

***Orientalmotor***

\* HP - P024 \*

HP-P024-4

**5-phase stepping motor unit**

**CRK Series**

**Built-in Controller (Stored Program)**

**Package**

---

**OPERATING MANUAL**



Thank you for purchasing an Oriental Motor product.

This Operating Manual describes product handling procedures and safety precautions.



- Please read it thoroughly to ensure safe operation.
- Always keep the manual where it is readily available.

## Table of contents

1	Safety precautions.....	3	11.3	Setting the switches.....	63
2	Overview of the CRK series built-in controller (Stored program) .....	5	12	Program Creation and Execution .....	66
3	System configuration .....	6	12.1	Overview of Operation .....	66
4	Introduction.....	7	12.2	Communication and Terminal Specifications.....	67
5	Precautions for use .....	8	12.3	Communication Mode.....	67
6	Preparation .....	10	12.4	Communication Timing .....	68
6.1	Checking the product.....	10	12.5	Creating a New Sequence.....	68
6.2	Combinations of motors and drivers ...	11	12.6	Editing an Existing Sequence .....	70
6.3	Names and functions of parts .....	14	12.7	Executing a Sequence .....	75
7	Installation .....	16	12.8	Error Messages .....	76
7.1	Location for installation .....	16	13	Command List .....	81
7.2	Installing the motor .....	16	14	Troubleshooting.....	288
7.3	Installing a load.....	18	14.1	Protective Functions .....	288
7.4	Permissible overhung load and permissible thrust load.....	20	14.2	Corrective Actions.....	290
7.5	Installing the driver.....	22	15	Additional Functions .....	293
7.6	Installing and wiring in compliance with EMC Directive.....	23	15.1	Software over travel .....	293
8	Connection .....	25	15.2	Hardware over travel .....	293
8.1	Connecting the motor .....	25	15.3	Position control .....	293
8.2	Connecting the power supply and grounding the driver.....	27	16	Alarms and warnings.....	294
8.3	Connecting the I/O signals.....	28	16.1	Alarms .....	294
8.4	Connecting an encoder.....	31	16.2	Warnings.....	297
8.5	Connecting the RS-485 communication cable .....	32	17	Inspection .....	298
9	Explanation of I/O signals.....	33	18	General specifications .....	299
9.1	Input signals .....	33	19	Options (sold separately) .....	300
9.2	Output signals.....	35	20	Sample Programs.....	301
10	Features .....	39	20.1	Repeated Positioning Operation .....	301
10.1	Overview.....	39	20.2	Speed Change On-The-Fly .....	302
10.2	Making the Motor Move .....	39	20.3	Speed Change On Input .....	303
10.3	Motion Types .....	40	20.4	Speed Change During Index Move ...	304
	3-sensor homing operation pattern .....	49	20.5	Looped Index Move .....	305
	2-sensor homing operation pattern .....	50	20.6	Executing Linked Operation.....	305
10.4	Stopping Motion.....	50	21	Multi-Drop Connections .....	306
10.5	Encoder input .....	51	21.1	Setting the Unit ID's .....	306
10.6	Misstep Detection function.....	51	21.2	Multi-axis mode.....	306
10.7	Self Correcting function .....	51	21.3	Multi-Drop Connection Procedure.....	306
10.8	How to recover from a deviation error .....	53	21.4	Multi-Drop Serial Communication Example .....	307
10.9	Encoder electronic gear settings.....	54	22	Timing Charts .....	308
10.10	Encoder Resolution .....	55	22.1	Execution of a Sequence .....	308
10.11	Support Functions.....	55	22.2	Stopping Operation .....	309
10.12	Protective Functions .....	58	22.3	Outputs .....	313
11	Control via RS-485 communication.....	59	22.4	Inputs.....	314
11.1	Guidance .....	59	22.5	Teaching Operation.....	315
11.2	Communication specifications .....	62	23	ASCII Data .....	316
			24	Command Format .....	317

# 1 Safety precautions

The precautions described below are intended to prevent danger or injury to the user and other personnel through safe, correct use of the product. Use the product only after carefully reading and fully understanding these instructions.

 <b>Warning</b>	Handling the product without observing the instructions that accompany a “ <b>Warning</b> ” symbol may result in serious injury or death.
 <b>Caution</b>	Handling the product without observing the instructions that accompany a “ <b>Caution</b> ” symbol may result in injury or property damage.
<b>Note</b>	The items under this heading contain important handling instructions that the user should observe to ensure safe use of the product.

## **Warning**

### General

- Do not use the product in explosive or corrosive environments, in the presence of flammable gases, locations subjected to splashing water, or near combustibles. Doing so may result in fire, electric shock or injury.
- Assign qualified personnel the task of installing, wiring, operating/controlling, inspecting and troubleshooting the product. Failure to do so may result in fire, electric shock or injury.
- The motor will lose its holding torque when the power supply or motor excitation turned OFF. If this product is used in a vertical application, be sure to provide a measure for the position retention of moving parts. Failure to provide such a measure may cause the moving parts to fall, resulting in injury or damage to the equipment.
- With certain types of alarms (protective functions), the motor may stop when the alarm generates and the holding torque will be lost as a result. This will result in injury or damage to equipment.
- When an alarm is generated, first remove the cause and then clear the alarm. Continuing the operation without removing the cause of the problem may cause malfunction of the motor and driver, leading to injury or damage to equipment.

### Connection

- Keep the driver's input-power voltage within the specified Range to avoid fire.
- For the driver's power supply, use a DC power supply with reinforced insulation on its primary and secondary sides. Failure to do so may result in electric shock.
- Connect any cables, lead wires or lead wire/connector assemblies securely according to the wiring diagram in order to prevent fire.
- Do not forcibly bend, pull or pinch the power supply cable or lead wires and motor cable or lead wires. Doing so may cause a fire. This will cause stress to the connecting section and may result in damage to equipment.

### Operation

- Turn OFF the driver power in the event of a power failure, or the motor may suddenly start when the power is restored and may cause injury or damage to equipment.
- Do not turn the motor excitation OFF while the motor is operating. The motor will stop and lose its holding ability, which may result in injury or damage to equipment.
- Configure an interlock circuit so that when a RS-485 communication error occurs, the entire system, including the driver, will operate safely.

### Repair, disassembly and modification

- Do not disassemble or modify the motor and/or driver. This may cause injury. Refer all such internal inspections and repairs to the branch or sales office from which you purchased the product.



## Caution

### General

- Do not use the motor and driver beyond its specifications, or injury or damage to equipment may result.
- Keep your fingers and objects out of the openings in the motor and driver, or fire or injury may result.
- Do not touch the motor and driver during operation or immediately after stopping. The surface is hot and may cause a skin burn(s).

### Transportation

- Do not hold the motor by the output shaft or by the motor's cable or lead wire/connector assembly. This may cause injury.

### Installation

- Install the motor and driver in an enclosure in order to prevent injury.
- Keep the area around the motor and driver free of combustible materials in order to prevent fire or a skin burn(s).
- Provide a cover over the rotating parts (output shaft) of the motor to prevent injury.

### Operation

- Use a motor and driver only in the specified combination. An incorrect combination may cause a fire.
- Provide an emergency stop device or emergency stop circuit external to the equipment so that the entire equipment will operate safely in the event of a system failure or malfunction. Failure to do so may result in injury.
- Before supplying power to the driver, turn all control inputs to the driver to OFF. Otherwise, the motor may start suddenly at power ON and cause injury or damage to equipment.
- Set the speed and acceleration/deceleration rate at reasonable levels. Otherwise, the motor will misstep and the moving part may move in an unexpected direction, resulting in injury or damage to equipment.
- Do not touch the rotating part (output shaft) during operation. This may cause injury.
- Before moving the motor directly with the hands, confirm that the power supply for the motor excitation is turned OFF and that motor current is cut off. Failure not to do so may result in injury.
- The motor surface temperature may exceed 70 °C (158 °F) even under normal operating conditions. If the operator is allowed to approach the running motor, attach a warning label as shown below in a conspicuous position. Failure to do so may result in skin burn(s).
- Immediately when trouble has occurred, stop running and turn OFF the driver power. Failure to do so may result in fire or injury.
- Static electricity may cause the driver to malfunction or suffer damage. While the driver is receiving power, do not touch the driver. Use only an insulated screwdriver to adjust the driver's switches.



Warning  
label

### Disposal

- To dispose of the motor and driver, disassemble it into parts and components as much as possible and dispose of individual parts/components as industrial waste. If you have any question, contact your nearest Oriental Motor branch or sales office

## 2 Overview of the CRK series built-in controller (Stored program)

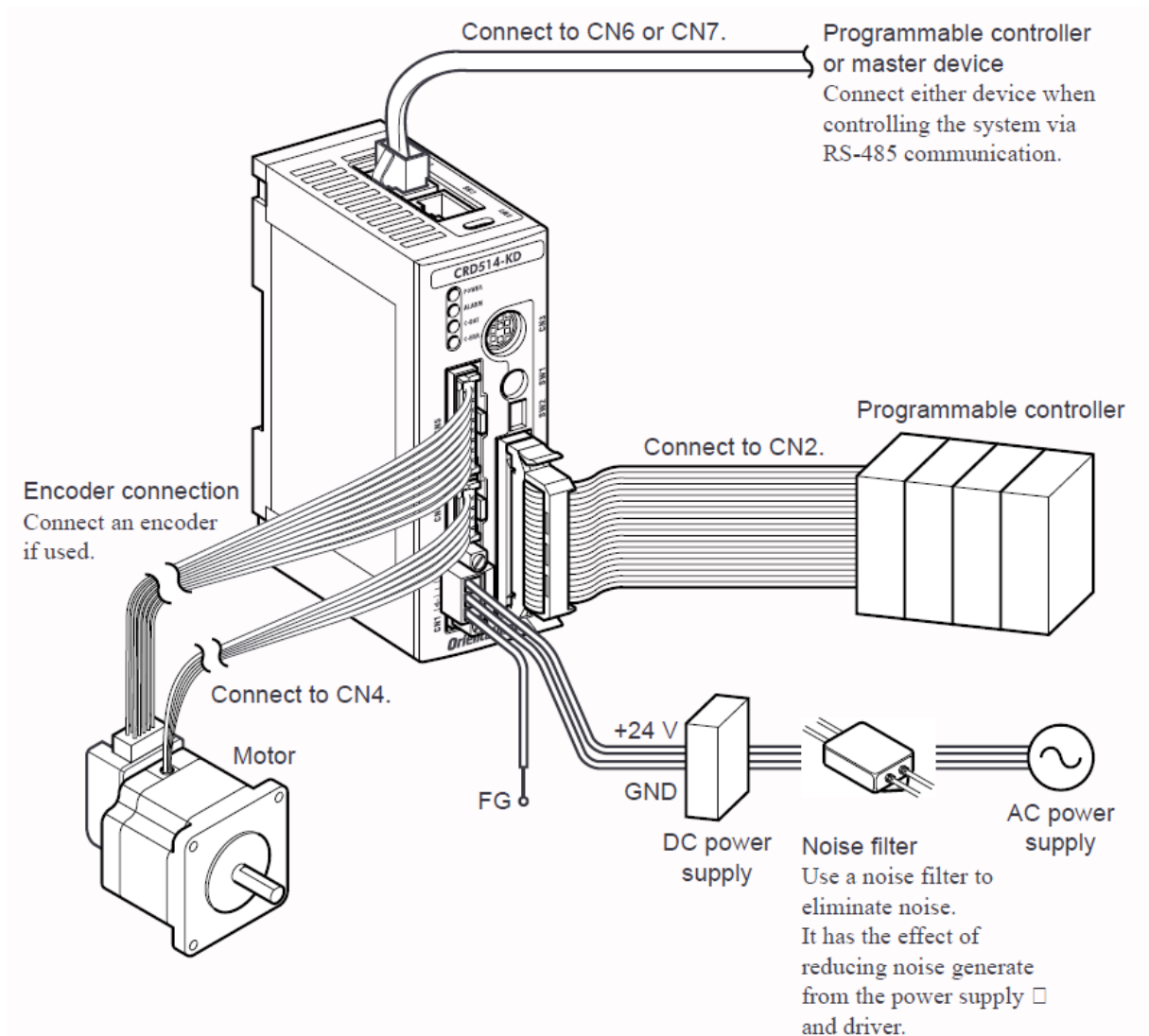
---

The CRK series built-in controller (Stored program) is a unit product consisting of a 5-phase stepping motor microstepping driver with built-in controller functions and a 5-phase stepping motor offering high torque with low vibration. The driver supports RS-485 communication, stand alone operation and I/O control. Operating data, parameters and stored programs can be set using RS-485 communication.

### ■ Main features

- Three operating patterns  
You can perform positioning operations, return-to-home operations and continuous operations. Up to 64 programs can be stored, and Linked motions are also possible.
- Low vibration, low noise  
The micro-step driver with smooth drive function achieves low vibration and low noise.
- Supporting RS-485 communication  
You can set operation data and parameters or issue operation start/stop commands from the master station. Up to 16 drivers can be connected to one master.
- Self Correction Function  
If a misstep condition occurs, the driver can automatically correct itself in order to ensure that the correct final position is reached and a SC (self correction has occurred) output signal will be output.
- Detection of misstep  
If the deviation between the encoder counter value and driver command position reaches or exceeds the set value, a STEP-OUT output signal will be output.
- Alarm and warning functions  
The driver provides alarms that are designed to protect the driver from overheating, poor connection, incorrect operation, etc. (protective functions), as well as warnings that are output before the corresponding alarms generate (warning functions).
- CRK Motion Creator GUI  
If you install the exclusive GUI tool, CRK Motion Creator, to your computer, just clicking your mouse can create motion, perform system configuration, write programs and upload/download programs and parameters easily. Of course, for the person who prefers to use a keyboard and the programming language; the device can be programmed via any terminal software on a PC, such as HyperTerminal. However, the CRK Motion Creator will greatly help you to save and load data between a PC and the CRK series built-in controller (Stored program). The CRK Motion Creator includes a motion creating function, sequence editing function, terminal function, data save/load function, and system parameter setting function. The latest version of the CRK Motion Creator program is available for download free at <http://www.orientalmotor.com/support/software/>

### 3 System configuration



# 4 Introduction

## ■ Before use

Only qualified personnel should work with the product.

Use the product correctly after thoroughly reading the section “1 Safety precautions” on p.3.

The product described in this manual has been designed and manufactured for use in general industrial machinery, and must not be used for any other purpose. For the driver's power supply, use a DC power supply with reinforced insulation on its primary and secondary sides.

Oriental Motor Co., Ltd. is not responsible for any damage caused through failure to observe this warning.

## ■ Structure of the manual

The CRK series built-in controller (Stored program) comes with the manuals specified below.

- CRK Series Built-in Controller (Stored Program) OPERATING MANUAL

This manual explains the product functions as well as how to install/connect and operate the product, among others.

- CRK Series Built-in Controller (Stored Program) Information Manual

This manual explains the safety precautions, connector pin assign and others.

After reading the above manuals, keep them in a convenient place so that you can reference them at any time.

## ■ CE Marking

Because the input power supply voltage of this product is 24 VDC, it is not subject to the Low Voltage Directive (LVD). However, install and connect this product as follows.

- The product is a type with machinery incorporated, so it should be installed within an enclosure.
- For the driver's power supply, use a DC power supply with reinforced insulation on its primary and secondary sides.
- Overvoltage category: I
- Pollution degree: Class 2
- Degree of protection:

Motor	<ul style="list-style-type: none"> <li>▪ High-resolution type</li> <li>▪ High-torque type</li> <li>▪ High-torque type with encoder</li> <li>▪ <b>TH geared type (CRK513P and CRK523P)</b></li> <li>▪ <b>PS geared type (CRK523P)</b></li> <li>▪ Standard type</li> <li>▪ Standard type with encoder</li> </ul>	IP20
	<ul style="list-style-type: none"> <li>▪ <b>TH geared type (CRK543, CRK544, CRK564 and CRK566)</b></li> <li>▪ <b>PS geared type (CRK543, CRK544, CRK564 and CRK566)</b></li> </ul>	IP30
Driver	IP20	

- EMC Directive

This product has received EMC measures under the conditions specified below.

Be sure to conduct EMC measures with the product assembled in your equipment by referring to “Installing and wiring in compliance with EMC Directive” on p.23

### Note

EMC directive was not tested for products with encoder attached. Please perform testing prior to use to ensure conformity.

Applicable standards

EMI	EN 61000-6-4:2007+A1:2011
	EN 55011:2009+A1:2010
EMS	EN 61000-6-2:2005

## ■ Hazardous substances

RoHS (Directive 2002/95/EC 27Jan.2003) compliant

## 5 Precautions for use

---

This section covers limitations and requirements the user should consider when using the product.

- Conduct the insulation resistance measurement or withstand voltage test separately on the motor and the driver.  
Conducting the insulation resistance measurement or withstand voltage test with the motor and driver connected may result in injury or damage to equipment.
- Do not apply a strong impact on the motor output shaft.  
If you are using a motor with encoder, an optical encoder is housed in the motor. To prevent damage to the encoder, handle the motor with care and avoid strong impact to the motor output shaft when transporting the motor or installing the load.
- Do not apply an overhung load and/or thrust load in excess of the specified permissible limit  
Operating it under an excessive overhung load and/or thrust load may damage the motor bearings (ball bearings). Be sure to operate the motor within the specified permissible limit of overhung load and thrust load. See page 19 for details.
- Motor case temperature
  - The motor case surface temperature may exceed 100 °C (212 °F) under certain conditions (ambient temperature, operating speed, duty cycle, etc.). Keeping the surface temperature of the motor case below 100 °C (212 °F) will also maximize the life of the motor bearings (ball bearings).
  - Use the motor with encoder in a condition where the encoder case temperature does not exceed 80 °C (176 °F).
- Operate the motor with a surface temperature not exceeding 100 °C (212 °F)  
The motor case's surface temperature may exceed 100 °C (212 °F) under certain conditions (ambient temperature, operating speed, duty cycle, etc.). Keeping the surface temperature of the motor case below 100 °C (212 °F) will also maximize the life of the motor bearings (ball bearings).
- Maximum static torque at excitation  
Maximum static torque at excitation represents a value obtained when the motor is excited using the rated current. When the motor is combined with a dedicated driver, the maximum static torque at excitation drops to approximately 50% (factory setting), due to the current down function, which suppresses the rise in motor temperature in a standstill state. Acceleration and operation of the motor at the maximum static torque at excitation is possible during start-up, but it only has approximately 50% holding power after the motor has stopped. When selecting a motor for your application, consider the fact that the holding power will be reduced to approximately 50% after the motor has stopped.
- Preventing electrical noise  
See "Installing and wiring in compliance with EMC Directive" on p. 23 for suggested measures with regard to reducing electrical noise.
- Regeneration  
The overvoltage alarm will generate depending on the operating condition. When an alarm is generated, review the operating conditions.
- EEPROM Write cycle  
Do not turn OFF the main power supply while data is being written to the EEPROM and 5 seconds after the completion of a data write. Doing so may abort the data write and cause an EEPROM error alarm to generate. The EEPROM can be rewritten approx. 100,000 times.



- Geared type

The relationship between the rotating direction of the motor shaft and that of the gear output shaft changes as follows, depending on the gear type and gear ratio.

Type of Gear	Gear ratio	Rotation direction (relative to the motor shaft direction)		
		Frame size [in. □ (mm □)]		
		0.98 (28)	1.65 (42)	2.36 (60)
TH geared	3.6:1	Opposite direction	Same direction	
	7.2:1			
	10:1			
	20:1	Same direction	Opposite direction	
	30:1			
PS geared	All ratios	Same direction		
Harmonic geared		Opposite direction		

#### Grease of geared motor

On rare occasions, a small amount of grease may ooze out from the geared motor. If there is concern over possible environmental damage resulting from the leakage of grease, check for grease stains during regular inspections. Alternatively, install an oil pen or other device to prevent leakage from causing further damage. Oil leakage may lead to problems in the customer's equipment or products.

# 6 Preparation

---

This chapter explains the items you should check, as well as the name and function of each part.

## 6.1 Checking the product

Verify that the items listed below are included. Report any missing or damaged items to the branch or sales office from which you purchased the product.

Verify the model number of the purchased unit against the number shown on the package label.

Check the model number of the motor and driver against the number shown on the nameplate.

The unit models and corresponding motor/driver combinations are listed on p.11.

When purchasing a unit model (motor and driver) product:

- Motor.....1 pc.
- Driver.....1 pc.
- CN1 power supply connector (3 terminals).....1 pc.
- CN2 I/O ribbon cable/connector assembly [1 m (3.3 ft.)].....1 pc.
- CN4 motor lead wire/connector assembly [0.6 m (2 ft.)].....1 pc.
- CN5 encoder lead wire/connector assembly [0.6 m (2 ft.)].....1 pc.  
(Encoder motor/driver models only)
- Information Manual.....1 copy
- Additional Items supplied with connector-type motor units  
Applicable products: High-resolution type, high-torque type, high-torque type with encoder, **TH** geared type, **PS** geared type, **Harmonic** geared type
  - Motor lead wire/connector assembly [0.6 m (2 ft.)].....1 pc.
- Additional Items supplied with motor units with encoder  
Applicable products: High-torque type with encoder (**CRK54**□**PRKD**)  
Standard type with encoder (**CRK54**□**RKD**, **CRK56**□**RKD**)
  - Encoder motor lead wire/connector assembly [0.6 m (2 ft.)] ..... 1 pc.

When purchasing a driver only product (for maintenance, replacement, etc):

- Driver.....1 pc.
- CN1 power supply connector (3 terminals).....1 pc.
- CN2 I/O ribbon cable/connector assembly [1 m (3.3 ft.)].....1 pc.
- CN4 motor lead wire/connector assembly [0.6 m (2 ft.)].....1 pc.
- Information Manual.....1 copy

## 6.2 Combinations of motors and drivers

### ■ Standard type

Frame Size [mm (in.)]	Unit model		Motor model		Driver model
	Single shaft	Double shaft	Single shaft	Double shaft	
□42 (1.65)	CRK543AKP	CRK543BKP	PK543NAW	PK543NBW	CRD507-KP
	CRK544AKP	CRK544BKP	PK544NAW	PK544NBW	
	CRK545AKP	CRK545BKP	PK545NAW	PK545NBW	
□60 (2.36)	CRK564AKP	CRK564BKP	PK564NAW	PK564NBW	CRD514-KP
	CRK566AKP	CRK566BKP	PK566NAW	PK566NBW	
	CRK569AKP	CRK569BKP	PK569NAW	PK569NBW	

### ■ Standard type with encoder (500 lines/rev, 3 channel, Line driver output)

Frame Size [mm (in.)]	Unit model		Motor model		Driver model
	Single shaft	Double shaft	Single shaft	Double shaft	
□42 (1.65)	CRK543RKP	-	PK543NAW-R27L	-	CRD507-KP
	CRK544RKP	-	PK544NAW-R27L	-	
	CRK545RKP	-	PK545NAW-R27L	-	
□60 (2.36)	CRK564RKP	-	PK564NAW-R27L	-	CRD514-KP
	CRK566RKP	-	PK566NAW-R27L	-	
	CRK569RKP	-	PK569NAW-R27L	-	

### ■ High Resolution type

Frame Size [mm (in.)]	Unit model		Motor model		Driver model
	Single shaft	Double shaft	Single shaft	Double shaft	
□28 (1.10)	CRK523PMAKP	CRK523PMBKP	PK523PMA	PK523PMB	CRD503-KP
	CRK524PMAKP	CRK524PMBKP	PK524PMA	PK524PMB	
	CRK525PMAKP	CRK525PMBKP	PK525PMA	PK525PMB	
□42 (1.65)	CRK544PMAKP	CRK544PMBKP	PK544PMA	PK544PMB	CRD507-KP
	CRK546PMAKP	CRK546PMBKP	PK546PMA	PK546PMB	
□60 (2.36)	CRK564PMAKP	CRK564PMBKP	PK564PMA	PK564PMB	CRD514-KP
	CRK566PMAKP	CRK566PMBKP	PK566PMA	PK566PMB	
	CRK569PMAKP	CRK569PMBKP	PK569PMA	PK569PMB	

### ■ High Resolution type with encoder (1000 lines/rev, 3 channel, Line driver output)

Frame Size [mm (in.)]	Unit model		Motor model		Driver model
	Single shaft	Double shaft	Single shaft	Double shaft	
□42 (1.65)	CRK544PMRKP	-	PK544PMA-R28L	-	CRD507-KP
	CRK546PMRKP	-	PK546PMA-R28L	-	
□60 (2.36)	CRK564PMRKP	-	PK564PMA-R28L	-	CRD514-KP
	CRK566PMRKP	-	PK566PMA-R28L	-	
	CRK569PMRKP	-	PK569PMA-R28L	-	

### ■ High Torque type

Frame Size [mm (in.)]	Unit model		Motor model		Driver model
	Single shaft	Double shaft	Single shaft	Double shaft	
□20 (0.79)	CRK513PAKP	CRK513PBKP	PK513PA	PK513PB	CRD503-KP
□28 (1.10)	CRK523PAKP	CRK523PBKP	PK523PA	PK523PB	
	CRK525AKP	CRK525BKP	PK525AW	PK525BW	
□42 (1.65)	CRK544PAKP	CRK544PBKP	PK544PA	PK544PB	CRD507-KP
	CRK546PAKP	CRK546PBKP	PK546PA	PK546PB	

### ■ High Torque type with encoder (500 lines/rev, 3 channel, Line driver output)

Frame Size [mm (in.)]	Unit model		Motor model		Driver model
	Single shaft	Double shaft	Single shaft	Double shaft	
□42 (1.65)	CRK544PRKP	-	PK544PA-R27L	-	CRD507-KP
	CRK546PRKP	-	PK546PA-R27L	-	

### ■ TH geared type

Frame Size [mm (in.)]	Unit model		Motor model		Driver model
	Single shaft	Double shaft	Single shaft	Double shaft	
□28 (1.10)	CRK523PAKP-T7.2	CRK523PBKP-T7.2	PK523PA-T7.2	PK523PB-T7.2	CRK503-KP
	CRK523PAKP-T10	CRK523PBKP-T10	PK523PA-T10	PK523PB-T10	
	CRK523PAKP-T20	CRK523PBKP-T20	PK523PA-T20	PK523PB-T20	
	CRK523PAKP-T30	CRK523PBKP-T30	PK523PA-T30	PK523PB-T30	
□42 (1.65)	CRK543AKP-T3.6	CRK543BKP-T3.6	PK543AW-T3.6	PK543BW-T3.6	CRD507-KP
	CRK543AKP-T7.2	CRK543BKP-T7.2	PK543AW-T7.2	PK543BW-T7.2	
	CRK543AKP-T10	CRK543BKP-T10	PK543AW-T10	PK543BW-T10	
	CRK543AKP-T20	CRK543BKP-T20	PK543AW-T20	PK543BW-T20	
	CRK543AKP-T30	CRK543BKP-T30	PK543AW-T30	PK543BW-T30	
□60 (2.36)	CRK564AKP-T3.6	CRK564BKP-T3.6	PK564AW-T3.6	PK564BW-T3.6	CRD514-KP
	CRK564AKP-T7.2	CRK564BKP-T7.2	PK564AW-T7.2	PK564BW-T7.2	
	CRK564AKP-T10	CRK564BKP-T10	PK564AW-T10	PK564BW-T10	
	CRK564AKP-T20	CRK564BKP-T20	PK564AW-T20	PK564BW-T20	
	CRK564AKP-T30	CRK564BKP-T30	PK564AW-T30	PK564BW-T30	

### ■ TH geared type with encoder (500 lines/rev, 3 channel, Line driver output)

Frame Size [mm (in.)]	Unit model		Motor model		Driver model
	Single shaft	Double shaft	Single shaft	Double shaft	
□42 (1.65)	CRK543RKPT3.6	-	PK543AWR27LT3.6	-	CRD507-KP
	CRK543RKPT7.2	-	PK543AWR27LT7.2	-	
	CRK543RKPT10	-	PK543AWR27LT10	-	
	CRK543RKPT20	-	PK543AWR27LT20	-	
	CRK543RKPT30	-	PK543AWR27LT30	-	
□60 (2.36)	CRK564RKPT3.6	-	PK564AWR27LT3.6	-	CRD514-KP
	CRK564RKPT7.2	-	PK564AWR27LT7.2	-	
	CRK564RKPT10	-	PK564AWR27LT10	-	
	CRK564RKPT20	-	PK564AWR27LT20	-	
	CRK564RKPT30	-	PK564AWR27LT30	-	

## ■ PS geared type

Frame Size [mm (in.)]	Unit model		Motor model		Driver model
	Single shaft	Double shaft	Single shaft	Double shaft	
□28 (1.10)	CRK523PAKP-PS5	CRK523PBKP-PS5	PK523PA-PS5	PK523PB-PS5	CRK503-KP
	CRK523PAKP-PS7	CRK523PBKP-PS7	PK523PA-PS7	PK523PB-PS7	
	CRK523PAKP-PS10	CRK523PBKP-PS10	PK523PA-PS10	PK523PB-PS10	
□42 (1.65)	CRK543AKP-PS25	CRK543BKP-PS25	PK543AW-PS25	PK543BW-PS25	CRD507-KP
	CRK543AKP-PS36	CRK543BKP-PS36	PK543AW-PS36	PK543BW-PS36	
	CRK543AKP-PS50	CRK543BKP-PS50	PK543AW-PS50	PK543BW-PS50	
	CRK545AKP-PS5	CRK545BKP-PS5	PK545AW-PS5	PK545BW-PS5	
	CRK545AKP-PS7	CRK545BKP-PS7	PK545AW-PS7	PK545BW-PS7	
	CRK545AKP-PS10	CRK545BKP-PS10	PK545AW-PS10	PK545AW-PS10	
□60 (2.36)	CRK564AKP-PS25	CRK564BKP-PS25	PK564AW-PS25	PK564BW-PS25	CRD514-KP
	CRK564AKP-PS36	CRK564BKP-PS36	PK564AW-PS36	PK564BW-PS36	
	CRK564AKP-PS50	CRK564BKP-PS50	PK564AW-PS50	PK564BW-PS50	
	CRK566AKP-PS5	CRK566BKP-PS5	PK566AW-PS5	PK566BW-PS5	
	CRK566AKP-PS7	CRK566BKP-PS7	PK566AW-PS7	PK566BW-PS7	
	CRK566AKP-PS10	CRK566BKP-PS10	PK566AW-PS10	PK566BW-PS10	

## ■ PS geared type with encoder (500 lines/rev, 3 channel, Line driver output)

Frame Size [mm (in.)]	Unit model		Motor model		Driver model
	Single shaft	Double shaft	Single shaft	Double shaft	
□42 (1.65)	CRK543RKPPS25	-	PK543AWR27LPS25	-	CRD507-KP
	CRK543RKPPS36	-	PK543AWR27LPS36	-	
	CRK543RKPPS50	-	PK543AWR27LPS50	-	
	CRK545RKPPS5	-	PK545AWR27LPS5	-	
	CRK545RKPPS7	-	PK545AWR27LPS7	-	
	CRK545RKPPS10	-	PK545AWR27LPS10	-	
□60 (2.36)	CRK564RKPPS25	-	PK564AWR27LPS25	-	CRD514-KP
	CRK564RKPPS36	-	PK564AWR27LPS36	-	
	CRK564RKPPS50	-	PK564AWR27LPS50	-	
	CRK566RKPPS5	-	PK566AWR27LPS5	-	
	CRK566RKPPS7	-	PK566AWR27LPS7	-	
	CRK566RKPPS10	-	PK566AWR27LPS10	-	

## ■ Harmonic geared type

Frame Size [mm (in.)]	Unit model		Motor model		Driver model
	Single shaft	Double shaft	Single shaft	Double shaft	
□20 (0.79)	CRK513PAKP-H50	CRK513PBKP-H50	PK513PA-H50S	PK513PB-H50S	CRK503-KP
	CRK513PAKP-H100	CRK513PBKP-H100	PK513PA-H100S	PK513PB-H100S	
□30 (1.18)	CRK523PAKP-H50	CRK523PBKP-H50	PK523HPA-H50S	PK523HPB-H50S	CRK507H-KP
	CRK523PAKP-H100	CRK523PBKP-H100	PK523HPA-H100S	PK523HPB-H100S	
□42 (1.65)	CRK543AKP-H50	CRK543BKP-H50	PK543AW-H50S	PK543BW-H50S	CRD507-KP
	CRK543AKP-H100	CRK543BKP-H100	PK543AW-H100S	PK543BW-H100S	
□60 (2.36)	CRK564AKP-H50	CRK564BKP-H50	PK564AW-H50S	PK564BW-H50S	CRD514-KP
	CRK564AKP-H100	CRK564BKP-H100	PK564AW-H100S	PK564BW-H100S	

## ■ Harmonic geared type with encoder (500 lines/rev, 3 channel, Line driver output)

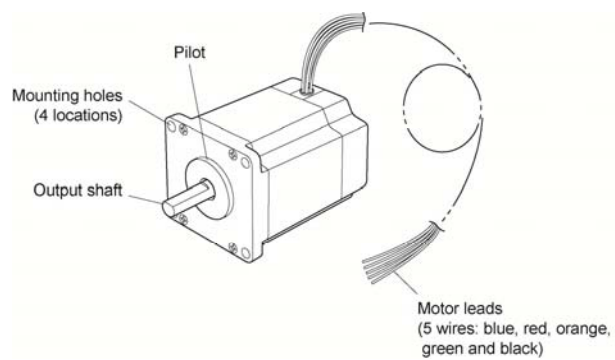
Frame Size [mm (in.)]	Unit model		Motor model		Driver model
	Single shaft	Double shaft	Single shaft	Double shaft	
□42 (1.65)	CRK543RKPH50	-	PK543AWR27LH50	-	CRD507-KP
	CRK543RKPH100	-	PK543AWR27LH100	-	
□60 (2.36)	CRK564RKPH50	-	PK564AWR27LH50	-	CRD514-KP
	CRK564RKPH100	-	PK564AWR27LH100	-	

## 6.3 Names and functions of parts

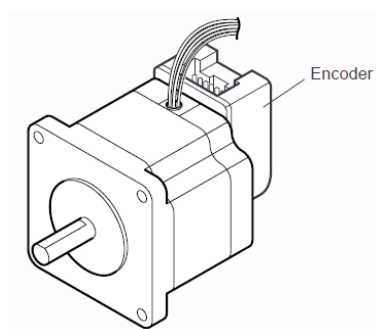
### ■ Motor

Illustration shows the PK56□ type.

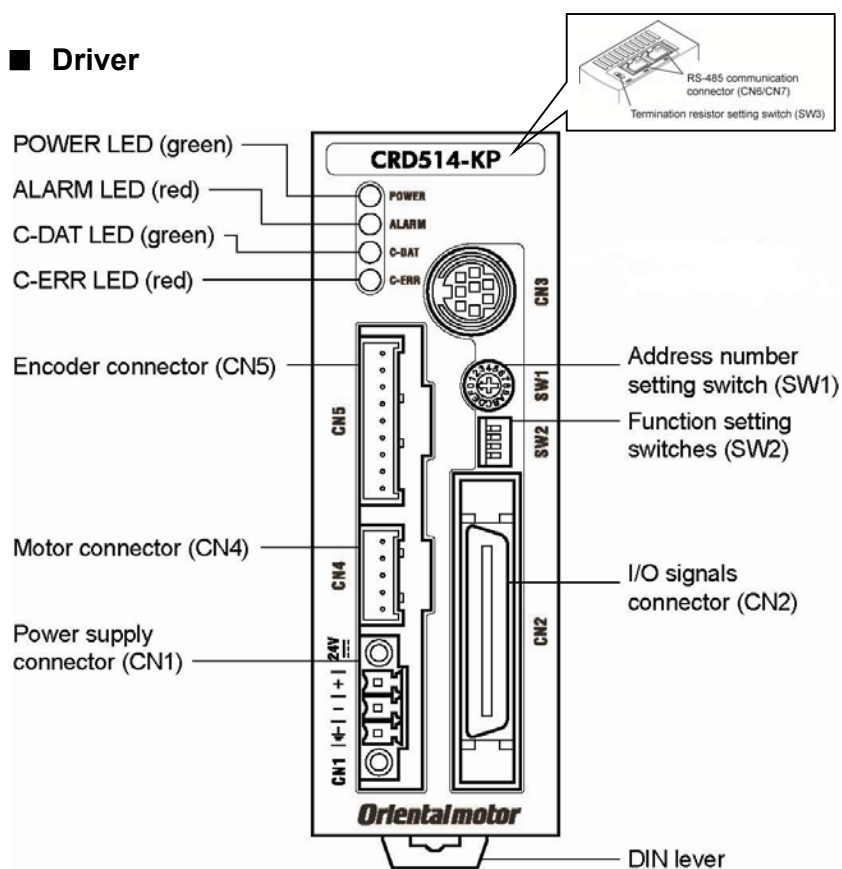
#### ▪ Standard type



#### ▪ Standard type with encoder



## ■ Driver



Name	Description	Reference
POWER LED (green)	This LED is lit while the main power is input.	—
ALARM LED (red)	This LED will blink when an alarm generates (a protective function is triggered). You can check the generated alarm by counting the number of times the LED blinks.	P.294
C-DAT LED (green)	This LED will blink or illuminate steadily when the driver is communicating with the master station properly via RS-485 communication.	—
C-ERR LED (red)	This LED will illuminate when a RS-485 communication error occurs with the master station.	—
Address number setting switch (SW1)	Use this switch when controlling the system via RS-485 communication. Set the address number of RS-485 communication.	P.63
Function setting switches (SW2)	Use this switches when controlling the system via RS-485 communication. No.1 to 3: Used to set the baud rate of RS-485 communication. No.4: Used to set device to single or multi-axis mode	P.64
Termination resistor setting switch (SW3)	Use this switch when controlling the system via RS-485 communication. Set the termination resistor (120 Ω) of RS-485 communication.	P.65
Power supply connector (CN1)	Connection for main power supply (+24 VDC) using the supplied connector.	P.27
I/O signals connector (CN2)	Connection for the I/O signals using the supplied connector cable.	P.28
Connector (CN3)	Not used	
Motor connector (CN4)	Connection for the motor.	P.25
Encoder connector (CN5)	Connection for the encoder.	P.31
RS-485 communication connectors (CN6/CN7)	Connection for the RS-485 communication cable.	P.32

# 7 Installation

This chapter explains the installation location and installation method of the motor and driver. Also covered in this section are the installation and wiring methods that are in compliance with the relevant EMC Directives.

## 7.1 Location for installation

The driver is designed and manufactured for installation in equipment.

Install it in a well-ventilated location that provides easy access for inspection. The location must also satisfy the following conditions:

- Inside an enclosure that is installed indoors (provide vent holes)
- Operating ambient temperature Motor:  $-10$  to  $+50$  °C ( $+14$  to  $+122$  °F) (non-freezing)  
Driver:  $0$  to  $+40$  °C ( $+32$  to  $+104$  °F) (non-freezing)
- Operating ambient humidity 85% or less (non-condensing)
- Area that is free of explosive atmosphere or toxic gas (such as sulfuric gas) or liquid
- Area not exposed to direct sun
- Area free of excessive amount of dust, iron particles or the like
- Area not subject to splashing water (rain, water droplets), oil (oil droplets) or other liquids
- Area free of excessive salt
- Area not subject to continuous vibration or excessive shocks
- Area free of excessive electromagnetic noise (from welders, power machinery, etc.)
- Area free of radioactive materials, magnetic fields or vacuum

## 7.2 Installing the motor

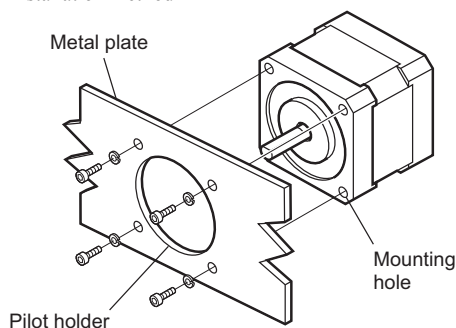
The motor can be installed in any direction.

Install the motor onto an appropriate flat metal plate having excellent vibration resistance and heat conductivity.

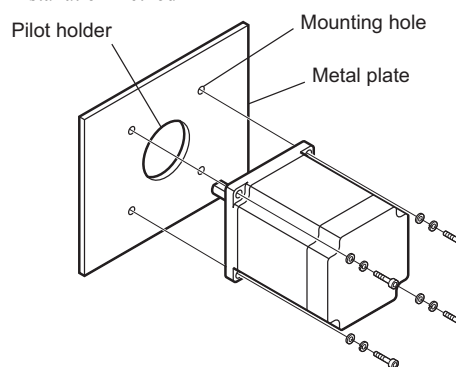
When installing the motor, secure it with four bolts (not supplied) through the four mounting holes. Do not leave a gap between the motor and metal plate.

**Note** Insert the pilot located on the motor's installation surface into the mounting plates.

Installation method A



Installation method B





## Screw size, tightening torque and installation method

Frame Size [mm (in.)]	Motor type	Motor model*		Nominal size	Tightening torque N·m (oz-in)]	Effective depth of bolt [mm (in.)]	Installation method
		Single shaft	Double shaft				
□20 (0.79)	High-torque type	PK513PA	PK513PB	M2	0.25 (35.4)	2.5 (0.098)	A
	<b>Harmonic</b> geared type	PK513PA-H□S	PK513PB-H□S	M2	0.25 (35.4)	5 (0.197)	
□28 (1.10)	High-resolution type	PK523PMA PK524PMA PK525PMA	PK523PMB PK524PMB PK525PMB	M2.5	0.5 (70.8)	2.5 (0.098)	A
	High-torque type	PK523PA PK525PA	PK523PB PK525PB				
	<b>TH</b> geared type	PK523PA-T□	PK523PB-T□			4 (0.15)	
	<b>PS</b> geared type	PK523PA-PS□	PK523PB-PS□	M3	1 (142)	6 (0.236)	
□30 (1.18)	<b>Harmonic</b> geared type	PK523HPA-H□S	PK523HPB-H□S	M3	1 (142)	6 (0.236)	
□42 (1.65)	High-resolution type	PK544PMA PK546PMA	PK544PMB PK546PMB	M3	1 (142)	4.5 (0.177)	A
	High-resolution type with encoder	PK544PMA-R28L PK546PMA-R28L	-				
	High-torque type	PK544PA PK546PA	PK544PB PK546PB				
	High-torque type with encoder	PK544PA-R27L PK546PA-R27L	-				
	Standard type with encoder	PK543NAW-R27L PK544NAW-R27L PK545NAW-R27L	-				
	Standard type	PK543NAW PK544NAW PK545NAW	PK543NBW PK544NBW PK545NBW	M4	2 (280)	8 (0.315)	
	<b>TH</b> geared type	PK543AW-T□	PK543BW-T□				
	<b>PS</b> geared type	PK543AW-PS□ PK545AW-PS□	PK543BW-PS□ PK545BW-PS□				
	<b>Harmonic</b> geared type	PK543AW-H□S	PK543BW-H□S				
□60 (2.36)	High-resolution type	PK564PMA PK566PMA PK564PMA	PK564PMB PK566PMB PK564PMB	M4	2 (280)	-	B
	High-resolution type with encoder	PK564PMA-R28L PK566PMA-R28L PK564PMA-R28L	-				
	Standard-type with encoder	PK564NAW-R27L PK566NAW-R27L PK569NAW-R27L	-				
	Standard-type	PK564NAW PK566NAW PK569NAW	PK564NAW PK566NAW PK569NAW				
	<b>TH</b> geared type	PK564AW-T□	PK564BW-T□	M5	2.5 (350)	8 (0.315)	A
	<b>PS</b> geared type	PK564AW-PS□ PK566AW-PS□	PK564BW-PS□ PK566BW-PS□				
	<b>Harmonic</b> geared type	PK564AW-H□S	PK564BW-H□S				

\* A □ within the model name represents the gear ratio

## 7.3 Installing a load

When connecting a load to the motor, align the centers of the motor's output shaft and load shaft. Flexible couplings are available as accessories.

### Note

- When coupling the load to the motor, pay attention to the centering of the shafts, belt tension, parallelism of the pulleys, and so on. Securely tighten the coupling and pulley set screws.
- Be careful not to damage the output shaft or bearings (ball bearing) when installing a coupling or pulley to the motor's output shaft.
- Do not modify or machine the motor's output shaft. Doing so may damage the bearings and destroy the motor.
- If you are using a motor with an encoder, an optical encoder is connected to the motor. To prevent damage to the encoder, handle the motor with care and avoid any strong impacts to the motor output shaft when transporting or installing the motor.

#### • Using a coupling

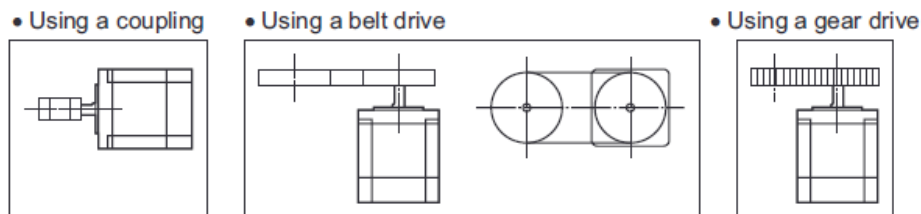
Align the centers of the motor's output shaft and load shaft in a straight line.

#### • Using a belt drive

Align the motor's output shaft and load shaft in parallel with each other, and position both pulleys so that the line connecting their centers is at a right angle to the shafts.

#### • Using a gear drive

Align the motor's output shaft and gear shaft in parallel with each other, and let the gears mesh at the center of the tooth widths.

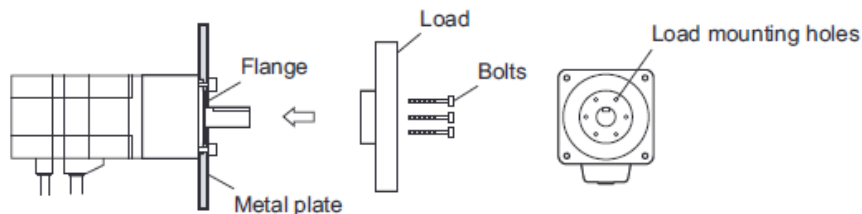


#### • Using a parallel key (geared motor)

Connect a load to the gear output shaft having a key groove, first provide a key groove on the load and fix the load with the gear output shaft using the supplied key.

#### • Installing in the flange surface (Harmonic geared type)

With a harmonic geared type, a load can be installed directly to the gear using the load mounting holes provided on the flange surface.

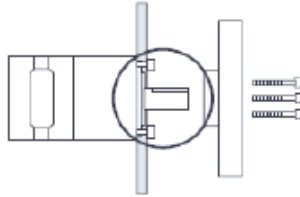


Motor model	Nominal size	Number of bolts	Tightening torque [N·m (oz-in)]	Effective depth of bolt [mm (in.)]
PK513-H□S	M2	3	0.35 (49)	3 (0.118)
PK523-H□S	M3	4	1.4 (198)	4 (0.157)
PK543-H□S	M3	6	1.4 (198)	5 (0.2)
PK564-H□S	M4	6	2.5 (350)	6 (0.24)

\* A □ within the model name represents the gear ratio

**Note**

- When installing a load on the flange surface, the load cannot be affixed using the key groove in the output shaft.
- Design an appropriate installation layout so that the load will not contact the metal plate or bolts used for installing the motor.



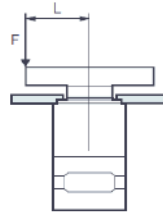
- Permissible moment load of the harmonic geared type

When installing an arm or table on the flange surface, calculate the moment load using the formula below if the flange surface receives any eccentric load. The moment load should not exceed the permissible value specified in the table.

Moment load:  $M \text{ [N-m (oz-in)]} = F \times L$

Motor type	Permissible moment load [N-m (oz-in)]
PK513-H□S	0.7 (99)
PK523-H□S	2.9 (410)
PK543-H□S	5.6 (790)
PK564-H□S	11.6 (1640)

\* A □ within the model name represents the gear ratio



## 7.4 Permissible overhung load and permissible thrust load

The overhung load and the thrust load on the motor's output shaft must be kept under the permissible values listed below.

**Note**

Failure due to fatigue may occur when the motor bearings and output shaft are subject to repeated loading by an overhung or thrust load that is in excess of the permissible limit.

Frame Size [mm (in.)]	Motor type	Motor model* <sup>1</sup>		Permissible overhung load [N (lb.)]					Permissible thrust load [N (lb.)]
				Distance from tip of motor shaft [mm (in.)]					
		Single shaft	Double shaft	0 (0)	5 (0.20)	10 (0.39)	15 (0.59)	20 (0.79)	
□20 (0.79)	High-torque type	PK513PA	PK513PB	12 (2.69)	15 (3.3)	-	-	-	0.05 (0.11)* <sup>2</sup>
	<b>Harmonic</b> geared type	PK513PA-H□	PK513PB-H□	50 (11.2)	75 (16.8)	-	-	-	60 (13.5)
□28 (1.10)	High-resolution type	PK523PMA	PK523PMB	25 (5.6)	34 (7.6)	52 (11.7)	-	-	0.11 (0.24)* <sup>2</sup>
		PK524PMA	PK524PMB						0.15 (0.33)* <sup>2</sup>
		PK525PMA	PK525PMB						0.2 (0.44)* <sup>2</sup>
	High-torque type	PK523PA	PK523PB	25 (5.6)	34 (7.6)	52 (11.7)	-	-	0.11 (0.24)* <sup>2</sup>
		PK525PA	PK525PB						0.2 (0.44)* <sup>2</sup>
	<b>TH</b> geared type	PK523PA-T□	PK523PB-T□	15 (3.3)	17 (3.8)	20 (4.5)	23 (5.2)	-	10 (2.2)
	<b>PS</b> geared type	PK523PA-PS□	PK523PB-PS□	45 (10.1)	60 (13.5)	80 (18)	100 (22)	-	20 (4.5)
□30 (1.18)	<b>Harmonic</b> geared type	PK523HPA-H□S	PK523HPB-H□S	110 (24)	135 (30)	175 (39)	250 (56)	-	140 (31)
□42 (1.65)	High-resolution type	PK544PMA	PK544PMB	20 (4.5)	25 (5.6)	34 (7.6)	52 (11.7)	-	0.3 (0.66)* <sup>2</sup>
		PK546PMA	PK546PMB						0.5 (0.11)* <sup>2</sup>
	High-resolution type with encoder	PK544PMA-R28L	-						0.3 (0.66)* <sup>2</sup>
		PK546PMA-R28L	-						0.5 (1.1)* <sup>2</sup>
	High-torque type	PK544PA	PK544PB						0.3 (0.66)* <sup>2</sup>
		PK546PA	PK546PB						0.5 (1.1)* <sup>2</sup>
	High-torque type with encoder	PK544PA-R27L	-						0.36 (0.79)* <sup>2</sup>
		PK546PA-R27L	-						0.56 (1.23)* <sup>2</sup>
	Standard type with encoder	PK543NAW-R27L	-						0.31 (0.68)* <sup>2</sup>
		PK544NAW-R27L							0.36 (0.79)* <sup>2</sup>
		PK545NAW-R27L							0.46 (1.01)* <sup>2</sup>
	Standard type	PK543NAW	PK543NBW PK544NBW PK545NBW						0.25 (0.55)* <sup>2</sup>
		PK544NAW							0.3 (0.66)* <sup>2</sup>
		PK545NAW							0.4 (0.88)* <sup>2</sup>
	<b>TH</b> geared type	PK543AW-T□	PK543BW-T□	10 (2.2)	14 (3.1)	20 (4.5)	30 (6.7)	-	15 (3.3)
<b>PS</b> geared type	PK543AW-PS□	PK543BW-PS□	109 (24)	127 (28)	150 (33)	184 (41)	-	50 (11.2)	
	PK545AW-PS□	PK545BW-PS□	73 (16.4)	84 (18.9)	100 (22)	123 (27)	-		
<b>Harmonic</b> geared type	PK543AW-H□S	PK543BW-H□S	180 (40)	220 (49)	270 (60)	360 (81)	510 (114)	220 (49)	

\*<sup>1</sup> □ within the model name represents the gear ratio.

\*<sup>2</sup> Indicates the motor mass [kg (lb)]. The thrust load should not exceed the motor's mass.

Frame Size [mm (in.)]	Motor type	Motor model* <sup>1</sup>		Permissible overhung load [N (lb.)]					Permissible thrust load [N (lb.)]
				Distance from tip of motor shaft [mm (in.)]					
		Single shaft	Double shaft	0 (0)	5 (0.20)	10 (0.39)	15 (0.59)	20 (0.79)	
□60 (2.36)	High-resolution type	PK564PMA	PK564PMB	90 (20.1)	100 (22)	130 (29.1)	180 (40.3)	270 (60.4)	0.65 (1.43)* <sup>2</sup>
		PK566PMA	PK566PMB						0.87 (1.91)* <sup>2</sup>
		PK564PMA	PK564PMB						1.5 (3.3)* <sup>2</sup>
	High-resolution type with encoder	PK564PMA-R28L	-	90 (20.1)	100 (22)	130 (29.1)	180 (40.3)	270 (60.4)	0.65 (1.43)* <sup>2</sup>
		PK566PMA-R28L							0.87 (1.91)* <sup>2</sup>
		PK564PMA-R28L							1.5 (3.3)* <sup>2</sup>
	Standard-type with encoder	PK564NAW-R27L	-	63 (14)	75 (16)	95 (21)	130 (29)	150 (33)	0.7 (1.54)* <sup>2</sup>
		PK566NAW-R27L							0.9 (1.98)* <sup>2</sup>
		PK569NAW-R27L							1.4 (3.08)* <sup>2</sup>
	Standard-type	PK564NAW	PK564NAW	63 (14)	75 (16)	95 (21)	130 (29)	150 (33)	0.6 (1.32)* <sup>2</sup>
		PK566NAW	PK566NAW						0.8 (1.76)* <sup>2</sup>
		PK569NAW	PK569NAW						1.3 (2.89)* <sup>2</sup>
	TH geared type	PK564AW-T□	PK564BW-T□	70 (15)	80 (18)	100 (22)	120 (27)	150 (33)	40 (9)
	PS geared type	PK564AW-PS25	PK564BW-PS25	330 (74)	360 (81)	400 (90)	450 (101)	520 (117)	100 (22)
		PK564AW-PS36	PK564BW-PS36						
		PK564AW-PS50	PK564BW-PS50						
	PK566AW-PS5	PK566BW-PS5	200 (45)	220 (49)	250 (56)	280 (63)	320 (72)	100 (22)	
	PK566AW-PS7	PK566BW-PS7	250 (56)	270 (60)	300 (67)	340 (76)	390 (87)		
	PK566AW-PS10	PK566BW-PS10							
Harmonic geared type	PK564AW-H□S	PK564BW-H□S	320 (72)	370 (83)	440 (99)	550 (123)	720 (162)	450 (101)	

\*<sup>1</sup> □ within the model name represents the gear ratio.

\*<sup>2</sup> Indicates the motor mass [kg (lb)]. The thrust load should not exceed the motor's mass.

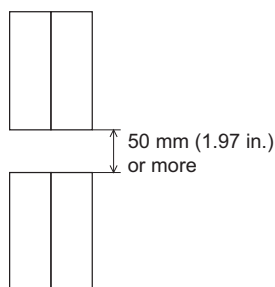
## 7.5 Installing the driver

### ■ Installation direction

Use a DIN rail 35 mm (1.38 in.) wide to mount the driver. Provide 50 mm (1.97 in.) clearances in the horizontal and vertical directions between the driver and enclosure or other equipment within the enclosure. Refer to the figure below for the required distances between adjacent drivers when two or more drivers are installed in parallel.

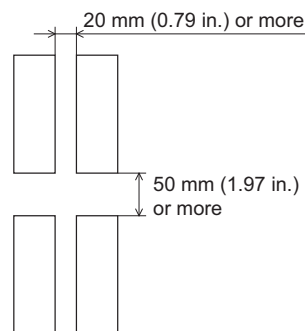
- CRD503-KP, CRD507-KP, CRD507H-KP

If two or more CRD503-KP, CRD507-KP or CRD507H-KP units are installed, they can be placed in contact with each other in the horizontal direction. Provide a clearance of 50 mm (1.97 in.) or more in the vertical direction.



- CRD514-KP

Provide a clearance of 20 mm (0.79 in.) or more in the horizontal direction, and 50 mm (1.97 in.) or more in the vertical direction.

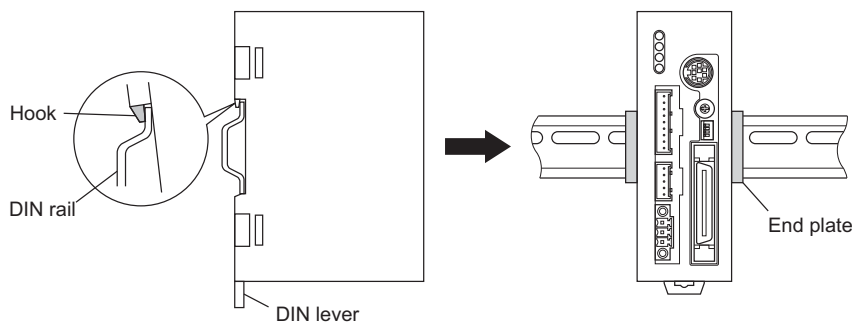


### Note

Be sure to install (position) the driver vertically. When the driver is installed in any position other than vertical, the heat radiation effect of the driver will drop.

### ■ Installation method

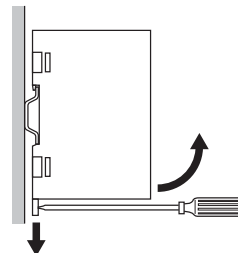
Push up the driver's DIN lever until it locks. Hang the hook at the rear to the DIN rail, and push in the driver. After installation, secure both sides of the driver with an end plate.



### Removing from DIN rail

Pull the DIN lever down until it locks using a flat tip screwdriver, and lift the bottom of the driver to remove it from the rail.

Use a force of about 10 to 20 N (2.2 to 4.5 lb.) to pull the DIN lever down to lock it. Excessive force may damage the DIN lever.



## 7.6 Installing and wiring in compliance with EMC Directive

Effective measures must be taken against the EMI that the motor and driver may give to adjacent control-system equipment, as well as the EMS of the motor and driver itself, in order to prevent a serious functional impediment in the machinery. The use of the following installation and wiring methods will enable the motor and driver to be compliant with the EMC directive. Refer to “CE Marking” on p.7 for the applicable standards. Oriental Motor conducts EMC measurements its motors and drivers in accordance with “Example of motor and driver wiring” on p.24. The user is responsible for ensuring the machine’s compliance with the EMC Directive, based on the installation and wiring explained below.

### ■ Power supply

These products are specifically designed for DC power supply input.

Use a DC power supply (such as a switching power supply) compliant with the EMC Directive.

### ■ Connecting noise filter for power supply line

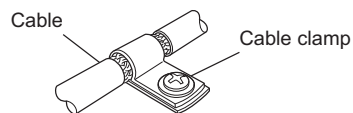
- Connect a noise filter in the DC power supply input part to prevent the noise generated in the driver from propagating externally through the power supply line.
- When using a power supply transformer, be sure to connect a noise filter to the AC input side of the power supply transformer.
- For a noise filter, use 10ESK1 (Tyco Electronics CORCOM), ZAG2210-11S (TDK Corporation) or equivalent product.
- Install the noise filter as close to the AC input terminal of DC power supply as possible. Use cable clamps and other means to secure the input and output cables or lead wires (AWG18: 0.75 mm<sup>2</sup> or more) firmly to the surface of the enclosure.
- Connect the ground terminal of the noise filter to the grounding point, using as thick and short a wire as possible.
- Do not place the AC input cable or lead wires (AWG18: 0.75 mm<sup>2</sup> or more) parallel with the noise filter output cable or lead wire (AWG18: 0.75 mm<sup>2</sup> or more). Parallel placement will reduce noise filter effectiveness if the enclosure’s internal noise is directly coupled to the power supply cable or lead wire by means of stray capacitance.

### ■ How to ground

The cable or lead wire used to ground the driver and noise filter must be as thick and short as possible so that no potential difference is generated. Choose a large, thick and uniformly conductive surface for the grounding point. Install the motor onto a grounded metal surface.

### ■ Wiring the power supply and signal cables or lead wires

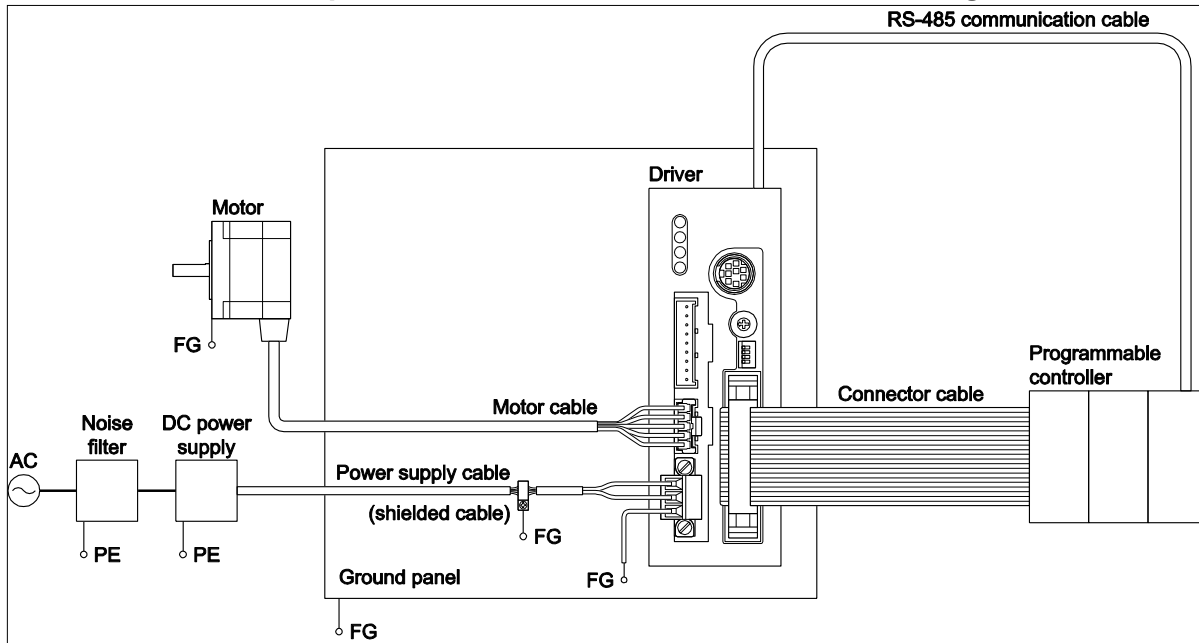
- Use a shielded cable of AWG22 (0.3 mm<sup>2</sup>) or more for the power supply cable, and keep it as short as possible.
- Use the supplied ribbon cable for the I/O signals cable, and keep it as short as possible.
- To ground a power supply cable, use a metal clamp or similar device that will maintain contact with the entire circumference of the cable. Attach a cable clamp as close to the end of the cable as possible, and connect it as shown in the figure.



### ■ Notes about installation and wiring

- Connect the motor, driver and other peripheral control equipment directly to the grounding point so as to prevent a potential difference from developing between grounds.
- When relays or electromagnetic switches are used together with the system, use noise filters and CR circuits to suppress surges generated by them.
- Keep cables as short as possible without coiling and bundling extra lengths.
- Place the power cables such as the motor and power supply cables as far apart [100 to 200 mm (3.94 to 7.87 in.)] as possible from the signal cables or lead wires. If they have to cross, cross them at a right angle. Place the AC input cable and output cable of a noise filter separately from each other.

## ■ Example of motor and driver installation and wiring



## ■ Precautions about static electricity

Static electricity may cause the driver to malfunction or suffer damage. While the driver is receiving power, handle the driver with care and do not come near or touch the driver. Always use an insulated screwdriver to adjust the driver's switches.

### Note

The driver uses parts that are sensitive to electrostatic charge. Before touching the driver, turn OFF the power to prevent electrostatic charge from generating. If an electrostatic charge is impressed on the driver, the driver may be damaged.



# 8 Connection

This chapter explains how to connect the driver, motor, I/O signals, power supply, and grounding method.

## Note

- Ensure that the connector(s) is plugged in securely. Insecure connector connection may cause malfunction or damage to the motor or driver.
- The CN2/CN4/CN5 connectors have a lock mechanism. When removing these connectors, release the connector lock first. Forcibly pulling out the connector without releasing the connector lock may damage the connector.
- To cycle the power or when plugging/unplugging the connector, turn OFF the power and then wait for at least 5 seconds.
- If the motor or power supply cables or lead wires generate an undesirable amount of noise, shield the cable or lead wires, or install a ferrite core.

## 8.1 Connecting the motor

Connect the supplied motor lead wire cable (5 wires) into the motor connector (CN4) on the driver. Next, connect the motor leads and the CN4 cable leads. The customer must provide a suitable terminal block, connectors and other items needed to interconnect the leads.

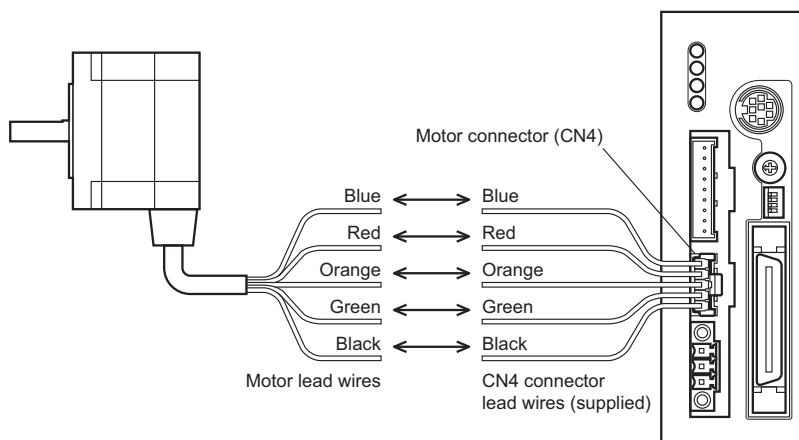
### ▪ Applicable products

Standard type with encoder, Standard type, **TH** geared type (**CRK543**, **CRK544**, **CRK564** and **CRK566**), **PS** geared type (**PK523-PS**)

### ▪ Connection method

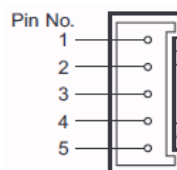
1. Connect the CN4 connector leads (5 pins) to the motor connector (CN4) on the driver.
2. Connect the motor leads and CN4 connector leads.

The customer must provide the terminal block, connectors and other items needed to interconnect the leads.



### ▪ CN4 pin assignments

Pin No.	Description
1	Blue motor lead
2	Red motor lead
3	Orange motor lead
4	Green motor lead
5	Black motor lead



### ▪ CN4 connector assembly parts

Connector housing	51103-0500 (Molex)
Contact	50351-8100 (Molex)
Crimping tool	57295-5000 (Molex)
Applicable lead wire size	AWG22 (0.32 mm <sup>2</sup> )

## ■ Connector-type motor

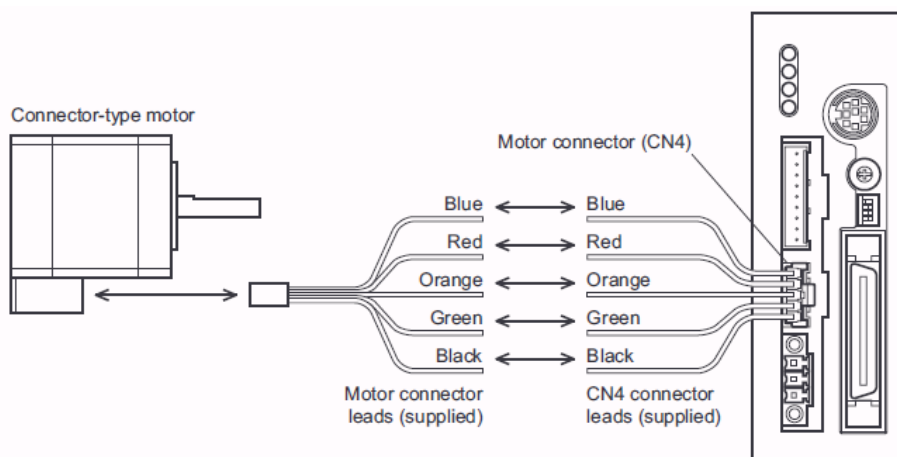
### ■ Applicable products

High-resolution type, High-resolution type with encoder, High-torque type, High-torque type with encoder, **TH** geared type (**CRK513P**□-T□, **CRK523P**□-T□), **PS** geared type (**PK523P**□-PS□)

### ■ Connection method

1. Connect the CN4 connector leads (5 pins) to the motor connector (CN4) on the driver.
2. Connect the motor connector leads (5 pins) to the motor.
3. Connect the motor connector leads and CN4 leads.

The customer must provide the terminal block, connectors and other items needed to interconnect the leads.

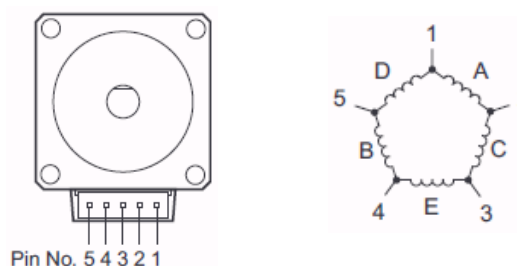


### ■ Connector assembly parts for motor's with a connector

Frame size [mm (in.)]	□20 (0.78) for <b>CRK51</b> □28 (1.10) for <b>CRK52</b>	□42 (1.65) for <b>CRK54</b>	□60 (2.36) for <b>CRK56</b>
Connector housing	51065-0500 (Molex)	51103-0500 (Molex)	51144-0500 (Molex)
Contact	50212-8100 (Molex)	50351-8100 (Molex)	50539-8100 (Molex)
Crimping tool	57176-5000 (Molex)	57295-5000 (Molex)	57189-5000 (Molex)
Applicable lead wire size	AWG24 (0.2mm <sup>2</sup> )	AWG22 (0.3mm <sup>2</sup> )	AWG22 (0.3mm <sup>2</sup> )

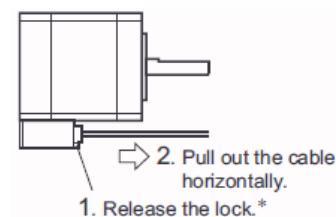
### ■ Connector pin assignments for motor's with a connector

Pin No.	Description
1	Blue motor lead
2	Red motor lead
3	Orange motor lead
4	Green motor lead
5	Black motor lead



### Note

- When connecting a motor, affix the cable or lead wires near the connector to prevent the connector from receiving stress due to flexing of the cable or lead wires. Make the cable's or lead wire's radius of curvature as large as possible.
- When disconnecting the leads from the connector type motor, pull the connector leads horizontally along the output shaft to remove. The motor may be damaged if force is applied in any other direction.
- The motor lead wire/connector assemblies that come with the **CRK54**□P, **CRK54**□PM and **CRK56**□PM have a connector with a lock mechanism. When removing these types of lead wire/connector assemblies, release the connector lock first. Forcibly pulling out the lead wire/connector assembly without releasing the connector lock may damage the motor and connector.



\* **CRK54**□P, **CRK54**□PM and **CRK56**□PM only.

## 8.2 Connecting the power supply and grounding the driver

### ■ Connecting the power supply

Use the CN1 connector (3 pins) to connect the power supply cable (AWG22: 0.3 mm<sup>2</sup>) to the power supply connector (CN1) on the driver.

Use a power supply capable of supplying the current capacity as shown below.

Driver model	CRD503-KP	CRD507-KP CRD507H-KP	CRD514-KP
Input power supply voltage	+24 VDC±10%		
Current capacity	0.70 A or more	1.4 A or more	2.5 A or more

### ■ Grounding the driver

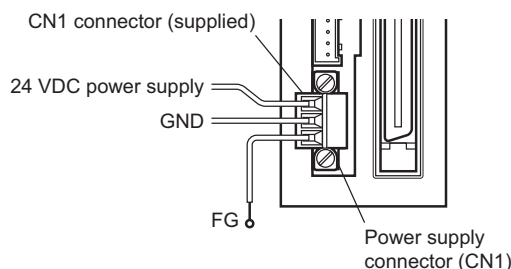
Ground the driver's Frame Ground Terminal (FG) as necessary.

Ground using a wire of AWG24 to 16 (0.2 to 1.25 mm<sup>2</sup>), and do not share the protective earth terminal with a welder or any other power equipment.

CN1 pin assignments

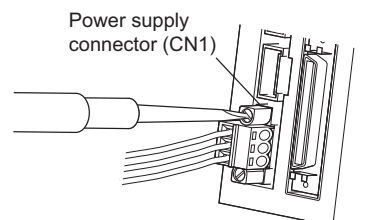
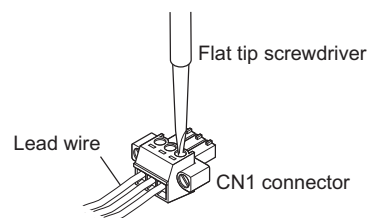
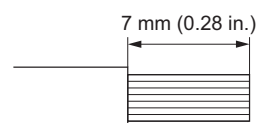
Pin No.	Name	Description
1	+24 VDC	+24 VDC power supply input
2	GND	Power supply GND
3	FG	Frame Ground

CN1 Connector part number:  
MC 1,5/3-STF-3,5 (Phoenix Contact)



### ■ Connecting method

- Strip the insulation cover of the lead wire by 7 mm (0.28 in.)
- Insert each lead wire into the CN1 connector and tighten the screw using a screwdriver (connector screw size: M2).  
Tightening torque: 0.22 to 0.25 N·m (31 to 35 oz-in)
- Insert the CN1 connector into power supply connector (CN1) and tighten the screws using a screwdriver (connector screw size: M2.5).  
Tightening torque: 0.4 N·m (56 oz-in)

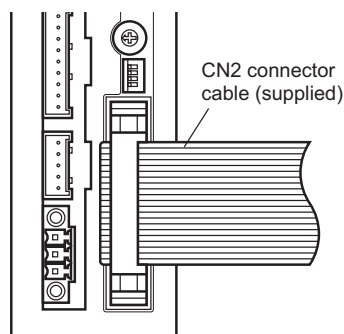


#### Note

- Pay attention to polarity when connecting the power supply. Connecting the power supply in reverse polarity may damage the driver.
- Do not wire the power supply cable of the driver in the same cable duct with other power lines or motor cables or lead wires. Doing so may cause malfunction due to electrical noise.

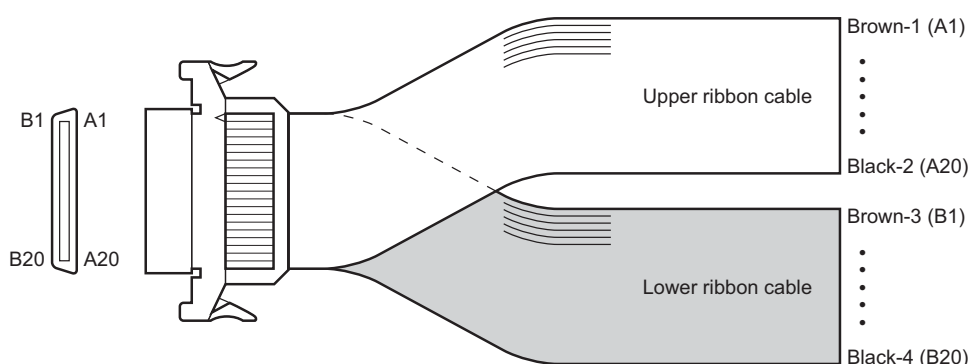
## 8.3 Connecting the I/O signals

Connect the CN2 connector ribbon cable (40 pins) to the I/O signals connector (CN2) on the driver.



