Smarter. Greener. Together.

Industrial Automation Headquarters

Delta Electronics, Inc.
Taoyuan Technology Center
18 Xinglong Road, Taoyuan District,
Taoyuan City 33068, Taiwan
TEL: 886-3-362-6301 / FAX: 886-3-371-6301

Asia

Delta Electronics (Jiangsu) Ltd.
Wujiang Plant 3
1688 Jiangxing East Road,
Wujiang Economic Development Zone
Wujiang City, Jiang Su Province, P.R.C. 215200
TEL: 86-512-6340-3008 / FAX: 86-769-6340-7290

Delta Greentech (China) Co., Ltd.
238 Min-Xia Road, Pudong District,
ShangHai, P.R.C. 201209
TEL: 86-21-58635678 / FAX: 86-21-58630003

Delta Electronics (Japan), Inc.
Tokyo Office
2-1-14 Minato-ku Shibadaimon,
Tokyo 105-0012, Japan
TEL: 81-3-5733-1111 / FAX: 81-3-5733-1211

Delta Electronics (Korea), Inc.
1511, Byucksan Digital Valley 6-cha, Gasan-dong,
Geumcheon-gu, Seoul, Korea, 153-704
TEL: 82-2-515-5303 / FAX: 82-2-515-5302

Delta Electronics Int'l (S) Pte Ltd.
4 Kaki Bukit Ave 1, #05-05, Singapore 417939
TEL: 65-6747-5155 / FAX: 65-6744-9228

Delta Electronics (India) Pvt. Ltd.
Plot No 43 Sector 35, HSIIDC
Gurgaon, PIN 122001, Haryana, India
TEL : 91-124-4874900 / FAX : 91-124-4874945

Americas

Delta Products Corporation (USA)
Raleigh Office
P.O. Box 12173,5101 Davis Drive,
Research Triangle Park, NC 27709, U.S.A.
TEL: 1-919-767-3800 / FAX: 1-919-767-8080

Delta Greentech (Brasil) S.A.
Sao Paulo Office
Rua Itapeva, 26 - 3° andar Edificio Itapeva One-Bela Vista
01332-000-São Paulo-SP-Brazil
TEL: 55 11 3568-3855 / FAX: 55 11 3568-3865

Europe

Delta Electronics (Netherlands) B.V.
Eindhoven Office
De Witbogt 20, 5652 AG Eindhoven, The Netherlands
TEL: +31 (0)40-8003800 / FAX: +31 (0)40-8003898

2016/12/30

Delta Multi-servo Drive Integrated Robot Controller
ASDA-MS Series User Manual

Delta Multi-servo Drive Integrated Robot Controller ASDA-MS Series User Manual

www.deltaww.com

Smarter. Greener. Together.

Preface

Thank you for purchasing ASDA-MS. This manual provides the related information of ASDA-MS series multi-servo drive integrated robot controller (MS controller) and ECMA series servo motors.

This manual includes:

- The installation and inspection of MS controller and servo motor
- Wiring of MS controller
- Steps of trial operation
- Control functions and tuning of MS controller
- Introduction to Delta robot language (DRL)
- Description of all parameters
- Description of communication protocol
- Inspection and maintenance
- Troubleshooting

Product features:

The MS controller integrates industrial robot and servo drive as the all-in-one controller. Its control kernel features complex computation, smooth motion path configuration, instant servo control circuit and complete system information, which enhances the efficiency for system operation. It supports 5 kinds of programming languages which comply with IEC61131-3 and PLCopen motion function block, which provides a platform for users to create customized functions and develop robot application programs. Through the general communication interface, the MS controller connects to machine vision system (DMV), sensor and central control system. With the high-speed bus, the system can be expanded with other motion axes and be integrated into a comprehensive platform.

How to use this manual:

This manual can be used as reference for operating ASDA-MS. It contains the information related to the product installation, setting, as well as instructions of how to use this product. Please read through Chapter 1 to Chapter 5 before tuning or setting your MS controller.

Table of contents and subject indexing are also provided for information searching.

DELTA technical services:

Please consult the distributors or DELTA customer service center if any problem occurs.

Safety Precautions

ASDA-MS series is a high resolution and open type servo drive. It should be installed in a shielded control box during operation. This MS controller uses precise feedback control and the digital signal processor (DSP) with high-speed calculation function to control the current output which generated by IGBT so as to operate three-phase permanent magnet synchronous motors (PMSM) and to achieve precise positioning.

ASDA-MS is applicable to industrial applications and is suggested to be installed in the control box. (Servo drives, wire rod and motors all should be installed in the environment which complies with the minimum requirement of UL50 Type 1.)

Pay special attention to the following safety precautions anytime during inspection, installation, wiring, operation and examination.

The symbols of “DANGER”, “WARNING” and “STOP” represent:

DANGER	**It indicates the potential hazards. It is possible to cause severe injury or fatal harm if the instructions are not followed.**
WARNING	**It indicates the potential hazards. It is possible to cause minor injury or lead to serious product damage or malfunction if the instructions are not followed.**
STOP	**It indicates the absolute prohibited activity. It is possible to cause damage to the product or product malfunction if the instructions are not followed.**

Inspection

DANGER	Please follow the instructions when using MS controller and servo motor, or it might cause fire or product malfunction.

Installation

DANGER	Do not expose the product to the environment which contains water, corrosive gas, or inflammable gas. Or it might result in electric shock or fire.

Wiring

DANGER

- Connect the ground terminals to class-3 ground system (Ground resistance should not exceed 100 Ω). Improper grounding may result in communication error, electric shock or fire.
- Do not connect the three-phase source to the motor output terminal U, V and W. Or it might cause fire or injury to persons.
- Please tighten the screws of the power and motor output terminal. Or it might cause fire.
- Please do the wiring according to the wire rod selection to prevent any danger.

Operation

- Before the operation, please change the parameter setting according to the application. If it is not adjusted to the correct setting value, it is possible to lead to malfunction of the machine or the operation might be out of control.
- Before the machine starts to operate, please be ensured that the emergency stop can be activated anytime.
- When applying to the power, make sure the motor will not rotate because of inertia of mechanism or other causes.

During the operation, it is prohibited to touch any rotating motor parts. Or injury to persons may occur.

- In order to prevent any accident, please separate the couplings and belts of the machine and isolate them. Then, conduct the initial trial run.
- If users fail to operate the machine properly after the servo motor connects to the equipment, it might damage the equipment and lead to injury to persons.
- In order to prevent the danger, it is strongly recommended to check if the MS controller can operate normally without load first. Then, operate the motor with load.
- Do not touch the heat sink of MS controller during operation. Or it is possible to cause scald due to the high temperature.

Maintenance and Inspection

- It is prohibited to touch the internal parts of the servo drive and servo motor. Or it might cause electric shock.
- It is prohibited to disassemble the panel of the servo drive when turning on the power. Or it might cause electric shock.
- Do not touch the ground terminal within 10 minutes after turning off the power. Or the residual voltage may cause electric shock.
- Do not disassemble the motor. Or it might cause electric shock or injury to persons.
- Do not change the wiring when the power is on. Or it might cause electric shock or injury to persons.
- Only the qualified electrical and electronics professionals can install, wire or maintain the MS controller and the servo motor.

Main Circuit Wiring

- Do not put the power cable and the encoder cable in the same channel and bond them together. Please separate the power cable and the encoder cable for at least 30 centimeters (= 11.8 inches).
- Please use stranded wires and multi-core shielded-pair wires for encoder cables and encoder feedback cables. The maximum length of command input cable is 3 meters (= 9.84 feet) and the maximum length of feedback cable is 20 meters (= 65.62 feet).
- The high voltage might remain in the servo motor even when the power is off. Remove the power source and wait for 10 minutes for discharging the capacitor before maintenance.

Avoid frequent on/off operation. If continuous power on and off is needed, please be ensured that the interval is one minute at least.

Terminal Wiring of the Main Circuit

- When wiring, please disassemble the terminal socket from MS controller.
- One terminal of the terminal socket for one electric wire only.
- When inserting the electric wires, do not short circuit the adjacent conductors.
- Before connecting to the power source, please inspect and be ensured that the wiring is correct.

The content of this instruction sheet may be revised without prior notice. Please consult the distributors or download the latest version at http://www.delta.com.tw/industrialautomation/.

Table of Contents

Before Operation

1 Inspection and Model Explanation

2 Installation

3 Wiring

Operation and Setting

6 Delta Robot Language

7 Coordinates System

Parameters Setting

8 Parameters

9 Communication

10 Absolute System

Troubleshooting

11 Troubleshooting

12 NC Code

Appendix

A Specifications

B Accessories

C

(This page is intentionally left blank.)

Inspection and Model Explanation

1

Before using ASDA-MS, please pay attention to the description about the inspection, nameplate, and model type. Suitable motor model for your MS controller can be found in the table of section 1.3.

1

1.1 Requirements for system installation

A complete and workable servo set should include:

(1) One MS controller and one servo motor.

(2) Two UVW motor power cables, the U, V and W wires can connect to the socket attached by MS controller and another side is the plug which could connect to the socket of the motor. And a green ground wire which should be connected to the ground terminal of MS controller. (selective purchase)

(3) An encoder cable which connects to the socket of the encoder. One side of it connects to MOTOR ENC. of MS controller and another side is the plug. (selective purchase)

(4) 4-PIN connector for USB1 (selective purchase)

(5) RJ45 connector for DMCNET high-speed communication (selective purchase)

(6) Power input of MS controller:
750 W: 2-PIN quick connector (24 V, 0 V)
750 W: 3-PIN quick connector (R, S, T)

(7) 2 sets of 6-PIN quick connector (UVW)

(8) STO connector

(9) BRK connector

(10) A plastic lever (for whole series)

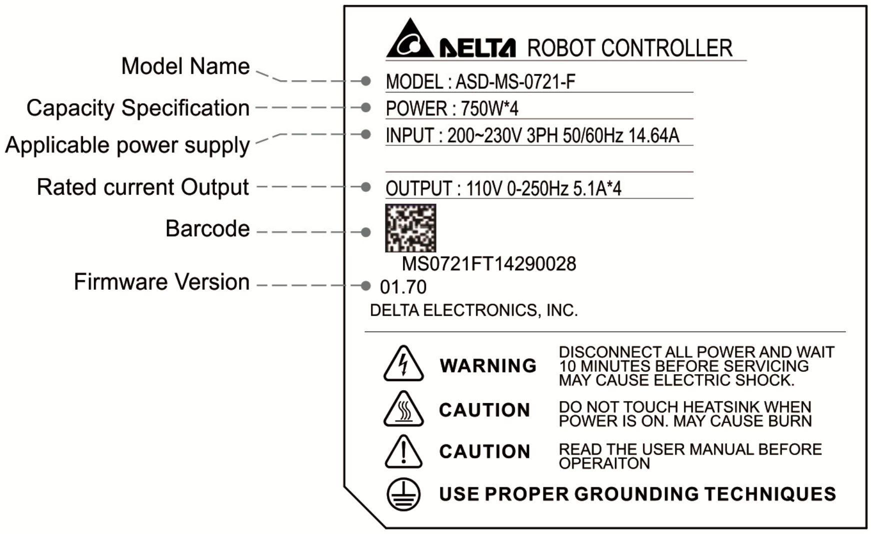
(11) An installation manual

1.2 Product model

1.2.1 Nameplate information

ASDA-MS series controller

■ Nameplate information



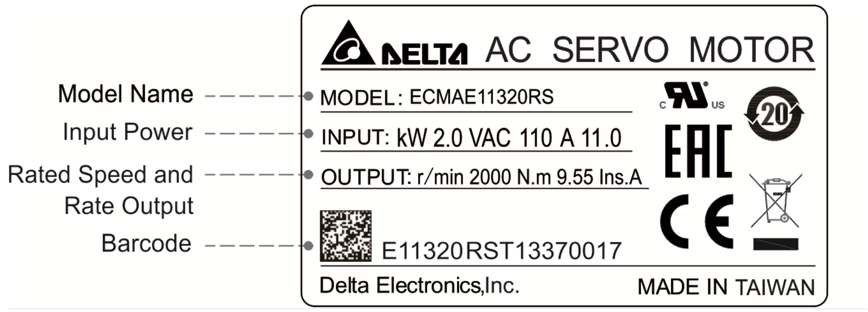
■ Serial number

MS0721F	T	14	29	0028
(1)	(2)	(3)	(4)	(5)

(1) Model name
(2) Production factory (T: Taoyuan; W: Wujiang)
(3) Year of production (14: year 2014)
(4) Week of production (form 1 to 52)
(5) Serial number (Production sequence of a week, starting from 0001)

ECMA series servo motor

■ Nameplate information



■ Serial number

E11320RS	T	14	37	0017
(1)	(2)	(3)	(4)	(5)

(1) Model name
(2) Production factory (T: Taoyuan; W: Wujiang)
(3) Year of production (14: year 2014)
(4) Week of production (from 1 to 52)
(5) Serial number (production sequence of a week, starting form 0001)

1

1.2.2 Model explanation

ASDA-MS series controller

ASD – MS – 07 21 – F

(1) (2) (3) (4) (5)

(1) Product name

AC Servo Drive

(2) Series

MS

(3) Rated output power

07 represents 750 W model

15 represents 1.5 kW model

(4) Input voltage and phase

21 represents 220 V, single-/three-phase MS controller

23 represents 220 V, three-phase MS controller

(5) Model type

Type	Full-closed loop control	EtherCAT	CANopen	DMCNET	E-CAM	Extension port for digital input
F	×	×	×	○	×	×

ECMA series servo motor

ECM A – C 1 06 02 E S

(1) (2) (3) (4) (5) (6) (7) (8)

(1) Product name ECM: Electronic Communication Motor

(2) Servo type A: AC servo

(3) Name of the series

Code	Rated voltage and speed	Code	Rated voltage and speed
C	220 V and 3,000 rpm	G	220 V and 1,000 rpm
E	220 V and 2,000 rpm	-	-

(4) Encoder type

Code	Spec.
1	Incremental type, 20-bit (for servo drive that is below 3 kW)
2	Incremental type, 17-bit
A	Absolute type (resolution of single turn: 17-bit; multi-turn: 16-bit)

(5) Motor frame size

Code	Spec.	Code	Spec.
04	40 mm	10	100 mm
06	60 mm	13	130 mm
08	80 mm	18	180 mm
09	86 mm	-	-

(6) Rated power output

Code	Spec.	Code	Spec.	Code	Spec.
01	100 W	05	500 W	10	1.0 kW
02	200 W	06	600 W		
03	300 W	07	700 W		
04	400 W	09	900 W		

(7) Type of shaft diameter and oil seal

	w/o brake w/o oil seal	with brake w/o oil seal	w/o brake with oil seal	with brake with oil seal
Round shaft (with fixed screw holes)	-	-	C	D
Keyway	E	F	-	-
Keyway (with fixed screw holes)	P	Q	R	S

(8) Shaft diameter

Standard shaft diameter: S

Specific shaft diameter: 3 = 42 mm; 7 = 14 mm

1.3 MS controller and corresponding servo motor

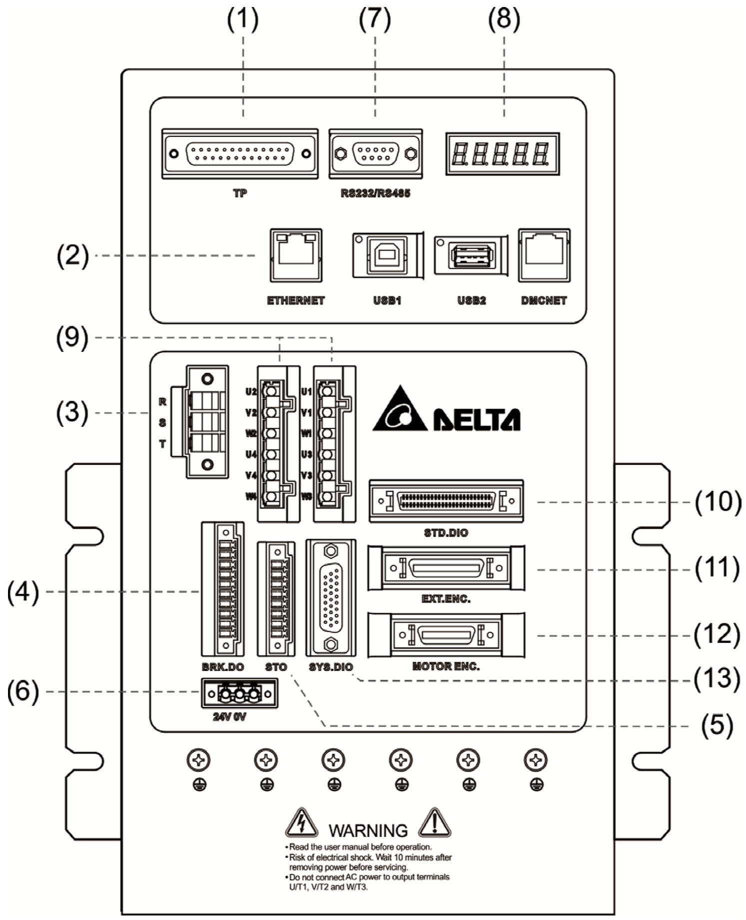
Motor							MS controller		
Motor series		Power	Output (W)	Model number	Rated current (Arms)	Max. instantaneous current (A)	Model number	Continuous output current (Arms)	Max. instantaneous output current (A)
Low inertia	ECMA-C 3000 r/min	Single-/ Three-phase	50	ECMA-C1040F□S	0.69	2.05	ASD-MS-0721-F	5.10	15.30
			100	ECMA-C△0401□S	0.90	2.70			
			200	ECMA-C△0602□S	1.55	4.65			
			400	ECMA-C△0604□S	2.60	7.80			
			400	ECMA-C△0804□7	2.60	7.80			
			750	ECMA-C△0807□S	5.10	15.30	ASD-MS-0721-F ASD-MS-1523-F	5.10 8.30	15.30 24.90
			750	ECMA-C△0907□S	5.10	15.30			
			1000	ECMA-C△0910□S	3.66	11.00	ASD-MS-1523-F	8.30	24.90
			1000	ECMA-C△1010□S	7.30	21.9			
Medium inertia	ECMA-E 2000 r/min	Single-/ Three-phase	400	ECMA-C△0604□H	2.60	7.80	ASD-MS-0721-F	5.10	15.30
			750	ECMA-C△0807□H	5.10	15.30	ASD-MS-0721-F ASD-MS-1523-F	5.10 8.30	15.30 24.90
High inertia	ECMA-C/G 3000 r/min	Single-/ Three-phase	300	ECMA-G△1303□S	2.50	7.50	ASD-MS-0721-F	5.10	15.30
			500	ECMA-E△1305□S	2.90	8.70			
			600	ECMA-G△1306□S	4.80	14.4			
			850	ECMA-F△1308□S	7.10	19.4	ASD-MS-1523-F	8.30	24.90
			900	ECMA-G△1309□S	7.50	22.5			
			1000	ECMA-E△1310□S	5.60	16.8			
			1500	ECMA-E△1315□S	8.30	24.9			

Note:

1. (△) in motor model names represents encoder type. △ = 1: incremental type, 20-bit; △ = 2: incremental type, 17-bit; △ = 3: 2500 ppr; △ = M: magnet type; △ = w: NICON encoder, 20-bit. The motor model listed above is for information checking. Please contact local distributors for product ordering.
2. (☐) in motor model names represents brake or keyway / oil seal.
3. The above table shows the specification of servo drive which has triple rated current. If the six times of rated current is required, please contact local distributors. For detailed specification of the servo motor and MS controller, please refer to Appendix A.
4. 1.5 kW model is coming soon.

1.4 Each part of MS controller

ASDA-MS series controller



(1) TP: Connects to HMI

(2) Communication: EtherNet, USB1, USB2, DMCNET

(3) RST: 200 V_{AC} input

(4) BRK.DIO: $24V_{DC}$ output, for releasing motor brake

(5) STO: Dual loop control DI

(6) 24V-0: Control power input, 24 V_{DC}

(7) Communication: RS-232, RS-485

(8) Display: 5-digit, 7-segment LED displays the state of MS controller

(9) UVW: 4-axis motor output; connects to motor power

(10) STD.DIO: User I/O, 24 digital inputs and 12 digital outputs

(11) EXT.ENC.: Feedback signal (A, B, Z phase)

(12) MOTOR.ENC.: 4 encoders; connects to the servo motor

(13) SYS.DIO: System I/O, 8 system digital inputs and 8 system digital outputs

(This page is intentionally left blank.)

Installation 2

Please follow the instruction mentioned in this chapter during installation. Information about specification of circuit breaker, fuse, EMI filter selection, and selection of regenerative resistor are also included.

2.1 Safety precautions

Please pay special attention to the followings:

If the connection between MS controller and the servo motor is over 20 meters, please thicken the connecting wire, UVW as well as the encoder cable. Please refer to section 3.1.6 Selection of wire rods. Do not select the wire that does not comply with the listed specification.

2.2 Ambient condition of storage

Before the installation, this product has to be kept in the shipping carton. In order to retain the warranty coverage and for the maintenance, please follow the instructions below when storage, if the product is not in use temporally:

- Store the product within an ambient temperature range of -20°C to +65°C.
- Store the product within a relative humidity range of 0% to 90% and a non-condensing environment.
- Avoid storing the product in the environment of corrosive gas.

2.3 Ambient condition of installation

MS controller: The ambient location should be free of over-heat device, water drop, vapor, dust and oily dust, corrosive and inflammable gas and liquid, airborne dust and metal particles. Check if the vibration will influence the electronic device of the electric box.

Motor: The ambient temperature of the motor is between 0°C and 40°C and the ambient conditions be free of over-heat device, water drop, vapor, dust and oily dust, corrosive and inflammable gas and liquid, airborne dust and metal particles.

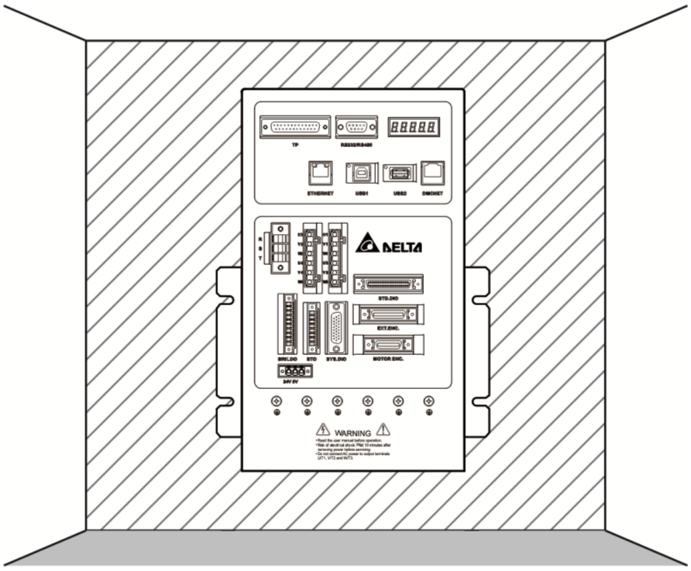
The ambient temperature of MS controller is between 0°C and 55°C. If the temperature is over 45°C, please place the product in a well-ventilated environment. During long-term operation, the ambient temperature should be under 45°C for ensuring its performance. If the product is installed in an electric box, make sure the size of the electric box and its ventilation condition will not overheat and endanger the internal electronic device.

2.4 Installation direction and space

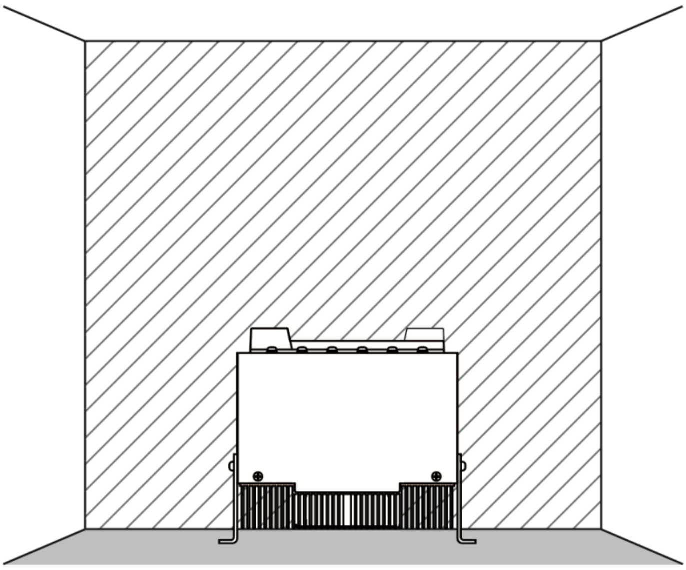
Attention:

- Incorrect installation may result in a drive malfunction.
- In order to ensure your MS controller is well ventilated, ensure that sufficient space is given to ASDA-MS. Or malfunction and damage will occur.
- Do not obstruct the ventilation holes and please correctly place MS controller. Or it might cause danger.

2



Correct



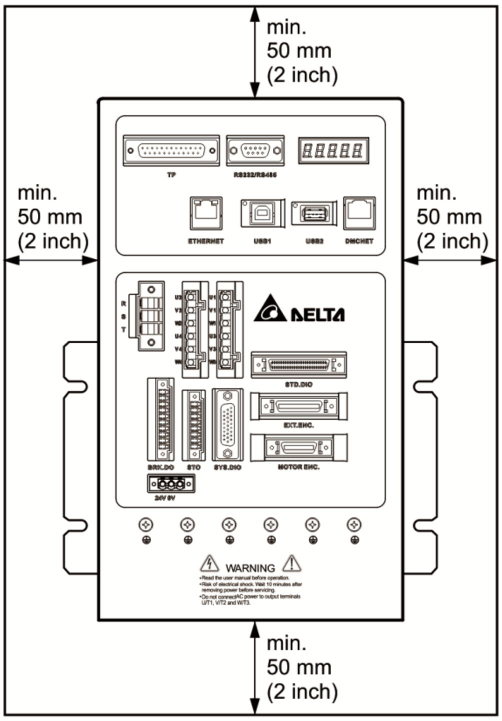
Incorrect

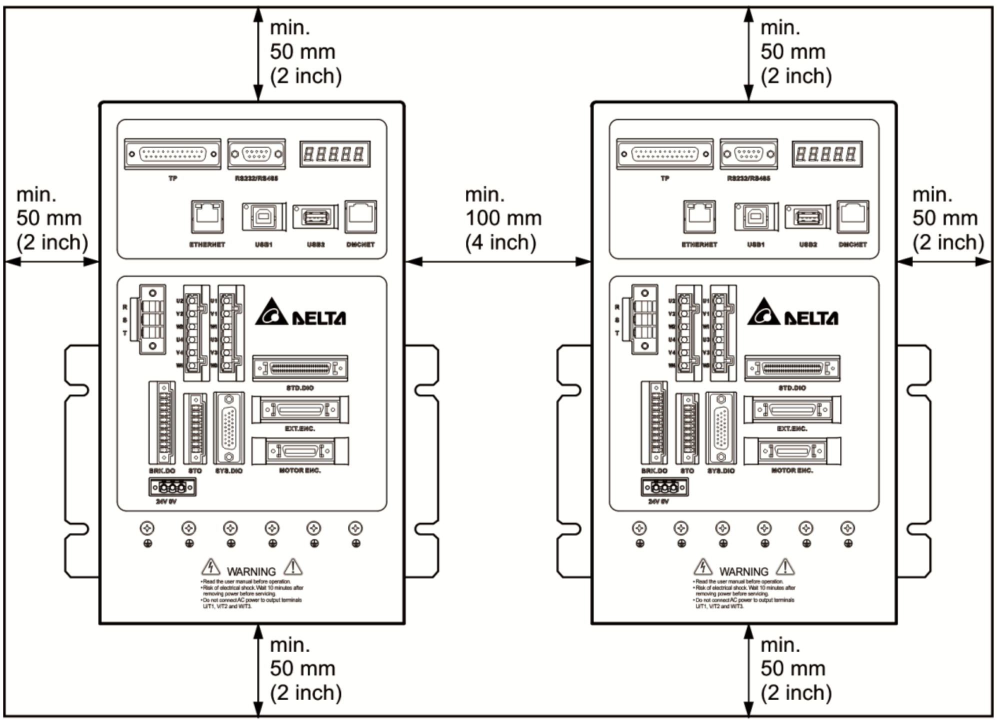
2

Heat dissipation requirements:

In order to have smaller wind resistance of the fan and increase the ventilation, please follow the suggested clearance value when installing one or more than one MS controllers. (Refer to the following diagrams)

Note: The above diagrams are not in equal proportion. Please refer to the annotation.





2.5 Specification of circuit breaker and fuse

MS controller model	Circuit breaker	Fuse (Class T)
ASD-MS-0721-F	30A	50A

Note:

1. If the servo drive equips with earth leakage circuit breaker for avoiding electric leakage, its current sensitivity should be over 200 mA and persists up to 0.1 seconds.
2. 1.5 kW model is coming soon.
3. Please apply the fuse and circuit breaker that complies with UL / CSA standard.

2.6 EMI Filters

All electronic equipment (including MS controller) generates high or low frequency noise during operation, which interfere the peripheral equipment via conduction or radiation. With EMI Filter and the correct installation, much interference can be eliminated. It is suggested to use Delta's EMI Filter to suppress the interference.

Item	Power	Servo Drive Model	Recommended EMI Filter	FootPrint
1	750 W	ASD-MS-0721-F	EMF023A21A	N
			EMF027A23A	

General precautions

In order to ensure the best performance of EMI Filter, apart from the instructions of MS controller installation and wiring, please follow the precautions mentioned below:

1. The MS controller and EMI Filter should be installed on the same metal plate.
2. The wiring should be as short as possible.
3. The metal plate should be well grounded.

Please follow the instructions of the user manual and make sure it meets the following specifications:

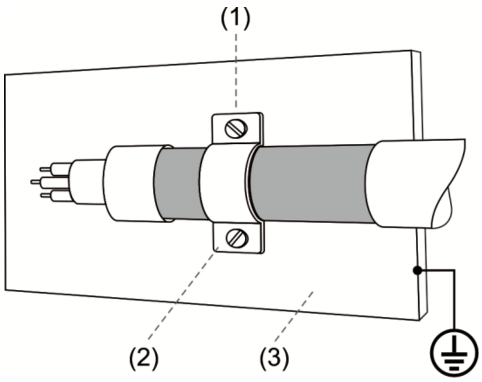
1. EN61000-6-4 (2001)
2. EN61800-3 (2004) PDS of category C2
3. EN55011+A2 (2007) Class A Group 1

2

Motor cable selection and installation precaution

The selection of motor cables and installation determines the performance of EMI Filter. Please follow the precautions mentioned below.

1. Use the cable that has braided shielding (double shielding would be better)
2. The shield on both sides of the motor cable should be grounded with the shortest cable length and the largest contact area.
3. Remove the protective paint of the U-shape saddle and metal plate in order to ensure the good contact. Please see the figure below.
4. It requires correct connection between the braided shielding of the motor cable and the metal plate. The braided shielding on both sides of the motor cable should be fixed by the U-shape saddle and metal plate. Please see the figure below for the correct connection.



(1) The protective paint of the U-shape saddle and metal plate should be removed in order to ensure the good contact.

(2) U-shape saddle

(3) Well-grounded metal plate

2.7 Regenerative resistor

Built-in regenerative resistor of 750 W model

The following table shows the specification of built-in regenerative resistor provided by ASDA-MS series:

MS controller (kW)	Specification of built-in regenerative resistor	
	Resistance (P1-52)(Ohm)	Capacity (P1-53)(Watt)
0.75	100	40

Wiring

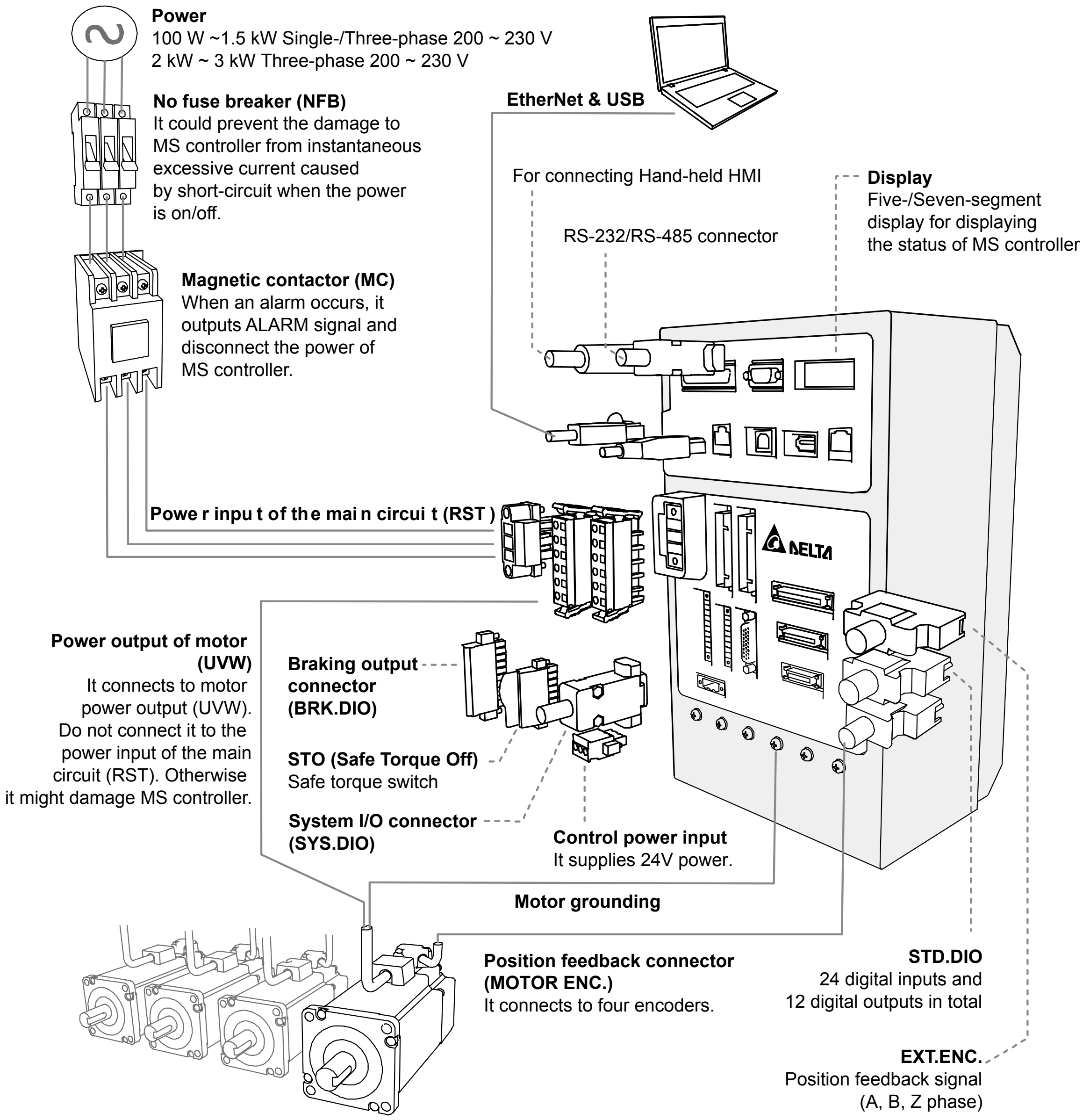
3

This chapter illustrates the electric circuit of power supply, connectors, and wiring for each mode of ASDA-MS.

3

3.1 System connection

3.1.1 Connecting to peripheral devices



Safety precautions:

1. Make sure power and wiring among R, S, T, 24 V and 0 V are correct.
2. Make sure wiring of motor power output U, V, W are correct.
3. Connect encoder wire to MOTOR ENC. correctly.
4. STO function can be used to disconnect motor’s power. If STO function is not used, the circuit in STO port has to be shorted out. (terminal block for shorting is available from Delta) Or the system cannot be activated.

3.1.2 Connectors and terminal blocks

<table>
<tr><th>Symbol</th><th>Name</th><th colspan="8">Description</th></tr>
<tr><td>24V, 0V</td><td>Power input of the control circuit</td><td colspan="8">Connect to 24 VDC power
(Power source needs to qualify Class 2 standard)</td></tr>
<tr><td>R, S, T</td><td>Power input of the main circuit</td><td colspan="8">Connect to three-phase AC power (Please refer to the specification of the model and input proper voltage.)</td></tr>
<tr><td rowspan="7">U, V, W
FG</td><td rowspan="7">Motor wire</td><td colspan="8">Connect to the servo motor</td></tr>
<tr><td>Symbol</td><td>Wire Color</td><td colspan="6">Description</td></tr>
<tr><td>U</td><td>Red</td><td colspan="6" rowspan="3">Three-phase main power cable of the motor</td></tr>
<tr><td>V</td><td>White</td></tr>
<tr><td>W</td><td>Black</td></tr>
<tr><td>FG</td><td>Green</td><td colspan="6">Connect to ground terminal ⏚ of MS controller</td></tr>
<tr></tr>
<tr><td>⏚ x 6</td><td>Ground terminal</td><td colspan="8">Connect to the ground wire of the power and servo motor.</td></tr>
<tr><td>TP</td><td>HMI connection port (optional)</td><td colspan="8">Connect to Delta HMI</td></tr>
<tr><td>RS232/RS485</td><td>Serial communication port</td><td colspan="8">Connect to other devices via RS-232 or RS-485 interfaces</td></tr>
<tr><td>Ethernet</td><td>Ethernet communication</td><td colspan="8">Connect to PC via Ethernet port</td></tr>
<tr><td>USB1</td><td>USB port
(Type B) (optional)</td><td colspan="8">Connect to PC via USB port</td></tr>
<tr><td>USB2</td><td>USB port</td><td colspan="8">Connect to USB storage</td></tr>
<tr><td>DMCNET</td><td>DMCNET communication port</td><td colspan="8">Connect to other devices via DMCNET</td></tr>
<tr><td>STO</td><td>Safe Torque Off I/O</td><td colspan="8">STO port</td></tr>
<tr><td>STD.DIO</td><td>User DI, DO</td><td colspan="8">24 sets of DI, 12 sets of DO</td></tr>
<tr><td>SYS.DIO</td><td>System DI, DO</td><td colspan="8">8 sets of SDI, 8 sets of SDO</td></tr>
<tr><td>BRK.DIO</td><td>Brake DO</td><td colspan="8">24V output for switching motor’s brake function</td></tr>
<tr><td>EXT.ENC.</td><td>Position feedback signal port</td><td colspan="8">Connect to external linear scale or encoder for full-closed loop control</td></tr>
<tr><td rowspan="8">Motor ENC.</td><td rowspan="8">Motor encoder connection port (optional)</td><td colspan="8">Connect to encoder in Delta motor.
Please refer to section 3.3</td></tr>
<tr><td>Symbol</td><td>Color</td><td colspan="6">Pin No</td></tr>
<tr><td>Axis</td><td>-</td><td>A</td><td>B</td><td>C</td><td>D</td><td>E</td><td>F</td></tr>
<tr><td>T+</td><td>Blue</td><td>5</td><td>11</td><td>17</td><td>23</td><td>14</td><td>20</td></tr>
<tr><td>T-</td><td>Blue/Black</td><td>7</td><td>13</td><td>19</td><td>25</td><td>15</td><td>21</td></tr>
<tr><td>+5V</td><td>Red/Red and black</td><td>4</td><td>10</td><td>16</td><td>22</td><td>2</td><td>8</td></tr>
<tr><td>GND</td><td>Black/Black and white</td><td>6</td><td>12</td><td>18</td><td>24</td><td>3</td><td>9</td></tr>
<tr><td>Reserved</td><td>-</td><td colspan="6">1, 26</td></tr>
</table>

3

Precautions for wiring:

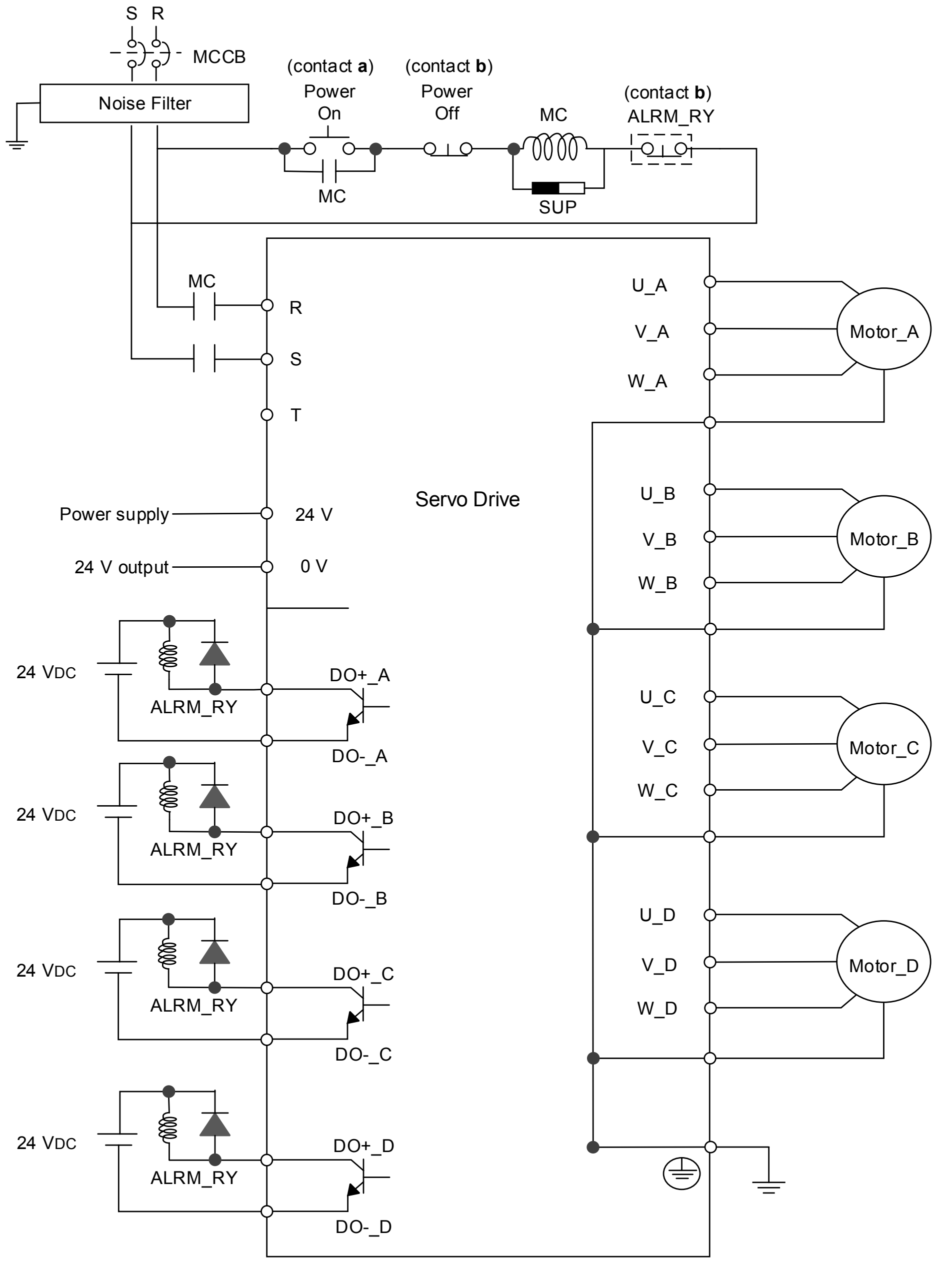
1. As soon as the power is off, do not touch R, S, T and U, V, W since the capacitance inside the servo drive still contains huge amount of electric charge. Wait until the charging light is off.
2. Separate R, S, T and U, V, W from the other wires. The distance should be at least 30 cm (11.8 inches).
3. If extension of connection cable for CN2 (encoder) or CN5 (position feedback) is required, please use an AWG26 shielded twisted-pair cable which conforms to UL2464 specifications. If it is over 20 meters (65.62 ft.), please choose to double the diameter of the signal cable to avoid excessive signal attenuation.
4. When DMCNET communication is applied, please conduct the standard wiring with shielded twisted-pair to ensure the communication quality.
5. More details about selection of wire rod, please refer to section 3.1.6

3.1.3 Wiring for power supply

There are two types of wiring method for power supply, single-phase and three-phase. In the diagram below, Power On is contact **a**, Power Off and ALRM_RY are contact **b**. MC is power relay (magnetic contactor) and self-holding power and the contact of main power circuit.

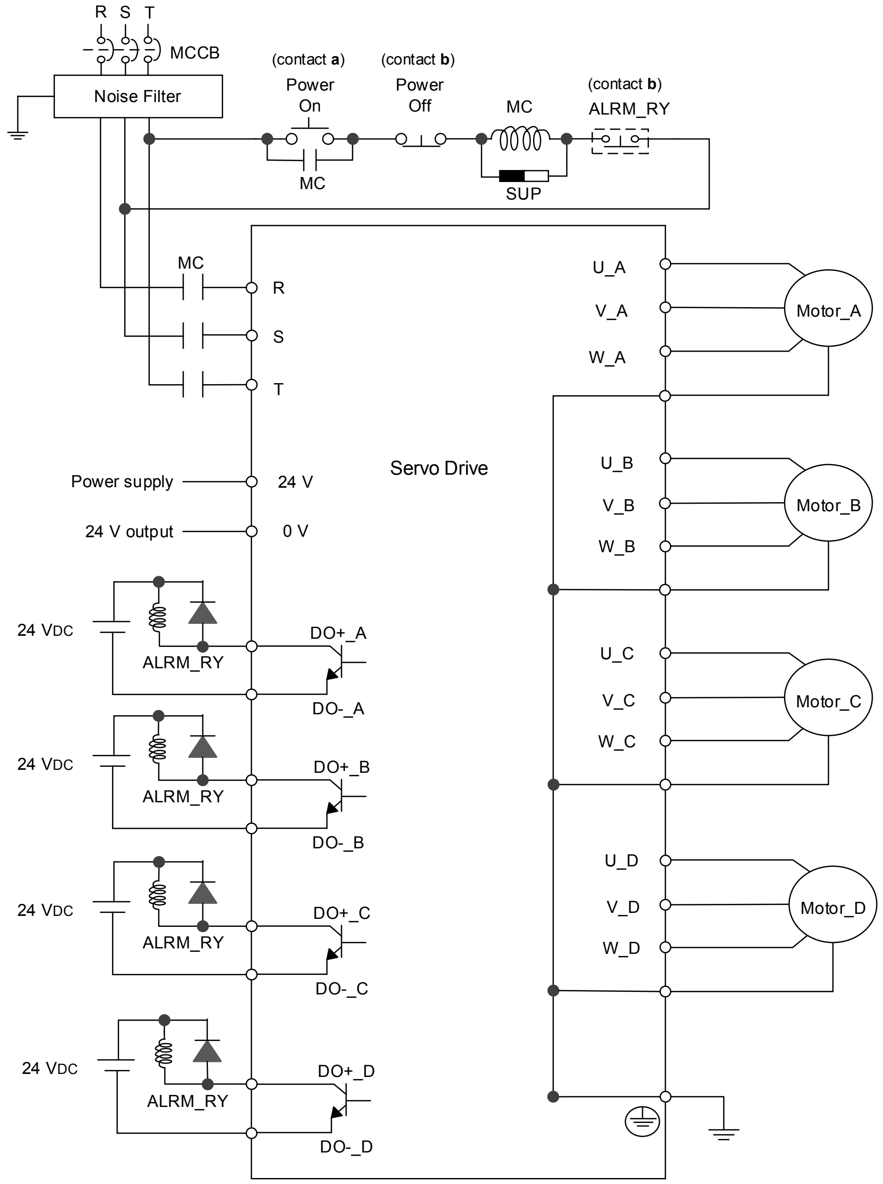
3

- Wiring method of single-phase power supply (suitable for 750 W model)

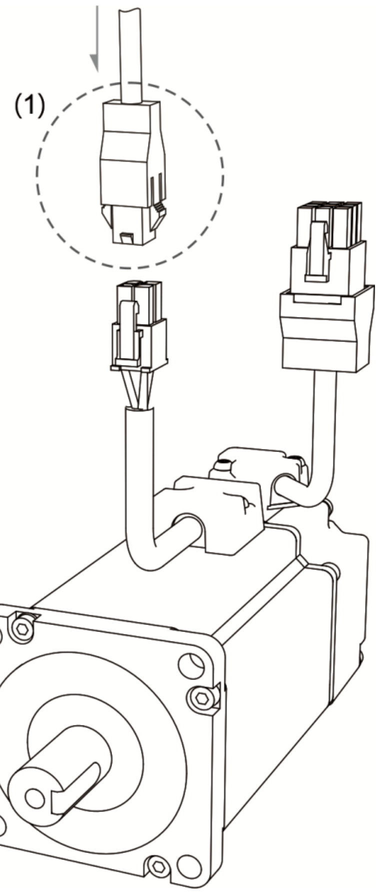


3

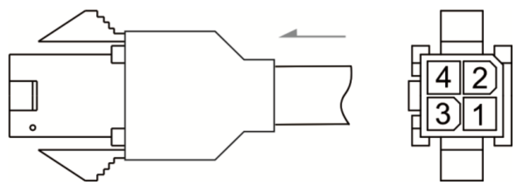
■ Wiring method of three-phase power supply (suitable for all series)



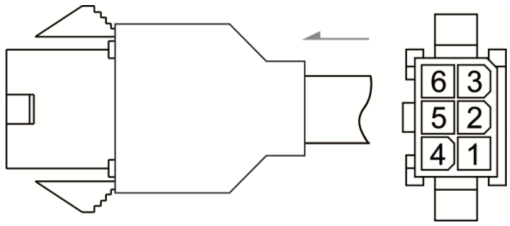
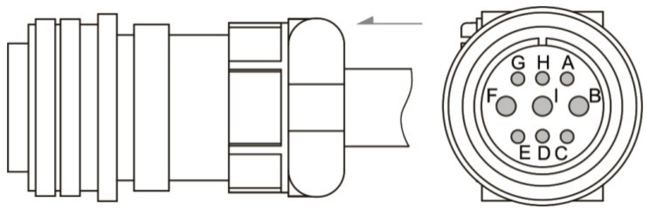
3.1.4 U, W, V connector of MS controller



(1) Please refer to the following table for UVW connector specifications

<table>
<tr><th>Motor model</th><th colspan="6">U, V, W connector</th></tr>
<tr><td rowspan="4">ECMA-C1040F☐S (50 W)
ECMA-C△0401☐S (100 W)
ECMA-C△0602☐S (200 W)
ECMA-C△0604☐S (400 W)
ECMA-C△0604☐H (400 W)
ECMA-C△0804☐7 (400 W)
ECMA-C△0807☐S (750 W)
ECMA-C△0807☐H (750 W)
ECMA-C△0907☐S (750 W)</td><td colspan="6">
</td></tr>
<tr><td colspan="6">Pin assignment</td></tr>
<tr><td>U (Red)</td><td>V (White)</td><td>W (Black)</td><td>CASE GROUND (Green)</td><td>BRAKE1 (Yellow)</td><td>BRAKE2 (Blue)</td></tr>
<tr><td>1</td><td>2</td><td>3</td><td>4</td><td>-</td><td>-</td></tr>
</table>

3

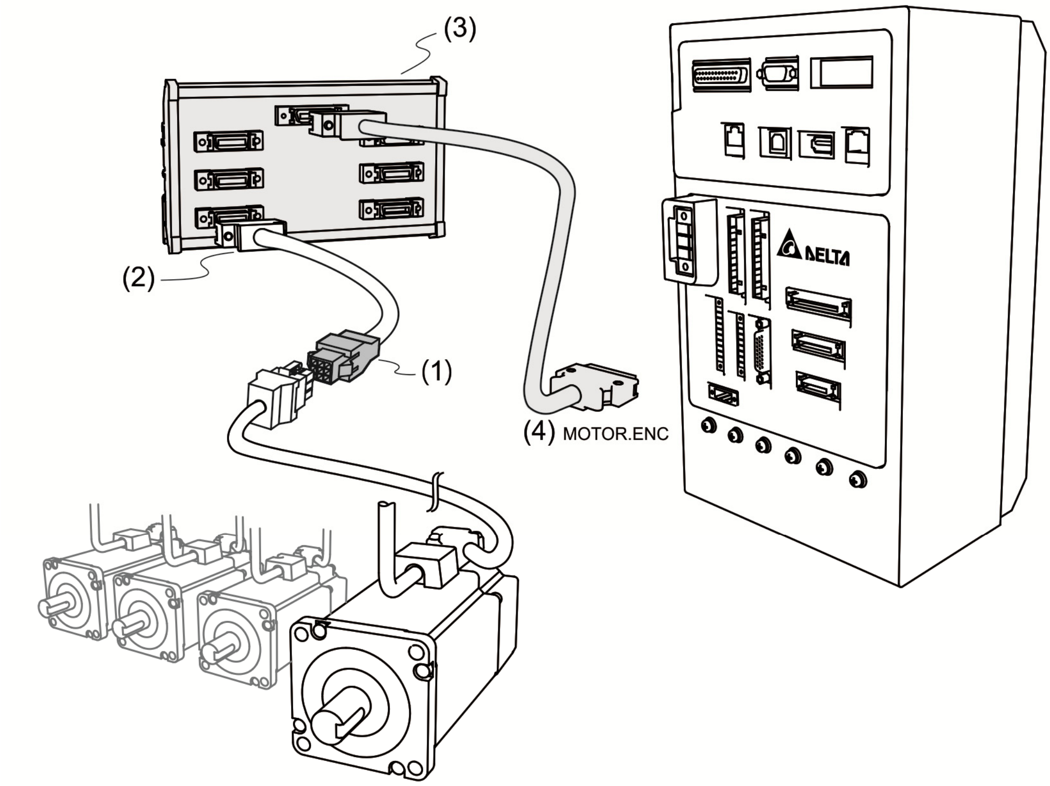
<table>
<tr><th>Motor model</th><th colspan="6">U, V ,W connector with brake</th></tr>
<tr><td rowspan="4">ECMA-C1040F□S (50 W)
ECMA-C△0401□S (100 W)
ECMA-C△0602□S (200 W)
ECMA-C△0604□S (400 W)
ECMA-C△0604□H (400 W)
ECMA-C△0804□7 (400 W)
ECMA-C△0807□S (750 W)
ECMA-C△0807□H (750 W)
ECMA-C△0907□S (750 W)</td><td colspan="6">
</td></tr>
<tr><td colspan="6">Pin assignment</td></tr>
<tr><td>U (Red)</td><td>V (White)</td><td>W (Black)</td><td>CASE GROUND (Green)</td><td>BRAKE1 (Yellow)</td><td>BRAKE2 (Blue)</td></tr>
<tr><td>1</td><td>2</td><td>4</td><td>5</td><td>3</td><td>6</td></tr>
<tr><td rowspan="4">ECMA-G△1303□S (300 W)
ECMA-E△1305□S (500 W)
ECMA-F△1305□S (500 W)
ECMA-G△1306□S (600 W)
ECMA-F△1308□S (850 W)
ECMA-G△1309□S (900 W)
ECMA-C△1010□S (1000 W)
ECMA-E△1310□S (1000 W)
ECMA-F△1313□S (1300 W)
ECMA-E△1315□S (1500 W)</td><td colspan="6">
</td></tr>
<tr><td colspan="6">Pin assignment</td></tr>
<tr><td>U (Red)</td><td>V (White)</td><td>W (Black)</td><td>CASE GROUND (Green)</td><td>BRAKE1 (Yellow)</td><td>BRAKE2 (Blue)</td></tr>
<tr><td>F</td><td>I</td><td>B</td><td>E</td><td>G</td><td>H</td></tr>
</table>

Wire rod selection: Please use a 600 V PVC cable. If it is longer than 30 meters, select the cable size according to voltage drop (wire impedance). See section 3.1.6 for more information.

Note: (□) in the motor model names represents brake or keyway / oil seal.

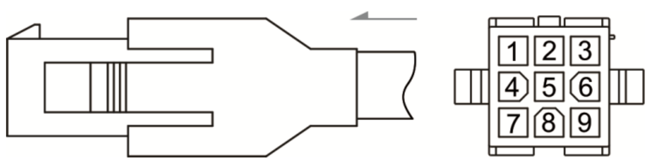
3.1.5 Specification of encoder connector

Encoder connection (Diagram 1):

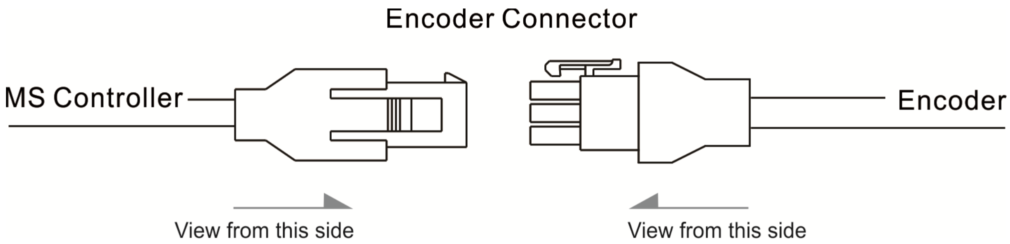


(1) Please refer to the information in this section
(2) Please refer to section 3.3
(3) About the adapting module, please refer to appendix B
(4) MOTOR.ENC connector

Note: This figure is just for illustration. Actual setup may vary according to the specifications of servo drive and motors.

Motor model	Encoder connector
ECMA-C1040F□S (50 W)	
ECMA-C△0401□S (100 W)	
ECMA-C△0602□S (200 W)	
ECMA-C△0604□S (400 W)	
ECMA-C△0604□H (400 W)	
ECMA-C△0804□7 (400 W)	
ECMA-C△0807□S (750 W)	
ECMA-C△0807□H (750 W)	
ECMA-C△0907□S (750 W)	
ECMA-C△0910□S (1000 W)	

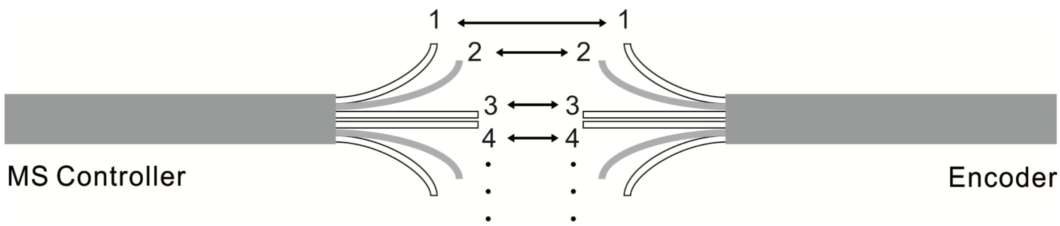
Specifications and pin assignment of encoder connector:



1 Blue T+	2 Green Reserved	3 Reserved
4 Blue/black T-	5 Green/Black Reserved	6 Reserved
7 Red/Red and white DC+5V	8 Black/Black and white GND	9 Shield

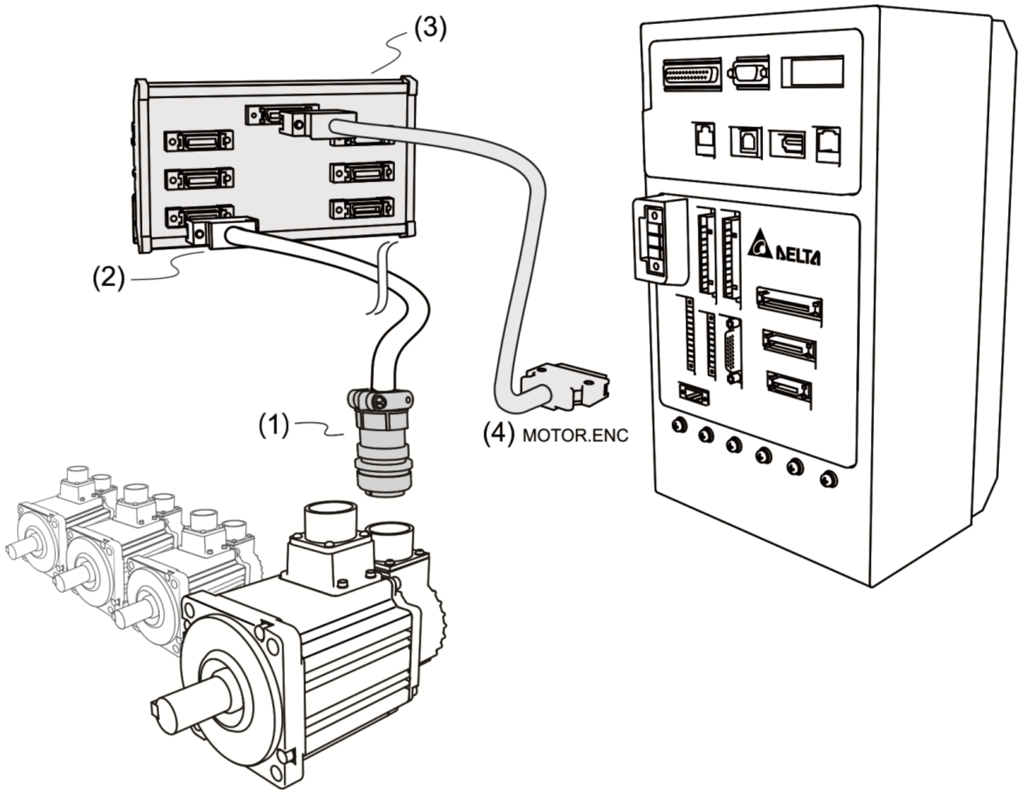
3 Reserved	2 Black Reserved	1 White T+
6 Reserved	5 Red/black Reserved	4 White/Red T-
9 Shield	8 Blue GND	7 Brown DC+5V

The wire color of MS controller is for reference only. Please refer to the actual parts.



To directly wire the cores without housing, please wire them according to the corresponding core number. For example, connect core No. 1 of the servo drive to No.1 of the motor encoder. Connect core No.2 of the servo drive to core No.2 of the motor encoder and so on. Please number the cores of the servo drive in sequence and then connect them to the encoder.

Encoder connection (Diagram 2):

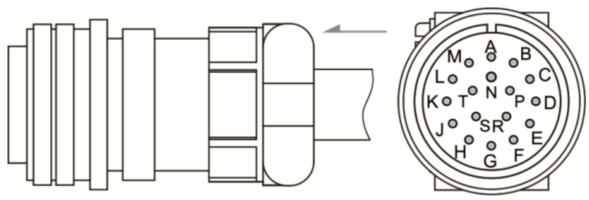


(1) , (2) please refer to section 3.3 “wiring for MOTOR ENC.”

(3) ASDPBSC2626: PCB adapter and wires (SCSI 26PIN SCSI 26 PIN, length: 0.5 M). Please refer to appendix B for more information

(4) MOTOR.ENC connector

Note: The diagram shows the connection between the servo drive and the encoder, which is not drawn by actual scale. The specification is subject to change according to the selected MS controller and motor models.

<table>
<tr><th>Motor model</th><th colspan="3">Military connector</th></tr>
<tr><td rowspan="7">ECMA-G△1303□S (300 W)
ECMA-E△1305□S (500 W)
ECMA-F△1305□S (500 W)
ECMA-G△1306□S (600 W)
ECMA-F△1308□S (850 W)
ECMA-G△1309□S (900 W)
ECMA-C△1010□S (1000 W)
ECMA-E△1310□S (1000 W)
ECMA-F△1313□S (1300 W)
ECMA-E△1315□S (1500 W)</td><td colspan="3"></td></tr>
<tr><td>Pin No.</td><td>Symbol</td><td>Color</td></tr>
<tr><td>A</td><td>T+</td><td>Blue</td></tr>
<tr><td>B</td><td>T -</td><td>Blue and black</td></tr>
<tr><td>S</td><td>DC+5</td><td>Red/Red and white</td></tr>
<tr><td>R</td><td>G D</td><td>Black/Black and white</td></tr>
<tr><td>L</td><td>BRAID SHIELD</td><td>–</td></tr>
</table>

Please use shielded stranded wire and connect it to BRAID and SHIELD. For more information, please refer to section 3.1.6.

Note: (□) in motor model names represents brake or keyway / oil seal.

3

3.1.6 Selection of wire rods

The recommended wire rods for connectors and signal wiring of MS controller are listed in the following table:

MS controller and corresponding servo motors		Power cable－wire diameter mm² (AWG)			
		L1c、L2c	R、S、T	U、V、W	P⊕、C
ASD-MS-0721-F	ECMA-C▵0401□S	1.3(AWG16)	2.1(AWG14)	0.82(AWG18)	2.1(AWG14)
	ECMA-C△0602□S				
	ECMA-C▵0604□S				
	ECMA-C▵0804□7				
	ECMA-C▵0807□S				
	ECMA-C▵0907□S				
	ECMA-E▵1305□S				
	ECMA-G▵1303□S				
	ECMA-G▵1306□S				
ASD-MS-1523-F	ECMA-C▵0910□S	1.3(AWG16)	3.3(AWG12)	1.3(AWG16)	3.3(AWG12)
	ECMA-C▵1010□S				
	ECMA-E▵1310□S				
	ECMA-E▵1315□S				
	ECMA-G▵1309□S				
	ECMA-C▵0807□S				
	ECMA-C▵0907□S				

MS controller	Encoder cable－wire diameter mm² (AWG)			
	Size	Number	Specification	Standard length
ASD-M-0721-F	0.13 (AWG26)	10 cores (4 pairs)	UL2464	3 m (9.84 inches)
ASD-M-1523-F	0.13 (AWG26)	10 cores (4 pairs)	UL2464	3 m (9.84 inches)

Note:
1. Please use shielded twisted-pair cable for encoder wiring in order to reduce the noise interference.
2. The shield should connect to the phase of SHIELD.
3. When wiring, please use the wire rods suggested mentioned above to avoid danger.
4. (□) in the servo motor model represents brake or keyway / oil seal.

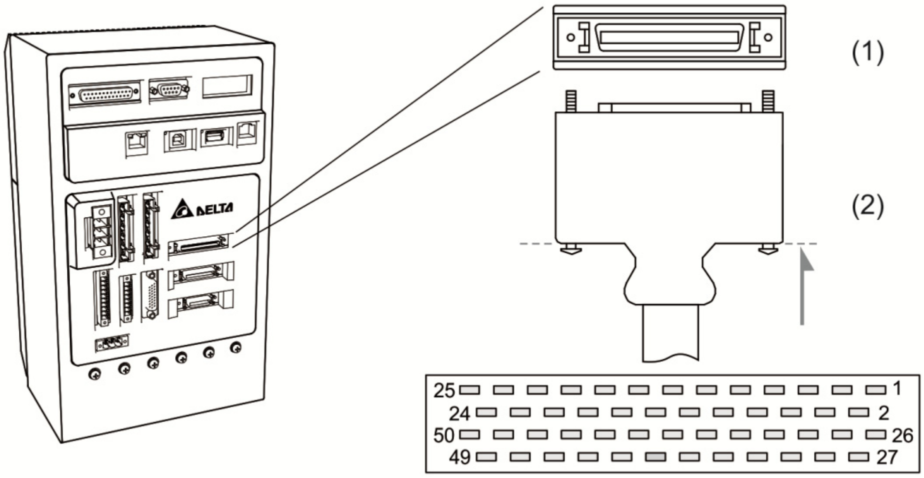
3.2 Wiring for STD.DIO and SYS.DIO I/O

3.2.1 I/O connector layout

3

MS controller provides 12 user-defined digital outputs and 24 user-defined digital inputs. In addition to that, there are 2 extra points for input and output respectively. These points can be set by P2-10, P2-11, P2-18 and P2-19. DI 1 to DI 6 and DI 13 to DI 18 use DI_COM1 for wiring. DI 7 to DI 12 and DI 14 to DI 24 use DI_COM 2 for wiring.

STD.DIO

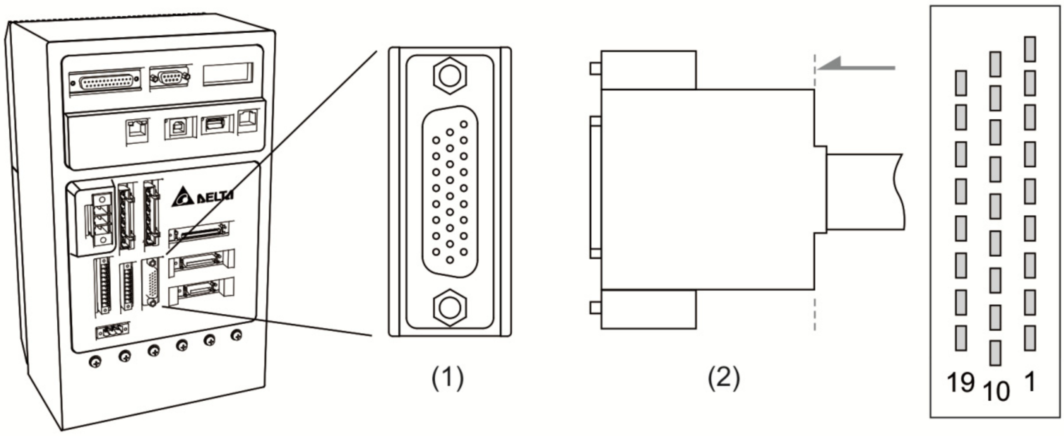


(1) STD.DIO connector (female); (2) STD.DIO connector (male)

Pin assignment:

Pin No	Symbol	Description	Pin No	Symbol	Description	Pin No	Symbol	Description
1	DI 1	Digital input	18	DO 05+	Digital output	35	DI 22	Digital input
2	DI 2	Digital input	19	DO 06+	Digital output	36	DI 23	Digital input
3	DI 3	Digital input	20	DO 07+	Digital output	37	DI 24	Digital input
4	DI 4	Digital input	21	DO 08+	Digital output	38	DI_COM 2	DI common 2
5	DI 5	Digital input	22	DO 09+	Digital output	39	DO 01-	Digital output
6	DI 6	Digital input	23	DO 10+	Digital output	40	DO 02-	Digital output
7	DI 13	Digital input	24	DO 11+	Digital output	41	DO 03-	Digital output
8	DI 14	Digital input	25	DO 12+	Digital output	42	DO 04-	Digital output
9	DI 15	Digital input	26	DI 7	Digital input	43	DO 05-	Digital output
10	DI 16	Digital input	27	DI 8	Digital input	44	DO 06-	Digital output
11	DI 17	Digital input	28	DI 9	Digital input	45	DO 07-	Digital output
12	DI 18	Digital input	29	DI 10	Digital input	46	DO 08-	Digital output
13	DI_COM 1	DI common 1	30	DI 11	Digital input	47	DO 09-	Digital output
14	DO 01+	Digital output	31	DI 12	Digital input	48	DO 10-	Digital output
15	DO 02+	Digital output	32	DI 19	Digital input	49	DO 11-	Digital output
16	DO 03+	Digital output	33	DI 20	Digital input	50	DO 12-	Digital output
17	DO 04+	Digital output	34	DI 21	Digital input	-	-	-

SYS.DIO



(1) SYS.DIO connector (female); (2) SYS.DIO connector (male)

Pin assignment:

Pin No	Symbol	Description	Pin No	Signal	Description	Pin No	Symbol	Description
1	SDO 3-	Digital output	10	SDO 2-	Digital output	19	SDO 5+	Digital output
2	SDO 3+	Digital output	11	SDO 2+	Digital output	20	SDO 5-	Digital output
3	SDO 4-	Digital output	12	SDO 7+	Digital output	21	SDO 6-	Digital output
4	SDO 4+	Digital output	13	SDO 7-	Digital output	22	SDO 6+	Digital output
5	SDI_COM	DI common	14	SDO 1+	Digital output	23	SDO 8-	Digital output
6	SDI_COM	DI common	15	SDO 1-	Digital output	24	SDO 8+	Digital output
7	SDI 5	Digital input	16	SDI 1	Digital input	25	SDI 2	Digital input
8	SDI 3	Digital input	17	SDI 6	Digital input	26	SDI 7	Digital input
9	SDI 8	Digital input	18	SDI 4	Digital input	-	-	-

3.2.2 I/O signal explanation

The following section details the signals listed in previous section:
There are various control modes in MS controller and their I/O settings may differ. Users can set their own DI/DO function depending upon their needs. However, the default setting of DI/DO should be able to fulfill most occasions.

Please choose the control mode according to your application. With the DI/DO table below, one can know the default DI/DO functions within the chosen control mode and pin numbers for wiring.

The table below list default DI/DO functions and pin numbers:

Default DO functions:

DO function	Control mode	Pin No		Description	Wiring method (refer to section 3.2.3)
		+	-		
SRDY	ALL	-	-	When MS controller is power-on with no alarm occurred in the control and motor power circuit, this DO is on.	C5/C6/ C7/C8
SON	None	-	-	When the servo motor is on, this DO is on.	
ALRM	ALL	28	27	When the alarm occurs, this DO is on. (Exception: forward / reverse limit, emergency stop, communication error and under voltage. These are WARN)	

Note: The item without default DI/DO number has no default functions. Users need to modify the corresponding parameters if wish to use them. Please refer to chapter 8 for detailed description.

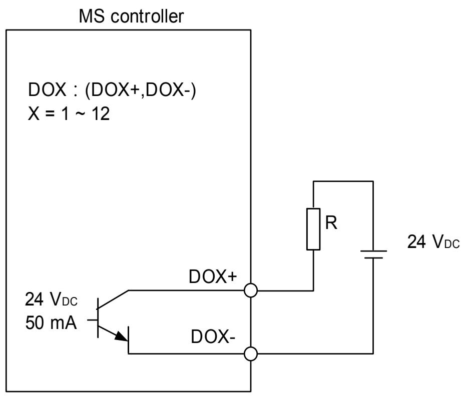
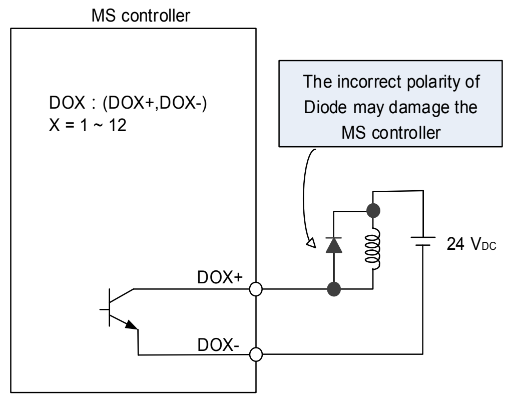
Default DI functions:

DI function	Control mode	Pin No	Description	Wiring method (refer to section 3.2.3)
ARST	ALL	-	This function is used to clear alarms (ALRM). When the servo drive is reset, SPDY will be on.	C9/C10 C11/C12

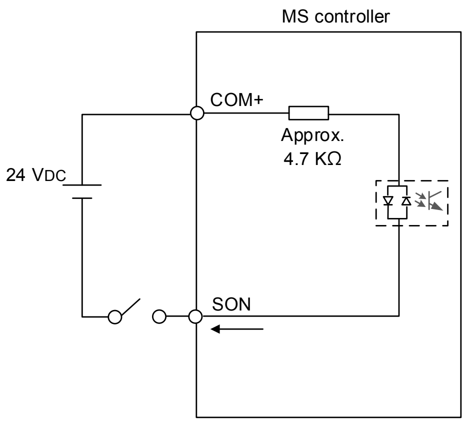
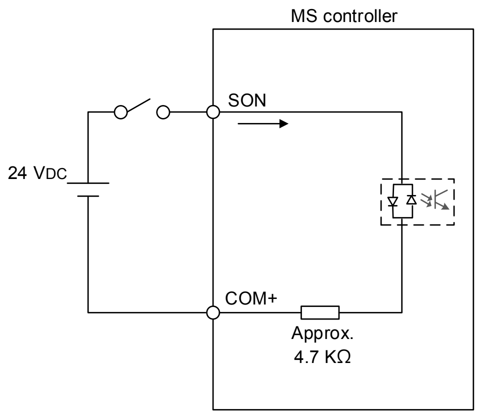
3

3.2.3 Wiring diagrams (digital I/O)

When the drive connects to inductive load, the diode has to be installed. (Permissible current: below 40 mA; Surge current: below 100 mA)

C1: DO wiring – the MS controller applies external power supply and the resistor is general load.	C2: DO wiring - the MS controller applies external power supply and the resistor is inductive load.
	

DI wiring - Input signals by relay or open-collector transistor.

NPN transistor (SINK mode) C3: DI wiring – the MS controller applies external power supply	PNP transistor (SOURCE mode) C4: DI wiring – the MS controller applies external power supply
	

WARNING

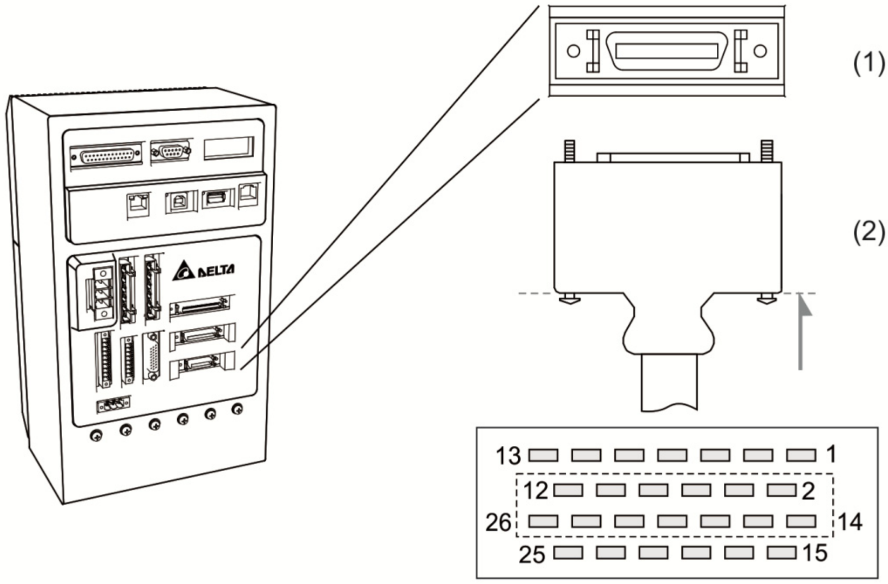
Caution: Do not apply to dual power or it may damage the MS controller.

3.2.4 User-defined DI and DO signal

If the default DI/DO function cannot meet the application requirement, users can set the function of DI 1~ DI 6 and DO 1 ~ DO 3 via the corresponding parameters P 2-10, P2-11 and P2-18. That is, the DI/DO functions can be specified by setting these parameters. (Please refer to chapter 8 for detailed description.)

3.3 Wiring for MOTOR ENC.

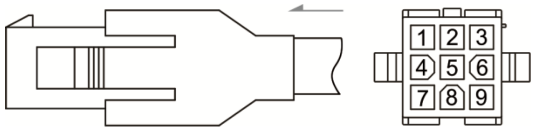
The CN2 encoder signal connector is illustrated as follows:



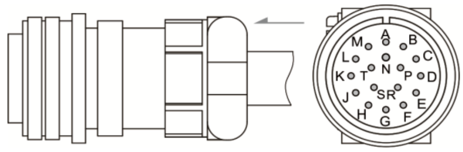
(1) Motor ENC. Connector (female); (2) Motor ENC. Connector (male)

3

Encoder connector



Quick connector

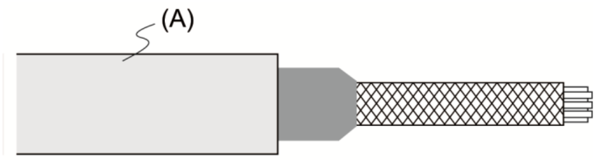


Military connector

Pin assignment:

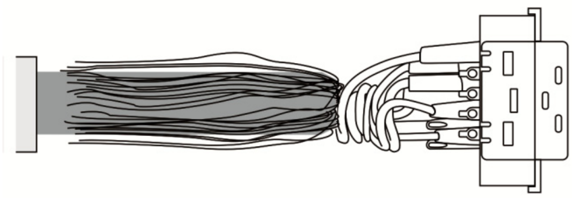
The end that connects to MS controller								The end that connects to the motor		
Symbol	Color	Pin No						Military connector	Quick connector	Color
Axis	-	A	B	C	D	E	F	-	-	-
T+	Blue	5	11	17	23	14	20	A	1	Blue
T-	Blue and black	7	13	19	25	15	21	B	4	Blue and black
+5V	Red/Red and white	4	10	16	22	2	8	S	7	Red/ Red and white
GND	Black/black and white	6	12	18	24	3	9	R	8	Black/Black and white
Reserved	-	1, 26						-	-	-
Shielding	Shielding	Case						L	9	-

The shielding procedures of CN2 encoder connector are as followings:



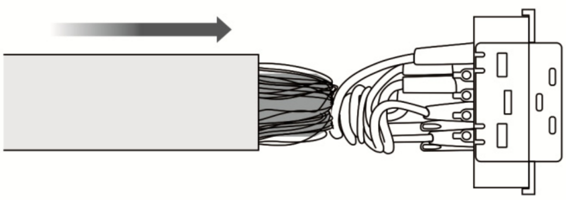
Step 1:

Cut through the cable and expose the core wire which covers the metal shielding. The reserved core wire length should be 20 ~ 30 m and then apply 45 mm heat shrink (A)



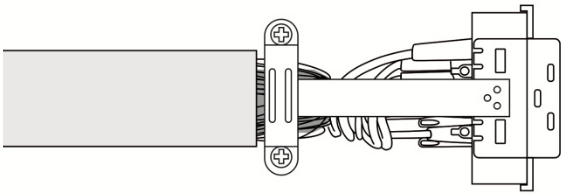
Step 2:

Spread the metal core wires with shielding and fold them upside down in downward direction. Refer to the table above to connect the pins one by one



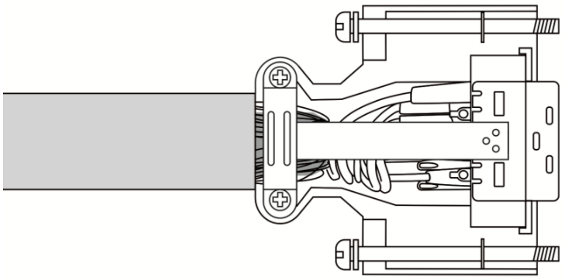
Step 3:

Keep about 5 ~ 10 mm metal shielding exposed (the length is about the same as the metal buckle’s width). Cover the remainder with heat shrink for isolation.



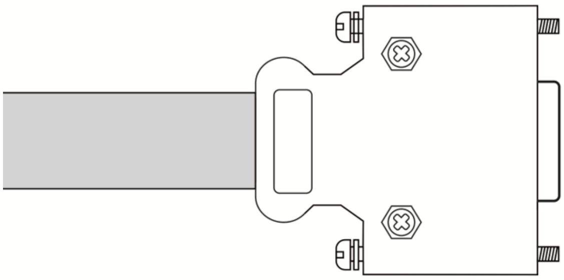
Step 3:

Screw the metal buckle to fix the metal shielding. The buckle has to cover all exposed metal shielding. The metal sheet on the top must join to the metal part of the connector.



Step 5:

Put it into the connector case as shown in figure.



Step 6:

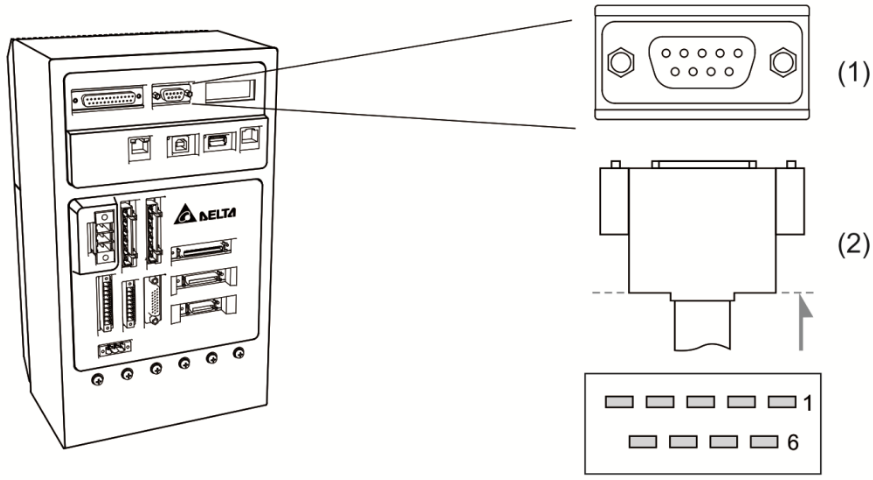
Screw the case and done.

3

3.4 Wiring for D-SUB communication connector

3.4.1 Communication port layout

MS controller communicates with PC via communication port. Users can write LUA programs to interact with HMI and PLC by Modbus communication protocol. MS controller is equipped with RS-232 and RS-485 interfaces (configurable by P3-05). RS-232 is more common and its cable length is up to 15 meters (50 feet). RS-485 can support longer transmitting distance and connect multiple MS controllers in parallel.



(1) RS-232, RS-485 communication connector (female); (2) RS-232, RS-485 connector (male)

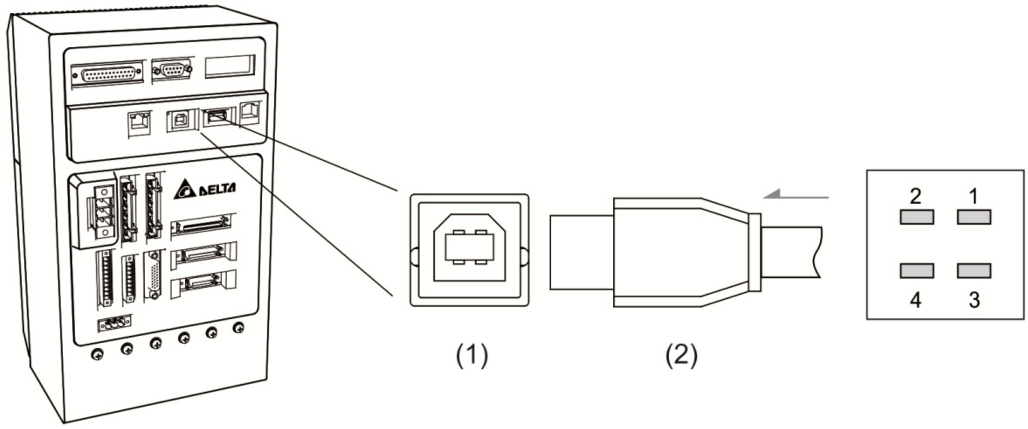
Pin assignment:

Pin No	Signal name	Symbol	Description
1	RS-485 data transmission	RS-485(+)	Differential signal (+) at MS side
2	RS-232 data receive	RS-232_RX	Data receive at MS to connect to data transmit at PC
3	RS-232 data transmission	RS-232_TX	Data transmit at MS to connect to data receive at PC
4	-	-	Reserved
5	Ground	GND	Ground for +5 V and signal
6	RS-485 data transmission	RS-485(-)	Differential signal (-) at MS side
7	-	-	Reserved
8	-	-	Reserved
9	-	-	Reserved

Note: When the field is relatively less noise, the cable length can be up to 15 meters. Further, if the baud rate is higher than 38400 bps, please keep the length less than 3 meters in order to guarantee accurate transmission.

3.5 USB port

USB1 port: for connecting to PC software in order to operate MS controller.



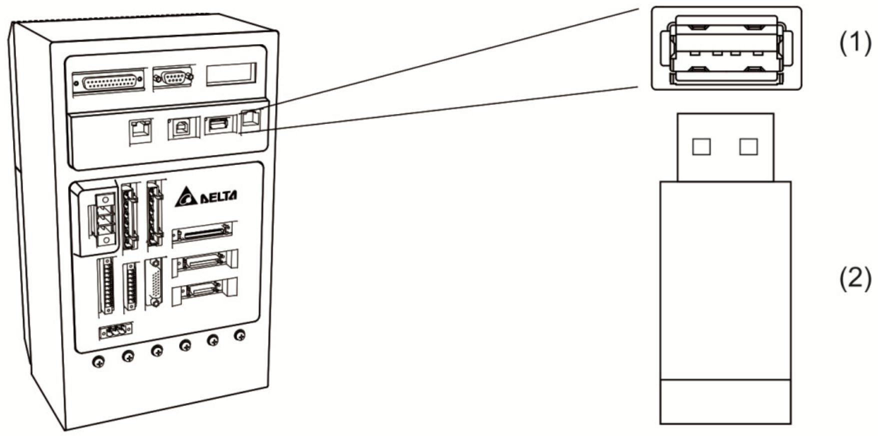
(1) USB1 port (2) USB1 connector

Pin assignment:

Pin No	Signal name	Description
1	V bus	DC +5 V (from external)
2	D-	Data-
3	D+	Data+
4	GND	Ground

3

USB2 port: for mass storage



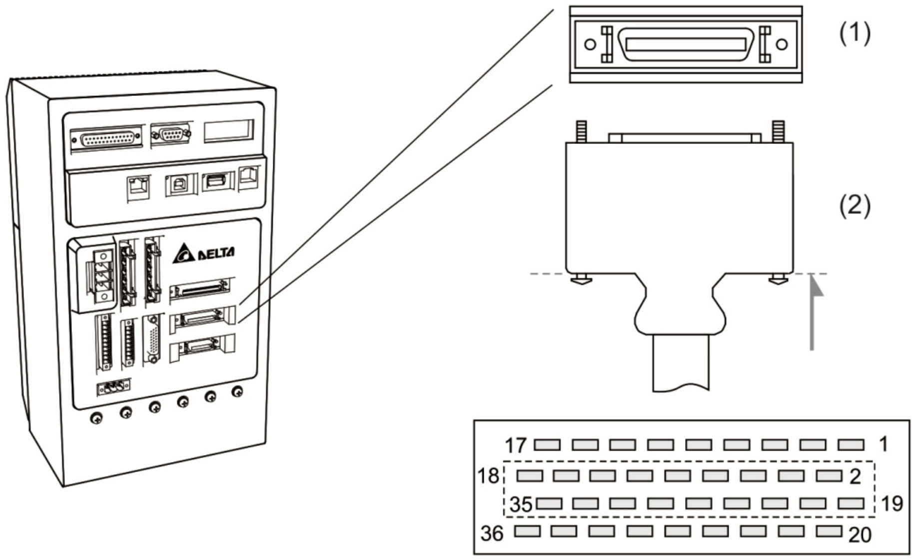
(1) USB2 port (2) USB2 connctor

Pin assignment:

Pin No	Signal name	Description
1	V bus	DC +5 V (from external)
2	D-	Data-
3	D+	Data+
4	GND	Ground

3.6 EXT.ENC. connector (position feedback signal for full-closed loop control)

External linear scale or encoder (A, B, Z phase) can connect to EXT. ENC. connector for full-closed loop control or conveyor tracking application (CVT). Please refer to parameters: P2-12 ~ P2-14 in chapter 8.



