**Oriental motor**

**Universal Controller**

# SCX10

# OPERATING MANUAL

C E

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.

This Operating Manual supports the firmware version 2.1x.

# Table of Contents

# 1  Before Use

## 1.1  Main Features

The **SCX10** universal controller is a compact and powerful programmable controller that can be connected with all Oriental Motor's standard pulse input drivers. The **SCX10** can also be connected to PLCs and computers by using variety of serial interfaces and I/Os.

### ■ Compact and Powerful

- Immediate and Program
  You may directly command each motion by a computer or PLC via serial ports, and immediately operate a motor. You may also store programs in the **SCX10**, and execute them via serial ports or I/Os. Up to 100 programs can be stored, and the motor will automatically be operated according to the stored sequence.
- Over 250 Commands
  The **SCX10** has various commands including motion control, I/O control, sequence control and status monitoring. For example, motion control is versatile with 6 types of stopping and pausing, 13 types of mechanical home seeking. Subroutines, math/logical operators and user variables are ready for sophisticated program creation.  With these, you can do almost anything you wish.
- Compact Size
  The **SCX10** is packed with powerful functions and many interfaces of a full-sized controller into a compact size. It can fit anywhere in your control panel.

### ■ Versatile Connections

- USB
  The **SCX10** has a USB port on the front panel. When you perform the initial setup of the **SCX10**, directly connect it to your computer.  Commercially available USB2.0 cables (mini-B type) are compatible.  This becomes an advantage during maintenance, since a special cable or converter is not required, unlike RS-485/232C products. USB can also be used for all functions, including network operation.
- CANopen
  The **SCX10** comes equipped with CANopen as standard.  Sequence selection and execution can be made through CANopen as well as commanding incremental motion.  CANopen for the **SCX10** is certified by CiA (CAN in Automation).
- RS-232C Daisy Chain (Multi Axis Control)
  While RS-232C is available as a standard connection to the master controller, daisy chain connection is also supported for multi axis control.  Other Oriental Motor products such as the *αSTEP*-**One** can be connected together.
- External Encoder Input
  The **SCX10** has external encoder input. Feedback position is always monitored, and can be used for general purpose as well as miss-step detection and mechanical home seeking (zero position).   The external encoder inputs are compatible with line driver, open collector and TTL signals. A 5V output for encoder power is also included.
- Configurable I/O
  While all connections to the drivers are included, the **SCX10** has 9 general inputs and 4 general outputs. Various commands such as start, stop, pause, end, brake and current can be assigned to any I/O point. Sequence programs can also be selected through these inputs.  Connect these I/O to PLC, switches, sensors and even other actuators. By using the powerful programming functions of the **SCX10**, simple systems can be configured without a PLC or computer.
- Differential Pulse Output
  The **SCX10** has photo-coupler compatible differential pulse outputs that can be connected with all Oriental Motor's standard pulse input drivers for less noise concerns in a variety of environments.

### ■ Simple Operations

- User Unit
  In the **SCX10**, actual motion distance of user application, such as "mm," "inch" and "revolution" is used, instead of pulse unit that is commonly used in pulse generates and motor controllers. The **SCX10** converts it to pulses for you. Any unit can be used.
- Friendly GUI
  While all commands for the **SCX10** can be executed using any general terminal software, a Windows based utility software, the **Immediate Motion Creator for CM/SCX Series (IMC)**, is provided. The **IMC** features include instant operation, easy programming and configuration without needing to know the **SCX10** commands. Real time monitoring of position and feedback, I/O status are also provided. Everything is intuitive. Once you install the **IMC** on your computer, you can make your desired motion in a few seconds.
- Three Mounting Options
  With it's standard configuration, the **SCX10** can be mounted on a DIN rail or a metal plate. If you take off the mounting plate, you may mount the **SCX10** onto a metal plate from the back side of it.

## 1.2 About Function Improvements

The **SCX10** is continuously improving functions and adding new features. The following lists show the major changes that have been made until now. Note that the utility software and the EDS file for CANopen are also changed in conjunction with updating the firmware of the main unit.

Added and changed functions with the firmware Ver.2.0x (Released in August to October, 2011)

| Feature | Content | Page |
|---|---|---|
| Reading the drivers' current position (absolute position) and home preset functions are added | 8.6 Driver Current Position Reading (**NX** Series driver, **ESMC** controller) | 100 |
| Torque limiting/push-motion operation/current cutback release functions are added | 8.5 Torque Limiting/Push-motion Operation (**NX** Series driver, **AR** Series driver) | 97 |
| | 12.3 Command Description<br>·TL: Torque Limiting/Push-motion Operation/Current Cutback Release | 350 |
| Limiting condition output (torque limiting /push-motion operation/current cutback) is added | 8.5 Torque Limiting/Push-motion Operation (**NX** Series driver, **AR** Series driver) | 97 |
| | 6.4.3 Output Signals<br>·LC (limiting condition) Output | 30 |
| | 6.5.4 Input Signals for Driver<br>·LC (limiting condition) Input | 39 |
| Sensor-less (push-motion type) home seeking operation is added | 8.2.5 Mechanical Home Seeking<br>·Sensor-less Mechanical Home Seeking Operation for "HOMETYP=12" | 87 |
| Driver operation ready (READY) /motor moving (MOVE) can be input | 6.4.3 Output Signals<br>·READY (operation ready) Output<br>·MOVE (motor moving) Output | 29<br>29 |
| Expanded the deviation counter clear selections during mechanical home seeking operation | 8.2.5 Mechanical Home Seeking<br>·HOMEDCL (deviation counter clear select at mechanical home seeking operation) | 86 |
| Stopping sequence in addition to motion, by PSTOP (panic stop) command or input (previously stopping motion only) | 6.4.2 Input Signals<br>·PSTOP (panic stop) Input | 25 |
| Name change of TIMING signal | See the following table* | - |
| Expansion of the applicable drivers and addition of the automated assignment function for the optimal driver I/O signals to each function/driver | 6.5.3 Change of Signal Assignment | 35 |
| Program memory is expanded to 6 kB from 2 kB | 15 Specifications<br>·"Programs" - "Size" | 372 |

∗ Name Change of TIMING Signal

| Function | Previous Firmware (Ver1.xx) | New Firmware (Ver.2.0x) |
|---|---|---|
| Assignment of the timing signal (differential) | DINTIM1 | DINTIMDEXTZ |
| Assignment of the timing signal (single-ended) | DINTIM2 | DINTIMS |
| Input status of the timing signal (differential) | DSIGTIM1 | DSIGTIMDEXTZ |
| Input status of the timing signal (single-ended) | DSIGTIM2 | DSIGTIMS |

The previous command names cannot be used in the new firmware. The functions of new commands remain the same.