CN2 Connector part number: FX2B-40SA-1.27R (Hirose Electric Co., Ltd.)

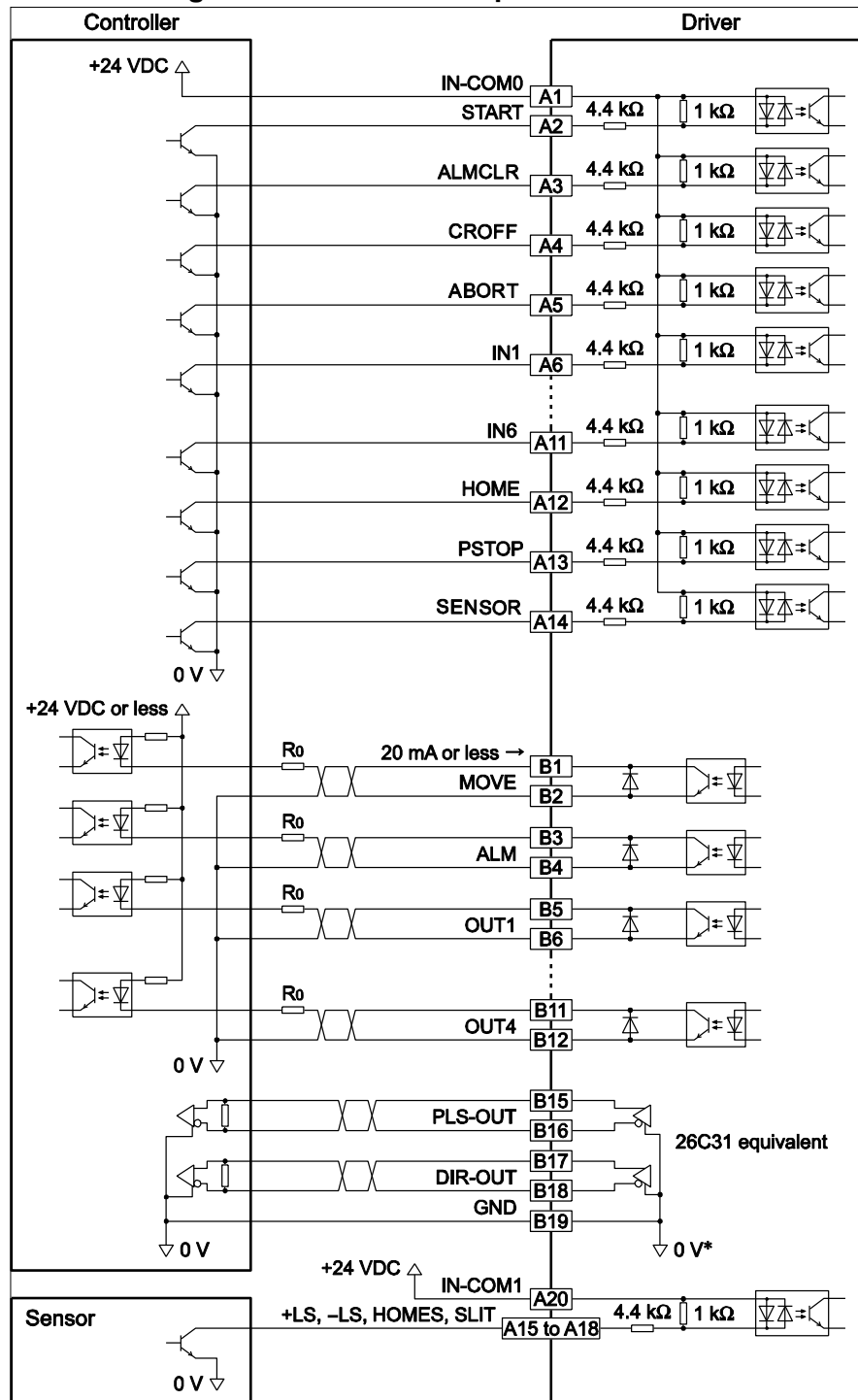
### ■ CN2 pin assignments



Lead wire color	Upper ribbon cable			Lead wire color	Lower ribbon cable		
	Pin No.	Signal name	Description		Pin No.	Signal name	Description
Brown-1	A1	IN-COM0	Input common	Brown-3	B1	MOVE+	Motor moving output
Red-1	A2	START	Start input	Red-3	B2	MOVE-	
Orange-1	A3	ALMCLR	Alarm Clear input	Orange -3	B3	ALM+	Alarm output
Yellow-1	A4	CROFF	Current OFF input	Yellow-3	B4	ALM-	
Green-1	A5	ABORT	Abort input	Green-3	B5	OUT1+	General output 1*2
Blue-1	A6	IN1	General inputs*1	Blue-3	B6	OUT1-	
Purple-1	A7	IN2		Purple-3	B7	OUT2+	General output 2*2
Gray-1	A8	IN3		Gray-3	B8	OUT2-	
White-1	A9	IN4		White-3	B9	OUT3+	General output 3*2
Black-1	A10	IN5		Black-3	B10	OUT3-	
Brown-2	A11	IN6		Brown-4	B11	OUT4+	General output 4*2
Red-2	A12	HOME	Homing operation input	Red-4	B12	OUT4-	
Orange -2	A13	PSTOP	Panic Stop input	Orange -4	B13	N.C.	Not used
Yellow-2	A14	SENSOR	Sensor input	Yellow-4	B14	N.C.	Not used
Green-2	A15	+LS	+ Limit switch input	Green-4	B15	PLS-OUT+	Pulse output (Line driver output)
Blue-2	A16	-LS	- Limit switch input	Blue-4	B16	PLS-OUT-	
Purple-2	A17	HOMES	Home sensor input	Purple-4	B17	DIR-OUT+	Direction output (Line driver output)
Gray-2	A18	SLIT	Slit sensor input	Gray-4	B18	DIR-OUT-	
White-2	A19	N.C.	Not used	White-4	B19	GND	GND
Black-2	A20	IN-COM1	Sensor input common	Black-4	B20	N.C.	Not used

\*1 The function of General Input 1(IN1) to 6(IN6) can be assigned unique functions using the "INxxx" commands.

## ■ Connecting to a current sink output circuit

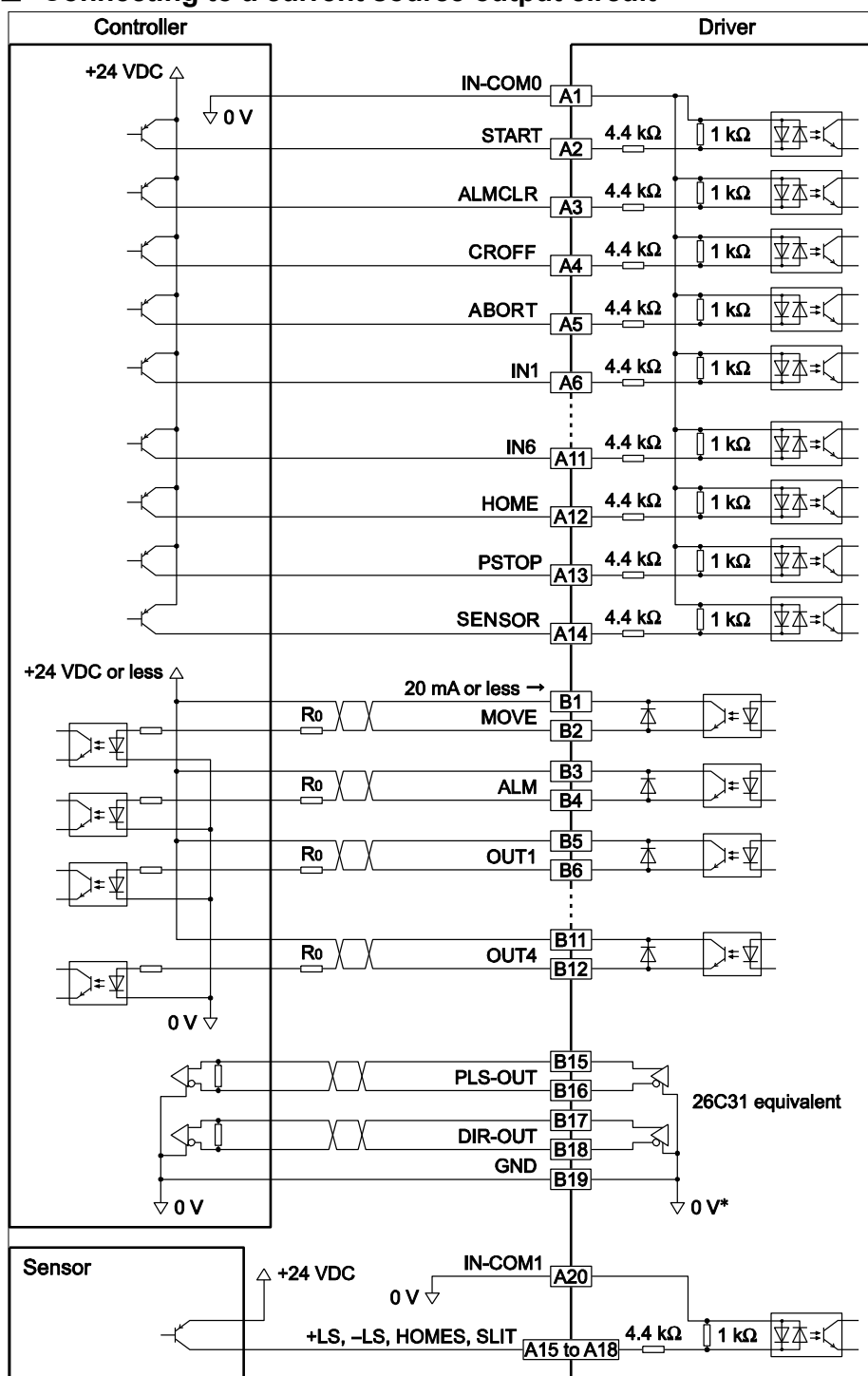


\* The GND line is used in common with CN1 (not insulated).

### Note

- Use input signals at 24 VDC.
- Use output signals at 24 VDC or less. If the current exceeds 20 mA, connect an external resistor  $R_0$ .
- The PLS-OUT output and DIR-OUT output are line driver outputs. When connecting a line receiver, be sure to connect pin No.B19 on the driver to the GND on the line receiver, and connect a termination resistor of 100  $\Omega$  or more between the driver and the input of the line receiver.

## ■ Connecting to a current source output circuit



\* The GND line is used in common with CN1 (not insulated).

### Note

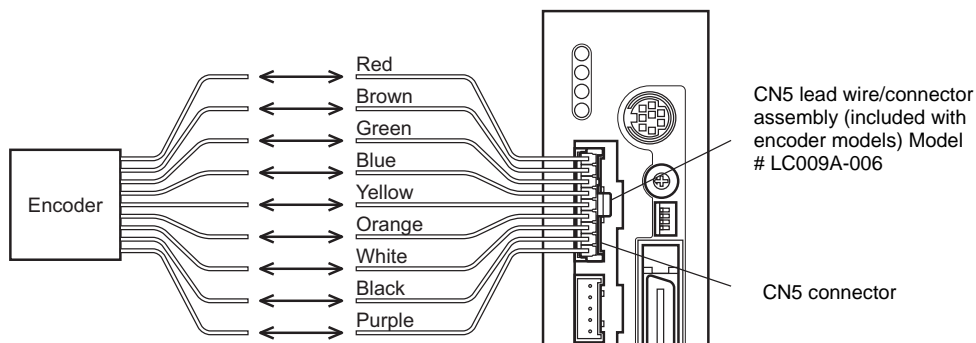
- Use input signals at 24 VDC.
- Use output signals at 24 VDC or less. If the current exceeds 20 mA, connect an external resistor  $R_0$ .
- The PLS-OUT output and DIR-OUT output are line driver outputs. When connecting a line receiver, be sure to connect pin No.B19 on the driver to the GND on the line receiver, and connect a termination resistor of 100  $\Omega$  or more between the driver and the input of the line receiver.

## 8.4 Connecting an encoder

If an encoder is to be used, connect the encoder using CN5.

Using the optional CN5 connector lead wire/connector assembly (9 pins; sold separately), connect the encoder to the encoder connector (CN5) on the driver. When extending the leads, use wires of AWG24 to 22 (0.2 to 0.3 mm<sup>2</sup>).

Refer to p.51 for the detailed specifications of this encoder.

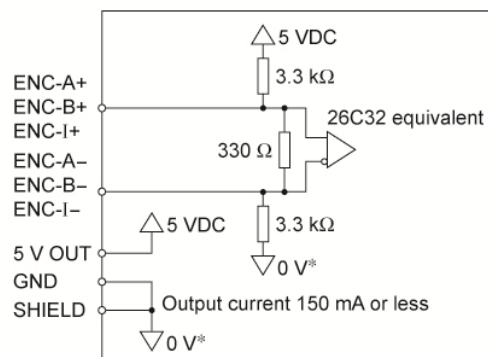


### • CN5 pin assignments

Pin No.	Signal name	Lead wire color	Description
1	ENC-A+	Red	Encoder input A- channel (Line receiver)
2	ENC-A-	Brown	
3	ENC-B+	Green	Encoder input B- channel (Line receiver)
4	ENC-B-	Blue	
5	ENC-I+	Yellow	Encoder input Index signal (Line receiver)
6	ENC-I-	Orange	
7	+5 VDC OUT	White	+5 VDC power supply output for encoder
8	GND	Black	GND
9	SHIELD	Purple	Shield (Connect to GND)

Applicable housing:  
51103-0900 (Molex)  
Applicable contact:  
50351-8100 (Molex)  
Specified crimping tool:  
57295-5000 (Molex)

### • Internal circuit diagram



\* The GND line is used in common with CN1 (not insulated).

### Note

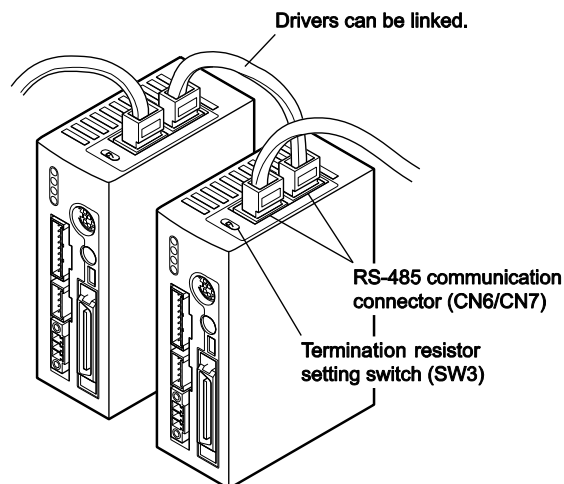
The current consumption of the encoder power supply should be kept to 150 mA or less. If the encoder power consumption exceeds 150 mA, provide an encoder power supply externally to the system. In this case, be sure to use a common GND line for the encoder power supply and encoder connector (CN5).

## 8.5 Connecting the RS-485 communication cable

Connect this cable if you want to control your product via RS-485 communication. Connect a RS-485 communication cable to CN6 or CN7 on the driver.

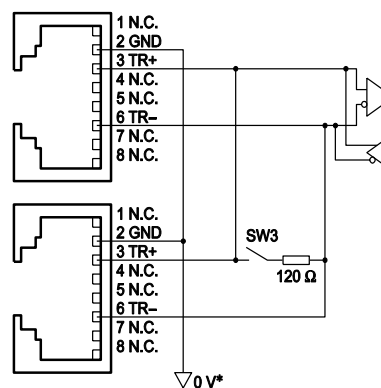
You can use the vacant connectors to link to a different driver. An optional driver link cable (sold separately) is available. See p.300.

You can also use a commercially available LAN cable to link drivers.



CN6/CN7 pin assignments

Pin No.	Signal name	Description
1	N.C.	Not used
2	GND	GND
3	TR+	RS-485 communication signal (+)
4	N.C.	Not used
5	N.C.	Not used
6	TR-	RS-485 communication signal (-)
7	N.C.	Not used
8	N.C.	Not used



\* The GND line is used in common with CN1 (not insulated).

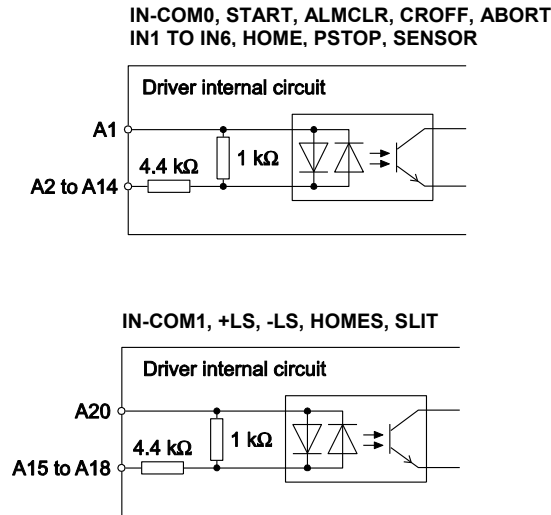


# 9 Explanation of I/O signals

## 9.1 Input signals

The following input signals of the driver are photocoupler inputs. The signal state represents the “ON: Carrying current” or “OFF: Not carrying current” state of the internal photocoupler rather than the voltage level of the signal.

**Note** Timing charts for the I/O signals can be found in Chapter 22, “Timing Charts” on page 308.



### ■ START input

This signal is used to start the sequence. Set the starting method using the STARTACT command. Additionally, the input logic can be changed using the STARTLV command. (The factory setting of this command is normally open.) The leading edge of this signal will cause the sequence to start.

### ■ ALMCLR input

This signal is used to reset the alarm that has been generated by the driver’s protective function. Input the ALMCLR signal once after removing the cause that has triggered the protective function. Additionally, the input logic can be changed using the ALMCLRLV command. (The factory setting of this command is normally open.) The trailing edge of the signal will cause the alarm to be cleared.

### ■ CROFF input

This signal is used to free the shaft by removing current to the motor. Additionally, the input logic can be changed using the CROFFLV command. (The factory setting of this command is normally open.) The leading edge of this signal will remove the current to the motor.

### ■ ABORT input

This signal is used to stop motion and the sequence. Additionally, the input logic can be changed using the ABORTLV command (The factory setting of this command is normally open.). When the ABORT input is ON, with the exception of when the START input is in the ABORT position when it is set to act as a toggle switch (STARTACT=1), no motion commands will be executed. The leading edge of this signal will cause the action.

## ■ IN1 to IN6 input

The IN1 through IN6 inputs can be used as input ports for general signals. The status of each port can be read using an IN command or INx (x=1–6) command. The general signals assignable to the IN1 through IN6 inputs are listed below.

Pause ..... INPAUSE  
Pause Clear..... INPAUSECL

## ■ HOME input

The homing operation starts when the HOME input turns ON.

## ■ PSTOP input

This signal is used to forcibly stop motion and the sequence. Set the stopping method using the ALMACT command. Additionally, the input logic can be changed using the PSTOPLV command. (The factory setting of this command is normally open.) The leading edge of the signal will cause the action.

## ■ SENSOR input

This signal is used to change the sensor operation. This signal is used for:

- Stopping motion during continuous operation.
- Offset motion on the fly during continuous operation.

Set the operation using the SENSORACT command. Additionally, the input logic can be changed using the SENSORLV command. (The factory setting of this command is normally open.) The leading edge of the signal will cause the action.

## ■ +LS input, -LS input

These signals are input from the applicable limit sensors. They are used to detect the home during return-to-home operation. In any other operation, these signals are used to stop the motor. The input logic can be changed using the OTLV command. (The factory setting of this command is normally open.)

### Note

If the +LS and -LS inputs are to be used in an operation other than return-to-home, set the “hardware over travel detection” parameter to “enable”.

## ■ HOMES input

These signals are input from the applicable HOME sensors. This input detects the mechanical home position when a return-to-home operation is executed in the 3-sensor mode. Additionally, the input logic can be changed using the HOMESLV command. (The factory setting of this command is normally open.) The leading edge of the signal will start the home seeking.

## ■ SLIT input

This signal is used to detect the home using a slit disc, etc. When detecting the home, use of the SLIT input in addition to the HOMES input and +LS/-LS inputs will increase the accuracy of home detection. The input logic can be changed using the SLITLV command. (The factory setting of this command is normally open.)

### Note

If the SLIT input is used, set the “SLIT detection with home-seeking” parameter to “enable”.

## ■ IN-COM0 input

This is a common terminal for input signals.

## ■ IN-COM1 input

This is a common terminal for sensor input signals.

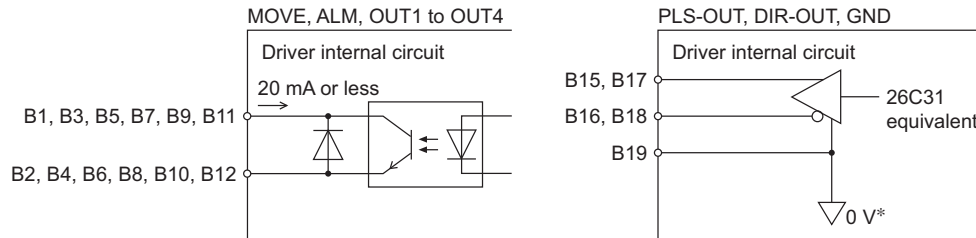
### Note

Use sensor input signals at 24 VDC±10%.

## 9.2 Output signals

The driver outputs signals in the photocoupler/open-collector output mode or line driver output mode. The signal state represents the “ON: Carrying current” or “OFF: Not carrying current” state of the internal photocoupler rather than the voltage level of the signal.

**Note** Timing charts for the I/O signals can be found in Chapter 22, “Timing Charts” on page 308.



\* The GND line is used in common with CN1 (not insulated).

### ■ MOVE output

The MOVE output becomes ON while operating the motor or during return-to-home operations. Even when the current motion has completed, the next motion cannot be started while the MOVE output is still ON.

### ■ ALM output

This signal is output when an alarm is generated by the device’s protective function. The reason for triggering of the protective function can be identified through the blink count of the alarm LED, or ALM command. To reset the ALM output, remove the cause of the alarm and then perform one of the following procedures after ensuring safety:

- Enter an ALMCLR command.
- Turn OFF the power, wait at least 10 seconds, and then turn it back on.
- Input ALMCLR signal.

Additionally, the output logic can be changed using the ALMLV command. (The factory setting of this command is normally open.) [OFF: No Alarm, ON: Alarm state]

### ■ OUT1 to OUT4 output

The “OUT1 signal mode selection” to “OUT4 signal mode selection” parameters are used to set the desired functions to be assigned to the OUT1 to OUT4 outputs, respectively. The following output signals can be assigned:

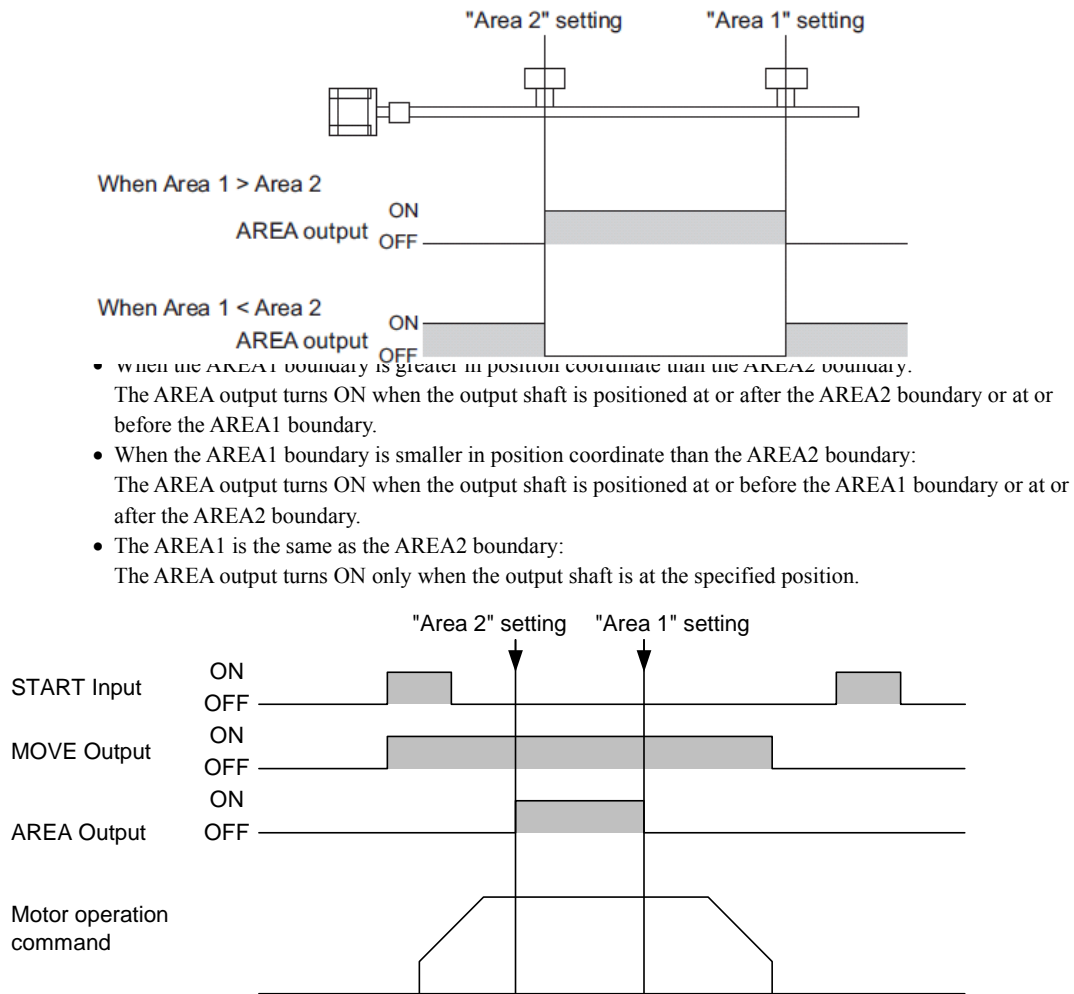
- AREA output
- HOMEP output (return-to-home ready complete output)
- PSTS output (pause status output)
- READY output (operation ready output)
- RUN output (sequence or operation is active)
- SC output (Self correcting operation was active)
- STO output (stepout detection output)
- TEMP output (temperature warning output)
- WNG output (warning output)

### ■ AREA output

The AREA output can be assigned to a general output.

This signal will be output when the commanded motor position is inside the area set by the “AREA 1” and “AREA2” parameters. This signal is also output while the motor is stopped.

**Note** If the AREA output is to be used during operation, set the width of the area so that the AREA output will remain ON for at least 1 ms. If the AREA output remains ON for less than 1 ms, the AREA output may not actually turn ON.



## ■ HOMEP output

The HOMEP output can be assigned to a general output.

This signal is output upon completion of return-to-home. It will turn ON when all of the following conditions are satisfied:

- The home is already set
- The command position has become 0
- The motor is stopped

The home can be set by the following methods:

- Successful completion of return-to-home operation
- Setting the position counter (PC) to 0 (zero) via RS-485 communication.

The home will be cancelled when either of the following operations is performed:

- Cycle the power.
- Stop the motor excitation (when the “stepout detection” parameter is set to “disable”)

## ■ PSTS output

The PSTS signal can be assigned to a general output.

This signal is output while the device is pausing with the PAUSE input signal. Additionally, the output logic can be changed using the PSTSLV command.

[OFF: No PAUSEing, ON: PAUSEing]

## ■ READY output

The READY output can be assigned to a general output.

This signal will be output when the driver becomes ready. Start operation after the READY output has turned ON. Additionally, the output logic can be changed using the READYLV command.

The READY output remains OFF in the following conditions:

- A sequence is running
- The motor is operating
- An alarm is present
- Any one of the HOME input and START input is ON
- The CROFF input is ON
- The ABORT input is ON (Not including ABORT status of START input when set to act as a toggle switch (STARTACT=1)).
- The PSTOP input is ON
- The motor is not excited
- Immediately after the power was turned ON

## ■ RUN output

The RUN output can be assigned to a general output.

This signal will be output when the driver is running a sequence, regardless if the motor is moving or not.

Additionally, the output logic can be changed using the RUNLV command.

[OFF: Not RUNning, ON: RUNning]

## ■ SC output

The SC output can be assigned to a general output. This signal is effective only when an encoder is connected.

This signal is output when a step deviation error has occurred and was corrected automatically. The SC output will turn OFF when the next motion command is executed or if the motor current is turned OFF. If the SC output is to be used, set the “SCEN” parameter to “enable”.

Additionally, the output logic can be changed using SCLV command.

## ■ STO output

The STO output can be assigned to a general output.

This signal becomes effective when an encoder is connected, and a deviation error occurs. This signal will be output when the deviation between the encoder counter value and driver command position reaches the value set in the “STOB” parameter. If the STO output is to be used, set the “STOEN” parameter to “enable”.

Additionally, the output logic can be changed using the STOLV command.

### Note

- While the motor is not excited, the STEPOUT output is always OFF. The signal will become effective once the motor has remained excited for at least 500 ms.
- The STEPOUT output remains OFF during return-to-home operation.

## ■ TEMP output

The TEMP output can be assigned to a general output.

This signal becomes effective when the temperature of the driver electronics (DTMP) exceeds the value set by the DTMPWNG command. Additionally, the output logic can be changed using the TEMPLV command.

## ■ WNG output

The WNG output is assigned to control output.

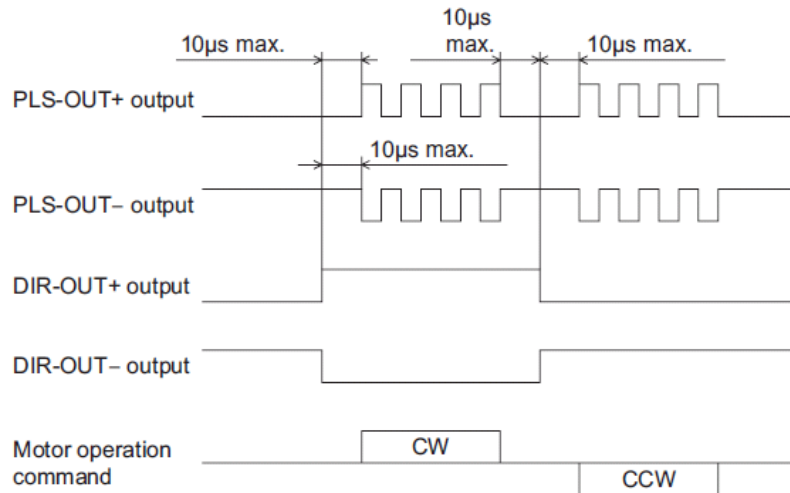
This signal is output when a warning generates. However, the operation will continue.

The WNG output will turn OFF automatically once the cause of the warning is removed. Additionally, the output logic can be changed using the WNGLV command.

## ■ PLS-OUT output, DIR-OUT output

The PLS-OUT output is used to output the driver's internal oscillation pulses. The number of pulses to be output corresponds to the commanded travel. The pulse frequency corresponds to the operating speed. The maximum output frequency is 500 kHz.

The DIR-OUT output is used to output the driver's internal direction command.

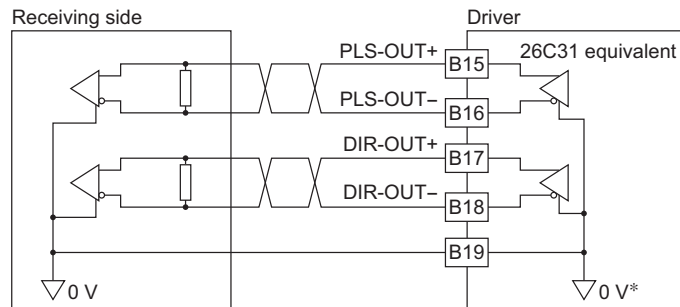


### Note

The PLS-OUT output and DIR-OUT output are line driver outputs. When connecting to a line receiver, be sure to connect pin No. B19 of CN2 with the GND line of the line receiver. Also connect a termination resistor of 100  $\Omega$  or more between the line receiver inputs.

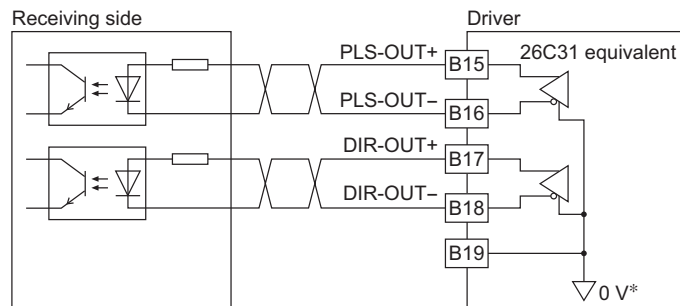
When the self correcting function operates, the number of pulses and the pulse frequency are different from the set motion profile. The number of pulses to be output corresponds to the adjusted travel amount. The pulse frequency corresponds to the corrective action speed. Do not use these signals for any other driver if the self correcting function is used in this driver.

### • Connecting to a line receiver



\* The GND line is used in common with CN1 (not insulated).

### • Connecting to a photocoupler



\* The GND line is used in common with CN1 (not insulated).

# 10 Features

---

This chapter introduces the main features of the CRK Built-in Controller (Stored Program) device.

## 10.1 Overview

The CRK Built-in Controller (Stored Program) is designed to make motion control simple and convenient. At the same time, the system has powerful enhanced features to maximize performance, and support functions to accelerate successful system integration. The following subjects are discussed in the sections which follow:

10.2	Making the Motor Move	Commanding motions and features available
10.3	Motion Types	Point-to-point, continuous, and home-seeking motions
10.4	Stopping Motion	Hard stops, soft stops, and system status after stopping
10.5	Encoder input	Using encoder input
10.6	Misstep Detection function	Detecting missed steps
10.7	Self Correcting function	Automated correction for missed steps
10.8	How to recover from a deviation error	Recovering from missed steps
0		Setting the encoder resolution in the device
	Encoder electronic gear settings	
10.10	Encoder Resolution	Encoder resolution
10.11	Support Functions	Teaching function, I/O test
10.12	Protective Functions	Controlling the system response to alarm conditions

## 10.2 Making the Motor Move

There are four ways to make the motor move:

- Programming mode
  - By configuring motion parameters and sending a motion start command via the serial port
  - By executing a sequence containing motion commands. Sequences can be started via the serial port (using the RUN command), or from the I/O port (using the START input).
  - A sequence named “CONFIG” will be run automatically upon power-up or device RESET.
- Stand alone
- Via IO
- CRK Motion Creator GUI. The latest version of the CRK Motion Creator GUI software is available for download at: <http://www.orientalmotor.com/support/software/software.html>

## 10.3 Motion Types

The supports three types of basic motion: point-to-point motions, continuous motions, and electrical and mechanical home seeking. This section explains each of these basic motion types.

### ■ Point-to-Point Motions

Point-to-point motions cause the motor to start moving from one position to another position, using a preset distance or destination. Motions start and stop at zero speed.

The motor accelerates to running velocity VR and continues to move at that velocity, as necessary, until decelerating to the final target position. Motion begins at starting speed VS, accelerates to VR over acceleration time TA, and finally decelerates back to VS over deceleration time TD before stopping.

- Commands and Parameters for Point-to-Point Motions ( ): default value

Command/ Parameter	Argument/Parameter Value	Function
MI	None	Start incremental motion, distance DIS
MA	-8,388,607 to +8,338,607	Start absolute motion to the specified destination
Command/ Parameter	Argument/Parameter Value	Function
DIS	-8,388,607 to +8,338,607 (0)	Distance for incremental motion [steps]
VS	1 to 500,000 pps (100)	Starting Velocity [pps]
VR	1 to 500,000 pps (1000)	Running Velocity [pps]
TA	0.001 to 1000 (0.5) <sup>□</sup>	Acceleration time [sec.] <sup>□□</sup>
TD	0.001 to 1000 (0.5)	Deceleration time [sec.] <sup>□□</sup>

#### Note

- See “10.4 Stopping Motion” on page 50 for information on stopping motions before they are finished.
- See the description of "Linked Motions" (below) for information on more complex motion profiles.

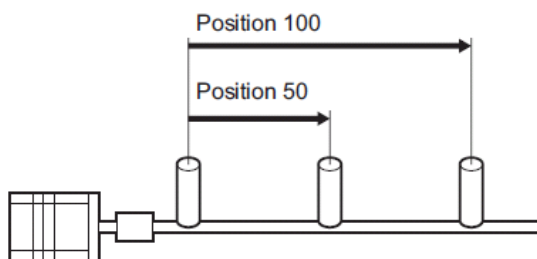
- Point-to-Point Motion Types

Two positioning modes are available for use in the positioning operation: absolute mode and incremental mode.

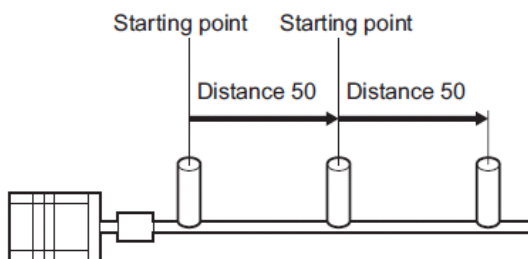
In absolute mode, the distance from electrical home is set.

In incremental mode, each device destination becomes the starting point for the next movement. This mode is suitable when the same distance is repeatedly used.

#### • Absolute Mode



#### • Incremental Mode

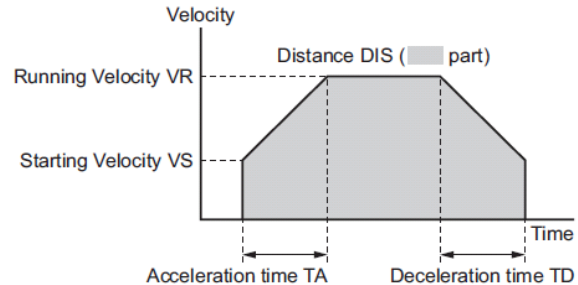




### • Example

Distance: 2000 steps (Incremental)  
 Running Velocity: 1000 pps  
 Starting Velocity: 500 pps  
 Acceleration time: 0.5 sec.  
 Deceleration time: 0.5 sec.

```
>DIS=2000
DIS=2000
>VR=1000
VR=1000
>VS=500
VS=500
>TA=0.5
TA=0.5
>TD=0.5
TD=0.5
>MI
>
```



## ■ Linked Motions

Linked motions are point-to-point motions which may be more complex than motions started with MA (move absolute) or MI (move incremental). Linked motions use up to four (4) running speeds between the start and stop position, and each segment of the motion has its own distance or destination. Segments can be (optionally) linked together: when two segments are linked, the system accelerates (or decelerates) to the second segment's running velocity when the first segment's distance has been traveled or destination has been reached. Motion does not stop between linked segments.

The maximum number of linked segments is four (4).

### Commands and Parameters for Linked Motions

Command/ Parameter	Argument/Parameter Value	Function
MIx (x=0-3)	None	Start linked motion at link segment 'x'
DISx (x=0-3)	-8,388,607 to +8,338,607 (0)	Distance or destination for link segment 'x' [steps]
VRx (x=0-3)	1 to 500,000 (1000)	Running velocity of link segment 'x' [pps]
INCABSx (x=0-3)	0, 1 (1)	Link type for link segment 'x' 0: Absolute 1: Incremental
LINKx (x=0-2)	0, 1 (0)	Link control for link segment 'x' 0: segment terminates linked motion 1: motion continues with next segment
VS	1 to 500,000 (100)	Starting velocity [pps]
TA	0.001 to 1000 (0.5)	Acceleration time [sec.]
TD	0.001 to 1000 (0.5)	Deceleration time [sec.]

( ): default value

#### Note

- See “10.4 Stopping Motion” on page 50 for information on stopping motions before they are finished.
- See notes below for how the acceleration and deceleration times are calculated for the different link segments.
- Link segments can be absolute or incremental, but all segments must execute in the same direction.
- Linked Motions cannot be paused and then continued: PAUSE causes a soft stop, and CONT is ignored.

### • Example

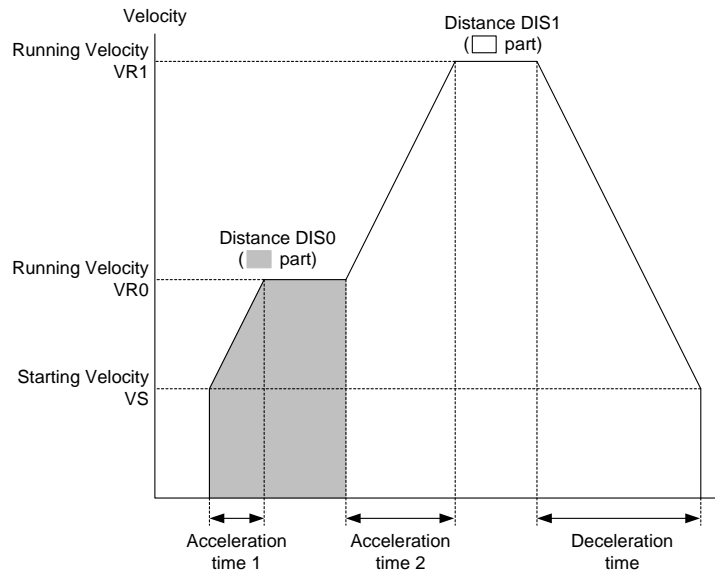
Number of linked segments: 2

Link Segment 0: Distance: 225 steps, Running Velocity: 1000 pps

Link Segment 1: Distance: 975 steps, Running Velocity: 2000 pps

Starting velocity: 500 pps

```
>DIS0 225
  DIS0=225
>DIS1 975
  DIS1=975
>VR0 1000
  VR0=1000
>VR1=2000
  VR1=2000
>INCABS0 1
  INCABS0=1 [INC]
>INCABS1 1
  INCABS1=1 [INC]
>LINK0 1
  LINK0=1
>LINK1 0
  LINK1=0
>TA 0.1
  TA=0.1
>TD 0.1
  TD=0.1
>VS 500
  VS=500
>MI0
>
```



If Acceleration time  $= \frac{TA}{VR0 - VS}$  and Deceleration time  $= \frac{TD}{VR0 - VS}$ , then

$$\text{Acceleration time 1} = \frac{TA}{VR0 - VS} \times (VR0 - VS) = TA$$

$$\text{Acceleration time 2} = \frac{TA}{VR0 - VS} \times (VR1 - VR0)$$

$$\text{Deceleration time} = \frac{TD}{VR0 - VS} \times (VR0 - VS)$$

## ■ Continuous Motions

Continuous motions cause the motor to accelerate or decelerate to a new constant speed and maintain that speed, with no predetermined final position. Motion continues until a new continuous motion command, a stop command, or sensor input to be turned on is executed.

Two continuous motion commands are available: MCP (Move Continuously, Positive) and MCN (Move Continuously, Negative). The new target velocity is determined by the value of running velocity VR at the time the command is executed. The motor accelerate or decelerate to the new target velocity over accelerations time TA or decelerations time TD.

The system changes speed over a fixed time interval. If speed is increasing (away from zero), acceleration time TA is used. If speed is decreasing (toward zero), deceleration time TD is used. (If the motor is stopped when the command is executed, speed changes immediately to starting velocity VS before ramping.)

Velocity change on the fly can be executed by setting a new value of running velocity VR and executing a continuous motion command again: see the example and the description below. Direction changes are not allowed: MCN is only permitted after a previous MCN, and MCP is only permitted after a previous MCP. The SENSOR input can be used to change speed and eventually stop after a predetermined distance: see the **Example** and discussion below.

**Note** See “10.4 Stopping Motion” on page 50 for information on stopping motions before they are finished.

• **Commands and Parameters for Continuous Operation**

Command/ Parameter	Argument/ Parameter Value	Function
MCP	None	Start moving continuously in the positive direction. Change velocity
MCN	None	Move continuous in the negative direction. Change velocity
VR	1 to 500,000 pps (1000)	Running velocity [pps]
VS	1 to 500,000 pps (100)	Starting velocity [pps]
TA	0.001 to 1000 (0.5)	Acceleration time [sec.] □□
TD	0.001 to 1000 (0.5)	Deceleration time [sec.] □□
SENSORACT	0 to 2 (2)	SENSOR input action 0: Hard stop 1: Soft stop 2: Soft stop at fixed distance from SENSOR signal
SCHGPOS	0 to 8,388,607 (0)	Distance from SENSOR input to the stop position [user unit] if SENSORACT=2
SCHGVR	1 to 500,00 pps (1000)	Velocity after SENSOR input [pps.] if SENSORACT=2

( ): default value

• **Example**

Conditions:

Running Velocity: 1000 pps

Starting Velocity: 500 pps

Direction: Positive

>VS=500

VS=500

>VR=1000

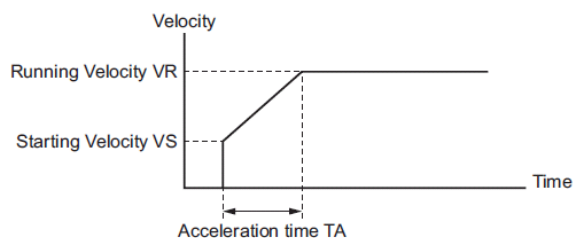
VR=1000

>TA=0.5

TA=0.5

>MCP

>



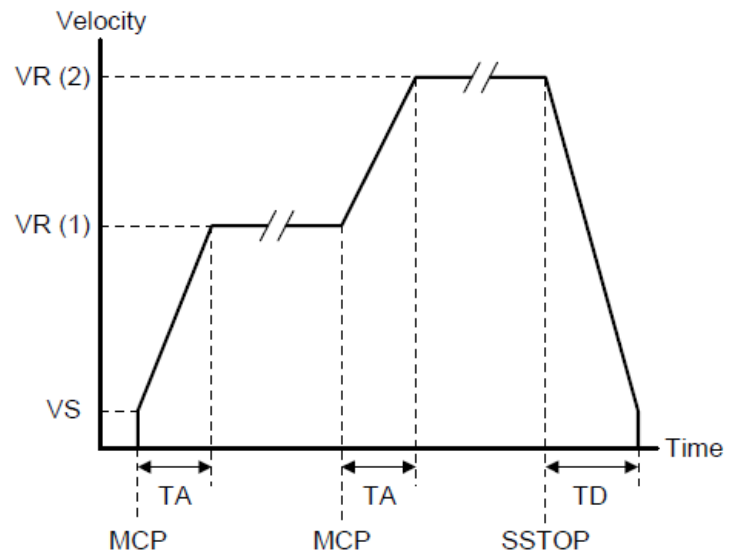
• **Velocity change on the fly**

When a new running velocity VR is set and a continuous motion command is executed during continuous motion, the velocity will change to the VR. If the value of new VR increases from the previous VR, the motor accelerates to the new VR over acceleration time TA determined at the time the continuous motion command is executed. If the value of new VR decreases from the previous VR, the motor decelerates to the new VR over deceleration time TD determined at the time the continuous motion command is executed.

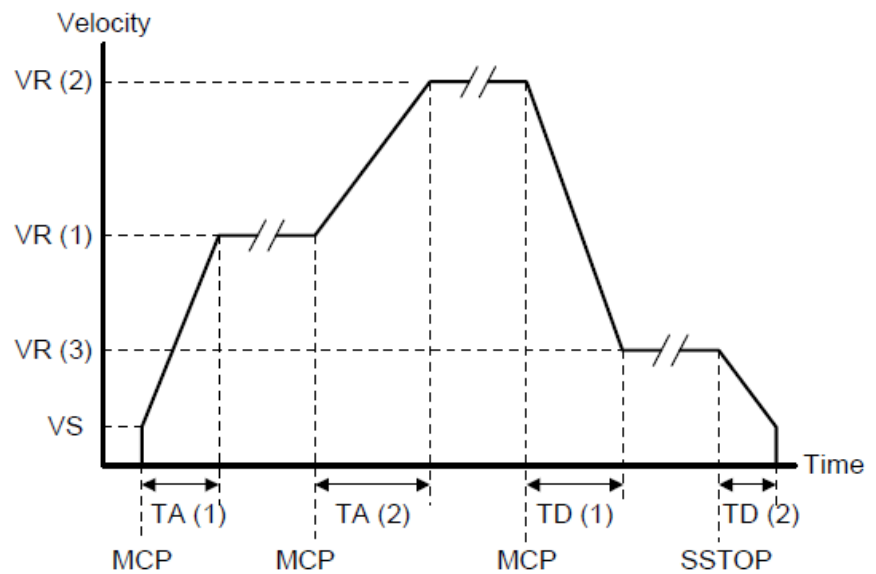
The speed change command can be accepted while the motor accelerate or decelerate from one speed to another. When a continuous motion command is executed during busy ramping, the motor will accelerate or decelerate again to new VR over TA/TD determined at the time the continuous motion command is executed.

• **Example**

```
>VS 100
VS=100
>VR 600
VR=600
>TA 0.4
TA=0.4
>TD 0.5
TD=0.5
>MCP
>VR 1000
VR=1000
>MCP
>SSTOP
```



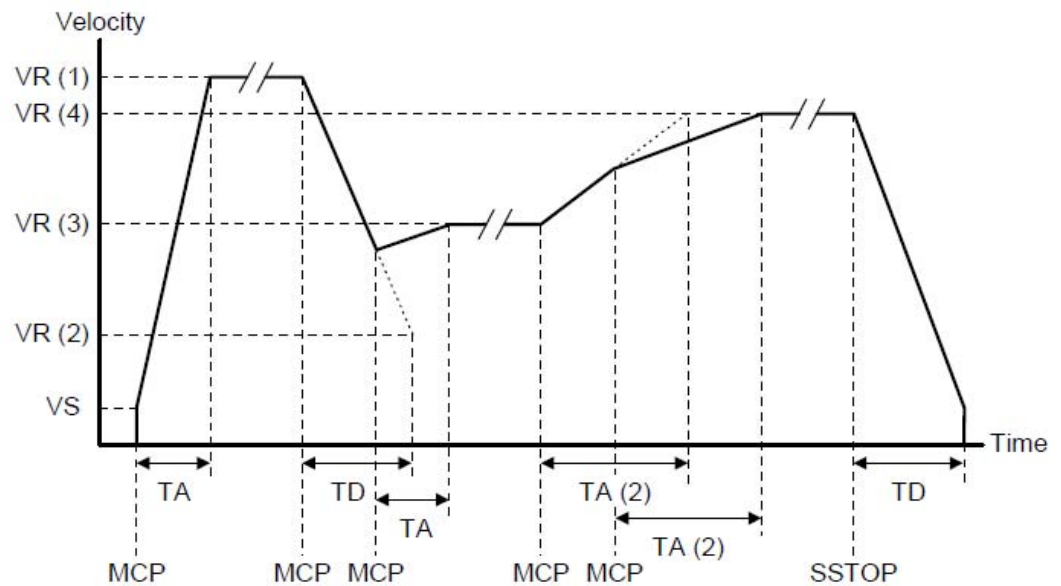
```
>VS 100
VS=100
>VR 600
VR=600
>TA 0.4
TA=0.4
>TD 0.5
TD=0.5
>MCP
>VR 1000
VR=1000
>TA 0.6
TA=0.6
>MCP
>VR 300
VR=300
>MCP
>TD 0.3
TD=0.3
>SSTOP
```



```

>VS 100
VS=100
>VR 1000
VR=1000
>TA 0.4
TA=0.4
>TD 0.6
TD=0.6
>MCP
>VR 300
VR=300
>MCP
>VR 600
VR=600
>MCP
>VR 900
VR=900
>TA 0.8
TA=0.8
>MCP
>MCP
>SSTOP

```

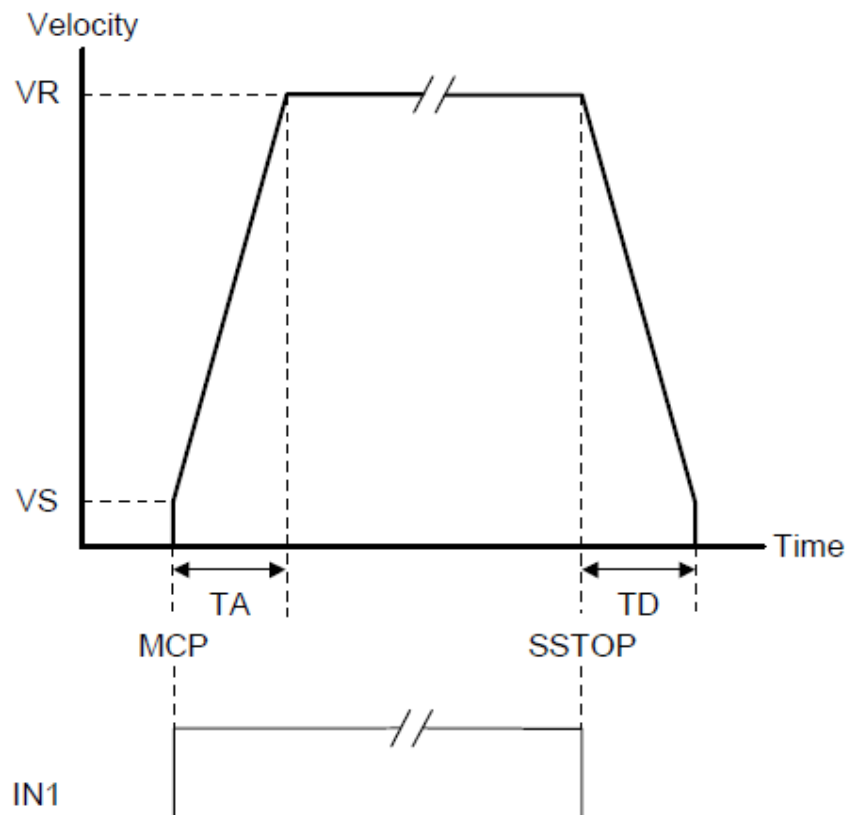


#### Note

Every time a continuous motion command MCP/MCN is executed, the system accelerates or decelerates the motor from the velocity at the time the command executes. Therefore the motor may not accelerate or decelerate when MCP/MCN command is executed repeatedly at short intervals. Be careful to use MCP/MCN command with the commands which execute statements repeatedly in the block, such as LOOP – ENDL, IF – ELSE – ENDIF, and WHILE – WEND. To avoid executing MCP/MCN command repeatedly while a motor accelerate or decelerate, check SIGMOVE status. See the example below.

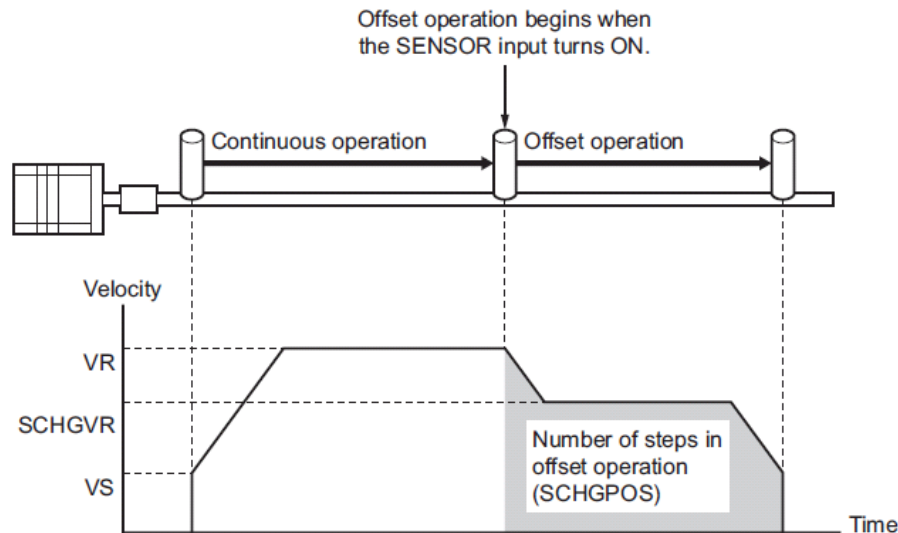
## Example program

(1) VS 100	Set starting velocity to 100 pps
(2) VR 1000	Set starting velocity to 1000 pps
(3) TA 0.5	Set acceleration time to 0.5 sec
(4) TD 0.5	Set deceleration time to 0.5 sec
(5) LOOP	Loop forever
(6) WHILE (IN1=1)	Begin WHILE block: execute while Input 1 is ON
(7) IF (SIGMOVE=0)	Begin IF block
(8) MCP	Execute MCP command if SIGMOVE is OFF (motor is not moving)
(9) ENDIF	End of IF block: back to IF (line 7)
(10) WEND	End of WHILE block: back to WHILE (line 6)
(11) SSTOP	Execute SSTOP command, if input 1 is OFF
(12) MEND	Wait for motion to end
(13) ENDL	End of LOOP block: back to LOOP (line 5)



- **SENSOR Action**

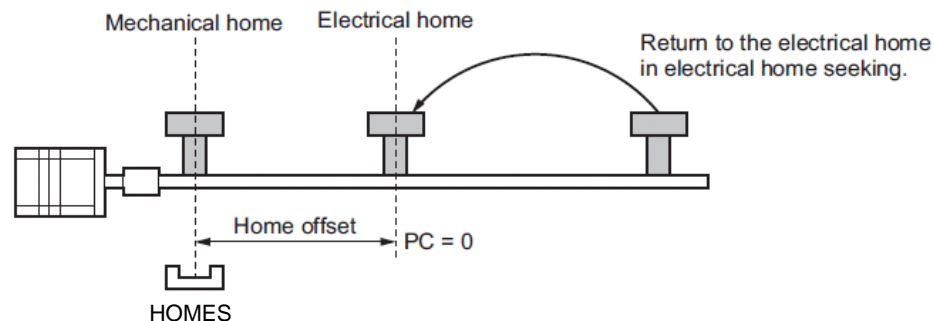
The SENSOR input can be used to stop continuous motions, with stop action determined by SENSORACT. If SENSORACT=0, the system performs a hard stop. If SENSORACT=1, the system performs a soft stop. If SENSORACT=2, the system changes velocity to SCHGVR, and stops at a distance SCHGPOS after the position at which the SENSOR signal was set. See “10.4 Stopping Motion” on page 50 for information on hard stops and soft stops. The picture below illustrates stopping action when SENSORACT=2.



## ■ Mechanical Home Seeking and Electrical Home Position

When the CRK Built-in Controller (Stored Program) device is started or reset, the position counter (PC) is set to position zero (0). The physical position at which PC=0 is called "electrical home". The electrical home position can be aligned with an external reference signal (or signals) through a process called "mechanical home seeking", in which the system moves until a predefined home input signal pattern has been found, and then moves a predefined distance (HOME OFS) from that position. (Mechanical home seeking is described in more detail in the next section.) When mechanical home seeking completes successfully, the final position is redefined as the new electrical home: position counter PC is reset to zero (0).

Position counter PC can also be set to any valid position value by direct assignment provided the motor is not moving.



## ■ Mechanical Home Seeking

When either a HOME input is turned ON or MGHP (MGHN) command is executed, a mechanical home seeking operation is started in the preset direction or the specified direction by MGHP (MGHN) command. When an offset from the mechanical home is set in the “position offset of home-seeking” parameter, the offset position becomes the home.

Two home detection modes are available: 3-sensor mode (high-speed operation) and 2-sensor mode (constant-speed operation). A desired mode can be set using the “home-seeking mode” parameter. The operation sequence varies depending on the starting direction and position of home detection.

**Note** See “10.4 Stopping Motion” on page 50 for information on stopping motions before they are finished.

### • Commands and Parameters for Mechanical Home Seeking

Command/Parameter	Argument/Parameter Value	Function
MGHP	None	Start seeking mechanical home in the + direction
MGHN	None	Start seeking mechanical home in the – direction
HOMEofs	–8,388,607 to 8,388,607 (0)	Offset for mechanical home seeking [steps]
HOMEVS	1 to 500,000 (100)	Starting velocity [pps]
HOMEVR	1 to 500,000 (1000)	Running velocity [pps.]
HOMETR	0.001 to 1000 (0.5)	Acceleration time [sec.]
HOMEDIR	0, 1 (1)	Homing start direction for HOME input 0: - (CCW), 1: + (CW)
HOMESEL	0, 1 (1)	Homing type 0: 2-sensor mode, 1: 3-sensor mode
SLITEN	0, 1 (0)	Enable use of SLIT input 0: Disabled, 1: Enable
TIMEN	0, 1, 2 (0)	Enable use of TIM signal or encoder Z signal 0: Disabled, 1: Use TIM signal, 2: Use encoder Z signal

( ): default value

**Note** Mechanical home seeking normally uses starting velocity HOMEVS for the final approach to the home signal(s).  
The TIMING signal is based on position command (or set point).  
The MGHP and MGHN commands have higher priority than the HOMEDIR parameter until the device is RESET or power has been cycled.

### Example: Mechanical Home Seeking with the 3-sensor mode

Conditions:

Running velocity: 200 pps

Starting velocity: 50 pps

Starting direction: positive

Acceleration time: 0.1 sec.

Deceleration time: 0.1 sec.

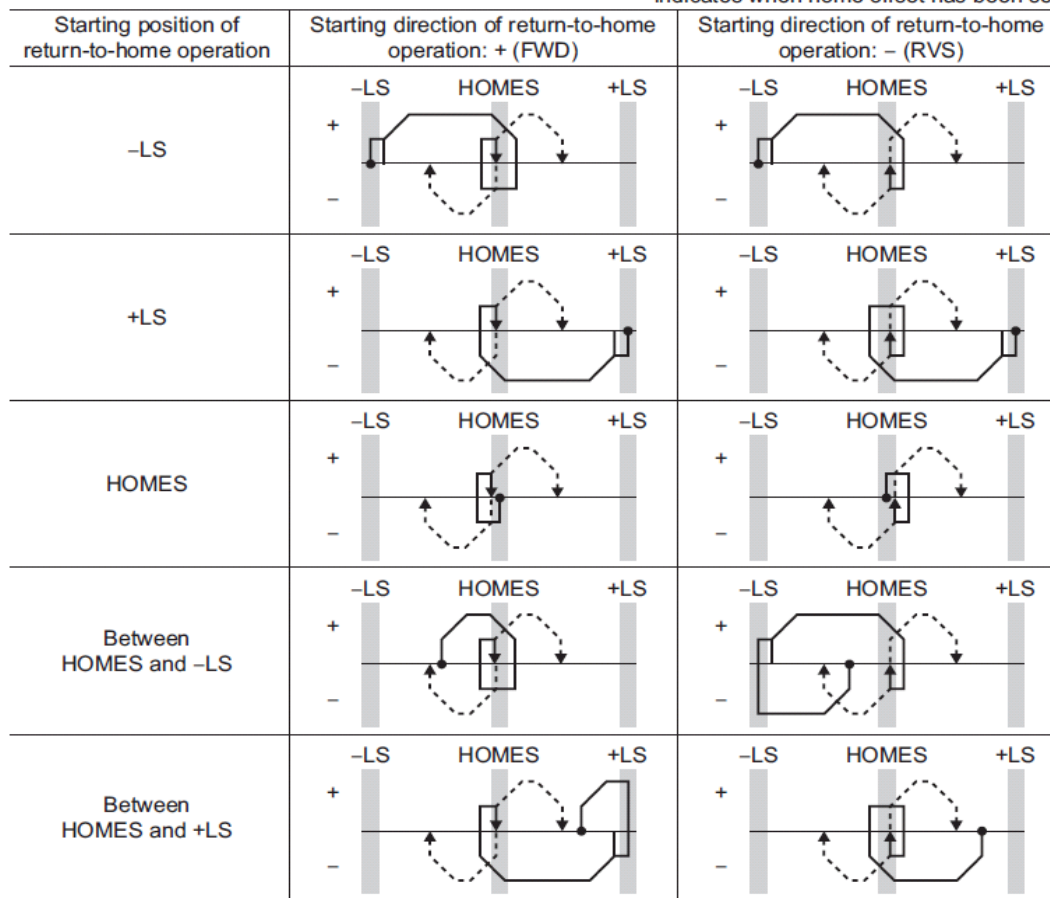
```
>HOMESEL=1
HOMESEL=1
>SLITEN=0
SLITEN=0
>HOMEVS=50
HOMEVS=50
>HOMEVR=200
HOMEVR=200
>HOMETR=0.1
HOMETR=0.1
>MGHP
>
```



### 3-sensor homing operation pattern

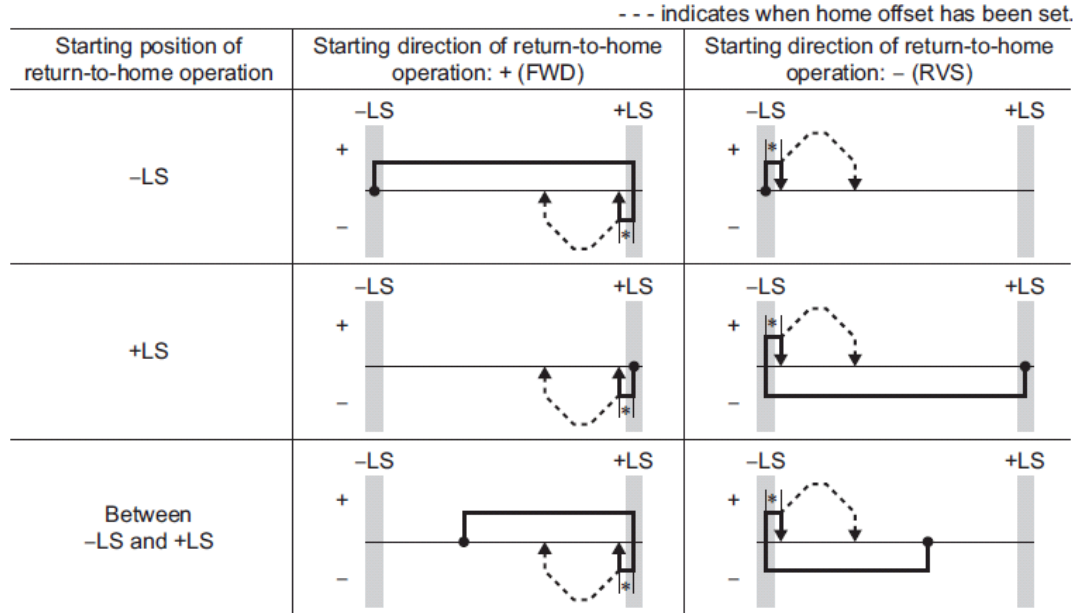
The home is detected using the three sensors of +LS, -LS and HOMES. The ON edge of HOMES defines the home. If the “SLIT detection with home-seeking” or “TIM signal detection with home-seeking” parameter is set, an AND gate will be applied to the ON edge of HOMES and the specified signal, thereby enabling more accurate home detection.

--- indicates when home offset has been set.



## 2-sensor homing operation pattern

The home is detected using +LS and -LS. When the motor pulls off of the limit sensor and both +LS and -LS turn OFF, the applicable position will be used to define the home. If the “SLIT detection with home-seeking” or “TIM signal detection with home-seeking” parameter is set, an AND gate will be applied to the OFF edge of +LS (or -LS) signal and the specified signal, thereby enabling more accurate home detection.



\* After backing off of the limit sensor, the equipment will move by the value set in the “backward steps in 2-sensor mode home-seeking” parameter (HOME2SB initial value: 200)

## 10.4 Stopping Motion

Commands and parameters for stopping motion are shown below.

### • Commands and Parameters for Stopping Motion

Command/Parameter	Argument/Parameter Value	Function
SSTOP	None	Soft stop: controlled deceleration over time
HSTOP	None	Hard stop: stop as quickly as possible
PSTOP	None	Panic stop: Hard stop, system state after stop is defined by ALMACT <sup>□</sup>
ABORT	None	Soft stop, abort sequence execution <sup>□</sup>
PAUSE	None	Pause motion (soft stop) <sup>□□</sup>
ABORTACT	0 to 1 (1)	0: Hard stop, Exit Sequence 1: Soft stop, Exit Sequence
ALMACT	1 to 2 (2)	1: Abort, Motor Current ON, Alarm ON 2: Abort, Motor Current OFF, Alarm ON

( ): default value

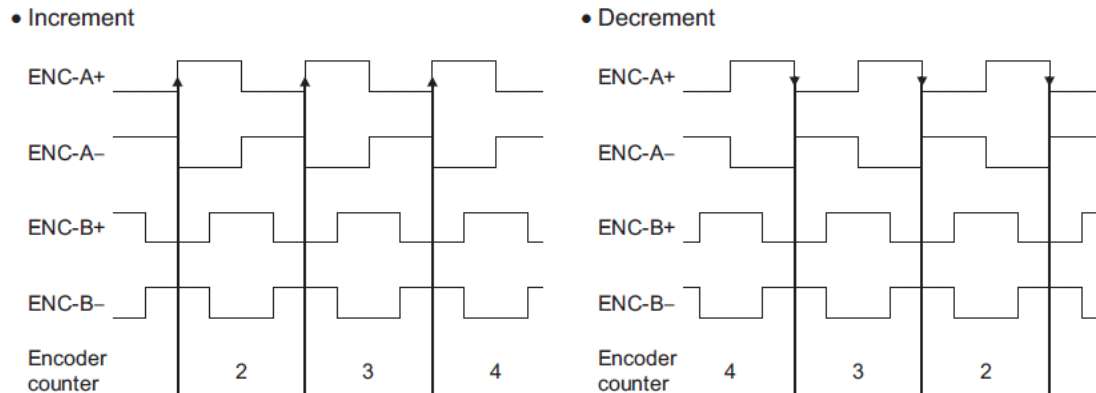
\* PAUSE can be assigned to an input port. ABORT and PSTOP have a dedicated input port.

#### Note

- The ESC key or character stops motion and aborts sequences, similar to ABORT.
- A motion that has been stopped with the PAUSE command can be continued (resumed) using the CONT command. See the entries for CONT and PAUSE in Chapter 13.
- See “9.1 Input signals” on page 33 for details on assigning signals to I/O ports.

## 10.5 Encoder input

- The driver counts up the 90° phase difference signal that is input from the encoder as a position feed-back signal. The encoder counter value can be read with a system parameter. The read value has been multiplied by 1.
- The encoder counter can be cleared to 0 by the system parameter. Also, a successful completion of return-to-home operation resets the encoder counter to 0.
- When an encoder is connected, the misstep detection function becomes available. Take note that the encoder input is counted even when the misstep detection function is not used.



This example assumes that the “motor rotation direction” parameter is set to “+ direction = CW”. If this parameter is set to “+ direction = CCW”, the counter value will decrease with each increment, and increase with each decrement.

## 10.6 Misstep Detection function

This function becomes effective when an encoder is connected. Specifically, the deviation between the command position and encoder counter is monitored. The sub-functions specified below become available when the “stepout detection” parameter is set to “enable”.

- Deviation error detection

When the deviation reaches the value set in the “stepout detection band” parameter (Initial Value: 7.2°), a deviation error will be recognized. If the base step angle of the motor is 0.72°, set the value of the “stepout detection band” parameter to 7.2°. Deviation error detection will start after the motor has remained excited for 500 msec. This function is disabled during return to mechanical home operation.

- STO output

This signal notifies a deviation error. Assign the STO output to one of OUT1 to OUT4 outputs.

- Alarm / warning

You can cause an alarm or warning to be generated upon detection of a deviation error.

- Generate an excessive position deviation alarm: Set the “stepout detection action” parameter to “alarm”.
- Generate an excessive position deviation warning: Set the “stepout detection action” parameter to “warning”.
- Do not generate an alarm or warning: Set the “stepout detection action” parameter to “no operation”.

- Command position update

The command position is corrected by the encoder counter while the motor is not excited. The command position will still be refreshed even when the motor output shaft is turned by an external force while the motor excitation is stopped.

## 10.7 Self Correcting function

This function becomes effective when an encoder is connected. Specifically, the deviation between the command position and encoder counter is monitored. The sub-functions specified below become available when the “self correcting” parameter is set to “enable”.

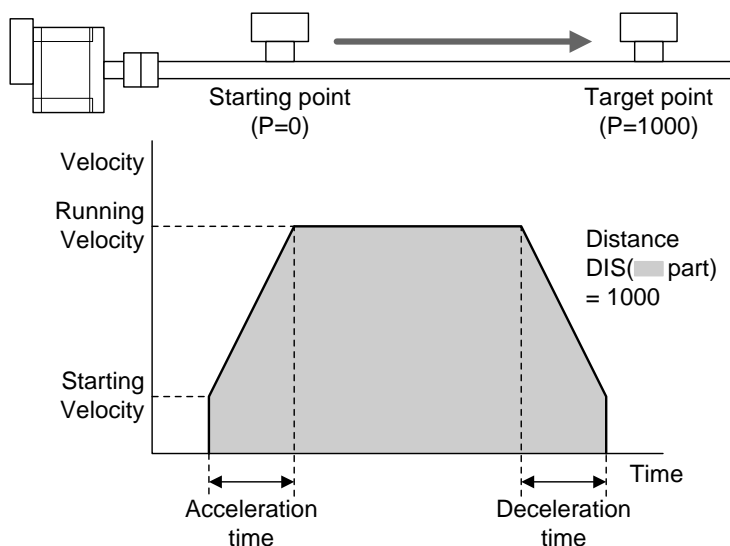
- Deviation error correction

When a misstep has occurred, the deviation error is corrected automatically.

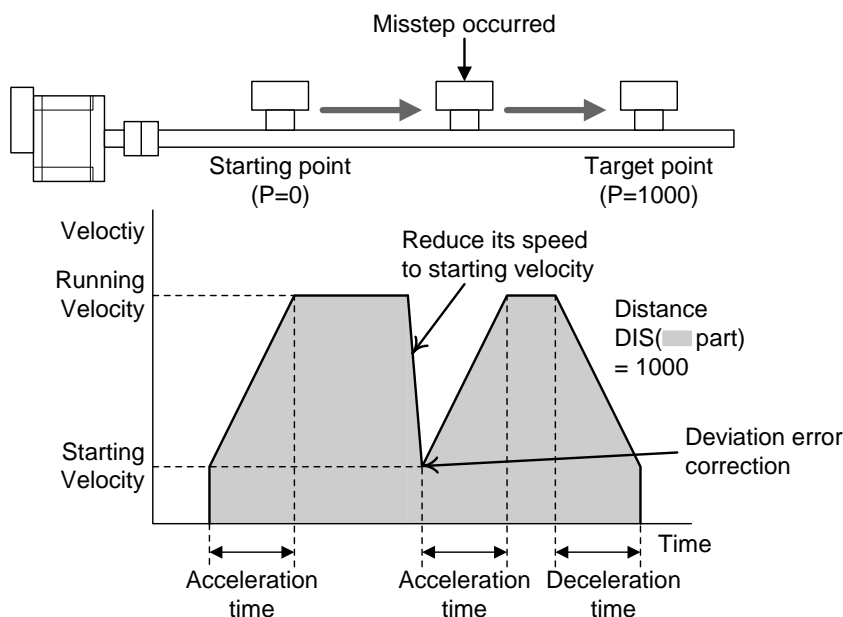
When a misstep is detected during motion, the driver will quickly reduce its speed to the “starting velocity” VS, correct the deviation error and then restart motion. By the time the motion is completed, the deviation error will have been corrected.

