etc.	Yes

Table 1-1: Specifications

	Item	Description
Optical Isolated Digital Input	+Limit Switch x 12 (PEL)	▶Sink or source type are selectable in all channels (all channels must be the same) ▶Input voltage range: 0 - 24V ▷Logic H: 14.4 - 24V ▷Logic L: 0 - 5V ▶Input resistor: 4.7kOhm @ 0.5W ▶DI change of state detection ▶Isolated voltage: 500Vrms ▶Bandwidth: 10kHz (0.1ms)
	-Limit Switch x 12 (MEL)	
	Proximity dog x 12 (ORG)	
	General Purposed Input x 2 (PCI board only)	
	Emergency Stop x 1	
Digital Output	DO x 2	▶Output type: ▷ Open-collector (PC3H7) ▶Sink Current: 6.5mA Min. ▶Isolated voltage: 500 VDC ▶Bandwidth: 10kHz (0.1 ms)

Table 1-1: Specifications

	Item	Description
Analog Out	DA x 2	▶Resolution: 16 bits ▶Settling Time: 10mS Max. ▶Output Range: ±10V ▶Output Coupling: DC ▶Output Impedance: 30W Max. ▶Output Driving: ±5mA max. ▶Power On State: Floating ▶Calibration: Self-Calibration ▶Gain Error: ±3% Max. ▶Offset Error: ▷1mV Max. for PCI board
Analog In	AD x 2 (Avaialble for cPCI board)	▶Resolution: 16 bits, no missing code ▶Sampling Rate: 250kS/s ▶Programmable Input Range: ±10V, ±5V, ±2.5V ▶Calibration: Self-Calibration ▶Gain Error: ±0.03% Max. ▶Offset Error: 0.2mV Max.

Table 1-1: Specifications

	Item	Description
Encoder Interface	32-bit Encoder input (A,B,Z) x 3 channel (PCI)	▶Incremental Encoder Input Max. Speed: 5Mhz ▶Input Voltage: 0 - 5Vdc ▶Logic H: 3 - 5V ▶Logic L: 0 - 2.4V ▶Input resistor: 220Ω @ 0.125W ▶Isolated voltage: 500Vrms
Pulse Output	2 channel differential pulses output (Available for cPCI board)	▶OUT/DIR, CW/CCW, AB phase selectable ▶Max. Output Frequence: 4.16Mhz ▶Isolated voltage: 500 Vrms
Aux. DIO	6 TTL Level Digital Output (at CN3 on Extension bracket of PCI board only)	▶Voltage output high: Typical: 5V, Min: 2.4v @ 15mA ▶Voltage output low: Typical: 0.3V @ 24mA, Max: 0.5V

Table 1-1: Specifications

1.2 Environmental Conditions

- Ambient Temperature Operation: 0 - 55°C
- Ambient Temperature Storage: -20 - 75°C
- Ambient Humidity Operation: 10 - 90%RH, avoid condensation
- Ambient Humidity Storage: 10 - 90%RH, avoid condensation
- Vibration Resistance
  - Confirms to JIS C 0911

Frequency	Acceleration	Amplitude of Vibration	Sweep
10~55Hz	-	0.075mm	10 times*
55~150Hz	1G	-	(1 ctave/minute)

Table 1-2: Vibration Resistance

- Shock resistance: Confirms to JIS C 0912 (10g, 3 directions, 3 times)
- Noise resistance: Noise voltage 1500V.P.P, Noise frequency 25 - 60Hz using noise simulator
- Operating tmosphere: Minimal corrosive gas, dust
- Cooling method: Self-cooling

Note: *One Octave: from initial frequency to double initial frequency or half initial frequency. For example: 10Hz -> 20Hz, 20Hz -> 40Hz -> 20Hz, 20Hz -> 10Hz. Each change is referred to as an octave.

1.3 Software Support

1.3.1 Programming Library

For customers who are programming their own applications, we provide Windows 95/98/NT/2000/XP DLLs for the PCI-8372+/8366+ and cPCI-8312 (H). It is shipped with these boards.

1.3.2 Motion Creator

Motion Creator is a Windows-based utility to setup cards, motors and system. It can also help users debug hardware and software problems. It also can let users set I/O logic parameters, which can be loaded in their own program. This product is bundled with this card. Refer to Chapter 5 for details.

2 Installation

This chapter describes how to install the PCI-8372+/8366+ or cPCI-8312 (H). Please follow these steps below to install the board.

2.1 What You Have

In addition to this User's Guide, the package should also include the following items:

- SSCNET Motion Control Card
- ADLINK All-in-one Compact Disc for driver installation
- User's Manual and Function Library. You can find the PDF files in the installed directory

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

2.2 PCI-8372+/8366+ Outline Drawing

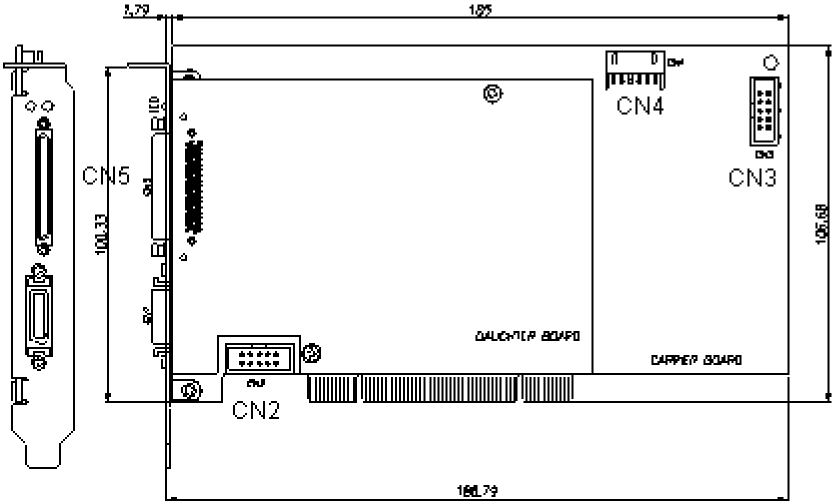


Figure 2-1: PCI-8372+/8366+ Mechanical Drawing

2.3 cPCI-8312(H) Outline Drawing

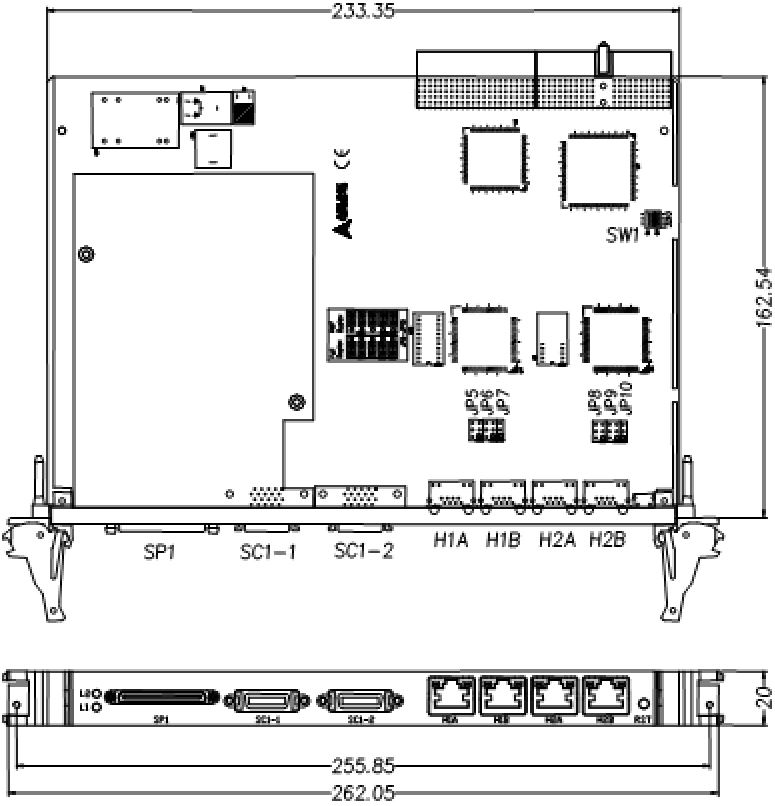


Figure 2-2: cPCI-8312(H) Mechanical Drawing

- SC1-1: SSCNET connector for Axis 0-5
- SC1-2: SSCNET connector for Axis 6-11
- H1A, H1B: First HSL Set
- H2A, H2B: Second HSL Set
- SP1: Daughter Board connector
- L1: Board Status LED in Green
- L2: Board Status LED in Red

- RST: Board Reset Button
- SW1: CardID
- JP5-JP7: H1A,H1B Communication Mode Selection
- JP8-JP10: H2A, H2B Communication Mode Selection

2.4 Hardware Installation

2.4.1 Installation Procedures

1. Turn off your computer and all accessories (printer, modem, monitor, etc.) connected to computer. Remove the cover from your computer.
2. Hardware installation:
   - ▷ For PCI board: Select a 32-bit PCI expansion slot. PCI slots are shorter than ISA or EISA slots and are usually white or ivory.
   - ▷ For CompactPCI board: Carefully push the board into the cPCI system through the groove. Be aware that the pin on the slot would be bent.
3. Before handling the PCI-8372+/8366+ or cPCI-8312 (H), discharge any static electric charge on your body by touching the metal case of the computer. Hold the edge and do not touch the components.
4. Position the board into the PCI/CompactPCI slot you selected.
5. For PCI/CompactPCI borad, secure the card in place at the rear panel of the system unit using screws removed from the slot.

2.4.2 LED Status

Please carefully read the following.

There are two LEDs present on the card’s bracket, Red and Green. These LEDs indicate the operation status of the card. If the system is turned on, these LEDs will blink together. This means it finishes the self-testing mode.

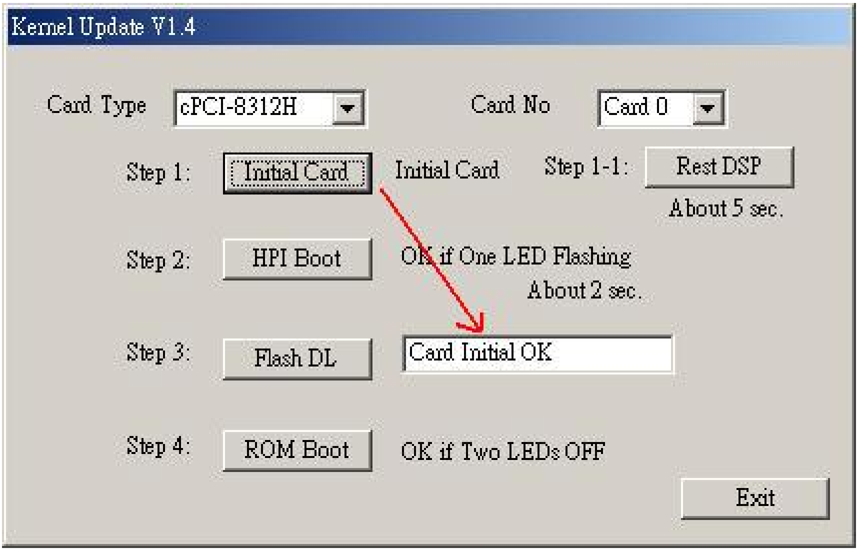
This sequence of self-testing is executed automatically when the system is reset or powered up. The procedure takes about two seconds. Over two seconds, the LEDs will be turned off. If not, there is something wrong with this board. This abnormality means that the card fails or the system’s power supply may be unstable. Users have to try downloading the DSP kernel again by KernelUpdate.EXE utility or change the power supply. If this board still in the faulty situation, please try to test it in another platform or replace a new board for testing again.

If the application runs the MDSP_initial() function and it is successfully executed, the LEDs will turn on and off about every 1 second. If the application program calls the MDSP_close() function, the two LEDs will be turn off.

2.4.3 KernelUpdate Utility of SSCNET card

To Reset DSP: Press Step 1 then Step 1-1

To Update Kernel: Press Step 1 - Step 4 (Ignore Step 1-1)



Kernel Update V1.4

Card Type: cPCI-8312H

Card No: Card 0

Step 1: Initial Card

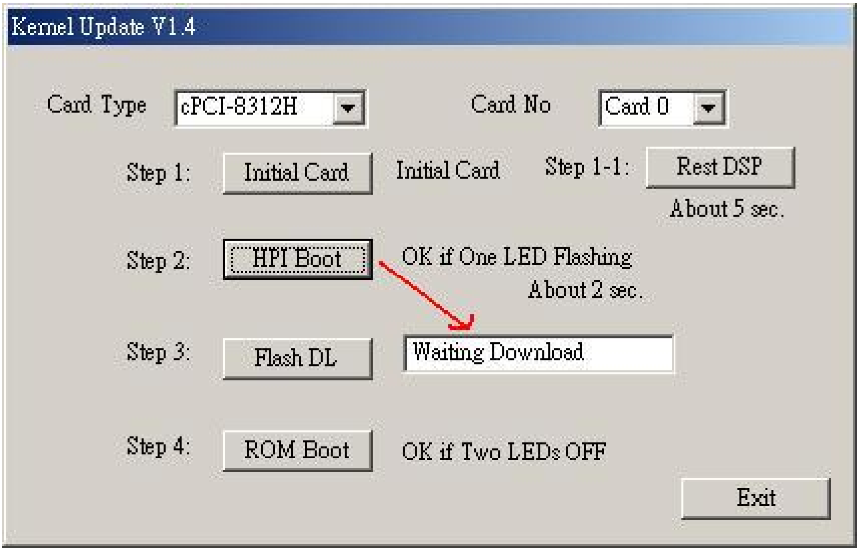
Step 1-1: About 5 sec.

Step 2: OK if One LED Flashing. About 2 sec.

Step 3: Card Initial OK

Step 4: OK if Two LEDs OFF

1. Select a card and initial it

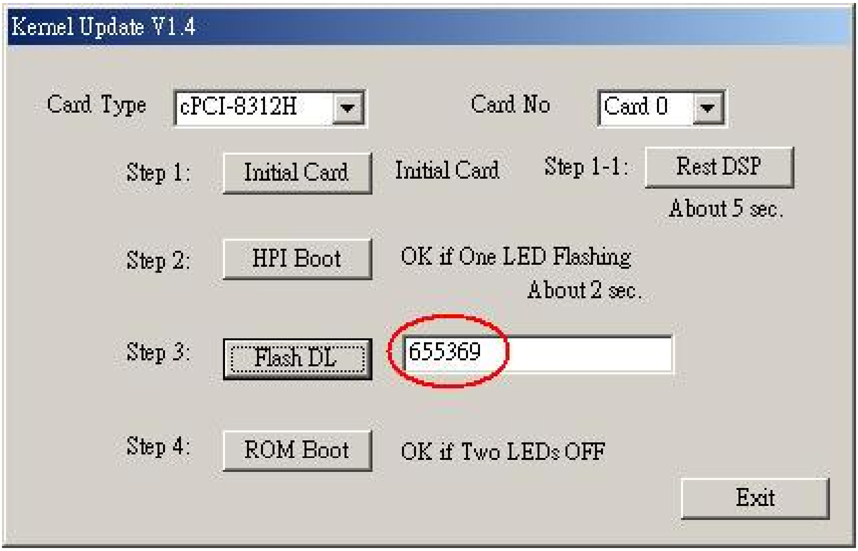


Kernel Update V1.4

Card Type: cPCI-8312H

Card No: Card 0

2. Press “HPI boot”



Kernel Update V1.4

Card Type: cPCI-8312H

Card No: Card 0

Step 1: Initial Card — Initial Card

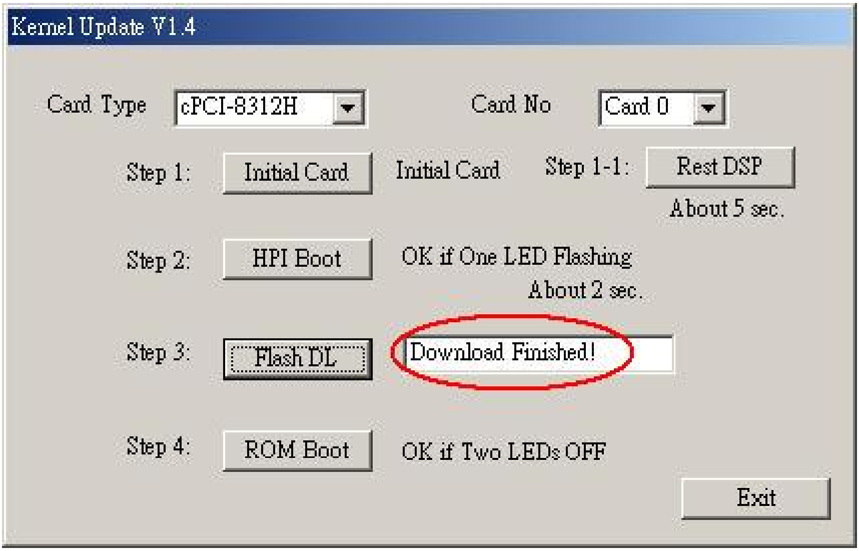
Step 1-1: Rest DSP — About 5 sec.

Step 2: HPI Boot — OK if One LED Flashing About 2 sec.

Step 3: Flash DL — 655369

Step 4: ROM Boot — OK if Two LEDs OFF

3. Press “Flash DL” button and select a kernel4.hex



Card Type: cPCI-8312H

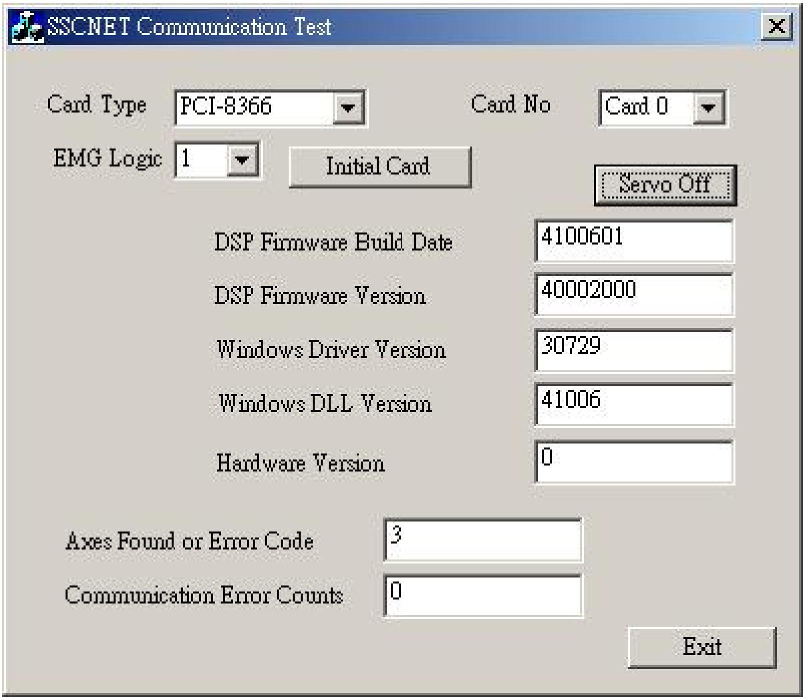
Card No: Card 0

Download Finished!

4. Wait the value become 0 and displays Download Finished
5. Press “ROM Boot” and wait about 5 sec and done.

2.4.4 SSCNET Communication Test Utility

We provide a test utility for SSCNET communication. After initialized, you can check the communication error counts from the dialog. Once it has communication errors, please disconnect the driver one by one and use a new cable to verify it.



Card Type: PCI-8366

Card No: Card 0

EMG Logic: 1

DSP Firmware Build Date: 4100601

DSP Firmware Version: 40002000

Windows Driver Version: 30729

Windows DLL Version: 41006

Hardware Version: 0

Axes Found or Error Code: 3

Communication Error Counts: 0

Figure 2-3: SSCNET Communication Test Utility

2.5 Software Driver Installation

1. Auto-Run from the ADLINK ALL-In-One CD, choose Motion Control and then SSCNET series baord
2. Follow the installation wizard
3. Shut down your computer, and insert the SSCNET series board into a slot and then power up the computer
4. When the installation is completed, the following folder will be created in the directory specified during installation (default directory "C:\Program Files\ADLINK\SSCNET").
   - ▷ **Library**: this folder contains files required for a project when programming an application.
   - ▷ **DSPKernel**: this folder contents a DSP kernel program with default settings. If a recovery of your system is required, use the Motion Creator utility to download the DSP kernel firmware.
   - ▷ **Utility**: Some utility for the board
   - ▷ **Manual**: An user's manual for a product
   - ▷ **Driver**: pci8372.sys and pci8366.sys
5. Execute "Motion Creator" in the Startup menu to confirm your hardware version by clicking the card list

2.6 CN1 Pin Assignment: SSCNet Connector on PCB

Receptacle: 10220-52A2JL

Manufacturer: 3M

No	Name	I/O	Function	No	Name	I/O	Function
1	GND	-	Signal Ground	11	GND	-	Signal Ground
2	TXD1+	O	Transmit+	12	TXD1-	O	Transmit -
3	TXD2+	O	Transmit+	13	TXD2-	O	Transmit -
4	RXD1+	I	Receive +	14	RXD1-	I	Receive -
5	GND	-	Signal Ground	15	GND	-	Signal Ground
6	RXD2+	I	Receive +	16	RXD2-	I	Receive -
7	EMG1+	O	Emergency1+	17	EMG1-	O	Emergency1-
8	EMG2+	O	Emergency2+	18	EMG2-	O	Emergency2-
9	NC	-		19	NC	-	
10	NC	-		20	NC	-	

Table 2-1: CN1 Pin Assignment

2.7 CN5 Pin Assignment: PCI-8372+/8366+ I/O Connector

No	Name	I/O	Function Axis	No	Name	I/O	Function Axis
1	A.COM	-	Analog Ground	35	DA1	I	Analog Output
2	PEL1/MDI1	I	Positive End Limit	36	DA2	I	Analog Output
3	MEL1/MDI2	I	Minus End Limit	37	PEL2/MDI4	I	Positive End Limit
4	ORG1/MDI3	I	Origin Signal	38	MEL2/MDI5	I	Minus End Limit
5	PEL3/MDI7	I	Positive End Limit	39	ORG2/MDI6	I	Origin Signal
6	MEL3/MDI8	I	Minus End Limit	40	PEL4/MDI10	I	Positive End Limit
7	ORG3/MDI9	I	Origin Signal	41	MEL4/MDI11	I	Minus End Limit
8	PEL5/MDI13	I	Positive End Limit	42	ORG4/MDI12	I	Origin Signal
9	MEL5/MDI14	I	Minus End Limit	43	PEL6/MDI16	I	Positive End Limit
10	ORG5/MDI15	I	Origin Signal	44	MEL6/MDI17	I	Minus End Limit
11	IPT_COM	I	Common for Digital Input	45	ORG6/MDI18	I	Origin Signal
12	EA1+	I	Encoder A-Phase (+)	46	EA2+	I	Encoder A-Phase (+)
13	EA1-	I	Encoder A-Phase (-)	47	EA2-	I	Encoder A-Phase (-)
14	EB1+	I	Encoder B-Phase (+)	48	EB2+	I	Encoder B-Phase (+)
15	EB1-	I	Encoder B-Phase (-)	49	EB2-	I	Encoder B-Phase (-)
16	EZ1+	I	Encoder Z-Phase (+)	50	EZ2+	I	Encoder Z-Phase (+)
17	EZ1-	I	Encoder Z-Phase (-)	51	EZ2-	I	Encoder Z-Phase (-)
18	PEL7/MDI19	I	Positive End Limit	52	PEL8/MDI22	I	Positive End Limit
19	MEL7/MDI20	I	Minus End Limit	53	MEL8/MDI23	I	Minus End Limit
20	ORG7/MDI21	I	Origin Signal	54	ORG8/MDI24	I	Origin Signal
21	PEL9/MDI25	I	Positive End Limit	55	PEL10/MDI28	I	Positive End Limit
22	MEL9/MDI26	I	Minus End Limit	56	MEL10/MDI29	I	Minus End Limit
23	ORG9/MDI27	I	Origin Signal	57	ORG10/MDI30	I	Origin Signal
24	PEL11/MDI31	I	Positive End Limit	58	PEL12/MDI34	I	Positive End Limit
25	MEL11/MDI32	I	Minus End Limit	59	MEL12/MDI35	I	Minus End Limit
26	ORG11/MDI33	I	Origin Signal	60	ORG12/MDI36	I	Origin Signal
27	IPT_COM	I	Common for Digital Input	61	IPT_COM	I	Common for Digital Input
28	DO_COM	I	Common for Digital Output	62	DI1	I	General Digital Input
29	EA3+	I	Encoder A-Phase (+)	63	DI2	I	General Digital Input
30	EA3-	I	Encoder A-Phase (-)	64	EMG	I	Emergency Stop Signal
31	EB3+	I	Encoder B-Phase (+)	65	EMG_COM	-	Emergency Stop Common
32	EB3-	I	Encoder B-Phase (-)	66	DO1	O	General Digital Output
33	EZ3+	I	Encoder Z-Phase (+)	67	DO2	O	General Digital Output
34	EZ3-	I	Encoder Z-Phase (-)	68	DO_COM	-	Common for Digital Output

Table 2-2: CN5 Pin Assignment

Note: *MDI# is for general purpose input if it is not used for motion.

2.8 SP1 Pin Assignment: cPCI-8312(H) I/O Connector

No	Name	I/O	Function Axis	No	Name	I/O	Function Axis
1	DO_COM	-	Common for Digital Output	35	DO1	O	General Digital Output
2	PEL1/MDI1	I	Positive End Limit	36	DO2	O	General Digital Output
3	MEL1/MDI2	I	Minus End Limit	37	PEL2/MDI4	I	Positive End Limit
4	ORG1/MDI3	I	Origin Signal	38	MEL2/MDI5	I	Minus End Limit
5	PEL3/MDI7	I	Positive End Limit	39	ORG2/MDI6	I	Origin Signal
6	MEL3/MDI8	I	Minus End Limit	40	PEL4/MDI10	I	Positive End Limit
7	ORG3/MDI9	I	Origin Signal	41	MEL4/MDI11	I	Minus End Limit
8	PEL5/MDI13	I	Positive End Limit	42	ORG4/MDI12	I	Origin Signal
9	MEL5/MDI14	I	Minus End Limit	43	PEL6/MDI16	I	Positive End Limit
10	ORG5/MDI15	I	Origin Signal	44	MEL6/MDI17	I	Minus End Limit
11	IPT_COM/						
EMG_COM	-	Common for Digital Input	45	ORG6/MDI18	I	Origin Signal	
12	EA1+	I	Encoder A-Phase (+)	46	EA2+	I	Encoder A-Phase (+)
13	EA1-	I	Encoder A-Phase (-)	47	EA2-	I	Encoder A-Phase (-)
14	EB1+	I	Encoder B-Phase (+)	48	EB2+	I	Encoder B-Phase (+)
15	EB1-	I	Encoder B-Phase (-)	49	EB2-	I	Encoder B-Phase (-)
16	EZ1+	I	Encoder Z-Phase (+)	50	EZ2+	I	Encoder Z-Phase (+)
17	EZ1-	I	Encoder Z-Phase (-)	51	EZ2-	I	Encoder Z-Phase (-)
18	PEL7/MDI19	I	Positive End Limit	52	PEL8/MDI22	I	Positive End Limit
19	MEL7/MDI20	I	Minus End Limit	53	MEL8/MDI23	I	Minus End Limit
20	ORG7/MDI21	I	Origin Signal	54	ORG8/MDI24	I	Origin Signal
21	PEL9/MDI25	I	Positive End Limit	55	PEL10/MDI28	I	Positive End Limit
22	MEL9/MDI26	I	Minus End Limit	56	MEL10/MDI29	I	Minus End Limit
23	ORG9/MDI27	I	Origin Signal	57	ORG10/MDI30	I	Origin Signal
24	PEL11/MDI31	I	Positive End Limit	58	PEL12/MDI34	I	Positive End Limit
25	MEL11/MDI32	I	Minus End Limit	59	MEL12/MDI35	I	Minus End Limit
26	ORG11/MDI33	I	Origin Signal	60	ORG12/MDI36	I	Origin Signal
27	IPT_COM/						
EMG_COM	-	Common for Digital Input	61	EMG	I	Emergency Stop Signal	
28	P_GND	-	Common for Pulse Interface	62	AD1	I	Analog Input
29	OUT1+	O	Pulse signal (+)	63	DIR1+	O	Dir. signal (+)
30	OUT1-	O	Pulse signal (-)	64	AD2	I	Analog Input

Table 2-3: SP1 Pin Assignment

No	Name	I/O	Function Axis	No	Name	I/O	Function Axis
31	OUT2+	O	Pulse signal (+)	65	DIR1-	O	Dir. signal (-)
32	OUT2-	O	Pulse signal (-)	66	DA1	O	Analog Output
33	DIR2+	O	Dir. signal (+)	67	DA2	O	Analog Output
34	DIR2-	O	Dir. signal (-)	68	A_COM	-	Analog Ground

Table 2-3: SP1 Pin Assignment

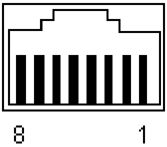
Note: *MDI# is for general purpose input if it is not used for motion

2.9 CN3 Pin Assignment: TTL output Connector on bracket

No	Name	I/O	Function Axis	No	Name	I/O	Function Axis
1	GND	-	Signal Ground	2	GND	-	Signal Ground
3	TDO1	O	TTL Output 1	4	TDO2	O	TTL Output 2
5	TDO3	O	TTL Output 3	6	TDO4	O	TTL Output 4
7	TDO5	O	TTL Output 5	8	TDO6	O	TTL Output 6
9	+5V	-	+5V Supply	10	NC	-	Not connected pin

Table 2-4: CN3 Pin Assignment

2.10 HS1A - HS2B Pin Assignments: HSL Communication Signal (RJ-45)



PIN	Signal
PIN 1	NC
PIN 2	NC
PIN 3	TXD+
PIN 4	RXD-
PIN 5	RXD+
PIN 6	TXD-
PIN 7	NC
PIN 8	NC

Table 2-5: HS1A - HS2B Pin Assignment

3 Signal Connections

Signal connections of all I/O's are described in this chapter. Refer to the contents of this chapter before wiring any cables between the 8372+/8366+ and any motor drivers

3.1 SSCNet Servo Driver Connection

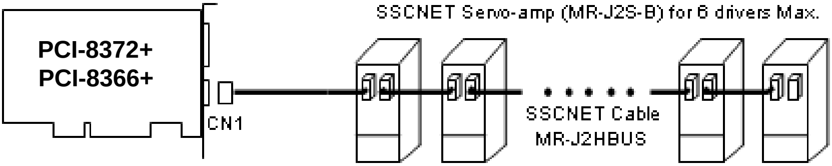


Figure 3-1: Wiring for 6 Axes (PCI-8372+/8366+)

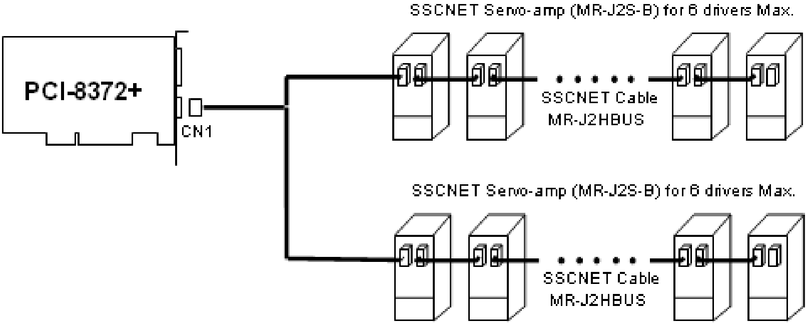
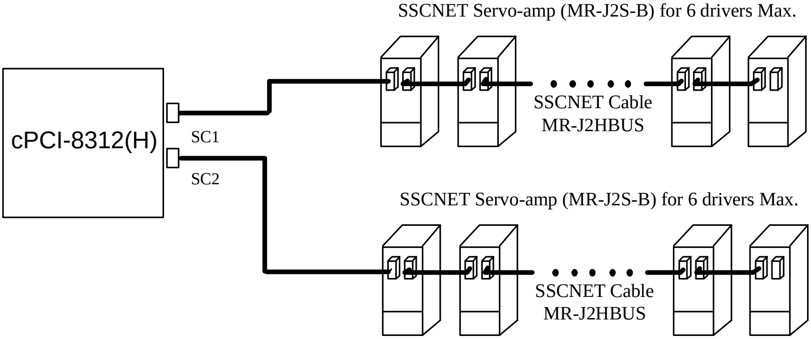

PCI-8372+
CN1
SSCNET Servo-amp (MR-J2S-B) for 6 drivers Max.
SSCNET Cable
MR-J2HBUS
SSCNET Servo-amp (MR-J2S-B) for 6 drivers Max.
SSCNET Cable
MR-J2HBUS

Figure 3-2: Wiring for 12 Axes (PCI-8372+)



SSCNET Servo-amp (MR-J2S-B) for 6 drivers Max.
cPCI-8312(H)
SC1
SC2
SSCNET Cable
MR-J2HBUS
SSCNET Servo-amp (MR-J2S-B) for 6 drivers Max.
SSCNET Cable
MR-J2HBUS

Figure 3-3: Wiring for cPCI-8312(H)

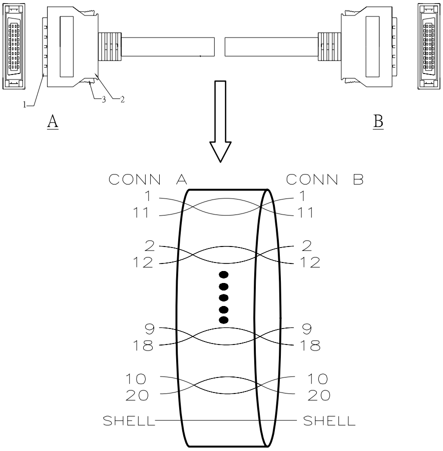


Figure 3-4: SSCNet Cable:

3.2 Encoder Feedback Signals: EA, EB and EZ

Pin No.		Name	Description
CN5	SP1		
12	12	EA1+	Encoder A-Phase (+)
13	13	EA1-	Encoder A-Phase (-)
14	14	EB1+	Encoder B-Phase (+)
15	15	EB1-	Encoder B-Phase (-)
16	16	EZ1+	Encoder Z-Phase (+)
17	17	EZ1-	Encoder Z-Phase (-)
46	46	EA2+	Encoder A-Phase (+)
47	47	EA2-	Encoder A-Phase (-)
48	48	EB2+	Encoder B-Phase (+)
49	49	EB2-	Encoder B-Phase (-)
50	50	EZ2+	Encoder Z-Phase (+)
51	51	EZ2-	Encoder Z-Phase (-)
29	--	EA3+	Encoder A-Phase (+)
30	--	EA3-	Encoder A-Phase (-)
31	--	EB3+	Encoder B-Phase (+)
32	--	EB3-	Encoder B-Phase (-)
33	--	EZ3+	Encoder Z-Phase (+)
34	--	EZ3-	Encoder Z-Phase (-)

Table 3-1: Encoder Feedback Signals: EA, EB and EZ

The encoder feedback signals include EA, EB, and EZ signals. EA and EB are used for position counting, and EZ is used for zero position indexing. The input circuit of the EA, EB, and EZ signals is shown in the diagram below.