Added and changed functions with the firmware Ver.2.10 (Released in October, 2012)

| Feature | Content | Page |
|---|---|---|
| The CON (current on) input has higher priority than the STRSW (current state at system start) parameter setting. | 6.4.2 Input Signals<br>・CON (current on) Input<br>12.3 Command Description<br>・STRSW : Current State at System Start | 27<br>343 |

| | | CON Input Signal | |
|---|---|---|---|
| | | OFF | ON |
| STRSW Parameter | 0 (OFF) | OFF | ON |
| | 1 (ON) | **OFF**<br>(Previously ON) | ON |

| Feature | Content | Page |
|---|---|---|
| Assignment of the resuming function from a pause status (CONT function) is added to I/O and remote I/O. | 6.4.2 Input Signals<br>・CONT (continue motion) Input | 26 |
| A choice for "no action when inputting SENSOR during continuous operation" (SENSORACT=3) is added to SENSORACT.<br>(In this choice, the SENSOR input is used with return-to-mechanical home operation only.) | 8.2.3 Continuous Motions<br>8.2.5 Mechanical Home Seeking | 79<br>83 |
| Sensor-less mechanical home seeking operation (HOMETYP=12) can be performed in the sequence, and also the MEND function can be used with HOMETYP=12. | 8.2.5 Mechanical Home Seeking<br>・Sensor-less Mechanical Home Seeking Operation for "HOMETYP=12"<br>12.3 Command Description<br>・MEND: Wait for Motion End | 87<br>284 |
| PABS, ABSSTS, DD, DIN, DOUT, DINSG, DOUTSG are added to the function that can be used via CANopen, and also PF becomes writable. | 10.7 Object Dictionary (SDO) | 149 |
| The encoder count (EC) becomes writable, and MAXEC as the maximum encoder count value is added. | 8.9 Encoder Function | 106 |
| ENDWAIT is added as the setting function of the waiting time for the motion end signal (END). | 8.8 END (motion end) Signal | 105 |

■ **Be Sure to Use the Latest Version of Utility Software Immediate Motion Creator for CM/SCX Series (IMC)**

Since the functions of the **SCX10** were expanded, the previous version of the **IMC** cannot be used. If the previous version of the **IMC** was installed in your PC, update it by using the CD-ROM provided. The new version of the **IMC** (Ver.2.10 or later) can be used with the previous versions of the **SCX10** firmware (Ver.1.xx to 2.02).

- To check the version of the **IMC** and the firmware version of the **SCX10**, use the "Help" - "Version Information" (or "About" for previous version) function of the **IMC**. (When checking the firmware version, it is required to connect the **SCX10** to your PC.)
- To update to the latest version of **IMC**, do so only when the previous version of the **IMC** is not running.
- The latest version of the **IMC** can be downloaded from the Oriental Motor Website. Check by the function for "Help" - "Check New Version" (or "Check Version") of the **IMC**.
- The latest **IMC** (Ver. 2.10 or later) does not support the Windows 2000 since the considerable period has past after the Microsoft discontinued to support it. Windows 8, Windows 7, Windows Vista and Windows XP are continuously supported. Windows 8 is also supported with the **IMC** Ver. 2.11 or later.

■ **Be Sure to Use the Correct Version of CANopen EDS Files**

The version of the EDS file that can be used depends on the firmware of the main unit. If the wrong combination is used, there may be the functions that cannot be used or the product may not work properly. Be sure to use the EDS file with the version that is applicable.

The following table is the combinations of the firmware and EDS file for CANopen.

| Firmware | Applicable EDS files |
|----------|---------------------|
| 1.xx | SCX10.eds |
| 2.0x | SCX10_2_1.eds |
| 2.1x | SCX10_3_0.eds |

## 1.3  System Configuration

A sample system configuration using the **SCX10** is provided below.
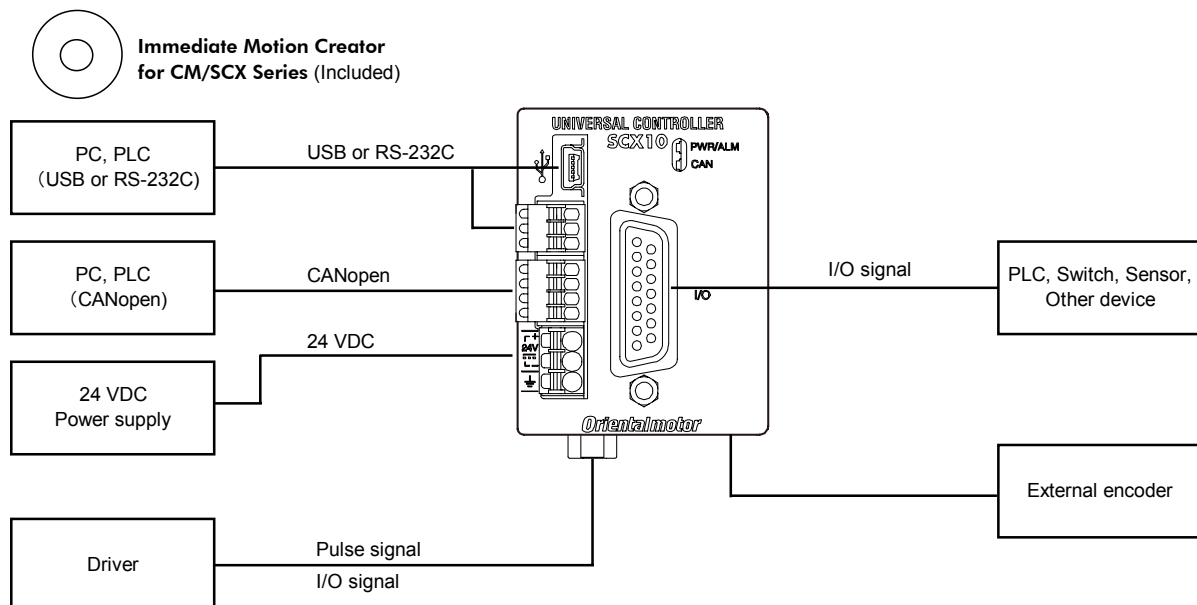
# 1.4 Operating Methods

There are two ways to make the motor move, immediate command and program execution.