### Example Point-to-Point Motion

- Normal Operation



- Operation when a misstep occurred



In the case when the motor is at rest, if a misstep has occurred, the driver performs an operation to return to the original position with the motion profile that is set at that time. During this motion, the motor current is set to the value set by the “stop current” CRSTOP parameter while the driver performs the operation to return to the original position. However, if the command position value is out of the range (-8,388,607 to 8,388,607), the driver will not perform the operation to return to the original position.

The value of the deviation error used to determine when a misstep has occurred depends on motor type. The driver distinguishes the motor type being used by the values of the “encoder electronic gear B” EGB parameter and the “motor resolution” MRES parameter. If the base step angle of the motor is 0.72° and the “motor resolution” MRES parameter is “0” (the number of divisions is “1”), set the value of the “encoder electronic gear B” EGB parameter to “500”. If the base step angle of the motor is 0.36° and the “motor resolution” MRES parameter is “0” (the number of divisions is “1”), set the value of the “encoder electronic gear B” EGB parameter to “1000”. (See also “10.9 Encoder electronic gear settings on page 54.”) This function is disabled during return to mechanical home operation.

- SC output

This signal indicates that a deviation error was corrected automatically. The SC output will turn OFF when the next command to start motion is executed or the motor current is turned OFF. Assign the SC output to one of OUT1 to OUT4 outputs.

- Command position update

The command position is corrected by the encoder counter during the following conditions:

- The motor is not excited.
- The ABORT input is ON, with the exception of when the START input is in the ABORT position when it is set to act as a toggle switch (STARTACT=1), no motion commands will be executed.
- An alarm is present

The command position will still be refreshed even when the motor output shaft is turned by an external force during the above conditions.

Notes

- If the “self correcting” parameter is set to “enable”, the “stepout detection” parameter is set to “disabled” internally. In this case, the STO output will not turn ON even if the deviation between the encoder counter value and driver command position reaches the value set in the “stepout detection band” parameter. Also if the “stepout detection action” parameter is set to “alarm” or “warning”, an alarm or warning is not generated when the deviation error was detected.
- If a motion command is executed while a deviation error is corrected when a motor stops, the “motion while in motion” alarm will occur. Make sure motions are not started while the MOVE output is ON.

## 10.8 How to recover from a deviation error

Perform one of the following operations to recover from a deviation error if the self correcting function is not used:

When the “stepout detection action” parameter is set to none or warning:

- Stop the motor excitation.
- Perform a return to mechanical home operation.
- RESET the CRK Plus or cycle power.

When the “stepout detection action” parameter is set to “alarm”:

When a deviation error is detected, an excessive position deviation alarm will generate. In this case, reset the alarm by following the procedure below:

1. Stop the motor excitation or clear the counter to recover from the deviation error.
2. Turn the ALMCLR input ON to reset the alarm.
3. Perform a return-to-home operation, if necessary.

**Note:** If an excessive position deviation alarm generates, turning the ALMCLR input ON alone will not reset the alarm. Be sure to recover from the deviation error first, and then reset the alarm.

## 10.9 Encoder electronic gear settings

Even when the motor resolution is different from the encoder resolution, you can still detect a deviation error by setting the encoder electronic gears. The encoder electronic gears are used to determine a deviation error and will not affect the encoder counter value.

Parameter	Description
Encoder electronic gear A (EGA)	Set the encoder resolution. Set to 500 if the encoder pulse count per motor revolution is 500 P/R.
Encoder electronic gear B (EGB)	Set the motor resolution. Set to 1000 if the pulse count required for one motor revolution is 1000 P/R.

Setting Examples

### For Standard Type Step Motors with a base step angle of 0.72°/step (500 steps/rev) (CRK5□ types)

Step angle (Steps/rev)	# of divisions	EGA parameter	EGB parameter	Step angle (Steps/rev)	# of divisions	EGA parameter	EGB parameter
0.72° (500)	1	500	500	0.0288° (12,500)	25	500	12500
0.36° (1000)	2	500	1000	0.018° (20,000)	40	500	20000
0.288° (1250)	2.5	500	1250	0.0144° (25,000)	50	500	25000
0.18° (2000)	4	500	2000	0.009° (40,000)	80	500	40000
0.144° (2500)	5	500	2500	0.0072° (50,000)	100	500	50000
0.09° (4000)	8	500	4000	0.00576° (62,500)	125	500	62500
0.072° (5000)	10	500	5000	0.0036° (100,000)	200	500	100000
0.036° (10,000)	20	500	10000	0.00288° (125,000)	250	500	125000

### For High Resolution Type Step Motors with a base step angle of 0.36°/step (1000 steps/rev) (CRK5□M types)

Step angle (Steps/rev)	# of divisions	EGA parameter	EGB parameter	Step angle (Steps/rev)	# of divisions	EGA parameter	EGB parameter
0.36° (1000)	1	1000	1000	0.0144° (25,000)	25	1000	25000
0.18° (2000)	2	1000	2000	0.009° (40,000)	40	1000	40000
0.144° (2500)	2.5	1000	2500	0.0072° (50,000)	50	1000	50000
0.09° (4000)	4	1000	4000	0.0045° (80,000)	80	1000	80000
0.072° (5000)	5	1000	5000	0.0036° (100,000)	100	1000	100000
0.045° (8000)	8	1000	8000	0.00288° (125,000)	125	1000	125000
0.036° (10,000)	10	1000	10000	0.0018° (200,000)	200	1000	200000
0.018° (20,000)	20	1000	20000	0.00144° (250,000)	250	1000	250000

#### Note

- The accuracy of deviation error detection is also affected by the encoder resolution and assembly accuracy. In addition, this accuracy varies depending on the operating speed and load. Accordingly, always check the accuracy of deviation error detection on the actual machine.
- If misstep occurs, the home position on the equipment side deviates from the home position recognized by the driver. If the operation is continued in this condition, the equipment may be damaged. Accordingly, take prompt actions if misstep is detected.
- If the motor step angle has been changed, be sure to change the value of the “encoder electronic gear B” parameter accordingly. Similarly if the encoder resolution has changed, be sure to change the value of the “encoder electronic gear A” parameter accordingly. If the gears are not set properly, the command position will not be updated correctly and a deviation error will be detected.

## 10.10 Encoder Resolution

If the misstep detection function or the self correction function is not used, the encoder resolution will not be limited in any way. Just connect an encoder meeting the required specification. If the misstep detection function or self correction function is used, use of an encoder with a resolution of 500 P/R is recommended for motors whose basic step angle is  $0.72^\circ$  (1000 P/R for motors whose basic step angle is  $0.36^\circ$ .)

## 10.11 Support Functions

### ■ Teaching Positions

The CRK Built-in Controller (Stored Program) device includes a position data array which can hold up to 64 pre-defined positions. Once defined, these positions can be used as targets for point-to-point motions. The positions are referenced as POS[1] through POS[64].

#### • Example

>MA POS [1] #Start absolute motion to the position specified by POS[1].

>W=POS [64] #Assign the value of POS[64] to variable W. (This statement is valid in a sequence only)

The position array data can be entered manually, but the system also provides a utility for "teaching" the positions using the TEACH command. The TEACH command starts the teaching process. While the teaching process runs:

- The system monitors and displays the system position command by pressing the "U" key
- Motor current can be toggled on and OFF if encoder is used and the stepout detection is set to enable (STOEN=1)

- If motor current is ON, the system can be commanded to move, positively or negatively, in increments of 1 step, or continuously at running velocity VR

- If motor current is OFF, the system can be repositioned using external force or torque

- At any time, the system position can be stored to any location in the position data array.

#### Note

- Motions use starting velocity VS, running velocity VR, and acceleration and deceleration times TA and TD.
- The position data array is not stored to EEPROM automatically; it must be saved using the (S) "save" key while teaching, or with the SAVEPOS command.
- The teach function is not available while a sequence is being executed, or motion is in progress. While teaching, sequences may not be executed and only the PSTOP, +LS, -LS, and CROFF inputs are acknowledged, if they are configured as inputs.

#### • Key functions, while TEACHing

Key	Function
V	Move continuously, in the negative direction (while key pressed). Soft stop when key is released.
B	Move negatively by one step.
N	Move positively by one step.
M	Move continuously, in the positive direction (while key pressed).Soft stop when key is released.
Q	Toggle current OFF and ON (Encoder required)
K	Set key interval detection time [millisecond]
U	Update position display
<SPACE>	Hard stop
<ESC>	Soft stop, terminate TEACHing
<Enter>	Store current position to a location in the position data array (System will prompt for location, 1-64)

#### Memo

- While teaching, continuous motions proceed while the V or M keys are pressed. The system stops the motor (over deceleration time TD) when it has not detected a key for the "key interval detection time". The key interval detection time can be adjusted. Smaller values make the system react quicker, but may result in "stuttering": motions may start and stop in a pulsing pattern. Larger values reduce the chance of stuttering, but take more time to react: controlling the final rest position is less accurate.
- Responsiveness is also very dependent on the host controller (e.g. PC or terminal) and its keyboard settings.
- Toggling current (with 'Q') is only recommended while the motor is stopped. A "current OFF" toggle may not be honored if a 'Q' character is sent within a stream of motion characters ('V', 'B', 'N', 'M').

- Example

```

***Teach mode***

(V)      : Move Cont. Neg.          (M): Move Cont. Pos.
(B)      : Move Incr. -1           (N): Move Incr. +1
(Q)      : Toggle Current ON/OFF (Encoder Required)
(K)      : Change Key Interval (50-500 [msec])
<Space>  : Immediate stop
<Enter>  : Data entry mode (Input POS number, then <Enter>)
<ESC>    : Exit teach mode

PC=      0  ←————— Current position

```

After TEACH command

```

***Teach mode***

(V)      : Move Cont. Neg.          (M): Move Cont. Pos.
(B)      : Move Incr. -1           (N): Move Incr. +1
(Q)      : Toggle Current ON/OFF (Encoder Required)
(K)      : Change Key Interval (50-500 [msec])
<Space>  : Immediate stop
<Enter>  : Data entry mode (Input POS number, then <Enter>)
<ESC>    : Exit teach mode

PC=      15410 Save to POS [1]  ←————— Set target position data
                                         array location: Input "1"

```

After &lt;ENTER&gt; received while teaching

```

***Teach mode***

(V)      : Move Cont. Neg.          (M): Move Cont. Pos.
(B)      : Move Incr. -1           (N): Move Incr. +1
(Q)      : Toggle Current ON/OFF (Encoder Required)
(K)      : Change Key Interval (50-500 [msec])
<Space>  : Immediate stop
<Enter>  : Data entry mode (Input POS number, then <Enter>)
<ESC>    : Exit teach mode

PC=      15410 Save to POS [1] Data set OK.

```

After &lt;ENTER&gt; received

```

***Teach mode***

(V)      : Move Cont. Neg.          (M): Move Cont. Pos.
(B)      : Move Incr. -1           (N): Move Incr. +1
(Q)      : Toggle Current ON/OFF (Encoder Required)
(K)      : Change Key Interval (50-500 [msec])
<Space>  : Immediate stop
<Enter>  : Data entry mode (Input POS number, then <Enter>)
<ESC>    : Exit teach mode

PC=      15410 Save to POS [1] Data set OK.
End teach mode  ←————— <ESC>

```

After &lt;ESC&gt; received



## ■ I/O Test

The CRK Built-in Controller (Stored Program) device provides a utility to help confirm proper I/O operation. OUTTEST starts a utility process to check I/O connections and levels. Inputs are continuously monitored and displayed, and outputs can be set or cleared, to confirm proper external connections.

Inputs and outputs are displayed as active (1) or inactive (0).

OUTTEST temporarily disables the actions of all assigned system input and output signals. The system will not react to inputs, and will not automatically control outputs. All output control is from the serial port.

Signal assignments are restored when the OUTTEST process terminates, and all outputs are restored to the state they were in when the OUTTEST process was started.

Outputs can be toggled, using the character displayed next to the signal name in the OUTTEST output.

Toggling an output changes its state as displayed, and changes the electrical state of the associated output port.

Toggle keystrokes or characters for each output are:

OUT1 (1)	OUT2 (2)	OUT3 (3)	OUT4 (4)	MOVE (M)
RUN (R)	AREA (E)	READY (D)	HOMEP (H)	ALM (A)
PSTS (P)	TEMP (T)	WNG (W)	STO (S)	SC (C)

A SPACE key sets all outputs to inactive (0).

An <ESC> key or character exits the OUTTEST process.

### Note

- Only keys for assigned output signals are available.
- OUTTEST is not permitted while a sequence is running, while a motion is in progress, or if the system is in an alarm state. While OUTTEST is running, sequences are not executable.

### • Example

```

*** Input Monitor -- Output Simulator ***
Input   : IN1 IN2 IN3 IN4 IN5 IN6 PAUSE PAUSECL
          : START ABORT ALMCLR CROFF HOME +LS -LS PSTOP SENSOR HOMES SLIT
Output  : ALM(A) MOVE(M) OUT1(1) OUT2(2) OUT3(3) OUT4(4) HOMEP(H) STO(S)

- Use (x) keys to toggle Outputs.
- Use <space> to set all outputs to zero.
- Use <esc> to exit OUTTEST mode.
- Use <enter> to update inputs status

--Inputs---      --- Dedicated IO signals ----      Outputs
1 2 3 4 5 6 -      (SEQ#)- S A A C N + - P S H S      -- A M 1 2 3 4
0 0 0 0 0 0 -      ( 0)- 0 0 0 0 0 0 0 0 0 0      -- 0 0 0 0 0 0

```

## 10.12 Protective Functions

The CRK Built-in Controller (Stored Program) device constantly monitors system conditions to detect potentially harmful conditions, such as overheating and over voltage. For some alarm conditions, the action(s) taken when the condition is detected can be controlled by ALMACT, to suit the application.

### • Alarm conditions affected by ALMACT

Condition	Description	Alarm Code
Hardware over travel	Positive or negative position limit signal detected	0x66
Software over travel	Position outside of programmed positive and negative position limits LIMP and LIMN	0x67
Panic stop	System executed a panic stop because of a PSTOP input or command	0x68

ALMACT controls the system response when any of the alarm conditions (above) are detected.

ALMACT	Action
1	Abort sequences and stop motion. Motor current ON, Alarm ON.
2	Abort sequences and stop motion. Motor current OFF, Alarm ON. □ default)

**Note** See “12.8 Error Messages” on page 76 for details of each alarm condition and system response.

The system can also be configured to automatically transmit a message when alarms or warnings are detected. Automatic message transmission is controlled by ALMMSG:

ALMMSG	Action
0	Do not automatically transmit alarm and warning messages (default)
1	Automatically transmit messages for alarms, but not warnings
2	Automatically transmit messages for alarms and warnings

**Note**

- See “12.8 Error Messages” on page 76 for message details
- Warnings are for informational purposes only, and do not effect system operation.

### ■ Alarm history

The ALM command shows the current alarm status, and the last 10 alarms.

Example

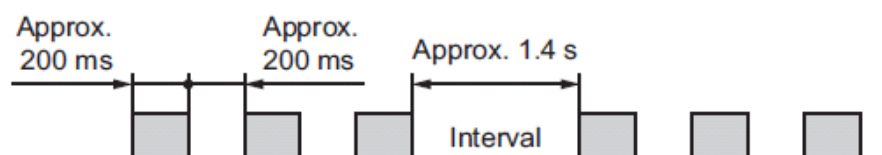
```
>ALM
ALARM = 30 ,   RECORD: 23 23 30 30 30 23 23 10 23 23.
>
```

**Note** The alarm history is automatically saved in non-volatile EEPROM, as a troubleshooting aid (warnings are not saved). The EEPROM has a nominal expected lifetime of 100,000 write cycles. Alarm conditions should be treated as exceptional, and not generated routinely by an application, if they could possibly occur at high frequency.

### ■ ALARM LED function

When an alarm generates, the ALM output will turn ON and the motor will either stop or be controlled by ALMACT. When an alarm generates, the ALARM LED will blink. The cause of the alarm can be checked by counting the number of times the ALARM LED blinks.

Example: Overvoltage alarm (number of blinks: 3)



# 11 Control via RS-485 communication

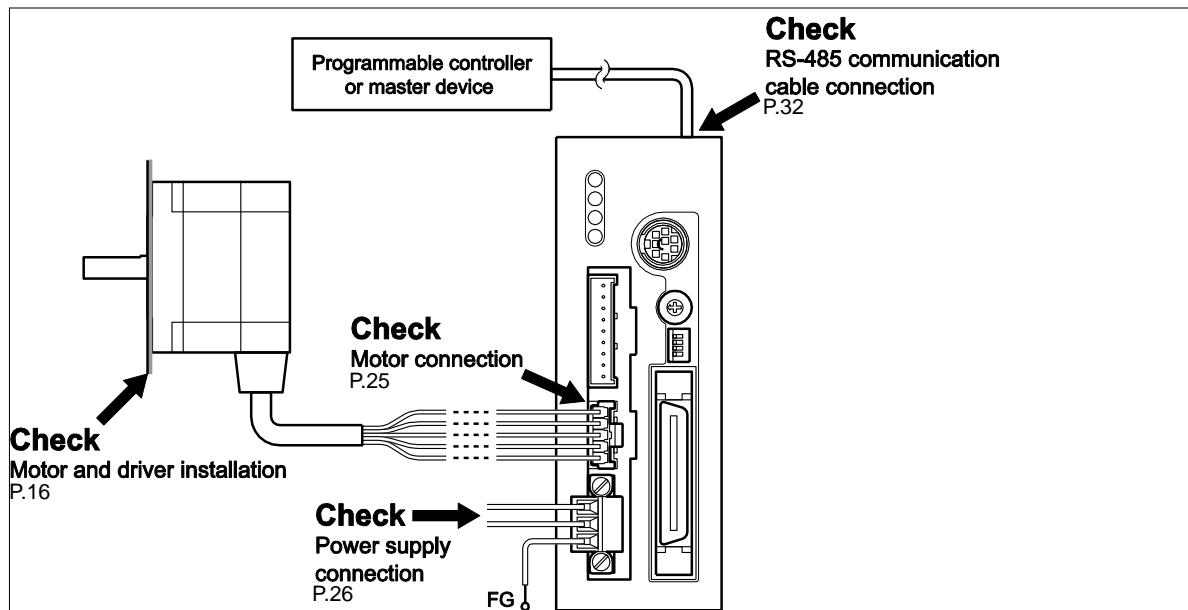
This chapter explains how to control the system using a programmable controller via RS-485 communication.

## 11.1 Guidance

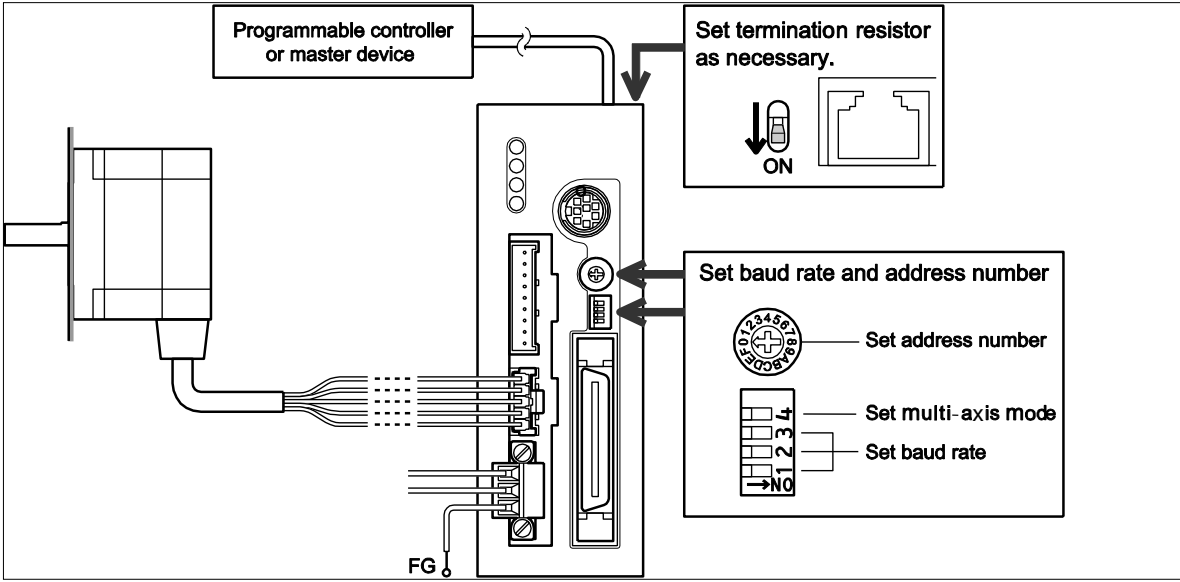
If you are new to the CRK Series built-in controller (Stored Program), read this section to understand the operating methods along the operation flow.

**Note** Before operating the motor, check the condition of the surrounding area to ensure safety.

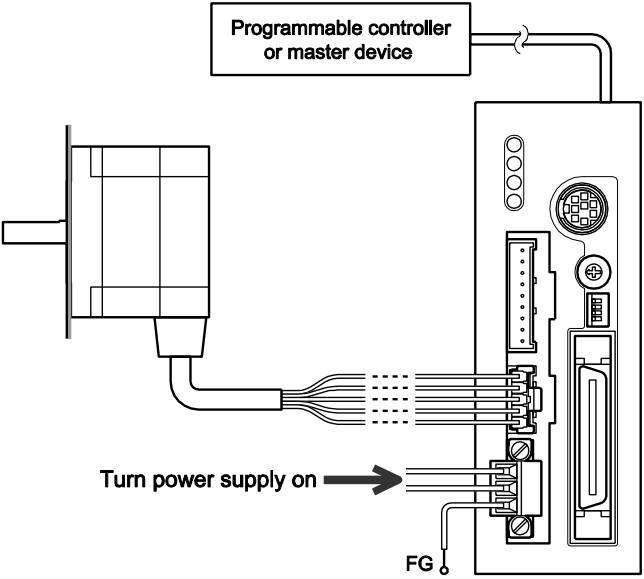
### STEP 1 Check the installation and connection



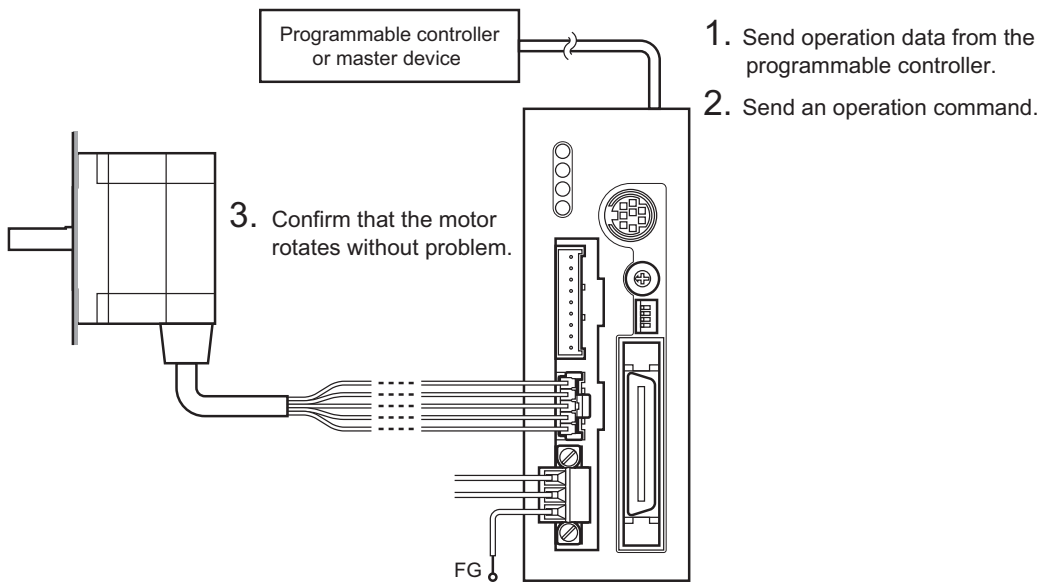
**STEP 2**    Set the switches



**STEP 3**    Turn on the power and set the parameters



## STEP 4 Operate the motor



## STEP 5 Were you able to operate the motor properly?

How did it go? Were you able to operate the motor properly?

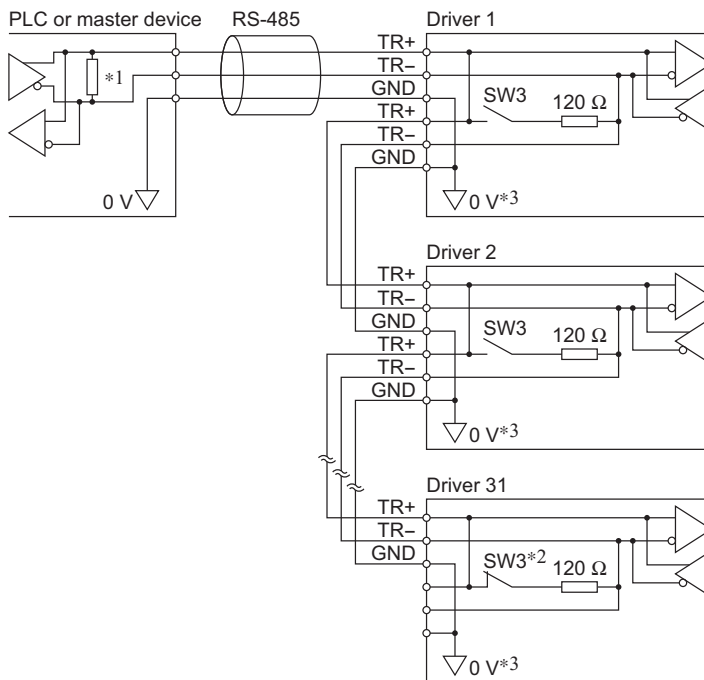
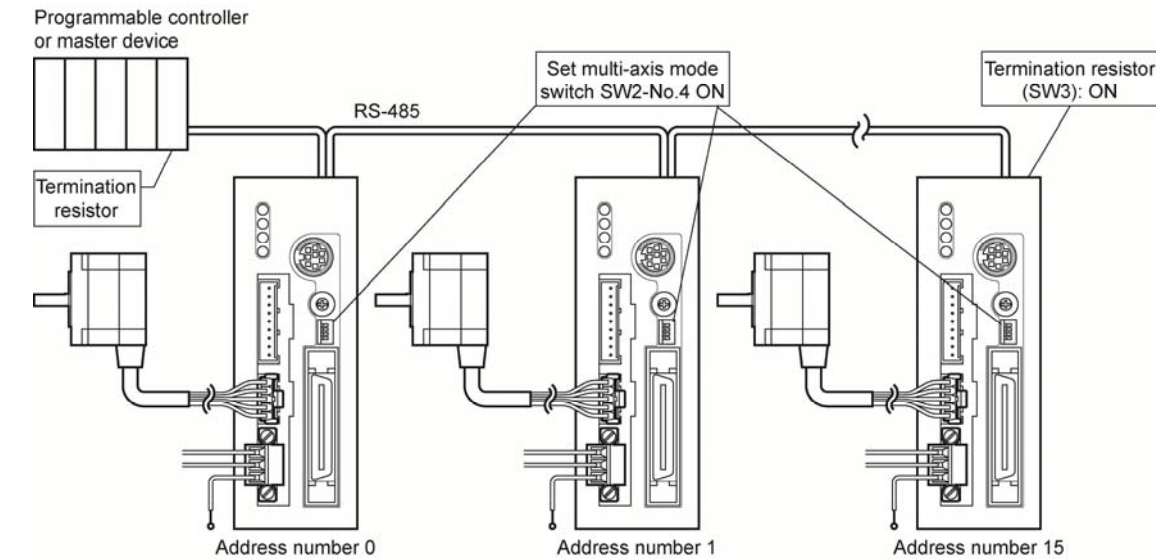
If the motor does not function, check the following points:

- Is an alarm present?
- Are the power supply, motor and RS-485 communication cables connected securely?
- Are the address number, baud rate and termination resistor set correctly?
- Is the C-ERR LED lit?
- Is the C-DAT LED lit?

## 11.2 Communication specifications

Electrical characteristics	In conformance with EIA-485 Use a twisted pair cable (TIA/EIA-568B CAT5e or higher is recommended) and keep the total wiring distance including extension to 50 m (164 ft.) or less.
Transmission mode	Half duplex
Baud rate	Selectable from 9600 bps, 19200 bps, 38400 bps, 57600 bps, 115200 bps
Physical layer	Asynchronous mode (8 bits, 1 stop bit, no parity)
Protocol	TTY (CR+LF)
Connection pattern	Up to 16 drivers can be connected to one programmable controller (master device).

### ■ Connection Example (for multi-axis)



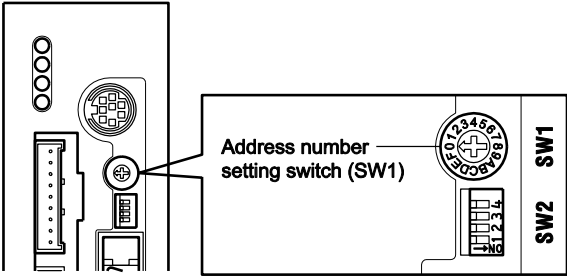
- \*1 Termination resistor 120 Ω
- \*2 Turn the termination resistor (SW3) to ON.
- \*3 The GND line is used in common with CN1 (not insulated).

# 11.3 Setting the switches

**Note** Be sure to turn OFF the driver power before setting the switches. If the switches are set while the power is still on, the new switch settings will not become effective until the driver power is cycled.

## ■ Address number

Set the address number using the address setting switch (SW1).  
 Make sure each address number you set for each driver is unique.  
 Factory setting SW1: 0 (address number 0)



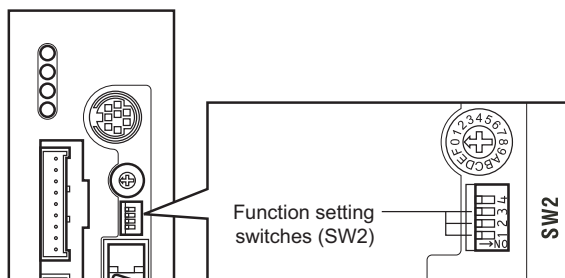
Address number	SW1
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F

## ■ Baud rate

Set the baud rate using SW2-No.1 to SW2-No.3 of the function setting switch (SW2).

The baud rate to be set should be the same as the baud rate of the programmable controller (master device).

Factory setting All OFF (9600 bps)



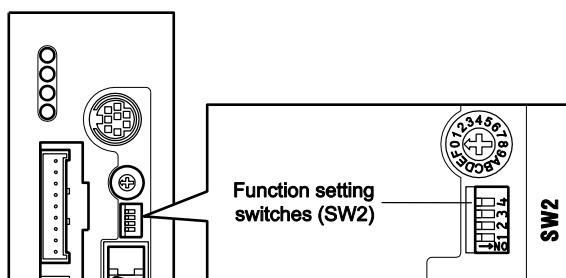
Baud rate (bps)	SW2-No.3	SW2-No.2	SW2-No.1
9600	OFF	OFF	OFF
19200	OFF	OFF	ON
38400	OFF	ON	OFF
57600	OFF	ON	ON
115,200	ON	OFF	OFF
115,200	ON	OFF	ON
115,200	ON	ON	OFF
115,200	ON	ON	ON

## ■ Multi-axis mode switch

Set the driver to single axis or multi-axis mode using function switch SW2-No.4.

When using more than 1 driver with a multi-drop connection, be sure to set SW2-No.4 to ON to enable the multi-axis mode.

Factory setting OFF (single axis mode)



SW2-No.4	Multi-axis mode
OFF	Disabled
ON	Enabled

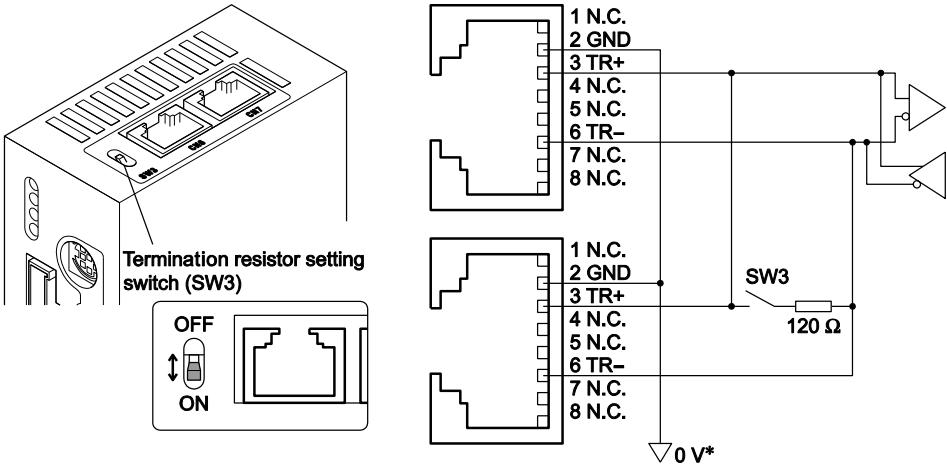


■ Termination resistor

Use a termination resistor for the driver located farthest away (positioned at the end) from the programmable controller (master device).

Turn the termination resistor setting switch (SW3) ON and set the termination resistor for RS-485 communication (120 Ω).

Factory setting      OFF (termination resistor disabled)



\* The GND line is used in common with CN1 (not insulated).

SW3	Termination resistor (120 Ω)
OFF	Disabled
ON	Enabled

# 12 Program Creation and Execution

This chapter explains the methods used to create new programs, edit existing programs and execute programs.

## 12.1 Overview of Operation

Commands and programs are created by entering commands and parameters from a terminal program. You can choose one of three operating modes (monitor mode, program-edit mode and sequence mode) to begin a desired task from a terminal.

### ■ Operation from Terminal (Monitor Mode)

Operation from the terminal is available when the device's power is input.

When operating from the terminal, you can create, delete, copy, lock and execute sequences. Additionally, motion can be started, stopped and the status of the device and I/O signals can be monitored.

### ■ Sequence Editing

- Sequences can be edited from the terminal

In this chapter, "Editing from the terminal" is explained.

The system enters this mode when "EDIT" is entered from the terminal.

In the sequence-edit mode, you can edit a sequence by changing, inserting or deleting specified lines. You can also perform a syntax check.

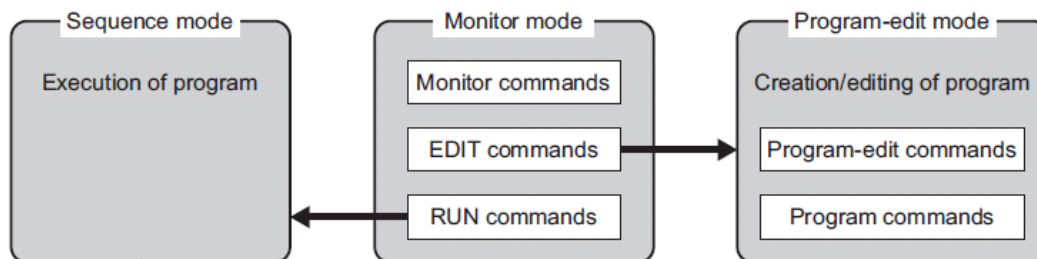
### ■ Executing Sequences

Sequences can be executed by either,

- Using the "RUN" command from the terminal.
- From I/O using the "START" and "INx" inputs.
- A sequence named "CONFIG" will be run automatically upon power-up or device RESET.

Sequence execution ends when any of the following conditions are satisfied:

- The END command or ABORT command written in the sequence is executed
- The PSTOP or ABORT input is turned ON
- The ESC key is pressed
- An alarm has been detected.



### ■ CRK Motion Creator GUI

The CRK Motion Creator GUI also includes a motion creating function, a sequence editing function, a terminal function, a data save/load function, and a system parameter setting function.

The latest version of the CRK Motion Creator is available for download free at

<http://www.orientalmotor.com/support/software/software.html>

In this manual, the CRK device is operated with terminal software. The same operation is possible when using the CRK Motion Creator terminal function.

**Note** The CRK Motion Creator program is not included in the CRK5xx-KP package.

## 12.2 Communication and Terminal Specifications

Please set up the terminal program used when creating a program to the following specifications.

### ■ Communication Specification

Item	Description
Electrical Characteristics	In conformance with EIA-485
Transmission method	Start-stop synchronous method, NRZ (Non-Return Zero), half-duplex
Data length	8 bits, 1 stop bit, no parity
Transmission speed	Selectable: 9600, 19200, 38400, 115,200 bps (as selected by SW2).
Protocol	ASCII (TTY + CRLF)
Connector specifications	RJ-45

### ■ Terminal Specifications

- ASCII mode
- VT 100 compatible recommended
- Handshake: None
- Transmission CR: C-R
- Word wrap: None
- Local echo: None
- Beep sound: ON

## 12.3 Communication Mode

This product uses the communication mode where one programmable controller acts as master (or host) and this driver serves as the slave. The communication can operate in a single axis mode or multi-drop mode using standard ASCII data transfer. The length of data is varies depending on the command or response type.

In single-axis mode (SW2-No.4=OFF), the driver will always respond to the host and no ID addressing from the host is required. The response data is terminated by a new line and a prompt ">". The driver is then ready to receive the next command.

In multi-axis mode (SW2-No.4=ON), only the driver which has the ID address setting matching the latest host command ID addressing (@id) will response to the master. The same is also applied to global commands. (id= 0-9, A-F).

When a device has been selected, it remains selected. The device changes its command line prompt to show it's ID. If a device with an ID=A is selected, the prompt changes from ">" to "A>". All commands will be processed by that device, until another @ prefix is sent with a different ID.

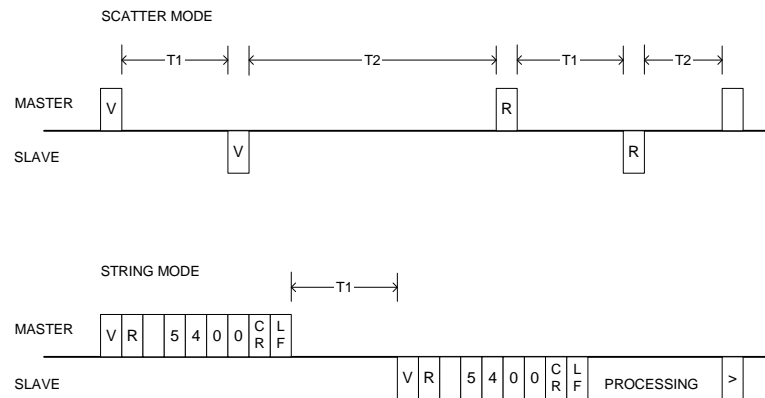
Data can be sent from the master one byte at a time like data entered from a keyboard or it can be sent as a string. The received data is echoed back to the master (if ECHO=1) as soon as the bus line becomes inactive for a predefined idle time interval.

**Note:** Special case:

- ID addressing (@id)

When a prefix "@" character is received by the slave, it will not be echo immediately until a valid and assigned id value is received. The new selected unit will then send both characters "@" and the id back to the host.

## 12.4 Communication Timing



-  $T_1$  :Idle time\* (Slave to master).

After received data from the host, the selected driver will process data immediately as it's entered. The return data to the host is buffered and delay until the bus becomes inactive for at least idle time interval.

-  $T_2$  :Delay time (Master to slave)

Depends on what kind of data that the host expected from the slave. If the host determined a data byte that it received from the slave is the last byte such as a single byte echoing or a prompt, the host can start sending new data as soon as 50 microseconds later.

Idle time\*:

The idle time is a minimum wait time required by the slave starting from the last character received to the time that the slave can initiate a transmission on its transmit pending data.

The idle time is fixed at 5 characters time length but limited to 1 millisecond minimum.

Idle time =  $(1 / \text{baud rate}) * 10 \text{ bits} * 5 \text{ characters}$  -> (idle time  $\geq 1\text{mS}$ )

**CAUTION:**

To avoid communication error such as master and slave transmitting data simultaneously, the master should not send command with character spacing between 4 to 6 character times (idle time  $\pm 1 \text{ char time}$ ).

## 12.5 Creating a New Sequence

Programs contain data with which to define device operation, such as the operating velocity and travel distance. When a sequence is started, the commands included in the sequence are executed sequentially. Sequences are stored in the device's memory. Program must adhere to the following specifications.

### ■ Sequence Specifications

Maximum number of programmable sequences	64 sequences (Name is user configurable)
Maximum sequence size	1.6KB maximum for total compiled sequences 4.2KB maximum (text + compiled)
Sequence execution by external input	START input executes a sequence selected by binary combination of IN1 to IN6.
Maximum number of selectable sequences by external input	64 sequences (0 to 63). Depending on INx assignment.
Automatic sequence execution at power up or after a RESET.	Sequence named CONFIG is executed at power up.
Sequence program name	10 characters maximum. 0 to 9, A to Z, a to z, can be used as characters. Using driver's command and/or parameter names for sequence names can cause confusion, and is not recommended. If sequence is saved by name, system assigns sequence number within 0 to 63. Assigned number is used for selection to start sequence by I/O.

#### Note

Device memory status can be checked either by the "DIR" command from the terminal or by the "M" command while editing a sequence.

## ■ Creating a New Sequence Example

1. Connect the device to the terminal.
2. Enter the terminal command “EDIT \*” (\* indicates the sequence name).  
Insert a space between “EDIT” and the sequence name.  
When the command is entered, a message indicating a blank sequence (New sequence) is displayed.  
Enter “I” (Insert).  
Subsequently, “(1)” is displayed and the system enters the sequence-edit mode.  
You can now create a sequence.

```
>EDIT SAMPLE1
New Sequence
Sequence Name   : SAMPLE1
Sequence Number : 0
Lines          : 0
Bytes          : 0
Bytes Free      : 1600
>>Command: I
( 1)_
```

3. Enter commands and parameters by referring to 13, “Command List,” to create a program.  
The following shows a sample program. This program, SAMPLE1, executes an incremental positioning operation at a starting velocity of 500 pps and operating velocity of 1000 pps, with a distance of 2500 steps.  
Insert a space or equal sign between the command and the parameter. See “Command Format” on page 317 as a reference.

```
>EDIT SAMPLE1
New Sequence
Sequence Name   : SAMPLE1
Sequence Number : 0
Lines          : 0
Bytes          : 0
Bytes Free      : 1600
>>Command: I
( 1) VS=500
( 2) VR=1000
( 3) DIS=2500
( 4) MI
( 5)_
```

4. When the program entry is complete, press the Enter key and enter “S” to save the program.  
The program is saved in the memory and a syntax check is performed. When an error in syntax is found, the line number on which the error was found is displayed together with the nature of the error.  
Finally, enter “Q” to complete the program and exit edit mode.

```
Bytes          : 0
Bytes Free      : 1600
>>Command: I
( 1) VS=500
( 2) VR=1000
( 3) DIS=2500
( 4) MI
( 5)_
>>Command: S
Compiling...OK
Saving.....OK
>>Command: Q
>_
```

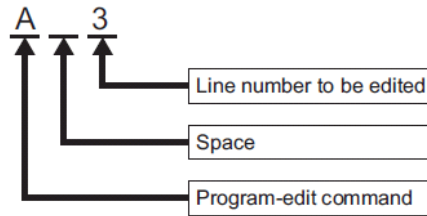
## 12.6 Editing an Existing Sequence

In the sequence-edit mode, existing sequences can be edited by alter inserting and deleting lines. The method used to enter commands is the same as when creating a new sequence.

1. Enter the monitor command “EDIT □” (□ indicates the sequence name or number).  
Insert a space between “EDIT” and the sequence name (or number).  
The system enters the sequence-edit mode.

```
>EDIT PROGRAM1
Sequence Name : PROGRAM1
Sequence Number : 1
Lines : 5
Bytes : 23
Bytes Free : 1577
>>Command: _
```

2. Enter a sequence-edit command and a line number according to the edit operation you wish to perform.  
Insert a space between the sequence-edit command and the line number.

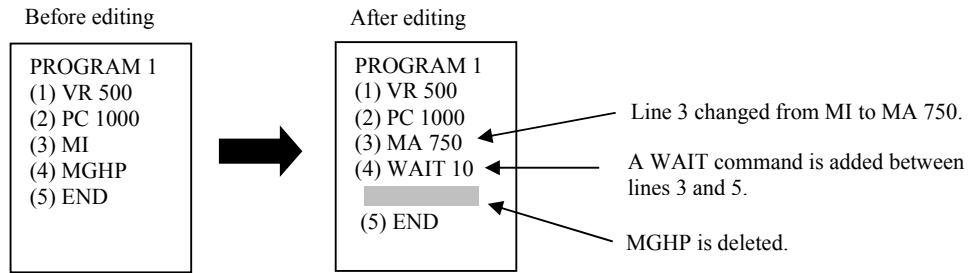


Command	Description
A	Alter (change)
D	Delete
I	Insert
X	Cut
P	Paste
C	Copy
S	Save, Compile
Q	Quit
H	Display help
L	List
M	Display memory status

**Note** | The “H” command will show the command help (above list) while editing a sequence.

## ■ Line Editing Example

This section explains the steps to edit PROGRAM1 as follows:



1. Enter “EDIT PROGRAM1” and press the Enter key.  
After the contents of Program1 are displayed, “>>Command:” is displayed and the monitor waits for editing input.

```
>EDIT PROGRAM1
Sequence Name   : PROGRAM1
Sequence Number : 1
Lines          : 5
Bytes          : 23
Bytes Free     : 1577
>>Command: _
```

2. Enter “L” to list the entire sequence, make sure which line to edit.

```
>EDIT PROGRAM1
Sequence Name   : PROGRAM1
Sequence Number : 1
Lines          : 5
Bytes          : 23
Bytes Free     : 1577
>>Command: L
( 1) VR=500
( 2) PC=1000
( 3) MI
( 4) MGHP
( 5) END
>>Command: _
```

3. Change line 3 from “MI” to “MA 5” using the following steps:
  - a. Enter “A 3” and press the Enter key.  
Line 3 becomes editable.

```
>EDIT PROGRAM1
Sequence Name   : PROGRAM1
Sequence Number : 1
Lines          : 5
Bytes          : 23
Bytes Free     : 1577
>>Command: A 3
( 3) MI_
```

- b. Delete “MI” with the Backspace key.

```
>EDIT PROGRAM1
Sequence Name   : PROGRAM1
Sequence Number : 1
Lines          : 5
Bytes          : 23
Bytes Free      : 1577
>>Command: A 3
( 3) _
```

- c. Enter “MA 750”.

```
>EDIT PROGRAM1
Sequence Name   : PROGRAM1
Sequence Number : 1
Lines          : 5
Bytes          : 23
Bytes Free      : 1577
>>Command: A 3
( 3) MA 750 _
```

- d. Press the Enter key.

Line 3 of PROGRAM1 is changed to “MA 750.” The command prompt is displayed and the monitor waits for the next program-edit command.

```
>EDIT PROGRAM1
Sequence Name   : PROGRAM1
Sequence Number : 1
Lines          : 5
Bytes          : 23
Bytes Free      : 1577
>>Command: A 3
( 3) MA 750 _
>>Command: _
```

4. Insert “WAIT 10” below line 3 using the following steps:

- a. Enter “I 4” and press the Enter key.

Line 4 is added, and the monitor waits for a command.

```
>EDIT PROGRAM1
Sequence Name   : PROGRAM1
Sequence Number : 1
Lines          : 5
Bytes          : 23
Bytes Free      : 1577
>>Command: A 3
( 3) MA 750 _
>>Command: I 4
( 4) _
```



b. Enter “WAIT 10”.

```
>EDIT PROGRAM1
Sequence Name   : PROGRAM1
Sequence Number : 1
Lines          : 5
Bytes          : 23
Bytes Free      : 1577
>>Command: A 3
( 3) MA 750_
>>Command: I 4
( 4) WAIT 10_
```

c. Press the Enter key.

“WAIT 10” is added to line 4 of PROGRAM1.

You will now insert a new line at line 5.

```
>EDIT PROGRAM1
Sequence Name   : PROGRAM1
Sequence Number : 1
Lines          : 5
Bytes          : 23
Bytes Free      : 1577
>>Command: A 3
( 3) MA 750_
>>Command: I 4
( 4) WAIT 10
( 5) _
```

d. Press the ENTER key.

A new line is inserted and each of the subsequent line numbers increases by one. The command prompt is displayed and the monitor waits for the next program-edit command.

```
>EDIT PROGRAM1
Sequence Name   : PROGRAM1
Sequence Number : 1
Lines          : 5
Bytes          : 23
Bytes Free      : 1577
>>Command: A 3
( 3) MA 750_
>>Command: I 4
( 4) WAIT 10
( 5)
>>Command: _
```

5. Delete “MGHP” from line 5 using the following steps:  
 Enter “D 5” and press the Enter key.  
 Line 5 is deleted, and each of the subsequent line numbers decreases in turn.  
 The command prompt is displayed and the monitor waits for the next program-edit command.

```

>EDIT PROGRAM1
Sequence Name   : PROGRAM1
Sequence Number : 1
Lines          : 5
Bytes          : 23
Bytes Free      : 1577
>>Command: A 3
( 3) MA 750_
>>Command: I 4
( 4) WAIT 10
( 5)
>>Command: D 5
>>Command: _

```

## ■ Ending the Edit Session

1. Enter the command “S” to end the session after saving the edited contents, then press the Enter key.  
 The edited contents are saved, and a syntax check is performed.  
 When an error in syntax is found, the line number on which the error was found is displayed together with the nature of the error.

```

Bytes          : 23
Bytes Free      : 1577
>>Command: A 3
( 3) MA 750_
>>Command: I 4
( 4) WAIT 10
( 5)
>>Command: D 5
>>Command: S
Compiling...OK
Saving.....OK
>>Command: _

```

2. Enter “Q” to quit the sequence editor.  
 A “>” (command prompt) is displayed.

```

>>Command: A 3
( 3) MA 5_
>>Command: I 4
( 4) WAIT 10
( 5)
>>Command: D 5
>>Command: S
Compiling...OK
Saving.....OK
>>Command: Q
> _

```

## 12.7 Executing a Sequence

You can execute sequences stored in the device's memory.  
There are three ways to execute a sequence.

### ■ Automatic execution upon power-up or after a device RESET

A sequence named "CONFIG" will be run automatically upon power-up or device RESET.

### ■ Executing a Sequence from the Terminal

1. Connect the device to the terminal.
2. Enter the terminal command "RUN □" (□ indicates either a sequence name or number).  
Insert a space between "RUN" and the sequence name (or number).  
When the command is entered, the system executes the sequence.

### ■ Executing a Sequence from I/O

1. Connect the START input. Connect IN1 to IN6 and ABORT inputs, as needed.
2. Assert IN1 to IN6 inputs to select the sequence to execute.  
Sequence is selected by the binary bit IN1 to IN6. (See the chart below)  
Inputs assigned to other function (PAUSE, PAUSECL) are always read as OFF. (E.g. INPAUSE=6 means IN6 is always read as OFF.)

Example of selection

Decimal Value	Input Port					
	IN1	IN2	IN3	IN4	IN5	IN6
1	ON					
2		ON				
4			ON			
8				ON		
16					ON	
32						ON
3	ON	ON				
6		ON	ON			
10		ON		ON		
63	ON	ON	ON	ON	ON	ON

#### Note

- Empty sections means that input is OFF.
- This selection can also be done with rotary digital switches.

3. Give a START input. System starts executing the desired sequence.  
There are two ways of action for START input. It is configured by STARTACT.

STARTACT	Operation
0	Setting START input from OFF to ON starts sequence execution. Setting START input from ON to OFF does not stop sequence. ABORT input is needed for aborting sequence.
1	Setting START input from ON to OFF starts sequence execution. Setting START input from OFF to ON aborts the sequence.

## 12.8 Error Messages

This section lists error messages that may be displayed on the terminal during program creation, syntax checking and program execution.

### ■ Error Messages Displayed during Program Creation

Unknown command: xxxx.	
Cause/action:	Input at Editor prompt did not match any of the single-character Editor commands (which can be seen by entering 'H' for [H]elp).
Invalid sequence name.	
Cause/action:	Given sequence name exceeds 10 characters
Attempted to edit a locked sequence.	
Cause/action:	At "R X Y"(rename), "D X"(delete), "E X"(edit), sequence X is locked.
Sequence directory full.	
Cause/action:	Tried to create a sequence, by [C]opy an existing sequence or [S]ave from the editor. No free directory entries available: all 100 are used.
Sequence editor memory full.	
Cause/action:	Editor memory is full, cannot add any more text.
Sequence storage memory full.	
Cause/action:	Sum of stored sequences + this attempt to [C]opy or [S]ave (from editor) would overflow available sequence storage memory. (EEPROM).
Invalid line number.	
Cause/action:	Editor command prompt expecting a line number. Found text, but wasn't a valid line number. (Example: 34c)
Invalid editor syntax.	
Cause/action:	Extra text is found after an editor command.
End line must follow start line	
Cause/action:	Many Editor commands can take both start and end line numbers ([A]lter, [D]elete, [L]ist, [C]opy, [C]ut...). The start line must be before the end line.

### ■ Error Messages Displayed during Syntax Check

Array index out of range.	
Cause/action:	Reference to POS[ ] data, index out of range. Can happen in any of MA POS[ ], POS[ ], POS[ ]=, =POS[ ].
Invalid argument.	
Cause/action:	Argument is invalid for the command. (MA xxx, WAIT xxx, VIEW xxx, etc)
Block depth too deep.	
Cause/action:	"Blocks" (WHILE-WEND, LOOP-ENDL, IF-ENDIF) can be "nested" inside each other. The driver permits up to 6 levels of nesting.
BREAKL outside LOOP block.	
Cause/action:	BREAKL is entered at the outside of LOOP block.
BREAKW outside WHILE block.	
Cause/action:	BREAKW is entered at the outside of WHILE block.
Conditional expression expected.	
Cause/action:	IF or WHILE statements require a conditional expression.
Invalid sequence number.	
Cause/action:	CALL by number detected invalid sequence number, number out of range (0 to 63), or fraction.
Invalid sequence reference.	
Cause/action:	Argument to CALL was not a valid sequence name.

Invalid text (missing separator?).	
Cause/action:	After a valid statement, found text before end-of-line. No separator (;), and not in comment.
Loop count must be positive integer.	
Cause/action:	Negative number is entered as argument for LOOP.
Invalid assignment.	
Cause/action:	Found something untranslatable involving an assignment. Note that '=' is required for all math operations.
Invalid ELSE-ENDIF block.	
Cause/action:	ELSE must be followed by ENDIF. ENDIF must be preceded by IF or ELSE.
Invalid IF block.	
Cause/action:	IF must be followed by ELSE or ENDIF. ELSE must be preceded by IF. ENDIF must be preceded by IF or ELSE.
Invalid LOOP block.	
Cause/action:	LOOP must be followed by ENDL. ENDL must be preceded by LOOP.
Invalid number.	
Cause/action:	Something that looked like a constant number contained unexpected text, or was out of range.
Invalid WHILE block.	
Cause/action:	WHILE must be followed by WEND. WEND must be preceded by WHILE.
Sequence needs block closure.	
Cause/action:	Compiler was still expecting ENDIF, ENDL, WEND when finished processing sequence.
String too long.	
Cause/action:	Assignment to user string variable, SAS, SACS arguments exceed limit of string length.
Text beyond END.	
Cause/action:	(Non-commented) text found beyond END statement.
Unknown command or parameter.	
Cause/action:	Command or parameter is not found.
Unsupported precision.	
Cause/action:	Numeric constant specified with too much precision (e.g. 1.2345).
Read-only parameter.	
Cause/action:	Attempt to modify a read-only parameter (e.g. IA)
Parameter cannot be displayed.	
Cause/action:	Attempt to "View" a non-viewable parameter (e.g. KB, KBQ).
WAIT must be positive.	
Cause/action:	Negative number is entered as argument for WAIT.

## ■ Error Messages Displayed during Program Execution

These are not displayed in multi axis mode.

Drive temperature over limit.	Cause/action: Driver temperature is out of specification. Revise the ventilation condition so that the ambient temperature of the device becomes 40°C (104°F) or below.
Drive voltage over limit.	Cause/action: The main circuit's inverter voltage has exceeded the upper limit. When an alarm has occurred during acceleration/deceleration: Reduce the inertial load or increase the acceleration/deceleration times.
EEPROM data corrupt.	Cause/action: EEPROM data is destroyed.
Both +LS, -LS ON.	Cause/action: Both the +LS and -LS are ON simultaneously. Check the logic setting for hardware limit sensors Normally Open (N.O.) or Normally Closed (N.C.).
LS detected, opposite HOME direction.	Cause/action: Opposite LS is detected from HOME direction. Connect the +LS and -LS correctly.
Abnormal LS status detected on HOME.	Cause/action: Mechanical home seeking could not be executed correctly. Check the hardware limits, installation of HOMES, wiring, and operation data used for the mechanical home seeking.
HOMES not detected between +LS and -LS on HOME (3 sensor mode).	Cause/action: Check the hardware limits, installation of HOMES, wiring, and operation data used for the mechanical home seeking.
TIMING, SLIT or Index signal not detected on HOMES at HOME	Cause/action: Check that the appropriate input signal is wired correctly.
Over travel: +LS or -LS detected.	Cause/action: The device has exceeded its hardware limit. Check the equipment.
Over travel: software position limit detected	Cause/action: The device has exceeded its software limit. Revise the operation data or change the software limit range.
PSTOP input detected.	Cause/action: Device has detected PSTOP input. Motion and sequence have stopped. Check your system for this PSTOP cause.
+LS or -LS detected during OFFSET motion	Cause/action: LS detection on offset motion.
Attempted to start unpermitted motion.	Cause/action: Impossible motion pattern is selected on motion start. Revise the operation data.
Sequence stack overflow	Cause/action: Stack area for user program has overflowed. Reduce the number of nested commands.
Attempted to call non-existent sequence.	Cause/action: Non-existent program is called. Delete the program, then enter it again.
Calculation result overflow	Cause/action: Calculation result over flow. Enter a program name that exists.
Parameter out of range	Cause/action: Parameter exceeds its setting range.
Division by Zero detected	Cause/action: Divide by zero was executed. Revise the program.
Attempted to modify PC while moving.	Cause/action: "PC" counter is updated while the device is operating or has lost its holding torque. Execute the PC counter while the device is at a standstill in the energized state.

Attempted to modify PC while moving.
Cause/action: "PC" counter is updated while the device is operating or loses its holding torque. Execute the PC counter while the device is at a standstill in the energized state.
ALMSET command detected
Cause/action: ALMSET command is detected.
Attempted to start motion while moving.
Cause/action: Prohibit motion command from being executed while motion.
Unexpected interrupt occurred.
Cause/action: Unexpected interrupted has occurred.
Sequence system internal error (xx)
Cause/action: Other error (program compatibility, etc) Display only "sub code" in ALM command.
Warning: Driver voltage over limit
Cause/action: Check the main power.

## ■ Error Messages Relating to Monitor Commands

Error: Command or parameter is unknown.
Cause/action: Text entered at the command prompt is not recognized (e.g. "DIV", "VY").
Error: Action is not allowed. (Motor is moving)
Cause/action: Command is attempted that is not executable while motor is running. Attempted to modify a parameter that may not be modified while motor is moving.
Error: Action is not allowed. (Sequence is running)
Cause/action: Command that starts motion is attempted while a sequence is running.
Error: Action is not allowed. (Alarm is ON)
Cause/action: Command is attempted that is not executable while alarm is ON.
Error: Action is not allowed. (Motion or I/O settings incompatible)
Cause/action: One of following situations is detected. - Motion command attempted while current is OFF. - CV command is attempted while decelerating during MI, MA motion
Error: Value is invalid.
Cause/action: Attempt to set parameter, non-numeric text found where numeric value expected (e.g. "DIS=abcde", "VR=3□_4").
Error: Argument is invalid.
Cause/action: Attempt to execute command, non-numeric text found where numeric argument expected (e.g. "MA abcde").
Error: Parameter is out of range.
Cause/action: Attempt to set parameter, value is out of range. (e.g. "VR=-0.1")
Error: Argument is out of range.
Cause/action: Attempted to execute command, argument is out of range. (e.g. "MA 5000000000000000")
Unsupported precision.
Cause/action: Acceleration / Deceleration time specified with precision over 3 decimal places. (e.g. "TA=1.2345")  Acceleration / Deceleration time specified with too much precision for its scale. Supported precision: 3 decimal places up to 1000.000
Error: EEPROM write failed.
Cause/action: Data writing failed while saving parameter to EEPROM (by CLEARALL, SAVEPRM, etc).

Error: Source sequence does not exist.	
Cause/action:	Sequence copy: source sequence does not exist. (e.g. "COPY X Y": Sequence X does not exist.)
Error: Sequence already exists.	
Cause/action:	Rename: (new name) already exists. (e.g. "REN X Y": Sequence Y already exists.)
Error: Sequence directory full.	
Cause/action:	Copy: Required creating a new sequence, all 64 sequences exist already. Tried to create a sequence, by copying an existing sequence or saving from the editor. No free directory entries available: all 64 sequences are used.
Error: Sequence storage memory full.	
Cause/action:	Copy: Not enough memory to create a new sequence. Sum of stored sequences and this attempt to copy or save (from editor) would overflow available sequence storage memory. (In EEPROM).
Error: Sequence executable memory full.	
Cause/action:	Copy: Not enough memory to create a new sequence. Sum of stored sequences and this attempt to copy or save (from editor) would overflow available sequence executable memory. (In RAM).
Error: Destination sequence is locked.	
Cause/action:	Sequence copy: attempt to overwrite a locked sequence. (e.g. "COPY X Y": Sequence Y already exists and is locked.)
Error: Sequence is locked.	
Cause/action:	Rename: Target sequence is locked. Delete: Target sequence is locked. (e.g. "REN X Y", "DEL X", "EDIT X", "S X" (Save in sequence editor): X is locked.)
Error: Sequence storage memory access failed!	
Cause/action:	EEPROM may not be in operation. Failed to properly pass data to or from sequence storage. Data may be corrupt or unusable.
Error: Invalid sequence name.	
Cause/action:	Sequence name may exceed 10 characters. Sequence name may contain unpermitted letters. (e.g. Name starting with digit, "N_", "S_" etc)



# 13 Command List

This chapter provides detailed information about each command and parameter.

In the tables below, the commands are grouped by functionality, for quick reference. After the tables, each command or parameter is described in detail, in alphabetical order.

## ■ Table Keys

n/a	not applicable
SAVE & RESET REQUIRED	n/a : Not applicable, or not required. For parameters, new value becomes active immediately. S : New value active immediately, but SAVEPRM command required for new value to be active after RESET or power cycle. SA : New value active immediately and saved automatically. R : RESET or a power cycle is required to activate the change. The value is saved automatically.
IN SEQ?	yes : Command or parameter can be used within sequences. - : Command or parameter cannot be used within sequences.
h	(after a value) Value is shown in hexadecimal notation
source	Sequence names or sequence numbers, as appropriate. Names can be Up to 10 letters or
target	numbers, and must start with a letter.
newname	

## Motion Commands

COMMAND	DESCRIPTION	DEFAULT VALUE	RANGE	SAVE & RESET REQUIRED	IN SEQ?	PAGE
CONT	Continue motion	n/a	n/a	n/a	Yes	115
CV	Change velocity	n/a	1 to 500,000 [pps]	n/a	Yes	122
HSTOP	Hard stop	n/a	n/a	n/a	Yes	155
MA	Move to absolute position	n/a	-8,388,607 to +8,338,607 [steps]	n/a	Yes	175
MCN, MCP	Move continuously, negative or positive	n/a	n/a	n/a	Yes	177
MGHN, MGHP	Seek mechanical home position	n/a	n/a	n/a	Yes	180
MI	Move incremental distance	n/a	n/a	n/a	Yes	181
MIx	Start linked incremental move	n/a	(x = 0 to 3)	n/a	Yes	182
PAUSE	Pause motion	n/a	n/a	n/a	Yes	201
PAUSECLR	Clear state of paused motion	n/a	n/a	n/a	Yes	203
PSTOP	Panic stop	n/a	n/a	n/a	Yes	212
SSTOP	Soft stop	n/a	n/a	n/a	Yes	259

## Motion Variables

COMMAND	DESCRIPTION	DEFAULT VALUE	RANGE	SAVE & RESET REQUIRED	IN SEQ?	PAGE
AREAx	AREA1 position and AREA2 position for AREA output signal	0	(x = 1 to 2) -8,388,607 to +8,338,607 [steps]	n/a	Yes	106
DIS	Incremental motion distance	0	-8,388,607 to +8,338,607 [steps]	S	Yes	126
DISx	Linked motion distance or destination	0	(x = 0 to 3) -8,388,607 to +8,338,607 [steps]	S	Yes	127
HOME2SB	2 Sensor Homing Position Back Steps	200	0 to 32,767 [steps]	SA	Yes	145
HOMEofs	Home offset position	0	-8,388,607 to +8,338,607 [steps]	SA	Yes	148
HOMETR	Homing acceleration and deceleration time	0.5	0.001 to 1000 [sec]	SA	Yes	152
HOMEVR	Homing Operation speed	1000	1 to 500,000 [pps]	SA	Yes	153
HOMEVS	Homing start speed	100	1 to 500,000 [pps]	SA	Yes	154
INCABSx	Linked move type	1	(x = 0 to 3) 0 [Absolute] to 1 [Incremental]	S	Yes	158
LINKx	Link control	0	(x = 0 to 2) 0 [No Link] to 1 [Link-to-next]	S	Yes	170
POS[x]	Position array data	0	(x = 1 to 64) -8,388,607 to +8,338,607 [steps]	SA	Yes	211

COMMAND	DESCRIPTION	DEFAULT VALUE	RANGE	SAVE & RESET REQUIRED	IN SEQ?	PAGE
SCHGPOS	Distance after SENSOR input	0	0 to +8,338,607 [steps]	SA	Yes	228
SCHGVR	Velocity after SENSOR input	1000	1 to 500,000 [pps]	SA	Yes	229
TA	Acceleration time	0.5	0.001 to 1000 [sec.]	S	Yes	267
TD	Deceleration time	0.5	0.001 to 1000 [sec.]	S	Yes	268
VR	Running velocity	1000	1 to 500,000 [pps]	S	Yes	279
VRx	Linked motion running velocity	1000	(x = 0 to 3) 1 to 500,000 [pps]	S	Yes	280
VS	Starting velocity	100	1 to 500,000 [pps]	S	Yes	281

## System Control

COMMAND	DESCRIPTION	DEFAULT VALUE	RANGE	SAVE & RESET REQUIRED	IN SEQ?	PAGE
;	Statement separator for multi-statement lines	n/a	n/a	n/a	Yes	90
<ESC>	(Escape): Abort operation(s)	n/a	n/a	n/a	–	92
ABORT	Abort sequences and motions	n/a	n/a	n/a	Yes	95
ABORTACT	Abort action	1	0 [Hardstop, Exit Sequence] 1 [Soft stop, Exit Sequence]	R	–	96
ALMACT	Alarm action	2	1 [Abort, Current On, Alarm On] 2 [Abort, Current OFF, Alarm ON]	R	–	99
ALMCLR	Clear alarm	n/a	n	n/a	–	100
ALMMSG	Alarm message action	0	0 [No messages] 1 [Messages, Alarms Only] 2 [Messages, Alarms and Warnings]	SA	–	103
ALMSET	Set user alarm	n/a	n/a	n/a	Yes	104
CLEARALL	Clear all programming (return to factory condition)	n/a	n/a	n/a	–	112
CLEARPOS	Clear POS[x] position array data	n/a	n/a	n/a	–	113
CRRUN	Run current	100	5 to 100 [% of Rated Current]	SA	Yes	119
CRSTOP	Stop current	50	5 to 50 [% of Rated Current]	SA	Yes	120
CURRENT	Current ON/OFF	1	0 [Motor Current OFF] 1 [Motor current ON]	n/a	Yes	121
DIRINV	Direction Invert	1	0 [positive motion is counterclockwise] 1 [positive motion is clockwise]	R	–	125
DTMPWNG	Drive temperature warning	85	0 to 85 [°C]	SA	Yes	129
EGA	Encoder Electrical Gear Ratio A	500	1 to 250,000	R	–	133
EGB	Encoder Electrical Gear Ratio B	500	1 to 250,000	R	–	133
HOMEDIR	Homing start direction	1	0 [Negative] 1 [Positive]	SA	Yes	146
HOMESEL	Homing type select	1	0 [2 sensors] 1 [3 sensors]	SA	Yes	150
INITPRM	Initialize parameters	n/a	n/a	n/a	–	160
LIMN, LIMP	Software position limits	LIMN=-8,388,607 LIMP=+ 8,388,607	-8,388,607 to +8,388,607 [steps]	SA	Yes	169
LSEN	Hardware overtravel limit enable	1	0 [Disable] 1 [Enable]	SA	–	174
MRES	Motor resolution	0	0 to 15 (See command description for details)	R	–	185
OTACT	Overtravel action	0	0 [Hard Stop] 1 [Soft Stop]	R	–	186
RESET	RESET device	n/a	n/a	n/a	–	218
SAVEPRM	Save parameters	n/a	n/a	n/a	–	224
SCEN	Self correcting enable	0	0 [Disabled] 1 [Enable]	R	-	226
SCTO	Self correcting timeout	1.000	0 to 10.000	SA	-	231
SENSORACT	SENSOR input action	2	0 [Hard Stop] 1 [Soft Stop] 2 [Soft stop at fixed distance from SENSOR signal]	R	–	232

COMMAND	DESCRIPTION	DEFAULT VALUE	RANGE	SAVE & RESET REQUIRED	IN SEQ?	PAGE
SLEN	Software position limit control	0	0 [Disabled] 1 [Enabled after homing]	SA	Yes	256
SLITEN	SLIT enable during homing operation	0	0 [Disabled] 1 [Enable]	SA	Yes	257
STARTACT	START input action	0	0 [Start Sequence when set active] 1 [Start Sequence when set active, Abort when set inactive]	R	–	260
STOACT	Step Out Action	0	0 [None] 1 [Warning] 2 [Alarm]	R	–	262
STOB	Step Out Alarm/Warning	7.2	0.1 to 360.0 [deg]	SA	Yes	263
STOEN	Step Out Detection enable	0	0 [Disabled] 1 [Enable]	R	–	264
STRSW	Current state at system start	1	0 [Current OFF] 1 [Current ON]	SA	–	266
TIMEN	Either TIM or ZSG enabled during homing operation	0	0 [Disabled] 1 [Enable TIM] 2 [Enable ZSG]	SA	Yes	271
WNGCLR	Clear Warning	n/a	n/a	n/a	-	286

## System Status

COMMAND	DESCRIPTION	DEFAULT VALUE	RANGE	SAVE & RESET REQUIRED	IN SEQ?	PAGE
DTMP	Drive temperature	n/a	n/a [°C]	n/a	Yes	128
EC	Encoder Counter	n/a	-2,147,483,647 to +2,147,483,647	n/a	Yes	130
PC	Position counter	0	-8,388,607 to +8,388,607 [steps]	n/a	Yes	206
PCI	Incremental position command	0	-8,388,607 to +8,388,607 [steps]	n/a	Yes	207
PE	Position error	n/a	-8,388,607 to +8,388,607 [steps]	n/a	Yes	208
PF	Position feedback	n/a	-8,388,607 to +8,388,607 [steps]	n/a	Yes	209
PFI	Incremental Position feedback	n/a	-8,388,607 to +8,388,607 [steps]	n/a	Yes	210
SIGALM	System ALM output signal	n/a	0 [OFF] 1 [ON]	n/a	Yes	234
SIGALMCLR	System ALMCLR input signal	n/a	0 [OFF] 1 [ON]	n/a	Yes	235
SIGAREA	System AREA output signal	n/a	0 [OFF] 1 [ON]	n/a	Yes	236
SIGCROFF	System CROFF input signal	n/a	0 [OFF] 1 [ON]	n/a	Yes	237
SIGHOME	System HOMING START input signal	n/a	0 [OFF] 1 [ON]	n/a	Yes	238
SIGHOMEP	System HOMEP output signal	n/a	0 [OFF] 1 [ON]	n/a	Yes	239
SIGHOMES	System HOMES input signal	n/a	0 [OFF] 1 [ON]	n/a	Yes	240
SIGLSN	System LSN input signal	n/a	0 [OFF] 1 [ON]	n/a	Yes	241
SIGLSP	System LSP input signal	n/a	0 [OFF] 1 [ON]	n/a	Yes	242
SIGMOVE	System MOVE output signal	n/a	0 [OFF] 1 [ON]	n/a	Yes	243
SIGPAUSE	System PAUSE input signal	n/a	0 [OFF] 1 [ON]	n/a	Yes	244
SIGPAUSECL	System PAUSECL input signal	n/a	0 [OFF] 1 [ON]	n/a	Yes	245
SIGPSTOP	System PSTOP input signal	n/a	0 [OFF] 1 [ON]	n/a	Yes	246
SIGPSTS	System PSTS output signal	n/a	0 [OFF] 1 [ON]	n/a	Yes	247
SIGREADY	System READY output signal	n/a	0 [OFF] 1 [ON]	n/a	Yes	248
SIGRUN	System RUN output signal	n/a	0 [OFF] 1 [ON]	n/a	Yes	249

COMMAND	DESCRIPTION	DEFAULT VALUE	RANGE	SAVE & RESET REQUIRED	IN SEQ?	PAGE
SIGSC	System SC output signal	n/a	0 [OFF] 1 [ON]	n/a	Yes	250
SIGSENSOR	System SENSOR input signal	n/a	0 [OFF] 1 [ON]	n/a	Yes	251
SIGSLIT	System SLIT input signal	n/a	0 [OFF] 1 [ON]	n/a	Yes	252
SIGSTO	System STEPOUT output signal	n/a	0 [OFF] 1 [ON]	n/a	Yes	253
SIGTEMP	System TEMP output signal	n/a	0 [OFF] 1 [ON]	n/a	Yes	254
SIGWNG	System WNG output signal	n/a	0 [OFF] 1 [ON]	n/a	Yes	255
TIMER	Running timer	n/a	0 to 500,000.000 [sec.]	n/a	Yes	272
VC	Velocity command	n/a	-500,000 to +500,000 [pps]	n/a	Yes	275

## I/O

COMMAND	DESCRIPTION	DEFAULT VALUE	RANGE	SAVE & RESET REQUIRED	IN SEQ?	PAGE
ABORTLV	ABORT input level	0	0 [Normally Open] 1 [Normally Closed]	R	–	97
ALMCLRLV	ALMCLR input level	0	0 [Normally Open] 1 [Normally Closed]	R	–	101
ALMLV	ALM output level	0	0 [Normally Open] 1 [Normally Closed]	R	–	102
AREALV	AREA output level	0	0 [Normally Open] 1 [Normally Closed]	R	–	105
CROFFLV	CROFF input level	0	0 [Normally Open] 1 [Normally Closed]	R	–	118
EVx	Configure event output	n/a	(x = 1 to 2) (See EVx entry for a full explanation.)	n/a	Yes	139
HOMELV	HOMING START input level	0	0 [Normally Open] 1 [Normally Closed]	R	–	147
HOMEPLV	HOME output level	0	0 [Normally Open] 1 [Normally Closed]	R	–	149
HOMESLV	HOME Switch input level	0	0 [Normally Open] 1 [Normally Closed]	R	–	151
IN	General input status	n/a	0 to 63	n/a	Yes	157
INITIO	Initialize I/O	n/a	n/a	n/a	–	159
INPAUSE	PAUSE signal input assignment	0	0 [unassigned] and 1 to 6	R	–	161
INPAUSECL	PAUSECL signal input assignment	0	0 [unassigned] and 1 to 6	R	–	162
INSG	System input signal status	0	0 to 8191	n/a	Yes	163
INx	Individual general input status	0	(x = 1 to 6) 0 [OFF] 1 [ON]	n/a	Yes	164
INxLV	INx input level	0	(x = 1 to 6) 0 [Normally Open] 1 [Normally Closed]	R	–	165
IO	Input/Output status	n/a	n/a	n/a	–	166
MOVELV	MOVE output level	0	0 [Normally Open] 1 [Normally Closed]	R	–	184
OTLV	Overtravel input level	0	0 [Normally Open] 1 [Normally Closed]	R	–	187
OUT	General output status	n/a	0 to 15	n/a	–	188
OUTAREA	AREA signal output assignment	0	0 [unassigned] and 1 to 4	R	–	189
OUTHOMEP	HOME output signal assignment	0	0 [unassigned] and 1 to 4	R	–	190
OUTPSTS	PSTS signal output assignment	0	0 [unassigned] and 1 to 4	R	–	191
OUTREADY	READY signal output assignment	0	0 [unassigned] and 1 to 4	R	–	192
OUTRUN	RUN signal output assignment	0	0 [unassigned] and 1 to 4	R	–	193
OUTSC	SC signal output assignment	0	0 [unassigned] and 1 to 4	R	–	194

COMMAND	DESCRIPTION	DEFAULT VALUE	RANGE	SAVE & RESET REQUIRED	IN SEQ?	PAGE
OUTSG	System output signal status	n/a	0 to 2047	n/a	Yes	195
OUTSTO	STEPOUT signal output assignment	0	0 [unassigned] and 1 to 4	R	–	196
OUTTEMP	TEMP signal output assignment	0	0 [unassigned] and 1 to 4	R	–	197
OUTTEST	I/O test utility	n/a	n/a	n/a	–	198
OUTWNG	WNG signal output assignment	0	0 [unassigned] and 1 to 4	R	–	199
OUTx	Individual general output control	0	(x = 1 to 4) 0 [OFF] 1 [ON]	n/a	Yes	200
PAUSECLLV	PAUSECL input level	0	0 [Normally Open] 1 [Normally Closed]	R	–	204
PAUSELV	PAUSE input level	0	0 [Normally Open] 1 [Normally Closed]	R	–	205
PSTOPLV	PSTOP input level	0	0 [Normally Open] 1 [Normally Closed]	R	–	213
PSTSLV	PSTS output level	0	0 [Normally Open] 1 [Normally Closed]	R	–	214
READYLV	READY output level	0	0 [Normally Open] 1 [Normally Closed]	R	–	215
RUNLV	RUN output level	0	0 [Normally Open] 1 [Normally Closed]	R	–	221
SCLV	SC output level	0	0 [Normally Open] 1 [Normally Closed]	R	–	230
SENSORLV	SENSOR input level	0	0 [Normally Open] 1 [Normally Closed]	R	–	233
SLITLV	SLIT input level	0	0 [Normally Open] 1 [Normally Closed]	R	–	258
STARTLV	START input level	0	0 [Normally Open] 1 [Normally Closed]	R	–	261
STOLV	STEPOUT output level	0	0 [Normally Open] 1 [Normally Closed]	R	–	265
TEMPLV	TEMP output level	0	0 [Normally Open] 1 [Normally Closed]	R	–	270
WNGLV	WNG output level	0	0 [Normally Open] 1 [Normally Closed]	R	–	287

## Monitor Commands

COMMAND	DESCRIPTION	DEFAULT VALUE	RANGE	SAVE & RESET REQUIRED	IN SEQ?	PAGE
ALM	Alarm status and history	n/a	00h to F2h	n/a	–	98
HELP	Display help information	n/a	n/a	n/a	–	140
REPORT	Display system status	n/a	n/a	n/a	–	217
TEACH	Teach Positions	n/a	n/a	n/a	–	269
TRACE	Sequence trace control	0	0 [Disabled] 1 [Enabled]	n/a	–	273
VER	Display firmware version	n/a	n/a	n/a	–	276
WNG	Warning status and history	n/a	00h to F2h	n/a	–	285

## Communications

COMMAND	DESCRIPTION	DEFAULT VALUE	RANGE	SAVE & RESET REQUIRED	IN SEQ?	PAGE
@	Select device	n/a	*, 0 to 9, A to Z	n/a	–	91
\	Global command	n/a	n/a	n/a	–	94
ECHO	Communications echo control	1	0 [Echo OFF] 1 [Echo ON]	SA	–	131
VERBOSE	Command response control	1	0 [Respond with data only] 1 [Respond with data and descriptive text]	SA	–	277

## Sequence Commands

COMMAND	DESCRIPTION	DEFAULT VALUE	RANGE	SAVE & RESET REQUIRED	IN SEQ?	PAGE
#	Sequence Comment	n/a	n/a	n/a	Yes	88
BREAKL	Break LOOP block	n/a	n/a	n/a	Yes	109
BREAKW	Break WHILE block	n/a	n/a	n/a	Yes	110
CALL	Call sequence as subroutine	n/a	Valid sequence name or number. May be a variable.	n/a	Yes	111
ELSE	Begin ELSE block: execute if IF is false	n/a	n/a	n/a	Yes	135
END	End sequence	n/a	n/a	n/a	Yes	136
ENDIF	End of IF block	n/a	n/a	n/a	Yes	137
ENDL	End of LOOP block	n/a	n/a	n/a	Yes	138
IF	Begin IF block: execute if true	n/a	n/a	n/a	Yes	156
KB	Keyboard Input	n/a	Depends on target variable	n/a	Yes	167
KBQ	Keyboard Input (quiet)	n/a	Depends on target variable	n/a	Yes	168
LOOP	Begin counted LOOP block	n/a	1 to 500,000,000. Must be integer. To make an infinite loop, omit count. Count may be a variable.	n/a	Yes	173
MEND	Wait for motion end	n/a	n/a	n/a	Yes	179
RET	Sequence Return	n/a	n/a	n/a	Yes	219
SACS	Send ASCII control string	n/a	Characters to transmit (up to 70). May contain embedded control characters. Does not append carriage return and line feed. No new prompt.	n/a	Yes	222
SAS	Send ASCII string	n/a	Characters to transmit (up to 70). Appends carriage return and line feed. New prompt.	n/a	Yes	223
VIEW	View parameter	n/a	Valid parameter or variable name	n/a	Yes	278
WAIT	Wait for specified time	n/a	0 – 500000.000 [sec.] May be a variable.	n/a	Yes	282
WEND	End of WHILE block	n/a	n/a	n/a	Yes	283
WHILE	Begin WHILE block: execute while true	n/a	n/a	n/a	Yes	284

## Math/Logical Operators (In sequences only)

COMMAND	DESCRIPTION	DEFAULT VALUE	RANGE	SAVE & RESET REQUIRED	IN SEQ?	PAGE
+, -, *, /, %, &,  , ^, <<, >>	Addition, subtraction, multiplication, division, modulo, AND, OR, XOR, left logic shift, right logic shift	n/a	n/a	n/a	Yes	89
a < b	a is smaller than b	n/a	n/a	n/a	Yes	93
a <= b	a is equal to or smaller than b	n/a	n/a	n/a	Yes	93
a = b	a is equal to b	n/a	n/a	n/a	Yes	93
a! = b	a is not equal to b	n/a	n/a	n/a	Yes	93
a >= b	a is equal to or larger than b	n/a	n/a	n/a	Yes	93
a > b	a is larger than b	n/a	n/a	n/a	Yes	93

## User Variables

COMMAND	DESCRIPTION	DEFAULT VALUE	RANGE	SAVE & RESET REQUIRED	MODE	IN SEQ?	PAGE
A to Z	User variables	0	-2,147,483,647 to 2,147,483,647	SA	0	Yes	107

## Sequence Management

COMMAND	DESCRIPTION	SYNTAX	IN SEQ?	PAGE
CLEARSEQ	Clear sequences	CLEARSEQ	–	114
COPY	Copy sequence	COPY source target	–	117
DEL	Delete sequence	DEL target	–	123
DIR	Sequence Directory	DIR [target]	–	124
EDIT	Edit sequence	EDIT [target]	–	132
LIST	List sequence contents	LIST target [startline] [endline]	–	171
LOCK	Lock Sequence	LOCK target	–	172
REN	Rename sequence	REN target newname	–	216
RUN	Run sequence	RUN target	–	220
UNLOCK	Unlock sequence	UNLOCK target	–	274

**# : Sequence Comment****Sequence Command**

<b>Execution Mode</b>	Sequence	
<b>Syntax</b>	#commenting text	
<b>Description</b>	<p>All text entered between the # symbol and the end of the line will not execute, but will be saved with the sequence. The # symbol is a means for commenting the commands within a sequence in order to describe the function of the commented sequence. Comments within a sequence are saved in EEPROM when the sequence is saved within the Editor Mode.</p> <p>Comments should not follow SAS or SACS commands on the same line. The text intended to be a comment will be transmitted as part of the SAS or SACS string.</p>	
<b>See Also</b>	EDIT, LIST	
<b>Example</b>	Command	Description
	>LIST 1	#List sequence 1
	(1) TA=0.5	#Acceleration Time, seconds
	(2) TD=0.5	#Deceleration Time, seconds
	(3) VS=10	#Starting Velocity, pps
	(4) VR=20	#Running Velocity, pps
	(5) DIS=10	#Distance of the move equals 10
	(6) MI	#Begin the index move
	(7) END	#End the sequence
	>	



**+, -, \*, /, %, &, |, ^, <<, >>****Math/Logical Operators****Execution Mode** Sequence**Syntax**  $Z = X \text{ n } Y$ 

X = Numeric Value or Variable

n = mathematical operation

Y = Numeric Value or Variable

Z = Variable

**Description** The following mathematical operations can be used in a program:

+ : Addition

- : Subtraction

\* : Multiplication

/ : Division

% : Modulo (remainder)

&amp; : AND (Boolean)

| : OR (Boolean)

^ : XOR (Boolean)

&lt;&lt; : Left logic shift (Shift to left bit)

&gt;&gt; : Right logic shift (Shift to right bit)

For simple constant assignments, the equals sign ("=") is not required. For assignment to a variable or to a mathematical expression, the equals sign is required.