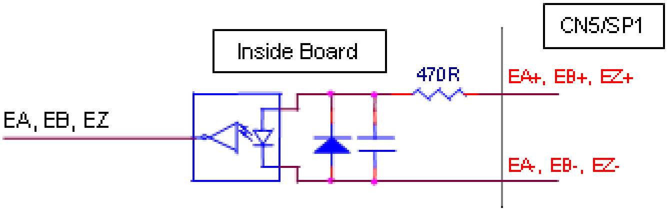


Figure 3-5: Encoder Feedback Signals

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

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

Connection to Line Driver Output

To drive the SSCNET board encoder input, the driver output must provide at least 3.5V across the differential pairs with at least 6 mA driving capacity. The ground level of the two sides must also be tied together.

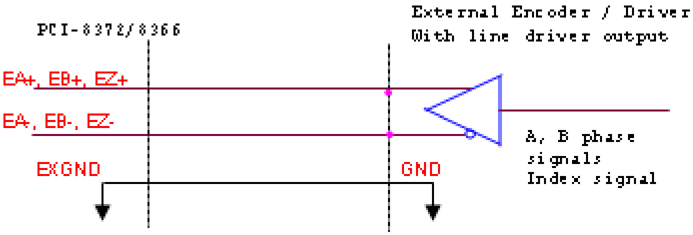


Figure 3-6: Line Drive Output Connection

Connection to Open Collector Output

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

Encoder Power (VDD)	External Resistor R
+5V	0 Ohms (None)
+12V	1.8kΩ
+24V	4.3kΩ

Table 3-2: Encoder Power

If=6mA max.

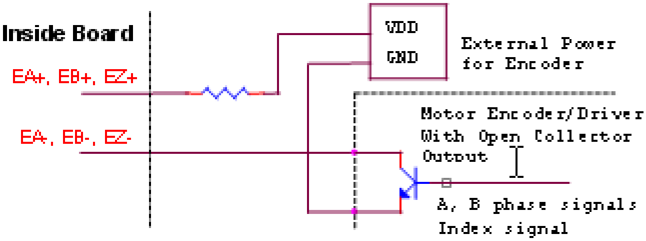


Figure 3-7: Open Collector Output Connection

3.3 PEL, MEL, ORG, EMG and General Purpose DI

Pin No.		Name	Description
CN5	SP1		
2	2	PEL1/MDI1	Positive End Limit / Axis 0
3	3	MEL1/MDI2	Minus End Limit / Axis 0
4	4	ORG1/MDI3	Origin Signal / Axis 0
37	37	PEL2/MDI4	Positive End Limit / Axis 1
38	38	MEL2/MDI5	Minus End Limit / Axis 1
39	39	ORG2/MDI6	Origin Signal / Axis 1
5	5	PEL3/MDI7	Positive End Limit / Axis 2
6	6	MEL3/MDI8	Minus End Limit / Axis 2
7	7	ORG3/MDI9	Origin Signal / Axis 2
40	40	PEL4/MDI10	Positive End Limit / Axis 3
41	41	MEL4/MDI11	Minus End Limit / Axis 3
42	42	ORG4/MDI12	Origin Signal / Axis 3
8	8	PEL5/MDI13	Positive End Limit / Axis 4
9	9	MEL5/MDI14	Minus End Limit / Axis 4
10	10	ORG5/MDI15	Origin Signal / Axis 4
43	43	PEL6/MDI16	Positive End Limit / Axis 5
44	44	MEL6/MDI17	Minus End Limit / Axis 5
45	45	ORG6/MDI18	Origin Signal / Axis 5
18	18	PEL7/MDI19	Positive End Limit / Axis 6
19	19	MEL7/MDI20	Minus End Limit / Axis 6
20	20	ORG7/MDI21	Origin Signal / Axis 6
52	52	PEL8/MDI22	Positive End Limit / Axis 7
53	53	MEL8/MDI23	Minus End Limit / Axis 7
54	54	ORG8/MDI24	Origin Signal / Axis 7
21	21	PEL9/MDI25	Positive End Limit / Axis 8
22	22	MEL9/MDI26	Minus End Limit / Axis 8
23	23	ORG9/MDI27	Origin Signal / Axis 8
55	55	PEL10/MDI28	Positive End Limit / Axis 9
56	56	MEL10/MDI29	Minus End Limit / Axis 9

Table 3-3: PEL, MEL, ORG, EMG and General Purpose DI

Pin No.		Name	Description
57	57	ORG10/MDI30	Origin Signal / Axis 9
24	24	PEL11/MDI31	Positive End Limit / Axis 10
25	25	MEL11/MDI32	Minus End Limit / Axis 10
26	26	ORG11/MDI33	Origin Signal / Axis 10
58	58	PEL12/MDI34	Positive End Limit / Axis 11
59	59	MEL12/MDI35	Minus End Limit / Axis 11
60	60	ORG12/MDI36	Origin Signal / Axis 11
62	--	DI1	General Digital Input
63	--	DI2	General Digital Input
27	27	IPT_COM	Common for Digital Input
11	11	IPT_COM	Common for Digital Input
61	--	IPT_COM	Common for Digital Input
64	61	EMG	Emergency Stop Signal
65	11,27	EMG_COM	Emergency Stop Common

Table 3-3: PEL, MEL, ORG, EMG and General Purpose DI

Note: MDI# is for general purpose input if it is not used for motion

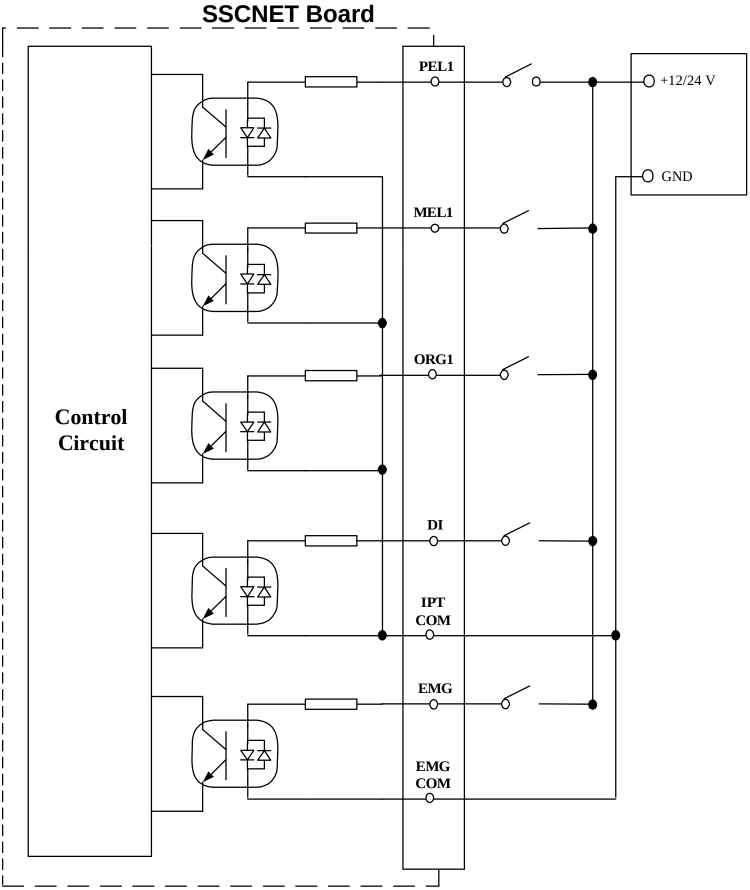


Figure 3-8: Source Type

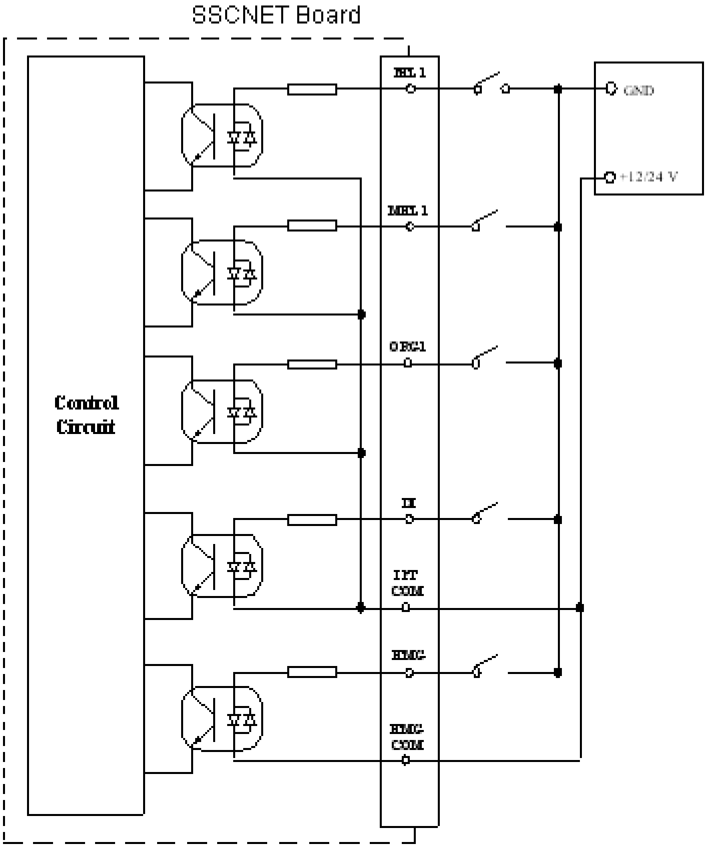


Figure 3-9: Skin Type

3.4 General Purpose DO

Pin No.		Name	Description
CN5	SP1		
28	1	DO_COM	Common for Digital Output
66	35	DO1	General Digital Output
67	36	DO2	General Digital Output
68	--	DO_COM	Common for Digital Output

Table 3-4: General Purpose DO Pinout

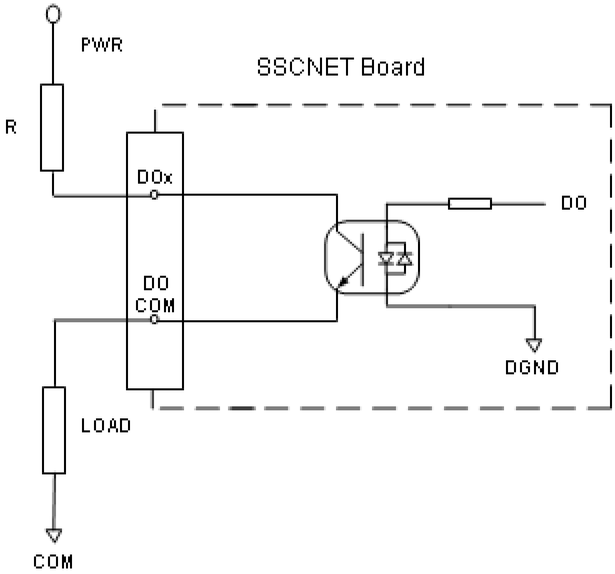


Figure 3-10: General Purpose DO

Note: For Example: R=4.7K and PWR=24V

3.5 TTL Output

The PCI-8372+/8366+ provides 6 general-purposed TTL digital outputs. The TTL output is available via CN3 of the bracket. Pin definition is defined in the below.

Pin No.	Name	Function
1	DGND	Digital ground
2	DGND	Digital ground
3	TDO1	Digital Output 1
4	TDO2	Digital Output 2
5	TDO3	Digital Output 3
6	TDO4	Digital Output 4
7	TDO5	Digital Output 5
8	TDO6	Digital Output 6
9	VCC	VCC +5V

Table 3-5: TTL Output Pinout

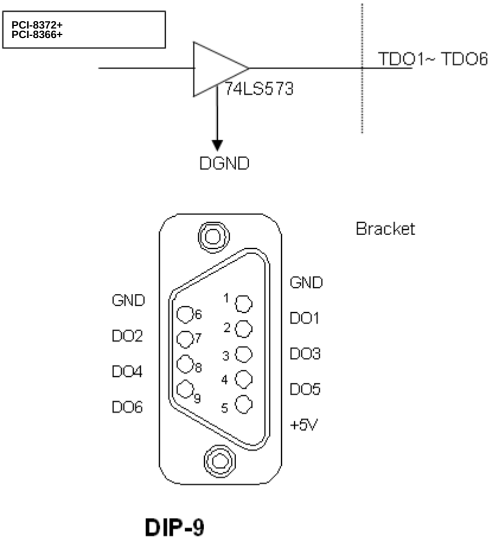


Figure 3-11: TTL Output

3.6 Analog Output

Pin No.		Name	Description
CN5	SP1		
1	68	A_COM	Common for Digital Output
35	66	DA1	Analog Output 1
36	67	DA2	Analog Output 2

Table 3-6: Analog Output Pinout

The SSCNET board has two bipolar analog output channels.

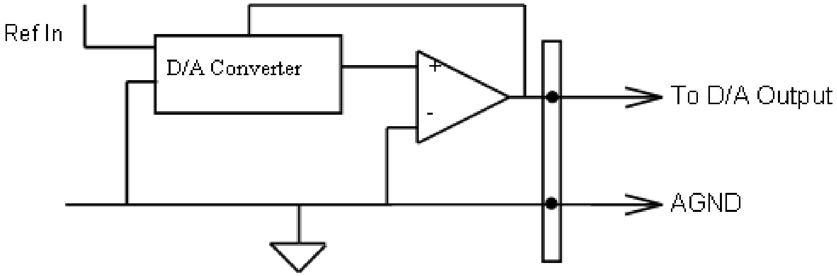


Figure 3-12: D/A Output Signals

3.7 Analog Input (cPCI-8312(H) Only)

Pin No.	Name	Description
SP1		
68	A_COM	Common for Digital Output
62	AD1	Analog Input 1
64	AD2	Analog Input 2

Table 3-7: Analog Input Pinout

The cPCI-8312(H) provides two single-ended analog input channels. The analog signal input range can be set as ±10V, ±5V or ±2.5V by software.

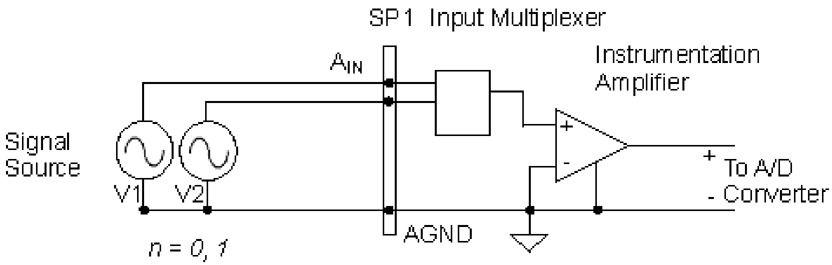


Figure 3-13: Analog Input

3.8 Pulse Output (cPCI-8312(H) Only)

Pin No.	Name	Description
SP1		
28	P_GND	Common ground of pulse interface
29	OUT1+	Pulse signal (+)
30	OUT1-	Pulse signal (-)
63	DIR1+	Dir Signal (+)
65	DIR1-	Dir Signal (-)
31	OUT2+	Pulse signal (+)
32	OUT2-	Pulse signal (-)
33	DIR2+	Dir Signal (+)
34	DIR2-	Dir Signal (-)

Table 3-8: Pulse Output Pinout

There are two axis pulse output signals on the cPCI-8312(H). For each axis, two pairs of OUT and DIR signals are used to transmit the pulse train and to indicate the direction. The OUT and DIR signals can also be programmed as CW and CCW signal pairs. In this section, the electrical characteristics of the OUT and DIR signals are detailed. Each signal consists of a pair of differential signals. For example, OUT2 consists of OUT2+ and OUT2- signals.

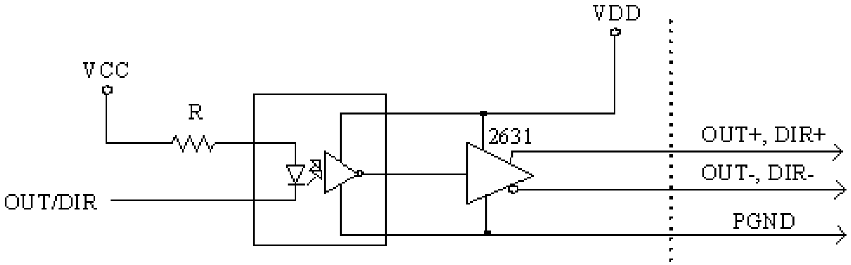


Figure 3-14: Wiring Diagram for OUT and DIR Signals

Warning: The sink current must not exceed 20mA or the 2631 will be damaged!

Non-differential type wiring example:

Choose either OUT/DIR+ and OUT/DIR- to connect to driver's OUT/DIR

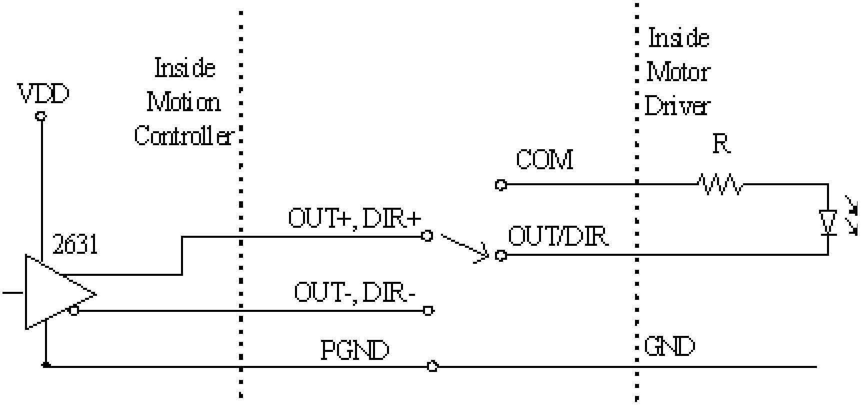


Figure 3-15: OUT/DIR SIgnal Selection

Notice that users can choose one pair of OUT/DIR from SSCNET board to connect driver's OUT/DIR. For example: Choose (+) pair, then OUT+ must be connected to driver's OUT pin and DIR+ must be connected to driver's DIR pin. Of course, OUT- and DIR- are useless. You can ignore it.

4 Operation Theory

This chapter describes the detail operation of the SSCNET board card

4.1 Architecture

4.1.1 HOST PC and SSCNET Board

The communication between the host PC and the SSCNET board is through a 16Kbyte Dual Port RAM that is integrated inside the SSCNET board. Both the Host CPU and DSP can read/write on it.

For the hardware level, the SSCNET board is a small microcomputer. It has its own processor (the DSP), address and data bus, its data memory, and peripherals use for SSCNet communication protocol. Thus, the host CPU does not pay any attention on the DSP. Only when the application program requests information or sends command motions to the SSCNet control board, the host PC needs to perform the read/write functions to the DPRAM.

4.1.2 SSCNet Communication

The SSCNET board controls servomotors through the SSCNet communication. The communication is a master-slave architecture. Every 0.888 ms, the SSCNET board (master) sends a command, in which the position command is involved, to each servo driver (the slave) and in return the servo drivers report back to the SSCNET board, providing the SSCNET board with information about its position, velocity, and other specified servo data.

- Communication is synchronous within the control cycle of the controller and the sending/receiving is executed every control cycle with CRC check.
- Broadcast transmission is conducted from the SSCNET board to the servo amplifier
- Transmission from the servo driver to the SSCNET board is conducted through a time-division system, and SSCNET board reads data in a batch format.

4.2 Frame Architecture

In this section, the frame architecture, which is the basis of all motion functions, is described.

4.2.1 Frame Introduction

A frame is a mathematical description of a piece of motion trajectory. When user gives a motion command, for example: start_sr_move(), the motion command will be translated into several frames. Each frame represents some pieces of the whole motion trajectory. Then, the frame data is downloaded to the SSCNET board.

As mentioned in previous sections, the SSCNET board is equipped with a DSP. It is in charge of calculating the frame data, so that the original motion trajectory information can be retrieved.

This is an example to illustrate how a frame works.

Suppose a user want an axis to move 10mm in distance. The acceleration time is 0.5 sec, deceleration time is 0.2 sec, and maximum velocity is 5mm/sec. So, he would call the function start_tr_move() function and provide the correct parameters in his application program. The library then splits the motion command into several frames, and downloads these frames to the SSCNET board. See flow chart below:

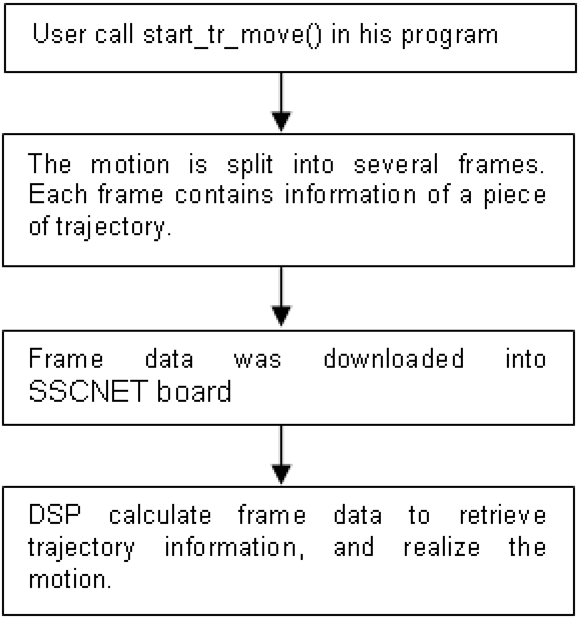

User call start_tr_move() in his program
The motion is split into several frames. Each frame contains information of a piece of trajectory.
Frame data was downloaded into SSCNET board
DSP calculate frame data to retrieve trajectory information, and realize the motion.

Figure 4-1: Frame Flowchart

Example of start_tr_move:

[Step 1]:

User calls start_tr_move(0, 10.0, 0, 5.0, 0, 0.5, 0.3) in his program.

The meaning of each parameter:

Axis No = 0, Dist = 10.0 mm, Stat velocity = 0,

Maximum velocity = 5.0 mm/sec, Final velocity = 0, Tacc = 0.5, Tdec = 0.3

[Step 2]:

DLL function start_tr_move() is invoked to solve the frames of this motion command. Assumes that the absolution position before start_tr_move is ‘0’. Start_tr_move will be disassembled into 3 frames.

- (1)X(t) = 10 * t^2 , t = 0 ~ 0.5
- (2)X(t) = 1.25 + 5 * t , t = 0 ~ 1.6
- (3)X(t) = 9.25 + 5*t - 16.666667 * t^2 , t = 0 ~ 0.3

[Step 3]:

Download frame data to the SSCNET board.

	t0	t1	t2	t3	Period
(1)	0	0	10	0	0.5
(2)	1.25	5	0	0	1.6
(3)	9.25	5	-16.667	0	0.3

Table 4-1: start_tr_move Data Table

[Step 4]:

The DSP of the SSCNET board calculates the frame data to obtain the trajectory information.

4.3 Single Motion

In this section, single motion functions are discussed. Single motion means the motion is commanded by one function call only. For example, start_sr_move(), this function will allow an axis to move a certain distance with a specified speed and accel/decel time.

Single motion functions can be categorized into the following types according to their functionality.

4.3.1 Single axis velocity motion

In this section, the following functions are discussed.

```
tv_move(Axis, StrVel, MaxVel, Tacc)
sv_move(Axis, StrVel, MaxVel, Tacc, Tlacc)
```

The single axis velocity motion function will allow the axis to accelerate from a starting velocity, ‘StrVel’, to a specified constant velocity, ’MaxVel’. The axis will continue to travel at this constant velocity until the velocity is changed by inserting the function tv_change(), sv_change() or stopped by the functions tv_stop(), sv_stop(), emg_stop().

Two kinds of acceleration method are available. By using tv_move(), the acceleration is constant as shown in the in left diagram below. By using sv_move(), the derivative of acceleration, the 'jerk', is a constant as Illustrated in the right diagram below.

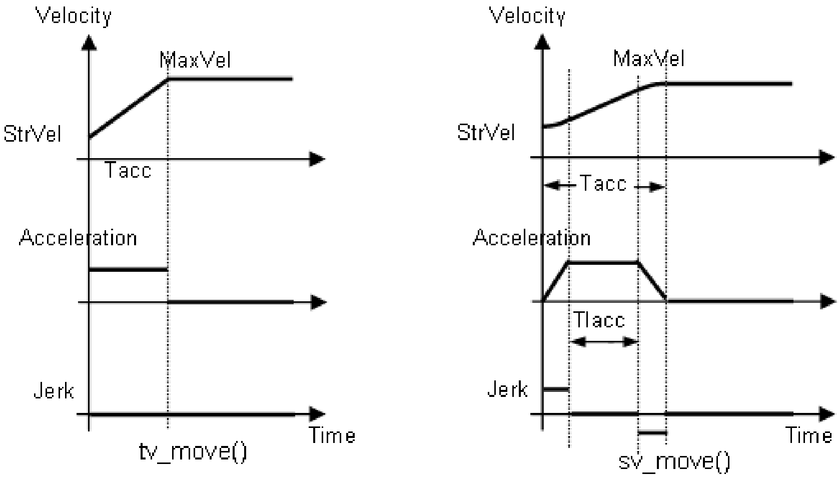


Figure 4-2: Constant Jerk Graph

4.3.2 Single axis P to P motion

In this section, the following functions are discussed.

```
start_tr_move(Axis, Dist, StrVel, MaxVel, FinVel,
      Tacc, Tdec)
start_sr_move(Axis, Dist, StrVel, MaxVel, FinVel,
      Tacc, Tdec, Tlacc, Tldec)
start_ta_move(Axis, Pos, StrVel, MaxVel, FinVel,
      Tacc, Tdec)
start_sa_move(Axis, Pos, StrVel, MaxVel, FinVel,
      Tacc, Tdec, Tlacc, Tldec)
```

Single axis P-to-P motion functions will allow the axis to move a specified distance or move to a specified position. The first four functions are pretty straightforward. 't', 'r', 's' and 'a' characterizes the function and provides information about the velocity profile and position method to achieve the target position.

- 't:' The velocity profile is 'Trapezoidal'. That is the acceleration and deceleration is a constant (shown in left diagram).
- 's:' The velocity profile is 'S-Curve'. This is a derivative of acceleration, 'jerk', and is a constant (shown in right diagram).
- 'r:' The axis moves a distance 'Relative' from a specified point. Specified by parameter 'Dist'.
- 'a:' The axis moves to an 'Absolute' position regardless of its current position. It is specified by the parameter 'Pos'.

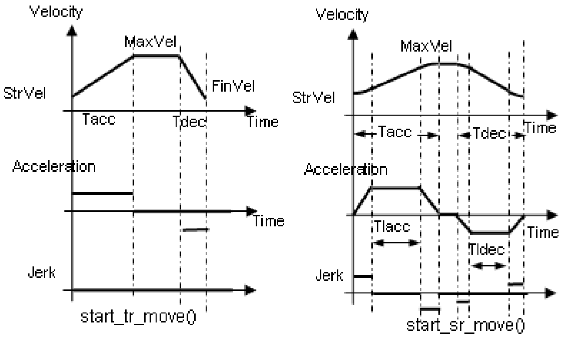
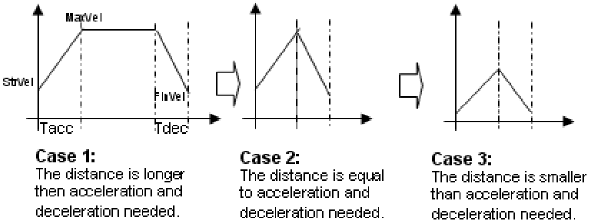


Figure 4-3: Single Axis Motion

The distance moved during acceleration and deceleration can be calculated using the following formula. (For both trapezoidal and S-curve profiles)

```
Dist_acc = 0.5 * (StrVel + MaxVel) * Tacc
Dist_dec = 0.5 * (FinVel + MaxVel) * Tdec
```

In some cases, the distance moved may not be long enough. For example, the 'Dist ' in start_tr_move() is too small or 'Pos' in start_sa_move() is too close to the current position. These 4 function calls mentioned above automatically slows down the velocity. The change in the velocity profile is illustrated in the diagram below.



Case 1: The distance is longer then acceleration and deceleration needed.

Case 2: The distance is equal to acceleration and deceleration needed.

Case 3: The distance is smaller than acceleration and deceleration needed.

Figure 4-4: Motion Function Graphs

Case 1 to case 2: The constant velocity period is reduced while the Tacc, Tdec, StrVel, MaxVel and FinVel remain unchanged.

Case 2 to case 3: The constant velocity period vanished, and, StrVel, MaxVel and FinVel become smaller according to the ratio described below. While the Tacc and Tdec remain unchanged.

```
New_StrVel = K * Original_StrVel
New_MaxVel= K * Original_MaxVel
New_FinVel = K * Original_FinVel
Where K = Dist/(Distacc_needed + Distdec_needed)
= Dist/(Distjust_case)
```

4.3.3 Multi axes velocity motion

In this section, the following functions are discussed.

```
tv_move_all(Length, *Axis, *StrVel, *MaxVel,
     *Tacc)
sv_move_all(Length, *Axis, *StrVel, *MaxVel,
     *Tacc, *Tlacc)
```

Multi axes velocity motion has exactly the same functionality as a single axis velocity motion except that multi axes velocity motion can be applied to 2 or more axes simultaneously with all applied axes beginning to move at the same time, and according to each axis's setting, each axis will move to its constant velocity as specified.