## ■ Immediate Command

Operate the motor by sending each command immediately from the master controller such as a computer or PLC via RS-232C, USB or CANopen. See "7 Start Up (Immediate Command)" on page 69 for more detail.
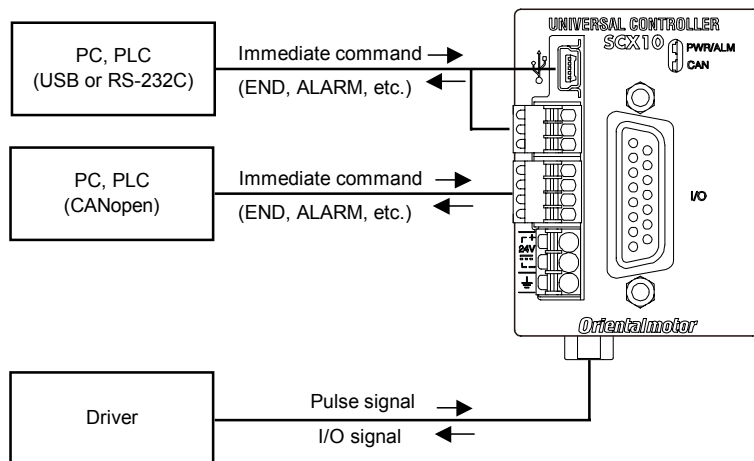


## ■ Program Execution

Create sequences using a computer, save the programs into the built-in memory of the **SCX10**, specify which sequence name or number to run, and input the start signal to execute the sequence. The program creation is made via USB or RS-232C, the program selection and execution are made via USB, RS-232C, CANopen or I/O selection. See "9 Program Creation and Execution" on page 116.
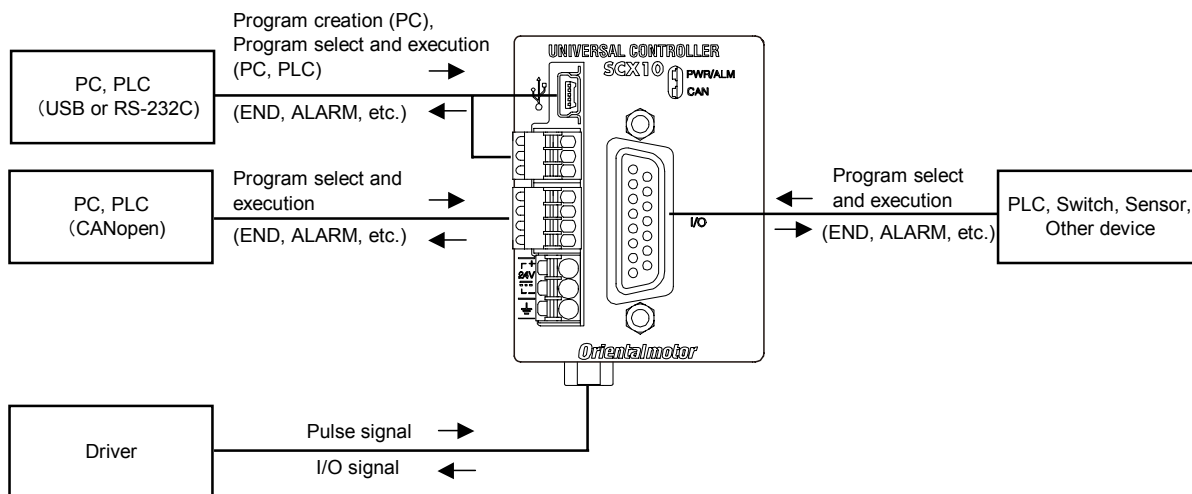


|  | Immediate Command | Program Creation | Program Select and Execution |
|---|---|---|---|
| USB | ✓ | ✓ | ✓ |
| RS-232C | ✓ | ✓ | ✓ |
| CANopen | ✓ | - | ✓ |
| I/O | - | - | ✓ |

**Note**
- USB and RS-232C cannot be connected and used at the same time.
- Do not use two or more masters (ex. a CANopen master and a RS-232C master) at the same time to avoid confusion.

**Memo** USB or RS-232C can be used for monitor and maintenance purposes while the **SCX10** is being operated by CANopen master.

## 1.5 Immediate Motion Creator for CM/SCX Series

General terminal software programs (ex. Windows HyperTerminal) can be used for all commands. If you install the exclusive utility software, **Immediate Motion Creator for CM/SCX Series** on your PC, just clicking your mouse can do test operations, program creation, position teaching, parameter setup and I/O configuration. Real time monitoring of position and feedback, along with I/O status and alarm status are also provided.

## 1.6 Standards and CE Marking

The CE Marking (EMC Directive) is affixed to the product is accordance with EN standards.

### ■ Installation Conditions (EN Standard)

This product is to be used as a component within other equipment.
Overvoltage category: Ⅰ
Pollution degree: Ⅱ
Protection against electric shock: Class Ⅲ

### ■ For EMC Directive

This product has received EMC compliance under the conditions specified in "5.3 Installing and Wiring in Compliance with EMC Directive" on page 16. The compliance of the final machinery with the EMC Directive will depend on such factors as the configuration, wiring, layout and risk involved in the control-system equipment and electrical parts. It therefore must be verified through EMC measures by the customer of the machinery.

Applicable Standards

| EMI | Emission Tests | EN61000-6-4 |
| | Radiated Emission Test | EN55011 Group1 ClassA |
| EMS | Immunity Tests | EN61000-6-2 |
| | Radiation Field Immunity Test | EN61000-4-3 |
| | Electrostatic Discharge Immunity Test | EN61000-4-2 |
| | Fast Transient / Burst Immunity Test | EN61000-4-4 |
| | Conductive Noise Immunity Test | EN61000-4-6 |

### ■ Hazardous Substance

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

# 2  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.

| ⚠ Warning | Handling the product without observing the instructions that accompany a "Warning" symbol may result in serious injury or death. |
|---|---|
| ⚠ Caution | Handling the product without observing the instructions that accompany a "Caution" symbol may result in injury or property damage. |
| **Note** | The items under this heading contain important handling instructions that the user should observe to ensure safe use of the product. |
| ⋮ Memo | This contains information relative to the description provided in the main text. |

---

⚠ 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 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 or injury.
- Do not transport, install the product, perform connections or inspections when the power is on.
  Always turn the power off before carrying out these operations. Failure to do so may result in electric shock.
- When the device's protective function is triggered, first remove the cause and then clear the protective function. Continuing the operation without determining the cause of the problem may cause malfunction of the device, leading to injury or damage to equipment.

## Installation

- Install the device in an enclosure in order to prevent injury.

## Connection

- Keep the device's input-power voltage within the specified range to avoid fire.
- For the device'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 the cables securely according to the wiring diagram in order to prevent fire.

## Operation

- Turn off the device 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.

## Repair, Disassembly and Modification

- Do not disassemble or modify the device. 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 device beyond its specifications, or injury or damage to equipment may result.