Note on Modulo operations:  $A \% B = A - (B * \text{sign}(A/B) * \text{floor}(|A/B|))$

Division by zero (0) or numeric overflow will cause an Alarm condition, stopping motion and halting sequence operation.

**See Also** A to Z**Example**

Command

Description

&gt;LIST 1

#List the user entered sequence

( 1) X=2

#The variable X is set equal to two

( 2) Y=PC

#Variable Y is set equal to the Position Counter Value

( 3) X=X\*Y

#X equals the previous value of X multiplied by Y

( 4) X

#Print the current value of X to the terminal

( 5) END

#End the sequence

&gt;PC

#Query the PC value

PC=10

#Device response

&gt;RUN 1

#Run sequence #1

&gt;20

#Device response

&gt;

**; : Statement Separator****System Control**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	Command; Command	
<b>Description</b>	The semicolon (;) allows for multiple command statements to be used on a single command line. The maximum number of characters per one line is 80 characters.	
<b>Note</b>	The semicolon cannot be used as a separator after an SACS or SAS command. The SAS and SACS commands transmit all following text (until the end of a line); no other statements can follow SAS or SACS on the same line.	
<b>Example</b>	<b>Command</b>	<b>Description</b>
	>VR 1000; DIS 2000; MI	#Set the running velocity to 1000 pps, distance to 2000 steps and then perform an index move
	VR=1000	#Device response
	DIS=2000	#Device response
	>	

**@: Select Device****Communications**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	@id	
<b>Range</b>	id = 0 to 9, A to F	
<b>Description</b>	<p>Makes a logical connection to a specific device in a multiple device, e.g. multi-drop configuration. That device can then be uniquely addressed with SW1 in the front panel. If the device ID is anything other than default ID (*), communication with the device requires using the @ commands to establish communication.</p>	
<b>Note</b>	<ul style="list-style-type: none"> <li>Each device used in a multi-drop communication configuration requires a unique device ID.</li> <li>Function switch (multi-axis mode) SW2-No.4 must set to ON for the driver to be in multi-axis mode for the “@” command to have any effect.</li> <li>The “@” character will no be echoed until an axis number is entered following it. Once an axis number has been entered, the “@” character will be displayed on the screen.</li> </ul>	
<b>Example</b>	Command	Description
	0>MGHP	#Device 0 go home
	0>@A	#Talk to Device A
	A>MGHP	#Device A go home

**<ESC>: (Escape) Abort Operation(s)****System Control**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	<ESC> (Escape key or character)	
<b>Description</b>	<ESC> represents an escape key or character (1Bh). <ESC> will abort motion, decelerating to a stop. <ESC> will abort an executing sequence. <ESC> will discard any characters on a line and send a carriage return and line feed (CR + LF), and new prompt.	
<b>See Also</b>	ABORT, ABORTACT, ALMACT, HSTOP, PSTOP, SSTOP, TD	
<b>Example</b>	Command	Description
	>VR 1000	#Set the running velocity to 1000 pps
	VR=1000	#Device response
	>MCN	#Move continuously in the negative rotation direction
	> <ESC>	#<ESC> received, motion begins decelerating to a stop
	>	#New prompt

## **a!=b, a<=b, a<b, a=b, a>=b, a>b: Conditional Operators      Math/Logical Operator**

**Execution Mode**      Sequence

**Description**      The following conditional operations may be used in a sequence, as part of an IF or WHILE statement. a and b can be constants or any variable available within sequences.

- a!=b : a is not equal to b
- a<=b : a is less than or equal to b
- a<b : a is less than b
- a=b : a is equal to b
- a>=b : a is greater than or equal to b
- a>b : a is greater than b

**See Also**      IF, WHILE

<b>Example</b>	Command	Description
	>LIST 2	#List sequence 2

( 1) IF (IN1!=0)	#If Input 1 does not equal the logic OFF state or 0, then;
( 2) DIS=100	#Set the distance to 100
( 3) MI	#Move Incrementally
( 4) ENDIF	#End the IF Statement
( 5) END	#End the sequence
>	

**\ : Global Command****Communications**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	\ (Command)	
<b>Description</b>	Global command operator. Attaching this operator before the command enables command to all the units. Applicable Commands: ABORT, CONT, CURRENT, CV, HSTOP, MA, MCN, MCP, MGHN, MGHP, MI, MIx, PAUSE, PAUSECLR, PSTOP, RESET, RUN, SSTOP, <ESC> key	
<b>See Also</b>	@, VERBOSE	
<b>Example</b>	Command	Description
	2> \MI	#Send the Global MI command to all devices
	2>	#Device response
<b>Note</b>	Support for \CURRENT is limited. \CURRENT=0 and \CURRENT=1 are supported (globally set current ON and OFF, respectfully), but \CURRENT, as a query (no arguments) will respond for only the active device.	

**ABORT: Abort Sequence and Motions****System Control**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	ABORT	
<b>Description</b>	<p>The ABORT command will stop the execution of a sequence.</p> <p>Commanding sequence ABORT while the motor is running will stop any sequence execution and cause the motor to come to a stop based on the ABORTACT setting.</p>	
<b>See Also</b>	<ESC>, ABORTACT, ALMACT, HSTOP, PSTOP, SSTOP	
<b>Example</b>	Command	Description
	>LIST 9	#List sequence 9
	( 1) TA=0.5	#Acceleration Time, seconds
	( 2) TD=0.1	#Deceleration Time, seconds
	( 3) VR=2000	#Set the running velocity to 2000 pps
	( 4) MCP	#Move continuously in the Positive direction
	>RUN 9	#Execute sequence #9
	>ABORT	#Abort sequence execution and decelerate the motor to a stop
	>	

**ABORTACT: Abort Action****System Control**

<b>Execution Mode</b>	Immediate																												
<b>Syntax</b>	ABORTACT n																												
<b>Range</b>	n= 0: Hard Stop (stop as quickly as possible) 1: Soft Stop (controlled deceleration over time)																												
<b>Initial Value</b>	1																												
<b>RESET</b>	RESET or recycle the power is required to activate the change. The parameter is saved automatically.																												
<b>Access</b>	READ and WRITE																												
<b>Description</b>	<p>ABORTACT establishes the motor action upon activation of the ABORT input and the ABORT command.</p> <p>If ABORTACT=0, the ABORT input and command stop the motor as quickly as possible (hard stop). ABORT behaves exactly the same as HSTOP.</p> <p>If ABORTACT=1, the ABORT input and command stop the motor by controlled deceleration (soft stop).</p> <p>ABORT behaves exactly the same as SSTOP.</p>																												
<b>Caution</b>	Ensure the ABORTACT is set properly prior to asserting the ABORT input or executing the ABORT command.																												
<b>See Also</b>	ABORT, ABORTLV, HSTOP, SSTOP																												
<b>Example</b>	<table> <thead> <tr> <th>Command</th><th>Description</th></tr> </thead> <tbody> <tr> <td>&gt;ABORTACT</td><td>#Check the ABORTACT setting</td></tr> <tr> <td>ABORTACT=1(1)</td><td>#Set for soft stop action</td></tr> <tr> <td>&gt;VS 100; VR 1000</td><td>#Set start velocity 100 pps, run velocity 1000 pps</td></tr> <tr> <td>VS=100</td><td></td></tr> <tr> <td>VR=1000</td><td></td></tr> <tr> <td>&gt;TA 0.05; TD 0.025</td><td>#Acceleration time 0.05, Deceleration time 0.025</td></tr> <tr> <td>TA=0.050</td><td></td></tr> <tr> <td>TD=0.025</td><td></td></tr> <tr> <td>&gt;MCP</td><td>#Start continuous motion, positive direction</td></tr> <tr> <td>&gt;VC</td><td>#Check velocity command</td></tr> <tr> <td>VC=1000</td><td>#Velocity has reached running speed</td></tr> <tr> <td>&gt;ABORT</td><td>#Stop: will be a soft stop because ABORTACT is 1</td></tr> <tr> <td>&gt;</td><td></td></tr> </tbody> </table>	Command	Description	>ABORTACT	#Check the ABORTACT setting	ABORTACT=1(1)	#Set for soft stop action	>VS 100; VR 1000	#Set start velocity 100 pps, run velocity 1000 pps	VS=100		VR=1000		>TA 0.05; TD 0.025	#Acceleration time 0.05, Deceleration time 0.025	TA=0.050		TD=0.025		>MCP	#Start continuous motion, positive direction	>VC	#Check velocity command	VC=1000	#Velocity has reached running speed	>ABORT	#Stop: will be a soft stop because ABORTACT is 1	>	
Command	Description																												
>ABORTACT	#Check the ABORTACT setting																												
ABORTACT=1(1)	#Set for soft stop action																												
>VS 100; VR 1000	#Set start velocity 100 pps, run velocity 1000 pps																												
VS=100																													
VR=1000																													
>TA 0.05; TD 0.025	#Acceleration time 0.05, Deceleration time 0.025																												
TA=0.050																													
TD=0.025																													
>MCP	#Start continuous motion, positive direction																												
>VC	#Check velocity command																												
VC=1000	#Velocity has reached running speed																												
>ABORT	#Stop: will be a soft stop because ABORTACT is 1																												
>																													



**ABORTLV: ABORT Input Level****I/O**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	ABORTLV n	
<b>Range</b>	n = 0: Normally Open 1: Normally Closed	
<b>Initial Value</b>	0	
<b>RESET</b>	RESET or recycle the power is required to activate the change. The parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	ABORTLV sets the active level of the ABORT input.	
<b>See Also</b>	ABORT, ABORTACT	
<b>Example</b>	Command	Description
	>ABORTLV 1	#Set the ABORT input to Normally Closed
	ABORTLV=0(1)	#Configure ABORTACT so that ABORT causes a hard stop
	>ABORTACT 0	#Establish the saved parameter values
	ABORTACT=1(0)	
	>RESET	
	Resetting system.	
	-----	
	CRD5xx-KP	
	CRK Series Built-in Controller	
	Software Version: A378 V.x.xx	
	Copyright 2009-2011	
	ORIENTAL MOTOR U.S.A. CORP.	
	-----	
	>ABORTLV	#Confirm the current ABORTLV setting
	ABORTLV=1(1)	
	>	

**ALM: Alarm Status and History****Monitor Commands**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	ALM	
<b>Range</b>	00 to F2	
<b>Initial Value</b>	00	
<b>Access</b>	READ	
<b>Description</b>	The ALM command displays the current alarm code, history of the last 10 alarm issues. See Chapter of "Protective Functions", for a list of all ALM codes and causes. The current ALM Code is overwritten upon device power up or RESET. The Alarm history is automatically saved in EEPROM.	
<b>See Also</b>	ALMLV, ALMACT, ALMCLR, ALMMMSG, ALMSET, CURRENT	
<b>Example</b>	Command	Description
	>ALM	#Query the current ALM code
	ALARM =00 , RECORD : 22 23 9A 23 68 68 66 60 66 66	
	>	

**ALMACT: Alarm Action****System Control**

<b>Execution Mode</b>	Immediate
<b>Syntax</b>	ALMACT n
<b>Range</b>	n = 1: Abort sequences and stop motion. Motor current remains ON (ALARM ON) 2: Abort sequences and stop motion. Turn Motor Current OFF (ALARM ON)
<b>Initial Value</b>	2
<b>RESET</b>	RESET or recycle the power is required to activate the change. The parameter is saved automatically.
<b>Access</b>	READ and WRITE
<b>Description</b>	This alarm action command is only related to alarms generated from motion. Establishes the motor current response after a PSTOP operation, or after hardware or software over travel errors.

**See Also** ALMLV, ALM, ALMCLR, PSTOP

Example	Command	Description
	>ALMACT 1	#Set the ALMACT to 1
	ALMACT=2(1)	
	>RESET	#Establish the saved parameter values
	Resetting system.	
-----		
	CRD5xx-KP	
	CRK Series Built-in Controller	
	Software Version: A378 V.x.xx	
	Copyright 2009-2011	
	ORIENTAL MOTOR U.S.A. CORP.	
-----		
	>ALMACT	#Query new value
	ALMACT=1(1)	
	>	

**ALMCLR: Clear Alarm****System Control**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	ALMCLR	
<b>Description</b>	The ALMCLR command attempts to clear the system alarm status. If the alarm condition is no longer present, the system will become fully operational again.	
<b>See Also</b>	ALM, ALMLV, ALMACT, ALMMSG, ALMSET, CURRENT	
<b>Example</b>	Command	Description
	>ALM	#Query ALM
	ALARM = 68 , RECORD : 68 68 66 60 66 66 60 68 66 66	
	>ALMCLR	#Clear the alarm condition, if possible.
	>ALM	
<b>Note</b>	ALARM = 00 , RECORD : 68 68 66 60 66 66 60 68 66 66	
	>	
Before issuing an ALMCLR command, remove the cause of the alarm. If the ALARM condition persists, the drive will enter the ALARM state again. Please see chapter 14 “Troubleshooting” on page 288 for a description of the causes of specific ALARM codes.		
Some alarm conditions cannot be cleared. Refer to chapter 14 “Troubleshooting” on page 288 to see which alarm conditions can and cannot be cleared.		

**ALMCLRLV: ALARM CLEAR Level****I/O**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	ALMCLRLV n	
<b>Range</b>	n = 0: Normally Open 1: Normally Closed	
<b>Initial Value</b>	0	
<b>RESET</b>	RESET or recycle the power is required to activate the change. The parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	ALMCLRLV is the active level of the Alarm Clear (ALMCLR) input.	
<b>See Also</b>	ALM, ALMLV, ALMACT, ALMCLR, ALMMSG, ALMSET, CURRENT	
<b>Example</b>	Command	Description
	>ALMCLRLV 1 ALMCLRLV=0 (1) >RESET Resetting system.	#Set the ALMCLR input as Normally Closed #Device response #Establish the saved parameter values
-----		
CRD5xx-KP CRK Series Built-in Controller Software Version: A378 V.x.xx Copyright 2009-2011 ORIENTAL MOTOR U.S.A. CORP.		
-----		
	>ALMCLRLV ALMCLRLV =1 (1) >	#Query the current ALMCLRLV setting #Device response

**ALMLV: ALARM Output Level****I/O**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	ALMLV n	
<b>Range</b>	n = 0: Normally Open 1: Normally Closed	
<b>Initial Value</b>	0	
<b>RESET</b>	RESET or recycle the power is required to activate the change. The parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	Sets the active level of the ALARM output. The ALARM Output will switch to the opposite state of the normal setting when an alarm condition occurs. For instance, when the ALMLV=1 (Normally Closed) and the device is in an alarm state, the ALARM output will change to an open level (Normally Open)..	
<b>See Also</b>	SIGALM, OUTSG, ALM, ALMCLR	
<b>Example</b>	Command	Description
	>ALMLV 1	#Set the ALM Output as Normally Closed
	ALMLV=0 (1)	
	>RESET	#Establish the saved parameter values
	Resetting system.	
	-----	
	CRD5xx-KP	
	CRK Series Built-in Controller	
	Software Version: A378 V.x.xx	
	Copyright 2009-2011	
	ORIENTAL MOTOR U.S.A. CORP.	
	-----	
	>ALMLV	#Query the current ALMLV setting
	ALMLV=1 (1)	#Device response
	>	

**ALMMMSG: Alarm Message Action****System Control**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	ALMMMSG n	
<b>Range</b>	n = 0: Do not automatically transmit alarm and warning messages (default) 1: Automatically transmit messages for alarms, but not warnings 2: Automatically transmit messages for alarms and warnings	
<b>Initial Value</b>	0	
<b>SAVEPRM</b>	The new value takes effect immediately and the parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	The system can automatically transmit a message when alarms or warnings are detected. ALMMMSG controls what types of messages are automatically transmitted. Warning messages are sent only if the detected warning condition is different from the last reported warning.	
<b>See Also</b>	ALMLV, ALM, ALMACT, ALMCLR, ALMSET	
<b>Example</b>	Command	Description
	>ALMMMSG 1	#Set the ALMMMSG to messaging alarm only
	ALMMMSG=1 [Alarm]	
	>	

**ALMSET: Set User Alarm****System Control**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	ALMSET	
<b>Description</b>	The ALMSET command allows the user to place the device in a forced Alarm State.	
<b>See Also</b>	ALMLV, ALM, ALMACT, ALMCLR, ALMMSG	
<b>Example</b>	Command	Description
	>LIST CHKINPUT	#List sequence CHKINPUT
	( 1) DIS 1000	#Set distance to 1000
	( 2) VR 500	#Set run velocity to 500 pps
	( 3) MI	#Start incremental motion
	( 4) WHILE (SIGMOVE=1)	#While system is moving...
	( 5) IF (IN1=1)	#If general purpose input #1 is active
	( 6) SAS Illegal sensor input entry!	#Transmit a message
	( 7) SSTOP	#Stop motion
	( 8) MEND	#Wait for stop to complete
	( 9) <b>ALMSET</b>	<b>#Force an alarm: sequence halts.</b>
	(10) ENDIF	#Terminate IF block
	( 11) WEND	#Terminate WHILE loop
	( 12) SAS Motion succeeded	#Send a success message
	>RUN CHKINPUT	#Run sequence CHKINPUT
	>Motion succeeded	#Successful
	>RUN CHKINPUT	#Run again
	>Illegal sensor input entry!	#Sequence aborted
	>ALM	#Check alarm
	ALARM =E0 , RECORD : E0 30 23 9A 23 68 68 66 60 66	
	>SIGALM	#Query the ALARM status signal
	SIGALM=1	#The device is in an ALARM state
	>ALMCLR	#Clear the alarm
	>	#Device response



**AREALV: AREA Output Level****I/O**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	AREALV n	
<b>Range</b>	n = 0: Normally Open 1: Normally Closed	
<b>Initial Value</b>	0	
<b>RESET</b>	RESET or recycle the power is required to activate the change. The parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	AREALV is the active level of the AREA output.	
<b>See Also</b>		
<b>Example</b>	Command >AREALV=1 AREALV=0(1) >RESET Resetting system.	Description #Set the AREA output logic to Normally Closed #RESET the device to initialize the modified AREA setting
	-----	
	CRD5xx-KP CRK Series Built-in Controller Software Version: A378 V.x.xx Copyright 2009-2011 ORIENTAL MOTOR U.S.A. CORP.	
	-----	
	>AREALV AREALV=1(1) >	#New value is active

**AREAx : Area Position****Motion Variables**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	AREA[x] n	
<b>Range</b>	-8,388,607 to +8,388,607	
<b>SAVEPRM</b>	The new value takes effect immediately and the parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	<p>Sets the range for AREA output. The AREA output will be ON when the motor is inside the area set by the area 1 and area 2.</p> <ul style="list-style-type: none"> <li>• When the area 1 boundary is greater in position coordinate than the area 2 boundary: The AREA output turns ON when the output shaft is positioned at or after the area 2 boundary or at or before the area 1 boundary.</li> <li>• When the area 1 boundary is smaller in position coordinate than the area 2 boundary: The AREA output turns ON when the output shaft is positioned at or before the area 1 boundary or at or after the area 2 boundary.</li> <li>• The area 1 is the same as the area 2 boundary: The AREA output turns ON only when the output shaft is at the specified position.</li> </ul>	
<b>See Also</b>	AREA, AREALV, SIGAREA	
<b>Example</b>	Command	Description
	>AREA1 1000	#Set the AREA1 position to 1000
	AREA1=1000	#Display the AREA1 value
	>AREA2 -1000	#Set the AREA2 position to -1000
	AREA2=-1000	#Display the AREA2 value
	>MA 0	#Start absolute motion to position 0
	>SIGAREA	#Check AREA status
	SIGAREA=1	#Current position is in the area specified
	>MA 1100	#Start absolute motion to position 1100
	>SIGAREA	#Check AREA status again
	SIGAREA=0	#Current position is out of the area
	>	

**A to Z: User Variables****User Variables**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	<p><math>\{A   B. . . Y   Z\} = n</math></p> <p>In sequence only: <math>\{A   B. . . Y   Z\} = \text{expression}</math></p> <p>Upper and lower case are permitted, but 'A' and 'a' reference the same variable. There are 26 variables.</p>	
<b>Range</b>	<p><math>n = -2,147,483,647</math> to <math>+2,147,483,647</math></p> <p>expression must evaluate to a value within the same range as n, and can be any of:</p> <ul style="list-style-type: none"> <li>- constant numeric value</li> <li>- any variable available to sequences</li> <li>- math expressions</li> </ul>	
<b>Initial Value</b>	0	
<b>Access</b>	READ and WRITE	
<b>Description</b>	<p>General purpose numeric variables.</p> <p>In immediate mode, A to Z may only be set and queried.</p> <p>Within a sequence, variables may also be used in the following conditions:</p> <ul style="list-style-type: none"> <li>• Targets or arguments for assignments (e.g. A=TIMER; DIS=A)</li> <li>• Loop Counters (e.g. LOOP Q)</li> <li>• Conditional Statement Values (e.g. if (VR&gt;X))</li> <li>• Arguments for a subroutine CALL (e.g. CALL S)</li> <li>• Parts of Mathematical Expressions (CRRUN=CRSTOP+I)</li> <li>• Targets for interactive data entry commands (X=KBQ)</li> </ul> <p>A sequence will not show the name of the variable (A – Z) when the value is displayed to the terminal. The reason for this operation is to reduce the amount of ASCII information sent out of the device to an external host controller or terminal.</p> <p>When a variable is assigned to TIMER, its value is in msec, not second. For example, if TIMER is 1.261 and a variable A is assigned to TIMER, the variable A become 1261 (One Thousand Two hundred Sixty One).</p> <p>For example:</p> <p>Sequence 1</p> <p>( 1) A=2    #Set the value of variable A</p> <p>( 2) A    #Display the value of A</p> <p>When sequence 1 executes the device displays the following:</p> <pre>&gt;   2    #Device response to line 2 (shown above) &gt;</pre> <p>If the variable name must be displayed on the same line as the value, use the SACS command followed on the next line by the display command.</p> <p>Like all other variables, these variables have global scope. If, for instance, variable “T” will be used to hold a particular dwell time, then variable “T” should not be used for anything else in the application.</p>	
<b>See Also</b>	POS [x], VIEW, SAS, SACS	
<b>Example</b>	<p>Command</p> <pre>&gt;B 100   B=100 &gt;LIST 1</pre> <p>( 1) A=KB</p> <p>( 2) LOOP A</p> <p>*Continued on next page...</p>	<p>Description</p> <pre>#Set the Variable B to a value of 100 (msec) #Device response #List sequence 1</pre> <p>#Query the user for the value of the variable A via the serial port</p> <p>#Use A as a loop counter</p>

( 3) MI	#Move incrementally
( 4) MEND	#Wait for motion to end
( 5) WAIT B	#Time delay, 'B' seconds
( 6) ENDL	#Terminate the LOOP
>DIS 1	#Set distance to 1
DIS=1	
>RUN 1	#Run sequence 1
>? 4	#Prompt the user for the value of A
	#Motion will execute 4 times

**BREAKL: Break LOOP Block****Sequence Commands**

<b>Execution Mode</b>	Sequence	
<b>Syntax</b>	BREAKL	
<b>Description</b>	Exits the innermost LOOP block. Often used to exit a LOOP based on the value of a conditional statement.	
<b>See Also</b>	BREAKW, ELSE, ENDIF, ENDL, IF, LOOP, WEND, WHILE	
<b>Example</b>	Command	Description
	>LIST 7	#List sequence 7
	( 1) LOOP	#Loop indefinitely
	( 2) IF (IN2=1)	#If INPUT2 is 1 (ON), the sequence proceeds to line 3.
	( 3) BREAKL	#Exit the loop and execute the line after the ENDL command
	( 4) ELSE	#Branch here if not true
	( 5) SAS HELLO	#Send HELLO via the ASCII Communication port
	( 6) ENDIF	#End the IF statement
	( 7) ENDL	#End the loop and return to the beginning of the loop at line 1
		#End the sequence
	( 8) END	
	>	

**BREAKW: Break WHILE Block****Sequence Commands**

<b>Execution Mode</b>	Sequence	
<b>Syntax</b>	BREAKW	
<b>Description</b>	Exits the innermost WHILE block. Often used to exit a WHILE block based on the value of a conditional statement.	
<b>See Also</b>	BREAKL, ELSE, ENDIF, ENDL, IF, LOOP, WEND, WHILE	
<b>Example</b>	Command	Description
	>LIST 8	#List sequence 8
	( 1) WHILE (IN1=0)	#Start WHILE block. Execute lines 2 through 4 while condition is true
	( 2) IF (IN2=1)	#If IN2 is 1 (ON), execute line 3
	( 3) <b>BREAKW</b>	#Exit the WHILE loop and execute the line after the WEND command
	( 4) ENDIF	#End the IF block
	( 5) WEND	#End the WHILE block, return to line 1
	( 6) END	#End the sequence
	>	

**CALL: Call Sequence as Subroutine****Sequence Commands**

<b>Execution Mode</b>	Sequence	
<b>Syntax</b>	CALL n	
<b>Range</b>	n = Valid sequence name or number, or variable.	
<b>Description</b>	<p>Executes a sequence as a subroutine, then returns to the calling sequence. If n is a variable name (e.g. CALL Q), then Q must be equal to a valid sequence number. Calling sequences by name can make sequences more readable, but requires an internal name lookup operation. That operation takes an unpredictable amount of time, which depends on system activity and the number of sequences that have been programmed. Calling sequences by number is fast and always executes in the same elapsed time, but is less readable. Calling by variable is just slightly slower than calling by number, and always executes in the same elapsed time. Calling by variable should only be used if necessary, to avoid calling the wrong (or a nonexistent) sequence.</p> <p>If the CALL'ed sequence executes without error, control returns to the CALL'ing sequence, at the statement following the CALL . Nesting is permitted. Sequence 1 can CALL sequence 2, which can CALL sequence 3, etc. Each CALL requires some internal memory, however, which is drawn from a dedicated "Sequence Stack". The Sequence Stack is also used by block operations (IF, WHILE, LOOP). If many calls are nested, and/or blocks are nested deeply within a sequence, the Sequence Stack may become exhausted, resulting in alarm condition: "Sequence stack overflow".</p> <p>If the target sequence does not exist, an alarm is triggered, and all sequence processing stops.</p> <p><b>Note:</b> Be sure to end a CALL'ed sequences with the RET command.</p>	
<b>See Also</b>	DIR, RET	
<b>Example</b>	Command	Description
	>LIST 1	#List sequence 1
	( 1) LOOP	#Start of an infinite Loop
	( 2) CALL 2	#Call the Sequence Number 2
	( 3) OUT1=1	#Turn on Output #1
	( 4) WAIT 0.5	#Wait 0.5 seconds
	( 5) IF (IN1=1)	#If input #1 is ON
	( 6) BREAKL	#Break out of the loop
	( 7) ENDIF	#End the IF statement
	( 8) ENDL	#End the loop
	( 9) END	#End Sequence
	>LIST 2	#List sequence 2
	( 1) DIS=1000	#Distance equals 1000
	( 2) MI	#Begin the Index Move
	( 3) MEND	#Wait for motion to end before the Call command in the Calling program.
	( 4) RET	In this example the line after the CALL 2 command in sequence #1 is line 3 and is the next line to execute after the Subroutine Sequence #2 completes executing.
	>	

**CLEARALL: Clear All Programming and Parameters****System Control**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	CLEARALL	
<b>Description</b>	Clears all parameters, POS [x] position array data and all sequences. The CLEARALL command will clear all of the input and output assignments.	
<b>Caution</b>	Use caution when clearing all parameter values, position array data, and sequences. Once the information is cleared it cannot be restored. The CLEARALL command writes to EEPROM. The EEPROM has a nominal expected lifetime of 100,000 write cycles. The CLEARALL command should not be used automatically (i.e. by a host controller) if it could possibly execute at high frequency.	
<b>See Also</b>	CLEARPOS, CLEARSEQ, INITPRM	
<b>Example</b>	Command	Description
	>CLEARALL Enter Y to proceed, other key to cancel. y Initializing Parameters..OK. Clearing POS[ ] Data.....OK. Clearing.....OK. >	#Initialize all parameters, clear all position array data and sequences



**CLEARPOS: Clear POS[x] Position Array Data****System Control**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	CLEARPOS	
<b>Description</b>	Clears all POS [x] position array data. Position data will set to 0.	
<b>Caution</b>	Use caution when clearing position array data. Once the data points are cleared, they cannot be restored. The CLEARPOS command writes to EEPROM. The EEPROM has a nominal expected lifetime of 100,000 write cycles. The CLEARPOS command should not be used automatically (i.e. by a host controller) if it could possibly execute at high frequency.	
<b>See Also</b>	CLEARALL, CLEARSEQ, INITPRM, TEACH	
<b>Example</b>	Command	Description
	> CLEARPOS	#Clear all position array data to 0
	Enter Y to proceed, other key to cancel. y	
	Clear POS[ ] Data.....OK.	
	>	

**CLEARSEQ: Clear sequences****Sequence Management**

<b>Execution Mode</b>	Immediate				
<b>Syntax</b>	CLEARSEQ				
<b>Description</b>	Clears all sequences from the nonvolatile memory (EEPROM). The amount of time required to delete the sequences varies based on the number of sequences saved in memory.				
<b>Caution</b>	Use caution when clearing all sequences. Once the sequences are deleted, they cannot be restored. The CLEARSEQ command writes to EEPROM. The EEPROM has a nominal expected lifetime of 100,000 write cycles. The CLEARSEQ command should not be used automatically (i.e. by a host controller) if it could possibly execute at high frequency.				
<b>See Also</b>	CLEARALL, CLEARPOS, DEL, EDIT				
<b>Example</b>	<table> <tr> <td>Command</td><td>Description</td></tr> <tr> <td>&gt;DIR</td><td>#List all sequences</td></tr> </table>	Command	Description	>DIR	#List all sequences
Command	Description				
>DIR	#List all sequences				

```

## Name      TextSize Locked
== =====
0 <nameless> 10
1 <nameless> 37
Total: 2
Executable memory: 43 bytes used of 1600 bytes total, 3 percent.
Storage memory: 98 bytes used of 4223 bytes total, 3 percent.
>CLEARSEQ                                     #Delete all sequences from memory
Enter Y to proceed, other key to cancel. y    #Device response sent to the terminal

Clearing.....OK.

>                                              #Device response sent to the terminal

```

**CONT: Continue Motion from PAUSE****Motion Commands**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	CONT	
<b>Description</b>	<p>Resumes a motion after a PAUSE command or PAUSE input has caused a motion to pause. The remaining portion of the interrupted motion is completed.</p> <p>Acceleration and deceleration times TA and TD, and start and running velocities VS and VR determine the motion profile while changing speed. If the paused motion was a point-to-point index (MI, MA), the former destination becomes the destination for the resumed motion.</p> <p>If the paused motion was a continuous motion, the former direction is assumed for the continued motion. If the paused motion was a mechanical home seeking operation (MGHP, MGHN), a CONT command restarts the process from the beginning: CONT has the same effect as re-issuing the original MGHx command.</p> <p>In all cases, the system uses the values of VS, VR, TA and TD in effect at the time the CONT command is executed.</p> <p>The CONT command has no effect if motion has not been previously PAUSE'd.</p> <p>If sequences are running, the START input can cause the same action as a CONT command.</p>	
<b>See Also</b>	INPAUSE, INPAUSECL, OUTPSTS, PAUSE, PAUSECLLV, PAUSECLR, PAUSELV, PSTSLV, SIGPAUSE, SIGPAUSECL, SIGPSTS	
<b>Note</b>	<p>PAUSE and CONT may effect processing time of sequences. For instance: if a sequence executes a MEND (wait for motion end) command, the sequence will be suspended while the motion is paused, and will not proceed beyond the MEND until the next end of motion (via a CONT, PAUSECLR, or new motion).</p> <p>Pause and Continue operations are not supported for Linked Motions (MIx). PAUSE during a Linked Motion causes a soft stop, and subsequent CONT commands are ignored.</p>	
<b>Example</b>	Command	Description
	>LIST CHKJAM	#List sequence CHKJAM
	( 1) DIS=10; VR=10	#Set motion parameter
	( 2) LOOP	#Start infinite loop
	( 3) MI	#Start move incremental
	( 4) WHILE (DTMP<70)	#Check if over heated
	( 5) IF (SIGMOVE=0)	#Check for motion end
	( 6) BREAKW	#Exit while loop, if so
	( 7) ENDIF	
	( 8) WEND	
	( 9) IF (SIGMOVE!=0)	#Check if moving
	( 10) PAUSE	#DTMP>70: PAUSE motion
	( 11) WAIT TD	#Wait for stop, send text, get response
	( 12) SAS System in trouble.	
	( 13) SACS Enter 1 to continue, other to stop:	
	( 14) A=KBQ; SACS ^M^J>	
	( 15) IF (A=1)	
	( 16) CONT; MEND	#CONTinue, if A=1
	( 17) ELSE	#Otherwise, report stopped
	( 18) SAS Operation stopped.	
	( 19) RET	#Return from sequence
	( 20) ENDIF	
	( 21) ENDIF	
	( 22) SAS Motion end, goto next.	#Send normal message
	*Continued on next page...	

( 23) WAIT 1	#Dwell 1 second, loop back to top.
( 24) ENDL	#End the loop
>RUN CHKJAM	#Execute sequence CHKJAM
>Motion end, goto next.	#Normal message
>Motion end, goto next.	#Normal message
>System in trouble.	#Driver is getting hot
>Enter 1 to continue, other to stop:1	#Prompt message -> Entry "1"
>Motion end, goto next.	#Normal message
>Motion end, goto next.	#Normal message
>System in trouble.	#Driver is getting hot
>Enter 1 to continue, other to stop:2	#Prompt message -> Entry "2"
>Operation stoppped.	#Finished message (Sequence
>	finish)

**COPY: Copy Sequence****Sequence Management**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	COPY source target	
<b>Range</b>	source and target can be any valid sequence number (0–63) or name (consisting of letters or numbers, 10 character maximum, must start with a letter)	
<b>Description</b>	Makes a copy of a sequence. The original program will still exist in memory upon execution of the COPY command. If the destination program already exists, a confirmation message, “Destination exists, overwrite? [y/n]” is displayed to prompt the user for confirmation.	
<b>See Also</b>	DEL, EDIT, REN	
<b>Example</b>	Command	Description
	>COPY 1 MASTER	#Copy Sequence #1 to sequence named MASTER
	>COPY REMOTE 2	#Copy Sequence REMOTE to Sequence #2

**CROFFLV: Current OFF Input Level****I/O**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	CROFFLV n	
<b>Range</b>	n = 0: Normally Open 1: Normally Closed	
<b>Initial Value</b>	0	
<b>RESET</b>	RESET or recycle the power is required to activate the change. The parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	Sets the active level for the CROFF input.	
<b>See Also</b>	CURRENT, INITIO, IO, SAVEPRM, SIGCROFF	
<b>Example</b>	Command >CROFFLV=1 CROFFLV=0(1) >RESET Resetting system.	Description #Set the CROFF input logic to Normally Closed #RESET the device to initialize the modified CROFF setting
-----		
	CRD5xx-KP CRK Series Built-in Controller Software Version: A378 V.x.xx Copyright 2009-2011 ORIENTAL MOTOR U.S.A. CORP.	
-----		
	>CROFFLV CROFFLV=1(1) >	#New value is active

**CRRUN: Run Current****System Control**

<b>Execution Mode</b>	Immediate and Program	
<b>Syntax</b>	CRRUN n	
<b>Range</b>	n = 5 to 100 (integer values), (% of Rated Current)	
<b>Initial Value</b>	100 (% of Rated Current)	
<b>SAVEPRM</b>	The new value takes effect immediately and the parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	<p>The motor run current value is set as a percentage of the rated current. The motor current setting takes place immediately.</p> <p>CRRUN controls current while accelerating, running at constant speed, and decelerating.</p>	
<b>See Also</b>	CROFFLV, CRSTOP	
<b>Example</b>	Command	Description
	>CRSTOP 25	#The motor stop current is set to 25% of the maximum applicable current value (rated current)
	>CRRUN 50	#Set the motor run current to 50% of the maximum applicable current value (rated current)
	CRRUN=50	
	>	

**CRSTOP: Stop Current****System Control**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	CRSTOP n	
<b>Range</b>	n = 5 to 50 (integer values), (% of Rated Current)	
<b>Initial Value</b>	50 (% Rated Current)	
<b>SAVEPRM</b>	The new value takes effect immediately and the parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	The motor stop current value is set as a percentage of the rated current. The motor current setting takes place immediately.	
<b>See Also</b>	CROFFLV, CRRUN	
<b>Example</b>	Command	Description
	>CRSTOP 25	#The motor stop current is set to 25% of the maximum applicable current
	CRSTOP=25	value (rated current)
	>CRRUN 50	#Set the motor run current to 50% of the maximum applicable current
	CRRUN=50	value (rated current)
	>	



**CURRENT: Current ON/OFF****System Control**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	CURRENT n	
<b>Range</b>	n = 0: Motor Current is OFF 1: Motor Current is ON	
<b>Initial Value</b>	1: Motor current is ON (Motor current at power up can be controlled with STRSW.)	
<b>Access</b>	READ and WRITE	
<b>Description</b>	Enables or disables the motor current.	
<b>See Also</b>	CROFFLV, CRRUN, CRSTOP, SIGCROFF, STRSW	
<b>Note</b>	When CURRENT=1, the actual amount of current and holding torque is controlled by the values (in percent of rated current) of CRSTOP, CRRUN. CROFF input must be inactive.	
<b>Example</b>	Command	Description
	>CURRENT 0	#Turn motor current OFF. Motor has no holding torque
	CURRENT=0	
	>CURRENT 1	#Turn motor current on. Motor now has holding torque
	CURRENT=1	
	>	

**CV: Change Velocity****Motion Commands**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	CV n	
<b>Range</b>	n = 1 to 500,000 pps	
<b>Description</b>	<p>The CV command can be used to change the running velocity during an incremental positioning index (MI) or absolute positioning index (MA). Velocity changes over acceleration time TA if speed is increasing (away from zero) and deceleration time TD if speed is decreasing (toward zero).</p> <p>The CV command can only be used when the motor is accelerating or at running velocity. The CV command is not executable while the motor is decelerating to the final target position. If CV is attempted in communications mode while the motor is decelerating, the device will send out a warning message. If CV is attempted within a sequence while the motor is decelerating, an alarm is set (70h).</p> <p>Changing the running velocity via the CV command will affect the time required to complete the original commanded motion profile.</p> <p>There are several other ways to change speeds while moving:</p> <ul style="list-style-type: none"> <li>- If moving continuously by MCP, set new VR, and execute MCP again.</li> <li>- If moving continuously by MCN, set new VR, and execute MCN again</li> <li>- If all motion parameters are known, use linked index motions. Refer to MIx.</li> </ul> <p>Use the SENSOR input with SCHGVR and SCHGPOS</p> <p>MA, MCN, MCP, MI, MIx, VR, VS, SCHGVR, SCHGPOS</p> <p>If successful, a CV command modifies running velocity VR. The new value of VR will be “n” (the argument to the CV command).</p>	
<b>See Also</b>		
<b>Important Interactions</b>		
<b>Example</b>	Command	Description
	>VR 300	#Set the running velocity to 300 pps
	VR=300	
	>DIS 10000	#Set the distance to 10000
	DIS=10000	
	>MI	#Start the Index Move
	>CV 500	#Change the running velocity to 500 pps
	>LIST 5	#List sequence 5
	( 1) TA=0.1	#Set the acceleration time, seconds
	( 2) TD=0.1	#Set the deceleration time, seconds
	( 3) VS=100	#Set the starting velocity, pps
	( 4) VR=1000	#Set the running velocity, pps
	( 5) DIS=2000	#Set the distance
	( 6) MI	#Execute an Index Move
	( 7) WAIT 0.5	#Wait 0.5 second
	( 8) CV 500	#Change the running velocity of the Index Move to 500 pps
	( 9) SAS SPEED	#Transmit ASCII string
	CHANGE	
	( 10) END	#End the program
	>	

**DEL: Delete Sequence****Sequence Management**

<b>Execution Mode</b>	Immediate				
<b>Syntax</b>	DEL [target]				
<b>Range</b>	target can be the name or number of any existing sequence.				
<b>Description</b>	<p>Deletes a sequence from EEPROM. The system will request confirmation of the DEL action.</p> <p>A deleted sequence cannot be recovered.</p> <p>If the sequence is locked, it cannot be deleted. Use the UNLOCK command to unlock the sequence before deleting.</p> <p>Sequences cannot be deleted while any sequence is running.</p>				
<b>See Also</b>	CLEARALL, CLEARSEQ, COPY, DIR, EDIT, LOCK, UNLOCK				
<b>Note</b>	To delete all sequences see the CLEARSEQ command.				
<b>Example</b>	<table> <tr> <th>Command</th><th>Description</th></tr> <tr> <td>&gt;DIR</td><td>#Display the stored programs</td></tr> </table>	Command	Description	>DIR	#Display the stored programs
Command	Description				
>DIR	#Display the stored programs				

```

## Name      TextSize Locked
== =====
0 test1      9
1 <nameless> 37
Total: 2
Executable memory: 27 bytes used of 1600 bytes total, 2 percent.
Storage memory: 87 bytes used of 4223 bytes total, 3 percent.
>DEL TEST1 #Delete the program TEST1 from memory
Enter Y to proceed, other key to cancel. y #Device response
>

```

**DIR: Sequence Directory****Sequence Management**

<b>Execution Mode</b>	Immediate
<b>Syntax</b>	DIR [target]
<b>Range</b>	target is optional. If given, it should be a valid sequence number (0–63) or name (up to 10 alphanumeric characters, starting with a letter).
<b>Description</b>	Lists directory information for one or all sequences in memory. If target is given, lists information for that sequence only, with summary. If target is not given, lists information for all sequences, with summary.
<b>See Also</b>	COPY, EDIT, REN
<b>Example</b>	>DIR #List the entire sequence directory

```
## Name      TextSize Locked
==
1 Master      940
2 ReSync      93
3 FastReturn   32
```

Total: 3

Executable memory: 690 bytes used of 1600 bytes total, 43 percent.

Storage memory: 2259 bytes used of 4223 bytes total, 54 percent.

>

```
>DIR RESYNC #List directory information for one sequence only
```

```
## Name      TextSize Locked
==
2 ReSync      93
```

Executable memory: 690 bytes used of 1600 bytes total, 43 percent.

Storage memory: 2259 bytes used of 4223 bytes total, 54 percent.

>

**DIRINV: Direction Invert****System Control**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	DIRINV n	
<b>Range</b>	n = 0: Motor rotates in the Counter-Clockwise (CCW) direction for positive distance values 1: Motor rotates in the Clockwise (CW) direction for positive distance values	
<b>Initial Value</b>	1	
<b>RESET</b>	RESET or recycle the power is required to activate the change. The parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	Inverts the direction of motor rotation. When using a gearhead, the direction of the gearhead output shaft may rotate in the opposite direction of the motor's rotation.	
<b>See Also</b>	DIS, MA, MCN, MCP, MGHN, MGHP, MI	
<b>Example</b>	<div>Command</div> <div>&gt;DIRINV=0</div> <div>DIRINV=1(0)</div> <div>&gt;RESET</div> <div>Resetting system.</div> <div>-----</div> <div>CRD5xx-KP</div> <div>CRK Series Built-in Controller</div> <div>Software Version: A378 V.x.xx</div> <div>Copyright 2009-2011</div> <div>ORIENTAL MOTOR U.S.A. CORP.</div> <div>-----</div> <div>&gt;DIRINV</div> <div>DIRINV=0(0)</div> <div>&gt;DIS 1000</div> <div>DIS=1000</div> <div>&gt;MI</div> <div>&gt;</div>	<div>Description</div> <div>#Invert the motor direction</div> <div>#Device response</div> <div>#RESET the device to initialize the modified DIRINV setting</div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <

**DIS: Incremental Motion Distance****Motion Variables**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	DIS n	
<b>Range</b>	n = -8,388,607 to +8,388,607	
<b>Initial Value</b>	0	
<b>SAVEPRM</b>	The new value takes effect immediately. However, SAVEPRM is required to save the parameter values in nonvolatile memory. Otherwise, the parameter value is reset to the last saved value at device power up. If no new parameter value was saved, then the value is set to the default (initial) value.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	Determines the distance to be moved for the MI (move incremental) command. The sign of DIS determines the direction of motion.	
<b>See Also</b>	CV, DIRINV, MA, MI, TA, TD, VS, VR	
<b>Example</b>	Command	Description
	>VS 500	#Set Starting speed VS to 500 pps
	VS=500	
	>VR 2000	#Set Running speed VR to 2000 pps
	VR=2000	
	>DIS 5000	#Set Distance to 5000
	DIS=5000	
	>TA 0.5	#Set acceleration time to 0.5 seconds
	TA=0.5	
	>TD 0.5	#Set deceleration time to 0.5 seconds
	TD=0.5	
	>MI	#Execute index motion
	>	

**DISx: Linked Motion Distance or Destination****Motion Variables**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	DISx n	
<b>Range</b>	x = 0 to 3 (Linked Motion Profiles defined by DISx, VRx, INCABSx, and LINKx) n = -8,388,607 to +8,388,607	
<b>Initial Value</b>	0	
<b>SAVEPRM</b>	The new value takes effect immediately. However, SAVEPRM is required to save the parameter values in nonvolatile memory. Otherwise, the parameter value is reset to the last saved value at device power up. If no new parameter value was saved, then the value is set to the default (initial) value.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	Determines the incremental distance or absolute destination for the linked index (MIx) motion commands. For incremental links, the sign of DISx determines the direction of motion. Linked motions can only be run in one direction: all linked must have the same effective direction of travel.	
<b>See Also</b>	INCABSx, MIx, LINKx, VRx	
<b>Example</b>	Command	Description
	>VR1 500	#Set the velocity for linked move #1 to 500 pps
	VR1=500	#Device response
	>DIS1 2000	#Set the distance for linked move #1 to 2000
	DIS1=2000	#Device response
	>INCABS1 1	#Set the move type for linked motion #1 to incremental
	INCABS1=1 [INC]	#Device response
	>LINK1 1	#Enable the linked operation for motion #1
	LINK1=1	#Device response
	>VR2 1000	#Linked move #2 velocity equals 1000 pps
	VR2=1000	#Device response
	>INCABS2 1	#Set the move type for linked motion #2 to incremental
	INCABS2=1 [INC]	#Device response
	>DIS2 4000	#Linked move #2: destination is position 4000
	DIS2=4000	#Device response
	>LINK2 0	#"Unlink" link2 from link3
	LINK2=0	#Device response
	>MI1	#Start the linked operation motion
	>	

**DTMP: Drive Temperature****System Status****Execution Mode** Immediate and Sequence**Syntax** DTMP**Range** n/a (Degrees Celsius)**Access** READ

**Description** DTMP indicates the temperature measured near the device electronics, in degrees Celsius. The system constantly monitors temperature of the driver. The temperature can trigger an alarm or warning if excessive. Warning limits are set by DTMPWNG, and can be used to trigger an output (TEMP) if these limits are exceeded.

**See Also** / (Forward slash), DTMPWNG, OUTTEMP, TEMPLV

**Example**

Command	Description
>DTMPWNG 70	#Set the device to trigger a warning when the drive temperature exceeds 70 degrees
DTMPWNG=70	Celsius
>DTMP	#Query the drive temperature value
DTMP=60	#Displays the current drive temperature value
>SIGTEMP	#Query temperature warning signal
SIGTEMP=0	#SIGTEMP is zero because drive is below warning limits
>	



**DTMPWNG: Drive Warning Temperature****System Control**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	DTMPWNG n	
<b>Range</b>	n = 40 to 85 (integer values) (Degrees Celsius)	
<b>Initial Value</b>	85	
<b>SAVEPRM</b>	The new value takes effect immediately and the parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	<p>DTMPWNG controls the drive temperature threshold used to control the SIGTEMP temperature warning signal, and TEMP output (if used).</p> <p>The system monitors temperature of the driver. The temperature warning triggers if the temperature exceeds its programmed warning limit. DTMPWNG can be used to provide an early warning of elevated drive temperature (via TEMP output), so that actions may be taken to avoid an alarm and shutdown (e.g. reduce motor current, reduce application throughput, etc.). The temperature warning feature could also be used in applications that are very temperature sensitive (e.g. motor current could be disabled and machine operation suspended until temperatures had reduced sufficiently).</p> <p>SIGTEMP reflects the temperature warning status of the drive. SIGTEMP will be zero (0) if the drive is below its limit, and one (1) if the drive is above its limit. SIGTEMP can be monitored over the serial port if the TEMP output is not configured.</p>	
<b>See Also</b>	DTMP, OUTTEMP, TEMPLV, SIGTEMP	
<b>Example</b>	Command	Description
	>DTMPWNG 70	#Set the device to trigger a warning when the drive temperature exceeds
	DTMPWNG=70	70 degrees Celsius
	>DTMP	#Query the drive temperature value
	DTMP=60	#Displays the current drive temperature value
	>SIGTEMP	#Query temperature warning signal
	SIGTEMP=0	#SIGTEMP is zero because drive is below warning limits
	>	

**EC: Encoder Counter****System Status**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	EC n	
<b>Range</b>	n = -2,147,483,647 to 2,147,483,647	
<b>Initial Value</b>	0	
<b>Access</b>	READ and WRITE	
<b>Description</b>	Sets the internal encoder pulse counter to any value within range.	
<b>See Also</b>	PC, PCI, PF, PFI, PE, EGA, EGB	
<b>Note</b>	If the EC value is changed by user, then PC value is automatically modified based on the ratio of EGA and EGB. And vice versa.	
<b>Example</b>	Command	Description
	>EGA ; EGB	#Query the Encoder Electrical Gear Ratio
	EGA=500(500)	
	EGB=500(500)	
	>EC 0	#Set the encoder counter to zero
	EC=0	#Displays the encoder counter value
	>PC	#Query the PC
	PC=0	#Displays the position counter value
	>DIS 2000	#Set distance to 2000
	>MI	#Execute the Index Move
	>EC	#Query the EC value
	EC=2000	#Displays the encoder counter value
	>	

**ECHO: Communications Echo Control****Communications**

<b>Execution Mode</b>	Immediate																
<b>Syntax</b>	ECHO n																
<b>Range</b>	n = 0: OFF, Commands are suppressed and not shown on the terminal 1: ON, Commands are echoes back to the terminal																
<b>Initial Value</b>	1																
<b>SAVEPRM</b>	The new value takes effect immediately and the parameter is saved automatically.																
<b>Access</b>	READ and WRITE																
<b>Description</b>	<p>Allows or suppresses the display of any characters being sent to the terminal via the device's communication port. The ECHO command is useful when the device is used with an operator interface (OIT or HMI) or a Host Controller where the echoing (repeating) of the entered characters is not necessary.</p> <p>The ECHO command defines the device's echo back setting (ON/OFF) for the user entered ASCII data on the terminal.</p> <p>If ECHO=0(OFF), the device will send no response for the entered ASCII data to the terminal.</p> <p>The function of displaying the queried parameter value or SAS (Send ASCII String) command from a program is not affected by ECHO=0. The queried parameter values and the SAS command entries will display on the terminal with ECHO=0.</p>																
<b>See Also</b>	VERBOSE																
<b>Example</b>	<table> <thead> <tr> <th>Command</th><th>Description</th></tr> </thead> <tbody> <tr> <td>&gt;VS</td><td>#Query the Starting Velocity</td></tr> <tr> <td>VS=1000</td><td>#Device response</td></tr> <tr> <td>&gt;ECHO 0</td><td>#Turn off the ECHO</td></tr> <tr> <td>ECHO=0</td><td>#Device response</td></tr> <tr> <td>&gt;ECHO=0</td><td>#Query ECHO setting: Note actual query text not echoed back (just response)</td></tr> <tr> <td>&gt;VS=1000</td><td>#Query the Starting Velocity. Again, query doesn't show: just response.</td></tr> <tr> <td>&gt;</td><td></td></tr> </tbody> </table>	Command	Description	>VS	#Query the Starting Velocity	VS=1000	#Device response	>ECHO 0	#Turn off the ECHO	ECHO=0	#Device response	>ECHO=0	#Query ECHO setting: Note actual query text not echoed back (just response)	>VS=1000	#Query the Starting Velocity. Again, query doesn't show: just response.	>	
Command	Description																
>VS	#Query the Starting Velocity																
VS=1000	#Device response																
>ECHO 0	#Turn off the ECHO																
ECHO=0	#Device response																
>ECHO=0	#Query ECHO setting: Note actual query text not echoed back (just response)																
>VS=1000	#Query the Starting Velocity. Again, query doesn't show: just response.																
>																	

**EDIT: Edit Sequence****Sequence Management**

Execution Mode	Immediate	
Syntax	EDIT [target]	
Range	target (optional): any valid sequence number (0–63) or name (consisting of letters and numbers, up to 10 characters, starting with a letter)	
Description	Enters the sequence editor, where sequences can be created or modified.	
	Every sequence must have a unique number. If [target] is unspecified, or specified as a new name, EDIT automatically assigns the lowest unused sequence number to the new sequence.	
	The editor uses its own prompt (>>Command:). Editing operations are performed by entering a one character command, and any relevant arguments. The editor commands are listed below: this information is also available by entering 'H' at the editor prompt ([ ] indicates an optional argument).	
	The ESCAPE character can also be used to quit the sequence editor.	
	Editor Command	Description
	I [x]	Insert line(s) before line x (end of sequence if no x)
	A x [y]	Alter line(s) x, or x to y
	D x [y]	Delete line(s) x, or x to y
	L [x] [y]	List line(s). All, or x to end, or x to y
	X x [y]	Cut line(s) to clipboard. x, or x to y
Important Interactions	C [x] [y]	Copy line(s) to clipboard. All, or x, or x to y
	P x	Paste lines from clipboard, ahead of x
	S	Save sequence, to existing location
	S x	Save sequence, by number (0–63)
	S sss	Save sequence, by name (10 char max)
	M	Display memory status
	H	Display this help reminder
	Q	Quit sequence editor
	- A sequence named CONFIG will run automatically at power up of the device or after A RESET command has been issued.	
	- While the sequence editor is active, sequences cannot be executed. The START input will have no affect. Likewise, when sequences are executing, sequences cannot be edited (an attempt to edit will result in an error message).	
Example	Command	Description
	>EDIT 0	#Create (or modify) Sequence # 0
	New Sequence	
	#Device response	
	Sequence Name : <no name>	
	#Device response	
	Sequence Number : 0	
	#Device response	
	Lines : 0	
	#Device response	
Bytes : 0		
#Device response		
Bytes Free : 1600		
#Device response		
>>Command:		
#<ESC> is sent to exit the Editor		
>		
#Back at the main system prompt		

**EGA, EGB: Encoder Electrical Gear Ratio****System Control**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	EGA n EGB n	
<b>Range</b>	n = 1 to 250,000 (integer value)	
<b>Initial Value</b>	EGA=500 EGB=500	
<b>RESET</b>	RESET or recycle the power is required to activate the change. The parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	Sets the encoder electronic gear A and B. These parameters are used in the misstep detection function and the self correction function. It does not affect the encoder counter value.  EGA: Set the encoder resolution. For instance, set to 500 if the encoder pulse count per motor revolution is 500 P/R. See table below.  EGB: Set the motor resolution. For instance, set to 1000 (MRES = 1 when a basic step angle of 0.72 degree ) if the pulse count required for one motor revolution is 1000 P/R. See table below.	
<b>See Also</b>	EC	
<b>Note</b>	See table on next page for details.	
<b>Example</b>	Command > EGA 1000 EGA=500(1000) > EGB 2500 EGB=500(2500) > RESET Resetting system.	Description #Set the EGA value #Display the EGA value #Set the EGB value #Display the EGB value #RESET the device to initialize the modified DIRINV setting
-----		
CRD5xx-KP CRK Series Built-in Controller Software Version: A378 V.x.xx Copyright 2009-2011 ORIENTAL MOTOR U.S.A. CORP.		
-----		
	> EGA; EGB EGA=1000(1000) EGB=2500(2500) >	#Query the EGA, EGB #New values are available

**For Standard Type Step Motors with 500 steps/rev (0.72°/step) (CRK5□ types)**

Step angle (Steps/rev)	# of divisions	EGA parameter	EGB parameter	Step angle (Steps/rev)	# of divisions	EGA parameter	EGB parameter
0.72° (500)	1	500	500	0.0288° (12,500)	25	500	12500
0.36° (1000)	2	500	1000	0.018° (20,000)	40	500	20000
0.288° (1250)	2.5	500	1250	0.0144° (25,000)	50	500	25000
0.18° (2000)	4	500	2000	0.009° (40,000)	80	500	40000
0.144° (2500)	5	500	2500	0.0072° (50,000)	100	500	50000
0.09° (4000)	8	500	4000	0.00576° (62,500)	125	500	62500
0.072° (5000)	10	500	5000	0.0036° (100,000)	200	500	100000
0.036° (10,000)	20	500	10000	0.00288° (125,000)	250	500	125000

**For High Resolution Type Step Motors with 1000 steps/rev (0.36°/step) (CRK5□M types)**

Step angle (Steps/rev)	# of divisions	EGA parameter	EGB parameter	Step angle (Steps/rev)	# of divisions	EGA parameter	EGB parameter
0.36° (1000)	1	1000	1000	0.0144° (25,000)	25	1000	25000
0.18° (2000)	2	1000	2000	0.009° (40,000)	40	1000	40000
0.144° (2500)	2.5	1000	2500	0.0072° (50,000)	50	1000	50000
0.09° (4000)	4	1000	4000	0.0045° (80,000)	80	1000	80000
0.072° (5000)	5	1000	5000	0.0036° (100,000)	100	1000	100000
0.045° (8000)	8	1000	8000	0.00288° (125,000)	125	1000	125000
0.036° (10,000)	10	1000	10000	0.0018° (200,000)	200	1000	200000
0.018° (20,000)	20	1000	20000	0.00144° (250,000)	250	1000	250000

**ELSE: Begin ELSE Block: execute if IF is false****Program Control**

<b>Execution Mode</b>	Sequence	
<b>Syntax</b>	ELSE	
<b>Description</b>	Branches to an alternate operation if the preceding conditional IF statement is not true.	
<b>See Also</b>	IF, ENDIF, WHILE, WEND	
<b>Example</b>	Command	Description
	>LIST 5	#List sequence 5
	( 1) IF (IN1=1)	#If input #1 is on, then do line 2
	( 2) VR=20	#Running Velocity=20 pps
	( 3) MA 0	#Move Absolute to position 0
	( 4) ELSE	#Branch on not true, if line 1 is not true, then do line 5
	( 5) MGHN	#Seek home in the negative direction
	( 6) ENDIF	#End of IF block
	>	

**END: End Sequence****Sequence Commands**

<b>Execution Mode</b>	Sequence	
<b>Syntax</b>	END	
<b>Description</b>	The END statement can be used to formally terminate sequence text. END behaves exactly the same as a return statement (RET), but END, if used, must be the last statement in the sequence. Any text following the END statement will cause an error when attempting to save the sequence. END is provided for compatibility with other Oriental Motor products. Its use is strictly optional: a sequence does not need an END as its last statement.	
<b>See Also</b>	RET	
<b>Example</b>	Command	Description
	>LIST 5	#List sequence 5
	( 1) IF (IN1=1)	#If input #1 is on, then do line 2
	( 2) MCP	#Move continuously, positive direction
	( 3) ELSE	#Branch on not true, if line 1 is not true, then do line 5
	( 4) MCN	#Move continuously, negative direction
	( 5) ENDIF	#End of IF block
	( 6) END	#End of sequence: optional
	>	



**ENDIF: End of IF Block****Sequence Commands**

<b>Execution Mode</b>	Sequence	
<b>Syntax</b>	ENDIF	
<b>Description</b>	Indicates the completion of a conditional IF statement.	
<b>See Also</b>	IF, ELSE, WHILE, WEND	
<b>Example</b>	Command	Description
	>LIST 5	#List sequence 5
	( 1) IF (IN1=1)	#If input #1 is on, then do line 2
	( 2) MCP	#Move continuously, positive direction
	( 3) ELSE	#Branch on not true, if line 1 is not true, then do line 4
	( 4) MCN	#Move continuously, negative direction
	( 5) <b>ENDIF</b>	<b>#End of IF block</b>
	( 6) END	#End of sequence: optional
	>	

**ENDL: End of LOOP Block****Sequence Commands**

<b>Execution Mode</b>	Sequence	
<b>Syntax</b>	ENDL	
<b>Description</b>	Terminates the innermost LOOP block	
<b>See Also</b>	LOOP, BREAKL	
<b>Example</b>	Command	Description
	>LIST 5	#List sequence 5
	( 1) DIS=5000	#Distance equals 5000
	( 2) LOOP 5	#Loop the following 5 times
	( 3) MI	#Do an Index Move
	( 4) MEND	#Wait for the move to end before executing the next command
	( 5) WAIT 1.0	#Wait 1 second
	( 6) ENDL	#End the loop block
	>	

**EVx: Configure Event Output****I/O**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	EVx OUTy = z m = n or EVx 0	
<b>Range</b>	x: Event channel number; 1 or 2 y: Output number; 1 to 4 z: Output logic level after trigger; 0 or 1 m: Event trigger source T: Trigger n seconds after motion start; 0.000 to 500.000 (second) D: Trigger after moving distance n from motion start. n=0.000 to 8,388,607 V: Trigger after reaching speed set point n. n=1 to 500,000 pps	
<b>Description</b>	Configures events which control outputs on-the-fly. Up to 2 events can be configured and active at the same time, using both event channels 1 and 2 EVx 0 clears (deactivates) the event. Once an event has been configured, it remains active until cleared. Clearing the event does not clear or reset the output itself. Event checking restarts at the beginning of a motion. The designated output will be set to the designated state when the designated condition has been met. To detect the transition, assure that the designated output is in the opposite state prior to the event occurring. The output used should not have an assigned system output signal (e.g. if OUTREADY=3, do not use output 3 for events). If the output has been assigned to a system output signal, no event-driven transitions will occur on the output.	
<b>Example</b>	Command	Description
	>EV1 OUT2=1 V=10	#Turn on Output#2 when reach speed of 10 pps
	EV1 OUT2=1 V=10	
	>EV2 OUT1=1 T=2	#Turn on Output#1 2 seconds after motion starts
	EV2 OUT1=1 T=2	
	>MCP	#Execute a continuous move in the positive direction
	>EV1 0; EV2 0	#Clear events number 1 and 2
	>	

**HELP: Display Help Information****Monitor Commands**

<b>Execution Mode</b>	Immediate
<b>Syntax</b>	HELP
<b>Description</b>	Displays help information. Each screen displays the command Syntax and a brief description. The SPACE key on the keyboard lists the next HELP screen. Any other keyboard key will exit the HELP screen mode.