The parameter 'Length' is used to indicate how many axes will be involved. The axes' numbers are stored in '*Axis', start velocity in '*StrVel', maximum velocity in '*MaxVel', Tacc in '*Tacc'.

Note: 1.Each axis runs independently. Thus, a stop function for each axis must be issued separately.

2.All axes must be on the same card.

4.3.4 Multi axes P to P motion

In this section, the following functions are discussed.

```
start_tr_move_all(Length, *Axis, *Dist, *StrVel,
     *MaxVel, *FinVel, *Tacc, *Tdec)
start_sr_move_all(Length, *Axis, *Dist, *StrVel,
     *MaxVel, *FinVel, *Tacc, *Tdec, *Tlacc,
     *Tldec)
start_ta_move_all(Length, *Axis, *Pos, *StrVel,
     *MaxVel, *FinVel,  *Tacc, *Tdec)
start_sa_move_all(Length, *Axis, *Pos, *StrVel,
     *MaxVel, *FinVel, *Tacc, *Tdec, *Tlacc,
     *Tldec)
```

Multi axes P-to-P motion has exactly the same functionality as single axis P-to-P motion except that multi axes P-to-P motion can be applied to 2 or more axes simultaneously with all applied axes beginning to move at the same time, and according to each axis's setting, each axis will move to its position or distance as specified

The parameter 'Length' is used to indicate how many axes will be involved. All motion parameters are passed to its function array just as single axis P to P motion.

Note: 1.Each axis runs independently. Thus, a stop function for each axis must be issued separately.

2.All axes must be on the same card.

4.3.5 Linear Interpolation

In this section, the following functions are discussed.

```
start_line_tr_move(Length, *AxisArray,
     *DistArray, StrVel, MaxVel, FinVel, Tacc,
     Tdec)
start_line_sr_move(Length, *AxisArray,
     *DistArray, StrVel, MaxVel, FinVel, Tacc,
     Tdec, Tlacc, Tldec)
start_line_ta_move(Length, *AxisArray, *PosArray,
     StrVel, MaxVel, FinVel, Tacc, Tdec)
```

```
start_line_sa_move(Length, *AxisArray, *PosArray,
      StrVel, MaxVel, FinVel, Tacc, Tdec, Tlacc,
      Tldec)
```

These four functions applies to any 2, any 3 or any 4 of the 12 axes in one card, so that these axes can “start simultaneously, and reach their ending points at the same time” and the ratio of speed between these axes is a constant value.

2-Axis Linear Interpolation

As in the diagram below, 2 axes linear interpolation means to move the XY (or any 2 of the 4 axis) position from P0 to P1. The 2 axes start and stop simultaneously, and the path is a straight line.

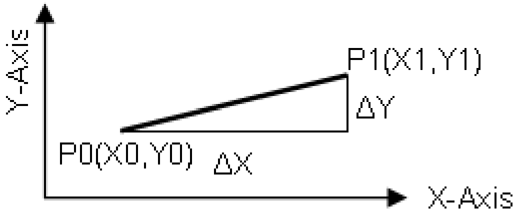


Figure 4-5: 2-Axis Linear Interpolation

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

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

When calling the 2 axes linear interpolation functions, it is the vector speed to define the start velocity, StrVel, and maximum velocity, MaxVel, Both trapezoidal and S-curve profile are available.

For example:

```
Axis[0] = 0;      Axis[1] = 2 ;      Dist[0] = 30
      ;      Dist [1] = 40
```

```
start_line_tr_move(2, Axis, Dist, 10.0, 50.0,
     15.0, 0.3, 0.2)
```

This cause the two axes (axes 0 & 2) to perform a linear interpolation movement, in which:

```
ΔX = 30 mm
ΔY = 40 mm
Start vector speed=10mm/sec, X speed=6mm/sec, Y
     speed = 8mm/sec
Max. vector speed=50mm/sec, X speed=30mm/sec,Y
     speed=40mm/sec
Final vector speed=15mm/sec, X speed=9mm/sec, Y
     speed =12mm/sec
Acceleration time = 0.3 sec
Deceleration time = 0.2 sec
```

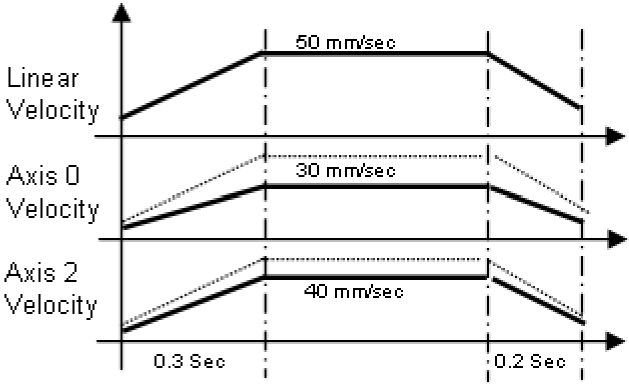


Figure 4-6: 2-Axis Linear Interpolation Example

3-Axis Linear Interpolation

Any 3 of the 12 axes of SSCNET board may perform 3 axes linear interpolation. As the figure below, 3 axes linear interpolation means to move the XYZ (if axes 0, 1, 2 are selected and assigned to be X, Y, Z respectively) position from P0 to P1 and start and stop simultaneously. The path is a straight line in space.

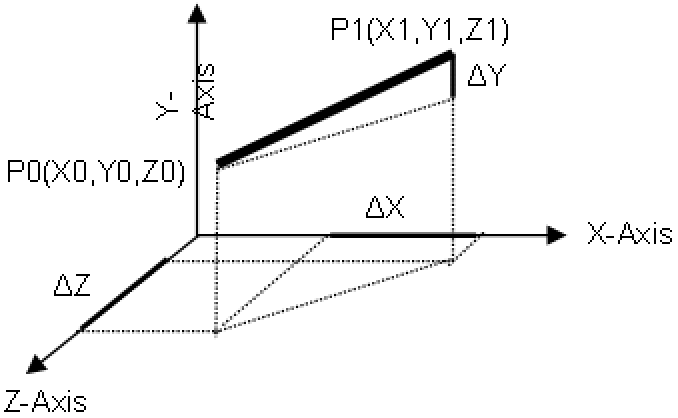


Figure 4-7: 3-Axis Linear Interpolation

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

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

When calling those 3 axes linear interpolation functions, it is the vector speed, which defines the start velocity, ‘StrVel’, maximum velocity, ‘MaxVel’ and final velocity ‘FinVel’. Both trapezoidal and S-curve profile are available.

For example:

```
Axis[0] = 0;  Axis[1] = 1;   Axis[2] = 2 ;
Dist[0] = 10; Dist[0] = 20;   Dist [1] = 30
start_line_tr_move(3, Axis, Dist, 10.0, 50.0,
     10.0, 0.3, 0.2)
```

This causes the two axes (axes 0 & 2) to perform a linear interpolation movement, in which:

```
ΔX = 10 mm
ΔY = 20 mm
```

```
ΔZ = 30 mm
Start vector speed=10mm/sec
X spped= 10/  = 2.67 mm/sec
Y spped= 2*10/  = 5.33 mm/sec
z spped= 3*10/  = 8.01 mm/sec
Max. vector speed = 50mm/sec
X spped= 50/  = 13.36 mm/sec
Y spped = 2*50/  = 26.72 mm/sec
z spped = 3*50/  = 40.08 mm/sec
Final speed=10mm/sec
X spped= 10/  = 2.67 mm/sec
Y spped = 2*10/  = 5.33 mm/sec
z spped = 3*100/  = 8.01 mm/sec
Acceleration time = 0.3 sec
Deceleration time = 0.2 sec
```

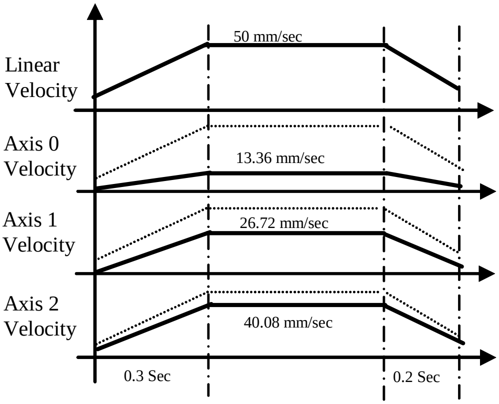


Figure 4-8: 3-Axis Linear Interpolation Example

4-Axis Linear Interpolation

In 4 axes linear interpolation, the speed ratio along X-axis, Y-axis, Z-axis and U-axis Is (ΔX: ΔY: ΔZ: ΔU), respectively, and the vector speed is:

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

Note: 1. Each axis runs independently. Thus, a stop function for each axis must be issued separately.

2. All axes must be of the same card

4.3.6 Circular Interpolation

Any 2 of the 12 axes of SSCNET board can perform circular interpolation. As the example below, the circular interpolation means XY (if axes 0, 1 are selected and assigned to be X, Y respectively) axes simultaneously start from initial point, (0,0) and stop at end point,(1800,600). The path between them is an arc, and the Max-Vel is the tangential speed

For example:

```
Axis[0] = 0;      Axis[1] = 2 ;      Dist[0] =
     1000 ;      Dist [1] = 0
start_arc_tr_move(2, Axis, Dist, -143.1, 10.0,
     50.0, 15.0, 0.1, 0.2)
This causes the two axes (axes 0 & 2) to perform
     a circular interpolation movement, in which:
Center distance X = 1000 mm
Center distance Y = 0 mm
Moving angle = -143 degree
Start vector speed=10mm/sec
Max. vector speed=50mm/sec
Final vector speed=15mm/sec
Acceleration time = 0.1 sec
Deceleration time = 0.2 sec
```

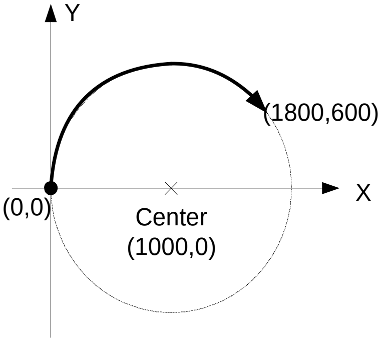


Figure 4-9: Circular interpolation

To specify a circular interpolation path, the following parameters must be clearly defined.

Center point: The coordinate of the center of arc (In absolute mode) or the off_set distance to the center of arc (In relative mode)

Angle: The moving angle, either clockwise (-) or counter clockwise (+)

4.3.7 Change Velocity on the Fly

In this section, the following functions are discussed.

```
tv_stop(Axis, Tdec)
sv_stop(Axis, Tdec)
emg_stop(Axis)
tv_change(Axis, SpeedFactor, Tacc)
sv_change(Axis, SpeedFactor, Tacc,)
```

The first three functions are used to stop a moving axis. The last two are used to adjust the moving speed of an axis. tv_stop() function stops the specified ‘Axis’ with a deceleration time period ,’Tdec’, and a “Trapezoidal” velocity profile during deceleration. See diagram below.

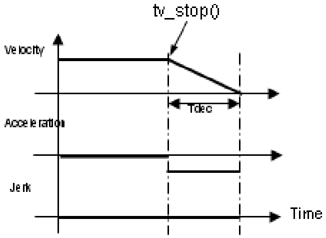


Figure 4-10: Stop a Moving Axis

The sv_stop() function stops a specified ‘Axis’ with deceleration time period, ’Tdec’, and a “S-Curve” velocity profile during deceleration. See diagram below.

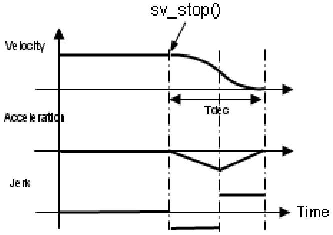


Figure 4-11: Stop with Deceleration

The emg_stop() function stops the a specified ‘Axis’ immediately without deceleration. See diagram below.

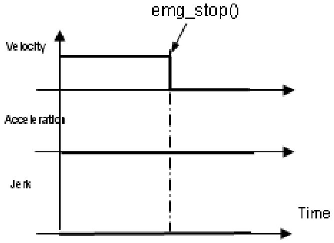


Figure 4-12: Immediate Stop

The tv_change() function changes the moving speed of a specified ‘Axis’ with acceleration time period, ’Tacc’, and a ‘Trapezoidal’ velocity profile during acceleration. The second parameter ‘SpeedFactor’ is used to define the new speed. For example, if the specified axis start its motion using tv_move() function and the ‘MaxVel’ is set to be 10 mm/sec. Then tv_change() is applied with ‘SpeedFactor’ = 1.5. The new speed is 1.5 * 10 = 15 mm/sec. As was shown below.

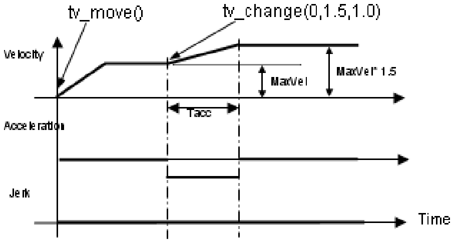


Figure 4-13: Moving Change

The sv_change() function changes the moving speed of a specified ‘Axis’ with acceleration time period, ’Tacc’, with a “S-Curve” velocity profile during acceleration.

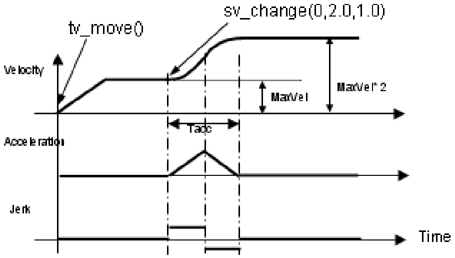


Figure 4-14: Change with S-Curve Velocity

If a second tv_change() or sv_change() function is applied, then ‘SpeedFactor’ will refer to the original ‘MaxVel’, ie, the original maximum velocity defined at the beginning of the motion function.

Note: 1. All change speed on the fly function calls can be applied any time when an axis is moving, no matter which function started its motion.

2. tv_change(), sv_change() with 'SpeedFactor' = 0 doesn't have the same affect as tv_stop() , sv_stop(). For tv_stop(), sv_stop() will complete its motion, while tv_change(), sv_change() will set speed to zero.

4.3.8 Position Compensation on the Fly

In this section, the following function is discussed.

```
set_position_compensate(I16 axis, F64
    Compen_value);
```

This function can be used to change the target position when an axis is commanded by the following single axis P-to-P motion functions:

```
start_tr_move() , start_ta_move()
start_sr_move() , start_sa_move()
```

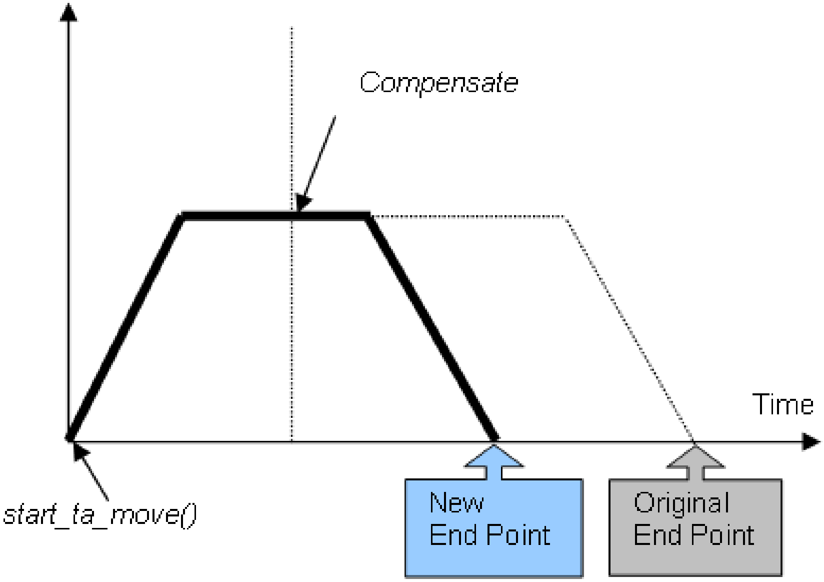


Figure 4-15: Position Compensation on the Fly

Theory of position compensation

This function is to change the target position defined originally by the previous motion functions. After changing position, the axis will move to the new target position and totally forget the original position. This operation can only be applied on the constant velocity section. Acceleration and deceleration section is not allowed for this function. The acceleration and deceleration rate, and StrVel and MaxVel are kept the same as the original setting

Constrains of position compensation :

1. It is applicable only after start_tr_move(), start_ta_move() start_sr_move(), and start_sa_move() functions. The moving distance must be long enough so that ‘MaxVel’ can be achieved.
2. It will not work if it is applied after the axis has entered the deceleration region.
3. The rest distance must be long enough for minus position compensation. The reset distance must be larger than deceleration section.

For example:

A trapezoidal absolute motion is applied:

```
start_ta_move(0, 100, 0, 10, 0, 0.5, 1).
```

This causes axis 0 to move to position 100mm, and the maximum velocity is 10 mm/sec. The necessary number of pulses to accelerate is 0.5*10*0.5 = 2.5mm. The necessary number of pulses to decelerate is 0.5*10*1 = 5 mm. Total distance is 100mm and it is larger than the summation of acceleration and deceleration distance. That means it can reach the maximum velocity. As for the deceleration distance is 5mm, the minus compensative command must be issued 5mm in advanced before end.

Refer to the following table. At position “AppliedPos” the set_position_compensate (0, Compen_Value) is applied.

Compen_Value	AppliedPos	Final Position	Note
5	10	105	OK

Table 4-2: set_position_compensate Values

Compen_Value	AppliedPos	Final Position	Note
15	90	115	OK
-5	95	95	OK
-5	96	100	Not allowed

Table 4-2: set_position_compensate Values

4.4 Home move

In this section, the following functions are discussed.

```
set_home_mode(Axis, HomeMode)
home_move(Axis, StartVel, MaxVel, FinVel, Tacc)
```

After configuring set_home_mode(), user may use the home_move() function to command the axis to start returning home. The ‘StrVel’ defines the starting velocity, the ‘Tacc’ define the acceleration time and the axis continues traveling at the constant velocity until it reaches the ORG switch.

Note: The sign of ‘MaxVel’ defines the moving direction, while the sign of ‘StrVel’ and ‘FinVel’ is meaningless. User must carefully define the direction of the home return motion, so that the axis can find the ORG switch correctly.

Mode 0: ORG only

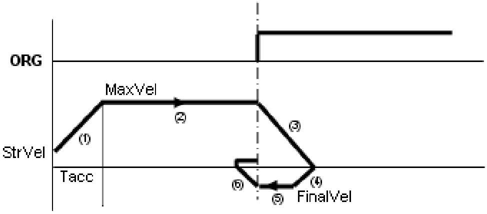


Figure 4-16: Mode 0 Home

1. Accelerate from StrVel to MaxVel.
2. Travel with constant velocity ‘MaxVel’ until ORG turns ON.
3. Slow done to stop.
4. Return and accelerate to ‘FinVel’.
5. Travel with constant velocity ‘FinVel’ until ORG turn Off.
6. Slow done to stop.
7. Searching ORG rising edge with velocity = 1 pulse per SSCNet cycle time until ORG turn ON, then stop and fin-Continuous Motion

In this section, the operation of continuous motion is introduced. To apply continuous motion function user must first construct the trajectory.

The procedures of constructing a continuous motion trajectory includes:

- Declaration for beginning of motion list
- Add Trajectory pieces
- Declare end of motion list

Beside on-line construction of the motion trajectory in the application program, an off-line method using the “Trajectory generator” and then save it to file, and using the ‘load_trajectory_file()’ function to load the motion trajectory will produce the same result.

- Load Trajectory file

After constructing the motion trajectory, user should be able to apply it to the continuous motion operation.

- Start/Stop command

4.4.1 Declaration for Beginning of Motion List

In this section, the following function is discussed.

```
start_motion_list(Length,  *AxisArray)
```

This function is used to declare the variables for the motion list describing a continuous motion trajectory. After the declaration for start_motion_list(), user can call the functions discussed in next section to piece-wisely extend the trajectory.

start_motion_list() automatically checks whether the previous motion list is finished or not. If the previous list is not completed it will return an error.

The first parameter ‘**Length**’ defines the total number of axes that will be involved in the continuous motion. The second parameter ‘*AxisArray’ is an array, and each array element stores the axis No.

For example:

If axis 0 and axis 5 are to perform a continuous motion, then the command line would be: start_motion_list(2, {0,5}) in the program.

If axis 0, axis 1 and axis 5 are to perform a continuous motion, then the command line would be: start_motion_list(3, {0,1,5}) in the program.

Note that all specified axis no. must be of the same card. And the ‘Length’ must not exceed ‘4’.

4.4.2 Add Trajectory pieces

In this section, the following functions are introduced.

```
add_line_tr_move(*DistArray, StrVel, MaxVel,
     FinVel, Tacc, Tdec)
add_line_sr_move(*DistArray, StrVel, MaxVel,
     FinVel, Tacc, Tdec, Tlacc, Tldec)
add_line_ta_move(*PosArray, StrVel, MaxVel,
     FinVel, Tacc, Tdec)
add_line_sa_move(*PosArray, StrVel, MaxVel,
     FinVel, Tacc, Tdec, Tlacc, Tldec)
add_arc_tr_move(*CenterArray, Angle, StrVel,
     MaxVel, FinVel, Tacc, Tdec)
add_arc_sr_move(*CenterArray, Angle, StrVel,
     MaxVel, FinVel, Tacc, Tdec, Tlacc, Tldec)
add_arc_ta_move(*CenterArray, Angle, StrVel,
     MaxVel, FinVel, Tacc, Tdec)
add_arc_sa_move(*CenterArray, Angle, StrVel,
     MaxVel, FinVel, Tacc, Tdec, Tlacc, Tldec)
```

```
add_arc2_sa_move(*AxisArray, *CenterPosArray,
     Angle, StrVel, MaxVel, FinVel, Tacc, Tlacc,
     Tdec, Tldec)
add_arc2_sr_move(*AxisArray, *CenterDistArray,
     Angle, StrVel, MaxVel, FinVel, Tacc, Tlacc,
     Tdec, Tldec)
add_arc2_ta_move(*AxisArray ,*CenterPosArray,
     Angle, StrVel, MaxVel, FinVel, Tacc, Tdec)
add_arc2_tr_move(*AxisArray, *CenterDistArray,
     Angle, StrVel, MaxVel, FinVel, Tacc, Tdec)
add_dwell(Sec)
smooth_enable(Flag, R)
```

These functions are used to construct a continuous motion trajectory. After declaration for the motion list, user can call these functions. Each function represents a piece of motion trajectory.

The functions could be categorized into four types of trajectories, and any combinations of added trajectory functions are possible.

```
Line: add_line_XX_move
Arc: add_arc_XX_move, add_arc2_XX_move
Dwell: add_dwell
Smooth: smooth_enable
```

Adding a line trajectory

When any of the following four functions are executed:

```
add_line_tr_move(),add_line_sr_move()
dd_line_ta_move(), add_line_sa_move()
```

A straight line will be added into the continuous motion trajectory. The parameter definitions of this added line functions the same as those in a single motion linear interpolation. The following is an example:

For example:

(Suppose both axes 0 & 2 are at position ‘0’)

```
start_motion_list(2, {0,2} )
add_line_tr_move({100,0}, 0, 100, 100, 1.0, 0)
add_line_sr_move({60,80}, 100, 100, 80, 0, 1, 0,
     0)
add_line_ta_move({320,110}, 80, 100, 100, 0.5, 0)
add_line_sa_move({320,160} 100, 100, 0, 0, 1.0,
     0, 0.5)
```

```
end_motion_list()
```

The resulting 2-D trajectory is:

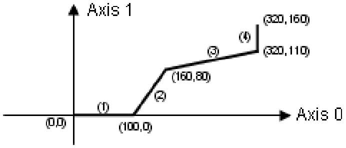


Figure 4-17: Example 2-D Trajectory

Adding an arc trajectory

When any of the following 8 functions are executed

```
add_arc_tr_move(), add_arc_sr_move()
add_arc_ta_move(), add_arc_sa_move()
add_arc2_tr_move(), add_arc2_sr_move()
add_arc2_ta_move(), add_arc2_sa_move()
```

A 2-D arc will be added to the continuous motion trajectory. The first 4 are used after the start_motion_list() function with ‘Length’=2, and the parameter definitions of this added arc functions the same as those of the single motion circular interpolation. The last 4 are used after the start_motion_list() function with ‘Length’ > 2. The additional parameter AxisArray defines the 2 axes that this arc belongs to. The following are 2 examples:

Example 1:

(Suppose both axes 0 & 2 are at position ‘0’)

```
start_motion_list(2, {0,2} )
add_line_ta_move({100,0}, 0 , 100, 100, 1.0, 0)
add_arc_ta_move({100,50}, -180, 100 , 100 , 100,
     0 , 0)
add_line_tr_move({-100,0}, 100, 100, 100, 0 , 0)
add_arc_sr_move({0,-50} ,-180, 100 , 100, 0 ,0,
     1.0 , 0 ,0)
end_motion_list()
```

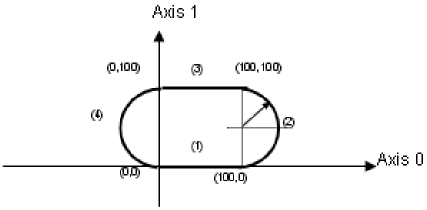


Figure 4-18: Example 1 - Arc Trajectory

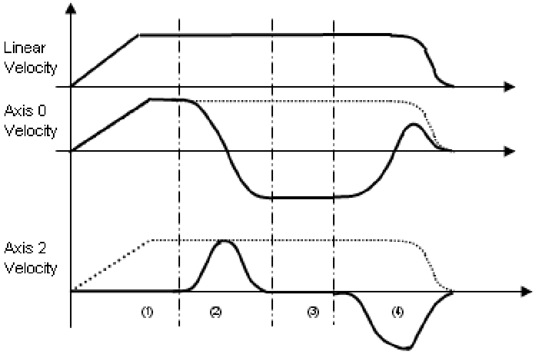


Figure 4-19: Velocity vs. Time

Example 2:

(Suppose both axes 0 ,1 & 2 are at position ‘0’)

```
start_motion_list(3, {0,1,2} )
add_line_ta_move({100,0,0}, 0 , 100, 100, 1.0, 0)
add_arc2_tr_move({1,2}, {0,50}, 180, 100 , 100 ,
     100, 0 , 0)
add_line_ta_move({0,0,0}, 100, 100, 0, 0 , 1)
end_motion_list()
```

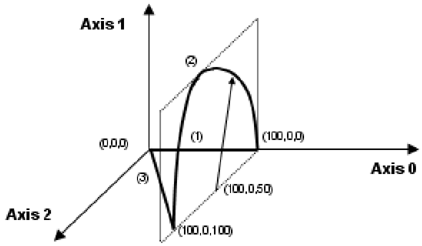


Figure 4-20: Example 2 - Arc Trajectory

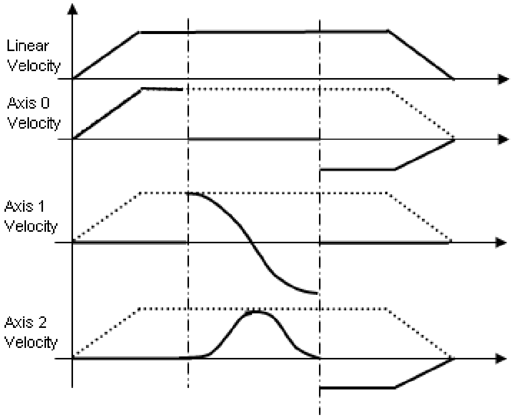


Figure 4-21: Velocity vs. Time

Add dwell

When add_dwell() function is executed, the motion will be freezed for a specified period of time, definied by parameter ‘Sec‘ in unit of second. The following is an example:

Example:

(Suppose both axes 0 & 2 are at position ‘0’)

```
start_motion_list(2, {0,2} )
add_line_tr_move({100,0}, 0 , 100, 0, 1.0,1.0)
```

```
add_dwell(0.5)
add_line_sr_move({60,80}, 0 , 150 , 0 , 0.5 , 0.5
     , 0 , 0)
add_dwell(1.0)
add_line_ta_move({320,110}, 80 , 100, 100, 0.5,
     0)
end_motion_list()
```

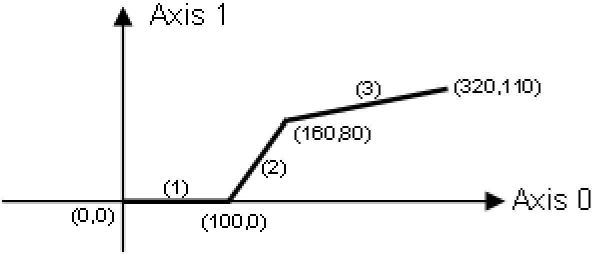


Figure 4-22: Adding Dwell Example

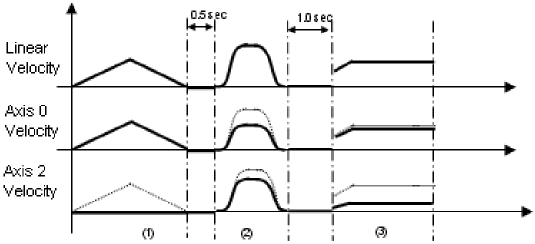


Figure 4-23: Velocity vs. Time

Smoothing trajectory

When smooth_enable() functions is executed, the motion trajectory thereafter will be “rounded”. The following figures show the rounding cases.

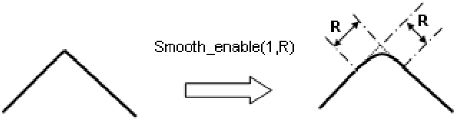


Figure 4-24: Line & Line

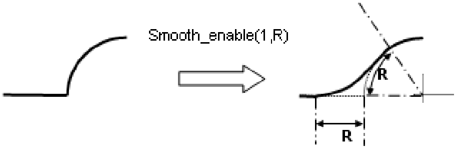


Figure 4-25: Line & Arc

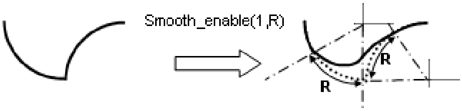


Figure 4-26: Arc & Arc

smooth_enable() function could be executed second or more rimes in order to disable smoothing or to change smoothing radius ‘R’ as program’s need.

For example:

(Suppose both axes 0 & 2 are at position ‘0’)

```
start_motion_list(2, {0,2} )
add_line_tr_move({100,0}, 0 , 100, 100, 1.0, 0)
add_line_sr_move({60,80}, 100 , 100 , 100 , 0 , 0
     , 0 , 0)
smooth_enable(1,50)
add_line_ta_move({320,110}, 100 , 100, 100, 0, 0)
smooth_enable(1,20)
add_line_sa_move({320,160} 100 , 100, 0 ,0, 1.0 ,
     0 , 0.5)
```

```
end_motion_list()
```

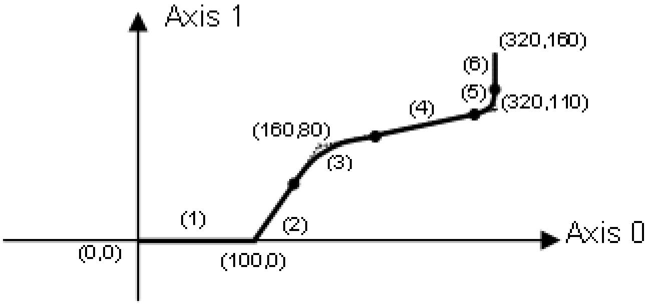


Figure 4-27: Smoothing Example

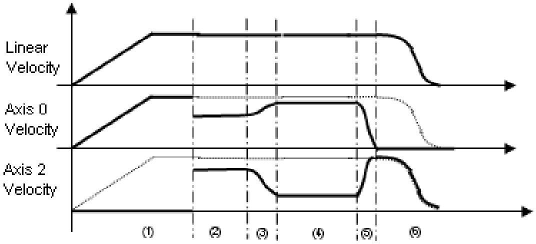


Figure 4-28: Velocity vs. Time

Note: 1. Smoothing is also applicable for 3D and 4D.

2. The smoothing trajectory guarantees continuous velocity and acceleration at the smoothing point.

4.4.3 Declaration for End of Motion List

In this section, the following function is discussed.

```
end_motion_list()
```

This function is used to declare the variables for the end of motion list, describing a continuous motion trajectory. After adding trajectories piece-wisely using the function calls discussed above, end_motion_list() must be called, so that the motion trajectory can be translated into frame data such that the SSCNET board can understand.