## Transportation

- Do not hold the device cable. This may cause damage or injury.

## Installation

- Keep the area around the device free of combustible materials in order to prevent fire or a skin burn (s).

## Conneciton

- When grounding the positive terminal of the power supply, do not connect any equipment (PC, etc.) whose negative terminal is grounded. Doing so may cause the driver and PC to short, damaging both.

## Operation

- To avoid injury, remain alert during operation so that the device can be stopped immediately in an emergency.
- Before supplying power to the device, turn all start inputs to the device to "OFF." Otherwise, the device may start suddenly and cause injury or damage to equipment.
- When an abnormality is noted, stop the operation immediately, or fire or injury may occur.

## Disposal

- When disposing of the device, treat it as ordinary industrial waste.

# 3  Precautions for Use

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

## ■ Preventing Electrical Noise

See "5.3 Installing and Wiring in Compliance with EMC Directive" on page 16 for measures with regard to noise.

## ■ EEPROM Write Cycle

Do not turn off the 24 VDC power supply while data is being written to the EEPROM and 5 seconds after the completion of 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.

# 4  Preparation

## 4.1  Checking the Product

- Universal controller (**SCX10**)                          1 unit
- CD-ROM                                                     1 pc.
  (**Immediate Motion Creator for CM/SCX Series** (utility software),
  Startup manual, Operating manual, CANopen EDS file,
  USB driver, .NET Framework 2.0)
- Connector set                                             1 set (packed in a bag)
  RS-232C connector (3 pins): 1
  CANopen connector (4 pins): 1
  Power connector (3 pins): 1
- Encoder connector housing/contact (8 pins)               1 set (packed in a bag)
- Startup manual                                            1 copy

## 4.2  Names and Functions of Parts

Driver I/O sink/source SW

Driver I/O 5/24 V SW
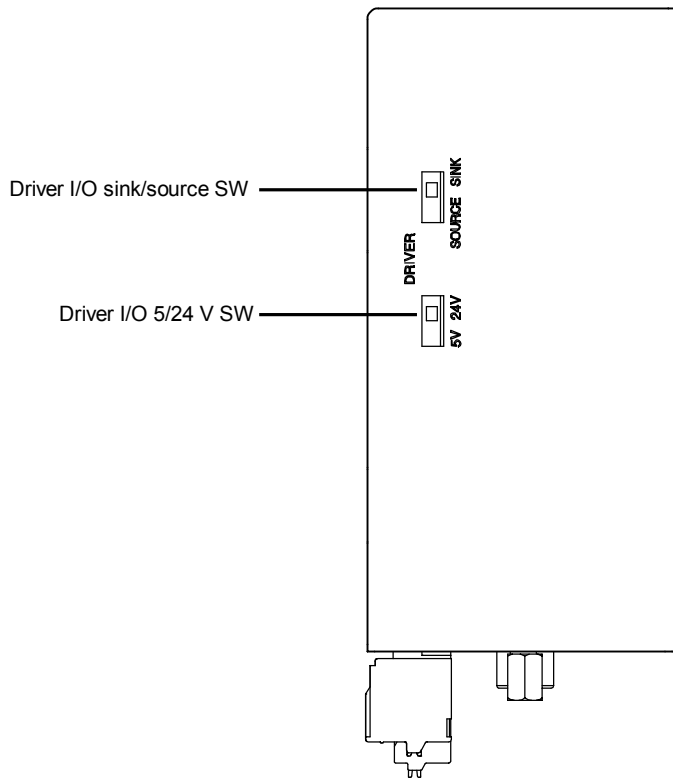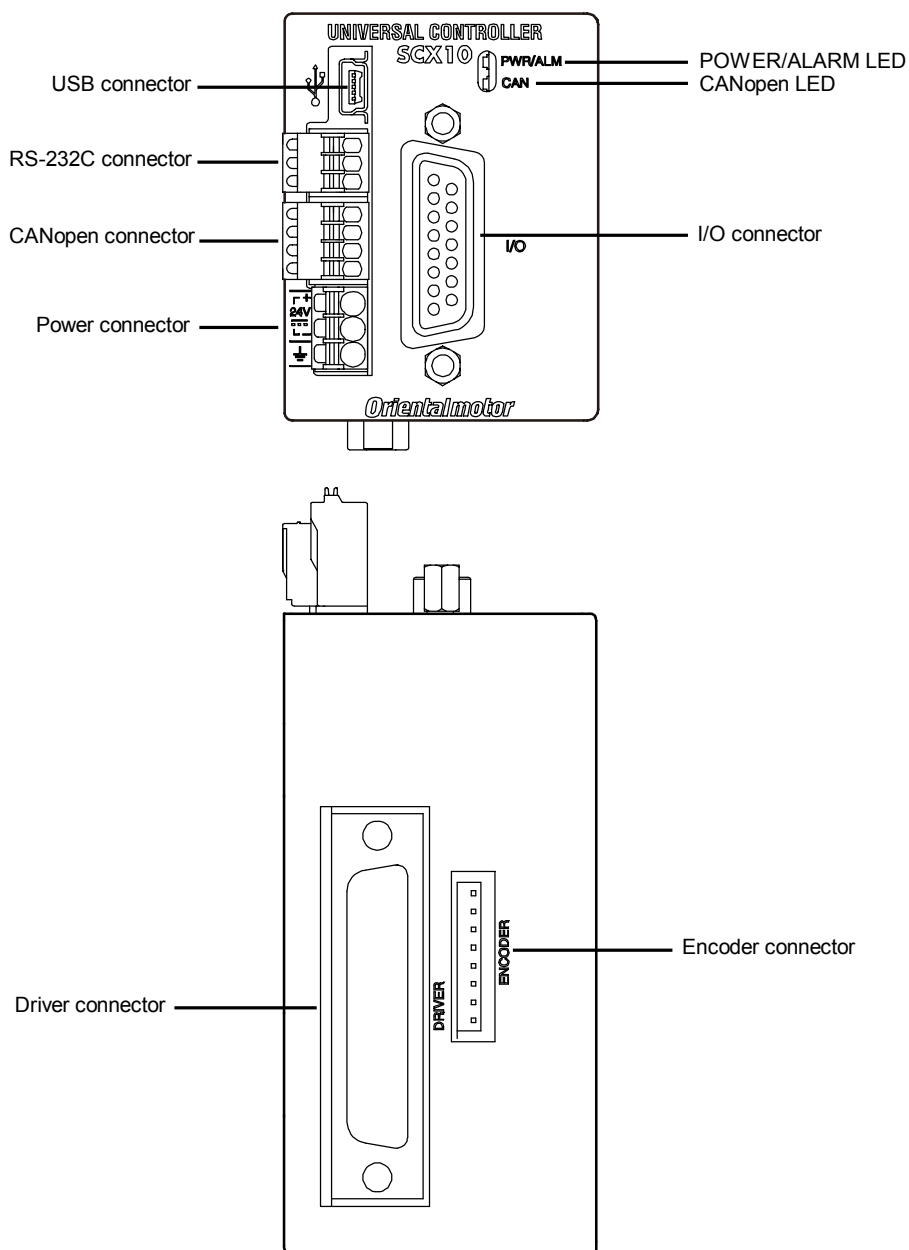
| Name | Description |
|---|---|
| POWER/ALARM LED (green/red) | Green: Lit when the power is on.<br>Red: The LED blinks when a protective function is triggered. The cause triggering the protective function can be identified by the number of blinks the LED emits. See "13 Troubleshooting" on page 367. |
| CANopen LED (green/red) | Green: Run<br>Red: Error (See "10.3 LED Indication" on page 133 for detail.) |
| Power connector | Connects to the power supply cable |
| I/O connector | Connects to the sensors, switches and/or master controller |
| RS-232C connector | Connects to the RS-232C cable |
| USB connector | Connects to the USB2.0 cable (mini-B type) |
| CANopen connector | Connects to the CANopen cable |
| Encoder connector | Connects to the external encoder |
| Driver connector | Connects to the driver |
| Driver I/O sink/source SW | Set the logic of the driver connector |
| Driver I/O 5/24 V SW | Set the voltage of the driver connector |