<b>Example</b>	<div> <div>Command</div> <div>&gt;HELP</div> <div> HELP : Help  VER : ROM Version Number  ALM : Show Alarm Code &amp; Alarm Record  ALMCLR : Alarm Clear  WNG : Show Warning Code &amp; Warning Record  WNGCLR : Warning Clear  DIRINV : Rotation Direction [0:+=CCW,1:+=CW]  STOEN : Step Out Detection Enable [0:Disable,1:Enable]  MRES : Motor Resolution [0-15]  EGA : Encoder Electrical Gear ratio A[1-250000]  EGB : Encoder Electrical Gear ratio B[1-250000]  LSEN : Hard Limit Enable [0:Disable,1:Enable]  AREA1 : AREA1 Position [-8388607..8388607step]  AREA2 : AREA2 Position [-8388607..8388607step]  CRRUN : Run Current Ratio [5..100 %]  CRSTOP : Stop Current Ratio [5..50 %]  HOMESEL : Homing Type [0:2sen,1:3sen]  HOMEVR : Homing Operation Speed [1..500000pps]  HOMETR : Homing Acc/Dec Time [0.001..1000.000s]  HOMEVS : Homing Start Speed [1..500000pps]  HOMEofs : Home Offset [-8388607..8388607step]  HOMEDIR : Homing Start Direction [0:-,1:+]  SLITEN : SLIT Enable at Homing [0:Disable,1:Enable]  TIMEN : TIM Enable at Homing [0:Disable,1:Tim Enable,2:Zsg Enable]  HOME2SB : 2sensor Homing Position Back Steps [0..32767step]  STOB : Step Out Detection Band [0.1..360.0deg]  DTMPWNG : Over Heat Warning [40..85 deg.C]  SLEN : Soft Limit Enable [0:Disable,1:Enable]  LIMP : Positive Soft Limit Position [-8388607..8388607step]  LIMN : Negative Soft Limit Position [-8388607..8388607step]  STOACT : Step out Action [0:none,1:Warning,2:Alarm]  MI : Move Incrementally  MA : Move Absolutely  CV :Change Velocity for Index  MCP : Move Continuous Positive  MCN : Move Continuous Negative  DIS : Incremental motion distance  VS : Starting Speed [1..500000pps]  VR : Running Velocity [1..500000pps]  TA : Common Acceleration Time [0.001..1000.000s]  TD : Common Deceleration Time [0.001..1000.000s] </div> </div> <div>Description</div> <div>#Display the Help information</div>
----------------	--

\*Continued on next page

PSTOP : Stop immediately, forcing ALARM  
 HSTOP : Stop immediately (hard stop)  
 SSTOP : Stop, decelerating (soft stop)  
 SCHGPOS : Distance from SENSOR on MCx  
 SCHGVR : Velocity on SCHGPOS motion  
 MI0 : Move via linked index, begin at linked index 0  
 MI1 : Move via linked index, begin at linked index 1  
 MI2 : Move via linked index, begin at linked index 2  
 MI3 : Move via linked index, begin at linked index 3  
 DIS0 : (-DIS3) Distance/Destination for linked index 'x' (x=0-3)  
 DIS1 :  
 DIS2 :  
 DIS3 :  
 VR0 : (-VR3) Velocity for linked index 'x' (x=0-3)  
 VR1 :  
 VR2 :  
 VR3 :  
 INCABS0 : (-INCABS3) Set positioning mode for index 'x' (x=0-3)  
 INCABS1 :  
 INCABS2 :  
 INCABS3 :  
 LINK0 : Configure link: linked index 0 and 1 (0:Link off/1:Link on)  
 LINK1 : Configure link: linked index 1 and 2 (0:Link off/1:Link on)  
 LINK2 : Configure link: linked index 2 and 3 (0:Link off/1:Link on)  
 PAUSE : Pause Motion  
 CONT : Resume Motion  
 PAUSECLR : Clear Paused Motion  
 MGHP : Find Home start in Positive direction  
 MGHN : Find Home, start in Negative direction  
 SCEN : Misstep Self Correcting Enable [0:Disable,1:Enable]  
 SCTO : Misstep Self Correcting Timeout [0-10.0s]  
 INSG : Display functional inputs [0-8191]  
 OUTSG : Display functional outputs [0-2047]  
 SIGPSTOP : Display input PSTOP [0,1]  
 SIGPAUSE : Display input PAUSE [0,1]  
 SIGPAUSECL : Display input PAUSECL [0,1]  
 SIGCROFF : Display input CROFF [0,1]  
 SIGHOME : Display input HOME start [0,1]  
 SIGHOMES : Display input HOME sensor [0,1]  
 SIGLSN : Display input -LS [0,1]  
 SIGLSP : Display input +LS [0,1]  
 SIGSENSOR : Display input SENSOR [0,1]  
 SIGSLIT : Display input SLIT [0,1]  
 SIGALMCLR : Display input ALMCLR [0,1]  
 SIGALM : Display output ALM [0,1]  
 SIGMOVE : Display output MOVE [0,1]  
 SIGRUN : Display output RUN [0,1]  
 SIGHOMEP : Display output HOMEP [0,1]  
 SIGPSTS : Display output PSTS [0,1]  
 SIGAREA : Display output AREA [0,1]

\*Continued on next page

SIGREADY : Display output READY [0,1]  
 SIGWNG : Display output WNG [0,1]  
 SIGTEMP : Display output TEMP [0,1]  
 SIGSTO : Display output STO [0,1]  
 SIGSC : Display output SC [0,1]  
 IN : Display general inputs [0-63]  
 IN1 : Display IN1 [0,1]  
 IN2 : Display IN2 [0,1]  
 IN3 : Display IN3 [0,1]  
 IN4 : Display IN4 [0,1]  
 IN5 : Display IN5 [0,1]  
 IN6 : Display IN6 [0,1]  
 OUT : Set/Clear/View all general outputs[0-15]  
 OUT1 : Set/Clear/View OUT1 level [0,1]  
 OUT2 : Set/Clear/View OUT2 level [0,1]  
 OUT3 : Set/Clear/View OUT3 level [0,1]  
 OUT4 : Set/Clear/View OUT4 level [0,1]  
 EV1 : EV1 parameter  
 EV2 : EV2 parameter  
     : Command Format; EVa OUTb=c d=e  
     : a; Event channel (1,2)  
     : b; Clear event (0) / OUT pin number (1-4)  
     : c: OUT level (0,1)  
     : d; Trigger source (Time, Velocity, Distance)  
     : e: Trigger level (Range depends on source)  
 CURRENT : Control Motor current (0:OFF/1:ON)  
 PC : Display command position  
 PE : Display position error (with encoder only)  
 PF : Display motor position (with encoder only)  
 EC : Display encoder counter  
 PCI : Display incremental command position  
 PFI : Display incremental feedback position(with encoder only)  
 VC : Display command velocity  
 DTMP : Display drive temperature [deg.C]  
 TIMER : Display general purpose timer [sec]  
 POS : Position data. POS[x] (x=1-64)  
 ALMSET : Set user alarm  
 ALMMSG : Alarm messaging (0:None/1:Alarm/2:Alarm+Warning)  
 RUN : Run sequence  
 ABORT : Abort sequence  
 DIR : Show sequence directory  
 LIST : List sequence  
 COPY : Copy sequence  
 DEL : Delete sequence  
 REN : Rename sequence  
 LOCK : Lock sequence  
 UNLOCK : Unlock sequence  
 EDIT : Edit sequence  
 TRACE : Control sequence tracing (0:OFF/1:ON)  
 A : (-Z) General purpose parameters  
 \*Continued on next page

B :  
 C :  
 D :  
 E :  
 F :  
 G :  
 H :  
 I :  
 J :  
 K :  
 L :  
 M :  
 N :  
 O :  
 P :  
 Q :  
 R :  
 S :  
 T :  
 U :  
 V :  
 W :  
 X :  
 Y :  
 Z :  
 ECHO : Control communications echo (0:OFF/1:ON)  
 VERBOSE : Control verbose responses (0:OFF/1:ON)  
 ABORTACT : ABORT Action [0:Hstop,1:Sstop]  
 OTACT : Over Travel Action [0:H,1:S]  
 ALMACT : ALARM option (1:Abort,C+ALM on/2:Abort,COFF,ALM on)  
 SENSORACT : SENSOR option (0:Hard Stop/1:Soft Stop/2:Offset Motion)  
 STARTACT : START option (0:Edge detect/1:Level detect+ABORT)  
 STRSW : Motor current at start up (0:OFF/1:ON)  
 INPAUSE : PAUSE pin number (0:No assignment/1-6:IN pin number)  
 INPAUSECL : PAUSECL pin number (0:No assignment/1-6:IN pin number)  
 PAUSELV : PAUSE input logic level (0:NO/1:NC)  
 PAUSECLLV : PAUSECL input logic level (0:NO/1:NC)  
 STARTLV : START input logic level (0:NO/1:NC)  
 ABORTLV : ABORT input logic level (0:NO/1:NC)  
 CROFFLV : CROFF input logic level (0:NO/1:NC)  
 OTLV : Hard limit input logic level (0:NO/1:NC)  
 HOMELV : HOME start input logic level (0:NO/1:NC)  
 HOMESLV : HOMES sensor input logic level (0:NO/1:NC)  
 SLITLV : SLIT input logic level (0:NO/1:NC)  
 ALMCLRLV : ALMCLR input logic level (0:NO/1:NC)  
 PSTOPLV : PSTOP input logic level (0:NO/1:NC)  
 SENSORLV : SENSOR input logic level (0:NO/1:NC)  
 OUTRUN : RUN pin number (0:No assignment/1-4:OUT pin number)  
 OUTHOME : HOME pin number (0:No assignment/1-4:OUT pin number)  
 OUTPSTS : PSTS pin number (0:No assignment/1-4:OUT pin number)  
 \*Continued on next page

OUTTEMP : TEMP pin number (0:No assignment/1-4:OUT pin number)  
 OUTAREA : AREA pin number (0:No assignment/1-4:OUT pin number)  
 OUTREADY : READY pin number (0:No assignment/1-4:OUT pin number)  
 OUTWNG : WNG pin number (0:No assignment/1-4:OUT pin number)  
 OUTSTO : STO pin number (0:No assignment/1-4:OUT pin number)  
 OUTSC : SC pin number (0:No assignment/1-4:OUT pin number)  
 MOVELV : MOVE output logic level (0:NO/1:NC)  
 ALMLV : ALM output logic level (0:NO/1:NC)  
 RUNLV : RUN output logic level (0:NO/1:NC)  
 HOMEPLV : HOMEPL output logic level (0:NO/1:NC)  
 PSTSLV : PSTS output logic level (0:NO/1:NC)  
 TEMPLV : TEMP output logic level (0:NO/1:NC)  
 AREALV : AREA output logic level (0:NO/1:NC)  
 READYLV : READY output logic level (0:NO/1:NC)  
 WNGLV : WNG output logic level (0:NO/1:NC)  
 STOLV : STO output logic level (0:NO/1:NC)  
 SCLV : SC output logic level (0:NO/1:NC)  
 IN1LV : (-IN6LV) input logic level, input 'x' (x=1-6) (0:NO/1:NC)  
 IN2LV :  
 IN3LV :  
 IN4LV :  
 IN5LV :  
 IN6LV :  
 OUT1LV : (-OUT4LV) output logic level,(0:NO/1:NC)  
 OUT2LV :  
 OUT3LV :  
 OUT4LV :  
 SAVEPRM : Save all parameters  
 INITPRI5RU\*QT Clear all sequences  
 CLEARPOS : Clear all POS[ ] data  
 CLEARALL : Initialize all parameters, clear all POS[ ], sequences  
 INITIO : Set all IN/OUT to general  
 REPORT : Display parameter report  
 TEACH : Begin Teaching target position data (POS[x])  
 OUTTEST : Simulate output, monitor input  
 IO : Display I/O status [bit]  
 RESET : Reset system



## HOME2SB: Backward steps in 2-sensor homing operation Motion Variables

---

<b>Execution Mode</b>	Immediate and Sequence
<b>Syntax</b>	HOME2SB n
<b>Range</b>	n = 0 to 32767
<b>Initial Value</b>	200
<b>SAVEPRM</b>	The new value takes effect immediately and the parameter is saved automatically.
<b>Access</b>	READ and WRITE
<b>Description</b>	Sets the travel amount after the motor pulls out from the LS sensor in 2-sensor homing operation.

### See Also

### Example

Command	Description
>HOME2SB 500	#Set the HOME2SB to 500
HOME2SB=500	#Display the HOME2SB value
>HOMESEL 0	#Set the HOMESEL to 2-sensor mode
HOMESEL=0	#Display the HOMESEL value
>MGHP	#Execute homing position in the positive direction
>	

**HOMEDIR: Homing Operation Direction****System Control**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	HOMEDIR n	
<b>Range</b>	n = 0: Negative Direction 1: Positive Direction	
<b>Initial Value</b>	1	
<b>SAVEPRM</b>	The new value takes effect immediately and the parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	Sets the starting direction for homing operation for the hardware input.	
<b>See Also</b>	HOMESEL, HOME2SB, HOMEofs, HOMEVR, HOMEVS, HOMETR	
<b>Note</b>	The MGHP and MGHn commands have higher priority than the HOMEDIR parameter until the device is RESET or power has been cycled.	
<b>Example</b>	Command	Description
	>HOMEDIR 0	#Set the HOMEDIR to 0 (negative)
	HOMEDIR=0	#Display the HOMEDIR value
	>MGHP	#Go home in positive direction
	>HOMEDIR	#Query the homing direction
	HOMEDIR=1	#Homing direction is positive
	>	

**HOMELV: HOME Input Level****I/O**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	HOMELV n	
<b>Range</b>	n = 0: Normally Open 1: Normally Closed	
<b>Initial Value</b>	0	
<b>RESET</b>	RESET or recycle the power is required to activate the change. The parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	Sets the active level for the HOME input.	
<b>See Also</b>	SIGHOME, HOMESEL, HOME2SB, HOME0FS, HOMEVR, HOMEVS, HOMETR	
<b>Example</b>	Command	Description
	>HOMELV=1 HOMELV=0(1) >RESET Resetting system.	#Set the HOME input logic to Normally Closed #RESET the device to initialize the modified HOME setting
-----		
CRD5xx-KP CRK Series Built-in Controller Software Version: A378 V.x.xx Copyright 2009-2011 ORIENTAL MOTOR U.S.A. CORP.		
-----		
>HOMELV HOMELV=1(1) >		#New value is active

**HOME OFS: Home Offset Position****Motion Variables**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	HOME OFS n	
<b>Range</b>	n = -8,388,607 to +8,388,607	
<b>Initial Value</b>	0	
<b>SAVEPRM</b>	The new value takes effect immediately and the parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	<p>HOME OFS is the distance to be moved as the last step of a mechanical home seeking operation (MGHN, MGHP). After the home seeking operation has established a valid home signal (or signal combination: see HOMESEL, SLITEN and TIMEN), the motor moves by the HOME OFS distance, sets that final position to be the origin (PC=0), and sets SIGHOMEP true (which will cause the HOMEP output to become active, if configured). The HOME OFS motion has start velocity HOMEVS, running velocity HOMEVR, and acceleration and deceleration times HOMETR. The default value of OFFSET is zero (0): the origin is established at the position where a valid home I/O signal pattern is found. Use OFFSET if the natural system origin differs from the home I/O signal location.</p>	
<b>See Also</b>	HOMESEL, SLITEN, TIMEN, HOMEVR, HOMEVS, HOMETR, HOMEDIR	
<b>Example</b>	Command	Description
	>HOMESEL 1	#3-sensor mode selected.
	HOMESEL=1	
	>HOME OFS -100	#OFFSET origin -100 from HOMES input
	HOME OFS=-100	
	>MGHP	#Seek mechanical home, approach from the positive direction.
	>SIGHOMES	#AFTER operation complete: check HOMES input
	SIGHOMES=0	#Input is inactive. We have moved away from the signal.
	>SIGHOMEP	#Check HOMEP output
	SIGHOMEP=1	#Signal is active. We are at PC=0 after a valid homing operation,
	>PC	#Check position counter PC.
	PC=0	#Origin. Expected position count after home.
	>MA 100	#Absolute move to 100
	>PC	#AFTER motion completes... check PC
	PC=100	#PC is 100
	>SIGHOMES	#Check Home input
	SIGHOMES=1	#Active. HOMES input and origin are separated by HOME OFS
	>	

**HOMEPLV: Home Position Output Level****I/O**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	HOMEPLV n	
<b>Range</b>	n = 0: Normally Open 1: Normally Closed	
<b>Initial Value</b>	0	
<b>RESET</b>	RESET or recycle the power is required to activate the change. The parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	Sets the active level for the HOMEPL output.	
<b>See Also</b>	SIGHOMEPL	
<b>Example</b>	Command	Description
	>HOMEPLV=1	#Set the HOMEPL output logic to Normally Closed
	HOMEPLV=0(1)	
	>RESET	#RESET the device to initialize the modified HOMEPL setting
	Resetting system.	
	-----	
	CRD5xx-KP	
	CRK Series Built-in Controller	
	Software Version: A378 V.x.xx	
	Copyright 2009-2011	
	ORIENTAL MOTOR U.S.A. CORP.	
	-----	
	>HOMEPLV	
	HOMEPLV=1(1)	#New value is active
	>	

**HOMESEL: Homing Type Selection****System Control**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	HOMESEL n	
<b>Range</b>	n = 0: 2-sensors type 1: 3-sensors type	
<b>Initial Value</b>	1	
<b>SAVEPRM</b>	The new value takes effect immediately and the parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	Set the type for homing operation.	
<b>See Also</b>	HOME OFS, SLITEN, TIMEN, LIMP, LIMN, SLEN	
<b>Example</b>	Command >LIMP 10000 LIMP=10000 >LIMN -10000 LIMN=-10000 >SLEN 1 SLEN=1 >HOMESEL 1 HOMESEL=1 >SLITEN 0 SLITEN=0 >TIMEN 0 TIMEN=0 >ALMMSG 2 ALMMSG=2 [Alarm+Warning] >MGHP >SIGHOMEP SIGHOMEP=1 >MCP >Over travel: software position limit detected. >PC PC=10000 >	Description #Set positive motion limit  #Set negative motion limit  #Set software limit enable  #Set Home type to 3-sensor.  #Set Slit sensor disable  #Set TIM signal disable  #Enable alarm messages  #Start seek mechanical home #MGHP finished, check HOMEP signal #Move continuously, positive #Detected limit #Check PC #Just over LIMP

**HOMESLV: Home Sensor Input Level****I/O**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	HOMESLV n	
<b>Range</b>	n = 0: Normally Open 1: Normally Closed	
<b>Initial Value</b>	0	
<b>RESET</b>	RESET or recycle the power is required to activate the change. The parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	Sets the active level for the HOMES input.	
<b>See Also</b>	SIGHOMES	
<b>Example</b>	Command	Description
	>HOMESLV=1	#Set the HOMES input logic to Normally Closed
	HOMESLV=0(1)	#RESET the device to initialize the modified HOMES
	>RESET	setting
	Resetting system.	
	-----	
	CRD5xx-KP	
	CRK Series Built-in Controller	
	Software Version: A378 V.x.xx	
	Copyright 2009-2011	
	ORIENTAL MOTOR U.S.A. CORP.	
	-----	
	>HOMESLV	
	HOMESLV=1(1)	#New value is active
	>	

**HOMETR: Acceleration/Deceleration Time for homing****Motion Variables**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	HOMETR n	
<b>Range</b>	n = 0.001 to 1000.000 (second)	
<b>Initial Value</b>	0.5	
<b>SAVEPRM</b>	The new value takes effect immediately and the parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	Sets the acceleration/deceleration rate for homing operation.	
<b>See Also</b>	MGHN, MGHP	
<b>Example</b>	Command	Description
	>LIST HOMING	#List sequence HOMING
	( 1) HOMEVS 100	#Start Homing Velocity: 100 pps
	( 2) HOMEVR 1000	#Running Homing Velocity: 1000 pps
	( 3) HOMETR 0.5	#Acceleration and Deceleration time: 0.5 sec
	( 4) MGHN	#Start seeking home in the negative direction
	( 5) MEND	#Wait for motion to complete
	>	



**HOMEVR: Running Velocity for homing operation****Motion Variables**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	HOMEVR n	
<b>Range</b>	n = 1 to 500,000 [pps]	
<b>Initial Value</b>	1000	
<b>SAVEPRM</b>	The new value takes effect immediately and the parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	Sets the operating speed for homing operation.	
<b>See Also</b>	MGHN, MGHP, HOMEVS, HOMETR, HOMEOPS	
<b>Example</b>	Command	Description
	>LIST HOMING	#List sequence HOMING
	( 1) HOMEVS 100	#Start Homing Velocity: 100 pps
	( 2) HOMEVR 1000	#Running Homing Velocity: 1000 pps
	( 3) HOMETR 0.5	#Acceleration and Deceleration time: 0.5 sec
	( 4) MGHN	#Start seeking home in the negative direction
	( 5) MEND	#Wait for motion to complete
	>	

**HOMEVS: Starting Velocity for homing operation****Motion Variables**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	HOMEVS n	
<b>Range</b>	n = 1 to 500,000 [pps]	
<b>Initial Value</b>	100	
<b>SAVEPRM</b>	The new value takes effect immediately and the parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	Sets the starting speed for homing operation.	
<b>See Also</b>	MGHN, MGHP, HOMEVR, HOMETR, HOMEOPS	
<b>Example</b>	Command	Description
	>LIST HOMING	#List sequence HOMING
	( 1) HOMEVS 100	#Start Homing Velocity: 100 pps
	( 2) HOMEVR 1000	#Running Homing Velocity: 1000 pps
	( 3) HOMETR 0.5	#Acceleration and Deceleration time: 0.5 sec
	( 4) MGHN	#Start seeking home in the negative direction
	( 5) MEND	#Wait for motion to complete
	>	

**HSTOP: Hard Stop****Motion Commands**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	HSTOP	
<b>Description</b>	HSTOP stops the motor as quickly as possible.	
<b>Caution</b>	The HSTOP command will attempt to cause the motor to stop rotating immediately. Use caution when stopping a high speed load using the HSTOP command. The actual distance traveled during a Hard Stop depends on velocity, load, and current settings.	
<b>See Also</b>	<ESC>, ABORT, ABORTACT, PSTOP, SSTOP	
<b>Note</b>	HSTOP should be used with care. At high speeds, or with high inertial loads, HSTOP may cause an alarm condition.	
<b>Example</b>	Command	Description
	>MCP	#Move the motor continuously in the positive direction
	>HSTOP	#Stop the motor as quickly as possible
	>	

**IF: Begin IF Block: execute if true****Sequence Commands**

<b>Execution Mode</b>	Sequence	
<b>Syntax</b>	IF (element1 {Conditional Operator} element2)	
<b>Description</b>	Conditional test and branch operation	
	Parentheses are required.	
	element1 and element2 may be any numeric variable available to sequences, or any numeric constant within the range -(Maximum Number) to +(Maximum Number).	
	Valid conditional operators are:	
	= : Equal to	
	!= : Not equal to	
	< : Less than	
	<= : Less than or equal to	
	> : Greater than	
	>= : Greater than or equal to	
<b>See Also</b>	IF statements must be followed (at some point) by a corresponding ENDIF statement, forming an IF "block". An ELSE statement may appear within the IF block.	
	When executed, the conditional expression is evaluated. If it evaluates to TRUE, sequence processing proceeds to the statement following the IF. If it evaluates to FALSE, sequence processing proceeds to the statement following the next ELSE (if used) or ENDIF (if ELSE is not used).	
	Block structures (IF-ENDIF, WHILE-WEND, LOOP-ENDL) may be nested, to six (6) levels deep.	
	ELSE, ENDIF, WHILE, BREAKW, WEND, LOOP, BREAKL, ENDL	
<b>Note</b>	Be careful when using MCP/MCN command. Refer to "10-3. Continuous Motions".	
<b>Example</b>	Command	Description
	>LIST 7	#List sequence 7
	( 1) IF (PC>25000)	#Compare position to 25000 steps
	( 2) SSTOP	#If true, soft stop...
	( 3) MEND	#Wait for motion to finish
	( 4) SAS End of motion	#Transmit "End of motion". Finished.
	( 5) ELSE	#Otherwise...
	( 6) IF (SIGTEMP=1)	#Check SIGTEMP temperature warning
	( 7) SSTOP	#If true, soft stop...
	( 8) MEND	#wait for motion to end
	( 9) CURRENT 0	#Turn current OFF
	( 10) SAS Cooling	#Transmit "Cooling". Done.
	( 11) ENDIF	#Close inner IF block
	( 12) ENDIF	#Close outer IF block
	>	

**IN: General Input Status****I/O**

<b>Execution Mode</b>	Immediate and Sequence
<b>Syntax</b>	IN
<b>Range</b>	0 to 63 (integer value)
<b>Description</b>	The IN command displays the current status of all the general purpose Inputs, as one integer number.

The general purpose inputs contribute to the value of IN as follows:

INx	Contribution to IN if active
IN6	32
IN5	16
IN4	8
IN3	4
IN2	2
IN1	1

For example, if IN=14 then Input #2 (2) is ON, Input #3 (4) is ON and Input#4 is ON (8).  
(2+4+8=14)

To check the status of a single general input, use the INx command.

**See Also** INITIO, INSG, INx, INxLV, IO, OUT, OUTSG, OUTTEST, OUTx, REPORT

**Important Interactions** If an input is assigned to a system input signal (INPAUSE, INPAUSECL) the IN command will always show that input OFF or 0. Inputs which have been assigned to system input signals do not affect IN. Use the INSG command to read the status of the assigned system input signals.

<b>Example</b>	Command	Description
	>IN	#Query the status of the general inputs
	IN=32	#Device response indicating Input #6 is ON
	>LIST 8	#List sequence 8
	( 1) SAS PRESS START	#Notify user to press start
	( 2) IF (IN=18)	#If Inputs #2 and #5 are ON then,
	( 3) MGHN	#Go home in the negative direction
	( 4) ELSE	#If the value of IN does not equal 18, then
	( 5) WHILE (IN=0)	#While all the inputs are OFF
	( 6) MI	#Execute an Index Move
	( 7) MEND	#Wait for move to complete
	( 8) WAIT 0.15	#Wait an additional 0.15 seconds
	( 8) WEND	#End the WHILE loop
	( 9) ENDIF	#End the IF block
	>	

**INCABSx: Linked Move Type****Motion Variables**

<b>Execution Mode</b>	Immediate and Sequence																																						
<b>Syntax</b>	INCABSx n																																						
<b>Range</b>	x = 0 to 3 (Linked Motion Profiles defined by DISx, INCABSx, VRx) n = 0: Absolute 1: Incremental																																						
<b>Initial Value</b>	1																																						
<b>SAVEPRM</b>	The new value takes effect immediately. However, SAVEPRM is required to save the parameter values in nonvolatile memory. Otherwise, the parameter value is reset to the last saved value at device power up. If no new parameter value was saved, then the value is set to the default (initial) value.																																						
<b>Access</b>	READ and WRITE																																						
<b>Description</b>	INCABSx determines whether DISx represents a distance or an absolute destination for linked index (Mlx) motion commands.																																						
<b>See Also</b>	DISx, VRx, LINKx, Mlx, TA, TD, VS																																						
<b>Important Interactions</b>	<p>Each of the four links can be incremental or absolute. Incremental and absolute can be used in combination, but all links executed together must move in the same direction.</p> <ul style="list-style-type: none"> <li>- For incremental links, motion direction is determined by the arithmetic sign of DIS.</li> <li>- For absolute links, motion direction is determined by the motor position at the start of that motion link.</li> </ul> <p>Generally, absolute links are not recommended when the motor position before linked operation cannot be predicted.</p>																																						
<b>Example</b>	<table> <thead> <tr> <th>Command</th><th>Description</th></tr> </thead> <tbody> <tr> <td>&gt;VR1 500</td><td>#Set the velocity for linked move #1 to 500 pps</td></tr> <tr> <td>VR1=500</td><td>#Device response</td></tr> <tr> <td>&gt;DIS1 2000</td><td>#Set the distance for linked move #1 to 2000</td></tr> <tr> <td>DIS1=2000</td><td>#Device response</td></tr> <tr> <td>&gt;INCABS1 1</td><td>#Set the move type for linked motion #1 to incremental</td></tr> <tr> <td>INCABS1=1 [INC]</td><td>#Device response</td></tr> <tr> <td>&gt;LINK1 1</td><td>#Enable the linked operation for motion #1</td></tr> <tr> <td>LINK1=1</td><td>#Device response</td></tr> <tr> <td>&gt;VR2 1000</td><td>#Linked move #2 velocity equals 1000 pps</td></tr> <tr> <td>VR2=1000</td><td>#Device response</td></tr> <tr> <td>&gt;INCABS2 1</td><td>#Set the move type for linked motion #2 to incremental</td></tr> <tr> <td>INCABS2=1 [INC]</td><td>#Device response</td></tr> <tr> <td>&gt;DIS2 4000</td><td>#Linked move #2: destination is position 4000</td></tr> <tr> <td>DIS2=4000</td><td>#Device response</td></tr> <tr> <td>&gt;LINK2 0</td><td>#"Unlink" link2 from link3</td></tr> <tr> <td>LINK2=0</td><td>#Device response</td></tr> <tr> <td>&gt;MI1</td><td>#Start the linked operation motion</td></tr> <tr> <td>&gt;</td><td></td></tr> </tbody> </table>	Command	Description	>VR1 500	#Set the velocity for linked move #1 to 500 pps	VR1=500	#Device response	>DIS1 2000	#Set the distance for linked move #1 to 2000	DIS1=2000	#Device response	>INCABS1 1	#Set the move type for linked motion #1 to incremental	INCABS1=1 [INC]	#Device response	>LINK1 1	#Enable the linked operation for motion #1	LINK1=1	#Device response	>VR2 1000	#Linked move #2 velocity equals 1000 pps	VR2=1000	#Device response	>INCABS2 1	#Set the move type for linked motion #2 to incremental	INCABS2=1 [INC]	#Device response	>DIS2 4000	#Linked move #2: destination is position 4000	DIS2=4000	#Device response	>LINK2 0	#"Unlink" link2 from link3	LINK2=0	#Device response	>MI1	#Start the linked operation motion	>	
Command	Description																																						
>VR1 500	#Set the velocity for linked move #1 to 500 pps																																						
VR1=500	#Device response																																						
>DIS1 2000	#Set the distance for linked move #1 to 2000																																						
DIS1=2000	#Device response																																						
>INCABS1 1	#Set the move type for linked motion #1 to incremental																																						
INCABS1=1 [INC]	#Device response																																						
>LINK1 1	#Enable the linked operation for motion #1																																						
LINK1=1	#Device response																																						
>VR2 1000	#Linked move #2 velocity equals 1000 pps																																						
VR2=1000	#Device response																																						
>INCABS2 1	#Set the move type for linked motion #2 to incremental																																						
INCABS2=1 [INC]	#Device response																																						
>DIS2 4000	#Linked move #2: destination is position 4000																																						
DIS2=4000	#Device response																																						
>LINK2 0	#"Unlink" link2 from link3																																						
LINK2=0	#Device response																																						
>MI1	#Start the linked operation motion																																						
>																																							

**INITIO: Initialize I/O****I/O**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	INITIO	
<b>Description</b>	<p>Cancels all Input or Output assignments.</p> <p>All system input signal assignment values (INPAUSE, INPAUSECL) and all system output signal assignment values (OUTREADY, OUTWNG, etc) are set to zero (0), unassigned.</p> <p>All Inputs and Outputs are reset for general purpose use.</p> <p>The command must be confirmed before it executes. A RESET is then required for this command to take effect. The old I/O assignments remain effective until A RESET is executed.</p> <p>INITIO does not change any signal level assignments (e.g. PAUSELV, etc.).</p>	
<b>See Also</b>	IN, INSG, INx, IO, OUT, OUTSG, OUTTEST, OUTx,	
<b>Example</b>	<p>Command</p> <p>&gt;INITIO</p> <p>Enter Y to proceed, other key to cancel. Y</p> <p>5(0)</p> <p>0(0)</p> <p>0(0)</p> <p>0(0)</p> <p>4(0)</p> <p>1(0)</p> <p>3(0)</p> <p>2(0)</p> <p>6(0)</p> <p>0(0)</p> <p>0(0)</p> <p>0(0)</p> <p>0(0)</p> <p>0(0)</p> <p>0(0)</p> <p>0(0)</p> <p>0(0)</p> <p>All I/O configurations are set to factory default.</p> <p>Execute RESET to activate new settings.</p> <p>&gt;</p>	<p>Description</p> <p>#RESET the current IO assignment to factory settings</p> <p>#Device response</p> <p>#Device response</p>

**INITPRM: Initialize Parameters****System Control**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	INITPRM	
<b>Description</b>	Reprograms all parameters to the original factory default setting. Execute A RESET command after INITPRM to activate the default settings. INITPRM cannot be executed while the motor is moving or a sequence is executing.	
<b>Caution</b>	When parameters are initialized to factory default settings, all previous values are lost. The INITPRM command writes to EEPROM. The EEPROM has a nominal expected lifetime of 100,000 write cycles. The INITPRM command should not be	
<b>See Also</b>	CLEARALL, CLEARPOS, CLEARSEQ, INITIO	
<b>Example</b>	Command >INITPRM Enter Y to proceed, other key to cancel. y Initializing Parameters..OK.  >RESET Resetting system. ----- CRD5xx-KP CRK Series Built-in Controller Software Version: A378 V.x.xx Copyright 2009-2011 ORIENTAL MOTOR U.S.A. CORP. ----- >	Description #RESET all of the motion parameters to default values #Once confirmed, memory overwritten, old values lost.  #RESET required to activate new factory default settings. #Ready



**INPAUSE: PAUSE Signal Input Assignment****I/O**

<b>Execution Mode</b>	Immediate
<b>Syntax</b>	INPAUSE n
<b>Range</b>	n = 0 to 6
<b>Initial Value</b>	0 (Unassigned)
<b>RESET</b>	RESET or recycle the power is required to activate the change. The parameter is saved automatically.
<b>Access</b>	READ and WRITE
<b>Description</b>	<p>INPAUSE assigns the PAUSE (Pause Motion) system input signal to one of the input pins.</p> <p>This signal can be assigned to any of the 6 inputs.</p> <p>If configured, the motor will stop when the signal becomes active (soft stop), but stay prepared to resume the same motion, if a CONT (continue) command is executed. A START input will also resume the motion. A PAUSECL (Pause Clear) input or PAUSECLR command clears the pause condition, effectively "forgetting" the remainder of the previously paused motion.</p> <p>The active level of the PAUSE input is determined by PAUSELV. Motion can be started, even with the PAUSE input active: the transition from inactive to active state triggers PAUSE action.</p> <p>A PAUSE input does not pause or suspend sequences.</p> <p>While motion is PAUSE'd, the system output signal PSTS (Pause Status) is true (1), and, if configured, the PSTS output is active. The signal becomes false and the output inactive if the motion is continued or the pause condition cleared.</p> <p>The PAUSE command performs the same function as a PAUSE input.</p>
<b>See Also</b>	CONT, IN, INxLV, INPAUSECL, IO, PAUSE, PAUSECLLV, PAUSELV, OUTPSTS, SIGPAUSE, SIGPAUSECL

Example	Command	Description
	>INPAUSE 6	#Assign the PAUSE input to Input #6
	INPAUSE = 0 (6)	
	>RESET	#Establish the saved parameter value
	Resetting system.	
	-----	
	CRD5xx-KP	
	CRK Series Built-in Controller	
	Software Version: A378 V.x.xx	
	Copyright 2009-2011	
	ORIENTAL MOTOR U.S.A. CORP.	
	-----	
	>INPAUSE	#Confirm new value
	INPAUSE=6(6)	
	>	

**INPAUSECL: Pause Clear Signal Input Assignment****I/O**

<b>Execution Mode</b>	Immediate
<b>Syntax</b>	INPAUSECL n
<b>Range</b>	n = 0 to 6
<b>Initial Value</b>	0 (Unassigned)
<b>RESET</b>	RESET or recycle the power is required to activate the change. The parameter is saved automatically.
<b>Access</b>	READ and WRITE
<b>Description</b>	INPAUSECL assigns the PAUSECL (Clear Pause) system input signal to one of the input pins.

This signal can be assigned to any of the 6 inputs.

If configured, a PAUSE condition will be cleared when the signal becomes active. If a motion had been PAUSE'd (by a PAUSE input or PAUSE command), the remainder of that motion is "forgotten": the motion cannot be continued. When a PAUSE condition is cleared, system output signal SIGPSTS becomes false (0). If configured, the PSTS output becomes inactive.

The active level of the PAUSE input is determined by PAUSELV. Motion can be started, even with the PAUSE input active: the transition from inactive to active state triggers PAUSE action.

**See Also** CONT, INPAUSE, IO, PAUSE, PAUSECLLV, PAUSECLR, PAUSELV, OUTPSTS, SIGPAUSE, SIGPAUSECL, SIGPSTS

Example	Command	Description
	>INPAUSECL 4	#Assign the PAUSECL input to Input #4
	INPAUSECL = 0 (4)	
	>RESET	#Establish the saved parameter value
	Resetting system.	

-----

CRD5xx-KP  
 CRK Series Built-in Controller  
 Software Version: A378 V.x.xx  
 Copyright 2009-2011  
 ORIENTAL MOTOR U.S.A. CORP.

-----

>INPAUSECL	#Confirm new value
INPAUSECL=4(4)	
>	

**INSG: System Input Signal Status****I/O**

<b>Execution Mode</b>	Immediate and Sequence
<b>Syntax</b>	INSG
<b>Range</b>	0 to 8191 (integer value)
<b>Access</b>	READ
<b>Description</b>	The INSG command displays the current status of all the system input signals, as one integer number.

The system input signals contribute to the value of INSG as follows:

Bit Location	Signal	Contribution to INSG if active
Bit 0	START	1
Bit 1	ABORT	2
Bit 2	PSTOP	4
Bit 3	HOME	8
Bit 4	+LS	16
Bit 5	-LS	32
Bit 6	HOMES	64
Bit 7	SENSOR	128
Bit 8	PAUSE	256
Bit 9	PAUSECL	512
Bit 10	CROFF	1024
Bit 11	ALMCLR	2048
Bit 12	SLIT	4096

INSG is the sum of the contribution of all active signals:

If INSG=2, the ABORT signal is active, and all other signals are inactive.

If INSG=192, the HOMES (64) and SENSOR (128) signals are active (64+128=192), and all other signals are inactive.

Be careful not to confuse INSG with IN (Input Status). IN reports the status of General Purpose Inputs (those inputs which are not assigned to a signal). INSG reports the status of system input signals.

**See Also** IN, INxLV, IO, OUTSG

<b>Example</b>	Command	Description
	>INSG	#Query the current Input Signal Value
	INSG=1024	#Device response: the CROFF signal is active
	>	

**INx: Individual General Input Status****I/O**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	INx	
<b>Range</b>	x = 0 to 6 0: Not Active 1: Active	
<b>Initial Value</b>	0	
<b>Access</b>	READ	
<b>Description</b>	INx returns the state of General Purpose Input “x”. The active level of each General Purpose Input is determined by INxLV. If the input has been assigned to a system input signal, then it is no longer "General Purpose". INx for these inputs will always return 0 (Not Active). Use the INSG command to check the status of the system input signals.	
<b>See Also</b>	INITIO, INSG, INxLV, IO, OUT, OUTSG, OUTTEST, OUTx	
<b>Example</b>	Command	Description
	>LIST JOG	#List sequence named "JOG"
	( 1) TA= 0.1; TD=0.1; VS=100; VR=2000	#Set motion parameters
	( 2) LOOP	#Start infinite loop
	( 3) IF (IN1=1)	#If input 1 is active
	( 4) MCP	#Move continuous, positive
	( 5) WHILE (IN1=1); WEND	#Wait for input 1 to clear
	( 6) SSTOP	#Soft Stop
	( 7) MEND	#Wait for stop to complete
	( 8) ENDIF	#End of IF block
	( 9) IF (IN2=1)	#If input 2 is active
	( 10) MCN	#Move continuous, negative
	( 11) WHILE (IN2=1); WEND	#Wait for input 2 to clear
	( 12) SSTOP	#Soft Stop
	( 13) MEND	#Wait for stop to complete
	( 14) ENDIF	#End of IF block
	( 15) ENDL	#End of LOOP block
	>	

**INxLV: INx Input Level****I/O**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	INxLV n	
<b>Range</b>	x = 0 to 6 n = 0: Normally Open 1: Normally Closed	
<b>Initial Value</b>	0	
<b>RESET</b>	RESET or recycle the power is required to activate the change. The parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	INxLV establishes the active level of General Purpose Input x. If input x has been assigned to a system input signal, then INxLV has no affect: the active level assigned to the signal is used.	
<b>See Also</b>	INITIO, INSG, INx, IO, OUT, OUTSG, OUTTEST, OUTx	
<b>Example</b>	Command	Description
	>IN1LV=1	#Set the IN1 input logic to Normally Closed
	IN1LV=0(1)	#RESET the device to initialize the modified IN1 setting
	>RESET	
	Resetting system.	
	-----	
	CRD5xx-KP CRK Series Built-in Controller Software Version: A378 V.x.xx Copyright 2009-2011 ORIENTAL MOTOR U.S.A. CORP.	
	-----	
	>IN1LV	
	IN1LV=1(1)	#New value is active
	>	

**IO: Input/Output Status****I/O****Execution Mode** Immediate**Syntax** IO**Description** IO displays the current status of General Purpose Inputs and Outputs and system input signals and system output signals.

Values are reported as 0:Inactive or 1:Active.

For inputs and outputs that have been assigned to a system input or output signal, the signal state is shown. A START input can start a sequence, determined by the binary value of IN. This value is shown in the I/O response under (SEQ#), and is the number of the sequence that would start if a START signal became active in this I/O state.

In the example below, Input 1-4 and Output 1-2 remain General Purpose and INPAUSE=5, INPAUSECL=6. All other I/O have been assigned to system signals. PSTOP (Panic Stop) is asserted, and ALARM output is set. General Purpose Input #1 is active, so IN=1, and Sequence 1 would start if the alarm condition were cleared and START became active.

**See Also** ABORTLV, ALMLV, ALMCLRLV, AREALV, CROFFLV, HOMELV, HOMEPLV, HOMESLV, HOMEPLV, INITIO, INPAUSE, INPAUSECL, IN, INxLV, MOVELV, OTLV, OUTAREA, OUTPSTS, OUTREADY, OUTRUN, OUTSC, OUTSTO, OUTTEMP, OUTWNG, PAUSECLLV, PAUSELV, PSTOPLV, PSTSLV, READYLV, RUNLV, SCLV, SENSORLV, SLITLV, STARTLV, STOLV, TEMPLV, WNGLV**Example**

Command	Description
>IO	#Display the IO status
Input : IN1 IN2 IN3 IN4 PAUSE PAUSECL	#Device response
: START ABORT ALMCLR CROFF HOME +LS -LS PSTOP SENSOR HOMES SLIT	
Output : ALM MOVE OUT1 OUT2 HOMEP STO	
--Inputs--- --- Dedicated IO signals ---- Outputs	
1 2 3 4 5 6 -(SEQ#)- S A A C H + - P S H S	-- A M 1 2 3 4
1 0 0 0 0 0 -( 1)- 0 0 0 0 0 0 0 0 1 0 0 0	-- 1 0 0 0 0 0
>	

**KB: Keyboard Input****Sequence Commands**

<b>Execution Mode</b>	Sequence	
<b>Syntax</b>	variable = KB	
<b>Range</b>	variable refers to any numeric variable which sequences can write to. Actual permitted range depends on variable	
<b>Description</b>	<p>KB transmits a data entry prompt over the serial port, accepts a numeric value from the serial port, and assigns that value to variable.</p> <p>The data entry prompt consists of a question mark and a space. The sequence waits for a valid numeric entry, terminated by any of (CR, LF, CR+LF, or LF+CR).</p> <p>If the data is not a valid numeric value (e.g. alphabetic text), the system retransmits the data entry prompt, and waits for a new entry.</p> <p>If the data is a valid numeric value, but represents an invalid value for the designated variable because of range or precision limits, an alarm will be triggered and sequence processing will stop.</p> <p>Sequence execution is effectively suspended while waiting to receive a valid numeric value.</p> <p>For similar operation without prompting, see KBQ (Keyboard Input Quiet).</p> <p>KB and KBQ are provided to enable interactive sequence operation when connected with a host computer, PLC, touch panel, etc. via the serial port. Along with normal variable display responses (which include extra characters), the VIEW command can be used to transmit a variable's value without extra characters.</p> <p>SAS (Send ASCII String) and SACS (Send ASCII Control String) can be used to transmit text information (with and without extra characters, respectively). Taken together, a complete interactive serial interface can be implemented.</p>	
<b>See Also</b>	KBQ, SAS, SACS, VIEW	
<b>Example</b>	<p>Command</p> <p>&gt;LIST 9</p> <p>( 1) VR=1000</p> <p>( 2) SACS How far do you want to go</p> <p>( 3) <b>DIS=KB</b></p> <p>( 4) DIS</p> <p>( 5) MI</p> <p>( 6) MEND</p> <p>&gt;RUN 9</p> <p>&gt;How far do you want to go? 2000</p> <p>2000</p> <p>&gt;</p>	<p>Description</p> <p>#List sequence 9</p> <p>#Set running velocity</p> <p>#Prompt user to enter desired distance</p> <p><b>#Output ? and wait for new value</b></p> <p>#Distance equals the entry value (KB).</p> <p>#Execute an Index Move of DIS steps</p> <p>#Wait for motion to end.</p> <p>#Execute sequence #9</p> <p>#Line 2 text, and numeric entry from Line 3</p> <p>#The distance value is displayed.</p> <p>#Motor moves 2000 steps</p>
<b>Note</b>	In a multi-drop configuration, all output from sequence commands is suppressed unless the device has been previously addressed (via @). The KB and KBQ commands will not receive input unless the device has been previously addressed.	

**KBQ: Keyboard Input (Quiet)****Sequence Commands**

<b>Execution Mode</b>	Sequence	
<b>Syntax</b>	variable = KBQ	
<b>Range</b>	variable refers to any numeric variable which sequences can write to. Actual permitted range depends on variable	
<b>Description</b>	<p>KBQ accepts a numeric value from the serial port, and assigns that value to variable.</p> <p>The sequence waits for a valid numeric entry, terminated by any of (CR, LF, CR+LF, or LF+CR). If the data is not a valid numeric value (e.g. alphabetic text), the data is ignored: the system continues to wait for a new entry.</p> <p>If the data is a valid numeric value, but represents an invalid value for the designated variable because of range or precision limits, an alarm will be triggered and sequence processing will stop.</p> <p>Sequence execution is effectively suspended while waiting to receive a valid numeric value.</p> <p>KBQ operation is essentially the same as for KB, without the leading prompt or trailing CR+LF pair. KBQ permits tighter control of serial output for applications requiring exact character-by-character control.</p> <p>KB and KBQ are provided to enable interactive sequence operation when connected with a host computer, PLC, touch panel, etc. via the serial port. Along with normal variable display responses (which include extra characters), the VIEW command can be used to transmit a variable's value without extra characters.</p> <p>SAS (Send ASCII String) and SACS (Send ASCII Control String) can be used to transmit text information (with and without extra characters, respectively). Taken together, a complete interactive serial interface can be implemented.</p>	
<b>See Also</b>	KB, SAS, SACS, VIEW	
<b>Example</b>	Command	Description
	>LIST 10	#List sequence 10
	( 1) VR=1000	#Set running velocity
	( 2) SACS How far do you want to go?	#Prompt user: Append ? and trailing space
	( 3) DIS=KBQ	#Wait for new value
	( 4) SACS ^M^JMoving :	#Transmit CR, LF, text
	( 5) VIEW DIS	#Transmit DIS value, no extra text
	( 6) MI	#Move incrementally, new DIS distance
	( 7) MEND	#Wait for motion to end.
	>RUN 10	#Execute sequence #10
	>How far do you want to go? -3750	#Line 2 text, and numeric entry from Line 3
	Moving :-3750	#Exact output of lines 4 and 5
		#Motor moves -3750 steps
<b>Note</b>	In a multi-drop configuration, all output from sequence commands is suppressed unless the device has been previously addressed (via @). The KB and KBQ commands will not receive input unless the device has been previously addressed.	



**LIMN, LIMP: Software Position Limits****System Control**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	LIMN n: Minimum permitted position LIMP n: Maximum permitted position	
<b>Range</b>	n = -8,388,607 to +8,388,607	
<b>Initial Value</b>	LIMN = -8,388,607, LIMP = +8,388,607	
<b>SAVEPRM</b>	The new value takes effect immediately and the parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	<p>When SLEN=1, software position limits LIMN and LIMP are enforced, provided the system has completed a homing action (MGHP, MGHN, HOME input or by setting PC=0).</p> <p>Moving outside software position limit range will cause the motor to stop, may cause an alarm (alarm code: 67h) and may disable motor current, depending on the value of ALMACT. Stop action (soft stop or hard stop) is fixed to a soft stop.</p> <p>Software limit checking is disabled while a homing operation is in process (MGHP, MGHN, HOME input).</p> <p>(A software position limit alarm may be triggered after a homing operation if PC=0 is not between LIMN and LIMP.)</p> <p>For absolute or incremental index moves (MA, MI), limit checking is performed before motion starts. If the final target position is outside the range, the motion will not occur, and the action defined by ALMACT will trigger.</p> <p>For continuous motions (MCN, MCP), any out of range condition is detected only as it happens. If the system is outside the software position limits, motions may still be started. After any alarm is cleared, MI or MA can be executed if their destination would bring the motor within limits. MCN or MCP can be executed, if the motor would move in the direction of the operational range.</p>	
<b>See Also</b>	SLEN, PC, MGHP, MGHN, ALM, ALMACT	
<b>Example</b>	Command >LIMP 10000 LIMP=10000 >LIMN -10000 LIMN=-10000 >SLEN 1 SLEN=1 >HOMESEL 1 HOMESEL=1 >SLITEN 0 SLITEN=0 >TIMEN 0 TIMEN=0 >ALMMSG 2 ALMMSG=2 [Alarm+Warning] >MGHP >SIGHOMEP SIGHOMEP=1 >MCP >Over travel: software position limit detected. >PC PC=10000 >	Description #Set positive motion limit  #Set negative motion limit  #Set software limit enable  #Set Home type to 3-sensor.  #Set Slit sensor disable  #Set TIM signal disable  #Enable alarm messages  #Start seek mechanical home #MGHP finished, check HOMEP signal #Move continuously, positive #Detected limit #Check PC #Just over LIMP

**LINKx: Link Control****Motion Variables**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	LINKx n	
<b>Range</b>	x = 0 to 2 (Linked Motion Profiles defined by DISx, INCABSx, VRx) n = 0: Segment (x) terminates motion 1: Link segment (x) to segment (x+1)	
<b>Initial Value</b>	0	
<b>SAVEPRM</b>	The new value takes effect immediately. However, SAVEPRM is required to save the parameter values in nonvolatile memory. Otherwise, the parameter value is reset to the last saved value at device power up. If no new parameter value was saved, then the value is set to the default (initial) value.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	LINKx control whether linked motion segment x is linked to the next segment, or not. If LINKx=0, the motion segment defined by DISx and VRx will terminate. If LINKx=1, the motion segment defined by DISx and VRx will not terminate: motion will proceed to motion segment (x+1).	
<b>See Also</b>	DISx, INCABSx, Mlx, TA, TD, VRx, VS	
<b>Example</b>	Command	Description
	>VR1 500	#Set the velocity for linked move #1 to 500 pps
	VR1=500	#Device response
	>DIS1 2000	#Set the distance for linked move #1 to 2000
	DIS1=2000	#Device response
	>INCABS1 1	#Set the move type for linked motion #1 to incremental
	INCABS1=1 [INC]	#Device response
	>LINK1 1	#Enable the linked operation for motion #1
	LINK1=1	#Device response
	>VR2 1000	#Linked move #2 velocity equals 1000 pps
	VR2=1000	#Device response
	>INCABS2 1	#Set the move type for linked motion #2 to incremental
	INCABS2=1 [INC]	#Device response
	>DIS2 4000	#Linked move #2: destination is position 4000
	DIS2=4000	#Device response
	>LINK2 0	#Unlink link2 from link3
	LINK2=0	#Device response
	>MI1	#Start the linked operation motion
	>	

**LIST: List Sequence Contents****Sequence Management**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	LIST target [start line] [end line]	
<b>Range</b>	target can be the name or number of any existing sequence [start line] is an optional line number. [end line] is an optional line number, if [startline] is specified. If given, it must not be less than [start line].	
<b>Description</b>	LIST lists the contents of a stored sequence. If [start line] and [end line] are not specified, the entire sequence is listed. If [start line] is specified, output starts with line [start line]. If [end line] is specified, output ends after line [end line].	
<b>See Also</b>	DIR, EDIT	
<b>Example</b>	Command	Description
	>LIST TEMPCHECK 6 11	#List sequence TEMPCHECK, from line 6 through 11
	( 6) IF (SIGTEMP=1)	#Partial contents of sequence TEMPCHECK
	( 7) SSTOP	
	( 8) MEND	
	( 9) CURRENT 0	
	(10) SAS Cooling	
	(11) ENDIF	
	>	

**LOCK: Lock Sequence****Sequence Management**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	LOCK target	
<b>Range</b>	target can be the name or number of any existing sequence	
<b>Description</b>	<p>LOCK prevents changes to a sequence.</p> <p>A locked sequence cannot be deleted, renamed, or overwritten (by COPY or EDIT).</p> <p>A locked sequence can still be loaded into the editor (with the EDIT command), but any changes must be saved to a new location.</p> <p>A locked sequence can be unlocked with the UNLOCK command.</p> <p>The sequence directory listing (DIR command) shows the lock status for all sequences.</p>	
<b>See Also</b>	DEL, DIR, EDIT, UNLOCK	
<b>Note</b>	A locked sequence will be cleared by CLEARSEQ or CLEARALL: the lock status offers no protection for these operations.	
<b>Example</b>	<p>Command</p> <p>&gt;LOCK PROG1</p> <p>&gt;DEL PROG1</p> <p>Error: Sequence is locked.</p> <p>&gt;DIR</p> <p>## Name      TextSize Locked</p> <p>== =====</p> <p>0 PROG1      37 Locked</p> <p>Total: 1</p> <p>Executable memory: 32 bytes used of 1600 bytes total, 2 percent.</p> <p>Storage memory: 77 bytes used of 4223 bytes total, 2 percent.</p> <p>&gt;</p>	<p>Description</p> <p>#Lock the sequence named PROG1 from deletion</p> <p>#Attempt to delete the PROG1 sequence</p> <p>#Device's response, unable to delete PROG1</p> <p>#Query the directory sequence</p>

**LOOP: Begin Counted LOOP Block****Sequence Commands**

<b>Execution Mode</b>	Sequence	
<b>Syntax</b>	LOOP [n]	
<b>Range</b>	n = 1 to 500,000,000 (integer value), loop count	
<b>Description</b>	<p>LOOP begins a "loop block" structure, which must be terminated later in the sequence by a corresponding ENDL (end loop) command.</p> <p>The statements between the LOOP and ENDL commands and will be executed 'n' times unless terminated (by a Break Loop (BREAKL) command, a Return (RET), an alarm condition, etc).</p> <p>Loop count 'n' is optional. If 'n' is not given, the block may execute forever. 'n' may be a positive constant, or any variable which a sequence can read. If the variable has a fractional component, it is ignored. The variable must have a positive value.</p> <p>Block structures (LOOP–ENDL, IF–ENDIF, WHILE–WEND) can be nested up to 6 levels deep.</p>	
<b>See Also</b>	BREAKL, ENDL, WHILE, WEND	
<b>Note</b>	Be careful when using MCP/MCN command. Refer to “10-3. Continuous Motions”.	
<b>Example</b>	Command	Description
	>LIST 27	#List sequence 27
	( 1) DIS=1000	#Distance equals 1000 steps
	( 2) <b>LOOP 5</b>	<b>#Loop the following 5 times</b>
	( 3) MI	#Do an Index Move
	( 4) MEND	#Wait for the move to end before executing the next command
	( 5) WAIT 1.0	#Wait 1 second
	( 6) ENDL	#End the loop
	>	

**LSEN: Hardware over travel detection****System Control**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	LSEN n	
<b>Range</b>	n = 0: Disable 1: Enable	
<b>Initial Value</b>	1	
<b>SAVEPRM</b>	The new value takes effect immediately and the parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	<p>LSEN enables or disables hardware position limit action.</p> <p>If the +LS and -LS inputs are to be used in an operation other than homing operation, set this parameter to “enable”.</p> <p>Moving outside hardware position limit range will cause the motor to stop, may cause an alarm (alarm code: 67h) and may disable motor current, depending on the value of ALMACT. Stop action (soft stop or hard stop) is defined by OTACT.</p> <p>For continuous motions (MCN, MCP), any out of range condition is detected only as it happens.</p> <p>This command may be used on test purpose when limit signal inputs want to be ignored.</p>	
<b>See Also</b>	SLEN, LIMN, LIMP	
<b>Example</b>	Command	Description
	> LSEN 0	#Set the hardware over travel detection disable
	LSEN=0	#Display the setting of LSEN
	>	

**MA: Move to Absolute Position****Motion Commands**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	MA n	
<b>Range</b>	n = -8,388,607 to +8,388,607	
<b>Description</b>	In immediate mode, 'n' can be a constant or any POS [x] position array variable.	
	In a sequence, 'n' can be a constant or any variable which can be read within a sequence.	
	MA starts a point-to-point motion to position “n”.	
	Motion velocity is determined by running velocity (VR). Start velocity (VS), acceleration time (TA), and deceleration time (TD) are effective. Speed may be changed while the motion is in progress, using the Change Velocity command (CV).	
	If the motion finishes successfully, the position set point (PC) should equal 'n'.	
<b>See Also</b>	Some combinations of effective distance, speeds and acceleration and deceleration times are not feasible.	
	For instance: if VR is very high, and TA and TD are very long, but the effective distance is very short, the system could cover too much distance accelerating to velocity VR over time TA. The system monitors for these conditions, and starts decelerating early if necessary. (Under these conditions, peak speed will be less than VR, and acceleration and deceleration times will be less than TA and TD.) The system is careful to preserve the actual motion distance, and the effective acceleration and deceleration rates.	
	MA is not accepted while the motor is moving, when current is OFF, or when the system has an active alarm condition. An attempt to execute MA while the motor is moving causes an error message in immediate mode, and causes an alarm and sequence termination (alarm code: A0h) if executed from a sequence.	
	MCN, MCP, MI, PC, TEACH, MEND, CV	
	MA starts an index motion, but does not wait for motion to end. Other commands can be issued in immediate mode or executed by a sequence while the motion is running, although most motion commands cannot be executed until the motion is complete.	
<b>Note</b>	To check that motion is finished, monitor SIGMOVE or SIGREADY. In a sequence, the MEND command provides a convenient way to suspend sequence processing until motion is finished.	
<b>Example</b>	Command	Description
	<pre> &gt;LIST MOVEABS  ( 1) PC=0                      #Set PC=0 ( 2) TA=0.1; TD=0.1           #Set ramp times ( 3) VS=100; VR=1000          #Set velocities ( 4) LOOP ( 5) SAS Position 1            #Message-1 ( 6) MA 250                    #Move to 250 ( 7) MEND; WAIT 1 ( 8) SAS Position 2            #Message-2 ( 9) MA 750                    #Move to 750 (10) MEND; WAIT 1 (11) SAS Position 3            #Message-3 (12) MA 500                    #Move to 500 (13) MEND; WAIT 1 (14) SAS Position 4            #Message-4 (15) MA 750                    #Move to 750 (16) MEND; WAIT 1 (17) SAS Position 5            #Message-5 *Continued on next page... </pre>	

( 18) MA 1000	#Move to 1000
( 19) MEND	
( 20) SAS End Session. Go to next.	#Message-6
( 21) WAIT 2	
( 22) ENDL	
>RUN MOVEABS	#Message-1
>Position 1	
>Position 2	
>Position 3	
>Position 4	
>Position 5	
>End Session. Go to next.	#Message-5
>Position 1	#Message-1
>Position 2	
>Position 3	
>Position 4	
>Position 5	
>End Session. Go to next.	
>	



**MCN, MCP: Move Continuously****Motion Commands**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	MCN MCP	
<b>Description</b>	<p>MCN and MCP start continuous motions, with no defined final position. MCN starts moving in the negative direction, and MCP starts moving in the positive direction.</p> <p>Motion velocity is determined by running velocity (VR). Start velocity (VS), acceleration time (TA) and deceleration time (TD) are effective.</p> <p>Motion continues until the system is commanded to stop or an alarm condition occurs.</p> <p>Velocity can be changed while a continuous motion is in progress, by changing the value of VR and re-issuing the MCN or MCP command. Ramp time will be TA if speed is increasing (away from zero) and TD if speed is decreasing (toward zero) for single speed motions. For motions where the speed is changed on the fly, the ramp times will be same as the accelerations and decelerations used in linked motions. A description of these ramp times can be found on page 41. The direction cannot be changed: MCN cannot be issued while an MCP motion is active, or vice versa. These conditions cause an error message if attempted at the command prompt, and an alarm (alarm code: A0h) if attempted in a sequence.</p> <p>MCN and MCP cannot be used while other motions are in progress (e.g. MI, MA), or while current is OFF, or while the system has an active alarm condition. These conditions also cause an error message if attempted at the command prompt, and an alarm (alarm code: A0h) if attempted in a sequence.</p>	
<b>See Also</b>	<ESC>, ABORT, DIRINV, PSTOP, INPAUSE, LIMN, LIMP, PAUSE, TA, TD, VR, VS	
<b>Note</b>	<p>MCN and MCP start continuous motions, but do not wait for motion to end. Other commands can be issued in immediate mode or executed by a sequence while the motion is running, although most motion commands cannot be executed until the motion is complete.</p> <p>To check that motion is finished, monitor SIGMOVE or SIGREADY. In a sequence, the MEND command provides a convenient way to suspend sequence processing until motion is finished. Be careful when using LOOP, IF, and WHILE command. Refer to “10-3. Continuous Motions”.</p>	
<b>Example</b>	<p>Command</p> <pre>&gt;LIST VCHANGE ( 1) TA 0.5; TD 0.5; VR 1000 ( 2) MCP ( 3) LOOP ( 4) IF (IN1=1) ( 5) VR=VR+10; MCP ( 6) SAS Increase speed by 10 pps ( 7) WAIT TA ( 8) WHILE (IN1=1); WEND ( 9) ENDIF (10) IF (IN2=1) (11) IF (VR!=1) (12) VR=VR-10; MCP (13) SAS Decrease speed by 10 pps (14) WAIT TD (15) WHILE (IN2=1); WEND (16) ELSE (17) SSTOP (18) SAS Reached endpoint, End Process (19) RET (20) ENDIF *Continued on next page...</pre>	<p>Description</p> <p>#Move continuously (positive)</p> <p>#Increase speed</p> <p>#Send message 1</p> <p>#Decrease speed</p> <p>#Send message 2</p> <p>#Soft stop</p> <p>#Send message 3</p>

```
( 21) ENDIF
( 22) ENDL
>RUN VCHANGE
>Increase speed by 10 pps      #Message 1
>Increase speed by 10 pps      #Message 1
>Increase speed by 10 pps      #Message 1
>Decrease speed by 10 pps      #Message 2
>Decrease speed by 10 pps      #Message 2
>Decrease speed by 10 pps      #Message 2
>Reached endpoint, End Process #Message 3: stopped.
>
```

**MEND: Wait for Motion End****Sequence Commands**

<b>Execution Mode</b>	Sequence	
<b>Syntax</b>	MEND	
<b>Description</b>	<p>MEND suspends sequence processing until motion is complete.</p> <p>Most motion commands start motions, but do not wait for motion to complete. Other operations can be performed while the motor is moving. MEND provides a simple way of synchronizing sequence execution with the end of a motion. When the motion completes (or if no motion is in progress), sequence execution proceeds to the statement following MEND.</p> <p>MEND is equivalent to WHILE (SIGMOVE=1); WEND. Most motion commands cannot be executed while another motion is in progress. To avoid errors, sequences should be designed to assure that each motion is complete before proceeding to another motion.</p>	
<b>Note</b>	The motion complete above means the command motion profile complete and it does not link to the actual motor shaft position even with an encoder feedback.	
<b>See Also</b>	SIGMOVE, SIGREADY, WHILE, WEND, IF, ENDIF	
<b>Example</b>	Command	Description
	>LIST MOVETIME	
	( 1) VS 100; VR 800; TA .05; TD .05	#Set motion parameters
	( 2) DIS 400	#Set distance DIS is 400
	( 3) T=0; Z=TIMER; MI	#T is zero, Z= start time. Move Incremental
	( 4) WHILE (VC<VR); WEND; T=TIMER-Z	#Wait for velocity to reach peak. Calc time.
	( 5) SACS ACTUAL ACCELERATION TIME:	#Message (last two chars are ^ and space)
	( 6) T	#Transmit acceleration time in msec
	( 7) MEND; T=TIMER-Z	#Wait for motion end, capture elapsed time
	( 8) SACS TOTAL MOVE TIME:	#Message (last two chars are ^ and space)
	( 9) T	#Transmit motion time in msec.
	>RUN MOVETIME	#Run MOVETIME
	>ACTUAL ACCELERATION TIME: 52	#First message
	>TOTAL MOVE TIME: 541	#Second message
	>	

**MGHN, MGHP: Seek Mechanical Home Position****Motion Commands**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	MGHN MGHP	
<b>Description</b>	<p>MGHN and MGHP start motion patterns, attempting to find a mechanical home position which links position zero (PC=0) to an application reference signal. MGHN starts moving in the negative direction, and MGHP starts moving in the positive direction.</p> <p>The process may involve moving in both directions before concluding. MGHN and MGHP differ in starting direction, and in direction upon final approach to the designated home signal (final approach is in the same direction as starting direction).</p> <p>The actual motion pattern and signal requirements are determined by HOMESEL. Depending on HOMESEL. See "Mechanical Home Seeking" in this specification for more information.</p> <p>The velocities and acceleration and deceleration times used for the home seeking process are determined by start velocity HOMEVS and run velocity HOMEVR, and acceleration and deceleration times HOMETR, at the time the process starts.</p> <p>If the home process completes successfully, the position counter (PC) is set to zero (0) and system output signal SIGHOMEPC is set to one (1). If configured, the HOMEPC output becomes active.</p> <p>Software position limits LIMN and LIMP are disabled while the homing process is active. If the system has been configured to use software position limits (SLEN=1) and the limits have been configured (LIMN and LIMP not both 0), the limits are enabled after successful completion of a homing process.</p> <p>MGHN and MGHP cannot be used while other motions are in progress (e.g. MI, MA), or while current is OFF, or while the system has an active alarm condition. These conditions also cause an error message if attempted at the command prompt, and an alarm (alarm code: A0h) if attempted in a sequence.</p>	
<b>See Also</b>	DIRINV, HOMESEL, HOMESLV, PC, OFFSET, OUTHOMEPC, OUTSG, SIGHOMEPC	
<b>Note</b>	<p>MGHN and MGHP start the home seeking process, but do not wait for the process to end. Other commands can be issued in immediate mode or executed by a sequence while the process is running, although motion commands cannot be executed until the process is complete.</p> <p>To check that motion is finished, monitor SIGMOVE or SIGREADY. In a sequence, the MEND command provides a convenient way to suspend sequence processing until motion is finished.</p>	
<b>Example</b>	Command	Description
	>LIST HOMING	#List sequence HOMING
	( 1) HOMEVS 10	#Start Homing Velocity: 10 pps
	( 2) HOMEVR 100	#Running Homing Velocity: 100 pps
	( 3) HOMETR 0.5	#Acceleration and Deceleration time: 0.5 sec
	( 4) MGHN	#Start seeking home in the negative direction
	( 5) MEND	#Wait for motion to complete
	>	

**MI: Move Incremental Distance****Motion Commands**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	MI	
<b>Description</b>	<p>MI starts a point-to-point incremental motion.</p> <p>The distance moved is determined by DIS, in user units. The direction of motion is determined by the arithmetic sign of DIS. Motion velocity is determined by running velocity (VR). Start velocity (VS), acceleration time (TA), and deceleration time (TD) are effective.</p> <p>Some combinations of distance, speeds and acceleration and deceleration times are not feasible. For instance: if VR is very high, and TA and TD are very long, but the distance is very short, the system could cover too much distance accelerating to velocity VR over time TA. The system monitors for these conditions, and starts decelerating early if necessary. (Under these conditions, peak speed will be less than VR, and acceleration and deceleration times will be less than TA and TD.) The system is careful to preserve the actual motion distance, and the effective acceleration and deceleration rates.</p> <p>MI is not accepted while the motor is moving, current is OFF, or while the system has an active alarm condition. An attempt to execute MI while the motor is moving causes an error message in immediate mode, and causes an alarm and sequence termination (alarm code: A0h) if executed from a sequence.</p>	
<b>See Also</b>	DIS, MA, TA, TD, VR, VS, CV	
<b>Note</b>	<p>MI starts an index motion, but does not wait for motion to end. Other commands can be issued in immediate mode or executed by a sequence while the motion is running, although most motion commands cannot be executed until the motion is complete.</p> <p>To check that motion is finished, monitor SIGMOVE or SIGREADY. In a sequence, the MEND command provides a convenient way to suspend sequence processing until motion is finished.</p>	
<b>Example</b>	Command	Description
	>LIST QCOUNTS	#List sequence QCOUNTS
	( 1) TA=0.1; TD=0.1	#Acceleration and deceleration times to 0.1
	( 2) VS=100; VR=1000;	#Program start and running speeds
	( 3) DIS=10000	#Set distance to 10000
	( 4) OUT3=0; OUT4=0	#Set general purpose outputs 3 and 4 inactive
	( 5) LOOP Q	#Loop, loop count given by variable Q
	( 6) MI	#Start moving incrementally, distance DIS
	( 7) MEND; PC	#Wait for motion to end, then send the value of PC
	( 8) OUT4=1	#Set Output 4 active
	( 9) WAIT 1	#Wait 1 second
	(10) OUT4=0	#Set Output 4 inactive
	(11) ENDL	#Endo of loop
	(12) MA 0	#Start moving to absolute position 0
	(13) MEND; PC	#Wait for motion to end, then send value of PC
	(14) OUT3=1	#Set Output 3 active
	>PC	#Display position command before starting
	PC=0	
	>Q	#Display value of variable Q
	Q=5	
	>run QCOUNTS	#Run Qcounts. Should do 5 incremental motions
	>1	#Output of PC after each incremental motion completes
	>2	
	>3	
	>4	
	>5	
	>0	#Output of PC after absolute move completes

**Mlx: Start Linked Index****Motion Commands**

<b>Execution Mode</b>	Immediate and Sequence														
<b>Syntax</b>	MIx														
<b>Range</b>	x = 0: Start with link segment 0 1: Start with link segment 1 2: Start with link segment 2 3: Start with link segment 3														
<b>Description</b>	<p>MIx starts a linked index motion beginning with link segment 'x' (0–3). The motion is point-to-point, but may be more complex than motions started with MA (Move Absolute) or MI (Move Incremental). Linked index motions can use up to four (4) running speeds between the start and stop position. The motion profile for each segment is defined by start velocity VS, acceleration and deceleration times TA and TD, and linked index parameters:</p> <ul style="list-style-type: none"> <li>- INCABSx determines whether segment 'x' is an absolute motion segment (INCABSx=0, move to a destination) or an incremental motion segment (INCABSx=1, move by a distance).</li> <li>- DISx is the destination (INCABSx=0) or distance (INCABS=1) of segment 'x'</li> <li>- VRx is the running speed for the segment 'x'.</li> </ul> <p>The segments can be linked together using LINKx. LINKx determines whether segment 'x' should stop (LINKx=0), or continue without stopping to execute the next segment (LINKx=1). (<b>Note:</b> There is no LINK3.)</p> <p>Motion can start with any link segment. The motor accelerates from VS to VRx over time TA. If LINKx=0, the motor will decelerate to a stop over time TD, after moving by or to DISx. If LINKx=1, the motor will continue at velocity VRx until the proper distance is covered or destination is reached (depending on DISx and INCABSx). Then, it will begin to execute the next segment, changing speeds as required.</p> <p>When changing speeds, acceleration time TA is used if speed is increasing away from zero, and deceleration time TD is used if speed is decreasing towards zero.</p> <p>Some combinations of distance, speeds, and acceleration and deceleration times are not feasible. For instance: if VRx is very high, and TA and TD are very long, but the effective distance is very short, the system could cover too much distance changing speed to velocity VRx. The system monitors for these conditions, and adjusts the motion profile if necessary. (Under these conditions, peak speed may be less than VRx, and acceleration and deceleration times may be less than TA and TD.) The system is careful to preserve the total motion distance or destination, and attempts to preserve the effective acceleration and deceleration rates. A sharp deceleration can occur if the effective distance of the last linked segment is small, and the previous link segment had a high running velocity. The system will stop at the correct final position, but cannot maintain the effective deceleration rate.</p>														
<b>See Also</b>	DISx, INCABSx, LINKx, MIx, TA, TD, VRx, VS														
<b>Note</b>	<p>MIx requires that all segments have the same effective direction of travel. If the first segment moves in the positive direction, then all linked segments which follow must move in the positive direction. If a MIx command is attempted which would result in both positive and negative motion, the Mix command is rejected. (An error message is generated in immediate mode. In a sequence, alarm 0x70h is set, and sequence processing terminates.)</p> <p>When using absolute links (INCABSx=0), motion direction depends on the motor position before the linked motion starts: careful planning is required to avoid an error or alarm.</p>														
<b>Example</b>	<table> <tr> <th>Command</th><th>Description</th></tr> <tr> <td>&gt;VR1 500</td><td>#Set the velocity for linked move #1 to 500 pps</td></tr> <tr> <td>VR1=500</td><td>#Device response</td></tr> <tr> <td>&gt;DIS1 2000</td><td>#Set the distance for linked move #1 to 2000</td></tr> <tr> <td>DIS1=2000</td><td>#Device response</td></tr> <tr> <td>&gt;INCABS1 1</td><td>#Set the move type for linked motion #1 to incremental</td></tr> <tr> <td>INCABS1=1 [INC]</td><td>#Device response</td></tr> </table>	Command	Description	>VR1 500	#Set the velocity for linked move #1 to 500 pps	VR1=500	#Device response	>DIS1 2000	#Set the distance for linked move #1 to 2000	DIS1=2000	#Device response	>INCABS1 1	#Set the move type for linked motion #1 to incremental	INCABS1=1 [INC]	#Device response
Command	Description														
>VR1 500	#Set the velocity for linked move #1 to 500 pps														
VR1=500	#Device response														
>DIS1 2000	#Set the distance for linked move #1 to 2000														
DIS1=2000	#Device response														
>INCABS1 1	#Set the move type for linked motion #1 to incremental														
INCABS1=1 [INC]	#Device response														

>LINK1 1	#Enable the linked operation for motion #1
LINK1=1	#Device response
>VR2 1000	#Linked move #2 velocity equals 1000 pps
VR2=1000	#Device response
>INCABS2 1	#Set the move type for linked motion #2 to incremental
INCABS2=1 [INC]	#Device response
>DIS2 4000	#Linked move #2: destination is position 4000
DIS2=4000	#Device response
>LINK2 0	#"Unlink" link2 from link3
LINK2=0	#Device response
>M11	#Start the linked operation motion
>	

**MOVELV: Move Output Level****I/O**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	MOVELV n	
<b>Range</b>	n = 0: Normally Open 1: Normally Closed	
<b>Initial Value</b>	0	
<b>RESET</b>	RESET or recycle the power is required to activate the change. The parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	Sets the active level for the MOVE output.	
<b>See Also</b>	SIGHOME P	
<b>Example</b>	Command	Description
	>MOVELV=1	#Set the MOVE output logic to Normally Closed
	MOVELV=0(1)	
	>RESET	#RESET the device to initialize the modified MOVE setting
	Resetting system.	
	-----	
	CRD5xx-KP	
	CRK Series Built-in Controller	
	Software Version: A378 V.x.xx	
	Copyright 2009-2011	
	ORIENTAL MOTOR U.S.A. CORP.	
	-----	
	>MOVELV	#New value is active
	MOVELV=1(1)	
	>	



**MRES: Motor Resolution Setting****System Control**

<b>Execution Mode</b>	Immediate
<b>Syntax</b>	MRES n
<b>Range</b>	n = 0 to 15
<b>Initial Value</b>	0
<b>RESET</b>	RESET or recycle the power is required to activate the change. The parameter is saved automatically.
<b>Access</b>	READ and WRITE
<b>Description</b>	Sets the motor step angle.

**For Standard Type Step Motors with 500 steps/rev (0.72°) (CRK5□ types)**

Setting	Step angle (Steps/rev)	# of divisions	Setting	Step angle (Steps/rev)	# of divisions
0	0.72° (500)	1	8	0.0288° (12,500)	25
1	0.36° (1000)	2	9	0.018° (20,000)	40
2	0.288° (1250)	2.5	10	0.0144° (25,000)	50
3	0.18° (2000)	4	11	0.009° (40,000)	80
4	0.144° (2500)	5	12	0.0072° (50,000)	100
5	0.09° (4000)	8	13	0.00576° (62,500)	125
6	0.072° (5000)	10	14	0.0036° (100,000)	200
7	0.036° (10,000)	20	15	0.00288° (125,000)	250

**For High Resolution Type Step Motors with 1000 steps/rev (0.36°/step) (CRK5□M types)**

Setting	Step angle (Steps/rev)	# of divisions	Setting	Step angle (Steps/rev)	# of divisions
0	0.36° (1000)	1	8	0.0144° (25,000)	25
1	0.18° (2000)	2	9	0.009° (40,000)	40
2	0.144° (2500)	2.5	10	0.0072° (50,000)	50
3	0.09° (4000)	4	11	0.0045° (80,000)	80
4	0.072° (5000)	5	12	0.0036° (100,000)	100
5	0.045° (8000)	8	13	0.00288° (125,000)	125
6	0.036° (10,000)	10	14	0.0018° (200,000)	200
7	0.018° (20,000)	20	15	0.00144° (250,000)	250

**See Also** EGA, EGB**Note**

- Step angles are theoretical values.
- With the geared type, the value of “step angle/gear ratio” will be the actual step angle.