This function takes no parameters.

4.4.4 Start/Stop command

In this section, the following functions are discussed.

```
start_cont_move(Void)
stop_cont_move(Void)
```

After building a trajectory by either on-line (start/end motion list) or off-line (load trajectory file) method, user can call the start_cont_move() function to perform the continuous motion trajectory.

If the user has program a second trajectory by either on-line (start/end motion list) or off-line (load trajectory file) method, the previous trajectory will be erased and cannot be retrieved except to reconstruct it again. The user must be careful with this, especially when multiple threading programs are implemented.

For example:

(Suppose both axes 0 & 2 are at position '0')

```
start_motion_list(2, {0,2} )
add_line_tr_move({100,0}, 0 , 100, 0, 1.0,1.0)
add_dwell(0.5)
add_line_sr_move({60,80}, 0 , 150 , 0 , 0.5 , 0.5
     , 0 , 0)
add_dwell(1.0)
add_line_ta_move({320,110}, 80 , 100, 100, 0.5,
     0)
end_motion_list()
start_cont_motion() (* motor start moving after
     this command*)
```

4.5 Motion Related IO

In addition to the SSCNet servo motor control capabilities, the SSCNET board has other I/O functions and can roughly be divided into 2 categories. They are the motion related I/O's and the general purposed I/O's. Motion related I/O's are input and output signals dedicated to motion. For example: PEL/MEL, position/velocity feedback…etc. This section will concentrate on the motion related I/O and their function calls.

4.5.1 Position control and feedback

In this section, the following functions are discussed.

```
get_position(Axis,*Pos_F, *Pos_C)
set_position(Axis,Pos)
get_target_pos(Axis,*TargetPos)
get_move_ratio(Axis, *PulsePerMM)
set_move_ratio(Axis, PulsePerMM)
```

Get position information

The SSCNET board controls servo drivers & motors via an SSCNet protocol. For each SSCNet cycle (0.888ms), the SSCNET board sends a command to and receives a response from the servo driver. Through command and response, an abundant amount of information is carried in and out, including position command and position feedback. The function call get_position() will retrieve such information.

The parameter '*Pos_F' retrieves the current position feedback, which is reported from the servo driver through SSCNet communication.

The parameter '*Pos_C' retrieves the current position command, which is calculated on each SSCNet cycle (0.888 ms) by the DSP. The command position is sent to the servo driver on each SSCNet cycle, and the servo drivers will guild its motor to this position.

Set position

The set_position() function allows users to set a current position counter value for the servo driver.

Get target position information

The target position is a software maintained variable, which is updated each time a new motion command is executed. This recorder value is the position where the servomotor will stop at when the motion completed.

Since the target position is a software recorder for the motion end position, it doesn't work under the following conditions:

- Case 1: Velocity motion is applied, because velocity motion have no end position information.
- Case 2: emg_stop(), tv_stop() and sv_stop() are executed, because motion stop before motion completed.

Once it is executed, the get_target_position() value is meaningless unless a position relatived motion function is executed.

Move ratio control

"Move ratio" means "How many command pulses will let the axes move 1.0 mm". Refer to the figure below; a servomotor is used to drive the moving part through a geared mechanism.

1. If the resolution of the motor is 8000 pulses/round, and,
2. The resolution of the gear mechanism is 10 mm/round. (i.e., part moves 100 mm if motor turns one round).

Then the "move ratio" will be 8000/10 = 800 pulses/mm.

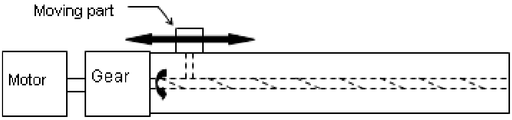


Figure 4-29: Move Ratio Control

All motion commands issued by the SSCNET board are in units of mm or mm/sec. Users need to set the "move ratio" value using set_move_ratio() according to the mechanical design.

If user want to check current "move ratio" value, get_move_ratio() function is helpful.

Note: If the set_cnt_to_axis() function is called to allow an encoder counter to work as a position feedback source, then the move ratio will refer to the pulses from the encoder counter rather than from the SSCNET motor driver.

4.5.2 Velocity Feedback

In this section, the following function is discussed.

```
get_velocity(Axis,*Vel_F, *Vel_C)
```

This function is used to retrieve the velocity information. Two velocity values can be retrieved.

- '*Vel_F': Feedback velocity, just as for position feedback, the SSCNET board receives the velocity feedback via SSCNet communication and is refreshes on each SSCNet cycle and is measured by the servo driver.
- '*Vel_C': Command velocity is calculated by the DSP of the SSCNET board. For each SSCNet cycle, it is calculated again.

Notice that the speed resolution is one pulse per 0.888ms. It is about 1126pps

4.5.3 Motion DIO status

In this section, the following functions are discussed.

```
set_PEL_config(Axis, Logic, mode)
set_MEL_config(Axis, Logic, mode)
set_ORG_config(Axis, Logic)
set_EMG_config(CardID, Logic)
get_PEL_status(Axis, *status)
get_MEL_status(Axis, *status)
get_ORG_status(Axis, *status)
get_EMG_status(CardID, *status)
```

The “motion DIO” mentioned here refers to the motions dedicated to the digital I/O signals including PEL, MEL, ORG, and EMG. Each axis has its own motion DIO signal except EMG. All axes from a single card shares the same EMG signal.

End-limit signals

The end-limit signals are used to stop the axis when they are active. There are two possible stop modes, one is “stop immediately”, and, the other is “decelerate to StrVel then stop”. The parameter ‘mode’ in set_PEL_config(), set_MEL_config() are used to select the mode. You can use either an ‘a’ contact switch or a ‘b’ contact switch by setting the parameter ‘Logic’.

PEL signal indicates the end-limit in the positive (plus) direction. The MEL signal indicates the end-limit in the negative (minus) direction. When the axis is moving towards the positive direction, the axis will be stopped when the PEL signal becomes active, while the MEL signal is no affect in this case, and vise versa. When the PEL is active, only the negative (minus) direction motion is allowed.

The PEL/MEL signals can generate an IRQ, if the interrupt service routine is enabled. Refer to section 4.9.

The PEL/MEL status can be monitored through the software function get_PEL_status() and get_MEL_status().

ORG signal

The ORG signal is used, when the axis is operating under the home return mode. There 1 home return mode (refer to section 4.4) and only one can be selected by setting the 'HomeMode' argument in the software function: set_home_mode().

The logic polarity of the ORG signal is selectable using the parameter 'Logic' of set_ORG_config(). The ORG status can be monitored using the software function get_ORG_status().

EMG signal

Each SSCNET board has an EMG signal input. Whenever this EMG signal becomes active, all the axes control by in the card will stop moving immediately.

The EMG signal is capable of generating an IRQ if an interrupt service routine is enabled, refer to section 4.9.

The logic polarity of the EMG signal is selectable using the parameter 'Logic' of set_EMG_config(). The EMG status can be monitored using the software function get_EMG_status().

4.5.4 Software limit

In this section, the following functions are discussed.

```
set_soft_limit(Axis, PLimit, Mlimit, ON_OFF)
get_soft_limit(Axis, *PLimit, *Mlimit, *ON_OFF)
```

The SSCNET board provides 2 software limits for each axis, one for the positive and one for the negative direction. Software limits are extremely useful in protecting a user's mechanical system, as it can operate as a physical limit switch, when configured correctly.

The software limit works because the DSP of the SSCNET board compares the current feedback position with the setting of the software limit value every SSCNet cycle. Once the feedback position is over the software limit, it stops the axis just as the PEL/MEL signals would. set_soft_limit() is used to configure the software limit.

- '**PLimit**' is used for software limit values in the positive direction.
- '**MLimit**' is used for software limit values in the negative (or minus) direction.
- '**ON_OFF**' is used to enable/disable the software limit function.

Users can read back current software limit setting using the get_soft_limit() function.

4.5.5 Motion Status

In this section, the following functions are discussed.

```
axis_status(Axis,*AxisStatus)
motion_status(Axis, *MotionStatus)
```

Axis status

The function call axis_status() is used to retrieve the servo driver's control status information. The parameter 'Axis' applies to the specified axis.

- '**AxisStatus**' – the control status of servo driver.

Bit	Name	Value & Description
0	Not_In_Control	1: Axis not in control 0: Axis is in control
1	In_Servo_Alarm	1: Axis is in servo alarm 0: Axis is not in servo alarm
2	Not_Ready_ON	1: Axis not Ready ON 0: Axis is Ready ON

Table 4-3: Axis Status

Bit	Name	Value & Description
3	Not_Servo_ON	1: Axis not Servo ON 0: Axis is Servo ON

Table 4-3: Axis Status

Not_In_Control: (Bit 0)

When initializing the SSCNET board card, the on-board DSP commands the SSCNet controller IC to search all axes for SSCNet servo drivers. If successful, this bit will be set to ‘0’, which means this axis has an SSCNet servo driver to control. Otherwise, this bit will be set to ‘1’. And no motion function could be executed on this axis.

In_Servo_Alarm: (Bit 1)

If the servo driver is in servo alarm state, this bit will be set to ‘1’, and no motion function could be executed on this axis.

Not_Ready_ON: (Bit 2), Not_Servo_ON: (Bit 3)

The servo on and ready on status is controllable using the set_servo_on() function, refer to section 4.8.4.

Motion status

The function call motion_status() is used to retrieve the motion status information. It is successful only when the ‘AxisStatus’ in axis_status() is ‘0’. That is this axis must be in control, no alarm, and ready/servo on. Check if the return code of motion_status() is equals to ‘0’. The parameter ‘Axis’ applies to the specified axis only.

- ‘**MotionStatus**’ – the motion status.

Bit	Name	Description
0	Ready_for_Motion	Axis is not moving and it is available for another move command
1	In_Motion	Axis is moving and can’t accept another move command
2	In_Home_Move	Axis is in moving in home procedure and can’t accept another move command

Table 4-4: Motion Status

Bit	Name	Description
3	In_V_Change	After lauching velocity change command, this bit will be ON till the change is done
4	In_P_Change	After lauching position change command, this bit will be ON till the change is done
5	MEL_ON	Axis touches the positive limit switch
6	PEL_ON	Axis touches the negative limit switch
7	ORG_ON	Axis touched the origin switch
8	EMG_ON	Emergency input pin is ON
9	P_Soft_ON	Axis is reached the positive software limit
10	M_Soft_ON	Axis is reached the negative software limit
11	EZ_ON	Axis touched the external Index switch
12	Stop_cmd_end	After v_stop() command ends, this bit will be ON
13	Stop_cmd_running	This bit will be ON if users lauched a v_stop() command
14	Interlock_Pause	Once the axis is paused be interlock procedure, this bit will be ON

Table 4-4: Motion Status

4.5.6 Motion Input as General Input

In this section, the following functions are discussed.

```
set_mio_mode(CardID,DI_Channel, Mode);
get_MDI_status(I16 CardID, I16 MDI_Channel);
```

MDI_Channel

The range is from 0 to 35 (The corresponding pin on SP1 is MDI# minus one)

Mode

Mode=0 makes the motion input function(EL/ORG) active, Mode=1 makes motion input function inactive.

For example:

If users want to make Axis3's PEL/MEL no effect in motion, they can use

```
set_mio_mode(CARD0, 9, 1);   // PEL
set_mio_mode(CARD0, 10, 1);  // MEL
```

and the ORG remains its function in motion.

You can get the return code from get_MDI_status(CARD0, 9) and get_MDI_status(CARD0, 10) to read input status. If you don't set the mode to 1, you still can read the MDI status by this function.

4.6 General Purpose IO

General purpose I/Os are input and output signals that user can freely use. For example: encode counters, isolated DIO…etc. In this section, all general purposed I/Os and their function calls are discussed.

4.6.1 Encoder Counter

In this section, the following functions are discussed.

```
set_cnt_iptmode(CardID, EncNo, IptMode)
set_cnt_to_axis(CardID, EncNo, Axis, Resolution)
set_cnt_value(CardID, EncNo, Value)
get_cnt_value(CardID, EncNo, *Value)
```

Each PCI-8372+/8366+ has 3 encoder counters and can be use to receive A/B phase signals from a linear encoder. Also these counters can be programmed to receive CW/CCW, OUT/DIR type signals.

Input circuit

The input circuits for EA, EB and EZ signals are shown below

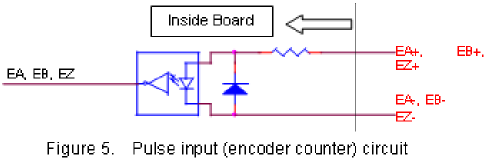


Figure 4-30: Pulse Input (Encoder Counter) Circuit

Note: The voltage across each differential pair of encoder input signals (EA+, EA-), (EB+, EB-) and (EZ+, EZ-) should be at least 3.5V or higher. Therefore, the output current must be observed when connecting to the encoder feedback or motor driver feedback as not to over drive the source.

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

Connection to Line Driver Output

To drive the SSCNET board encoder input, the driver output must provide at least 3.5V across the differential pairs with at least 6 mA driving capacity. The ground level of the two sides must also be tied together.

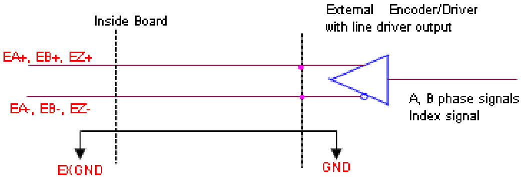


Figure 4-31: Line Driver Circuit

Connection to Open Collector Output

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

Encoder Power (VDD)	External Resistor R
+5V	0 Ω (None)
+12V	1.8kΩ
+24V	4.3kΩ

Table 4-5: Encoder Resistor

- +If=6mA max.

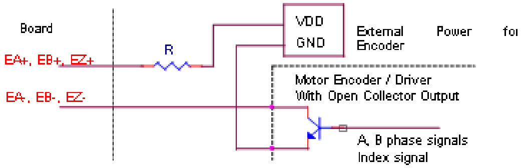


Figure 4-32: Open Collector Circuit

Configuring encoder counter

Each encoder counter can be configured to receive one of the following three types of signals using the function call set_cnt_iptmode().

1. A/B phase (Quadrature pulse signal)
2. CW/CCW (Dual pulses signal)
3. OUT/DIR (Single Pulse signal)

Set counter channel as position feedback of certain axis

The 3 general-purposed counters may work as position feedback source for each axis. The second parameter of the set_cnt_to_axis() function defines which counter is used, the third parameter defines which axis, and the last parameter declares the resolution of the counter in units of pulses. 'Resolution' is defined as the number of pulses counted by a counter when the SSCNet motor rotates one revolution. For example:

set_cnt_to_axis(0, 0, 1, 10000.0), this function will

1. Set counter '0' as position feedback source for axis '1'
2. Command the PCI-8372+ encoder to count 10000.0 pulses when motor goes one revolution.

A/B phase

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

A timing waveform is illustrated below.

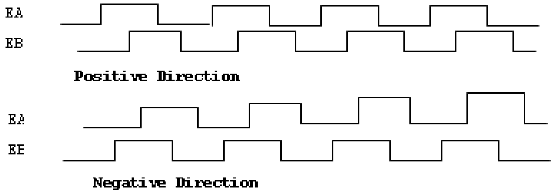


Figure 4-33: A/B Phase Timing

CW/CCW Mode

In this mode, the pulse from EA causes the counter to count up, while EB will cause the counter to count down.

OUT/DIR Mode

- In this mode, the pulse from EB decides on whether the counter should increase or decrease, whereas EA count the number of pulses.

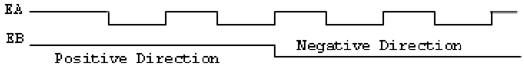


Figure 4-34: OUT/DIR Pulses

The index input (EZ) signal of the encoder is used as the “ZERO” index. This signal is common to most rotational motors. EZ can be used to define the absolute position of the mechanism. When a rising edge of EZ signal is received, it will clear the encoder counter value to ‘0’.

Counter value read/write

To read the encoder counter value, use the get_cnt_value() function. The parameter ‘*Value’ returns the counter value. To set the encoder counter value, use set_cnt_value(). The counter value will be set as the parameter ‘Value’.

4.6.2 DIO

In this section, the following functions are discussed.

```
get_di_status(CardID, ChNo, *Sts)
set_do_value(CardID, ChNo, Value)
```

Each PCI-8372+ board has 2-isolated digital output and 2 isolated digital input channels. Use the get_di_status() function to retrieve the current DI status, and set_do_value() to set the DO value.

4.6.3 DA

In this section, the following functions are discussed.

```
set_da_config(CardID, ChNo, Cfg)
set_da_value(CardID,ChNo,Value)
```

Each SSCNET board has 2 analog voltage output channels and can be independently configured using the set_da_config() function to set the DA to either be a direct DA output or for a velocity profile output. The default setting is DA direct output.

By using direct DA output, users can control the DA value using the set_da_value() function. While using velocity profile output will cause the DA output value to be proportional to the current command velocity. E.g. when a motor is rotating at 3000 rpm, the DA output is 10V and the DA output would be –10V if the rotating speed is -3000 rpm.

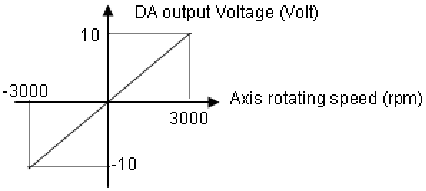


Figure 4-35: DA Output

4.6.4 AD

In this section, the following functions are discussed.

```
set_ad_function(CardID,Enable, AD_gain, AD_Last,
     AD2_src)
get_da_value(CardID,ChNo,*Value)
```

There are two analog input channels on the cPCI-8212H. It is used for sensing anlog output device ranged from –10V to +10V. Users can choose the gain by 1, 2, or 4. It means that the input voltage range could be +/-10V, +/-5V and +/- 2.5V for optimizing input resolution. There is an internal analog input channel which is for double checking. It is called AD2. The source of this channel could be choosed from internal +5V, ground, DA0, or DA1. It is very useful in debugging.

4.6.5 Analog channel auto calibration

In this section, the following functions are discussed.

```
tune_ref_5V(CardID, Value)
save_auto_k_value(CardID, Channel, Value)
get_auto_k_value(CardID, Channel, *Value)
tune_ad_offset_gain(CardID, Step, Value)
tune_da_offset(CardID, Step, Value)
reload_auto_k_setting(CardID)
```

In the past, the calibration of analog I/O needs many VRs to finished it. Now, SSCNET board has built-in the electric VRs in PLD. Users needn’t use screw driver to tune the value of offset or gain anymore. They need only set the values like tuning VR via those functions we provided. Users needn’t to tune these value because we have done this when this board is produced. The procedure of tuning these analog channels are as following:

1. Tune the on board +5V generator to exactly +5.0000V by measuring it from JP1 connector on daughter board. PCI-8372+/8366+ don't have this feature.
2. Execute tune_ad_offset_gain() by Step=1, Value=128. PCI-8372+/8366+ don't have this feature.
3. Execute tune_ad_offset_gain() by Step=3, Value=128. PCI-8372+/8366+ don't have this feature.
4. Calibrate AD offset using tune_ad_offset_gain() for Step=0. Check AD2's value as 0.0. PCI-8372+/8366+ does not have this feature.
5. Calibrate AD offset using tune_ad_offset_gain() for Step=1. Check AD2's value as 0.0. PCI-8372+/8366+ does not have this feature.
6. Calibrate AD gain using tune_ad_offset_gain() for Step=2. Check AD2's value as 5.0. PCI-8372+/8366+ does not have this feature.
7. Execute tune_da_offset() by Step=1, Value=128.
8. Execute tune_da_offset() by Step=3, Value=128.
9. Calibrate DA offset using tune_da_offset() for Step=0. Check AD2's value as 0.0.
10. Calibrate DA offset using tune_da_offset() for Step=1. Check AD2's value as 0.0.
11. Calibrate DA offset using tune_da_offset() for Step=2. Check AD2's value as 0.0.
12. Calibrate DA offset using tune_da_offset() for Step=3. Check AD2's value as 0.0.
13. Execute save_auto_k_value() and the tuning value above will be saved in EEPROM on the board. These values will be restored by reload_auto_k_value() when board is initialized.

4.7 Driver Management

4.7.1 Driver parameter

In this section, the following functions are discussed.

```
get_servo_para(Axis, ParaNo,*Value)
set_servo_para(Axis, ParaNo, Value)
get_servo_para_all(Axis, *Value)
set_servo_para_all(Axis, *Value)
save_servo_para(I16 Axis)
set_servo_para_default(I16 Axis)
```

With the SSCNET board, servo parameters read/write becomes very easy using function calls listed above.

To read a current parameter setting, user can call get_servo_para() or get_servo_para_all(). get_servo_para() retrieves certain parameter values, while get_servo_para_all() will retrieve all parameter settings.

To set a new value for the servo parameters, user can call set_servo_para() or set_servo_para_all(). set_servo_para() will set new values for specified parameters only, while set_servo_para_all() will set all parameters' value.

After a servo parameter tuning process, user may use save_servo_para() to store current parameter setting. These values are stored in a Flash ROM of the SSCNET board.

Whenever the user wants to restore default parameter setting, the function set_servo_para_default() can be used. This will reset all parameters to the factory setting.

The following table is a simplified list of parameters. For more information, refer to the "MR-J2SB Instruction Manual".

Symbol	Name	MR-J2SB Instruction Manual Parameter	Unit	Setting Range
*AMS	Amp setting	Pr.01		0000H~0001H
*REG	Regenerative resistor	Pr.02		0000H~0011H
*MTY	For manufacturer's settings	Pr.03		0080H

Table 4-6: MR-J2SB Parameters

Symbol	Name	MR-J2SB Instruction Manual Parameter	Unit	Setting Range
*MCA	For manufacturer’s settings	Pr.04		0000H
*MTR	For manufacturer’s settings	Pr.05		1
*FBP	Feedback pulse number	Pr.06		0,1,6,7,225
*POL	Direction of motor rotation	Pr.07		0,1
ATU	Auto-tuning	Pr.08		0000H~0004H
RSP	Servo response setting	Pr.09		0001H~000FH
TLP	Forward rotation torque limits	Pr.10	%	0~Maximum torque
TLN	Reverse rotation torque limits	Pr.11	%	0~Maximum torque
DG2	Moment of inertia ratio of load	Pr.12	0.1	0~3000
PG1	Position control gain 1	Pr.13	rad/sec	4~2000
VG1	Speed control gain 1	Pr.14	rad/sec	20~8000
PG2	Position control gain 2	Pr.15	rad/sec	1~1000
VG2	Speed control gain 2	Pr.16	rad/sec	20~20000
VIC	Speed integration compensation	Pr.17	msec	1~1000
NCH	Mechanical resonance control filter	Pr.18		0~031FH
FFC	Feed forward gain	Pr.19	%	0~100
INP	In position range	Pr.20	pulse	0~50000
MBR	Electromagnetic brake sequence output	Pr.21	msec	0~1000
MOD	Monitor output mode	Pr.22		0000H~0B0B H
OP1	Optional function 1	Pr.23		0000H~0001H
OP2	Optional function 2	Pr.24		0000H~0110H
LPF	Low pass filter	Pr.25		0000H~1210H
OP4	For manufacturer’s settings	Pr.26		0000H
MO1	Monitor output 1 offset	Pr.27	mv	-999~999
MO2	Monitor output 2 offset	Pr.28	Mv	-999~999

Table 4-6: MR-J2SB Parameters

Symbol	Name	MR-J2SB Instruction Manual Parameter	Unit	Setting Range
MOA	For manufacturer's settings	Pr.29		0001H
ZSP	Zero speed	Pr.30	rpm	0~10000
ERZ	Error excess alarm level	Pr.31	kpulse	1~1000
OP5	Option function 5	Pr.32		0000H~0002H
OP6	For manufacturer's settings	Pr.33		0000H~0113H
VPI	PI-PID change position droop	Pr.34		0~50000
TTT	For manufacturer's settings	Pr.35		0000H
VDC	Speed integration compensation	Pr.36		0~1000
OP7	For manufacturer's settings	Pr.37		0010H
ENR	Encoder output pulse	Pr.38		0~32768
	For manufacturer's settings	Pr.39		0000H
*BLK	Parameter block	Pr.40		0000H~000E H

Table 4-6: MR-J2SB Parameters

4.7.2 Data monitoring

In this section, the following functions are discussed.

```
set_monitor_channel(Axis, Channel_0, Channel_1,
     Channel_2, Channel_3)
set_monitor_config(Axis, Trigger_Select,
     Trigger_Level, SamplePeriod,
     PreTriggerSampleNo, SampleNumber)
get_instant_monitor_data(Axis,*Data_0,*Data_1,*D
     ata_2,  *Data_3)
start_monitor(Axis)
check_monitor_ready(Axis, *status)
get_monitor_data(Axis, *Data)
```

The firmware in the SSCNET board gives each axis 4 monitoring channels. Users can use these monitoring channels to monitor a variety of I/O data, such as Speed feedback, INP (in position)… etc.

To be able to use the monitoring function, users must understand the configuring and operating procedures.

Configuring procedures:

Configuring procedure is necessary before a monitor function can be started. There are two main instructions during configuration:

1. set_monitor_channel

This function is used to set the monitoring target. This function must be executed before monitoring can started.

The first parameter ‘Axis’ specifies the axis. The remaining four parameters are used for the monitoring targets. The relationship between set values and monitoring targets are list below.

Value	Description	Unit
FF	Not Used	
00	Feedback pulse accumulation	Pulse
01	(Reserved)	
02	Motor revolution speed	0.1rpm
03	(Reserved)	
04	Accumulated pulse	Pulse
05	(Reserved)	
06	Regenerative load factor	%
07	Execution load factor	%
08	Peak load factor	%
09	Bus voltage	
0A	Load inertia ratio	
0B	ABS counter	Rev
0C	Position within one revolution	Pulse
0D	(Reserved)	
0E	F/B present value	Pulse
0F	(Reserved)	
10	Position droop	Pulse
11	(Reserved)	
12	Speed command	0.1rpm
13	(Reserved)	0.1rpm
14	Speed feedback	0.1rpm
15	(Reserved)	0.1rpm
16	Current command	0.1%
17	Current feedback	0.1%
18	ZCT (Bottom)	Pulse
19	(Reserved)	
1A	Present revolution counts	Rev

Table 4-7: Monitoring Targets

Value	Description	Unit
1B	Origin revolution counts	Rev
1C	Origin position within one revolution	Pulse
1D	(Reserved)	
1E	(Reserved)	
1F	(Reserved)	
20	Alarm status AL-1	
21	Alarm status AL-2	
22	Alarm status AL-3	
23	Alarm status AL-4	
24	Alarm status AL-5	
25	Alarm status AL-6	
26	Alarm status AL-7	
27	Alarm status AL-8	
28	Alarm status AL-9	
29	Alarm status AL-E	
2A	(Reserved)	
2B	(Reserved)	
2C	(Reserved)	
2D	(Reserved)	
2E	(Reserved)	
2F	(Reserved)	
30	Alarm history #1,#2	
31	Alarm history #3,#4	
32	Alarm history #5,#6	
33	Alarm history #7,#8	
34	Alarm history #9,#10	
35	(Reserved)	
36	(Reserved)	
37	(Reserved)	
38	Parameter error NO.Pr01 to Pr16	
39	Parameter error NO.Pr17 to Pr32	
3A	Parameter error NO.Pr33 to Pr40	

Table 4-7: Monitoring Targets

Value	Description	Unit
3B	(Reserved)	
3C	(Reserved)	
3D	(Reserved)	
3E	(Reserved)	
3F	(Reserved)	
A5	INP (in position)	Active: 1, Inactive: 0
B0	Velocity Command	Pulse/sec
B2	DA1 Value	
B3	DA2 Value	
B4	Speed Feedback	
B6	External Encoder Feedback	
B8	Command Pulse	

Table 4-7: Monitoring Targets

2. set_monitor_config

This function is used to set the monitoring configuration, such as sampling period, trigger condition…etc. This function must be executed before monitoring can start.

The first parameter ‘Axis’ specifies which axis. Other parameters are listed below.

Parameter Name	Description
Trigger_Select	This variable is used to define the trigger source.
	Trigger_Select:
	Value = 0: No trigger
	Value = 1: CH0 as trigger source, going high
	Value = 2: CH1 as trigger source, going high
	Value = 3: CH2 as trigger source, going high
	Value = 4: CH3 as trigger source, going high
	Value = -1: CH0 as trigger source, going low
	Value = -2: CH1 as trigger source, going low
	Value = -3: CH2 as trigger source, going low
	Value = -4: CH3 as trigger source, going low
TriggerLevel	Define the trigger level
SamplePeriod	This variable is used to define the sample period.
	Value = 1: 0.888 ms
	Value = 2: 2 * 0.888 ms
	Value = 3: 3 * 0.888 ms
	Value = 4: 4 * 0.888 ms
PreTriggerSampleNo	Define the Number of samples before Trigger
	Value = 1 ~ 1023
SampleNumber	Define the Total Number of samples
	Value = 1 ~ 1023

Table 4-8: Axis Parameters

Operating procedures:

There are 2 operation modes, real time data reading and normal monitoring.

1. get_instant_monitor_data

After the monitoring channels have been set by set_monitor_channel(), the get_instant_monitor_data() function can be used to retrieve monitoring data.

The first parameter ‘Axis’ specifies which axis. The remaining four parameters are used to retrieve monitoring data from the monitoring channels.

This function returns a value immediately and carries out real time monitoring of specified monitoring targets.

2. Normal monitoring

Normal monitoring is data sampling with the help of the on-board DSP. The DSP take charge of storing all sampled data according to configuration set by set_monitor_config(). The following steps are necessary to operate in normal monitoring.

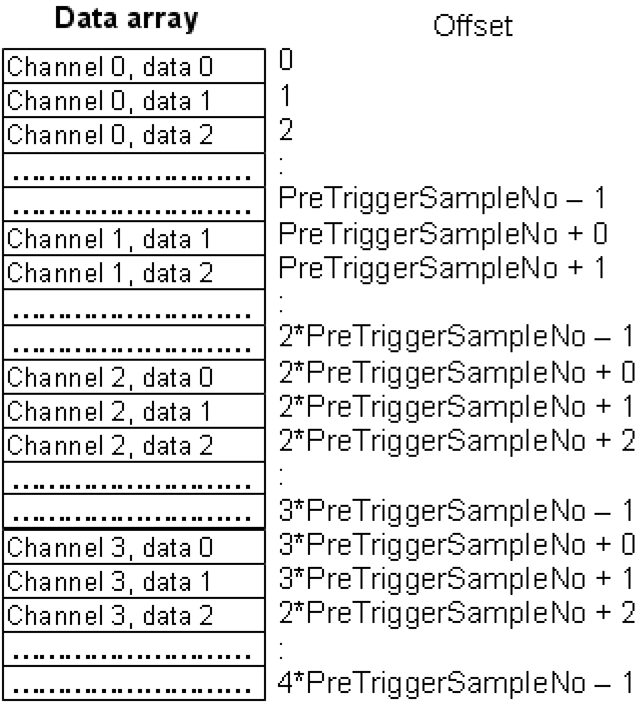
Step 0: Set monitor channel and configuration using function call: set_monitor_channel(), set_monitor_config()

Step 1: Start normal monitoring using the function call: start_monitor(). This will start the DSP.

Step 2: Check if the monitoring has completed by calling the function: check_monitor_ready(). The parameter ‘status’ returns a value, if the monitoring process has completed.

Step 3: If monitoring has completed, then call get_monitor_data() to retrieve the monitored data.

The size of the ‘data’ array of the function get_monitor_data(), which is used to read the monitored data must be 4 times the value of ‘PreTriggerSampleNo’ in set_monitor_config() and when the function get_monitor_data() returns a value, the ‘data’ is stored in the following format.