(1) EXT.ENC. connector (female); (2) EXT.ENC. connector (male)

Pin assignment:

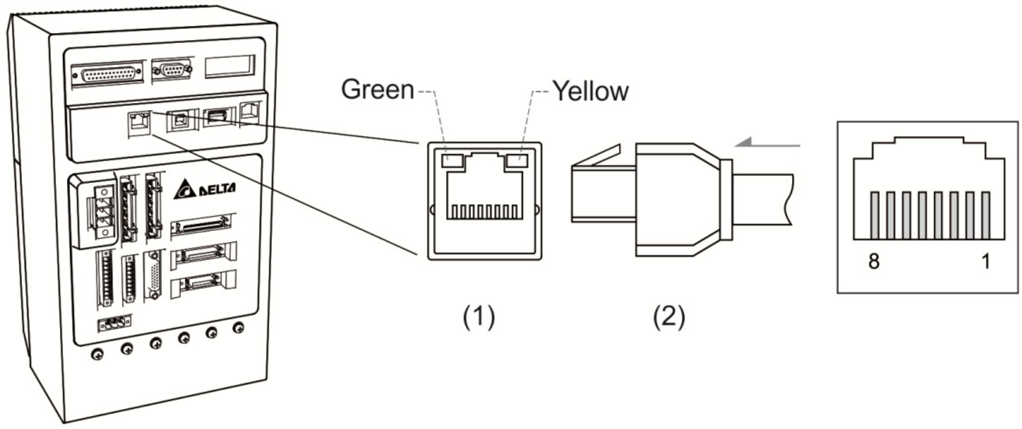
Signal name	Symbol	Description	Pin No.			
			Axis 1	Axis 2	Axis 3	Axis 4
A phase input	OPT_A	Linear scale A phase output	5	17	23	35
/A phase input	OPT_/A	Linear scale /A phase output	6	18	24	36
B phase input	OPT_B	Linear scale B phase output	3	15	21	33
/B phase input	OPT_/B	Linear scale /B phase output	4	16	22	34
Z phase input	OPT_Z	Linear scale Z phase output	1	13	19	31
/Z phase input	OPT_/Z	Linear scale /Z phase output	2	14	20	32
Encoder ground	GND	Ground	8	11	26	29
Encoder power	+5V	Linear scale + 5 V power	7	12	25	30
-	-	Reserved	9	10	27	28

3

3.7 Connector of Ethernet / DMCNET communication

MS controller supports Ethernet / DMCNET communication protocols. Parameters in MS controller can be set via Ethernet. Besides, external connected servo drives and modules can be controlled via DMCNET communication. The wiring diagrams are shown below:

Ethernet:

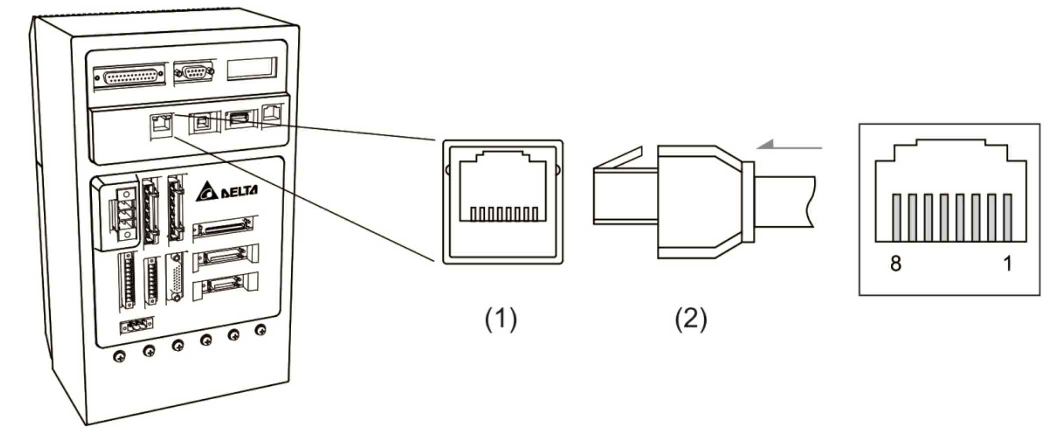


(1) Ethernet connector (female); (2) Ethernet connector (male)

Pin assignment:

Pin No	Symbol	Description
1	TXP	Ethernet TX+
2	TXN	Ethernet TX-
3	RXP	Ethernet RX+
4	-	Reserved
5	-	Reserved
6	RXN	Ethernet RX-
7	-	Reserved
8	-	Reserved

DMCNET:



(1) DMCNET connector (female); (2) DMCNET connector (male)

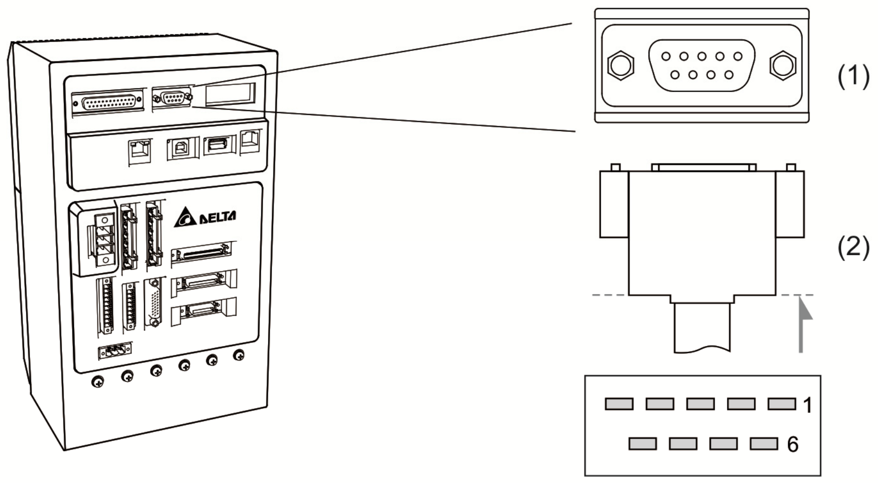
Pin assignment:

Pin No	Symbol	Description
1	DMC_A1	DMCNET 1+
2	DMC_B1	DMCNET 1-
3	DMC_A2	DMCNET 2+
4	-	Reserved
5	-	Reserved
6	DMC_B2	DMCNET 2-
7	-	Reserved
8	-	Rreserved

3

3.8 Connector of HMI TP communication

Delta HMI can access parameters in MS controller via communication. Normally open and normally closed functions for emergency stop are defined in different pin on this interface. Please refer to the table below:



(1) TP communication connector (female); (2) TP communication connector (male)

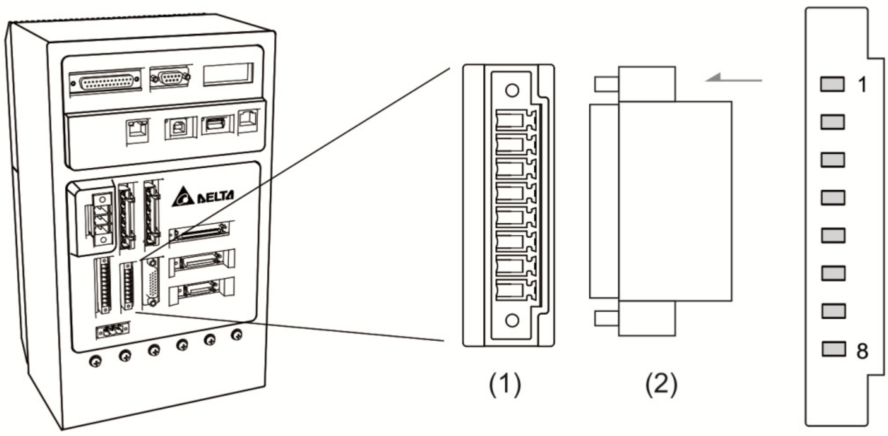
Pin assignment:

Pin No	Signal name	Description	Pin No	Signal name	Description
1	HMI_TX+	Ethernet TX+	14	HMI_RX+	Ethernet RX+
2	HMI_TX-	Ethernet TX-	15	HMI_RX-	Ethernet RX-
3	-	Reserved	16	-	Reserved
4	-	Reserved	17	-	Reserved
5	PW	24 V	18	PW	24 V
6	GND	0 V	19	GND	0 V
7	E_STOP_NO+	Emergency stop (NO)	20	ENC_EXA	MPG input (A)
8	E_STOP_NO-	Emergency stop (NO)	21	ENC_EXB	MPG input (B)
9	E_STOP_NC+	Emergency stop (NC)	22	ENSW_NC+	Jog (NC)
10	E_STOP_NC-	Emergency stop (NC)	23	ENSW_NC-	Jog (NC)
11	GND	0 V	24	PW	24 V
12	GND	0 V	25	PW	24 V
13	GND	0 V	-	-	-

3

3.9 STO connector

3.9.1 STO pin assignment



(1) STO connector (female); (2) STO connector (male)

Pin assignment:

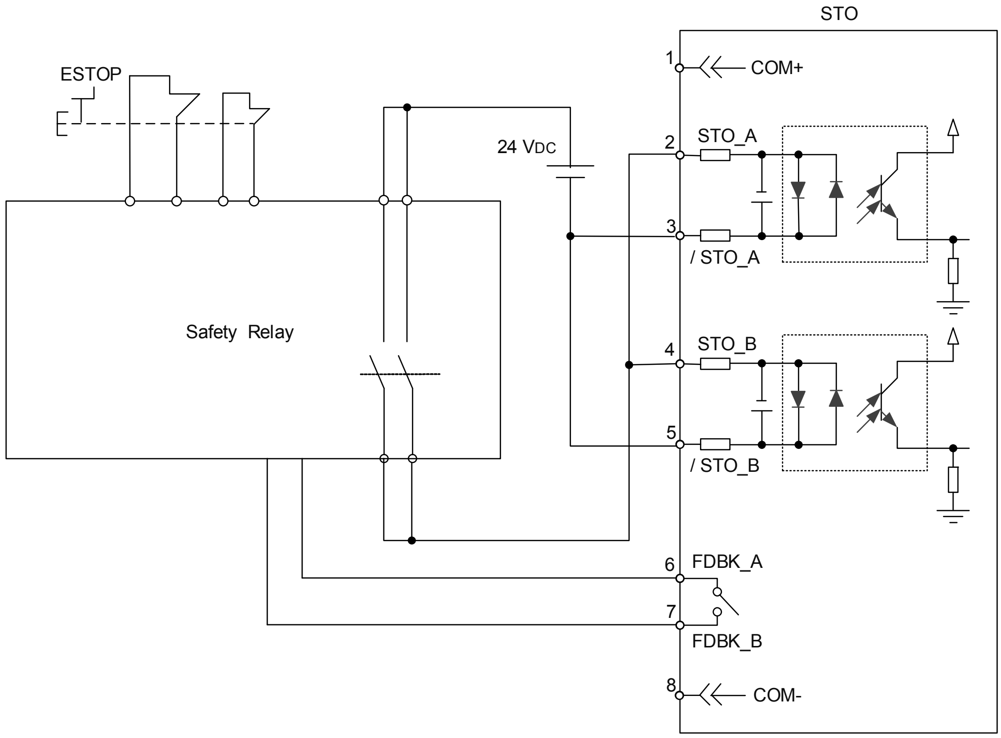
Pin No	Symbol	Description
*1	COM+	VDD (24 V), same as pin 5 on CN1
2	STO_A	STO input A+
3	/STO_A	STO input A-
4	STO_B	STO input B+
5	/STO_B	STO input B-
6	FDBK_A	STO alarm output A: BJT Output Maximum voltage/current: 80 VDC / 0.5 A
7	FDBK_B	STO alarm output B: BJT Output Maximum voltage/current: 80 VDC / 0.5 A
8	COM-	Ground for VDD(24V)

WARNING

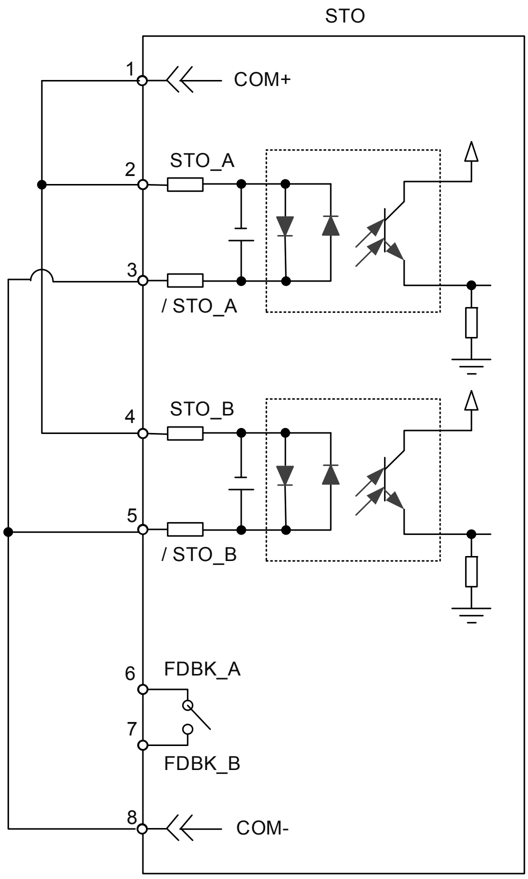
***1 Caution: Do not apply to dual power to COM+ or it may damage MS controller.**

3

Wiring for STO and safety relay:



Wiring for disabling STO:



3.9.2 STO safety function

Principle of STO

STO function is controlled by the motor current from two individual circuits. It cuts off the power supply to the motor when needed and the motor is now free of torque force. The following table elaborates how this function works.

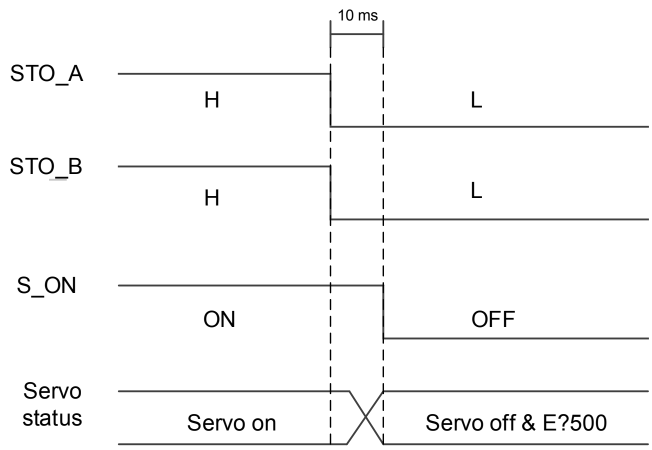
3

Pin assignment:

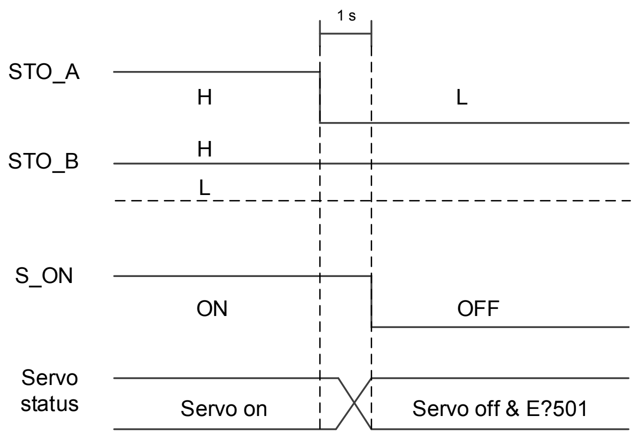
STO signal	Status of Opto-Isolator			
STO_A、/STO_A	ON	ON	OFF	OFF
STO_B、/STO_B	ON	OFF	ON	OFF
Output status	Ready	Torque off (STO_B lost)	Torque off (STO_A lost)	Torque off (STO Mode)

(1) Description of STO alarm:

See the diagram below. When the motor runs normally (SERVO ON) but STO_A and STO_B signals are both gone for 10 ms at the same time, AL500 will occur and the MS controller will be in the state of SERVO OFF.



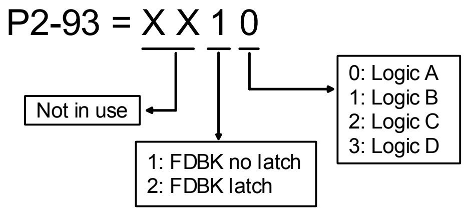
When the motor runs normally (SERVO ON) but one of the safety signal source is gone for 1s, AL501 or AL502 will occur. Then, the servo drive will be in the state of SERVO OFF.



3

3.9.3 Related parameter of STO function

By setting parameter P2-93, users can determine the FDBK status and whether FDBK will latch if STO alarm occurs. The setting of P2-93 is shown as follows:



See the table below. Four logics (Logic A, B, C, and D) are presented to standardize the FDBK status when different STO alarm occurs. Users can select the corresponding logic according to the demands. (In this table, “Open” means FDBK+ and FDBK- of STO port are open circuit. Take Logic C as the example. When AL500 occurs, FDBK+ and FDBK- of STO port are short circuited.)

Status of MS controller		FDBK_A & FDBK_B Statuses							
		Logic A		Logic B		Logic C		Logic D	
Parameter P2.093		XX10	XX20	XX11	XX21	XX12	XX22	XX13	XX23
FDBK behavior		No Latch	Latch	No Latch	Latch	No Latch	Latch	No Latch	Latch
No STO alarm occurs		Open		Close		Open		Close	
Alarm occurs	AL500	Close		Open		Close		Open	
	AL501	Close		Open		Open		Close	
	AL502	Close		Open		Open		Close	
	AL503	Close		Open		Open		Close	

Note: Open = open circuit; Close = short circuit

FDBK behavior (Latch /No Latch):

If FDBK is latched, when STO alarm occurs, status of FDBK will remain even when the alarm has been cleared. Please note that when more than one alarm occurs, the MS controller will only display AL500.

Example of Latch:

If Logic C P2-93 = XX22 is set, the FDBK status will be close when safety signal is lost and AL005 occurs. Since FDBK is selected as Latch, even when the safety signal is back to normal, FDBK status will remain close. Use the approaches below to reset.

1. Reconnect the power supply. FDBK status returns to “open”.
2. Do not reconnect the power supply. Instead, set P2-93 to XX12 to make FDBK status return to “open”. Then set P2-93 to XX22 again. This step is to set FDBK behavior to Latch.

After the FDBK status restores, alarms can be cleared by normal corrective actions. In this case, AL500 can be cleared by DI.Alm Reset.

Example of No Latch:

If Logic C P2-93 is set to XX12, the FDBK status will be “close” when the safety signal is lost and AL500 occurs. Since FDBK is selected as No Latch, safety signals return to normal and the FDBK status automatically changes from short-circuited to normal when AL500 occurs. Setting P2-93 to XX12 again is not required.

After the FDBK status restores, alarms can be cleared by normal corrective actions. In this case, AL500 can be cleared by DI.Alm Reset.

Relevant parameter (Please refer to Chapter 8 for detailed information)

Para. No.	Description
P2-93	STO FDBK control

3

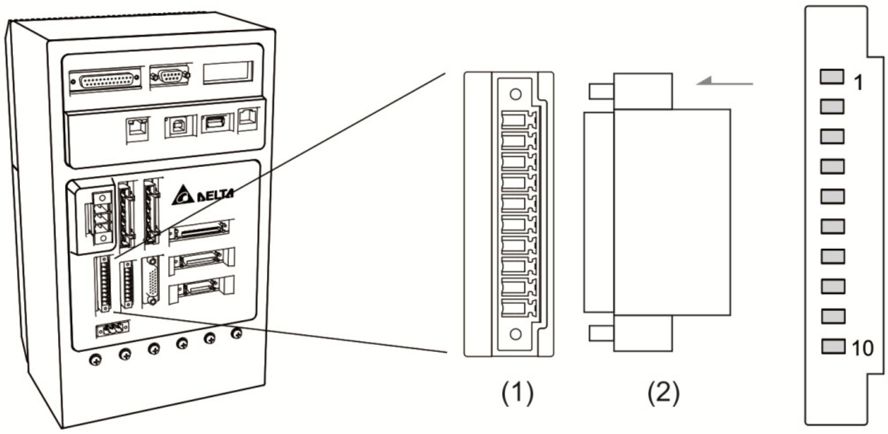
3.9.4 STO related alarms

Alarm code	Description
E?500	STO is activated
E?501	STO_A lost (signal lost or error)
E?502	STO_B lost (signal lost or error)
E?503	STO error

Please refer to chapter 11 for detailed description.

3.10 BRK.DO connector

MS controller provides BRK.DO for disengaging the motor brake. This function can be also achieved by communication function.



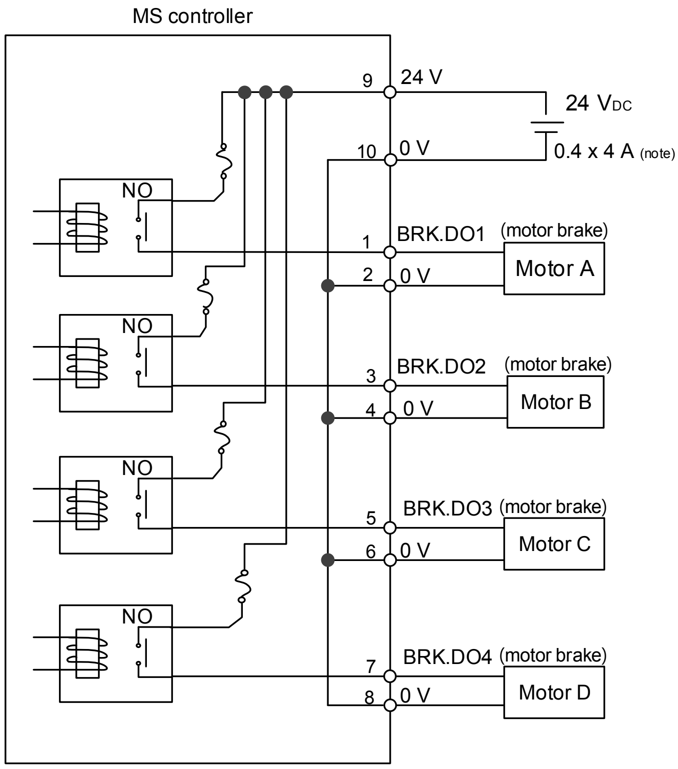
(1) BRK.DO connector (female); (2) BRK.DO connector (male)

Pin assignment:

Pin No	Signal name	Description
1	BRK.DO1	Digital output (A)
2	0V	0 V (A)
3	BRK.DO2	Digital output (B)
4	0V	0 V (B)
5	BRK.DO3	Digital output (C)
6	0V	0 V (C)
7	BRK.DO4	Digital output (D)
8	0V	0 V (D)
9	24V	24 V input
10	0V	0 V input

3

Wiring for BRK.DO: (this diagram is only for 750W model)



Note: The required power differs according to the number of the motors. Please refer to the table below

Number of motor(s)	Current (A)
1	1 x 0.4 = 0.4
2	2 x 0.4 = 0.8
3	3 x 0.4 = 1.2
4	4 x 0.4 = 1.6

Panel Display and Parameters Setting

4

This chapter describes panel display of MS controller and its operation. User can monitor the operation state and see if any alarm occurs via the panel.

4.1 Status display

4.1.1 System state

Displayed symbol	Description
boot.	System start-up or system reboot
rUn	System ready
StoP	PLC stops operating
PWoFF	System stops in 3 seconds. You can turn off the power or re-power on the system.
bbFlH	EC005 Nand Flash problem solving
USbLd	USB updating and loading.
FWUPd	Firmware updating
rEtrY	Fail to update the firmware. Retrying.
rEbot	Firmware update successful. Rebooting the system.
noUPd	Error occurs on firmware file to be updated. Firmware update is not carried out.
FLASh	Firmware burning.
donE	Burning complete.
FAIL	Fail to burn the firmware. Update failure after 3 attempts.

4.1.2 Alarm message

Displayed symbol	Description
 (Error d: Axis 13 Error code: 013)	Most left number (the first number): Fixed display “**E**” The second number: Alarm type (1) Controller: “**C**” (2) Group: “**1.**“ or “**2.”** (The decimal point represents Group) (3) Axis: Display in vigesimal format. See details below: Axis 1 to 6 is represented by 1 ~ 6 Axis 7 to 12 - reserved Axis 13 to 18 is represented by D ~ I (4) User: “**U**” The last three numbers: Error Code. Please refer to the next section (description of parameter P0-01) or Chapter 11 (Troubleshooting) for its definition.

4.2 Description of P0-01

When setting P0-01 to 0 for clearing the error, the panel will display:

When accessing parameters via communication, if any alarm is not cleared, the returned value will be a non-zero and 32-bit value while the first 16 bits represent Index and the later 16 represent alarm code. See the table below.

Index (16-bit)				Error Code (16-bit)
U	Z	Y	X	Error Code (Word)
Group No. or Axis No.		Reserved (0x0)	Type	
(1)		(2)	(3)	(4)

(1) U and Z: They represent the number for Group or Axis with 4-bit data size.

(2) Y: System reserved (0x0)

(3) X: 4-bit data size

0x0: Controller

0x1: Group

0x2: Axis

0x3: User (User-defined)

0x4 ~ 0xF: System reserved

(4) Error Code: Please refer to Chapter 11, Troubleshooting

For example, when the panel displays Ed013, the software reads P0-01 from the controller and the returned value will be 0x0D020013.

Index (16-bit)				Error Code (16-bit)
U	Z	Y	X	013
0D		Reserved (0x0)	2	

The controller displays the alarm message of axis 13, Group type and 013 is its alarm code.

4.3 Edit parameters

Parameter editing section provided by DRAS manages all parameters of the controller and servo drive. Users can read, write or browse parameters here. It is divided into three sections, (1) tool bar, (2) parameter tree and (3) parameter list. See figure 4.3.1.

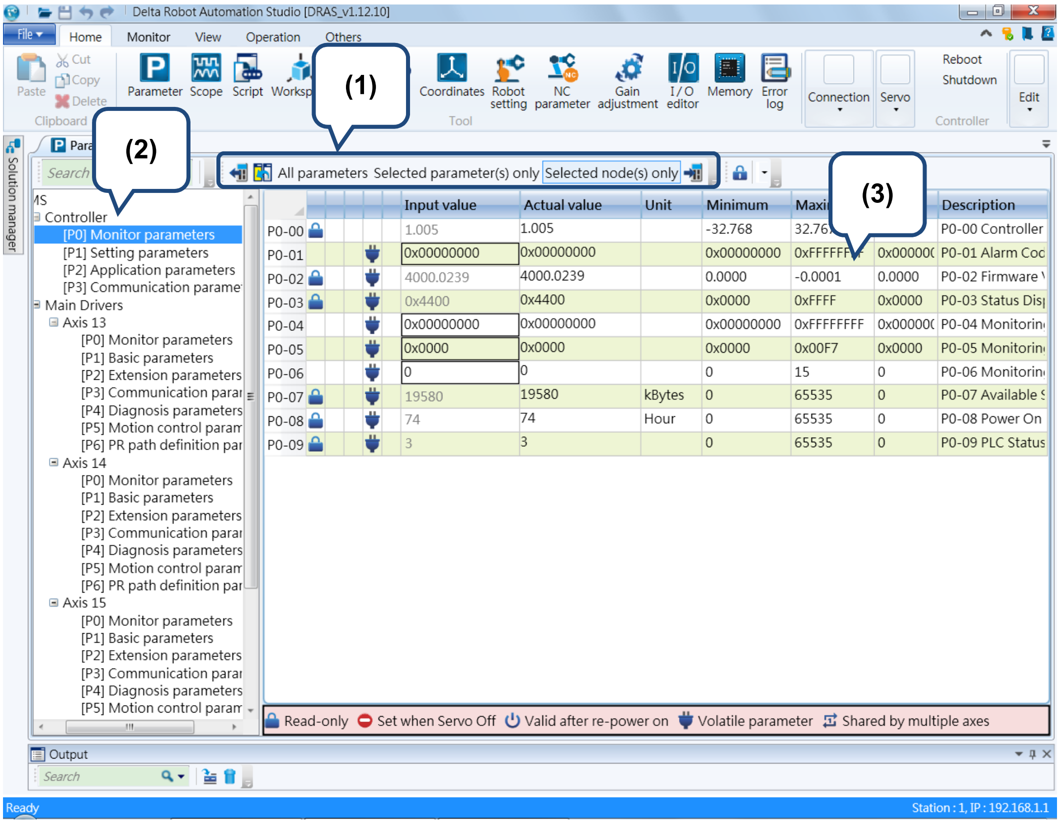


Figure 4.3.1 Parameter

4.3.1 Tool bar

Please refer to the following table of each function in tool bar.

Purpose	Icon	Function
Parameters editing		Read parameters from the controller
		Write parameters value into MS controller.
		Replace input values with actual values
	All parameters	Edit all parameters
	Selected parameters(s) only	Edit the selected parameter(s) only
	Selected node(s) only	Edit the selected node(s) only
		Stop current operation
Other		Lock or unlock the hidden parameter
	Regular display mode	Switch the display mode. Three display modes are available.

4

Purpose	Icon	Function
		1. Regular display mode 2. Only display items which actual value is different from the default value 3. Only display items which actual value is different from the input value
		Input the keyword to find the related parameters

Table 4.3.1 Tool bar

4.3.2 Parameter tree

Parameter tree is used to present parameter groups in a tree structure. There are more than 3000 parameters in the controller. With the parameter tree, it is easier to edit and search the needed parameters. By clicking on a node of the parameter tree, you can view all parameters of the node in the parameter list on the right hand side. See the figure 4.3.2.1.

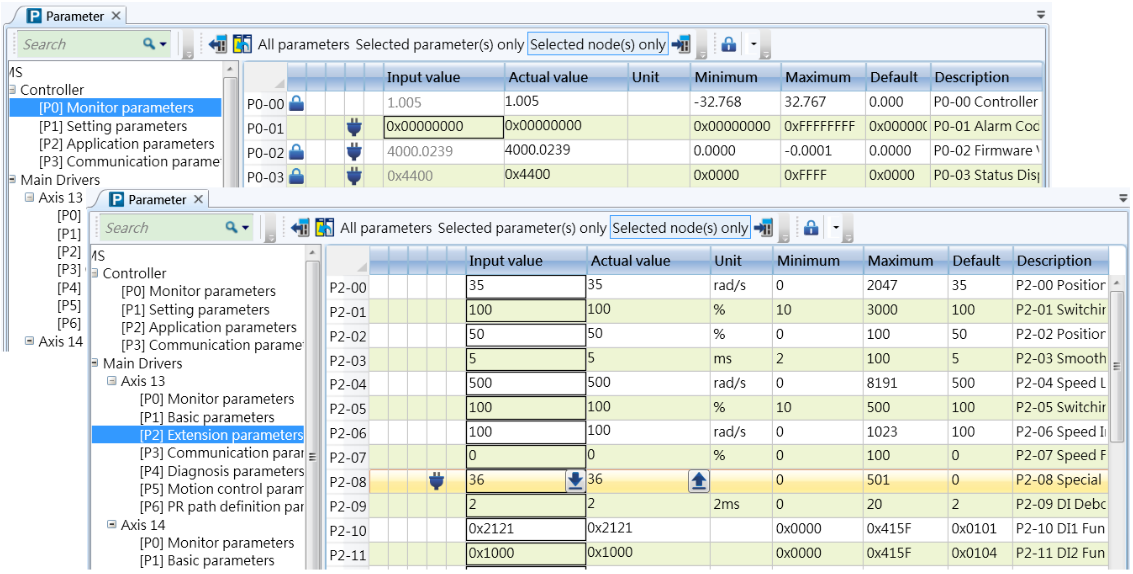


Figure 4.3.2.1 Selected items of parameter tree and parameter list

4.3.3 Parameter list

Parameter list includes parameter's actual value, minimum and maximum value, default value, description and input value. The symbol showed next to the input value represents the parameter's property. Please refer to the detailed description of each property below.

Icon of parameter property	Description
	This parameter is read-only.
	When it is in Servo On status, parameters cannot be set.
	The parameter setting is valid after re-power on.
	When the power is off, the parameter value will be reset to the default.

Table 4.3.2 Parameter property

See figure 4.3.3.1. User can edit and write the value in **Input value** column. However, parameter value is read-only in **Actual value** column. If you wish to change one parameter value, please write the value in **Input value** column first. Then, download it to the controller by either clicking on the **Download** button or directly pressing the **Enter** key. See the figure below. The **Write** button also can be used to download the parameter value.

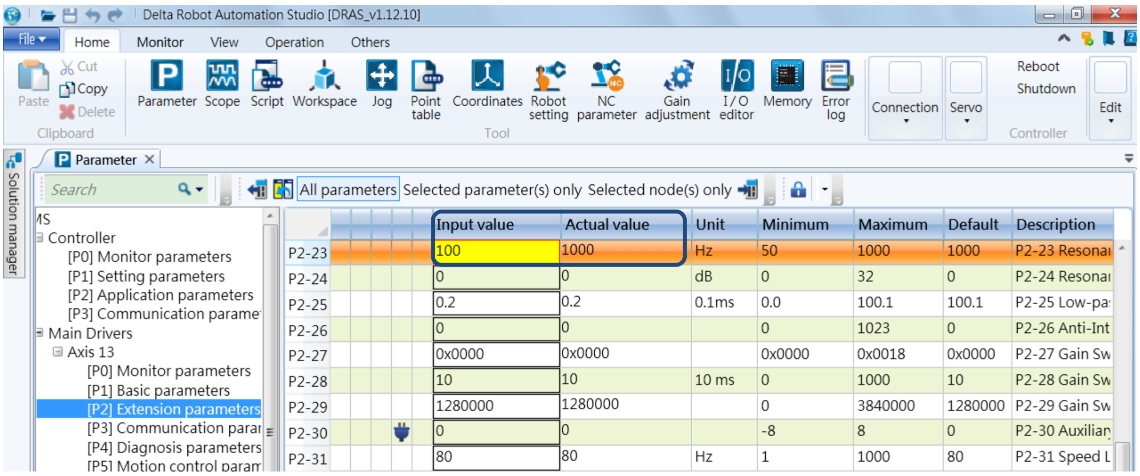


Figure 4.3.3.1 Input value and Actual value columns

When the value in **Input value** column is different from the value in **Actual value**, its background color will be yellow. The value in **Input value** column can be updated to the same value in **Actual value** by clicking the button of **Replace input values with actual values**.

Users can read and set one or multiple parameters. Three modes are provided to set parameters, which are **All parameters**, **Selected parameter(s) only** and **Selected node(s) only**. See figure 4.3.3.2.

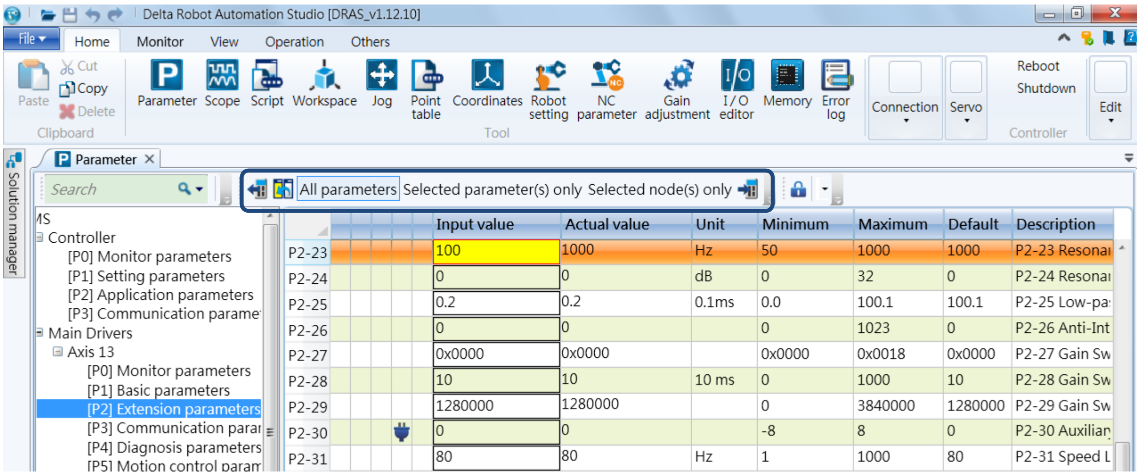


Figure 4.3.3.2 Tool bar - Parameters

In this section, user can select one mode to set parameters. The option **All parameters** shows all parameters of the controller and servo drive. **Selected parameter(s) only** shows the parameters you selected with orange background (figure 4.3.3.3). **Selected node(s) only** option shows all parameters of the selected node (blue background) in parameter list. See figure 4.3.3.4.

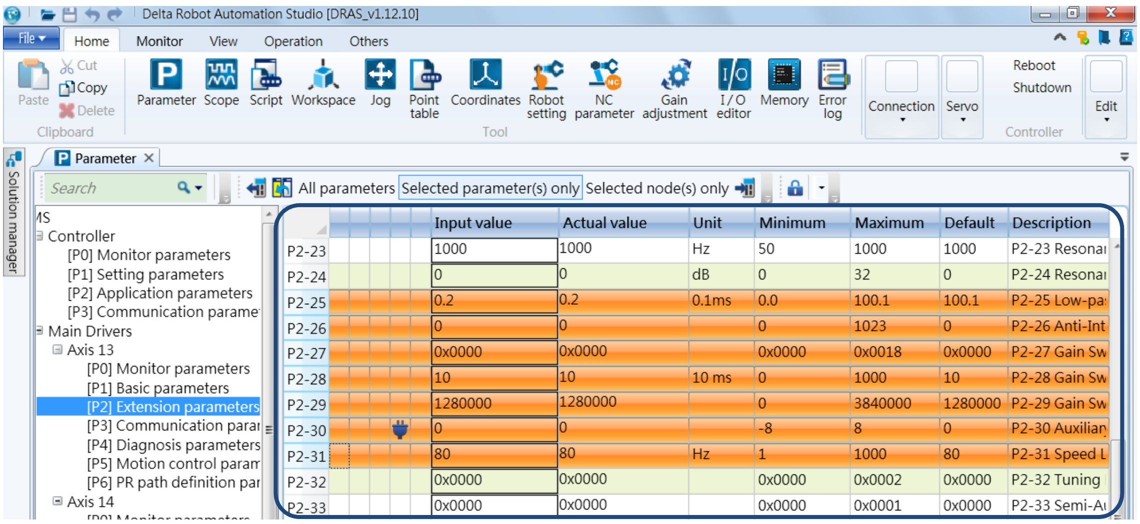


Figure 4.3.3.3 Select more than one parameter

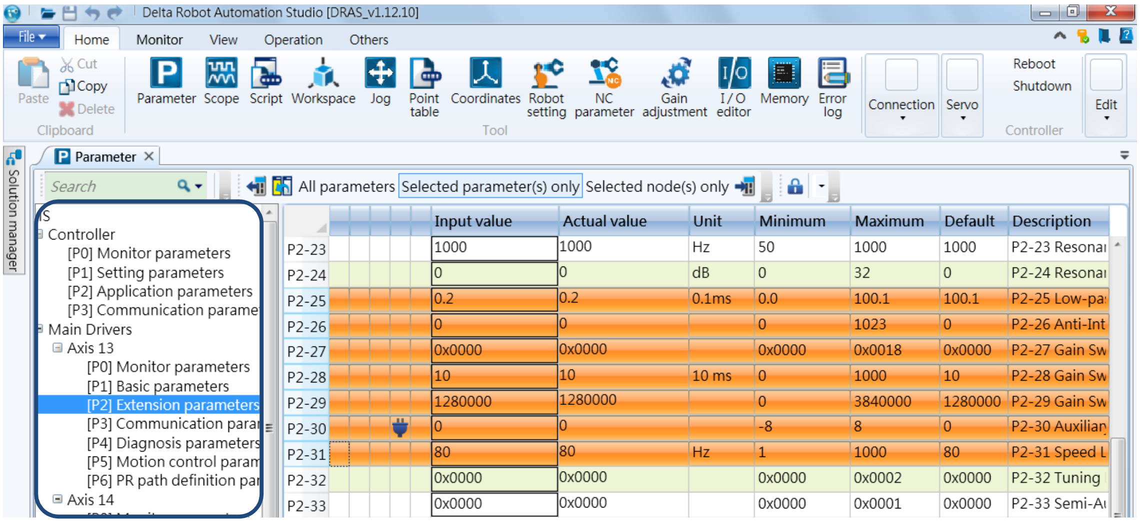


Figure 4.3.3.4 Selected node of parameter tree

To search a parameter, input the keyword in the search box and press Enter. Then, you will find the result in the parameter list. See figure 4.3.3.5.

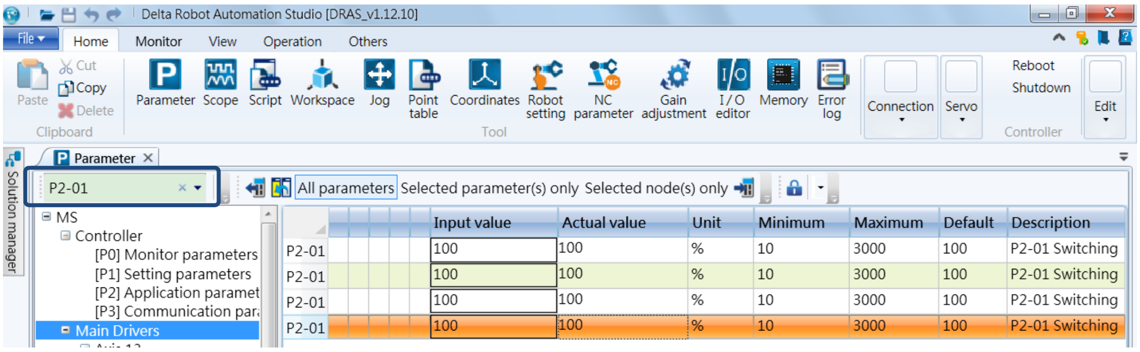


Figure 4.3.3.5 Search tool

In **Parameter,** data can be imported/exported by a parameter (.dpar) file. Please note that exporting the file is for recording the actual value of all parameters; importing is for updating all data in **Input value** column. To update data with a parameter file, you can import the file first to the controller/servo drive and then click the write icon.

Three display modes are available. See figure 4.3.3.6. Users can select the item from parameter list after selecting the display mode.

1. Regular display mode: Display all parameters.
2. Only display items which actual value is different from the default value: Filter those items that actual value is different from the default value and display in parameter list.
3. Only display items which actual value is different from the input value: Filter those items that actual value is different from the input value and display in parameter list.

4

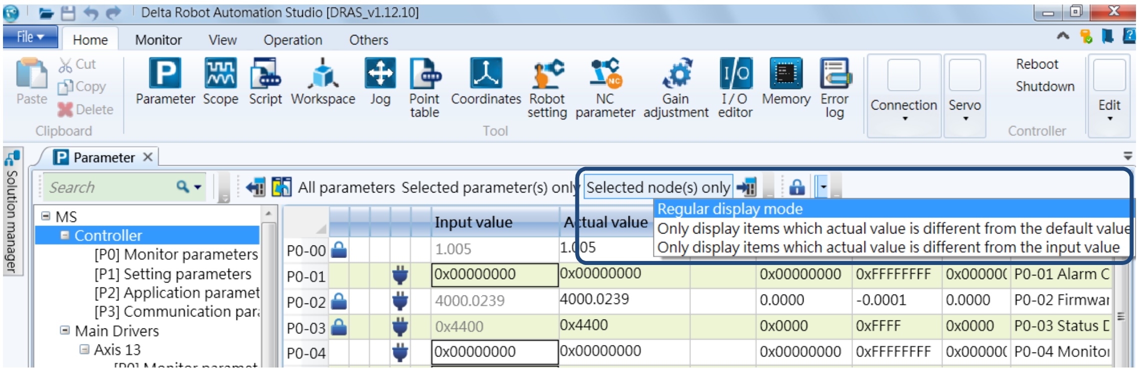


Figure 4.3.3.6 Switch the display mode

4.3.4 Parameter editing

View relevant parameters

The parameter editing section enables users to preview the related parameters of the controller. Start the DRAS and connect it with the controller. Then, click on **Parameter**. A window for editing parameters will pop up. The parameter tree for selecting the parameter group is on the left side of the window. Once you select the parameter group, the detailed information of the selected ones will appear on the right hand side. Then, select the parameter to be viewed and click on the read icon. The parameters values will be displayed in **Actual value** column.

Steps:

(1) Start the DRAS software.

(2) Make sure DRAS is connected with the controller. If not, please complete the communication setting in **Connection**.

(3) Click on **Parameter** to open the setting page.

(4) Select the parameter group from parameter tree.

(5) Select the parameters to be viewed.

(6) Click on the read icon. The parameters values will be displayed in **Actual value** column.

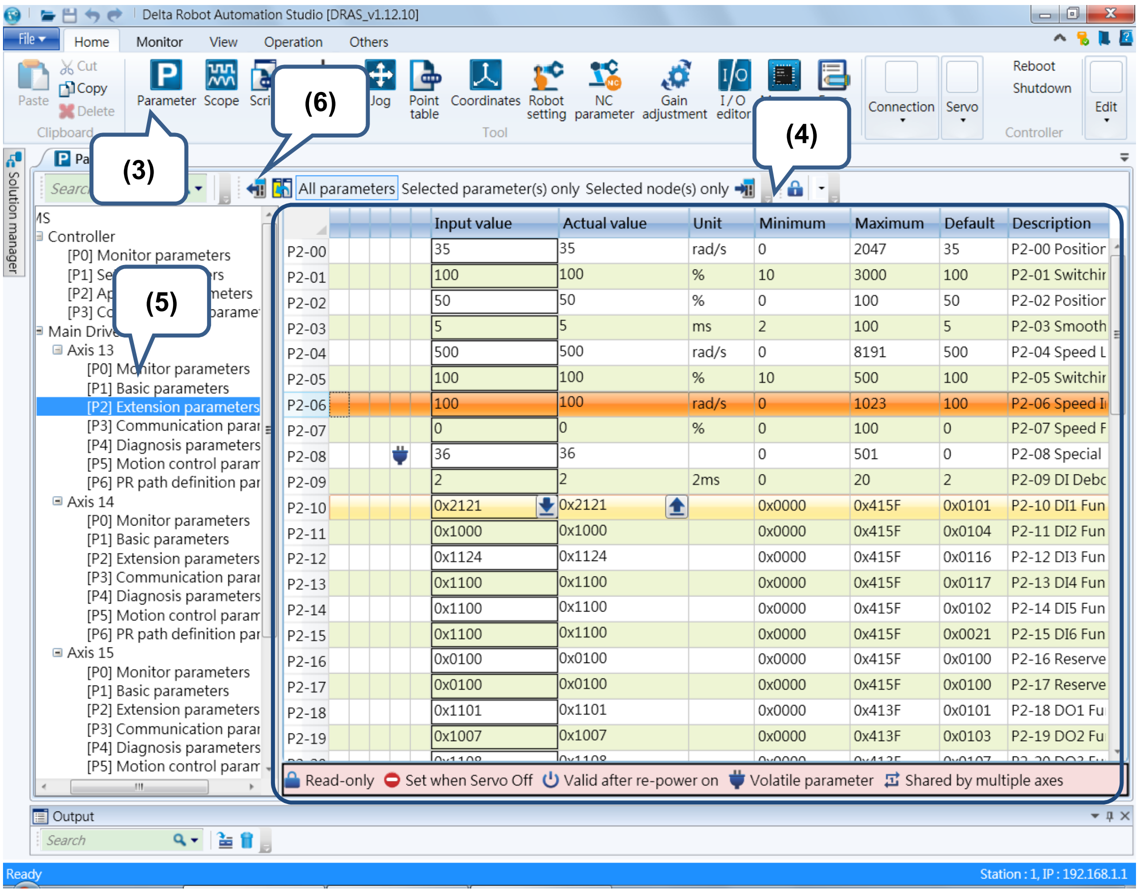


Figure 4.3.4.1 View relevant parameters

Set relevant parameters

The parameter editing section also enables users to modify the related parameters of the controller or servo drive. Start the DRAS and connect it with the controller. Then, click on **Parameter**. A window for editing parameters will pop up. In **Parameter** page, select the parameter group to be edited and edit them on the right hand side. Input the modified value into the corresponded column (see figure 4.3.3.1). Three ways are available for writing values into the controller, (1) click on the icon in **Input value** column (see figure 4.3.3.1), (2) input the value and directly pressing the **Enter** key and (3) click on the **Write** button. Set the value by one of the means and the DRAS will upload the selected parameters into the controller or servo drive.

Steps:

(1) Start DRAS.

(2) Make sure DRAS is connected with the controller. If not, please complete the communication setting in **Connection**.

(3) Click on **Parameter** to open the setting page.

(4) Select the parameter group from parameter tree.

(5) Input the value.

(6) Then, upload it to the controller or servo drive by (1) clicking on the **Download** button, (2) directly pressing the **Enter** key or (3) clicking on the **Write** button.

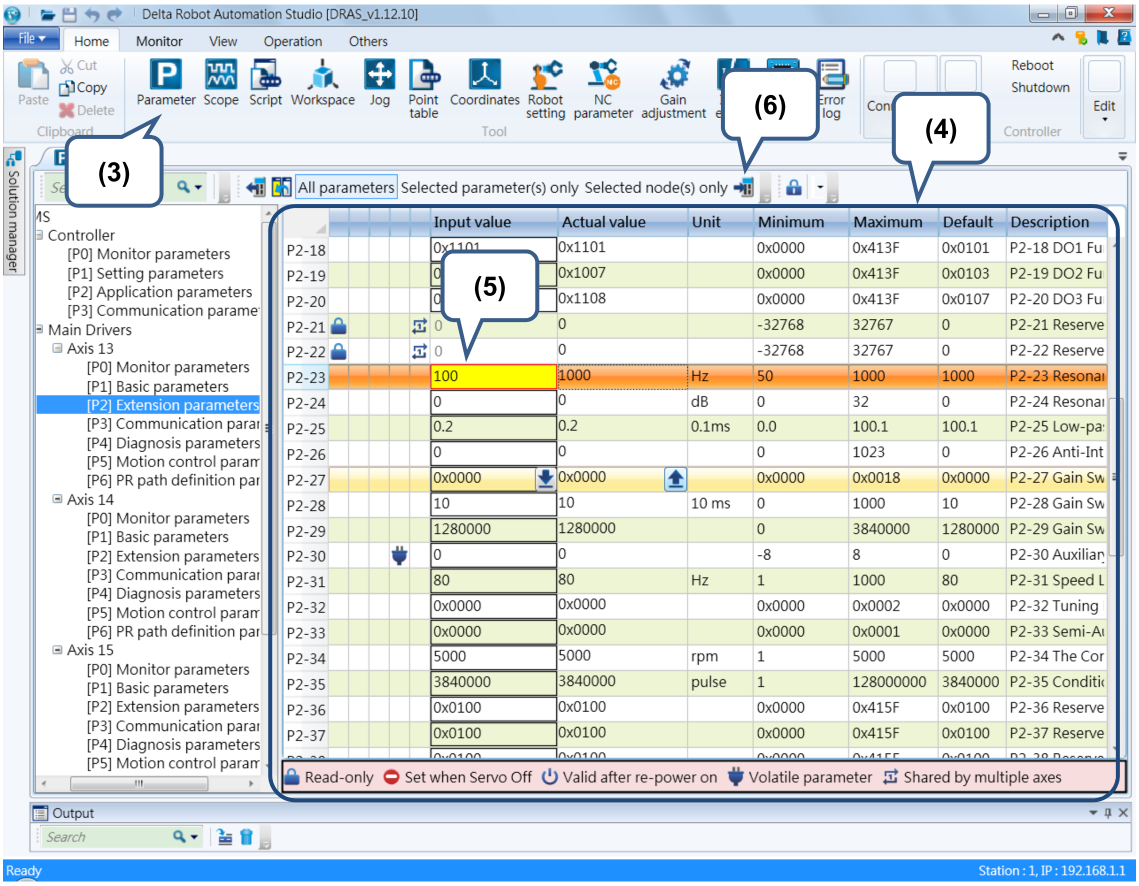


Figure 4.3.4.2 Set relevant parameters

Open/Save parameter file

Parameter's actual value can be saved as the parameter file in the parameter editing section. And the value in parameter file can be uploaded to the DRAS as well. Start the DRAS and connect it with the controller before saving the parameter file. Then, click on **Parameter**. A window for editing parameters will pop up. Select **All parameters** and click on the **Read** button. Parameters of the controller will be loaded into DRAS. Click on **File** > **Save as** and the parameter's actual value can be saved as the file. To open the parameter file, you need to click on **File** > **Open** and select the parameter file.

> WARNING
>
> **Warning: Use the right firmware version. Applying the wrong version might cause parameter file error.**

Steps:

(1) Start DRAS.

(2) Make sure DRAS is connected with the controller. If not, please complete the communication setting in **Connection**.

(3) Click on **Parameter** to open the setting page.

(4) Select **All parameters** on tool bar.

(5) Click the **Read** button and load the controller parameters into DRAS.

(6) Select **File** and click on **Save as**.

(7) And select **File** > **Open** to open parameter file.

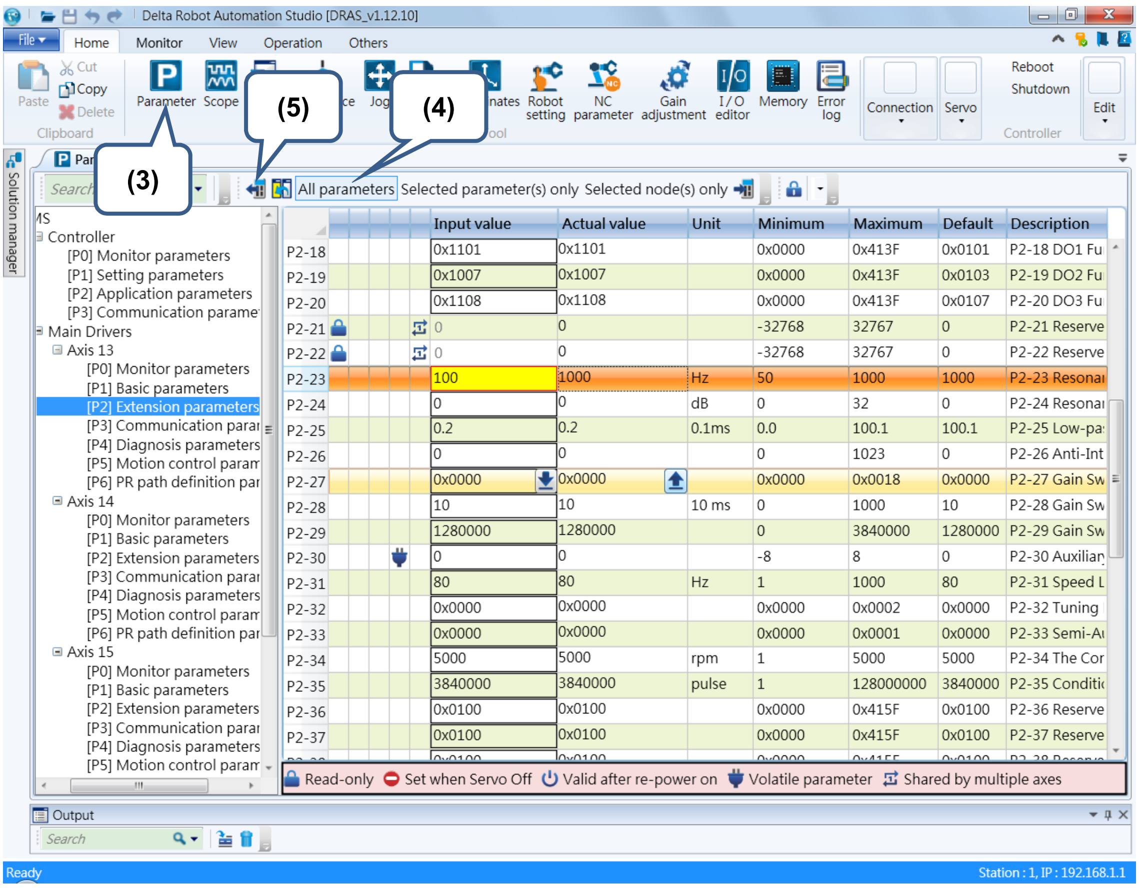


Figure 4.3.4.3 Save as parameter file

4

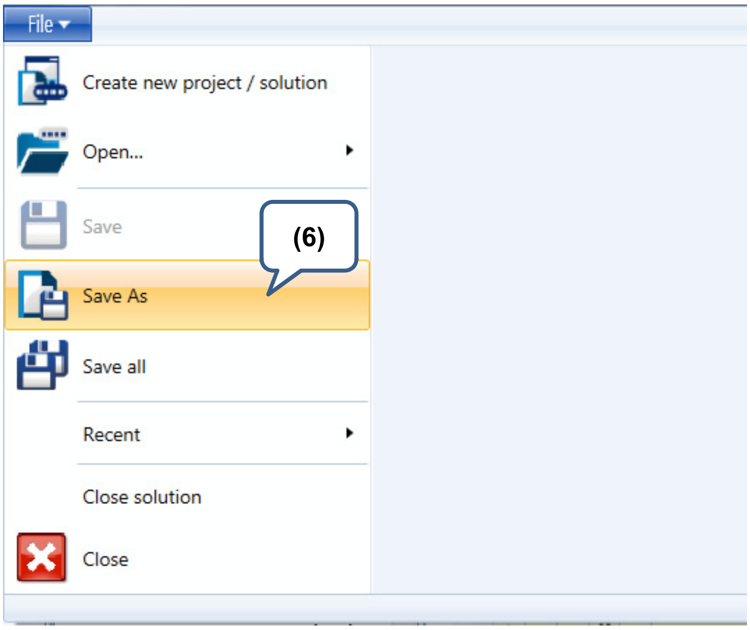


Figure 4.3.4.4 Save as parameter file

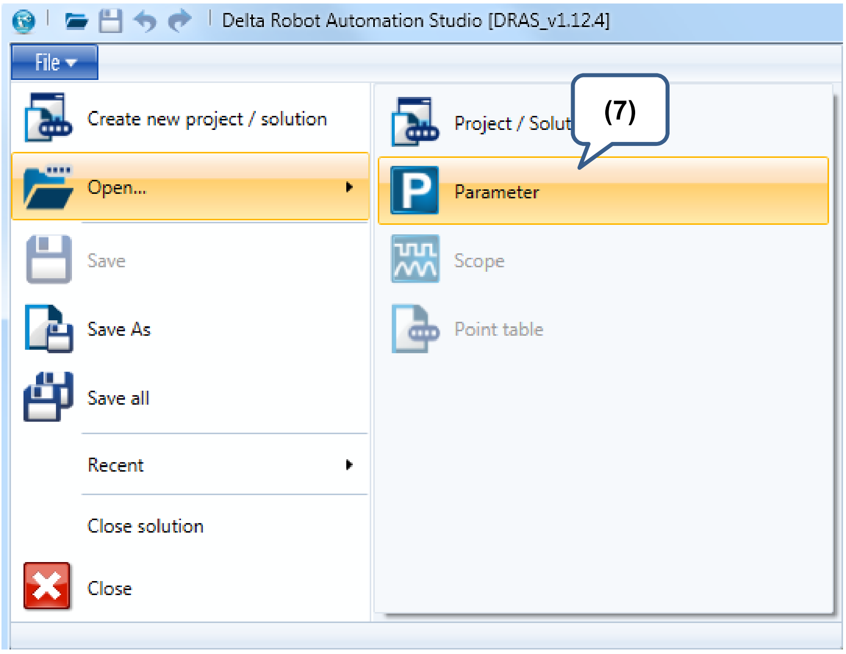


Figure 4.3.4.5 Open parameter file

Trial Operation and Tuning 5

This chapter illustrates how to do trial operation and the basic procedure of tuning. For your safety, please conduct the first inspection (without load) and then carry out further trial with load.

5

5.1 Apply power to MS controller

Please follow the instructions below.

(1) Make sure the wiring between the motor and MS controller is correct:

U, V, W and FG have to connect to cable red, white, black and green respectively. If the wiring is incorrect, the motor cannot work normally. Please refer to section 3.1 for wiring.

DANGER **Caution: Do not connect the power (R, S, T) to the output terminal (U, V, W) of MS controller. Or it might damage the controller.**

(2) Power on:

If it displays alarm ED013 when power is on:

This is the warning of emergency stop:

Please open DRAS and select **Home** > **I/O editor**. Check if any system DI (0 ~ 7) is set to emergency stop (EMGS). See figure 5.1.1.

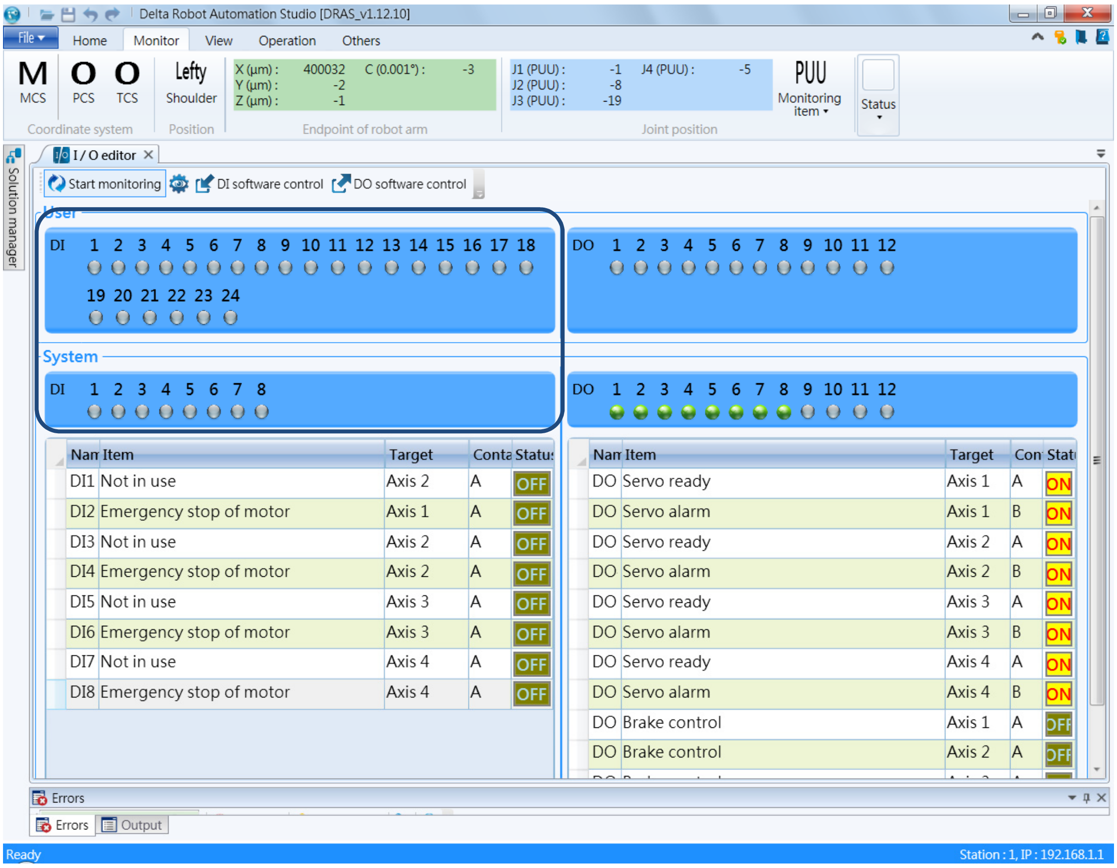


Figure 5.1.1 I/O editor

5

If EMGS signal is not required to set in DI, please go to **Home** > **Parameter** and set P2-11 to 1. See figure 5.1.2. Then, click on Alarm reset to clear the alarm. See figure 5.1.3.

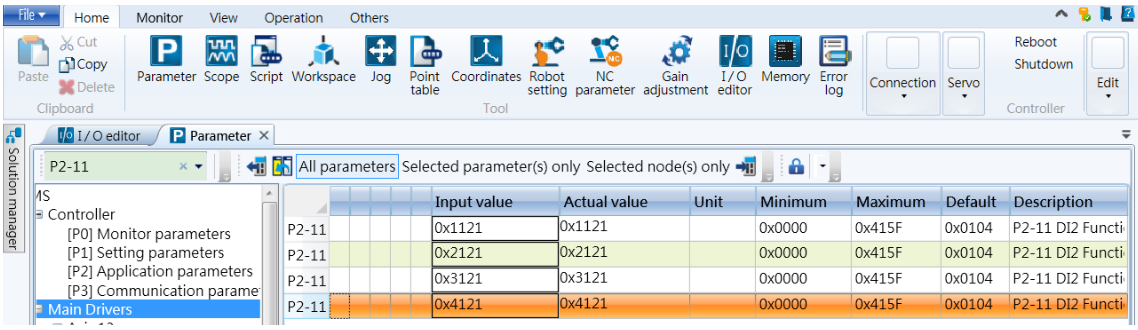


Figure 5.1.2 Setting the value of parameter P2-11

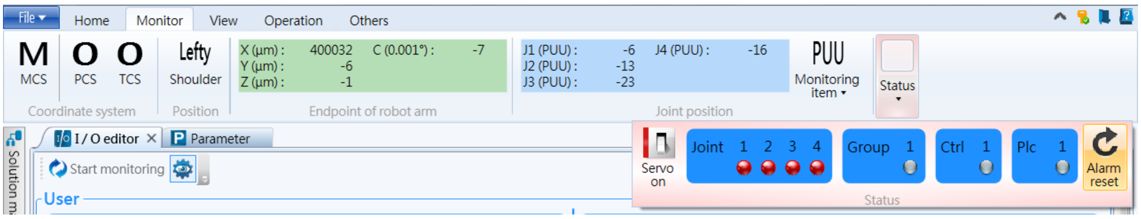


Figure 5.1.3 Alarm reset

5.2 Robot setting

Please complete the robot setting via DRAS before trial run. Select **Robot setting** and you will see parameters setting screen. See figure 5.2.1.

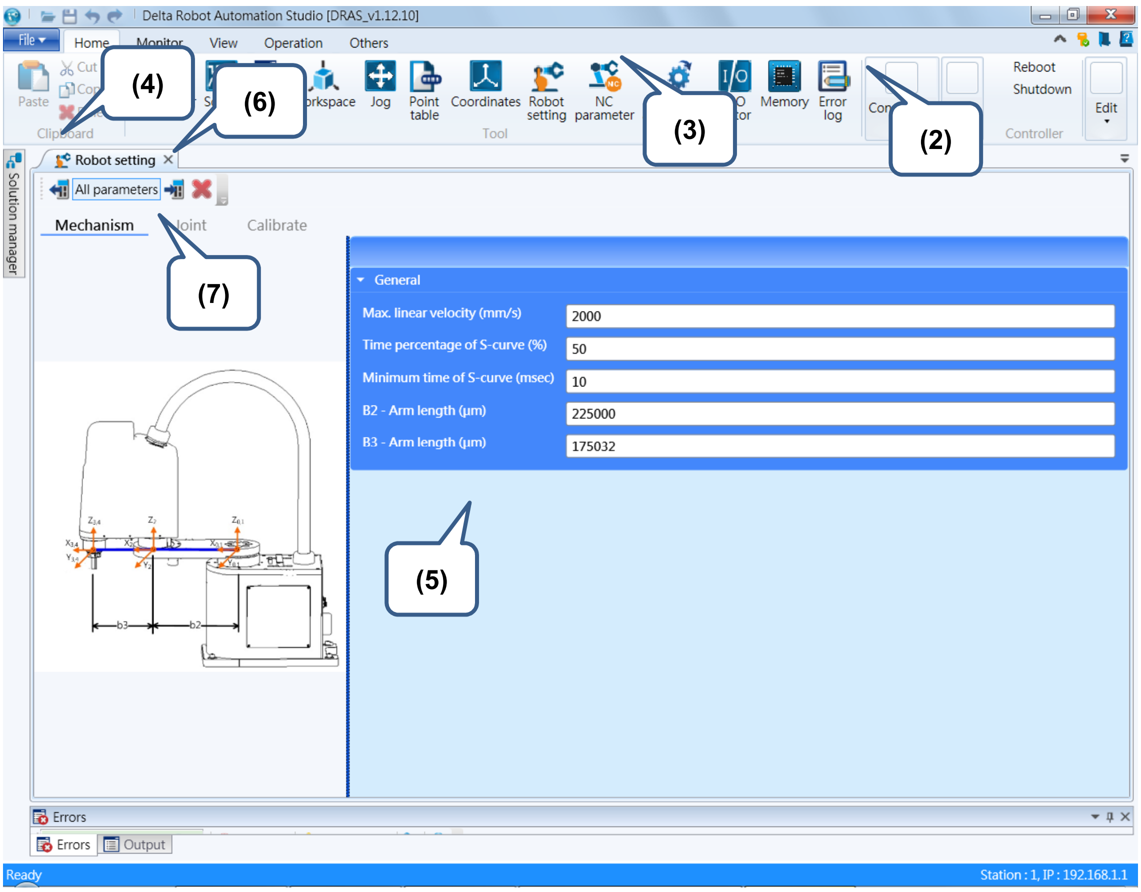


Figure 5.2.1 Steps of robot setting

Left hand side of Robot setting screen shows the robot diagram; robot related parameters are displayed on the right hand side. Click the **Read** icon on tool bar and the parameters will be updated from the current MS controller. Please fill in correct parameters according to the selected robot type. Then, click on the **Write** icon to update the controller data to the current parameters values.

If the parameter shows in red color, it means the value does not match to the value in the controller. It is suggested to write or read parameters first.

Please follow the steps below to complete robot setting:

(1) Start DRAS.

(2) Make sure DRAS is connected with MS controller. If it is not connected, please go to **Connection** to complete the setting.

(3) Select **Robot** setting tab.

(4) Click the **Read** icon on tool bar to update the data from MS controller.

(5) Adjust the parameters setting according to the robot type.

(6) Then, click on the **Write** icon to update the controller data to the current parameters values.

(7) Complete the setting in **Joint** setting page.

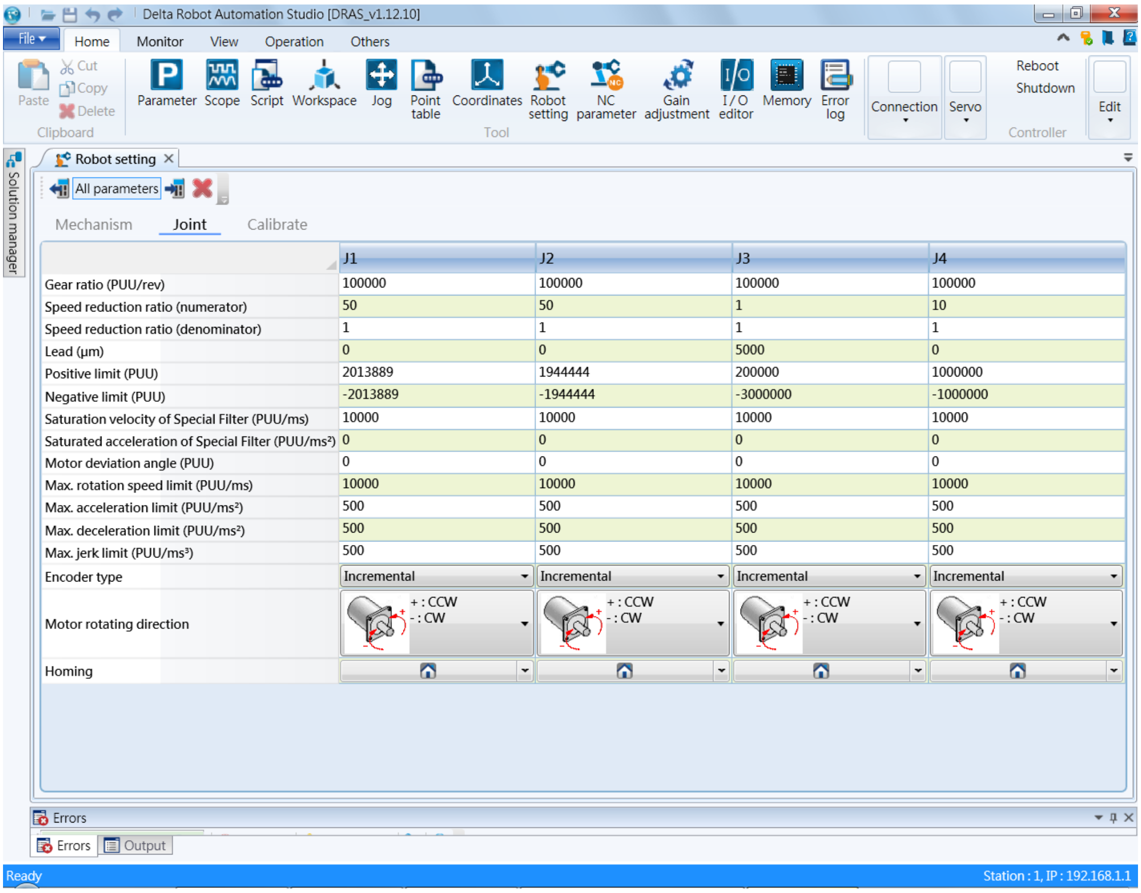


Figure 5.2.2 Joint setting

5.3 Jog trial run

WARNING

For your safety, make sure emergency stop and limit protection can function well before running with the servo motor.

Users can test the servo motor with jog trial run. It is recommended to set JOG at low speed for the first operation. Make sure the motor base is firmly fastened so as to avoid dangers caused by the counterforce during operation.

The Jog function can be carried out in **Jog** page of DRAS. Start DARS and connect it with MS controller. Click on the **Servo on** button to enable your MS controller. Then, select **Jog**. You can find Jog and Teach function in Jog tab. See figure 5.3.1. Select **ACS** in Jog mode. You will see the **Jog** button for controlling each axis. Each motor's status and robot position will be displayed on the right hand side.

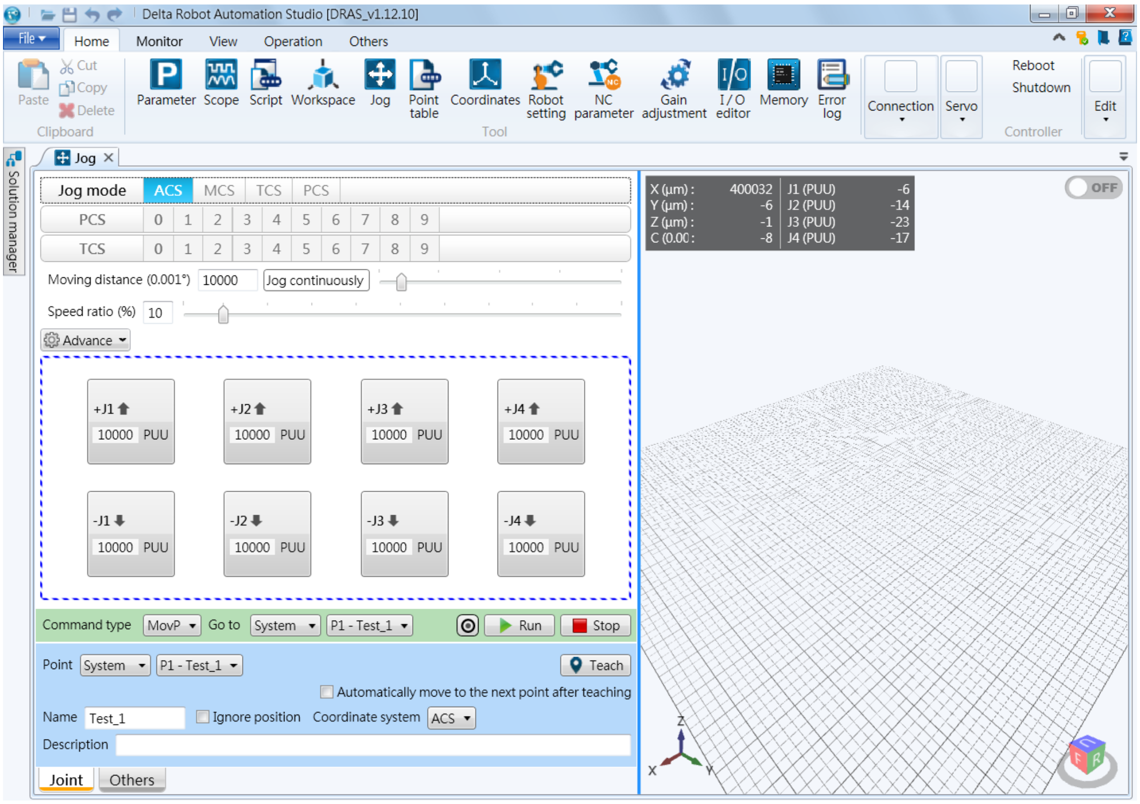


Figure 5.3.1 Jog function page

Before applying the Jog function, check the encoder type and complete the setting of P2-69 first. If the encoder is incremental type, set P2-69 to 0x0000 via Parameter setting page. And click the **Homing** button of the axis in **Advance**. This is for setting the current motor position as the origin. If the encoder is absolute type, you just need to set P2-69 to 0x0001. In Jog tab, you can adjust the moving distance, speed ratio, acceleration (ACC), deceleration (DEC) and JERK. In ACS mode, unit of the moving distance is PUU. Also, it is better to adjust to a lower speed ratio, such as 20% for the trial run. The unit of ACC and DEC is PUU/ms^2, we advise you to choose low velocity for acceleration and deceleration, e.g. 20 PUU/ms^2. Then, Jerk can be set according to the value of ACC and DEC. When the ratio of ACC/JERK or DEC/JERK is larger, the motor's running curve is smoother. Its unit is PUU/ms^2. Next, click on the **Jog** button to operate the motor. Pressing **+**, the motor runs in forward direction; on the other hand, while pressing **-**, the motor runs in reverse direction. Users can monitor motor state via Monitor section on the right. If the motor stops, please check the wiring between UVW cable and encoder. If the motor runs abnormally, check if the wiring sequence of UVW cable is correct.

See the steps below for Jog trial run without load:

WARNING **For your safety, make sure emergency stop and limit protection can function well before running with the servo motor.**

(1) Start DRAS.

(2) Make sure DRAS is connected with MS controller. If it is not connected, please go to **Connection** to complete the setting.

(3) Click **Servo on** button to enable your servo system.

(4) Then, click on **Jog**.

(5) Select **ACS** in Jog mode to control each axis.

(6) Before applying the Jog function, check the encoder type and complete the setting of P2-69 first. If the encoder is incremental type, set P2-69 to 0x0000 via Parameter setting page. And click the **Homing** button of the axis in **Advance**. This is for setting the current motor position as the origin. If the encoder is absolute type, you just need to set P2-69 to 0x0001.

(7) Adjust the moving distance, speed ratio, acceleration (ACC), deceleration (DEC) and JERK. Those values are not advised to set too high so as to avoid any potential danger.

(8) Then, operate the motor by pressing the Jog buttons.

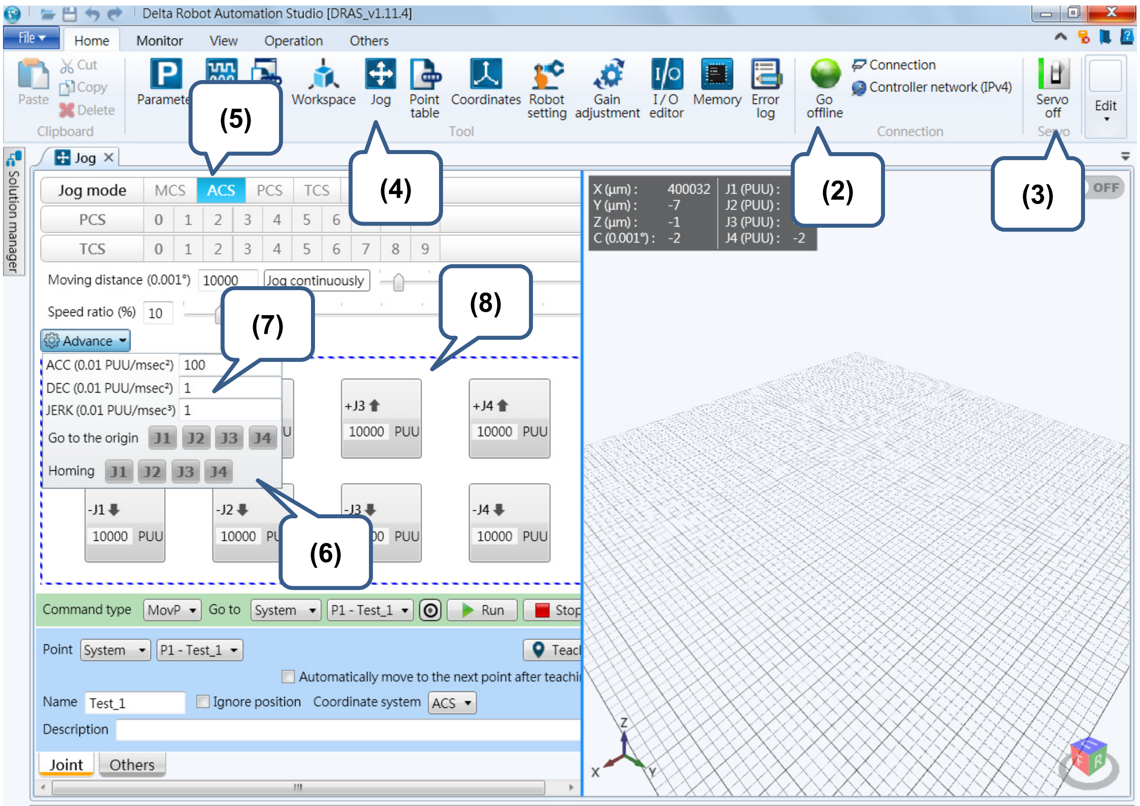


Figure 5.3.2 Steps of Jog trial run without load

5.4 Tuning procedure

For having a stable system and optimize the performance of MS controller, please correctly set the gain value first. Users can complete the gain adjustment more easily and efficiently via the software. The procedure is illustrated as follows.

5.4.1 Flow chart of tuning procedure

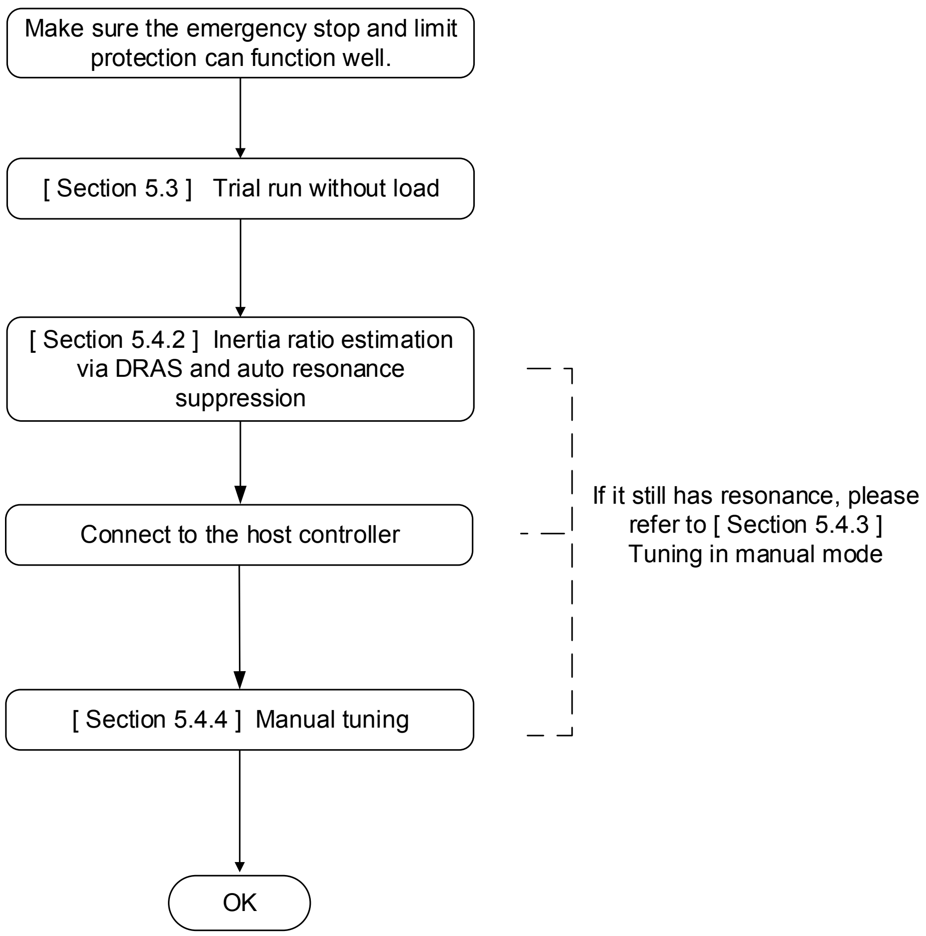


Figure 5.4.1.1 Tuning procedure

5

5.4.2 Gain adjustment via the software

Users can find **Gain adjustment** interface in DRAS. Start DRAS and select Gain adjustment. This function page has two modes, **Gain estimation** and **Gain Calculation**. See figure 5.4.2.1.

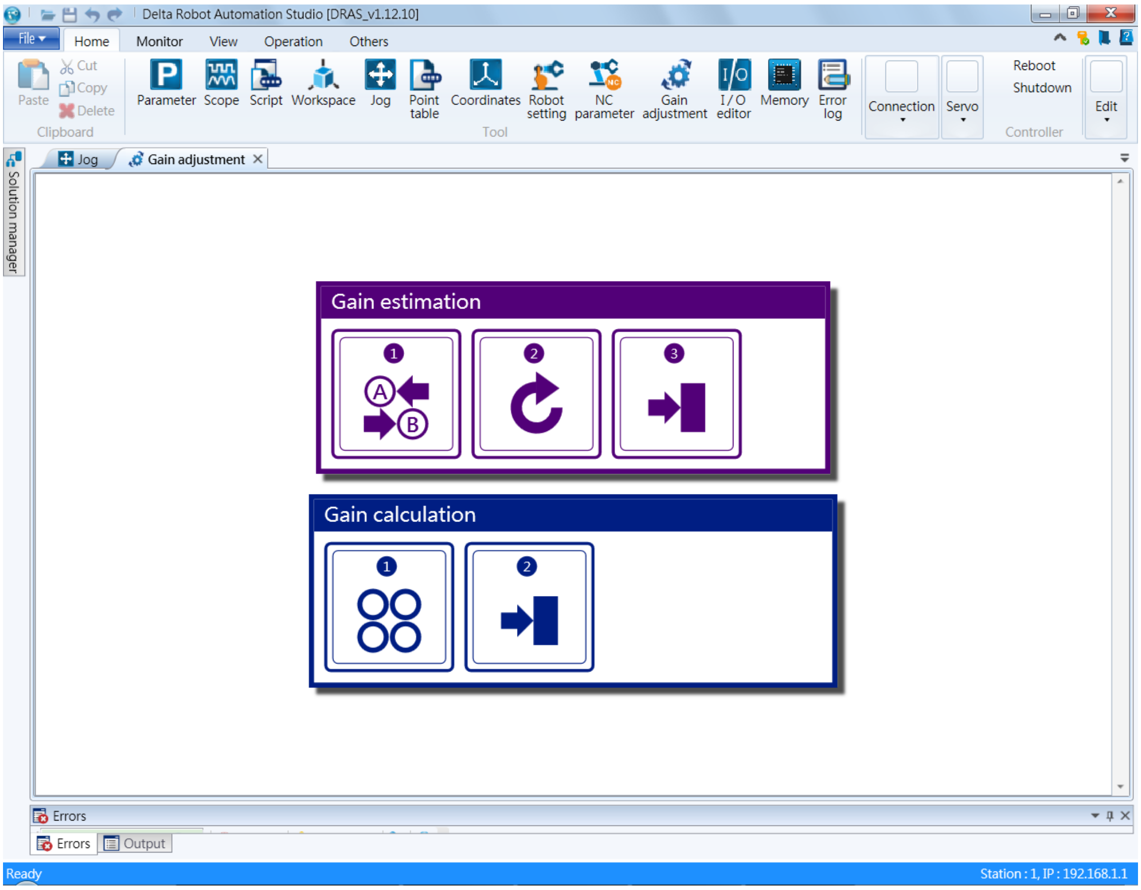


Figure 5.4.2.1 Gain adjustment function

These two modes can be applied when you have already "known the rigidity, bandwidth and inertia ratio" or "have not known the rigidity, bandwidth and inertia ratio", respectively.

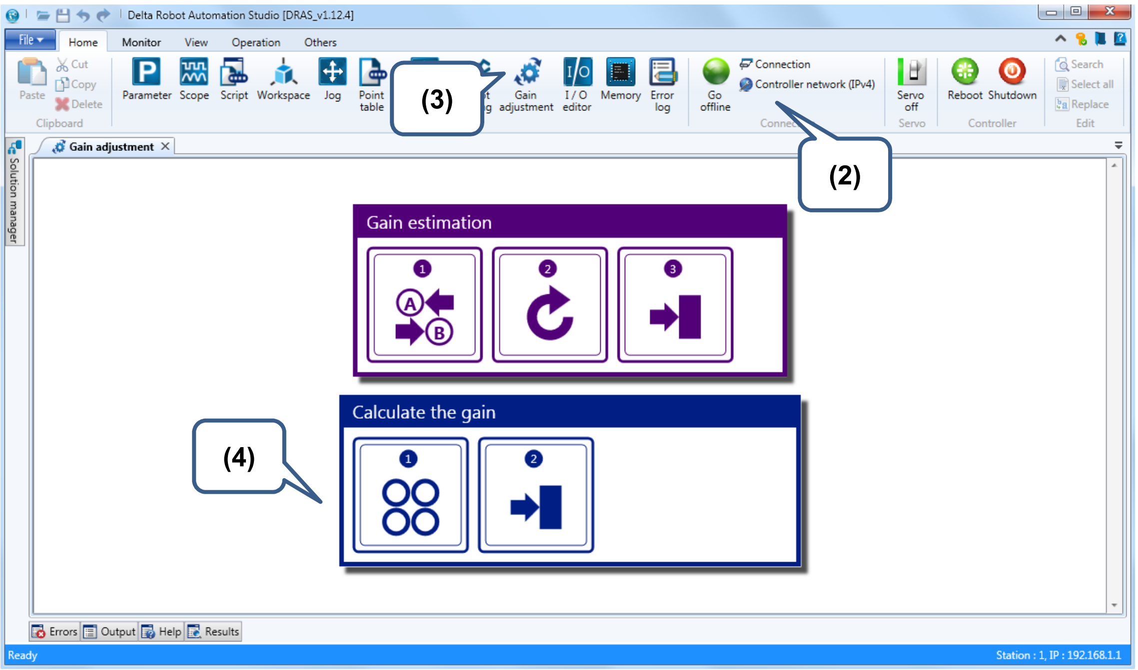
- Steps of gain calculation (knowing the rigidity, bandwidth and inertia ratio):

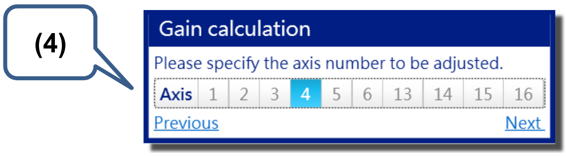
(1) Start DRAS.

(2) Make sure DRAS is connected with MS controller. If it is not connected, please go to **Connection** to complete the setting.

(3) Select **Gain adjustment** to open the setting page.

(4) Select **Calculate the gain** and specify the axis to be adjusted.