**Example**

Command

&gt;MRES 1

MRES=0(1)

&gt;

&gt;RESET

Resetting system.

Description

#Set the motor resolution to 1000 P/R in case of the stepping motor which has a basic step angle of 0.72 degree.

#RESET the device to initialize the modified MRES setting

-----  
CRD5xx-KP

CRK Series Built-in Controller

Software Version: A378 V.x.xx

Copyright 2009-2011

ORIENTAL MOTOR U.S.A. CORP.  
-----

&gt;MRES

#New value is active

MRES=1(1)

**OTACT: Over travel Action****System Control**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	OTACT n	
<b>Range</b>	n = 0: Hard Stop (stop as quickly as possible) 1: Soft Stop (controlled deceleration over time)	
<b>Initial Value</b>	0	
<b>RESET</b>	RESET or recycle the power is required to activate the change. The parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	<p>OTACT establishes the stop action when the system detects an over travel input signal (-LS or +LS). If OTACT=0, the system will stop the motor as quickly as possible (hard stop). Stop action is exactly the same as HSTOP.</p> <p>If OTACT=1, the system will stop the motor by a controlled deceleration over time (soft stop). Stop action is exactly the same as SSTOP.</p> <p>Action after stop (alarm or no alarm, current ON or OFF) is controlled by ALMACT.</p>	
<b>Caution</b>	Use caution when using the Soft Stop option. The additional distance traveled during a Soft Stop depends on system speed and other parameters. Be sure that the load will not strike any physical obstacles for a significant range beyond the over travel detectors.	
<b>See Also</b>	HSTOP, SSTOP, ALMACT, SIGLSP, SIGLSN, OTLV	
<b>Example</b>	Command	Description
	>OTACT 0	#Set the over travel action to Hard Stop.
	OTACT=0(0)	
	>ALMACT 2	#Set Alarm Action to 2 (stop, alarm, current OFF)
	ALMACT=2(2)	
	>LIMN -50	#Set negative position limit (typically inside hardware limit)
	LIMN=-50	
	>LIMP 50	#Set positive position limit (typically inside hardware limit)
	LIMP=50	
	>SLEN 1	#Enable software limit checking (after home operation)
	SLEN=1	
	>MGHP	#Seek home, start in positive direction (if successful, LIMN and LIMP position limits become active)
	>	

**OTLV: Over travel Input Level****I/O**

<b>Execution Mode</b>	Immediate
<b>Syntax</b>	OTLV n
<b>Range</b>	n = 0: Normally Open 1: Normally Closed
<b>Initial Value</b>	0
<b>RESET</b>	RESET or recycle the power is required to activate the change. The parameter is saved automatically.
<b>Access</b>	READ and WRITE
<b>Description</b>	OTLV is the active level of the over travel limit inputs, -LS and +LS.
<b>See Also</b>	SIGLSN, SIGLSP
<b>Example</b>	Command

Command	Description
>OTLV=1	#Set the LS input logic to Normally Closed
OTLV=0(1)	
>RESET	#RESET the device to initialize the modified LS setting
Resetting system.	
-----	
CRD5xx-KP	
CRK Series Built-in Controller	
Software Version: A378 V.x.xx	
Copyright 2009-2011	
ORIENTAL MOTOR U.S.A. CORP.	
-----	
>OTLV	
OTLV=1(1)	#New value is active
>	

**OUT: General Output Status****I/O**

<b>Execution Mode</b>	Immediate and Sequence										
<b>Syntax</b>	OUT										
<b>Range</b>	n = 0 to 15 (integer value)										
<b>Access</b>	READ and WRITE										
<b>Description</b>	<p>OUT displays or sets the value of all the general purpose outputs, as one integer number.</p> <p>The general purpose outputs contribute to the value of OUT as follows:</p> <table> <tr> <td>OUTx</td><td>Contribution to OUT if active</td></tr> <tr> <td>OUT4</td><td>8</td></tr> <tr> <td>OUT3</td><td>4</td></tr> <tr> <td>OUT2</td><td>2</td></tr> <tr> <td>OUT1</td><td>1</td></tr> </table> <p>For example, if OUT=10 then OUT2 #2 (2) is ON, and Output#4 is ON (8). (2+8=10)</p> <p>To check or change the status of a single general output, use the OUTx command.</p> <p>All general purpose outputs are in the inactive (OFF) state immediately following system startup.</p>	OUTx	Contribution to OUT if active	OUT4	8	OUT3	4	OUT2	2	OUT1	1
OUTx	Contribution to OUT if active										
OUT4	8										
OUT3	4										
OUT2	2										
OUT1	1										
<b>Caution</b>	All outputs are OFF when device power is OFF.										
<b>See Also</b>	INITIO, IO, IN, OUTTEST, OUTSG, OUTx, REPORT										
<b>Important Interactions</b>	If an output is assigned to a system output signal (OUTHOMEP, OUTREADY, etc) the OUT command will not effect or reflect the electrical I/O port state. The port is always controlled by its assigned signal. Use the OUTSG command to read the status of the assigned system output signals.										
<b>Example</b>	<p>Command                      Description</p> <p>&gt;IO                              #Check IO Status</p> <p>                                    #Response: <b>Note</b> that STO has been assigned to Output 1. Outputs 2–4 are general purpose.</p> <p>Input : IN1 IN2 IN3 IN4 PAUSE PAUSECL</p> <p>      : START ABORT ALMCLR CROFF HOME +LS -LS PSTOP SENSOR HOMES SLIT</p> <p>Output : ALM MOVE STO OUT2 OUT3 OUT4</p> <p>--Inputs---      --- Dedicated IO signals ----- Outputs</p> <p>1 2 3 4 5 6 -(SEQ#)- S A A C H + - P S H S    — A M 1 2 3 4</p> <p>0 0 0 0 0 0 -( 0)-    0 0 0 0 0 0 0 0 0 0    — 0 0 0 0 0 0</p> <p>&gt;                              #All outputs reported OFF.</p> <p>&gt;OUT 15                        #Set OUT to 15 (all outputs on)</p> <p>OUT=15</p> <p>&gt;IO                              #Check IO Status again.</p> <p>Input : IN1 IN2 IN3 IN4 PAUSE PAUSECL</p> <p>      : START ABORT ALMCLR CROFF HOME +LS -LS PSTOP SENSOR HOMES SLIT</p> <p>Output : ALM MOVE STO OUT2 OUT3 OUT4</p> <p>--Inputs---              --- Dedicated IO signals ----- Outputs</p> <p>1 2 3 4 5 6 -(SEQ#)- S A A C H + - P S H S    — A M 1 2 3 4</p> <p>0 0 0 0 0 0 -( 0)-    0 0 0 0 0 0 0 0 0 0    — 0 0 0 1 1 1</p> <p>&gt;                              #All outputs are on... expect Output 1. Output 1 active state cannot be effected by</p> <p>OUT</p>										

**OUTAREA: AREA Signal Output Assignment****I/O**

<b>Execution Mode</b>	Immediate
<b>Syntax</b>	OUTAREA n
<b>Range</b>	n = 0 to 4
<b>Initial Value</b>	0 (Unassigned)
<b>RESET</b>	RESET or recycle the power is required to activate the change. The parameter is saved automatically.
<b>Access</b>	READ and WRITE
<b>Description</b>	<p>OUTAREA is the output port assigned to system output signal AREA. If OUTAREA is zero (0), the AREA signal is not assigned to an output.</p> <p>This signal will be output when the motor output shaft is inside the area set by the “area 1” and “area 2” parameters. This signal is also output while the motor is stopped.</p> <p>The active level of the AREA output is controlled by AREALV.</p>

**See Also** SIGAREA, AREALV, OUTSG, AREAn

Example	Command	Description
	>OUTAREA 2	#Assign the AREA output assignment to Output #2
	OUTAREA=0(2)	
	>RESET	#Establish the saved parameter values
	Resetting system.	
	-----	
	CRD5xx-KP	
	CRK Series Built-in Controller	
	Software Version: A378 V.x.xx	
	Copyright 2009-2011	
	ORIENTAL MOTOR U.S.A. CORP.	
	-----	
	>OUTAREA	
	OUTAREA=2(2)	#Confirm the new assignment
	>	

**OUTHOMEP: Home Position Signal Output Assignment****I/O**

<b>Execution Mode</b>	Immediate										
<b>Syntax</b>	OUTHOMEP n										
<b>Range</b>	n = 0 to 4										
<b>Initial Value</b>	0 (Unassigned)										
<b>RESET</b>	RESET or recycle the power is required to activate the change. The parameter is saved automatically.										
<b>Access</b>	READ and WRITE										
<b>Description</b>	<p>OUTHOMEP is the output port assigned to system output signal HOMEP (Home Positioning Status). If OUTHOMEP is zero (0), the HOMEP signal is not assigned to an output.</p> <p>The HOMEP output is set to its active state while on a valid HOME position, and inactive otherwise. The HOMEP output is always OFF until the system has successfully executed a homing operation (MGHN, MGHP). After a successful homing operation, the HOMEP is ON when position command PC=0, and SIGHOME=0 otherwise.</p> <p>The HOMEP is continuously updated by the system.</p> <p>The active level of the HOMEP output is controlled by HOMEPLV.</p>										
<b>See Also</b>	HOMEPLV, OUTSG, SIGHOMEP										
<b>Example</b>	<table> <tr> <th>Command</th><th>Description</th></tr> <tr> <td>&gt;OUTHOMEP 2</td><td>#Assign the HOMEP output assignment to Output #2</td></tr> <tr> <td>OUTHOMEP=0(2)</td><td></td></tr> <tr> <td>&gt;RESET</td><td>#Establish the saved parameter values</td></tr> <tr> <td>Resetting system.</td><td></td></tr> </table>	Command	Description	>OUTHOMEP 2	#Assign the HOMEP output assignment to Output #2	OUTHOMEP=0(2)		>RESET	#Establish the saved parameter values	Resetting system.	
Command	Description										
>OUTHOMEP 2	#Assign the HOMEP output assignment to Output #2										
OUTHOMEP=0(2)											
>RESET	#Establish the saved parameter values										
Resetting system.											

-----

CRD5xx-KP  
 CRK Series Built-in Controller  
 Software Version: A378 V.x.xx  
 Copyright 2009-2011  
 ORIENTAL MOTOR U.S.A. CORP.

-----

>OUTHOMEP  
 OUTHOMEP=2(2) #Confirm the new assignment  
 >

**OUTPSTS: Pause Status Signal Output Assignment****I/O**

<b>Execution Mode</b>	Immediate
<b>Syntax</b>	OUTPSTS n
<b>Range</b>	n = 0 to 4
<b>Initial Value</b>	0 (Unassigned)
<b>RESET</b>	RESET or recycle the power is required to activate the change. The parameter is saved automatically.
<b>Access</b>	READ and WRITE
<b>Description</b>	<p>OUTPSTS is the output port assigned to system output signal PSTS (Pause Status). If OUTPSTS is zero (0), the PSTS signal is not assigned to an output.</p> <p>The PSTS output is set to its active state when a motion has been paused, by PAUSE command or PAUSE input.</p> <p>The PSTS output is set to its inactive state when motion has not been paused, when a paused motion has been resumed with by a CONT command or START input, and when a paused motion has been cleared by a PAUSECL (Pause Clear) input or PAUSECLR command.</p> <p>The active level of the PSTS output is controlled by PSTSLV.</p>

**See Also**

SIGPSTS, PSTSLV, OUTSG, PAUSE, PAUSECLR, CONT

**Example**

Command	Description
>OUTPSTS 2	#Assign the PSTS output assignment to Output #2
OUTPSTS=0(2)	
>RESET	#Establish the saved parameter values
Resetting system.	
-----	
CRD5xx-KP	
CRK Series Built-in Controller	
Software Version: A378 V.x.xx	
Copyright 2009-2011	
ORIENTAL MOTOR U.S.A. CORP.	
-----	
>OUTPSTS	
OUTPSTS=2(2)	#Confirm the new assignment
>	

**OUTREADY: READY Status Signal Output Assignment****I/O**

<b>Execution Mode</b>	Immediate
<b>Syntax</b>	OUTREADY n
<b>Range</b>	n = 0 to 4
<b>Initial Value</b>	0 (Unassigned)
<b>RESET</b>	RESET or recycle the power is required to activate the change. The parameter is saved automatically.
<b>Access</b>	READ and WRITE
<b>Description</b>	OUTREADY is the output port assigned to system output signal READY. If OUTREADY is zero (0), the AREA signal is not assigned to an output. This signal will be output when the driver becomes ready. Start operation after the READY output has turned ON. The READY output remains OFF in the following conditions:

- The motor is operating
- An alarm is present
- Any one of the HOME input and START input is ON
- The CROFF input is ON
- The ABORT input is ON (Not including ABORT status of START input when set to act as a toggle switch (STARTACT=1)).
- The motor is not excited
- Immediately after the power was turned on

The active level of the READY output is controlled by READYLV.

**See Also**

SIGREADY, READYLV, OUTSG

**Example**

Command

Description

>OUTREADY 2

#Assign the READY output assignment to Output #2

OUTREADY=0(2)

>RESET

#Establish the saved parameter values

Resetting system.

-----

CRD5xx-KP

CRK Series Built-in Controller

Software Version: A378 V.x.xx

Copyright 2009-2011

ORIENTAL MOTOR U.S.A. CORP.

-----

>OUTREADY

#Confirm the new assignment

OUTREADY=2(2)

>



**OUTRUN: RUN Signal Output Assignment****I/O**

<b>Execution Mode</b>	Immediate
<b>Syntax</b>	OUTRUN n
<b>Range</b>	n = 0 to 4
<b>Initial Value</b>	0 (Unassigned)
<b>RESET</b>	RESET or recycle the power is required to activate the change. The parameter is saved automatically.
<b>Access</b>	READ and WRITE
<b>Description</b>	<p>OUTRUN is the output port assigned to system output signal RUN. If OUTRUN is zero (0), the RUN signal is not assigned to an output.</p> <p>The RUN output is set to its active state while the sequences are executing.</p> <p>The RUN output is set to its inactive state when sequences are not executing.</p> <p>The active level of the RUN output is controlled by RUNLV.</p>

**See Also** SIGRUN, RUNLV, OUTSG

Example	Command	Description
	>OUTRUN 2	#Assign the RUN output assignment to Output #2
	OUTRUN=0(2)	
	>RESET	#Establish the saved parameter values
	Resetting system.	
	-----	
	CRD5xx-KP	
	CRK Series Built-in Controller	
	Software Version: A378 V.x.xx	
	Copyright 2009-2011	
	ORIENTAL MOTOR U.S.A. CORP.	
	-----	
	>OUTRUN	
	OUTRUN=2(2)	#Confirm the new assignment
	>	

**OUTSC: SC Signal Output Assignment****I/O**

<b>Execution Mode</b>	Immediate
<b>Syntax</b>	OUTSC n
<b>Range</b>	n = 0 to 4
<b>Initial Value</b>	0 (Unassigned)
<b>RESET</b>	RESET or recycle the power is required to activate the change. The parameter is saved automatically.
<b>Access</b>	READ and WRITE
<b>Description</b>	OUTSC is the output port assigned to system output signal SC. If OUTSC is zero (0), the SC signal is not assigned to an output. This signal is effective when an encoder is connected. This signal is output when a step deviation error has occurred and was corrected automatically. The SC output will turn OFF when the next motion command is executed or if the motor current is turned OFF. If the SC output is to be used, set the Self Correcting (SCEN) parameter to “enable”. The active level of the SC output is controlled by SCLV.

**See Also**

SIGSC, SCLV, OUTSG, SCEN, SCTO, EGA, EGB

**Example**

Command	Description
>OUTSC 2	#Assign the SC output to Output #2
OUTSC=0(2)	
>RESET	#Establish the saved parameter values
Resetting system.	

-----

CRD5xx-KP  
CRK Series Built-in Controller  
Software Version: A378 V.x.xx  
Copyright 2009-2011-2011  
ORIENTAL MOTOR U.S.A. CORP.

-----

>OUTSC	
OUTSC=2(2)	#Confirm the new assignment
>	

**OUTSG: System Output Signal Status****I/O****Execution Mode** Immediate and Sequence**Syntax** OUTSG n**Range** n = 0 to 2047**Initial Value** 0**Access** READ**Description** OUTSG displays the current status of all the system output signals, as one integer number.

The system output signals contribute to the value of OUTSG as follows:

Bit Location	Signal	Contribution to OUTSG if active
Bit 0	MOVE	1
Bit 1	RUN	2
Bit 2	AREA	4
Bit 3	HOME P	8
Bit 4	ALM	16
Bit 5	PSTS	32
Bit 6	TEMP	64
Bit 7	READY	128
Bit 8	STO	256
Bit 9	WNG	512
Bit 10	SC	1024

OUTSG is the sum of the contribution of all active signals:

- If OUTSG=2, the RUN signal is active, and all other signals are inactive.

- If OUTSG=192, the TEMP (64) and READY (128) signals are active (64+128=192), and all other signals are inactive.

Be careful not to confuse OUTSG with OUT (Output Status). OUT reports the status of General Purpose Outputs (those outputs which are not assigned to a signal). OUTSG reports the status of system output signals.

System output signals are always maintained in their appropriate state, even in the signals are not assigned to outputs.

**See Also** SIGMOVE, SIGRUN, SIGAREA, SIGHOME P, SIGALM, SIGPSTS, SIGTEMP, SIGREADY, SIGSC, SIGSTO, SIGWNG, OUT**Example**

Command	Description
>OUTSG	#Query the status of the system output signals
OUTSG=1	#OUTSG equals 1, indicating motion is occurring
>	

**OUTSTO: STO Signal Output Assignment****I/O**

<b>Execution Mode</b>	Immediate
<b>Syntax</b>	OUTSTO n
<b>Range</b>	n = 0 to 4
<b>Initial Value</b>	0 (Unassigned)
<b>RESET</b>	RESET or recycle the power is required to activate the change. The parameter is saved automatically.
<b>Access</b>	READ and WRITE
<b>Description</b>	OUTSTO is the output port assigned to system output signal STO. If OUTSTO is zero (0), the STO signal is not assigned to an output.

This signal becomes effective when an encoder is connected, and a deviation error occurs.

This signal will be output when the deviation between the encoder counter value and driver command position reaches the value set in the “stepout detection band” STOB parameter. If the STO output is to be used, set the “stepout detection” STOEN parameter to “enable”.

The active level of the STO output is controlled by STOLV.

**See Also** EGA, EGB, SIGSTO, STOLV, OUTSG, STOB, STOEN, STOACT

**Example**

Command	Description
>OUTSTO 2	#Assign the STO output assignment to Output #2
OUTSTO=0(2)	
>RESET	#Establish the saved parameter values
Resetting system.	
-----	
CRD5xx-KP	
CRK Series Built-in Controller	
Software Version: A378 V.x.xx	
Copyright 2009-2011	
ORIENTAL MOTOR U.S.A. CORP.	
-----	
>OUTSTO	#Confirm the new assignment
OUTSTO=2(2)	
>	

**OUTTEMP: TEMP Signal Output Assignment****I/O**

<b>Execution Mode</b>	Immediate
<b>Syntax</b>	OUTTEMP n
<b>Range</b>	n = 0 to 4
<b>Initial Value</b>	0 (Unassigned)
<b>RESET</b>	RESET or recycle the power is required to activate the change. The parameter is saved automatically.
<b>Access</b>	READ and WRITE

**Description** OUTTEMP is the output port assigned to system output signal TEMP (Temperature Warning).  
 If OUTTEMP is zero (0), the TEMP signal is not assigned to an output.  
 The TEMP output is set to its active state if drive electronics temperature DTMP exceeds drive temperature warning limit DTMPWNG.  
 The TEMP output is set to its inactive state when DTMP is below its respective warning levels.  
 The active level of the TEMP output is controlled by TEMPLV.

**See Also** SIGTEMP, TEMPLV, OUTSG, DTMP, DTMPWNG

<b>Example</b>	Command	Description
	>OUTTEMP 2	#Assign the TEMP output assignment to Output #2
	OUTTEMP=0(2)	
	>RESET	#Establish the saved parameter values
	Resetting system.	
-----		
	CRD5xx-KP	
	CRK Series Built-in Controller	
	Software Version: A378 V.x.xx	
	Copyright 2009-2011	
	ORIENTAL MOTOR U.S.A. CORP.	
-----		
	>OUTTEMP	
	OUTTEMP=2(2)	#Confirm the new assignment
	>	

**OUTTEST: I/O Test Utility****I/O****Execution Mode** Immediate**Syntax** OUTTEST

**Description** OUTTEST starts a utility process to check I/O connections and levels. Inputs are continuously monitored and displayed, and outputs can be set or cleared, to confirm proper external connections. Inputs and outputs are displayed as active (1) or inactive (0). OUTTEST temporarily disables the actions of all assigned system input and output signals. The system will not react to inputs, and will not automatically control outputs. All output control is from the serial port. Signal assignments are restored when the OUTTEST process terminates, and all outputs are restored to the state they were in when the OUTTEST process was started. Outputs can be toggled, using the character displayed next to the signal name in the OUTTEST output. Toggling an output changes its state as displayed, and changes the electrical state of the associated output port. Toggle keystrokes or characters for each output are:

**See Also** IN, INSG, OUT, OUTSG, IO**Example**

Command	Description
>OUTTEST	#Start the OUTTEST process
>	#Assignments and toggle keys shown here.
*** Input Monitor -- Output Simulator ***	
Inputs : IN1 IN2 IN3 IN4 PAUSE PAUSECL	#Device response
: START ABORT ALMCLR CROFF HOME +LS -LS PSTOP SENSOR HOMES SLIT	
Outputs : ALM(A) MOVE(M) OUT1(1) OUT2(2) HOMEP(H) STO(S)	
- Use (x) keys to toggle Outputs.	
- Use <space> to set all outputs to zero.	
- Use <esc> to exit OUTTEST mode.	
- Press <enter> to update inputs status.	
--Inputs--- --- Dedicated IO signals ----- Outputs	
1 2 3 4 5 6 -(SEQ#)-	S A A C H + - P S H S — A M 1 2 3 4
0 0 0 0 0 0 -( 0 )-	0 0 0 0 0 0 0 0 0 0 — 0 0 0 0 0 0
>	#Active (1) or inactive (0) states shown here
	#Escape entered: OUTTEST ends

**OUTWNG: Warning Output Status Assignment****I/O**

<b>Execution Mode</b>	Immediate
<b>Syntax</b>	OUTWNG n
<b>Range</b>	n = 0 to 4
<b>Initial Value</b>	0 (Unassigned)
<b>RESET</b>	RESET or recycle the power is required to activate the change. The parameter is saved automatically.
<b>Access</b>	READ and WRITE
<b>Description</b>	<p>OUTWNG is the output port assigned to system output signal WNG. If OUTWNG is zero (0), the WNG signal is not assigned to an output.</p> <p>This signal is output when a warning generates. However, the operation will continue.</p> <p>The WNG output will turn OFF automatically once the cause of the warning is removed.</p> <p>The active level of the WNG output is controlled by WNGLV.</p>

**See Also** SIGWNG, WNGLV, OUTSG

Example	Command	Description
	>OUTWNG 2	#Assign the WNG output assignment to Output #2
	OUTWNG=0(2)	
	>RESET	#Establish the saved parameter values
	Resetting system.	
	-----	
	CRD5xx-KP	
	CRK Series Built-in Controller	
	Software Version: A378 V.x.xx	
	Copyright 2009-2011-2011	
	ORIENTAL MOTOR U.S.A. CORP.	
	-----	
	>OUTWNG	#Confirm the new assignment
	OUTWNG=2(2)	
	>	

**OUTx : Individual General Output Control****I/O**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	OUTx=n	
<b>Range</b>	x = 1 to 4	
	n = 0: Not Active 1: Active	
<b>Description</b>	<p>OUTx controls the state of General Purpose Output 'x'.</p> <p>If the output has been assigned to a system output signal, then it is no longer "General Purpose".</p> <p>OUTx for these outputs has no affect on the output pins. Use OUTSG to check the status of assigned system output signals.</p>	
<b>See Also</b>	INITIO, OUT, OUTSG, OUTTEST	
<b>Example</b>	Command	Description
	>LIST HOMEDIR	#Sequence to output motion direction while seeking home
	( 1) WHILE (SIGMOVE=1)	#While system is moving
	( 2) IF (VC>0)	#If moving in positive direction
	( 3) OUT1=1	#General Purpose Output 1 active
	( 4) ELSE	
	( 5) OUT1=0	#Else, General Purpose Output 1 inactive
	( 6) ENDIF	
	( 7) IF (VC<0)	#If moving in negative direction
	( 8) OUT2=1	#General Purpose Output 2 active
	( 9) ELSE	
	(10) OUT2=0	#Else, General Purpose Output 2 inactive
	(11) ENDIF	#End of IF block
	(12) WEND	#End of WHILE block
	(13) OUT1=0; OUT2=0	#No longer moving: set both General Purpose Outputs inactive.
	>	



**PAUSE: Pause Motion****Motion Commands**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	PAUSE	
<b>Description</b>	<p>PAUSE interrupts a motion, stopping the motor by controlled deceleration (soft stop). See SSTOP for details on the velocity profile during deceleration.</p> <p>The system remembers the motion that was in process, so that it may be resumed later. See the CONT (Continue Motion) command for details on continuing motions after a PAUSE command.</p> <p>After a PAUSE command, the system sets system output signal SIGPSTS to one (1). If SIGPSTS has been assigned to an output, that output is set to its active state.</p> <p>The system remains in a "paused" state, until motion is continued (see CONT), or the state is explicitly cleared (with a PAUSECL input or PAUSECLR command), or another motion command is executed. If no motion is in process when a PAUSE command is issued, the PAUSE command has no effect.</p> <p>Motions may also be paused by assigning system input signal PAUSE to an input. Operation after PAUSE becomes active is identical to issuing a PAUSE command.</p>	
<b>See Also</b>	SSTOP, CONT, INITIO, INPAUSE, INPAUSECL, OUTPSTS, OUTSG, PAUSECLLV, PAUSECLR, PAUSELV, PSTSLV, SIGPAUSE, SIGPAUSECL, SIGPSTS	
<b>Note</b>	<p>PAUSE and CONT may affect processing time of sequences. For instance: if a sequence executes a MEND (wait for motion end) command, the sequence will be suspended while the motion is paused, and will not proceed beyond the MEND until the next end of motion (via a CONT, PAUSECL input or PAUSECLR command, or new motion).</p> <p>Pause and Continue operations are not supported for Linked Motions (MIx). PAUSE during a Linked Motion causes a soft stop, and subsequent CONT commands are ignored.</p>	
<b>Example</b>	<p>Command</p> <p>&gt;LIST CHKJAM</p> <p>( 1) DIS=1000; VR=100</p> <p>( 2) LOOP</p> <p>( 3) MI</p> <p>( 4) WHILE (DTMP&lt;70)</p> <p>( 5) IF (SIGMOVE=0)</p> <p>( 6) BREAKW</p> <p>( 7) ENDIF</p> <p>( 8) WEND</p> <p>( 9) IF (SIGMOVE!=0)</p> <p>(10) PAUSE</p> <p>(11) WAIT TD</p> <p>(12) SAS System in trouble.</p> <p>(13) SACS Enter 1 to continue, other to stop:</p> <p>(14) A=KBQ; SACS ^M^J&gt;</p> <p>(15) IF (A=1)</p> <p>(16) CONT; MEND</p> <p>(17) ELSE</p> <p>(18) SAS Operation stopped.</p> <p>(19) RET</p> <p>(20) ENDIF</p> <p>(21) ENDIF</p> <p>(22) SAS Motion end, goto next.</p> <p>(23) WAIT 1</p> <p>(24) ENDL</p>	<p>Description</p> <p>#List sequence CHKJAM</p> <p>#Set motion parameter</p> <p>#Start infinite loop</p> <p>#Start move incremental</p> <p>#Check if over heated</p> <p>#Check for motion end</p> <p>#Exit while loop, if so</p> <p>#Check if moving</p> <p>#DTMP&gt;70: PAUSE motion</p> <p>#Wait for stop, send text, get response</p> <p>#CONTinue, if A=1</p> <p>#Otherwise, report stopped</p> <p>#Return from sequence</p> <p>#Send normal message</p> <p>#Dwell 1 second, loop back to top.</p>

>RUN CHKJAM	#Execute sequence CHKJAM
>Motion end, goto next.	#Normal message
>Motion end, goto next.	#Normal message
>System in trouble.	#Driver is getting hot
>Enter 1 to continue, other to stop:1	#Prompt message -> Entry "1"
>Motion end, goto next.	#Normal message
>Motion end, goto next.	#Normal message
>System in trouble.	#Driver is getting hot
>Enter 1 to continue, other to stop:2	#Prompt message -> Entry "2"
>Operation stopped.	#Finished message (Sequence finish)
>	

**PAUSECLR: Clear State of Paused Motion****Motion Commands**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	PAUSECLR	
<b>Description</b>	<p>PAUSECLR clears the state of any paused motion.</p> <p>Any remaining motion is “forgotten”, and the previously paused motion cannot be continued (the CONT command will be ignored). If motion was paused, the PSTS signal will be cleared by the PAUSECLR command.</p> <p>If motion was not paused when a PAUSECLR command is issued, the PAUSECLR command has no effect.</p> <p>Paused motions may also be cleared by assigning system input signal PAUSECL to an input. Operation after PAUSECL becomes active is identical to issuing a PAUSECLR command.</p>	
<b>See Also</b>	SSTOP, CONT, INITIO, INPAUSE, INPAUSECL, OUTPSTS, OUTSG, PAUSECLLV, PAUSE, PAUSELV, PSTSLV, SIGPAUSE, SIGPAUSECL, SIGPSTS	
<b>Example</b>	Command	Description
	>LIST RESUME	#List sequence RESUME
	( 1) IF (CURRENT=1)	#If motor current applied
	( 2) CONT	#Continue previous motion
	( 3) ELSE	#Otherwise (no motor current)
	( 4) PAUSECLR	#Clear pause status
	( 5) ENDIF	
	>	

**PAUSECLLV: Pause Clear Input Level****I/O**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	PAUSECLLV n	
<b>Range</b>	n = 0: Normally Open 1: Normally Closed	
<b>Initial Value</b>	0	
<b>RESET</b>	RESET or recycle the power is required to activate the change. The parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	PAUSECLLV is the active level of the Pause Clear (PAUSECL) input, if used.	
<b>See Also</b>	INPAUSECL, SIGPAUSECL	
<b>Example</b>	Command	Description
	>PAUSECLLV=1	#Set the PAUSECL input logic to Normally Closed
	PAUSECLLV=0(1)	
	>RESET	#RESET the device to initialize the modified PAUSECL
	Resetting system.	setting
-----		
	CRD5xx-KP	
	CRK Series Built-in Controller	
	Software Version: A378 V.x.xx	
	Copyright 2009-2011-2011	
	ORIENTAL MOTOR U.S.A. CORP.	
-----		
	>PAUSECLLV	
	PAUSECLLV=1(1)	#New value is active
	>	

**PAUSELV: PAUSE Input Level****I/O**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	PAUSELV n	
<b>Range</b>	n = 0: Normally Open 1: Normally Closed	
<b>Initial Value</b>	0	
<b>RESET</b>	RESET or recycle the power is required to activate the change. The parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	PAUSELV is the active level of the Pause (PAUSE) input, if used.	
<b>See Also</b>	INPAUSE, SIGPAUSE	
<b>Example</b>	Command	Description
	>PAUSELV=1	#Set the PAUSE input logic to Normally Closed
	PAUSELV=0(1)	
	>RESET	#RESET the device to initialize the modified PAUSE setting
	Resetting system.	
	-----	
	CRD5xx-KP	
	CRK Series Built-in Controller	
	Software Version: A378 V.x.xx	
	Copyright 2009-2011-2011	
	ORIENTAL MOTOR U.S.A. CORP.	
	-----	
	>PAUSELV	#New value is active
	PAUSELV=1(1)	
	>	

**PC: Position Counter****System Status**

<b>Execution Mode</b>	Immediate and Sequence																								
<b>Syntax</b>	PC n																								
<b>Range</b>	n = -8,388,607 to +8,388,607																								
<b>Initial Value</b>	0																								
<b>Access</b>	READ and WRITE READ only while motion is in progress																								
<b>Description</b>	<p>PC is the position counter (or set point).</p> <p>PC is the position that the system has been instructed to go to. In case of using with encoder the feedback position is counted as PF (Position Feedback). The difference between PC and PF is the position error (PE).</p> <p>PC is set to zero (0) at system startup.</p> <p>PC is continuously updated by the system:</p> <ul style="list-style-type: none"> <li>- In normal operations, PC is updated by the internal motion profiler.</li> <li>- If current is OFF, PC is continuously set to feedback position PF (to maintain zero position error while the system is freewheeling).</li> <li>- PC is automatically set to zero (0) after successful completion of a home seeking operation (MGHN, MGHP).</li> </ul> <p>PC can be modified directly, if no motion is in progress (in immediate mode or in sequences). If PC is changed in this way, PF (Position Feedback, actual motor position) is simultaneously changed by the same amount. Changing PC by direct assignment does not cause motion.</p>																								
<b>See Also</b>	MA, MGHN, MGHP, MI, PCI, PE, PF, PFI																								
<b>Note</b>	If the PC value is changed by user, then EC value is automatically modified based on the ratio of EGA and EGB. And vice versa.																								
<b>Example</b>	<table> <tr> <th>Command</th><th>Description</th></tr> <tr> <td>&gt;LIST</td><td>#List sequence named "Origin"</td></tr> <tr> <td>ORIGIN</td><td></td></tr> <tr> <td>( 1) MGHP</td><td>#Seek home: start in the positive direction</td></tr> <tr> <td>( 2) MEND</td><td>#Wait for home operation to finish: home operation sets PC to 0</td></tr> <tr> <td>( 3) PC=45</td><td>#This position is actually 45</td></tr> <tr> <td>( 4) MA 0</td><td>#Go to position zero (PC=0), 45 away from HOMES input location</td></tr> <tr> <td>&gt;RUN</td><td></td></tr> <tr> <td>ORIGIN</td><td></td></tr> <tr> <td>&gt;PC</td><td>#Check the position counter after motion complete</td></tr> <tr> <td>PC=0</td><td></td></tr> <tr> <td>&gt;</td><td></td></tr> </table>	Command	Description	>LIST	#List sequence named "Origin"	ORIGIN		( 1) MGHP	#Seek home: start in the positive direction	( 2) MEND	#Wait for home operation to finish: home operation sets PC to 0	( 3) PC=45	#This position is actually 45	( 4) MA 0	#Go to position zero (PC=0), 45 away from HOMES input location	>RUN		ORIGIN		>PC	#Check the position counter after motion complete	PC=0		>	
Command	Description																								
>LIST	#List sequence named "Origin"																								
ORIGIN																									
( 1) MGHP	#Seek home: start in the positive direction																								
( 2) MEND	#Wait for home operation to finish: home operation sets PC to 0																								
( 3) PC=45	#This position is actually 45																								
( 4) MA 0	#Go to position zero (PC=0), 45 away from HOMES input location																								
>RUN																									
ORIGIN																									
>PC	#Check the position counter after motion complete																								
PC=0																									
>																									

**PCI: Incremental Position Command****System Status**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	PCI	
<b>Range</b>	n = -8,388,607 to +8,388,607	
<b>Access</b>	READ	
<b>Description</b>	<p>PCI is the change in position counter PC since the last motion started.</p> <p>PCI is continuously updated by the system.</p> <p>PCI is set to zero (0) at system startup. PCI is undefined immediately after a mechanical home seeking operation completes (MGHN, MGHP).</p>	
<b>See Also</b>	PC, PE, PF, PFI	
<b>Example</b>	Command	Description
	>LIST AREAOUT2	
	( 1) DIS 3000; VR=1000	#Set distance, velocity
	( 2) MI	#Start move incremental
	( 3) SAS Motion started	#Send message #1
	( 4) WHILE (PCI<1000)	#Wait for PCI to reach 1000
	( 5) WEND	
	( 6) SAS Passed 1000	#Send message #2
	( 7) WHILE (PCI<2000)	#Wait for PCI to reach 2000
	( 8) WEND	
	( 9) SAS Passed 2000	#Send message #3
	(10) MEND	#Wait for motion end
	(11) SAS Reached target	#Send message #4
	(12) END	
	>	
	>RUN AREAOUT2	#Start sequence
	>Motion started	#Message #1
	>Passed 1000	#Message #2
	>Passed 2000	#Message #3
	>Reached target	#Message #4

**PE: Position Error****System Status**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	PE	
<b>Range</b>	n = -8,388,607 to +8,388,607	
<b>Access</b>	READ	
<b>Description</b>	<p>PE is the position error, or the difference between the position counter (PC) and the feedback position (PF). <math>PE = PC - PF</math>.</p> <p>PE is continuously updated by the system, and can be used to monitor the systems response to load conditions.</p>	
<b>See Also</b>	EGA, EGB, PC, PCI, PF, PFI, EC	
<b>Example</b>	Command >LIST CHECKLOAD  ( 1) MCP ( 2) WHILE (IN1=0) ( 3) D=PE-E ( 4) E=E*9 ( 5) E=E+D ( 6) E=E/10 ( 6) WEND ( 7) SSTOP ( 8) MEND ( 9) IF (E>3) ( 10) SAS Load increasing, clean machine. ( 11) ENDIF >	Description #List sequence CHECKLOAD  #Start continuous motion, positive #While Input 1 is OFF. #Capture position error, and... #...Form a simple moving... #...average in 10 samples.  #End of WHILE block #When IN1=1: Start soft stop #Wait for motion to stop #E = averaged position error #if high, send reminder #End of IF block



**PF: Motor Position****System Status**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	PF	
<b>Range</b>	n = -8,388,607 to +8,388,607	
<b>Access</b>	READ	
<b>Description</b>	<p>PF is the feedback position, measured by the encoder (only for encoder type).</p> <p>PF is continuously updated by the system.</p> <p>PF can deviate from the position counter PC, depending on load conditions. The difference between PC and PF is the position error PE.</p> <p>PF cannot be set directly, but does get changed when PC is changed. PF is calculated by the following equation. <math>PF = EC \times (EGB/EGA)</math></p> <p>For example, if PC=0 and PF=1 with some constant load, setting PC=10 adjusts PF to 10 (exact value may vary with load and any small shaft motion).</p>	
<b>See Also</b>	EGA, EGB, PC, PCI, PE, PFI, EC	
<b>Example</b>	Command	Description
	>LIST AREAOUT3	
	( 1) DIS 9000; VR=1000	#Set distance, velocity
	( 2) PC=0	#RESET PC to zero (PF also adjusted)
	( 3) MI	#Start move incremental
	( 4) SAS Motion started	#Send message #1
	( 5) WHILE (PF<3000)	#Wait for PF to reach 3000
	( 6) WEND	
	( 7) SAS Passed 3000	#Send message #2
	( 8) WHILE (PF<6000)	#Wait for PF to reach 6000
	( 9) WEND	
	( 10) SAS Passed 6000	#Send message #3
	( 11) MEND	#Wait for motion end
	( 12) SAS Reached target	#Send message #4
	( 13) END	
	>	
	>RUN AREAOUT3	#Start sequence
	>Motion started	#Message #1
	>Passed 3000	#Message #2
	>Passed 6000	#Message #3
	>Reached target	#Message #4
	>	

**PFI: Incremental Motor Position****System Status**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	PFI	
<b>Range</b>	n = -8,388,607 to +8,388,607	
<b>Access</b>	READ	
<b>Description</b>	<p>PFI is the change in feedback position PF since the last motion started.</p> <p>PFI is continuously updated by the system.</p> <p>PFI is set to zero (0) at system startup. PFI is undefined immediately after a mechanical home seeking operation completes (MGHN, MGHP).</p>	
<b>See Also</b>	EGA, EGB, PC, PCI, PE, PF, EC	
<b>Example</b>	Command	Description
	>LIST AREAOUT2	
	( 1) DIS 3000; VR=1000	#Set distance, velocity
	( 3) MI	#Start move incremental
	( 4) SAS Motion started	#Send message #1
	( 5) WHILE (PFI<1000)	#Wait for PFI to reach 1000
	( 6) WEND	
	( 7) SAS Passed 1000	#Send message #2
	( 8) WHILE (PFI<2000)	#Wait for PFI to reach 2000
	( 9) WEND	
	( 10) SAS Passed 2000	#Send message #3
	( 11) MEND	#Wait for motion end
	( 12) SAS Reached target	#Send message #4
	( 13) END	
	>	
	>RUN AREAOUT2	#Start sequence
	>Motion started	#Message #1
	>Passed 1000	#Message #2
	>Passed 2000	#Message #3
	>Reached target	#Message #4

**POS [x]: Position Array Data****Motion Variables**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	POS [x] n	
<b>Range</b>	x = 1 to 64	
	n = -8,388,607 to +8,388,607	
<b>Initial Value</b>	0	
<b>SAVEPRM</b>	The new value takes effect immediately and the parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	The POS [x] variables provide an array of 64 data values, intended primarily to store predefined positions.	
	The POS [x] variables may be used in immediate mode as arguments to the MA (Move Absolute) command, e.g.:	
	MA POS [7]	
	... will start an absolute motion to the position stored in POS [7].	
	POS [x] data may be entered directly if known, or positions can be interactively found and stored using the TEACH function. See the chart "Teaching Position" for more information on the TEACH function.	
<b>See Also</b>	All POS [x] data can be cleared (initialized to zero) with the CLEARPOS command.	
	MA, PC, TEACH, CLEARPOS, CLEARALL	
<b>Example</b>	Command	Description
	>POS[1]	#Query the value established for POS [1]
	POS[1]=112	
	>MA POS[1]	#Move to POS[1]
	>PC	#When motion is finished, query the position counter value
	PC=112	#Moved as expected, PC=POS[1]
	>POS[2] 236	#Set POS [2] to 236
	POS[2]=236	
	>MA POS[2]	#Move to POS[2]
	>PC	#When motion is finished, query the position command value
	PC=236	#Moved as expected, PC=POS[2]
	>	

**PSTOP: Panic Stop****Motion Commands**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	PSTOP	
<b>Description</b>	<p>PSTOP stops the motor as quickly as possible (hard stop), and then takes the alarm action determined by ALMACT, which may involve setting an alarm (alarm 68h), aborting sequences, and possibly disabling motor current.</p> <p>The PSTOP function may also be executed via the PSTOP input. See the INPSTOP command to assign the PSTOP input.</p> <p>There are several different ways to stop the motor. See the Motor Stop Command list for more information.</p>	
<b>Caution</b>	<p>The PSTOP command will attempt to cause the motor to stop rotating immediately. Use caution when stopping a high speed load using the PSTOP command. The actual distance traveled during a Panic Stop depends on velocity, load, and current settings.</p>	
<b>See Also</b>	<ESC>, MA, MCN, MCP, MGHN, MGHP, MI, SSTOP, HSTOP, ALMACT, ABORT, ABORTACT	
<b>Example</b>	Command	Description
	>VR 4000	#Set the velocity to 4000 pps
	VR=4000	#Device response
	>MCP	#Move continuously in the positive direction
	>PSTOP	#Stop the motor as quickly as possible
	>	

**PSTOPLV: Panic Stop Input Level****I/O**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	PSTOPLV n	
<b>Range</b>	n = 0: Normally Open 1: Normally Closed	
<b>Initial Value</b>	0	
<b>RESET</b>	RESET or recycle the power is required to activate the change. The parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	PSTOPLV is the active level of the Panic Stop (PSTOP) input.	
<b>See Also</b>	SIGPSTOP	
<b>Example</b>	Command	Description
	>PSTOPLV=1	#Set the PSTOP input logic to Normally Closed
	PSTOPLV=0(1)	#RESET the device to initialize the modified PSTOP setting
	>RESET	
	Resetting system.	
	-----	
	CRD5xx-KP	
	CRK Series Built-in Controller	
	Software Version: A378 V.x.xx	
	Copyright 2009-2011-2011	
	ORIENTAL MOTOR U.S.A. CORP.	
	-----	
	>PSTOPLV	#New value is active
	PSTOPLV=1(1)	
	>	

**PSTSLV: Pause Status Output Level****I/O**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	PSTSLV n	
<b>Range</b>	n = 0: Normally Open 1: Normally Closed	
<b>Initial Value</b>	0	
<b>RESET</b>	RESET or recycle the power is required to activate the change. The parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	PSTSLV is the active level of the Pause Status (PSTS) output, if used.	
<b>See Also</b>	OUTPSTS, SIGPSTS	
<b>Example</b>	Command	Description
	>PSTSLV=1	#Set the PSTS output logic to Normally Closed
	PSTSLV=0(1)	
	>RESET	#RESET the device to initialize the modified PSTS setting
	Resetting system.	
	-----	
	CRD5xx-KP	
	CRK Series Built-in Controller	
	Software Version: A378 V.x.xx	
	Copyright 2009-2011	
	ORIENTAL MOTOR U.S.A. CORP.	
	-----	
	>PSTSLV	
	PSTSLV=1(1)	#New value is active
	>	

**READYLV: READY Status Output Level****I/O**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	READYLV n	
<b>Range</b>	n = 0: Normally Open 1: Normally Closed	
<b>Initial Value</b>	0	
<b>RESET</b>	RESET or recycle the power is required to activate the change. The parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	READY is the active level of the READY output, if used.	
<b>See Also</b>	OUTREADY, SIGREADY	
<b>Example</b>	Command	Description
	>READYLV=1	#Set the READY output logic to Normally Closed
	READYLV=0(1)	
	>RESET	#RESET the device to initialize the modified READY
	Resetting system.	setting
	-----	
	CRD5xx-KP	
	CRK Series Built-in Controller	
	Software Version: A378 V.x.xx	
	Copyright 2009-2011	
	ORIENTAL MOTOR U.S.A. CORP.	
	-----	
	>READYLV	
	READYLV=1(1)	#New value is active
	>	

**REN: Rename Sequence****Editor Command**

<b>Execution Mode</b>	Immediate
<b>Syntax</b>	REN target newname
<b>Range</b>	'target' must be the name or number of an existing sequence. 'newname' must be a valid sequence name (consisting of letters or numbers, 10 character maximum, must start with a letter).
<b>Description</b>	REN renames an existing sequence. The new name must be unique. REN can also be used to name a sequence which was created by number only, and has no name. 'target' cannot be renamed if it is locked.
<b>See Also</b>	COPY, DIR, EDIT, DEL, LOCK, UNLOCK
<b>Example</b>	<div>Command</div> <div>Description</div> <div>&gt;DIR</div> <div>#Check the names of all sequences</div>

```
## Name      TextSize Locked
== =====
1 PROG1      37
2 MOVE1      24
```

Total: 2

Executable memory: 4 bytes used of 1600 bytes total, 0 percent.

Storage memory: 18 bytes used of 4223 bytes total, 0 percent.

>REN PROG1 PROG2 #Rename PROG1 to the new name of PROG2

>DIR

```
## Name      TextSize Locked
== =====
1 PROG2      37
2 MOVE1      24
```

Total: 2

Executable memory: 4 bytes used of 1600 bytes total, 0 percent.

Storage memory: 18 bytes used of 4223 bytes total, 0 percent.

>REN PROG2 MOVE1 #Can't rename if new name exists already.

Error: Sequence already exists.

>



**REPORT: Display System Status****Monitor Commands****Execution Mode** Immediate**Syntax** REPORT**Description** REPORT displays a system status summary.

The REPORT command can be an effective tool for troubleshooting problems with the system. The REPORT command displays the status and active level of all of the system's inputs and outputs, the values of important parameters, the value of position counter PC, and the alarm and warning history.

**See Also** ALM, DIR, IO**Example**

Command	Description
>REPORT	#Get a system status summary: sample result follows

```

/ I/O REPORT /---(NO:Normally Open,NC:Normally Close)-----

START(NO)= 0 ABORT(NO)= 0 ALMCLR(NO)= 0 CROFF(NO)= 0
HOME(NO)= 0 +LS(NO)= 0 -LS(NO)= 0 PSTOP(NO)= 0
SENSOR(NO)= 0 HOMES(NO)= 0 SLIT(NO)= 0
IN1(NO)= 0 IN2(NO)= 0 IN3(NO)= 0 IN4(NO)= 0 PAUSE(NO)= 0 PAUSECL(NO)= 0
ALM(NO)= 0 MOVE(NO)= 0
OUT1(NO)= 0 OUT2(NO)= 0 HOMEP(NO)= 0 STO(NO)= 0

/ PARAMETER REPORT /-----

STRSW= 1 DIRINV= 1 CRRUN= 100 CRSTOP= 50
VS= 100 VR= 1000 TA= 500 TD= 500 DIS= 0
LIMP= 8388607 LIMN= -8388607 EGA= 500 EGB= 500

/ POSITION REPORT /-----

PC = 1863 EC = 1863

/ ALARM HISTORY /-----

ALARM = 00 , RECORD : 31 30 23 98 9E 70 30 00 00 00
No alarm.
>

```

**RESET: RESET Device****System Control**

Execution Mode	Immediate	
Syntax	RESET	
Description	<p>RESET Resets the device.</p> <p>Performing a RESET operation is similar to cycling power, but may respond quicker.</p> <p>Several events occur when the device is reset:</p> <ol style="list-style-type: none"><li>1) Motor current is disabled. The motor may move, depending on load conditions: ensure the device is not supporting a vertical load as the load may drop when the device is reset.</li><li>2) The system transmits a message: "Resetting system."</li><li>3) All outputs are set to an open (non-conducting) state.</li><li>4) The parameters saved in EEPROM are established. Any parameter that was not saved is lost. (Use SAVEPRM to save parameter data if desired, before issuing A RESET command.)</li><li>5) Alarm conditions are checked, and the alarm code is updated accordingly.</li><li>6) If motor current is permitted (depending on alarm state, Current Off (CROFF) input, and STRSW setting), current is enabled, and begins to increase from 0 to CRSTOP value.</li><li>7) Outputs are set to appropriate states.</li><li>8) The immediate mode command prompt is transmitted (&gt;). If VERBOSE=1, a system startup banner message appears before the prompt. If a terminal or terminal emulation program is communicating with the system, the terminal screen may clear prior to the banner, depending on emulation mode.</li><li>9) Inputs are read and appropriate actions taken.</li><li>10) If no alarm is set, no sequences are running, and a sequence named CONFIG exists, the CONFIG sequence will begin running automatically.</li></ol> <p>Many parameters do not become effective until the new values have been saved and the system RESET or power cycled. RESET is a convenient way to finish reconfiguring the system without cycling power.</p>	
Caution	<p>When the device is reset, any parameter that was not saved is lost.</p> <p>Use SAVEPRM to save parameter data, if desired, before issuing A RESET command.</p> <p>When the device is reset motor current is disabled (at least momentarily), resulting in no holding torque. Be sure that the load cannot move accidentally. Vertical loads which can freefall should be supported via mechanical brake or other means.</p>	
See Also	STRSW, CRSTOP, CROFFLV, VERBOSE, SAVEPRM	
Example	<div>Command</div> <div>&gt;READYLV=1</div> <div>    READYLV=0(1)</div> <div>&gt;RESET</div> <div>Resetting system.</div>	<div>Description</div> <div>#Set the READY output logic to Normally Closed</div> <div>#RESET the device to initialize the modified READY setting</div>
<div>-----</div> <div>CRD5xx-KP</div> <div>    CRK Series Built-in Controller</div> <div>    Software Version: A378 V.x.xx</div> <div>    Copyright 2009-2011</div> <div>    ORIENTAL MOTOR U.S.A. CORP.</div> <div>-----</div> <div>&gt;READYLV</div> <div>    READYLV=1(1)</div> <div>&gt;</div> <div>#New value is active</div>		

**RET: Sequence Return****Sequence Commands**

<b>Execution Mode</b>	Sequence	
<b>Syntax</b>	RET	
<b>Description</b>	<p>RET terminates processing of the current sequence.</p> <p>If the sequence was CALL'ed from another sequence, the original sequence will resume at the statement following the CALL statement. If the sequence was started with the START input or the RUN command, RET terminates all sequence execution.</p> <p>To unconditionally terminate all sequence processing, use ABORT.</p> <p>All sequences automatically return when all statements have been processed: a RET is not required at the end of sequences (but may be used, if desired).</p>	
<b>See Also</b>	ABORT, CALL, END	
<b>Example</b>	Command	Description
	>LIST MAIN	#List sequence MAIN
	( 1) VR 1000	#Set running velocity to 1000 pps
	( 2) LOOP 10	#Do contents, 10 times
	( 3) MI	#...Start incremental motion
	( 4) CALL WATCHER	#...Call sequence WATCHER
	( 5) ENDL	#End of LOOP block
	( 6) MA 0	#Start absolute move back to 0
	( 7) CALL WATCHER	#Call sequence WATCHER
	>LIST WATCHER	#List sequence WATCHER
	( 1) WHILE (SIGMOVE=1)	#While moving...
	( 2) IF (IN4=1)	#...If Input 4 is asserted
	( 3) CV 500	#.....Change speed to 500 pps
	( 4) MEND	#.....Wait for motion to end
	( 5) <b>RET</b>	<b>#.....and return to caller</b>
	( 6) ENDIF	#...End of IF block
	( 7) WEND	#End of WHILE block
	>	

**RUN: Run Sequence****Sequence Commands**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	RUN [target]	
<b>Range</b>	'target' must be the name or number of an existing sequence.	
<b>Description</b>	<p>RUN starts execution of a sequence.</p> <p>Sequences can also be started with the dedicated START input.</p> <p>Sequences cannot be started if the system has an active alarm condition.</p> <p>Control returns to the command prompt, and sequence execution continues in the background until complete or aborted. Sequences abort automatically if an alarm is detected or the dedicated ABORT input is activated. Sequences can be manually aborted with the ABORT command or an ESCAPE key or character.</p> <p>RUN cannot be used inside sequences. To execute one sequence from within another, use the CALL command.</p>	
<b>Important Interactions</b>	Sequences cannot be edited while a sequence is executing. The system prevents the editor from starting.	
<b>See Also</b>	EDIT, DIR, ABORT, <ESC>	
<b>Example</b>	Command	Description
	>LIST MAIN	#List sequence MAIN
	( 1) VR 1000	#Sequence listing.
	( 2) LOOP 10	
	( 3) MI	
	( 4) CALL WATCHER	
	( 5) ENDL	
	( 6) MA 0	
	( 7) CALL WATCHER	
	>RUN MAIN	#Run sequence MAIN
	>SIGRUN	#Commands can still be executed, while...
	SIGRUN=1	#...sequences execute (SIGRUN=1).
	>	

**RUNLV: RUN Output Level****I/O**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	RUNLV n	
<b>Range</b>	n = 0: Normally Open 1: Normally Closed	
<b>Initial Value</b>	0	
<b>RESET</b>	RESET or recycle the power is required to activate the change. The parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	RUNLV is the active level of the Sequence Running (RUN) output, if used.	
<b>See Also</b>	OUTRUN, SIGRUN	
<b>Example</b>	Command	Description
	>RUNLV=1	#Set the RUN output logic to Normally Closed
	RUNLV=0(1)	
	>RESET	#RESET the device to initialize the modified RUN setting
	Resetting system.	
	-----	
	CRD5xx-KP	
	CRK Series Built-in Controller	
	Software Version: A378 V.x.xx	
	Copyright 2009-2011	
	ORIENTAL MOTOR U.S.A. CORP.	
	-----	
	>RUNLV	
	RUNLV=1(1)	
	>	#New value is active

**SACS: Send ASCII Control String****Sequence Commands**

<b>Execution Mode</b>	Sequence	
<b>Syntax</b>	SACS string	
<b>Range</b>	string : a series of ASCII characters or control codes, maximum 70 characters.	
<b>Description</b>	<p>SACS transmits an ASCII string out the serial port. The string begins with the first non-space character following the SACS command, and continues to the last non-space character on the line. SACS does not append a line terminator (carriage return and linefeed), but instead allows a user to embed ASCII control codes within the string. The normal system prompt is not automatically refreshed immediately after an SACS command. SACS permits almost complete control over the actual contents of the output.</p> <p>SACS supports the normal range of printable ASCII characters, plus most ASCII control codes. Control codes are entered by prefixing a printable character with a caret (^). For instance, to transmit an ASCII "BEL" code (usually interpreted as "beep speaker", or similar), use ^G. For a Carriage Return, followed by a Line Feed, use ^M^J. (SACS, Send ASCII String, automatically appends a Carriage Return + Linefeed pair, and may be easier to use in some applications.)</p> <p>There are several exceptions and extensions to the normal ASCII interpretation of control codes:</p> <ul style="list-style-type: none"> <li>- ^@ (ASCII value NULL, binary 0) is not supported.</li> <li>- ^, followed by a space, transmits one space character (this permits leading or trailing space characters in the output)</li> <li>- ^^ transmits a single caret (^). ^^^ transmits an ASCII CTRL-^, 1Eh.</li> </ul> <p>Other commands and comments cannot follow an SACS command on the same line: they will be considered part of the ASCII string.</p> <p>For other control codes and their usual interpretations, see Appendix.</p>	
<b>Note</b>	When using the CRK Motion Creator GUI, a "Serial Port Framing Error (Check baud rate or connection)" message may be displayed in some instances when the SACS command is used in a sequence while it is running. This is a non-fatal error and the message can be ignored.	
<b>See Also</b>	KB, KBQ, SAS, VIEW	
<b>Example</b>	<p>Command</p> <p>&gt;LIST REFRESH</p> <p>( 1) # VT-100 EMULATION</p> <p>( 2) # 1) CLEAR DISPLAY</p> <p>( 3) # 2) HOME CURSOR</p> <p>( 4) # 3) TRANSMIT PREFERRED PROMPT</p> <p>( 5) SACS ^[[2J^[[H--&gt;</p> <p>&gt;RUN REFRESH</p> <p>&gt;</p> <p>--&gt;</p>	<p>Description</p> <p>#List Sequence "REFRESH"</p> <p>#Comments, in sequence</p> <p>#Transmitted control codes below cause VT-100 displays to clear screen and "home" the cursor. Then: transmit a custom prompt</p>
<b>Note</b>	In a multi-drop configuration, all output from sequence commands is suppressed unless the device has been previously addressed (via @). The KB and KBQ commands will not receive input unless the device has been previously addressed.	

**SAS: Send ASCII String****Sequence Commands**

<b>Execution Mode</b>	Sequence	
<b>Syntax</b>	SAS string	
<b>Range</b>	string : a series of ASCII characters, maximum 70 characters.	
<b>Description</b>	<p>SAS transmits an ASCII string out the serial port, verbatim, appends a Carriage Return and Line Feed pair, and refreshes the system prompt. The ASCII string begins with the first non-space character following the SAS command, and continues to the last non-space character on the line.</p> <p>Other commands and comments cannot follow an SAS command on the same line: they will be considered part of the ASCII string.</p>	
<b>Note</b>	<p>When using the CRK Motion Creator GUI, a “Serial Port Framing Error (Check baud rate or connection)” message may be displayed in some instances when the SAS command is used in a sequence while it is running. This is a non-fatal error and the message can be ignored.</p>	
<b>See Also</b>	SACS, VIEW, KB, KBQ	
<b>Example</b>	Command	Description
	>LIST TRANSMIT2	#List sequence TRANSMIT2
	( 1) SAS Distance:	#Send characters "Distance:", Carriage Return, Linefeed, reprompt.
	( 2) DIS	#Display value of DIS and reprompt
	( 3) SACS Distance:	#Send characters "Distance:", with 1 trailing space
	( 4) VIEW D	#Display value of DIS on SAME line, no reprompt
	( 5) SACS ^M^J	#Send Carriage Return and Line Feed
	>RUN TRANSMIT2	
	>Distance:	
	>1125	
	>Distance: 0	
<b>Note</b>	<p>In a multi-drop configuration, all output from sequence commands is suppressed unless the device has been previously addressed (via @). The KB and KBQ commands will not receive input unless the device has been previously addressed.</p>	

**SAVEPRM: Save Parameters****System Control**

<b>Execution Mode</b>	Immediate
<b>Syntax</b>	SAVEPRM
<b>Description</b>	<p>SAVEPRM saves all parameters that normally not saved for example like VR, VS, TA, TD, DIS and all of related to link index to nonvolatile memory (EEPROM).</p> <p>SAVEPRM affects the values of most parameters at system start (following a power cycle or RESET command). The saved values become the Initial Values of each parameter after a restart. These parameters will have a SAVEPRM entry in this chapter.</p> <p>SAVEPRM requires confirmation. A 'y' (not case sensitive) must be sent before the operation proceeds: any other character aborts the operation.</p>

The EEPROM has a nominal expected lifetime of 100,000 write cycles, which should be sufficient for almost all applications. The SAVEPRM command should not be used automatically (i.e. by a host controller) if it could possibly execute at high frequency. Saving once per day, for instance, yields a nominal expected lifetime of almost 275 years. Saving once per minute reduces the expected lifetime to about 70 days, and is certainly not recommended.

The system keeps a counter of how many times EEPROM has been written (by SAVExxx commands, CLEARxxx commands, or INITxxx commands). This counter is displayed each time any of these commands is executed, for reference.

**Caution** The SAVEPRM command writes to EEPROM. The EEPROM has a nominal expected lifetime of 100,000 write cycles. The SAVEPRM command should not be used automatically (i.e. by a host controller) if it could possibly execute at high frequency.

**See Also** CLEARALL, RESET, INITPRM, CLEARPOS

**Example**

Command	Description
>VR	#Check value of running velocity
VR=500	#Default value, 500 pps
>VR 1000	#Set running velocity 1000
VR=1000	
>RESET	#RESET the system
Resetting system.	

-----

CRD5xx-KP  
 CRK Series Built-in Controller  
 Software Version: A378 V.x.xx  
 Copyright 2009-2011  
 ORIENTAL MOTOR U.S.A. CORP.

-----

>VR	#Check value of running velocity
VR=500	#Didn't stick! Still default 500 pps
>VR 1000	#Set back to 1000 pps
VR=1000	
>SAVEPRM	#SAVEPRM: this will become new startup value
Enter Y to proceed, other key to cancel. Y	#Confirm SAVEPRM
Saving Parameters.....OK.	
>RESET	#RESET the system again
Resetting system.	

-----

CRD5xx-KP  
 CRK Series Built-in Controller  
 Software Version: A378 V.x.xx



Copyright 2009-2011  
ORIENTAL MOTOR U.S.A. CORP.

---

>VR

#Check value again

VR=1000

#OK. We have new startup value

>

**SCEN: Self Correcting Enable****System Control**

<b>Execution Mode</b>	Immediate
<b>Syntax</b>	SCEN n
<b>Range</b>	n = 0: Disable 1: Enable
<b>Initial Value</b>	0
<b>RESET</b>	RESET or recycle the power is required to activate the change. The parameter is saved automatically.
<b>Access</b>	READ and WRITE
<b>Description</b>	Sets whether to enable or disable the self correcting mode. Specifically, the deviation between the command position and encoder counter is monitored. When a misstep has occurred, the deviation error is corrected automatically. When a misstep is detected during motion, the driver will quickly reduce its speed to the “starting velocity” VS, correct the deviation error and then restart motion. By the time the motion is completed, the deviation error will have been corrected.

In the case when the motor is at rest, if a misstep has occurred, the driver will attempt to return to the original position with the motion profile that is set at that time. During this motion, the motor current is set to the value set by the “stop current” CRSTOP parameter while the driver performs the operation to return to the original position. However, if the command position value is out of the range (-8,388,607 to 8,388,607), the driver will not perform the operation to return to the original position.

The value of the deviation error used to determine when a misstep has occurred depends on motor type. The driver distinguishes the motor type being used by the values of the “encoder electronic gear B” EGB parameter and the “motor resolution” MRES parameter. If the base step angle of the motor is 0.72° and the “motor resolution” MRES parameter is “0” (the number of divisions is “1”), set the value of the “encoder electronic gear B” EGB parameter to “500”. If the base step angle of the motor is 0.36° and the “motor resolution” MRES parameter is “0” (the number of divisions is “1”), set the value of the “encoder electronic gear B” EGB parameter to “1000”. (See also “10-9. Encoder electronic gear settings”)

- Note**
- An encoder must be used when utilizing the self correcting function.
  - This function is disabled during return to mechanical home operation.
  - If the “self correcting” parameter is set to “enable”, the “stepout detection” parameter is set to “disabled” internally. In this case, the STO output will not turn ON even if the deviation between the encoder counter value and driver command position reaches the value set in the “stepout detection band” parameter. Also if the “stepout detection action” parameter is set to “alarm” or “warning”, an alarm or warning is not generated when the deviation error was detected.
  - If a motion command is executed while a deviation error is corrected when a motor stops, the “motion while in motion” alarm will occur. Make sure motions are not started while the MOVE output is ON.

**See Also** SCTO, SIGSC, OUTSC, SCLV, EGA, EGB

<b>Example</b>	<b>Command</b>	<b>Description</b>
	>SCEN 1	#Enable the self correcting function
	SCEN=0(1)	
	>RESET	#Reset the device to initialize the modified SCEN setting
	Resetting system.	

-----

CRD5xx-KP  
CRK Series Built-in Controller  
Software Version: A378 V.x.xx  
Copyright 2009-2011  
ORIENTAL MOTOR U.S.A. CORP.

-----

>SCEN  
SCEN=1(1)  
>

#The new setting is active

**SCHGPOS: Distance After SENSOR Input****Motion Variables**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	SCHGPOS n	
<b>Range</b>	n = 0 to +8,388,607	
<b>Initial Value</b>	0	
<b>SAVEPRM</b>	The new value takes effect immediately and the parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	<p>SCHGPOS is the distance used for SENSOR offset operation.</p> <p>SENSOR offset operation allows the system to stop a continuous motion (MCN, MCP) a specified distance after a SENSOR input is detected. The system will change speed to SCHGVR, and eventually decelerate to a stop after the designated distance.</p> <p>SENSORACT must be 2 (offset operation).</p>	
<b>See Also</b>	SCHGVR, SENSORACT, SENSORLV	
<b>Example</b>	Command	Description
	>LIST REGISTER	#List Sequence "REGISTER"
	( 1) SCHGPOS 1000	#Set sensor offset distance to 1000
	( 2) SCHGVR 500	#Set sensor offset speed to 500 pps
	( 3) VR 1000	#Set running velocity to 1000 pps
	( 4) MCP	#Move continuous (positive)
	( 5) WHILE (SIGSENSOR=0)	#Wait for SENSOR input turn on
	( 6) WEND	#System Switch to index mode
	( 7) WHILE (SIGMOVE=1)	#Wait for motion complete
	( 8) WEND	
	( 8) SACS Motion Completed	
	>	

**SCHGVR: Velocity After SENSOR Input****Motion Variables**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	SCHGVR n	
<b>Range</b>	n = 1 to 500,000 pps	
<b>Initial Value</b>	1000	
<b>SAVEPRM</b>	The new value takes effect immediately and the parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	<p>SCHGVR is the velocity used for SENSOR offset operation.</p> <p>SENSOR offset operation allows the system to stop a continuous motion (MCN, MCP) a specified distance after a SENSOR input is detected. The system will change speed to SCHGVR, and eventually decelerate to a stop after the designated distance.</p> <p>SENSORACT must be 2 (offset operation).</p>	
<b>See Also</b>	SCHGPOS, SENSORACT, SENSORLV	
<b>Example</b>	Command	Description
	>LIST REGISTER	#List Sequence "REGISTER"
	( 1) SCHGPOS 1000	#Set sensor offset distance to 1000
	( 2) SCHGVR 500	#Set sensor offset speed to 500 pps
	( 3) VR 1000	#Set running velocity to 1000 pps
	( 4) MCP	#Move continuous (positive)
	( 5) WHILE (SIGSENSOR=0)	#Wait for SENSOR input turn on
	( 6) WEND	#System Switch to index mode
	( 7) WHILE (SIGMOVE=1)	#Wait for motion complete
	( 8) WEND	
	( 8) SACS Motion Completed	
	>	

**SCLV: SC Output Level****I/O**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	SCLV n	
<b>Range</b>	n = 0: Normally Open 1: Normally Closed	
<b>Initial Value</b>	0	
<b>RESET</b>	RESET or recycle the power is required to activate the change. The parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	SCLV is the active level of the Self Correction (SC) output.	
<b>See Also</b>	SCEN, SCTO, SIGSC, OUTSC, EGA, EGB	
<b>Example</b>	Command	Description
	>SCLV 1	#Set SC active level to Normally Closed
	SCLV=0(1)	
	>RESET	#RESET to activate new settings
	Resetting system.	
	-----	
	CRD5xx-KP	
	CRK Series Built-in Controller	
	Software Version: A378 V.x.xx	
	Copyright 2009-2011-2011	
	ORIENTAL MOTOR U.S.A. CORP.	
	-----	
	>SCLV	
	SCLV=1(1)	#New value is active
	>	

**SCTO: Self Correcting Timeout****System Control**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	SCTO n	
<b>Range</b>	n = 0 to 10.000 [sec]	
<b>Initial Value</b>	1.000	
<b>SAVEPRM</b>	The new value takes effect immediately and the parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	Sets the maximum time for when the device determines that an over load (missed steps keep occurring) condition exists. If the overload condition persists for longer than the value set with the SCTO parameter, an alarm is generated.	
<b>See Also</b>	OUTSC, SCEN, SCLV, SIGSC, EGA, EGB	
<b>Example</b>	Command	Description
	>SCTO 5.000	#Set SCTO to 5 seconds
	SCTO=5.000	#Display the new value
	>	

**SENSORACT: SENSOR Input Action****System Control**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	SENSORACT n	
<b>Range</b>	n = 0: Hard Stop (stop as quickly as possible) 1: Soft Stop (controlled deceleration over time) 2: Soft stop at fixed distance from SENSOR signal	
<b>Initial Value</b>	2	
<b>RESET</b>	RESET or recycle the power is required to activate the change. The parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	<p>SENSORACT establishes the stop action taken when the system detects a SENSOR input signal while executing a continuous motion (MCN, MCP).</p> <p>If SENSORACT=0, the system will stop the motor as quickly as possible (hard stop). Stop action is exactly the same as HSTOP.</p> <p>If SENSORACT=1, the system will stop the motor by a controlled deceleration over time (soft stop). Stop action is exactly the same as SSTOP.</p> <p>If SENSORACT=2, the system will change speed to SCHGVR, and bring the motor to a stop at a distance SCHGPOS from the position at which the SENSOR signal was detected.</p> <p>SENSOR input behavior is different during home seeking operations (MGHN, MGHP).</p> <p>SENSORACT does not affect the use of the SENSOR input during home seeking.</p>	
<b>See Also</b>	SCHGPOS, SCHGVR, SENSORLV	
<b>Example</b>	Command >SENSORLV 1 SENSORLV=0(1) > <b>SENSORACT 1</b> SENSORACT=2(1) >RESET Resetting system.	Description #Set SENSOR active level to Normally Closed #Set SENSORACT to 1: Soft Stop #RESET to activate new settings
<hr/> CRD5xx-KP CRK Series Built-in Controller Software Version: A378 V.x.xx Copyright 2009-2011 ORIENTAL MOTOR U.S.A. CORP. <hr/> >		



**SENSORLV: SENSOR Input Level****I/O**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	SENSORLV n	
<b>Range</b>	n = 0: Normally Open 1: Normally Closed	
<b>Initial Value</b>	0	
<b>RESET</b>	RESET or recycle the power is required to activate the change. The parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	SENSORLV is the active level of the SENSOR input.	
<b>See Also</b>	SIGSENSOR	
<b>Example</b>	Command	Description
	>SENSORLV 1	#Set SENSOR active level to Normally Closed
	SENSORLV=0(1)	#Set SENSORACT to 1: Soft Stop
	>SENSORACT 1	
	SENSORACT=2(1)	
	>RESET	#RESET to activate new settings
	Resetting system.	
	-----	
	CRD5xx-KP	
	CRK Series Built-in Controller	
	Software Version: A378 V.x.xx	
	Copyright 2009-2011	
	ORIENTAL MOTOR U.S.A. CORP.	
	-----	
	>	

**SIGALM: System ALARM Output Signal****System Status**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	SIGALM	
<b>Range</b>	0: No alarm	
	1: Active alarm condition	
<b>Initial Value</b>	0	
<b>Access</b>	READ	
<b>Description</b>	SIGALM is the system alarm signal.	
	SIGALM is continuously updated by the system. If the system has an active alarm condition, SIGALM will be one (1), otherwise it will be zero (0).	
	The active level of the output can be set with ALMLV.	
	Although SIGALM is available within sequences, it is not really meaningful there. If an alarm is detected, all sequence processing is aborted: sequences can never detect SIGALM=1.	
<b>See Also</b>	ALMLV, ALMACT, OUTSG	
<b>Example</b>	Command	Description
	>ALMSET	#Set the device in an alarm condition
	>SIGALM	#Query the status of the ALARM signal
	SIGALM=1	#Alarm condition present
	>ALMCLR	#Clear the alarm condition
	>SIGALM	#Query the status of the ALARM signal again
	SIGALM=0	
	>	

**SIGALMCLR: Functional ALMCLR Signal****System Status**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	SIGALMCLR	
<b>Range</b>	0: OFF	
	1: ON	
<b>Initial Value</b>	0	
<b>Access</b>	READ	
<b>Description</b>	Display the status of the internal ALMCLR (Device ALARM CLEAR) signal. Allows the user to read the active level of the assigned input and the device condition.	
<b>See Also</b>	ALMCLRLV, INSG, OUTSG	
<b>Example</b>	Command	Description
	>SIGALMCLR	#Query the functional status of the ALMCLR input
	SIGALM=1	#Device response when input is active
	>	

**SIGAREA: System AREA Output Signal****System Status****Execution Mode** Immediate and Sequence**Syntax** SIGAREA**Range** 0: OFF  
1: ON**Initial Value** 0**Access** READ**Description** SIGAREA is the system AREA output signal state.  
SIGAREA is continuously updated by the system, and reflects the state of the AREA output.  
This signal will be output when the motor output shaft is inside the area set by the “Area 1” and “Area 2” parameters. This signal is also output while the motor is stopped.  
The active level of the output can be set with AREALV.**See Also** AREALV, AREA n, OUTSG

<b>Example</b>	Command	Description
	>AREA1 1000	#Set the AREA1 position to 1000
	AREA1=1000	#Display the AREA1 value
	>AREA2 -1000	#Set the AREA2 position to -1000
	AREA2=-1000	#Display the AREA2 value
	>MA 0	#Start absolute motion to position 0
	>SIGAREA	#Check AREA status
	SIGAREA=1	#Current position is in the area specified
	>MA 1100	#Start absolute motion to position 1100
	>SIGAREA	#Check AREA status again
	SIGAREA=0	#Current position is out of the area
	>	

**SIGCROFF: System Current Off Input Signal****System Status****Execution Mode** Immediate and Sequence**Syntax** SIGCROFF**Range** 0: CROFF input inactive  
1: CROFF input active**Initial Value** 0**Access** READ**Description** SIGCROFF is the system external Current Off (CROFF) input signal state.  
SIGCROFF is continuously updated by the system, and reflects the state of the Current Off (CROFF) input, if used.

The active level of the input can be set with CROFFLV.

SIGCROFF does not reflect the actual state of motor current. For instance, if current has been explicitly disabled (by CURRENT=0), while CROFF input is active, SIGCROFF is zero (0).