Data array	Offset
Channel 0, data 0	0
Channel 0, data 1	1
Channel 0, data 2	2
…………………………	:
…………………………	PreTriggerSampleNo – 1
Channel 1, data 1	PreTriggerSampleNo + 0
Channel 1, data 2	PreTriggerSampleNo + 1
…………………………	:
…………………………	2*PreTriggerSampleNo – 1
Channel 2, data 0	2*PreTriggerSampleNo + 0
Channel 2, data 1	2*PreTriggerSampleNo + 1
Channel 2, data 2	2*PreTriggerSampleNo + 2
…………………………	:
…………………………	3*PreTriggerSampleNo – 1
Channel 3, data 0	3*PreTriggerSampleNo + 0
Channel 3, data 1	3*PreTriggerSampleNo + 1
Channel 3, data 2	2*PreTriggerSampleNo + 2
…………………………	:
…………………………	4*PreTriggerSampleNo – 1

Table 4-9: Data Array Offset

4.7.3 Servo Information

In this section, the following function is discussed.

```
get_servo_info(Axis,*ServoInfo)
```

This function is used to retrieve the servo driver's status information. The parameter '*ServoInfo' carries the information about servo driver's status in individual bit's:

Bit 0	In Ready-ON
Bit 1	In Servo-ON
Bit 2	In course of in-Position

Table 4-10: Servo Bit Information

Bit 3	In course of zero speed
Bit 4	Pass through Z phase of servo motor
Bit 5	In Torque limit
Bit 6	In Alarm
Bit 7	In warning
Bit 8	Reserve
Bit 9	Reserve
Bit 10 ~ 14	Reserved
Bit 15	In course of Speed limit
Bit 16 ~ 31	Reserved

Table 4-10: Servo Bit Information

4.7.4 Servo On

In this section, the following function is discussed.

```
set_servo_on(Axis,ON_OFF)
```

After this function is execute with 'ON_OFF' = 1, the servo driver of specified axis starts to control its servomotor. Motion functions can now be applied to the axis.

In most cases, Servo driver should be at servo ON status, except that, "Before set_position() function, the servo driver must be at Servo OFF status."

4.7.5 Driver information

In this section, the following functions are discussed.

```
understand_driver(Axis, *Class_Code)
understand_motor(Axis, *MotorType, *Capacity,
     *RateRPM, *RateCurrent, *MaxRPM, *MaxTorq,
     *PPR, *ENCInfo, *OptionalInfor)
```

When booting, the SSCNET board will gather some static information about the servo driver and servomotor. This information is kept by the SSCNET board and users can retrieve the info using the following two functions.

- To read the servo driver's static info, use understand_driver()
- To read servo motor's static info, use understand_motor()

4.7.6 Servo Alarm

In this section, the following functions are discussed.

```
get_alarm_no(Axis,*AlarmNo)
alarm_reset(Axis)
```

When a fault occurs, the servo driver will stop the motor and report an alarm number on the LED display of the servo driver. The SSCNET board will also be acknowledged because the servo driver also reports the alarm condition through SSCNet communication. If this is the case, users can exam the fault condition by:

1. Examining the return code from the function call.
2. Set an IRQ. Allow a IRQ to be generated, if an alarm occurs. Refer to section 4.9.

After noticing the occurrence of the alarm, users can use get_alarm_no() to retrieve the alarm code, which is displayed on the LED of the servo driver. After removing the alarm condition, users can use the alarm_reset() function to recover the driver from the alarm state.

4.8 Control Gain Tuning

In this section, the following functions are discussed.

```
set_auto_tune(Axis, Mode, RSP, GD2)
get_auto_tune(Axis, *Mode, *RSP, *GD2)
set_control_gain(Axis, PG1, VG1, VIC,PG2, VG2,
     FFC)
get_control_gain(Axis, *PG1, *VG1, *VIC,*PG2,
     *VG2, *FFC)
set_notch_filter(Axis, Mode, NotchFrequency,
     NotchDepth)
get_notch_filter(Axis, *Mode, *NotchFrequency,
     *NotchDepth)
set_LP_filter(Axis,ON_OFF)
```

```
get_LP_filter(Axis,*ON_OFF)
```

4.8.1 Control Gains

The first 4 functions are used to set/read the gain controls of the servomotor control system. There are 6 control gains, and they are PG1, VG1, VIC, PG2, VG2, and FFC. The following are some simple description of the control gains, for more information refers to the "Instruction Manual" of the MR-J2S-B servo driver.

PG1: Position loop gain 1

Increase this value to improve tractability in response to the position command.

VG1: Velocity loop gain 1

Normally this gain value does not need to be changed. A higher set value increases the response level but is likely to generate vibration and/or noise.

VIC: Velocity integral compensation

Used to set the integral time constant of a speed loop. A higher set value increases the response level but is likely to generate vibration and/or noise.

PG2: Position loop gain 2

This gain is used to increase the response due to level load disturbance. A higher set value increases the response level but is likely to generate vibration and/or noise.

VG2: Velocity loop gain 2

Set this gain when vibration occurs to machines with low rigidity or with large backlash. A higher set value increases the response level but is likely to generate vibration and/or noise.

FFC: Feed foreword gain

Used to set the velocity feed foreword gain. When it is set to 100%, drop pulses will be almost zero at constant-speed operation. Note that higher set values will increase response but will enlarge the overshoot during sudden acceleration/deceleration.

Auto-Tuning mode

The MR-J2S-B servo driver has as read-time auto tuning function, which can automatically estimate the machines characteristic and set the optimum control gain values in real time.

There are two Auto-Tuning modes:

Auto-Tuning mode 1:

In this mode, the load inertia moment of a machine is estimated, and all control gains are set automatically, creating a machine response frequency that matches the user's requirements (RSP). The servo driver is factory-set to this mode.

Under this mode, set_control_gain() does not work and will return errors.

Auto-Tuning mode 2:

Under this mode, the user must specify the load inertia moment for the machine (GD2) with all other control gains set automatically, creating a machine response frequency that matches the user's requirements (RSP).

Under this mode, set_control_gain() does not work and will return errors.

Manual setting mode

If the user is not satisfied with the adjustment of auto-tuning, he/ she can make manual adjustments.

Manual mode 1:

Under this mode, the user can specify the following control gains including GD2, PG1, VG2, and VIC, while PG2 VG1 and FFC are set automatically.

Under this mode, set_control_gain() can be used to set PG1, VG2, and VIC. Setting for PG2, VG1 and FFC is automatically ignored.

Manual mode 2:

Under this mode, the user can specify all control gains. The function call set_control_gain() can be used to set PG1, VG2, VIC, PG2, VG2 and FFC.

The function call set_auto_tune() is used to select the auto-tuning or manual tuning mode with the parameter 'Mode' specifying the operation mode. If Auto-tuning mode is selected, 'RSP' specifies the user's machine response frequency requirements, while in manual mode it's not applicable. 'GD2' specifies the load inertia moment of a machine.

- The function call get_auto_tune() can be used to read settings from the servo driver.
- The function call set_control_gain() is used to set the control gains when manual mode operation is selected.
- The function call get_auto_tune() can be used to read settings from the servo driver.
- The following table is a list of selectable gains under different operation modes:

'Mode' Value	Description	RSP	GD2	PG1	VG1	VIC	PG2	VG2	FFC
1	Auto-Tuning mode 1	M	A	A	A	A	A	A	A
2	Manual mode 2	--	M	M	M	M	M	M	M
3	Auto-Tuning mode 2	M	M	A	A	A	A	A	A
4	Manual mode 1	--	M	M	A	M	A	M	A
0*	Interpolation mode	--	A	M	M	A	A	A	A

Table 4-11: Selectable Gains

- A: automatically set
- M: manually specify
- --: Not used
- * Interpolation mode is normally not used.

4.8.2 Mechanical resonance suppression filter

The functions set_notch_filter() and get_notch_filter() are related to the mechanical resonance suppression filter function.

If a mechanical system has a natural resonance point, increasing the servo system response may cause the mechanical system to produce resonance (vibration) at its resonance frequency. Using the notch filter and adaptive vibration suppression control functions can suppress the resonance of the mechanical system.

Notch filter

The notch filter is a filter function, which decreases the gain of the specific frequency (notch frequency) and gain decreasing depth.

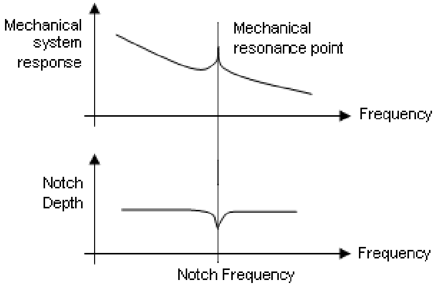


Figure 4-36: Notch Filter

Note: The machine resonance suppression filter is a delay factor for the servo system. Hence, vibration may increase if you set a wrong resonance frequency or a too deep notch.

If the resonance frequency of the machine is unknown, decrease the notch frequency from a higher to lower order. The optimum notch frequency is set at the point where vibration is minimal.

The notch frequency is set with the parameter 'NotchFrequency'. The setting values and corresponding notch frequency are listed in the table below.

Setting	Frequency	Setting	Frequency	Setting	Frequency	Setting	Frequency
0	Invalid	8	562.5	16	281.3	24	187.5
1	4500	9	500	17	264.7	25	180
2	2250	10	450	18	250	26	173.1
3	1500	11	409.1	19	236.8	27	166.7
4	1125	12	375	20	225	28	160.1
5	900	13	346.2	21	214.3	29	155.2
6	750	14	321.4	22	204.5	30	150
7	642.9	15	300	23	195.7	31	145.2

Table 4-12: Notch Frequency Settings

A deeper notch provides better resonance suppression but increases the phase delay and may increase vibration.

The notch frequency can be set with the parameter ‘NotchDepth’. The setting values and corresponding notch gain are listed in the table below.

Setting	Depth (Gain)
0	-40db
1	-14db
2	-8db
3	-4db

Table 4-13: Notch Gain Settings

Adaptive vibration suppression control

Adaptive vibration suppression control is a function in which the servo driver detects machine’s resonance and set a adaptive filter (also a notch filter) automatically to suppress mechanical system vibration.

Note: This adaptive filter has nothing to do with previous Notch filter setting.

Since the adaptive filter characteristics (notch frequency and depth) are set automatically, user need not be conscious of the resonance frequency of a mechanical system. Also, while adaptive vibration suppression control is valid, the servo driver always detects machine resonance, and if the resonance frequency changes, it changes the filter characteristics in response to that frequency.

The Adaptive vibration suppression control function can be set with the parameter ‘Mode’. The setting value and suppression control function are listed in the table below.

Setting	Control Selection
0	Invalid - The adaptive vibration suppression control is not used.

Table 4-14: Suppression Control Settings

Setting	Control Selection
1	Valid_0 - The adaptive vibration suppression control is enabled with normal sensitivity of detecting machine resonance.
2	Valid_1 - The adaptive vibration suppression control is enabled with large sensitivity of detecting machine resonance.
3	Hold - filter characteristic generated so far is held, and detection of machine resonance is stopped.

Table 4-14: Suppression Control Settings

Note: 1. The mode setting does not affect the notch filters functionality set by 'NotchFrequency' and 'NotchDepth'

2. Adaptive vibration control is factory-set to be invalid.

3. Adaptive vibration control is useful only when machine resonance is between 150 ~ 500 Hz. It has no effect on the resonance frequency outside this range.

4. Under operating conditions in which sudden disturbance is imposed during operation, the detection of the resonance frequency may malfunction temporarily, causing machine vibration. In such a case, set the adaptive vibration suppression control mode to be 3 (Hold) to fix the characteristics of the adaptive vibration suppression control filter.

4.8.3 Low pass filter

The functions set_LP_filter() and get_LP_filter() are related to low pass filter functions.

When a ball-screw or the like is used, resonance of high frequency may occur as the response of the servo system is increased. To prevent this, the low-pass filer is factory-set to be valid with a torque command. The filtering frequency of this low pass filter is automatically adjusted to the value according to the expression below:

$$\text{Filter Frequency (Hz)} = \frac{\text{VG2 setting}^{*}\ 10}{2 * \pi * (1 + \text{GD2 setting} * 0.1)}$$

The low pass filter can be enabled or disabled using the parameter ‘**ON_OFF**’.

- ‘**ON_OFF**’ = 0, Disabled
- ‘**ON_OFF**’ = 1, Enabled

Note: In a mechanical system where rigidity is extremely high and resonance s difficult to occur, setting the low pass filter to be ‘Disabled’ may increase the servo system response to shorten the settling time.

4.9 Interrupt control

In this section, the following functions are discussed.

```
int_control(CardID, Flag)
set_int_factor(CardID, Source, IntFactor)
get_int_status(CardID, Source, *IntStatus)
set_int_event(CardID, *HEvent)
link_interrupt(CardID, *callbackAddr)
```

The SSCNET board can generate an interrupt for certain conditions. Refer to the figure below:

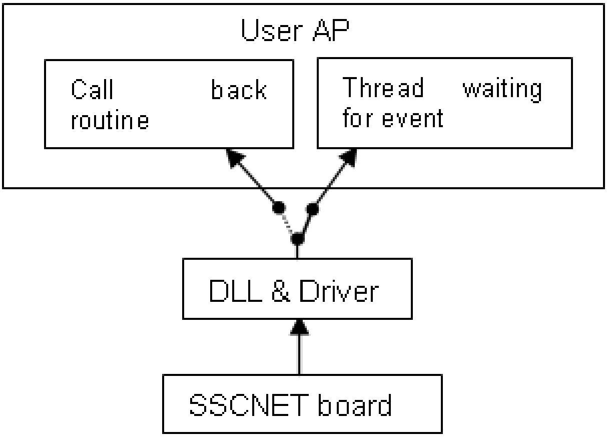


Figure 4-37: Interrupt Control

Users can either set a call back routine that will be executed when an interrupt occurs, or create a thread to wait for an event that will be triggered when an interrupt occurs.

To enable or disable the interrupt generated from the SSCNET board, use the int_control() function. It acts as an ON_OFF switch. Once disabled, the SSCNET board will cease to generate any interrupt signals to the host system.

In addition to int_control(), users need to define the conditions under which an interrupt signal should occurs by using the set_int_factor() function in order to successfully introduce an interrupt signal to the host system. The SSCNET board has 3 possible sources of interrupts; it includes the motion axes, general purposed I/O, and the DSP (or system). The second parameter 'Source' of set_int_factor() is use to specify the source with 'IntFactor' specifying the interrupt conditions for this specified source.

Refer to the tables below.

“Source” = 0 - 11, for Axis 0 - Axis 11 respectively.

Bit of ‘IntFactor’	Name	Description
0	PEL	Positive Limit Switch
1	MEL	Negative Limit Switch
2	ORG	Home Switch
3	RDY	Servo Ready
4	INP	In Position
5	EZ	Index signal passed
6	ZSPD	Zero Speed
7	TLC	Torque Limit reached
8	ALM	Alarm signal on
9	WRN	Servo Warning on
10	HOME	Home Move completed
11	MTC	Motion Completed
12	CPBF	Curve Parameter Buffer Full
13	EPD	Position deviation is too large
14	CMP1	Position compare1 is true
15	CMP2	Position compare2 is true

Table 4-15: Axis Interrupts

“Source” = 12 for system interrupt

Bit of ‘IntFactor’	Description
0	System Error
1	Emergency Stop
2	Cyclic Timer Interrupt

Table 4-16: System Interrupts

“Source” = 13 for GPIO interrupt

Bit of ‘IntFactor’	Description	
0	General purposed DI, Channel 0	
1	General purposed DI, Channel 1	
8		Compare_Counter_CH0

Table 4-17: GPIO Interrupts

Bit of ‘IntFactor’	Description	
9		Compare_Counter_CH1
10		Compare_Counter_CH2

Table 4-17: GPIO Interrupts

After setting the interrupt source and factors, the interrupt signal can be detected by using either a call back routine or a event waiting thread.

Note:

- For the PCI-8372+, the number of controllable axis is “12”, Thus:
  - Source: 0 - 11 is for axis 0 - 11 individually,
  - Source: 12 for system,
  - Source: 13 for GPIO.
- For the PCI-8366+ the number of controllable axes is “6”,
  - Source: 0 - 5 is for axis 0 - 5 individually,
  - Source: 6 for system,
  - Source: 7 for GPIO.

By call back routine

The link_interrupt() function helps users to set up a call back routine. Each SSCNET board has its own call back routine. This routine will be executed once an interrupt occurs. Note, during routine execution, the next interrupt is on hold until the routine has ended.

By thread

The set_int_event() function helps users to set up a event handle. This event will be fired once an interrupt occurs. Therefore, users can create an independent thread to wait for the event handle.

4.10 Position Compare Function

In this section, the following functions are discussed.

```
set_compare(Axis, CMP1Pos, CMP1Dir, CMP2Pos,
     CMP2Dir)
set_single_compare(Axis, Channel, CMP_Pos)
check_compare(Axis, *status)
```

Each axis of the SSCNET board has 2-position compare channels. After setting the channels using set_compare(), the DSP of the SSCNET board compares the feedback position on each SSCNET cycle with “CMP#Pos” for each channel. The comparison includes the direction. The user can specify the compare succeed condition to be any of the following:

```
Direction = 0, whenever feedback across
ComparePos
Direction = 1, feedback > ComparePos
Direction = 2, feedback >= ComparePos
Direction = 3, feedback < ComparePos
Direction = 4, feedback <= ComparePos
```

In order to understand the compared status of each specified axis, a second function check_compare() is helpful. The compared status will be reset to false each time set_compare() is executed. If the comparison of channel1 comes into existence, bit0 of “*status” will become ‘1’. If the comparison of channel1 comes into existence, bit1 of “*status” will become ‘1’.

Other method to obtain comparison result is through an interrupt. When a comparison comes into existence, the SSCNET board will generate an interrupt signal. Users need to set the interrupt to enabled and correct the interrupt factor so that the program can accept interrupt signals.

4.11 Interlock Function

In this section, the following function is discussed.

```
set_interlock(CardID, Flag, Axis_X, Axis_Y, X1,
     X2, Y1, Y2, Time)
get_interlock(CardID, *Enable, *Axis_X, *Axis_Y,
     *X1, *X2, *Y1, *Y2, *Time)
```

The SSCNET board provides one interlock function for each card. This function is used for collision avoidance for 2-axis operation, “Axis_X”, “Axis_Y”, and one specific area,”X1”,”X2”,”Y1”,”Y2”. Once the axis' position is inside this section, it is said that it has entered into the interlock area. The slow-down and speed-up algorithm is done by the on-board DSP. The following graph explains this action.

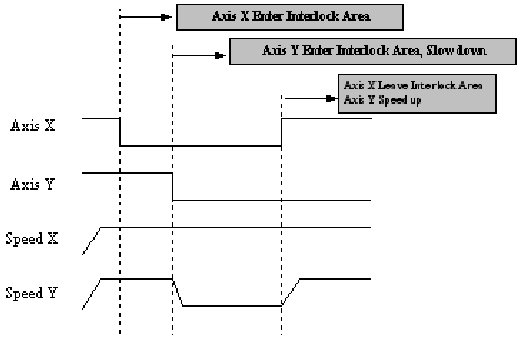

Axis X Enter Interlock Area
Axis Y Enter Interlock Area, Slow down
Axis X Leave Interlock Area
Axis Y Speed up
Axis X
Axis Y
Speed X
Speed Y

Figure 4-38: DSP Action Graph

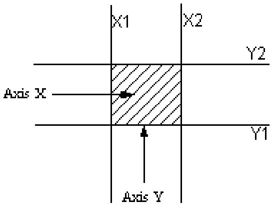


Figure 4-39: Interlock Area

Interlock function is much like a crossroad semaphore. In some applications, two independent axes will work on the same region occasionally. In the past, users must take care these two axes' movement to prevent collision. Now, SSCNET motion control card has this feature inside. Users don't need to worried about this problem. The axis will automatically slow down when the other axis is inside the predefined region and speed up again when the other axis leave from this region. It is much useful in this situation.

The encoder update rate is one SSCNET cycle. The slow down action will be token in the same cycle.

Coding Guide

```
set_interlock(CardID, Enable, Axis_X, Axis_Y, X1,
     X2, Y1, Y2, Time);
```

```
get_interlock(…) will retrieve above parameters
     for users’ to check.
```

Note: 1. Enable=1 means enable this function and 0 means disable

2. Time means slow down time when interlock happens

3. X1,X2,Y1,Y2 form a interlock region

4.12 Absolute Position System

In this section, the following function is discussed.

```
get_abs_position(Axis, *ABS_Pos)
save_abs_position(Axis)
clear_abs_data_on_flash(I16 CardID)
```

SSCNET board provide absolute position system for SSCNET motor driver. Users need only execute homing procedure once then the board can keep the absolute position in the ROM. When the machine restart next time, it will restore the absolute position of each axis from the ROM. The machine needn’t to doing the home procedure again.

Mitsubishi servo drivers use a battery to keep its encoder’s value inside at absolute position mode. The position value is on servo driver and users must turn on absolute position mode in parameter “AMS” before using this feature. Of course, users need to install a battery to keep the position value on driver’s side permanently.

In order to have absolute position feature on SSCNET motion control card, we must read the absolute position value from servo driver and retrieve the origin position information from FLASH ROM of SSCNET motion control card when card is initialized. After that we will calculate an absolute position of users’ machine according to these two information.

The formal procedure to use this features are as followings:

- Launch home_move() function to complete homing.
- Check if the home position is correct
- Launch save_abs_position() to store the ABS position as an origin position reference.
- Next time, when SSCNET motion control card starts, users don't need to launch home_move() anymore. They can only launch get_position() function to get an absolute position.

If the servo parameter07 is set to '1', remember to set this ABS position to operating position counter via the get_abs_position() and set_position() functions. Don't use the get_abs_position() in polling cycle because it is much slower than the get_position() function.

The Coding Guides are as followings

- Physical homing: (Axis doesn't know its origin position or program needs)
  - home_move(Axis0…);
  - WaitforSingleObject(Axis0Event, TimeOut) or Polling motion_done();
  - save_abs_position(Axis0…);
- Non-Physical homing (Axis already knew its Origin Position)
  - set_position(AxisNo…);

4.13 Compared Trigger Output

In this section, the following function is discussed.

```
map_dout_and_comparator(CardID, Dout_CH, AxisNo,
     CompNo, Dout_mode)
set_compare_table_dir(CardID, Table_ChNo, Dir)
link_dout_and_compare_table(CardID, DO_ChNo,
     StartI, EndI, *Table_Data)
```

For some applications, motion control must work with vision system. The vision system includes a CCD camera that needs to capture images at a specific location. These locations are discontinuous, but very closed at most cases. If users need to perform a high speed picture capturing on the fly, they have to consider the continuous compared with triggering pulse output feature.

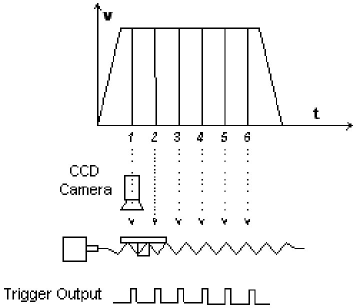


Figure 4-40: Trigger Output

SSCNET motion board has a compare mechanism of each axis that is operated by DSP. DSP will compare the receiving counter with users' desired value and do the actions in one SSCNET cycle if the position is achieved. Besides, we provide triggering pulse output when the compare condition happens. The triggering pulse is performed by DO channel. It will output a specific pulse width when compare condition happens.

First, users must set one of the digital output channels as one comparator's triggering output. Map more than one comparators to single digital output channel is not allowed, but reverse case is allowed.

Second, users must build up an array that contains the table of compare points. You can assign a region by giving start and end index of this array for the table of compare points.

Third, users must assign the compare direction of the table. It is useful because users needn't rebuild the table reversely again if they want to do a reverse comparasion.

The triggering pulse is as below. The pulse width is about 1ms which is decided by SSCNET cycle time. Our suggestion triggering frequency is less than 500Hz.

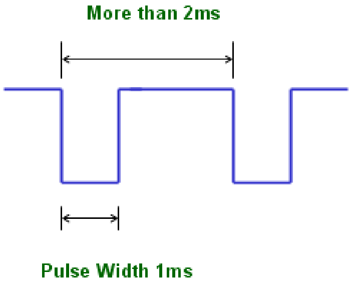


Figure 4-41: Triggering Frequency Under 500Hz

Before using this feature, users must map on-board digital output channel to axis' comparator. The mapping could be one ouptput channel to one comparator or two output channels to one comparator. For example, users can map Dout Channel 0 to axis2' comparator0 and Dout Channel 1 to axis3' comparator1. Or users can map Dout Channel 0 and Channel 1 to the same comparator for dual synchronous triggering pulse output.

After choosing the output channel and comparator, users must build a compare point table for digital output channel. Using the same table to map different digital output channel is allowable. The first element in compare table must be the smallest. The maximum number of point in the table is 100. This value is limited by DSP firmware. The triggering pulse width is about 1ms and compare accuracy is about +/-1ms.

Finally, set the compare direction in the table. Once users build a table, the table will remain on SSCNET board. The table can be compared either from upper side or lower side. The same table can be reused in different direction by changing the parameter, Dir. For example, Table contents 100,200,300. If they choose decreasing direction, the compare ordering will be 300,200,100 and vice versa.

The compare condition will be greater than or equal than depends on the compare direction

The first elements in compare table must be smallest like this order –300, -200, -100

```
< Example: Dual trigger pulses output by
     comparing two Table in one axis >
F32
     Table_Data1[10]={1500,2000,2500,3000,3500,4
     000,4500,5000,5500,6000};
F32
     Table_Data2[10]={1800,2300,2800,3300,3800,4
     300,4800,5300,5800,6300};
I16 AxisNo=1;
// Normal high setting
map_dout_and_comparator(0, 0, AxisNo, 0 ,1);
map_dout_and_comparator(0, 1, AxisNo, 1 ,1);
// Build Compare Table1,2 from index 0 to 9,
     totally 10 points
link_dout_and_compare_table(0, 0, 0,
     9,Table_Data1);
link_dout_and_compare_table(0, 1, 0,
     9,Table_Data2);
// Set Compare direction increaing
set_compare_table_dir(0, 0, 0 );
set_compare_table_dir(0, 1, 0 );
// Start Move from 0 to 8000, See results Picture
     1
start_tr_move(AxisNo, 8000,0, 60000,0,0,0);
// Wait for last move finished ( Users must
     handle this by any method )
// Set Compare direction decreasing
set_compare_table_dir(0, 0, 1 );
set_compare_table_dir(0, 1, 1 );
// Start Move from 8000 to 0, See results Picture
     2
start_tr_move(AxisNo, -8000,0, 60000,0,0,0);
<< Results Analyzing >>
Trigger pulse rate = 500/60000 = 8.33ms period
     time ( 1ms tolerance is acceptable )
Real Output in Picture 1 & 2 is about this value.
```

Results Pictures

- Picture one: Positive Move (CH1 in scope is DO Channel 0, CH2 in scope is DO Channel 1.)

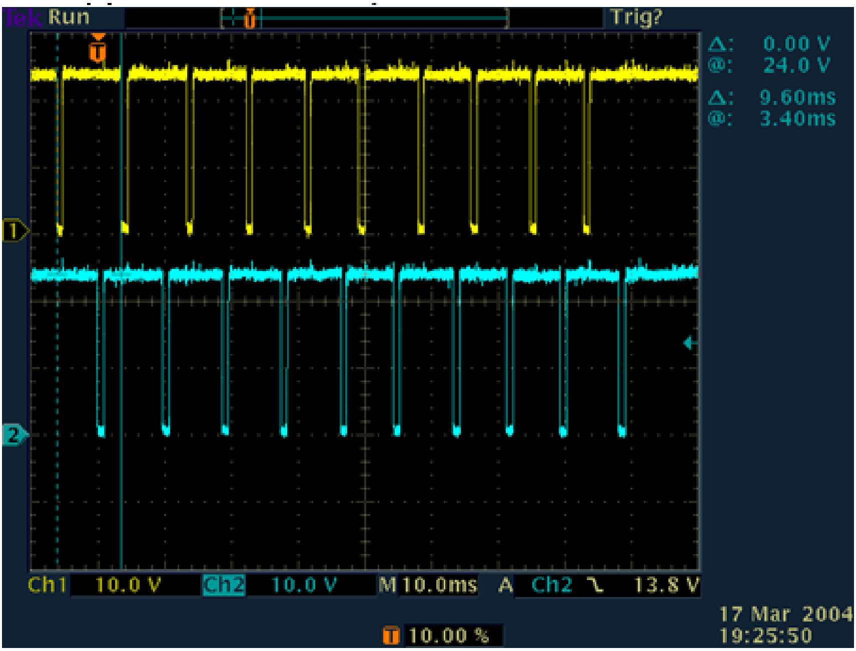


Figure 4-42: Positive Move

- Picture two: Negative Move (CH1 in scope is DO Channel 0, CH2 in scope is DO Channel 1.)

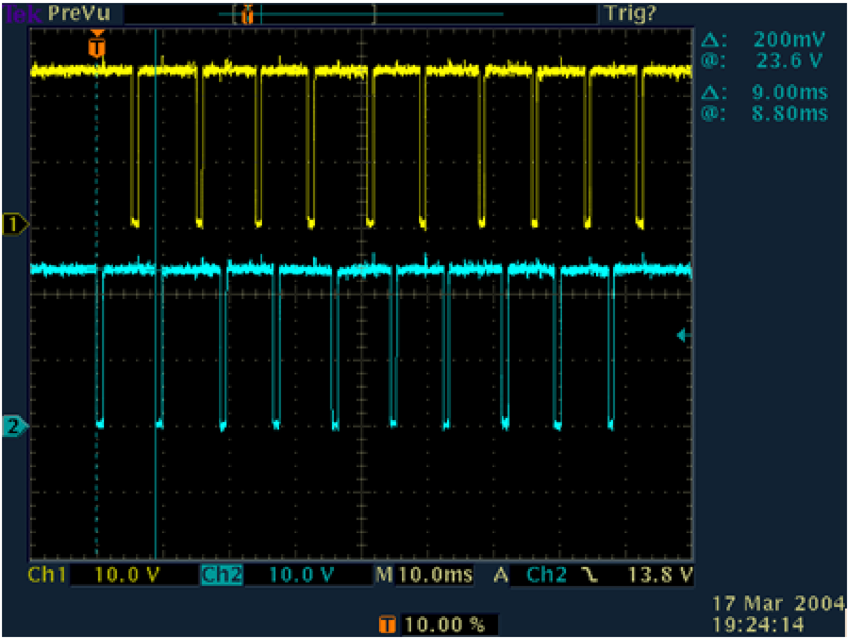


Figure 4-43: Negative Move

4.14 Sequence Motion Control

SSCNet has the property of the deterministic time, which is 0.888 ms. Theoretically, the motion command will be passed down to DSP with hand-shaking way. It takes two or three cycle times to complete the delivery and execute the motion command. Consequently, it is sure to waste some time and has slower response.

In order to improve the response time, the motion control board can let users have their own motion patterns downloaded into onboard DSP and realize the precisely timing control. All motion patterns can be executed in the DSP layer with the on-board RAM. The delivering time of motion command from computer to SSCNet board can be eliminated. It can increase the response time and get high performance control.

First, users depend on the timing chart (velocity profile) of desired motion to segment them and get many frames. Frame is the basic unit in this sequence motion control. Properly group some of those frames into one pattern. Simply put, a pattern contains many frames inside. The motion pattern can be reused and in the form of T-curve, S-curve, combined T-curve and S-curve, or arbitrary velocity profile. Consequently, you can plan it at your will.

Next, you have to consider the synchronism. In fact, some controlled axes may have the synchronous relation with each other. In a word, they are time-dependent. Consequently, you can group them as one sequence. DSP will execute the sequence based on the time relation to realize the synchronous motion. The sequence may conatins the pattern information of multi axes.

In other case, certain axis may refer to the other axis' condition and start to move. For example, axis 0 is planed to move while the maximum velocity of axis 1 is achieved. At this moment, you can use API to completely describe the motion patterns and achieve the time-dependent motion. The sequence contains the pattern information of one axis.

Sequence is the container of patterns. Pattern is the container of frames. Sequence also has the time-dependent description to complete the motion behavior that users want to realize.

Finally, every sequence designs triple pattern buffers to meet the motion continuity. While the patterns are executed continuously, the first pattern buffer will be passed down to DSP and the remaining two buffers would wait to be executed as the concept of queue. This design can make sure the continuity of motion. Users also can judge the motion status. If certain status comes into existence, you can replace the patterns in the sequence at your will.

4.14.1 Conceptual Flow Chart

1. Create Frames

The trapezoidal velocity profile has three frames and S-curve has seven frames. If you are not familiar with it, please refer to 4.2.