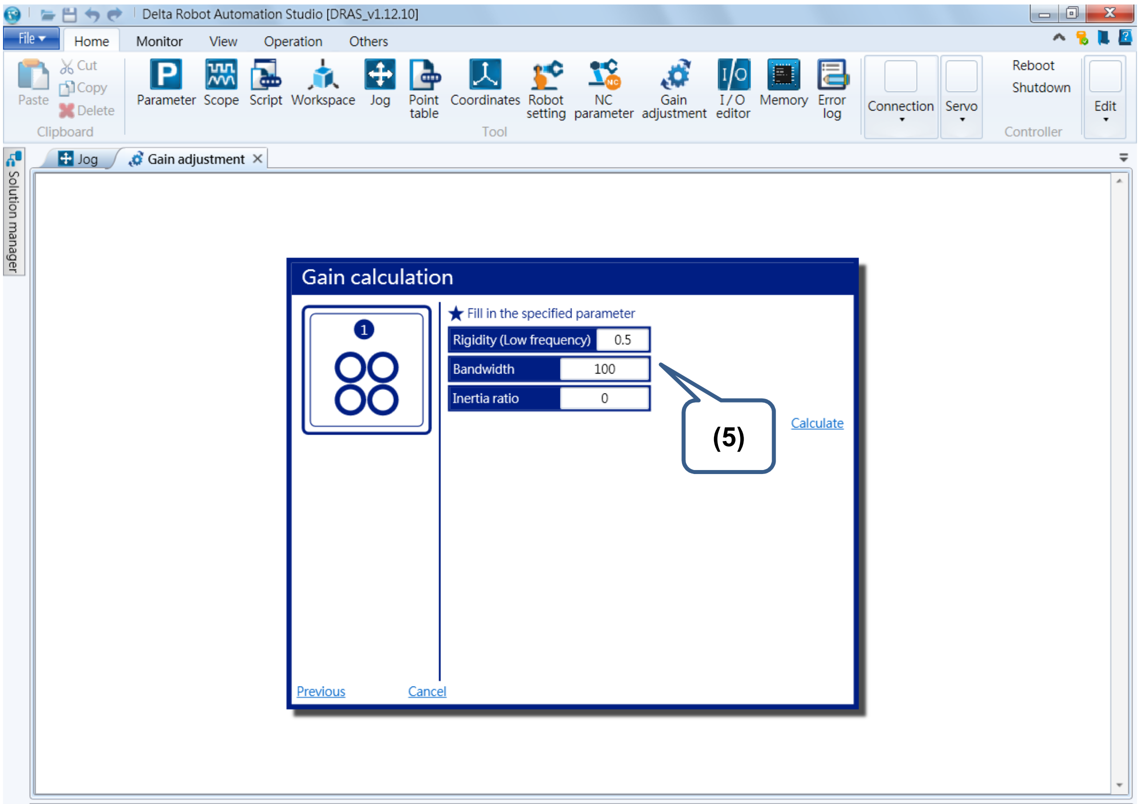


5

(5) Input the parameter values of rigidity (low frequency), bandwidth and inertia ratio.

(6) Click on **Calculate** and you can see the result on the screen.

(7) Then, click on **Write the result to the controller** to update the value of MS controller.



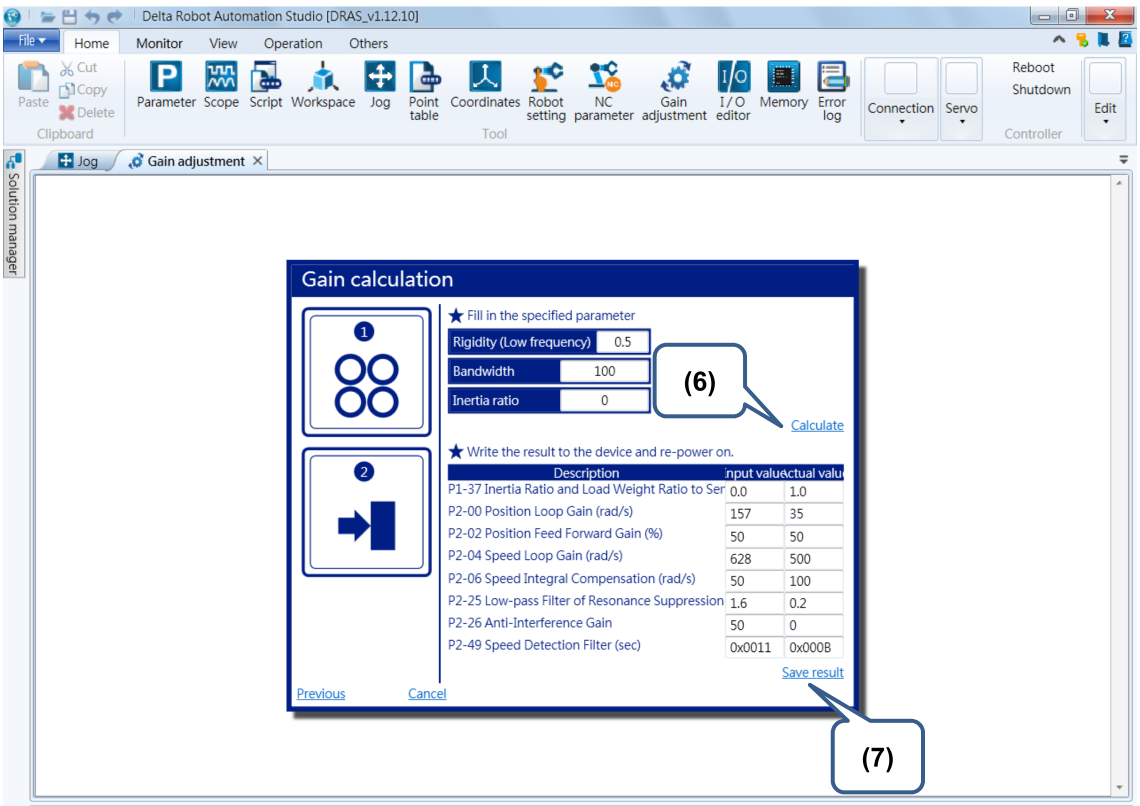


Figure 5.4.2.2 Steps to adjust the gain via the software (knowing the rigidity, bandwidth and inertia ratio).

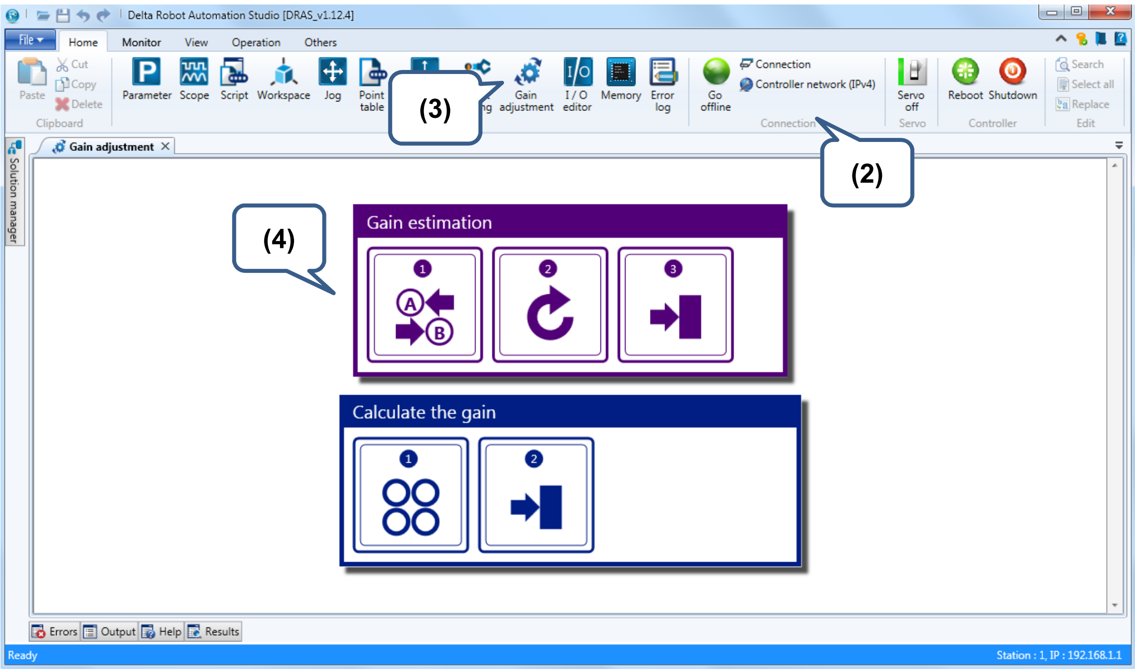
- Steps of gain estimation (with the unknown rigidity, bandwidth and inertia ratio):

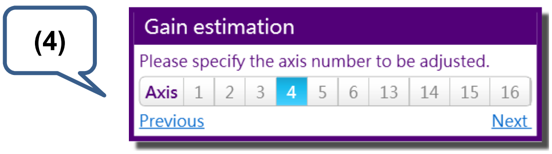
(1) Start DRAS.

(2) Make sure DRAS is connected with MS controller. If it is not connected, please go to **Connection** to complete the setting.

(3) Select **Gain adjustment** to open the setting page.

(4) Select **Gain estimation** and specify the axis to be adjusted.





(5) Position two points (A and B) of the motor via the Jog function.

(6) Input the correct motion parameters and time interval.

(7) Click on **Start** to estimate the inertia ratio.

(8) When the value of inertia ratio is stable (small variation), press the **Stop** button to stop operation.

(9) Then, click on **Calculate** to acquire the suggested values.

(10) Click on **Write the result to the controller** to complete gain estimation.

(11) When applying the Jog function, the motor will run back and forth until the system is stable and has no howling sound.

(12) If the system performance is not satisfactory, you can gradually increase the bandwidth and repeat step (10) to (12) until it meets the requirement. When it has howling sound during operation, it is suggested to multiply the current bandwidth by 0.8 as the frequency.

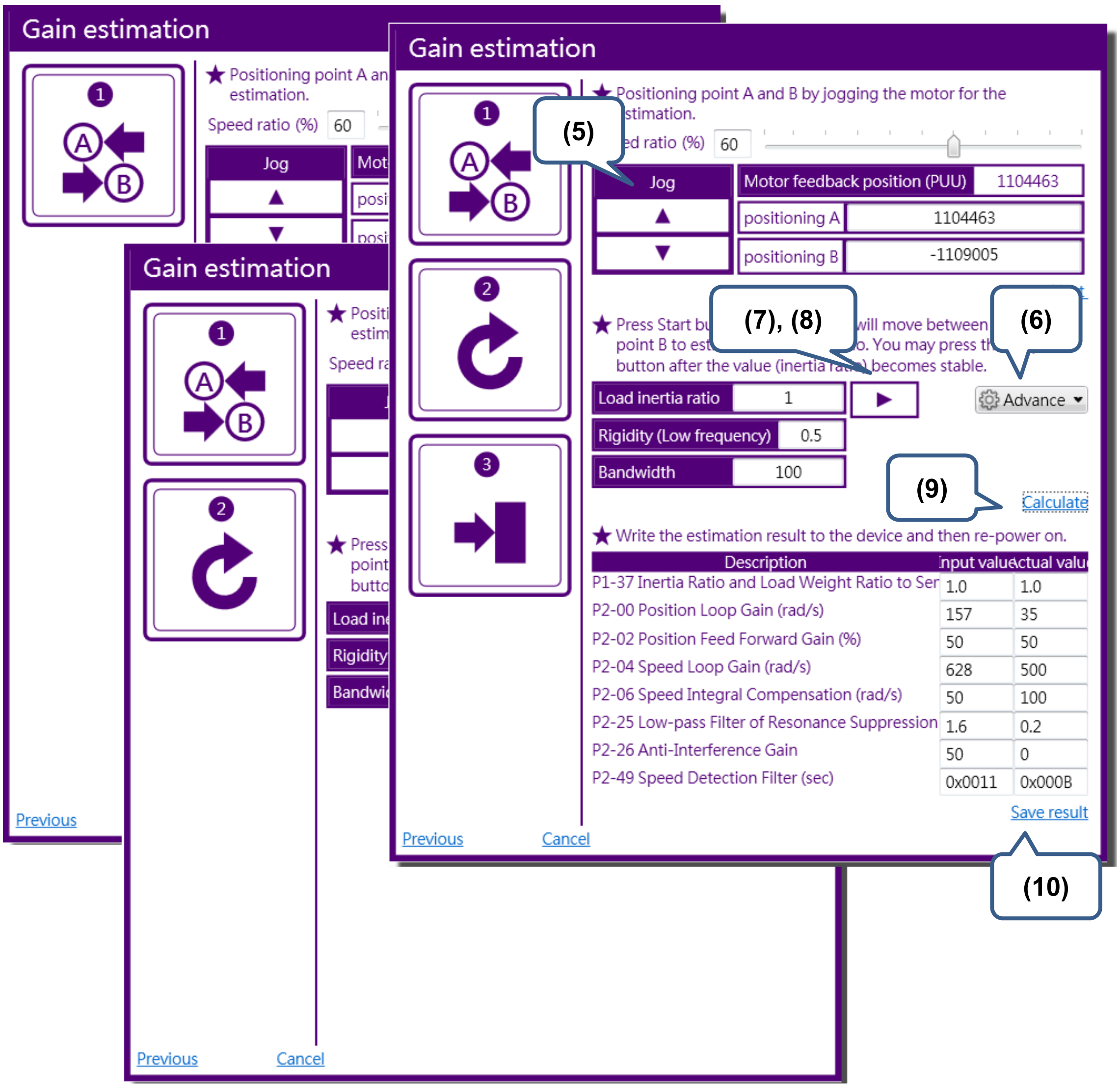


Figure 5.4.2.3 Steps to adjust the gain via the software (with the unknown rigidity, bandwidth and inertia ratio).

5

5.4.3 Mechanical resonance suppression

In default setting, the system is set as auto mode for mechanical resonance suppression. Please set P2-47 to 1.

Three groups of Notch filter are provided. Two of them can be set to auto resonance suppression. All of them can be set to manual adjustment (set P2-47 to 0). When P2-47 is set to 1, the system automatically detects the resonance frequency and the resonance setting will be set to two Notch filters. If P2-47 is set to 0, users need to input the resonance frequency and attenuation rate. See the setting steps below:

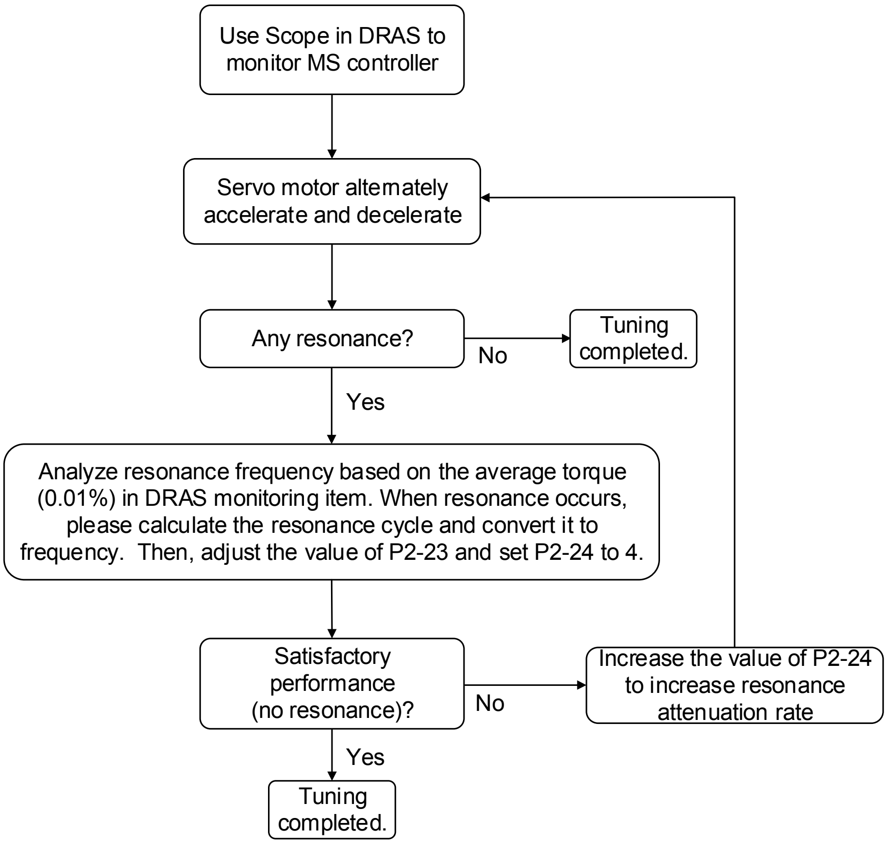


Figure 5.4.3.1 Mechanical resonance suppression

5

5.4.4 Tuning in manual mode

After connecting to the host controller, users have to adjust the response of the servo system. The followings are the related descriptions of gain adjustment.

- Position control gain (KPP, parameter P2-00)

  This parameter determines the response of position loop. The bigger KPP value will cause the higher response frequency of position loop. And it will bring smaller following error, smaller position error, and shorter settling time. However, if the value is set too big, the machinery will vibrate or causing overshoot when positioning. The calculation of position loop frequency response is as follows:

  Position loop frequency response $(\mathrm{Hz}) = \frac{\mathrm{KPP}}{2\pi}$

- Position feed forward gain (PFG, parameter P2-02)

  It can reduce the position error and shorten the settling time. However, if the value is set too big, it might cause overshoot. If the setting of e-gear ratio is bigger than 10, it might cause the noise as well.

- Speed control gain (KVP, parameter P2-04)

  This parameter determines the response of speed loop. The bigger KVP value will cause the higher response frequency of speed loop and better following. However, if the value is set too big, it would easily cause machinery resonance. The response frequency of speed loop must be 4 ~ 6 times higher than the response frequency of position loop. Otherwise, the machinery might vibrate or causing overshoot when positioning. The calculation of speed loop frequency response is as follows:

  Speed loop frequency response $\mathrm{fv} = \left(\frac{\mathrm{KVP}}{2\pi}\right) \times \left[\frac{\left(1+\frac{\mathrm{P1-37}}{10}\right)}{\left(1+\frac{\mathrm{JL}}{\mathrm{JM}}\right)}\right] \mathrm{Hz}$

  JM: Motor inertia; JL: Load inertia; P1-37: 0.1 times

  When P1-37 (estimation or setting) equals the real inertia ratio (JL/JM), the real speed loop frequency response will be: $fv = \frac{\mathrm{KVP}}{2\pi}\ \mathrm{Hz}$

- Speed integral compensation (KVO, parameter P2-06)

  The higher the KVI value is, the better capability of eliminating the deviation will be. However, if the value is set too big, it might easily cause the vibration of machinery. It is suggested to set the value as follows:

  $\mathrm{KVI\ (Parameter\ P2-06)} \leq 1.5 \times \mathrm{Speed\ Loop\ Frequency\ Response}$

- Low-pass filter of resonance suppression (NLP, parameter P2-25)

  The high value of inertia ratio will reduce the frequency response of speed loop. Therefore, the KVP value must be increased to maintain the response frequency. During the process of increasing KVP value, it might cause machinery resonance. Please use this parameter to eliminate the noise of resonance. The bigger the value is, the better the capability of improving high-frequency noise will be. However, if the value is set too big, it would cause the instability of speed loop and overshoot. It is suggested to set the value as the following:

  $\mathrm{NLP\ (Parameter\ P2-25)} \leq \frac{10000}{6 \times \mathrm{Speed\ Loop\ Frequency\ Response\ (Hz)}}$

- Anti-interference gain (DST, parameter P2-26)

  This parameter is used to strengthen the ability of resisting external force and gradually eliminate overshoot during acceleration / deceleration. Its default value is 0. It is suggested not to adjust the value in manual mode, unless it is for fine-tuning via the software.

  When manually calculate the gain value mentioned above, update the related data to the controller via Parameter setting page of DRAS to complete tuning. See figure 5.4.4.1.

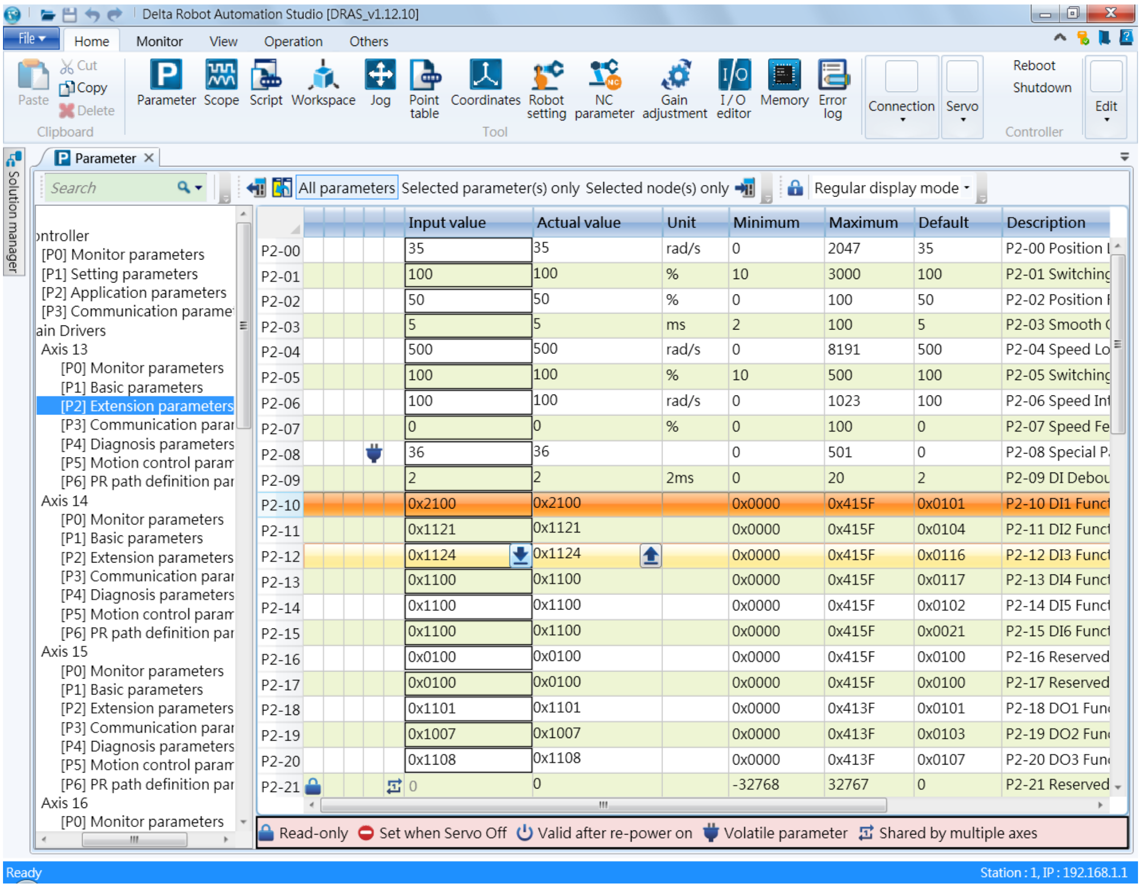


Figure 5.4.4.1 Parameter

Steps of tuning in manual mode:

(1) Manually calculate the parameters value of tuning.

(2) Start DRAS.

(3) Make sure DRAS is connected with MS controller. If it is not connected, please go to **Connection** to complete the setting.

(4) Click on **Parameter** to open the setting page.

(5) Select parameters to be edited.

(6) Input the value.

(7) Write parameters value into MS controller.

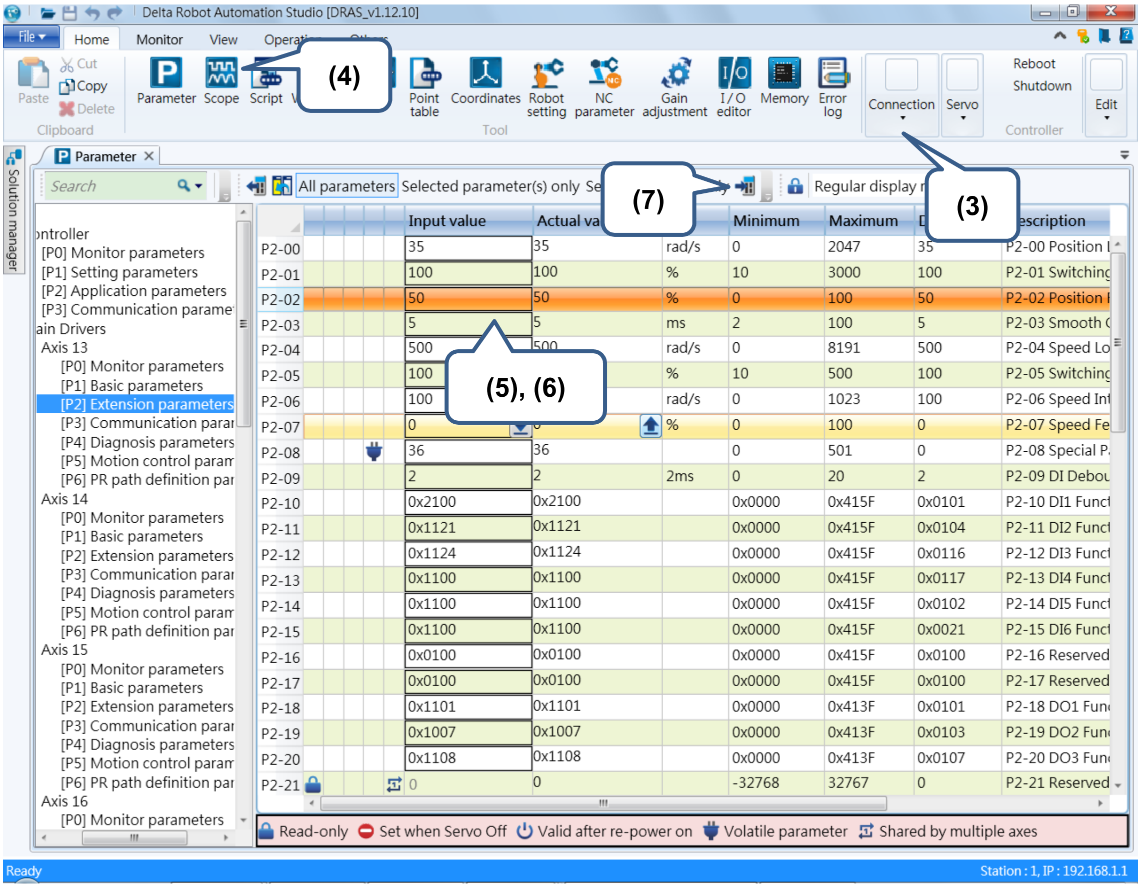


Figure 5.4.4.2 Steps of tuning in manual mode

5.4.5 Filter setting

When there is a fierce change in position, filter can be used to improve the operation. However, it might cause command delay. When it is applied to the application of multi-axis synchronous control, filters setting of each axis must be the same. This is for synchronizing the command of each axis after processing by the filter (Each axis has the same delay time).

Relevant parameters: Please refer to Chapter 8 for detailed description.

Parameter	Function
P1-08	Smooth constant of position command (Low-pass filter)
P1-68	Position command (Moving filter)

Delta Robot Language 6

DRL (DELTA Robot Language) is a lua-based robot language. It provides a robot-motion-related function library and helps users to do programming. In this chapter, you will find descriptions about instructions of DRL function library and their examples.

6

6.1 Robot settings

Through the **Robot settings** of DRAS, users can establish a complete robot motion procedure. Apart from functions of script, download, and operation, the **Robot settings** include functions such as debugging, syntax check, code completion and function completion. See Figure 6.1.1, the **Robot settings** section has two parts, (1) Tool bar and (2) Script. Attention: To run the script successfully, the script content has to be saved in file named after “main.lua”.

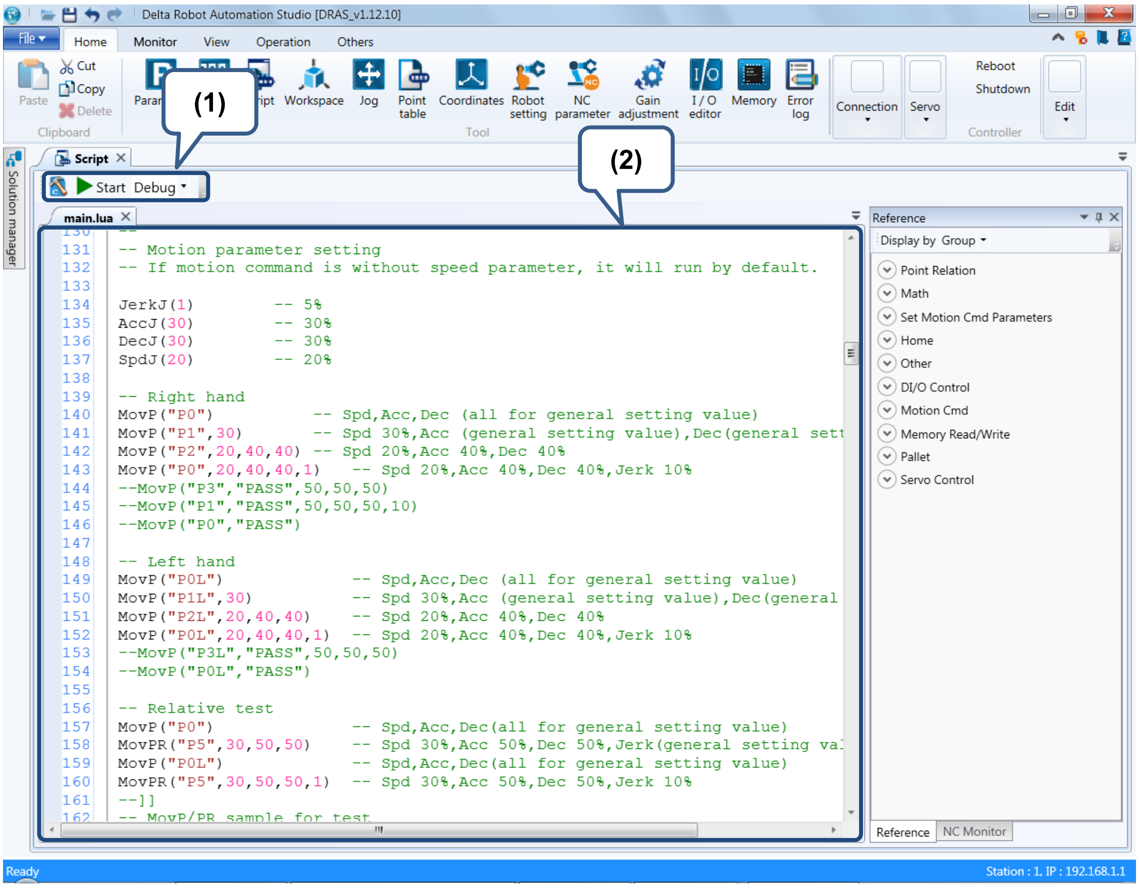


Figure 6.1.1 Robot settings

6.1.1 Tool bar of Robot settings

Each part of the tool bar is explained in the table 6.1.1.

Icon	Function	Keyboard shortcut	Description
	Syntax check	F6	Check the syntax of the current StartUp project.
	Start	F5	Start the current StartUp project.
	Stop	-	Stop running the program.
	Continue	F5	Debug function - Continue
	Pause	-	Debug function - Pause
	Step over	F10	Debug function - Step over
	Step into	F11	Debug function - Step into
	Step out	Shift+F11	Debug function - Step out

Table 6.1.1.1 Tool bar function of the robot setting

6

6.1.2 Script

The **Script** includes four parts, breakpoint, line number, folding, and script editing. See Figure 6.1.2.1. The script editing section is for editing the script. And its code is written by a Lua-based language, DELTA Robot Language (DRL). As the syntax of DRL and Lua is identical, users can write the script with syntax of Lua as well as using the provided motion functions for further programming. When finishing editing the script, you can use the tool bar functions to run the script.

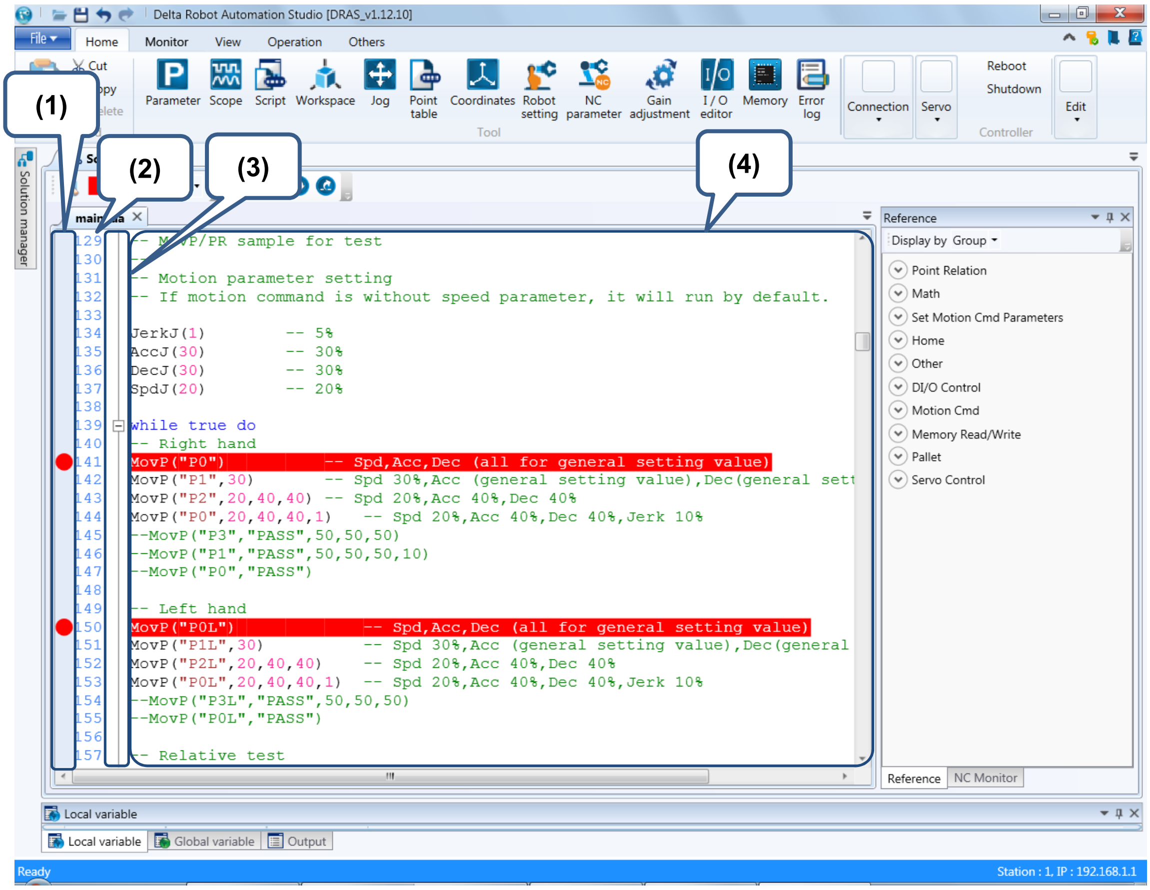


Figure 6.1.2.1 Script editing section

(1) Breakpoint switch; (2) Line number; (3) folding; (4) Program editing section

6.2 Reserved Keywords

The reserved keywords cannot be used for naming the variables in the program. The keywords are: **and**, **break**, **do**, **else**, **elseif**, **end**, **false**, **for**, **function**, **if**, **in**, **local**, **nil**, **not**, **or**, **repeat**, **return**, **then**, **true**, **until**, and **while.**

6.3 Point (P) expression

P[k].< expression>

- Point is expressed as **P[k]**. **k** stands for the point number.
- MS controller will copy the point data (taught by users) in the non-volatile memory to point **P** before running the Lua script. Modifying **P** point data in the Lua script will not change the data stored in non-volatile memory. The purpose of this design is to prevent the non-volatile point data from being modified accidentally. To edit the point data in non-volatile memory, please use function **SetPointToMem().**
- **P[**point name**]** will return the point number.
- Data type of **P** is listed as follows:

Name	Length	Description
No.	-	Point number
Name	16 Bytes	Name the point. This name can be used in the program to replace the point number.
x	FLOAT32	Spatial coordinates X. Unit: µm
y	FLOAT32	Spatial coordinates Y. Unit: µm
z	FLOAT32	Spatial coordinates Z. Unit: µm
a	FLOAT32	Spatial coordinates A. Unit: 0.001°
b	FLOAT32	Spatial coordinates B. Unit: 0.001°
c	FLOAT32	Spatial coordinatesC. Unit: 0.001°
Shoulder[1]	BOOL	Posture of the robot shoulder: Right(0) / Left(1)
Elbow[2]	BOOL	Posture of the robot elbow: Up(0) / Down(1); Hand: Right(0) / Left(1)
Flip	BOOL	Flip of robot wrist: No(0) /Yes(1)
PS	BOOL	Robot posture setting: Posture determined by controller (0) / Posture defined by point setting (1)
UF	UINT16	Product coordinate system (PCS)
TF	UINT16	Tool coordinate system (TCS)
Coord[3]	UINT16	Coordinate system of the point: MCS(0); PCS(1); TCS(2); ACS(3)

6

Note:
1. **Elbow** can be changed by variables. **HAND_RIGHT** stands for right hand; **HAND_LEFT** stands for left hand.
2. **Coord** can be changed by variables, which are _MCS, _PCS, _TCS, and _ACS.

Example:

```
h1=-1200.0                  -- Set variable h1 to -1200.0.
P[1].x = 1000.0             --Set spatial coordinates X of P[1] to 1000.0 μm.
P[1].y = 2000.0             --Set spatial coordinates Y of P[1] to 2000.0 μm.
P[1].z = h1                 --Set spatial coordinates Z of P[1] to h1.
MovP(1)                     --Move to P[1] by PTP motion command.
P2 = P["P2"]                –Read the point number named P2.
P[P2].x = 3000.0            --Set spatial coordinates X of P[P2] to 3000.0μm
P[P2].Elbow = HAND_RIGHT --Set P[P2] to right-hand posture.
P[P2].Coord = _ACS          --Set coordinate system of P[P2] to ACS.
```

P

"Pn"| n

- Point can be expressed in two ways in the program. One is expressed by names with quotations marks " " , the other is expressed by point number.
- n: Point number

Example:

```
A[1]=LOCX("P1")             -- Pass the x-coordinate of point named P1 to A[1]
MovP(2)                     --Move to P[2] by PTP motion command.
MovL("P3")                  --Move to the point named P3 by linear motion command
```

LOCx

LOCx(P, Value)

- Read or input point data. This function modifies point data **P** but not the data in non-volatile memory. It only changes the point data **P** in the current script written in Lua.
- **X**: Includes spatial coordinates X, Y, Z, A, B, and C, as well as F (left/right hand posture)
- **P**: Target point.
- **Value**: The input value.
- Unit of spatial coordinates X, Y, and Z is μm.
- Unit of spatial coordinates A, B, and C is 0.001°.

Example:

```
A[1]=LOCX("P1")             --Pass the x-coordinate of point named P1 to A[1]
LOCZ(2,100)                 --Set spatial coordinate z of P[2] to 100 μm.
LOCF("P3",1)                -- Set point named P3 to right-hand posture.
```

P.new

P.new (x, y, z, a, b, c, Elbow, Shoulder, Flip, PS, UF, TF, Coord)

- Create a new point. It will be stored in the current Lua script instead of the non-volatile memory.
- Return: Point data array.
- `x`: Spatial coordinate x in the unit of μm
- `y`: Spatial coordinate y in the unit of μm
- `z`: Spatial coordinate z in the unit of μm
- `a`: Spatial coordinate a in the unit of 0.001°
- `b`: Spatial coordinate b in the unit of 0.001°
- `c`: Spatial coordinate c in the unit of 0.001°
- `Elbow`: Posture of robot elbow:

  0: Up; 1: Down
- `Shoulder`: Posture of robot shoulder:

  0: Right; 1: Left
- `Flip`: Flip of robot wrist:

  0: No; 1: Yes
- `PS`: Robot posture setting:

  Disable: 0; Enable: 1
- `UF`: Number of PCS
- `TF`: Number of TCS
- `Coord`: Coordinate system of the point include four types:

  (1)_MCS; (2)_PCS; (3)_TCS; (4)_ACS.

Example:

```
PNew = P.new(300010, 201000, -5300, 0, 0, 0, 1, 1, 0, 1, 0, 0, _MCS)
--Create a point variable named PNew. In its content, x = 300010, y = 201000, and z = -5300,
a = 0, b = 0, c = 0, Elbow = Down, Shoulder, Shoulder = Left, and Flip = None, PS = 1, UF = 0, TF = 0,
and Coord = _MCS.

MovP(PNew)

--Move to point PNew in PTP movement.
```

6

P. SetTable

P.SetTable (nPoint)

- Set the point format. Please ensure its format is correct. If no value is input in the point data, the value will automatically be set to 0.
- Return: point data array.
- **nPoint**: The point to be set. If it is not set, the default is { x = 0, y = 0, z = 0, c = 0, Elbow = HAND_RIGHT, PS = 0, UF = 0, TF = 0, Coord = _MCS }

Example:

```
PNew = {x = 300000, y = 150000, z = 0 }
--Create a new point variable named PNew. In its content, x = 300000, y = 150000, and z = 0.
PNew2 = P.SetTable(PNew)
--Set PNew and transmit the point data array to PNew2.
MovP(PNew2)
--Move to PNew2 in PTP movement.
```

P + P

P + P

- Point compensation. If it is point data array, the **x**, **y**, **z**, and **c** can be used for compensation with positive or negative value. The value of **Elbow**, **PS**, **UF**, **TF**, and **Coord** is dermined by the 1st parameter following the equation; if it is not a point data array, **Elbow**, **PS**, **UF**, **TF**, and **Coord** become 0.
- The compensation will not change the value of the original point; the value after compensation will be specified as a new point variable.
- The usage of point compensation is elaborated as follows:
  1. **P + P**: Addition or deduction of the two points.
  2. **P.X(Value)**: x value for the compensation
  3. **P.Y(Value)**: y value for the compensation
  4. **P.Z(Value)**: z value for the compensation
  5. **P.C(Value)**: c value for the compensation
- It will be stored in the Lua script but not the non-volatile memory.
- Return: Point data array

Example:

```
NewP1 = P[P["Pmeta1"]] + P.X(2000) + P.Z(-3000)
-- The point named Pmeta1's x-coordinates plus 2000 and Z-coordinate plus -3000.
Elbow, PS, UF, TF, and Coord are the original point data of "Pmeta1" and will be stored in variable
NewP1.
MovP(NewP1)             --Move to NewP1 by PTP motion command.

NewP2 = P[P["P2"]] + P[P["SHIFT1"]] - P[P["SHIFT2"]]
--Add value x, y, z, and c of point named P2, SHIFT1, and SHIFT2 together.
Elbow, PS, UF, TF, and Coord are the original point data of P2 and stored in variable NewP2.
```

```
MovP(NewP2)            --Move to point NewP2 by PTP motion command.

NewP3 = NewP1 + NewP2

--Add value x, y, z, and c of NewP1 and NewP2 together.

Elbow, PS, UF, TF, and Coord are the original point data of NewP1 and are stored in variable NewP3.

MovP(NewP3)            --Move to poin NewP3 by PTP motion command.
```


SetPointToMem

SetPointToMem(Point, PointIdx, PointName)

- Write the points to the non-volatile memory in MS controller (PLC zone). If the point number already contains point data, this original data will be replaced by the new ones.
- **Point**: The point to be written in. You can input the point name, point number, or data array.
- **PointIdx**: Set the point number, which range is 1 ~ 1024.
- **PointName**: Set the point name.

Example:

```
NewP1 = P[P["Pmeta1"]] + P.X(2000) + P.Z(-3000)
--The point named Pmeta1'sxX-cooridnate plus 2000 and Z-coordinate plus -3000.
Elbow, PS, UF, TF, and Coord are the original point data of Pmeta1 and are stored in variable NewP1.

SetPointToMem(NewP1, 100, "newP1")
--Write the point data NewP1 to the non-volatile memory of MS controller (PLC zone), which point
number is 100 and is named after newP1.

MovP(100)              --Move to P[100] by PTP motion command.

SetPointToMem(100, 101, "newP2")
--Copy point P[100] to the non-volatile memory of MS, and its point number is 101 and named newP2.

MovP(101)              --Move to P[101] by PTP motion command.

SetPointToMem("newP2", 102, "newP3")
--Copy the point named newP2 to the non-volatile memory of MS controller, and its point number is 102
and named newP3.

MovP(102)              --Move to P[102] by PTP motion command.
```

6

CopyPoint

CopyPoint(Point)

- Copy the point. The return value is data array.
- **`Point`**: The point to be written in. You can input the point name, point number, or data array.

Example:

```
CopyP = CopyPoint("P1") --Copy the data of the point named "P1" to variable CopyP.
NewP = CopyP + P.X(10000) + P.Y(20000)  --The point NewP's X-coordinate plus 10000 and
                                          Y-coordinate plus 20000.

MovP(NewP)             --Move to point NewP by PTP motion command.
```

6.4 Instruction descriptions

The instructions can be divided into categories of flow control, motion parameters, motion control, I/O operation, servo, read/write of the memory, pallet, and time. The usage and examples will be elaborated in this section.

Instruction list

Flow of Control	
Instruction	Example
If…then…else…end	if … then … [elseif … then …] [else …] end
While	while …do…end
For	for …do…end
Repeat	repeat…until…
Goto	goto <label>
function	function(…) … end
DELAY	DELAY(t)

Motion Parameters	
Instruction	Example
AccJ	AccJ(Value)
DecJ	DecJ(Value)
SpdJ	SpdJ(Value)
JerkJ	JerkJ(Value)
AccL	AccL(Value)
DecL	DecL(Value)
SpdL	SpdL(Value)
JerkL	JerkL (Value)
MaxSpdL	MaxSpdL (Value)
MaxAccL	MaxAccL (Value)
MaxSpdJ	MaxSpdJ (Value)
MaxAccJ	MaxAccJ (Value)
SetPassMode	SetPassMode(PassMode)
SetPassDistance	SetPassDistance(Value)
SetPassTime	SetPassTime(Value)
SetWaitCmdMode	SetWaitCmdMode(WaitCmdMode)
SetMArchPPS	SetMArchPPS(IsSetMArchPPS)

Motion Control	
Instruction	Example
MovP	MovP(Point, BMode, Spd, Acc, Dec, Jerk)
MovL	MovL(Point, BMode, Spd, Acc, Dec, Jerk)
MovPR	MovPR(Point, BMode, Spd, Acc, Dec, Jerk)
MovLR	MovPR(Point, BMode, Spd, Acc, Dec, Jerk)
MArchP	MArchP(Point, h1, h2, h3, Spd, Acc, Dec, Jerk)
MArchL	MArchL(Point, h1, h2, h3, Spd, Acc, Dec, Jerk)
MArchPT	MArchPT(Point, h1, h2, h3, Spd, Acc, Dec, Jerk)
MArchLT	MArchLT(Point, h1, Spd, Acc, Dec, Jerk)
MovJ	MovJ(Axis_idx, Point, BMode, Spd, Acc, Dec, Jerk)
MovCIRC	MovCIRC(ECirc, PCirc, ArcMode, BMode, Spd, Acc, Dec, Jerk)
MovL_EX	MovL_EX(Point, DO_Count, <dis_percent, expression>, BMode, Spd, Acc, Dec, Jerk)
MovCIRC_DIR	MovCIRC_DIR(ECirc, PCirc, ArcMode, OriChoiceMode, OriControlMode, BMode, Spd, Acc, Dec, Jerk)
MovCIRC_EX	MovCIRC_EX(BMode, Spd, Acc, Dec, Jerk)
StopAxis	StopAxis(Axis_idx, BMode, Dec, Jerk)
StopGroup	StopGroup(BMode, Dec, Jerk)

DI/O Operation	
Instruction	Example
DI	DI(di_idx)
DO	DO(do_idx, Switch)
User_DI	User_DI(di_idx)
User_DO	User_DO(do_idx, Switch)
Sys_DI	Sys_DI(di_idx)
Sys_DO	Sys_DO(do_idx)
Remote_DI	Remote_DI(method, station_idx, di_idx)
Remote_DO	Remote_DO(method, station_idx, do_idx, Switch)
User_DIs	User_DI(nDIGrpIdx)
User_DOs	User_DO(nDOGrpIdx, nDOGrpValue)
Sys_DIs	Sys_DI(nDIGrpIdx)
Sys_DOs	Sys_DO(nDOGrpIdx)
Remote_DIs	Remote_DI(method, station_idx, nDIGrpIdx)
Remote_DOs	Remote_DO(method, station_idx, nDOGrpIdx, nDOGrpValue)
WaitDIO	WaitDIO(expression, delayTime)

Servo	
Instruction	Example
ServoOn	ServoOn(ax_idx)
ServoOff	ServoOff(ax_idx)
ServoOnGroup	ServoOnGroup( )
ServoOffGroup	ServoOffGroup( )

Read and Write of the Memory	
Instruction	Example
ModbusRead16	ModbusRead16(Adress)
ModbusRead32	ModbusRead32(Adress)
ModbusWrite16	ModbusWrite16(Adress, Value)
ModbusWrite32	ModbusWrite32(Adress, Value)
PLCMB3Read16	PLCMB3Read16(Adress)
PLCMB3Read32	PLCMB3Read32(Adress)
PLCMB3Write16	PLCMB3Write16(Adress, Value)
PLCMB3Write32	PLCMB3Write32(Adress, Value)

Pallet	
Instruction	Example
PalletDef	PalletDef(Pallet_idx, x_idx, y_idx, z_idx, PPoint1, PPoint2, PPoint3, PPoint4, PPoint5)
PalletLength	PalletLength(Pallet_idx)
PalletP	PalletP(Pallet_idx, P_idx) PalletP(Pallet_idx, x_idx, y_idx, z_idx)

Time	
Instruction	Example
timerInit	timerInit()
timerPass	timerPass(tTime)

6

6.4.1 Flow of control

Instruction: if…then…else… end

if … then … [elseif … then …] [else …] end

- Syntax of the **if** statement is as follows:

```
if specified condition then
    statement
end
```

- If the specified condition is true, the statement is executed; if false, the statement will not be executed.

Example:

```
if DI(1) == 1 then
--If DI(1) is ON, then move to the point named P1 by MovP instruction.
   MovP("P1")
elseif DI(2) == 0 then
--If DI(2) is OFF, then move to the point named P2 by MovL instruction.
   MovL("P2")
else   --In else condition, move to the third point by MovP instruction.
   MovP(3)
end
```

Instruction: while

while …do…end

- Syntax of the **while** loop is as follows:

```
while condition do
    statement
end
```

- Repeatedly execute the body of the **while** loop as long as the specified condition is true. If the given condition is false, then the loop ends.

Example:

```
a = {5,4,3,2,1}

i = 1

sum = 0

while a[i] do
--Test if the value is set in a[i]. If the condition is true, execute the statement that follows; if the condition is false, the loop ends.

    sum = sum + a[i]

    i = i + 1

end
```

Instruction: for

for …do…end

- Syntax of the **for** loop is as follows:

  ```
  for var=exp1,exp2,exp3 do
      statement
  end
  ```

- The **for** loop can be applied when you already knew the loop executing time. At the first time the loop is being executed, the variable will be set to the initial value. It will test if the variable is in the range between the initial and end value. If the condition is true, execute the statement; if false, the **for** loop ends. Each time the **for** loop ends, the variable value increases or decreases. Again, it tests if this variable is within the range between the initial and end value. When the increment/decrement is not set, the value will be set to 1 automatically.

Example:

```
a = {5,4,3,2,1}

i = 1

sum = 0

for i=1,5 do
--The value of variable i equals 1. When the increment/decrement is not set, the value will be set to 1 automatically. The condition is set within [1,5].

    sum = sum + a[i]

    i = i + 1

end
```

6

Instruction: repeat

repeat…until…

- Syntax of the **repeat** loop is as follows:

```
repeat
     statement
until specified condition
```

- If the specified condition is true, the **repeat** loop ends; if the condition is false, continue to execute the statement.

Example:

```
a = {5,4,3,2,1}
i = 1
sum = 0
repeat
   sum = sum + a[i]
   i = i + 1
until i > #a      --#a: The array length of "a". Continue to execute the repeat loop and the loop
                  ends when I > #a.
```

Instruction: goto

goto <label>

- Jump to the line that is labeled and then execute the program that follows the labeled line.
- Prefix "::" and suffix "::" to label the string.
- The **goto** label does not require "::"

Example:

```
a = {5,4,3,2,1}
i = 1
sum = 0
::START::               --Prefix :: and suffix :: to label the string.
sum = sum + a[i]
i = i + 1
if i < 6 then
   goto START           --the goto label does not require "::"
end
```

Instruction: function

function(…) … end

- Functions can be defined by users. The syntax is as follows.

```
Function function name (input variable 1, input variable 2,...)
    statement
end
```

Example:

```
function MyFunction()
    MovP("P2")      --Move to the point named P2 by MovP instruction.
    MovP("P3")      --Move to the point named P3 by MovP instruction.
end

MovP("P1")
MyFunction();           --Execute the instructions in MyFunction.
```

Instruction: DELAY

DELAY(t)

- Delay time
- **t**: Delay time (unit: sec)
- The minimum delay time is 0.000001 seconds.

Example:

```
DELAY(0.5)        --A delay of 0.5 sec.
```

6

6.4.2 Instructions of motion parameters

Instructions of motion parameters can be used to set the speed, acceleration, deceleration, and jerk for programming. The setting values will be recorded once the setting is complete. If no speed, acceleration, deceleration or jerk is set, the program will automatically refer and use the previous settings.

Instruction: AccJ

AccJ(Value)

- Set the acceleration of PTP motion.
- **`Value`**: The acceleration value, which unit is %.

Example:

```
AccJ(50)        --Set the acceleration of PTP motion to 50%.
```

Instruction: DecJ

DecJ(Value)

- Set the deceleration during PTP motion.
- **`Value`**: The deceleration value, which unit is %.

Example:

```
DecJ(50)    --Set the deceleration of PTP motion to 50%.
```

Instruction: SpdJ

SpdJ(Value)

- Set the max. speed of PTP motion.
- **`Value`**: The max. speed, which unit is %.

Example:

```
SpdJ(50)        --Set the max. speed of PTP motion to 50%.
```

Instruction: JerkJ

JerkJ(Value)

- Set the jerk of PTP motion.
- **Value**: the jerk, which unit is %.
- Adjusting Acc/Jerk ratio can change the smoothness of the motion path. The greater the ratio is, the smoother the path will be.

Example:

```
JerkJ(50)   --Set the jerk of PTP motion to 50%.
```

Instruction: AccL

AccL(Value)

- Set the acceleration of linear motion.
- Switch the input mode according to **MovL_MODE**. The default is **REAL_SPEED** mode.
  1. **REAL_SPEED**: Input the actual speed.
  2. **PERCENT_SPEED**: Input the value in percentage.
- **Value**: Acceleration
  1. **REAL_SPEED**: The actual acceleration, which unit is mm/sec^2.
  2. **PERCENT_SPEED**: Value in percentage.

Example:

```
-- REAL_SPEED mode
MovL_MODE = REAL_SPEED
AccL(5000)   --Set the actual acceleration of linear motion to 5000 mm/sec2.
-- PERCENT_SPEED mode
MovL_MODE = PERCENT_SPEED
MaxAccL(5000000)          --Set the max. acceleration of linear motion to 5000000 mm/sec2.
AccL(0.1)    --Set the acceleration of linear motion to 0.1% and acutal acceleration to 5000 mm/sec2.
```

6

Instruction: DecL

DecL(Value)

- Set the deceleration of the linear motion.
- Switch the input mode according to **MovL_MODE**. The default is **REAL_SPEED** mode.
  1. **REAL_SPEED**: Input the actual deceleration value.
  2. **PERCENT_SPEED**: Input the value in percentage.
- **Value**: deceleration
  1. **REAL_SPEED**: The actual deceleration value, which unit is mm/sec^2.
  2. **PERCENT_SPEED**: Value in percentage.

Example:

```
-- REAL_SPEED mode
MovL_MODE = REAL_SPEED
DecL(5000)            --Set the actual deceleration of linear motion to 5000 mm/sec².
-- PERCENT_SPEED mode
MovL_MODE = PERCENT_SPEED
MaxAccL(5000000)      --Set the max. acceleration of linear motion to 5000000 mm/sec²
DecL(0.1)  --Set the deceleration of linear motion to 0.1% and actual deceleration to 5000 mm/sec² .
```

Instruction: SpdL

SpdL(Value)

- Set the max. speed of linear motion.
- Switch the input mode according to **MovL_MODE**. The default is **REAL_SPEED** mode.
  1. **REAL_SPEED**: Input the actual max. speed.
  2. **PERCENT_SPEED**: Input the value in percentage.
- Value: max. speed
  1. **REAL_SPEED**: The actual max. speed, which unit is mm/sec.
  2. **PERCENT_SPEED**: Value in percentage.

Example:

```
-- REAL_SPEED mode
MovL_MODE = REAL_SPEED
SpdL(200)             --Set the actual max. speed of linear motion to 200 mm/sec.
-- PERCENT_SPEED mode
MovL_MODE = PERCENT_SPEED
MaxSpdL(2000)         -- Set the max. speed of linear motion to 2000 mm/sec.
SpdL(10)       -- Set the max. speed of linear motion to10% and actual max. speed to 200 mm/sec.
```

Instruction: JerkL

JerkL(Value)

- Set the jerk of linear motion.
- Adjusting the Acc/Jerk ratio can change the smoothness of the motion path. The greater the ratio is, the smoother the motion path will be.
- **`Value`**: The actual jerk, which unit is mm/sec^3.
- Switch the input mode according to **`MovL_MODE`**. The default is **`REAL_SPEED`** mode.
  1. **`REAL_SPEED`**: Input the actual jerk.
  2. **`PERCENT_SPEED`**: Input the value in percentage.
- Value:
  1. **`REAL_SPEED`**: The actual jerk, which unit is mm/sec^3.
  2. **`PERCENT_SPEED`**: Value in percentage.

Example:

```
-- REAL_SPEED mode
MovL_MODE = REAL_SPEED
JerkL(5000000)          --Set the actual jerk of linear motion to 5000000 mm/sec^3.
-- PERCENT_SPEED mode
MovL_MODE = PERCENT_SPEED
MaxAccL(5000000)     --Set the max. acceleration/deceleration of linear motion to 5000000
                       mm/sec^2.
                     --Set the max. jerk of linear motion to 5000000000 mm/sec^3
JerkL(0.1)  --Set the jerk of linear motion to 0.1% and actual jerk to 5000000 mm/sec^3.
```

Instruction: MaxSpdL

MaxSpdL(Value)

- Set the max. speed limit of linear motion.
- **`Value`**: The max. speed setting value, which unit is mm/sec.

Example:

```
MaxSpdL(2000)        --Set the max. speed limit of linear motion to 2000 mm/sec.
```

6

Instruction: MaxAccL

MaxAccL(Value)

- Set the max. acceleration, deceleration, and jerk of linear motion.
- **Value**: The max. acceleration. When this value is set first, the deceleration and jerk will be set automatically.
  1. The max. acceleration: The unit is mm/sec^2.
  2. The max. deceleration: The unit is mm/sec^2 and the value is the same as the max. acceleration.
  3. The max. jerk: The unit is mm/sec3 and the value is 1000 times of the max. acceleration.

Example:

```
MaxAccL(100)
--Set the max acceleration limit of linear motion to 100 mm/sec^2
--Set the max. deceleration of linear motion to 100 mm/sec^2, which value is the same as the max. acceleration.
-- Set the max. jerk of linear motion to 100000 mm/sec3, which value is 1000 times of the max. acceleration.
```

Instruction: MaxSpdJ

MaxSpdJ(Value)

- Set the max. speed limit of PTP motion.
- **Value**: The max. speed limit, which unit is PUU/msec.

Example:

```
MaxSpdJ(8500)        --Set the max. speed limit of linear motion to 8500 PUU/msec.
```

Instruction: MaxAccJ

MaxAccJ(Value)

- Set the max. acceleration, deceleration, and jerk of PTP motion.
- **Value**: the max. acceleration. When this value is set first, the deceleration and jerk will be set automatically.
  1. The max. acceleration: The unit is PUU/msec^2.
  2. The max. deceleration: The unit is PUU/msec^2 and the value is the same as the max. deceleration.
  3. The max. jerk: The unit is PUU/msec^3 and the value is the same as the max. acceleration.

Example:

```
MaxAccJ(100)
--Set the max. speed limit of PTP motion to 100 PUU/msec2.
--Set the max. deceleration limit of PTP motion to 100 PUU/msec2,which value is the same as the max. acceleration.
--Set the max. jerk of PTP motion to 100 PUU/msec3, which is the same as the max. acceleration.
```

Instruction: SetPassMode

SetPassMode(PassMode)

- Set the mode for instructions overlap.
- **PassMode**: Two modes are available, distance overlap and time overlap.
- To apply the distance overlap mode, input **TM_DIS_PASS**.
- To apply the time overlap mode, input **TM_TIME_PASS**.

Example:

```
MovP("SQURE1",30,50,50,5)
SetPassMode( TM_DIS_PASS )      --Set to distance overlap mode.
SetPassDistance(10)             --Set the interrupt distance to 10 mm.
MovL_MODE = PERCENT_SPEED
MaxSpdL(2000)                   --The max. speed limit of linear motion is 2000 mm/sec.
MaxAccL(100000)                 -- The max. deceleration of linear motion is 100000 mm/sec2
MovL("SQURE2","PASS",30,30,30,3)
MovL("SQURE3","PASS",30,30,30,3)
MovL("SQURE4","PASS",30,30,30,3)
MovL("SQURE1",30,30,30,3)

SetPassMode( TM_TIME_PASS )     --Set to time overlap mode.
SetPassTime( 100 )              --Set the interrupt time to 100%.
MovL("SQURE2","PASS",30,30,30,3)
MovL("SQURE3","PASS",30,30,30,3)
MovL("SQURE4","PASS",30,30,30,3)
MovL("SQURE1",30,30,30,3)
```

6

Instruction: SetPassDistance

SetPassDistance(Value)

- Set the distance for instructions overlap.
- **Value**: The overlap distance in the unit of mm.

Example:

```
SetPassMode( TM_DIS_PASS )      --Set to distance overlap mode.
SetPassDistance(20)       -- Set the interrupt distance to 20 mm.
MovL("P4", "BLENDSTART",80,40,40,40)
MovL("P3", "PASS",80,40,40,40)
--Distance overlap mode will be applied for connecting the two paths. The instructions start overlapping at 20 mm away from P4.
```

Instruction: SetPassTime

SetPassTime(Value)

- Set the time percentage for instructions overlap. Setting to 100% means the whole deceleration zone of the current instruction will overlap the next instruction.
- **Value**: The overlap time in the unit of percentage.

Example:

```
MovL_MODE = PERCENT_SPEED
MaxSpdL(2000)              --Set the max. speed limit of linear motion to 2000 mm/sec.
MaxAccL(100000)       --Set the max. acceleration/deceleration of linear motion to 100000 mm/sec^2
SetPassMode( TM_TIME_PASS )     --Set to time overlap mode.
SetPassTime(100)           --Set the interruption time to 100%.
MovL("P4", "BLENDSTART",80,40,40,40)
MovL("P3", "PASS",80,40,40,40)
--Time overlap mode will be applied to the two paths. The first instruction's whole deceleration zone (100%) will overlap the next instruction.
```

Instruction: SetWaitCmdMode

SetWaitCmdMode(WaitCmdMode)

- Set the waiting mode, which determines the time to execute the next instruction. The time for executing the instruction is determined by whether interpolator buffer is available and whether motor is in position (including feedback and command). If the interpolator buffer is available, it means it can save another motion instruction. Whether previous instruction has been executed completely or not, the next command will be sent to MS controller as long as the buffer is available. When the motor is in position (feedback and command), it means the next instruction is sent to MS controller only when the current one is executed completely. The default mode is "motor in position" (**MOTION_INPOSITION**).
- **WaitCmdMode**: Waiting mode of the instruction
  1. Interpolator is available: Enter **MOTION_WAITBUFFER**.
  2. Motor in position (feedback): Enter **MOTION_INPOSITION**.
  3. Motor in position (command): Enter **MOTION_DONE**.

Example:

```
SetWaitCmdMode(MOTION_WAITBUFFER)  --Set the waiting mode to "interpolator buffer available"

MovP("P1")

MovP("P2")
--Issue MovP("P1"). Then, execute MovP ("P2") as long as the interpolator buffer is available.
Meanwhile, MovP("P1") might still being executed.

SetWaitCmdMode(MOTION_INPOSITION)

--Set the waiting mode to "motor in position (feedback).

MovP("P3")

MovP("P4")

--When it reaches the point named P3 by PTP motion command, it can start executing MovP("P4").

SetWaitCmdMode(MOTION_DONE)

--Set the waiting mode to "motor in position" (command).

MovP("P5")

MovP("P6")

--Move in PTP motion and reach point P5, and then start execute MovP("P6").
```

6

Instruction: SetMArchPPS

SetMArchPPS(IsSetMArchPPS)

- Set whether to discard the posture setting when **`MArchP`** is executed and during rising. The default is false.
- **`IsSetMArchPPS`**: Set whether to discard the posture setting during rising.
  1. True: discard posture setting
  2. False: apply the posture setting of the point

Example:

```
SetMArchPPS(true)  –When executing MarchP and ascending, discard the posture setting.
MArchP("P1",100,50,50)

SetMArchPPS(false) –When executing MarchP and ascending,apply the posture setting of the point.
MArchP("P2",100,50,50)
```

6.4.3 Instructions of motion control

There are multiple types of input (**Spd**, **Acc**, **Dec**, **Jerk** are the instructions which are not compulsory). You can do the programming according to your requirement. The combination is as follows.

(1) If no parameters are input: Values of **Spd**, **Acc**, **Dec**, and **jerk** will be the setting values of motion parameters.

(2) If **Spd**, **Acc**, and **Dec** are input: Values of **Spd**, **Acc**, and **Dec** will be the input values. **Jerk** is the value set by the motion parameter.

(3) If all **Spd**, **Acc**, **Dec**, and **Jerk** are input: **Spd**, **Acc**, **Dec**, and **Jerk** will be the input values.

Please note that the parameter sequence has to be identical to that mentioned above. If there is an absent parameter value, the instruction will refer to the value of previous motion parameter.

Regarding the instructions of linear motion, such as **MovL**, **MovLR**, **MArchL**, and **MovCIRC**, you can choose from two input modes, **REAL_SPEED** and **PERCENT_SPEED** mode, to input the speed, acceleration, deceleration, and jerk. In **REAL_SPEED** mode, the values have to be the actual value which units are usually mm and sec. In **PERCENT_SPEED** mode, it requires using **MaxSpdL** and **MaxAccL** to set the max. speed, acceleration, deceleration, and jerk first. And the input for motion control instructions will be in percentage format. Apart from the three combinations mentioned above, linear motion instructions in **PERCENT_SPEED** mode have one more input type:

(4) When only one value in percentage is input: All **Spd**, **Acc** and **Dec** will use this input value while Jerk will be the value set by the instruction of motion parameter.

Please refer to the description and examples of motion control instructions in the following paragraph for more detail.

6

Instruction: MovP

MovP(Point, BMode, Spd, Acc, Dec, Jerk)

- Multiple axes make PTP motion based on the absolute coordinates.
- **Point**: The target point. It can be expressed in two ways: point number and point name (Use quotation mark to quote the name).
- **BMode**:
  1. "PASS": When it is set to "PASS", this path will overlap the previous one for continuous motion. (The waiting mode of the instruction has to be 'interpolator buffer available' (**MOTION_WAITBUFFER**).)
  2. "ABORT": Interrupt the previous motion command and execute the current one. (The waiting mode of the instruction has to be 'interpolator buffer available' (**MOTION_WAITBUFFER**).)
  3. "BLENDSTART": Execute the next line as long as the instruction is issued.
- **Spd**: The max. speed in the unit of %. If **Spd** is not set, the system will refer to the max. speed set by the motion parameter.
- **Acc**: Acceleration setting in the unit of %. If **Acc** is not set, it will refer to the acceleration set by the motion parameter.
- **Dec**: Deceleration setting in the unit of %. If **Dec** is not set, it will refer to the deceleration set by the motion parameter.
- **Jerk**: Jerk setting in the unit of %. If **Jerk** is not set, it will refer to the jerk set by the motion parameter.

Example:

```
JerkJ(20)     --Set the jerk of PTP motion to 20%.
AccJ(30)      --Set the acceleration of PTP motion to 30%.
DecJ(30)      --Set the deceleration of PTP motion to 30%.
SpdJ(20)      --Set the max. speed of PTP motion to 20%
MovP(1)
 --Move to P[1] in PTP motion. The speed, acceleration, deceleration, and jerk values are set by
   motion parameters.
MovP(1, "BLENDSTART")
MovP(2, "PASS")
 --Make a detour around P[1] and move to P[2] for continuous PTP motion. The speed, acceleration,
   deceleration, and jerk values are set by motion parameters.
MovP(3,100,50,50)
--Move to P[3] in PTP motion and set the speed to 100% and acceleration/deceleration to 50%.
MovP("P3", "BLENDSTART",80,40,40)
MovP("P4", "PASS",80,40,40)
 --Move in PTP motion. Set the speed to 80% and acceleration/deceleration to 40%. Make a detour
   around P3 and then move continuously to P4.
MovP("P5",80,40,40,10)
```

--Set the speed to 80%, acceleration/deceleration to 40% and jerk to 10%. Move to P5 in PTP motion.

```
SetWaitCmdMode(MOTION_WAITBUFFER)  --Set the waiting mode to 'interpolator buffer available'.
MovP("P1")        --After MovP("P1")is issued, execute the next instruction as long as the interpolator
                  buffer is available.
DELAY(0.5)        --A delay of 0.5 sec
MovP("P2", "ABORT")
--Execute MovP("P2") whether MovP("P1") has been executed completely by robot or not.
```


Instruction: MovL

MovL(Point, BMode, Spd, Acc, Dec, Jerk)

- Multiple axes make linear motion based on absolute coordinates.
- **Point**: The target point. It can be expressed in two ways: point number or point name (quote the name with quotation mark " ").
- **BMode**:
  1. "PASS": When it is set to "PASS", this path will overlap the previous one for continuous motion. (The waiting mode of the instruction has to be 'interpolator buffer available' (**MOTION_WAITBUFFER**).)
  2. "ABORT": Interupt the previous motion instruction and execute the current one. (The waiting mode of the instruction has to be 'interpolator buffer available' (**MOTION_WAITBUFFER**).)
  3. "BLENDSTART": Execute the next line as long as the instruction is issued.
- Switch the input mode according to **MovL_MODE**. The default is **REAL_SPEED** mode.
- **REAL_SPEED** mode

  Spd: The max. speed in the unit of mm/sec. If **Spd** is not set, it will refer to the max. speed set by the motion parameter.

  Acc: Acceleration setting in the unit of mm/sec^2. If **Acc** is not set, it will refer to the acceleration set by the motion parameter.

  Dec: Deceleration setting in the unit of mm/sec^2. If **Dec** is not set, it will refer to the deceleration set by the motion parameter.

  Jerk: Jerk setting in the unit of mm/sec^3，. If **Jerk** is not set, it will refer to the jerk set by the motion parameter.
- **PERCENT_SPEED** mode

  Spd: The max. speed in the unit of %. If **Spd** is not set, it will refer to the max. speed set by the motion parameter.

  Acc: Acceleration setting in the unit of %. If **Acc** is not set, it will refer to the acceleration set by the motion parameter.

  Dec: Deceleration setting in the unit of %. If **Dec** is not set, it will refer to the deceleration set by the motion parameter.

  Jerk: Jerk setting in the unit of %. If **Jerk** is not set, it will refer to the jerk set by the motion

6

parameter.

Example:

```
-- REAL_SPEED mode
MovL_MODE = REAL_SPEED
JerkL(5000000)  --Set the actual jerk of linear motion to 5000000 mm/sec3.
AccL(25000)     -- Set the actual acceleration of linear motion to 25000 mm/sec2.
DecL(25000)     --Set the actual deceleration of linear motion to 25000 mm/sec2.
SpdL(150)           --Set the actual max. speed of linear motion to 150 mm/sec.
MovL("P1")
 --Move to the point named P1 in linear motion. Its speed, acceleration, deceleration and jerk are set by
   instructions of motion parameters.
MovL(1, "BLENDSTART")
MovL(2, "PASS")
 --Make a detour around P[1] and move to P[2] in linear motion. Its speed, acceleration, deceleration,
   and jerk values are set by motion parameters.
MovL(3,100,5000,5000)
 -- Move to P[3] in linear motion at the speed of 100 mm/sec and the acceleration/deceleration of 5000
   mm/sec2.
MovL("P4", "BLENDSTART",80,4000,4000)
MovL("P3", "PASS",80,4000,4000)
 --Move in linear motion at the speed of 80 mm/sec and acceleration/deceleration of 4000 mm/sec2.
   Make a detour around the point named P4 and move to P3.
MovL("P5",80,4000,4000,3000000)
 --Move to the point named P5 in linear motion at the speed of 80 mm/sec, the
   acceleration/deceleration of 4000 mm/sec2 and the jerk of 3000000 mm/sec3.


-- PERCENT_SPEED mode
MovL_MODE = PERCENT_SPEED
MaxSpdL(2000)           --Set the max. speed of linear motion to 2000 mm/sec.
MaxAccL(5000000) --Set the max. acceleration and deceleration of linear motion to 5000000 mm/sec2.
JerkL(30)          --Se the actual jerk of linear motion to 30%.
AccL(60)           --Set the actual acceleration of linear motion to 60%.
DecL(60)           --Set the actual deceleration of linear motion to 60%.
SpdL(50)           --Set the actual max. speed of linear motion to 50%.
MovL("P1")
 --Move to the point named P1 in linear motion. Its speed, acceleration, deceleration and jerk are set by
   instructions of motion parameters.
MovL(2)
 --Move to the point named P[2] in linear motion. Its speed, acceleration, deceleration and jerk are set
   by instructions of motion parameters.
MovL("P1",60)
 --Move in linear motion. Set the speed to 60% of its maximum speed. Set the
   acceleration/deceleration to 60% of their maximum.Then, move to the point named P1.
```

```
MovL(3,80,50,50)
```
--Move in linear motion. Set the speed to 80% of its maximum speed. Set the acceleration/deceleration to 50% of their maximum. Then, move to the point named P[3].

```
MovL("P4", "BLENDSTART",80,40,40)

MovL("P3", "PASS",80,40,40)
```
--Move in linear motion. Set the speed to 80% of its maximum speed. Set the acceleration/deceleration to 40% of their maximum. Then, make a detour around P4 and move continuously to P[3].

```
MovL("P5",80,40,40,20)
```
--Move in linear motion. Set the max. peed to 80%, max. acceleration/deceleration to 40%, and max. jerk to 20%. Move to point named P5.

```
SetWaitCmdMode(MOTION_WAITBUFFER)  --Set the waiting mode to 'interpolator buffer available'.
MovL("P1")        --After MovL("P1") is issued and interpolator buffer is available, execute the next
                    instruction.

DELAY(0.5)        --A delay of 0.5 sec.

MovL("P2", "ABORT")
```

--Whether the robot has executed MovL("P1") completely, execute MovL("P2") directly.

Instruction: MovL_EX

MovL_EX(Point, DO_Count, <dis_percent, expression>, BMode , Spd, Acc, Dec, Jerk)

- Multiple axes move in linear motion based on absolute coordinates and it can control DI/O.
- **Point**: The target point. It can be expressed in two ways: point number and point name (Use quotation mark to quote the name).
- **DO_Count**: Number of DO command
- **<dis_percent, expression>** Fill in the numbers of "**dis_percent**" and "**expression**" according to DO_Count.
  1. **dis_percent**: Percentage of the path. If this given percentage is reached, execute the DI/O function in the expression.
  2. **expression**: Expression. You can type the function relevant to DI/O operation. Please use quotation mark to quote the expression.
- **BMode**:
  1. "PASS": When it is set to "PASS", this path will overlap the previous one for continuous motion. (The waiting mode of the instruction has to be 'interpolator buffer available' (**MOTION_WAITBUFFER**).)
  2. "ABORT": Interrupt the previous motion instruction and execute the current one. (The waiting mode of the instruction has to be 'interpolator buffer available' (**MOTION_WAITBUFFER**).)
  3. "BLENDSTART": Execute the next line as long as the instruction is issued.
- Switch the input mode according to the setting of **MovL_MODE**. The default is **REAL_SPEED** mode.
- **REAL_SPEED** mode

  Spd: The max. speed in the unit of mm/sec. If **Spd** is not set, the system will refer to the max. speed set by the motion parameter.

  Acc: Acceleration setting in the unit of mm/sec^2. If **Acc** is not set, the system will refer to the acceleration set by the motion parameter.

  Dec: Deceleration setting in the unit of mm/sec^2. If **Dec** is not set, it will refer to the deceleration set by the motion parameter.

  Jerk: Jerk setting in the unit of mm/sec^3. If **Jerk** is not set, it will refer to the jerk set by the motion parameter.
- **PERCENT_SPEED** mode

  Spd: The max. speed in the unit of %. If **Spd** is not set, it will refer to the max. speed set by the motion parameter.

  Acc: Acceleration setting in the unit of %. If **Acc** is not set, it will refer to the acceleration set by the motion parameter.

  Dec: Deceleration setting in the unit of %. If **Dec** is not set, it will refer to the deceleration set by the motion parameter.

  Jerk: Jerk setting in the unit of %. If **Jerk** is not set, it will refer to the jerk set by the motion parameter.

Example:

```
-- REAL_SPEED mode

MovL_MODE = REAL_SPEED

JerkL(5000000)   --Set the actual jerk of linear motion to 5000000 mm/sec3

AccL(25000)      -- Set the actual acceleration of linear motion to 25000 mm/sec2.

DecL(25000)      --Set the actual deceleration of linear motion to 25000 mm/sec2.

SpdL(150)        --Set the actual max. speed of linear motion to 150 mm/sec.


MovL_EX("P0",3, 20,"User_DO(1, 'ON')", 30,"User_DO(2, 'OFF')", 80,"User_DO(3, 'ON')")
--Move to the point named P0 in linear motion. The speed, acceleration, deceleration, and jerk are set
  by instructions of motion parameters. When the moving distance reaches 20%, execute User_DO
  function and set User DO1 to ON. When the distance reaches 30%, execute User_DO function and
  set User DO2 to OFF. When the moving distance reaches 800%, execute User_DO function and set
  User DO3 to On.
```

6

Instruction: MovPR

MovPR(Point, BMode, Spd, Acc, Dec, Jerk)

- Move in PTP motion based on the relative coordinates.
- **Point**: the target point. It can be expressed in two ways: point number and point name (Use quotation mark to quote the name).
- **BMode**:
  1. "PASS": When "PASS" is set, it means the current path will overlap the previous one for continuous motion. (The waiting mode of the instruction has to be 'interpolator buffer available' (**MOTION_WAITBUFFER**).)
  2. "ABORT": Interrupt the previous motion instruction and execute the current one. (The waiting mode of the instruction has to be 'interpolator buffer available' (**MOTION_WAITBUFFER**).)
  3. "BLENDSTART": Execute the next line as long as the instruction is issued.
- **Spd**: The max. speed in the unit of %. If **Spd** is not set, it will refer to the max. speed set by the motion parameter.
- **Acc**: Acceleration setting in the unit of %. If **Acc** is not set, it will refer to the acceleration set by the motion parameter.
- **Dec**: Deceleration setting in the unit of %. If **Dec** is not set, it will refer to the deceleration set by the motion parameter.
- **Jerk**: Jerk setting in the unit of %. If **Jerk** is not set, it will refer to the jerk set by the motion parameter.

Example:

```
JerkJ(20)     --Set the jerk of PTP motion to 20%.
AccJ(30)      --Set the acceleration of PTP motion to 30%.
DecJ(30)      --Set the deceleration of PTP motion to 30%.
SpdJ(20)      --Set the max. speed of PTP motion to 20%
MovPR(1)
 --Move to P[1] in PTP motion based on the relative coordinates. The speed, acceleration,
   deceleration, and jerk are set by instructions of motion parameters.
MovPR(2, “BLENDSTART”)
MovPR(1, “PASS”)
 --Make a detour around point P[2] in PTP motion and then go to point P[1] based on the relative
   coordinates. The speed, acceleration, deceleration, and jerk are set by instructions of motion
   parameters.
MovPR(3,100,50,50)
 --Move in PTP motion. Set the speed to 100%, the acceleration/deceleration to 50% and move to point
   P[3].
MovPR(“P4”, “BLENDSTART”,80,40,40)
MovPR(“P3”, “PASS”,80,40,40)
 --Move in PTP motion. Set the speed to 80% and the acceleration/deceleration to 40%. Make a detour
   around the point named P4 and then move continuously to P3.
MovPR(“P5”,100,50,50,10)
```

```
--Move in PTP motion. Set the speed to 100%, the acceleration/ deceleration to 50% and the jerk to
  10%. Then, move to P5 based on the relative coordinates.


SetWaitCmdMode(MOTION_WAITBUFFER) --Set the waiting mode to "interpolator buffer

 available".
MovPR("P1")       --After MovPR("P1") is issued, execute the next instruction as soon as the
                  interpolator buffer is available.

DELAY(0.5)        --A delay of 0.5 sec.

MovPR("P2", "ABORT")

--Whether the robot has executed MovPR("P1") completely, execute MovPR("P2") directly.
```

6

Instruction: MovLR

MovLR(Point, BMode, Spd, Acc, Dec, Jerk)

- Multiple axes move in linear motion based on relative coordinates.
- **Point**: The target point. It can be expressed in two ways: point number and point name (Use quotation mark to quote the name).
- **BMode**:
  1. "PASS": When "PASS" is set, it means the current path will overlap the previous one for continuous motion. (The waiting mode of the instruction has to be 'interpolator buffer available' (**MOTION_WAITBUFFER**).)
  2. "ABORT": Interrupt the previous motion instruction and execute the current one. (The waiting mode of the instruction has to be 'interpolator buffer available' (**MOTION_WAITBUFFER**).)
  3. "BLENDSTART": Execute the next line as long as the instruction is issued.
- Switch the input mode according **MovL_MODE**. The default is **REAL_SPEED** mode.
- **REAL_SPEED** mode

  Spd: The max. speed in the unit of mm/sec. If **Spd** is not set, it will refer to the max. speed set by the motion parameter.

  Acc: Acceleration setting in the unit of mm/sec^2. If **Acc** is not set, it will refer to the acceleration set by the motion parameter.

  Dec: Deceleration setting in the unit of mm/sec^2. If **Dec** is not set, it will refer to the deceleration set by the motion parameter.

  Jerk: Jerk setting in the unit of mm/sec^3，. If **Jerk** is not set, it will refer to the jerk set by the motion parameter.
- **PERCENT_SPEED** mode

  Spd: The max. speed in the unit of %. If it is not set, it will refer to the max. speed set by the motion parameter.

  Acc: Acceleration setting in the unit of %. If it is not set, it will refer to the acceleration set by motion parameters.

6

Dec: Deceleration setting in the unit of %. If it is not set, it will refer to the deceleration set by motion parameters.

Jerk: Jerk setting in the unit of %. If it is not set, it will refer to the jerk set by motion parameters.

Example:

```
-- REAL_SPEED mode
MovL_MODE = REAL_SPEED
JerkL(5000000)      --Set the actual jerk of linear motion to 5000000 mm/sec3
AccL(25000)         --Set the actual acceleration of linear motion to 25000 mm/sec2
DecL(25000)         --Set the actual deceleration of linear motion to 25000 mm/sec2
SpdL(150)       --Set the actual max. speed of linear motion to 150 mm/sec
MovLR("P1")
 --Move to P1 in linear motion based on the set relative value. The speed, acceleration, deceleration,
   and jerk are set by instructions of motion parameters.
MovLR(2, "BLENDSTRT")
MovLR(1, "PASS")
 --Make a detour around P[2] in linear motion and move to P[1] based on the set relative value. The
   speed, acceleration, deceleration, and jerk are set by instructions of motion parameters.
MovLR(3,100,5000,5000)
 --Set the speed to 100 mm/sec and move in linear motion. Set the acceleration/deceleration to 5000
   mm/sec2 and move to P[3].
MovLR("P4", "BLENDSTART",80,4000,4000)
MovLR("P3", "PASS",80,4000,4000)
 --Set the speed to 80 mm/sec and move in linear motion. Set the acceleration/deceleration to 4000
   mm/sec2. Make a detour around the point named P4 and then move to P3 based on the relative
   coordinates.
MovLR("P5" ,80,4000,4000,3000000)
 --Move in linear motion. Set the speed to 80 mm/sec, the acceleration/deceleration to 4000 mm/sec2
   and the jerk to 3000000 mm/sec3. Then, move to the point named P5.


-- PERCENT_SPEED mode
MovL_MODE = PERCENT_SPEED
MaxSpdL(2000)           --Set the max. speed of linear motion to 2000 mm/sec.
MaxAccL(5000000)        --Set the max. acceleration of linear motion to 5000000 mm/sec2.
JerkL(30)               --Set the actual jerk of linear motion to 30%.
AccL(60)                --Set the actual acceleration of linear motion to 60%.
DecL(60)                --Set the actual deceleration of linear motion to 60%.
SpdL(50)                --Set the actual max. speed of linear motion to 50%
MovLR("P1")
 --Move to the point named P1 in linear motion. The speed, acceleration, deceleration, and jerk are set
   by instructions of motion parameters.
MovLR(2)
```

--Move to point P[2] in linear motion. The speed, acceleration, deceleration, and jerk are set by instructions of motion parameters.

```
MovLR("P1",60)
```

--Move in linear motion. Set the speed to 60% of the max. speed and the acceleration/deceleration to 60% of their maximum. Then, move to the point named P1 based on the relative coordinates.

```
MovLR(3,80,50,50)
```

--Move in linear motion. Set the speed to 80% of the max. speed and the acceleration/deceleration to 50% of their maximum. Then, move to P[3] based on the relative coordinates.

```
MovLR("P4", "BLENDSTART",80,40,40)

MovLR("P3", "PASS",80,40,40)
```

--Move in linear motion. Set the speed to 80% of the maximum speed. Set the acceleration/deceleration to 40% of their maximum. Make a detour around P4 and go to P3 based on the relative coordinates.

```
MovLR("P5",80,40,40,10)
```

--Move in linear motion. Set the speed to 80% of the maximum speed. Set the acceleration/deceleration to 40% of their maximum and the jerk to 10% of its maximum. Move to P5 based on the relative coordinates.

```
SetWaitCmdMode(MOTION_WAITBUFFER)  --Set the waiting mode to "interpolator buffer available".
MovLR("P1")       --After MovLR("P1") is issued, execute the next instruction as long as the
                  interpolator buffer is available.
DELAY(0.5)        --A delay of 0.5 sec.
MovLR("P2", "ABORT")
```

--Whether the robot has executed MovLR("P1") completely, execute MovLR("P2") directly.

6

Instruction: MArchP

MArchP(Point, h1, h2, h3, Spd, Acc, Dec, Jerk)

- Multiple axes move to the target position in PTP and arched motion.
- **Point**: The target point. It can be expressed in two ways, point number and point name (Use quotation mark to quote the name.)
- **h1**: The max. ascending height of Z-axis in the unit of mm.
- **h2**: The ascending height (the max. safety height) of Z-axis in the unit of mm. It shall not exceed h1.
- **h3**: The descending depth of Z-axis (the max. safety depth) in the unit of mm. It shall not exceed h1.
- **Spd**: The max. speed in the unit of %. If **Spd** is not set, it will refer to the max. speed set by the motion parameter.
- **Acc**: The acceleration in the unit of %. If **Acc** is not set, it will refer to the acceleration set by the motion parameter.
- **Dec**: The deceleration in the unit of %. If **Dec** is not set, it will refer to the deceleration set by the motion parameter.
- **Jerk**: The jerk in the unit of %. If **Jerk** is not set, it will refer to the jerk set by the motion parameter.
- Input all of the parameters **P**, **h2**, **h1**, and **h3**, or the instruction cannot be executed.

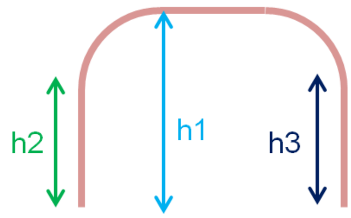


Illustration of MArchP

Example:

```
JerkJ(20)      --Set the jerk of PTP motion to 20%.
AccJ(30)       --Set the acceleration of PTP motion to 30%.
DecJ(30)       --Set the deceleration of PTP motion to 30%.
SpdJ(20)       --Set the max. speed of PTP motion to 20%.
MArchP("P1",100,50,40)
 --Go to the point named P1 in PTP and arched motion. Set the max. ascending height to 100 mm,
   max. safety height to 50 mm, and the safety descending depth to 40 mm in Z-axis. The speed,
   acceleration, deceleration, and jerk are set by the instructions of motion parameters.
MArchP(2,100,50,40,10,5,5)
 --Go to point P[2] in PTP and arched motion. Set the max. ascending height to 100 mm, max. safety
   height to 50 mm, and the safety descending depth to 40 mm in Z-axis. Set the speed to 10% and
   acceleration/deceleration to 5%.
MArchP("P3",100,50,40,10,5,5,2)
 --Go to the point named P3 in PTP and arched motion. Set the max. ascending height to 100 mm, the
   max. safety height to 50 mm, and the safety descending depth to 40 mm in Z-axis. Set the speed to
   10%, the acceleration/deceleration to 5% and the jerk to 2%.
```

Instruction: MArchL

MArchL(Point, h1, h2, h3, Spd, Acc, Dec, Jerk)

- Multiple axes go to the target position in linear and arched motion.
- **Point**: The target point. It can be expressed in two ways: point number and point name (Use quotation mark to quote the name).
- **h1**: The max. ascending height of Z-axis in the unit of mm.
- **h2**: Ascending height of the Z-axis in the unit of mm (the max. safety height). It should not exceed the max. ascending height.
- **h3**: The descending depth of Z-axis (the max. safety depth) in the unit of mm. It shall not exceed h1.
- Switch the input mode according to the setting of **MovL_MODE**. The default is **REAL_SPEED** mode.
- **REAL_SPEED** mode

  Spd: The max. speed in the unit of mm/sec. If **Spd** is not set, it will refer to the max. speed set by the motion parameter.

  Acc: Acceleration setting in the unit of mm/sec^2. If **Acc** is not set, it will refer to the acceleration set by the motion parameter.

  Dec: Deceleration setting in the unit of mm/sec^2. If **Dec** is not set, it will refer to the deceleration set by the motion parameter.

  Jerk: Jerk setting in the unit of mm/sec^3. If **Jerk** is not set, it will refer to the jerk set by the motion parameter.
- **PERCENT_SPEED** mode

  Spd: The max. speed in the unit of %. If **Spd** is not set, it will refer to the max. speed set by the motion parameter.

  Acc: Acceleration setting in the unit of % If **Acc** is not set, it will refer to the acceleration set by the motion parameter.

  Dec: Deceleration setting in the unit of $^{\%}$. If **Dec** is not set, it will refer to the deceleration set by the motion parameter.

  Jerk: Jerk setting in the unit of %. If **Jerk** is not set, it will refer to the jerk set by the motion parameter.
- Input all of the parameters **P**, **h2**, **h1**, and **h3**, or the instruction cannot be executed.