**See Also** CROFFLV, INSG, CURRENT**Example**

Command	Description
( 1)LOOP	#Start infinite loop
( 2)WHILE (SIGCROFF=1)	#While CROFF (current OFF) active
( 3)WEND	#Repeat this line
( 4)IF (PC!=0)	#If Position counter moved off 0...
( 5)MA 0	#... Move back to 0
( 6)MEND	#... and wait for motion to complete
( 7)ENDIF	#End IF block
( 8)ENDL	#end LOOP block
>	

**SIGHOME: System HOME Input Signal****System Status****Execution Mode** Immediate and Sequence**Syntax** SIGHOME**Range** 0: OFF  
1: ON (Executing homing operation)**Initial Value** 0**Access** READ**Description** SIGHOME is the system external Homing operation (HOME) input signal state.  
SIGHOME is continuously updated by the system, and reflects the state of the HOME input.  
The homing operation starts when the HOME input turns ON.  
The active level of the input can be set with HOMELV.**See Also** HOMELV**Example**

Command	Description
>SIGHOME	#Query the functional status of the HOME input
SIGHOME=0	#Device responses when input is active.
>	

**SIGHOME: System Home Position Output Signal****System Status**

<b>Execution Mode</b>	Immediate and Sequence
<b>Syntax</b>	SIGHOME
<b>Range</b>	0: Away from HOME position 1: At HOME position
<b>Initial Value</b>	0
<b>Access</b>	READ
<b>Description</b>	<p>SIGHOME is the system HOME signal, active while on a valid HOME position, and inactive otherwise.</p> <p>SIGHOME is always 0, until the system has successfully executed a homing operation (MGHN, MGHP). After a successful homing operation, SIGHOME=1 when position counter PC=0, and SIGHOME=0 otherwise.</p> <p>SIGHOME is continuously updated by the system.</p> <p>The active level of the output can be set with HOMEPLV.</p>

**See Also**

OUTHOME, HOMEPLV, OUTSG, MGHP, MGHN, PC

**Example**

Command	Description
>PC	#Check position counter PC
PC = 0	#Zero (typical for startup)
>SIGHOME	#Check SIGHOME
SIGHOME=0	#Zero (0): not at home.
>MA 0	#Absolute move, to PC=0
>SIGHOME	#Check SIGHOME again
SIGHOME=1	#Now 1, because of PC=0
>PC 22	#Set PC to 22
PC=22	
>SIGHOME	#Check SIGHOME
SIGHOME=0	#Not at home
>MA 0	#Absolute move, to PC=0
>SIGHOME	#Check HOME
SIGHOME=1	#OK: this is the new home
>	

**SIGHOMES: System Home Sensor Input Signal****System Status**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	SIGHOMES	
<b>Range</b>	0: Not at home position 1: At home position	
<b>Initial Value</b>	0	
<b>Access</b>	READ	
<b>Description</b>	<p>SIGHOMEP is the system external Home Position Sensor (HOMES) input signal state.</p> <p>SIGHOMEP is continuously updated by the system, and reflects the state of the HOMEP output.</p> <p>This input detects the mechanical home position when a return-to-home operation is executed in the 3-sensor mode.</p> <p>The active level of the input can be set with HOMESLV.</p>	
<b>See Also</b>	HOMESLV	
<b>Example</b>	Command	Description
	>LIST SLIPCHECK	#List sequence SLIPCHECK
	( 1) MA 0	#Return to home position (PC=0)
	( 2) MEND	#Wait for motion to complete
	( 3) IF (SIGHOMES!=1)	#If HOMES input not active...
	( 4) SAS No home input at home position.	#...Problem. Transmit messages
	( 5) SAS Check linkage and sensor.	
	( 6) ALMSET	#Set an alarm
	( 7) ENDIF	#End of IF block
	>	



**SIGLSN: System Limit Switch Negative Input Signal****System Status**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	SIGLSN	
<b>Range</b>	0: Negative Limit Sensor inactive 1: Negative Limit Sensor active	
<b>Initial Value</b>	0	
<b>Access</b>	READ	
<b>Description</b>	<p>SIGLSN is the system external Negative Limit Sensor (-LS) input signal state.</p> <p>SIGLSN is continuously updated by the system, and reflects the state of the -LS input, if used. If -LS has not been assigned to an input, SIGLSN is always zero (0).</p> <p>The active level of the input can be set with OTLV.</p>	
<b>See Also</b>	OTLV, OTACT, INSG, HOMETYP, MGHP, MGHN	
<b>Example</b>	Command	Description
	>LSEN 0	#Disable the software limits
	LSEN=0	
	>	
	>LIST FIXLIMITS	#List sequence FIXLIMITS
	 ( 1) IF (SIGLSN=1)	
	( 2) MCP	#If SIGLSN=1, negative limit sensor active
	( 3) WHILE (SIGLSN=1); WEND	#Start moving continuously, positive direction
	( 4) ENDIF	#While the sensor is still active, wait
	( 5) IF (SIGLSP=1)	#End IF block
	( 6) MCN	#If SIGLSP=1, positive limit sensor active
	( 7) WHILE (SIGLSP=1); WEND	#Start moving continuously, negative direction
	( 8) ENDIF	#While the sensor is still active, wait
	( 9) SSTOP	#End IF block
	(10) MEND	#Stop the motor (soft stop)
	>ALM	#Wait for stop to finish
		#Check alarm
	ALARM=66 , RECORD : 66 70 E0 06 00 00 00 00 00 00	
	>ALMCLR	#Limit sensor alarm: clear it.
	>RUN FIXLIMITS	#Run sequence FIXLIMITS to get back within limits
	>SIGLSN; SIGLSP	#Check limits
	SIGLSN=0	#Negative...
	SIGLSP=0	#...and positive limit sensors inactive. Recovered.
	>	

**SIGLSP: System Limit Switch Positive Input Signal****System Status**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	SIGLSP	
<b>Range</b>	0: Positive Limit Sensor inactive 1: Positive Limit Sensor active	
<b>Initial Value</b>	0	
<b>Access</b>	READ	
<b>Description</b>	<p>SIGLSP is the system external Positive Limit Sensor (+LS) input signal state.</p> <p>SIGLSP is continuously updated by the system, and reflects the state of the +LS input, if used. If +LS has not been assigned to an input, SIGLSP is always zero (0).</p> <p>The active level of the input can be set with OTLV.</p>	
<b>See Also</b>	OTLV, OTACT, INSG, HOMETYP, MGHP, MGHN	
<b>Example</b>	Command	Description
	>LSEN 0	#Disable the software limits
	LSEN=0	
	>	
	>LIST FIXLIMITS	#List sequence FIXLIMITS
	( 1) IF (SIGLSN=1)	#If SIGLSN=1, negative limit sensor active
	( 2) MCP	#Start moving continuously, positive direction
	( 3) WHILE (SIGLSN=1); WEND	#While the sensor is still active, wait
	( 4) ENDIF	#End IF block
	( 5) IF (SIGLSP=1)	#If SIGLSP=1, positive limit sensor active
	( 6) MCN	#Start moving continuously, negative direction
	( 7) WHILE (SIGLSP=1); WEND	#While the sensor is still active, wait
	( 8) ENDIF	#End IF block
	( 9) SSTOP	#Stop the motor (soft stop)
	(10) MEND	#Wait for stop to finish
	>ALM	#Check alarm
	ALARM=66 , RECORD : 66 70 E0 06 00 00 00 00 00 00	
	>ALMCLR	#Limit sensor alarm: clear it.
	>RUN FIXLIMITS	#Run sequence FIXLIMITS to get back within limits
	>SIGLSN; SIGLSP	#Check limits
	SIGLSN=0	#Negative...
	SIGLSP=0	#...and positive limit sensors inactive. Recovered.
	>	

**SIGMOVE: System MOVE Output Signal****System Status**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	SIGMOVE	
<b>Range</b>	0: Motor is not executing a motion command 1: Motor is executing a motion command	
<b>Initial Value</b>	0	
<b>Access</b>	READ	
<b>Description</b>	<p>SIGMOVE is the system MOVE signal, active (1) while the system is executing any motion, and inactive (0) otherwise.</p> <p>SIGMOVE is updated based on position counter (PC). When the system is changing PC (intentionally moving), SIGMOVE=1, and SIGMOVE=0 otherwise. The motor may actually move, because of loading conditions, even if the system is not executing a motion.</p> <p>SIGMOVE is continuously updated by the system.</p> <p>The active level of the output can be set with MOVELV.</p>	
<b>See Also</b>	MOVELV, OUTSG	
<b>Example</b>	<p>Command</p> <p>&gt;LIST GOHOME</p> <p>( 1) SAS Home Requested</p> <p>( 2) IF (SIGMOVE=1)</p> <p>( 3) SAS System moving, please wait...</p> <p>( 4) MEND</p> <p>( 5) ENDIF</p> <p>( 6) SAS Returning to home position.</p> <p>( 7) MA 0</p> <p>( 8) MEND</p> <p>( 9) SAS At home position.</p> <p>&gt;</p>	<p>Description</p> <p>#List sequence GOHOME</p> <p>#Transmit "Home Requested"</p> <p>#If motion in progress</p> <p>#Transmit wait message</p> <p>#Wait for motion to finish</p> <p>#End IF block</p> <p>#Transmit returning message</p> <p>#Move to position zero</p> <p>#Wait for motion to complete</p> <p>#Transmit finished message</p>

**SIGPAUSE: System PAUSE Input Signal****System Status**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	SIGPAUSE	
<b>Range</b>	0: PAUSE input not asserted 1: PAUSE input asserted	
<b>Initial Value</b>	0	
<b>Access</b>	READ	
<b>Description</b>	<p>SIGPAUSE reflects the state of the external Motion Pause (PAUSE) input.</p> <p>SIGPAUSE will be active (1) when the PAUSE signal has been assigned to an input (INPAUSE != 0) and the PAUSE input is asserted. If PAUSE has not been assigned to an input, SIGPAUSE is always zero (0).</p> <p>The PAUSE input is used to interrupt a motion and decelerate to a stop. The motion can be resumed later using the Continue command (CONT), if sequences are executing, asserting the START input. The Pause Clear (PAUSECL) input or PAUSECLR command can be used to abandon the remainder of a PAUSE'd motion.</p> <p>The active level of the input can be set with PAUSELV.</p>	
<b>See Also</b>	INPAUSE, PAUSELV, INSG, PAUSE, PAUSECLR, CONT	
<b>Example</b>	Command	Description
	>LIST WATCHPAUSE	#List sequence WATCHPAUSE
	( 1) MA X	#Start motion, to position in variable 'X'
	( 2) WHILE (PC != X)	#While position counter still not 'X'
	( 3) IF (SIGPAUSE=1)	#If PAUSE input detected
	( 4) WHILE (SIGPAUSE=1); WEND	#Wait for PAUSE input to clear
	( 5) CONT	#Resume motion
	( 6) ENDIF	#End of IF block
	( 7) WEND	#End of WHILE block
	>	

**SIGPAUSECL: System Pause Clear Input Signal****System Status**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	SIGPAUSECL	
<b>Range</b>	0: PAUSECL input not asserted 1: PAUSECL input asserted	
<b>Initial Value</b>	0	
<b>Access</b>	READ	
<b>Description</b>	<p>SIGPAUSECL reflects the state of the external Pause Clear (PAUSECL) input.</p> <p>SIGPAUSECL will be active (1) when the PAUSECL signal has been assigned to an input (INPAUSECL != 0) and the PAUSECL input is asserted. If PAUSECL has not been assigned to an input, SIGPAUSECL is always zero (0).</p> <p>The Pause Clear (PAUSECL) input causes a previously paused motion to be abandoned.</p> <p>The active level of the input can be set with PAUSECLLV.</p>	
<b>See Also</b>	INPAUSECL, PAUSECLLV, INSG, PAUSE, PAUSECLR, CONT	
<b>Example</b>	Command	Description
	>LIST DIRSWITCH	#List sequence DIRSWITCH
	( 1) IF (SIGPAUSECL=1)	#Re-use PAUSECL as direction select, if I/O budget tight
	( 2) DIS=D	#If asserted, distance = +D
	( 3) ELSE	
	( 4) DIS=-D	#Otherwise, distance = -D
	( 5) ENDIF	#End of IF block
	( 6) MI	#Start incremental motion
	( 7) MEND	#Wait for motion end
	>D 100	
	D=100	
	>RUN DIRSWITCH	

**SIGPSTOP: System Panic Stop Input Signal****System Status**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	SIGPSTOP	
<b>Range</b>	0: PSTOP input inactive 1: PSTOP input active	
<b>Initial Value</b>	0	
<b>Access</b>	READ	
<b>Description</b>	<p>SIGPSTOP is the system external Panic Stop (PSTOP) input signal state.</p> <p>The PSTOP input stops motion as quickly as possible when activated (hard stop), and then takes action defined by ALMACT. While PSTOP is active, motions cannot be started.</p> <p>SIGPSTOP is continuously updated by the system, and reflects the state of the PSTOP input, if used. If PSTOP has not been assigned to an input, SIGPSTOP is always zero (0).</p> <p>The active level of the input can be set with PSTOPLV.</p>	
<b>See Also</b>	PSTOPLV, INSG, PSTOP, ALMACT	
<b>Example</b>	Command	Description
	( 1) IF (SIGPSTOP=1)	#Check PSTOP before moving
	( 2) SAS NoGo: STOP input active.	#If active, transmit message...
	( 3) RET	#...and return (No move, avoid an alarm.)
	( 4) ENDIF	#End of IF block
	( 5) MI	#Start incremental motion
	( 6) MEND	#Wait for motion to end.
	>	

**SIGPSTS: System Pause Status Output Signal****System Status**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	SIGPSTS	
<b>Range</b>	0: Motion not paused. 1: Motion paused.	
<b>Initial Value</b>	0	
<b>Access</b>	READ	
<b>Description</b>	<p>SIGPSTS is the system Pause Status (PSTS) signal, active when a motion has been paused, and inactive otherwise.</p> <p>SIGPSTS is continuously updated by the system. When motion is paused (by PAUSE input or PAUSE command), PSTS becomes active. It remains active until motion is continued (by CONT command, or START input if sequences are executing), or cleared (by PAUSECLR input or PAUSECLR command), or another motion is started.</p> <p>The active level of the output can be set with PSTSLV.</p>	
<b>See Also</b>	OUTPSTS, PSTSLV, OUTSG, PAUSE, PAUSECLR, CONT	
<b>Example</b>	Command	Description
	>LIST PULSEOUT	#List sequence PULSEOUT
	( 1) MA 0	#Start absolute move to position 0
	( 2) WHILE (PC != 0)	#While we aren't there yet
	( 3) IF (SIGPSTS=1)	#If we are "paused" (by PAUSE input)
	( 4) A=TIMER % 250	#Variable A = TIMER modulo 1: ramp from 0 to 0.249
	( 5) IF (A>=125)	#Toggle OUT4 based on value of A
	( 6) OUT4=1	#A>125, OUT4=1
	( 7) ELSE	
	( 8) OUT4=0	#A<125, OUT4=0
	( 9) ENDIF	#End IF block. Results in 4 "blinks" per second
	(10) ELSE	
	(11) OUT4=0	#If not paused: OUT4=0
	(12) ENDIF	#End IF block
	(13) WEND	#End WHILE block
	>	

**SIGREADY: System Ready Status Output Signal****System Status**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	SIGREADY	
<b>Range</b>	0: Not ready to operate 1: Ready to operate	
<b>Initial Value</b>	0	
<b>Access</b>	READ	
<b>Description</b>	<p>SIGREADY is the system Ready Status (READY) signal, active when the driver becomes ready. SIGREADY is continuously updated by the system, and reflects the state of the READY output. The READY output remains OFF in the following conditions:</p> <ul style="list-style-type: none"> <li>• The motor is operating</li> <li>• An alarm is present</li> <li>• Any one of the HOME input and START input is ON</li> <li>• The CROFF input is ON</li> <li>• The ABORT input is ON (Not including ABORT status of START input when set to act as a toggle switch (STARTACT=1)).</li> <li>• The PSTOP input is ON</li> <li>• The motor is not excited</li> <li>• Immediately after the power was turned on</li> </ul> <p>The active level of the output can be set with READYLV.</p>	
<b>See Also</b>	READYLV	
<b>Example</b>	Command	Description
	>LIST READY	#List sequence READY
	( 1) VR 1000	#Sequence listing
	( 2) TA 0.5;TD 0.5	
	( 3) DIS 5000	
	( 4) SIGREADY	#System is ready to operate
	( 5) MI	
	( 6) SIGREADY	#System is busy running motion
	( 7) MEND	
	( 8) SIGREADY	#System is ready again to operate
	( 9) END	
	>RUN READY	#Run sequence READY
	SIGREADY=1	#SIGREADY is 1 before index motion
	SIGREADY=0	#SIGREADY is 0 while index motion
	SIGREADY=1	#SIGREADY is 1 after the index motion
	>	



**SIGRUN: System RUN Output Signal****System Status**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	SIGRUN	
<b>Range</b>	0: Sequences not executing 1: Sequences executing	
<b>Initial Value</b>	0	
<b>Access</b>	READ	
<b>Description</b>	<p>SIGRUN is the system RUN signal, active (1) while the system is executing any sequence, and inactive (0) otherwise.</p> <p>SIGRUN is continuously updated by the system.</p> <p>SIGRUN can be polled from the serial port to check for sequence completion. Because SIGRUN is always one (1) when sequences are executing, it has no real utility in sequences.</p> <p>The active level of the output can be set with RUNLV.</p>	
<b>See Also</b>	OUTRUN, RUNLV, OUTSG, RUN, ABORT	
<b>Example</b>	Command	Description
	>LIST	#List sequence GOHOME
	GOHOME	
	( 1) TA 10	
	( 2) TD 0.5	
	( 3) VR 100	
	( 4) DIS 500	
	( 5) MI	
	( 6) MEND	
	>RUN	#Run sequence GOHOME
	GOHOME	#Host system periodically polls SIGRUN to test for completion
	>SIGRUN	#Sequence is still running
	SIGRUN=1	
	>SIGRUN	#Sequence is still running
	SIGRUN=1	
	>SIGRUN	#Sequence is still running
	SIGRUN=1	
	>SIGRUN	#Sequence is still running
	SIGRUN=1	
	>SIGRUN	#Sequence is still running
	SIGRUN=1	
	>SIGRUN	#Sequence is finished
	SIGRUN=0	
	>	

**SIGSC: System SC Output Signal****System Status**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	SIGSC	
<b>Range</b>	0: Self Correction function was not activated 1: Self Correction function was activated	
<b>Initial Value</b>	0	
<b>Access</b>	READ	
<b>Description</b>	<p>SIGSC is the system external Self Correction (SC) output signal state.</p> <p>SIGSC is continuously updated by the system, and reflects the state of the SC output.</p> <p>The signal is only effective when an encoder is connected.</p> <p>This signal is output when a step deviation error has occurred and was corrected automatically. The SC output will turn OFF when the next motion command is executed or if the motor current is turned OFF. If the SC output is to be used, set the Self Correcting (SCEN) parameter to enable.</p> <p>The active level of the SC output can be set with SCLV command.</p>	
<b>See Also</b>	SCEN, SCTO, OUTSC, SCLV, EGA, EGB	
<b>Note</b>	<ul style="list-style-type: none"> <li>• While the motor is not excited, the SC output is always OFF. The signal will become effective once the motor has remained excited for at least 500 msec.</li> <li>• The SC output remains OFF during homing operation.</li> </ul>	
<b>Example</b>	<p>Command</p> <p>&gt; SIGSC</p> <p>SIGSC=0</p> <p>&gt;</p>	<p>Description</p> <p>#Query the Self Correction function signal status</p> <p>#Display the signal status</p>

**SIGSENSOR: System SENSOR Input Signal****System Status**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	SIGSENSOR	
<b>Range</b>	0: SENSOR input inactive 1: SENSOR input active	
<b>Initial Value</b>	0	
<b>Access</b>	READ	
<b>Description</b>	<p>SIGSENSOR is the system external Sensor (SENSOR) input signal state.</p> <p>SIGSENSOR is continuously updated by the system, and reflects the state of the SENSOR input. The SENSOR input can stop continuous motions MCN and MCP: the actual stopping behavior is determined by SENSORACT.</p> <p>The active level of the input can be set with SENSORLV.</p>	
<b>See Also</b>	SENSORLV, SENSORACT, INSG	
<b>Example</b>	Command	Description
	>SENSORACT	#Check Sensor Action (SENSORACT)
	SENSORACT=0(0)	#0: Hard stop when sensor detected.
	>LIST SIMPLEHOME	#List sequence SIMPLEHOME
	( 1) VR 1000	#Running velocity to 1000 pps
	( 2) MCP	#Move continuous, positive direction
	( 3) MEND	#Wait for motion to stop re: SENSOR
	( 4) IF (SIGSENSOR=1)	#If SENSOR still active (we stopped at right location)
	( 5) PC=0	#Set position counter PC to 0. This is "home"
	( 6) ELSE	
	( 7) ALMSET	#Sensor no longer active, motion overshoot. Force alarm.
	( 8) ENDIF	#End of IF block
	>	

**SIGSLIT: System Slit Sensor Input Signal****System Status**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	SIGSLIT	
<b>Range</b>	0: SLIT input inactive 1: SLIT input active	
<b>Initial Value</b>	0	
<b>Access</b>	READ	
<b>Description</b>	<p>SIGSLIT is the system external Slit Sensor (SLIT) input signal state.</p> <p>SIGSLIT is continuously updated by the system, and reflects the state of the SLIT input.</p> <p>The SLIT input can also affect homing operations, depending on SLITEN: refer to the mechanical homing entry for more detail.</p> <p>The active level of the input can be set with SLITLV.</p>	
<b>See Also</b>	SLITLV, SLITEN, INSG	
<b>Example</b>	Command	Description
	>SIGSLIT	#Query the functional status of the SLIT input
	SIGSLIT=0	#Device responses when input is active.
	>	

**SIGSTO: System STO Output Signal****System Status**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	SIGSTO	
<b>Range</b>	0: Deviation normal 1: Deviation abnormal (misstep detected)	
<b>Initial Value</b>	0	
<b>Access</b>	READ	
<b>Description</b>	<p>SIGSTO is the system external STO output signal state.</p> <p>SIGSTO is continuously updated by the system, and reflects the state of the STO output.</p> <p>This signal becomes effective when an encoder is connected, and a deviation error occurs.</p> <p>This signal will be output when the deviation between the encoder counter value and driver command position reaches the value set in the “stepout detection band” parameter. If the STO output is to be used, set the “stepout detection” parameter to “enable”.</p> <p>The active level of the STO output can be set with STOLV.</p>	
<b>See Also</b>	EGA, EGB, STOLV, STOEN, STOB, STOACT, OUTSG	
<b>Note</b>	<ul style="list-style-type: none"> <li>• While the motor is not excited, the STO output is always OFF. The signal will become effective once the motor has remained excited for at least 500 ms.</li> <li>• The STO output remains OFF during homing operation.</li> <li>• The STO output will not turn ON if the SCEN parameter is set to “enable”.</li> </ul>	
<b>Example</b>	Command	Description
	>SIGSTO	#Query the signal status SIGSTO
	SIGSTO=0	#Display the signal status SIGSTO
	>	

**SIGTEMP: System TEMP Output Signal****System Status**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	SIGSTO	
<b>Range</b>	0: No temperature warning 1: Temperature warning	
<b>Initial Value</b>	0	
<b>Access</b>	READ	
<b>Description</b>	<p>SIGTEMP is the system TEMP signal, active when drive temperature is above warning levels, and inactive otherwise.</p> <p>SIGTEMP is continuously updated by the system. If drive temperature DTMP exceeds drive temperature warning level DTMPWNG, SIGTEMP will be one (1). Otherwise, SIGTEMP will be zero (0).</p> <p>The active level of the output can be set with TEMPLV.</p>	
<b>See Also</b>	OUTTEMP, OUTSG, TEMPLV, DTMP, DTMPWNG	
<b>Example</b>	Command	Description
	>LIST MAINACTION	#List sequence MAINACTION
	( 1) LOOP 10	#Repeat 10 times
	( 2) MI	#Start incremental motion
	( 3) MEND	#Wait for motion to end
	( 4) WHILE (IN6=1); WEND	#Wait for IN6 to become 0
	( 5) WHILE (IN6=0); WEND	#Wait for IN6 to become 1 again
	( 6) ENDL	#End LOOP block
	( 7) MA 0	#Start absolute motion to position 0
	( 8) MEND	#Wait for motion stop
	( 9) IF (SIGTEMP=1)	#If SIGTEMP=1, DTMP getting high
	( 10) SACS ^M^JHigh Temp.^G	#Transmit control string, then 'beep'
	( 11) ENDIF	
	>	

**SIGWNG: System Warning Output Signal****System Status**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	SIGWNG	
<b>Range</b>	0: No warning	
	1: Active warning condition	
<b>Initial Value</b>	0	
<b>Access</b>	READ	
<b>Description</b>	SIGWNG is the system warning signal.	
	SIGWNG is continuously updated by the system. If the system has an active warning condition, SIGWNG will be one (1), otherwise it will be zero (0).	
	The active level of the output can be set with WNGLV.	
	WNGLV, OUTSG	
<b>See Also</b>	WNGLV, OUTSG	
<b>Example</b>	Command	Description
	>SIGWNG	#Query the signal status SIGWNG
	SIGWNG=0	#The current status is displayed

**SLEN: Software Position Limit Control****System Control**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	SLEN n	
<b>Range</b>	n = 0: Software position limits are disabled 1: Software position limits are enabled after homing	
<b>Initial Value</b>	0	
<b>SAVEPRM</b>	The new value takes effect immediately and the parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	<p>SLEN enables or disables software position limit action.</p> <p>When SLEN=1, software position limits LIMN and LIMP are enforced, provided the system has completed a homing action (MGHP, MGHN, HOME input or by setting PC=0). Moving outside the software position limit range will cause the motor to stop, may cause an alarm (alarm code: 67h) and may disable motor current, depending on the value of ALMACT. Stop action (soft stop or hard stop) is fixed to soft stop.</p> <p>Software limit checking is disabled while a homing operation is in process (MGHP, MGHN, HOME input).</p> <p>(A software position limit alarm may be triggered after a homing operation if PC=0 is not between LIMN and LIMP).</p> <p>For absolute or incremental index moves (MA, MI), limit checking is performed before motion starts. If the final target position is outside the range, the motion will not occur, and the action defined by ALMACT will trigger.</p> <p>For continuous motions (MCN, MCP), any out of range condition is detected only as it happens. If the system is outside the software position limits, motions may still be started. After any alarm is cleared, MI or MA can be executed if their destination would bring the motor within limits. MCN or MCP can be executed, if the motor would move in the direction of the operational range.</p>	
<b>See Also</b>	LIMP, LIMN, PC, MGHP, MGHN, ALM, ALMACT	
<b>Example</b>	Command	Description
	>ALMACT 2	#Set Alarm Action to 2 (stop, alarm, current OFF)
	ALMACT=0(2)	
	>LIMN -50	#Set negative position limit (typically inside hardware limit)
	LIMN=-50	
	>LIMP 50	#Set positive position limit (typically inside hardware limit)
	LIMP=50	
	>SLEN 1	#Enable software limit checking (after home operation)
	SLEN=1	
	>RESET	#Establish the saved parameter values
	Resetting system.	
	-----	
	CRD5xx-KP	
	CRK Series Built-in Controller	
	Software Version: A378 V.x.xx	
	Copyright 2009-2011	
	ORIENTAL MOTOR U.S.A. CORP.	
	-----	
	>MGHP	#Seek home, start in positive direction (if successful,
	>	LIMN and LIMP position limits become active)



**SLITEN: Slit Signal Input Enable****System Control**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	SLITEN n	
<b>Range</b>	n = 0: Disable 1: Enable	
<b>Initial Value</b>	0	
<b>SAVEPRM</b>	The new value takes effect immediately and the parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	Sets whether or not to concurrently use the SLIT input for homing operation.	
<b>See Also</b>	SLITLV, HOMESEL, SIG SLIT	
<b>Example</b>	Command	Description
	>LIMP 10000	#Set positive motion limit
	LIMP=10000	
	>LIMN -10000	#Set negative motion limit
	LIMN=-10000	
	>SLEN 1	#Set software limit enable
	SLEN=1	
	>HOMESEL 1	#Set Home type to 3-sensor.
	HOMESEL=1	
	> <b>SLITEN 0</b>	<b>#Set Slit sensor disable</b>
	SLITEN=0	
	>TIMEN 0	#Set TIM signal disable
	TIMEN=0	
	>ALMMSG 2	#Enable alarm messages
	ALMMSG=2	
	[Alarm+Warning]	#Start seek mechanical home
	>MGHP	#MGHP finished, check HOMEP
	>SIGHOMEP	signal
	SIGHOMEP=1	#Move continuously, positive
	>MCP	#Detected limit
	>Over travel: software position limit detected.	
	>PC	#Check PC
	PC=10000	#Just over LIMP
	>	

**SLITLV: SLIT Input Level****I/O**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	SLITLV n	
<b>Range</b>	n = 0: Normally Open 1: Normally Closed	
<b>Initial Value</b>	0	
<b>RESET</b>	RESET or recycle the power is required to activate the change. The parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	SLITLV is the active level of the Slit Sensor (SLIT) input.	
<b>See Also</b>	SIGSLIT, SLITEN	
<b>Example</b>	Command	Description
	>SLITLV=1	#Set the SLIT input logic to Normally Closed
	SLITLV=0(1)	
	>RESET	#RESET the device to initialize the modified SLIT
	Resetting system.	setting
	-----	
	CRD5xx-KP	
	CRK Series Built-in Controller	
	Software Version: A378 V.x.xx	
	Copyright 2009-2011	
	ORIENTAL MOTOR U.S.A. CORP.	
	-----	
	>SLITLV	
	SLITLV=1(1)	#New value is active
	>	

**SSTOP: Soft Stop****Motion Commands**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	SSTOP	
<b>Description</b>	SSTOP stops the motor with a controlled deceleration. The motor decelerates to start velocity VS over deceleration time TD, and then stops completely.	
<b>See Also</b>	TD, HSTOP, PSTOP, ABORT	
<b>Example</b>	Command	Description
	>TD 1.0	#Set the deceleration time to 1.0 second.
	TD=1.0	#Device response
	>VS 200	#Set the starting velocity to 200 pps
	VS=200	#Device response
	>VR 4000	#Set the running velocity to 4000 pps
	VR=4	#Device response
	>MCP	#Move continuously in the positive direction
	>SSTOP	#Slow down and stop the motor
	>DIS 10000	#Distance equals 10000
	DIS=10000	#Device response
	>MI	#Move incremental
	>SSTOP	#Slow down and stop the motor
	>	

**STARTACT: START Input Action****System Control**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	STARTACT n	
<b>Range</b>	n = 0: START input starts sequence execution when asserted. 1: START input starts sequence execution when asserted, and aborts sequence execution and motion when cleared.	
<b>Initial Value</b>	0	
<b>RESET</b>	RESET or recycle the power is required to activate the change. The parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	STARTACT determines the action associated with the dedicated START input. START can be configured to start sequences only (STARTACT=0), or to act as a toggle (STARTACT=1): starting sequences when set to its active level and aborting sequences (and motions) when set to its inactive level.	
<b>See Also</b>	<ESC>, ABORT, STARTLV	
<b>Example</b>	Command >STARTACT 1 STARTACT=0(1) >RESET Resetting system.	Description #Set the START input action to level detect #RESET to activate new settings
-----		
CRD5xx-KP CRK Series Built-in Controller Software Version: A378 V.x.xx Copyright 2009-2011 ORIENTAL MOTOR U.S.A. CORP.		
-----		
>STARTACT STARTACT=1(1) >		#Confirm new value.

**STARTLV: START Input Level****I/O**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	STARTLV n	
<b>Range</b>	n = 0: Normally Open 1: Normally Closed	
<b>Initial Value</b>	0	
<b>RESET</b>	RESET or recycle the power is required to activate the change. The parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	Sets the active level of the dedicated START input.	
<b>See Also</b>	STARTACT	
<b>Example</b>	Command	Description
	>STARTLV=1	#Set the START input logic to Normally Closed
	STARTLV=0(1)	
	>RESET	#RESET the device to initialize the modified START
	Resetting system.	setting
	-----	
	CRD5xx-KP	
	CRK Series Built-in Controller	
	Software Version: A378 V.x.xx	
	Copyright 2009-2011	
	ORIENTAL MOTOR U.S.A. CORP.	
	-----	
	>STARTLV	
	STARTLV=1(1)	#New value is active
	>	

**STOACT: Step Out Detection Action****System Control**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	STOACT n	
<b>Range</b>	n = 0: No operation 1: Warning 2: Alarm	
<b>Initial Value</b>	0	
<b>SAVEPRM</b>	RESET or recycle the power is required to activate the change. The parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	Sets the operation to be performed when the deviation between the command position and encoder counter value reaches the stepout detection band.	
<b>See Also</b>	EGA, EGB, STOEN, STOB, SIGSTO, OUTSTO, STOLV	
<b>Example</b>	Command	Description
	>STOACT 2	#Set for the alarm action
	STOACT=0(2)	
	>STOB 3.6	#Set the Step Out Detection Band to 3.6
	STOB=3.6	degree
	>STOEN 1	
	STOEN=0(1)	#Set the Step Out Detection enable
	>RESET	#RESET the device to initialize the
	Resetting system.	modified STOEN setting
	-----	
	CRD5xx-KP	
	CRK Series Built-in Controller	
	Software Version: A378 V.x.xx	
	Copyright 2009-2011	
	ORIENTAL MOTOR U.S.A. CORP.	
	-----	
	>STOACT	
	STOACT=2(2)	#New value is active
	>	

**STOB: Step Out Detection Band****System Control**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	STOB n	
<b>Range</b>	n = 0.1 to 360.0 deg	
<b>Initial Value</b>	7.2	
<b>SAVEPRM</b>	The new value takes effect immediately and the parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	Sets the judgment condition for misstep detection using the deviation (angle) between the command position and encoder counter value.	
<b>See Also</b>	EGA, EGB, STOEN, STOACT, SIGSTO, OUTSTO, STOLV	
<b>Note</b>	Initial Value is a reference value designed for a standard resolution type of a 5-phase stepping motor which has a basic step angle of 0.72 degree as a reference value. For high resolution type which has a basic step angle of 0.36 degree, the Initial Value as a reference value should be divided by two.	
<b>Example</b>	Command	Description
	>STOACT 2 STOACT=0(2) >STOB 3.6 STOB=3.6 >STOEN 1 STOEN=0(1) >RESET Resetting system.	#Set for the alarm action  #Set the Step Out Detection Band to 3.6 degree  #Set the Step Out Detection enable  #RESET the device to initialize the modified STOEN setting
-----		
CRD5xx-KP CRK Series Built-in Controller Software Version: A378 V.x.xx Copyright 2009-2011 ORIENTAL MOTOR U.S.A. CORP.		
-----		
>STOEN STOEN=1(1) >		#New value is active

**STOEN: Step Out Detection Enable****System Control**

<b>Execution Mode</b>	Immediate
<b>Syntax</b>	STOEN n
<b>Range</b>	n = 0: Disable 1: Enable
<b>Initial Value</b>	0
<b>RESET</b>	RESET or recycle the power is required to activate the change. The parameter is saved automatically.
<b>Access</b>	READ and WRITE
<b>Description</b>	Sets whether to enable or disable the misstep detection function.
<b>Note</b>	To enable Step Out Detection function, an encoder is required.
<b>See Also</b>	EGA, EGB, STOB, STOACT, SIGSTO, OUTSTO, STOLV
<b>Example</b>	<div> <div>Command</div> <div>Description</div> </div> <div> <div>&gt;STOACT 2</div> <div>#Set for the alarm action</div> </div> <div> <div>STOACT=0(2)</div> <div></div> </div> <div> <div>&gt;STOB 3.6</div> <div>#Set the Step Out Detection Band to 3.6</div> </div> <div> <div>STOB=3.6</div> <div>degree</div> </div> <div> <div>&gt;STOEN 1</div> <div>#Set the Step Out Detection enable</div> </div> <div> <div>STOEN=0(1)</div> <div>#RESET the device to initialize the</div> </div> <div> <div>&gt;RESET</div> <div>modified STOEN setting</div> </div> <div> <div>Resetting system.</div> <div></div> </div> <div> <div>-----</div> <div></div> </div> <div> <div>CRD5xx-KP</div> <div></div> </div> <div> <div>CRK Series Built-in Controller</div> <div></div> </div> <div> <div>Software Version: A378 V.x.xx</div> <div></div> </div> <div> <div>Copyright 2009-2011</div> <div></div> </div> <div> <div>ORIENTAL MOTOR U.S.A. CORP.</div> <div></div> </div> <div> <div>-----</div> <div></div> </div> <div> <div>&gt;STOEN</div> <div>#New value is active</div> </div> <div> <div>STOEN=1(1)</div> <div></div> </div> <div> <div>&gt;</div> <div></div> </div>



**STOLV: STO Output Level****I/O**

<b>Execution Mode</b>	Immediate
<b>Syntax</b>	STOLV n
<b>Range</b>	n = 0: Normally Open 1: Normally Closed
<b>Initial Value</b>	0
<b>RESET</b>	RESET or recycle the power is required to activate the change. The parameter is saved automatically.
<b>Access</b>	READ and WRITE
<b>Description</b>	Sets the active level of the dedicated STO output, if used.
<b>See Also</b>	EGA, EGB, SIGSTO, STOEN, STOB, STOACT, OUTSTO
<b>Example</b>	<div> <div>Command</div> <div>Description</div> </div> <div> <div>&gt;STOLV=1</div> <div>#Set the STO output logic to Normally Closed</div> </div> <div> <div>STOLV=0(1)</div> <div></div> </div> <div> <div>&gt;RESET</div> <div>#RESET the device to initialize the modified</div> </div> <div> <div>Resetting system.</div> <div>STOLV setting</div> </div> <div>-----</div> <div> <div>CRD5xx-KP</div> <div>CRK Series Built-in Controller</div> <div>Software Version: A378 V.x.xx</div> <div>Copyright 2009-2011</div> <div>ORIENTAL MOTOR U.S.A. CORP.</div> </div> <div>-----</div> <div> <div>&gt;STOLV</div> <div></div> </div> <div> <div>STOLV=1(1)</div> <div>#New value is active</div> </div> <div> <div>&gt;</div> <div></div> </div>

**STRSW: Current State at System Start****System Control**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	STRSW n	
<b>Range</b>	n = 0: Motor current OFF at system start 1: Motor current ON at system start	
<b>Initial Value</b>	1	
<b>SAVEPRM</b>	The new value takes effect immediately and the parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	<p>STRSW enables or disables motor current immediately after system start.</p> <p>If STRSW=0, no current is supplied to the motor windings after system start (initial value of CURRENT is 0). The motor freewheels. Motor current must be explicitly enabled (by setting CURRENT to 1) to develop holding torque and permit motions.</p> <p>If STRSW=1, the system supplies current to the motor after a successful startup (current level determined by CRSTOP).</p>	
<b>See Also</b>	CURRENT, CRRUN, CRSTOP	
<b>Example</b>	<p>Command</p> <p>&gt;STRSW 0</p> <p>STRSW=0 [Current OFF at start up]</p> <p>&gt;RESET</p> <p>Resetting system.</p> <p>-----</p> <p>CRD5xx-KP</p> <p>CRK Series Built-in Controller</p> <p>Software Version: A378 V.x.xx</p> <p>Copyright 2009-2011</p> <p>ORIENTAL MOTOR U.S.A. CORP.</p> <p>-----</p> <p>&gt;CURRENT</p> <p>CURRENT=0</p> <p>&gt;</p>	<p>Description</p> <p>#Configure for CURRENT=0 at start up</p> <p>#RESET to see CURRENT status at start up</p> <p>#CURRENT=0 after restart</p>

**TA: Acceleration Time****Motion Variables**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	TA n	
<b>Range</b>	n = 0.001 to 1000.000 (second)	
<b>Initial Value</b>	0.500	
<b>SAVEPRM</b>	The new value takes effect immediately. However, SAVEPRM is required to save the parameter values in nonvolatile memory. Otherwise, the parameter value is reset to the last saved value at device power up. If no new parameter value was saved, then the value is set to the default (initial) value.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	<p>TA is the time used to accelerate the motor (increase velocity away from zero)</p> <p>TA affects the initial ramp time for:</p> <ul style="list-style-type: none"> <li>- MA (Move Absolute)</li> <li>- MI (Move Incremental)</li> <li>- MCN and MCP (Move Continuously, negative and positive)</li> <li>- MGHN and MGHP (Seek home, start negative and positive)</li> </ul> <p>TA also affects the time required to change speeds, when speeds are increasing (in an absolute sense), for the following motion types:</p> <ul style="list-style-type: none"> <li>- CV (Change Velocity)</li> <li>- MCN and MCP (Move Continuously, negative and positive)</li> <li>- MIx (Linked index)</li> </ul> <p>If speeds are decreasing (toward zero), deceleration time TD determines ramp time.</p>	
<b>See Also</b>	CV, MA, MCN, MCP, MGHN, MGHP, MI, MIx, TD	
<b>Example</b>	Command	Description
	>VS 500	#Set Starting speed VS to 500 pps
	VS=500	
	>VR 2000	#Set Running speed VR to 2000 pps
	VR=2000	
	>DIS 5000	#Set Distance to 5000
	DIS=5000	
	>TA 0.5	#Set acceleration time to 0.5 seconds
	TA=0.5	
	>TD 0.5	#Set deceleration time to 0.5 seconds
	TD=0.5	
	>MI	#Execute index motion
	>	

**TD: Deceleration Time****Motion Variables**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	TD n	
<b>Range</b>	n = 0.001 to 1000.000 (second)	
<b>Initial Value</b>	0.500	
<b>SAVEPRM</b>	The new value takes effect immediately. However, SAVEPRM is required to save the parameter values in nonvolatile memory. Otherwise, the parameter value is reset to the last saved value at device power up. If no new parameter value was saved, then the value is set to the default (initial) value.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	<p>TD is the time used to decelerate the motor (decrease velocity toward zero).            TD affects the final ramp time for:</p> <ul style="list-style-type: none"> <li>- MA (Move Absolute)</li> <li>- MI (Move Incremental)</li> <li>- MCN and MCP (Move Continuously, negative and positive)</li> <li>- MGHN and MGHP (Seek home, start negative and positive)</li> <li>- SSTOP (Soft Stop)</li> <li>- ABORT (Abort sequences and motions)</li> <li>- &lt;ESC&gt; (ESCAPE character: equivalent to ABORT)</li> </ul> <p>TD also affects the time required to change speeds, when speeds are decreasing (in an absolute sense),            for the following motion types:</p> <ul style="list-style-type: none"> <li>- CV (Change Velocity)</li> <li>- MCN and MCP (Move Continuously, negative and positive)</li> <li>- MIx (Linked index)</li> </ul> <p>If speeds are increasing (away from zero), acceleration time TA determines ramp time.</p>	
<b>See Also</b>	CV, MA, MCN, MCP, MGHN, MGHP, MI, MIx, TA	
<b>Example</b>	Command	Description
	>VS 500	#Set Starting speed VS to 500 pps
	VS=500	
	>VR 2000	#Set Running speed VR to 2000 pps
	VR=2000	
	>DIS 5000	#Set Distance to 5000
	DIS=5000	
	>TA 0.5	#Set acceleration time to 0.5 seconds
	TA=0.5	
	>TD 0.5	#Set deceleration time to 0.5 seconds
	TD=0.5	
	>MI	#Execute index motion
	>	

**TEACH: Teach Positions****Motor Commands**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	TEACH	
<b>Description</b>	<p>TEACH starts a utility process to find and store target positions into the position data array (POS [x]). While the TEACH process runs, the motor can be moved until an intended target position is reached, and then that position value can be stored in the POS [x] array. The motor can move actively, using menu keys to move continuously or by small increments. The motor can also be externally positioned after toggling current OFF.</p> <p>The POS [x] array data can be used as the target destination for absolute motions (MA). In sequences, POS [x] can be used anywhere a variable is permitted.</p> <p>For a full explanation of the TEACH utility, refer to another section “Teaching Positions”.</p>	
<b>See Also</b>	POS [x]	
<b>Example</b>	Command	Description
	>TEACH	#Start the TEACH process
	*** Teach mode ***	
	<p>(V) : Move Cont. Neg. (M) : Move Cont. Pos.</p> <p>(B) : Move Incr. -1 (N) : Move Incr. +1</p> <p>(Q) : Toggle current ON/OFF (Encoder required)</p> <p>(K) : Change Key Interval (50-500[msec])</p> <p>(U) : Update display position -----</p> <p>&lt;Space&gt; : Immediate Stop</p> <p>&lt;Enter&gt; : Data entry mode (Input POS number, then &lt;Enter&gt;)</p> <p>&lt;ESC&gt; : Exit teach mode</p>	
	PC= 0	

**TEMPLV: TEMP Output Level****I/O**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	TEMPLV n	
<b>Range</b>	n = 0: Normally Open 1: Normally Closed	
<b>Initial Value</b>	0	
<b>RESET</b>	RESET or recycle the power is required to activate the change. The parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	TEMPLV is the active level of the Temperature Warning (TEMP) output, if used.	
<b>See Also</b>	OUTTEMP, SIGTEMP	
<b>Example</b>	Command	Description
	>TEMPLV=1	#Set the TEMP output logic to Normally Closed
	TEMPLV=0(1)	
	>RESET	#RESET the device to initialize the modified TEMP
	Resetting system.	setting
-----		
	CRD5xx-KP	
	CRK Series Built-in Controller	
	Software Version: A378 V.x.xx	
	Copyright 2009-2011	
	ORIENTAL MOTOR U.S.A. CORP.	
-----		
	>TEMPLV	
	TEMPLV=1(1)	#New value is active
	>	

**TIMEN: Timing Signal Enable****System Control****Execution Mode** Immediate and Sequence**Syntax** TIMEN n

**Range** n = 0: Disable  
 1: Enable internal TIM signal  
 2: Enable Index signal

**Initial Value** 0**SAVEPRM** The new value takes effect immediately and the parameter is saved automatically.**Access** READ and WRITE**Description** Sets whether or not to concurrently use the TIM signal for homing operation.**See Also** HOMESSEL

Example	Command	Description
	>LIMP 10000	#Set positive motion limit
	LIMP=10000	
	>LIMN -10000	#Set negative motion limit
	LIMN=-10000	
	>SLEN 1	#Set software limit enable
	SLEN=1	
	>HOMESSEL 1	#Set Home type to 3-sensor.
	HOMESSEL=1	
	>SLITEN 0	#Set Slit sensor disable
	SLITEN=0	
	>TIMEN 0	#Set TIM and Index signal disable
	TIMEN=0	
	>ALMMSG 2	#Enable alarm messages
	ALMMSG=2 [Alarm+Warning]	
	>MGHP	#Start seek mechanical home
	>SIGHOME P	#MGHP finished, check HOME P
	SIGHOME P=1	signal
	>MCP	#Move continuously, positive
	>Over travel: software position limit detected.	#Detected limit
	>PC	#Check PC
	PC=10000	#Just over LIMP
	>	

**TIMER: Running Timer****System Status**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	TIMER n	
<b>Range</b>	n = 0 to 500000.000 (second)	
<b>Initial Value</b>	0	
<b>Access</b>	READ and WRITE	
<b>Description</b>	<p>TIMER is a running timer, counting seconds.</p> <p>TIMER is set to zero (0.000) at system start, and counts up from that time, with millisecond resolution.</p> <p>TIMER overflows at 500,000 seconds (about 5.8 days), and is restarted from zero.</p> <p>TIMER can be set to any value within its range, for synchronization.</p> <p>Since user variables can only be set as integer numbers, when TIMER is to be set to a user variable, the user variable should be set to 1000 times bigger than the desired value of TIMER and the unit is changed to [msec] from [sec]. For instance, if TIMER is 0.025 [sec] and is set to a user variable, the user variable will be set to 25 [msec].</p>	
<b>See Also</b>	ALM, WAIT	
<b>Example</b>	Command	Description
	>LIST WATCH	#List sequence WATCH
	( 1) T=TIMER+60000	#Set T to be 60 seconds greater than timer
	( 2) WHILE (TIMER<T)	#While TIMER < T (true for about 1 minute)
	( 3) IF (IN2=1)	#If input 2 is asserted
	( 4) ALMSET	#Set an alarm
	( 5) ENDIF	#End IF block
	( 6) WEND	#End WHILE block
	>	



**TRACE: Sequence Trace Control****Monitor Commands**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	TRACE n	
<b>Range</b>	n = 0: Trace is disabled 1: Trace is enabled	
<b>Initial Value</b>	0	
<b>Access</b>	READ and WRITE	
<b>Description</b>	TRACE enables or disables tracing of sequence statements. When sequence tracing is enabled (TRACE=1), sequence statements are displayed as they are executed, one statement at time, surrounded by "curly braces" { and }.	
<b>See Also</b>	RUN, ABORT, LIST	
<b>Note</b>	Enabling sequence tracing alters sequence timing, because of the time required to transmit the trace information. Sequences execute slower when TRACE=1.	
<b>Example</b>	Command	Description
	>LIST TOGGLEATVR	#List sequence TOGGLEATVR
	( 1) LOOP 3	#List output...
	( 2) MI	
	( 3) WHILE (VC!=VR); WEND	
	( 4) OUT4=1-OUT4	
	( 5) MEND	
	( 6) ENDL	
	>TRACE 1	#Enable Tracing
	>RUN TOGGLEATVR	#Run sequence TOGGLEATVR
	>{ LOOP 3 }	#First executing statement, surrounded by { }
	{ MI }	#Next statement
	{ WHILE (VC!=VR) }	#Next statement: <b>Note</b> NOT the entire line
	{ WEND }	#End WHILE block...
	{ WHILE (VC!=VR) }	#...back to WHILE statement
	{ OUT4=1-OUT4 }	#WHILE test failed, proceed beyond WEND
	{ MEND }	#Wait for motion end
	{ ENDL }	#End LOOP block, back to top-of-loop
	{ MI }	#Actual to-of-loop is first statement within loop
	{ WHILE (VC!=VR) }	#Repeat...
	{ WEND }	
	{ WHILE (VC!=VR) }	
	{ OUT4=1-OUT4 }	
	{ MEND }	
	{ ENDL }	
	{ MI }	
	{ WHILE (VC!=VR) }	
	{ WEND }	
	{ WHILE (VC!=VR) }	
	{ OUT4=1-OUT4 }	
	{ MEND }	
	{ ENDL }	#Loop count exhausted, sequence is finished.

**UNLOCK: Unlock Sequence****Sequence Management**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	UNLOCK [target]	
<b>Range</b>	target can be the name or number of any existing sequence	
<b>Description</b>	<p>UNLOCK unlocks a sequence that has been previously locked with the LOCK command.</p> <p>A locked sequence cannot be deleted, renamed, or overwritten (by COPY or EDIT).</p> <p>The sequence directory listing (DIR command) shows the lock status for all sequences.</p>	
<b>See Also</b>	DIR, EDIT, LOCK	
<b>Example</b>	Command	Description
	>DEL REGISTER	#Delete sequence REGISTER
	Error: Sequence is locked.	#Can't: sequence is locked
	>UNLOCK REGISTER	#Unlock sequence REGISTER
	>DEL REGISTER	#Delete sequence REGISTER
	Enter Y to proceed, other key to cancel. Y	#OK now. Confirm.
	>	

**VC: Velocity Command****System Status**

<b>Execution Mode</b>	Immediate and Sequence
<b>Syntax</b>	VC
<b>Range</b>	-500,000 to +500,000
<b>Initial Value</b>	0
<b>Access</b>	READ
<b>Description</b>	<p>VC is the instantaneous velocity command, or set-point.</p> <p>This value is controlled by the system's motion profiler in internal profiler mode. The sign reflects the motion direction.</p> <p>VC reflects the velocity that the system is supposed to be running at. The actual shaft velocity may vary from VC.</p>

**See Also****Example**

Command	Description
>VR	#Check target running velocity VR
VR=4000	#4000 pps
>TA	#Check acceleration time TA
TA=20	#20 seconds
>MCP	#Start moving continuously, positive direction
>VC	#Check commanded velocity, while accelerating
VC=892	#Slowly increasing toward VR
>VC	
VC=1614	
>VC	
VC=2293	
>VC	
VC=3007	
>VC	
VC=3721	
>	

**VER: Display Firmware Version****Monitor Commands****Execution Mode** Immediate**Syntax** VER**Description** VER displays the system's firmware version information.**Example** Command

Description

&gt;VER

#Display the firmware version

A378 V.1.03 / Date Oct.20.2010

#Typical response

&gt;

**VERBOSE: Command Response Control****Communications**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	VERBOSE n	
<b>Range</b>	n = 0: Respond with data only 1: Respond with data and descriptive text	
<b>Initial Value</b>	1	
<b>SAVEPRM</b>	The new value takes effect immediately and the parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	<p>VERBOSE controls the amount of information that the system transmits in response to commands. When VERBOSE=1 (the default), extra information is transmitted to establish the context of the response. VERBOSE=1 is preferred for human communications.</p> <p>When VERBOSE=0, the extra information is suppressed. Fewer characters are transmitted, reducing the amount of time required to communicate, and reducing the amount of data to be interpreted. VERBOSE=0 is preferred if an intelligent host machine will be automatically controlling the system via the serial port.</p> <p>The examples below show the differences in several responses.</p>	
<b>See Also</b>	ECHO	
<b>Example</b>	Command	Description
	>VERBOSE	#Check VERBOSE setting
	VERBOSE=1	#VERBOSE=1: extra text
	>PC	#Check position set point
	PC=150	#Response includes "PC=", value
	>VR	#Check running velocity
	VR=10	#Response includes "VR="
	>ALMMSG	#Check ALMMSG setting
	ALMMSG=2[Alarm+Warning]	#Response includes "ALMMSG=", value, and explanation
	>VERBOSE 0	#Set VERBOSE=0 (suppress extra text)
	0	#Immediately effective. Only new value returned
	>PC	#Check position set point
	150	#Only value returned
	>VR	#Check running velocity
	10	#Only value returned
	>ALMMSG	#Check ALMMSG
	2	#Only value returned
	>	

**VIEW: View Parameter****Communications**

<b>Execution Mode</b>	Sequence	
<b>Syntax</b>	VIEW element	
<b>Range</b>	'element' can be the name of any parameter or variable available in sequences.	
<b>Description</b>	<p>VIEW transmits the value of a parameter or variable without any extra characters.</p> <p>When a value is transmitted in response to a simple query (using just the parameter or variable name), the system transmits the numeric value, followed by a carriage return, a linefeed, and a new prompt.</p> <p>The VIEW command only transmits the numeric value, permitting tighter control of the response.</p>	
<b>See Also</b>	KB, KBQ, SACS, SAS, VERBOSE	
<b>Example</b>	Command	Description
	>LIST SAYPOS	#List sequence SAYPOS
	( 1) SAS POSITION:	#Send ASCII string "POSITION:", + CR + LF + prompt
	( 2) PF	#Display value of actual position, + CR + LF + prompt
	( 3) SACS	#Send ASCII string "POSITION:" with trailing space
	POSITION: ^	
	( 4) <b>VIEW PF</b>	<b>#Display value of actual position: no extra text</b>
	>RUN SAYPOS	#Run sequence SAYPOS
	>POSITION:	#SAS: results in new line, new prompt
	>14655	#First PF: results in new line, new prompt
	>POSITION: 14655	#SACS output, VIEW output: no new line, no new prompt
<b>Note</b>	<p>In a multi-drop configuration, all output from sequence commands is suppressed unless the device has been previously addressed (via @). The KB and KBQ commands will not receive input unless the device has been previously addressed.</p>	

**VR: Running Velocity****Motion Variables**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	VR n	
<b>Range</b>	n = 1 to 500,000 pps	
<b>Initial Value</b>	1000	
<b>SAVEPRM</b>	The new value takes effect immediately. However, SAVEPRM is required to save the parameter values in nonvolatile memory. Otherwise, the parameter value is reset to the last saved value at device power up. If no new parameter value was saved, then the value is set to the default (initial) value.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	VR is the running velocity for motions. VR specifies the peak target speed for the motion. VR is always positive: the direction for the motion is determined by start vs. end positions (for point to point motions), or by choice of positive or negative motion command (MCN vs. MCP).	
<b>See Also</b>	CV, MA, MCN, MCP, MI,	
<b>Important Interactions</b>	The Change Velocity (CV) command overwrites VR with the value designated in the CV command.	
<b>Example</b>	Command	Description
	>VS 500	#Set Starting speed VS to 500 pps
	VS=500	
	>VR 2000	#Set Running speed VR to 2000 pps
	VR=2000	
	>DIS 5000	#Set Distance to 5000
	DIS=5000	
	>TA 0.5	#Set acceleration time to 0.5 seconds
	TA=0.5	
	>TD 0.5	#Set deceleration time to 0.5 seconds
	TD=0.5	
	>MI	#Execute index motion
	>	

**VRx: Linked Motion Running Velocity****Motion Variables**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	VRx n	
<b>Range</b>	x = 0 to 3: Linked motion segment	
	n = 1 to 500,000 pps	
<b>Initial Value</b>	1000	
<b>SAVEPRM</b>	The new value takes effect immediately. However, SAVEPRM is required to save the parameter values in nonvolatile memory. Otherwise, the parameter value is reset to the last saved value at device power up. If no new parameter value was saved, then the value is set to the default (initial) value.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	VRx is the running velocity for linked motion segment 'x'. VRx specifies the peak target speed for the segment, in user units per second.	
	VRx is always positive: the direction for the motion segment is determined by the start and end positions for the entire linked index.	
<b>See Also</b>	INCABSx, Mlx, LINKx	
<b>Example</b>	Command	Description
	>VR1 500	#Set the velocity for linked move #1 to 500 pps
	VR1=500	#Device response
	>DIS1 2000	#Set the distance for linked move #1 to 2000
	DIS1=2000	#Device response
	>INCABS1 1	#Set the move type for linked motion #1 to incremental
	INCABS1=1 [INC]	#Device response
	>LINK1 1	#Enable the linked operation for motion #1
	LINK1=1	#Device response
	>VR2 1000	#Linked move #2 velocity equals 1000 pps
	VR2=1000	#Device response
	>INCABS2 1	#Set the move type for linked motion #2 to incremental
	INCABS2=1 [INC]	#Device response
	>DIS2 4000	#Linked move #2: destination is position 4000
	DIS2=4000	#Device response
	>LINK2 0	#"Unlink" link2 from link3
	LINK2=0	#Device response
	>MI1	#Start the linked operation motion
	>	



**VS: Starting Velocity****Motion Variables**

<b>Execution Mode</b>	Immediate and Sequence	
<b>Syntax</b>	VS n	
<b>Range</b>	n = 1 to 500,000 pps	
<b>Initial Value</b>	100	
<b>SAVEPRM</b>	The new value takes effect immediately. However, SAVEPRM is required to save the parameter values in nonvolatile memory. Otherwise, the parameter value is reset to the last saved value at device power up. If no new parameter value was saved, then the value is set to the default (initial) value.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	<p>VS is the starting velocity for motions.</p> <p>All motions start with velocity VS and then accelerate to VR over acceleration time TA. All motions decelerate from VR to VS over deceleration time, TD, then stop.</p> <p>Speed changes between zero (0) speed and VS is instantaneous. (Note that this is a velocity command, and not actual motor velocity: the motor cannot physically change speeds instantaneously). The sudden change in speed may or may not be desirable. In applications with high static friction, VS may help the system start or finish motions better. VS might also be used to avoid any very low resonant speed.</p>	
<b>See Also</b>	MA, MCN, MCP, MI, MIx	
<b>Example</b>	Command	Description
	>VS 500	#Set Starting speed VS to 500 pps
	VS=500	
	>VR 2000	#Set Running speed VR to 2000 pps
	VR=2000	
	>DIS 5000	#Set Distance to 5000
	DIS=5000	
	>TA 0.5	#Set acceleration time to 0.5 seconds
	TA=0.5	
	>TD 0.5	#Set deceleration time to 0.5 seconds
	TD=0.5	
	>MI	#Execute index motion
	>	

**WAIT: Wait for Specified Time****Sequence Commands**

<b>Execution Mode</b>	Sequence	
<b>Syntax</b>	WAIT n	
<b>Range</b>	n = 0.001 to 500,000.000 (second)	
<b>Description</b>	WAIT causes sequence execution to wait for the indicated time, before proceeding to the next statement.	
<b>See Also</b>	KB, KBQ, TIMER, MEND	
<b>Example</b>	Command	Description
	>LIST TENTIMES	#List sequence TENTIMES
	( 1) MA 0	#Start absolute motion, to position 0
	( 2) MEND	#Wait for motion to finish
	( 3) OUT4 1	#Turn output 4 on
	( 4) WAIT 3.0	#Wait 3 seconds before proceeding
	( 5) OUT4 0	#Turn output 4 OFF
	( 6) LOOP 10	#Loop: execute contents 10 times
	( 7) DIS 1000	
	( 8) MI	#Start incremental motion (distance DIS)
	( 9) MEND	#Wait for motion to finish
	(10) OUT4 1	#Turn output 4 on
	(11) WAIT Q	#Wait before proceeding, wait time in variable Q
	(12) OUT4 0	#Turn output 4 OFF
	(13) ENDL	#End of LOOP block
	>Q 500	#Unit is (msec) with user variables
	Q=5	
	>RUN TENTIMES	

**WEND: End of WHILE Block****Sequence Commands**

<b>Execution Mode</b>	Sequence	
<b>Syntax</b>	WEND	
<b>Description</b>	<p>WEND terminates the innermost WHILE block in a sequence.</p> <p>Processing returns to the WHILE which started the block, for re-evaluation. If the WHILE condition fails, processing continues with the statement following the WEND statement.</p>	
<b>See Also</b>	ENDIF, ENDL, IF, LOOP, WHILE, BREAKW	
<b>Example</b>	Command	Description
	>LIST CHKJAM	#List sequence CHKJAM
	( 1) DIS=10; VR=10	#Set motion parameter
	( 2) LOOP	#Start infinite loop
	( 3) MI	#Start move incremental
	( 4) WHILE (DTMP<70)	#Check if over heated
	( 5) IF (SIGMOVE=0)	#Check for motion end
	( 6) BREAKW	#Break out of while loop, if so
	( 7) ENDIF	
	( 8) <b>WEND</b>	<b>#End while loop</b>
	( 9) IF (SIGMOVE!=0)	#Check if moving
	( 10) PAUSE	#DTMP>70: PAUSE motion
	( 11) WAIT TD	#Wait for stop, send text, get response
	( 12) SAS System in trouble.	
	( 13) SACS Enter 1 to continue, other to stop:	
	( 14) A=KBQ; SACS ^M^J>	
	( 15) IF (A=1)	
	( 16) CONT; MEND	#CONTinue, if A=1
	( 17) ELSE	#Otherwise, report stopped
	( 18) SAS Operation stopped.	
	( 19) RET	#Return from sequence
	( 20) ENDIF	
	( 21) ENDIF	
	( 22) SAS Motion end, goto next.	
	( 23) WAIT 1	#Send normal message
	( 24) ENDL	#Dwell 1 second, loop back to top.
	>RUN CHKJAM	#Execute sequence CHKJAM
	>Motion end, goto next.	#Normal message
	>Motion end, goto next.	#Normal message
	>System in trouble.	#Driver is getting hot
	>Enter 1 to continue, other to stop:1	#Prompt message -> Entry "1"
	>Motion end, goto next.	#Normal message
	>Motion end, goto next.	#Normal message
	>System in trouble.	#Driver is getting hot
	>Enter 1 to continue, other to stop:2	#Prompt message -> Entry "2"
	>Operation stopped.	#Finished message (Sequence finished)
	>	

**WHILE: Begin WHILE Block: execute while true****Sequence Commands**

<b>Execution Mode</b>	Sequence																																						
<b>Syntax</b>	WHILE (element1 {Conditional Operator} element2)																																						
<b>Description</b>	<p>WHILE begins a conditional iterative block.</p> <p>Statements between the opening WHILE statement and the closing WEND statement execute while the conditional expression is true.</p> <p>Parenthesis are required.</p> <p>element1 and element2 may be any numeric variable available to sequences, or any numeric constant within the range -(Maximum Number) to +(Maximum Number).</p> <p>Valid conditional operators are:</p> <ul style="list-style-type: none"> <li>• = : Equal to</li> <li>• != : Not equal to</li> <li>• &lt; : Less than</li> <li>• &lt;= : Less than or equal to</li> <li>• &gt; : Greater than</li> <li>• &gt;= : Greater than or equal to</li> </ul> <p>WHILE statements must be followed (at some point) by a corresponding WEND statement, forming a WHILE "block". BREAKW statements may appear within the WHILE block, terminating iteration and breaking out of the block.</p> <p>When executed, the conditional expression is evaluated. If it evaluates to TRUE, sequence processing proceeds to the statement following the WHILE. If it evaluates to FALSE, sequence processing proceeds to the statement following the closing WEND statement.</p> <p>The conditional expression is evaluated at the beginning of the block only, once per iteration. If the expression evaluates to TRUE when the WHILE statement executes, the contents of the WHILE block will be executed. The expression will be reevaluated at the next iteration: it is not tested during execution of the enclosed block statements.</p> <p>Block structures (IF-ENDIF, WHILE-WEND, LOOP-ENDL) may be nested, to six (6) levels deep.</p>																																						
<b>See Also</b>	IF, LOOP, WEND, BREAKW																																						
<b>Note</b>	Be careful when using with MCP/MCN command. Refer to "10-3. Continuous Motions".																																						
<b>Example</b>	<table> <tr> <th>Command</th><th>Description</th></tr> <tr> <td>&gt;LIST RUNTIMEOUT</td><td>#List sequence RUNTIMEOUT</td></tr> <tr> <td>( 1) IF (IN6=1)</td><td>#If Input 6 is asserted...</td></tr> <tr> <td>( 2) MCN</td><td>#Start moving continuously, negative direction</td></tr> <tr> <td>( 3) ELSE</td><td>#Otherwise...</td></tr> <tr> <td>( 4) MCP</td><td>#Start moving continuously, positive direction</td></tr> <tr> <td>( 5) ENDIF</td><td>#End of IF block</td></tr> <tr> <td>( 6) TIMER=0</td><td>#RESET running TIMER to 0</td></tr> <tr> <td>( 7) WHILE (IN5=1)</td><td>#Begin WHILE block: execute while Input 5 is asserted</td></tr> <tr> <td>( 8) IF (SIGTEMP=1)</td><td>#If temperature warning</td></tr> <tr> <td>( 9) ALMSET</td><td>#...set alarm (will automatically abort sequence and motion)</td></tr> <tr> <td>(10) ENDIF</td><td>#End of IF block</td></tr> <tr> <td>(11) IF (TIMER &gt; 5000)</td><td>#If TIMER greater than 5 seconds...</td></tr> <tr> <td>(12) BREAKW</td><td>#BREAK out of WHILE block: next statement follows WEND</td></tr> <tr> <td>(13) ENDIF</td><td>#End of IF block</td></tr> <tr> <td>(14) WEND</td><td>#End of WHILE block: back to WHILE and reevaluate</td></tr> <tr> <td>(15) SSTOP</td><td>#Start Soft Stop</td></tr> <tr> <td>(16) MEND</td><td>#Wait for motion to end</td></tr> <tr> <td>&gt;</td><td></td></tr> </table>	Command	Description	>LIST RUNTIMEOUT	#List sequence RUNTIMEOUT	( 1) IF (IN6=1)	#If Input 6 is asserted...	( 2) MCN	#Start moving continuously, negative direction	( 3) ELSE	#Otherwise...	( 4) MCP	#Start moving continuously, positive direction	( 5) ENDIF	#End of IF block	( 6) TIMER=0	#RESET running TIMER to 0	( 7) WHILE (IN5=1)	#Begin WHILE block: execute while Input 5 is asserted	( 8) IF (SIGTEMP=1)	#If temperature warning	( 9) ALMSET	#...set alarm (will automatically abort sequence and motion)	(10) ENDIF	#End of IF block	(11) IF (TIMER > 5000)	#If TIMER greater than 5 seconds...	(12) BREAKW	#BREAK out of WHILE block: next statement follows WEND	(13) ENDIF	#End of IF block	(14) WEND	#End of WHILE block: back to WHILE and reevaluate	(15) SSTOP	#Start Soft Stop	(16) MEND	#Wait for motion to end	>	
Command	Description																																						
>LIST RUNTIMEOUT	#List sequence RUNTIMEOUT																																						
( 1) IF (IN6=1)	#If Input 6 is asserted...																																						
( 2) MCN	#Start moving continuously, negative direction																																						
( 3) ELSE	#Otherwise...																																						
( 4) MCP	#Start moving continuously, positive direction																																						
( 5) ENDIF	#End of IF block																																						
( 6) TIMER=0	#RESET running TIMER to 0																																						
( 7) WHILE (IN5=1)	#Begin WHILE block: execute while Input 5 is asserted																																						
( 8) IF (SIGTEMP=1)	#If temperature warning																																						
( 9) ALMSET	#...set alarm (will automatically abort sequence and motion)																																						
(10) ENDIF	#End of IF block																																						
(11) IF (TIMER > 5000)	#If TIMER greater than 5 seconds...																																						
(12) BREAKW	#BREAK out of WHILE block: next statement follows WEND																																						
(13) ENDIF	#End of IF block																																						
(14) WEND	#End of WHILE block: back to WHILE and reevaluate																																						
(15) SSTOP	#Start Soft Stop																																						
(16) MEND	#Wait for motion to end																																						
>																																							

**WNG; Warning Status and History****Monitor Commands**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	WNG	
<b>Range</b>	00 to F2	
<b>Initial Value</b>	00	
<b>Access</b>	READ	
<b>Description</b>	The WNG command displays the current alarm code, history of the last 10 warning issues, a brief alarm code Description, and the elapsed time for the latest alarm code and warning message. See Chapter 14 "Troubleshooting", on page 288 for a list of all ALARM codes and causes. The Warning history is not saved in EEPROM.	
<b>See Also</b>	ALMLV, ALMACT, ALMCLR, ALMMSG, ALMSET, CURRENT	
<b>Example</b>	Command	Description
	>WNG	#Query the current WARNING code
	WNG =00 , RECORD : 22 23 9A 23 68 68 66 60 66 66	
	>	

**WNGCLR: Clear Warning****System Control****Execution Mode** Immediate**Syntax** WNGCLR**Description** The WNGCLR command attempts to clear the system warning status.**See Also** WNG, WNGLV**Example** Command

Description

&gt;WNG

#Query WNG

WNG =68 , RECORD : 68 68 66 60 66 66 60 68 66 66

&gt;WNGCLR

#Clear the warning condition, if possible.

&gt;WNG

WNG =00 , RECORD : 68 68 66 60 66 66 60 68 66 66

&gt;

**WNGLV: Warning Output Level****I/O**

<b>Execution Mode</b>	Immediate	
<b>Syntax</b>	WNGLV n	
<b>Range</b>	n = 0: Normally Open 1: Normally Closed	
<b>Initial Value</b>	0	
<b>RESET</b>	RESET or recycle the power is required to activate the change. The parameter is saved automatically.	
<b>Access</b>	READ and WRITE	
<b>Description</b>	WNGLV is the active level of the Warning (WNG) output, if used.	
<b>See Also</b>	OUTWNG, SIGWNG	
<b>Example</b>	Command	Description
	>WNGLV=1 WNGLV=0(1) >RESET Resetting system.	#Set the WNG output logic to Normally Closed  #RESET the device to initialize the modified WNG setting
-----		
CRD5xx-KP CRK Series Built-in Controller Software Version: A378 V.x.xx Copyright 2009-2011 ORIENTAL MOTOR U.S.A. CORP.		
-----		
>WNGLV		#New value is active
	WNGLV=1(1) >	

# 14 Troubleshooting

This chapter explains the system's protective functions and procedures for troubleshooting alarm conditions.

## 14.1 Protective Functions

This section covers the system's protective functions and methods used to recover from alarm conditions.

- Most alarm conditions cause motion and sequence processing to stop, and many cause the system to disable motor current and lose holding torque. The system should be used in a way that prevents personal injury or damage to equipment if an alarm condition occurs.
- When an alarm occurs, determine and correct the cause of the alarm before attempting to restore normal operation. Some alarms can be cleared with the ALMCLR command; others require resetting the system or cycling input power. (A few alarms indicate serious system malfunction, and cannot be cleared.) The cause of the alarm should always be corrected before attempting to clear the system alarm state.

### ■ Types of Protective Functions and Check Methods



#### Warning

The device has protective functions to protect itself from rising ambient temperatures, poor connections, abnormal input power and other similar conditions.

When a protective function is triggered, the ALARM LED on the front side of the device blinks and the ALM output, if configured, is set to its active state.

Depending on the type of protective function, current to the motor may be disabled, resulting in a loss of holding torque.

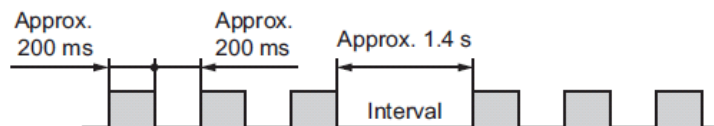
- How to check protective functions

The type of protective function that has been activated can be checked using the following two methods:

- 1) Count how many times the ALARM LED blinks on the back side of the device.

An Example of the ALARM LED's blinking cycle is shown in the figure below.

Example: Overvoltage protection



- 2) Check the alarm code using the ALM command.

- Clearing alarm conditions

Before clearing alarm conditions, always correct the cause of the alarm.

To clear an alarm condition, perform one of the following:

- Enter an ALMCLR command, for alarm conditions that ALMCLR can clear (refer to table above).
- Enter a RESET command.
- Turn the ALMCLR input ON and the OFF. (The alarm will be reset at the OFF edge of the input.)
- Turn OFF the power, wait for the green POWER LED to turn OFF, then turn power back on.