# 5  Installation

## 5.1  How to Install the SCX10

You can install the **SCX10** by one of the three following methods.

1. Mounting to a DIN rail
   a. Loosen the M4 screw on the **SCX10**, using a screwdriver.
   b. Press the hook under the M4 screw onto the DIN rail, and push the upper hook of the **SCX10** onto the DIN rail.
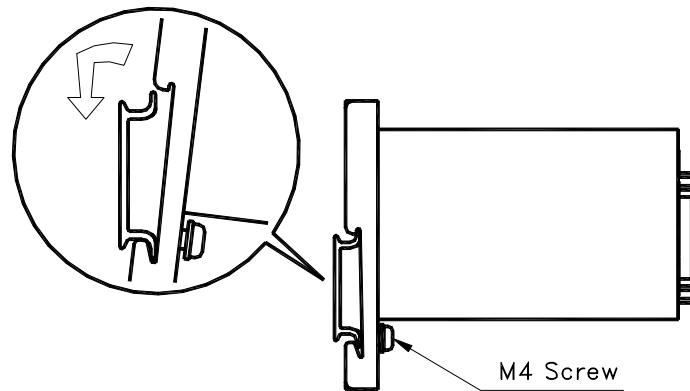


   c. Tighten the M4 screw on the **SCX10**, using a screwdriver.
      * Tightening torque: 1.4 N・m
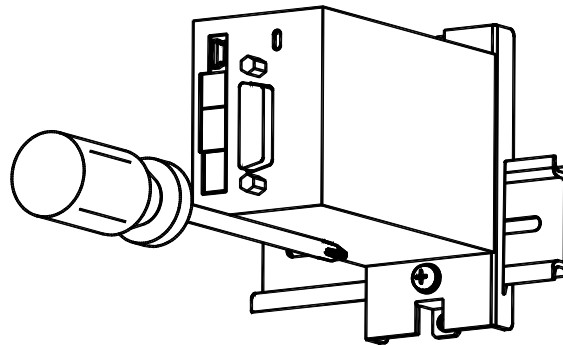
## 2. Mounting to a plate (Installation method A)

Secure the controller mounting bracket to a plate using two screws (M4, not included).
∗ Tightening torque: 1.4 N・m
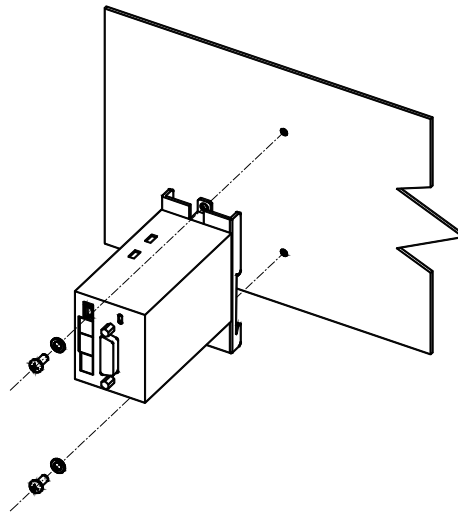
## 3. Mounting to a plate (Installation method B)

Remove the mounting bracket from the **SCX10**.
Secure the **SCX10** to a plate from the backside using three screws (M3, not included).
The thickness of the **SCX10** housing = effective depth of screw holes is 2 mm. Do not use screws with a length that will allow them to enter the **SCX10** more than 4 mm (including the 2 mm housing thickness) from the back. The use of longer screws can short-circuit in the **SCX10**.
∗ Tightening torque: 0.5N・m
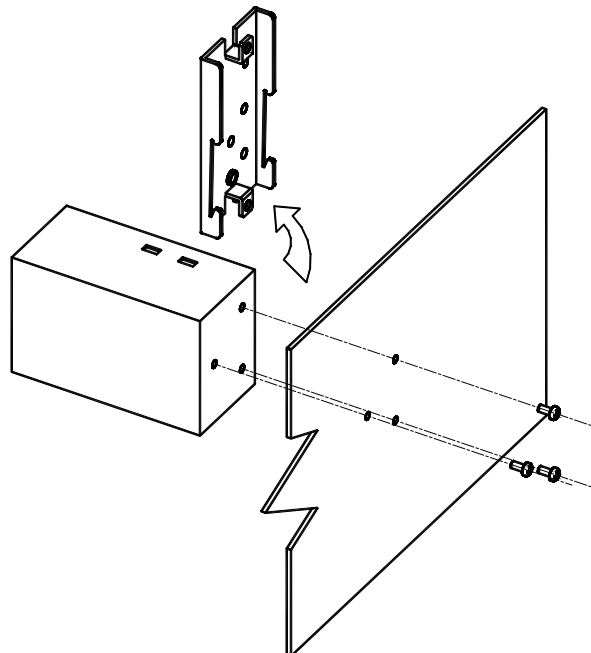
## 5.2   Installing the Driver

Refer to the driver's Operating Manual and install it at an appropriate distance from other equipment

## 5.3   Installing and Wiring in Compliance with EMC Directive

This device has been designed and manufactured for incorporation in general industrial machinery. The EMC Directive requires that the equipment incorporating this product comply with the directive.
The installation and wiring method for the motor and device are the basic methods that would effectively allow the customer's equipment to be compliant with the EMC Directive.
The compliance of the final machinery with the EMC Directive will depend on such factors as configuration, wiring, layout and risk involved in the control-system equipment and electrical parts. It therefore must be verified through EMC measures by the customer of the machinery. For the EMC Directives, see "1.6 Standards and CE Marking" on page 8.