First of all, users have to prepare the timing chart (velocity profile) of all controlled axes. The following is the demonstrated timing chart for handler control:

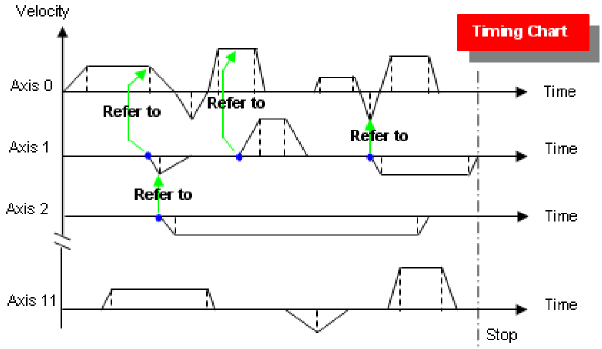


Figure 4-44: Conceptual Flow Chart - Timing A

As the diagram, you can see the timing chart of the 4 axes. In this case, we only control 4 axes –axis 0, 1, 2 and 11. As soon as having the complete timing chart, you can segment the velocity profile and obtain the frames that are based on the rule introduced earlier. The spot in the figure is the starting condition. Some axes will start to move based on the other axes' condition.

Here, we have to be aware of one thing. The synchronous relation should be noted. In this example, we have three dependent axes, which are axis 0, 1, 2. Axis 11 is independent of these three axes. In a word, those three axes have the synchronous relation.

Right now, we have to label the frame index. The following diagram shows that:

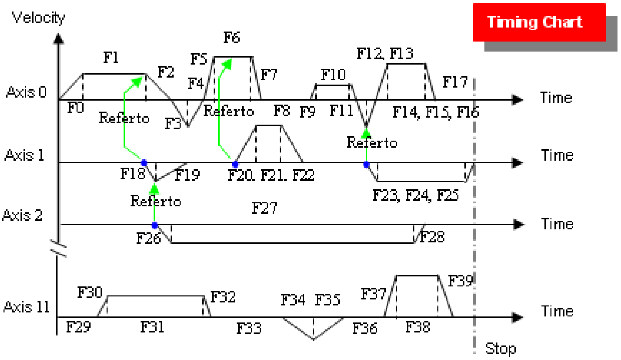

Velocity
Timing Chart
Axis 0
Axis 1
Axis 2
Axis 11
Time
Stop
F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 F10 F11 F12, F13 F14, F15, F16 F17
Referto
F18 F19 F20 F21 F22 F23, F24, F25
F26 F27 F28
F29 F30 F31 F32 F33 F34 F35 F36 F37 F38 F39

Figure 4-45: Conceptual Flow Chart - Timing B

There are totally 40 frames in this example.

2. Create Pattern

In this step, we can group several frames into one pattern. For example, we can have the patterns as follows:

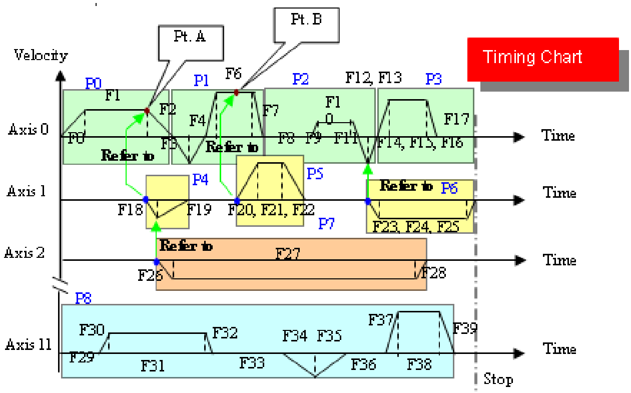

Pt. A
Pt. B
Velocity
Timing Chart
Axis 0
Axis 1
Axis 2
Axis 11
Time
Stop
P0 P1 P2 P3 P4 P5 P6 P7 P8
F0 F1 F2 F3 F4 F6 F7 F8 F9 F10 F11 F12, F13 F14, F15, F16 F17
Refer to
F18 F19 F20, F21, F22 F23, F24, F25
F26 F27 F28
F29 F30 F31 F32 F33 F34 F35 F36 F37 F38 F39

Figure 4-46: Conceptual Flow Chart - Pattern

The dash block represents the pattern. We mainly divide the timing chart of axis 0 into 4 patterns. Pattern 4 to 6 will activate depending on the axis 0. Pattern 7 will activate depending on axis 1. Here, users can have three selections to meet the requirement:

- Position compare: While the axis moves through certain position, it can let the other axis start to move. Like pattern 5, it will start to move while the axis 0 passes through point B.
- Velocity transition: The pattern can activate while the velocity of the other axis is at the end of the acceleration or the beginning of the deceleration. Like pattern 4, it will start to move while the velocity of axis 0 passes through point A.
- External I/O signal: You can also use the I/O signal as trigger to let the pattern start to move.

Then, Pattern 8 is an independent one. The following table is the summary:

Pattern Index	Content
Pattern 0	F0 to F2
Pattern 1	F3 to F7
Pattern 2	F8 to F13
Pattern 3	F14 to F17
Pattern 4	F18, F19
Pattern 5	F20, F21, F22
Pattern 6	F23, F24, F25
Pattern 7	F26, F27, F28
Pattern 8	F29 to F39

Table 4-18: Pattern Index

3. Sequence with Three Pattern Buffers

The sequence conatins three pattern buffers and realizes the desired motion. Every sequence has three pattern buffers in order to execute the pattern smoothly. Users have to input the patterns into the command buffers. The concept can be shown as the following diagram:

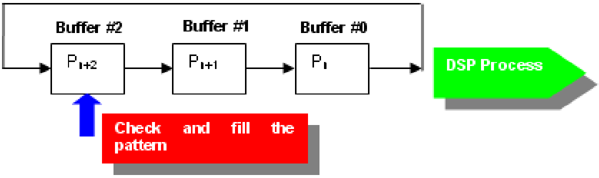


Figure 4-47: Conceptual Flow Chart - Buffers A

The sequence is an abstract object. It collects several patterns as a group. The pattern is a substantial object. It contains the information of frames.

Inside the board, the first three patterns can be stored in those three pattern buffers in advance. While the pattern Pn is executed by DSP, the remaining two patterns, Pn+1 and Pn+2, will be pushed forward and wait to be executed. If you have more than three patterns, you can use API function to check that the buffer status is full or not. It not, the next pattern can be put into the buffer. It is shown as follows:

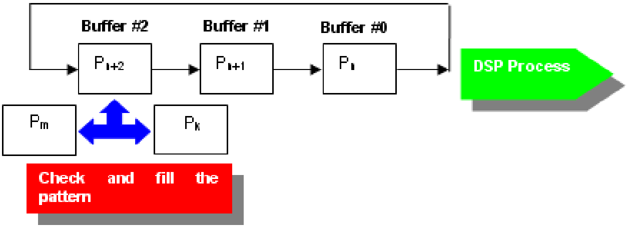


Figure 4-48: Conceptual Flow Chart - Buffers B

In some cases, users may have multiple selectable patterns based on specific situation. For example, they are Pm and Pk. Currently, you can depend on your condition to put Pm or Pk into the empty buffer. As for simple case, you can just throw the next pattern into the empty buffer if no any other specific condition should be considered and no selectable patterns are in hand.

In this case, we let every axis to be as a sequence. Group the patterns and we can have the sequence as follows:

Sequence	Contains
Sequence 0	P0 to P3
Sequence 1	P4 to P6
Sequence 2	P7
Sequence 3	P8

Table 4-19: Sequences

Then, you can use the API to link the synchronous relation.

- P4 will start to move referring to point A of axis 0.
- P5 will start to move referring to point B of axis 0.

The same rule is for pattern 6 and 7.

4.14.2 Coding Example 1: Using C Language

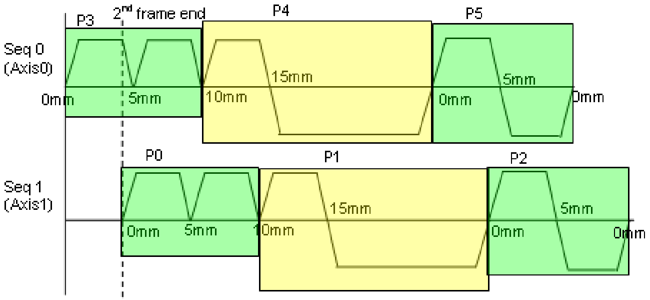


Figure 4-49: Coding Example 1

1. Variables Setting

```
I16 FirstFrame,LastFrame;
I16 AxisNo;
I16 SynAxes;
I16 PatternNo;
I16 WaitAxis,WaitCondition;
```

2. Create Patterns for Sequence 1

```
AxisNo=1;
SynAxes=0x02;
PatternNo=0;

// Pattern 0
FirstFrame=0;
LastFrame=add_frame_ta_move(AxisNo, FirstFrame,
     0, 5, 0, 10, 0, 0.1, 0.1);
LastFrame=add_frame_ta_move(AxisNo, LastFrame, 5,
     10, 0, 10, 0, 0.1, 0.1);
set_pattern(0, PatternNo, FirstFrame, LastFrame -
     FirstFrame, SynAxes);
PatternNo++;

// Pattern 1
FirstFrame=LastFrame;
LastFrame=add_frame_ta_move(AxisNo, FirstFrame,
     10, 15, 0, 10, 0, 0.1, 0.1);
LastFrame=add_frame_ta_move(AxisNo, LastFrame,
     15, 0, 0, 10, 0, 0.1, 0.1);
set_pattern(0, PatternNo, FirstFrame, LastFrame -
     FirstFrame, SynAxes);
PatternNo++;

// Pattern 2
FirstFrame=LastFrame;
LastFrame=add_frame_ta_move(AxisNo, FirstFrame,
     0, 5, 0, 10, 0, 0.1, 0.1);
LastFrame=add_frame_ta_move(AxisNo, LastFrame, 5,
     0, 0, 10, 0, 0.1, 0.1);
set_pattern(0, PatternNo, FirstFrame, LastFrame-
     FirstFrame,SynAxes);
PatternNo++;
```

3. Create Patterns for Sequence 0

```
AxisNo=0;
SynAxes=0x01;

// Pattern 3
FirstFrame= LastFrame;
LastFrame=add_frame_ta_move(AxisNo, FirstFrame,
     0, 5, 0, 10, 0, 0.1, 0.1);
```

```
LastFrame=add_frame_ta_move(AxisNo, LastFrame, 5,
     10, 0, 10, 0, 0.1, 0.1);
set_pattern(0, PatternNo, FirstFrame, LastFrame-
     FirstFrame, SynAxes);
PatternNo++;

// Pattern 4
FirstFrame=LastFrame;
LastFrame=add_frame_ta_move(AxisNo, FirstFrame,
     10, 15, 0, 10, 0, 0.1, 0.1);
LastFrame=add_frame_ta_move(AxisNo, LastFrame,
     15, 0, 0, 10, 0, 0.1, 0.1);
set_pattern(0, PatternNo, FirstFrame, LastFrame-
     FirstFrame, SynAxes);
PatternNo++;

// Pattern 5
FirstFrame=LastFrame;
LastFrame=add_frame_ta_move(AxisNo, FirstFrame,
     0, 5, 0, 10, 0, 0.1, 0.1);
LastFrame=add_frame_ta_move(AxisNo, LastFrame, 5,
     0, 0, 10, 0, 0.1, 0.1);
set_pattern(0, PatternNo, FirstFrame, LastFrame-
     FirstFrame,SynAxes);
PatternNo++;
```

4. Start Sequence 0 & 1 at the same time

```
reset_seq_buffer(0,1);
// First pattern of this sequence will wait
     pattern3’s frame 2 of axis0 to start
WaitAxis=0;
WaitCondition=3  // Pattern 3 (P3)
            SynAxes=0x02
insert_pattern_to_seq_buffer(0, 1, 0,
     SyncAxes,1,WaitAxis, WaitCondition,2);
     insert_pattern_to_seq_buffer(0, 1, 1,
     SynAxes,0,0,0,0);
insert_pattern_to_seq_buffer(0, 1, 2,
     SynAxes,0,0,0,0);

reset_seq_buffer(0, 0);
             SynAxes=0x01;
```

```
insert_pattern_to_seq_buffer(0, 0, 3,
     SynAxes,0,0,0,0);
insert_pattern_to_seq_buffer(0, 0, 4,
     SynAxes,0,0,0,0);
insert_pattern_to_seq_buffer(0, 0, 5,
     SynAxes,0,0,0,0);

start_seq_move(0, 0x3);
```

Working with more than 3 patterns in one sequence

If the patterns are morer than three, users must know how to use the sequecen command buffers. There are three command buffers in each sequence. Users can use the command buffer to fulfill the continuous sequence motion.

Before inserting a new pattern into sequence command buffer, users must use the following function to check if the buffer is full.

```
I16 check_seq_buffer(I16 CardID, I16 SeqNo);
```

If the function returns 1, it means the sequence buffer is ready for next command. If the function returns 0. It means all sequence command buffers are full.

```
while ( check_seq_buffer(CardID, SeqNo ) == 0  );
     // wait buffer empty
```

Pause and resume a sequence

Sometimes, users need to pause sequences and resume them. They will affect all the axes in the sequence and if the SeqNoBit value contents more than one sequence. All the sequences will have the same results after the command is issued.

```
I16 pause_seq_move(I16 CardID, I16 SeqNoBit, F64
     Dec_Time);
I16 resume_seq_move(I16 CardID, I16 SeqNoBit, F64
     Acc_Time);
```

4.14.3 Coding Example 2: Compare Start Condition

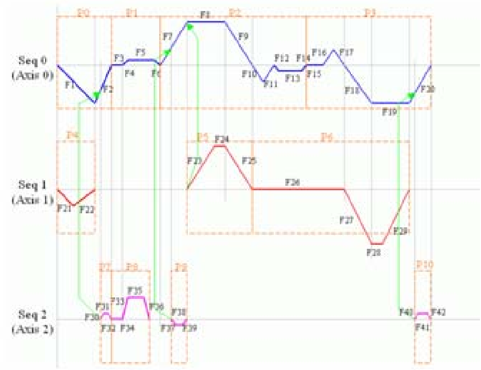


Figure 4-50: Coding Example 2

1. Variables Setting

```
I16 FirstFrame, LastFrame;
I16 AxisNo;
I16 SynAxes;
I16 PatternNo;
I16 WaitAxis, StartCondition;
```

2. Create Patterns for Sequence 0

```
AxisNo = 0;
SynAxes = 0x1;
PatternNo = 0;

// Pattern 0
FirstFrame = 0;
LastFrame =
     add_frame_ta_move(AxisNo,FirstFrame,0,-
     45,0,30,0,2,1);
set_pattern(0,PatternNo,FirstFrame, LastFrame-
     FirstFrame, SynAxes);
PatternNo++;

// Pattern 1
FirstFrame = LastFrame;
```

```
LastFrame = add_frame_dwell(AxisNo,FirstFrame,-
     45,1);
LastFrame = add_frame_ta_move(AxisNo,LastFrame,-
     45,-44.8,0,1,0,0.1,0.1);
set_pattern(0,PatternNo,FirstFrame, LastFrame-
     FirstFrame, SynAxes);
PatternNo++;

// Pattern 2
FirstFrame = LastFrame;
LastFrame = add_frame_ta_move(AxisNo,FirstFrame,-
     44.8,45,0,20,0,1,1);
LastFrame =
     add_frame_ta_move(AxisNo,LastFrame,45,41.5,
     0,2,0,2,2);
LastFrame =
     add_frame_ta_move(AxisNo,LastFrame,41.5,41.
     4,0,0.1,0,0.01,0.01);
set_pattern(0,PatternNo,FirstFrame, LastFrame-
     FirstFrame, SynAxes);
PatternNo++;

// Pattern 3
FirstFrame = LastFrame;
LastFrame =
     add_frame_dwell(AxisNo,FirstFrame,41.4,2);
LastFrame =
     add_frame_ta_move(AxisNo,LastFrame,41.4,45,
     0,5,0,1,1);
LastFrame =
     add_frame_ta_move(AxisNo,LastFrame,45,-
     45,0,10,0,2,2);
set_pattern(0,PatternNo,FirstFrame, LastFrame-
     FirstFrame, SynAxes);
PatternNo++;
```

3. Create Patterns for Sequence 1

```
AxisNo = 1;
SynAxes = 0x2;

// Pattern 4
FirstFrame = LastFrame;
```

```
LastFrame =
     add_frame_ta_move(AxisNo,FirstFrame,0,-
     15,0,15,0,1,1);
set_pattern(0,PatternNo,FirstFrame, LastFrame-
     FirstFrame, SynAxes);
PatternNo++;

// Pattern 5
FirstFrame = LastFrame;
LastFrame = add_frame_ta_move(AxisNo,FirstFrame,-
     15,45,0,30,0,1,1);
set_pattern(0,PatternNo,FirstFrame, LastFrame-
     FirstFrame, SynAxes);
PatternNo++;

// Pattern 6
FirstFrame = LastFrame;
LastFrame =
     add_frame_dwell(AxisNo,FirstFrame,45,1);
LastFrame =
     add_frame_ta_move(AxisNo,LastFrame,45,0,0,1
     0,0,1,1);
set_pattern(0,PatternNo,FirstFrame, LastFrame-
     FirstFrame, SynAxes);
PatternNo++;
```

4. Create Patterns for Sequence 2

```
AxisNo = 2;
SynAxes = 0x4;

// Pattern 7
FirstFrame = LastFrame;
LastFrame =
     add_frame_ta_move(AxisNo,LastFrame,0,0.05,0
     ,0.1,0,0.01,0.01);
set_pattern(0,PatternNo,FirstFrame, LastFrame-
     FirstFrame, SynAxes);
PatternNo++;

// Pattern 8
FirstFrame = LastFrame;
```

```
LastFrame =
     add_frame_dwell(AxisNo,FirstFrame,0.05,1);
LastFrame =
     add_frame_ta_move(AxisNo,LastFrame,0.05,0.2
     3,0,1,0,0.1,0.1);
set_pattern(0,PatternNo,FirstFrame, LastFrame-
     FirstFrame, SynAxes);
PatternNo++;

// Pattern 9
FirstFrame = LastFrame;
LastFrame =
     add_frame_ta_move(AxisNo,LastFrame,0.23,0,0
     ,1,0,0.1,0.1);
set_pattern(0,PatternNo,FirstFrame, LastFrame-
     FirstFrame, SynAxes);
PatternNo++;

// Pattern 10
FirstFrame = LastFrame;
LastFrame =
     add_frame_ta_move(AxisNo,LastFrame,0,0.05,0
     ,1,0,0.01,0.01);
set_pattern(0,PatternNo,FirstFrame, LastFrame-
     FirstFrame, SynAxes);
PatternNo++;
```

5. Insert pattern in sequences

```
// sequence 0
SynAxes = 0x1;
reset_seq_buffer(0,0);
WaitAxis = 0;
StartCondition = 0;
insert_pattern_to_seq_buffer(0,0,0,SynAxes,0,Wai
     tAxis,StartCondition,0);
insert_pattern_to_seq_buffer(0,0,1,SynAxes,0,Wai
     tAxis,StartCondition,0);
insert_pattern_to_seq_buffer(0,0,2,SynAxes,0,Wai
     tAxis,StartCondition,0);

// sequence 1
SynAxes = 0x2;
reset_seq_buffer(0,1);
```

```
WaitAxis = 0;
StartCondition = 0; // pattern no
insert_pattern_to_seq_buffer(0,1,4,SynAxes,0,Wai
     tAxis,StartCondition,0);
WaitAxis = 0;
StartCondition = 2;
insert_pattern_to_seq_buffer(0,1,5,SynAxes,1,Wai
     tAxis,StartCondition,1);
WaitAxis = 0;
StartCondition = 0;
insert_pattern_to_seq_buffer(0,1,6,SynAxes,0,Wai
     tAxis,StartCondition,0);

// sequence 2
SynAxes = 0x4;
reset_seq_buffer(0,2);
WaitAxis = 0;
StartCondition = 200;
insert_pattern_to_seq_buffer(0,2,7,SynAxes,1,Wai
     tAxis,StartCondition,-40);
WaitAxis = 0;
StartCondition = 0;
insert_pattern_to_seq_buffer(0,2,8,SynAxes,0,Wai
     tAxis,StartCondition,0);
WaitAxis = 0;
StartCondition = 102;
insert_pattern_to_seq_buffer(0,2,9,SynAxes,1,Wai
     tAxis,StartCondition,0);
```

6. Start sequence move and wait for buffer being empty

```
start_seq_move(0,7);

while( check_seq_buffer(0,0) == 0) ;
SynAxes = 0x1;
WaitAxis = 0;
StartCondition = 0;
insert_pattern_to_seq_buffer(0,0,3,SynAxes,0,Wai
     tAxis,StartCondition,0);

while( check_seq_buffer(0,2) == 0) ;
SynAxes = 0x4;
WaitAxis = 0;
StartCondition = 203;
```

```
insert_pattern_to_seq_buffer(0,2,10,SynAxes,1,Wa
     itAxis,StartCondition,-40);
```

7. Test Results

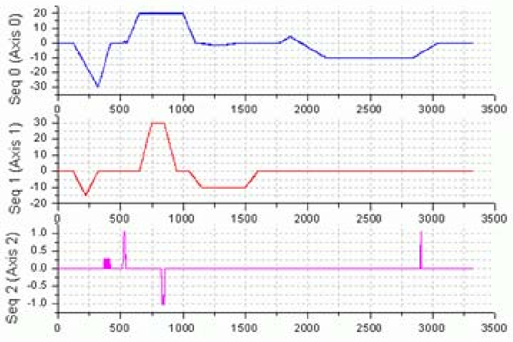


Figure 4-51: Test Results

5 Motion Creator

After installing all the hardware according to Chapters 2 and 3, it is necessary to correctly configure all cards and double check before running. This chapter gives guidelines for establishing a control system and manually testing the SSCNET board cards to verify correct operation. Motion Creator provides a simple yet powerful means to setup, configure, test and debugging a motion control system with the SSCNET board cards installed.

Note: Motion Creator is only available for Windows NT/2000/XP with a screen resolution higher than 800x600 and does not run under a DOS environment.

5.1 Overview

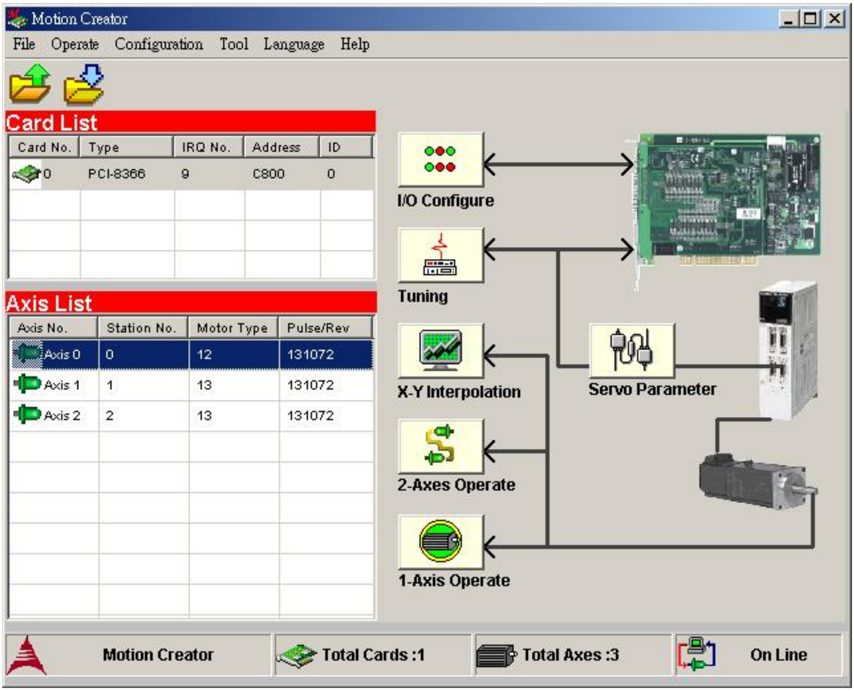
Motion Creator offers the following features and functionality:

- Language support for English, Chinese Traditional and Japanese
- 32-bit operation under Windows95/98/2000 and Windows NT
- Access and configuration of Multi-Axes control system
- Ability to access all of the servo driver parameter.
- Direct access to the general purpose I/O
- Full tuning capability for all servo driver and motion parameter
- XY-Interpolation
- Support for absolute and relative, trapezoidal and S-Curve, home return, and Continuous motion

Note: Motion Creator is available for Windows 2000 or Windows NT with the screen resolution higher than 800x600 environment and cannot run under a DOS environment.

5.2 Main Window

The diagram below is the Main window of Motion Creator when the program is executed. From the main window all SSCNET board cards inserted in the system and all axes connect are listed.



Card List

Card No.	Type	IRQ No.	Address	ID
0	PCI-8366	9	C800	0

Axis List

Axis No.	Station No.	Motor Type	Pulse/Rev
Axis 0	0	12	131072
Axis 1	1	13	131072
Axis 2	2	13	131072

Figure 5-1: Motion Creator Main Window

5.2.1 Component description

Toolbar

Use Motion Creator’s toolbar, to access the following functions

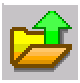


Figure 5-2: Load Servo Parameter From File

Load the servo parameter file from saved file. This file records the servo parameters of all axes in all cards.

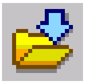


Figure 5-3: Save Servo Parameter to File

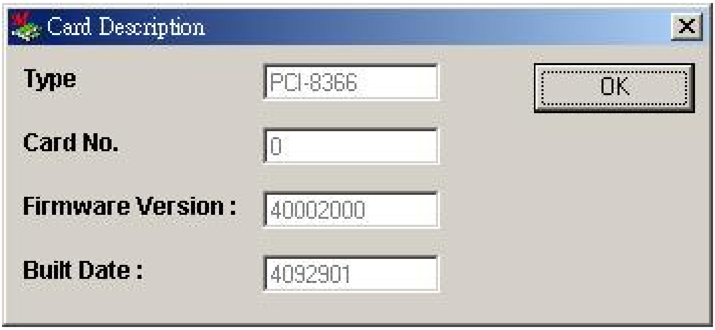
Save the servo parameters to file with a file extension “par”.

Language Support

Click the “Language” item in the menu bar, and select the language you want to display. (The language you select must be available in your operation system, for example: only Chinese-Traditional and English are available in Windows NT Chinese-Traditional Version.

Card list table

This table lists all SSCNET board cards plugged in the PCI-Bus. The "Card No" column displays the card index, the "Type" column display the card type, the "IRQ" column displays the IRQ number of the card, and the "Address" column displays the PCI-Bus base address of the card. The “ID” number column displays the card ID of the card. If the “ID” is a minus value. It means card initial fail. If you double click the card in the card list, it will displays the on board DSP firmware version as bellow:



Type PCI-8366

Card No. 0

Firmware Version : 40002000

Built Date : 4092901

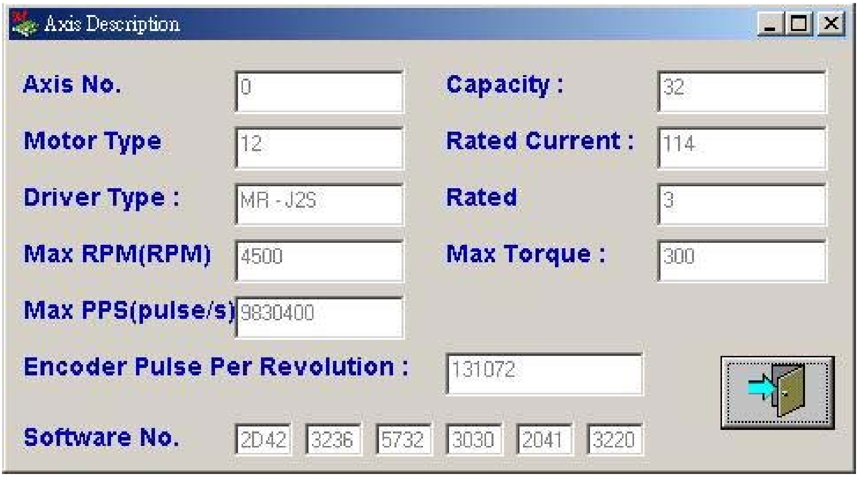
Figure 5-4: Card List Table

Axis list table

If you click one of the cards in the card list table, the axis list table lists all the axes connected to this card. The "Station No" column display the ID of each axis, the "Axis No" column display the index of the axis, the "Motor Type" display the motor type of the axis, the “Pulse/Rev” indicate the pulse per revolution of the axis. The default value is 131072.

Axis information

If you double click the axis in the axis list table, the axis information window will appear, and display the information of the axis.



Axis No.	0	Capacity :	32
Motor Type	12	Rated Current :	114
Driver Type :	MR - J2S	Rated	3
Max RPM(RPM)	4500	Max Torque :	300
Max PPS(pulse/s)	9830400		

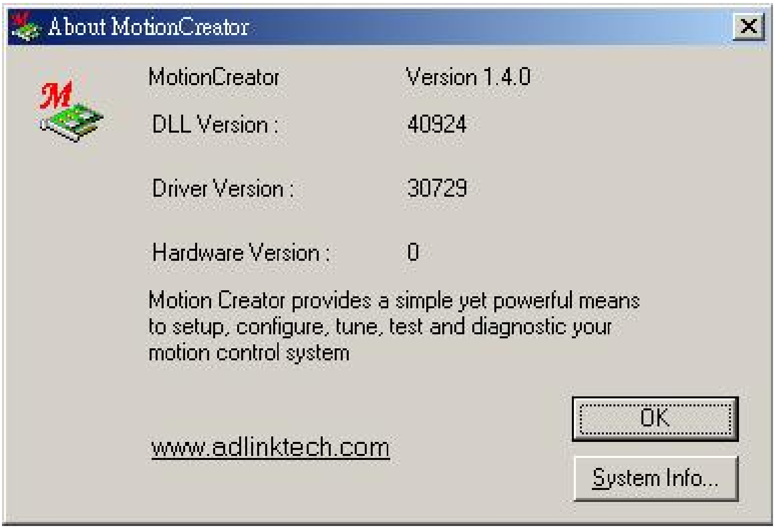
Encoder Pulse Per Revolution : 131072

Software No. 2D42 3236 5732 3030 2041 3220

Figure 5-5: Axis Information

Software version Information

Check the software version from the help menu bar. It looks like below:



MotionCreator Version 1.4.0

DLL Version : 40924

Driver Version : 30729

Hardware Version : 0

Motion Creator provides a simple yet powerful means to setup, configure, tune, test and diagnostic your motion control system

www.adlinktech.com

Figure 5-6: Software Version Information

Command buttons

The functionality of command buttons are described following

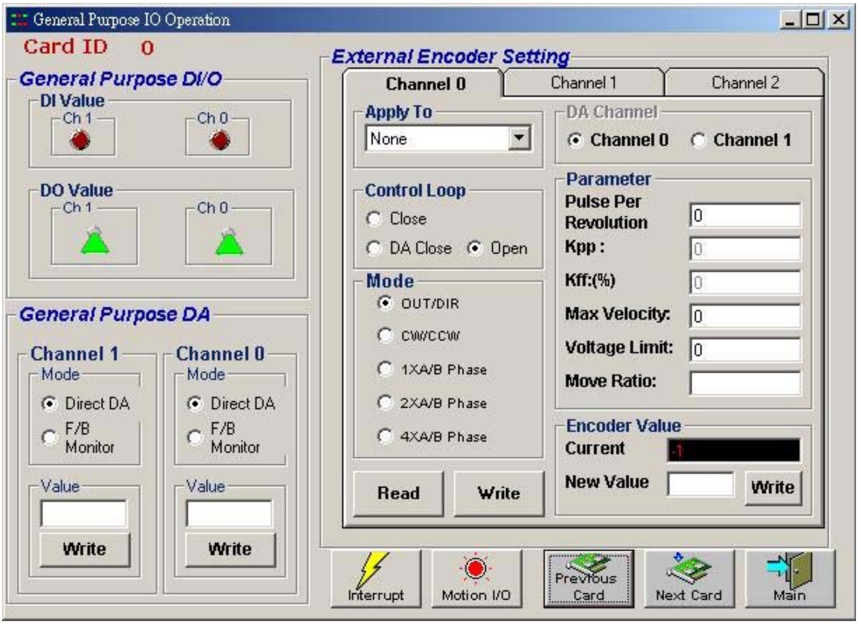
- I/O Configure
  - General purpose digital input and output
  - General purpose Analog output
  - External Encoder setting
- Tuning
  - Trigger setting
  - Basic servo driver parameter setting
  - Multiple channel display and adjustment
- XY-Interpolation
  - Circular interpolation
  - Linear interpolation
  - 2-D graph of command and feedback trajectory
- 2-Axes Operate
  - Two-axes motion
  - Driver status display
  - Relative, absolute and repeat motion mode
  - Velocity profile display
- 1-Axes Operate
  - Driver status display
  - Support Trapezoidal, S-Curve, Home return, Continuous motion
  - Relative, absolute and repeat motion mode
  - Velocity profile display
  - On the fly change of Velocity and position
- Servo Parameter
  - Servo driver parameter configuration
  - Default setting
  - Parameter description
  - Servo Parameter

5.2.2 Operation Steps

1. Check if all the SSCNET cards, which are plugged into the PCI-Bus show on the “Card List” table, then click each card in the card list table and check if all the axes are displayed. If not all of the axes listed in the table, please quit MotionCreator and restart again.
2. Select the axis in the “Axis List” table
3. Clicks the command button to operate.