Types of protective functions

Protective Function	Description	Alarm Code	ALARM LED Blinks	System Action	ALMCLR Effect
No alarm	No alarm	0x00	OFF	Normal Operation	—
Stack overflow	Sequence memory "stack" exhausted	0x90	1	Motion and sequence processing stop.	Clears alarm
Sequence reference error	Attempt to call a non-existing sequence as a subroutine	0x94			
Calculation over flow	Sequence calculation result exceeded numerical limits	0x98			
Parameter Range error	Attempt to set a parameter to a value outside its range	0x99			



Protective Function	Description	Alarm Code	ALARM LED Blinks	System Action	ALMCLR Effect
Zero division	Attempt to divide by zero	0x9A	1	Motion and sequence processing stop.	Clears alarm
PC counter execution error	Attempt to modify position counter PC while a motion was in process	0x9D			
User variable reference error	Attempt to access a non-existing user-defined variable	0x9E			
Parameter write error	Attempt to change a parameter under invalid conditions (e.g. if prohibited while moving)	0x9F			
Motion while in motion	Attempt to execute a motion while an incompatible motion is in progress	0xA0			
User alarm	ALMSET command intentionally executed	0xE0			
Driver overheat	Drive temperature is out of specification	0x21	2	Motor current disabled (no holding torque)	Clears alarm if condition corrected
Over load	The cumulative value of the applied load exceeding the maximum torque exceeded the value set in the SCTO parameter.	0x30			Clears alarm
Over voltage	DC input voltage out of specification (high)	0x22	3	Motor current disabled (no holding torque)	No effect
Over position error *Appropriate encoder has to be used with your motor	The deviation between the encoder counter value and command position reached STOB parameter and STOACT parameter was set to “alarm”.	0x10	4	Defined by STOACT setting	Clears alarm if condition corrected
Panic stop	System executed a panic stop because of a PSTOP input or command	0x68	6	Defined by ALMACT setting	Clears alarm
LS logic error	Positive and negative position limit signals on simultaneously	0x60	7	Motion and sequence processing stop	Clears alarm
LS connected in reverse	Positive or negative position limit signal detected opposite home seeking direction	0x61			
HOME operation failed	Unstable or unexpected position limit signal detected while seeking home position	0x62			
HOMES not found	No HOME input detected between position limit signals while seeking home position	0x63			
TIM, Index, SLIT signal error	Timing position or Index or SLIT input expected with HOME input: not found	0x64			
Hardware over travel	Positive or negative position limit signal detected	0x66		Defined by ALMACT and OTACT settings	
Software over travel	Position outside of programmed positive and negative position limits	0x67			
LS detection during home offset motion	Positive or negative position limit signal detected while moving to OFFSET position after homing	0x6A		Motion and sequence processing stop	
Motion parameter error	Attempt to execute motion with incompatible motion parameters	0x70			
EEPROM error	User data in non-volatile EEPROM memory is corrupt	0x41	9	Motor current disabled (no holding torque)	No effect
System error	System detected unexpected internal logic state	0xF0	ON	Motor current disabled (no holding torque)	No effect
Memory error	Internal memory access error	0xF1			
Sequence internal error	Sequence code invalid or corrupt	0xF2			

## 14.2 Corrective Actions

If device operation is not normal, check this section and take appropriate action. If operation is still not normal, contact your nearest Oriental Motor office.

**Memo** Perform failure diagnosis using the following methods:

- Check the alarm code using the ALM command.
- Count how many times the ALARM LED blinks.

Phenomenon	Alarm Code	ALARM LED Blinks	Protective Function	Description	Action
Motion and sequence execution stop	0x90	1	Stack overflow	Sequence memory "stack" exhausted	Restructure sequences to reduce the number of nested blocks or subroutine calls
	0x94		Sequence reference error	Attempt to call a non-existing sequence as a subroutine	Revise the CALL statement or rename the intended target sequence
	0x98		Calculation overflow	Sequence calculation result exceeded numerical limits	Check math operations, make sure they cannot overflow
	0x99		Parameter Range error	Attempt to set a parameter to a value outside its Range	Make sure all assignments stay within defined limits
	0x9A		Zero division	Attempt to divide by zero	Check division operations, test divisor for zero before division
	0x9D		PC counter execution error	Attempt to modify position counter PC while a motion was in process	Make sure that PC is only changed when motor is stopped
	0x9E		User variable reference error	Attempt to Access a non-existing user-defined variable	Make sure the target user-defined variable exists: use the correct name in sequence
	0x9F		Parameter write error	Attempt to change a parameter under invalid conditions (e.g. if prohibited while moving)	Make sure that motion is not occurring
	0xA0		Motion while in motion	Attempt to execute a motion while an incompatible motion is in progress	Make sure motions are not started before a previous motion is complete. Use MEND, poll SIGMOVE, or monitor the MOVE output to detect motion complete.
	0xE0		User alarm	ALMSET command intentionally executed	If a user alarm was not expected, check sequence programming for inappropriate ALMSET command(s)

Phenomenon	Alarm Code	ALARM LED Blinks	Protective Function	Description	Action
Motion and sequence execution stop	0x60	7	LS logic error	Positive and negative position limit signals on simultaneously	<ul style="list-style-type: none"> <li>- Check limit sensors and wiring.</li> <li>- Check input signal configuration.</li> <li>- Check the logic setting for limit sensors (OTLV): Normally open (N.O.) or Normally closed (N.C.).</li> </ul>
	0x61		LS connected in reverse	Positive or negative position limit signal detected opposite home seeking direction	
	0x62		HOMES operation failed	Unstable or unexpected position limit signal detected while seeking home position	
	0x63		HOME not found	No HOMES input detected between position limit signals while seeking home position	Check HOMES sensor wiring and connections
	0x64		TIM, Index or SLIT signal error	Timing position or Index or SLIT input expected with HOMES input: not found	Selected mechanical home seeking operation (see HOMESL) requires a valid SLIT input and/or a valid Timing position and/or Index signal while HOMES input active. Make sure HOMES and other required input(s) can be active at the same location.
	0x6A		LS detected during home offset motion	Positive or negative position limit signal detected while moving to OFFSET position after homing	Make sure that the OFFSET distance, measured from the HOME signal position, does not trigger a limit sensor
	0x70		Motion parameter error	Attempt to execute motion with incompatible motion parameters	<ul style="list-style-type: none"> <li>- Make sure current is enabled (CURRENT=1).</li> <li>- Home seeking: make sure required inputs are configured.</li> <li>- Linked indexing: make sure all linked segments execute in the same direction.</li> </ul>
Motion and sequence execution stop.  Motor may or may not have holding torque, depending on ALMACT.	0x10	4	Over position error	The deviation between the encoder counter value and command position reached STOB parameter and STOACT parameter was set to "alarm".	<ul style="list-style-type: none"> <li>- Check for machine jam</li> <li>- Reduce load.</li> <li>- Increase current.</li> <li>- Reduce running velocity.</li> <li>- Increase acceleration or deceleration times.</li> <li>- Increase value of STOB parameter</li> </ul>
	0x68	6	Panic stop	System executed a panic stop because of a PSTOP input or command	If a panic stop was unexpected: <ul style="list-style-type: none"> <li>- Check PSTOP input configuration.</li> <li>- Check sequence programming for inappropriate PSTOP command(s).</li> </ul>
	0x66	7	Hardware over travel	Positive or negative position limit signal detected	<ul style="list-style-type: none"> <li>- Check motion parameters.</li> <li>- Make sure home position is correct.</li> <li>- Check limit sensors and wiring.</li> <li>- Check input signal configuration.</li> <li>- Check the logic setting for limit sensors (OTLV): Normally open (N.O.) or Normally closed (N.C.).</li> </ul>
	0x67		Software over travel	Position outside of programmed positive and negative position limits	<ul style="list-style-type: none"> <li>- Check motion parameters.</li> <li>- Check software position limits.</li> <li>- Make sure home position is correct.</li> </ul>

Phenomenon	Alarm Code	ALARM LED Blinks	Protective Function	Description	Action
The motor lacks holding torque.	0x21	2	Drive overheat	Drive temperature exceeds programmed limit DTMPMAX	<ul style="list-style-type: none"> <li>- Reduce motion duty cycle.</li> <li>- Reduce current.</li> <li>- Increase ventilation.</li> <li>- Reduce ambient temperature.</li> </ul>
	0x30		Over load	The cumulative value of the applied load exceeding the maximum torque exceeded the value set in the SCTO parameter.	Reduce the load or increase the acceleration/deceleration time.
	0x22	3	Overvoltage	DC input voltage out of specification (high)	Check power supply. Can also occur while slowing a large inertial load (regenerative braking). Reduce load inertia or increase deceleration time.
	0x41	9	EEPROM error	User data in non-volatile EEPROM memory is corrupt	Contact Oriental Motor to arrange for inspection or repair.
	0xF0	ON	System error	System detected unexpected internal logic state	
	0xF1		Memory error	Internal memory access error	
	0xF2		Sequence internal error	Sequence code invalid or corrupt	

# 15 Additional Functions

This chapter explains the useful functions that facilitate operation, functions that become available when an encoder is connected, and the like.

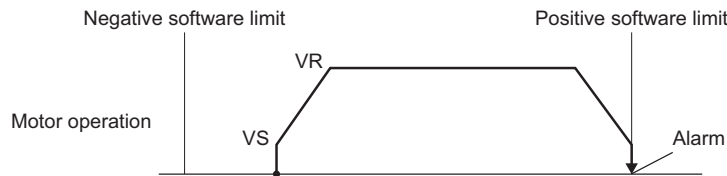
## 15.1 Software over travel

The software over travel is a function that limits the range of movement via software settings.

When a positioning operation or continuous operation is started where the position specified by a “positive software limit” parameter or “negative software limit” parameter is to be exceeded, the motor will decelerate to a stop and software over travel alarm will generate.

If software over travel is to be used, set the “software over travel” parameter to “enable”.

The operation pattern shown below applies when an operation where a soft limit is to be exceeded is started.



Software over travel will become effective after the home position is set. For the method to set the home, refer to p.47.

## 15.2 Hardware over travel

Hardware over travel is a function that limits the range of movement using limit sensors ( $\pm$ LS).

If the +LS or -LS signal is input during positioning operation or continuous operation, the motor will stop and a hardware over travel alarm will generate.

When the +LS or -LS input is turned ON during positioning operation or continuous operation, the motor will stop and hardware over travel alarm will generate.

You can change the input logic for  $\pm$ LS sensors using the “LS contact configuration” parameter. The stopping method to be applied upon detection of a limit sensor signal can be set using the “over travel action” parameter.

If the limit sensor input is turned ON, issuing a positioning operation start command will only generate an alarm and the operation will not be started.

Continuous operation can be performed in - direction while the +LS input is ON, or in + direction while the +LS input is ON. The motor operates at the starting speed within range between the limit sensors. Once the motor deviates from the limit sensor range, it will operate at the operating speed.

## 15.3 Position control

The driver has an internal oscillating-pulse counter. The command position can be read from this counter using the RS-485 communication. You can also check the command position by counting the number of times a PLS-OUT or DIR-OUT output signal has been output.

The control range of command positions is -2,147,483,648 to 2,147,483,647.

The command position will be cleared to 0 once the return-to-home operation ends successfully. If a RESET is performed using the PC “position counter” command, the command position will change to the value set with the PC parameter.

If an encoder is connected and the “stepout detection” parameter is set to “enable”, the command position will be refreshed by the encoder counter value while the motor is not excited.

# 16 Alarms and warnings

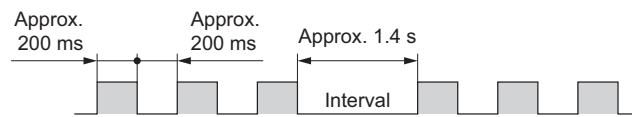
The driver provides alarms that are designed to protect the driver from overheating, poor connection, incorrect operation, etc. (protective functions), as well as warnings that are output before the corresponding alarms generate (warning functions).

## 16.1 Alarms

When an alarm generates, the ALM output will turn OFF and the motor will stop. When an alarm generates, the ALARM LED will blink. The cause of the alarm can be checked by counting the number of times the ALARM LED blinks.

Present alarms can be checked using the RS-485 communication. You can also check the records of up to ten most recent alarms starting from the latest one, or clear the alarm records.

**Example:** Overvoltage alarm (number of blinks: 3)



### ■ Alarm Reset

Perform one of the RESET operations specified below.

Before resetting an alarm, always remove the cause of the alarm and ensure safety.

- Turn the ALMCLR input to ON and then OFF. (The alarm will be RESET at the OFF edge of the input.)
- Perform an alarm RESET using the RS-485 communication.
- Cycle the power.

#### Note

Some alarms cannot be reset with the ALMCLR input or RS-485 communication. Check the following table to identify which alarms meet this condition.  
To reset these alarms, you must cycle the power.

Alarm type	No. of ALARM LED blinks	Alarm code	Motor operation	Alarm Reset <sup>□*</sup>	Cause	Remedial action
Overheat	2	21	The motor current is cut off.	Possible	The internal temperature of the driver exceeded 85 °C (185 °F).	Review the ventilation condition in the enclosure.
Over load		30			The cumulative value of the applied load exceeding the maximum torque exceeded the value set in the SCTO parameter.	Reduce the load or increase the acceleration/deceleration time.
Overvoltage	3	22	The motor current is cut off.	Not possible	<ul style="list-style-type: none"> <li>The internal voltage exceeded the permissible value due to regeneration, etc.</li> <li>The power supply voltage exceeded the allowable value.</li> </ul>	<ul style="list-style-type: none"> <li>If this alarm generates during operation, reduce the load or increase the acceleration/deceleration time.</li> <li>Check the power supply voltage.</li> </ul>
Over position error *Appropriate encoder has to be used with your motor	4	10	The motor stops.	Possible	The deviation between the encoder counter value and command position reached the stepout detection band when the “stepout detection action” parameter was set to “alarm”.	<ul style="list-style-type: none"> <li>Reduce the load, or increase the acceleration/deceleration time.</li> <li>Check the setting of “stepout detection band” parameter.</li> <li>Check the setting of “encoder electronic gear” parameter.</li> </ul>
±LS both sides active	7	60	The motor stops.	Possible	Both the +LS and -LS signals were detected when LS detection was enabled.	Check the sensor logic and setting of “LS contact configuration” parameter.
Reverse limit sensor connection		61	The motor stops.	Possible	The LS opposite to the operating direction has detected during a return-to-home operation.	Check the ±LS wiring.
Home seeking error		62	The motor stops.	Possible	Return-to-home operation did not complete normally.	<ul style="list-style-type: none"> <li>An unanticipated load may have been applied during the return-to-home operation. Check the load.</li> <li>If the installation positions of ±LS and HOMES are close to one another, the return-to-home sequence may not end properly, depending on the starting direction of return-to-home operation. Review the sensor installation positions and the starting direction of return-to-home operation.</li> <li>Return-to-home operation may have been performed in a condition where both +LS and -LS were detected. Check the sensor logic and the setting of “LS contact configuration” parameter.</li> </ul>
No HOMES		63	The motor stops.	Possible	The HOMES is not detected at a position between +LS and -LS during return-to-home operation in 3-sensor mode.	<ul style="list-style-type: none"> <li>Set a HOMES between +LS and -LS.</li> <li>Check the HOMES wiring.</li> </ul>

\* Reset alarm using the ALMCLR input or RS-485 communication.

Alarm type	No. of ALARM LED blinks	Alarm code	Motor operation	Alarm Reset* □	Cause	Remedial action
TIM, Index, SLIT input error	7	64	The motor stops.	Possible	None of the SLIT input, TIM output and Index output could be detected during return-to-home operation.	<ul style="list-style-type: none"> <li>Adjust the connection condition of the motor output shaft and load as well as the HOMES position so that at least one of the SLIT input, TIM output and Index output will turn ON while HOMES is ON.</li> <li>If the SLIT input, TIM output and Index output are not used with HOMES, set the “TIM signal detection with home-seeking” parameter and “SLIT detection with home-seeking” parameter to “disable.”</li> </ul>
Over travel		66	The motor stops.	Possible	A +LS or -LS signal was detected when hardware over travel was enabled.	Reset the alarm using the ALMCLR input, and then pull out from the LS sensor via continuous operation or return-to-home operation.
Software over travel		67	The motor stops.	Possible	A software limit was reached when software over travel was enabled.	Perform the operation within the range between the software limits. In single-motion operation, check to see if the position exceeds the soft limit. In linked-motion operation, check to see if the result of linked position exceeds the soft limit.
Home seeking offset error		6A	The motor stops.	Possible	A limit sensor signal was detected during offset movement as part of return-to-home operation.	Check the setting of “position offset of home-seeking” parameter.
Invalid operation data		70	The motor stops.	Possible	<ul style="list-style-type: none"> <li>Five or more data may be linked.</li> <li>Data of different directions may be linked</li> </ul>	<ul style="list-style-type: none"> <li>Keep the number of operation data to be linked to 4 or less.</li> <li>Link operation data having the same direction.</li> </ul>
EEPROM error	9	41	The motor current is cut off.	Not possible	The stored data was damaged.	Initialize the parameters using the RS-485 communication.

\* Reset alarm using the ALMCLR input or RS-485 communication.



## 16.2 Warnings

When a warning generates, the WNG output will turn ON. The motor will continue to operate. Once the cause of the warning is removed, the WNG output will turn OFF automatically. The WNG command shows the current warning status, and the last 10 warnings..

**Note** You can also clear the warning records by turning OFF the driver power.  
The warning history is saved in the non-volatile EEPROM

### ■ Descriptions of warnings

Warning type	Warning code	Cause	Remedial action
Over position error *Appropriate encoder has to be used with your motor	10	The deviation between the encoder counter value and command position reached the stepout detection band when the “stepout detection action” parameter was set to “alarm”.	<ul style="list-style-type: none"> <li>• Reduce the load, or increase the acceleration/deceleration time.</li> <li>• Check the setting of “stepout detection band” parameter.</li> <li>• Check the setting of “encoder electronic gear” parameter.</li> </ul>
Overheat	21	The temperature inside the driver exceeded the value set in the “overheat warning” parameter.	Review the ventilation condition in the enclosure.
Overvoltage	22	<ul style="list-style-type: none"> <li>• The internal voltage exceeded the value set in the “overvoltage warning” parameter due to regeneration, etc.</li> <li>• The power supply voltage exceeded the value set in the “overvoltage warning” parameter.</li> </ul>	<ul style="list-style-type: none"> <li>• If this alarm generates during operation, reduce the load or increase the acceleration/deceleration time.</li> <li>• Check the power supply voltage.</li> </ul>

# 17 Inspection

---

It is recommended that periodic inspections be conducted for the items listed below after each operation of the motor.

If an abnormal condition is noted, discontinue any use and contact your nearest office.

## ■ During inspection

- Are any of motor mounting screws loose?
- Check for any unusual noises in the motor's bearings (ball bearings) or other moving parts.
- Are there any scratches, signs of stress or loose driver connection in the motor lead wires?
- Are the motor's output shaft (or gear output shaft) and load shaft out of alignment?
- Is there any looseness at the driver mounting points on the DIN rail?
- Is there any loose driver connector?
- Is there attachment of dust, etc., on the driver?
- Are there any strange smells or appearances within the driver?

### Note

The driver uses semiconductor elements, so be extremely careful when handling them. Static electricity may damage the driver.

# 18 General specifications

		Motor	Driver
Degree of protection		IP30	IP20
Operation environment	Ambient temperature	−10 to +50 °C (+14 to +122 °F) (non-freezing)	0 to +40 °C (+32 to +104 °F) (non-freezing)
	Humidity	85% or less (non-condensing)	
	Altitude	Up to 1000 m (3300 ft.) above sea level	
	Surrounding atmosphere	No corrosive gas, dust, water or oil	
Storage environment	Ambient temperature	−20 to +60 °C (−4 to +140 °F) (non-freezing)	−25 to +70 °C (−13 to +158 °F) (non-freezing)
	Humidity	85% or less (non-condensing)	
	Altitude	Up to 3000 m (10000 ft.) above sea level	
	Surrounding atmosphere	No corrosive gas, dust, water or oil	
Shipping environment	Ambient temperature	−20 to +60 °C (−4 to +140 °F) (non-freezing)	−25 to +70 °C (−13 to +158 °F) (non-freezing)
	Humidity	85% or less (non-condensing)	
	Altitude	Up to 3000 m (10000 ft.) above sea level	
	Surrounding atmosphere	No corrosive gas, dust, water or oil	
Insulation resistance		Applied 500 VDC 100 MΩ or more · Between coil and case	Applied 500 VDC 100 MΩ or more · Between FG terminal and power supply terminal
Withstand voltage		Applied 1.5 kVAC (1 kVAC for PK54□) 50/60 Hz for 1 minute. · Between coil and case	Applied 500 VAC 50/60 Hz for 1 minute, leak current 10 mA or less · Between FG terminal and power supply terminal

# 19 Options (sold separately)

---

## ■ RS-485 communication jumper cable

Use this cable to link multiple drivers with the RS-485 communication connectors (CN6, CN7).

**Model:** CC001-RS4 [0.1 m (0.3 ft.)]

## ■ Encoder motor connection cable

Use this cable/lead wire assembly when connecting an encoder to the drivers' encoder connector (CN5)

**Model:** LC09A-006 [0.6m (2 ft.)]

## ■ CRK Motion Creator GUI

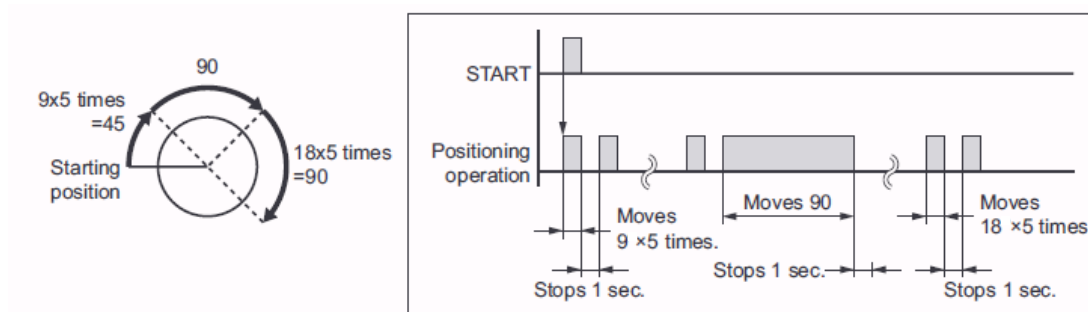
A graphical user interface (GUI) tool, CRK Motion Creator, is available for use with your computer. By just clicking your mouse, you can create motion, perform system configuration, write programs and upload/download programs and parameters easily. Of course, for the person who prefers to use a keyboard and the programming language; the device can be programmed via any terminal software on a PC, such as HyperTerminal. However, the CRK Motion Creator will greatly help you to save and load data between a PC and the CRK series built-in controller (Stored program). The CRK Motion Creator includes a motion creating function, sequence editing function, terminal function, data save/load function, and system parameter setting function. The latest version of the CRK Motion Creator program is available for download free at

<http://www.orientalmotor.com/support/software/software.html>

# 20 Sample Programs

This chapter provides sample programs.

## 20.1 Repeated Positioning Operation

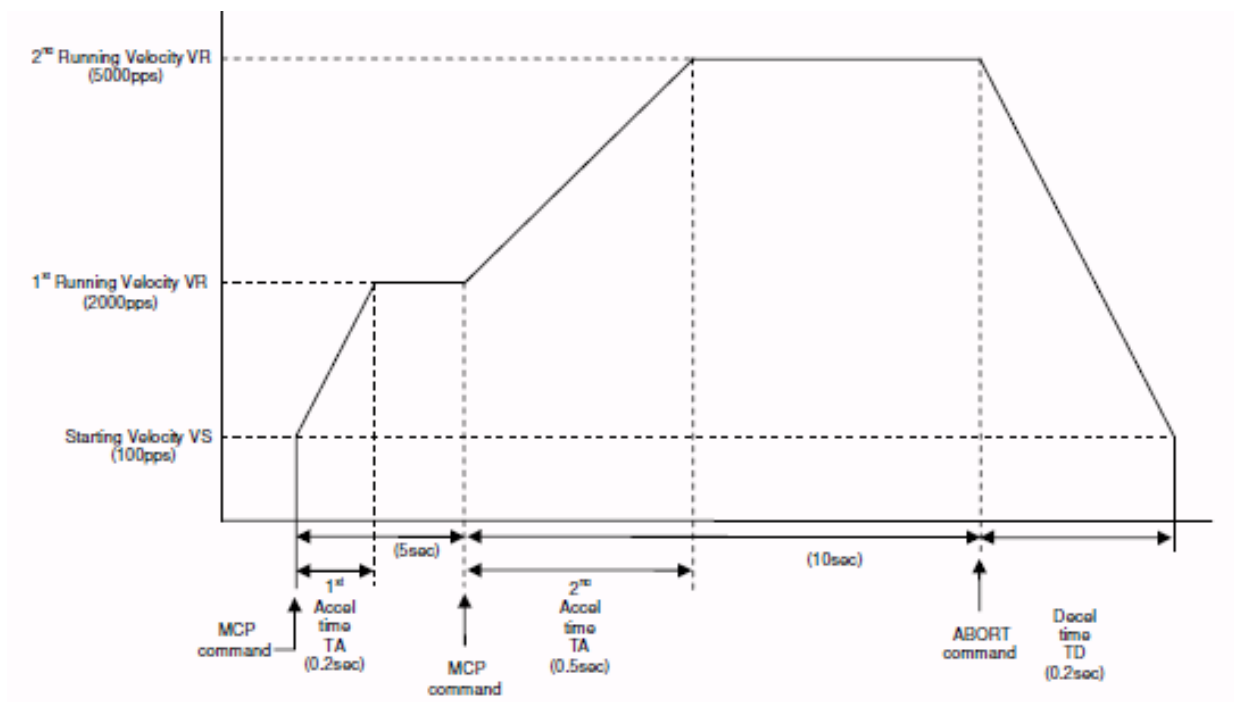


### ■ “Main” Program

```
(1) TA 0.1
(2) TD 0.1
(3) VS=10
(4) VR=360
(5) LOOP 5
(6) DIS=9
(7) MI
(8) MEND
(9) WAIT 1
(10) ENDL
(11) DIS=90
(12) MI
(13) MEND
(14) WAIT 1
(15) LOOP 5
(16) DIS=18
(17) MI
(18) MEND
(19) WAIT 1
(20) ENDL
(21) END
```

The acceleration time is set to 0.1sec.  
 The deceleration time is set to 0.1sec.  
 The starting velocity is set to 10 pps.  
 The running velocity is set to 360 pps.  
 Lines 6 through 9 are repeated five times.  
 The distance is set to 9 steps.  
 Incremental positioning operation is executed.  
 The program waits until the motion is ended.  
 The program waits 1 sec.  
 The LOOP statement is ended.  
 The distance is set to 90 steps.  
 Incremental positioning operation is executed.  
 The program waits until the motion is ended.  
 The program waits 1 sec.  
 Lines 16 through 19 are repeated five times.  
 The distance is set to 18 steps.  
 Incremental positioning operation is executed.  
 The program waits until the motion is ended.  
 The program waits 1 sec.  
 The LOOP statement is ended.  
 The program is ended.

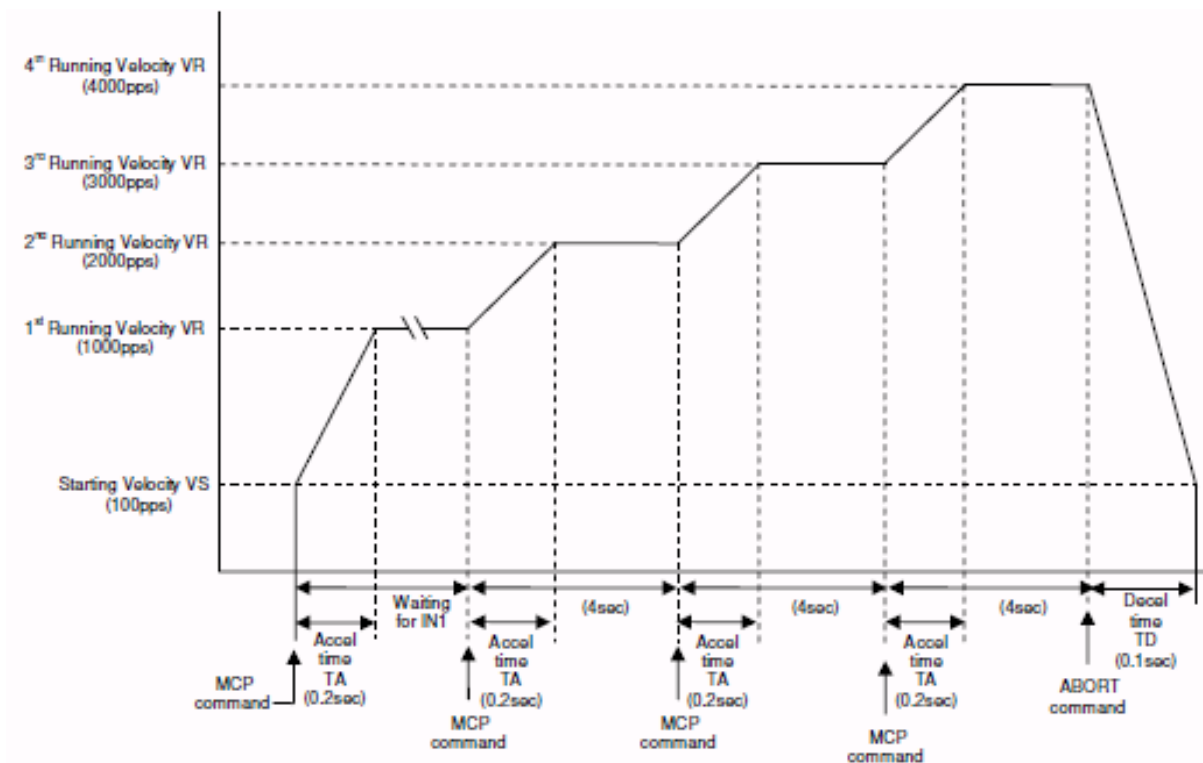
## 20.2 Speed Change On-The-Fly



### ■ “VelocityChangeScan” Program

- |              |   |
|--------------|---|
| (1) VS 100   | Set starting velocity to 100pps                             |
| (2) VR 2000  | Set 1st running velocity to 2000pps                         |
| (3) TA .2    | Set 1st accel time to 0.2sec                                |
| (4) TD .2    | Set decel time to 0.2sec                                    |
| (5) MCP      | Move continuously in the positive direction                 |
| (6) WAIT 5   | Wait 5sec   |
| (7) TA .5    | Set 2nd accel time to 0.5sec                                |
| (8) VR 5000  | Set 2nd running velocity to 5000pps                         |
| (9) MCP      | Move continuously in the positive direction at new velocity |
| (10) WAIT 10 | Wait 10sec  |
| (11) ABORT   | Abort sequence execution                                    |

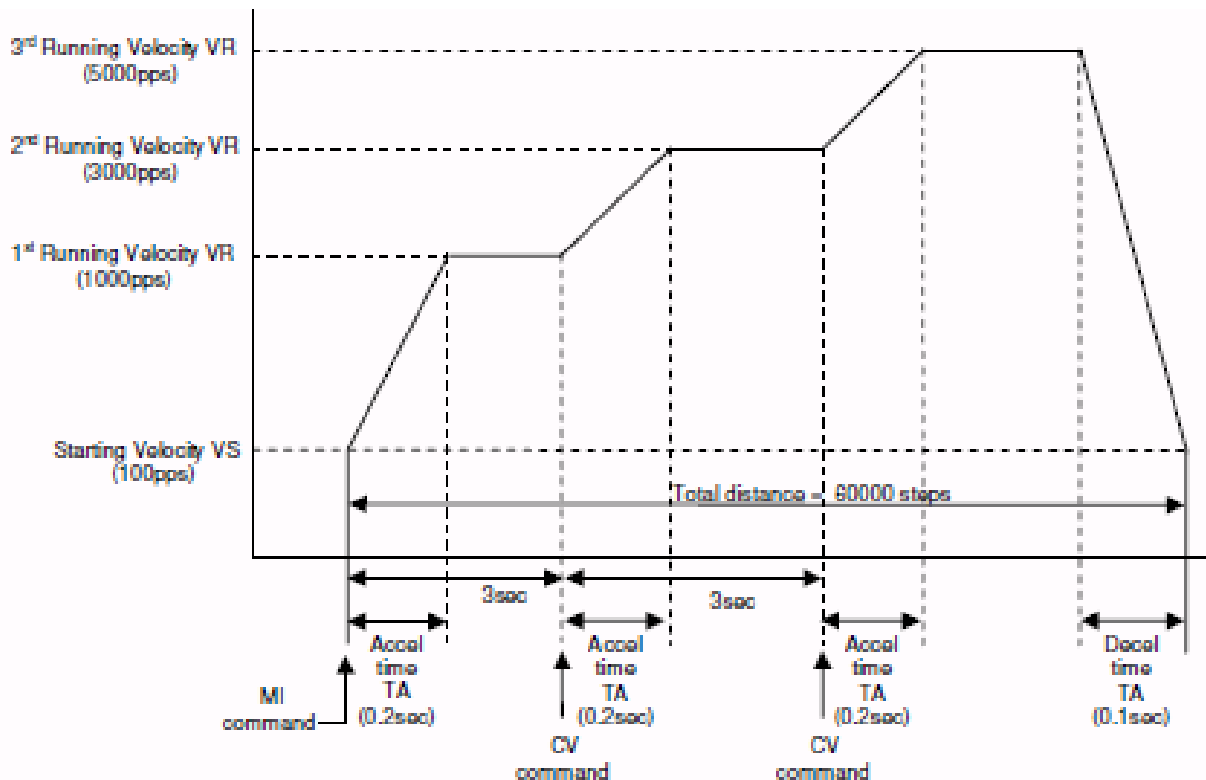
## 20.3 Speed Change On Input



### ■ “VelocityChangeInput” Program

- |                       |   |
|-----------------------|---|
| (1) VS 100            | Set starting velocity to 100pps             |
| (2) VR 1000           | Set 1st running velocity to 1000pps         |
| (3) TA .2             | Set accel time to 0.2sec                    |
| (4) TD .1             | Set decel time to 0.1sec                    |
| (5) MCP               | Move continuously in the positive direction |
| (6) WHILE (IN1=0)     |   |
| (7) WEND              | Wait until input 1 turns ON                 |
| (8) LOOP 3            | Loop 3 times                                |
| (9) VR=VR+1000        | Increase velocity by 1000pps                |
| (10) MCP              | Move continuously in the positive direction |
| (11) SAS SPEED CHANGE | Send characters "SPEED CHANGE" to terminal  |
| (12) WAIT 4           | Wait 4sec                                   |
| (13) ENDL             | End loop                                    |
| (14) ABORT            | Abort sequence execution                    |

## 20.4 Speed Change During Index Move

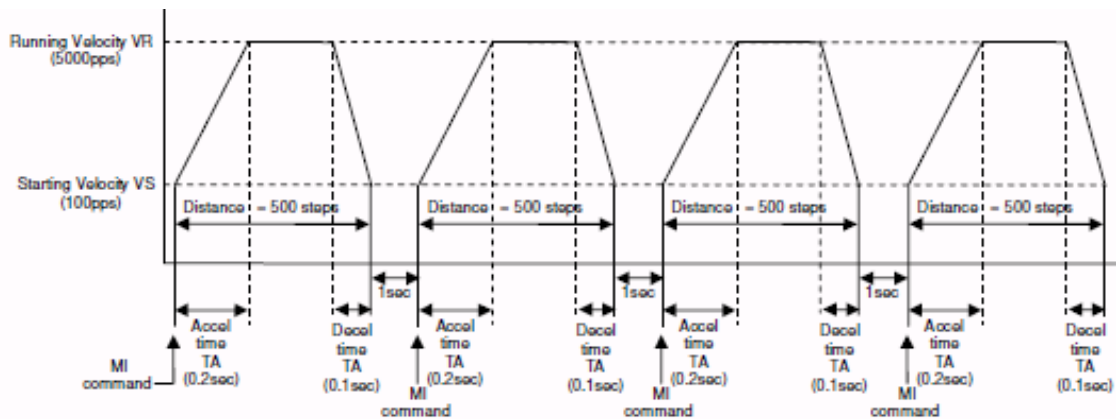


### ■ “VelocityChangeIndex” Program

- |               |                                    |
|---------------|------------------------------------|
| (1) pc =0     | Set PC to 0                        |
| (2) VS 100    | Set starting velocity to 100pps    |
| (3) VR 1000   | Set running velocity to 1000pps    |
| (4) TA .2     | Set accel time to 0.2sec           |
| (5) TD .1     | Set decel time to 0.1sec           |
| (6) DIS 60000 | Set distance to 60000 steps        |
| (7) MI        | Move incrementally                 |
| (8) WAIT 3    | Wait 3sec                          |
| (9) CV 3000   | Change running velocity to 3000pps |
| (10) WAIT 3   | Wait 3sec                          |
| (11) CV 5000  | Change running velocity to 5000pps |



## 20.5 Looped Index Move

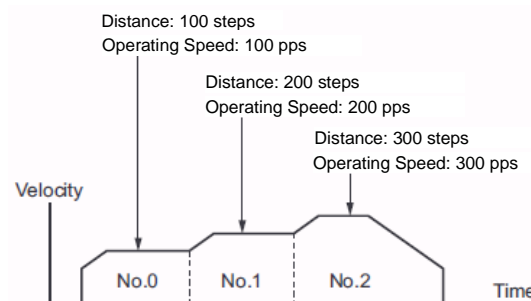


### ■ “LoopedIndex” Program

- |             |                                 |
|-------------|---------------------------------|
| (1) VS 100  | Set starting velocity to 100pps |
| (2) VR 5000 | Set running velocity to 5000pps |
| (3) TA .2   | Set accel time to 0.2sec        |
| (4) TD .1   | Set decel time to 0.1sec        |
| (5) DIS 500 | Set distance 500 steps          |
| (6) LOOP 4  | Loop 4 times                    |
| (7) MI      | Move incrementally              |
| (8) MEND    | Wait for motion end             |
| (9) WAIT 1  | Wait 1sec                       |
| (10) ENDL   | End of loop                     |

## 20.6 Executing Linked Operation

LINKx	Setting Value
LINK0	1 (linked)
LINK1	1 (linked)
LINK2	0 (one-shot)



- |                |   |
|----------------|---|
| (1) DIS 0=100  | The distance for operation number 0 is set to 100 steps.                              |
| (2) DIS 1=200  | The distance for operation number 1 is set to 200 steps.                              |
| (3) DIS 2=300  | The distance for operation number 2 is set to 300 steps.                              |
| (4) VR 0=100   | The operating speed for operation number 0 is set to 100 pps.                         |
| (5) VR 1=200   | The operating speed for operation number 1 is set to 200 pps.                         |
| (6) VR2=300    | The operating speed for operation number 2 is set to 300 pps.                         |
| (7) INCABS 0=1 | The positioning mode for operation number 0 is set to incremental.                    |
| (8) INCABS 1=1 | The positioning mode for operation number 1 is set to incremental.                    |
| (9) INCABS 2=1 | The positioning mode for operation number 2 is set to incremental.                    |
| (10) LINK 0=1  | Operation number 0 is set to linked.  |
| (11) LINK 1=1  | Operation number 1 is set to linked.  |
| (12) LINK 2=0  | Operation number 2 is set to one shot linked.   |
| (13) MI 0      | Start the operation to start at operation number 0. (Numbers 0 through 2 are linked.) |
| (14) END       | The program is ended.   |

# 21 Multi-Drop Connections

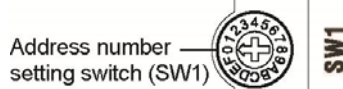
This chapter describes the procedure used to connect two or more devices via a multi-drop connection (up to 16 devices).

## 21.1 Setting the Unit ID's

Set the address number using the address setting switch (SW1).

Factory setting:

SW1: 0, (address number 0) Do not use duplicate axis numbers.

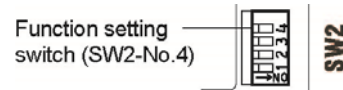


## 21.2 Multi-axis mode

Set the driver to multi-axis mode using the multi-axis mode setting switch (SW2-No.4) to ON.

Factory setting:

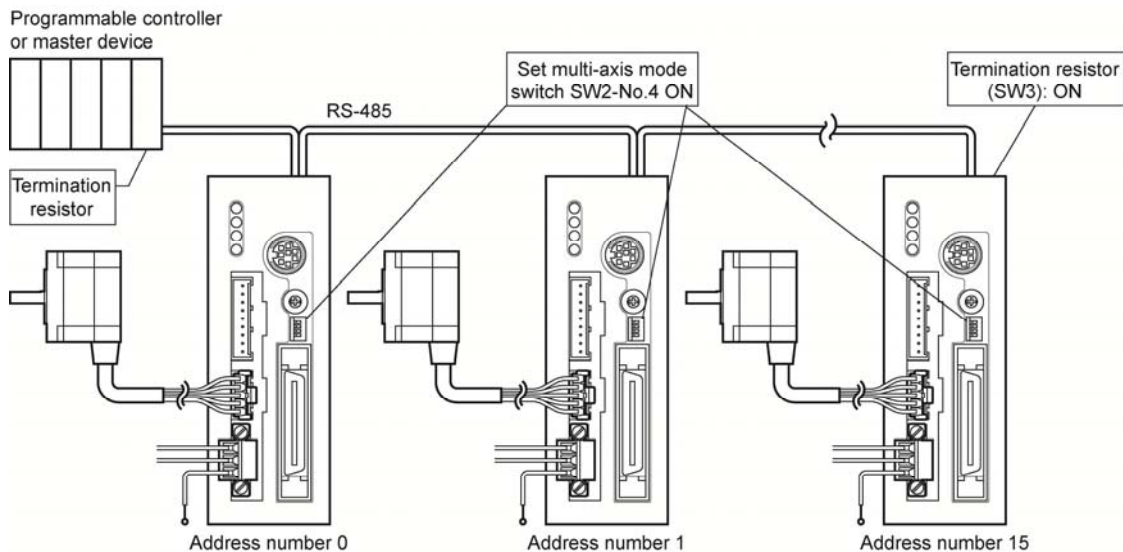
SW2-No.4: OFF, (single axis mode).



## 21.3 Multi-Drop Connection Procedure

Use the RS-485 communication pins (TR+, TR-, GND) of the RS-485 communication connector.

An example of connecting three drivers via a multi-drop is shown below.



### Note

- The maximum distance when using a multi-drop connection should be 50 m (164 feet).
- Wire the RS-485 signal lines over the shortest possible distance. It is recommended that the signal lines be shielded to protect them from noise interference.
- An optional RS-485 jumper cable (part number CC001-RS4) is available for connecting multiple devices together with the CN6 and CN7 connectors.

## 21.4 Multi-Drop Serial Communication Example

Call the specific device used for communication via the @command. When the power is turned on, the communication device is set to the one whose axis number is 0.

Example) Connection to the device whose axis number is 1 to the communication line.

When the power is turned on, the communication device is set to the one whose axis number is 0.

As a result, a prompt (“>”) is not output.

@1 : Executing a “@1” command connects device 1.  
1> : A prompt (“1>”) is output.  
@2 : Connect to device 2.  
2> : A prompt (“2>”) is output.

**Note**

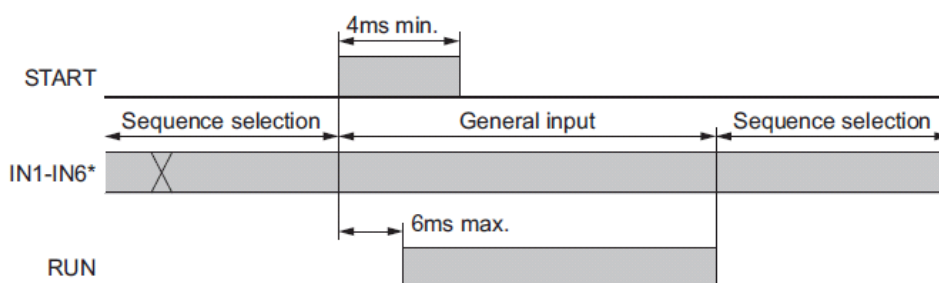
- The “@” character will not be echoed until an axis number is entered following it. Once an axis number has been entered, the “@” character will be displayed on the screen.

# 22 Timing Charts

This chapter includes timing charts that describe the operation of the CRK series built-in controller (stored program)

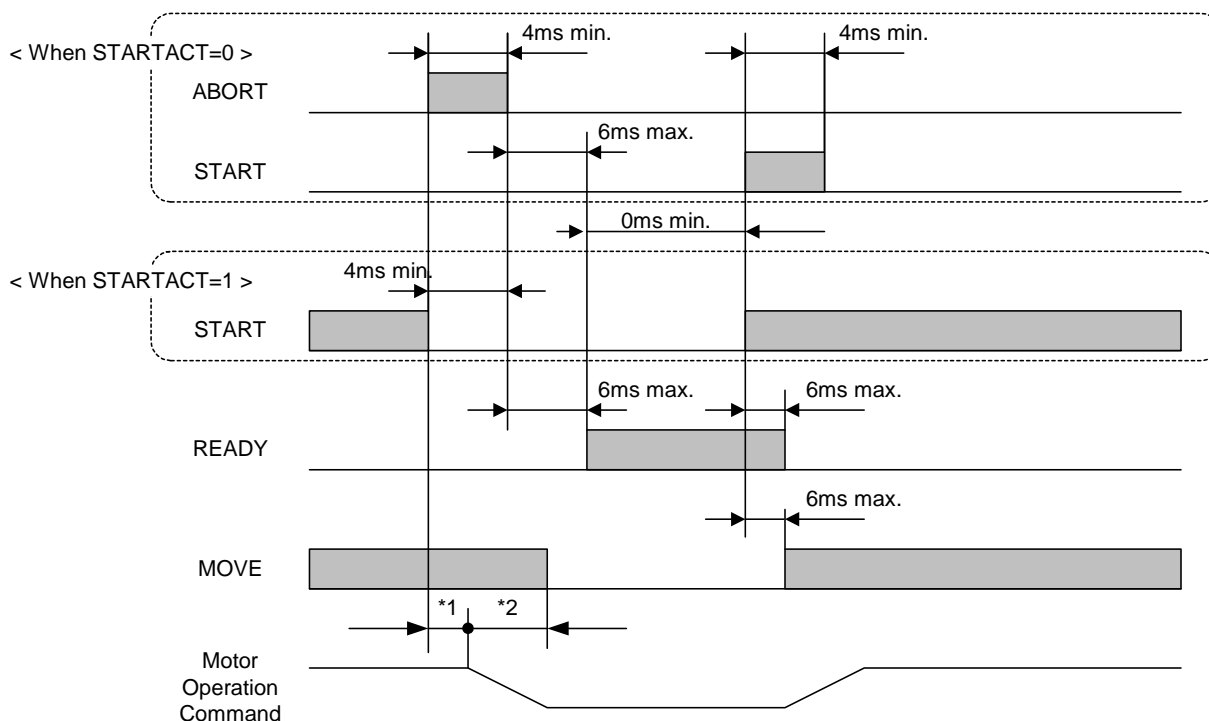
## 22.1 Execution of a Sequence

### ■ Selection and Execution of a Sequence



\* Only inputs that are not assigned are read.  
Inputs assigned to another function are always read as "0".

### ■ Execution and Stopping a Sequence (START, ABORT, READY, MOVE)

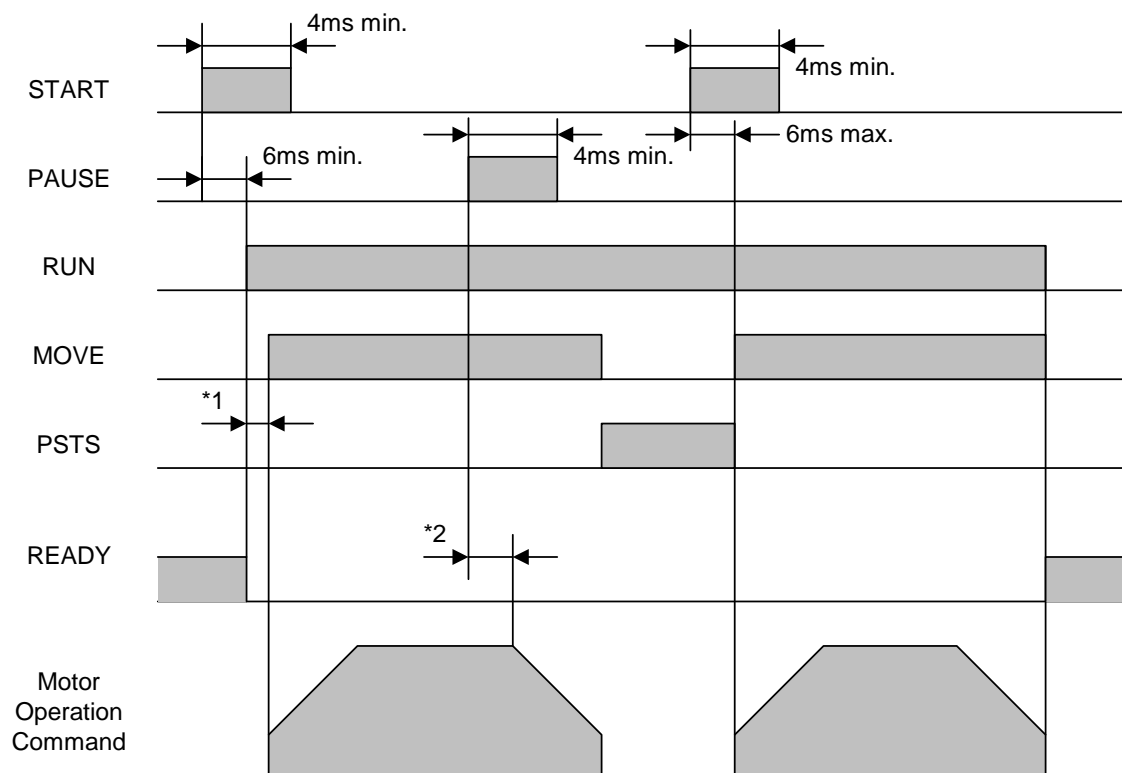


\*1 The specific time varies depending on the command speed.

\*2 The specific period varies depending on the setting of deceleration time.

## 22.2 Stopping Operation

### ■ Pausing Index Operation (PAUSE, PSTS)

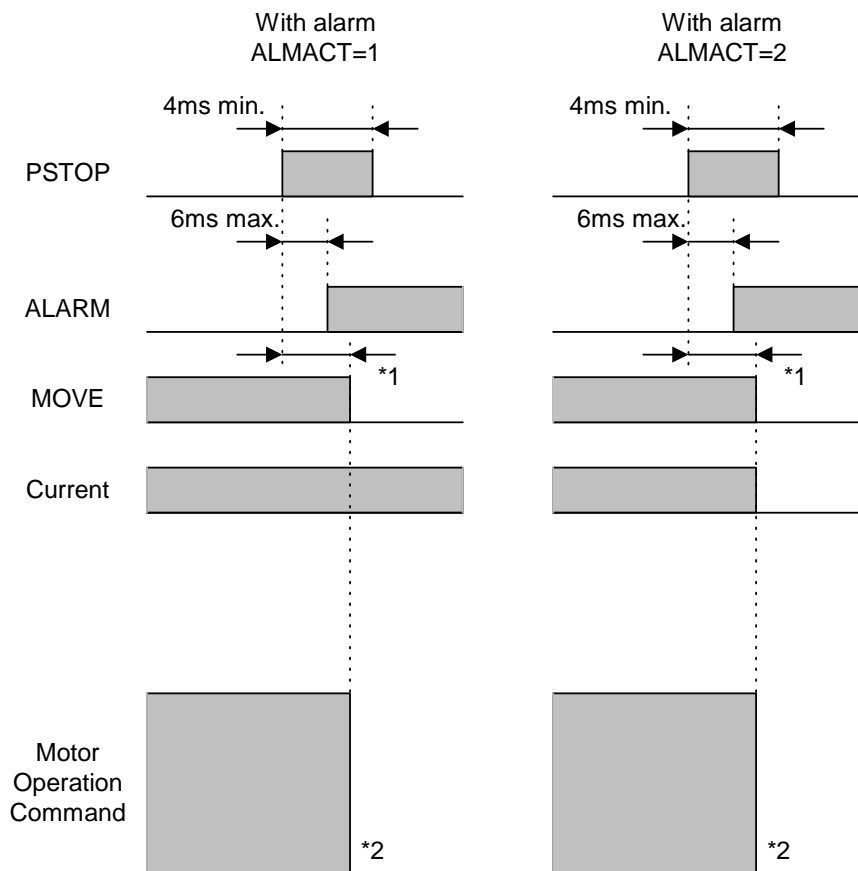


\*1 The specific time varies depending on the sequence

\*2 The specific time varies depending on the command speed.

**Note:** The START input will clear a PAUSE state.

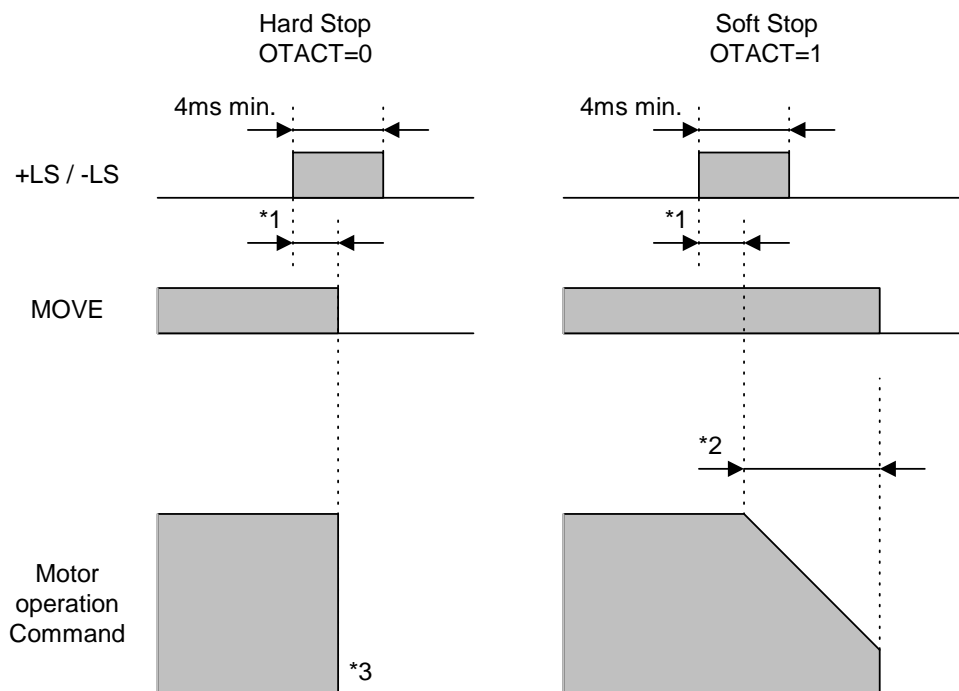
### When the PSTOP Input is Turned ON



\*1 The specific time varies depending on the command speed.

\*2 Motor might not stop immediately due to load condition.

### ■ When the (+LS, -LS) Input is Used

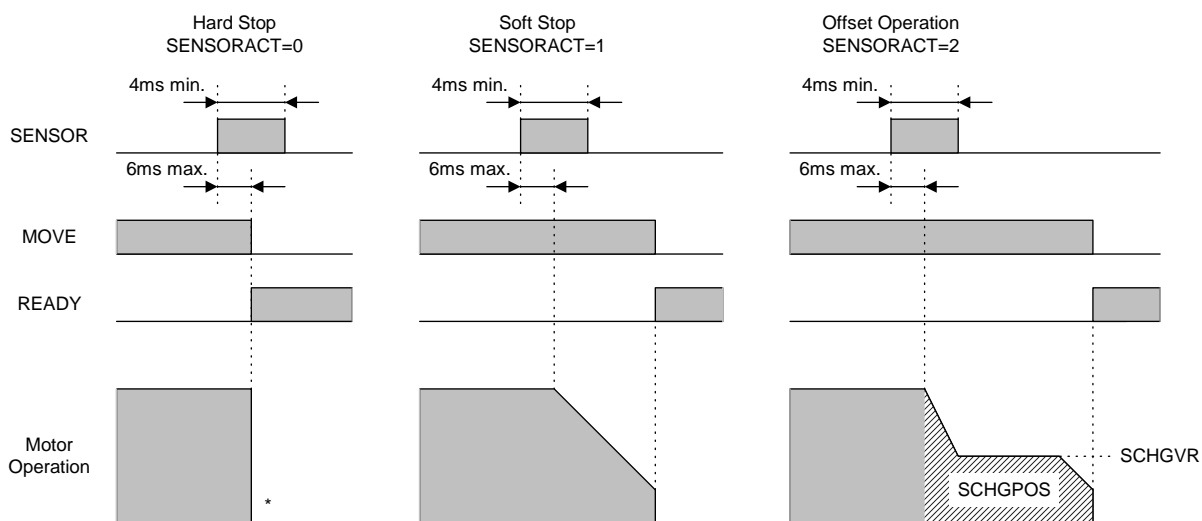


\*1 The specific time varies depending on the command speed.

\*2 The specific period varies depending on the setting of deceleration time.

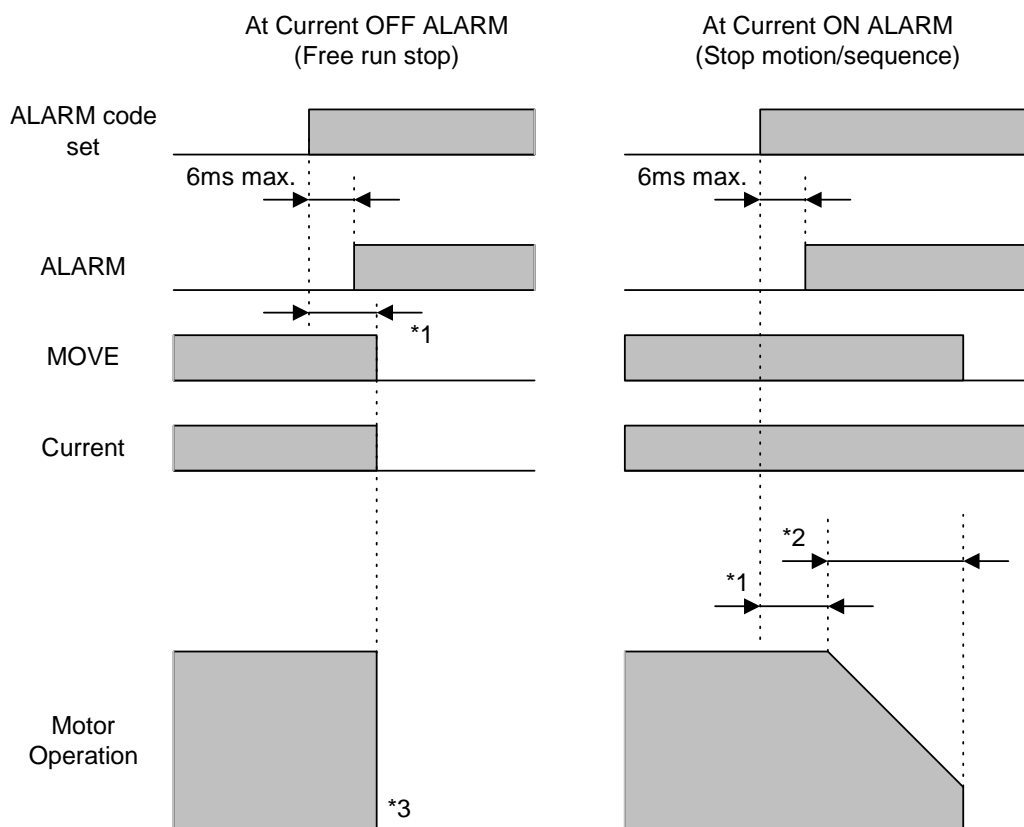
\*3 Motor might not stop immediately due to load condition.

## ■ When the SENSOR Input is Used



\* Motor might not stop immediately due to load condition.

## ■ When an ALARM has Occurred



\*1 The specific time varies depending on the command speed.

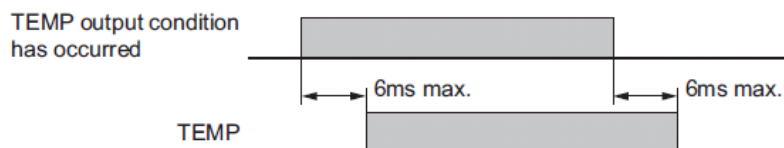
\*2 The specific period varies depending on the setting of deceleration time.

\*3 Motor might not stop immediately due to load condition.

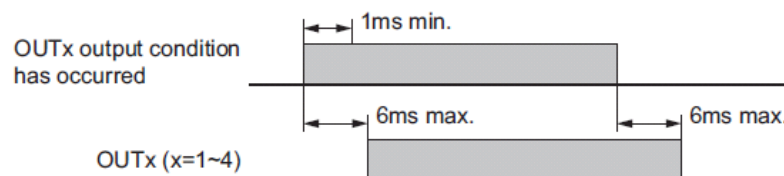


## 22.3 Outputs

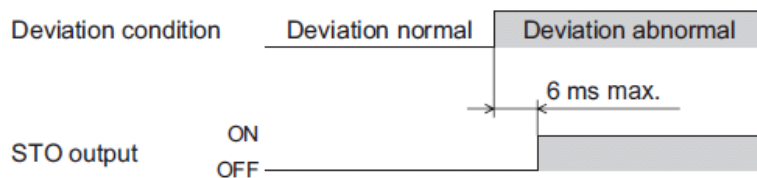
### ■ TEMP output



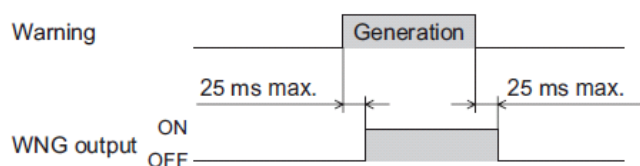
### ■ OUT1-OUT4 output



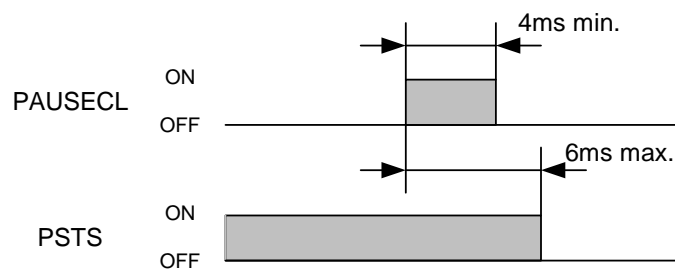
### ■ STO Output



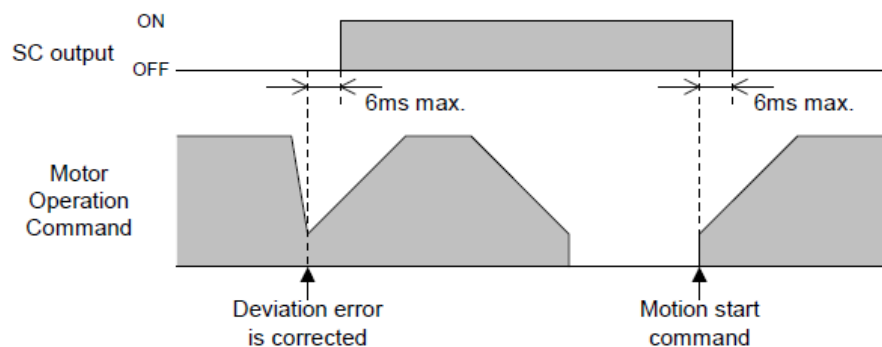
### ■ WNG Output



### ■ PSTS Output

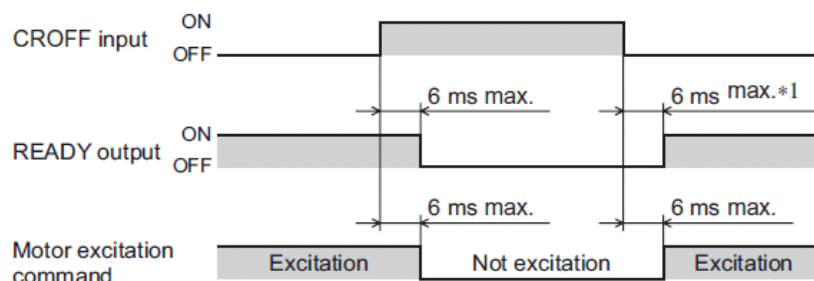


## ■ SC Output



## 22.4 Inputs

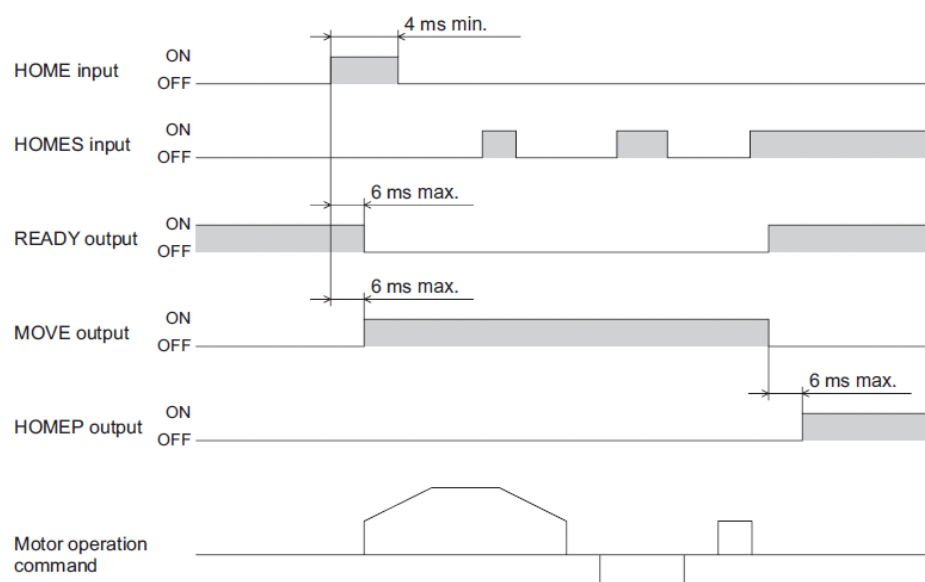
### ■ CROFF Input



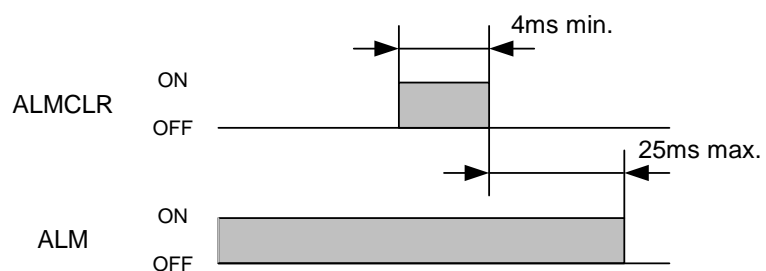
\*1 If STOEN (stepout detection enable) parameter is set to "enable," this period becomes 500 ms or less. If the parameter is set to "disable," the period becomes 6 ms or less.

### ■ HOME Input

Example: Return-to-home operation in the 3-sensor mode

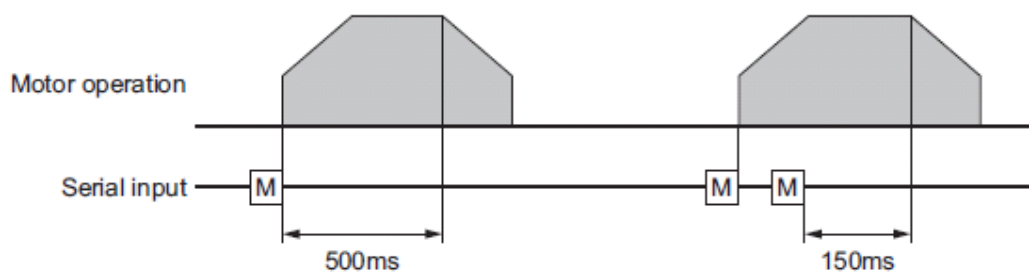
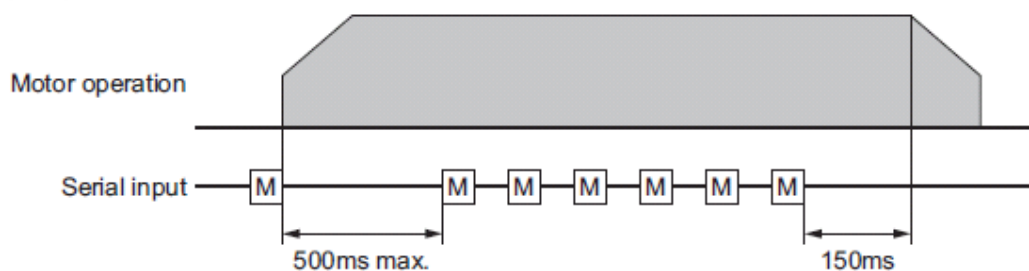


## ■ ALMCLR Input

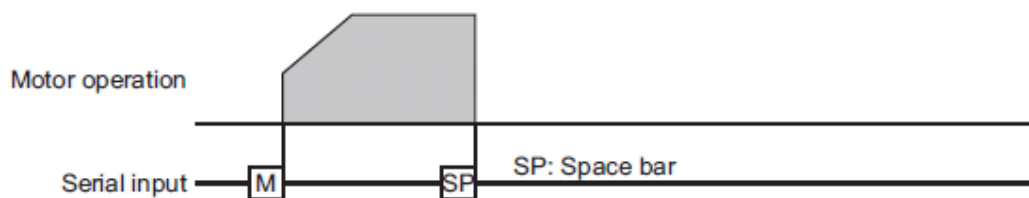


## 22.5 Teaching Operation

- When holding the key down  
**M** : CW scan input key on qwerty keyboard.



- To stop motion immediately



# 23 ASCII Data

This chapter explains ASCII data available in the CRK series built-in controller (stored program) device.

## ■ Abbreviation for ASCII Data

For convenience, following expressions are used in this manual.

	Description	ASCII Hex (Dec)
[BEL]	BELl	07h (7)
[BS]	Back Space	08h (8)
[LF]	Line Feed	04h (10)
[CR]	Carriage Return	0Dh (13)
[SP]	SPace	20h (32)
[ESC]	ESCape	1Bh (27)
[EOL]	End Of Line Any of the following combinations [CR] [LF] [CR] [LF] [LF] [CR]	

## ■ Valid ASCII Data for Serial Communication

Following are the values that can be entered from the terminal. Any other value is not accepted unless it is specified for specific features.

Receiving Data	Operation	
[20h] to [69h]	Input data buffer count is under 80:	Echo back entered value, store into input buffer.
	Input data buffer count is 80:	Send [BEL].
[EOL]	Start parsing commands. Clear input buffer.	
[BS]	Any data exist in input buffer:	Send [BS] [SP] [BS]. Clear the last data n input buffer.
	No data exist in input buffer.	Send [BEL].
[ESC]	Send [CR][LF][>]. Stop executing sequence program, stop motion. Clear input buffer.	

# 24 Command Format

This chapter shows the command format. Spaces between each word are accepted. Case {Upper/Lower} of the character does not matter unless specified. Decimal point number is accepted in some of the parameters.

**Memo** ¶ See “ASCII Data” on page 316 for abbreviation for ASCII data.

## Parameters

An “=” between a parameter and parameter value is required. If the parameter value is a constant, a space can be used instead of an “=”.

- Format

[Parameter] [=] [Parameter value]

[Parameter] [SP] [Parameter value (constant)]

- Examples

Condition	Example
Parameter value is constant	DIS=1234, DIS 1234
Parameter value is variable	DIS=A (Available in sequence only)
Parameter value is equation	DIS=A*□15 (Available in sequence only)

## Commands

Spacing between command and argument (if needed argument) by at least one space is required.

- Format

[Command] [SP] [Argument]

- Examples

Condition	Example
No parameter	MI
Parameter is constant	MA 1234
Parameter is variable	MA POS [1]
Parameter is string	RUN Test

## Multiple-Statements on a Line

Multiple statements can be written on a single line. A “;” (semicolon) divides each statement on the line. Spaces around semicolon are accepted. The maximum number of characters on a one line is 80.

- Example

>DIS 1234; VR 3; TA 0.5; TD 0.1; MI [EOL]

**Memo** ¶ VERBOSE parameter defines the response display. The following shows some examples.

VERBOSE=1 (default)	VERBOSE=0
>PC PC=123 >DIS=5678 DIS=5678 >HOMESEL HOMESEL=1	>PC 123 >DIS=5678 5678 >HOMESEL 1

**Note** ¶ Also the ECHO command defines the echo back ON/OFF for entered ASCII data. Default is ECHO=1 (ON) echo back. If ECHO=0 (OFF), there will no reply for the entered ASCII data. Display of parameter readout or SAS command from sequence is not affected by ECHO=, they are always displayed (See page 223 for SAS command).

- Unauthorized reproduction or copying of all or part of this manual is prohibited.  
If a new copy is required to replace an original manual that has been damaged or lost, please contact your nearest Oriental Motor branch or sales office.
- Oriental Motor shall not be liable whatsoever for any problems relating to industrial property rights arising from use of any information, circuit, equipment or device provided or referenced in this manual.
- Characteristics, specifications and dimensions are subject to change without notice.
- While we make every effort to offer accurate information in the manual, we welcome your input. Should you find unclear descriptions, errors or omissions, please contact the nearest sales office.
- ***Oriental motor*** is a registered trademark or trademark of Oriental Motor Co., Ltd., in Japan and other countries.  
Other product names and company names mentioned in this manual may be registered trademarks or trademarks of their respective companies and are hereby acknowledged. The third-party products mentioned in this manual are recommended products, and references to their names shall not be construed as any form of performance guarantee. Oriental Motor is not liable whatsoever for the performance of these third-party products.

© Copyright ORIENTAL MOTOR USA CORPORATION. 2011

• Please contact your nearest Oriental Motor office for further information.

ORIENTAL MOTOR U.S.A. CORP.  
1001 Knox St. Torrance, CA 90502  
Technical Support Tel:(800)468-3982  
8:30 A.M. to 5:00 P.M., P.S.T. (M-F)  
7:30 A.M. to 5:00 P.M., C.S.T. (M-F)  
[www.orientalmotor.com](http://www.orientalmotor.com)

ORIENTAL MOTOR DO BRASIL LTDA.  
Tel:+55-11-3266-6018  
[www.orientalmotor.com.br](http://www.orientalmotor.com.br)

ORIENTAL MOTOR (EUROPA) GmbH  
Headquarters Düsseldorf, Germany  
Technical Support Tel:00 800/22 55 66 22  
[www.orientalmotor.de](http://www.orientalmotor.de)

ORIENTAL MOTOR (UK) LTD.  
Tel:01256-347090  
[www.oriental-motor.co.uk](http://www.oriental-motor.co.uk)

ORIENTAL MOTOR (FRANCE) SARL  
Tel:01 47 86 97 50  
[www.orientalmotor.fr](http://www.orientalmotor.fr)

ORIENTAL MOTOR ITALIA s.r.l.  
Tel:02-93906346  
[www.orientalmotor.it](http://www.orientalmotor.it)

ORIENTAL MOTOR ASIA PACIFIC PTE. LTD.  
Singapore  
Tel:1800-8420280  
[www.orientalmotor.com.sg](http://www.orientalmotor.com.sg)

ORIENTAL MOTOR (MALAYSIA) SDN. BHD.  
Tel:1800-806161  
[www.orientalmotor.com.my](http://www.orientalmotor.com.my)

ORIENTAL MOTOR (THAILAND) CO., LTD.  
Tel:1800-888-881  
[www.orientalmotor.co.th](http://www.orientalmotor.co.th)

ORIENTAL MOTOR (INDIA) PVT. LTD.  
Tel:+91-80-41125586  
[www.orientalmotor.co.in](http://www.orientalmotor.co.in)

TAIWAN ORIENTAL MOTOR CO., LTD.  
Tel:0800-060708  
[www.orientalmotor.com.tw](http://www.orientalmotor.com.tw)

SHANGHAI ORIENTAL MOTOR CO., LTD.  
Tel:400-820-6516  
[www.orientalmotor.com.cn](http://www.orientalmotor.com.cn)

INA ORIENTAL MOTOR CO., LTD.  
Korea  
Tel:080-777-2042  
[www.inaom.co.kr](http://www.inaom.co.kr)

ORIENTAL MOTOR CO., LTD.  
Hong Kong Branch  
Tel:+852-2427-9800

ORIENTAL MOTOR CO., LTD.  
Headquarters Tokyo, Japan  
Tel:03-6744-0361  
[www.orientalmotor.co.jp](http://www.orientalmotor.co.jp)