### ■ Connecting Mains Filter for Power Source Line

Install a mains filter on the input side of the DC power supply in order to prevent the noise generated within the driver from propagating outside via the DC power-source line.

- Install the mains filter as close to the AC input terminal of the DC power source as possible, and use cable clamps and other means to secure the input and output cables (AWG18: 0.75 $mm^2$ or more) firmly to the surface of the enclosure.
- Connect the ground terminal of the mains filter to the grounding point, using as thick and short a wire as possible.
- Do not place the AC input cable (AWG18: 0.75 $mm^2$ or more) parallel with the mains filter output cable (AWG18: 0.75 $mm^2$ or more). Parallel placement will reduce mains filter effectiveness if the enclosure's internal noise is directly coupled to the power-supply cable by means of stray capacitance.

### ■ Connecting the 24 VDC Power Supply

Use a 24 VDC power supply conforming to the EMC Directive.
Use a shielded cable for wiring and wire/ground the 24 VDC power supply over the shortest possible distance. Refer to "Wiring the Power Supply Cable and I/O Signals Cable" below for how to ground the shielded cable.

### ■ How to Ground

The cable used to ground the driver, motor and mains 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.

### ■ Wiring the Power Supply Cable and I/O Signals Cable

Use a shielded cable of AWG24 (0.2 $mm^2$) or more for the power supply cable and signal cable, and keep it as short as possible.
To ground a shielded cable, use a metal clamp or similar device that will maintain contact with the entire circumference of the shielded cable. Attach a cable clamp as close to the end of the cable as possible, and connect it as shown in the figure.



Shield cable

Cable clamp
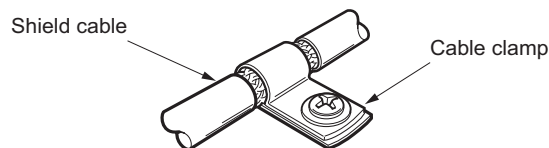
## ■ Notes about Installation and Wiring

- 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 mains 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. If they have to cross, cross them at a right angle. Place the AC input cable and output cable of a mains filter separately from each other.

> **Note** Be sure to connect the protective earth lead wire of the motor cable to the protective earth terminal of the driver. If not connected, an error via USB communication may occur.

## ■ Example of SCX10 Module and Driver Installation and Wiring
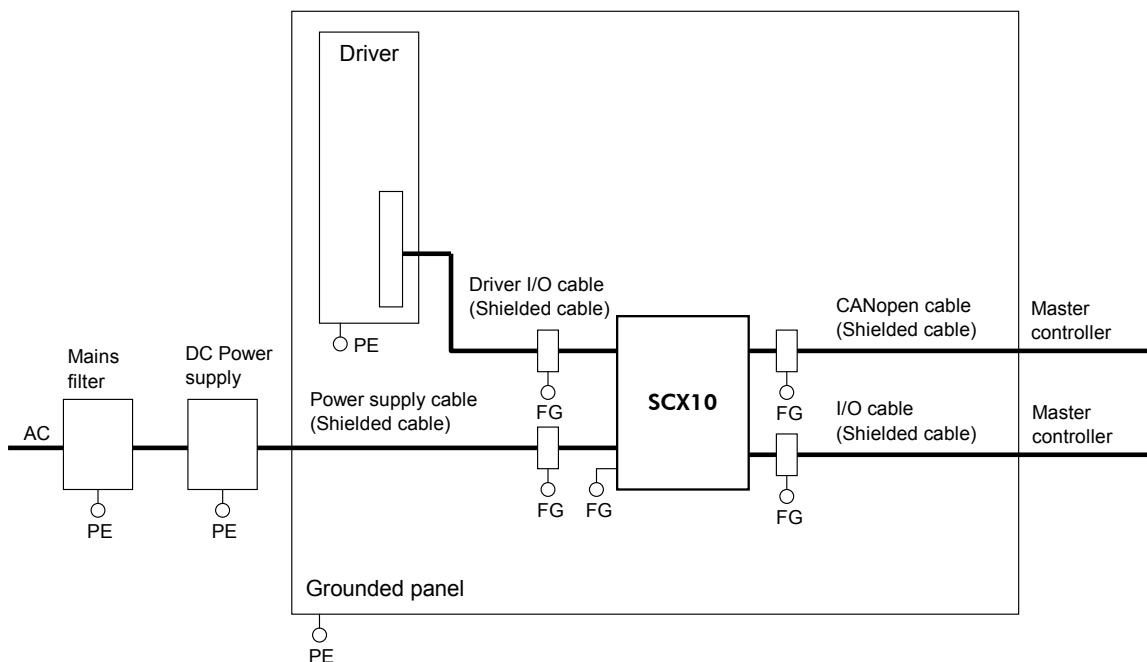


## ■ Precautions about Static Electricity

Static electricity may cause the **SCX10** to malfunction or suffer damage. While the **SCX10** is receiving power, handle the **SCX10** with care and do not come near or touch the **SCX10**.

> **Note** The **SCX10** uses parts that are sensitive to electrostatic charge. Before touching the **SCX10**, turn off the power to prevent electrostatic charge from generating.
> If an electrostatic charge is impressed on the **SCX10**, the **SCX10** may be damaged.

# 6 Connection

This chapter explains the methods for connecting to the computer, PLC, sensors, switches, external encoder, and the power supply, as well as the grounding method, connection examples and control inputs/outputs.