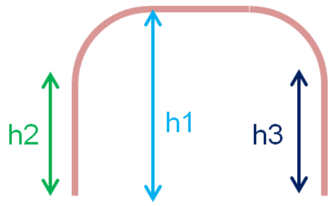


Illustration of MArchL

6

Example:

```
-- REAL_SPEED mode

MovL_MODE = REAL_SPEED

JerkL(5000000)    --Set the actual jerk of linear motion to 5000000 mm/sec3.

AccL(25000)       --Set the actual acceleration of linear motion to 25000 mm/sec2.

DecL(25000)       --Set the actual deceleration of linear motion to 25000 mm/sec2.

SpdL(150)         --Set the actual max. speed of linear motion to150 mm/sec.

MArchL("P1",100,50,40)
 --Go to the point named P1 in linear and arched motion. Set the max. ascending height to 100 mm,
   safety height to 50 mm, and the safety descending depth to 40 mm in Z-axis. The speed,
   acceleration/deceleration, and jerk are set by instructions of motion parameters.

MArchL(2,100,50,40,100,5000,5000)
 --Go to P[2] in linear and arched motion. Set the max. ascending height to 100 mm, the max. safety
   height to 50 mm, and the safety descending depth to 40 mm in Z-axis. Set the speed to 100 mm/sec
   and the acceleration/deceleration to 5000 mm/sec2.

MArchL("P3",100,50,40,100,5000,5000,3000000)
 --Go to the point named P3 in linear and arched motion. Set the max. ascending height to 100 mm, the
   max. safety height to 50 mm, and the safety descending depth to 40 mm in Z-axis. Set the speed to
   100 mm/sec, the acceleration/deceleration to 5000 mm/sec2and the jerk to 3000000 mm/sec3


-- PERCENT_SPEED mode

MovL_MODE = PERCENT_SPEED

MaxSpdL(2000)          --Set the max. speed of linear motion to 2000 mm/sec.

MaxAccL(5000000)       --Set the max. acceleration of linear motion to 5000000 mm/sec2.

JerkL(30)         --Set the actual jerk of linear motion to 30%.

AccL(60)          --Set the actual acceleration of linear motion to 60%

DecL(60)          --Set the actual deceleration of linear motion to 60%

SpdL(50)          --Set the actual max. speed of linear motion to 50%.

MArchL("P1",100,50,40)
 --Go to the point named P1 in linear and arched motion. Set the max. ascending height to 100 mm,
   safety height to 50 mm, and the safety descending depth to 40 mm in Z-axis. The speed,
   acceleration/deceleration, and jerk are set by instructions of motion parameters.

MArchL("P2",100,50,40,60)
 --Go to the point named P2 in linear and arched motion. Set the max. ascending height to 100 mm, the
   max. safety height to 50 mm, and the safety descending depth to 40 mm in Z-axis. Set the speed to
   60% and the acceleration/deceleration to 60%.

MArchL(2,100,50,40,80,50,50)
 --Go to the point named P[2] in linear and arched motion. Set the max. ascending height to 100 mm,
   the max. safety height to 50 mm, and the safety descending depth to 40 mm in Z-axis. Set the speed
   to 80% and the acceleration/deceleration to 50%.

MArchL("P3",100,50,40,80,50,50,5)
 --Go to the point named P3 in linear and arched motion. Set the max. ascending height to 100 mm, the
   max. safety height to 50 mm, and the safety descending depth to 40 mm in Z-axis. Set the speed to
   80%, the acceleration/deceleration to 50%, and the jerk to 5%.
```

Instruction: MArchPT

MArchPT(Point, h1, h2, h3, Spd, Acc, Dec, Jerk)

- Multiple axes go to the target position in PTP and arched motion. **MArchP** is for arched motion with 'distance' overlap while **MArchPT** is for arched motion with 'time' overlap.
- **Point**: The target point. It can be expressed in two ways, point number and point name (Use quotation mark to quote the name.)
- **h1**: The max. ascending height of Z-axis in the unit of mm.
- **h2**: The ascending height (the max. safety height) of Z-axis in the unit of mm. It shall not exceed h1.
- **h3**: The descending depth of Z-axis in the unit of mm. It shall not exceed h1.
- **Spd**: The max. speed in the unit of %. If **Spd** is not set, it will refer to the max. speed set by the motion parameter.
- **Acc**: Acceleration setting in the unit of %. If **Acc** is not set, it will refer to the acceleration set by the motion parameter.
- **Dec**: Deceleration setting in the unit of %. If **Dec** is not set, it will refer to the deceleration set by the motion parameter.
- **Jerk**: Jerk setting in the unit of %. If **Jerk** is not set, it will refer to the jerk set by the motion parameter.
- Input all of the parameters **P**, **h2**, **h1**, **h3**, or this instruction cannot be executed.

Example:

```
JerkJ(20)     --Set the jerk of PTP motion to 20%.
AccJ(30)      --Set the acceleration of PTP motion to 30%.
DecJ(30)      --Set the deceleration of PTP motion to 30%.
SpdJ(20)      --Set the max. speed of PTP motion to 20%.
MArchPT("P1",100,50,40)
 --Go to the point named P1 in PTP and arched motion in the form of time overlap. Set the max.
   ascending height to 100 mm, safety height to 50 mm, and the safety descending depth to 40 mm in
   Z-axis. The speed, acceleration/deceleration, and jerk are set by instructions of motion parameters.
MArchPT(2,100,50,40,10,5,5)
 --Go to point P[2] in PTP and arched motion in the form of time overlap. Set the max. ascending height
   to 100 mm, the max. safety height to 50 mm, and the safety descending depth to 40 mm in Z-axis.
   Set the speed to 10% and the acceleration/deceleration to 5%.
MArchPT("P3",100,50,40,10,5,5,2)
 --Go to point P3 in PTP and arched motion in the form of time overlap. Set the max. ascending height
   to 100 mm, the max. safety height to 50 mm, and the safety descending depth to 40 mm in Z-axis.
   Set the speed to 10%, the acceleration/deceleration to 5%, and the jerk to 2%.
```

6

Instruction: MArchLT

MArchLT(Point, h1, Spd, Acc, Dec, Jerk)

- Multiple axes go to the target position in linear and arched motion. **MArchLT** is for arched motion with 'time' overlap while **MArchL** is for arched motion with 'distance' overlap.
- **Point**: The target point. It can be expressed in two ways, point number and point name (Use quotation mark to quote the name.)
- **h1**: The max. ascending height of Z-axis in the unit of mm.
- Switch the input mode according to **MovL_MODE**. The default is **REAL_SPEED** mode.
- **REAL_SPEED** mode

  Spd: The max. speed in the unit of mm/sec. If **Spd** is not set, it will refer to the max. speed set by the motion parameter.

  Acc: Acceleration setting in the unit of mm/sec^2. If **Acc** is not set, it will refer to the acceleration set by the motion parameter.

  Dec: Deceleration setting in the unit of mm/sec^2. If **Dec** is not set, it will refer to the deceleration set by the motion parameter.

  Jerk: Jerk setting in the unit of mm/sec^3. If **Jerk** is not set, it will refer to the jerk set by the motion parameter.
- **PERCENT_SPEED** mode

  Spd: The max. speed in the unit of %. If **Spd** is not set, it will refer to the max. speed set by the motion parameter.

  Acc: Acceleration setting in the unit of %. If **Acc** is not set, it will refer to the acceleration set by the motion parameter.

  Dec: Deceleration setting in the unit of %. If **Dec** is not set, it will refer to the deceleration set by the motion parameter.

  Jerk: Jerk setting in the unit of %. If **Jerk** is not set, it will refer to the jerk set by the motion parameter.
- Fill in both parameter **P** and **h1**, or this instruction cannot be executed.

Example:

```
-- REAL_SPEED mode
MovL_MODE = REAL_SPEED
JerkL(5000000)      --Set the actual jerk of linear motion to 5000000 mm/sec3.
AccL(25000)         --Set the actual acceleration of linear motion to 25000 mm/sec2.
DecL(25000)         --Set the actual deceleration of linear motion to 25000 mm/sec2.
SpdL(150)           --Set the actual max. speed of linear motion to150 mm/sec.
MArchLT("P1",100)
 --Go to the point named P1 in linear and arched motion in the form of time overlap. Set the max.
   ascending height in Z-axis to 100 mm . The speed, acceleration/deceleration, and jerk are set by
   instructions of motion parameters.
MArchLT(2,100,100,5000,5000)
 --Go to point P[2] in linear and arched motion in the form of time overlap. Set the max. ascending
```

6

height in Z-axis to 100 mm , the speed to 100 mm/sec, and the acceleration/deceleration to 5000 mm/sec^2.

```
MArchLT("P3",100,100,5000,5000,3000000)
```

--Go to the point named P3 in linear and arched motion in the form of time overlap. Set the max. ascending height in Z-axis to 100 mm , the speed to 100 mm/sec, the acceleration/deceleration to 5000 mm/sec^2, and the jerk to 3000000 mm/sec^3.

-- PERCENT_SPEED mode

```
MovL_MODE = PERCENT_SPEED
MaxSpdL(2000)        --Set the max. speed of linear motion to 2000 mm/sec.
MaxAccL(5000000)     --Set the max. acceleration/deceleration of linear motion to 5000000 mm/sec^2.
JerkL(30)            --Set the actual jerk of linear motion to 30%.
AccL(60)             --Set the actual acceleration of linear motion to 60%
DecL(60)             --Set the actual deceleration of linear motion to 60%
SpdL(50)             --Set the actual max. speed of linear motion to 50%.
MArchLT("P1",100)
```

--Go to the point named P1 in linear and arched motion in the form of time overlap. Set the max. ascending height in Z-axis to 100 mm . The speed, acceleration/deceleration, and jerk are set by instructions of motion parameters.

```
MArchLT("P2",100,60)
```

--Go to the point named P2 in linear and arched motion in the form of time overlap. Set the max. ascending height in Z-axis to 100 mm, the speed to 60% and the acceleration/deceleration to 60%.

```
MArchLT(2,100,80,50,50)
```

--Go to P[2] in linear and arched motion in the form of time overlap. Set the max. ascending height in Z-axis to 100 mm, the speed to 80% and the acceleration/deceleration to 50%.

```
MArchLT("P3",100,80,50,50,5)
```

--Go to the point named P3 in linear and arched motion in the form of time overlap. Set the max. ascending height in Z-axis to 100 mm, the speed to 80%, the acceleration/deceleration to 50%, and the jerk to 5%.

6

Instruction: MovJ

MovJ(Axis_idx, Point, BMode, Spd, Acc, Dec, Jerk)

- Single axis moves in PTP motion based on the absolute coordinates.
- **`Axis_idx`**: Motor number, expressed with "Jn" or n. It can be set in two ways, point number and point name (Use quotation mark to quote the name).
- **`Point`**: The target point. Value X is regarded as PUU value.
- **`BMode`**:
  1. "PASS": When "PASS" is set, it means the current path will overlap the previous one for continuous motion. (The waiting mode of the instruction has to be 'interpolator buffer available' (**`MOTION_WAITBUFFER`**).)
  2. "ABORT": Interrupt the previous motion instruction and execute the current one. (The waiting mode of the instruction has to be 'interpolator buffer available' (**`MOTION_WAITBUFFER`**).)
  3. "BLENDSTART": Execute the next line as long as the instruction is issued.
- **`Spd`**: Set the max. speed in the unit of %.
- **`Acc`**: Acceleration setting in the unit of %.
- **`Dec`**: Deceleration setting in the unit of %.
- **`Jerk`**: Jerk setting in the unit of %.

Example:

```
JerkJ(20)    --Set the jerk of PTP motion to 20%.
AccJ(30)     --Set the acceleration of PTP motion to 30%.
DecJ(30)     --Set the deceleration of PTP motion to 30%.
SpdJ(20)     --Set the max. speed of PTP motion to 20%.
MovJ("J1","P1")
 --Axis J1 moves to the point named P1 in PTP motion. The speed, acceleration, deceleration, and jerk
   are set by instructions of motion parameters.
MovJ(2,2,80,10,10)
 --Axis J2 moves to point P[2] in PTP motion. Set the speed to 80% and the acceleration/deceleration
   to 10%.
MovJ("J3","P3",80,10,10,2)
 --Axis J3 moves to the point named P3 in PTP motion. Set the speed to 80%, the
   acceleration/deceleration to 10% and the jerk to 2%.
```

Instruction: MovCIRC

MovCIRC(ECirc, PCirc, ArcMode, BMode, Spd, Acc, Dec, Jerk)

- Make a circular motion based on the absolute coordinates. There are two ways to form a circle. (1) Form a circle with 3 points, current position, arc/circle's end position, arc/circle's passing position. (2) Form a circle with 2 points, current position and arc/circle's center
- **ECirc**: End point of the arc/circle. It can be specified in two ways, point number and point name (use quotation mark to quote the name).
- **PCirc**:
  1. The passing point of an arc/circle. It can be specified in two ways, point number and point name (use quotation mark to quote the name).
  2. The circle center. It can be specified in two ways, point number and point name (use quotation mark to quote the name).
- **ArcMode**: Arc/circle mode selection. Available modes are as follows:
  1. **CM_BORDER_ARC**: Move and form an arc with 3 points based on absolute coordinates.
  2. **CM_BORDER_CIRC**: Move and form a circle with 3 points based on absolute coordinates.
  3. **CM_CENTER_ARC_CCW**: Move and form an arc (with 2 points and 1 center point) in counterclockwise direction based on absolute coordinates.
  4. **CM_CENTER_ARC_CW**: Move and form an arc (with 2 points and 1 center point) in clockwise direction based on absolute coordinates.
  5. **CM_CENTER_CIRC_CCW**: Move and form a circle (with 2 points and 1 center point) in counterclockwise direction based on absolute coordinates.
  6. **CM_CENTER_CIRC_CW**: Move and form a circle (with 2 points and 1 center point) in clockwise direction based on absolute coordinates.
- **BMode**:
  1. "PASS": When "PASS" is set, it means the current path will overlap the previous one for continuous motion. (The waiting mode of the instruction has to be 'interpolator buffer available' (**MOTION_WAITBUFFER**).)
  2. "ABORT": Interrupt the previous motion instruction and execute the current one.(The waiting mode of the instruction has to be 'interpolator buffer available' (**MOTION_WAITBUFFER**).)
  3. "BLENDSTART": Execute the next line as long as the instruction is issued.
- Switch the input mode according to the setting of **MovL_MODE**. The default is **REAL_SPEED** mode.
- **REAL_SPEED** mode

  Spd: The max. speed in the unit of mm/sec. If **Spd** is not set, it will refer to the max. speed set by the motion parameter.

  Acc: Acceleration setting in the unit of mm/sec^2. If **Acc** is not set, it will refer to the acceleration set by the motion parameter.

  Dec: Deceleration setting in the unit of mm/sec^2. If **Dec** is not set, it will refer to the

6

deceleration set by the motion parameter.

Jerk: Jerk setting in the unit of mm/sec^3. If **Jerk** is not set, it will refer to the jerk set by the motion parameter.

- **PERCENT_SPEED** mode

  Spd: The max. speed in the unit of %. If **Spd** is not set, it will refer to the max. speed set by the motion parameter.

  Acc: Acceleration setting in the unit of %. If **Acc** is not set, it will refer to the acceleration set by the motion parameter.

  Dec: Deceleration setting in the unit of %. If **Dec** is not set, it will refer to the deceleration set by the motion parameter.

  Jerk: Jerk setting in the unit of %. If **Jerk** is not set, it will refer to the jerk set by the motion parameter.

- All parameters of **ECirc**, **PCirc**, **ArcMode** have to be complete, or they cannot be executed.

Example:

```
-- REAL_SPEED mode
MovL_MODE = REAL_SPEED
JerkL(1000000)        --Set the jerk to 1000000 mm/sec3.
AccL(5000)     --Set the acceleration to 5000 mm/sec2.
DecL(5000)     --Set the deceleration to 5000 mm/sec2.
SpdL(150)      --Set the speed to150 mm/sec.
MovCIRC("CEnd", "CAux", CM_BORDER_ARC)
 --Regard the current position as the 1st point, point CEnd as the end point and CAux as the passing
   point to make an arc motion (based on these three points). The speed, acceleration, deceleration,
   and jerk are set by instructions of motion parameters.
MovCIRC("CEnd", "CAux", CM_BORDER_CIRC,150,20000,20000)
 --Regard the current position as the 1st point, point CEnd as the end point and CAux as the passing
   point to make a circular motion (based on these three points). Set the speed to 150 mm/sec, the
   acceleration to 20000 mm/sec2, and deceleration to 20000 mm/sec2.
MovCIRC("CEnd", "CAux", CM_BORDER_CIRC, "BLENDSTART",300,10000,
10000)
MovCIRC("CEnd2", "CAux2", CM_BORDER_CIRC, "PASS",300,10000,10000)
 --Regard the current position as the 1st point, point CEnd as the end point and CAux as the passing
   point to make a circular motion (based on these three points). Set the speed to 300 mm/sec, the
   acceleration to 10000 mm/sec2, and deceleration to 10000 mm/sec2. And make a circular motion
   while previous circular motion command is being executed for continuous motion.
MovCIRC("CEnd", "CAux", CM_BORDER_CIRC,150,20000,20000,2000000)
 --Regard the current position as the 1st point, point CEnd as the end point and CAux as the passing
   point to make a circular motion (based on these three points). Set the speed to 150 mm/sec, the
   acceleration to 20000 mm/sec2, deceleration to 20000 mm/sec2, and jerk to 2000000 mm/sec3.


-- PERCENT_SPEED mode
MovL_MODE = PERCENT_SPEED
MaxSpdL(2000)          --Set the max. speed of linear motion to 2000 mm/sec.
```

6

```
MaxAccL(5000000)          --Set the max. acceleration of linear motion to 5000000 mm/sec2.

JerkL(30)          --Set the actual jerk of linear motion to 30%.

AccL(60)          --Set the actual acceleration of linear motion to 60%.

DecL(60)          --Set the actual deceleration of linear motion to 60%.

SpdL(50)          --Set the actual max. speed of linear motion to 50%.

MovCIRC("CEnd", "CAux", CM_BORDER_ARC)
 --Regard the current position as the 1st point, point CEnd as the end point and CAux as the passing
   point to make an arc motion (based on these three points). The speed, acceleration, deceleration,
   and jerk are set by instructions of motion parameters.

MovCIRC("CEnd", "CAux", CM_BORDER_ARC,50)
 --Regard the current position as the 1st point, point CEnd as the end point and CAux as the passing
   point to make arc motion (based on these three points). Set the speed to 50%, the acceleration to
   50%, and deceleration to 30%.

MovCIRC("CEnd", "CAux", CM_BORDER_CIRC,50,30,30)
 --Regard the current position as the 1st point, point CEnd as the end point and CAux as the passing
   point to make a circular motion (based on these three points). Set the speed to 50%, the acceleration
   to 30%, and deceleration to 30%.

MovCIRC("CEnd", "CAux", CM_BORDER_CIRC, "BLENDSTART",60,40,40)

MovCIRC("CEnd2", "CAux2", CM_BORDER_CIRC, "PASS",60,40,40)
 --Regard the current position as the 1st point, point CEnd as the end point and CAux as the passing
   point to make a circular motion (based on these three points). Set the speed to 60%, the acceleration
   to 40%, and deceleration to 40%. And make a circular motion while previous circular motion
   command is being executed for continuous motion.

MovCIRC("CEnd", "CAux", CM_BORDER_CIRC,50,30,30,5)
 --Regard the current position as the 1st point, point CEnd as the end point and CAux as the passing
   point to make circular motion (based on these three points). Set the speed to 50%, the acceleration
   to 30%, and deceleration to 30%, and jerk to 5%.


SetWaitCmdMode(MOTION_WAITBUFFER)  --Set the waiting mode to 'interpolator buffer available'.

MovCIRC("CEnd", "CAux", CM_BORDER_CIRC)
-- After MovCIRC("CEnd", "CAux", CM_BORDER_CIRC) is issued, execute the next instruction as
long as the interpolator buffer is available.

DELAY(0.5)        --A delay of 0.5 sec.

MovCIRC("CEnd2", "CAux2", CM_BORDER_CIRC, "ABORT")
--Whether the robot has completely executed MovCIRC("CEnd", "CAux", CM_BORDER_CIRC),
directly execute MovCIRC("CEnd2", "CAux2", CM_BORDER_CIRC, "ABORT").


MovCIRC("CEnd", "CCenter", CM_CENTER_ARC_CCW)
--Regard the current position as the 1st point, point CEnd as the end point and CCenter as the center
to make circular motion in counterclockwise direction (based on these two points). The speed,
acceleration, deceleration, and jerk are set by instructions of motion parameters.

MovCIRC("CEnd", "CCenter", CM_CENTER_CIRC_CW)
--Regard the current position as the 1st point, point CEnd as the end point and CCenter as the center
to make circular motion in clockwise direction (based on these two points). The speed, acceleration,
deceleration, and jerk are set by instructions of motion parameters.
```

6

Instruction: MovCIRC_DIR

MovCIRC_DIR (ECirc, PCirc, ArcMode, OriChoiceMode, OriControlMode, Spd, Acc, Dec, Jerk)

- Before an arc motion starts on absolute coordinates, adjust the end point direction. This instruction requires to work with **MovCIRC_EX()**.
- **ECirc**: End point of the arc/circle. It can be specified in two ways, point number and point name (use quotation mark to quote the name).
- **PCirc**:
  1. The passing point of an arc or a circle. It can be specified in two ways, point number and point name (use quotation mark to quote the name).
  2. The circle center. It can be specified in two ways, point number and point name (use quotation mark to quote the name).
- **ArcMode**: Arc/circle mode selection. Available modes are as follows:
  1. **CM_BORDER_ARC**: Move and form an arc with 3 points based on absolute coordinates.
  2. **CM_BORDER_CIRC**: Move and form a circle with 3 points based on absolute coordinates.
  3. **CM_CENTER_ARC_CCW**: Move and form an arc (with 2 points and 1 center point) in counterclockwise direction based on absolute coordinates.
  4. **CM_CENTER_ARC_CW**: Move and form an arc (with 2 points and 1 center point) in clockwise direction based on absolute coordinates.
  5. **CM_CENTER_CIRC_CCW**: Move and form a circle (with 2 points and 1 center point) in counterclockwise direction based on absolute coordinates.
  6. **CM_CENTER_CIRC_CW**: Move and form a circle (with 2 points and 1 center point) in clockwise direction based on absolute coordinates.
- **OriChoiceMode**:
  1. **CIRC_TANGENT_FORWARD**: Start from the tangent point and move in circular motion in forward direction.
  2. **CIRC_TANGENT_REVERSE**: Start from the tangent point and move in circular motion in reverse direction.
  3. **CIRC_CENTRIPETAL**: Start from the tangent point and point to centripetal direction.
  4. **CIRC_CENTRIFUGAL**: Start from the tangent point and point to centrifugal direction.
- **OriControlMode**:
  1. **CIRC_SPACE_ANGLE_VAR**: Angle of the tool when moving - variable
  2. **CIRC_SPACE_ANGLE_FIX**: Angle of the tool when moving - fixed
  3. **CIRC_PATH_ANGLE_VAR**: Moving direction - variable (Reserved)
  4. **CIRC_PATH_ANGLE_FIX**: Moving direction - fixed
- **BMode**:
  1. “PASS": When “PASS” is set, it means the current path will overlap the previous one for continuous motion. (The waiting mode of the instruction has to be ‘interpolator buffer available (**MOTION_WAITBUFFER**).)
  2. "ABORT": Interrupt the previous motion instruction and execute the current one. (The waiting mode of the instruction has to be ‘interpolator buffer available’

(`MOTION_WAITBUFFER`).)

3. "BLENDSTART": Execute the next line as long as the instruction is issued.

- Switch the input mode according to the setting of **MovL_MODE**. The default is **REAL_SPEED** mode.
- **REAL_SPEED** mode

  Spd: The max. speed in the unit of mm/sec. If **Spd** is not set, it will refer to the max. speed set by the motion parameter.

  Acc: Acceleration setting in the unit of mm/sec^2. If **Acc** is not set, it will refer to the acceleration set by the motion parameter.

  Dec: Deceleration setting in the unit of mm/sec2. If **Dec** is not set, it will refer to the deceleration set by the motion parameter.

  Jerk: Jerk setting in the unit of mm/sec^3. If **Jerk** is not set, it will refer to the jerk set by the motion parameter.
- **PERCENT_SPEED** mode

  Spd: The max. speed in the unit of %. If **Spd** is not set, it will refer to the max. speed set by the motion parameter.

  Acc: Acceleration setting in the unit of %. If **Acc** is not set, it will refer to the acceleration set by the motion parameter.

  Dec: Deceleration setting in the unit of %. If **Dec** is not set, it will refer to the deceleration set by the motion parameter.

  Jerk: Jerk setting in the unit of %. If **Jerk** is not set, it will refer to the jerk set by the motion parameter.
- All parameters of **ECirc**, **PCirc**, and **ArcMode** have to be complete, or they cannot be executed.

Example:

```
-- REAL_SPEED mode
MovL_MODE = REAL_SPEED
JerkL(1000000)        --Set the jerk to 1000000 mm/sec3.
AccL(5000)        --Set the acceleration to 5000 mm/sec2.
DecL(5000)        --Set the deceleration to 5000 mm/sec2.
SpdL(150)         --Set the speed to150 mm/sec.

MovP(CBeg,30,50,50,2)
 MovCIRC_DIR("CEnd", "CAux", CM_BORDER_ARC, CIRC_TANGENT_FORWARD,
 CIRC_PATH_ANGLE_FIX)
MovCIRC_EX()
 --Move and form an arc in CCW direction with 3 points - Fixed moving direction - Tangent
```

6

```
MovP(CBeg,30,50,50,2)

MovCIRC_DIR("CEnd", "CAux", CM_BORDER_CIRC, CIRC_CENTRIPETAL, CIRC_PATH_ANGLE_FIX)

-- Move and form a circle in CCW direction with 3 points - Fixed moving direction - centripetal


MovP(CBeg,30,50,50,2)

MovCIRC_DIR("CEnd", "CAux", CM_CENTER_ARC_CW, CIRC_TANGENT_REVERSE,

CIRC_PATH_ANGLE_FIX)

MovCIRC_EX()
-- Move and form an arc in CW direction with 2 points and 1 circle center - Fixed moving direction -
   Arctangent

MovP(CBeg,30,50,50,2)

MovCIRC_DIR("CEnd", "CAux", CM_CENTER_CIRC_CW, CIRC_CENTRIFUGAL,

CIRC_PATH_ANGLE_FIX)

MovCIRC_EX()
-- Move and form a circle in CW direction with 2 points and 1 center point - Fixed moving direction -
   Centrifugal
```

6

Instruction: MovCIRC_EX

MovCIRC_DIR (Spd, Acc, Dec, Jerk)

- Execute an arc motion based on the absolute coordinates and set the angle for reaching the end point. This instruction will determine the motion path, the moving direction, and the tool's angle to reach the end point according to parameters of **ECirc**, **PCirc**, **ArcMode**, **OriChoiceMode**, and **OriControlMode**.
- Switch the input mode according to the setting of **MovL_MODE**. The default is **REAL_SPEED** mode.
- **REAL_SPEED** mode

  Spd: The max. speed in the unit of mm/sec. If **Spd** is not set, it will refer to the max. speed set by the motion parameter.

  Acc: Acceleration setting in the unit of mm/sec^2. If **Acc** is not set, it will refer to the acceleration set by the motion parameter.

  Dec: Deceleration setting in the unit of mm/sec^2. If **Dec** is not set, it will refer to the deceleration set by the motion parameter.

  Jerk: Jerk setting in the unit of mm/sec^3. If **Jerk** is not set, it will refer to the jerk set by the motion parameter.
- **PERCENT_SPEED** mode

  Spd: The max. speed in the unit of %. If **Spd** is not set, it will refer to the max. speed set by the motion parameter.

  Acc: Acceleration setting in the unit of %. If **Acc** is not set, it will refer to the acceleration set by the motion parameter.

  Dec: Deceleration setting in the unit of %. If **Dec** is not set, it will refer to the deceleration set by the motion parameter.

  Jerk: Jerk setting in the unit of %. If **Jerk** is not set, it will refer to the jerk set by the motion parameter.

Example:

```
-- REAL_SPEED mode
MovL_MODE = REAL_SPEED
JerkL(1000000)          --Set the jerk to 1000000 mm/sec3.
AccL(5000)        --Set the acceleration to 5000 mm/sec2.
DecL(5000)        --Set the deceleration to 5000 mm/sec2.
SpdL(150)         --Set the speed to150 mm/sec.

MovP(CBeg,30,50,50,2)
 MovCIRC_DIR("CEnd", "CAux", CM_BORDER_ARC, CIRC_TANGENT_FORWARD,
 CIRC_PATH_ANGLE_FIX)
MovCIRC_EX()
```

6

```
--Move and form an arc in CCW direction with 3 points-   Fixed moving direction - Tangent


MovP(CBeg,30,50,50,2)
 MovCIRC_DIR("CEnd", "CAux", CM_BORDER_CIRC, CIRC_CENTRIPETAL, CIRC_PATH_ANGLE_FIX)
MovCIRC_EX()
 -- Move and form a circle in CCW direction with 3 points - Fixed moving direction - Centripetal


MovP(CBeg,30,50,50,2)
 MovCIRC_DIR("CEnd", "CAux", CM_CENTER_ARC_CW, CIRC_TANGENT_REVERSE,
 CIRC_PATH_ANGLE_FIX)
MovCIRC_EX()
 -- Move and form an arc in CW direction with 2 points and 1 circle center - Fixed moving direction -
   Arctangent



 MovP(CBeg,30,50,50,2)
 MovCIRC_DIR("CEnd", "CAux", CM_CENTER_CIRC_CW, CIRC_CENTRIFUGAL,
 CIRC_PATH_ANGLE_FIX)
 MovCIRC_EX()
 -- Move and form a circle in CW direction with 2 points and 1 circle center - Fixed moving direction -
   Centrifugal
```

Instruction: StopAxis

StopAxis (Axis_idx, BMode, Dec, Jerk)

- Stop the motion of single axis. This axis will decelerate to stop.
- **Axis_idx**: Motor number, expressed with "Jn" or n. It can be set in two ways, point number and point name (Use quotation mark to quote the name).
- **BMode**:
  1. "PASS": Execute the next line once the instruction is issued. An alarm may occur if motion command is issued right after StopAxis.
  2. "ABORT": Interrupt the previous motion instruction and execute the current one. (The waiting mode of the instruction has to be 'interpolator buffer available' (**MOTION_WAITBUFFER**).)
  3. "BLENDSTART": Execute the next line as long as the instruction is issued.
- **Dec**: Deceleration setting in the unit of %. If **Dec** is not set, it will refer to the deceleration set by the motion parameter.
- **Jerk**: Jerk setting in the unit of %. If **Jerk** is not set, it will refer to the jerk set by the motion parameter.
- Inputting parameters **BMode**, **Dec**, and **Jerk** is optional. The function library will adopt the motion parameters that are set in advance.

Example:

```
SetWaitCmdMode(MOTION_WAITBUFFER)
-- Switch the mode of motion instruction to MOTIOIN_WAITBUFFER.
MovJ(13, "JPoint")
--The 13th axis moves to the point named "JPoint".
if User_DI(1) then – If the condition of User DI 1 is true, stop the motion of the 13th axis.
    StopAxis(13)
 End
```

6

Instruction: StopGroup

StopGroup(BMode, Dec, Jerk)

- Stop the motion of the group. The group will start decelerate to stop. Only 1 group is supported for now.
- **BMode**:
  1. "PASS": Execute the next line once the instruction is issued. An alarm may occur if motion command is issued right after StopAxis.
  2. "ABORT": Interrupt the previous motion instruction and execute the current one. (The waiting mode of the instruction has to be 'interpolator buffer available (**MOTION_WAITBUFFER**).)
  3. "BLENDSTART": Execute the next line as long as the instruction is issued.
- **Dec**: Deceleration setting in the unit of %. If **Dec** is not set, it will refer to the deceleration set by the motion parameter.
- **Jerk**: Jerk setting in the unit of %. If **Jerk** is not set, it will refer to the jerk set by the motion parameter.
- Inputting parameters of **BMode**, **Dec**, and **Jerk** is optional as the function library will refer to the set motion parameters when executing the instruction.

Example:

```
SetWaitCmdMode(MOTION_WAITBUFFER)
-- Switch the mode of motion instruction to MOTIOIN_WAITBUFFER.
MovP("P1")
--The end effector goes to the point named P1.
if User_DI(2) then – If condition of User DI 2 is true, stop the motion of the group.
    StopGroup()
 End
```

6.4.4 DI/O operation

I/O instructions are for controlling User DI/O. They can access I/O status or set DO to on or off.

Instruction: DI

DI(di_idx)

- Access User DI.
- `Di_idx`: Number of DI point, which range is 0 ~ 23.
- Returning 0 signifies off; Returning 1 signifies on.

Example:

```
if DI(1) == 1 then      --If DI 1 is 1, execute the following statement.
    MovL("P1")      --Move to the point named P1 in linear motion.
end
```

Instruction: DO

DO(do_idx, Switch)

- Access or write the User DO.
- `do_idx`: Number of the DO point, which range is 0 ~ 11
- `Switch`: The write in signal, which is set to on or off.
- Returning 0 signifies off; Returning 1 signifies on.

Example:

```
if DO(1) == 1 then      --if DO 1 is 1, execute the following statement.
    DO(2, "ON")      --Set DO 2 to on.
end
```

Instruction: User_DI

User_DI(di_idx)

- Access the the User DI. Returning 0 signifies off; Returning 1 signifies on.
- **Di_idx**: Number of DI point, which range is 1 ~ 24.

Example:

```
if User_DI(1) == 1 then      --If D I1 is 1, then execute the following statement.
   MovL("P1")      --Move to the point named P1 in linear motion.
end
```

Instruction: User_DO

User_DO(do_idx, Switch)

- Access or write the User DO. Returning 0 signifies off; Returning 1 signifies on.
- **do_idx**: Number of the DO point, which range is 1 ~ 12.
- **Switch**: The write-in signal, which is set to on or off.

Example:

```
if User_DO(1) == 0 then      --If DO1 is 0, execute the following statement.
   User_DO(1,"ON")   --Set DO1 to on.
end
```

Instruction: Sys_DI

Sys_DI(di_idx)

- Access the System DI. Returning 0 signifies off; Returning 1 signifies on.
- **di_idx**: Number of the DI point, which range is 1 ~ 8.

Example:

```
if Sys_DI(1) == 1 then      --If DI 1 is 1, execute the following statement.
    MovL("P1")    --Move to the point named P1 in linear motion.
end
```

Instruction: Sys_DO

Sys_DO(do_idx)

- Access the System DO. Returning 0 signifies off; Returning 1 signifies on.
- **do_idx**: Number of the DO point, which range is 1 ~ 8.

Example:

```
if Sys_DO(2) == 0 then      --If DO 2 is 0, execute the following statement.
    MovL("P2")    --Move to the point named P2 in linear motion.
end
```

Instruction: Remote_DI

Remote_DI(method, station_idx, di_idx)

- Access the DI of the expansion axis. Returning 0 signifies off; Returning 1 signifies on.
- **Method**: The connection method; only DMCNET is available currently.
- **Station_idx**: Station number of the expansion axis.
- **di_idx**: Number of DI point, which starts from 1.

Example:

```
if Remote_DI(DMCNET, 1, 1) == 1 then
--If the first DMCNET station of DI 1 is 1, then execute the following statement.
    MovL("P1")    --Move to the point named P1 in linear motion.
end
```

6

Instruction: Remote_DO

Remote_DO(method, station_idx, do_idx, Switch)

- Access the DO of the expansion axis. Returning 0 signifies off; Returning 1 signifies on.
- Method: The connection method; only DMCNET is available currently.
- Station_idx: Station number of the module.
- do_idx: Number of DO point, which starts from 1.
- Switch: Write-in signal, which can be set to on or off. If no value is set, the system will access the DO status.

Example:

```
if Remote_DO(DMCNET, 1, 1) == 1 then
-- If the first DMCNET station of DO 1 is 1, then execute the following statement.
    Remote_DO(DMCNET, 2, 2, "ON")
    -- Set the second DMCNET station of DO 2 to on.
end
```

Instruction: DIO

DIO(target, station_idx, dio_idx)

- Access DI/O status, including User DI/O, System DI/O and DMCNET DI/O. Returning true signifies that DI/O status is on while false signifies off.
- **Target**: The selected DI/O. Setting is as follows.
  1. UDI: User DI
  2. UDO: User DO
  3. SDI: System DI
  4. SDO: System DO
  5. DMCDI: DMCNET expanded DI
  6. DMCDO: DMCNET expanded DO
- **Station_idx**: Station number of the module or port of User and System DI/O.
- **dio_idx**: Number of DI/O point, which starts from 1.

Example:

```
if DIO(UDI,1,5) then
--If User DI 5 of Port 1 is on, then execute the following instruction.
    MovL("P1")      --Move to the point named P1 in linear motion.
end
if DIO(DMCDI,5,1) then
--If DI 5 of the DMCNET station number 1 is on, then execute the following instruction.
    MovL("P2")      --Move to the point named P2 in linear motion.
end
```

Instruction: User_DIs

User_DIs(nDIGrpIdx)

- Access the User DI. Access 16 DIs per time (16 bits) and return the status of 16 DIs (16 bits).
- **nDIGrpIdx**: DI Group No; 16 DIs (16 bits) form a group.

Example:

```
if User_DIs(1) == 0x0055   then
-- If User DI Group1 (User DI 1 ~ DI 16) returns 0x0055 (0b 0000 0000 0101 0101), then execute the following instruction.
    MovL("P1")     --Move to the point named P1 in linear motion.
end
```

Instruction: User_DOs

User_DOs(nDOGrpIdx, nDOGrpValue)

- Access or write the User DO. Access or write 16 DIs per time (16 bits) and return the status of 16 DOs (16 bits).
- **nDOGrpIdx**: DO Group No.; 16 DOs (16 bits) form a group.
- **nDOGrpValue**: The write-in signal, which writes in the status of 16 bits.

Example:

```
if User_DOs(1) == 0x00FF then
-- If Group1 of User DO (User DO 1 ~ DO 16) returns 0x00FF (0b 0000 0000 1111 1111), then execute the following instruction.
    User_DOs(1,0x0050)
--Set Group1 of User DO (User DO 1 ~ DO 16) to 0x0050 (0b 0000 0000 0101 0000)
end
```

6

Instruction: Sys_DIs

Sys_DIs(nDIGrpIdx)

- Access System DI. Access 16 DIs (16 bits) per time and return the status of 16 Dis (16 bits).
- **nDIGrpIdx**: DI Group No.; 16 Dis (16 bits) form a group.

Example:

```
if Sys_DIs(1) == 0x0055 then
--If Group1 of System DI (System DI 1 ~ DI 16) returns 0x0055 (0b 0000 0000 0101 0101), then
  execute the following instruction.
    MovL(“P1”)      --Move to the point named P1 in linear motion.
end
```

Instruction: Sys_DOs

Sys_DOs(nDOGrpIdx)

- Access System DO. Access 16 DOs (16 bits) per time and return the status of 16 DOs (16 bits).
- **nDOGrpIdx**: DO Group No.; 16 DOs (16 bits) form a group.

Example:

```
if Sys_DOs(1) == 0x0055 then
--If Group1 of System DO (System DO1~DO16) returns 0x0055 (0b 0000 0000 0101 0101), then
  execute the following instruction.
    MovL(“P2”)      --Move to the point named P2 in linear motion.
end
```

Instruction: Remote_DIs

Remote_DIs(method, station_idx, nDIGrpIdx)

- Access and write DI of the expansion axis. Access 16 DIs (16 bits) per time and return the status of 16 DIs (16 bits).
- **method**: The connection method; only DMCNET is available currently.
- **station_idx**: Station number of the module.
- **nDIGrpIdx**: DI Group No.; 16 DIs(16 bits) form a group.

Example:

```
if Remote_DIs(DMCNET, 1, 1) == 0x0055 then
-- If DI Group1 of the 1st DMCNET station (DI1~DI16) returns 0x0055 (0b 0000 0000 0101 0101), then
   execute the following instruction.
    MovL("P1")      --Move to the point named P1 in linear motion.
end
```

Instruction: Remote_DOs

Remote_DOs(method, station_idx, nDOGrpIdx, nDOGrpValue)

- Access and write digital output of the expansion axis. Access 16 DOs (16 bits) per time and return the status of 16 DOs (16 bits).
- method: The connection method; only DMCNET is available currently.
- station_idx: Station number of the module.
- nDOGrpIdx: DO Group No.; 16 DOs (16 bits) form a group.
- nDOGrpValue: The write-in signal, which writes in the status of 16 bits.

Example:

```
if Remote_DOs(DMCNET, 1, 1) == 0x0055 then
--If DO Group1 of the 1st DMCNET station (DO1~DO16) returns 0x0055 (0b 0000 0000 0101 0101),
  then execute the following instruction.
    Remote_DO(DMCNET, 1, 2, 0x0050)
    --Set DO Group2 of the 1st DMCNET station (DO17~DO32) to 0x0050 (0b 0000 0000 0101 0000)
end
```

6

Instruction: WaitDIO

WaitDIO(expression, delayTime)

- Wait for DI/O's condition is fulfilled and then execute the next line of code. You can set delay time to skip this instruction.
- **`expression`**: Expression of the instruction, which has to be in the form of string. It works with instruction of **`DI/O (target, station_idx, dio_idx)`**. And you can use operators (including **`and`**, **`or`**, and **`not`**) for creating more DI/O test conditions. Placing **`not`** before a DI/O command means it is to test whether the confition is OFF; if **`not`** isn't placed before a DI/O command, it is to test whether it is ON. **(The statement is read from left to right and the conditions are tested in pairs.)**
- **`delayTime`**: Set the delay time in the unit of second. Once the delay time is over, execute the next line of code.

Example:

```
WaitDIO("DIO(UDI,1,5) and not DIO(UDO,1,6) and DIO(DMCDO,4,1)")
-- Wait for the following condition is fulfilled: User DI 5 = on, User DO 6 = off, and DO 1 of DMCNET
   station 4 = on.

MovL("P1")     --Move to the point named P1 in linear motion.

WaitDIO("DIO(UDI,1,2) or DIO(SDO,1,3)", 5)
--Wait for the condition, User DI 2 = on or System DO 3 = on, execute the next line if the waiting time
  exceeds 5 seconds.

MovL("P2")     --Move to the point named P2 in linear motion.
```

6.4.5 Servo

Instruction: ServoOn

ServoOn(ax_idx)

- Enable the servo.
- **ax_idx**: Servo number. Servo number of MS controller is13 ~ 16.

Example:

```
ServoON(1)                  --Enable Servo 1.
```

Instruction: ServoOff

ServoOff(ax_idx)

- Servo Off
- **ax_idx**: Servo number. Servo number of MS controller is13 ~ 16.

Example:

```
ServoOff(1)               --Disable Servo 1.
```

Instruction: ServoOnGroup

ServoOnGroup ( )

- Switch all the axes in Group1 to Servo On status.

Example:

```
ServoOnGroup()              --Enable Servo Group 1.
```

Instruction: ServoOffGroup

ServoOffGroup ( )

- Switch all axes of Group2 to Servo Off status.

Example:

```
ServoOffGroup()             --Disable Servo Group 1.
```

6

6.4.6 Read and Write of the memory

Instruction: ModbusRead16

ModbusRead16(Address)

- Read the address of Modbus (16 bits).
- **`Address`**: The read address in hexadecimal format and the range is 0x0000 ~ 0x97FF.

Example:

```
ModbusRead16(0x00E0)      --Read value from address 0x00E0 of the Modbus zone.
```

Instruction: ModbusRead32

ModbusRead32(Address)

- Read the address of Modbus (32 bits).
- **`Address`**: The read address in hexadecimal format, which range is 0x0000 ~ 0x97FF.

Example:

```
ModbusRead32(0x0140)      --Read value from address 0x0140 of the Modbus zone.
```

Instruction: ModbusWrite16

ModbusWrite16(Address, Value)

- Write to the address of Modbus (16 bits).
- **`Adress`**: The write-in address in hexadecimal format, which range is 0x0000~ 0x97FF.
- **`Value`**: The value to be written.

Example:

```
ModbusWrite16(0x01A0, 10)      --Write 10 to address 0x01A0 of the Modbus zone.
```

Instruction: ModbusWrite32

ModbusWrite32(Address, Value)

- Write to the address of Modbus (32 bits).
- **`Address`**: The write-in addresse in hexadecimal format, which range is 0x0000 ~ 0x97FF.
- **`Value`**: The value to be written.

Example:

```
ModbusWrite32(0x01B0, 50)  -- Write 50 to address 0x01B0 of the Modbus zone
```

Instruction: PLCMB3Read16

PLCMB3Read16(Address)

- Read the address of PLC (16 bits).
- Address: The read address in decimal format, which range 0 ~ 77823.

Example:

```
PLCMB3Read16(448)        --Read the value from address 448 of the MB3 zone.
```

Instruction: PLCMB3Read32

PLCMB3Read32(Address)

- Read the address of PLC (32 bits).
- Address: The read address in decimal format, which range is 0 ~ 77823.

Example:

```
PLCMB3Read32(640)      --Read the value from address 640 of the MB3 zone.
```

Instruction: PLCMB3Write16

PLCMB3Write16(Address, Value)

- Write to the address of PLC (16 bits).
- Address: The write-in address in decimal format, which range is 0 ~ 77823.
- Value: The value to be written.

Example:

```
PLCMB3Write16(932, 10)      --Write 10 to address 932 in the MB3 zone.
```

Instruction: PLCMB3Write32

PLCMB3Write32(Address, Value)

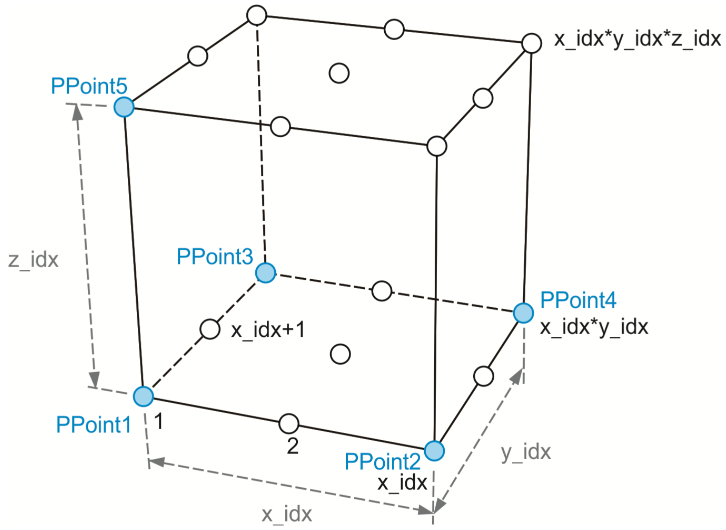
- Write to the address of PLC (32 bits).
- Address: The write-in address in decimal format, which range is 0 ~ 77823.
- Value: The value to be written.

Example:

```
PLCMB3Write32(948, 50)      --Write 50 to address 948 in the MB3 zone.
```

6

6.4.7 Pallet



Demonstration of defining pallet

Instruciton: PalletDef

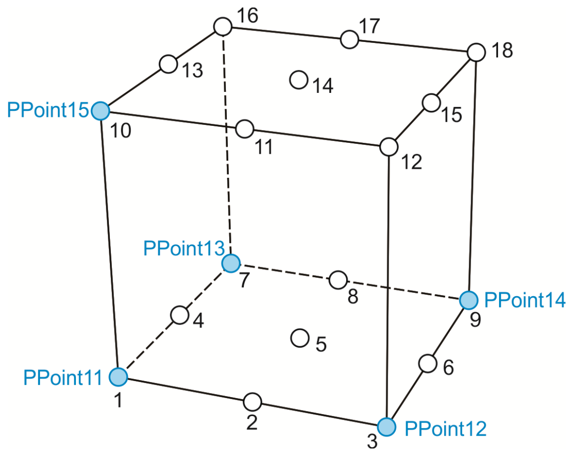
PalletDef(Pallet_idx, x_idx, y_idx, z_idx, PPoint1, PPoint2, PPoint3, PPoint4, PPoint5)

- Define the pallet: Shape the pallet based on the reference point.
- **Pallet_idx**: The pallet number to be defined.
- **x_idx**: Number of points on X-axis.
- **y_idx**: Number of points on Y-axis.
- **z_idx**: Number of points on Z-axis.
- **PPoint1**: Reference point 1 of the pallet to be defined. See the figure above. Input the point name, point number or point array.
- **PPoint2**: Reference point 2 of the pallet to be defined. See the figure above. Input the point name, point number or point array.
- **PPoint3**: Reference point 3 of the pallet to be defined. See the figure above. Input the point name, point number or point array.
- **PPoint4**: Reference point 4 of the pallet to be defined. See the figure above. Input the point name, point number or point array. If this pallet only has one layer (**z_idx** = 1) and you know the exact size of it, then this parameter can be ignored.
- **PPoint5**: Reference point 5 of the pallet to be defined. See the figure above. Input the point name, point number or point array. If this pallet has only one layer (**z_idx** = 1), then this parameter can be ignored.

Example:

```
PalletDef(1,3,3,2, "Pallet11","Pallet12","Pallet13",

"Pallet14","Pallet15")
--Define the points of pallet No. 1: x*y*z = 2*5*3. "Pallet11","Pallet12","Pallet13", "Pallet14" and
 "Pallet15" are the 5 reference points of the pallet. See the figure below.
```



Define the pallet: x,y,z = 3*3*2

Instruction: PalletLength

PalletLength (Pallet_idx)

- Access the point number of the pallet.
- **`Pallet_idx`**: The pallet number to be defined.

Example:

```
Length = PalletLength (1)
--Access the point number of pallet No.1. If the pallet is as illustrated in the above figure, then 3*3*2 =
18.
```

6

Instruction: PalletP

PalletP(Pallet_idx, P_idx)

PalletP(Pallet_idx, x_idx, y_idx, z_idx)

- Access the defined points on the pallet and return the point array. There are two ways. One is to define the pallet by specifying the pallet number, and the other is by specifying the coordinates. With the later way, the point you access is not necessarily to be the pallet point that is already defined; you can also expand the accessing area by specifying the coordinates that haven't been defined on the pallet. For example, input **`Pallet(1,-1,-2,1)`**. See the figure below.
- **`Pallet_idx`**: The pallet number to be defined.
- **`P_idx`**: The point number defined by the pallet.
- **`x_idx`**: Target X-coordinate.
- **`y_idx`**: Target Y-coordinate.
- **`z_idx`**: Target Z-coordinate.

(-1,4)						
		(1,3)	(2,3)	(3,3)		
		(1,2)	(2,2)	(3,2)		
		(1,1)	(2,1)	(3,1)		
(-1,-1)						
(-1,-2)						(5,-2)

X-Y plan of the pallet definition (Z = 1)

Example 1:

```
PalletDef(1,3,3,2, "Pallet11","Pallet12","Pallet13",
"Pallet14","Pallet15")
--Define the points of pallet No.1: x*y*z = 3*3*2. "Pallet11","Pallet12","Pallet13", "Pallet14" and
  "Pallet15" are the 5 reference points of the pallet. See the figure below.
User_DO(1, "OFF")  – Assume that when DO 1 is set to off, it means the gripper releases.
PGet = P[10]        –Define the point for getting the object.
for i = 1, PalletLength(1) do
    MovP( PGet+P.Z(10000) )  –Move to the position above the point for getting the object.
    MovP( PGet )              – Move to the point for getting the object.
    User_DO(1, "ON")          –Gripper holds.
    PalletPoint = PalletP(1,i)     –Access the point of Pallet No.1.
    MovP( PalletPoint + P.Z(10000) )  – Move to the point above the Pallet point.
    MovP( PalletPoint)                – Move to the Pallet point.
    User_DO(1, "OFF")         –Gripper releases.
end
```

Example 2:

```
PalletDef(1,3,3,2, "Pallet11","Pallet12","Pallet13",

"Pallet14","Pallet15")
--Define the points of pallet No. 1: x*y*z = 3*3*2. "Pallet11","Pallet12","Pallet13", "Pallet14" and
  "Pallet15" are the 5 reference points of the pallet.

User_DO(1, "OFF")  –Assume that when DO 1 is off, the gripper releases.

PGet = P[10]        –Define the point for getting the object.

for z = 1, 2 do

  for y = 1, 3 do

    for x = 1, 3 do

        MovP( PGet+P.Z(10000) )  –Move to the position above the point for getting the object.

        MovP( PGet )              –Move to the point for getting the object.

        User_DO(1, "ON")          –Gripper holds.

        PalletPoint = PalletP(1,x,y,z)    –Access the point of pallet No.1

        MovP( PalletPoint + P.Z(10000) )  –Move to the position above the Pallet point.

        MovP( PalletPoint)                –Move to the Pallet point.

        User_DO(1, "OFF")         – Gripper releases.

    end

  end

end
```

6

6.4.8 Time

Instruction: timerInit

timerInit()

- Access the current time, which unit is ms.

Example:

```
tTime = timerInit()    –Current time in the unit of ms.
```

Instruction: timerPass

timerPass(tTime)

- Calculate the difference between current time and input time, which unit is ms.

Example:

```
tTime = timerInit()          –Current time in the unit of ms.
MovP( “P1” )
wTime = timerPass(tTime)  –The difference from current time, which unit is ms.
if wTime < 1000 then        –if wTime is less than 1000 ms, then execute the following program.
    User_DO(1, “ON”)
end
```

6.5 Operators

Operators	Description
+	
-	
*	
/	
^	square
AND	
OR	
XOR	
>	
>=	
<	
<=	
==	
~=	Not equal to
ABS(x)	Returns absolute value
ACOS(x)	The inverse cosine and sine (in radians)
ASIN(x)	The inverse tangent (in radians)
ATAN(x)	The inverse tangent (in radians)
ATAN2(y, x)	Returns the arc tangent of y/x (in radians).
CEIL(x)	The smallest integral value no less than x.
COS(x)	Cosine (in radians)
COSH(x)	The hyperbolic cosine of x.
DEG(x)	Convert from radians to degrees.
EXP(x)	Returns the value e^x
FLOOR(x)	The greatest integer no greater than x.
FMOD(x, y)	Returns the remainder of the division of x by y.
LOG10(x)	Natural logarithm of x to the base 10.
LOG(x)	Natural logarithm of x.
MAX(x, …)	Returns the max. value of the parameter.

6

Operators	Description
MIN(x, …)	Returns the min.value of the parameter.
MODF(x)	Returns two numbers, the integral part of x and the fractional part of x
POW(x, y)	Returns x^y
RAD(x)	Convert from degrees to radians.
SIN(x)	Sine (in radians)
SINH(x)	Hyperbolic sine
SQRT(x)	Returns the square root of x
TAN(x)	Returns the tangent of x (in radians).
TANH(x)	Hyperbolic tangent

6.6 System function library

System function library consists of several DRL function modules and include the control instructions mentioned above. It is usually named “system.luz” and has to be written to the controller so that it can be shared by all motion control projects. Apart from the function provided by Delta, you can create and extend your own function library. The following section will tell you how to create a system function library.

6.6.1 Search for relevant information

You can find information about the function library in the window of **Create library** in DRAS. Click on the existing system function library and you will find the detailed information.

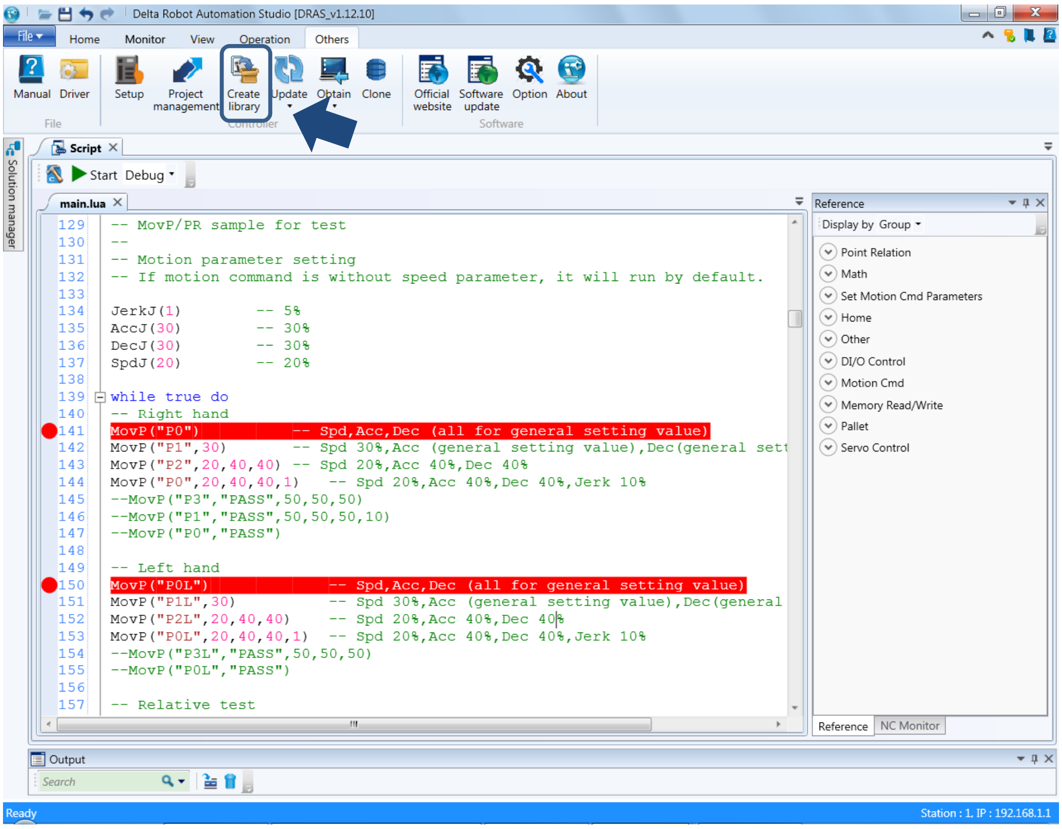


Figure 6.6.1.1 Others - Create system function library

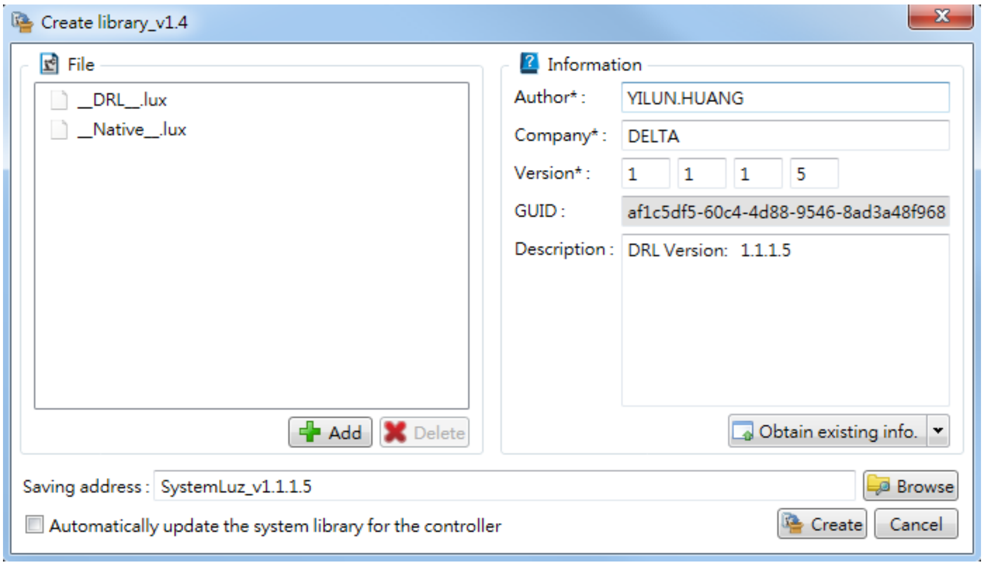


Figure 6.6.1.2 Information of the function library

6.6.2 Create new function library

You can create a new function library via “Create library”. Firstly, add a complete function library to the left side of the file section. Then, fill in the information in the Information section.

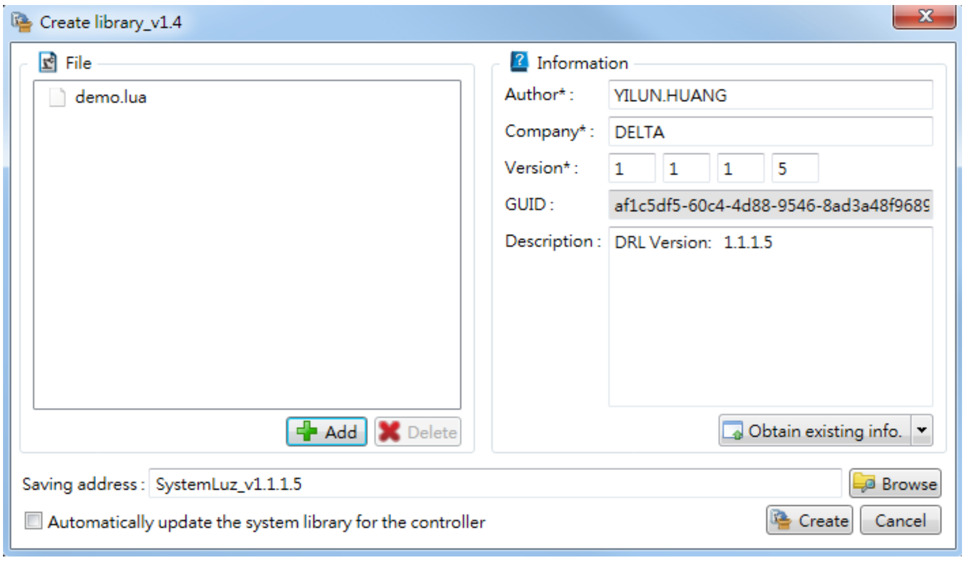


Figure 6.6.2.1 Create a new function library

Note: It is suggested to name the function module with a special name format such as (__XXX__.lua). So, the same filename used by the script and function module can be avoided.

6

6.6.3 Extension

Extending the function library is the best way to keep the original function module. After acquiring the function module encapsuled in the library, you can add a complete function module and modify the relevant information. Then, an extensive system function library can be created. Please note that if the function modules are highly dependent on each other, you will need to check the adding sequence of the file. Generally, the original module should be followed by the expanded ones.

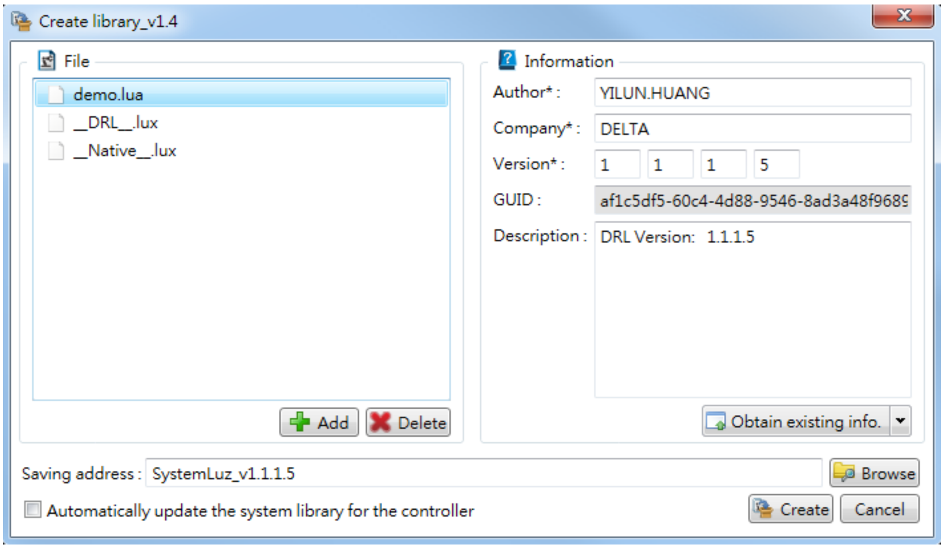


Figure 6.6.3.1 Extension function library

Note: The function module encapsuled in system.luz will be encoded and saved as .lux file. So, the contents can not be viewed and it cannot be restored as a .lua file.

6.6.4 Usage

To share the system function library among all DRL projects, please update the function library to the controller first. And you may start using it once the update is complete.

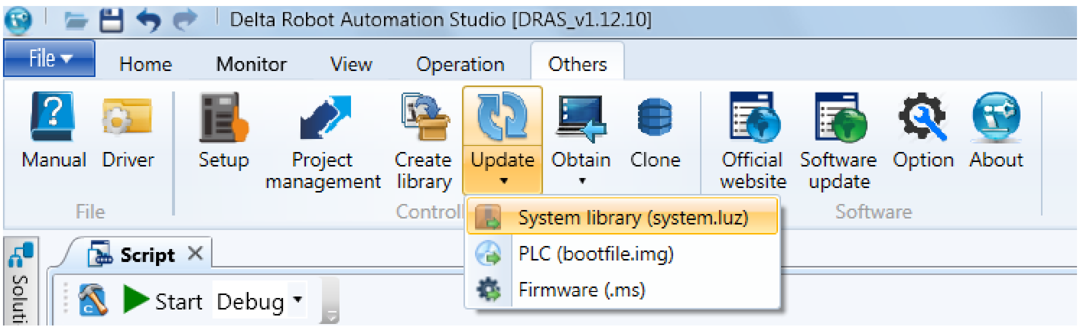


Figure 6.6.4.1 Update system library

6.6.5 How to create a function module?

Function module can be created by DRL script. However, only local/global functions are suitable for the module. And please avoid put other flow control instructions in the non-function block.

Correct

```
function DRA.MArchLC(TargetPosKey, h1, h2 ,h3, ...)
    if TargetPosKey == nil then
        return
    end
    local tmph1 = h1 * 1000
    local tmph2 = h2 * 1000
    local tmph3 = h3 * 1000
    DRA.MotionArchC(TargetPosKey, ARCH_TYPE_MOVE_LIN, tmph1, tmph2, tmph3, ... )
end

function DRA.MArchPC(TargetPosKey, h1, h2, h3, ...)
    if TargetPosKey == nil then
        return
    end
    local tmph1 = h1 * 1000
    local tmph2 = h2 * 1000
    local tmph3 = h3 * 1000
    DRA.MotionArchC(TargetPosKey, ARCH_TYPE_MOVE_P2P, tmph1, tmph2, tmph3, ... )
end
```

Figure 6.6.5.1 Correct coding

Wrong

```
i = 0
while true do
    i = i + 1
end

function DRA.MArchLC(TargetPosKey, h1, h2 ,h3, ...)
    if TargetPosKey == nil then
        return
    end
    local tmph1 = h1 * 1000
    local tmph2 = h2 * 1000
    local tmph3 = h3 * 1000
    DRA.MotionArchC(TargetPosKey, ARCH_TYPE_MOVE_LIN, tmph1, tmph2, tmph3, ... )
end
```

Figure 6.6.5.2 Wrong coding

It is suggested to add a summary after programming completed. By doing so, a code-completion box with detailed description will be available.

```
--- <summary>*多軸Line拱形運動(距離插斷)\n以多軸Line運動且距離插斷的方式做拱形移動至該目標位置</summary
--- <argument name="Point">點位編號或點位名稱</argument>
--- <argument name="h1">Z軸最高上升高度，單位為mm</argument>
--- <argument name="h2">Z軸上升高度(最高安全高度)，不可高於Z軸最高上升高度，單位為mm</argument>
--- <argument name="h3">Z軸下降高度(最低安全高度)，不可高於Z軸最高上升高度，單位為mm</argument>
--- <argument name="...">可輸入BMode, Spd, Acc, Dec, Jerk\n
--- BMode：當設為"PASS"則表示此段路徑會與下一段有設定"PASS"的路徑做重疊的連續移動，沒有設定表示此段路徑結
--- Spd：最大運動速度設定，若沒有設定則以運動參數的最大運動速度設定為主\n
--- Acc：運動加速度設定，若沒有設定則以運動參數的加速度設定為主\n
--- Dec：運動減速度設定，若沒有設定則以運動參數的減速度設定為主\n
--- Jerk：運動加加速度設定，若沒有設定則以運動參數的加加速度設定為主\n
--- (依據MovL_MODE的模式來切換輸入的單位，REAL_SPEED模式單位為mm/sec，PERCENT_SPEED模式單位為%，預設
--- </argument>
function MArchL(Point, h1, h2 ,h3, ...)
    if h1 == nil or h2 == nil or h3 == nil then motion.ProgramStop(string.format("MArchL(P,
    if type(h1) ~= "number" or type(h2) ~= "number" or type(h3) ~= "number" then motion.Pro
    MotionArch(Point, ARCH_TYPE_MOVE_LIN, h1 * 1000, h2 * 1000, h3 * 1000, ... )
end
```

Figure 6.6.5.3 Summary

```
MArchL("Key001", |
        function MArchL(Point, h1, h2 ,h3, ...)
        *多軸Line拱形運動(距離插斷)
        以多軸Line運動且距離插斷的方式做拱形移動至該目標位置
        h1：Z軸最高上升高度，單位為mm
```

Figure 6.6.5.4 Code-completion box

When creating function modules via DRAS, inputting “---” symbols at the previous line will auto prompt the tag required in the function. Relevant contents of each tag will also be elaborated.

```
---
function LOCC(Point,Value)
    local pos_idx = 1
    if type(Point) == "string" then
        pos_idx = P[Point]
        if pos_idx == nil then
```

Figure 6.6.5.5 Inputting “---”in the previous line of the function

```
--- <summary></summary>
--- <argument name="Point"></argument>
--- <argument name="Value"></argument>
function LOCC(Point,Value)
    local pos_idx = 1
    if type(Point) == "string" then
        pos_idx = P[Point]
```

Figure.6.6.6.6 Auto prompted tags (in green)

<Summary> tag is mainly for describing the the function purpose. <argument> tag is mainly for presenting the definition and unit.

```
--- <summary>*點位C資訊\n讀取或寫入點位C資訊</summary>
--- <argument name="Point">點位編號或點位名稱</argument>
--- <argument name="Value">欲寫入的數值</argument>
function LOCC(Point,Value)
    local pos_idx = 1
    if type(Point) == "string" then
        pos_idx = P[Point]
```

Figure 6.6.5.7 Input the summary

Note: When inputing relevant descriptions in the tags, use symbols such as \n and \t

Coordinate System 7

This chapter introduces the coordinate systems applied by MS controller, including machine coordinate system (MCS), product coordinate system (PCS), tool coordinate system (TCS) and axes coordinate system (ACS). You can find detailed information about user interface and commands related to coordinate system.

7

7.1 Descriptions of coordinate system used by MS controller

Four coordinate systems are in MS controller. We will describe the function of each coordinates here.

- Machine coordinate system (MCS)
- Product coordinate system (PCS)
- Tool coordinate system (TCS)
- Axis coordinate system (ACS)

Setting coordinate system is required for operation (jog) of robot arm and point settings. Please observe the following rules. Number 0 of PCS is default and defined as MCS. It cannot be edited by users.

1. Number 0 of TCS is defined as the TCS when no tool is fitted. It cannot be edited by users.
2. When the point belongs to MCS, then the point setting of PCS will be invalid.
3. When the point belongs to ACS, the point setting of PCS and TCS will be invalid.
4. Before using or switching to PCS or TCS, make sure point teaching on the specified coordinate system has been completed. Or an error will occur.
5. Users can access the coordinate system when the motion command is incomplete and the motor is still running. However, users cannot switch the coordinate system.

7.1.1 Machine coordinate system (MCS)

MCS is the default coordinate system, which cannot be edited by users. It is the Cartesian coordinate system. In ASDA-MS, MCS and robot arm's coordinates are defined as the same coordinate system. See figure 7.1.1.1.

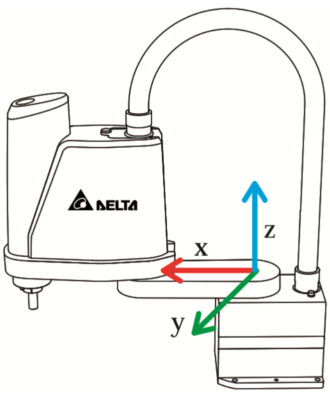


Figure 7.1.1.1 MCS

7

7.1.2 Product coordinate system (PCS)

Users can employ PCS to define the position of objects such as a workpiece or a bench. See figure 7.1.2.1. It also belongs to Cartesian coordinate system.

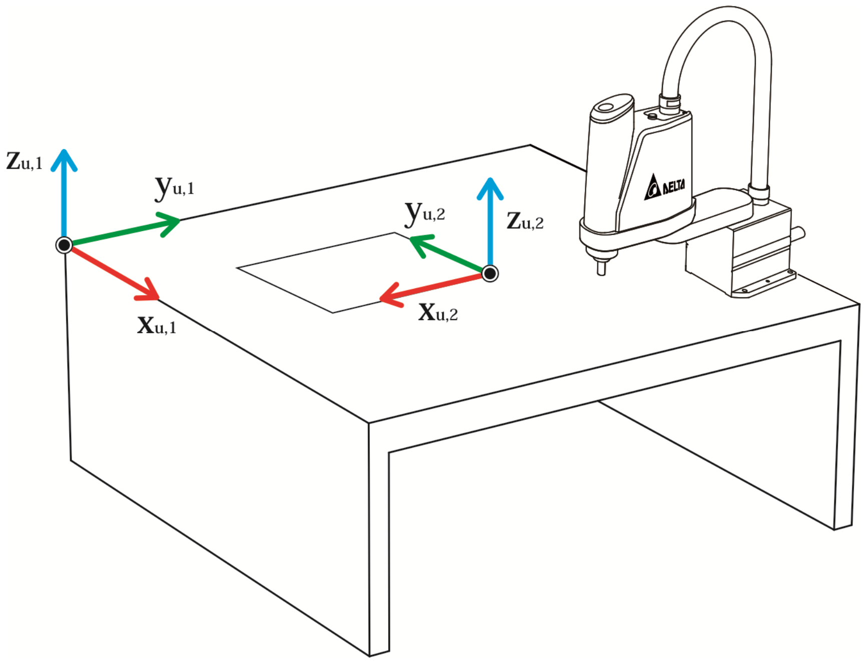


Figure 7.1.2.1 PCS

Since PCS is defined by users, you have to set the relation among the defined coordinate systems and MCS into MS controller via Teach function beforehand. Otherwise, the controller will be unable to apply the coordinate system. Please refer to the following sections for Teach function.

7.1.3 Tool coordinate system (TCS)

TCS is also a user-defined coordinate system. Its original point is usually identical to the end point of the tool. Like PCS, you need to set the relation between the defined TCS and MCS and write this data into MS controller via Teach function. Otherwise, you will be unable to use it. See figure 7.1.3.1. TCS also belongs to Cartesian coordinate system.

TCS is used to show the specific coordinate system that currently applied by the robot arm. Thus, selecting different TCS means the robot arm applies different tools. However, it does not change the position of robot arm.

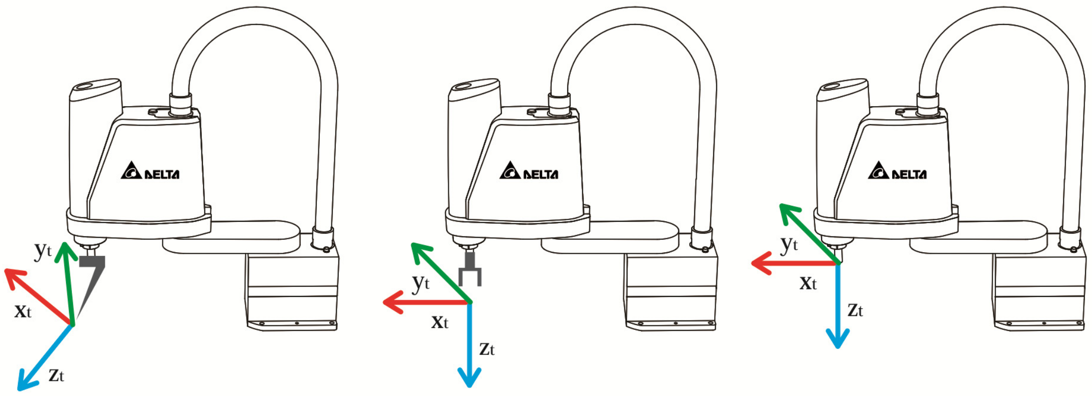


Figure 7.1.3.1 TCS

7.1.4 Axis coordinate system (ACS)

ACS differs from Cartesian coordinate system. The axis is not always in vertical angle. ACS is created by the motor position of each axis on robot arm. Take the 4-axis SCARA supported by MS as the example. ACS is created by 4-axis motor position. And the unit of each axis in ACS is PUU.

7.1.5 Introduction to operation interface

Here provides a brief introduction of coordinate system related interfaces. See detailed descriptions in later sections. Function shows on interface will be displayed in **bold** in this chapter.

- **Jog**
- **Script**
- **Point table**
- **Coordinates**
- **Parameter**

[Jog]

Please enable Jog function. See figure 7.1.5.1.

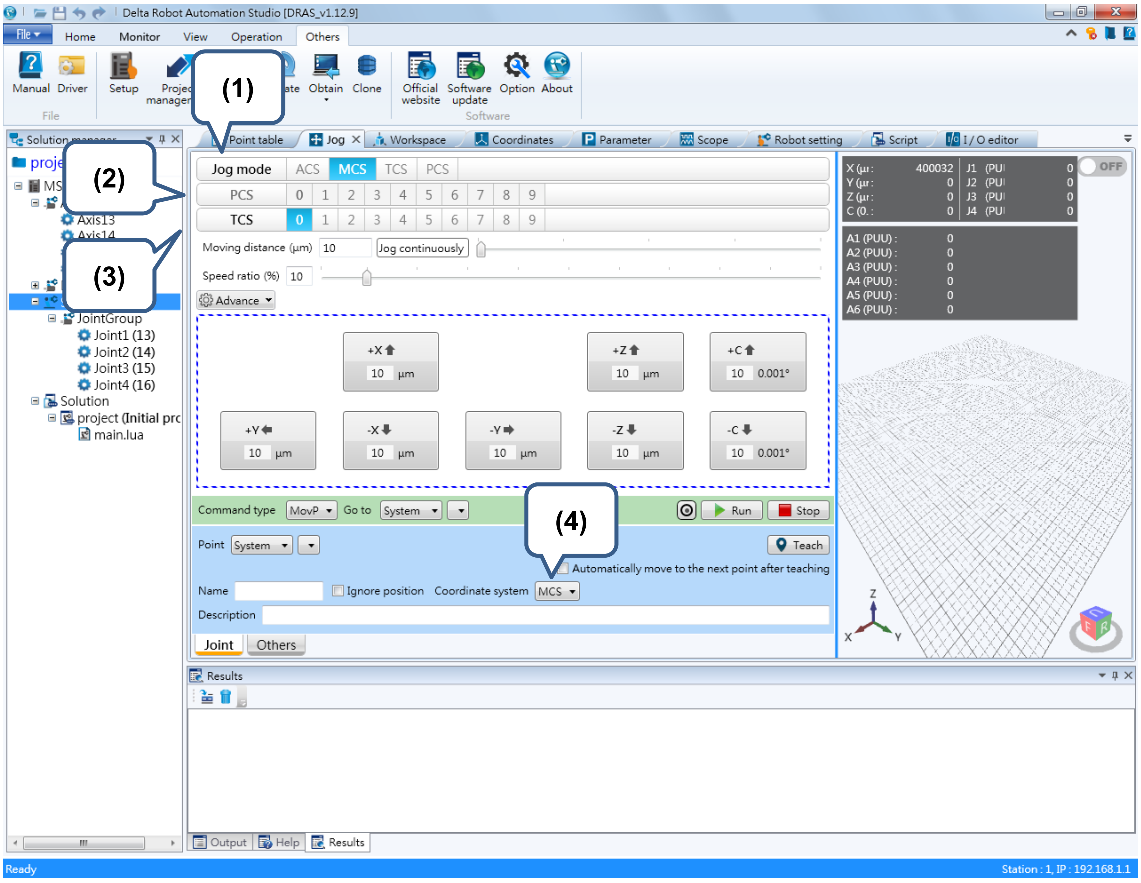


Figure 7.1.5.1 Jog function

(1) **Jog mode** selection; (2) PCS number selection; (3) TCS number selection; (4) Coordinate system selection for point teaching

In **Jog** panel, four main sections can be used to edit the coordinate system:

(1) Jog mode:

Four coordinate systems are available, which are MCS, ACS, PCS and TCS.

(2) PCS number:

When applying jog function, you can choose from number 0 to 9 in PCS. Number 0 is defined as MCS. Number 1 ~ 9 are user-defined. The PCS number can be selected only when it is PCS or TCS in **jog mode**. Moreover, if the selected coordinate system has not been completed point teaching, an error will occur.

(3) TCS number:

When applying jog function, you can choose from number 0 to 9 in TCS. Select number 0 when no tool is fitted. Others are user-defined. PCS number can be selected only when it is MCS, PCS or TCS in **jog mode**. Moreover, if the selected coordinate system has not been completed point teaching, an error will occur.

(4) Coordinate system for point teaching:

Users should select the coordinate system for point teaching. Different mode has different selection.

In MCS mode, MCS and ACS are available.

In ACS mode, PCS and ACS are available.

In PCS mode, PCS and ACS are available.

In TCS mode, PCS and ACS are available.

[Script]

Script is for setting parameters of point position. Main parameters are:

1. **UF**: It is used to set PCS number (0 ~ 9) for the point position.
2. **TF**: It is used to set TCS number (0 ~ 9) for the point position.
3. **Coord**: It is used to set the coordinate system for point position. (0: MCS; 1: PCS; 3: ACS)

Example:

```
P[“example”].x = -25000
P[“example”].y = 40000
P[“example”].z = -100000
P[“example”].a = 0.0
P[“example”].b = 0.0
P[“example”].c = 0.0
P[“example”].Elbow = HAND_RIGHT
P[“example”].PS = 0
P[“example”].UF = 0
P[“example”].TF = 0
P[“example”].Coord = 0
```

7

[Point table]

Please click on **Point table** to enable the function. See the figure 7.1.5.3.

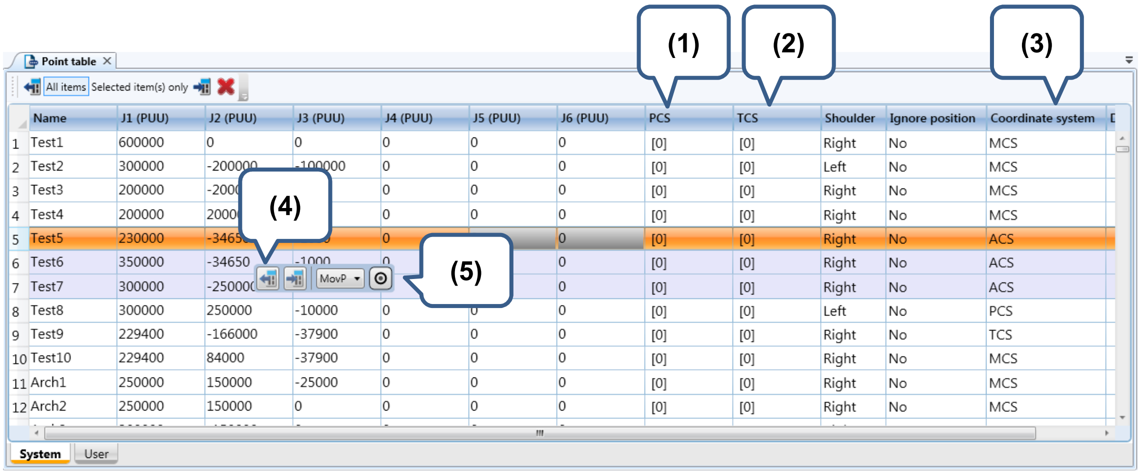


Figure 7.1.5.3 Point table

(1) PCS number:

You can choose from number 0 to 9 in PCS. Number 0 is defined as MCS. Number 1 ~ 9 are user-defined. Please remember to complete point teaching for the specified coordinate system. Otherwise, an error will occur.

(2) TCS number:

You can choose from number 0 to 9 in TCS. Number 0 is defined for the TCS that has no tool fitted while the others are user-defined. Please remember to complete point teaching for the specified coordinate system. Otherwise, an error will occur.

(3) Coordinate system for point teaching:

Four coordinate systems are available, which are MCS, PCS, TCS and ACS.

(4) Read/Write:  is the button of Read and  is the button of Write.

(5) Shortcut panel for jog function: Users can select the command type from the shortcut panel. The jog button is on the right hand side. Click your mouse and the motor will run to the target position. The motor will not stop running unless you release the mouse.

[Coordinates]

Click on **Coordinates** and the screen will be shown as figure 7.1.5.4. Users can complete point teaching for PCS and TCS. Please see detailed information of Teach function in later sections.

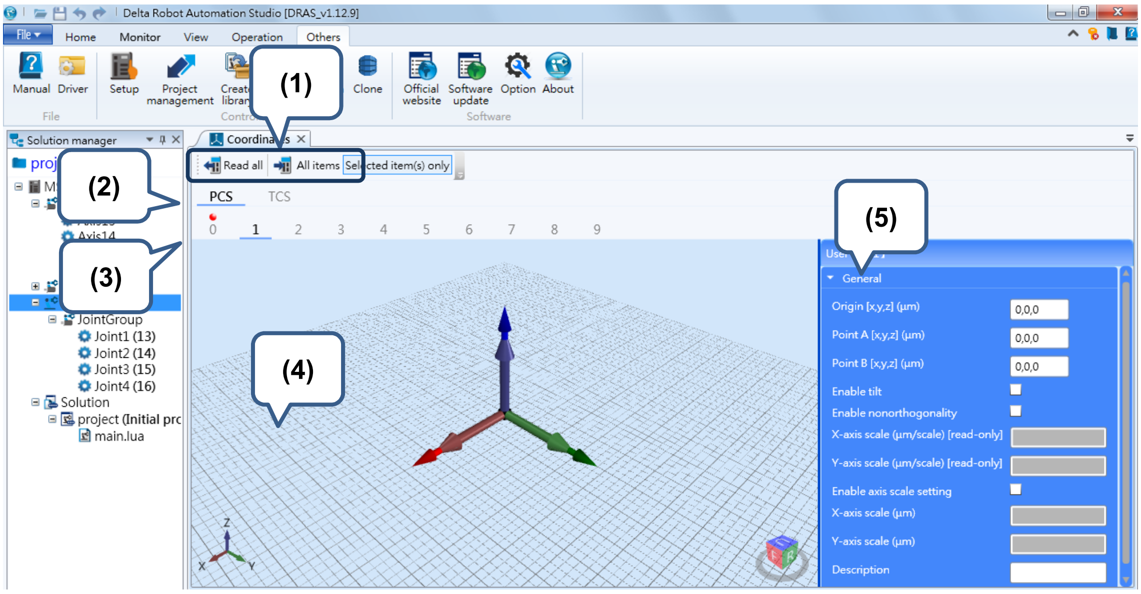


Figure 7.1.5.4 Coordinates

(1) Read/Write:  is the button of Read and  is the button of Write.

(2) PCS/TCS: It is used to switch the current display between PCS and TCS.

(3) Number of coordinate system: Users can switch the number of coordinate system.

(4) Preview for coordinate system: Users can preview the position in the selected coordinate system.

(5) Point teaching for coordinate system:

PCS:

Origin: Write/read the origin of PCS. (Unit: um)

Point A: Write/read the information of point X in PCS. (Unit: um)

Point B: Write/read the information of point Y in PCS. (Unit: um)

Enable tilt: Check this box to enable the Teach function for PCS in non-horizontal level.

Enable nonorthogonality: Check this box to enable the Teach function for PCS in non-orthogonal situation.

X-axis scale (read-only): It displays the scale ratio of axis X in um.

Y-axis scale (read-only): It displays the scale ratio of axis Y in um.

Enable axis scale setting: Check this box and you will be able to define the scaling length of XY axis for PCS.

X-axis scale: It sets the scaling length of axis X in PCS.

Y-axis scale: It sets the scaling length of axis Y in PCS.

Description: Input the description of this coordinate system.

Teach: Input the robot arm position.

TCS:

w: Input the length from tool to flange. (Unit: um)

h: Input the height from tool to flange. (Unit: um)

e: Input the offset angle of tool installation. (Unit: 0.001 degree)

A: Input the rotation angle of axis X in TCS. (Unit: 0.001 degree)

B: Input the rotation angle of axis Y in TCS. (Unit: 0.001 degree)

C: Input the rotation angle of axis Z in TCS. (Unit: 0.001 degree)

Four point method (w, h, e): Specify the tool size by four-point teaching in TCS.

Three point method (A, B, C): Specify the orientation of TCS by three-point teaching.

Description: Input the description of this coordinate system.

[Parameter]

Click on **Parameter** to open parameter editing panel. Go to **MS** > **Controlle**r > **P2** (parameter group) to see coordinate system related parameters, which are P2-06, P2-07, P2-08 and P2-09. See the figure below. These four parameters can be used to edit MS coordinate system. Please refer to the later section for further information.

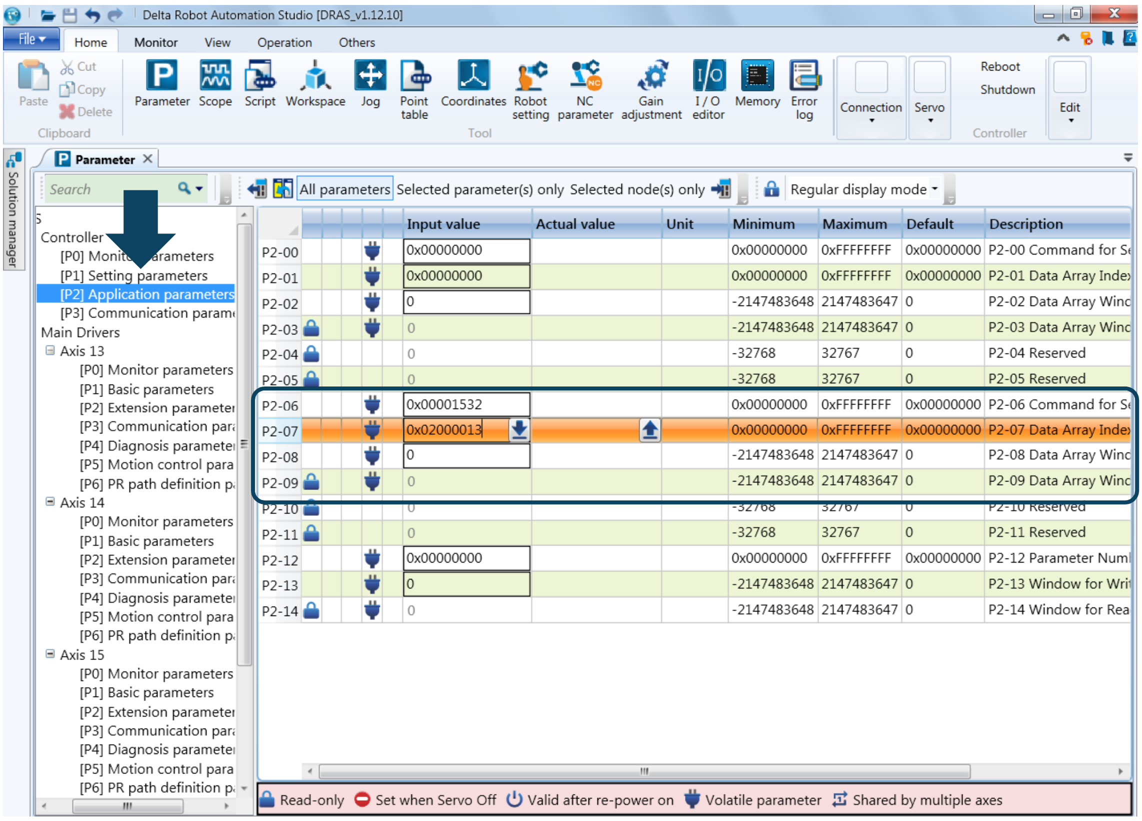


Figure7.1.5.5 Parameter

Following lists the related parameters. Please refer to Chapter 8 for detailed description.

Parameters for setting coordinate system:

Application parameters				
Number	Function	Default value	Unit	Data size
P2-06 ⛉	Command for setting coordinate system parameters	-		32-bit
P2-07 ⛉	Data array index for coordinate system parameters	0		32-bit
P2-08 ⛉	Data array window for writing coordinate system parameters	0	-	32-bit
P2-09 ⛉	Data array window for reading coordinate system parameters	0	-	32-bit

7.2 Machine coordinate system

Machine coordinate system (MCS) is a fundamental coordinate system of robot arm. In DRAS, you can use MCS in **Jog**, **Script**, **Point table** and **Parameter** panels.

7.2.1 Use MCS in [Jog]

Select **MCS** in **Jog mode** so that users can apply Jog function in MCS. You will be able to select PCS number now.