5.3 General Purpose IO Operation Window (PCI-8372+/8366+)

General Purpose IO Operation Window appears when clicking “I/O Configure” button in the Main window. Figure “I/O Configure” shows the General Purpose IO Operation Window.



General Purpose IO Operation

Card ID 0

General Purpose DI/O

DI Value: Ch 1 ●, Ch 0 ●

DO Value: Ch 1, Ch 0

General Purpose DA

Channel 1 — Mode: ● Direct DA ○ F/B Monitor — Value: (empty) — Write

Channel 0 — Mode: ● Direct DA ○ F/B Monitor — Value: (empty) — Write

External Encoder Setting — Channel 0 | Channel 1 | Channel 2

Apply To: None

DA Channel: ● Channel 0 ○ Channel 1

Control Loop: ○ Close ○ DA Close ● Open

Mode: ● OUT/DIR ○ CW/CCW ○ 1XA/B Phase ○ 2XA/B Phase ○ 4XA/B Phase

Parameter

Pulse Per Revolution: 0

Kpp : 0

Kff:(%) 0

Max Velocity: 0

Voltage Limit: 0

Move Ratio:

Encoder Value

Current: -1

New Value:

Figure 5-7: General Purpose IO Operation Window

5.3.1 Component description

The General Purpose IO Operation Window is divided into several frames. Each frame is described as follows:

General Purpose DI/O

There are two digital input and 2 digital output channels in SSCNET board

1. The circular buttons show the status of two digital input channels.
2. Click the rectangle button to write the digital output value for each digital output channel.

General Purpose DA

There are two analog output channels in SSCNET board.

The current value textboxes read back the current value of two analog output channels.

Enter the analog output value in the textbox then click the “Set Value” button to write the analog output value.

External Encoder Setting

- SSCNET board includes three external encoder channels.

Value

- If the external encoder channels are used, and the signals are connected, you can read the encoder value for each channel.
- Enter the new value of encoder in the textbox then click the “Set” button to write the value.

Apply To

- Specify the axis that uses the external encoder signal

Mode

- Select the attribute of each external encoder signal, if the external encoder signals are connected.

Control Loop

- The attribute of the control loop for the external encoder

Parameter

- The corresponding parameter for the external encoder

5.3.2 Operation Steps

The General Purpose IO Operation Window accesses the digital input, output and analog output value of the SSCNET board. The operation steps are described as follows:

General Purpose DI/O

- Digital input: the circular buttons display and update the current status of two digital input channels in the scan rate of 100 ms.
- Digital output: click the rectangle button to write the digital output value for each digital output channel.

General Purpose DA

- Analog output: enter the analog output value in the textbox then click the “Set Value” button to write the analog output value.
- The current value textboxes read back the current value of two analog output channels automatically.

External Encoder Setting

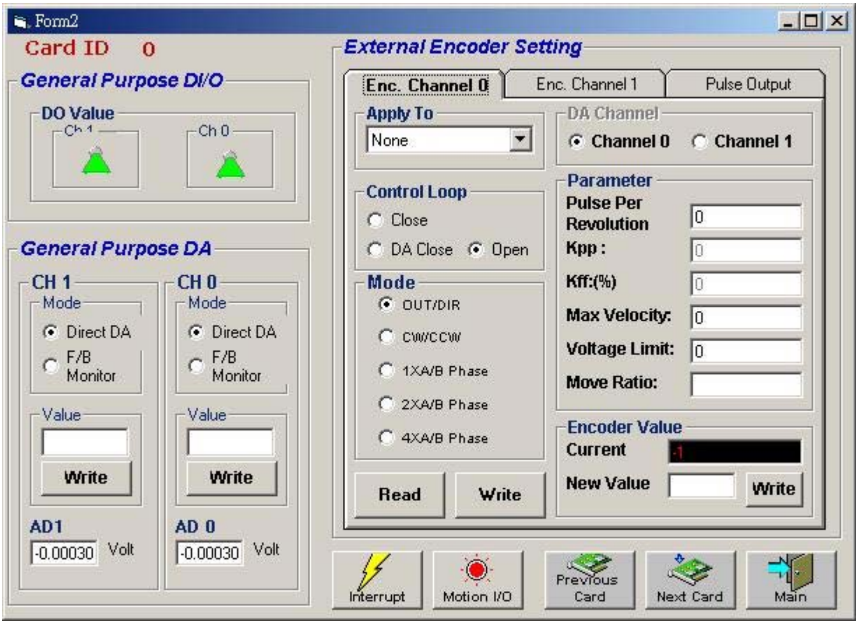
- Mode: Select the attribute of each external encoder signal, if the external encoder signals are connected.
- Apply To: select the axis that uses the external encoder
- Control Loop: select open or close loop control.
- Parameter: enter the corresponding parameter for control loop

Read and write encoder value

- If the external encoder channels are used, and the signals are connected, you can read the encoder value for each channel.
- Enter the new value of encoder in the textbox then click the “Set” button to write the value.

5.4 General Purpose IO Operation Window (cPCI-8312H)

General Purpose IO Operation Window appears when clicking “I/O Configure” button in the Main window. Figure “I/O Configure” shows the General Purpose IO Operation Window.



Card ID 0

Apply To: None

DA Channel: ● Channel 0 ○ Channel 1

Control Loop: ○ Close ○ DA Close ● Open

Mode: ● OUT/DIR ○ CW/CCW ○ 1XA/B Phase ○ 2XA/B Phase ○ 4XA/B Phase

Pulse Per Revolution: 0

Kpp : 0

Kff:(%) 0

Max Velocity: 0

Voltage Limit: 0

Move Ratio:

Encoder Value Current: -1

CH 1 Mode: ● Direct DA ○ F/B Monitor

CH 0 Mode: ● Direct DA ○ F/B Monitor

AD1 -0.00030 Volt

AD 0 -0.00030 Volt

Figure 5-8: General Purpose IO Operation Window

Component description

The General Purpose IO Operation Window is divided into several frames. Each frame is described as follows:

General Purpose DO

There are two digital output channels in SSCNET board

Click the rectangle button to write the digital output value for each digital output channel.

General Purpose DA/AD

There are two analog output and input channels in SSCNET board.

The current value textboxes read back the current value of two analog output channels.

Enter the analog output value in the textbox then click the "Set Value" button to write the analog output value.

The AD0/AD1 will read back the current analog input value of two channels.

External Encoder Setting

- SSCNET board includes three external encoder channels.

Value

- If the external encoder channels are used, and the signals are connected, you can read the encoder value for each channel.
- Enter the new value of encoder in the textbox then click the "Set" button to write the value.

Apply To

- Specify the axis that uses the external encoder signal

Mode

- Select the attribute of each external encoder signal, if the external encoder signals are connected.

Control Loop

- The attribute of the control loop for the external encoder

Parameter

- The corresponding parameter for the external encoder

5.4.1 Operation Steps

The General Purpose IO Operation Window accesses the digital input, output and analog output value of the SSCNET board. The operation steps are described as follows:

General Purpose DI/O

- Digital output: click the rectangle button to write the digital output value for each digital output channel.

General Purpose AD/DA

- Analog output: enter the analog output value in the textbox then click the “Set Value” button to write the analog output value.
- The current value textboxes read back the current value of two analog output channels automatically.
- The AD0/AD1 will read back the current analog input value of two channels.

External Encoder Setting

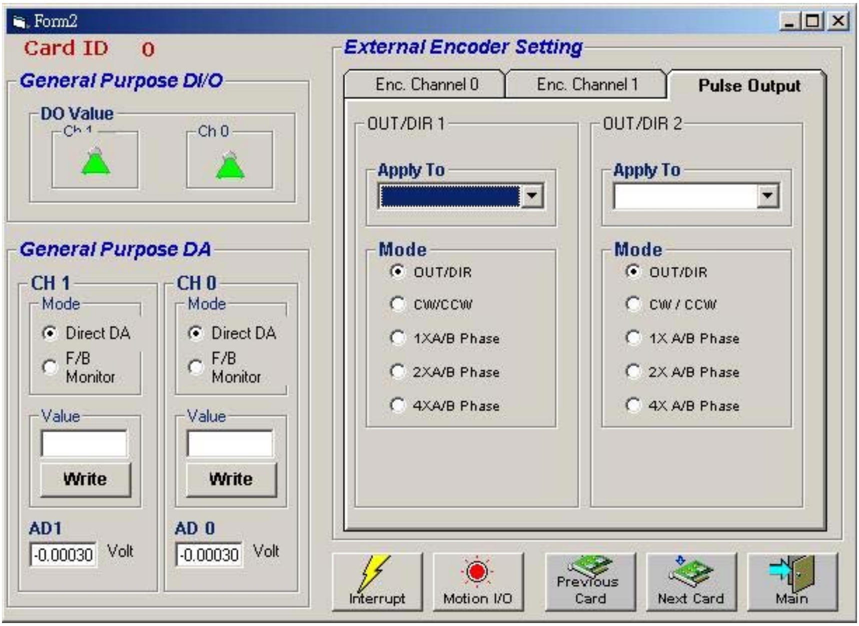
- Mode: Select the attribute of each external encoder signal, if the external encoder signals are connected.
- Apply To: select the axis that uses the external encoder
- Control Loop: select open or close loop control.
- Parameter: enter the corresponding parameter for control loop

Read and write encoder value

- If the external encoder channels are used, and the signals are connected, you can read the encoder value for each channel.
- Enter the new value of encoder in the textbox then click the “Set” button to write the value.

5.4.2 Pulse Output Page

This SSCNET board provide two channels of pulse output function. Users can use these two channel to control stepper or any other pulse input command motor.



Form2

Card ID 0

General Purpose DI/O

DO Value: Ch 1, Ch 0

General Purpose DA

CH 1 — Mode: ● Direct DA ○ F/B Monitor — Value: — AD1 -0.00030 Volt

CH 0 — Mode: ● Direct DA ○ F/B Monitor — Value: — AD 0 -0.00030 Volt

External Encoder Setting

Enc. Channel 0 | Enc. Channel 1 | Pulse Output

OUT/DIR 1 — Apply To: — Mode: ● OUT/DIR ○ CW/CCW ○ 1XA/B Phase ○ 2XA/B Phase ○ 4XA/B Phase

OUT/DIR 2 — Apply To: — Mode: ● OUT/DIR ○ CW / CCW ○ 1X A/B Phase ○ 2X A/B Phase ○ 4X A/B Phase

Figure 5-9: Pulse Output

5.4.3 Component description

Apply To

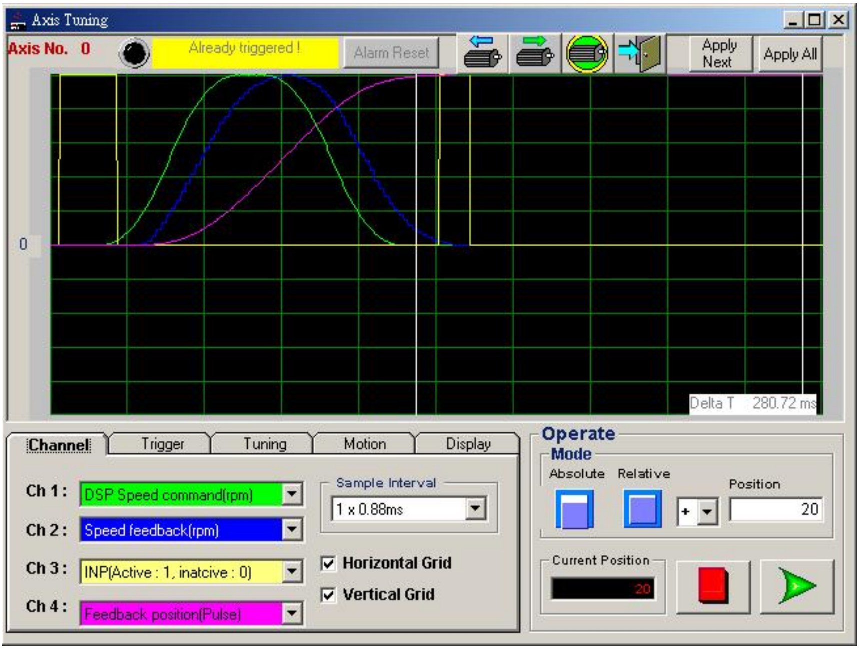
- Specify the axis that uses pulse output function. Notice that the axis number can’t be overlapped by SSCNET axis.

Mode

- Select the attribute of pulse output signal.

5.5 Tuning Window

Tuning Window appears when clicking “Tuning” button in the Main window. The following figure shows the Tuning Window. This window displays the response diagrams of selected channels by setting the motion parameter in “Single-Axis Operation Window” and trigger setting in this window.



Axis No. 0

Already triggered !

0

Delta T 280.72 ms

Ch 1 : DSP Speed command(rpm)

Ch 2 : Speed feedback(rpm)

Ch 3 : INP(Active : 1, inatcive : 0)

Ch 4 : Feedback position(Pulse)

Sample Interval: 1 x 0.88ms

☑ Horizontal Grid

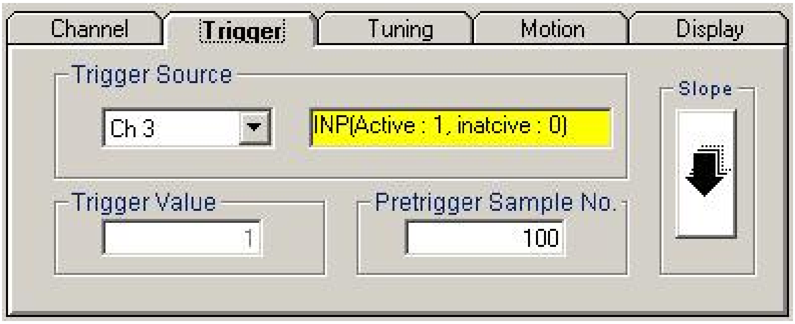
☑ Vertical Grid

Position: + 20

Current Position: 20

Figure 5-10: Tuning Window

5.5.1 Component Description



Trigger Source: Ch 3 INP(Active : 1, inatcive : 0)

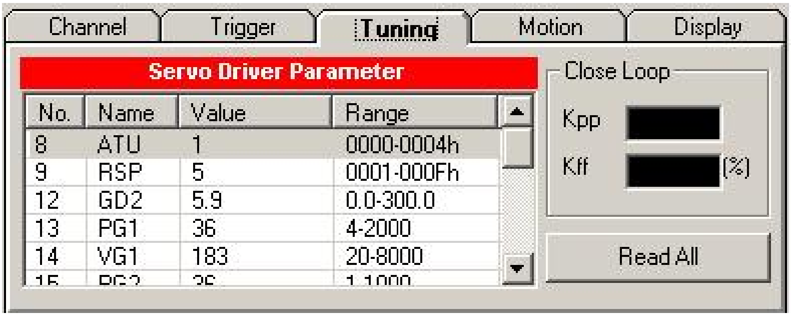
Trigger Value: 1

Pretrigger Sample No.: 100

Figure 5-11: Trigger Setting Frame

This frame provides a flexible choice to configure the trigger. Once the signal is triggered, the data from the four channels will be plotted on the response diagram.

- Source: select one of the channel signal to be the trigger source
- Value: trigger value
- Slope: specify the rising edge or falling edge trigger
- Sample Number: total amount of the gathering data
- Pretrigger sample No.: amount of the pretrigger data



No.	Name	Value	Range
8	ATU	1	0000-0004h
9	RSP	5	0001-000Fh
12	GD2	5.9	0.0-300.0
13	PG1	36	4-2000
14	VG1	183	20-8000

Figure 5-12: Parameter Tuning Frame

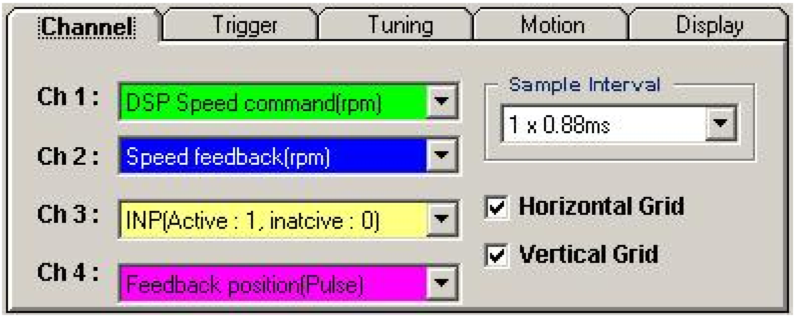
This frame affords an easy way to access a set of fundamental servo parameter.

Read All: read the current servo parameter

Channel Selection Frame

This frame is used to set the signal of each channel.

Sample interval: the sample interval between signals. The units of the X-Axis is sample interval.



Channel | Trigger | Tuning | Motion | Display

Ch 1 : DSP Speed command(rpm)

Ch 2 : Speed feedback(rpm)

Ch 3 : INP(Active : 1, inatcive : 0)

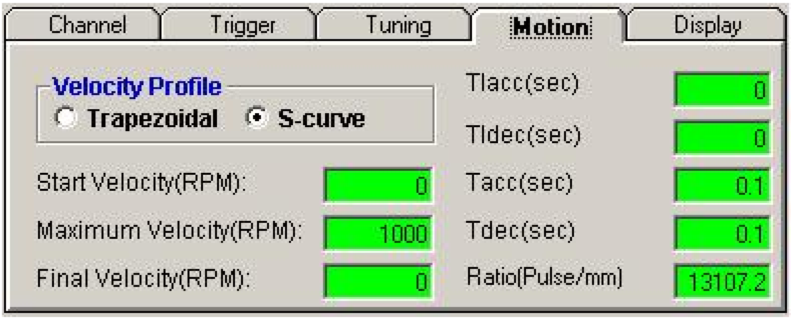
Ch 4 : Feedback position(Pulse)

Sample Interval: 1 x 0.88ms

☑ Horizontal Grid

☑ Vertical Grid

Figure 5-13: Channel Selection Frame



Channel | Trigger | Tuning | Motion | Display

Velocity Profile: ○ Trapezoidal ● S-curve

Start Velocity(RPM): 0

Maximum Velocity(RPM): 1000

Final Velocity(RPM): 0

Tlacc(sec): 0

Tldec(sec): 0

Tacc(sec): 0.1

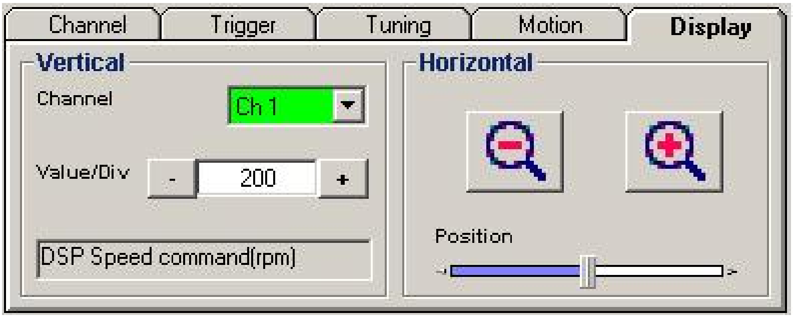
Tdec(sec): 0.1

Ratio(Pulse/mm): 13107.2

Figure 5-14: Motion Frame

This frame is used to construct a motion.

- Velocity profile: Select the Trapezoidal or S-Curve velocity profile.
- Start Velocity Set the start velocity of motion in unit of PRM.
- Maximum Velocity: Set the maximum velocity of motion in unit of PRM.
- Final Velocity: Set the finvel velocity of motion in unit of PRM.
- Tacc: Set the total acceleration time in unit of second.
- Tdec: Set the total deceleration time in unit of second.
- Tlacc: Set the linear acceleration time in unit of second.
- Tldec: Set the linear deceleration time in unit of second.
- Ratio: Set the move ratio between pulse and displacement



Channel: Ch 1

Value/Div: 200

DSP Speed command(rpm)

Figure 5-15: Display Frame

- Vertical:
  - Channel: Select the channel you want to adjust.
  - Scale: Adjust he current scale of selected channel.
  - Position: Shift the data of selected channel.
- Horizontal:
  - Zoom in
  - Zoom out
  - Position

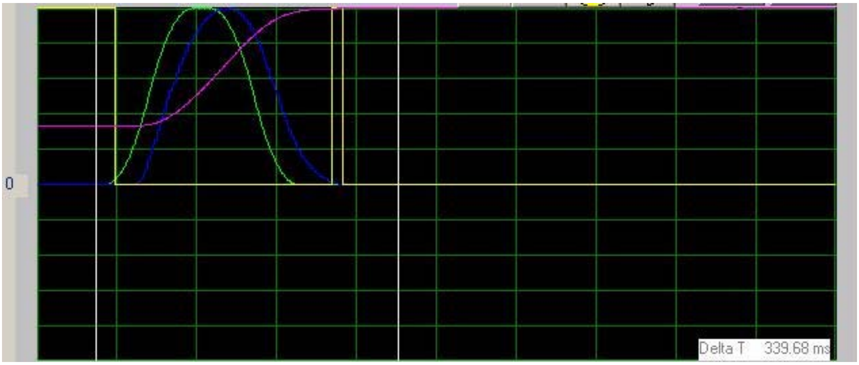


Figure 5-16: Response Diagram

This diagram displays the waveform from four channels in different colors.

Timing Line

There are two timing lines in the response diagram, and the time difference between two lines will shows in the left corner of response diagram.

Play Keys

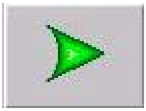


Figure 5-17: Play Button

Click this button will cause SSCNET board start to move.

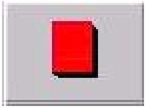


Figure 5-18: Stop Button

Click “Stop” button will cause SSCNET board to decelerate to stop.

5.5.2 Operation Steps

The operation steps are description as follows:

- Click “Channel” tab to specify the signal data, and sample interval
- Click “Trigger” tab to set trigger source, trigger value, slope, Sample Number, and pretrigger sample No.
- Click “Motion” tab to set the motion parameter.
- Click “Play button” to cause SSCNET board start to move.
- If the signal is triggered, and the data is shown on the response diagram, then click the “Display” tab to adjust the data.
- If you want to change the response of servomotor, click “Tuning” tab to modify the servo parameter to change the response

5.5.3 Example

- Click “Channel” tab, select “F/B present value” for channel 1, “INP(In Position)” for channel 2, “Speed Command” for channel 3, “Speed Feedback” for channel 4, and set “Sample Interval” = 1x0.88 ms.
- Click “Trigger” tab, select “Ch 2” to be the trigger source, trigger value = 1, down slope, 1 x0.88 ms sample interval.
- Click “Motion” tab, select “relative ” motion mode, “Distance” = 20000, "Trapezoidal" velocity profile, "Start Velocity" = 1000, "Maximum Velocity" = 3000, "Final Velocity" = 1000, "Tacc" = 0.1, "Tdec" = 0.1.
- Click the "Play" button to start motion.
- When the “INP signal” is changing from 1 to 0, the signal is triggered and system starts to record the data for four channels by the 0.88ms time interval, and the data is adjusted and shown in the response diagram.
- If the data is shown on the response diagram correctly, you can click “Display” tab to scale or shift the data.

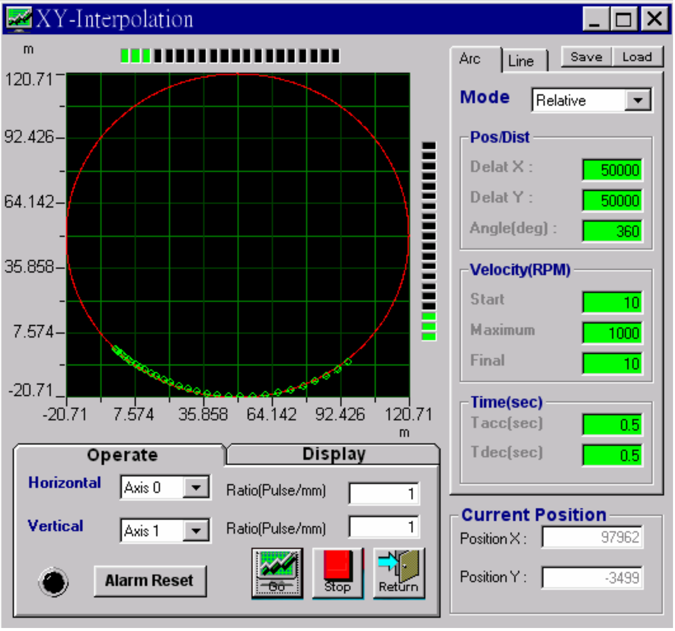
Timing

- ▶ Move the cursor to the “timing line”.
- ▶ Drag the “timing line to” any position.
- ▶ The textbox in left corner in “Response Diagram” will indicate the time difference between two lines.

Note: The range of Y-Axis in the response diagram is –1000 to 1000, if the scaled data exceeds this range, it will not display on the diagram.

5.6 XY-Interpolation Window

XY-Interpolation Window appears when clicking “XY-Interpolation” button in the Main window. The following figure shows the XY-Interpolation Window.



Mode Relative

Pos/Dist
Delat X : 50000
Delat Y : 50000
Angle(deg) : 360

Velocity(RPM)
Start 10
Maximum 1000
Final 10

Time(sec)
Tacc(sec) 0.5
Tdec(sec) 0.5

Horizontal Axis 0 Ratio(Pulse/mm) 1
Vertical Axis 1 Ratio(Pulse/mm) 1

Current Position
Position X : 97962
Position Y : -3499

Figure 5-19: XY-Interpolation Window

5.6.1 Component description

Position Graph

This graph shows the feedback and command position of the interpolation dynamically.

Parameter Page

The parameter page affords a friendly and intelligent interface to configure the interpolation motion.

Control Panel

The control panel starts or stops the interpolation motion, set the horizontal, vertical axis for the interpolation, and scale or shift the data.

Position Display

The position display shows the current position of the horizontal and vertical axis in the unit of pulse.

Velocity Display

The Velocity display shows the current speed ratio (current speed/ motor maximum speed) of the horizontal and vertical axis.

5.6.2 Operation steps

The operation steps of XY-Interpolation Window are described as following:

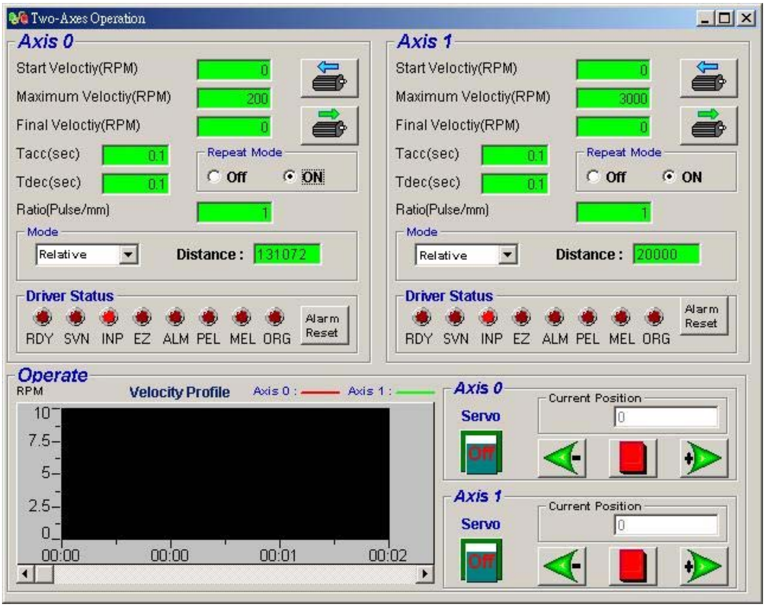
- Click “Tab 0” to select the circular or liner mode of Interpolation
- Select the relative or absolute interpolation mode
- Input the require parameters, the require parameters are in the green background color
- Select the axes for horizontal and vertical direction
- Press the “Go button” to start motion
- Click “Display” tab to scale or shift the data.

Note: 1. The XY-Interpolation is available when more than two axes exist in your system.

2. If alarm happens (for example: accelerate too fast), the motion will be interrupted, you must click the “alarm reset” button to reset the alarm status.

5.7 Two-Axes Operation Window

Two-Axes Operation Window appears when clicking “2-Axis Operate” button in the Main window. The following figure shows the Two-Axes Operation Window. This window affords the simple control of motion (relative or absolute trapezoidal mode), and displays the velocity profile, driver status for the users.



Two-Axes Operation

Axis 0

Start Velociiy(RPM) 0
Maximum Velociiy(RPM) 200
Final Velociiy(RPM) 0
Tacc(sec) 0.1
Tdec(sec) 0.1
Ratio(Pulse/mm) 1
Repeat Mode: ○ Off ● ON
Mode: Relative
Distance : 131072

Axis 1

Start Velociiy(RPM) 0
Maximum Velociiy(RPM) 3000
Final Velociiy(RPM) 0
Tacc(sec) 0.1
Tdec(sec) 0.1
Ratio(Pulse/mm) 1
Repeat Mode: ○ Off ● ON
Mode: Relative
Distance : 20000

Operate

Axis 0 — Servo: Off — Current Position: 0

Axis 1 — Servo: Off — Current Position: 0

Figure 5-20: Two-Axes Operation Window

5.7.1 Component description

Axis 0, Axis 1 frame

These frames display the required parameters for motion; the parameters are described below:

- Start Velocity: Set the start velocity of motion in unit of PRM. In “Absolute Mode” or “Relative Mode”, only the value is effective. ie, -100.0 is the same as 100.0. In “Cont. Move”, both the value and sing is effective. –100.0 means 100.0 in minus direction.

- Maximum Velocity: Set the maximum velocity of motion in unit of PRM. In “Absolute Mode” or “Relative Mode”, only the value is effective. ie, -5000.0 is the same as 5000.0. In “Cont. Move”, both the value and sing is effective. –5000.0 means 5000.0 in minus direction.
- Final Velocity: Set the final velocity of motion in unit of PRM. In “Absolute Mode” or “Relative Mode”, only the value is effective. ie, -5000.0 is the same as 5000.0. In “Cont. Move”, both the value and sing is effective. –5000.0 means 5000.0 in minus direction.
- Tacc: Set the total acceleration time in unit of seconds.
- Tdec: Set the total deceleration time in unit of seconds.
- Ratio (Pulse/mm): set the move ratio between pulse and displacement
- Relative or absolute mode for each axis:
- Absolute Mode: “Position” will be used as absolution target position for motion
- Relative Mode: “Distance will” be used as relative displacement for motion.
- Repeat mode: When “On” is selected, the motion will go in repeat mode (Positive distance <--> Negative distance or Positive position <--> Negative position).

Driver Status frame

This frame monitors the driver status in the update rate of 50ms.

Velocity Chart

This chart displays the velocity profile of each axis

Play keys

- Right play button: Click this button will cause SSCNET board start to outlet pulses according to previous setting.
- In “Relative Mode”, it cause axis move Positive Distance.
- In “Absolute Mode”, it cause axis move to Positive Position.
- Left play button: Click this button will cause SSCNET board start to outlet pulses according to previous setting.
- In “Relative Mode”, it cause axis move Negative Distance.
- In “Absolute Mode”, it cause axis move to Negative Position.
- Stop button: Click “Stop” button will cause SSCNET board to decelerate to stop. The deceleration time is defined in parameter “Tdec”.