## 6.1 Overview

### ■ System Configuration

A sample system configuration using the **SCX10** is provided below.
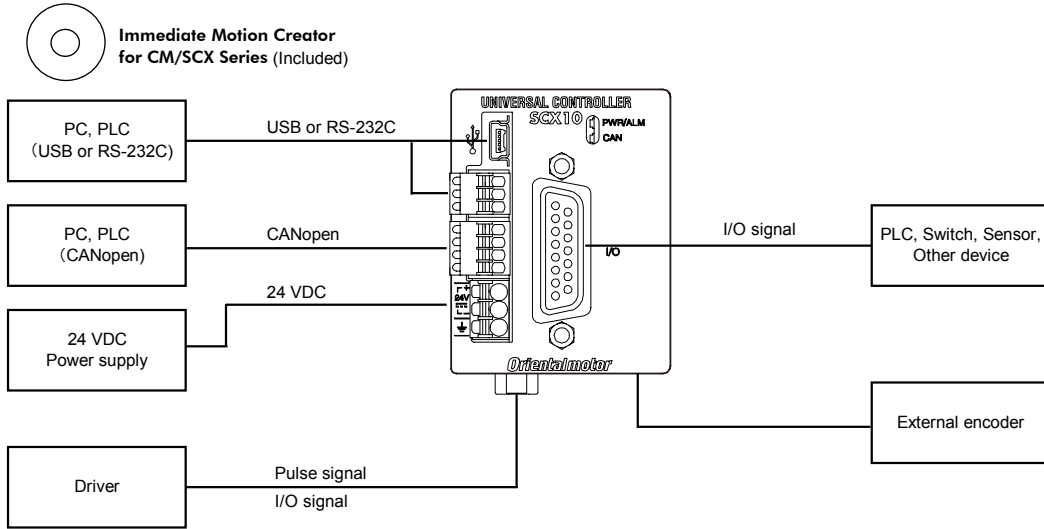


> **Memo**
> - Making all connections is not necessary. Choose the necessary connections according to your needs by referring the contents and the chart below.
> - For the initial set up, only a power supply and a computer (USB or RS-232C connection) are required. See "7 Start Up (Immediate Command)" on page 69.

### ■ Contents

6.2 Connecting the Power Supply
6.3 Connecting the USB and Installation of Utility Software
6.4 Connecting the I/O Signals
   6.4.1 Pin Assignments
   6.4.2 Input Signals
   6.4.3 Output Signals
   6.4.4 Connection Example of I/O
6.5 Connecting the Driver
   6.5.1 I/O Voltage and Logic
   6.5.2 Signal Assignments
   6.5.3 Change of Signal Assignment
   6.5.4 Input Signals for Driver
   6.5.5 Output Signals for Driver
   6.5.6 Connection Example of Driver
6.6 Connecting the RS-232C
6.7 Connecting the CANopen
6.8 Connecting the External Encoder

Interface and Availability

| | Immediate Command | Program Creation | Program Select and Execution |
|---|---|---|---|
| USB | ✓ | ✓ | ✓ |
| RS-232C | ✓ | ✓ | ✓ |
| CANopen | ✓ | - | ✓ |
| I/O | - | - | ✓ |

## 6.2 Connecting the Power Supply

### ■ Connecting to the Power Supply

Use the power supply connector (3 pins) to connect the power supply cable (AWG24 to 16: 0.2 to 1.5 mm$^2$) to the main power supply connector on the **SCX10**.

### ■ Grounding SCX10

Ground the driver's Frame Ground Terminal (FG) as necessary.
Use a grounding wire of a size equivalent to or larger than the power-supply cable (AWG24 to 16: 0.14 to 1.5 mm$^2$).
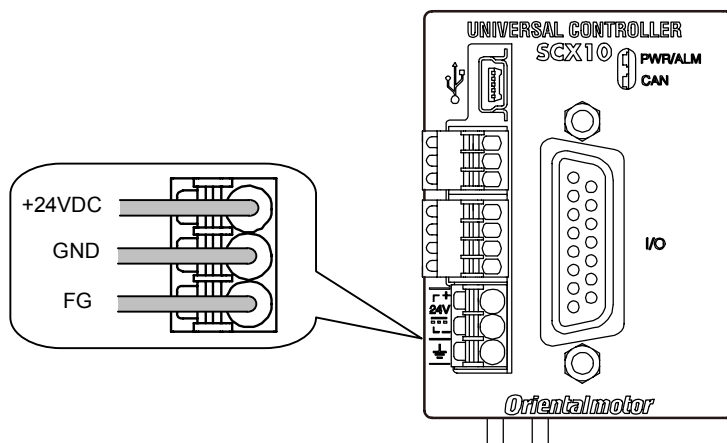


Applicable Lead Wire

| Connector | FMC 1,5/3-ST-3,5 (PHOENIX CONTACT) |
|---|---|
| Applicable lead wire size | AWG24 to 16 (0.2 to 1.5 mm$^2$) |

### ■ Connection Method

1. Strip the lead wire insulation by 10 mm.

2. Push the spring (orange) of the connector with a flat-tip screwdriver, to open a terminal port.
   Recommended flat-tip screwdriver: a tip of 2.5 mm in width, 0.4 mm in thickness

3. Insert the cable while pushing down the flat-tip screwdriver.

4. Release the flat-tip screwdriver. The lead wire will be attached.

## 6.3  Connecting the USB and Installation of Utility Software

The USB connection can be used for all the operations including initial setup, test operation, program creation, I/O configuration and real time monitoring, using general terminal software or supplied utility software as well as user program. Everything you can perform via USB can also be performed via RS-232C.

### ■ Specification

∗ The USB on the **SCX10** talks to the virtual COM port on the computer.

| Item | Description |
|---|---|
| Electrical characteristics | In conformance with USB2.0 (Full Speed) |
| Transmission method | Start-stop synchronous method, NRZ (Non-Return Zero), full-duplex |
| Data length | 8 bits, 1 stop bit, no parity |
| Transmission speed | 9600, 19200, 38400, 57600, 115200 bps (9600 is factory setting.) Selected by the USBBAUD parameter |
| Protocol | TTY (CR+LF) |

Terminal Specification
- ASCII mode
- VT100 compatible recommended
- Handshake: None
- Transmission CR: C-R
- Word wrap: None
- Local echo: None
- Beep sound: ON

Conncector
- USB mini-B

| Note | Be aware that Windows automatically changes the COM port number when a **SCX10** is replaced. |

| Memo | Generally, the maximum number of COM ports in a Windows PC is 256. Since the COM port number on a PC increases every time different **SCX10** is connected via USB, setting data to more than 256 pieces of **SCX10** cannot be accomplished using a PC. When it is required such as for mass production, use RS-232C connection or USB to RS-232C converter so as to be RS-232C connection on the **SCX10**. |

### ■ USB Driver Installation (Other than Windows 8)

Insert the supplied CD-ROM into the CD-ROM drive of the computer, power on the **SCX10** and connect to a USB port using a mini-B cable. (Prepare a commercially available USB2.0 cable (mini-B type). A cable with ferrite cores that has the effect of exogenous noise suppression is recommended.) You will then be asked to install the USB driver. See the procedure according to the type of Windows as follows.

∗ In the case of Windows 8, a little while after **SCX10** is powered on and connected to a USB port using a mini-B cable, USB driver is installed automatically.