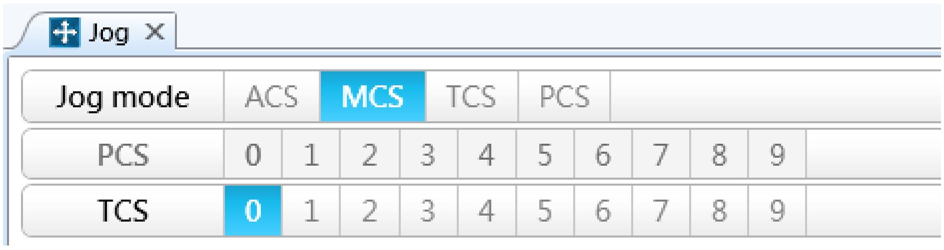


Figure 7.2.1.1 Select MCS in **Jog mode**

When selecting **MCS** in **Jog mode**, only MCS or ACS is available from **Coordinate system** drop-down menu. MCS is used for recording the end point of robot arm. And ACS is used for recording the current position of each axis.

7.2.2 Use MCS in [Script]

To set MCS as the coordinate system, you need to set `Coord` to 0 in the script. See the example of MCS declaration below. When `Coord` is set to 0, `UF` is invalid.

Example:

```
P[“example”].x = -25000
P[“example”].y = 40000
P[“example”].z = -100000
P[“example”].a = 0.0
P[“example”].b = 0.0
P[“example”].c = 0.0
P[“example”].Elbow = HAND_RIGHT
P[“example”].PS = 0
P[“example”].UF = 0
P[“example”].TF = 0
P[“example”].Coord = 0      -- It means to select MCS as the coordinate system
```

7.2.3 Use MCS in [Point table]

Users can set the coordinate system of each point in **Point table**. Enable point table and select **MCS** in **Coordinate system**. Then, press the **Write** button to complete the setting of the point. When the Coordinate system is set to MCS, the number of PCS will be in valid.

Point table ×

All items Selected item(s) only

	Name	X (μm)	Y (μm)	Z (μm)	A (0.001°)	B (0.001°)	C (0.001°)	PCS	TCS	Shoulder	Ignore position	Coordinate system
1	P1	207133	15354	-49992	0	0	0	[0]	[0]	Right	No	MCS

Figure 7.2.3.1 Point table

7.2.4 Use MCS in [Parameter]

In **Parameter** section, users can switch the coordinate system by modifying the value of jog parameter and access the relevant information.

Switch the coordinate system:

For instance, to switch the coordinate system to MCS, users should set P2-06 to 0x00010020.

Steps for accessing feedback position of MCS:

1. Set P2-06 to 0x00000024 (This is for accessing the feedback data of MCS.)
2. Set P2-07 to 0x00000013 (This is for accessing index 0 of data array).
3. Read the value of P2-09. The returned value is the feedback position of axis X (um).
4. Set P2-07 to 0x01000013 (This is for accessing index 1 of data array).
5. Read the value of P2-09. The returned value is the feedback position of axis Y (um).
6. Set P2-07 to 0x02000013 (This is for accessing index 2 of data array).
7. Read the value of P2-09. The returned value is the feedback position of axis Z (um).
8. Set P2-07 to 0x03000013 (This is for accessing index 3 of data array).
9. Read the value of P2-09. The returned value is the feedback rotation angle A of axis X (0.001°).
10. Set P2-07 to 0x04000013 (This is for accessing index 4 of data array).
11. Read the value of P2-09. The returned value is the feedback rotation angle B of axis Y (0.001°).
12. Set P2-07 to 0x05000013 (This is for accessing index 5 of data array).
13. Read the value of P2-09. The returned value is the feedback rotation angle C of axis Z (0.001°).

Steps for accessing the position command of MCS:

1. Set P2-06 to 0x00000124 (This is for accessing the position command of MCS.)
2. Set P2-07 to 0x00000013 (This is for accessing index 0 of data array).
3. Read the value of P2-09. The returned value is the position command of axis X (um).
4. Set P2-07 to 0x01000013 (This is for accessing index 1 of data array).
5. Read the value of P2-09. The returned value is the position command of axis Y (um).
6. Set P2-07 to 0x02000013 (This is for accessing index 2 of data array).
7. Read the value of P2-09. The returned value is the position command of axis Z (um).

8. Set P2-07 to 0x03000013 (This is for accessing index 3 of data array).
9. Read the value of P2-09. The returned value is the rotation angle A of axis X (0.001°).
10. Set P2-07 to 0x04000013 (This is for accessing index 4 of data array).
11. Read the value of P2-09. The returned value is the rotation angle B of axis Y (0.001°).
12. Set P2-07 to 0x05000013 (This is for accessing index 5 of data array).
13. Read the value of P2-09. The returned value is the rotation angle C of axis Z (0.001°).

7.3 Product coordinate system

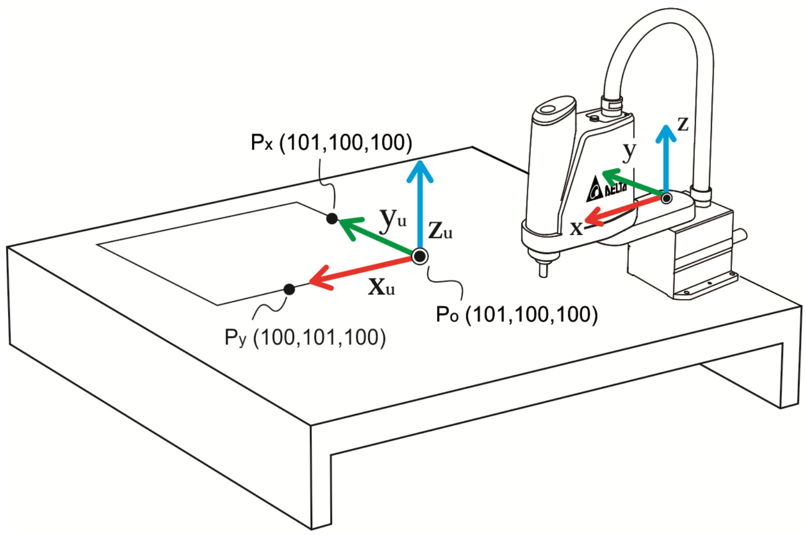
MS controller provides 10 sets of PCS (number 0 ~ 9). Number 0 is defined as MCS, which cannot be modified. Before using PCS number 1 ~ 9, users need to convert the coordinate system into PCS via the Teach function and write it to MS controller. Direct entry method or Three point method can be applied for point teaching. The **Coordinate** or **Parameter** panel can be used as the input interface. This section will firstly introduce the method of point teaching for coordinate system. Then, it will focus on how to use PCS in **Jog**, **Script**, **Point table** and **Parameter** panels.

7.3.1 Teach PCS in [Coordinates]

Three point method

The three point method is for converting coordinates, from PCS to MCS or from MCS to PCS. By entering the origin (P_o), point X (P_x) and point Y (P_y) in MCS, the system will be able to convert them into points in PCS.

Please go to **Coordinate** and observe the following steps to complete point teaching for PCS. See figure 7.3.1.1. Take PCS as the example. Two methods are available to input the point information.



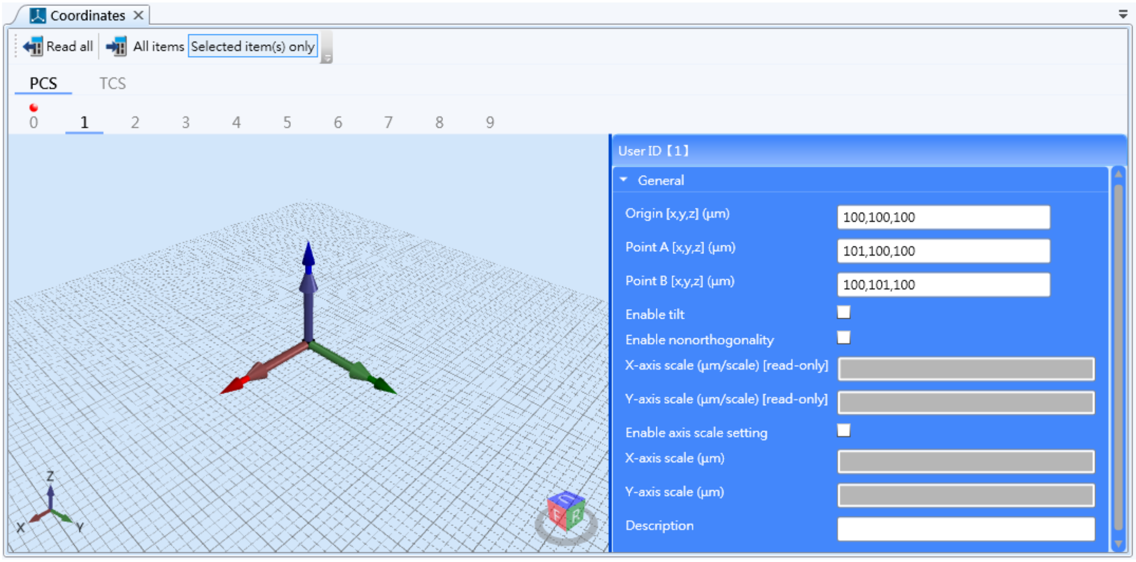


Figure 7.3.1.1 Example of teach PCS and three point method

Method 1: Directly input the point

1. Input the P_o, P_x and P_y value of MCS in the field of Origin, Point A and Point B respectively. (see the figure above)
2. Then, click on the **Write** button. The MS system will automatically convert the coordinate system to PCS. And the point teaching of PCS is complete. If no error occurs, it means point teaching for PCS is done.

Method 2: Acquire the point by jog mode

1. Firstly, use the jog function to move the arm to P_o. Then, press the **Teach** button to input the current coordinates value.
2. Repeat Step 1 and continue to teach P_x and P_y. (See figure 7.3.1.1)
3. Then, click on the **Write** button. The MS system will automatically convert the coordinate system to MCS. And the point teaching of PCS is complete. If no error occurs, it means point teaching for PCS is done.

Then, if you want to verify the point teaching of PCS, please go to **Jog** and select the specified PCS which point teaching has been completed. You will see the change of Cartesian coordinates in Monitoring section.

7.3.2 Teach PCS in [Parameter]

You can find functions related to PCS in **Parameter**. This section will firstly introduce Teach function. Please refer to section 7.3.6 for other detailed information.

Three point method

Please follow the steps below to complete three point method via setting parameters. See the example with the use of Three point method below.

1. Write Origin (P_o):
   - 1.1. Set P2-07 to 0x00010013 (Ready to write X-axis value).
   - 1.2. Set P2-08 to 100.
   - 1.3. Set P2-07 to 0x01010013 (Ready to write Y-axis value).
   - 1.4. Set P2-08 to 100.
   - 1.5. Set P2-07 to 0x02010013 (Ready to write Z-axis value).
   - 1.6. Set P2-08 to 100.
   - 1.7. Set P2-06 to 0x01011142 (Complete writing the P_o value of PCS number 1)
2. Write Point x (P_x):
   - 2.1. Set P2-07 to 0x00010013 (Ready to write X-axis value).
   - 2.2. Set P2-08 to 101.
   - 2.3. Set P2-07 to 0x01010013 (Ready to write Y-axis value).
   - 2.4. Set P2-08 to 100.
   - 2.5. Set P2-07 to 0x02010013 (Ready to write Z-axis value).
   - 2.6. Set P2-08 to 100.
   - 2.7. Set P2-06 to 0x01012142 (Complete writing the P_x value of PCS number 1)
3. Write Point Y (P_y):
   - 3.1. Set P2-07 to 0x00010013 (Ready to write X-axis value).
   - 3.2. Set P2-08 to 100.
   - 3.3. Set P2-07 to 0x01010013 (Ready to write Y-axis value).
   - 3.4. Set P2-08 to 101.
   - 3.5. Set P2-07 to 0x02010013 (Ready to write Z-axis value).
   - 3.6. Set P2-08 to 100.
   - 3.7. Set P2-06 to 0x01013142 (Complete writing the P_y value of PCS number 1)
4. Write the point and start to calculate:
   - 4.1. Set P2-06 to 0x01000141.

If no error occurs, it means point teaching of PCS is complete. Similarly, if you want to verify point teaching for PCS, please go to **Jog** and select the specified PCS which point teaching has been completed. In addition, users can set P2-06 to 0x01010040 to switch to PCS.

Direct entry method

Please follow the steps below to complete the teaching by directly entry method via setting parameters. Take Three point method from the previous section as the example.

1. Write Origin (P_o):
   - 1.1. Set P2-07 to 0x00010013 (Ready to write X-axis value).
   - 1.2. Set P2-08 to 100.
   - 1.3. Set P2-07 to 0x01010013 (Ready to write Y-axis value).
   - 1.4. Set P2-08 to 100.
   - 1.5. Set P2-07 to 0x02010013 (Ready to write Z-axis value).
   - 1.6. Set P2-08 to 100.
   - 1.7. Set P2-06 to 0x01011142 (Complete writing the P_o value of PCS number 1)
2. Write the rotation angle of X, Y, Z:
   - 2.1. Set P2-07 to 0x00010013 (Ready to write the rotation angle of axis X).
   - 2.2. Set P2-08 to 0.
   - 2.3. Set P2-07 to 0x01010013 (Ready to write the rotation angle of axis Y).
   - 2.4. Set P2-08 to 0.
   - 2.5. Set P2-07 to 0x02010013 (Ready to write the rotation angle of axis Z).
   - 2.6. Set P2-08 to 0.
   - 2.7. Set P2-06 to 0x01012142 (Complete writing the rotation angle of axis X, Y, Z in PCS number 1)
3. Write the point and start to calculate:
   - 3.1. Set P2-06 to 0x01000141.

If no error occurs, it means the point teaching of PCS is complete. Please apply the same way of Three point method to verify the result. In addition, users can set P2-06 to 0x01010040 to switch to PCS.

7.3.3 Use PCS in [Jog]

When point teaching of PCS is complete, you can start to use PCS. Please select **PCS** in **Jog mode**. Then, select the specified PCS which point teaching has been completed so that you can use the jog function on PCS.

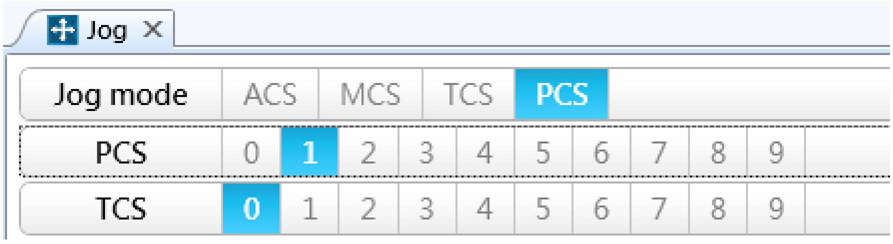


Figure 7.3.3.1 Select PCS in **Jog mode**

Now, the coordinate system is switched to PCS. Users can see the coordinate value in monitoring tab. Please note that before selecting the number of coordinate system, you should make sure point teaching is complete. Otherwise, an error might occur.

When selecting PCS in **jog mode**, only PCS and ACS are available from **Coordinate system** drop-down menu. Selecting PCS means the teach point belongs to the applied Product coordinate system. ACS is used for recording the current position of each axis.

7.3.4 Use PCS in [Script]

To set PCS as the coordinate system, you need to set **`Coord`** to 1 in script. Then, set PCS number in **`UF`**. When it is set to 0, **`UF`** is invalid and no switching command is required. When the program runs to the point, the system will automatically switch to the coordinate system of that point.

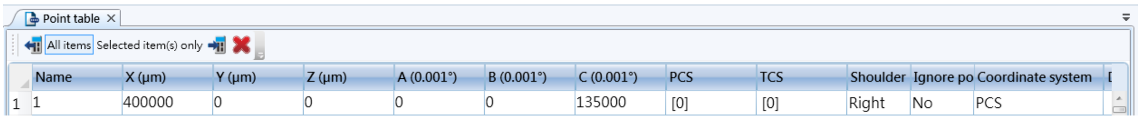
Please note that before using PCS, you need to complete point teaching of PCS. If point teaching has not been done, error will occur when running to the point even when the PCS number is inputted to the script. See the example for point declaration of PCS:

```
P["example"].x = -25000
P["example"].y = 40000
P["example"].z = -100000
P["example"].a = 0.0
P["example"].b = 0.0
P["example"].c = 0.0
P["example"].Elbow = HAND_RIGHT
P["example"].PS = 0
P["example"].UF = 1       -- It means you select PCS number 1.
P["example"].TF = 0
P["example"].Coord = 1    -- It means PCS is selected as the coordinate system.
```

7.3.5 Use PCS in [Point table]

Apart from **Script**, **Point table** can also be used to set up PCS. Go to **Point table** and directly select the **PCS** number and **Coordinate system** for the point. Then, click on **Write** button to complete the setting.

7



	Name	X (μm)	Y (μm)	Z (μm)	A (0.001°)	B (0.001°)	C (0.001°)	PCS	TCS	Shoulder	Ignore po	Coordinate system
1	1	400000	0	0	0	0	135000	[0]	[0]	Right	No	PCS

Figure 7.3.5.1 Point table

Be aware that although users can directly change the PCS number via this panel, if point teaching has not been done, error will occur when running to the point. Please make sure the point teaching for specified coordinate system is complete before using the point.

7.3.6 Use PCS in [Parameter]

Read the data created by Teach function of PCS (Three point method)

Note: Make sure point teaching for the specified coordinate system is complete via Three point method. Otherwise the accessing value might be incorrect.

For example, accessing P_o of PCS number 1:

1. Set P2-06 to 0x01001142 (This is for accessing P_o of PCS number 1).
2. Set P2-07 to 0x00000013 (This is for accessing index 0 of data array).
3. Read P2-09 and the returned value is the X-axis value of P_o (um)
4. Set P2-07 to 0x01000013 (This is for accessing index 1 of data array).
5. Read P2-09 and the returned value is the Y-axis value of P_o (um)
6. Set P2-07 to 0x02000013 (This is for accessing index 2 of data array).
7. Read P2-09 and the returned value is the Z-axis value of P_o (um)

Follow the steps above to access the data of P_x or P_y that created by Teach function. All you need to do is to change the write-in command in step 1 to 0x01002142 (P_x) or 0x01003142 (P_y).

Read the data created by Teach function of PCS (Direct entry method)

Note: Make sure point teaching for the specified coordinate system is complete via Direct entry method. Otherwise the accessing value might be incorrect.

For example, accessing the data of PCS number 1 that created by teach function:

1. Set P2-06 to 0x01001242 (This is for accessing the angle of PCS number 1).
2. Set P2-07 to 0x00000013 (This is for accessing index 0 of data array)
3. Read the value of P2-09. The returned value is the rotation angle A of axis X (0.001°).
4. Set P2-07 to 0x01000013 (This is for accessing index 1 of data array)
5. Read the value of P2-09. The returned value is the rotation angle B of axis Y (0.001°).
6. Set P2-07 to 0x02000013 (This is for accessing index 2 of data array)
7. Read the value of P2-09. The returned value is the rotation angle C of axis Z (0.001°).

Switch to PCS

Take PCS number 1 as the example:

Set P2-06 to 0x01010040 to switch the coordinate system to PCS number 1.

Clear PCS data

Take PCS number 1 as the example:

Set P2-06 to 0x01010F41 to clear the data of PCS number 1.

Steps for accessing the spatial feedback position of PCS:

1. Set P2-06 to 0x00000044 (This is for accessing the feedback data of PCS.)
2. Set P2-07 to 0x00000013 (This is for accessing index 0 of data array).
3. Read the value of P2-09. The returned value is the X-axis position (um).
4. Set P2-07 to 0x01000013 (This is for accessing index 1 of data array).
5. Read the value of P2-09. The returned value is the Y-axis position (um).
6. Set P2-07 to 0x02000013 (This is for accessing index 2 of data array).
7. Read the value of P2-09. The returned value is the Z-axis position (um).
8. Set P2-07 to 0x03000013 (This is for accessing index 3 of data array).
9. Read the value of P2-09. The returned value is the rotation angle A of axis X (0.001°).
10. Set P2-07 to 0x04000013 (This is for accessing index 4 of data array).
11. Read the value of P2-09. The returned value is the rotation angle B of axis Y (0.001°).
12. Set P2-07 to 0x05000013 (This is for accessing index 5 of data array).
13. Read the value of P2-09. The returned value is the rotation angle C of axis Z (0.001°).

7.4 Tool coordinate system

MS controller provides 10 sets of TCS (number 0 ~ 9). Number 0 is defined as the TCS that has no tool fitted and cannot be modified. Before using number 1 ~ 9 of TCS, users need to convert the coordinate system to TCS by completing point teaching and write it to MS controller. Users can use Direct entry method or Four point method (w, h, e) to teach the tool size or Three point method (A, B, C) to teach the orientation of TCS. The **Coordinate** provides graphical user interface for Teach function. **Parameter** also can be used as the input interface through parameter editing. This section will firstly introduce the method of point teaching for coordinate system. Then, it will focus on how to use TCS in **Jog**, **Script**, **Point table** and **Parameter** panels.

7.4.1 Teach TCS in [Coordinates]

Direct Entry Method

Whey applying Direct entry method, you should input the width (w), height (h) and angle (e) of TCS for converting between TCS and MCS. Please go to **Coordinate** and observe the following steps to complete point teaching of TCS.

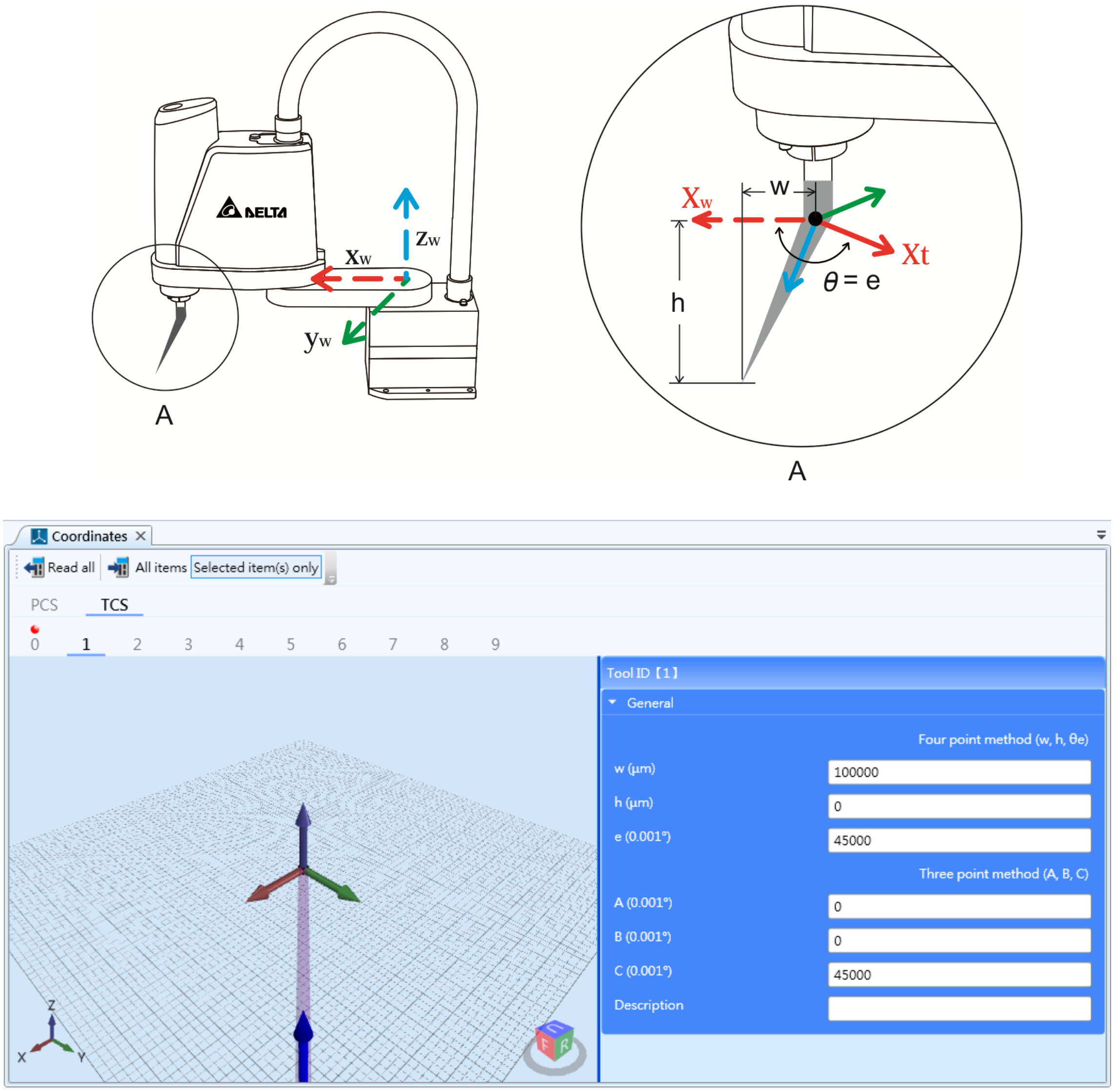


Figure 7.4.1.1 Point teaching example of TCS

7

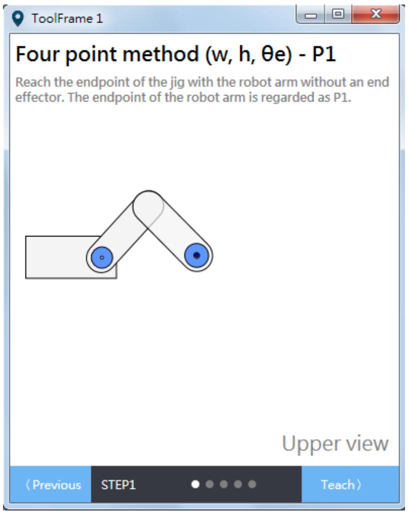
Direct entry method

1. Input the width of TCS, height of TCS and angle of TCS into the field of **w**, **h**, and **e** respectively (See the figure above.).
2. Then, click on the **Write** button. The MS system will automatically convert the coordinate system to MCS. And the point teaching of TCS is complete. If no error occurs, it means point teaching of TCS is done.

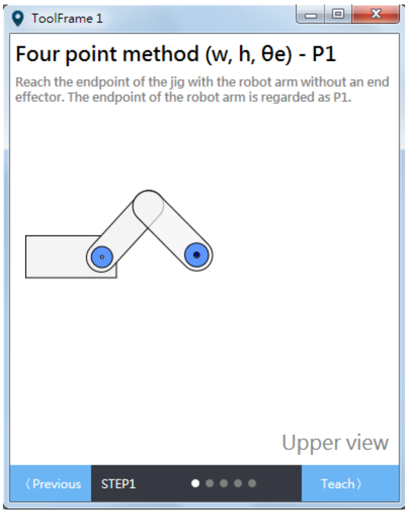
Four point method (w, h, e)

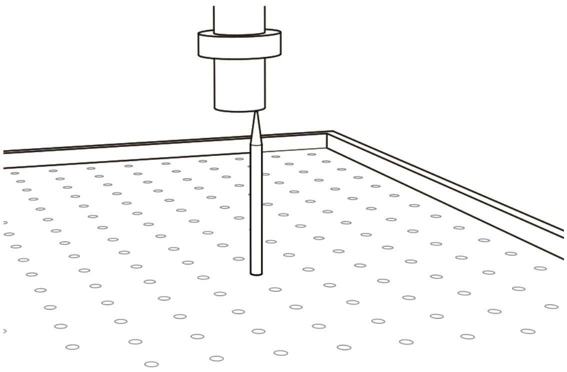
This method conducts point teaching by reaching the four endpoints of the jig with the robot arm. Thus, the jig should have a sharp point. Please select MCS as the coordinate system.

1. Click on **Four point method (w, h, e)** for enabling Teach function. See the figure below.

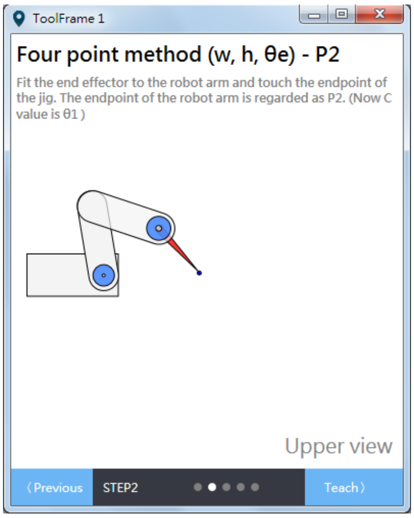


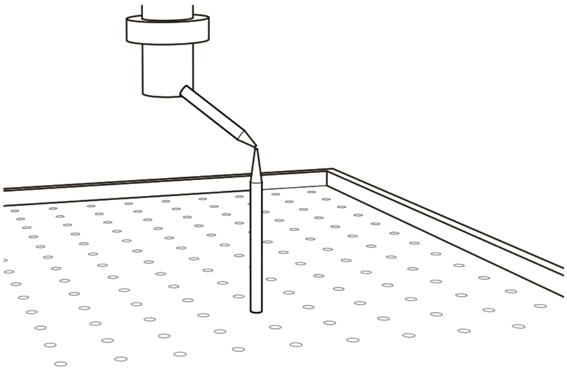
2. The first point is for measuring the tool height. Please reach the jig with the flange.



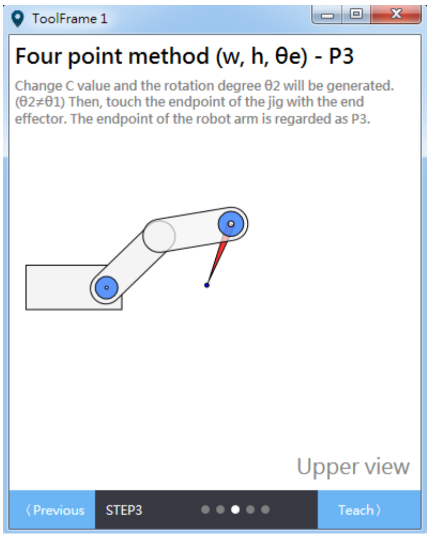


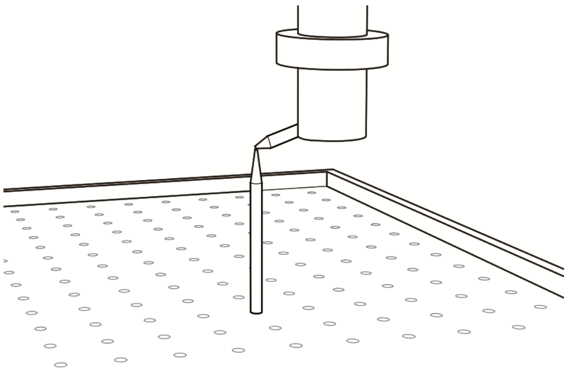
3. When the end effector is fitted to the robot arm, touch the endpoint of the jig with the end effector. Now, C value is θ1.



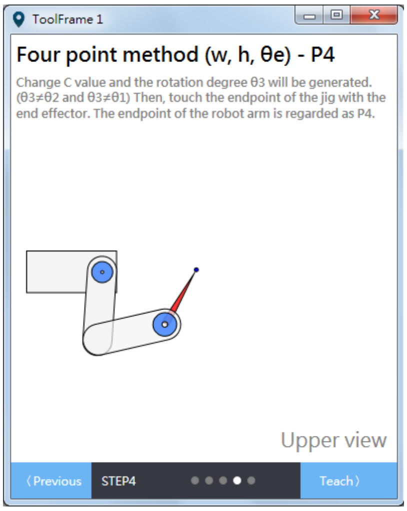


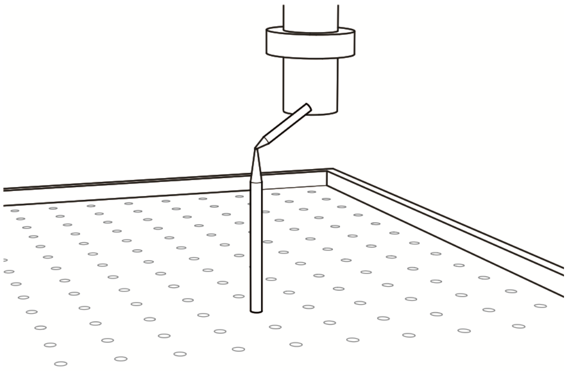
4. Change C value to θ2, which cannot equal to θ1. Then, touch the endpoint of the jig with the end effector.





5. Change C value to θ3, which cannot equal to θ2 and θ1. Then, touch the endpoint of the jig with the end effector.



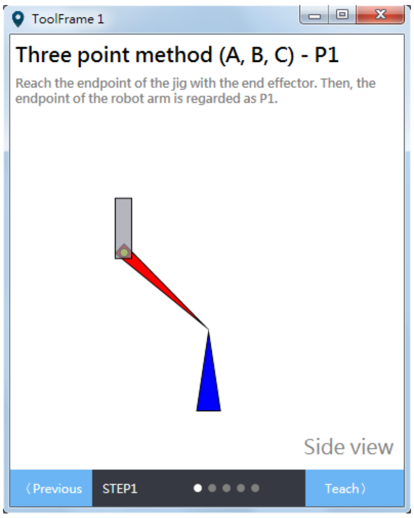


6. After calculation, you can acquire the tool dimension.

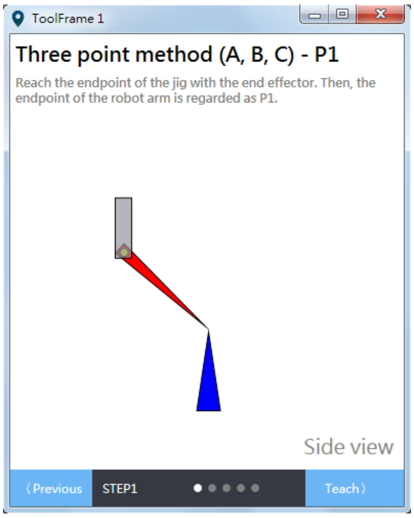
Three point method (A, B, C)

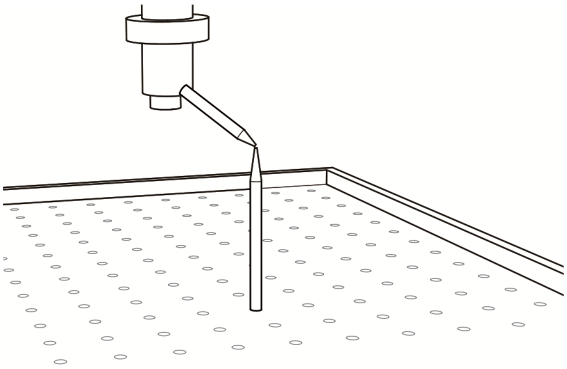
This method can be applied for teaching the orientation of TCS by reaching the three points of the jig with robot arm. The jig should have the sharp point. Please select MCS as the coordinate system.

1. Click on **Three point method (A, B, C)** for enabling Teach function. See the figure below.

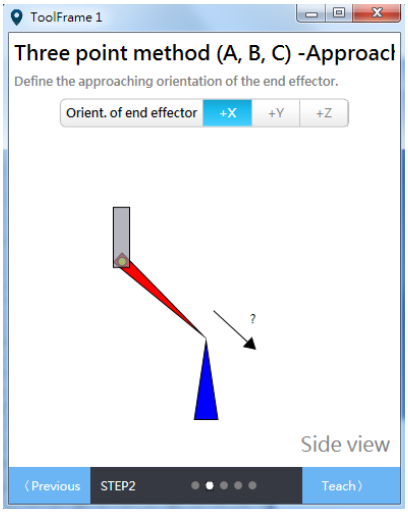


2. Please touch the endpoint of the jig with the end effector for the first point.

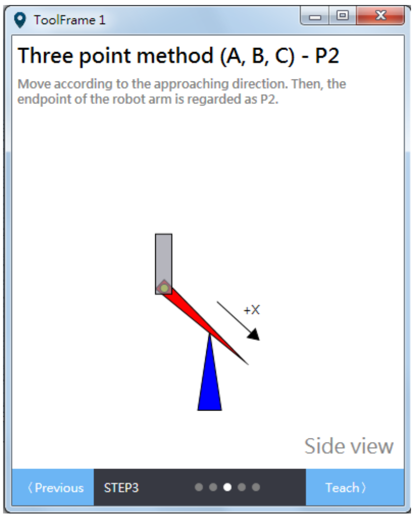


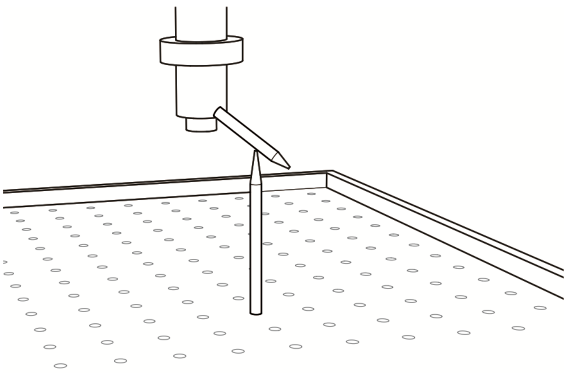


3. Select the orientation of TCS. It can be +X, +Y and +Z.

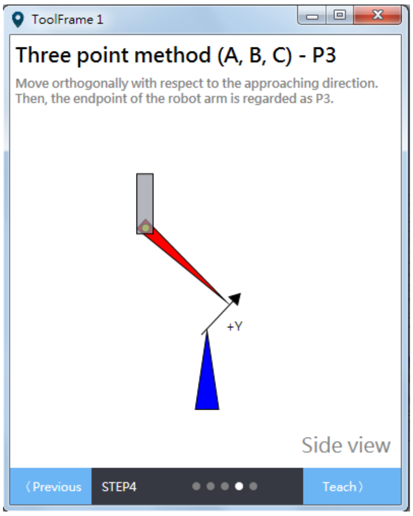


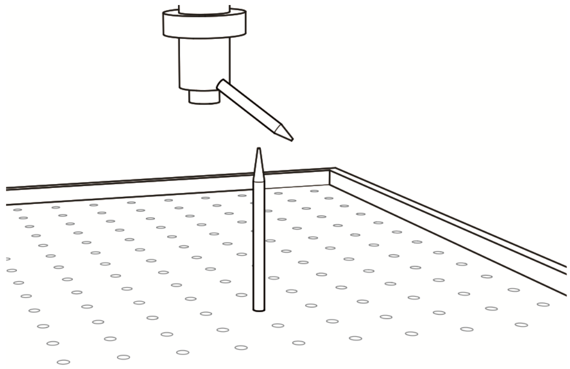
4. Use jog function to move the tool toward the specified direction. And touch the endpoint of the jig with the effector that approaches in this direction.





5. Then, move towards another specified direction of TCS. (If teaching this orientation is not necessary, you can move it to any position.)





6. After calculation, you can acquire the approaching orientation of TCS.

Please go to **Jog** and select the TCS number that just completed teaching, you can see the variation of Cartesian coordinates in monitoring section for verification.

7.4.2 Teach TCS in [Parameter]

Direct Entry Method

Please follow the steps below to complete teaching with direct entry method via setting parameters. See the example below.

1. Input the width (w), height (h) and angle (e):
   1.1. Set P2-07 to 0x00010013 (Ready to write the value of w).
   1.2. Set P2-08 to 1000.
   1.3. Set P2-07 to 0x01010013 (Ready to write the value of h).
   1.4. Set P2-08 to 2000.
   1.5. Set P2-07 to 0x02010013 (Ready to write the value of e).
   1.6. Set P2-08 to 10000.
   1.7. Set P2-06 to 0x01011132 (Completely writing the value of w, h, e).
2. Input TCS information and start to calculate:
   2.1. Set P2-06 to 0x01010231 (Write in and calculate data of TCS number 1)

If no error occurs, it means the point teaching of TCS is complete. You can switch to TCS and see the value of Cartesian coordinates in monitoring section for verification. Another way to switch the coordinate system to TCS is to set P2-06 to 0x01010050.

7.4.3 Use TCS in [Jog]

When point teaching of TCS is complete, you can start to use TCS. Please select **TCS** in **Jog mode**. Then, select the specified TCS which point teaching has been completed so that you can use the jog function on TCS. The setting of **TCS** will not influence the setting of **PCS**. Therefore, uses can select the number for **PCS**.

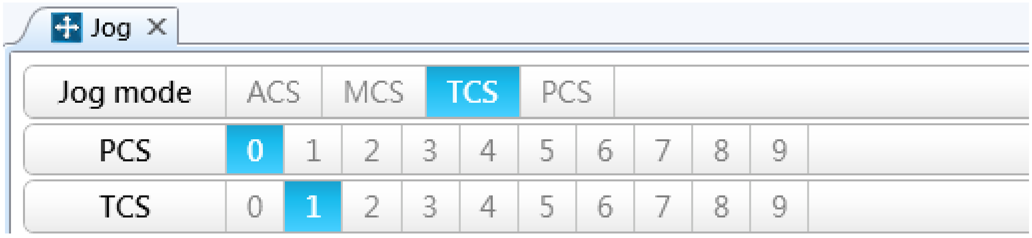


Figure 7.4.3.1 Select TCS in Jog mode

Now the coordinate system is switched to TCS. You can see the coordinate value in monitoring section. Please note that before selecting the number of coordinate system, make sure point teaching has been done. Otherwise, an error will occur.

When selecting **TCS** in **Jog mode**, only PCS and ACS are available in **Coordinate system** drop-down menu. Selecting PCS means the teaching point belongs to the applied PCS. And ACS is used for recording the current position of each axis (PUU).

7.4.4 Use TCS in [Script]

In **Script**, **`TF`** is used to select the TCS number. And TCS number 0 is defined as the TCS that has no tool fitted. As for **`Coord`**, it can be set to 0 (MCS) or 1 (PCS). If it is set to 0, **`UF`** is invalid. When it is set to 1, then you can input the TCS number in **`UF`**. Once the point is set, commands for converting coordinates is not required. When the program runs to the point, the system will automatically switch to the specified coordinate system.

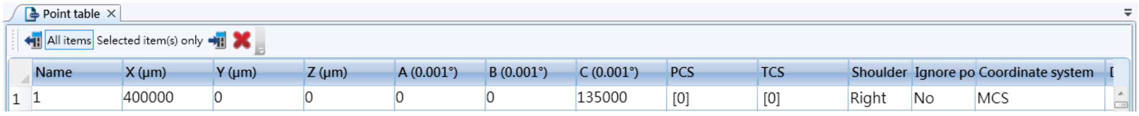
Please note that before using PCS, you need to complete the point teaching of TCS first. Users still can input the number of coordinate system in the script. However, if the coordinate system has not carried out point teaching, an error will occur. See the example for point declaration of TCS:

```
P["example"].x = -25000
P["example"].y = 40000
P["example"].z = -100000
P["example"].a = 0.0
P["example"].b = 0.0
P["example"].c = 0.0
P["example"].Elbow = HAND_RIGHT
P["example"].PS = 0
P["example"].UF = 1        -- It means PCS number 1 is selected.
P["example"].TF = 1        -- It means TCS number 1 is selected.
```

```
P["example"].Coord = 1      -- It can be set to 0 or 1.
```

7.4.5 Use TCS in [Point table]

Apart from **Script**, **Point table** can also be used to set up PCS. Go to **Point table** and directly select the number of TCS and PCS for the point. Then, select **MCS** or **PCS** as the coordinate system. Click on **Write** button to complete the setting.



	Name	X (μm)	Y (μm)	Z (μm)	A (0.001°)	B (0.001°)	C (0.001°)	PCS	TCS	Shoulder	Ignore po	Coordinate system	[
1	1	400000	0	0	0	0	135000	[0]	[0]	Right	No	MCS	

Figure 7.4.5.1 Point table

Make sure point teaching is correctly carried out and complete. Otherwise, an error will occur when the system runs to that point.

7.4.6 Use TCS in [Parameter]

Read the data created by Teach function of TCS (Direct entry method)

For example, accessing the data of TCS number 1 that created by Teach function:

1. Set P2-06 to 0x01001132 (It is for accessing the data of TCS number 1).
2. Set P2-07 to 0x00000013 (It is for accessing index 0 of data array)
3. Read the value of P2-09. The returned value is **w** (um).
4. Set P2-07 to 0x01000013 (It is for accessing index 1 of data array)
5. Read the value of P2-09. The returned value is **h** (um).
6. Set P2-07 to 0x02000013 (It is for accessing index 2 of data array)
7. Read the value of P2-09. The returned value is **e** (0.001°).

Switch the coordinate system:

Example: switch to TCS number 1

Set P2-06 to 0x01010030 to switch to TCS number 1.

Switch to TCS

Example: enable TCS number 1

Set P2-06 to 0x01010050 to switch the coordinate system to PCS number 1.

Clear TCS data

Example: clear TCS number 1

Set P2-06 to 0x01010F31 to clear the data of TCS number 1.

7.5 Axis coordinate system

Axis coordinate system (ACS) is created by each axis of the robot arm. Take SCARA supported by MS as the example, ACS is created by the four axes of SCARA. In DRAS, you can use **ACS** in **Jog**, **Script**, **Point table** and **Parameter** panels.

7.5.1 Use ACS in [Jog]

Select **ACS** in **Jog mode** to switch the coordinate system to ACS. However, you will be unable to select the number of **PCS** and **TCS**. And the displayed value of PCS and TCS in monitoring section is from the previous selection.

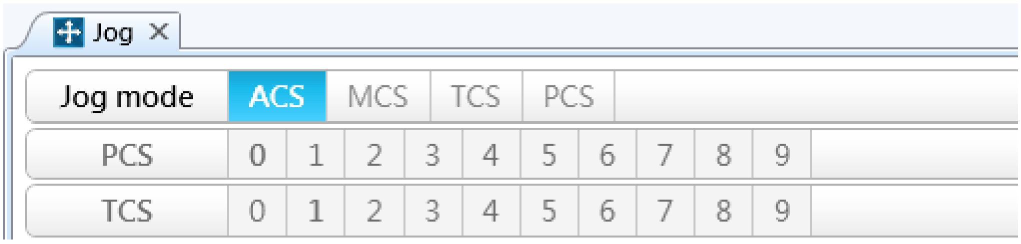


Figure 7.5.1.1

When selecting **ACS** in **Jog mode**, only **PCS** and **ACS** are available in **Coordinate system** drop-down menu. Selecting PCS means the teaching point belongs to the applied PCS. And ACS is used for recording the current position of each axis (PUU).

7.5.2 Use ACS in [Script]

In **Script**, when **Coord** is set to 3, it means the coordinate system is set to ACS. See the example of the point declaration of ACS below. When **Coord** is set to 3, **UF** and **TF** are in valid.

Example:

```
P[“example”].x = -25000
P[“example”].y = 40000
P[“example”].z = -100000
P[“example”].a = 0.0
P[“example”].b = 0.0
P[“example”].c = 0.0
P[“example”].Elbow = HAND_RIGHT
P[“example”].PS = 0
P[“example”].UF = 0
P[“example”].TF = 0
P[“example”].Coord = 3     -- It means to select ACS as the coordinate system
```

7

7.5.3 Use ACS in [Point table]

Users can set the coordinate system of each point in **Point table**. Go to **Point table** and select **ACS** in **Coordinate system**. Then, click on the **Write** button. You will see the background color becomes light purple. When you select **ACS** for **Coordinate system**, PCS and TCS value will be invalid.

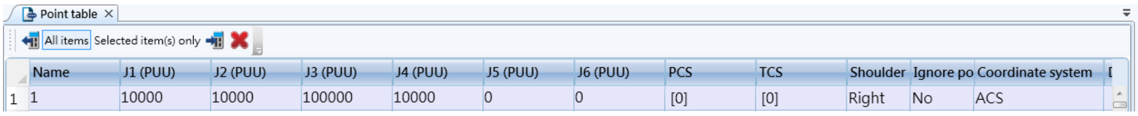


Figure 7.5.3.1 Use ACS in Point table

7.5.4 Use ACS in [Parameter]

In **Parameter**, you can access the feedback information of ACS.

Access the motor feedback position in ACS (PUU)

1. Set P2-06 to 0x00000014 (This is for accessing the feedback data of ACS.)
2. Set P2-07 to 0x00000013 (This is for accessing index 0 of data array)
3. Read the value of P2-09. The returned value is the feedback position of the first axis (PUU).
4. Set P2-07 to 0x01000013 (This is for accessing index 1 of data array).
5. Read the value of P2-09. The returned value is the feedback position of the second axis (PUU).
6. Set P2-07 to 0x02000013 (This is for accessing index 2 of data array).
7. Read the value of P2-09. The returned value is the feedback position of the third axis (PUU).
8. Set P2-07 to 0x03000013 (This is for accessing index 3 of data array).
9. Read the value of P2-09. The returned value is the feedback position of the fourth axis (PUU).

Access the motor command position in ACS (PUU)

1. Set P2-06 to 0x00000114 (This is for accessing ACS command.)
2. Set P2-07 to 0x00000013 (This is for accessing index 0 of data array).
3. Read the value of P2-09. The returned value is the command position of the first axis (PUU).
4. Set P2-07 to 0x01000013 (This is for accessing index 1 of data array).
5. Read the value of P2-09. The returned value is the command position of the second axis (PUU).
6. Set P2-07 to 0x02000013 (This is for accessing index 2 of data array).
7. Read the value of P2-09. The returned value is the command position of the third axis (PUU).
8. Set P2-07 to 0x03000013 (This is for accessing index 3 of data array).
9. Read the value of P2-09. The returned value is the command position of the fourth axis (PUU).

Access the motor feedback position in ACS (angle)

1. Set P2-06 to 0x00000314 (This is for accessing the feedback data of ACS.)
2. Set P2-07 to 0x00000013 (This is for accessing index 0 of data array).
3. Read the value of P2-09. The returned value is the feedback position of the first axis (0.001°).
4. Set P2-07 to 0x01000013 (This is for accessing index 1 of data array).
5. Read the value of P2-09. The returned value is the feedback position of the second axis (0.001°).
6. Set P2-07 to 0x02000013 (This is for accessing index 2 of data array).
7. Read the value of P2-09. The returned value is the feedback position of the third axis (0.001°).
8. Set P2-07 to 0x03000013 (This is for accessing index 3 of data array).
9. Read the value of P2-09. The returned value is the feedback position of the fourth axis (0.001°).

Access the motor command position in ACS (angle)

1. Set P2-06 to 0x00000414 (This is for accessing ACS command.)
2. Set P2-07 to 0x00000013 (This is for accessing index 0 of data array).
3. Read the value of P2-09. The returned value is the command position of the first axis (PUU).
4. Set P2-07 to 0x01000013 (This is for accessing index 1 of data array).
5. Read the value of P2-09. The returned value is the feedback position of the second axis (0.001°).
6. Set P2-07 to 0x02000013 (This is for accessing index 2 of data array).
7. Read the value of P2-09. The returned value is the feedback position of the third axis (0.001°).
8. Set P2-07 to 0x03000013 (This is for accessing index 3 of data array).
9. Read the value of P2-09. The returned value is the feedback position of the fourth axis (0.001°).

(This page is intentionally left blank.)

Parameters

8

This chapter introduces the motion control functions supported by MS controller and provides descriptions of parameters setting, including the assignment of digital input (DI) and digital output (DO).

8.1 Parameter definition

Parameters of MS controller can be categorized by controller and servo drive groups:

Controller parameters are divided into four groups. The first character after the start code P is the group character and the following two characters are the parameter character. As for the communication address, it is the combination of group number along with the two-digit number in hexadecimal. The definition of parameter groups is as follows:

Group 0: Monitoring parameters (example: P0-xx)
Group 1: Setting parameters (example: P1-xx)
Group 2: Application parameters (example: P2-xx)
Group 3: Communication parameters (example: P3-xx)

Servo drive parameters are divided into seven groups. The first character after the start code P is the group character and the following two characters are the parameter character. As for the communication address, it is the combination of group number along with the two-digit number in hexadecimal. The definition of parameter groups is as follows:

Group 0: Monitoring parameters (example: P0-xx)
Group 1: Basic parameters (example: P1-xx)
Group 2: Extension parameters (example: P2-xx)
Group 3: Communication parameters (example: P3-xx)
Group 4: Diagnosis parameters (example: P4-xx)
Group 5: Motion control parameters (example: P5-xx)
Group 6: PR parameters (example: P6-xx)

Control mode description:

Sz: Speed control
Tz: Torque control
DMC: DMCNET control

Special symbol description:

Icon of parameter property	Description
	Read-only parameter. Users can only read the status.
	Parameter cannot be set when it is in servo on status.
	Parameter setting will be valid after re-power on the servo drive.
	When the power is off, the parameter value will be the default value.
	It is shared by multiple axes.

8

8.2 List of controller parameters

Monitoring parameters			
Parameter No.	Function	Default value	Unit
P0-00	Controller firmware version	Factory setting	-
P0-01	Alarm code display of the controller	-	-
P0-02	Firmware version of motion control	Factory setting	-
P0-03	Status display of robot arm	-	-
P0-04	Monitoring variables setting	-	-
P0-05	Monitoring station number setting	-	-
P0-06	Monitoring channel setting	-	-
P0-07	Available space of Script	-	kBytes
P0-08	Power on hours count	-	Hour
P0-09	PLC status display	-	-

Setting parameters			
Parameter No.	Function	Default value	Unit
P1-00	Robot type setting	0x0000	-
P1-01	PLC mode setting	-	-
P1-02	PLC program mode setting	-	-
P1-06	Change password of Script	0x123456	-
P1-07	Mask setting of digital output	0x0fff0000	-
P1-08	Advanced function setting	-	-
P1-09	Window for setting the advanced function	-	-
P1-10	Window for setting the advanced function (16-bit)	-	-

Application parameters			
Parameter No.	Function	Default value	Unit
P2-00	Command for setting mechanism parameters	-	-
P2-01	Data array index for mechanism parameters	0	-
P2-02	Data array window for writing mechanism parameters	0	-
P2-03	Data array window for reading mechanism parameters	0	-
P2-06	Command for setting coordinate system parameters	-	-
P2-07	Data array index for coordinate system parameters	0	-

8

Application parameters			
Parameter No.	Function	Default value	Unit
P2-08	Data array window for writing coordinate system parameters	0	-
P2-09	Data array window for reading coordinate system parameters	0	-
P2-12	Index of (CVT) parameter	0	-
P2-13	Window for writing conveyor tracking parameters	0	-
P2-14	Window for reading conveyor tracking parameters	0	-

Communication parameters			
Parameter No.	Function	Default value	Unit
P3-00	Address setting	0x0001	-
P3-01	Transmission speed	0x0003	bps
P3-02	Communication protocol	0x0006	-
P3-05	Communication mechanism	0x0000	-
P3-06	USB function switch	0x0000	-
P3-08	Monitoring mode	0x0000	-
P3-20	EtherNet network state	-	-
P3-21	EtherNet IP address	-	-
P3-22	EtherNet subnet mask	-	-
P3-23	EtherNet default gateway	-	-
P3-24	EtherNet network setting	0x00000000	-
P3-25	EtherNet IP address setting	0xC0A80101	-
P3-26	EtherNet subnet mask setting	0xFFFFFF00	-
P3-27	EtherNet default gateway setting	0xC0A80101	-
P3-29	DMCNET function setting	0x0001	-
P3-30	DMCNET function control	0x0000	-
P3-31	DMCNET slave status (No.1)	0x00000000	-
P3-32	DMCNET slave status (No.2)	0x00000000	-
P3-33	DMCNET slave status (No.3)	0x00000000	-
P3-34	DMCNET slave status (No.4)	0x00000000	-
P3-35	DMCNET slave status (No.5)	0x00000000	-
P3-36	DMCNET slave status (No.6)	0x00000000	-
P3-37	DMCNET slave status (No.7)	0x00000000	-
P3-38	DMCNET slave status (No.8)	0x00000000	-

Communication parameters			
Parameter No.	Function	Default value	Unit
P3-39	DMCNET slave status (No.9)	0x00000000	-
P3-40	DMCNET slave status (No.10)	0x00000000	-
P3-41	DMCNET slave status (No.11)	0x00000000	-
P3-42	DMCNET slave status (No.12)	0x00000000	-

Icon of parameter property	Description
	Read-only parameter. Users can only read the status.
	Parameter cannot be set when it is in servo on status.
	Parameter setting will be valid after re-power on the servo drive.
	When the power is off, the parameter value will be the default value.
	It is shared by multiple axes.

8

8.3 Description of controller parameters

P0-xx Monitoring parameters

P0-00	**Controller firmware version**		**Address: 0000H 0001H**
Default:	Factory setting	Control mode:	ALL
Unit:	-	Range:	-
Format:	DEC	Data size:	16-bit

Settings:

It displays the firmware version of the controller.

P0-01	**Alarm code display of the controller (7-segment display)**		**Address: 0002H 0003H**
Default:	-	Control mode:	ALL
Unit:	-	Range:	32 Bits; Set P0-01 to 0 can clear the alarm.
Format:	HEX	Data size:	32-bit

Settings:

It displays the alarm code. Please refer to Chapter 11 for detailed information.

Format of the alarm code				
INDEX (16 bits)				ERROR CODE (16 Bits)
U	Z	Y	X	ERROR CODE (WORD)
NO		Reserved (0x0x)	TYPE	

NO: Group or Axis NO.

TYPE:

0x0: Controller

0x1: Group

0x2: Axis

0x3: User

0x4 ~ 0xF: System reserved

P0-02	**Firmware version of motion control**		**Address: 0004H 0005H**
Default:	Factory setting	Control mode:	ALL
Unit:	-	Range:	-
Format:	DEC	Data size:	32-bit

Settings:

It displays the firmware version of the motion control.

P0-03	Status display of robot arm		Address: 0006H 0007H
Default:	-	Control mode:	ALL
Unit:	-	Range:	-
Format:	HEX	Data size:	16-bit

Settings:

It displays the status of robot arm.

Format: 0xUZYX

Robot type	Setting value	Description
SCARA	0x4400	4-axis Scara robot
	0x4500	5-axis Scara robot
DELTA	0x3300	3-axis parallel robot arm (rotary drive type)
	0x3301	3-axis parallel robot arm (linear drive type 1)
	0x3302	3-axis parallel robot arm (linear drive type 2)
	0x3400	4-axis parallel robot arm (rotary drive type)
	0x3401	4-axis parallel robot arm (linear drive type 1)
	0x3402	4-axis parallel robot arm (linear drive type 2)

P0-04	Monitoring variables setting		Address: 0008H 0009H
Default:	-	Control mode:	ALL
Unit:	-	Range:	-
Format:	HEX	Data size:	32-bit

Settings:

<table>
<tr><td rowspan="2">Format
Description</td><td>H</td><td>G</td><td>F</td><td>E</td><td>D</td><td>C</td><td>B</td><td>A</td></tr>
<tr><td>31 ~ 28</td><td>27 ~ 24</td><td>23 ~ 20</td><td>19 ~ 16</td><td>15 ~ 12</td><td>11 ~ 8</td><td>7 ~ 4</td><td>3 ~ 0</td></tr>
<tr><td rowspan="2">Monitoring variables [Monitor ID]</td><td rowspan="2">E</td><td rowspan="2">0</td><td rowspan="2" colspan="2">Group or axis number [GrpAxNo]</td><td>0</td><td>0</td><td colspan="2">Axis monitoring ID [ID]</td></tr>
<tr><td colspan="2">Group monitoring ID [ExtraID]</td><td>F</td><td>F</td></tr>
<tr><td>PLC M3 section</td><td>F</td><td>0</td><td colspan="6">SV section: 00 0000 ~ 00 03FF
DV section: 00 0400 ~ 00 3FFF
DH section: 00 4000 ~ 01 2FFF</td></tr>
</table>

P0-05	Monitoring station number setting		Address: 000AH 000BH
Default:	-	Control mode:	ALL
Unit:	-	Range:	-
Format:	HEX	Data size:	16-bit

Settings:

This function is not available now.

8

P0-06	Monitoring channel setting		Address: 000CH 000DH
Default:	-	Control mode:	ALL
Unit:	-	Range:	-
Format:	DEC	Data size:	16-bit

Settings:

This is for setting monitoring variable of the corresponded channel. Before setting P0-04, please select the channel to be monitored first.

Range	0 ~ 100	101 ~ 116
Function	N/A	Monitoring mode for debugging 16 channels (Set P3-08 to 1 for enabling this function)

P0-07	Available space of Script		Address: 000EH 000FH
Default:	-	Control mode:	ALL
Unit:	kBytes	Range:	-
Format:	DEC	Data size:	16-bit

Settings:

It displays the available space of FTP in **Script**.

P0-08	Power on hours count		Address: 0010H 0011H
Default:	-	Control mode:	ALL
Unit:	Hour	Range:	-
Format:	DEC	Data size:	16-bit

Settings:

It displays the total controller on-time.

P0-09	PLC status display		Address: 0012H 0013H
Default:	-	Control mode:	ALL
Unit:		Range:	-
Format:	DEC	Data size:	16-bit

Settings:

It displays the current status of PLC.

Number	State	Number	State	Number	State
0	PlcOn	3	PlcRunning	6	PlcStopping
1	PlcLoading	4	PlcHaltRequested	7	PlcStop
2	PlcStarting	5	PlcHalt	8	PlcResetting

P1-xx Setting parameters

P1-00 ⏻	Robot type setting		Address: 0100H 0101H
Default:	-	Control mode:	ALL
Unit:	-	Range:	-
Format:	HEX	Data size:	16-bit

Settings:

It is for setting the robot type that connects to the controller.

Robot type	Setting value	Description
SCARA	0x4400	4-axis Scara robot
	0x4500	5-axis Scara robot
DELTA	0x3300	3-axis parallel robot arm (rotary drive type)
	0x3301	3-axis parallel robot arm (linear drive type 1)
	0x3302	3-axis parallel robot arm (linear drive type 2)
	0x3400	4-axis parallel robot arm (rotary drive type)
	0x3401	4-axis parallel robot arm (linear drive type 1)
	0x3402	4-axis parallel robot arm (linear drive type 2)

P1-01 ⏻	PLC mode setting		Address: 0101H 0102H
Default:	-	Control mode:	ALL
Unit:	-	Range:	-
Format:	HEX	Data size:	16-bit

Settings:

It sets the PLC mode.

Setting value	Description
0	The system switches to the default PLC.
1	The system switches to the user-defined PLC.

P1-02 ⛉	PLC program mode setting		Address: 0103H 0104H
Default:	-	Control mode:	ALL
Unit:	-	Range:	-
Format:	HEX	Data size:	16-bit

Settings:

It sets the PLC mode. See detailed information below:

Setting value	State	Setting value	State
0	Stop PLC program	2	Warm start PLC program
1	Cold start PLC program	4	Hot start PLC program

8

P1-03~P1-05	Reserved

P1-06	Change password of Script		Address: 010CH 010DH
Default:	0x123456	Control mode:	ALL
Unit:	-	Range:	-
Format:	HEX	Data size:	32-bit

Settings:

It sets the FTP password of script.

Steps:

1. Input the old password.
2. Input the new password.
3. Input the new password again.
4. When the status code is 1, it means new password is set successfully. However, when it shows 0, it means password update failure.

FTP password format: It can be 6 to 8 characters and you can choose from a ~ f in small letter and 0 ~ 9. 0 cannot be set as the first character.

When changing the password via modbus communication, please input the password in hexadecimal format, such as 0x123456. And it will display ASCII code when log on ftp, such as 123456.

P1-07	Mask setting of digital output		Address: 010EH 0110FH
Default:	0x0FFF0000	Control mode:	ALL
Unit:	-	Range:	-
Format:	HEX	Data size:	32-bit

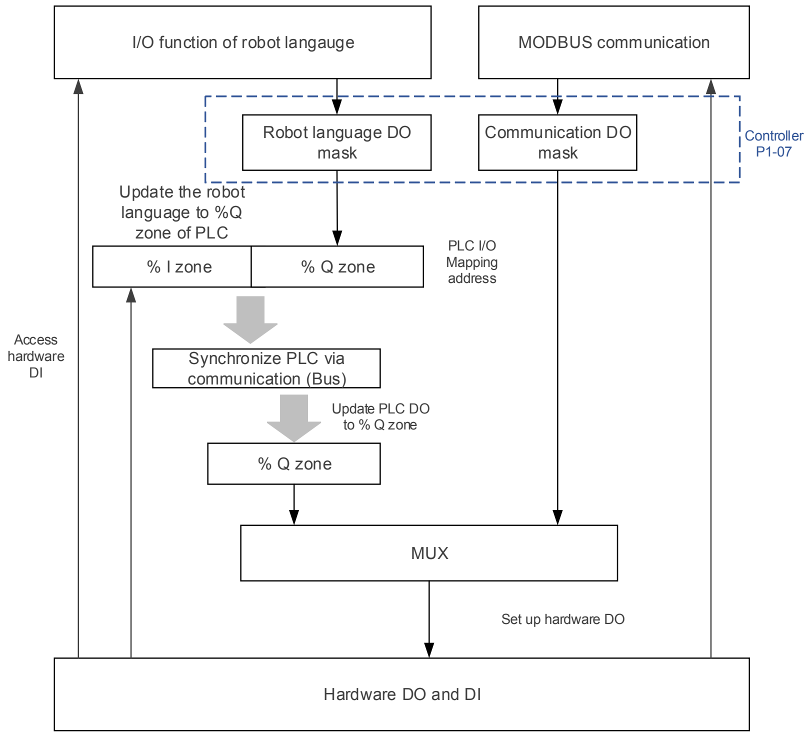
Settings:

Mask for digital output	
Robot language DO mask	Communication DO mask
16-bit	16-bit

To split one 32-bit word to two 16-bit words, it can be used to record 12 User DOs. 1 means DO is on; 0 means DO is off. System default is available for PLC and Lua DO (robot language DO mask) only. To use DO via Modbus communication, please enable the low word first.

Note: If the system default is not available for Modbus, when the startup setting value is 0x00000FFF, it will restore to the default (0x0FFF0000) automatically.

Mask setting of digital output:



P1-08 ⛉	Advanced function setting		Address: 0110H 0111H
Default:	-	Control mode:	ALL
Unit:	-	Range:	-
Format:	HEX	Data size:	32-bit

Settings:

It is for internal use only.

P1-09 ⛉	Window for setting the advanced function		Address: 0112H 0113H
Default:	-	Control mode:	ALL
Unit:	-	Range:	-
Format:	HEX	Data size:	32-bit

Settings:

It is for internal use only.

P1-10 ⛉	Window for setting the advanced function (16-bit)		Address: 0110H 0111H
Default:	-	Control mode:	ALL
Unit:	-	Range:	-
Format:	HEX	Data size:	16-bit

Settings:

It is for internal use only.

P2-xx Application parameters

P2-00	Command for setting mechanism parameters		Address: 0200H 0201H
Default:	-	Control mode:	ALL
Unit:	-	Range:	0x00000000 ~ 0xFFFFFFFF
Format:	HEX	Data size:	32-bit

Settings:

P2-00 ~ P2-03 are mechanism parameters, which setting will be valid only when they are set together.

P2-01	Data array index for mechanism parameters		Address: 0202H 0203H
Default:	-	Control mode:	ALL
Unit:	-	Range:	0x00000000 ~ 0xFFFFFFFF
Format:	HEX	Data size:	32-bit

Settings:

Format of the setting value: 0xDCBAUZYX

Description / Format		D	C	B	A	U	Z	Y	X
		Reserved	Index	Reserved		Command code			
Index of data array	Specify the index of data array to be read	-	[Note]		0	0x0013			
	Specify the index of data array to be written				1	0x0013			

Note: The size of data array is 8, please input the index of data array to be wrote and read.

P2-02	Data array window for writing mechanism parameters		Address: 0204H 0205H
Default:	-	Control mode:	ALL
Unit:	-	Range:	-2147483648 ~ +2147483647
Format:	DEC	Data size:	32-bit

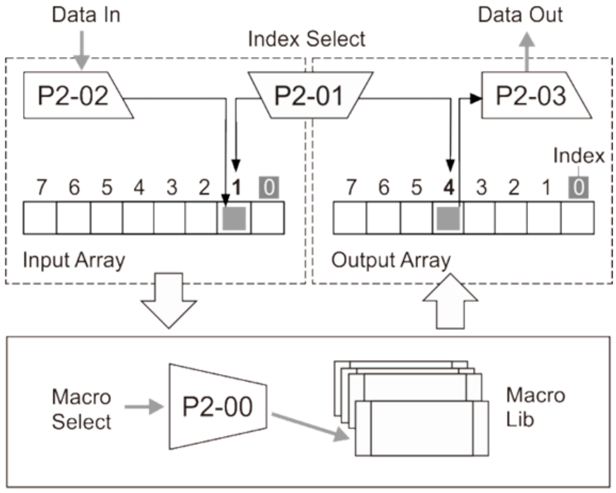
Settings:

Write the value to index 0 of data array. Please refer to the description of P2-01.

P2-03	Data array window for reading mechanism parameters		Address: 0206H 0207H
Default:	-	Control mode:	ALL
Unit:	-	Range:	-
Format:	DEC	Data size:	32-bit

Settings:

Read the value from index 0 of data array. Please refer to the description of P2-01.



Please refer to the table below for robot arm status and setting:

Item		D	C	B	A	U Z Y X	Format	Value / Description
Description		Group	Index	-	Read / Write 3*	Command code		
Status	Arm posture	1*	-	-	R	0x0031	BIN	Bit00: 0: right shoulder / 1: left shoulder Bit01: 0: right elbow (hand) / 1: left elbow (hand) Bit02: 0: non-flip (wrist) / 1: flip (wrist) Bit03: Reserved Bit04 ~ 07: It displays the group of PCS that is currently applied. [0D]: It is identical to MCS. [1d ~ 09d]: It represents PCS. Bit08 ~ 11: It displays the group of TCS that is currently applied. Bit09 ~ 15: Reserved.
	Applied coordinate system					0x0041	HEX	2h: Machine coordinate system 3h: Tool coordinate system 4h: Product coordinate system
Mechanism parameters	Max. speed in each direction (X, Y, Z) (mm/s)		2*		R/W	0x0012	DEC	Max. resultant velocity of robot arm (X, Y, Z)
	Gear ratio					0x0022	DEC	J1 ~ J6 gear ratio
	Length of robot arm (um)					0x0032	DEC	Length of the 1st~ 6th robot arm
	Speed reduction ratio					0x0042	DEC	J1 ~ J6 speed reduction ratio. **Please input 1 if no speed**

8

								reduction is required.
	Lead (um)					0x0052	DEC	J1 ~ J6 screw lead. **Please input 0 if no screw lead is required.**
	Software positive limit (PUU)					0x0062	DEC	J1 ~ J6 Software positive limit
	Software negative limit (PUU)					0x0072	DEC	J1 ~ J6 Software negative limit
	Software positive limit (0.001 degree)				R	0x0082	DEC	J1 ~ J6 Software positive limit
	Software negative limit (0.001 degree)					0x0092	DEC	J1 ~ J6 Software negative limit

Note:
1. Different axis can be formed multiple groups (Each group can be regarded as one robot arm). Please input the specified group in this field.
2. Setting value of each item differs from different robot types. Take 4-axis SCARA as the example. Gear ratio, speed reduction ratio, software positive limit and negative are set to 4. And the length of robot arm is set to 2.
3. 0 means “R-read”; 1 means “W-write”.

Example 1: Read robot arm’s posture of group 1.

Set P2-00 to 0x10000031. The returned value of P2-03 is 0010b, which represents right shoulder/left elbow (hand).

Example 2: Read robot arm’s coordinate system of group 1.

Set P2-00 to 0x10000041. The returned value of P2-03 is 0002h, which means MCS is the coordinate system.

Example 3: Set the length of S400 type robot arm to group 0.

The default length of arm 1 is 225000 um and 175032 um is the default length of arm 2. See the setting steps below.

1. Set P2-01 to 0x00010013.
2. Set P2-02 to 225000.
3. Set P2-00 to 0x00010032.
4. Set P2-01 to 0x00010013.
5. Set P2-02 to 175032.
6. Set P2-00 to 0x01010032.

Example 4: Read the length of S400 type robot arm in group 0. See the setting steps below.

1. Set P2-00 to 0x00000032.
2. The returned value of P2-03 is the length of arm 1. Its unit is um.
3. Set P2-00 to 0x01000032.
4. The returned value of P2-03 is the length of arm 2. Its unit is um.

P2-04~P2-05	Reserved

P2-06 ⛉	Command for setting coordinate system parameters		Address: 020CH 020DH
Default:	-	Control mode:	ALL
Unit:	-	Range:	0x00000000 ~ 0xFFFFFFFF
Format:	HEX	Data size:	32-bit

Settings:

P2-06 ~ P2-09 are for setting coordinate system. The following table shows the definition of returned error code.

Error code	Item	Description
0x10	Group does not exist	Please check if the group to be accessed exists.
0x20	Window is not supported	Please check if the window is in the command list.
0x30	The coordinate system group you tried to access exceeded the supported range	Please check if the group number is between 0 and 9.
0x40	Motor is not ready	Make sure the motor is correctly connected.
0x50	Group [0] cannot be taught	-
0x60	PCS-Three points are collinear when Three Point Method is applied	Please check if the teaching points are correct.
0x70	PCS-The angle is illegal when Direct Entry Method is applied	Please check if the input angle is correct.
0x80	-	-
0x90	PCS-An error has occurred when applying Three Point Method	-
0xA0	The reading / writing bit of the array is not correctly specified	Please check if the bit of read/write is correct.
0xB0	The axis number you tried to access exceeds the range	Please check the number of supporting grouped axes for the robot arm.
0xC0	Exceeds the array length	Please check if the array index is between 0 and 7.
0xD0	Coordinate system is not created	Please check if the group is correctly taught by the system.
0xE0	-	Reserved
0xF0	-	Reserved

P2-07 ⛉	Data array index for coordinate system parameters		Address: 020EH 020FH
Default:	-	Control mode:	ALL
Unit:	-	Range:	0x00000000 ~ 0xFFFFFFFF
Format:	HEX	Data size:	32-bit

Settings:

Returned error code: Please refer to P2-06.

8

Format \ Description	D	C	B	A	U Z Y X
	-	Index	-		Command code
Specify the array index to be read.	-	[Note]		0	0x0013
Specify the array index to be written.				1	0x0013

Note: The size of data array is 8. Please input the index of data array to be written/read.

P2-08 ☋	Data array window for writing coordinate system parameters		Address: 0210H 0211H
Default:	-	Control mode:	ALL
Unit:	-	Range:	-2147483648 ~ +2147483647
Format:	DEC	Data size:	32-bit

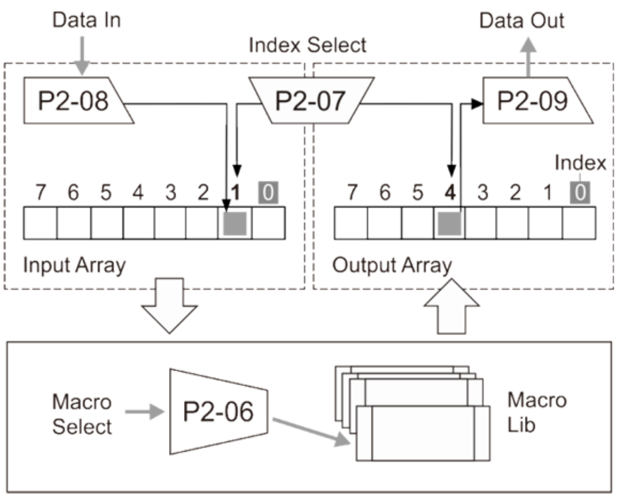
Settings:

Write the value to data array in accordance with the specified index. Returned error code: Please refer to P2-06.

P2-09 🔒	Data array window for reading coordinate system parameters		Address: 0212H 0213H
Default:	-	Control mode:	ALL
Unit:	-	Range:	-
Format:	DEC	Data size:	32-bit

Settings:

Read the value from data array in accordance with the specified index. Returned error code: Please refer to P2-06.



8

Item	Write / Read Description		Set commands via P2-06: D Group	C Index	B -	A Read / Write 3*	U Z Y X Command code	Set values via P2-08 Read values via P2-09
Teaching point of coordinate system	TCS-Direct Entry Method		1*	2*	-	R/W	0x1132	Tool size w, h, e
	PCS-Three Point Method						0x1142	Origin (X-, Y-, Z-coordinate)
							0x2142	Point X (X-, Y-, Z-coordinate)
							0x3142	Point Y (X-, Y-, Z-coordinate)
							0x4142	Scaling ratio Sx, Sy, Sz
	PCS-Direct Entry Method						0x1242	Angle A, B, C
Calculate the value according to the data created by Teach function / Save the data created by teach function	TCS-Direct Entry Method					W	0x0231	-
	PCS-Three Point Method						0x0141	
	PCS-Direct Entry Method						0x0241	
Switch the coordinate system	MCS			-		W	0x0020	-
	TCS			2*			0x0030	
Enable the coordinate system	PCS					W	0x0040	-
	TCS						0x0050	
Clear the data of coordinate system	TCS					W	0x0F31	-
	PCS						0x0F41	
Read coordinate system	ACS	Motor feedback position		-		R	0x0014	Save the position of each axis in data array according to different robot types (Unit: PUU)
		Motor command position					0x0114	
		Motor feedback position					0x0314	It returns different axial position in accordance with the robot type (Unit: 0.001 degree).
		Motor command position					0x0414	
	MCS	Feedback position of spatial coordinate system					0x0024	3 positions (um), (X, Y, Z) 3 angles (0.001 degree), (A, B, C)
		Command position of spatial coordinate system					0x0124	
	PCS	Feedback position of spatial coordinate system					0x0044	

Note:

1. Group can be formed by different axes. (Each group can be regarded as one robot arm). Please input the specified group in this field.
2. Input the specified group (max. 10). **Group 0 cannot be modified. However, the other 9 groups can be defined by users.**
3. 0 means “R-read”; 1 means “W-write”.

Attention: Before motion command is complete and motor stops, the coordinate system cannot be switched but read.

P2-10~P2-11	Reserved

8

P2-12 ~ P2-14 are conveyor tracking (CVT) commands

P2-12 ⛉	Index of CVT parameters		Address: 0218H 0219H
Default:	-	Control mode:	ALL
Unit:	-	Range:	0x00000000 ~ 0xFFFFFFFF
Format:	HEX	Data size:	32-bit

Settings:

Format: 0xDCBAUZYX

Index format:

D	C	B	A	U	Z	YX
-	-	-	-	-	CVT group number	Index number

Index definition

Index number	Item	Write/read description	Write or Read
0x00	Specify the controlling type that corresponded to CVT group	Y: 0 means it corresponds to single axis 1 means it corresponds to the group X: It represents the corresponded number of controlling type	W/R
0x01	Command source of conveyor speed	0: external encoder	W/R
0x02	Channel number of conveyor speed source	The number should be between 0 and 3.	W/R
0x03	Conversion coefficient for converting the speed unit of the conveyor (numerator)	-	W/R
0x04	Conversion coefficient for converting the speed unit of the conveyor (denominator)	It cannot be 0.	W/R
0x06	PCS number applied by CVT function	The number should be between 1 ~ 9.	W/R
0x07	CVT function switch	0: Disable; 1: Enable	W/R
0x08	Enable speed compensation for the conveyor	0: Enable; 1: Disable	W/R
0x0D	Offset distance in X-coordinates direction of camera coordinate system	Unit: um	W/R
0x0E	Offset distance in Y-coordinates direction of camera coordinate system	Unit: um	W/R
0x12	Rotation angle of axis Z-coordinates in camera coordinate system	Unit: 0.001°	W/R
0x13	Offset distance in X-coordinates direction of template coordinate system	Unit: um	W/R
0x14	Offset distance in Y-coordinates direction of template coordinate system	Unit: um	W/R
0x18	Rotation angle of axis Z-coordinates in template coordinate system	Unit: 0.001°	W/R
0x19	Offset distance in X-coordinates direction of workpiece coordinate system	Unit: um	W/R
0x1A	Offset distance in Y-coordinates direction of workpiece coordinate system	Unit: um	W/R
0x1E	Rotation angle of axis Z in workpiece coordinate system	Unit: 0.001°	W/R
0x1F	Calculated value of conveyor speed (Calculated conveyor speed)	Unit: um/ms	R
0x20	Reset CVT parameters	Rest parameters of CVT group	W

P2-13	Window for writing conveyor tracking parameters		Address: 021AH 021BH
Default:	-	Control mode:	ALL
Unit:	-	Range:	-2147483648 ~ +2147483647
Format:	HEX	Data size:	32-bit

Settings:

This is a write-in parameter.

Parameters write-in format: Please refer to index definition of P2-12.

P2-14	Window for reading conveyor tracking parameters		Address: 021CH 021DH
Default:	-	Control mode:	ALL
Unit:	-	Range:	-2147483648 ~ +2147483647
Format:	DEC	Data size:	32-bit

Settings:

Read parameters.

Parameters reading format: Please refer to index definition of P2-12.

See the following example for setting CVT parameters.

Example 1: This is for setting number 0 of CVT to the robot arm (Controlling type 0)

1. Set P2-12 to 0x00000000.
2. Set P2-13 to 0x00000010 (10 refers to number 0 of controlling type)

Example 2: This is for setting the command source of conveyor speed and channel 0 as external encoder.

1. Set P2-12 to 0x00000001.
2. Set P2-13 to 0x00000000.
3. Set P2-12 to 0x00000002.
4. Set P2-13 to 0x00000000.

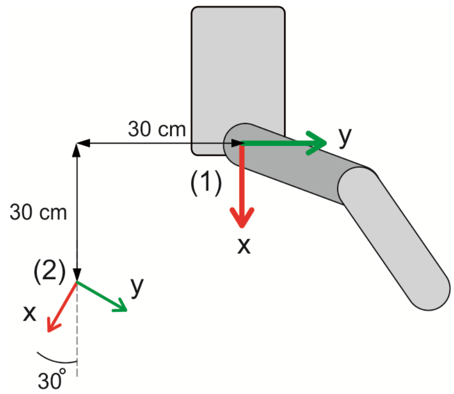
Example 3: This is for setting parameters of camera coordinate system. (See the figure below.)

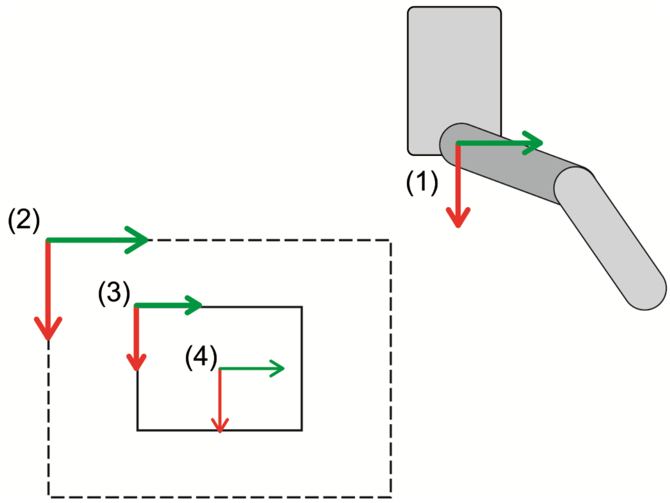
1. Set P2-12 to 0x0000000D.
2. Set P2-13 to 300000 (300000 um in X-axis direction).
3. Set P2-12 to 0x0000000E.
4. Set P2-13 to 300000 (300000 um in Y-axis direction).
5. Set P2-12 to 0x00000012.
6. Set P2-13 to 30000 (Axis Z rotates 30 degrees).

Workpiece coordinate system is relative to template coordinate system;

Template coordinate system is relative to camera coordinate system (It is usually provided by DMV);

Camera coordinate system is relative to machine coordinate system.





(1) Machine coordinate system (2) Camera coordinate system (3) Product coordinate system (4) Template coordinate system

P3-xx Communication parameters

P3-00 ⏻	Address setting		Address: 0300H 0301H
Default:	01	Control mode:	ALL
Unit:	-	Range:	0x01 ~ 0xF7
Format:	HEX	Data size:	16-bit

Settings:

The communication address setting is divided into Y, X (hexadecimal):

	0	0	Y	X
Range	-	-	0 ~ F	0 ~ F

When using Modbus to communicate, this station number is the absolute address of the controller. This controller occupies three ADR, which are P3-00, (P3-00) + 1 and (P3-00) + 2. When the ADR is P3-00, it means to access controller parameters; when ADR is (P3-00) + 1, it means to access PLC; when ADR is (P3-00) +2, it means to access error log.

When the communication address is set to 0xFF, the controller will automatically reply and receive data regardless of the address. However, P3-00 cannot be set to 0xFF. Since (P3-00) + 1 has already exceeded ADR setting range, when P3-00 is set to 0xF7 (247), it cannot access PLC communication and error log.

P3-01	Transmission speed		Address: 0302H 0303H
Default:	3	Control mode:	ALL
Unit:	Bps	Range:	0 ~ 5
Format:	HEX	Data size:	16-bit

Settings:

Definition of each setting value:

0: 4800	1: 9600	2: 19200
3: 38400	4: 57600	5: 115200

P3-02	Communication protocol		Address: 0304H 0305H
Default:	6	Control mode:	ALL
Unit:	-	Range:	0 ~ 8
Format:	HEX	Data size:	16-bit

Settings:

Definition of each setting value:

0: 7, N, 2 (MODBUS, ASCII)	1: 7, E, 1 (MODBUS, ASCII)	2: 7, O, 1 (MODBUS, ASCII)
3: 8, N, 2 (MODBUS, ASCII)	4: 8, E, 1 (MODBUS, ASCII)	5: 8, O, 1 (MODBUS, ASCII)
6: 8, N, 2 (MODBUS, RTU)	7: 8, E, 1 (MODBUS, RTU)	8: 8, O, 1 (MODBUS, RTU)

P3-03 ~ P3-04	Reserved

P3-05	Communication mechanism		Address: 030AH 030BH
Default:	0	Control mode:	ALL
Unit:	-	Range:	0 ~ 1
Format:	HEX	Data size:	16-bit

Settings:

The communication mechanism setting is divided into Y, X (hexadecimal):

	0	0	Y	X
Function	-	-	Master and slave station setting	Communication interface
Range	-	-	0 ~ 1	0 ~ 1

Definition of Y value:

0: Modbus Slave	1: Modbus Master

Definition of X value:

0: RS-232	1: RS-485

P3-06 ⏻	USB function switch		Address: 030CH 030DH
Default:	0	Control mode:	ALL
Unit:	-	Range:	0 ~ 1
Format:	HEX	Data size:	16-bit

Settings:

Use USB port as Serial or EtherNet

Function of USB port:

0: USB-Serial

1: USB-EtherNet

Note:
1. Baud rate of USB-Serial is 921600 bps.
2. IP of USB-EtherNet is 192.168.240.1. DHCP server is provided, thus, the PC IP will be 192.168.240.100.
3. Please refer to Appendix C, Install USB-Serial driver.

P3-07	Reserved

8

P3-08	Monitoring mode		Address: 030CH 030DH
Default:	0	Control mode:	ALL
Unit:	-	Range:	0x0 ~ 0xF3
Format:	HEX	Data size:	16-bit

Settings:

Setting of monitoring mode is divided into L and H (hexadecimal):

Digit	-	-	L	H
Function	-	-	Monitoring time for debugging	Monitoring mode
Range	0	0	0 ~ F	0 ~ 3

Definition of each setting value:

Definition of setting value H:

0: Disable monitoring function.
1: Monitoring for debugging. Its sampling time is determined by L, which monitors 8 channels, 32 bits or 16 channels, 16 bits.
2: High-speed monitoring. Its sampling frequency is 2 K, which monitors 8 channels, 32 bits or 16 channels, 16 bits.
1: High-speed monitoring. Its sampling frequency is 4 K, which monitors 4 channels, 32 bits or 8 channels, 16 bits.

L: It is the sampling time for monitoring. Unit: ms.

It will record one message every 2^L ms for controller to perform status analysis. Each monitoring data includes 8-channel data (32-bit x 8) or 16-channel data (16-bit x 16).

L will be valid only when H is set to 1.

P3-09~P3-19	Reserved

P3-20	EtherNet network status		Address: 0328H 0329H
Default:	-	Control mode:	ALL
Unit:	-	Range:	-
Format:	HEX	Data size:	32-bit

Settings: EtherNet network status can be divided into Z, Y and X (hexadecimal):

	0	0	0	0	0	Z	Y	X
Function	-	-	-	-	-	DHCP state	IP setup	Cable state
Range	-	-	-	-	-	0 ~ 2	0 ~ 1	0 ~ 1

Definition of Z value:

0: EtherNet is not ready	1: IP is acquired.	2: IP is not acquired.

Definition of Y value:

0: Static IP	1: DHCP

Definition of X value:

0: Cable plugged	1: Cable unplugged

Note: Setting of Z value is valid only when Y value is set to 1.

P3-21 🔒	EtherNet IP address		Address: 032AH 032BH
Default:	-	Control mode:	ALL
Unit:	-	Range:	-
Format:	HEX	Data size:	32-bit

Settings:

It displays EtherNet IP address (in hexadecimal format). If the IP is 192.168.1.1, then it will display 0xC0A80101.

P3-22 🔒	EtherNet subnet mask		Address: 032CH 032DH
Default:	-	Control mode:	ALL
Unit:	-	Range:	-
Format:	HEX	Data size:	32-bit

Settings:

It displays EtherNet subnet mask (in hexadecimal format).

P3-23 🔒	EtherNet default gateway		Address: 032EH 032FH
Default:	-	Control mode:	ALL
Unit:	-	Range:	-
Format:	HEX	Data size:	32-bit

Settings:

It displays EtherNet default gateway (in hexadecimal format).

P3-24	EtherNet network setting		Address: 0330H 0331H
Default:	0	Control mode:	ALL
Unit:	-	Range:	0x00 ~ 0x11
Format:	HEX	Data size:	32-bit

Settings:

8

The EtherNet network setting is divided into Y, X (hexadecimal):

	0	0	0	0	0	0	Y	X
Function	-	-	-	-	-	-	IP setup	Trigger
Range	-	-	-	-	-	-	0 ~ 1	0 ~ 1

Definition of Y value:

0: Static IP	1: DHCP

Definition of X value:

0: Default	1: Start

Note: When X value becomes 1, EtherNet setting will be initialized. (It will be triggered once only when the value turns to 1 from 0.)

P3-25	EtherNet IP address setting		Address: 0332H 0333H
Default:	0xC0A80101	Control mode:	ALL
Unit:	-	Range:	0x00000000 ~ 0xFFFFFFFF
Format:	HEX	Data size:	32-bit

Settings:

EtherNet IP address is set by hexadecimal format. Default is 192.168.1.1 = 0xC0A80101.

P3-26	EtherNet subnet mask setting		Address: 0334H 0335H
Default:	0xFFFFFF00	Control mode:	ALL
Unit:	-	Range:	0x00000000 ~ 0xFFFFFFFF
Format:	HEX	Data size:	32-bit

Settings:

EtherNet subnet mask is set by hexadecimal format. Default is 255.255.255.0 = 0xFFFFFF00.

P3-27	EtherNet default gateway setting		Address: 0336H 0337H
Default:	0xC0A80101	Control mode:	ALL
Unit:	-	Range:	0x00000000 ~ 0xFFFFFFFF
Format:	HEX	Data size:	32-bit

Settings:

EtherNet default gateway is set by hexadecimal format. Default is 192.168.1.1 = 0xC0A80101.