5.7.2 Operation Steps

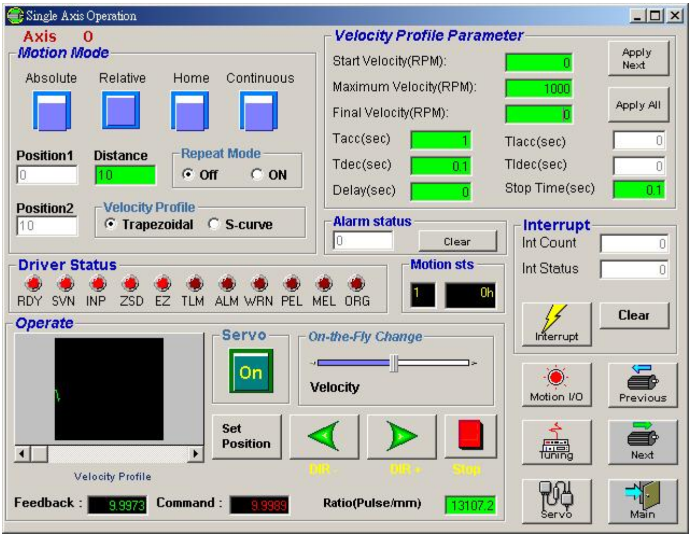
- Select the motion mode, and input the require parameters for each axis, the require parameters are in the green background color.
- Click the “Servo” button to keep the “Servo-On” state
- Click right play, and left play button to start the motion.
- Click “Stop” to stop the motion.

When the motion starts, the feedback velocity profile will display in the velocity chart, and the driver status is also display in this window.

Note: If alarm happens, the motion will be interrupted, you must click the “alarm reset” button to reset the alarm status.

5.8 Single Axis Operation Window

Single Axis Window appears when clicking “1-Axis Operate” button in the Main window. The following figure shows the Single Axis Window. This window supports the full control of a single axis motion, and displays the velocity profile and driver status.



Single Axis Operation

Axis 0

Motion Mode

Position1: 0

Distance: 10

Repeat Mode: ● Off ○ ON

Position2: 10

Velocity Profile: ● Trapezoidal ○ S-curve

Velocity Profile Parameter

Start Velocity(RPM): 0

Maximum Velocity(RPM): 1000

Final Velocity(RPM): 0

Tacc(sec): 1

Tdec(sec): 0.1

Delay(sec): 0

Tlacc(sec): 0

Tldec(sec): 0

Stop Time(sec): 0.1

Alarm status: 0

Interrupt

Int Count: 0

Int Status: 0

Motion sts: 1 0h

Feedback : 9.9973 Command : 9.9989 Ratio(Pulse/mm) 13107.2

Figure 5-21: Single Axis Operation Window

5.8.1 Component description

Motion Mode Frame

This frame provides selection of all modes for single axis motion; each mode is described below.

Motion Parameters Frame

This frame displays the require parameters for motion, the parameters are described below:

- Start Velocity: Set the start velocity of motion in unit of PRM. In “Absolute Mode” or “Relative Mode”, only the value is effective. ie, -100.0 is the same as 100.0. In “Cont. Move”, both the value and sing is effective. –100.0 means 100.0 in minus direction.

- Maximum Velocity: Set the maximum velocity of motion in unit of PRM. In “Absolute Mode” or “Relative Mode”, only the value is effective. ie, -5000.0 is the same as 5000.0. In “Cont. Move”, both the value and sing is effective. –5000.0 means 5000.0 in minus direction.
- Final Velocity: Set the final velocity of motion in unit of PRM. In “Absolute Mode” or “Relative Mode”, only the value is effective. ie, -5000.0 is the same as 5000.0. In “Cont. Move”, both the value and sing is effective. –5000.0 means 5000.0 in minus direction.
- Ratio (Pulse/mm): set the move ratio between pulse and displacement
- Tacc: Set the total acceleration time in unit of seconds.
- Tdec: Set the total deceleration time in unit of seconds.
- Tlacc: Set the linear acceleration time in unit of seconds.
- Tldec: Set the linear deceleration time in unit of seconds.

Driver Status frame

This frame monitors the driver status in the update rate of 50ms.

Motion Status frame

Left part is the motion done status and the right part is the motion status in heximal.

Velocity Chart

This chart displays the velocity profile of each axis

Play keys

Right play button: Clicking this button will cause the SSCNET board to start outputting pulses according to a previous setting.

- In “Relative Mode”, it cause axis move Positive Distance
- In “Absolute Mode”, it cause axis move to Positive Position
- In “Continuous Mode”, it cause axis start to move according to the velocity setting

Left play button: Clicking this button will cause the SSCNET board to start outputting pulses according to a previous setting.

- ▶ In “Relative Mode”, it cause axis move Negative Distance
- ▶ In “Absolute Mode”, it cause axis move to Negative Position
- ▶ In “Continuous Mode”, it cause axis start to move according to the velocity setting

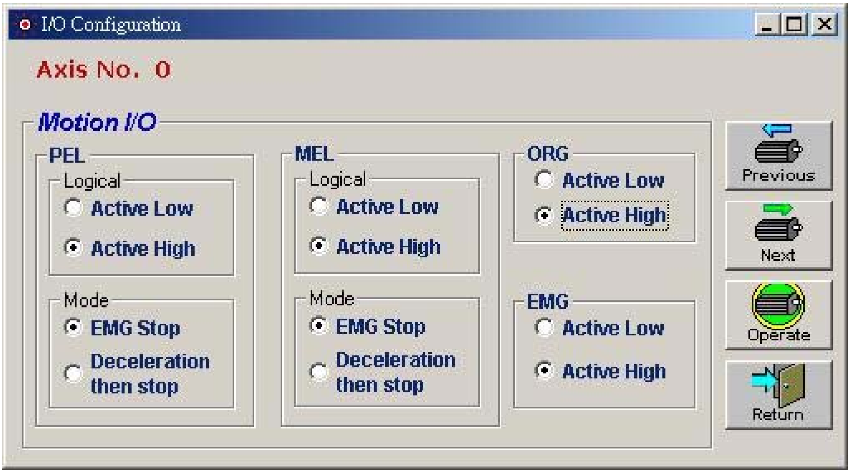
Stop button: Clicking the “Stop” button will cause the SSCNET board to decelerate to stop. The deceleration time for a "Trapezoidal Velocity Profile" is defined in parameter “Tdec”, and for "S-Curve Velocity Profile" is defined in parameter “Tdec”, "Tldec".

Change Velocity On The Fly Button: click this button to change the velocity of current motion. The new velocity must be defined in “Maximum Velocity”

5.8.2 Motion I/O Configration Window

If you press the “Motion I/O” button, you will see a window below:

It is for users to set their dedicated I/O of the axis. The setting will be automatically saved by MotionCreator.



I/O Configuration

Axis No. 0

Motion I/O

PEL
Logical
○ Active Low
● Active High
Mode
● EMG Stop
○ Deceleration then stop

MEL
Logical
○ Active Low
● Active High
Mode
● EMG Stop
○ Deceleration then stop

ORG
○ Active Low
● Active High

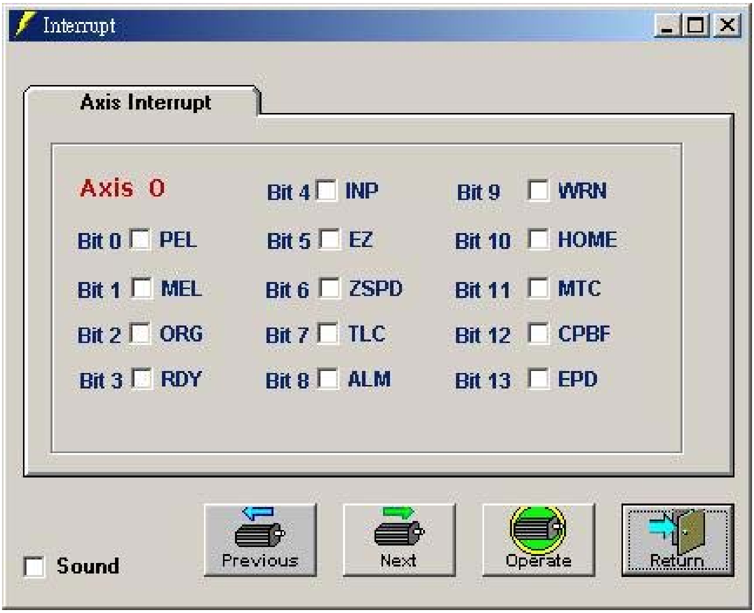
EMG
○ Active Low
● Active High

Figure 5-22: Motion I/O Configration Window

5.8.3 Interrupt Configration Window

If you press the “Interrupt” button, you will see a window below:

It is for users to set interrupt factor the axis. The setting will be automatically saved by MotionCreator. It is userful to users to test the interrupt functions.



Interrupt

Axis Interrupt

Axis 0

Bit 0 □ PEL	Bit 4 □ INP	Bit 9 □ WRN
Bit 1 □ MEL	Bit 5 □ EZ	Bit 10 □ HOME
Bit 2 □ ORG	Bit 6 □ ZSPD	Bit 11 □ MTC
Bit 3 □ RDY	Bit 7 □ TLC	Bit 12 □ CPBF
	Bit 8 □ ALM	Bit 13 □ EPD

□ Sound

Figure 5-23: Interrupt Configration Window

5.8.4 Operation Steps

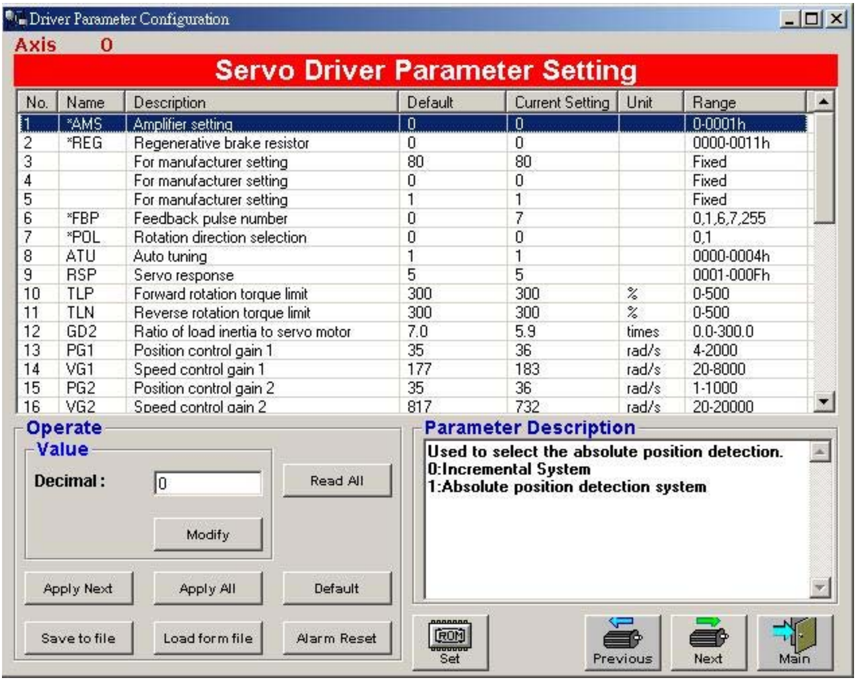
- Selecting a motion mode:
  - Absolute Mode: “Position1” and “position2” will be used as absolute target position for motion
  - Relative Mode: “Distance will” be used as relative displacement for motion.
  - Home Mode: The Motion keeps going until the ORG signal is active.
  - Continuous Mode: The Motion keeps going until the “stop” button is clicked.
  - Repeat Mode: When “On” is selected, the motion will go in repeat mode (forward?? backward or position1 ?? position2). It is only effective when “Relative Mode” or “Absolute Mode” is selected.
  - Velocity Profile: Select the velocity profile. Both Trapezoidal and S-Curve are available for “Absolute Mode”, “Relative Mode” and “Continuous Mode”.
- Input the require parameters, the require parameters are in the green background color.
- Click the “Servo” button to keep the “Servo-On” state
- Click right play, and left play button to start the motion.
- Change Position On The Fly Button: When this button is enabled, users can change the target position of current motion. The new position must be defined in “Position2”.
- Change Velocity On The Fly Button: When this button is enabled, users can change the velocity of current motion. The new velocity must be defined in “Maximum Velocity”
- Stop Motion: Click “Stop” will cause SSCNET board to decelerate to stop.

When the motion starts, the command and feedback velocity profile will display in the velocity chart, and the driver status is also displayed in this window.

Note: If alarm happens, the motion will be interrupted, you must click the “alarm reset” button to reset the alarm status.

5.9 Driver Parameter Configuration Window

Driver Parameter Configuration Window appears when clicking "Servo Parameter" button from the Main window. The following figure shows the Driver Parameter Configuration Window. This window supports full access to all servo driver parameters.


Driver Parameter Configuration
Axis 0
Servo Driver Parameter Setting

No.	Name	Description	Default	Current Setting	Unit	Range
1	*AMS	Amplifier setting	0	0		0-0001h
2	*REG	Regenerative brake resistor	0	0		0000-0011h
3		For manufacturer setting	80	80		Fixed
4		For manufacturer setting	0	0		Fixed
5		For manufacturer setting	1	1		Fixed
6	*FBP	Feedback pulse number	0	7		0,1,6,7,255
7	*POL	Rotation direction selection	0	0		0,1
8	ATU	Auto tuning	1	1		0000-0004h
9	RSP	Servo response	5	5		0001-000Fh
10	TLP	Forward rotation torque limit	300	300	%	0-500
11	TLN	Reverse rotation torque limit	300	300	%	0-500
12	GD2	Ratio of load inertia to servo motor	7.0	5.9	times	0.0-300.0
13	PG1	Position control gain 1	35	36	rad/s	4-2000
14	VG1	Speed control gain 1	177	183	rad/s	20-8000
15	PG2	Position control gain 2	35	36	rad/s	1-1000
16	VG2	Speed control gain 2	817	732	rad/s	20-20000

Decimal : 0

Used to select the absolute position detection.
0:Incremental System
1:Absolute position detection system

Figure 5-24: Driver Parameter Configuration Window

5.9.1 Component description

Servo Driver Parameter Table

This table lists all the accessible servo driver parameters, and the attribute of the parameter. Each column in this table is described as following

- Name: the short name of the parameter. For any parameter whose symbol is preceded by *, set the parameter value and switch power off once, then switch it on again to make that parameter setting valid.

- Description: explains the meaning of the parameter briefly
- Default Value: the default setting of the parameter
- Current Value: the current value of the parameter
- Unit: the unit of the parameter
- Setting Range: the range of the parameter.

Parameter Description Frame

This frame explains the parameter more completely, and illustrates the meaning of each setting value of the parameter.

Operate Frame

This frame includes several command buttons, and are described as following

- Read All: reads all of the servo driver parameter from servo driver, and displays the value in the "Current Value" column of Servo Driver Parameter Table
- Default: modify the setting value of all servo driver parameters to default value
- Save to File: save the current setting of all servo driver parameters into a file
- Load from File: modify the setting value of all servo driver parameters from a existing file
- Apply Next: apply the current parameter setting to next axis
- Apply All: apply the current parameter setting to all axes in your system

Value Frame

This frame shows the current setting of the parameter in decimal or hexadecimal format

- "Modify" button: modify the current setting of the parameter

5.9.2 Operation Steps

- Click “Read Parameter” button to read current value of all parameters from servo driver.
- Click the parameter you want to adjust in the parameter list table.
- Input the value, and click the “modify” button to modify the setting value of the parameter.
- Click “Write Parameter” button to adjust the parameters that you have modified.
- You can also click the “default” button to modify all the parameters to default setting, then click “Write Parameter” button to adjust all the parameters to default value.

Note: For any parameter whose symbol is preceded by *, set the parameter value and switch power off once, then switch it on again to make that parameters setting valid.

6 Appendix

6.1 MR-J2S-B Alarm List

When any alarm has occurred, eliminate its cause, ensure safety, then deactivate the alarm, and restart operation. Not doing so can cause injury.

AL.10	Undervoltage	Power supply voltage dropped. MR-J2S-B: 160V or lessMR-J2S-oB1: 83V or less
AL.12	Memory alarm 1	RAM memory fault
AL.13	Clock alarm	Printed board fault
AL.15	Memory alarm 2	EEPROM fault
AL.16	Encoder alarm 1	Communication error occurred between encoder and servo amplifier.
AL.17	Board alarm	CPU/parts fault
AL.19	Memory alarm 3	ROM memory fault
AL.1A	Motor combination alarm	Combination of servo amplifier and servo motor is wrong.
AL.20	Encoder alarm 2	Communication error occurred between encoder and servo amplifier.
AL.24	Main circuit error	Ground fault occurred at the servo motor outputs (U, V, W phases) of the servo amplifier.
AL.25	Absolute position erase	Absolute position data in error-Power was switched on for the first time in the absolute position detection system.
AL.30	Regenerative alarm	The permissible regenerative power of the built-in regenerative brake resistor or regenerative brake option is exceeded.Regenerative transistor fault
AL.31	Overspeed	Speed has exceeded the instantaneous permissible speed.

Table 6-1: MR-J2S-B Alarm List

AL.32	Overcurrent	Current that flew is higher than the permissible current of the servo amplifier.
AL.33	Overvoltage	Converter bus voltage input value-exceeded 400V.
AL.34	CRC error	Bus cable is faulty.
AL.35	Command pulse frequency alarm	The pulse frequency of the input command pulses is too high.
AL.36	Transfer error	Bus cable or printed board is faulty.
AL.37	Parameter alarm	Parameter setting is wrong.
AL.45	Main circuit device overheat	Main circuit overheated abnormally.
AL.46	Motor overheat	Servo motor temperature rise actuated the thermal protector.
AL.50	Overload 1	Load exceeded overload protection characteristic of servo amplifier.Load ratio 300%: 2.5s or more-Load ratio 200%: 100s or more
AL.51	Overload 2	Machine collision etc. caused max. output current to flow successively for several seconds.Servo motor locked: 1s or more
AL.52	Error excessive	Droop pulse value of the deviation counter exceeded the parameter No.31 setting value .
AL.8E	Serial communication alarm	Serial communication fault occurred between servo amplifier and communication device (e.g. personal computer).
88	Watchdog	CPU, parts faulty

Table 6-1: MR-J2S-B Alarm List

6.2 MR-J2S-B Warning List

If E6, E7, E9 or EE occurs, the servo off status is established. If any other warning occurs, operation can be continued but an alarm may take place or proper operation may not be performed. Eliminate the cause of the warning according to this section. Use the optional servo configuration software to refer to the cause or warning.

AL.92	Open battery cable warning	Absolute position detection system battery voltage is low.
AL.96	Home position setting warning	Home position return could not be made in the precise position.
AL.9F	Battery warning	Voltage of battery for absolute position detection system reduced.
AL.E0	Excessive regenerative load warning	There is a possibility that regenerative power may exceed permissible regenerative power of built-in regenerative brake resistor or regenerative brake option.
AL.E1	Overload warning	There is a possibility that overload alarm 1 or 2 may occur.
AL.E3	Absolute position counter warning	Absolute position encoder pulses faulty.
AL.E4	Parameter warning	Parameter outside setting rang.
AL.E6	Servo emergency stop	EM1-SG are open.
AL.E7	Controller emergency stop warning.	
AL.E9	Main circuit off warning	Servo was switched on with main circuit power off.
AL.EE	SCCNET error warning	The servo system controller connected is not SSCNET-compatible..

Table 6-2: MR-J2S-B Warning List

6.3 Driver Parameter List

Symbol	Name	MR-J2SB Instruction Manual parameter	Unit	Setting range
*AMS	Amp setting	Pr.01		0000H~0001H
*REG	Regenerative resistor	Pr.02		0000H~0011H
*MTY	For manufacturer’s settings	Pr.03		0080H
*MCA	For manufacturer’s settings	Pr.04		0000H
*MTR	For manufacturer’s settings	Pr.05		1
*FBP	Feedback pulse number	Pr.06		0,1,6,7,225
*POL	Direction of motor rotation	Pr.07		0,1
ATU	Auto-tuning	Pr.08		0000H~0004H
RSP	Servo response setting	Pr.09		0001H~000FH
TLP	Forward rotation torque limits	Pr.10	%	0~Maximum torque
TLN	Reverse rotation torque limits	Pr.11	%	0~Maximum torque
DG2	Moment of inertia ratio of load	Pr.12	0.1	0~3000
PG1	Position control gain 1	Pr.13	rad/sec	4~2000
VG1	Speed control gain 1	Pr.14	rad/sec	20~8000
PG2	Position control gain 2	Pr.15	rad/sec	1~1000
VG2	Speed control gain 2	Pr.16	rad/sec	20~20000
VIC	Speed integration compensation	Pr.17	msec	1~1000
NCH	Mechanical resonance control filter	Pr.18		0~031FH
FFC	Feed forward gain	Pr.19	%	0~100
INP	In position range	Pr.20	pulse	0~50000
MBR	Electromagnetic brake sequence output	Pr.21	msec	0~1000
MOD	Monitor output mode	Pr.22		0000H~0B0BH
OP1	Optional function 1	Pr.23		0000H~0001H
OP2	Optional function 2	Pr.24		0000H~0110H
LPF	Low pass filter	Pr.25		0000H~1210H
OP4	For manufacturer’s settings	Pr.26		0000H
MO1	Monitor output 1 offset	Pr.27	mv	-999~999

Table 6-3: Driver Parameter List

Symbol	Name	MR-J2SB Instruction Manual parameter	Unit	Setting range
MO2	Monitor output 2 offset	Pr.28	Mv	-999~999
MOA	For manufacturer’s settings	Pr.29		0001H
ZSP	Zero speed	Pr.30	rpm	0~10000
ERZ	Error excess alarm level	Pr.31	kpulse	1~1000
OP5	Option function 5	Pr.32		0000H~0002H
OP6	For manufacturer’s settings	Pr.33		0000H~0113H
VPI	PI-PID change position droop	Pr.34		0~50000
TTT	For manufacturer’s settings	Pr.35		0000H
VDC	Speed integration compensation	Pr.36		0~1000
OP7	For manufacturer’s settings	Pr.37		0010H
ENR	Encoder output pulse	Pr.38		0~32768
	For manufacturer’s settings	Pr.39		0000H
*BLK	Parameter block	Pr.40		0000H~000EH

Table 6-3: Driver Parameter List

6.4 Handshake Procedure

SSCNET board is composed of a DSP and other control units on it. The DSP is a microprocessor for managing all devices on the board. Once the CPU on host PC needs to communicate with DSP, it must use dual port RAM on SSCNET board to do it. On the same way, the DSP must communicate host CPU via dual port RAM. The commander must check if he can send the command and the responser must give him some ready signal for this procedure. This is so called handshake. It takes time in handshake. The handshake latency is 0.888ms because of the SSCNET protocol. Some procedure needs a series of handshaking and they will be introduced in the following sections.

6.4.1 Card Initial Procedure

The initial procedure is very complicated in SSCNET board. Once the function “MDSP_initial()” is lauched, the following flow char will be taken:

Step	Action	OK Reponse	Error Response	Error Reason
1	Power ON	LED Flash one by one and off	No LED Flashing or LED always ON	ROM data corrupt. Please download ROM data again. Use Kernelupdate.exe to do it.
2		Initial Board		
3		“DSP_OK”=1 LED turns off		

Table 6-4: Card Initial Procedure

Step	Action	OK Reponse	Error Response	Error Reason
4	MDSP_initial()	No Error	Card_ID_Out_Of_Range	The CardNo parameter of this function invalid
5		No Error	Card_Reinitialized	In the same program, card dosen't close normally then want to initial again
6		Check "DSP_OK"=1	Card_Not_Ready TimeOut=200ms	Use KernelUpdate.exe to reset DSP and try again
7		Check DSP Initial Status	Card_ReClose_Fail TimeOut=10000ms	Use KernelUpdate.exe to reset DSP and try again
8		Tell DSP start searching axes. The LED will flash	DSP_Initial_Time_Out TimeOut=10000ms	Use KernelUpdate.exe to reset DSP and try again
9		The servo drivers will display "b#"	Maximun_Number_Of_Card_Exceed	Close program and open again
10		Check DSP ready for FPGA download	FPGA_Handshake_Time_Out TimeOut=100ms	Use KernelUpdate.exe to reset DSP and try again
11		Load daughter board's FPGA code	FPGA_Download_Time_Out TimeOut=5000ms	Use KernelUpdate.exe to reset DSP and try again
12		If everything is okay, it will return CardID in lowbyte and total-axes-found in highbyte		

Table 6-4: Card Initial Procedure

6.4.2 Card Close Procedure

Every time the program ends, MDSP_inital() must be lauched to make sure that the PC resources will be released. It is good for next program starts.

Step	Action	OK Reponse	Error Response	Error Reason
1	MDSP_close()	LED will turn off	LED is still flashing or LED is always ON or OFF DSP_Close_Time_Out TimeOut=5000ms	Restart User's program or use KernelUpdate.exe to reset DSP and try again
2		The servo drivers will display "AA"		

Table 6-5: Card Close Procedure

6.4.3 Card Soft Reset Procedure

We strongly recommend you using kernelupdate.exe utility to reset board. The following table describe the procedure of MDSP_reset().

Step	Command	OK Reponse	Error Response	Error Reason
1	MDSP_reset()	LED Flash one by one and off	No LED Flashing or LED always ON	ROM data corrupt. Please download ROM data again. Use Kernelupdate.exe to do it.
2		Initial Board		
3		"DSP_OK"=1 LED turns off	DSP_Reset_Time_Out	
4		No Error	Card_ID_Out_Of_Range	The CardNo parameter of this function invalid
5		No Error	Card_Reinitialized	In the same program, card dosen't close normally then want to initial again
6		Check "DSP_OK"=1	Card_Not_Ready TimeOut=200ms	Use KernelUpdate.exe to reset DSP and try again
7		Check DSP Initial Status	Card_ReClose_Fail TimeOut=10000ms	Use KernelUpdate.exe to reset DSP and try again
8		Tell DSP start searching axes. The LED will flash	DSP_Initial_Time_Out TimeOut=10000ms	Use KernelUpdate.exe to reset DSP and try again
9		The servo drivers will display "b#"	Maximun_Number_Of_Card_Exceed	Close program and open again

Table 6-6: Card Soft Reset Procedure

6.4.4 Motion Command Procedure

After the motion command is issued by uses with parameters, the DLL will calcaulte the frames and transfer them to DSP. When all the frames are transferred, the DLL will set a "motion go" command and the motor will be started frame by frame. It takes some handshake time during this procedure. The following table shows the running steps of a motion command: start_ta_move()

Step	Action Item	Error Response	Error Reason
1	Check Card	Card_Not_Ready	“DSP_OK” is 0, please reset the card.
2		Card_Not_Initial	MDSP_Initial() failed, please restart program
3	Check DSP	DSP_Not_Ready	“Initial_Status” is not at finished state. Please restart program.
4	Check Axis	Axis_Not_In_Control	Axis is out of control. Check connection and restart program
5		Axis_Servo_Alarm	Axis is in servo alarm. Use alarm reset to remove this status
6		Axis_Is_Not_Ready_ON	Axis is not ready, use servo_on() command or check connection
7		Axis_Is_Not_Servo_ON	Axis is not servo on, use servo_on() command or check connection
8	Check Motion Status	Axis_Prepare_For_Motion	Axis is prepare frames for motion, if you are very sure the motion is ended, use stop command to cancel it.
9		Axis_Busy_For_Motion	Axis is busy for motion, if you are very sure the motion is ended, use stop command to cancel it.
10		Axis_In_EMG_ON	The EMG signal is ON. Check EMG logic and switch.
11		Axis_In_PEL_ON	The axis is going to a direction which PEL ON. Check PEL logic and switch
12		Axis_In_MEL_ON	The axis is going to a direction which MEL ON. Check MEL logic and switch
13	Frame download command	Axis_Hand_Shake_Failed	Frame download failed. Please use stop command to cancel this motion
15	Motion Go command	Axis_Not_Response	“Motion Command Go” is set to DSP but can’t see “In Motion” status ON. Please use stop command to cancel this motion
16	Finish	No Error	The function will response a positive value represents the total frames need to be run

Table 6-7: Motion Command Procedure

6.4.5 Motion Command Timing

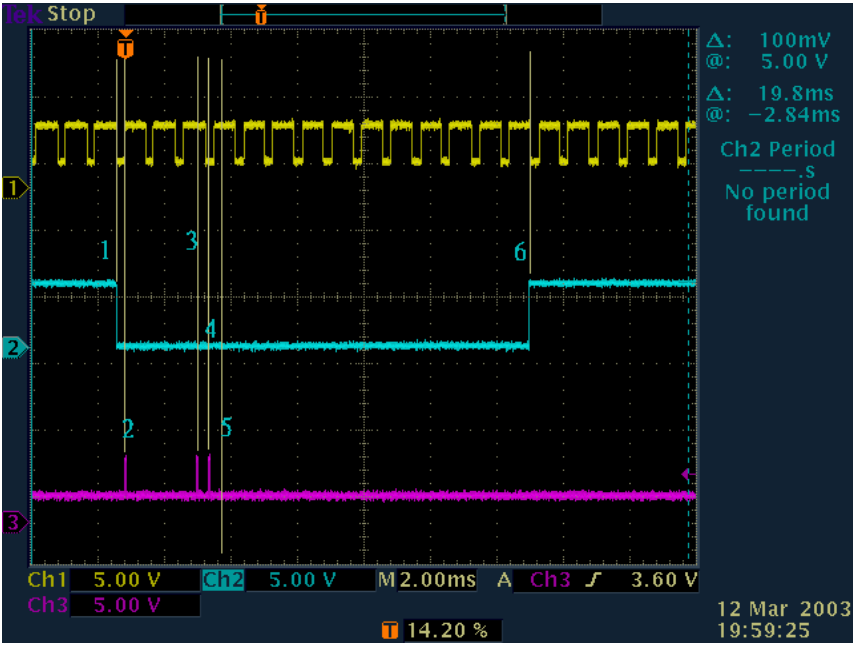


Figure 6-1: PCI-8372+ Single Motion Command Timing Chart

Signal Channel:

- [1] DSP processing time synchronized with SSCNET cycle (low voltage level duration)
- DSP codes has two process: Synchronized and Non-Synchronized process, this channel displays the processing time of synchronized process.
- [2] start_tr_move command processing time at host side (low voltage level duration)
- [3] DSP response time for Host motion command (high voltage level duration)

Label number:

- (1)start_tr_move() command starts
- (1~2) Trajectory calculation time on host
- (2)Send motion-download command to DSP. DSP will take some time (the Peak) to transfer the trajectory data and set a “transferring done flag” for host.
- *(2~3) Host waits the “transferring done flag” and get a mutex from system for continue
- (3)Send motion-go command to DSP and DSP will take some time(the Peak) to set a “motion go” flag for DSP synchronized process
- (3~4) DSP enter the synchronized process
- (4)DSP starts calculating first position for servo driver and put it on SSCNET data stream
- (4~5) The position data is sent to servo driver through cable
- (5)Servo motor runs according to the position data
- ** (5~6) Host start_ta_move() object is in ending process
- (6)start_ta_move function leaves

Note: * Waiting mutex time is uncertain. It takes 2~10ms in average.

** C++ object dis-constructing time.

Conclusion: From Command Launched to Motor started takes about 3.5 SSCNET cycle

6.5 cPCI-8312H High Speed Link Initial Guide

cPCI-8312H has two master chips of High Speed Link on the board. So it has all the features of HSL just like PCI-7852. In this chapter, we will introduce how to to initial the HSL functions on this board.

1. MDSP_Initial()

This function is not only for SSCNET but also initializing the board on Windows system. It will register the board's resources on system. The HSL functions are executable from this information only after the MDSP_Initial() is successfully issued.

2. HSL_Start() or HSL_Auto_Start()

This function will start to search all the modules on HSL network.

3. HSL_Slave_Live()

Use this function to check the status of searched HSL module

After these procedures, you can use all the functions of HSL. Please refer to HSL user's manual or module user guide for details.

Warranty Policy

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

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

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

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