P3-28	Reserved

P3-29	DMCNET function setting		Address: 033AH 033BH
Default:	0x0001	Control mode:	ALL
Unit:	-	Range:	0x0000 ~ 0xFFFF
Format:	HEX	Data size:	16-bit

Settings:

DMCNET setting

Format of the setting value: P3-29.UZYX

P3-29.X = 0 (Disable DMCNET function) (not supported now)

P3-29.X = 1 (Enable DMCNET function) (default mode)

P3-29.Y = 0 (Master mode) (default mode)

P3-29.Y = 1 (Slave mode) (not supported now)

P3-29.Z (System reserved)

P3-29.U (System reserved)

P3-30	DMCNET function control		Address: 033CH 033DH
Default:	-	Control mode:	ALL
Unit:	-	Range:	0x0000 ~ 0xFFFF
Format:	HEX	Data size:	16-bit

Settings:

DMCNET control

Format of the setting value: P3-30.UZYX

P3-30.X = 0 (State: Scan ends normally)

P3-30.X = 1 (Command: Start to scan the node of slave station)

P3-30.X = 2 (Command: Save the scanning result in non-volatile area, P3-31.Low-word ≦ P3-31.High-word)

P3-30.X = 4 (State: System verification)

P3-30.X = E (State: System uninitialized after start-up)

P3-30.X = F (State: Scan failure/overtime/verification not match)

P3-30.Y (System reserved)

P3-30.Z (System reserved)

P3-30.U (System reserved)

8

P3-31	DMCNET slave status (No.1)		Address: 033EH 033FH
Default:	-	Control mode:	ALL
Unit:	-	Range:	-
Format:	HEX	Data size:	32-bit

Settings:

It is divided into Low Word and High Word:

High Word is used for recording the device type that scanned by the system; Low Word is used for recording the device type that verified by the system.

Number for device type:

0: Not Connected

1: A2-F series servo drive

2: M-F series servo drive

3: ASD-DMC-RM32NT (Remote extension module with 32 digital output points; transistor type output)

4: ASD-DMC-RM64NT (Remote extension module with 64 digital output points; transistor type output)

5: ASD-DMC-RM32PT (Remote extension module with 16 digital input points/16 digital output points; transistor type output)

6: ASD-DMC-RM32MN (Remote extension module with 32 digital input points; NPN/PNP)

7: ASD-DMC-RM64MN (Remote extension module with 64 digital input points; NPN/PNP)

8: ASD-DMC-RM04PI-MODE2 (4-axis remote extension stepping module; PDO mode)

9: ASD-DMC-RM04PI-MODE1 (4-axis remote extension stepping module; SDO mode)

A: ASD-DMC-RM04AD (Remote extension module with 4 analog input points)

B: ASD-DMC-RM04DA (Remote extension module with 4 analog output points)

C: HMC-RIO3232RT5 (Remote extension module with 32 digital input points / 32 digital output points; Relay/ transistor type output)

P3-32	DMCNET slave status (No.2)		Address: 0340H 0341H
Default:	-	Control mode:	ALL
Unit:	-	Range:	-
Format:	HEX	Data size:	32-bit

Settings:

Please refer to P3-31 for parameter definition.

P3-33 🔒	DMCNET slave status (No.3)		Address: 0342H 0343H
Default:	-	Control mode:	ALL
Unit:	-	Range:	-
Format:	HEX	Data size:	32-bit

Settings:

Please refer to P3-31 for parameter definition.

P3-34 🔒	DMCNET slave status (No.4)		Address: 0344H 0345H
Default:	-	Control mode:	ALL
Unit:	-	Range:	-
Format:	HEX	Data size:	32-bit

Settings:

Please refer to P3-31 for parameter definition.

P3-35 🔒	DMCNET slave status (No.5)		Address: 0346H 0347H
Default:	-	Control mode:	ALL
Unit:	-	Range:	-
Format:	HEX	Data size:	32-bit

Settings:

Please refer to P3-31 for parameter definition.

P3-36 🔒	DMCNET slave status (No.6)		Address: 0348H 0349H
Default:	-	Control mode:	ALL
Unit:	-	Range:	-
Format:	HEX	Data size:	32-bit

Settings:

Please refer to P3-31 for parameter definition.

P3-37 🔒	DMCNET slave status (No.7)		Address: 034AH 034BH
Default:	-	Control mode:	ALL
Unit:	-	Range:	-
Format:	HEX	Data size:	32-bit

Settings:

Please refer to P3-31 for parameter definition.

8

P3-38 🔒	DMCNET slave status (No.8)		Address: 034CH 034DH
Default:	-	Control mode:	ALL
Unit:	-	Range:	-
Format:	HEX	Data size:	32-bit

Settings:

Please refer to P3-31 for parameter definition.

P3-39 🔒	DMCNET slave status (No.9)		Address: 034EH 034FH
Default:	-	Control mode:	ALL
Unit:	-	Range:	-
Format:	HEX	Data size:	32-bit

Settings:

Please refer to P3-31 for parameter definition.

P3-40 🔒	DMCNET slave status (No.10)		Address: 0350H 0351H
Default:	-	Control mode:	ALL
Unit:	-	Range:	-
Format:	HEX	Data size:	32-bit

Settings:

Please refer to P3-31 for parameter definition.

P3-41 🔒	DMCNET slave status (No.11)		Address: 0352H 0353H
Default:	-	Control mode:	ALL
Unit:	-	Range:	-
Format:	HEX	Data size:	32-bit

Settings:

Please refer to P3-31 for parameter definition.

P3-42 🔒	DMCNET slave status (No.12)		Address: 0354H 0355H
Default:	-	Control mode:	ALL
Unit:	-	Range:	-
Format:	HEX	Data size:	32-bit

Settings:

Please refer to P3-31 for parameter definition.

8.4 List of servo drive parameters

Monitor and general output setting parameters						
Parameter No.	Function	Default value	Unit	Control mode		
				DMC	Sz	Tz
P0-00	Firmware version	Factory	-	O	O	O
P0-01	Alarm code display of the servo drive	-	-	O	O	O
P0-08	Total on-time of the servo drive	0	Hour	O	O	O
P0-46	Servo digital output (DO) status display	0	-	O	O	O

Filter and resonance suppression parameters						
Parameter No.	Function	Default value	Unit	Control mode		
				DMC	Sz	Tz
P1-25	Low-frequency vibration suppression (1)	100	Hz	O		
P1-26	Low-frequency vibration suppression gain (1)	0	-	O		
P1-27	Low-frequency vibration suppression (2)	100	Hz	O		
P1-28	Low-frequency vibration suppression gain (2)	0	-	O		
P1-29	Auto mode for low-frequency vibration	0	-	O		
P1-30	Low-frequency vibration detection	500	pulse	O		
P1-34	Acceleration constant of S-curve	200	ms		O	
P1-35	Deceleration constant of S-curve	200	ms		O	
P1-36	Acceleration/deceleration constant of S-curve	0	ms	O	O	
P2-23	Resonance suppression (Notch filter) (1)	1000	Hz	O	O	O
P2-24	Resonance suppression (Notch filter) attenuation rate (1)	0	dB	O	O	O
P2-49	Speed detection and jitter suppression	0	sec	O	O	O

Icon of parameter property	Description
	Read-only parameter. Users can only read the status.
	Parameter cannot be set when it is in servo on status.
	Parameter setting will be valid after re-power on the servo drive.
	When the power is off, the parameter value will be the default value.
	It is shared by multiple axes.

Gain and switching parameters						
Parameter No.	Function	Default value	Unit	Control mode		
				DMC	Sz	Tz
P1-37	Inertia ratio and load weight ratio of servo motor	10	0.1 times	O	O	O
P2-00	Position control gain	35	Rad/s	O		
P2-01	Rate of change for position control gain	100	%	O		
P2-02	Position feed forward gain	50	%	O		
P2-03	Smooth constant of position feed forward	5	ms	O		
P2-04	Speed control gain	500	Rad/s	O	O	O
P2-05	Rate of change for speed control gain	100	%	O	O	O
P2-06	Speed integral compensation	100	Rad/s	O	O	O
P2-07	Speed feed forward gain	0	%	O	O	O
P2-26	Anti-interference gain	0	0.001	O	O	O

Position control parameters						
Parameter No.	Function	Default value	Unit	Control mode		
				DMC	Sz	Tz
P1-01	Input setting of control mode and control command	0xB	-	O	O	O
P1-44	E-Gear ratio (Numerator) (N1)	128	pulse	O	O	O
P1-45	E-Gear ratio (Denominator) (M1)	10	pulse	O	O	O
P5-08	Forward software limit	2147483647	PUU	O		
P5-09	Reverse software limit	-2147483648	PUU	O		

Configuration of digital input/digital output						
Parameter No.	Function	Default value	Unit	Control mode		
				DMC	Sz	Tz
P2-10	DI1 configuration	0x101	-	O	O	O
P2-11	DI2 configuration	104	-	O	O	O
P2-18	DO1 configuration	0x101	-	O	O	O
P3-06	Control switch of digital input (DI)	0	-	O	O	O
P4-07	Multiple function of digital input	0	-	O	O	O

8

8.5 Description of servo parameters

P0-xx Monitoring parameters

P0-00	Firmware version		Address: 0000H 0001H
Default:	Factory setting	Control mode:	ALL
Unit:	-	Range:	-
Format:	DEC	Data size:	16-bit

Settings: It displays the firmware version of the servo drive.

P0-01	Alarm code display of the servo drive		Address: 0002H 0003H
Default:	-	Control mode:	ALL
Unit:	-	Range:	0x0000 ~ 0xFFFF, set P0-01 to 1 to clear the alarm.
Format:	HEX	Data size:	16-bit

Settings: It displays the alarm code. Please refer to Chapter 11 for detailed information.

Alarm list			
Code	Item	Code	Item
001	Overcurrent	020	Serial communication timeout
002	Overvoltage	021	Reserved
003	Under voltage (When it is in servo on status, the RST voltage is not enough)	022	RST leak phase
004	Motor combination error	023	Early warning for overload
005	Regeneration error	024	Encoder initial magnetic field error
006	Overload	025	Internal error of the encoder (memory and counter are in error)
007	Overspeed	026	Unreliable internal data of the encoder
008	Abnormal pulse command	027	The internal of the motor is in error
009	Excessive deviation of position command	028	Encoder voltage error or the internal of the encoder is in error
010	Reserved	029	Gray code error
011	Encoder error	030	Motor crash error
012	Adjustment error	031	Incorrect wiring of motor power cable
013	Emergency stop	034	Internal communication of the encoder is in error
014	Reverse limit error	044	Warning of servo function overload
015	Forward limit error	060	The absolute position is lost
016	IGBT overheat	061	Encoder under voltage
017	Abnormal EEPROM	062	The multi-turn of absolute encoder overflows

Alarm list			
Code	Item	Code	Item
018	Abnormal signal output	069	Wrong motor type
019	Serial communication error	099	EEPROM must be updated

Alarm list of DMCNET communication			
Code	Item	Code	Item
185	DMCNET Bus hardware error	-	-

Alarm list of motion control			
Code	Item	Code	Item
201	An error occurs when loading DMCNET data	301	DMCNET synchronization failure
283	Software positive limit	302	The synchronized signal of DMCNET is sent too fast
285	Software negative limit	303	The synchronized signal of DMCNET is sent too slow
289	Feedback position counter overflows	304	DMCNET IP command failed

P0-08	Total on-time of the servo drive		Address: 0010H 0011H
Default:	0	Control mode:	ALL
Unit:	hour	Range:	0 ~ 65535
Format:	DEC	Data size:	16-bit

Settings:

It displays the total on-time of the servo drive.

P0-46	Servo digital output (DO) status display		Address: 005CH 005DH
Default:	0	Control mode:	ALL
Unit:	-	Range:	-
Format:	HEX	Data size:	16-bit

Settings:

Bit00: SRDY (servo ready)

Bit01: SON (servo activated)

Bit02: ZSPD (zero speed detection)

Bit03: TSPD (target speed reached)

Bit04: TPOS (target position reached)

Bit05: TQL (torque limit is activated)

Bit06: ALRM (servo alarm)

Bit07: BRKR (brake control signal output)

Bit08: HOME (homing completed)

Bit09: OLW (early warning for motor overload)

Bit10: WARN (This DO is on when servo warning occurs, EMGS, under voltage or communication error occurs)

Bit11: Reserved

Bit12: Reserved

Bit13: Reserved

Bit14: Reserved

Bit15: Reserved

P0-49	**Update encoder absolute position**		**Address: 0062H 0063H**
Default:	0	Control mode:	ALL
Unit:	-	Range:	0x00 ~ 0x22
Format:	HEX	Data size:	16-bit

Settings:

Format of the setting value: 0xUZYX

U	Z	Y	X
Reserved	Reserved	Absolute position	Encoder data setting

Encoder data setting:

1: It updates the encoder data to P0-50 ~ P0-52.

2: It updates the encoder data to P0-50 ~ P0-52 and clears the position error. Then, the motor’s current position will be reset to the absolute position that corresponds to P0-51 and P0-52.

P0-50	**Encoder status**		**Address: 0064H 0065H**
Default:	0	Control mode:	ALL
Unit:	-	Range:	-
Format:	HEX	Data size:	16-bit

Settings:

Bit 00: 1 means the absolute position is lost; 0 means normal.

Bit 01: 1 means the battery is under voltage; 0 means normal.

Bit 02: 1 means multi-turn overflows; 0 means normal.

Bit 03: 1 means PUU overflows; 0 means normal.

Bit 04: 1 means the absolute coordinate system has not been created; 0 means normal.

Bit 05 ~ Bit15: reserved (0).

8

P0-51	Encoder absolute position - Multi-turn		Address: 0066H 0067H
Default:	0	Control mode:	ALL
Unit:	Revolution (turn)	Range:	-32768 ~ +32767
Format:	DEC	Data size:	32-bit

Settings:

It displays the turns of encoder absolute position.

P0-52	Encoder absolute position - Pulse number in one single turn or PUU		Address: 0068H 0069H
Default:	0	Control mode:	ALL
Unit:	Pulse or PUU	Range:	-2147483648 ~ 2147483647
Format:	DEC	Data size:	32-bit

Settings:

When bit 1 of P2-70 is set to 1 to read the pulse number, this parameter represents the pulse number of encoder absolute position in one single turn.

When bit 1 of P2-70 is set to 0 to read the PUU number, this parameter represents PUU number of motor absolute position.

P1-xx Basic parameters

P1-01	Input setting of control mode and control command		Address: 0102H 0103H
Default:	0xB	Control mode:	ALL
Unit:	-	Range:	0X1001, 0x1004, 0x1005, 0x100B
Format:	HEX	Data size:	16-bit

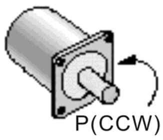
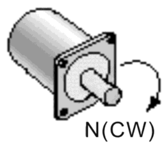
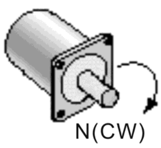
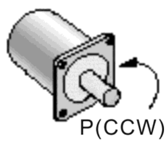
Settings:

Format: U Z Y X

X: Control mode setting

Setting value	Mode	Description
01	PR	Position control mode
04	Sz	Internal speed command
05	Tz	Internal torque command
0B	DMCNET	Communication mode

Y: Direction of torque output

	0	1
Forward direction		
Reverse direction		

Z: DIO setting

0: When switching modes, DIO setting (P2-10 ~ P2-18) remains the same.

1: When switching modes, DIO setting (P2-10 ~ P2-18) will be reset to the default.

U: Reserved.

P1-25	Low-frequency vibration suppression (1)		Address: 0132H 0133H
Default:	1000	Control mode:	ALL
Unit:	0.1 Hz	Range:	10 ~ 1000
Format:	DEC	Data size:	16-bit

Settings:

This is for setting the value of the first low-frequency vibration suppression. When P1-26 is set to 0, the first low-frequency filter will be disabled.

8

P1-26	Low-frequency vibration suppression gain (1)		Address: 0134H 0135H
Default:	1000	Control mode:	ALL
Unit:	-	Range:	0 ~ 9 (0: Disable low-frequency filter)
Format:	DEC	Data size:	16-bit

Settings:

This is for setting the gain of the first low-frequency vibration suppression. Increasing the setting value brings better position response. However, if the value is set too big, the motor will not be able to operate smoothly. It is suggested to set the value to 1.

P1-27	Low-frequency vibration suppression (2)		Address: 0136H 0137H
Default:	1000	Control mode:	ALL
Unit:	-	Range:	10 ~ 1000
Format:	DEC	Data size:	16-bit

Settings:

This is for setting the value of the second low-frequency vibration suppression. When P1-28 is set to 0, the second low-frequency filter will be disabled.

P1-28	Low-frequency vibration suppression gain (2)		Address: 0138H 0139H
Default:	1000	Control mode:	ALL
Unit:	-	Range:	0 ~ 9 (0: Disable the first low-frequency filter)
Format:	DEC	Data size:	16-bit

Settings:

This is for setting the the gain of second low-frequency vibration suppression. Increasing the setting value brings better position response. However, if the value is set too big, the motor will not be able to operate smoothly. It is suggested to set the value to 1.

P1-29	Auto mode for low-frequency vibration suppression		Address: 013AH 013BH
Default:	0	Control mode:	ALL
Unit:	-	Range:	0 ~ 1
Format:	DEC	Data size:	16-bit

Settings:

0: Disable the auto mode.

1: Enable this function. And disable it once the vibration is suppressed.

Description of auto mode setting:

When the value is set to 1, it is in auto mode for vibration suppression. When the vibration is not being detected or the vibration frequency is stable, the parameter will be set to 0 and save the result of low frequency vibration suppression to P1-25 automatically.

P1-30	Low-frequency vibration detection		Address: 013CH 013DH
Default:	500	Control mode:	ALL
Unit:	Pulse	Range:	1 ~ 8000
Format:	DEC	Data size:	16-bit

Settings:

When auto suppression is enabled (P1-29 is set to 1), it will search the frequency in accordance with the set detection level. The lower the value is, the more sensitive the detection will be. However, the system might misjudge the noise or regard the low-frequency vibration as the one to be suupporessed. If the value is bigger, it will make more precise judgment. Nevertheless, if the vibration of the mechanism is smaller, it might not detect the frequency of low-frequency vibration.

P1-32	Motor stop mode		Address: 0140H 0141H
Default:	0	Control mode:	ALL
Unit:	Pulse	Range:	0 ~ 0x20
Format:	HEX	Data size:	16-bit

Settings:

Format: U Z Y X

X: Reserved

Y: Options for using the dynamic brake when the servo is off or an alarm (including EMGS) occurs.

0: Use dynamic brake

1: Motor free run

2: Use dynamic brake first. Then, start racing after the speed is slower than the value set by P1-38. When the motor reaches PL or NL, please refer to the setting of P5-03 for setting the deceleration time. If the setting time is 1 ms, the motor stops instantaneously.

Z: Reserved.

U: Reserved.

8

P1-34	Acceleration constant of S-curve		Address: 0144H 0145H
Default:	200	Control mode:	Sz
Unit:	ms	Range:	1 ~ 65500
Format:	DEC	Data size:	16-bit

Settings:

It is the acceleration constant. The time that motor accelerates from 0 to the rated speed can be set individually by P1-34, P1-35 and P1-36.

P1-35	Deceleration constant of S-curve		Address: 0146H 0147H
Default:	0	Control mode:	ALL
Unit:	ms	Range:	1 ~ 65500
Format:	DEC	Data size:	16-bit

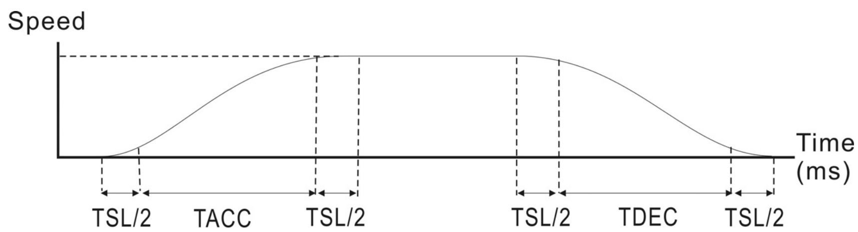
Settings:

It is the deceleration constant. The time that motor decelerates from rated speed to 0 can be set individually by P1-34, P1-35 and P1-36.

P1-36	Acceleration/deceleration constant of S-curve		Address: 0148H 0149H
Default:	0	Control mode:	ALL
Unit:	ms	Range:	1 ~ 65500
Format:	DEC	Data size:	16-bit

Settings:

It is the acceleration / deceleration constant of S-curve.



P1-34: It is for setting the acceleration time of S-curve.

P1-35: It is for setting the deceleration time of S-curve.

P1-36: It is for setting the smoothing time of S-curve acceleration / deceleration.

P1-34, P1-35 and P1-36 can be set individually.

P1-37	Inertia ratio and load weight ratio of servo motor		Address: 014AH 014BH
Default:	10	Control mode:	ALL
Unit:	0.1 times	Range:	0 ~ 2000
Format:	DEC	Data size:	16-bit

Settings:

Inertia ratio of the servo motor (rotary motor) (J_load / J_motor):

J_motor: It is the rotor inertia of the servo motor.

J_load: It is the total equivalent inertia of external mechanical load.

P1-38	Trigger level of zero-speed signal		Address: 014CH 014DH
Default:	10	Control mode:	ALL
Unit:	0.1 r/min	Range:	0 ~ 2000
Format:	DEC	Data size:	16-bit

Settings:

This is for setting the trigger level of zero-speed signal (ZSPD). When the motor speed (in forward/reverse direction) is slower than the setting value, the digital output will be enabled.

P1-42	Delay time of brake when servo is on		Address: 0154H 0155H
Default:	0	Control mode:	ALL
Unit:	ms	Range:	0 ~ 1000
Format:	DEC	Data size:	16-bit

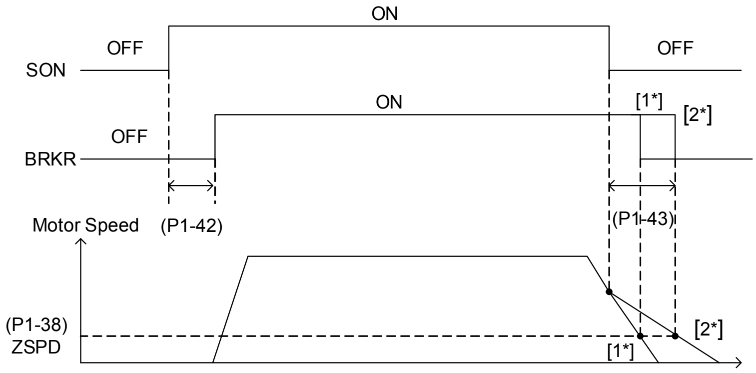
Settings:

This is for setting the delay time from servo in servo on state to signal of brake (BRKR) on.

P1-43	Delay time of brake when servo is off		Address: 0156H 0157H
Default:	0	Control mode:	ALL
Unit:	ms	Range:	-1000 ~ 1000
Format:	DEC	Data size:	16-bit

Settings:

This is for setting the delay time from servo in servo off state to signal of brake (BRKR) off.



Note:

1. If the delay time specified by P1-43 is not over yet and the motor speed is slower than the value of P1-38, the signal of brake (BRKR) is off.
2. If the delay time of P1-43 is up and the motor speed is higher than the value of P1-38, the signal of brake (BRKR) is off.
3. If P1-43 is set to a negative value and the servo is in servo off state due to alarm (except AL022) or emergency stop, its setting value will be 0.

P1-44 ⊖	E-Gear ratio (Numerator) (N1)		Address: 0158H 0159H
Default:	128	Control mode:	ALL
Unit:	Pulse	Range:	1 ~(2^{29}-1)
Format:	DEC	Data size:	32-bit

Settings:

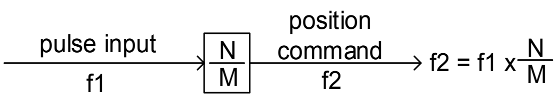
Please refer to P1-45 for e-gear ratio (numerator) setting.

P1-45 ⊖	E-Gear ratio (Denominator) (M1)		Address: 015AH 015BH
Default:	10	Control mode:	ALL
Unit:	Pulse	Range:	1 ~(2^{31}-1)
Format:	DEC	Data size:	32-bit

Settings:

This is for setting e-gear ratio (denominator). If the setting is incorrect, the servo motor will easily have sudden unintended acceleration. Please observe the following setting rules:

Setting of pulse input:



Range of command pulse input: 1／50＜Nx／M＜25600

Note: The setting value cannot be modified when it is in servo on state.

P1-48	Speed reached (DO: MC_OK)		Address: 0160H 0161H
Default:	0x0000	Control mode:	ALL
Unit:	Pulse	Range:	0x0000 ~ 0x0011
Format:	HEX	Data size:	16-bit

Settings:

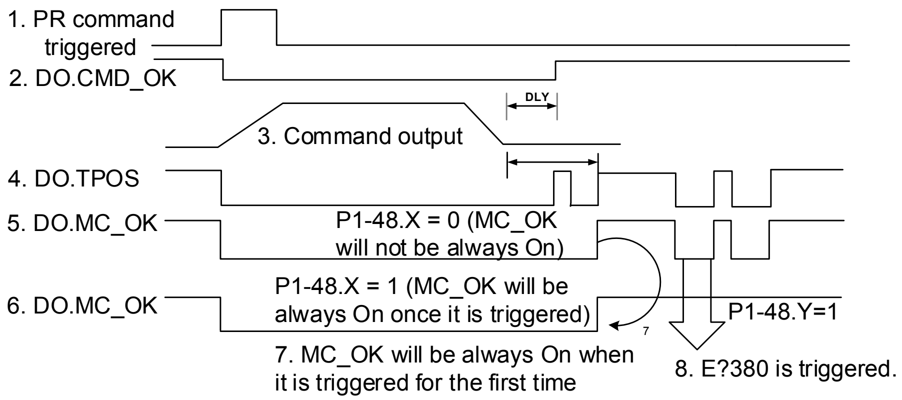
Sequential setting of DO.MC_OK (DO code: 0x17):

Format: 00YX

X = 0: The digital output status will not be remained; 1: It will remain the digital output status

Y = 0: Do not show E?380 when position deviation occurs; 1: Show E? 380 when position deviation occurs.

Diagram:



Description:

1. Command triggered: It means the new PR command is effective. Position command starts to output and clear signal 2, 4, 5, 6 at the same time. Command triggering source: DI.CTRG, EV1/EV2, and P5-07 (triggered via software).
2. CMD_OK: It means the position command is completely outputted and can set the delay time (DLY).
3. Command output: Output the profile of position command according to the setting of acceleration / deceleration.
4. TPOS: It means the position error of the servo drive is within the range set by P1-54.
5. MC_OK: It means the position command is complete. Then, DO.CMD_OK and DO.TPOS are both on.
6. MC_OK (remains the digital output status): It is the same as 5. However, once this DO is on, its status will be remained regardless the status signal 4.
7. The output profile is determined by parameter P1-48.X.
8. Position deviation: When number 7 happens, if signal 4 (or 5) is off, it means the position is deviated and E?380 can be triggered. This alarm can be set via parameter P1-48.Y.

8

P1-54	Range of position reached		Address: 016CH 016DH
Default:	12800	Control mode:	ALL
Unit:	Pulse	Range:	0 ~ 1280000
Format:	DEC	Data size:	32-bit

Settings:

In position mode, when the deviation pulse number is smaller than the setting range of P1-54, DO.TPOS is on.

P1-55	Max. speed limit		Address: 016EH 016FH
Default:	Same as the rated speed of each model	Control mode:	ALL
Unit:	r/min	Range:	0 ~ max. speed
Format:	DEC	Data size:	16-bit

Settings:

This is for setting the max. speed of the servo motor. Its default is the rated speed.

P1-57	Motor crash protection (torque percentage)		Address: 0172H 0173H
Default:	0	Control mode:	ALL
Unit:	%	Range:	0 ~ 300
Format:	DEC	Data size:	16-bit

Settings:

This is for setting the protection level. (For the percentage of rated torque, setting P1-57 to 0 means to disable the function; setting the value to 1 or above means to enable the function)

P1-59	Motor crash protection (protection time)		Address: 0174H 0175H
Default:	1	Control mode:	ALL
Unit:	ms	Range:	1 ~ 1000
Format:	DEC	Data size:	16-bit

Settings:

This is for setting the protection time. When reaching the protection level, E?030 will occur after exceeding the protection time.

8

P2-xx Extension parameters

P2-00	Position control gain		Address: 0200H 0201H
Default:	35	Control mode:	ALL
Unit:	rad/s	Range:	0 ~ 2047
Format:	DEC	Data size:	16-bit

Settings:

When the value of position loop gain is increased, the position response can be enhanced and the position error can be reduced. However, if the value is set too big, it may easily cause vibration and noise.

P2-01	Rate of change for speed control gain		Address: 0202H 0203H
Default:	100	Control mode:	ALL
Unit:	%	Range:	10 ~ 500
Format:	DEC	Data size:	16-bit

Settings:

Switch the changing rate of position loop gain according to the gain switching condition.

P2-02	Position feed forward gain		Address: 0204H 0205H
Default:	50	Control mode:	ALL
Unit:	%	Range:	0 ~ 100
Format:	DEC	Data size:	16-bit

Settings:

If the position command is changed smoothly, increasing the gain value can reduce the position error. If not, decreasing the gain value can tackle the problem of mechanical vibration.

P2-03	Smooth constant of position feed forward gain		Address: 0206H 0207H
Default:	5	Control mode:	ALL
Unit:	ms	Range:	2 ~ 100
Format:	DEC	Data size:	16-bit

Settings:

If the position command is changed smoothly, decreasing the value can reduce the position error. If not, then increasing the value can tackle the problem of mechanical vibration.

8

P2-04	Speed control gain		Address: 0208H 0209H
Default:	500	Control mode:	ALL
Unit:	rad/s	Range:	0 ~ 8191
Format:	DEC	Data size:	16-bit

Settings:

Increasing the value of speed loop gain can enhance the speed response. However, if the value is set too big, it could easily cause resonance and noise.

P2-05	Rate of change for speed control gain		Address: 020AH 020BH
Default:	100	Control mode:	ALL
Unit:	rad/s	Range:	10 ~ 500
Format:	DEC	Data size:	16-bit

Settings:

This parameter switches the changing rate of speed loop gain according to the gain switching condition.

P2-06	Speed integral compensation		Address: 020CH 020DH
Default:	100	Control mode:	ALL
Unit:	rad/s	Range:	0 ~ 1023
Format:	DEC	Data size:	16-bit

Settings:

Increasing the value of speed loop gain can enhance the speed response and diminish the deviation of speed control. However, if the value is set too big, it could easily cause resonance and noise.

P2-07	Speed feed forward gain		Address: 020EH 020FH
Default:	0	Control mode:	ALL
Unit:	%	Range:	0 ~ 100
Format:	DEC	Data size:	16-bit

Settings:

When the speed control command runs smoothly, increasing the gain value can reduce the speed command error. If the command does not run smoothly, decreasing the gain value can reduce the mechanical vibration during operation.

P2-08 ⛉	Special parameter with write-in function		Address: 0210H 0211H
Default:	0	Control mode:	ALL
Unit:	-	Range:	0 ~ 65535
Format:	DEC	Data size:	16-bit

Settings:

Write-in function:

Code	Function
10	Reset single-axis parameters (Connect to the power again after reset)
11	Reset all-axis parameters (Connect to the power again after reset)
30, 28	Update the firmware

P2-10	DI 1 configuration		Address: 0212H 0213H
Default:	101	Control mode:	ALL
Unit:	-	Range:	0 ~ 0X 315Fh (the last two codes are DI code)
Format:	HEX	Data size:	16-bit

Settings:

Format: U Z Y X

YX: Input function selection: Please refer to table 8.1.

Z: Input contact: **a** or **b** contact

0: Set this input contact as normally closed (**b** contact)

1: Set this input contact as as normally closed (**a** contact)

U: Axial selection: Select the axis that corresponds to this DI.

0: Set the axis to 0 and this DI is shared by 4 axes.

1: Set the axis to 1 and this DI is used by axis 1.

2: Set the axis to 2 and this DI is used by axis 2.

3: Set the axis to 3 and this DI is used by axis 3.

4: Set the axis to 4 and this DI is used by axis 4.

Please re-power on your MS controller after modifying parameters.

Attention: P3-06 can be used to determine the DI control, either by external terminal or parameter P4-07.

Note:

1. Three functions are provided by the DI shared by 4 axes.
   a. Servo on: The setting value is 0101 (**a** contact) and 0001 (**b** contact).
   b. Alarm reset: The setting value is 0102 (**a** contact) and 0002 (**b** contact).
   c. Emergency stop: The setting value is 0103 (**a** contact) and 0003 (**b** contact).
2. When switching the mode, if the DIO setting value is reset, the axial selection will restore to the default.

8

P2-11	DI 2 configuration		Address: 0216H 0217H
Default:	104	Control mode:	ALL
Unit:	-	Range:	0 ~ 0x315F (the last two codes are DI code)
Format:	HEX	Data size:	16-bit

Settings:

Format: U Z Y X; Please refer to P2-10.

P2-18	DO 1 configuration		Address: 0224H 0225H
Default:	101	Control mode:	ALL
Unit:	-	Range:	0 ~ 0x313F (the last two codes are DO code)
Format:	HEX	Data size:	16-bit

Settings:

Format: U Z Y X

YX: Output function selection: Please refer to table 8.2.

Z: Output contact: **a** or **b** contact

0: Set this output contact as normally closed (contact **b**)

1: Set this output contact as as normally closed (contact **a**)

U: Axial selection: Select the axis that corresponds to this DO.

1: Set the axis to 1 and this DO is used by axis 1.

2: Set the axis to 2 and this DO is used by axis 2.

3: Set the axis to 3 and this DO is used by axis 3.

4: Set the axis to 4 and this DO is used by axis 4.

Please re-power on your MS controller after modifying parameters.

Note: When switching the mode, if the DIO setting value is reset, the axial selection will restore to the default.

P2-23	Resonance suppression (Notch filter) (1)		Address: 022EH 022FH
Default:	1000	Control mode:	ALL
Unit:	Hz	Range:	50 ~ 1000
Format:	DEC	Data size:	16-bit

Settings:

This is the first setting value of resonance frequency. If P2-24 is set to 0, this function is disabled.

P2-24	Resonance suppression (Notch filter) attenuation rate (1)		Address: 0230H 0231H
Default:	0	Control mode:	ALL
Unit:	dB	Range:	0 ~ 32
Format:	DEC	Data size:	16-bit

Settings:

This is the first resonance suppression (Notch filter) attenuation rate. When this parameter is set to 0, function of Notch filter will be disabled.

P2-25	Low-pass filter of resonance suppression		Address: 0232H 0233H
Default:	2 (1 kW below) or 5 (other models)	Control mode:	ALL
Unit:	0.1 ms	Range:	0 ~ 1000
Format:	DEC	Data size:	16-bit

Settings:

This is for setting the time constant for the low-pass filter of resonance suppression. When P2-25 is set to 0, the function of low-pass filter is disabled.

P2-26	Anti-interference gain		Address: 0234H 0235H
Default:	0	Control mode:	ALL
Unit:	-	Range:	0 ~ 1023 (0: Disable ths function)
Format:	DEC	Data size:	16-bit

Settings:

Increasing the value of this parameter can increase the damping of speed loop. It is suggested to set P2-26 that equals to the value of P2-06. Please observe the rules below for adjusting P2-26:

1. In speed mode, increasing the value of this parameter can avoid speed overshoot.
2. In position mode, decreasing the value of this parameter can avoid position overshoot.

P2-35	Warning for excessive deviation of position command		Address: 0246H 0247H
Default:	3840000	Control mode:	ALL
Unit:	Pulse	Range:	1 ~ 128000000
Format:	DEC	Data size:	32-bit

Settings:

This is for setting the warning condition for excessive deviation of position command.

8

P2-49	Speed detection and jitter suppression		Address: 0262H 0263H
Default:	0	Control mode:	ALL
Unit:	-	Range:	0x00 ~ 0x1F
Format:	DEC	Data size:	16-bit

Settings: This is for setting the filter of speed estimation.

Setting value	Bandwidth of speed estimation (Hz)
00	2500
01	2250
02	2100
03	2000
04	1800
05	1600
06	1500
07	1400
08	1300
09	1200
0A	1100
0B	1000
0C	950
0D	900
0E	850
0F	800
10	750
11	700
12	650
13	600
14	550
15	500
16	450
17	400
18	350
19	300
1A	250
1B	200
1C	175
1D	150
1E	125
1F	100

P2-53	Position integral compensation		Address: 026AH 026BH
Default:	0	Control mode:	ALL
Unit:	rad/s	Range:	0 ~ 1023
Format:	DEC	Data size:	16-bit

Settings:

When increasing the value of position integral, it can reduce the position steady-state error.

However, it may easily cause position overshoot and noise if the value is set too big.

P2-69 ⏻	Encoder setting (absolute type)		Address: 028AH 028BH
Default:	0	Control mode:	ALL
Unit:	-	Range:	0000h ~ 0111h
Format:	HEX	Data size:	16-bit

Settings:

Format: U Z Y X

X: Encoder type setting

0: Incremental type; An absolute type motor can be used as an incremental type.

1: Absolute type (It is only applicable to absolute type motor. If it is applied to incremental type motor, E?069 will occur.)

Y: Pulse command setting when the absolute position is lost

0: When E?060 or E?06A occurs, the MS controller will not receive any pulse command.

1: When E?060 or E?06A occurs, the MS controller can receive pulse command.

Z: Absolute position will not overflow when applying indexing function

0: Index coordinate is lost when the absolute position overflows.

1: Index coordinate is non-volatile, but the absolute position will not remain when the power is off.

U: Reserved.

The parameter setting is valid after the servo drive is re-powered on.

P2-70	Absolute type data accessing setting		Address: 028CH 028DH
Default:	0	Control mode:	ALL
Unit:	-	Range:	-
Format:	HEX	Data size:	16-bit

Settings:

Bit 0: Unit setting when accessing data via DI/DO. 1: pulse; 0: PUU

Bit 1: Unit setting when accessing data via parameter. 1: pulse; 0: PUU

Bit 2: Setting of overflow warning. 1: No overflow warning; 0: Overflow warning, including E?289 (PUU) and E?062 (pulse)

Bit 3 ~ Bit 15: reserved (0).

P2-71 ♡	Reset the absolute position to 0		Address: 028EH 028FH
Default:	0	Control mode:	ALL
Unit:	-	Range:	0000 ~ 0001h
Format:	HEX	Data size:	16-bit

Settings:

When P2-71 is set to 1, the current absolute position will be cleared to 0. This function is the same as DI.ABSC. Since the write-in function of P2-71 is protected by P2-08, users have to set P2-08 to 271 first to enable this function.

P2-93	STO FDBK control		Address: 02BAH 02BCH
Default:	0	Control mode:	-
Unit:	-	Range:	0x0010 ~ 0x0023
Format:	HEX	Data size:	16-bit

Settings:

Bit 0: Select the logic for FDBK status.

Bit 1: Determine if FDBK should be latched.

P3-xx Communication parameters

P3-06	Control switch of digital input (DI)		Address: 030CH 030DH
Default:	0	Control mode:	ALL
Unit:	-	Range:	0x0000 ~ 0x3FFF
Format:	HEX	Data size:	16-bit

Settings:

This is for setting DI control switch. Each bit of this parameter determines one input source of DI signal:

Bit 0 ~ Bit 5 are used for DI 1 ~ DI 6. See the descriptions below:

0: DI status is controlled by the external device.

1: DI status is controlled by P4-07.

Please refer to P2-10 for DI function configuration.

P4-xx Diagnosis parameters

P4-06	DO setting via software (readable and writable)		Address: 040CH 040DH
Default:	0	Control mode:	ALL
Unit:	-	Range:	0 ~ 0xFF
Format:	HEX	Data size:	16-bit

Settings:

7	6	5	4	3	2	1	0	Bit
0x37	0x36	0x35	0x34	0x33	0x32	0x31	0x30	Corresponded DO code
F	E	D	C	B	A	9	8	Bit
0x3F	0x3E	0x3D	0x3C	0x3B	0x3A	0x39	0x38	Corresponded DO code

When setting the DO number of each axis, setting parameter for each axis is also necessary.

When P2-18 = 0x1130, the output of axis [1] DO#1 is the status of P4-06 bit 0.

When P2-18 = 0x2130, the output of axis [2] DO#1 is the status of P4-06 bit 0.

When P2-18 = 0x3130, the output of axis [3] DO#1 is the status of P4-06 bit 0.

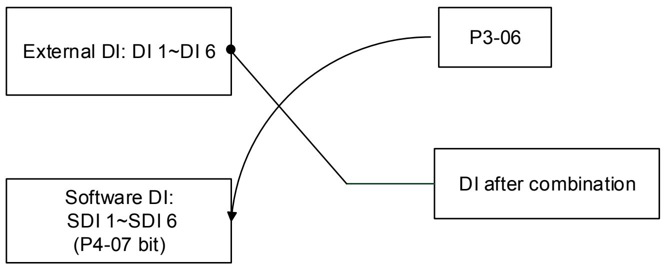
When P2-18 = 0x4130, the output of axis [4] DO#1 is the status of P4-06 bit 0.

Please set DO code (0x30 ~ 0x3F) first and complete the setting in P4-06.

P4-07	Multiple functions of digital input		Address: 040EH 040FH
Default:	0	Control mode:	ALL
Unit:	-	Range:	0 ~ 0x3FFF
Format:	HEX	Data size:	16-bit

Settings:

The source of DI input signal can be external terminal (DI 1 ~ DI 6) or software SDI 1 ~ SDI 6 (Bit 0 ~ 5 of corresponding parameter P4-07), which is determined by P3-06. The corresponding bit of P3-06 is 1, which means the DI signal source is software SDI (P4-07). If the corresponding bit is 0, then the DI signal source is hardware DI. See the figure below:



Read parameters: It shows the DI status after combining external DI and software DI.

Write parameters: It writes software SDI status. Example:

When the reading value of P4-07 is 0x0011, it means DI 1 and DI 5 are On.

When the writing value of P4-07 is 0x0011, is means SDI 1 and SDI 5 are On.

Please refer to P2-10 for DI function configuration.

P5-xx Motion control parameters

P5-08	Forward software limit		Address: 0510H 0511H
Default:	2147483647	Control mode:	ALL
Unit:	PUU	Range:	-2147483648 ~ +2147483647
Format:	DEC	Data size:	32-bit

Settings:

In position mode, if the motor rotates in forward direction and its command position exceeds the setting value of P5-08, E?283 will occur.

P5-09	Reverse software limit		Address: 0512H 0513H
Default:	-2147483648	Control mode:	ALL
Unit:	PUU	Range:	-2147483648 ~ +2147483647
Format:	DEC	Data size:	32-bit

Settings:

In position mode, if the motor rotates in reverse direction and its command position exceeds the setting value of P5-09, E?285 will occur.

P6-xx PR parameters

P6-01	Origin definition		Address: 0602H 0603H
Default:	0	Control mode:	ALL
Unit:	PUU	Range:	-2147483648 ~ +2147483647
Format:	DEC	Data size:	32-bit

Settings:

This is for setting the value of origin.

Table 8.1 Description of digital input (DI)

Setting value: 0x01			
DI	Description	Triggering method	Control mode:
SON	When this DI is on, the servo drive is activated (servo on)	Level triggered	ALL

Setting value: 0x02			
DI	Description	Triggering method	Control mode:
ARST	After the alarm has been cleared, this DI can be used to clear the alarm signal.	Rising-edge triggered	ALL

Setting value: 0x21			
DI	Description	Triggering method	Control mode:
EMGS	When this DI is on, the motor stops urgently (emergency stop).	Level triggered	ALL

Setting value: 0x22			
DI	Description	Triggering method	Control mode:
NL	Reverse inhibit limit (**b** contact).	Level triggered	ALL

8

Setting value: 0x23			
DI	Description	Triggering method	Control mode:
PL	Forward inhibit limit (**b** contact).	Level triggered	ALL

Setting value: 0x46			
DI	Description	Triggering method	Control mode:
STOP	Motor stops.	Rising-edge triggered	ALL

Table 8.2 Description of digital output (DO)

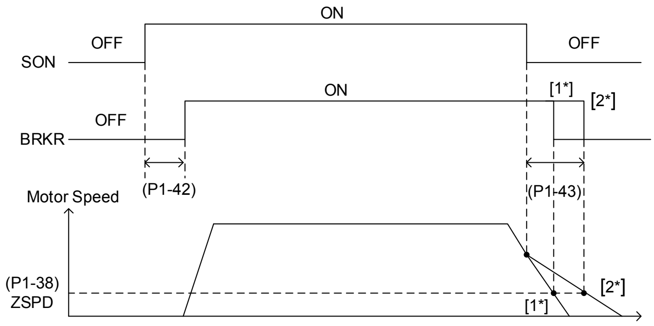
Setting value: 0x01			
DI	Description	Triggering method	Control mode:
SRDY	When the main circuit power is applied to the servo drive, this DO is on if no alarm occurs.	Level triggered	ALL

Setting value: 0x02			
DI	Description	Triggering method	Control mode:
SON	When the servo is activated (servo on), this DO is on if no alarm occurs.	Level triggered	ALL

Setting value: 0x03			
DI	Description	Triggering method	Control mode:
ZSPD	When the motor speed is slower than the setting speed of P1-38, this DO is on.	Level triggered	ALL

Setting value: 0x05			
DI	Description	Triggering method	Control mode:
TPOS	In position mode, when the deviation pulse number is smaller than the setting range of P1-54, DO.TPOS is on.	Level triggered	ALL

Setting value: 0x07			
DI	Description	Triggering method	Control mode:
ALRM	When an alarm occurs, this DO is on. (Except forward/reverse limit, communication error, under voltage and fan error)	Level triggered	ALL

Setting value: 0x08			
DI	Description	Triggering method	Control mode:
BRKR	When the signal of brake control is outputted, please adjust the setting of parameter P1-42 and P1-43. 	Level triggered	ALL

Setting value: 0x0B			
DI	Description	Triggering method	Control mode:
HOME	When homing is completed, it means the position coordinate system and position counter are available and this DO is on. When MS controller connects to the power, this DO is off. After homing is complete, this DO is on. During the operation, this DO is on until the position counter overflows (including command or feedback). Then, this DO turns off. When homing command is triggered, this DO is off. However, after homing is complete, this DO is on.	Level triggered	ALL

Setting value: 0x11			
DI	Description	Triggering method	Control mode:
WARN	Warning outputs (forward/reverse limit, communication error, under voltage and fan error)	Level triggered	ALL

Setting value: 0x13			
DI	Description	Triggering method	Control mode:
SNL	Software limit (reverse limit).	Level triggered	ALL

Setting value: 0x14			
DI	Description	Triggering method	Control mode:
SPL	Software limit (forward limit).	Level triggered	ALL

8

Setting value: 0x15			
DI	Description	Triggering method	Control mode:
Cmd_OK	When PR position command is complete, it will be in PR mode and this DO is on. When it is executing the PR command, this DO is off. When the command is complete, this DO turns on. This DO is used for referring the command status. Please refer to DO.TPOS for motor position command.	Level triggered	ALL

Setting value: 0x15			
DI	Description	Triggering method	Control mode:
MC_OK	When signals of DO.Cmd_OK and DO.TPOS are both on, DO.MC_OK is on. Please refer to servo parameter P1-48 for further information.	Level triggered	ALL

Parameters for SPF and feed rate modulation

Special-Process-Filter (SPF) - (This function is disabled in default setting. It is a non-volatile parameter.)

When the robot is moving on Cartesian coordinate system, overspeed or excessive acceleration can occur. After setting the max. speed (unit: PUU/ms) and acceleration (unit: PUU/ms^2) for each axis, the motor's actual speed and acceleration will be limited by this setting. When the acceleration is set to 0d, this function is disabled.

Example:

Setting the max. speed of axis 12 to 5000 PUU/ms and acceleration to 500 PUU/ms^2, SPF function is enabled.

1. Set P2-01 to 0x00010013.
2. Set P2-02 to 5000d.
3. Set P2-00 to 0x0C010014.
4. Set P2-01 to 0x00010013.
5. Set P2-02 to 500d.
6. Set P2-00 to 0x0C010024.

Adaptive feed rate modulation - (This function is disabled in default setting. It is a non-volatile parameter.)

When the axial speed exceeds the motor limit, SPF function can be applied for protection. However, if the speed interpolation command remains the same, it might cause position error. The adaptive feed rate modulation can be used to minimizae the position error in accordance with different situations.

Enable adaptive feed rate modulation of group 0:

1. Set P2-07 to 0x00010013.
2. Set P2-08 to 1h.
3. Set P2-06 to 0x00010016.

(This page is intentionally left blank.)

Communications 9

This chapter provides description of MODBUS which is used for setting, reading and writing general parameters via communication. For motion control network, please refer to the related documentations of DMCNET. Information about structures of ASCII, RTU and TCP mode is also provided in this chapter.

9.1 Communication parameters setting

Apart from parameter P3-00 (Address setting), the following parameters, such as P3-01 (Transmission speed), P3-02 (Communication protocol) and P3-05 (Communication mechanism), are essential and must be set when applying RS-232 / RS-485 to connect MS controller.

P3-24 (EtherNet network setting), P3-25 (EtherNet IP address setting), P3-26 (EtherNet subnet mask setting) and P3-27 (EtherNet default channel setting) are required when applying EtherNet communication. P3-06 (USB function switch) should be applied when using USB. The rest such as P3-08 (Monitoring mode) is optional.

Relevant parameters: Please refer to Chapter 8 for detailed description

Parameter	Function
P3-00	Address setting
P3-01	Transmission speed
P3-02	Communication protocol
P3-05	Communication mechanism
P3-06	USB function switch
P3-24	EtherNet network setting
P3-25	EtherNet IP address setting
P3-26	EtherNet subnet mask setting
P3-27	EtherNet default channel setting

9

9.2 MODBUS communication protocol

There are three modes of MODBUS network communication: ASCII (American Standard Code for Information Interchange), RTU (Remote Terminal Unit) and TCP (Transmission Control Protocol). Users could set the communication protocol (ASCII and RTU) required by RS-232/RS-485 via P3-02. Please note that TCP can only be applied in EtherNet communication and RTU is for USB-Serial only. MS controller also supports functions of accessing data (03H), writing one character (06H) and writing multiple characters (10H). Please refer to the following descriptions.

Code Description

ASCII mode:

In ASCII mode, data are transmitted in ASCII (American Standard Code for Information Interchange) format. For instance, when transmitting data 64H between two stations (Master and Slave), the master will send 36H to represent "6" and 34H to represent "4".

ASCII code for digits 0 to 9 and characters A to F are as follows:

Character	‘0’	‘1’	‘2’	‘3’	‘4’	‘5’	‘6’	‘7’
ASCII code	30H	31H	32H	33H	34H	35H	36H	37H
Character	‘8’	‘9’	‘A’	‘B’	‘C’	‘D’	‘E’	‘F’
ASCII code	38H	39H	41H	42H	43H	44H	45H	46H

RTU mode:

Every 8-bit data consists of two 4-bit characters (hexadecimal). If data 64H is transmitted between two stations, it will be transmitted directly, which is more efficient than ASCII mode.

TCP mode:

It's identical to RTU mode.

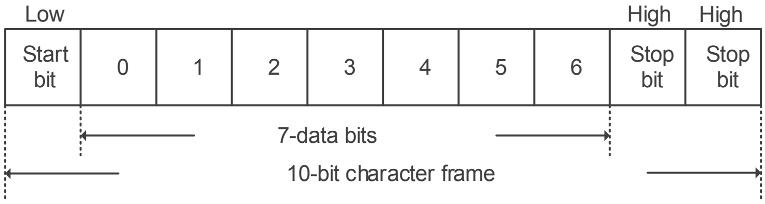
9

Character structure

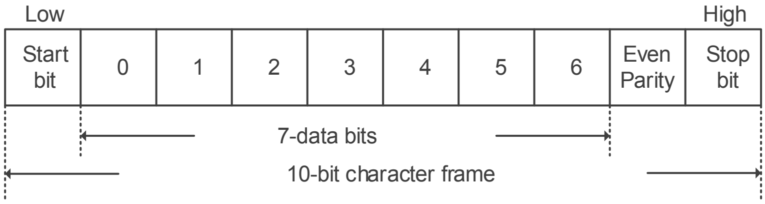
Characters will be encoded into the following framing and transmitted in serial. The checking method of different bit is as follows.

10-bit character frame (for 7-bit character)

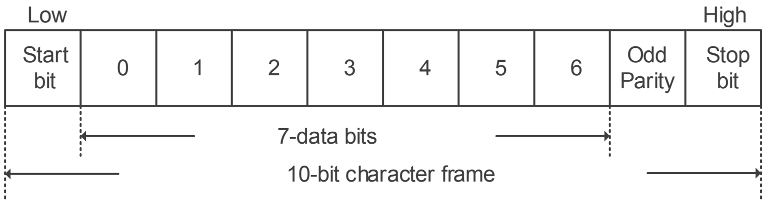
7N2



7E1

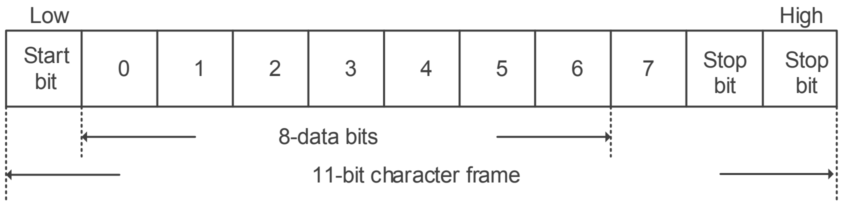


7O1

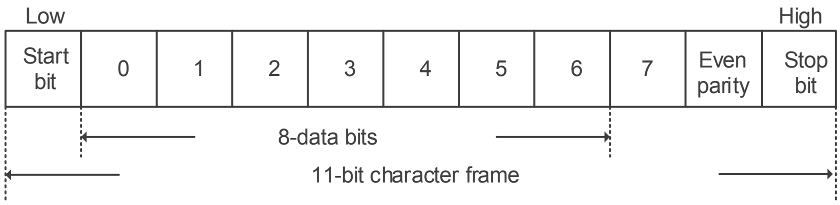


11-bit character frame (for 8-bit character)

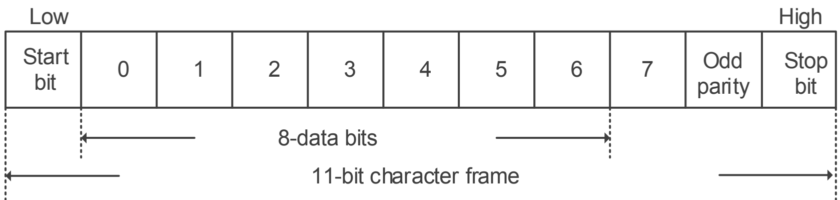
8N2



8E1



8O1



Communication data structure

Definitions of data frame for three modes are as below:

ASCII mode:

Start	Start character ":" (3AH)
Slave Address	Communication address: 1 byte consists of 2 ASCII codes
Function	Function code: 1 byte consists of 2 ASCII codes
Data (n-1)	Data content: n word = 2n-byte consists of n x 4 ASCII codes, n ≤ 10
…….	
Data (0)	
LRC	Error check: 1 byte consists of 2 ASCII codes
End 1	End code 1: (0DH)(CR)
End 0	End code 0：(0AH)(LF)

The start character of communication in ASCII mode is colon ":" (ASCII code: 3AH). ADR comprises two characters in ASCII code. The end code is CR (Carriage Return) and LF (Line Feed). The communication address, function code, data content and error checking LRC (Longitudinal Redundancy Check), etc. are between the start character and end code.

RTU mode:

Start	A silent interval of more than 10 ms
Slave Address	Communication address: 1 byte
Function	Function code: 1 byte
Data (n-1)	Data content: n-word = 2n-byte, n ≤ 10
…….	
Data (0)	
CRC	Error check: 1 byte
End 1	A silent interval of more than 10 ms

The start and the end of the communication in RTU (Remote Terminal Unit) mode are silent intervals. The communication address, function code, data content and error checking CRC (Cyclical Redundancy Check), etc. are between the start and the end.

TCP mode:

Start	Beginning of TCP packet
Transaction ID	Transaction ID: 2 byte
Protocol ID	Protocol ID: 2 byte
Length	Field length: 2 byte
Unit ID	Communication address: 1 byte
Function	Function code: 1 byte
Data (n-1)	Data content: n-word = 2n-byte, n ≤ 10
…….	
Data (0)	
End 1	End of TCP packet

TCP (Transmission Control Protocol) mode transmits data in a complete TCP packet. It starts with a TCP packet and ends with the same packet. The transaction ID, protocol ID, length, communication address, function code, data content and error checking CRC (Cyclical Redundancy Check), etc. are between the start and the end.

Example 1: function code 03H, accessing multiple words:

The master issues command to the 1st slave and reads continuous 2 words starting from the start data address 0200H. In the response message from the slave, the content of start data address 0200H is 00B1H, and the content of the 2nd data address is 1F40H. The maximum allowable data in one single access is 10. The calculation of LRC and CRC will be described in the following section.

ASCII mode:

Command Message (Master):

Start	‘:’
Slave Address	‘0’
	‘1’
Function	‘0’
	‘3’
Start Data Address	‘0’
	‘2’
	‘0’
	‘0’
Data Number (Word)	‘0’
	‘0’
	‘0’
	‘2’
LRC Check	‘F’
	‘8’
End 1	(0DH)(CR)
End 0	(0AH)(LF)

Response Message (Slave):

Start	‘:’
Slave Address	‘0’
	‘1’
Function	‘0’
	‘3’
Data Number (In byte)	‘0’
	‘4'
Content of Start Data Address 0200H	‘0’
	‘0’
	‘B’
	‘1’
Content of Second Data Address 0201H	‘1’
	‘F’
	‘4’
	‘0’
LRC Check	‘E’
	‘8’
End 1	(0DH)(CR)
End 0	(0AH)(LF)

9

RTU mode:

Command Message (Master):

Slave Address	01H
Function	03H
Start Data Address	02H (High)
	00H (Low)
Data Number (In word)	00H
	02H
CRC Check Low	C5H (Low)
CRC Check High	B3H (High)

Response Message (Slave):

Slave Address	01H
Function	03H
Data Number (In byte)	04H
Content of Start Data Address 0200H	00H (High)
	B1H (Low)
Content of Second Data Address 0201H	1FH (High)
	40H (Low)
CRC Check Low	A3H (Low)
CRC Check High	D4H (High)

Note: Before and after transmission in RTU mode, 10 ms of silent interval is needed.

TCP mode:

Command Message (Master):

Transaction ID	00H (High)
	01H (Low)
Protocol ID	00H (High)
	00H (Low)
Length	00H (High)
	06H (Low)
Unit ID	01H
Function	03H
Start Data Address	02H (High)
	00H (Low)
Data Number (In word)	00H
	02H

Response Message (Slave):

Transaction ID	00H (High)
	01H (Low)
Protocol ID	00H (High)
	00H (Low)
Length	00H (High)
	07H (Low)
Unit ID	01H
Function	03H
Data Number (In byte)	04H
Content of Start Data Address 0200H	00H (High)
	B1H (Low)
Content of Second Data Address 0201H	1FH (High)
	40H (Low)

Note: Length in TCP mode indicates the number of bytes in the message to follow.

Example 2: function code 06H, writing single words:

The master issues command to the 1st slave and writes data 0064H to address 0200H. The slave sends response message to the master after writing is completed. The calculation of LRC and CRC will be described in the following section.

ASCII mode:

Command Message (Master):

Start	‘:’
Slave Address	‘0’
	‘1’
Function	‘0’
	‘6’
Start Data Address	‘0’
	‘2’
	‘0’
	‘0’
Data Content	‘0’
	‘0’
	‘6’
	‘4’
LRC Check	‘9’
	‘3’
End 1	(0DH)(CR)
End 0	(0AH)(LF)

Response Message (Slave):

Start	‘:’
Slave Address	‘0’
	‘1’
Function	‘0’
	‘6’
Start Data Address	‘0’
	‘2'
	‘0’
	‘0’
Data Content	‘0’
	‘0’
	‘6’
	‘4’
LRC Check	‘9’
	‘3’
End 1	(0DH)(CR)
End 0	(0AH)(LF)

RTU mode:

Command Message (Master):

Address	01H
Slave Function	06H
Start Data Address	02H (High)
	00H (Low)
Data Content	00H (High)
	64H (Low)
CRC Check Low	89H (Low)
CRC Check High	99H (High)

Response Message (Slave):

Address	01H
Slave Function	06H
Start Data Address	02H (High)
	00H (Low)
Data Content	00H (High)
	64H (Low)
CRC Check Low	89H (Low)
CRC Check High	99H (High)

Note: Before and after transmission in RTU mode, 10 ms of silent interval is needed.

TCP mode:

Command Message (Master):

Transaction ID	00H (High)
	01H (Low)
Protocol ID	00H (High)
	00H (Low)
Length	00H (High)
	06H (Low)
Unit ID	01H
Slave Function	06H
Start Data Address	02H (High)
	00H (Low)
Data Content	00H (High)
	64H (Low)

Response Message (Slave):

Transaction ID	00H (High)
	01H (Low)
Protocol ID	00H (High)
	00H (Low)
Length	00H (High)
	06H (Low)
Unit ID	01H
Slave Function	06H
Start Data Address	02H (High)
	00H (Low)
Data Content	00H (High)
	64H (Low)

Note: Length in TCP mode indicates the number of bytes in the message to follow.

Example 3: function code10H, writing multiple words:

The master issues command to the 1st slave and writes data 0BB8H and 0000H to the start data address 0112H. That is to say, 0BB8H is written into 0112H and 0000H is written into 0113H. The maximum allowable data in one single access is 10. The slave sends the response message to the master after the writing is completed. The calculation of LRC and CRC will be described in the following section.

ASCII mode:

Command Message (Master):

Start	‘:’
Slave Address	‘0’
	‘1’
Function	‘1’
	‘0’
Start Data Address	‘0’
	‘1’
	‘1’
	‘2’
Data Number (In Word)	‘0’
	‘0’
	‘0’
	‘2’
Data Number (In Byte)	‘0’
	‘4’
Content of the 1st Data	‘0’
	‘B’
	‘B’
	‘8’
Content of the 2nd Data	‘0’
	‘0’
	‘0’
	‘0’
LRC Check	‘1’
	‘3’
End 1	(0DH)(CR)
End 0	(0AH)(LF)

Response Message (Slave):

Start	‘:’
Slave Address	‘0’
	‘1’
Function	‘1’
	‘0’
Start Data Address	‘0’
	‘1’
	‘1’
	‘2’
Data Number	‘0’
	‘0’
	‘0’
	‘2’
LRC Check	‘D’
	‘A’
End 1	(0DH)(CR)
End 0	(0AH)(LF)

9

RTU mode:

Command Message (Master):

Slave Address	01H	
Function	10H	
Start Data Address	01H (High)	
	12H (Low)	
Data Number (In Word)	00H (High)	
	02H (Low)	
Data Number (In Byte)	04H	
Content of the 1st Data	0BH (High)	
	B8H (Low)	
Content of the 2nd Data	00H (High)	
	00H (Low)	
CRC Check Low	FCH (Low)	
CRC Check High	EBH (High)	

Response Message (Slave):

Slave Address	01H
Function	10H
Start Data Address	01H (High)
	12H (Low)
Data Number (In Word)	00H (High)
	02H (Low)
CRC Check Low	E0H (Low)
CRC Check High	31H (High)

Note: Before and after transmission in RTU mode, 10 ms of silent interval is needed.

TCP mode:

Command Message (Master):

Transaction ID	00H (High)
	01H (Low)
Protocol ID	00H (High)
	00H (Low)
Length	00H (High)
	0BH (Low)
Unit ID	01H
Function	10H
Start Data Address	01H (High)
	12H (Low)
Data Number (In Word)	00H (High)
	02H (Low)
Data Number (In Byte)	04H
Content of the 1st Data	0BH (High)
	B8H (Low)
Content of the 2nd Data	00H (High)
	00H (Low)

Response Message (Slave):

Transaction ID	00H (High)
	01H (Low)
Protocol ID	00H (High)
	00H (Low)
Length	00H (High)
	06H (Low)
Unit ID	01H
Function	10H
Start Data Address	01H (High)
	12H (Low)
Data Number (In Word)	00H (High)
	02H (Low)

Note: Length in TCP mode indicates the number of bytes in the message to follow.

LRC and CRC transmission error check

The error check of ASCII mode is LRC (Longitudinal Redundancy Check) and CRC (Cyclical Redundancy Check) is for RTU mode. TCP mode inspects the error from the bottom. Additional error check of LRC or CRC is not required. See the description below:

LRC (ASCII mode):

Start	‘:’
Slave Address	‘7’
	‘F’
Function	‘0’
	‘3’
Start Data Address	‘0’
	‘5’
	‘C’
	‘4’
Data Number	‘0’
	‘0’
	‘0’
	‘1’
LRC Check	‘B’
	‘4’
End 1	(0DH)(CR)
End 0	(0AH)(LF)

The calculation of LRC is to add up all the byte, round down the carry and take 2’s complement.

For example:

7FH + 03H + 05H + C4H + 00H + 01H = 14CH, round down carry 1 and take 4CH.

2’s complement of 4CH is B4H.

9

CRC (RTU mode) :

The calculation description of CRC value is as the follows:

1. Load a 16-bit register of FFFFH, which is called "CRC" register.
2. (The low byte of CRC register) XOR (The first byte of command), and save the result in CRC register.
3. Check the least significant bit (LSB) of CRC register. If the bit is 0, right move one bit; If the bit is 1, then right move one bit and (CRC register) XOR (A001H). Repeat this step for 8 times.
4. Repeat the procedure from step 2 to step 3 until all byte is processed. The content of CRC register is the CRC value.

After calculating the CRC value, fill in the low word of CRC value in command message, and then the high word. For example, if the result of CRC calculation is 3794H, 94H should be filled in low word and 37H in high word which is shown as below:

ARD	01H
CMD	03H
Start Data Address	01H (High)
	01H (Low)
Data Number (In word)	00H (High)
	02H (Low)
CRC Check Low	94H (Low)
CRC Check High	37H (High)

Example of CRC program:

Calculate CRC value in C language. This function needs two parameters:

```
unsigned char* data;
unsigned char length
The function returns the CRC value as a type of unsigned integer.
unsigned int crc_chk(unsigned char* data, unsigned char length) {
    int j;
    unsigned int reg_crc=0xFFFF;

    while( length-- ) {
        reg_crc^= *data++;
        for (j=0; j<8; j++ ) {
            if( reg_crc & 0x01 ) { /*LSB(bit 0 ) = 1 */
                reg_crc = (reg_crc >> 1)^0xA001;
            } else {
                reg_crc = (reg_crc>>1);
            }
        }
    }
    return reg_crc;
}
```

Example of PC communication program:

```
#include<stdio.h>
#include<dos.h>
#include<conio.h>
#include<process.h>
#define PORT 0x03F8   /*  the address of COM 1  */
#define THR 0x0000
#define RDR 0x0000
#define BRDL 0x0000
#define IER 0x0001
#define BRDH 0x0001
#define LCR 0x0003
#define MCR 0x0004
#define LSR 0x0005
#define MSR 0x0006
unsigned char rdat[60];
/* read 2 data from address 0200H of ASD with address 1 */
unsigned char
tdat[60]={':','0','1','0','3','0','2','0','0','0','0','0','2','F','8','\r','
\n'};
void main() {
int I;
outportb(PORT+MCR,0x08);       /*  interrupt enable  */
outportb(PORT+IER,0x01);           /*  interrupt as data in  */
outportb(PORT+LCR,( inportb(PORT+LCR) | 0x80 ) );
/*  the BRDL/BRDH can be access as LCR.b7 == 1  */
```

9

```
outportb(PORT+BRDL,12);
outportb(PORT+BRDH,0x00);
outportb(PORT+LCR,0x06);        /* set prorocol
                                <7,E,1> = 1AH,        <7,O,1> = 0AH
                                <8,N,2> = 07H    <8,E,1> = 1BH
                                <8,O,1> = 0BH                  */

for( I = 0; I<=16; I++ ) {
     while( !(inportb(PORT+LSR) & 0x20) );   /*  wait until THR empty  */
     outportb(PORT+THR,tdat[I]);             /*  send data to THR  */
}
I = 0;
while( !kbhit() ) {
     if( inportb(PORT+LSR)&0x01 ) {  /*  b0==1, read data ready  */
          rdat[I++] = inportb(PORT+RDR); /*   read data from RDR  */
     }
}
}
```

9.3 Setting and accessing of communication parameters

For parameter details, please refer to Chapter 8. Descriptions of parameters can be written or read via communication.

Parameters are divided into 4 groups: Group 0: Monitoring parameters, Group 1: Basic parameters, Group 2: Application parameters and Group 3: Communication parameters.

Set parameters via communication:

Parameters which can be set via communication include:

Group 0, except (P0-00), (P0-02 ~ P0-03) and (P0-07 ~ P0-08).

Group 1, except (P1-02 ~ P1-05).

Group 2, except (P2-03 ~ P2-05), (P2-09 ~ P2-11) and (P2-14).

Group 3, except (P3-03 ~ P3-04), (P3-07), (P3-09 ~ P3-23), (P3-28 ~ P3-29) and (P3-31 ~ P3-42).

Please note that:

(P3-01) While changing to a new communication speed, the next data will be written with the new transmission speed after the new value is set.

(P3-02) While changing to a new communication protocol, the next data will be written with the new communication protocol after the new value is set.

(P3-06) The USB setting will be valid after writing the new setting value and re-power on MS controller.

Accessing parameters via communication:

Parameters which can be accessed via communication include:

Group 0 (P0-00 ~ P0-08)

Group 1 (P1-00 ~ P1-10)

Group 2 (P2-00 ~ P2-14)

Group 3 (P3-00 ~ P3-42)

(This page is intentionally left blank.)

Absolute System 10

This chapter introduces the application of absolute servo system, including the wiring and installation of absolute type encoder, setting steps and operation when initializing absolute position for the first time.

Note

A complete absolute servo system should include ASDA-MS controller, absolute motor and a backup battery box. With the battery that supplies power to the system, the encoder is able to work even when power is off. Moreover, absolute type encoder can continuously record the motor's actual position anytime even when the motor shaft is rotated after power off. The absolute servo system must work with absolute motor. Using incremental type motor with parameters of absolute system will cause alarm E?069.

When using an absolute motor, as soon as it applies to the power, the motor speed should not exceed 250 rpm. When operating with the battery, make sure the maximum speed does not exceed 200 rpm.

Check if your motor is absolute type. See the model name below:

ECMA - □ A □ □ □ □ □ □

A: ABS type motor

Please correctly install the battery to the encoder. One MS controller uses one single battery box. Please use Delta's encoder cable for connecting to Delta's battery box. See the following descriptions for the specifications of battery box and its accessories.

10.1 Battery box (Absolute type) and wiring rods

10.1.1 Specifications

Precautions

Please carefully read through the following safety precautions. Use batteries in accordance with the specification so as to avoid damages or dangers.

WARNING

- The installation location shall have no water drop, corrosive gas and inflammable gas.
- Correctly place the battery into the battery box so as to avoid short circuiting.
- Do not short circuit the positive electrode and negative electrode of the battery; or install the battery in reverse direction.
- It is suggested to use new batteries only. This is for avoiding losing electric energy or shortening the lifetime of new batteries.
- Please follow the instructions when wiring battery box, or danger may occur.

STOP

- Do not place the battery in a high-temperature environment (over 100℃) or it might result in fire or explosion.
- The batteries are non-rechargeable. Do not charge the batteries or it might result in explosion.
- Do not directly weld on the surface of the battery.

Battery specifications

Item	Li/SOCl2 Cylindrical Battery
Type	ER14505
Delta part number	ASD-CLBT0100
International standard size	AA
Standard voltage	3.6 V
Standard capacity	2700 mAh
Maximum continuous discharge current	100 mA
Maximum pulse current	200 mA
Dimensions (D x H)	14.5 x 50.5 mm
Weight	Approx. 19 g
Operating temperature	-40 ~ +85℃

10

Battery life

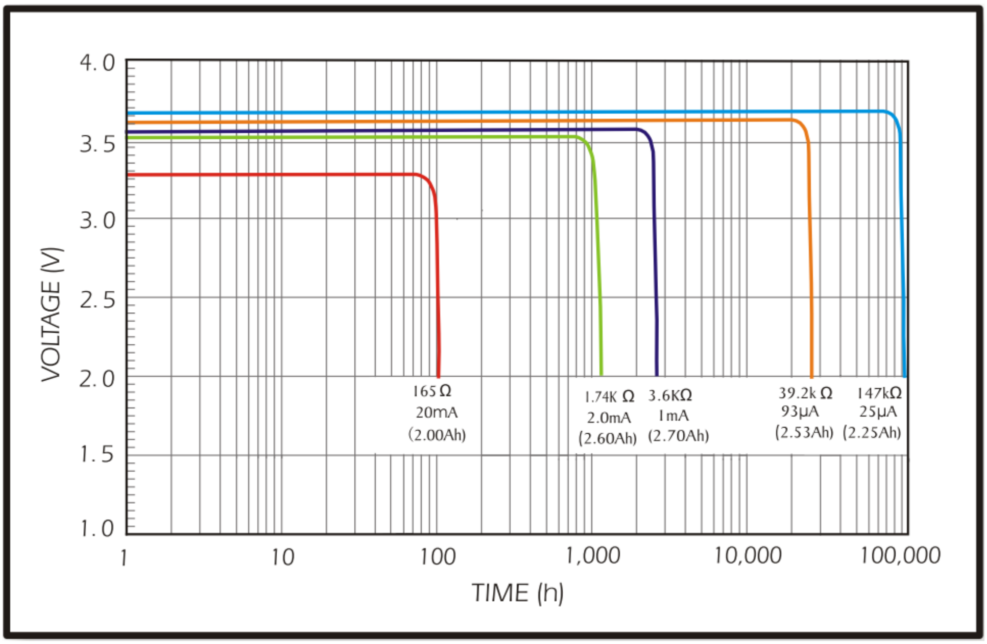


Figure 10.1.1.1 Curve of discharge current
(The above figure comes from EVE Energy Co. ER14505 Discharge Characteristics)

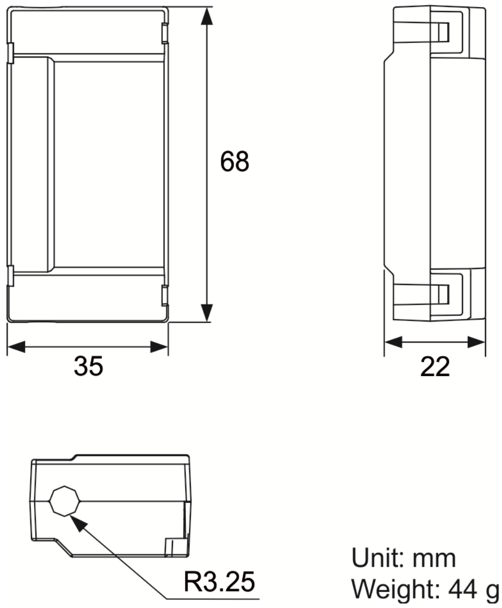
1. The above figure illustrates the discharge current curve generated by constant current test. See the testing result shown on the graph above. When the power consumption of an absolute encoder is 65 uA or lower and the voltage of the battery keeps 3 V or higher, the expected battery life is about 21900 hr, approximately 2.5 years (Note). Therefore, the lowest voltage level of battery for an absolute encoder is set to 3.1 V.
2. The battery life expectancy is about 5 years and is able to provide 3.6 V or higher voltage at normal temperature and humidity level.

Note: The battery life was measured when one single battery box is connected to one MS controller and one servo motor.

10.1.2 Battery box dimensions

Single battery box

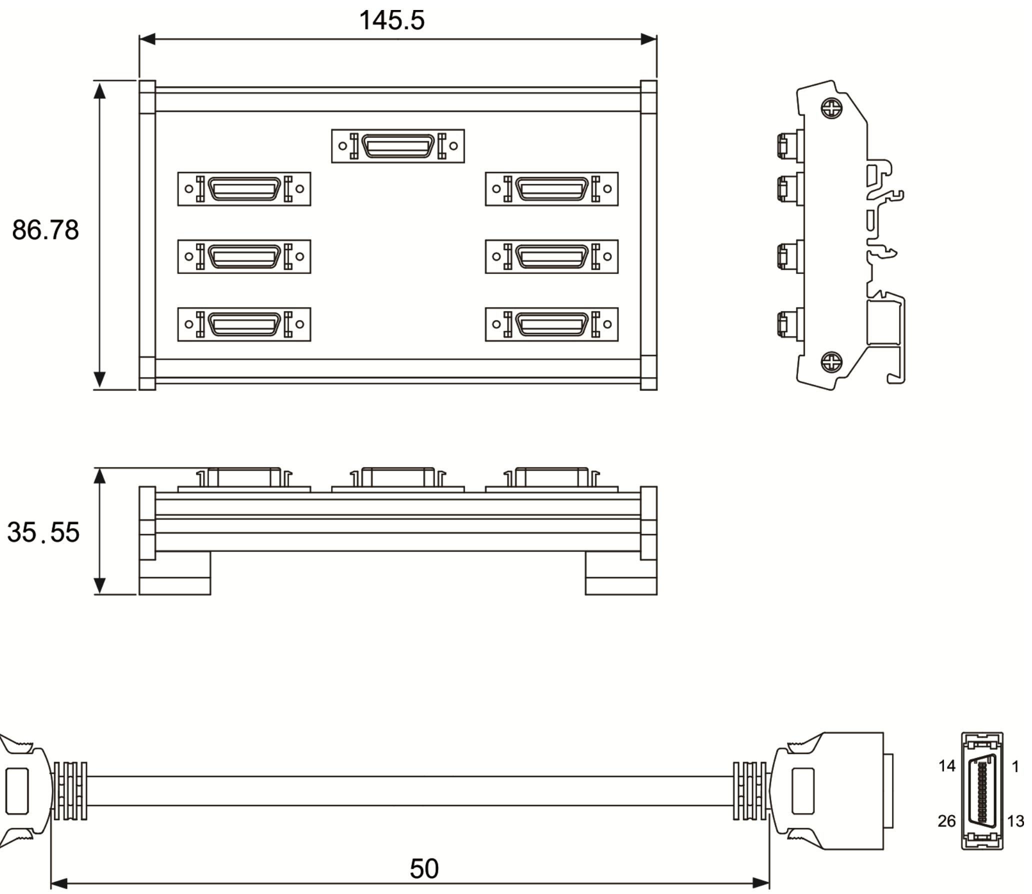
Delta part number: ASD-MDBT0100



Unit: mm
Weight: 44 g

Encoder conversion module

Part Number: ASDPBSC2626



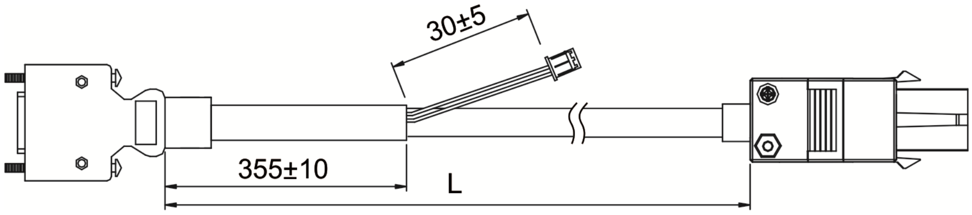
Unit: mm

10

10.1.3 Connection cable for absolute encoder

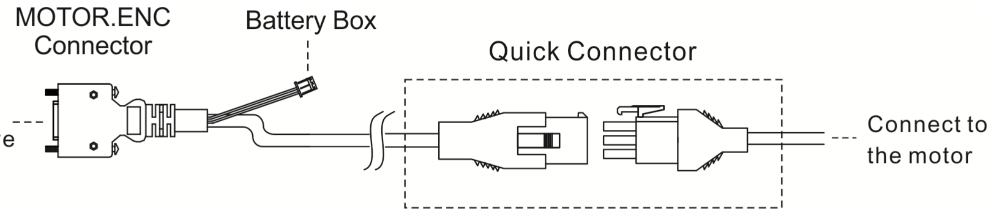
A. Quick connector

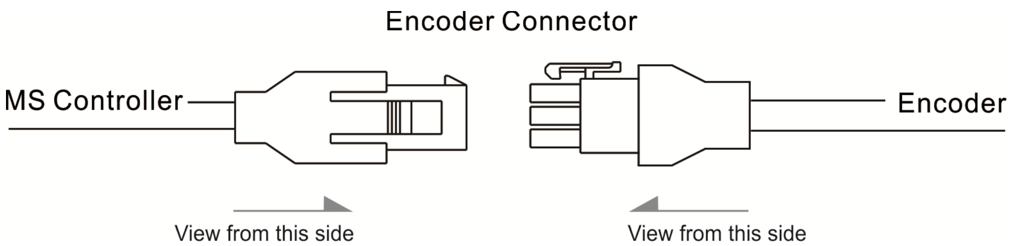
Delta Part Number: ASD-A2EB0003, ASD-A2EB0005



Connection method:

Note Please follow the instructions below when conduct wiring. Wrong wiring might result in explosion.





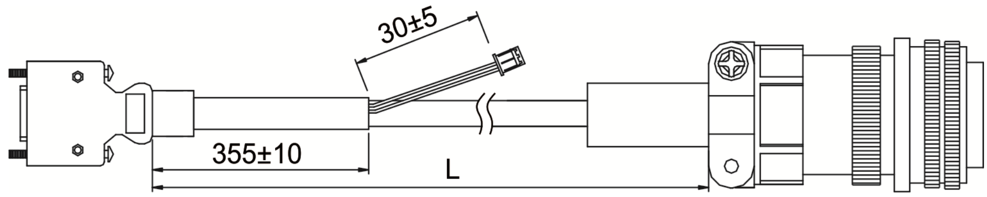
1 Blue T+	2 Green Reserved	3 Reserved
4 Blue/Black T-	5 Green/Black Reserved	6 Reserved
7 Red/Red&WHE DC+5V	8 BLK/BLK&WHE GND	9 Shield

3 Reserved	2 Black Reserved	1 White T+
6 Reserved	5 Red/Black Reserved	4 White/Red T-
9 Shield	8 Blue GND	7 Brown DC+5V

The wire color is for reference only. It should base on the real object.

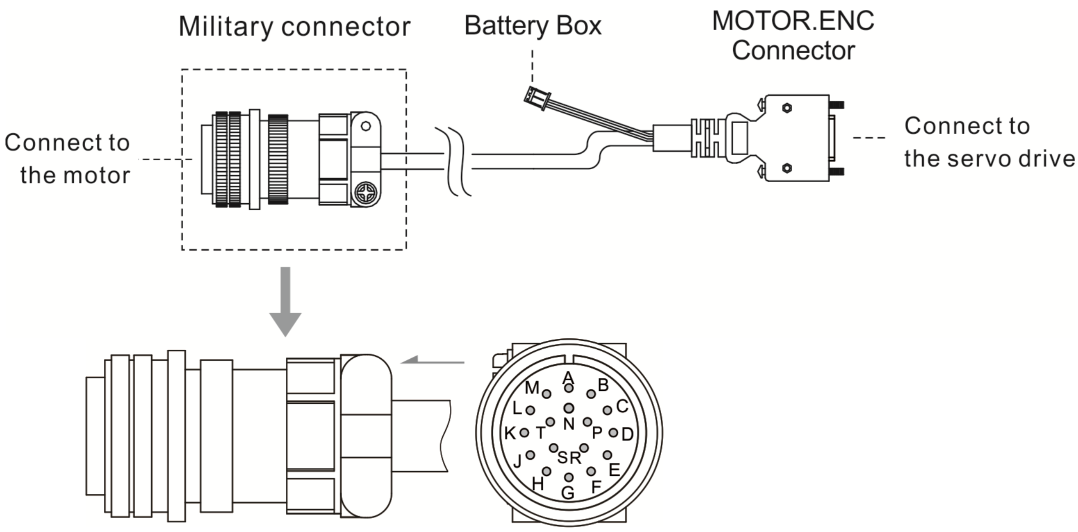
B. Military connector

Delta Part Number: ASD-A2EB1003, ASD-A2EB1005



Connection method:

Note Please follow the instructions below when conduct wiring. Wrong wiring might result in explosion.



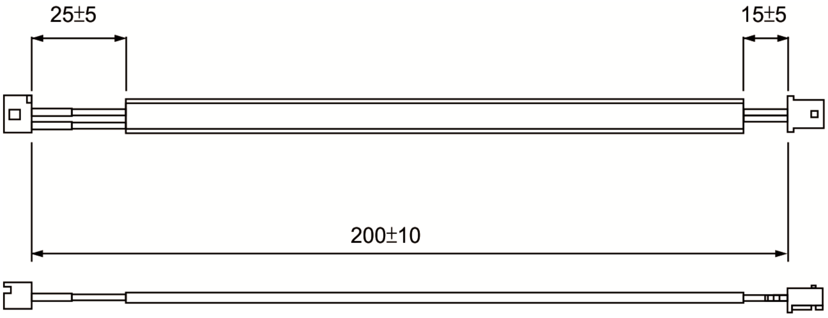
Pin No.	Terminal	Color
A	T+	Blue
B	T -	Blue & Black
S	+5 VDC	Red/Red&White
R	GND	Black/Black&White
L	BRAID SHIELD	–

10

10.1.4 Battery box cable

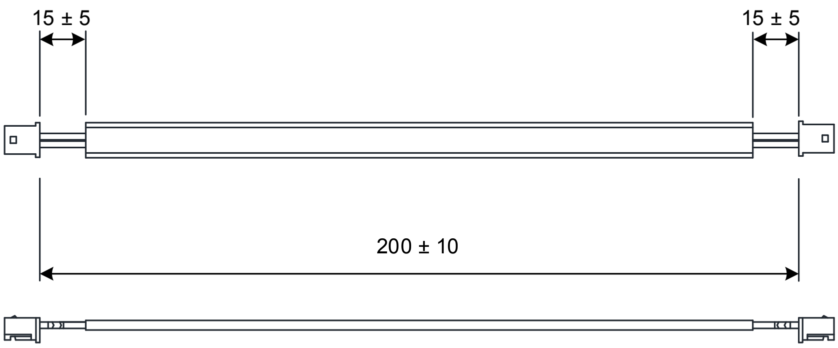
Battery box cable AW

Delta Part Number: 3864573700



Battery box cable IW

Delta Part Number: 3864811900

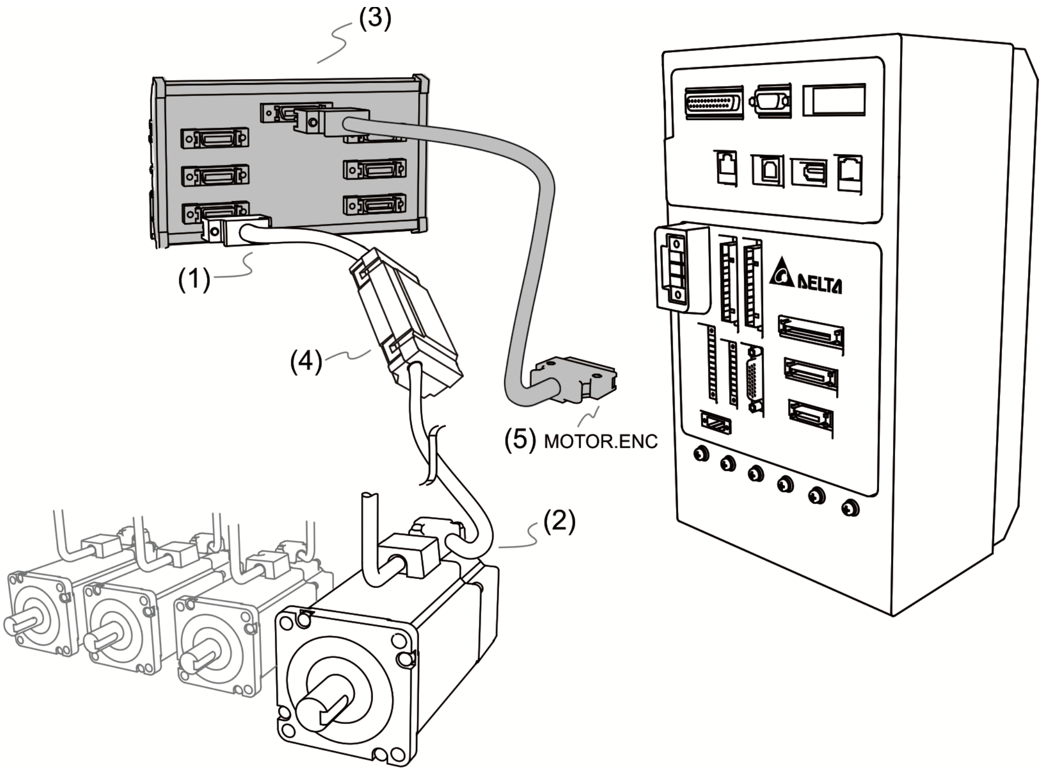


10

10.2 Installation

10.2.1 Install battery box in servo system

Single battery box (standard wiring)



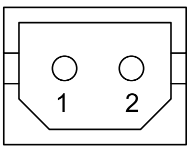
Please refer to section 3.1.5 for the wiring of (1) and (2).

(3) See the following for the definition of battery connection cable:

Note Please follow the instructions below when conduct wiring. Wrong wiring might result in explosion.

Pin No.	Terminal
1	BAT+
2	BAT-

(4) Connect to the power supply of single battery box. See the details below:



Pin No.	Terminal	Cable color
1	BAT+	Red
2	BAT-	Black

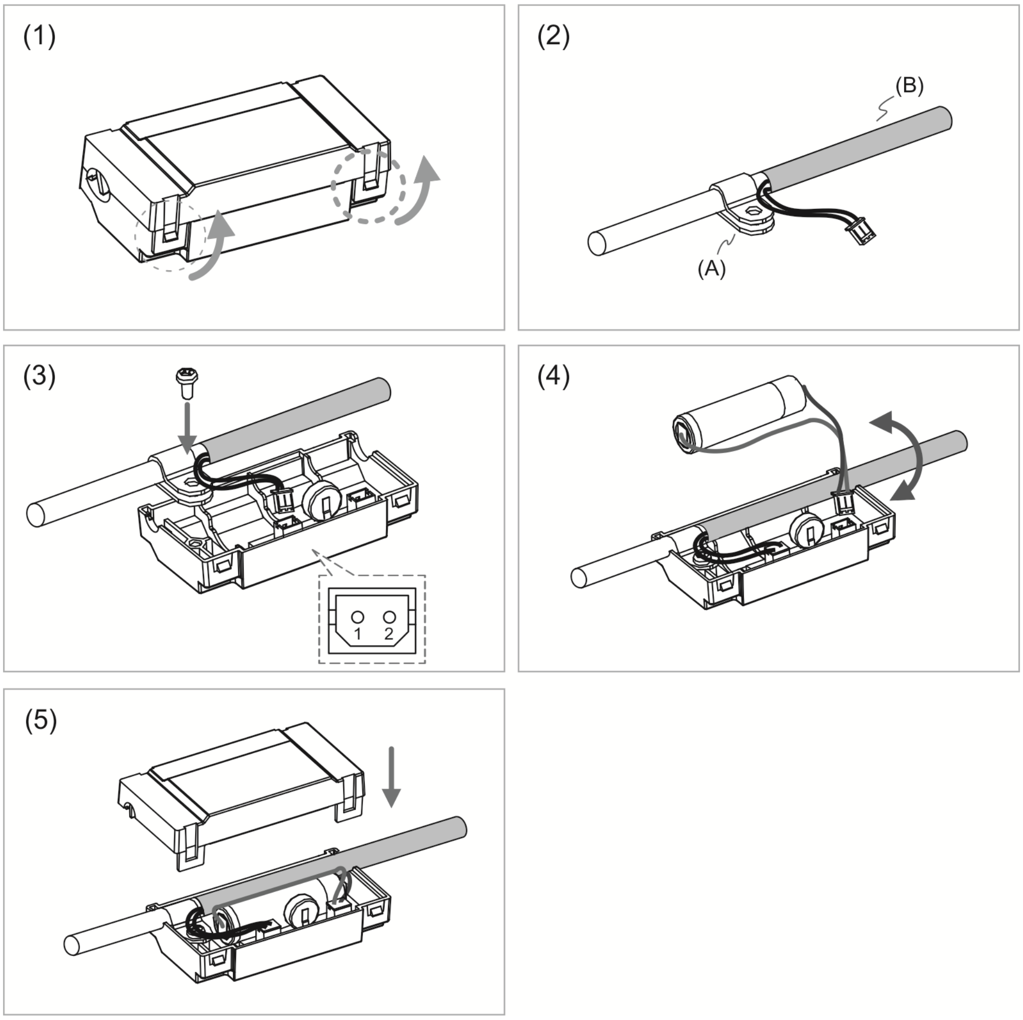
(5) MOTOR.ENC connector

Note: This is the wiring diagram of connecting single battery box, which is not drawn to scale. For different models of MS controller and motors, the connection cables may differ.

10

10.2.2 How to install the battery

Single battery box



(1) Loosen the hooks on both sides to open the lid of battery box.

(2) Put the metal clip on connection cable. Please note that the metal clip should be put close to the heat shrink.

(A) Metal clip；(B) Heat shrink

(3) Plug the connection cable and tighten the screw.

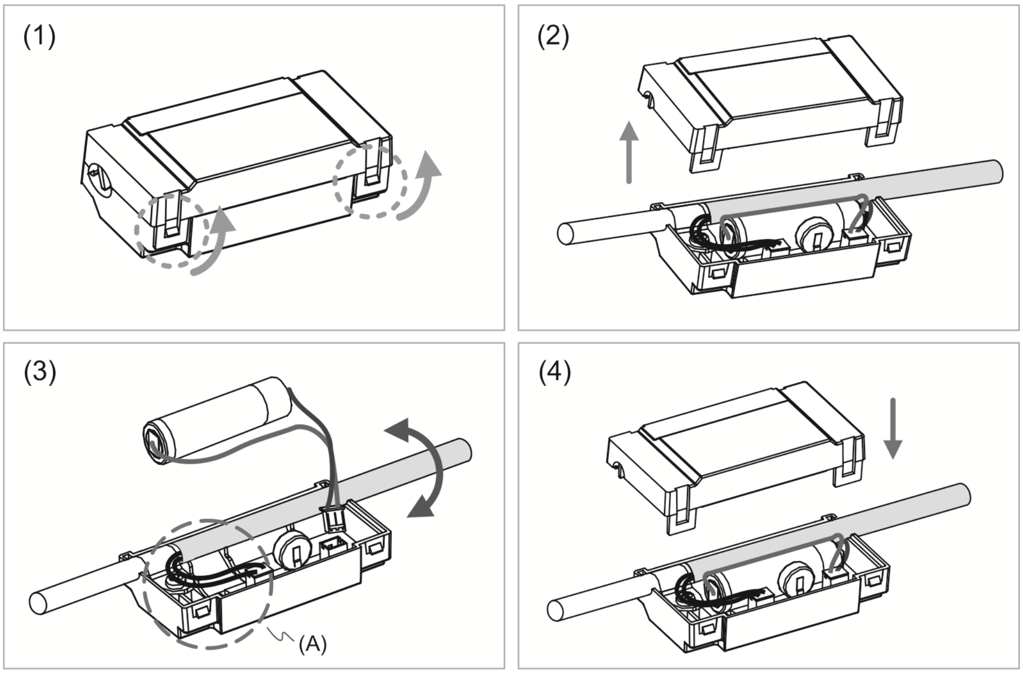
(4) Install a new battery and connect it to the cable.

(5) Place the cable into the box and cover the lid back.

10.2.3 How to replace a battery

For avoid data lost, please replace a new battery when any of the circumstances happens mentioned hereunder: 1. MS controller shows alarm E?061, which means the voltage is too low (please refer to Chapter 11 for further information). Users can use P0-02 (monitoring variable 26h) to check the battery power. When it displays 31, it means the voltage is under 3.1 V. When the voltage is under 2.7 V, motor's position record might be lost. Please execute homing after replacing a new battery.
Note For avoiding data loss, it is recommended to replace the new battery when the MS controller still has power supply.

Single battery box



(1) Loosen the hooks on both sides to open the lid of battery box.
(2) Fully open the top cover.
(3) Disconnect the connector and remove the old battery. Then, replace with the new one and connect the connection cable again.
(A) Please replace the battery when the power is still supplied to MS controller. Do not remove the power cable, otherwise it might cause data lost.
(4) Place the cable into the box and cover the lid back.

10.3 Parameters of absolute system

Parameter	Function
P2-69	Absolute encoder setting
P2-71	Absolute position homing

10.4 Alarm list for absolute function and monitoring variables

Display	Alarm name
E?028	Encoder voltage error or the internal of the encoder is in error
E?029	Gray code error
E?034	Internal communication of the encoder is in error
E?060	The absolute position is lost
E?061	Encoder under voltage
E?062	The multi-turn of absolute encoder overflows
E?069	Wrong motor type
E?289	Feedback position counter overflows

Monitoring variables

Code	Name	Description
038 (26h)	Voltage level of the battery	The battery’s voltage level of the absolute encoder

10.5 System initialization and operation procedures

10.5.1 System initialization

E?060 will occur when the absolute system is enabled for the first time. This is because the coordinate system has not been created. The alarm will be cleared until the setting of coordinate system is complete. Low battery power or the failure of power supply will lead to coordinate system loss and the occurrence of E?060. In absolute system, when the motor's rotation number exceeds the range from -32768 to 32767, E?062 will occur. In terms of PUU, the position value should be between -2147483648 and 2147483647, or E?289 will occur.

10.5.2 PUU number

PUU number is a 32-bit absolute data with positive and negative sign. When the motor is running in forward direction, the PUU number will increase; when it is in reverse direction, the PUU number will decrease. The forward direction does not mean the motor running in clockwise direction. It should be defined by P1-01.Z.

Range of the maximum counting number is from -32768 to +32767. E?062 will occur when the cycle number exceeds the range (overflows). If the PUU number exceeds the range between -2147483648 and 2147483647, the position counter overflows and E?289 occurs. Users have to re-initialize the system to clear alarms (E?062 or E?289).

See the following examples:

Example 1: When P1-44 = 128 and P1-45 = 0, the motor needs 100000 PUU to run a cycle.
And 2147483647÷100000≒21474.8. Thus, once the motor runs over 21474.8 (< 32767) cycles in forward direction, E?289 occurs.

Example 2: When P1-44 = 128 and P1-45 = 1, the motor needs 10000 PUU to run a cycle.
And 2147483647÷10000≒214748.3. Thus, once the motor runs over 32767 (< 214748.3) cycles in forward direction, E?062 occurs.

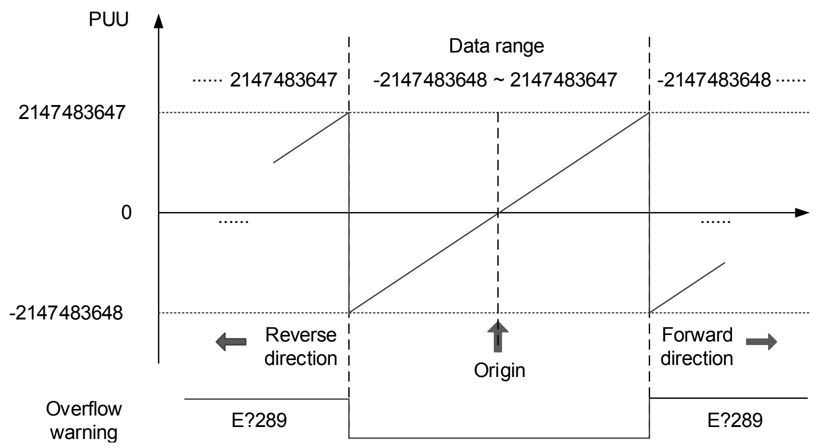


Figure 10.5.2.1 Absolute position of PUU counting

10

10.5.3 To initialize the absolute coordinate via parameters

Users can set P2-71 to 1 to initialize the coordinates via communication. As soon as P2-71 is set to 1, the absolute system will be reset. Since the write-in function of P2-71 is protected by P2-08, users have to set P2-08 to 271 first. Thus, you should set P2-08 to 271 first. Then, set P2-71 to 1. Please note that this method can be applied to all modes except DMCNET. For DMCNET mode, please execute homing to reset the coordinate.

Note:

1. After initializing the absolute coordinate system, any change on P1-01.Z or e-gear ratio (P1-44 and P1-45) will change the setting of absolute coordinate system. If the setting of the above parameters is changed, please re-initialize the coordinate system.
2. Please initialize the absolute coordinate to clear alarm E?060.

Troubleshooting 11

This chapter provides alarm descriptions and corrective actions that can be used for troubleshooting.

11

Alarms can be categorized into four groups, which are Controller, User, Group and Axis. See detailed information below:

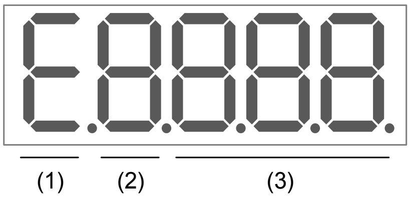
“Controller”: Alarm generated by the controller; this type alarm is reserved for now.

“User”: User-defined alarm in PLC program.

“Group”: Alarm generated by the axial group, which can be composed of any axis.

“Axis”: Alarm generated by each axis.

See the display of 7-segment LED below:

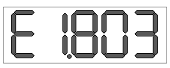


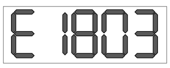
(1) “**E**” is fixed displayed as alarm.

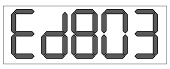
(2) Alarm type

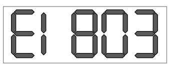
Controller:	Display with English character “**C**”. This type of alarm is reserved for now.
User:	Display with English character “**U**”
Group:	Display with number “**1. ~ 2.**”. *In the following alarm list, “**?**” is used to represent the unknown number.
Axis:	Axis 1 ~ 6: display with “**1**” ~ “**6**” Axis 7 ~ 12: Reserved. Axis 13 ~ 18: display with English character “**D**” ~ “**I**” *In the following alarm list, “**?**” is used to represent the unknown number and English character.

Example:


E1.803, the alarm from group 1 of Group type.


E1803, the alarm from axis 1 of Axis type.


ED803, the alarm from axis 13 of Axis type.


EI803, the alarm from axis 18 of Axis type.

(3) Alarm code

11.1 Alarm list

Group:

Display	Alarm name	Error type		Servo state	
		ALM	WARN	ON	OFF
E?801	Homing of all axes is not complete	○			○
E?803	Motion command conflict	○			○
E?80A	Motion command is not ready	○			○
E?80B	Unknown motion command	○			○
E?80C	Incorrect buffer for motion command	○			○
E?813	Grouped axis error	○			○
E?814	Axis error	○			○
E?815	Exceed the software limit	○			○
E?821	Arm posture does not match	○			○
E?822	PTP motion exceeds the working space	○			○
E?823	Linear motion exceeds the working space	○			○
E?824	Motion of the axis exceeds the working space	○			○
E?825	Forward kinematics computing error	○			○
E?827	Group does not exist	○			○
E?829	Error occurs when switching coordinate system	○			○
E?82A	PCS switching error	○			○
E?82B	TCS switching error	○			○
E?82C	Motion command exceeds the working area	○			○
E?82D	Arm’s motion exceeds the setting range	○			○
E?832	Internal communication packet is lost	○			○
E?833	Internal communication checksum error	○			○
E?841	The arc command exceeds the range	○			○
E?842	Arc cannot be formed	○			○
E?843	Arc command error	○			○
E?851	Conveyor tracking- result timeout when transmitting visual recognition	○			○
E?852	Conveyor tracking- the speed exceeds the limit	○			○
E?853	Conveyor tracking-PCS error	○			○
E?861	Jogging of TP MPG is too fast	○		○	
E?862	Jogging of TP MPG is in progress	○		○	

Note:
1. When the occurring alarm is not mentioned in the above table, please contact local distributors.
2. “**?**” represents the number “**1. ~ 2.**” in group type alarm.

11

Axis:

Display	Alarm name	Error type		Servo state	
		ALM	WARN	ON	OFF
E?001	Overcurrent	○			○
E?002	Overvoltage	○			○
E?003	Under voltage		○		○
E?004	Motor combination error	○			○
E?005	Regeneration error	○			○
E?006	Overload	○			○
E?007	Overspeed	○			○
E?009	Excessive deviation of position command	○			○
E?011	Encoder error	○			○
E?012	Adjustment error	○			○
E?013	Emergency stop		○		○
E?014	Reverse limit error		○		○
E?015	Forward limit error		○	○	
E?016	IGBT overheat	○			○
E?017	Abnormal EEPROM	○			○
E?018	Abnormal signal output	○			○
E?019	Serial communication error	○			○
E?020	Serial communication timeout		○	○	
E?022	RST leak phase		○		○
E?023	Early warning for overload		○	○	
E?024	Encoder initial magnetic field error	○			○
E?025	Internal error of the encoder	○			○
E?026	Unreliable internal data of the encoder	○			○
E?027	The internal of the motor is in error	○			○
E?028	Encoder voltage error or the internal of the encoder is in error	○			○
E?029	Gray code error	○			○
E?030	Motor crash error	○			○
E?031	Incorrect wiring of motor power cable	○			○
E?034	Internal communication of the encoder is in error	○			○
E?044	Warning of servo function overload		○		○
E?060	The absolute position is lost		○		○
E?061	Encoder under voltage		○	○	
E?062	The multi-turn of absolute encoder overflows		○	○	
E?067	Encoder temperature warning		○	○	
E?069	Wrong motor type	○			○

Display	Alarm name	Error type		Servo state	
		ALM	WARN	ON	OFF
E?06A	The absolute position is lost				
E?06C	MS controller connects to CA type and CW type motor simultaneously	○			○
E?06D	Check procedure is in error when power is supplied to the encoder	○			○
E?070	Encoder does not complete the command which is issued by servo drive		○		○
E?099	EEPROM must be updated	○			○
E?09A	DSP AD1 error	○			○
E?09B	DSP AD2 error	○			○
E?111	Buffer overflow occurs when receiving data via DMCNET	○			○
E?185	DMCNET Bus hardware error	○			○
E?201	An error occurs when loading DMCNET data	○			○
E?235	Position overflows	○			○
E?245	Positioning is overtime	○			○
E?283	Software positive limit		○	○	
E?285	Software reverse limit		○	○	
E?289	Feedback position counter overflows	○			○
E?301	DMCNET synchronization failure	○			○
E?302	The synchronized signal of DMCNET is sent too fast	○			○
E?303	The synchronized signal of DMCNET is sent too slow	○			○
E?304	DMCNET IP command failed	○			○
E?500	STO function is enabled	○			○
E?501	STO_A lost (Signal loss or signal error)	○			○
E?502	STO_B lost (Signal loss or signal error)	○			○
E?503	STO_error	○			○
E?555	System failure	○			○

Note:
1. "**?**" represents the number "**1. ~ 2.**" in group type alarm.

11

Controller:

Display	Alarm name	Error type		Servo state	
		ALM	WARN	ON	OFF
EC001	PLC timeout	○			○
EC002	PLC image download failed	○			○
EC003	PLC Exception	○			○
EC004	Motion module failure	○			○
EC005	Controller failure	○			○
EC006	Write-in error (continuous 30 sec)	○			○
EC007	DMCNET device setting does not match	○			○
EC008	Robot parameter file loading error	○			○
EC009	Robot type is different from the setting	○			○
EC010	Frequent alarm reset	○			○

Note:
1. Please refer to the detailed description for EC003, exception code.

11.2 Causes and corrective actions

Group:

E?801 Homing of all axes is not complete	
Causes	Homing of all axes is not complete
Checking methods and corrective actions	If robot arm moves based on spatial coordinates (X-, Y- and Z-axis direction) before homing of all axes is not complete, this alarm will occur. Please conduct homing for all axes.
How to clear?	Alarm reset

E?803 Motion command conflict	
Causes	Different motion commands are combined for one path.
Checking methods and corrective actions	Check if single-axis PTP command, multi-axis PTP command and multi-axis linear command are combined in one path. These motion commands cannot be combined together. Please replace with other motion commands or avoid command combination.
How to clear?	Alarm reset

E?80A Motion command is not ready	
Causes	Motion command interpreter can only buffer two commands. If the buffer zone is full, the third motion command will not be interpreted.
Checking methods and corrective actions	Check if the PLC or Lua motion command edited by users has buffer mode (ButterMode = Buffered). Do not issue more than two motion commands continuously.
How to clear?	Alarm reset

E?80B Unknown motion command	
Causes	Motion command cannot be identified.
Checking methods and corrective actions	Please refer to the motion command list and check if the command code you applied is correct. If not, please use the motion command supported by Delta (Refer to Chapter 6 (6.4) Command description)
How to clear?	Alarm reset

E?80C Incorrect buffer for motion command	
Causes	Incorrect buffer for motion command
Checking methods and corrective actions	Make sure the motion command is stored in the correct buffer zone (single-axis and multi-axis commands should be stored in different buffer zones). Please refer to motion command table.
How to clear?	Alarm reset

E?813 Grouped axis error	
Causes	1. Grouped axis error occurs during command interpreting. 2. Grouped axis error occurs when executing motion command.
Checking methods and corrective actions	1. Please use DRAS to check the alarm occurrence of grouped axis and troubleshoot the problem. 2. Please use DRAS to see if the servo can be normally activated. 3. Check if the motion command is executed when homing. If yes, please complete homing first before executing the motion command.
How to clear?	Alarm reset

E?814 Axis error	
Causes	1. Axis error occurs during command interpreting. 2. Axis error occurs when executing motion command.
Checking methods and corrective actions	Please use DRAS to check alarm occurrence of the axis and troubleshoot the problem.
How to clear?	Alarm reset

E?815 Exceed the software limit	
Causes	The target position of single axis exceeds the software limit.
Checking methods and corrective actions	Please make sure the target position of each axis is within the range between software positive and negative limit (set by servo parameter P5-08 and P5-09). If the position is not within the range, please change the target position or reset the limit value in accordance with the actual application.
How to clear?	Alarm reset

E?821 Arm posture does not match	
Causes	Robot arm's current posture does not match the posture setting for reaching the target.
Checking methods and corrective actions	1. Moving command (MovL) does not support robot arm switching. Please check if the arm's current posture is identical to the posture setting for reaching the target. If not, please change the posture or apply another motion command. 2. Please select "Discard posture setting" for the robot arm. Posture of the robot arm will be determined by the controller.
How to clear?	Alarm reset

E?822 PTP motion exceeds the working space	
Causes	Target position of PTP command exceeds the working space.
Checking methods and corrective actions	Please make sure the target position of each axis is within the range between software positive and negative limit (set by servo parameter P5-08 and P5-09). If the position is not within the range, please change the target position or reset the limit value in accordance with the actual application.
How to clear?	Alarm reset

E?823 Linear motion exceeds the working space	
Causes	The target position of linear command exceeds the working space.
Checking methods and corrective actions	Please make sure the target position of each axis is within the range between software positive and negative limit (set by servo parameter P5-08 and P5-09). If the position is not within the range, please change the target position or reset the limit value in accordance with the actual application.
How to clear?	Alarm reset

E?824 Motion of the axis exceeds the working space	
Causes	The axis exceeds the working space when executing the motion command.
Checking methods and corrective actions	Please make sure the target position of each axis is within the range between software positive and negative limit (set by servo parameter P5-08 and P5-09). If the position is not within the range, please change the target position or reset the limit value in accordance with the actual application.
How to clear?	Alarm reset

E?825 Forward kinematics computing error	
Causes	Forward kinematics computing error
Checking methods and corrective actions	1. Please make sure the target position of each axis is within the range between software positive and negative limit (set by servo parameter P5-08 and P5-09). If the position is not within the range, please change the target position or reset the limit value in accordance with the actual application. 2. Check if the motion path is within the setting range. 3. Contact the original manufacturer for the correct robot dimensions.
How to clear?	Alarm reset

E?827 Group does not exist	
Causes	The specified group does not exist
Checking methods and corrective actions	Please check if the specified group is group 1. If not, please specify the group to 1.
How to clear?	Alarm reset

E?829 Error occurs when switching coordinate system	
Causes	The coordinate system to be switched to does not exist.
Checking methods and corrective actions	Check if the specified number of coordinate system is between 0 and 9. If not, please input the correct number. The system supports machine coordinate system (MCS), tool coordinate system (TCS), product coordinate system (PCS) and axes coordinate system (ACS) for now.
How to clear?	Alarm reset

E?82A PCS switching error	
Causes	Product coordinate system switching error
Checking methods and corrective actions	1. Check if the specified number of coordinate system is between 0 and 9. If not, please input the correct number. 2. Check and test if the Teach function works properly. (Please refer to Chapter 7 (7.3) for detailed information.)
How to clear?	Alarm reset

E?82B TCS switching error	
Causes	Tool coordinate system switching error
Checking methods and corrective actions	1. Check if the specified number of coordinate system is between 0 and 9. If not, please input the correct number. 2. Check and test if the Teach function works properly. (Please refer to Chapter 7 (7.4) for detailed information.)
How to clear?	Alarm reset

E?82C Motion command exceeds the working area	
Causes	Motion command exceeds the set working area or is within the restricted area.
Checking methods and corrective actions	Check if the target position of motion command exceeds the working area or is within the restricted area. If yes, please change the target position through DRAS.
How to clear?	Alarm reset

E?82D Arm's motion exceeds the setting range	
Causes	Robot arm exceeds the set working area or is within the restricted area.
Checking methods and corrective actions	1. Check if the arm's current position of motion command exceeds the working area or is within the restricted area. 2. If the arm is in non-working area, please disable **Working area setup** and move the arm to the working area.
How to clear?	Alarm reset

E?832 Internal communication packet is lost	
Causes	The internal communication packet is lost for 3 times
Checking methods and corrective actions	1. Check if servo parameter P0-00 can be accessed through DRAS. Please contact Delta if this alarm occurs. 2. Check if the system is resetting the parameter. If yes, please reset the alarm afterwards. And contact Delta if this alarm occurs again.
How to clear?	Alarm reset

E?833 Internal communication checksum error	
Causes	Communication packet checksum error occurs for 3 times
Checking methods and corrective actions	1. Check if servo parameter P0-00 can be accessed through DRAS. Please contact Delta if this alarm occurs. 2. Eliminate the source of disturbance around MS controller. If the alarm still occurs, please contact Delta.
How to clear?	Alarm reset

E?841 The arc command exceeds the range	
Causes	Target position exceeds the working space.
Checking methods and corrective actions	Please make sure the target position of each axis is within the range between software positive and negative limit (set by servo parameter P5-08 and P5-09). If the position is not within the range, please change the target position or reset the limit value in accordance with the actual application.
How to clear?	Alarm reset

E?842 Arc cannot be formed	
Causes	Arc cannot be formed by the input data
Checking methods and corrective actions	Make sure the input data for forming an arc is correct. The arc cannot be formed in following conditions: issuing three target positions for consisting an arc when three positions are in the same line, the radius is 0, or the circle center is located on the circumference. Please issue the position command again in accordance with the condition to form an arc. (Refer to Chapter 6 (6.4.3) for detailed information.)
How to clear?	Alarm reset

E?843 Arc command error	
Causes	Arc command error
Checking methods and corrective actions	Make sure the input data for forming an arc is correct. The arc cannot be formed in following conditions: issuing three target positions for constituting an arc when three positions are in the same line, the radius is 0, or the circle center is located on the circumference. Please issue the position command again in accordance with the condition to form an arc. (Refer to Chapter 6 (6.4.3) for detailed information.)
How to clear?	Alarm reset

E?851 Conveyor tracking- result timeout when transmitting visual recognition	
Causes	1. Material transmission error 2. Visual recognition is not triggered
Checking methods and corrective actions	1. Make sure the visual recognition is triggered before activating the robot system. 2. Make sure the system configuration and setting of visual recognition are all correct.
How to clear?	Alarm reset

E?852 Conveyor tracking- the speed exceeds the limit	
Causes	Conveyor exceeds the speed limit
Checking methods and corrective actions	Slow down the conveyor speed
How to clear?	Alarm reset

E?853 Conveyor tracking- PCS error	
Causes	PCS number setting error in conveyor tracking application
Checking methods and corrective actions	Make sure the PCS number in conveyor tracking application is within the setting range (cannot be 0 and no larger than 9).
How to clear?	Alarm reset

E?861 Jogging of TP MPG is too fast	
Causes	Jogging of TP MPG is too fast
Checking methods and corrective actions	Reduce the jog speed
How to clear?	Alarm reset

E?862 Jogging of TP MPG is in progress	
Causes	Jogging of TP MPG is in progress and other commands cannot be executed.
Checking methods and corrective actions	Stop jogging and execute the commands which should be completed beforehand.
How to clear?	Alarm reset

Axis:

E?001 Overcurrent	
Causes	1. MS controller output is short-circuited. 2. Motor wiring is wrong. 3. IGBT is abnormal. 4. Setting of command is wrong. 5. A fierce change in position/speed/torque.
Checking methods and corrective actions	1. Check if the wiring between motor and MS controller is correct and see if the wire is short-circuited. Eliminate the short circuit and prevent metal conductor from being exposed. 2. Refer to the user manual for wiring steps and correct the wiring between motor and MS controller. 3. Use monitoring variable (0x10) to check if the temperature of IGBT exceeds 117 degrees. If yes, please send your MS controller back to local distributors or contact Delta. 4. Please check if the parameter setting matches the default value in accordance with the actual watt of motor and MS controller. Check servo parameter H1-00 and H1-01: 750 W (H1-00 = 1d, H1-01 = 4d); 1500 W (H1-00 = 1d, H1-01 = 6d). If the setting value exceeds the range, please set the value back to the default and modify it. 5. Check if the acceleration / deceleration time is less than 10 ms. Please modify the command or enable the function of moving filter.
How to clear?	Alarm reset

E?002 Overvoltage	
Causes	1. The input voltage of the main circuit is higher than the rated allowable voltage. 2. The hardware of MS controller is damaged.
Checking methods and corrective actions	1. Check if the input voltage of the main circuit is within the rated allowable value. If not, please serial connect the regulator to convert the voltage into the rated range. 2. Check if the input voltage of the main circuit is within the rated range. If issue persists, please send your MS controller back to the distributors or contact Delta.
How to clear?	Alarm reset

E?003 Under voltage	
Causes	1. The input voltage of the main circuit is lower than the rated allowable voltage. 2. No power supply for the main circuit. 3. Wrong power input.
Checking methods and corrective actions	1. Use the voltmeter to see if the voltage of the main circuit is normal. 2. Refer to the user manual for wiring steps and correct the wiring between motor and MS controller. (Please refer to Chapter 3 for detailed information.) 3. Check the power system and see if it confirms to the specifications. Please serial connect the regulator to convert the voltage into the rated range.
How to clear?	Alarm will be cleared when the voltage returns to normal range.

E?004 Motor combination error	
Causes	1. The encoder is loose. 2. Motor combination error.
Checking methods and corrective actions	1. Check the encoder connector and reinstall it if it is loose. 2. Please refer to Chapter 1 and change a correct motor.
How to clear?	Re-power on the MS controller.

E?005 Regeneration error	
Causes	1. Select a wrong regeneration resistor or the external regeneration resistor is not connected. 2. Set P1-53 to zero when the regeneration resistor is not applied. 3. Parameter setting error.
Checking methods and corrective actions	1. Calculate the value for regenerative resistor again and reset the value of P1-52 and P1-53. If the alarm has not been cleared, please send MS controller back to Delta. 2. Make sure servo parameter P1-53 is set to zero when the regeneration resistor is not applied. 3. Correctly reset the value of P1-52 and P1-53 again.
How to clear?	Alarm reset

E?006 Overload	
Causes	1. The load is over the rated range and the servo drive is in a persistent overload condition. 2. The setting of the control system parameter is inappropriate. 3. Wiring between motor and encoder is wrong. 4. The encoder is defective.
Checking methods and corrective actions	1. Use monitoring variable (11d) to see if the average torque [%] is over 100%. Please refer to Appendix A, Overload features for further information. And reduce the load to clear the alarm. 2.1 Check if there is any mechanical vibration. If yes, please adjust the control loop gain. 2.2 Check if the acceleration/deceleration constant is set too fast. Lengthen the acceleration/deceleration setting time. 3. Check the wiring of U, V, W and the encoder. And rewire it. 4. Send the motor back to distributors or contact Delta.
How to clear?	Alarm reset

E?007 Overspeed	
Causes	Inappropriate setting of parameter P2-34.
Checking methods and corrective actions	Check if the setting value of P2-34 is too small (Condition for over speed warning). Check if the setting value of P2-34 is too small.
How to clear?	Alarm reset

E?009 Excessive deviation of position command	
Causes	1. Inappropriate setting of parameter P2-35 (Excessive position deviation warning). 2. Gain value is set too small. 3. Torque limit is set too low. 4. Excessive external load. 5. Improper setting of E-gear ratio.
Checking methods and corrective actions	1. Make sure the setting of parameter P2-35 is correct. If it is set too small, please increase the setting value. 2. Correctly adjust the gain value through the software in accordance with the mechanism. 3. Check the torque limit. Correctly adjust the gain value in accordance with the mechanism. 4. Reduce the external load or evaluate the motor capacity again. 5. Please input the ratio range: 1/50 < P1-44 / P1-45 < 25600. And make sure the proportion of P1-44 and P1-45 is appropriate. If the value exceeds the

E?009 Excessive deviation of position command	
	range, please modify the value.
How to clear?	Alarm reset

E?011 Encoder error	
Causes	1. Encoder wiring is wrong. 2. Encoder connection is loose. 3. Encoder is damaged.
Checking methods and corrective actions	1. Check if the wiring follows the instructions mentioned in the user manual. If not, conduct the wiring again. 2. Check if the connection between MS controller and encoder is loose and connect it again. 3. Please refer to Chapter 1 for replacing another motor.
How to clear?	Re-power on the MS controller.

E?012 Adjustment error	
Causes	Abnormal current adjustment.
Checking methods and corrective actions	Reset the power supply. If issue persists, send your MS controller back to distributors or contact Delta.
How to clear?	Re-power on the MS controller.

E?013 Emergency stop	
Causes	The Emergency stop button is pressed.
Checking methods and corrective actions	Check if the emergency stop button is enabled. This button is a normally closed button. Please release the button to clear the alarm.
How to clear?	Alarm reset

E?014 Reverse limit error	
Causes	1. Reverse limit switch is activated. 2. Servo system is instable.
Checking methods and corrective actions	1. Check if the reverse limit switch is activated. If yes, please release the switch. 2. Please adjust the parameter value or evaluate the motor capacity again. Check the watt of the applied motor and servo drive: Servo parameter H1-00 and H1-01: 750W: (H1-00 = 1d, H1-01 = 4d); 1500W: (H1-00 = 1d, H1-01 = 6d)
How to clear?	Alarm reset

E?015 Forward limit error	
Causes	1. Forward limit switch is activated. 2. Servo system is instable.
Checking methods and corrective actions	1. Check if the forward limit switch is activated. If yes, please release the switch. 2. Please adjust the parameter value or evaluate the motor capacity again. Check the watt of the applied motor and servo drive: 750W: (H1-00 = 1d, H1-01 = 4d); 1500W: (H1-00 = 1d, H1-01 = 6d)
How to clear?	Alarm reset

E?016 IGBT overheat	
Causes	1. The MS controller is in a persistent overload condition. 2. MS controller output is short-circuited.
Checking methods and corrective actions	1. Check if the motor is overloaded or over current. Then, increase the motor's capacity or reduce the load. 2. Check the output wiring of MS controller and make sure the wiring is correct.
How to clear?	Alarm reset

E?017 Abnormal EEPROM	
Causes	1. Parameter write-in error. This error occurs when parameters are restored to the default and servo drive type is wrong. 2. Data in ROM is damaged or there is no data in ROM. It occurs when the system is in servo-on status. Send your MS controller back to distributors or contact Delta.
Checking methods and corrective actions	See the monitoring variable (0x1D), which format is XGAB (x = 1 ~ 4; G = parameter group code; AB = hexadecimal code of parameter) 320Ah represents servo parameter P2-10; 3610h represents servo parameter P6-16. Make sure the parameter value is within the setting range. If an error occurs when applying to the power, it means one of the parameters exceeds the setting range. Please correct it and re-power on your MS controller. If the error occurs during normal operation, it means an error occurs when writing the parameter. Use alarm reset to clear the alarm. When it displays 100Xh, it means you select the wrong motor type. Please correctly set H1-00 and H1-01 in accordance with the applied motor and servo drive: 750W: (H1-00 = 1d, H1-01 = 4d); 1500W: (H1-00 = 1d, H1-01 = 6d)
How to clear?	Alarm reset

E?018 Abnormal signal output	
Causes	1. Encoder error. 2. The output pulse exceeds the hardware allowable range.
Checking methods and corrective actions	Check the log (P4-00 ~ P4-05) and see if the alarm exists with encoder errors (E?011, E?024, E?025, E?026). Then, carry out the corrective actions when any error occurs. Please correctly set parameter P1-76 and P1-46 with the following conditions: P1-76 > motor speed and $\frac{\text{Motor speed}}{60} \times \mathrm{P1}-46 \ \times 4 \ < 19.8 \ \times 10^6$
How to clear?	Alarm reset

E?019 Serial communication error	
Causes	1. Improper setting of communication parameters. 2. Incorrect communication address. 3. Incorrect communication value.
Checking methods and corrective actions	1. Check the setting value of communication parameter. Then, correctly set P3-03 and P3-04 or restore the value to the default. 2. Check and correctly set the communication address. 3. Check the accessing value and make sure the value is correct.
How to clear?	Alarm reset

E?020 Serial communication timeout	
Causes	1. Improper setting of the timeout parameters. 2. The servo drive has not received the communication command for a long time.
Checking methods and corrective actions	1. Check parameters setting and correctly set the value. 2. Check if the communication cable is loose or broken and correctly wire it.
How to clear?	Alarm reset

E?022 RST leak phase	
Causes	RST leak phase
Checking methods and corrective actions	Check if RST power cable is loose or no power is applied. This alarm occurs when 1.5 kW (or below) MS controller is not connected to three-phase power supply; for 2 kW (or above) MS controller, the alarm occurs when one single phase is not connected to the power supply. Correctly connect the MS controller to the power. If the issue persists, please send your MS controller to local distributors or contact Delta.
How to clear?	Alarm reset

E?023 Early warning for overload	
Causes	Early warning for overload
Checking methods and corrective actions	1. Check if your MS controller is overloaded and refer to the corrective actions of E?006 for troubleshooting. 2. Check if the value of P1-56 is set too small. If yes, please increase the value, which should be over 100 for disabling the warning function.
How to clear?	Alarm reset

E?024 Encoder initial magnetic field error	
Causes	Encoder initial magnetic field error (The magnetic field of the encoder U, V, W signal is in error.)
Checking methods and corrective actions	1. Make sure the servo motor is properly grounded and connect UVW connector (color green) to the heat sink of MS controller. 2. Make sure the encoder cable is separated from the power supply or high-current cable to avoid interference. 3. Please use the shielding cables for the encoder. If issue persists, please send your MS controller back to the distributors or contact Delta.
How to clear?	Re-power on the MS controller.

E?025 Internal error of the encoder	
Causes	1. Internal error of the encoder (Internal memory and counter are in error). 2. When applying to the power, the motor rotates because of mechanical inertia or other causes.
Checking methods and corrective actions	1. Make sure the servo motor is properly grounded and connect UVW connector (color green) to the heat sink of MS controller. 2. Make sure the encoder cable is separated from the power supply or high-current cable to avoid interference. 3. Please use the shielding cables for the encoder. 4. Make sure the motor shaft stands still when power is on.
How to clear?	Re-power on the MS controller.

E?026 Unreliable internal data of the encoder	
Causes	Encoder error (Internal data error occurs for three times continuously.)
Checking methods and corrective actions	1. Make sure the servo motor is properly grounded and connect UVW connector (color green) to the heat sink of MS controller. 2. Make sure the encoder cable is separated from the power supply or high-current cable to avoid interference. 3. Please use the shielding cables for the encoder.
How to clear?	Re-power on the MS controller.

E?027 The internal of the motor is in error	
Causes	Encoder reset error
Checking methods and corrective actions	1. Make sure the encoder communication cable is properly connected and applied shielding mesh. 2. Make sure the power supply is stable with 24 V input power.
How to clear?	Re-power on the MS controller.

E?028 Encoder voltage error or the internal of the encoder is in error	
Causes	1. Voltage level of the battery is too high. 2. Internal error of the encoder.
Checking methods and corrective actions	1.1 Check if there is a charging circuit in MS controller or the battery installation is wrong (voltage > 3.8 V). Please use voltmeter to measure the battery voltage and see if it exceeds 3.8 V. 2.1 Make sure the encoder is absolute type. 2.2 Check if the motor is properly grounded. Connect UVW connector (color green) to the heat sink of MS controller. 2.3 Check if the high-current cable interferes with the encoder cable. Please separate the encoder cable from the high-current cable.

E?028 Encoder voltage error or the internal of the encoder is in error	
	2.4 Check the wiring of encoder cable and apply shielding mesh.
How to clear?	Re-power on the MS controller.

E?029 Gray code error	
Causes	Absolute position error
Checking methods and corrective actions	Re-power on to operate the motor and check if the alarm occurs again. If issue persists, please change the encoder.
How to clear?	Re-power on the MS controller.

E?030 Motor crash error	
Causes	1. Check if the function of motor crash protection (P1-57) is enabled. If yes, please set P1-57 to 0. 2. See if the value of P1-57 is set too small and the time set by P1-58 is too short. Please set P1-57 according to the actual toque. Improper setting might inadvertently trigger the signal or lose the protection function.
Checking methods and corrective actions	Re-power on to operate the motor and check if the alarm occurs again. If issue persists, please change the encoder.
How to clear?	Re-power on the MS controller.

E?031 Incorrect wiring of motor power cable	
Causes	Incorrect wiring of motor power cable
Checking methods and corrective actions	Check if the motor power cable (U, V, W, GND) is firmly connected. Please conduct wiring and ground properly by following the instructions mentioned in user manual.
How to clear?	Re-power on the MS controller.

E?034 Internal communication of the encoder is in error	
Causes	Internal communication of the encoder is in error
Checking methods and corrective actions	1. Check the battery wiring. Then, wire it again and re-power on the system. 2. The internal communication error of absolute type encoder occurs. Please replace it with another motor.
How to clear?	Re-power on the MS controller.

E?044 Warning of MS controller function overload	
Causes	Warning of MS controller function overload
Checking methods and corrective actions	Set servo parameter P2-66 bit 4 to 1 to clear the alarm.
How to clear?	Re-power on the MS controller.

E?060 The absolute position is lost	
Causes	1. Voltage level of the battery is too low. 2. The battery is replaced when the control power is off. 3. The absolute coordinate has not been initialized after the absolute function is enabled. 4. Poor connection of the battery power circuit. 5. E-gear ratio setting is changed.
Checking methods and corrective actions	1. Check if the battery voltage is less than 2.8 V. 2. Do not change or remove the battery when the control power is off. 3. Complete the initialization of absolute coordinate: Method 1: Complete the setting via servo parameters: P2-08 = 271d and P2-71 = 1d. Method 2: Use homing function in DARS to rebuild the absolute coordinate system. (Please refer to Chapter 10 for detailed information.) 4.1 Check if the battery installation and wiring are both correct. 4.2 Check the encoder wiring. 4.3 Check the wiring between battery box and MS controller. 5. See if E-gear ratio is changed. Corrective action: Execute homing again.
How to clear?	Re-power on the MS controller.

E?061 Encoder undervoltage	
Causes	Voltage level of the battery is too low
Checking methods and corrective actions	1. Check from the panel and see if the battery voltage is less than 3.1 V (tentative specification). 2. Measure the battery voltage and see if it is less than 3.1 V (tentative specification). Replace the battery when the control power is on.
How to clear?	The alarm will be cleared automatically.

E?062 The multi-turn of absolute encoder overflows	
Causes	Motor's rotation cycle exceeds the range
Checking methods and corrective actions	Check if the motor's operation turn is within the range between -32768 and +32767. Please, execute homing again.
How to clear?	Re-power on the MS controller.

E?067 Encoder temperature warning	
Causes	Encoder temperature warning (85 ~ 100˚C)
Checking methods and corrective actions	Use DARS to set servo parameter P0-02 to 120d and check if the encoder temperature is identical to the motor. If the encoder temperature is higher, please improve heat dissipation or reduce operation load. If the encoder temperature is 30°C higher, please send the motor back to distributors.
How to clear?	Re-power on the MS controller.

E?069 Wrong motor type	
Causes	Incremental type motor does not support absolute function.
Checking methods and corrective actions	1. Check to see if your servo motor is with incremental type or absolute type encoder. 2. Check servo parameter P2-69 and correctly set the value.
How to clear?	Re-power on the MS controller.

E?06A The absolute position is lost	
Causes	1. The absolute coordinate has not been initialized after the absolute function is enabled. 2. E-gear ratio setting is changed.
Checking methods and corrective actions	1. Initialize the absolute coordinate: Method 1: Complete the setting via servo parameters: P2-08 = 271d and P2-71 = 1d. Method 2: Use homing function in DARS to rebuild the absolute coordinate system. 2. Check if E-gear ratio is changed. If yes, please execute homing.
How to clear?	Re-power on the MS controller.

E?06C MS controller connects to CA type and CW type motor simultaneously	
Causes	MS controller connects to CA type and CW type motor simultaneously.
Checking methods and corrective actions	Check if MS controller connects to CA type and CW type motor simultaneously
How to clear?	Connect the MS controller to the same type of motor and re-power on.

E?06D Check procedure is in error when power is supplied to the encoder	
Causes	This is a protective alarm. When the check procedure error occurs on one axis, other axes will stop operating. Motor that shows this alarm is normal.
Checking methods and corrective actions	Please check the motor that does not show this alarm and conduct corrective actions for its error.
How to clear?	Carry out corrective actions for those motors **that did not show this alarm**. Then, re-power on the MS controller.

E?070 Encoder does not complete the command which is issued by servo drive	
Causes	Command is not completed when writing the barcode into the encoder.
Checking methods and corrective actions	Check if the wiring is correct or firmly connected. If not, please correctly conduct the wiring again.
How to clear?	Re-power on the MS controller.

E?099 EEPROM must be updated	
Causes	EEPROM must be updated
Checking methods and corrective actions	Make sure EEPROM is upgraded. 1. Set servo parameter P2-08 to 30 first, then 28. 2. EEPROM is updated completely when the value of P2-08 is 999. During the updating process, MS controller should connect to the power all the time.
How to clear?	Re-power on the MS controller.

E?09A DSP AD1 error	
Causes	DSP AD1 error
Checking methods and corrective actions	Check the value of P0-14 and see if the firmware version is correct. (The value of P0-14 should be 0xFFFFAxxx)
How to clear?	Burn the firmware again.

E?09B DSP AD2 error	
Causes	DSP AD2 error
Checking methods and corrective actions	Check the value of P0-15 and see if the firmware version is correct. (The value of P0-15 should be 0xFFFFBxxx)
How to clear?	Burn the firmware again.

11

E?111 Buffer overflow occurs when receiving data via DMCNET	
Causes	MS controller receives more than two packets in 1 ms.
Checking methods and corrective actions	Make sure the host controller only receives or sends one packet in 1 ms.
How to clear?	Re-power on the MS controller.

E?185 DMCNET Bus hardware error	
Causes	DMCNET Bus hardware is in error or the communication packet is lost.
Checking methods and corrective actions	1. Make sure the communication cable is firmly connected. 2. Check the communication quality. (It is suggested to use common grounding and shielding cables.) If the communication quality is poor, try to change the communication cable. 3. Make sure the terminal resistor is installed.
How to clear?	Re-power on the MS controller.

E?201 An error occurs when loading DMCNET data	
Causes	An error occurs when loading DMCNET data
Checking methods and corrective actions	1. If the alarm is cleared when re-power on, it means the error occurs when accessing the data in previous time. Please re-power on the system and read DMCNET data again. 2. If the error persists after re-power on, it means the data in EEPROM is damaged. Please restore the system to the default. Set servo parameter P2-08 to 30 first, then 28.
How to clear?	Re-power on the MS controller.

E?235 Position overflows	
Causes	Incremental system: The motor keeps operating in one direction in PR mode. This causes position feedback register (FB_PUU) overflows. And the coordinate system cannot display the correct position. Issuing the absolute positioning command at this moment will result in error. Absolute system: This error will occur when issuing the absolute positioning command in following situations: 1. Feedback position register (FB_PUU) overflows. 2. Setting of P1.01.Z is changed but homing has not been completed yet. 3. E-gear ratio (servo parameter P1-44 and P1-45) is changed but homing has not been completed yet. 4. Function of returning to original point is triggered but homing has not been

E?235 Position overflows	
	completed yet. 5. E?060 or E?062 occurs.
Checking methods and corrective actions	Conduct homing.
How to clear?	Alarm will be cleared after re-power on

E?245 Positioning is overtime	
Causes	PR positioning is overtime.
Checking methods and corrective actions	If this alarm occurs, please directly send your MS controller back to Delta without making any modification.
How to clear?	N/A

E?283 Software positive limit	
Causes	The position exceeds the software positive limit
Checking methods and corrective actions	Check if the position exceeds the range set by servo parameter P5-08. Please set the limit in accordance with the actual application. If it does not exceed the setting range, please set it to the max. value (2147483647).
How to clear?	Alarm reset

E?285 Software negative limit	
Causes	The position exceeds the software negative limit
Checking methods and corrective actions	Check if the position exceeds the range set by servo parameter P5-09. Please set the limit in accordance with the actual application. If it does not exceed the setting range, please set it to the max. value (-2147483648).
How to clear?	Alarm reset

Note: The software positive / negative limit is determined by the position command. It is because the command always arrives first and then the feedback. When the limit protection is activated, the actual position might not exceed the limit yet. Thus, setting an appropriate decelerating time could satisfy the demand. Please refer to the description of P5-03.

E?289 Feedback position counter overflows	
Causes	Feedback position counter overflows
Checking methods and corrective actions	If this alarm occurs, please directly send your MS controller back to Delta without making any modification.
How to clear?	N/A

11

E?301 DMCNET synchronization failure	
Causes	DMCNET synchronization failure
Checking methods and corrective actions	1. Check if the communication quality is poor. If yes, please use shielded cable. 2. Check if the host controller successfully sends SYNC signal and make sure the sequence is synchronized. 3. Check if the setting of P3-09 is appropriate. It is better to use the default value.
How to clear?	Alarm reset

E?302 The synchronized signal of DMCNET is sent too fast	
Causes	The synchronized signal is sent too fast
Checking methods and corrective actions	1. Check if the setting of P3-09 is appropriate. It is better to use the default value. 2. Check if the host controller successfully sends SYNC signal and make sure the sequence is synchronized.
How to clear?	Alarm reset

E?303 The synchronized signal of DMCNET is sent too slow	
Causes	The synchronized signal is sent too slow.
Checking methods and corrective actions	1. Make sure the communication quality is good and use shielded cable. 2. Check if the host controller successfully sends SYNC signal and make sure the sequence is synchronized. 3. Check if the setting of P3-09 is appropriate. It is better to use the default value.
How to clear?	Alarm reset

E?304 DMCNET IP command failed	
Causes	The computing time of IP mode takes too long.
Checking methods and corrective actions	Please disable the monitoring function.
How to clear?	Alarm reset

E?500 STO function is enabled	
Causes	Safety function (STO) is enabled.
Checking methods and corrective actions	Safety function (STO) is enabled. Please check the activation causes.
How to clear?	Alarm reset

E?501 STO_A lost (Signal loss or signal error)	
Causes	STO_A loses the enable signal or STO_A signal does not synchronize with STO_B signal for more than 1 second.
Checking methods and corrective actions	Make sure the wiring of STO_A is correct.
How to clear?	Alarm reset

E?502 STO_B lost (Signal loss or signal error)	
Causes	STO_B loses the enable signal or STO_A signal does not synchronize with STO_B signal for more than 1 second.
Checking methods and corrective actions	Make sure the wiring of STO_B is correct.
How to clear?	Alarm reset

E?503 STO_error	
Causes	STO self-diagnostic error
Checking methods and corrective actions	Make sure the wiring of STO_A and STO_B is correct.
How to clear?	Alarm reset

E?555 System failure	
Causes	DSP processing error
Checking methods and corrective actions	If this alarm occurs, please directly send your MS controller back to Delta without making any modification.
How to clear?	N/A

Controller:

EC001 PLC timeout	
Causes	1. Due to the large PLC program, it takes too much execution time. 2. Switch to Debug mode.
Checking methods and corrective actions	1. Make sure PLC Debug mode is disabled. 2. Switch to a longer cycle.
How to clear?	Alarm reset

EC002 PLC image download failure	
Causes	The library version of PLC image does not match the firmware version.
Checking methods and corrective actions	Check if the controller parameter P1-01 is set to 1. If yes, then the error is caused by firmware update failure or wrong updated version. Please update the PLC image, which can match the firmware.
How to clear?	Alarm reset

EC003 PLC Exception	
Causes	PLC execution error
Checking methods and corrective actions	Please refer to the following information for troubleshooting.

Error message	Exception Code
PlcExcNon	0
ExcOutOfMemory	1
ExcDivisionByZero	2
ExcIndexOutOfRange	3
ExcIllegalCast	4
ExcStackOverflow	5
ExcNullReference	6
ExcMissingMethod	7
ExcThreadCreation	8
ExcThreadAbort	9
ExcSynchronizationLockException	10
ExcBreakpointIllegal	11
ExcBreakpoint	12
ExcExecutionEngine	13
ExcExternal	16
PlcExcString	32
PlcExcWatchDogExceeded	33
PlcExcMaximumCpuLoadExceeded	34
PlcExcSystem	35

EC003 PLC Exception		
	PlcExcEnd	36
How to clear?	Alarm reset	

EC004 Motion module failure	
Causes	Motion module function error
Checking methods and corrective actions	If this alarm occurs, please directly send your MS controller back to Delta without making any modification.
How to clear?	N/A

EC005 Controller failure	
Causes	Controller function error
Checking methods and corrective actions	If this alarm occurs, please directly send your MS controller back to Delta without making any modification.
How to clear?	N/A

EC006 Write-in error (continuous 30 sec)	
Causes	Error occurs when continuously write-in the program for 30 seconds.
Checking methods and corrective actions	Check if the continuously write-in is caused by PLC, RL program or Modbus error. If issue persists, it is better to remove all external devices first. Then, resume the default setting of PLC and start debugging.
How to clear?	Alarm reset

EC007 DMCNET device setting does not match	
Causes	When starting up MS controller, the scan result of external devices does not match the setting of non-volatile parameters.
Checking methods and corrective actions	Check the connection status of the device with DMCNET communication. And make sure the setting of controller parameter P3-31 ~ P3-42 matches the external devices.
How to clear?	Alarm reset. To change the setting, please re-scan the external devices and save the result to P3-31 ~ P3-42.

11

EC008 Robot parameter file loading error	
Causes	Robot parameter file format error or file damage results in loading error
Checking methods and corrective actions	Do not switch MS controller to servo on state. Disconnect the power for 10 minutes. If this issue persists, please directly send your MS controller back to Delta without making any modification.
How to clear?	Re-power on the MS controller.

EC009 Robot type is different from the setting	
Causes	Robot type set by parameter does not match the content loaded from the motion module.
Checking methods and corrective actions	1. Check the firmware version. Then, see if the setting of P1-00 and P0-03 is not identical resulting from firmware update failure or wrong firmware version. Update the firmware again if necessary. 2. Set the controller parameter P1-00 again. And make sure the applied robot type is supported by the firmware version. Then, re-power on your MS controller.
How to clear?	Re-power on the MS controller.

EC010 Frequent alarm reset	
Causes	Alarm is reset for over 5 times in 1 second.
Checking methods and corrective actions	1. Check if the program resets the alarm too frequently, such as continuously setting controller parameter P0-01 to 0. 2. To prevent the alarm from being auto-cleared by the system, users can add a reminder function in the application framework. So, user is able to clear it manually when alarm occurs.
How to clear?	Re-power on the MS controller.

NC Code

12

This chapter introduces NC code functions of ASDA-MS. NC code is mainly applied to the application for controlling the motion path.

12

12.1 Specification of NC code

Following table lists the start character of NC code and its function supported by ASDA-MS.

Start character	Description
A	Axis A (rotation axis in X-axis direction)
B	Axis B (rotation axis in Y-axis direction)
C	Axis C (rotation axis in Z-axis direction)
D	Angle
F	Feed rate
G	G code (general function)
I	(1) The offset from the starting point to the arc center in X-axis direction specified by command G02/G03; (2) Others
J	(1) The offset from the starting point to the arc center in Y-axis direction specified by command G02/G03; (2) Others
K	(1) The offset from the starting point to the arc center in Z-axis direction specified by command G02/G03; (2) Others
M	M code (auxiliary function)
N	Program line number
P	(1) Pause time of the fixed cycle; (2) Pause time of G04; (3) Parameter of G10
R	Radius
U	It is identical to axis A.
V	It is identical to axis B.
W	It is identical to axis C.
X	Axis X
Y	Axis Y
Z	Axis Z

The following lists the G codes supported by ASDA-MS:

G Code	Group	G code function
G00	01 Motion mode	Fast positioning
G01		Linear interpolation
G02		Arc / helical interpolation in clockwise direction
G03		Arc / helical interpolation in counterclockwise direction
G04	00	Dwell
G09		Exact stop
*G17	02 Plane selection	Select the X-Y plane
G18		Select the Z-X plane
G19		Select the Y-Z plane
*G50	11 Scaling setting	Cancel scaling up/down
G51		Scaling up/down
G52	00	Select the local coordinate system
G53		Select mechanical coordinate system (Absolute)
*G54	14 Coordinates system selection	1st workpiece coordinates system
G55		2nd workpiece coordinates system
G56		3rd workpiece coordinates system
G57		4th workpiece coordinates system
G58		5th workpiece coordinates system
G59		6th workpiece coordinates system
G68	16 Coordinate system rotation setting	Rotate the coordinate system
*G69		Cancel coordinate system rotation
G90	03 Absolute / Incremental setting	Absolute coordinate system
G91		Incremental coordinate system

G code with the * mark means it is the default when G code is initialized.

12

12.2 Detailed description of G Code

G code can be used to execute the motion with MS controller. The following paragraph will introduce the supported G codes and their functions.

- G00 Fast positioning

Format: G00 X_Y_Z_

Description:
1. Select an axis (axis X, Y, or Z).
2. If the current motion is in rapid traverse (G00) already, setting G00 again is not required as it is a continuous effective command.
3. The max. speed is 2 m/sec. (The max. speed can be set via parameters)

- G01 Linear interpolation

Format: G01 X_Y_Z_F_

Description:
1. Select an axis (axis X, Y, or Z).
2. If the current motion is linear interpolation (G01) already, setting G01 again is not required as it is a continuous effective command.
3. The F value specifies the feed rate.
4. If F value is not specified, the program will refer to the previous setting feed rate.

- G02/G03 Clockwise/Counterclockwise _ Arc / Helical interpolation_ Center mode

Format: G02/G03 X_Y_Z_I_J_K_F_

Description:
1. Specify the plane by G17, G18, and G19.
2. X, Y, and Z specify the target position.
3. If X-Y plane is selected, then you have to set X and Y values.
4. If X, Y, and Z specify the current tool position, the motion path will be a circle.
5. An error will occur when the deviation of “the distance from current position to the center” and “target position to the center” exceeds 0.0002 inch or 0.002 mm.
6. I, J and K define the relative coordinates from the current position to the center.
7. I, J and K are used for selection G17, G18 and G19.
8. Choose 1 axis from I, J and K. The unselected axis will be 0.
9. The F value specifies the feed rate.
10. If F value is not specified, the program will refer to the previous setting feed rate.
11. If the coordinates of the 3rd axis is identical to the current position, the motion path will be in arc; if different, it is helical.

- G02/G03 Clockwise/Counterclockwise _ Arc method / Helical interpolation _ radius method

Format: G02/G03 X_Y_Z_R(+/-)_F_

Description:
1. Specify the plane by G17, G18, and G19.
2. X, Y, and Z specify the target position.
3. Target position cannot be identical to the current position.
4. If X-Y plane is selected, then you have to set X and Y values.
5. R is the radius.
6. R must be greater than the sum of target position plus current position/2 ($R > \frac{Target\ position + Current\ position}{2}$)
7. R with positive sign (+): the arc angle is ≤ 180°;
   R with negative sign (-): the arc angle is in the range of 180° ~360°.
8. The F value specifies the feed rate.
9. If F value is not specified, the program will refer to the previous setting feed rate.
10. If the coordinates of the 3rd axis is identical to the current position, the motion path will be in arc; if different, it is helical.

- G04 Dwell time

Format: G04 X_
G04 P_

Description:
1. Unit of X is second, which can be a decimal.
2. Unit of P is msec, which can only be an integral.
3. It is a non-continuous effective G code.

- G09 Exact stop

Format: G09

Description: It is a non-continuous effective G code.

- G17/G18/G19 Plane selection

Format: G17

Description:
1. Select the X-Y plane.
2. It is a continuous effective G code.

Format: G18

Description:
1. Select the Z-X plane.
2. It is a continuous effective G code.

Format: G19

Description: 1. Select the Y-Z plane.

2. It is a continuous effective G code.

- G50 Cancel scaling up/down

Format: G50

Description: Cancel the scaling up/down of the coordinate system

- G51 Scaling up/down

Format: G51 X_Y_Z_P_

G51 X_Y_Z_I_J_K_

Description:
1. X, Y, and Z set the center position for scaling up/down, which is an absolute coordinates.
2. P sets the scaling ratio for all axes and has to be an integral. If X_Y_Z_P_ is set, all axes will scale up/down with the same scaling. When P1300 is specified, the scaling ratio will be 1.3.
3. I, J, and K set the scaling ratio for individual axis. If X_Y_Z_ I_J_K_ is set, each axis will scale up/down according to I, J, and K respectively. I, J, and K have to be integrals. When I1300 is set, the scaling ratio will be 1.3.

- G52 Local coordinate system setup

Format: G52 X_Y_Z_

Description:
1. X, Y, and Z specify the origin of the workpiece coordinate system.
2. Specify the workpiece coordinate system by G54 ~ G59 first.
3. The setting immediately becomes active once the workpiece coordinate system is set. To cancel the setting of local coordinate system, please set X, Y, and Z to 0.

- G53 Go to the specified mechanical coordinates in rapid traverse

Format: G53 X_Y_Z_F_

Description:
1. X, Y, and Z specify the target mechanical coordinates.
2. It is valid only when being applied with G90 Absolute coordinate system.
3. This command enables the tool to go to the specified position in rapid traverse with the feed rate you set or the feed rate set in G1 or G0.
4. It is a non-continuous effective G code.

- G54 ~ G59 Workpiece coordinate system setting

Format: G54 ~ G59

Description:
1. Specify the workpiece coordinate system.
2. It is a continuous effective G code, which remains active in the following blocks.
3. You can input the coordinate data in MDI mode or by G10.

- G68 Workpiece coordinate system rotation setting

Format: G68 X_Y_Z_R_

Description:
1. Specify the plane by G17, G18, and G19.
2. X, Y, and Z set the center of rotation.
3. R sets the degree of rotation. Set positive values for counterclockwise rotation and negative values for clockwise rotation.
4. If X, Y, and Z are not specified and G68 is being executed, the current position will be regraded as the center of rotation.
5. If R is not specified, it will refer to the parameter setting.

- G69 Cancel workpiece coordinate system rotation

Format: G69

Description: Cancel the rotation of workpiece coordinate system.

- G90 Coordinate system

Format: G90

Description: Specify the position with absolute coordinates.

- G91 Coordinate system

Format: G91

Description: Specify the position in incremental form.

12

12.3 How to use DRAS software?

The main programming language of ASDA-MS is Delta robot language (DRL). To execute the NC code for making the motion path, it requires machine language to call the NC code. See the operation flow chart below.

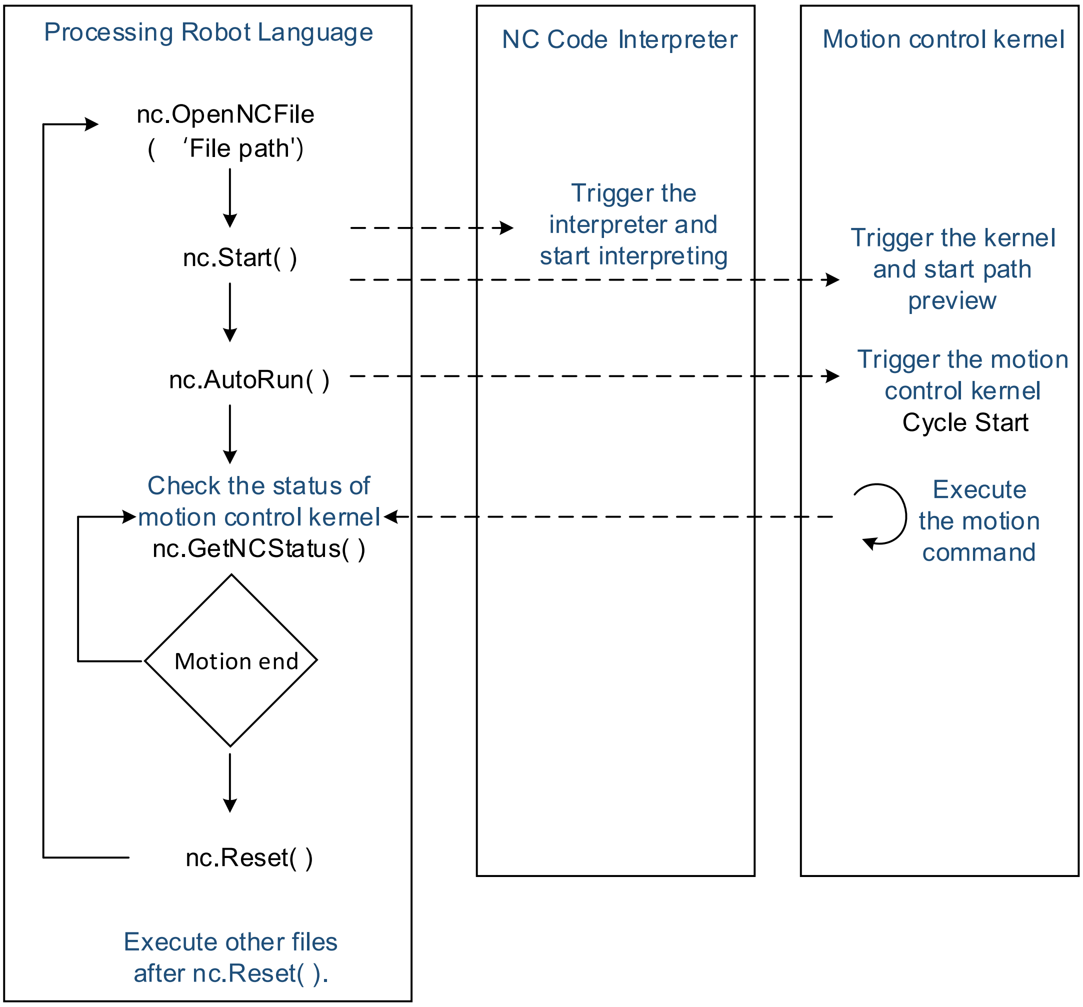


Figure 12.3.1 Flow chart of NC code execution

State diagram of nc.GetNCStatus( ) is as follows.

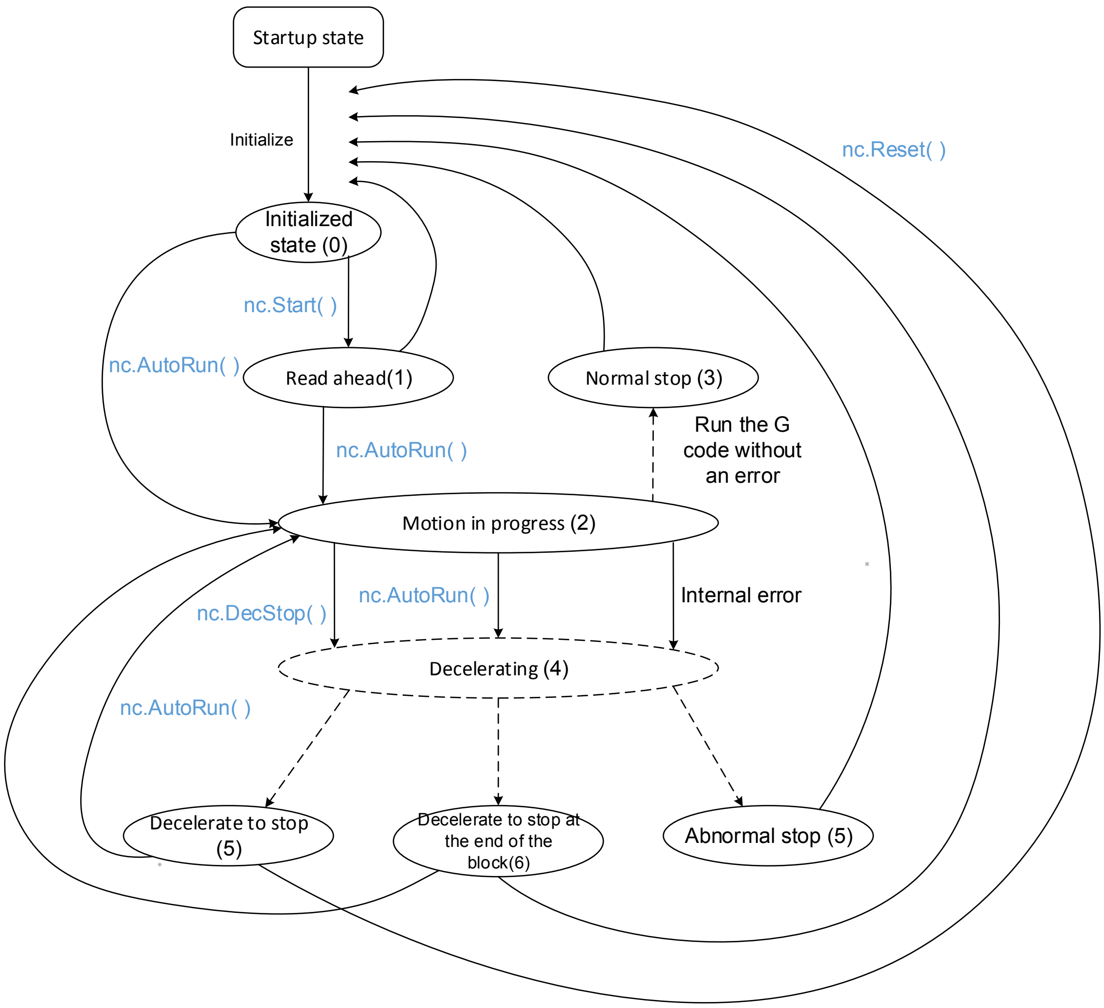
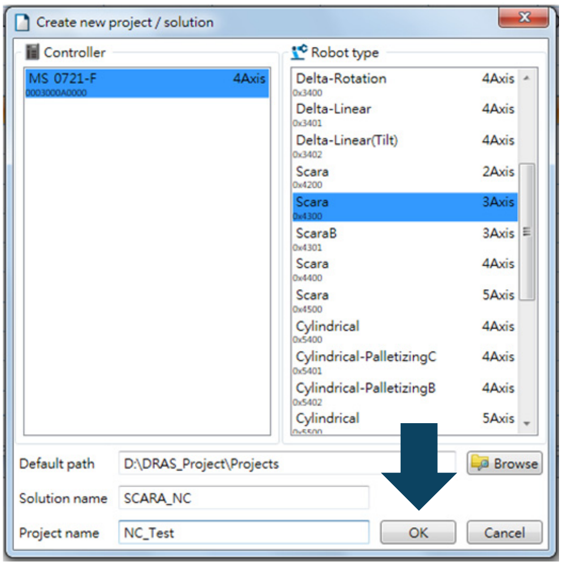


Figure 12.3.2 NC state diagram

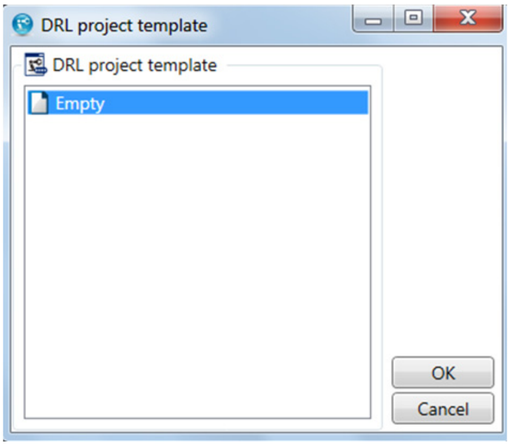
12

The example of executing NC program is as follows:

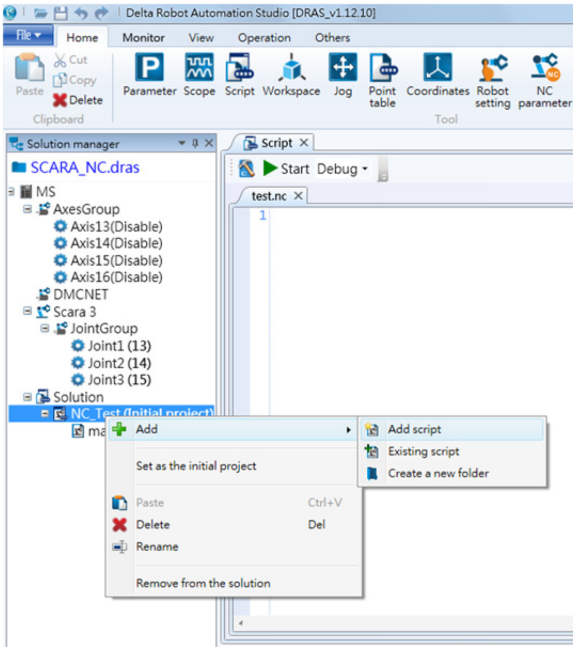
1. After opening the startup project, click on **OK**.



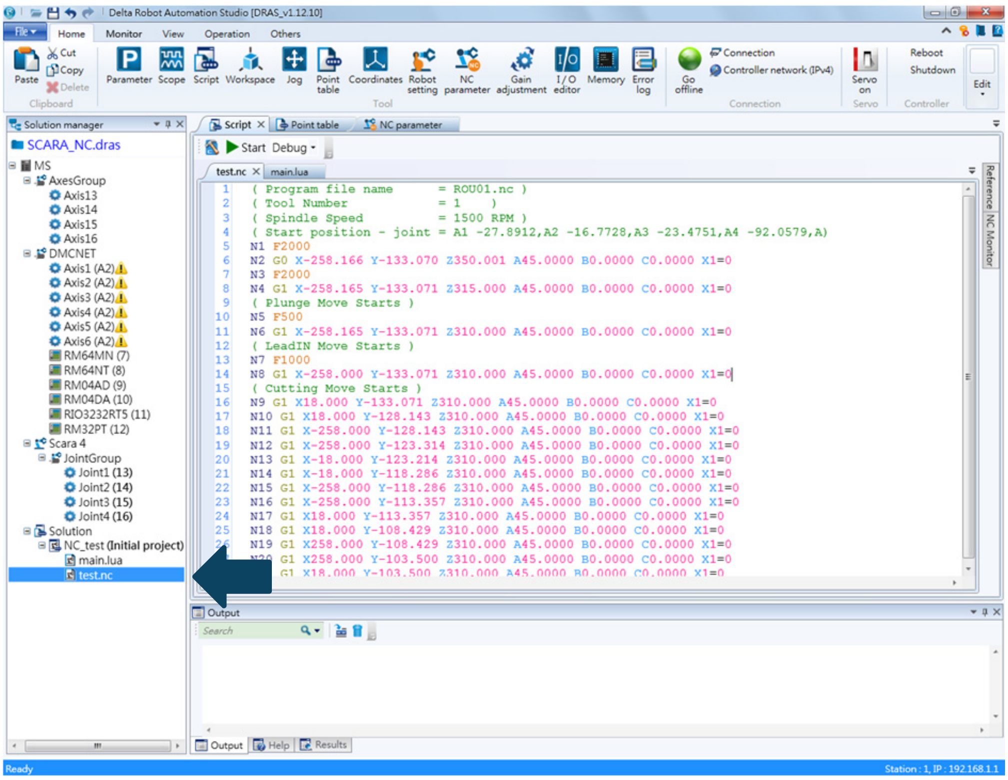
2. Click on **Empty** > **OK** in DRL project template.



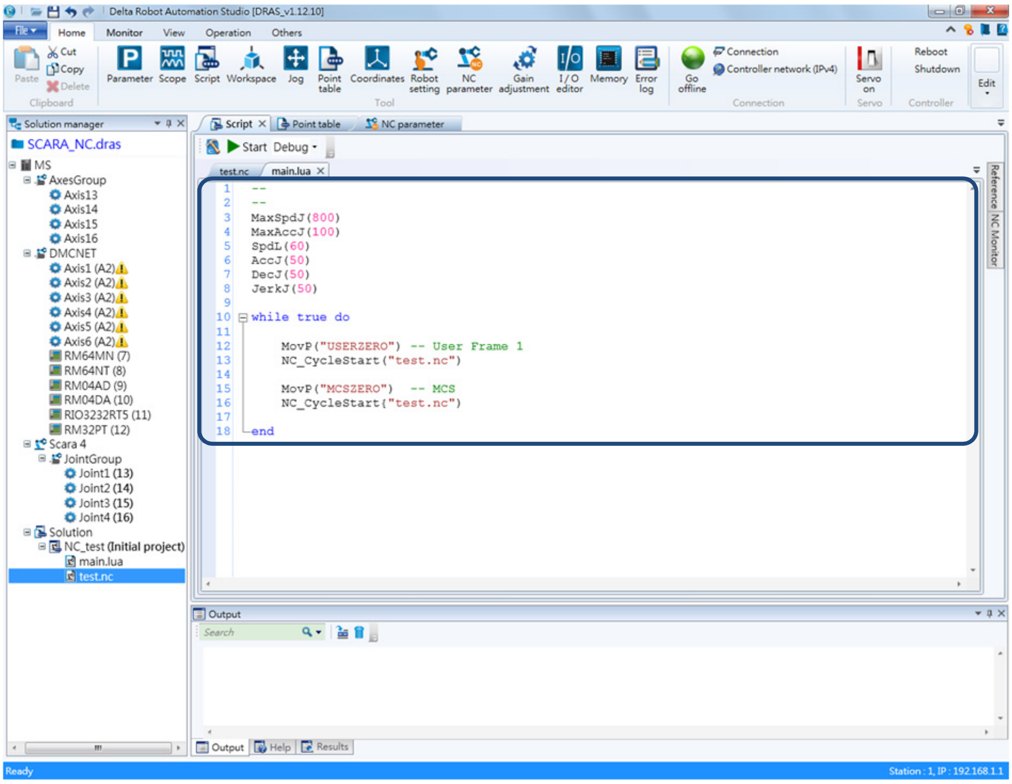
3. Right click on the project and click **Add** > **Add script**.



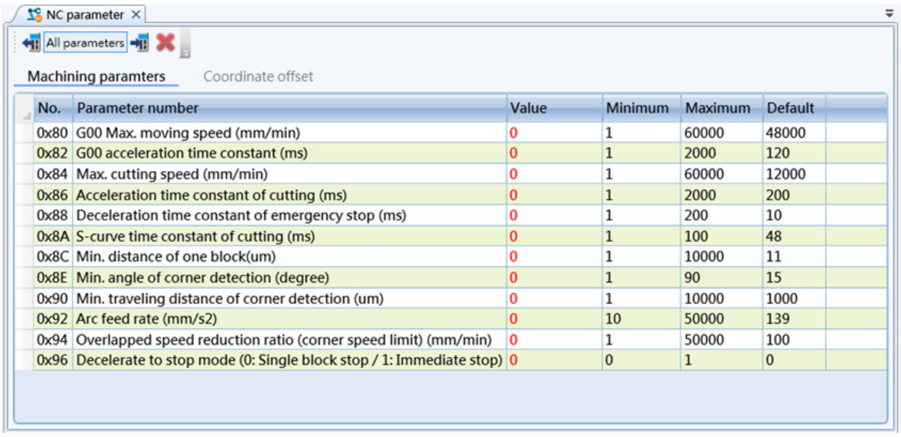
4. Rename the script1.lua file as a ".nc" file, such as "test.nc". Edit the NC code and then save it.



5. Go to the main.lua page to edit the robot language for triggering the NC program. The example is shown as follows:



12.4 NC parameters



No.	Parameter number	Value	Minimum	Maximum	Default
0x80	G00 Max. moving speed (mm/min)	0	1	60000	48000
0x82	G00 acceleration time constant (ms)	0	1	2000	120
0x84	Max. cutting speed (mm/min)	0	1	60000	12000
0x86	Acceleration time constant of cutting (ms)	0	1	2000	200
0x88	Deceleration time constant of emergency stop (ms)	0	1	200	10
0x8A	S-curve time constant of cutting (ms)	0	1	100	48
0x8C	Min. distance of one block(um)	0	1	10000	11
0x8E	Min. angle of corner detection (degree)	0	1	90	15
0x90	Min. traveling distance of corner detection (um)	0	1	10000	1000
0x92	Arc feed rate (mm/s2)	0	10	50000	139
0x94	Overlapped speed reduction ratio (corner speed limit) (mm/min)	0	1	50000	100
0x96	Decelerate to stop mode (0: Single block stop / 1: Immediate stop)	0	0	1	0

Parameter 0x80 ~ 0x8A are used for the speed relevant settings. Parameter 0x80 and 0x82 set the speed of G00 command. Parameter 0x84 ~ 0x86 are for setting the speed for cutting command G01. And 0x8A is for specifying the S-curve time constant. To modify the speed and activate the setting, please stop executing the motion path first. The setting is invalid when speed is modified during NC code is still running.

Parameter 0x8C ~ 0x94 are for analyzing the path. 0x8C is to see if the statement is true by referring to the minimum distance of one block; 0x8E~0x90 are for testing the statement by referring to the minimum angle/traveling distance of corner detection; 0x92 ~ 0x94 are for calculating the speed for connecting two blocks.

12

12.5 NC coordinate system

The diagram below shows the concept of the two coordinate systems applied in MS controller; the down-left is the robot arm coordinate system and top-right is the NC coordinate system. Users can use G92 command to specify the “NC mechanical origin” as the “origin of PCS” in NC system. That is, the PCS will be shifted to the origin set by G92. G54 ~ G59 can also be used for switching from G92 coordinate system to other systems. And G52 system is the local coordinate system converted from the system created by G54 ~ G59. Therefore, machining can be done in the NC coordinate system with G codes (such as G51 Scaling up/down command and G68 Coordinate system rotation).

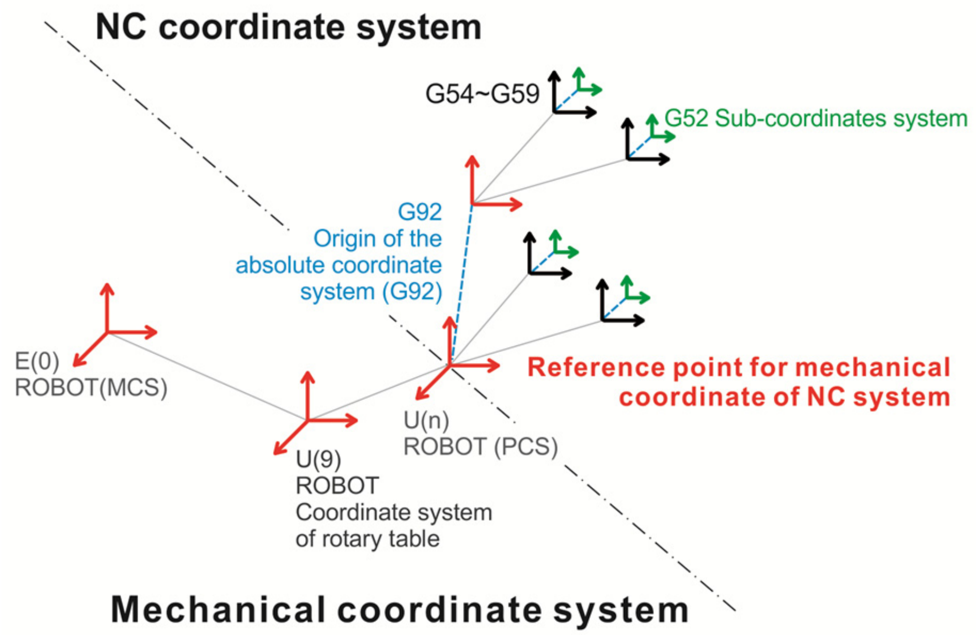
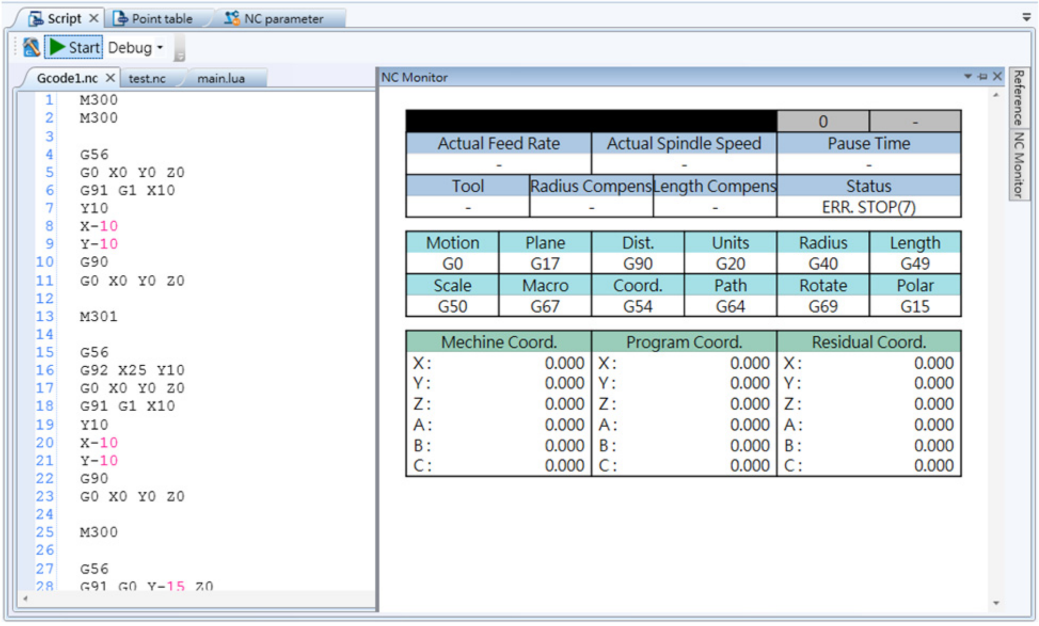


Figure 12.5.1 NC program coordinates (Absolute / Workpiece)

12.6 NC system monitoring

When using the NC functions of DRAS software, you can click on the **NC Monitor** at the right hand side of the window to check the actual feed rate, actual spindle speed, and dwell time. In addition, you can also see the coordinate values in the mechanical and program coordinate system.



(This page is intentionally left blank.)

Specifications Appendix A

A

Specifications of ASDA-MS controller

MS controller		750 W (4-axis)	1.5 kW (4-axis)
		07	15
Power	Phase / Voltage	Three-phase / Single-phase 220 VAC	Three-phase 220 VAC
	Permissible voltage	Single-phase / Three-phase 220 ~ 230 VAC, -15% ~ 10%	Three-phase 220 ~ 230 VAC, -15% ~ 10%
	Control power	24 VDC, -10% ~ 10%	
	Input current (3 PH) Unit: Arms	12.4	24.8
	Input current (1 PH) Unit: Arms	23.8	44.5
	Continuous output current Unit: Arms	5.1 (for each axis)	8.3 (for each axis)
Dimensions (W) x (H) x (D) mm / Weight		175 mm x 300 mm x 159 mm / 5.6 kg	
Cooling method		Fan cooling	
Encoder resolution / Feedback resolution		20-bit (1280000 p/rev)	
Main circuit control		SVPWM control	
Control mode		Manual / Auto	
Regenerative resistor		Built-in	
Robot control	Programming language	IEC61131-3 PLC 5 kinds of languages ((LD, FBD, SFC, IL, ST) and Delta robot language (DRL)	
	Motion mode	Point to point motion, linear interpolation and arc interpolation	
	Memory	20 MB: for program and data editing 16 KB: for PLC SV/DV variable (volatile) 60 KB: for PLC DH variable (non-volatile) 1 K: position data is for global variable (can be shared by different program) Max. 32 K: position data is for editing the program	
Input / Output	Standard I/O	Users I/O: 24 digital inputs and 12 digital outputs System I/O: 8 digital inputs and 8 digital outputs	
	Brake output	4 brake outputs	
Communication interface	Ethernet	1 channel	
	RS-232 / RS-485	1 connector (it can be switched between two communication modes)	
	DMCNET	1 channel	
	USB Host	1 connector	

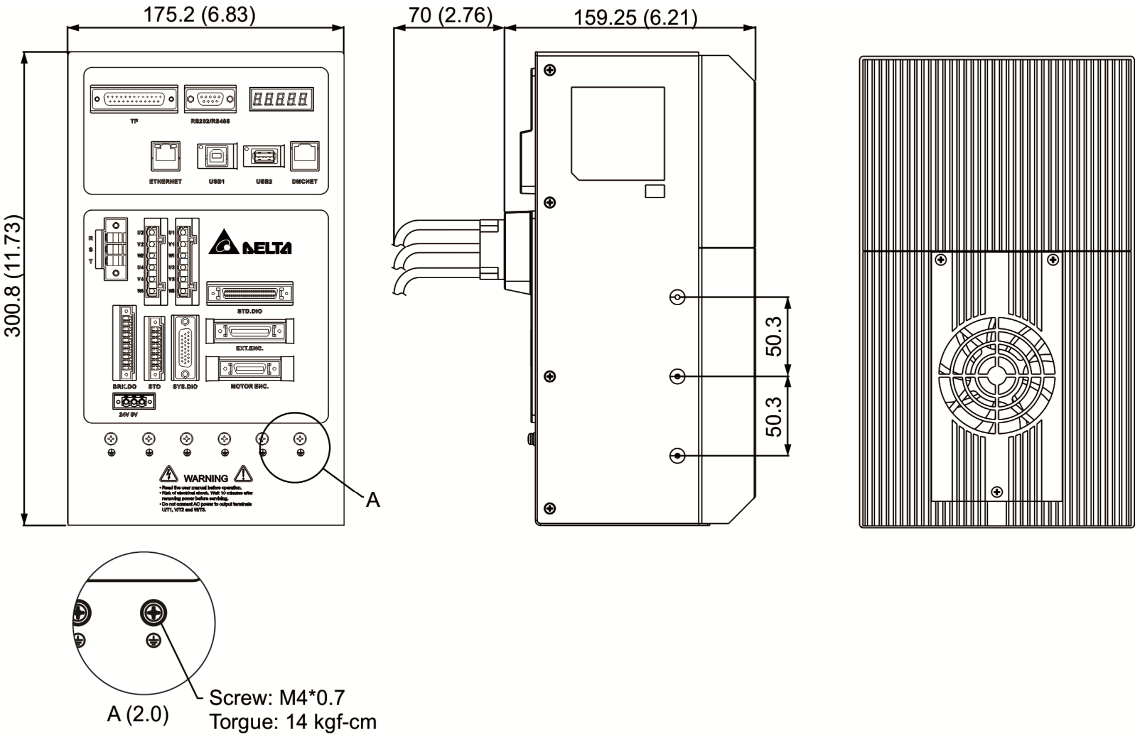
		750 W (4-axis)	1.5 kW (4-axis)
		07	15
Environment	Installation site	Indoors (avoid direct sunlight), no corrosive fog (avoid fume, flammable gas and dust)	
	Altitude	Below altitude 1000 m	
	Atmospheric pressure	86 kPa ~ 106 kPa	
	Operating Temperature	0˚C ~ 55˚C (If operating temperature is above 45°C, forced cooling will be required)	
	Storage Temperature	-20˚C ~ 65˚C	
	Humidity	Under 0 ~ 90% RH (non-condensing)	
	Vibrating	9.80665 m/s2 (1 G) less than 20 Hz, 5.88 m/ s2 (0.6 G) 20 to 50 Hz	
	IP rating	IP20	
	Power system	TN system[*1]	
	Approvals	IEC/EN 61800-5-1, UL 508C, C-tick	

Note:

*1. TN system: The neutral point of the power system connects to the ground directly. The exposed metal components connect to the ground via the protective earth conductor.

2. 1.5 kW model is coming soon.

Dimensions of ASDA-MS controller



Weight	4.5

Note:

1. Dimensions are in millimeters; Weights are in kilograms (kg)
2. Dimensions and weights of the servo drive may be updated without prior notice.

Specifications of servo motor (ECMA series)

Low inertia series

ECMA	C104	C△04	C△06		C△08		C△09		C△10
	0F	01	02	04□S	04	07	07	10	10
Rated power (kW)	0.05	0.1	0.2	0.4	0.4	0.75	0.75	1.0	1.0
Rated torque (N-m)[*1]	0.159	0.32	0.64	1.27	1.27	2.39	2.39	3.18	3.18
Max. torque (N-m)	0.477	0.96	1.92	3.82	3.82	7.16	7.14	8.78	9.54
Rated speed (r/min)	3000						3000		3000
Max. speed (r/min)	5000						3000		5000
Rated current (A)	0.69	0.90	1.55	2.60	2.60	5.10	3.66	4.25	7.30
Max. instantaneous current (A)	2.05	2.70	4.65	7.80	7.80	15.30	11.00	12.37	21.90
Power rating (kW/s)	12.27	27.7	22.4	57.6	24.0	50.4	29.6	38.6	38.1
Rotor inertia (× 10^{-4}kg.m^2)	0.0206	0.037	0.177	0.277	0.68	1.13	1.93	2.62	2.65
Mechanical constant (ms)	1.14	0.75	0.80	0.53	0.74	0.63	1.72	1.20	0.74
Torque constant-KT (N-m/A)	0.23	0.36	0.41	0.49	0.49	0.47	0.65	0.75	0.44
Voltage constant-KE (mV/(r/min))	9.8	13.6	16.0	17.4	18.5	17.2	24.2	27.5	16.8
Armature resistance (Ohm)	12.70	9.30	2.79	1.55	0.93	0.42	1.34	0.897	0.20
Armature inductance (mH)	26.0	24.0	12.07	6.71	7.39	3.53	7.55	5.70	1.81
Electric constant (ms)	2.05	2.58	4.30	4.30	7.96	8.36	5.66	6.23	9.30
Insulation class	Class A (UL), Class B (CE)								
Insulation resistance	> 100 MΩ, 500 VDC								
Insulation strength	1.8k Vac,1 sec								
Weight (w/o brake) (kg)	0.42	0.5	1.2	1.6	2.1	3.0	2.9	3.8	4.3
Weight (with brake) (kg)	--	0.8	1.5	2.0	2.9	3.8	3.69	5.5	437
Max. radial loading (N)	78.4	78.4	196	196	245	245	245	245	490
Max. axial loading (N)	39.2	39.2	68	68	98	98	98	98	98
Power rating (kW/s) (with brake)	--	25.6	21.3	53.8	22.1	48.4	29.3	37.9	30.4
Rotor inertia (× 10^{-4}kg.m^2) (with brake)	--	0.04	0.19	0.30	0.73	1.18	1.95	2.67	3.33
Mechanical constant (ms) (with brake)	--	0.81	0.85	0.57	0.78	0.65	1.74	1.22	0.93
Brake holding torque [Nt-m (min)] *2	--	0.3	1.3	1.3	2.5	2.5	2.5	2.5	8.0
Brake power consumption (at 20˚C)[W]	--	7.3	6.5	6.5	8.2	8.2	8.2	8.2	18.7
Brake release time [ms (Max)]	--	5	10	10	10	10	10	10	10
Brake pull-in time [ms (Max)]	--	25	70	70	70	70	70	70	70
Vibration grade (μm)	15								
Operating temperature (˚C)	0˚C ~ 40˚C								

Storage temperature (˚C)	-10˚C ~ 80˚C
Operating humidity	20 ~ 90%RH (non-condensing)
Storage humidity	20 ~ 90%RH (non-condensing)
Vibration capacity	2.5 G
IP rating	IP65 (when waterproof connectors are used, or when an oil seal is used to be fitted to the rotating shaft (an oil seal model is used))
Approvals	

Note:

*1. The rated torque is the continuous permissible torque between 0 ~ 40˚C operating temperature which is suitable for the following heat sink dimension.

ECMA-_ _ 04 / 06 / 08: 250 mm x 250 mm x 6 mm

ECMA-_ _ 10: 300 mm x 300 mm x 12 mm

ECMA-_ _ 13: 400 mm x 400 mm x 20 mm

Material: Aluminum – F40, F60, F80, F100, F130

*2. The built-in brake of the servo motor is for keeping it in stop state. Do not use it to decelerate or as the dynamic brake.

3. As for the information about motors with magnetic encoder, please refer to the corresponded model.

4. (△) in motor model names represents encoder type. Please refer to Chapter 1 for further information.

A

Medium/Medium-high inertia series

ECMA	C△06	C△08
	04□H	07□H
Rated power (kW)	0.40	0.75
Rated torque (N-m)[*1]	1.27	2.39
Max. torque (N-m)	3.82	7.16
Rated speed (r/min)	3000	
Max. speed (r/min)	5000	
Rated current (A)	2.6	5.1
Max. instantaneous current (A)	7.8	15.3
Power rating (kW/s)	21.70	19.63
Rotor inertia (× 10^{-4}kg.m^2)	0.743	2.910
Mechanical constant (ms)	1.42	1.60
Torque constant-KT (N-m/A)	0.49	0.47
Voltage constant-KE (mV/(r/min))	17.4	17.2
Armature resistance (Ohm)	1.55	0.42
Armature inductance (mH)	6.71	3.53
Electric constant (ms)	4.30	8.36
Insulation class	Class A (UL), Class B (CE)	
Insulation resistance	> 100 MΩ, 500 VDC	
Insulation strength	1.8k Vac,1 sec	
Weight (w/o brake) (kg)	1.8	3.4
Weight (with brake) (kg)	2.2	3.9
Max. radial loading (N)	196	245
Max. axial loading (N)	68	98
Power rating (kW/s) (with brake)	21.48	19.30
Rotor inertia (× 10^{-4}kg.m^2) (with brake)	0.751	2.960
Mechanical constant (ms) (with brake)	1.43	1.62
Brake holding torque [Nt-m (min)] *2	1.3	1.3
Brake power consumption (at 20˚C)[W]	6.5	8.2
Brake release time [ms (Max)]	10	10
Brake pull-in time [ms (Max)]	70	70
Vibration grade (μm)	15	
Operating temperature (˚C)	0˚C ~ 40˚C	
Storage temperature (˚C)	-10˚C ~ 80˚C	
Operating humidity	20 ~ 90%RH (non-condensing)	

A

ECMA	C△06	C△08
	04□H	07□H
Storage humidity	20 ~ 90%RH (non-condensing)	
Vibration capacity	2.5G	
IP rating	IP65 (when waterproof connectors are used, or when an oil seal is used to be fitted to the rotating shaft (an oil seal model is used))	
Approvals	CE cULus	

Note:

*1. The rated torque is the continuous permissible torque between 0 ~ 40˚C operating temperature which is suitable for the following heat sink dimension.

ECMA-_ _ 04 / 06 / 08: 250 mm x 250 mm x 6 mm

ECMA-_ _ 10: 300 mm x 300 mm x 12 mm

ECMA-_ _ 13: 400 mm x 400 mm x 20 mm

Material: Aluminum – F40, F60, F80, F100, F130

*2. The built-in brake of the servo motor is for keeping it in stop state. Do not use it to decelerate or as the dynamic brake.

3. As for the information about motors with magnetic encoder, please refer to the corresponded model.

4. (△) in motor model names represents encoder type. Please refer to Chapter 1 for further information.

A

High inertia series

ECMA	E△13			F△13	G△13		
	05	10	15	08	03	06	09
Rated power (kW)	0.5	1.0	1.5	0.85	0.3	0.6	0.9
Rated torque (N-m)[*1]	2.39	4.77	7.16	5.41	2.86	5.73	8.59
Max. torque (N-m)	7.16	14.3	21.48	13.8	8.59	17.19	21.48
Rated speed (r/min)	2000			1500	1000		
Max. speed (r/min)	3000			3000	2000		
Rated current (A)	2.9	5.6	8.3	7.1	2.5	4.8	7.5
Max. instantaneous current (A)	8.7	16.8	24.9	19.4	7.5	14.4	22.5
Power rating (kW/s)	7.0	27.1	45.9	21.52	10.0	39.0	66.0
Rotor inertia (× 10^{-4}kg.m^2)	8.17	8.41	11.18	13.6	8.17	8.41	11.18
Mechanical constant (ms)	1.91	1.51	1.10	2.43	1.84	1.40	1.06
Torque constant-KT (N-m/A)	0.83	0.85	0.87	0.76	1.15	1.19	1.15
Voltage constant-KE (mV/(r/min))	30.9	31.9	31.8	29.2	42.5	43.8	41.6
Armature resistance (Ohm)	0.57	0.47	0.26	0.38	1.06	0.82	0.43
Armature inductance (mH)	7.39	5.99	4.01	4.77	14.29	11.12	6.97
Electric constant (ms)	12.96	12.88	15.31	12.55	13.55	13.55	16.06
Insulation class	Class A (UL), Class B (CE)						
Insulation resistance	> 100 MΩ, 500 VDC						
Insulation strength	1500 VAC, 60 sec						
Weight (w/o brake) (kg)	6.8	7.0	7.5	8.6	6.8	7.0	7.5
Weight (with brake) (kg)	8.2	8.4	8.9	10.0	8.2	8.4	8.9
Max. radial loading (N)	490	490	490	490	490	490	490
Max. axial loading (N)	98	98	98	98	98	98	98
Power rating (kW/s) (with brake)	6.4	24.9	43.1	19.78	9.2	35.9	62.1
Rotor inertia (× 10^{-4}kg.m^2) (with brake)	8.94	9.14	11.90	14.8	8.94	9.14	11.9
Mechanical constant (ms) (with brake)	2.07	1.64	1.19	2.65	2.0	1.51	1.13
Brake holding torque [Nt-m (min)] *2	10.0	10.0	10.0	10.0	10.0	10.0	10.0
Brake power consumption (at 20˚C)[W]	19.0	19.0	19.0	19.0	19.0	19.0	19.0
Brake release time [ms (Max)]	10	10	10	10	10	10	10
Brake pull-in time [ms (Max)]	70	70	70	70	70	70	70
Vibration grade (μm)	15						
Operating temperature (˚C)	0˚C ~ 40˚C						
Storage temperature (˚C)	-10˚C ~ 80˚C						

ECMA	E△13			F△13	G△13		
	05	10	15	08	03	06	09
Operating humidity	20 ~ 90%RH (non-condensing)						
Storage humidity	20 ~ 90%RH (non-condensing)						
Vibration capacity	2.5G						
IP rating	IP65 (when waterproof connectors are used, or when an oil seal is used to be fitted to the rotating shaft (an oil seal model is used))						
Approvals	CE cULus						

Note:

*1. The rated torque is the continuous permissible torque between 0 ~ 40˚C operating temperature which is suitable for the following heat sink dimension.

ECMA-_ _ 04 / 06 / 08: 250 mm x 250 mm x 6 mm

ECMA-_ _ 10: 300 mm x 300 mm x 12 mm

ECMA-_ _ 13: 400 mm x 400 mm x 20 mm

Material: Aluminum – F40, F60, F80, F100, F130

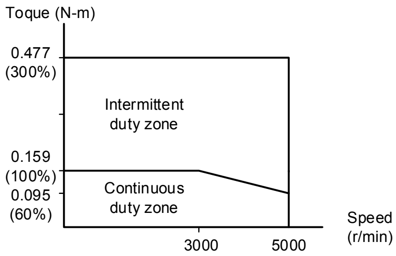
*2. The built-in brake of the servo motor is for remaining the item in stop status. Do not use it to decelerate or as the dynamic brake.

3. As for the information about motors with magnetic encoder, please refer to the corresponded model.

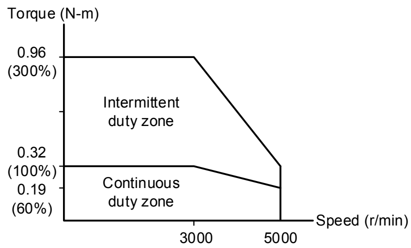
4. (△) in motor model names represents encoder type. Please refer to Chapter 1 for further information.

A

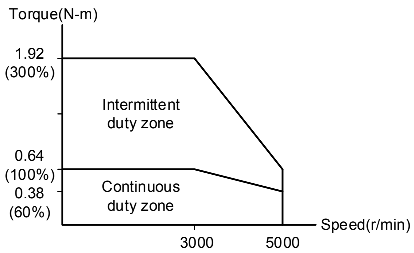
Torque features (T-N curves)



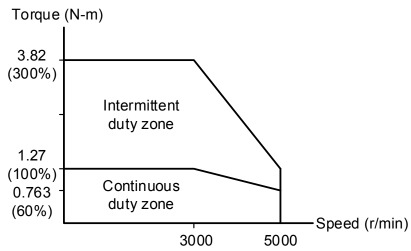
ECMA-C1040F□S



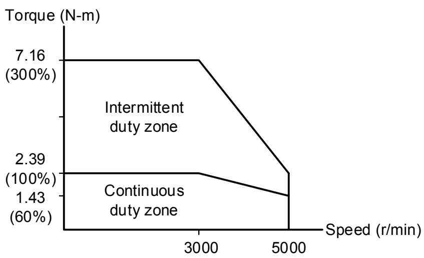
ECMA-CΔ0401□S



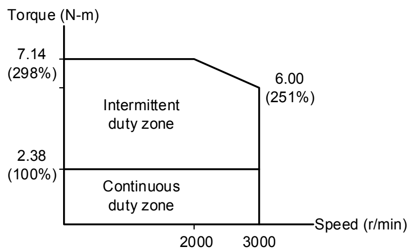
ECMA-CΔ0602□S



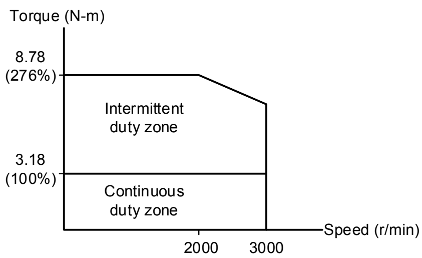
ECMA-CΔ0604□S, ECMA-CΔ0604□H
ECMA-CΔ0804□7



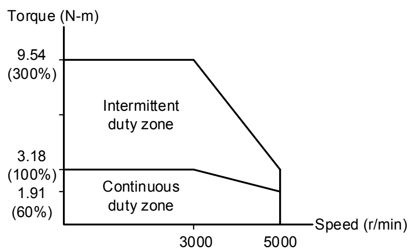
ECMA-CΔ0807□S, ECMA-CΔ0807□H



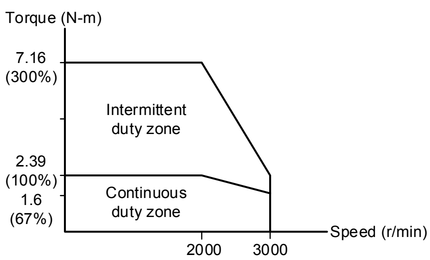
ECMA-CΔ0907□S



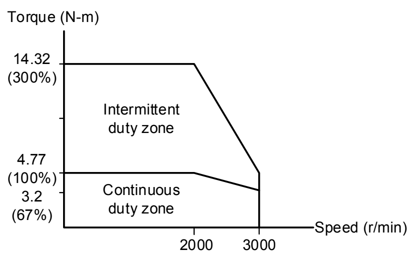
ECMA-CΔ0910□S



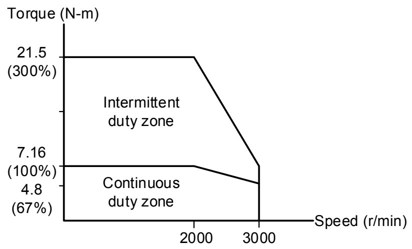
ECMA-CΔ1010□S



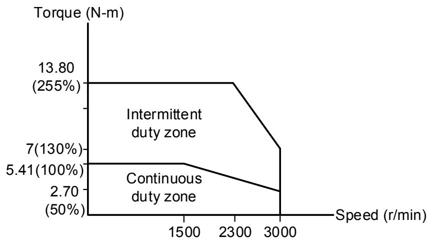
ECMA-EΔ1305□S



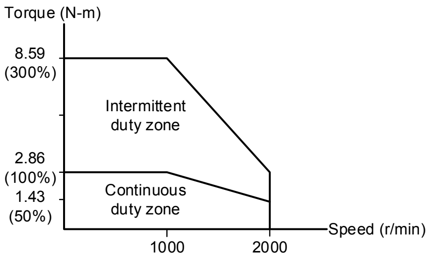
ECMA-EΔ1310□S



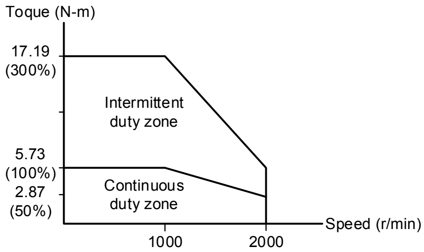
ECMA-E∆1315□S



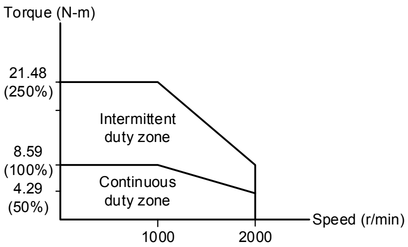
ECMA-F∆1308□S



ECMA-G∆1303□S



ECMA-G∆1306□S



ECMA-G∆1309□S

A

Overload features

Definition of overload protection

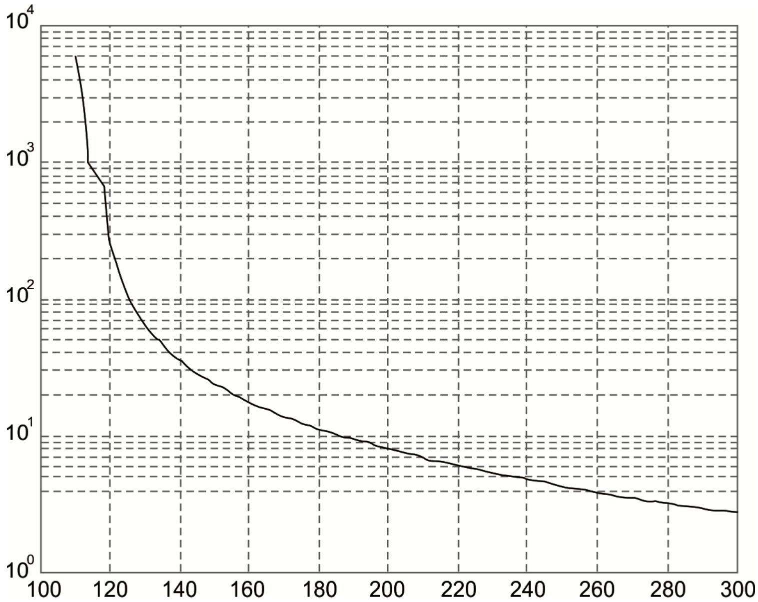
The overload protection is to prevent the motor from overheating.

Causes of overload

1. The motor's rated torque exceeds the range and the operation time is too long.
2. The inertia ratio is set too big and the motor frequently accelerates / decelerates.
3. Connection error between power cable and encoder wiring.
4. Error of servo gain setting causes resonance of the motor.
5. The motor with brake operates without releasing the brake.

Graph of load and operating time

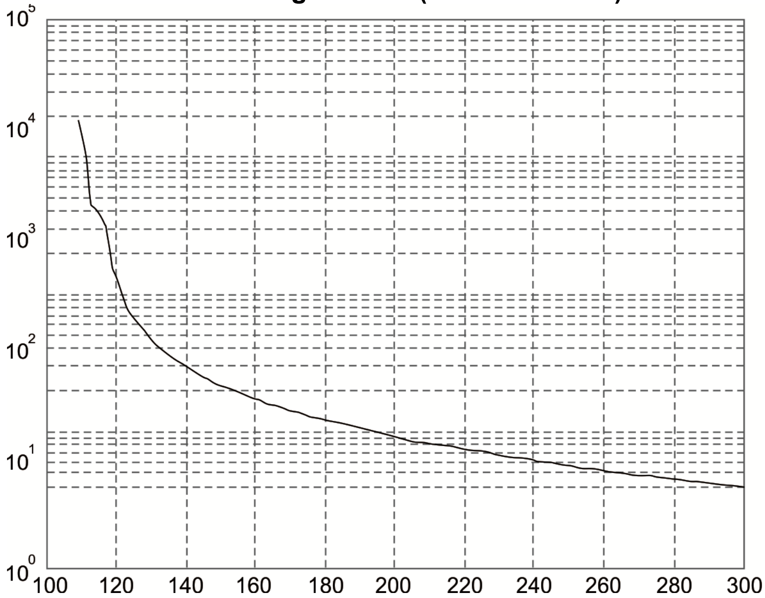
Low inertia (ECMA C series)



Load	Operating time
120%	236.8 s
140%	35.2 s
160%	17.6 s
180%	11.2 s
200%	8 s
220%	6.1 s
240%	4.8 s
260%	3.9 s
280%	3.3 s
300%	2.8 s

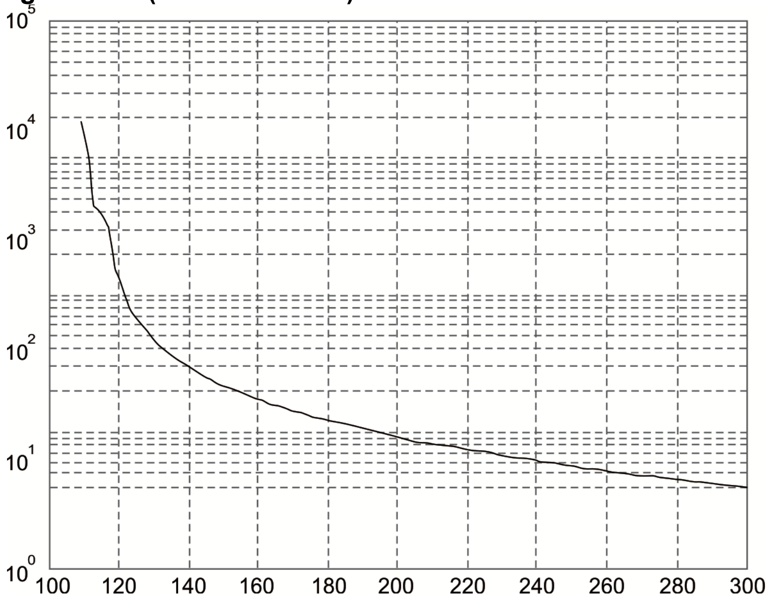
A

Medium and medium-high inertia (ECMA E series)



Load	Operating time
120%	527.6 s
140%	70.4 s
160%	35.2 s
180%	22.4 s
200%	16 s
220%	12.2 s
240%	9.6 s
260%	7.8 s
280%	6.6 s
300%	5.6 s

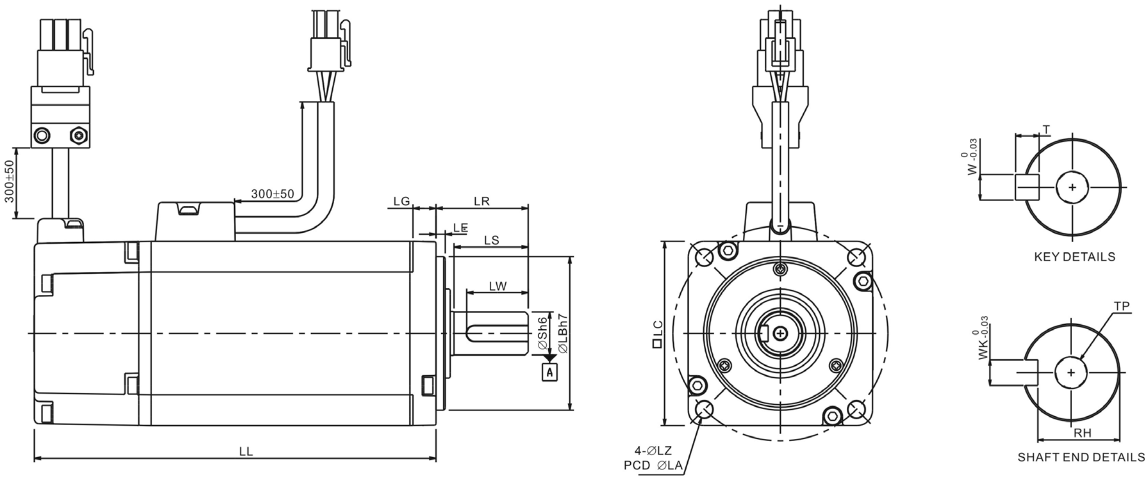
High inertia (ECMA G series)



Load	Operating time
120%	527.6 s
140%	70.4 s
160%	35.2 s
180%	22.4 s
200%	16 s
220%	12.2 s
240%	9.6 s
260%	7.8 s
280%	6.6 s
300%	5.6 s

Dimensions of servo motor

Motor frame size: 86 mm and below

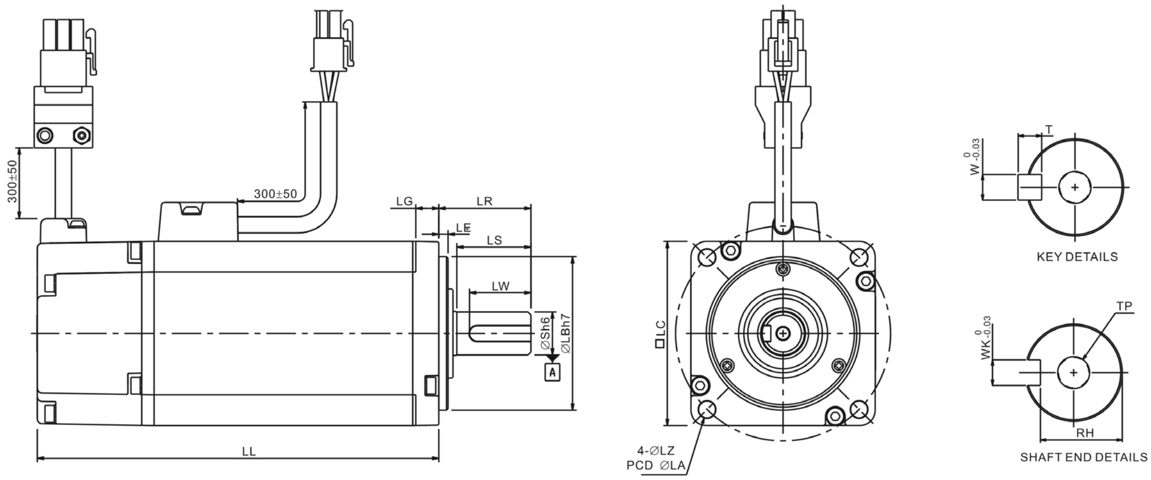


Model	C1040F□S	C△0401□S	C△0602□S	C△0604□S	C△0604□H
LC	40	40	60	60	60
LZ	4.5	4.5	5.5	5.5	5.5
LA	46	46	70	70	70
S	$8(^{+0}_{-0.009})$	$8(^{+0}_{-0.009})$	$14(^{+0}_{-0.011})$	$14(^{+0}_{-0.011})$	$14(^{+0}_{-0.011})$
LB	$30(^{+0}_{-0.021})$	$30(^{+0}_{-0.021})$	$50(^{+0}_{-0.025})$	$50(^{+0}_{-0.025})$	$50(^{+0}_{-0.025})$
LL (w/o brake)	79.1	100.6	105.5	130.7	145.8
LL (with brake)	--	136.6	141.6	166.8	176.37
LS	20	20	27	27	27
LR	25	25	30	30	30
LE	2.5	2.5	3	3	3
LG	5	5	7.5	7.5	7.5
LW	16	16	20	20	20
RH	6.2	6.2	11	11	11
WK	3	3	5	5	5
W	3	3	5	5	5
T	3	3	5	5	5
TP	--	M3 Depth 8	M4 Depth 15	M4 Depth 15	M4 Depth 15

Note:

1. Dimensions are in millimeters.
2. Dimensions and weights of the servo drive may be updated without prior notice.
3. (□) in the model names represent shaft end/brake or the number of oil seal.
4. (△) in motor model names represents encoder type. Please refer to Chapter 1 for further information.

Motor frame size: 86 mm and below

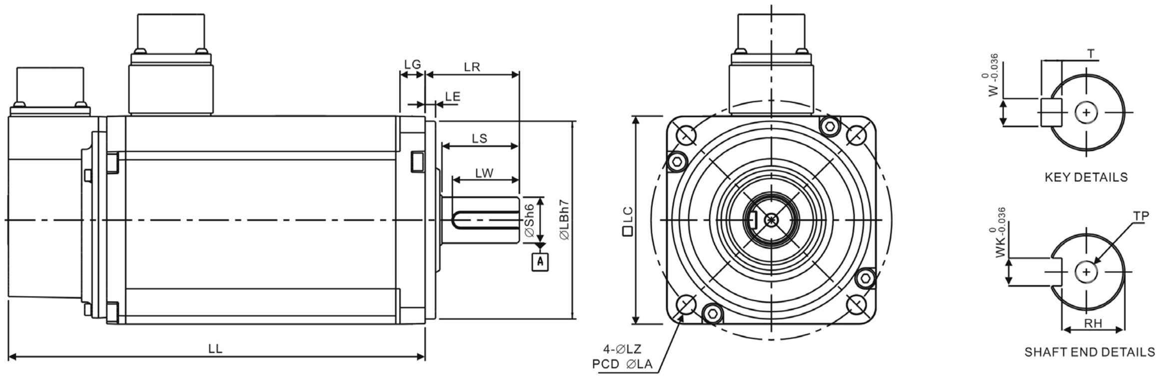


Model	C△0804□7	C△0807□S	C△0807□H	C△0907□S	C△0910□S
LC	80	80	80	86	86
LZ	6.6	6.6	6.6	6.6	6.6
LA	90	90	90	100	100
S	$14(^{+0}_{-0.011})$	$19(^{+0}_{-0.013})$	$19(^{+0}_{-0.013})$	$16(^{+0}_{-0.011})$	$16(^{+0}_{-0.011})$
LB	$70(^{+0}_{-0.030})$	$70(^{+0}_{-0.030})$	$70(^{+0}_{-0.030})$	$80(^{+0}_{-0.030})$	$80(^{+0}_{-0.030})$
LL (w/o brake)	112.3	138.3	151.1	130.2	153.2
LL (with brake)	152.8	178.0	189.0	161.3	184.3
LS	27	32	32	30	30
LR	30	35	35	35	35
LE	3	3	3	3	3
LG	8	8	8	8	8
LW	20	25	25	20	20
RH	11	15.5	15.5	13	13
WK	5	6	6	5	5
W	5	6	6	5	5
T	5	6	6	5	5
TP	M4 Depth 15	M6 Depth 20	M6 Depth 20	M5 Depth 15	M5 Depth 15

Note:

1. Dimensions are in millimeters.
2. Dimensions and weights of the servo drive may be updated without prior notice.
3. (□) in the model names represent shaft end/brake or the number of oil seal.
4. (△) in motor model names represents encoder type. Please refer to Chapter 1 for further information.

Motor frame size: 100 mm ~ 130 mm

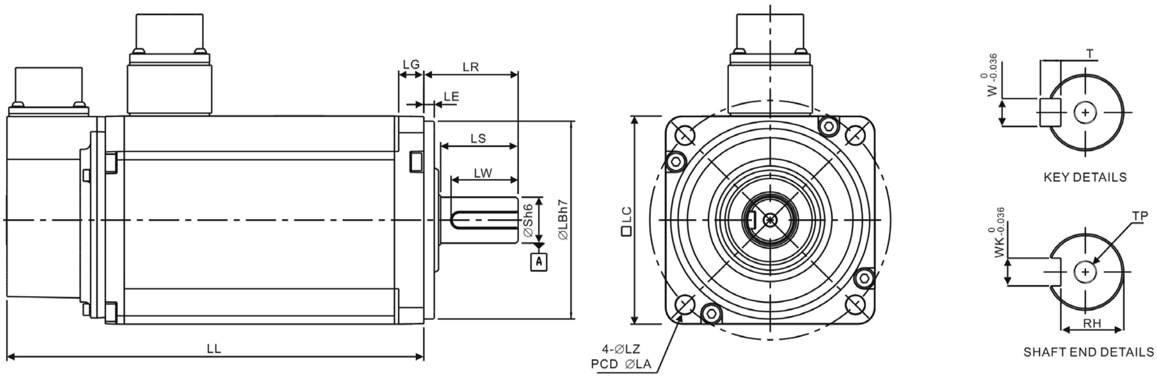


Model	E△1010□S	E△1305□S	E△1310□S	E△1315□S	F△1308□S
LC	100	130	130	130	130
LZ	9	9	9	9	9
LA	115	145	145	145	145
S	$22\left(^{+0}_{-0.013}\right)$	$22\left(^{+0}_{-0.013}\right)$	$22\left(^{+0}_{-0.013}\right)$	$22\left(^{+0}_{-0.013}\right)$	$22\left(^{+0}_{-0.013}\right)$
LB	$95\left(^{+0}_{-0.035}\right)$	$110\left(^{+0}_{-0.035}\right)$	$110\left(^{+0}_{-0.035}\right)$	$110\left(^{+0}_{-0.035}\right)$	$110\left(^{+0}_{-0.035}\right)$
LL (w/o brake)	153.3	147.5	147.5	167.5	152.5
LL (with brake)	192.5	183.5	183.5	202.0	181.0
LS	37	47	47	47	47
LR	45	55	55	55	55
LE	5	6	6	6	6
LG	12	11.5	11.5	11.5	11.5
LW	32	36	36	36	36
RH	18	18	18	18	18
WK	8	8	8	8	8
W	8	8	8	8	8
T	--	7	7	7	7
TP	M6 Depth 20	M6 Depth 20	M6 Depth 20	M6 Depth 20	M6 Depth 20

Note:

1. Dimensions are in millimeters.
2. Dimensions and weights of the servo drive may be updated without prior notice.
3. (□) in the model names represent shaft end/brake or the number of oil seal.
4. (△) in motor model names represents encoder type. Please refer to Chapter 1 for further information.

Motor frame size: 100 mm ~ 130 mm



Model	G△1303□S	G△1306□S	G△1309□S
LC	130	130	130
LZ	9	9	9
LA	145	145	145
S	$22(^{+0}_{-0.013})$	$22(^{+0}_{-0.013})$	$22(^{+0}_{-0.013})$
LB	$110(^{+0}_{-0.035})$	$110(^{+0}_{-0.035})$	$110(^{+0}_{-0.035})$
LL (w/o brake)	147.5	147.5	163.5
LL (with brake)	183.5	183.5	198.0
LS	47	47	47
LR	55	55	55
LE	6	6	6
LG	11.5	11.5	11.5
LW	36	36	36
RH	18	18	18
WK	8	8	8
W	8	8	8
T	7	7	7
TP	M6 Depth 20	M6 Depth 20	M6 Depth 20

Note:

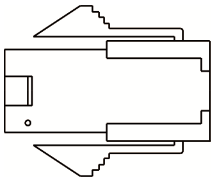
1. Dimensions are in millimeters.
2. Dimensions and weights of the servo drive may be updated without prior notice.
3. (□) in the model names represent shaft end/brake or the number of oil seal.
4. (△) in motor model names represents encoder type. Please refer to Chapter 1 for further information.

Accessories Appendix B

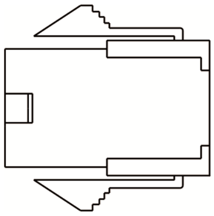
B

Power connector

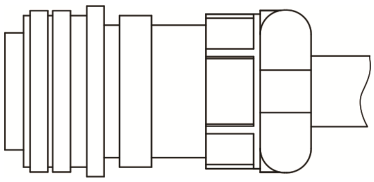
Delta Part Number: ASDBCAPW0000



Delta Part Number: ASDBCAPW0100



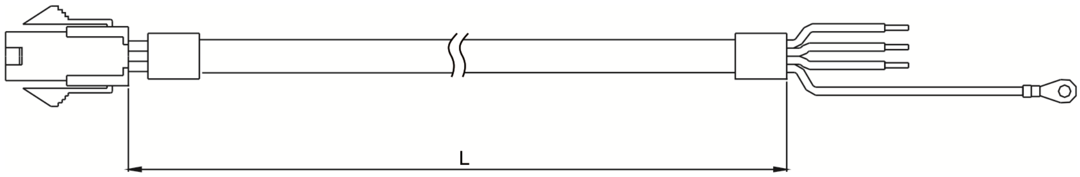
Delta Part Number: ASD-CAPW1000



B

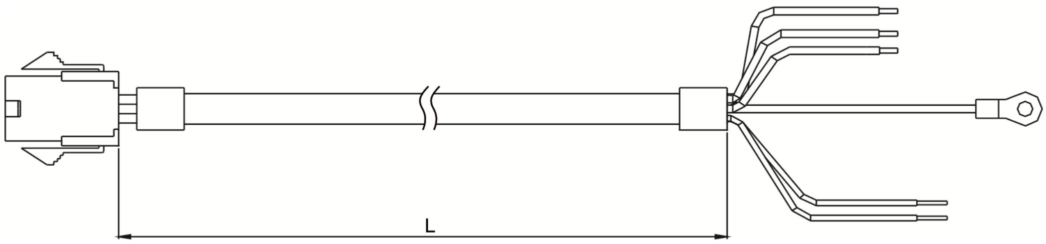
Power cable

Delta Part Number: ASD-ABPW0003, ASD-ABPW0005



Title	Part No.	L	
		mm	inch
1	ASD-ABPW0003	3000 ± 100	118 ± 4
2	ASD-ABPW0005	5000 ± 100	197 ± 4

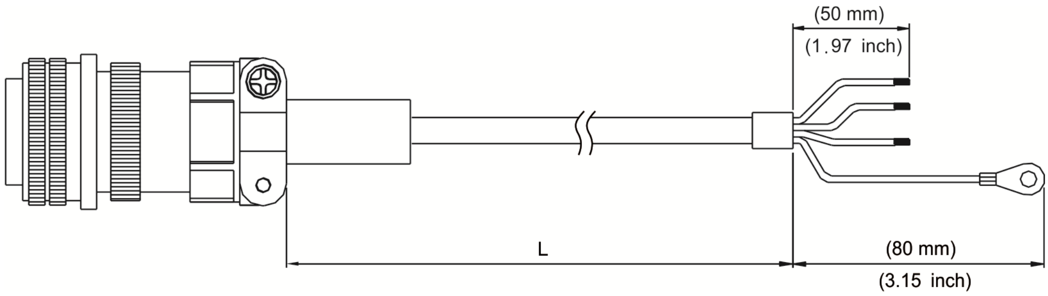
Delta Part Number: ASD-ABPW0103, ASD-ABPW0105



Title	Part No.	L	
		mm	inch
1	ASD-ABPW0103	3000 ± 100	118 ± 4
2	ASD-ABPW0105	5000 ± 100	197 ± 4

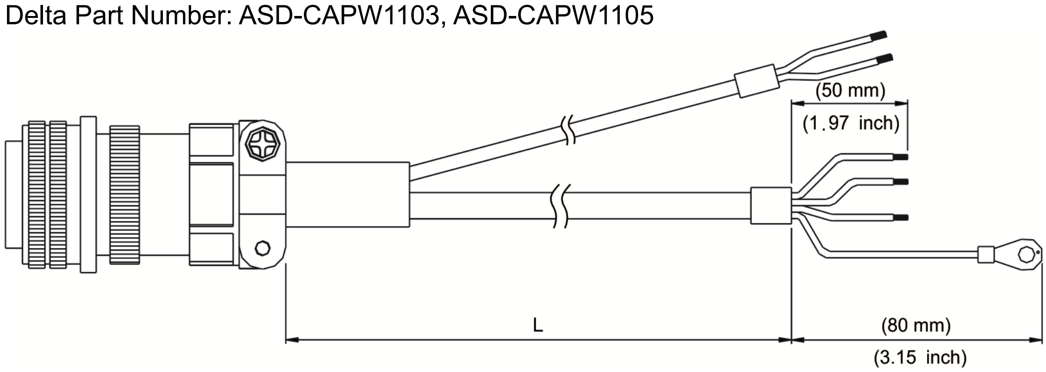
B

Delta Part Number: ASD-CAPW1003, ASD-CAPW1005



Title	Part No.	L	
		mm	inch
1	ASD-CAPW1003	3000 ± 100	118 ± 4
2	ASD-CAPW1005	5000 ± 100	197 ± 4

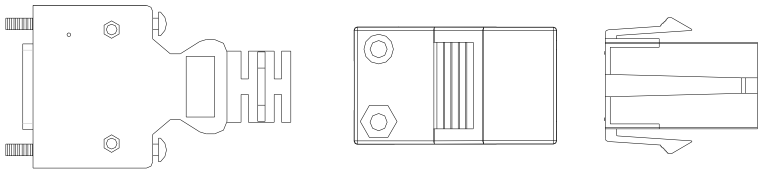
Delta Part Number: ASD-CAPW1103, ASD-CAPW1105



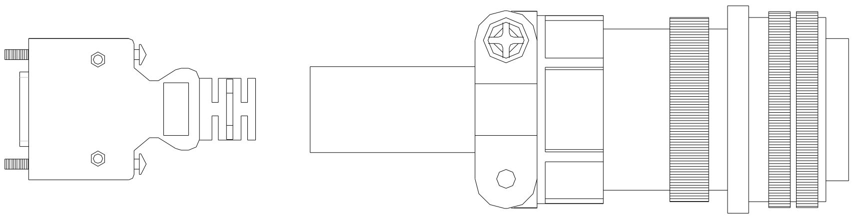
Title	Part No.	L	
		mm	inch
1	ASD-CAPW1103	3000 ± 100	118 ± 4
2	ASD-CAPW1105	5000 ± 100	197 ± 4

Encoder connector

Delta Part Number: ASD-ABEN0000



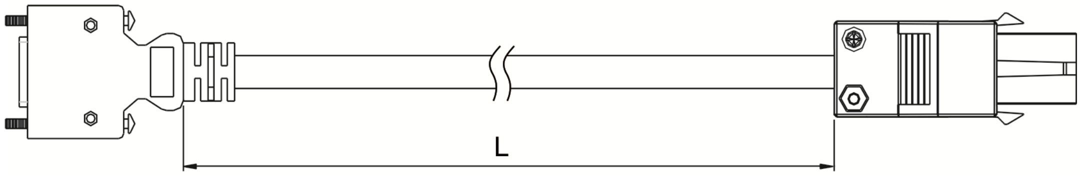
Delta Part Number: ASD-CAEN1000



B

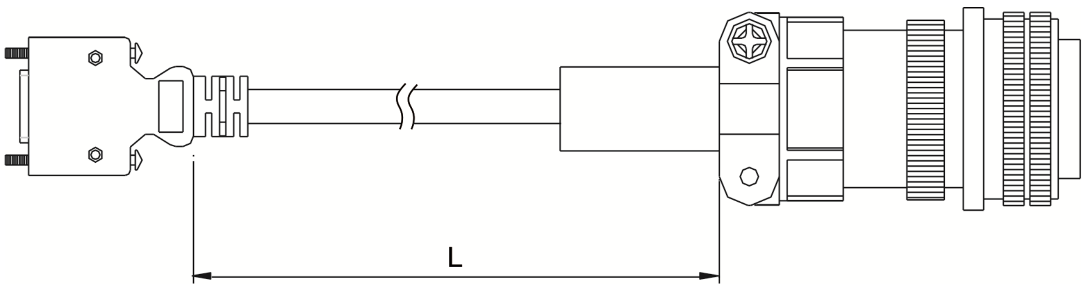
Encoder cable

Delta Part Number: ASD-ABEN0003, ASD-ABEN0005



Title	Part No.	L	
		mm	inch
1	ASD-ABEN0003	3000 ± 100	118 ±4
2	ASD-ABEN0005	5000 ± 100	197 ± 4

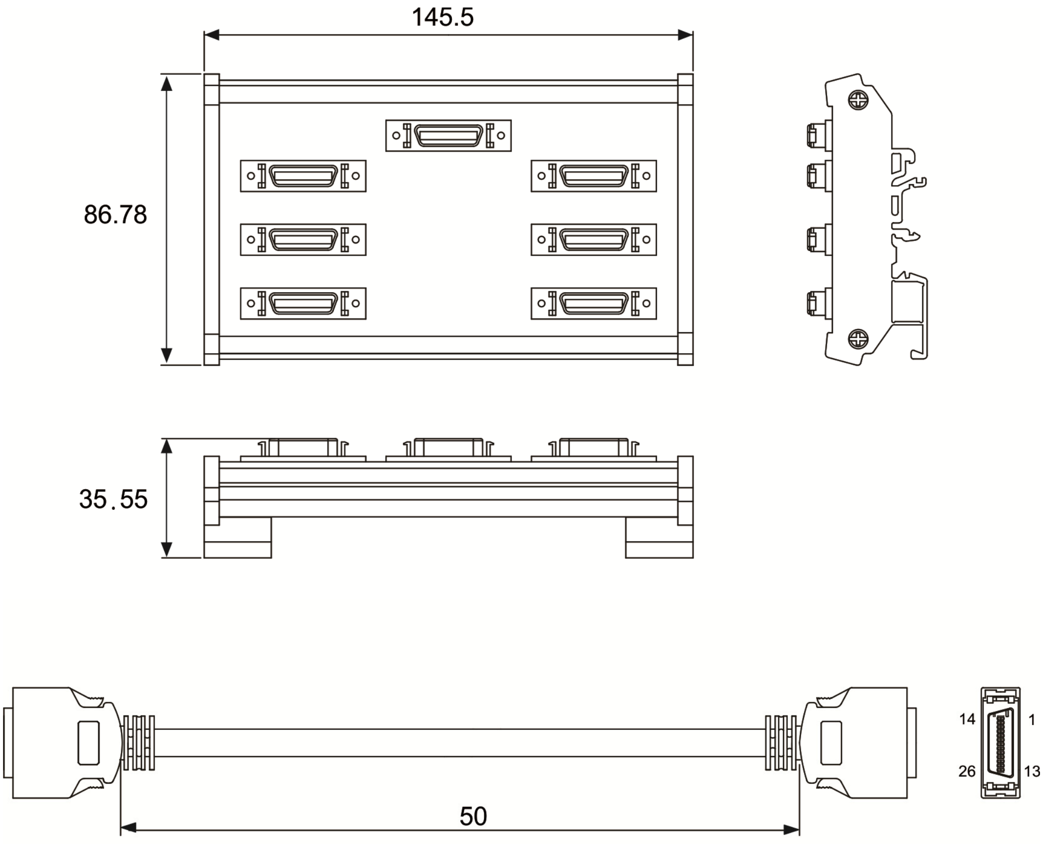
Delta Part Number: ASD-CAEN1003, ASD-CAEN1005



Title	Part No.	L	
		mm	inch
1	ASD-CAEN1003	3000 ± 100	118 ±4
2	ASD-CAEN1005	5000 ± 100	197 ± 4

Encoder conversion module

Part Number: ASD-PBSC2626

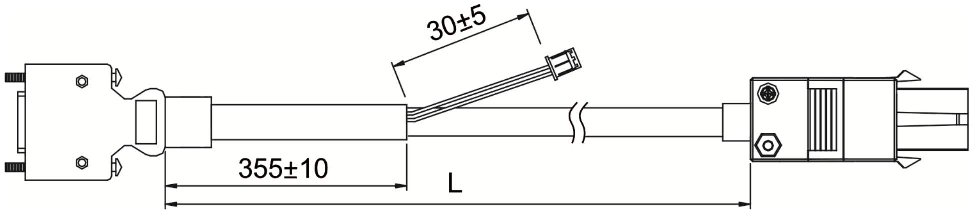


Unit: mm

B

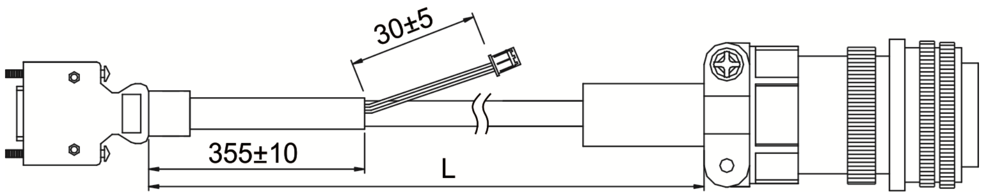
Encoder cable (Absolute type)

Delta Part Number: ASD-B2EB0003, ASD-B2EB0005



Title	Part No.	L	
		mm	inch
1	ASD-B2EB0003	3000 ± 100	118 ± 4
2	ASD-B2EB0005	5000 ± 100	197 ± 4

Delta Part Number: ASD-B2EB1003, ASD-B2EB1005



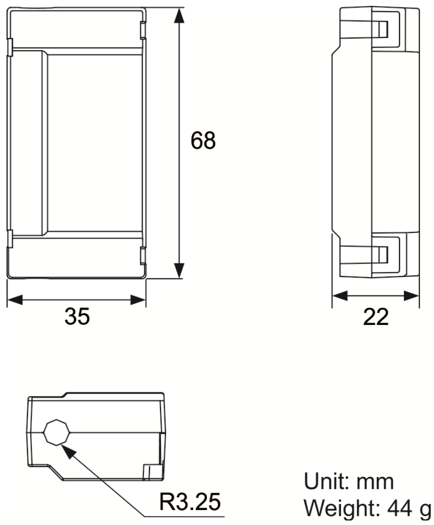
Title	Part No.	L	
		mm	inch
1	ASD-B2EB1003	3000 ± 100	118 ± 4
2	ASD-B2EB1005	5000 ± 100	197 ± 4

B

Battery box (Absolute type)

Single battery box

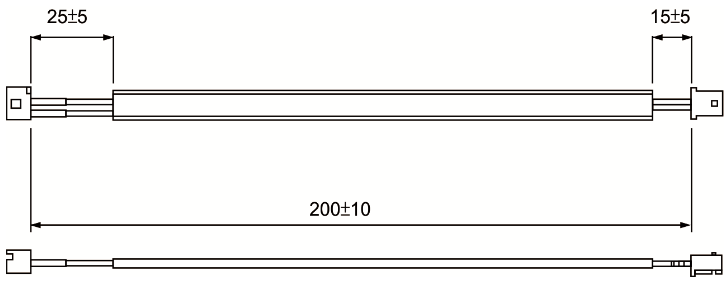
Delta Part Number: ASD-MDBT0100



Unit: mm
Weight: 44 g

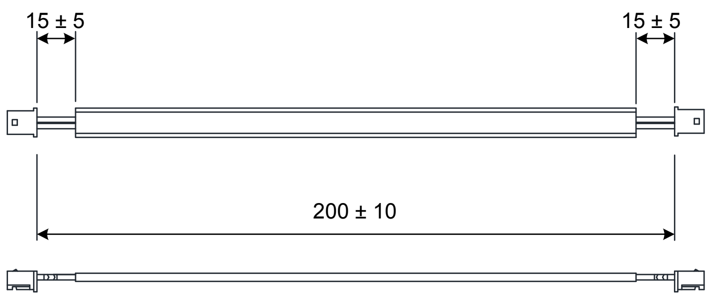
Battery box cable AW

Delta Part Number: 3864573700



Battery box cable IW

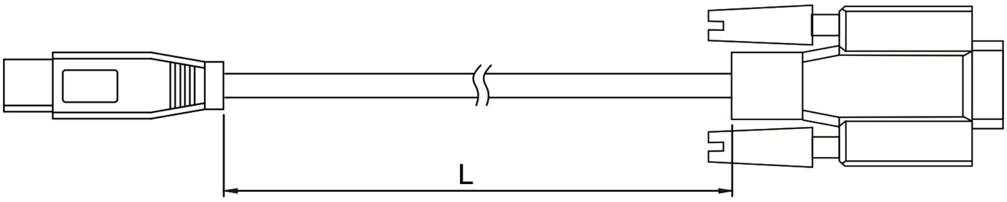
Delta Part Number: 3864811900



B

RS-232 communication cable

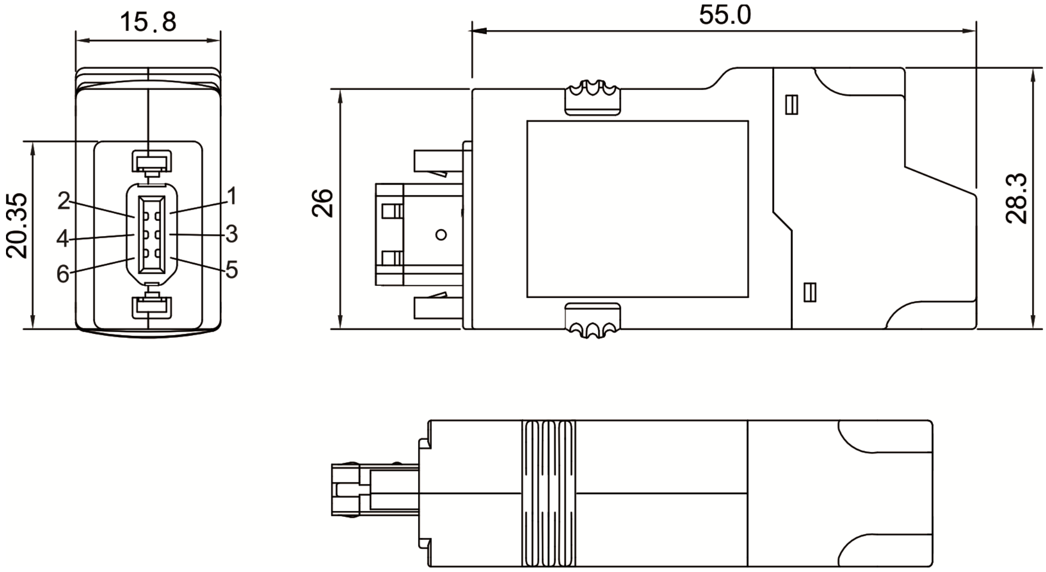
Delta Part Number: ASD-CARS0003



Title	Part No.	L	
		mm	inch
1	ASD-CARS0003	3000 ± 100	118 ±4

RS-485 connector

Delta Part Number: ASD-CNIE0B06



Optional accessories

750 W MS controller with 50 W low-inertia motor

MS controller	ASD-MS-0721-F
Low-inertia motor	ECMA-C1040F□S
Motor power cable (without brake)	ASD-ABPW000X
Power connector (with brake)	ASDBCAPW0000
Motor power cable (with brake)	ASD-ABPW010X
Power connector (with brake)	ASDBCAPW0100
Encoder cable (Incremental type)	ASD-ABEN000X
Encoder cable (Absolute type)	ASD-A2EB000X
Encoder connector	ASD-ABEN0000

(X = 3 indicates that the cable length is 3 m; X = 5 indicates that the cable length is 5 m)

750 W MS controller with 100 W low-inertia motor

MS controller	ASD-MS-0721-F
Low-inertia motor	ECMA-C△0401□S
Motor power cable (without brake)	ASD-ABPW000X
Power connector (with brake)	ASDBCAPW0000
Motor power cable (with brake)	ASD-ABPW010X
Power connector (with brake)	ASDBCAPW0100
Encoder cable (Incremental type)	ASD-ABEN000X
Encoder cable (Absolute type)	ASD-A2EB000X
Encoder connector	ASD-ABEN0000

(X = 3 indicates that the cable length is 3 m; X = 5 indicates that the cable length is 5 m)

Note:
1. The last number at the end of servo motor model is the model name ASDA-MS. Please refer to the product that you purchased for model name information.
2. (□) in motor model names represents brake or keyway / oil seal.
3. (△) in motor model names represents encoder type. Please refer to Chapter 1 for further information.

B

750 W MS controller with 200 W low-inertia motor

MS controller	ASD-MS-0721-F
Low-inertia motor	ECMA-C△0602□S
Motor power cable (without brake)	ASD-ABPW000X
Power connector (with brake)	ASDBCAPW0000
Motor power cable (with brake)	ASD-ABPW010X
Power connector (with brake)	ASDBCAPW0100
Encoder cable (Incremental type)	ASD-ABEN000X
Encoder cable (Absolute type)	ASD-A2EB000X
Encoder connector	ASD-ABEN0000

(X = 3 indicates that the cable length is 3 m; X = 5 indicates that the cable length is 5 m)

750 W MS controller with 400 W low-inertia motor

MS controller	ASD-MS-0721-F
Low-inertia motor	ECMA-C△0604□S ECMA-C△0604□H ECMA-C△0604□7
Motor power cable (without brake)	ASD-ABPW000X
Power connector (with brake)	ASDBCAPW0000
Motor power cable (with brake)	ASD-ABPW010X
Power connector (with brake)	ASDBCAPW0100
Encoder cable (Incremental type)	ASD-ABEN000X
Encoder cable (Absolute type)	ASD-A2EB000X
Encoder connector	ASD-ABEN0000

(X = 3 indicates that the cable length is 3 m; X = 5 indicates that the cable length is 5 m)

750 W MS controller with 500 W medium-inertia motor

MS controller	ASD-MS-0721-F
Medium-inertia motor	ECMA-E△1305□S
Motor power cable (without brake)	ASD-CAPW100X
Power connector (with brake)	ASD-CAPW110X
Motor power cable (with brake)	ASD-CAPW1000
Power connector (with brake)	ASD-CAEN100X
Encoder cable (Incremental type)	ASD-A2EB100X
Encoder cable (Absolute type)	ASD-CAEN1000
Encoder connector	ASD-ABEN0000

(X = 3 indicates that the cable length is 3 m; X = 5 indicates that the cable length is 5 m)

750 W MS controller with 300 W high-inertia motor

MS controller	ASD-MS-0721-F
High-inertia motor	ECMA-G△1303□S
Motor power cable (without brake)	ASD-CAPW100X
Power connector (with brake)	ASD-CAPW110X
Motor power cable (with brake)	ASD-CAPW1000
Power connector (with brake)	ASD-CAEN100X
Encoder cable (Incremental type)	ASD-A2EB100X
Encoder cable (Absolute type)	ASD-CAEN1000
Encoder connector	ASD-ABEN0000

(X = 3 indicates that the cable length is 3 m; X = 5 indicates that the cable length is 5 m)

Note:

1. The last number at the end of servo motor model is the model name ASDA-MS. Please refer to the product that you purchased for model name information.
2. (□) in motor model names represents brake or keyway / oil seal.
3. (△) in motor model names represents encoder type. Please refer to Chapter 1 for further information.

B

750 W MS controller with 750 W low-inertia motor

MS controller	ASD-MS-0721-F
Low-inertia motor	ECMA-C△0807□S ECMA-C△0807□H ECMA-C△0907□S
Motor power cable (without brake)	ASD-ABPW000X
Power connector (with brake)	ASDBCAPW0000
Motor power cable (with brake)	ASD-ABPW010X
Power connector (with brake)	ASDBCAPW0100
Encoder cable (Incremental type)	ASD-ABEN000X
Encoder cable (Absolute type)	ASD-A2EB000X
Encoder connector	ASD-ABEN0000

(X = 3 indicates that the cable length is 3 m; X = 5 indicates that the cable length is 5 m)

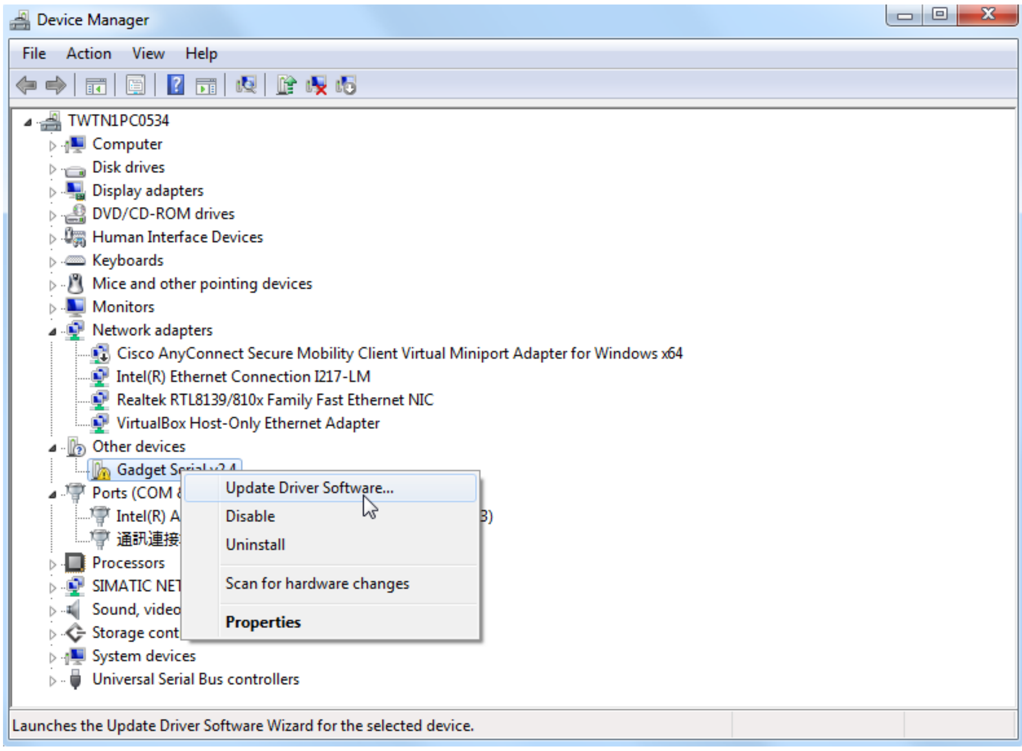
Other accessories (applicable to ASDA-MS series)	
Name	Delta part number
RS-232 communication cable	ASD-CARS0003
RS-485 connector	ASD-CNIE0B06

Appendix C

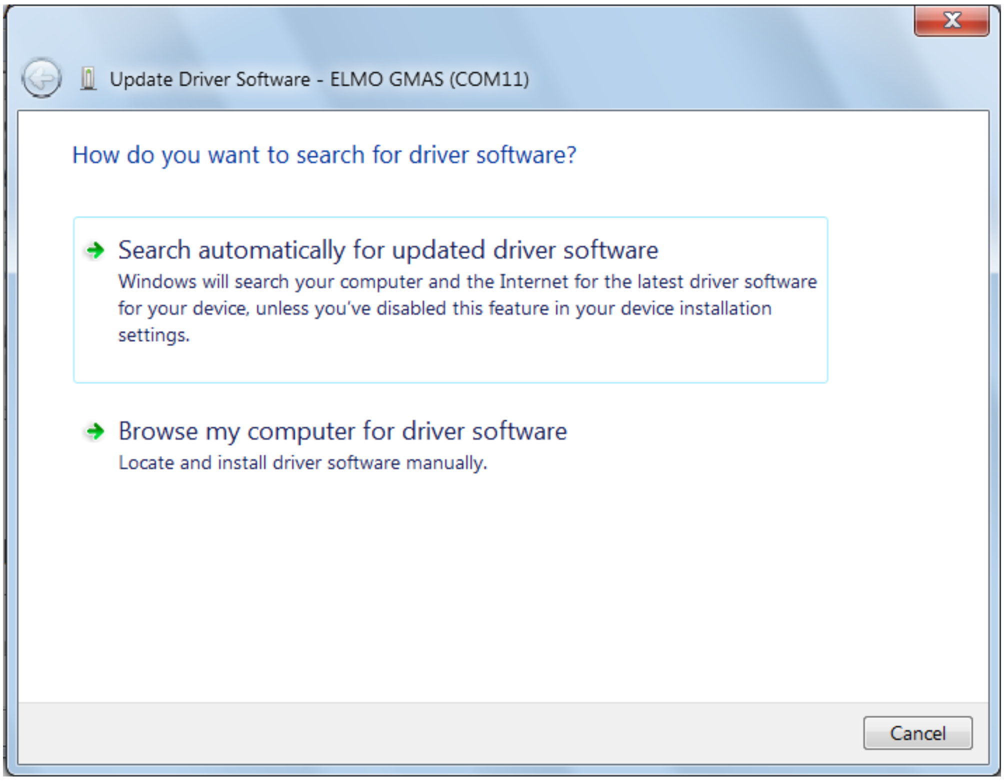
Install USB-Serial driver software

Steps of installing USB-Serial driver:

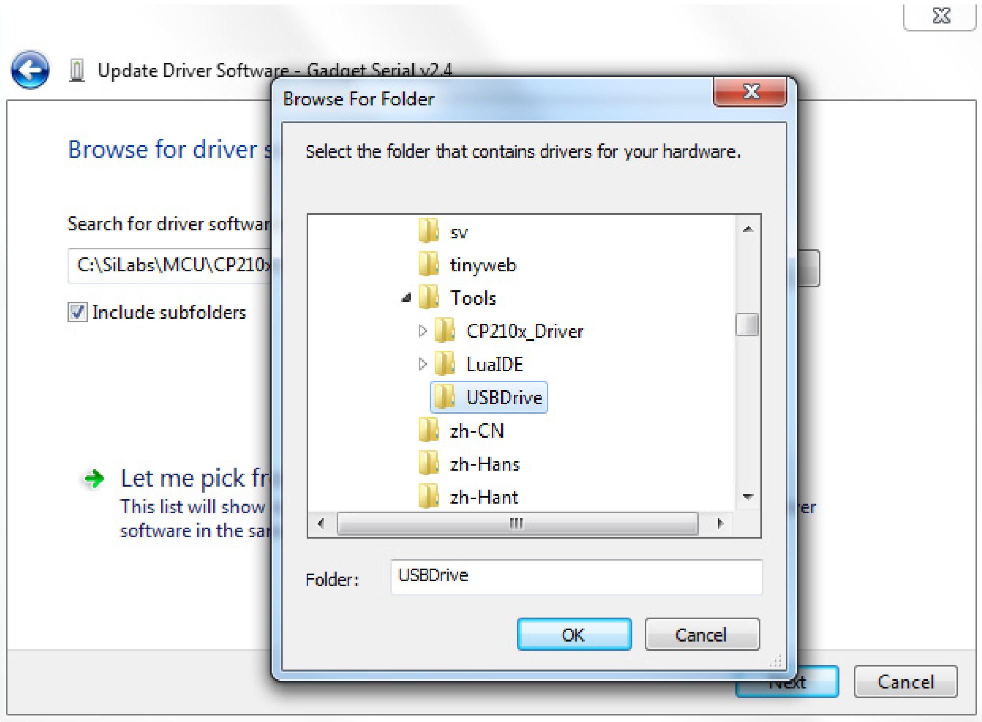
Step 1: Open **Devise Manager**, and click **Other Devices**. Then right click **Gadget Serial v2.4**, and select **Update Driver Software** from the drop-down menu.



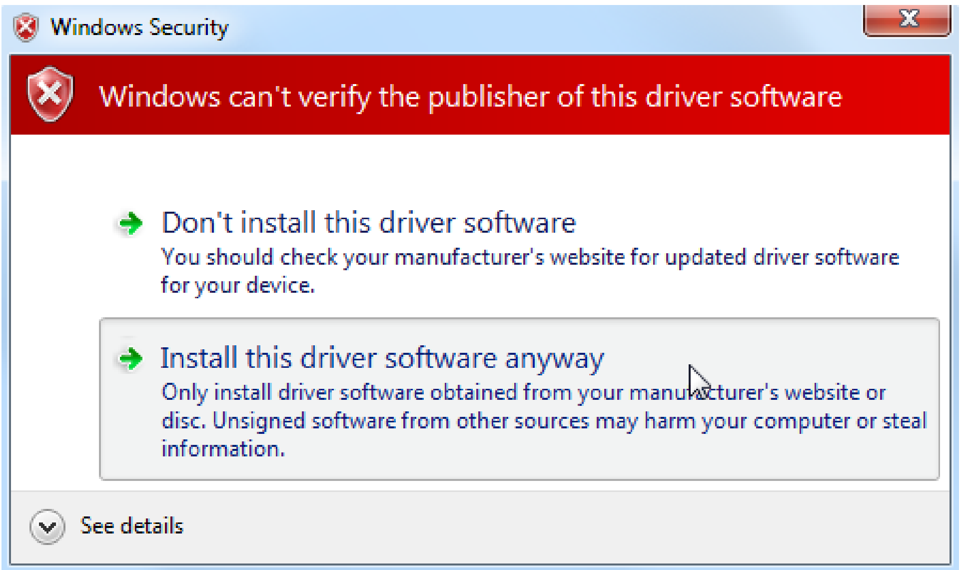
Step 2: Select **Browse my computer for driver software**.



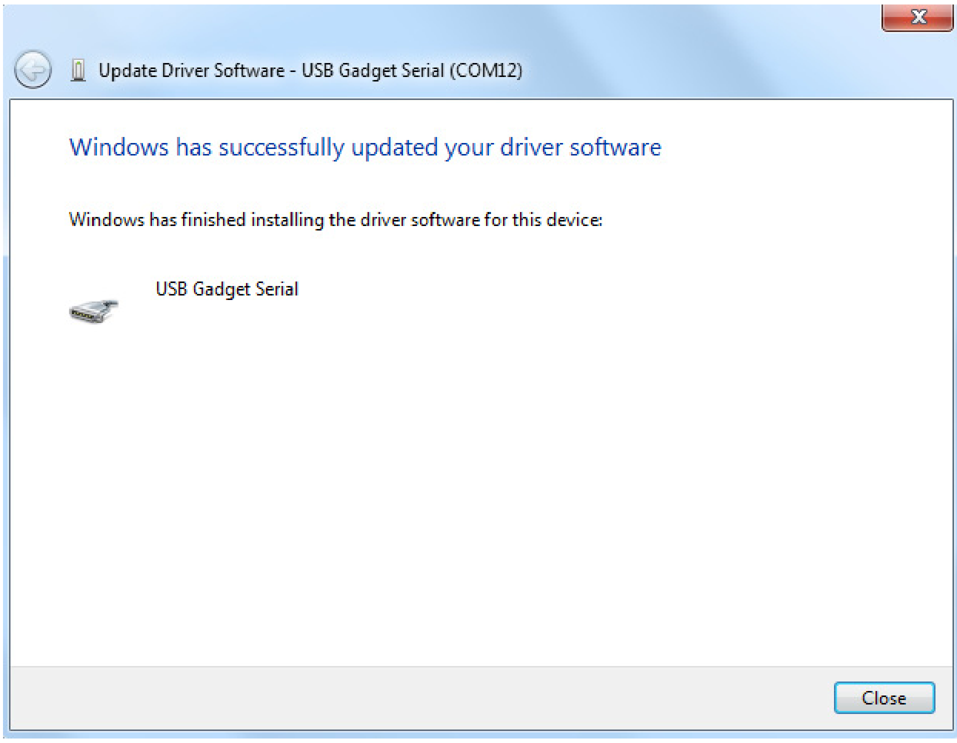
Step 3: Press the **Browse** button to open the folder of linux-cdc-acm.inf. Then, select **USBDrive**.



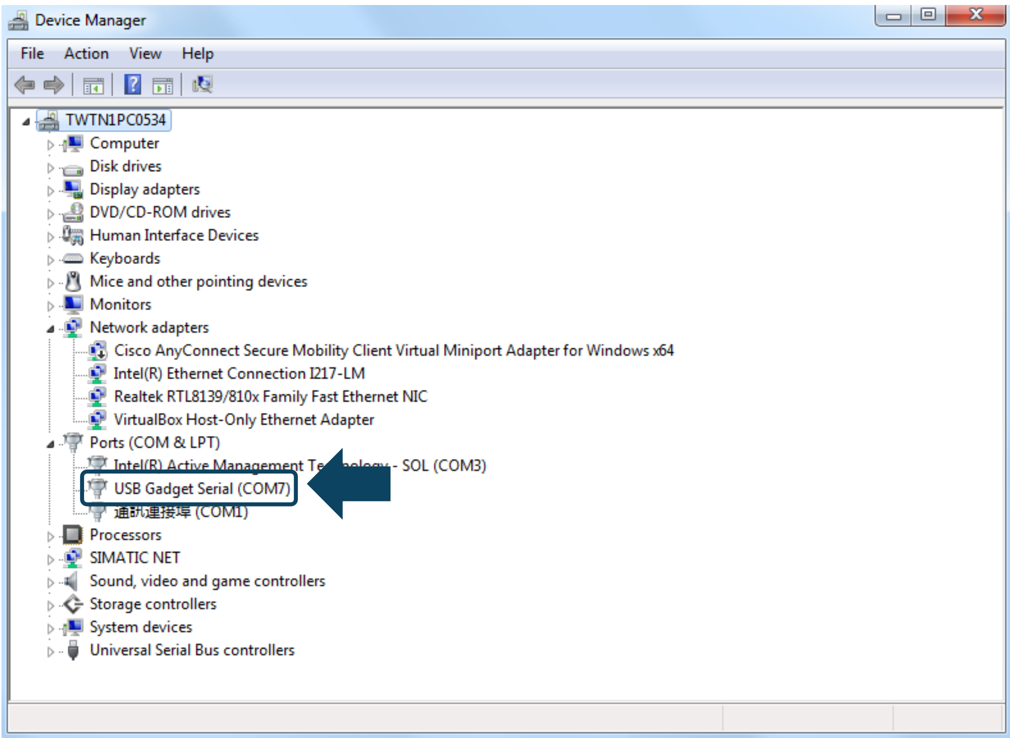
Step 4: Please select **Install this driver software anyway**.



Installation complete.



A device named **USB Gadget Seril (COM?)** will appear under Ports (COM & LPT) in device manager.

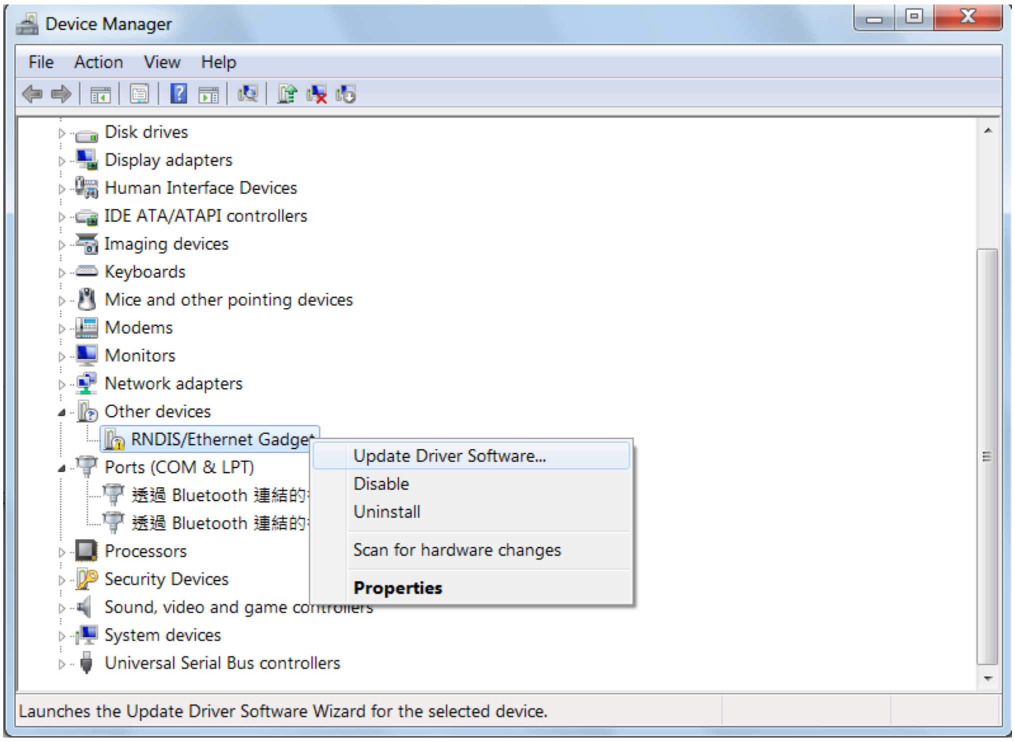


Note: The question mark **(?)** in **USB Gadget Serial(COM?)** is a variable, which will be automatically specified by PC.

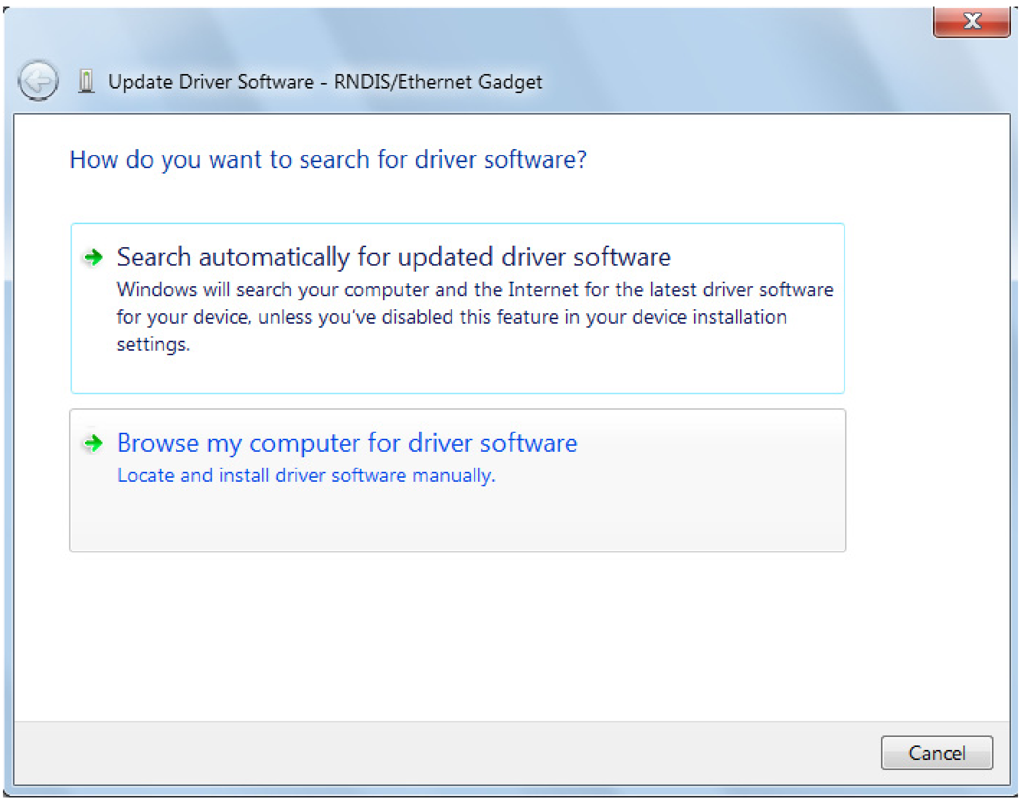
Install USB-EtherNet driver software

Steps of installing USB-EtherNet driver:

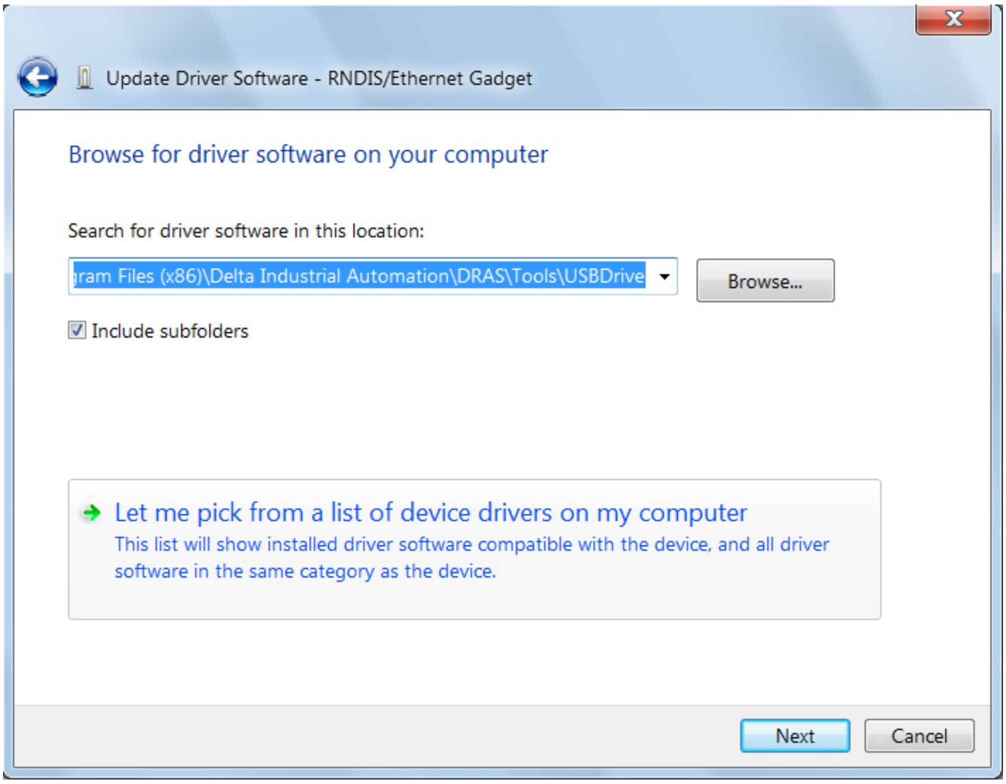
Step 1: Open **Devise Manager**, and click **Other Devices**. Then right click **RNDIS/EtherNetGadget**, and select **Update Driver Software** from the drop-down menu.



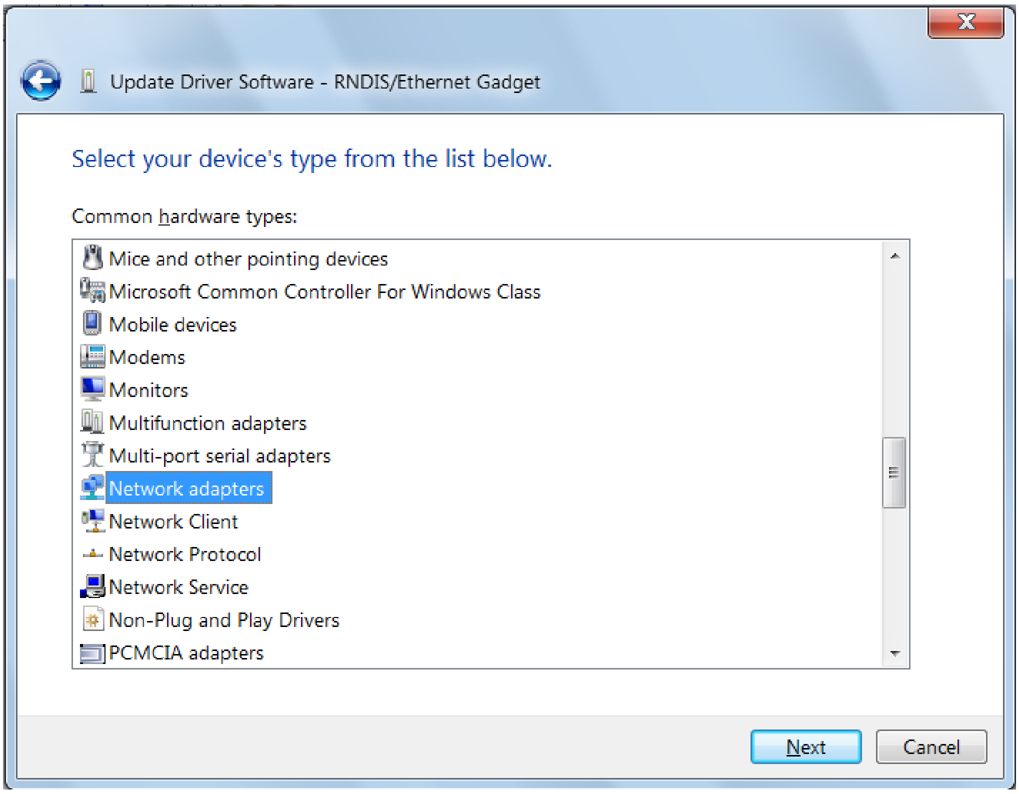
Step 2: Select **Browse my computer for driver software**.



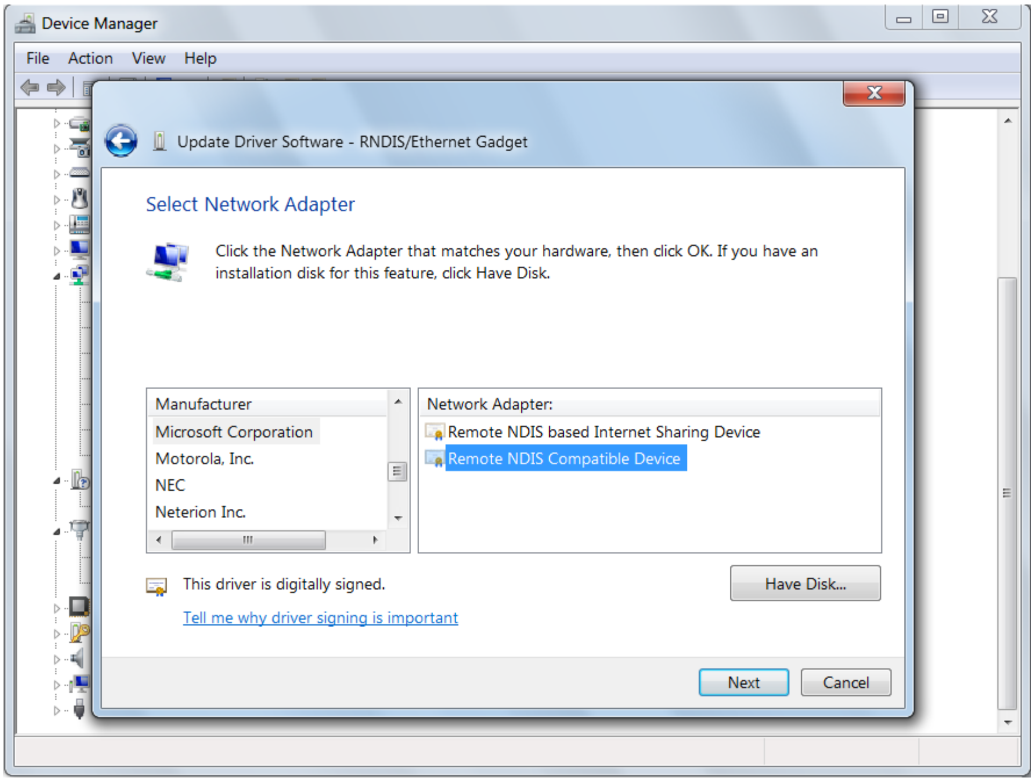
Step 3: Select **Let me pick from a list of device drivers on my computer**.



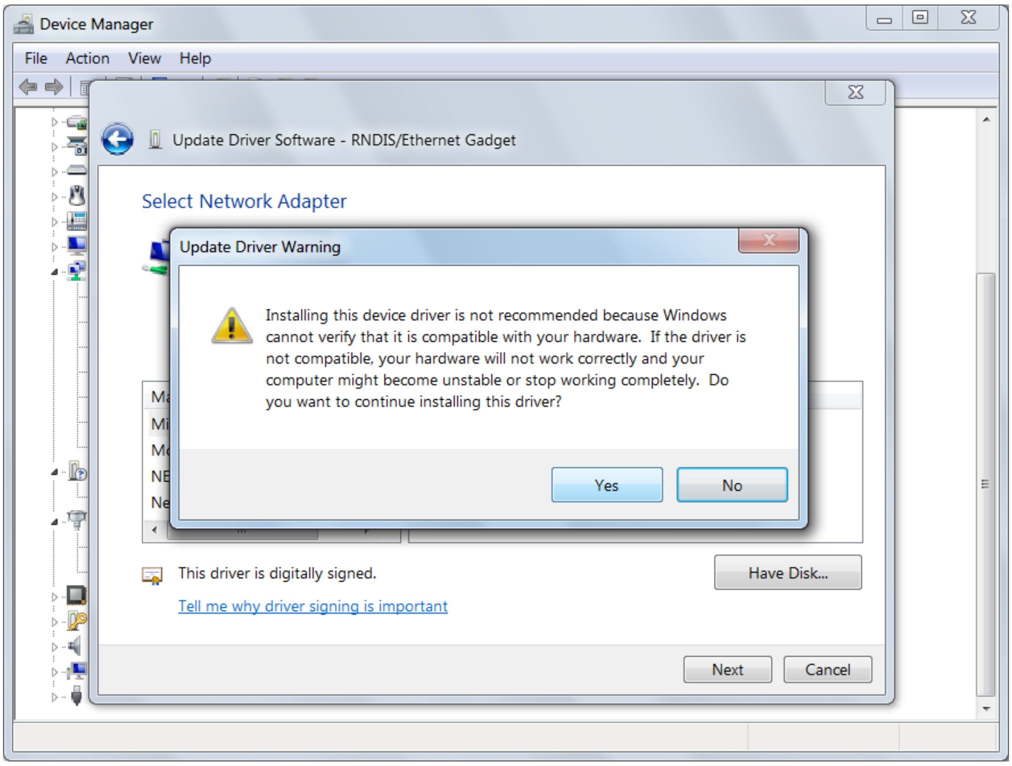
Step 4: Then, select **Network adapters**.



Step 5: In this step, you should select **Microsoft Corporation** as manufacturer first. Then, choose **Remote NDIS Compatible Device**.

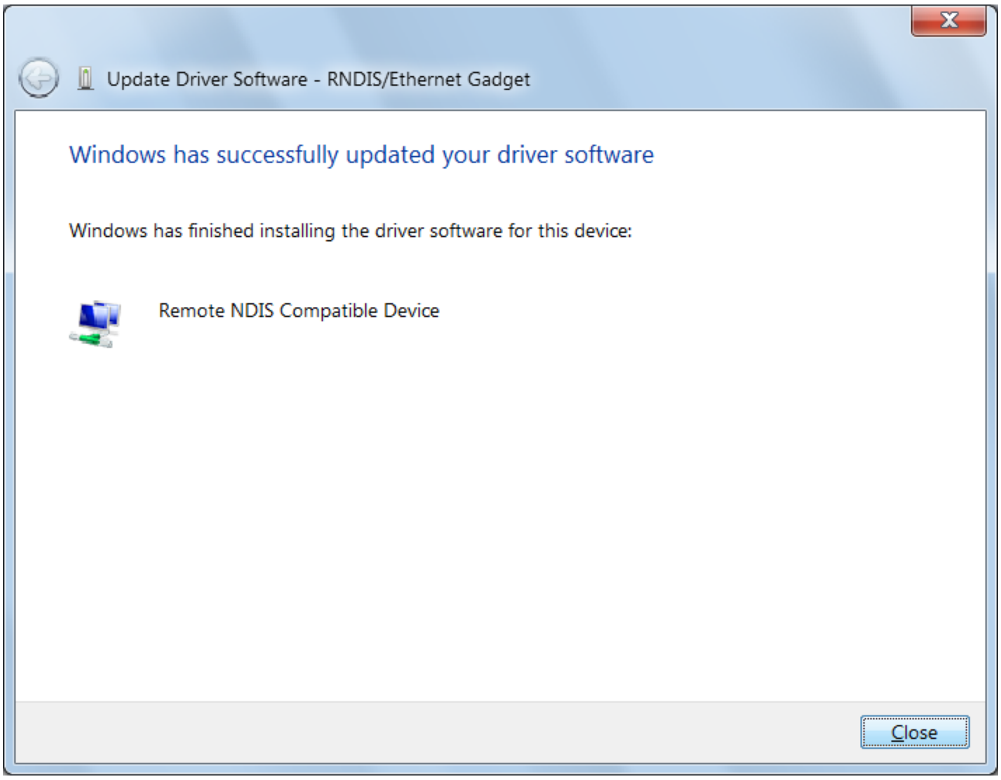


Step 6: And select Yes to continue installing this driver.

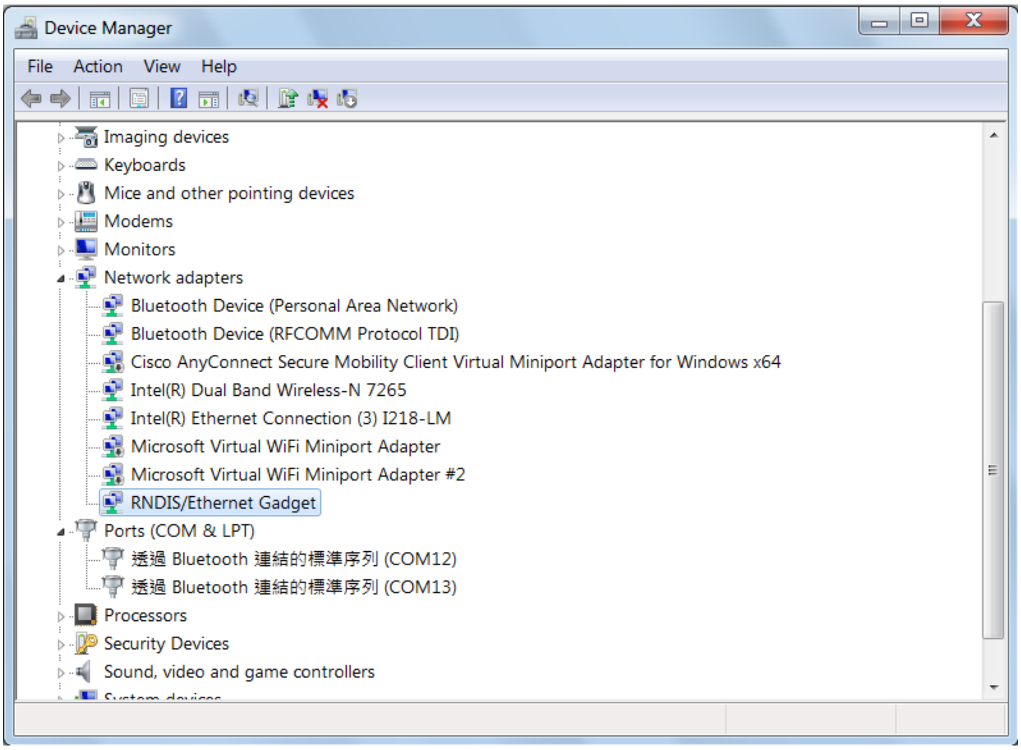


C

Installation complete.



You will see **RNDIS/EtherNet Gadget** under Network adapters.



Revision History

Date of Release	Version	Revision
December, 2016	V1.0 (First version)	-
-	-	-
-	-	-

For more information about ASDA-MS user manual, please refer to

(1) Delta Robot Automation Studio (DRAS) User Guide (Released on March, 04, 2016)

(This page is intentionally left blank.)