**Windows 7:**
1. Open "Devices and Printers" in the Control Panel.
2. Right click on "FT232R USB UART" and select "Update Driver Software."
3. Select "Browse my computer for driver software."
4. Click "Browse" and select the applicable CD-ROM drive, check the box next to "Include subfolders" and Click "Next."
5. After successful installation, click "Close."
6. Go back to the Device Manager, right click on "USB Serial Port" and select "Update Driver Software." Repeat same procedure as the above FT232R USB UART installation.

**Windows Vista:**
1. The installation of the FT232R USB UART is asked by Windows when the **SCX10** is connected. Select "Locate and install driver software," and click "Next." After successful installation, click "Close."
2. The installation of the USB Serial Port is then asked for by Windows.
3. Click "Next." After successful installation, click "Close."

**Windows XP:**
1. The installation of the FT232R USB UART is asked for by Windows when the **SCX10** is connected. Select "Install the software automatically," and click "Next." After successful installation, click "Finish."
2. The installation of the USB Serial Port is then asked for by Windows. Select "Install the software automatically," and click "Next."
3. After successful installation, click "Finish."

## ■ Installation of Utility Software "Immediate Motion Creator for CM/SCX Series (IMC)"

While all commands for the **SCX10** can be made using general terminal software, the supplied Windows based utility software, **Immediate Motion Creator for CM/SCX Series (IMC)**, gives you instant operation, easy programming and configuration without having to know any commands. Since important settings and functions (such as I/O assignment, automatic setting of the driver I/O, etc.) are included, it is recommended to use the **IMC**.

Functions:
- Motion Creator: Select motion type, put desired values in and click the start button to begin motions instantly.
- Program Editor: Double click listed commands in a desired order to create a sequence program, and click a button to save it to the device or your PC.
- Terminal: Use as a general terminal software. Most commands can be used by typing.
- Teach/Jog: Move motors and store positions. Stored positions can be used for programmed motions.
- System Config: Indicate and Set system parameters and I/O assignments
- Real Time Monitor: I/O, Alarms (including history), Busy, Motor Position and Encoder Position

- System Requirements

   - Windows 8, Windows 7, Windows Vista, Windows XP SP2 or later
   - .NET Framework 2.0 - 3.5
   - SVGA monitor 800 x 600 or greater
   - USB or RS-232C port
   - CD-ROM drive

- Installation and Uninstallation

   Insert the supplied CD-ROM into your CD-ROM drive. Open the Explorer, select the applicable CD-ROM drive, open the IMC folder, double click on "setup.exe" and follow the on screen instructions.
   To uninstall, use the "Add/Remove Programs" function in the Windows Control Panel.

   | Note | When updating the installed version of the **IMC**, do so when the existing **IMC** is not running. |
   |---|---|

- .NET Framework Installation (For Windows 8 and Windows XP)

   **IMC** runs on the Microsoft .NET Framework 2.0-3.5 and they are not automatically installed with Windows 8 and Windows XP. Unless you have separately installed them, install them as instructed below.

   (The .NET Framework 3.5 is normally installed with Windows 7 and the .NET Framework 3.0 is normally installed with Windows Vista.) Visit the Microsoft .NET Framework website if detailed information is required.

   **Windows 8:**

   When you first launch the **IMC** after it is installed, the message prompting you to install the .NET Framework 3.5 is appeared. Install it by following the instructions appeared. (An Internet connection is required.)

   **Windows XP:**

   Install the .NET Framework 2.0 software that is on the supplied CD-ROM, under the DotNet_Framework2_0 folder.

- Start

   1. Connect the **SCX10** by USB2.0 cable (mini-B type)
   2. Power on the **SCX10**
   3. Click "Start" – "All Programs"- "ORIENTAL MOTOR" – "IMC for CM SCX" – "Immediate Motion Creator for CM SCX Series." The COM port selection window will appear.
   4. Select the COM port that is connected to the **SCX10**. The **IMC** will be launched.

   The **IMC** is made to be intuitive to use. For instructions, refer to the HELP menu in the **IMC**, as necessary.
   Right click on the **IMC** screen and select [Help] will launch the **IMC** Help with the description of function that the mouse cursor is on.

- Update

  After installation, click "Help" - "Check New Version" on the pull down menu with the Internet connection. If a newer version of this software is available, continue to the download and update actions.

- About VERBOSE and ECHO

  The **IMC** may alter automatically the ECHO (Echo ON/OFF) and VERBOSE (respond with data and description/respond with data only) parameters of the **SCX10** for communications and ease of use. The ECHO and VERBOSE parameters cannot be set in the System Config window on the **IMC**. When changing the ECHO and/or VERBOSE setting is required, follow the procedure below.

  1. Turn on the power to the **SCX10**
     (If the message of reconnection attempt failure has shown while using the **IMC**, first turn off the power, wait for a few seconds and restart.)
  2. Launch the **IMC** software
  3. Click the [Terminal] tab
  4. Type "ECHO=0" or "ECHO=1", press the Enter key and "VERBOSE=0" or "VERBOSE=1", then press the Enter key (both parameters must be typed)
  5. Type "SAVEPRM" and press the Enter key, then type "Y" and press the Enter key
  6. After "OK" is indicated, exit the **IMC**  (Do not operate other functions before exiting)
  7. Turn off the power to the **SCX10** for the new settings to become effective

## ■ Setting the Baud Rate

Since the USB on the **SCX10** talks to the virtual COM port on the computer, the baud rate for the COM port and the baud rate for the **SCX10** need to be the same.
The default USB baud rate of the **SCX10** is 9600 bps, same as the default baud rate of a general Windows computer. If the baud rate on the computer or the **SCX10** is changed, the baud rate must also be changed on the other.

- When Using the **IMC**, Provided Utility Software

  Use of the highest baud rate, 115200 bps is recommended if there is no problem for the usage environment. Set it to the **SCX10** according to the following steps.

  1. Turn on the power for the **SCX10** and connect to the computer.
  2. Launch the **IMC**.
  3. Click the [System Config] tab.
  4. Click the "USB Baud rate" located at the upper-center, and select "115200 bps" from the drop-down list.
  5. Click the [SAVE and RESET] to enable the change. At that time, the **IMC** also change the baud rate of computer side to 115200 bps.

  The setting is completed. The **SCX10** is already communicating with your computer at 115200 bps.

  > **Note** When starting communication with the **SCX10**, be sure to set the baud rate of the **IMC** to the same baud rate that has been set to the **SCX10**. If you are unsure about the baud rate of the **SCX10**, use the [Scan Baud Rate] button on the Serial Port Settings window. If the wrong baud rate has been set in the Serial Port Settings window, not only will the communication not be established, but it may also be possible that the communication will never be established even if the baud rate is correctly set afterwards. If this communication problem occurs, turn off the power to the **SCX10**, wait for a few seconds and restart. Take the same action any time when communication is likely to be disconnected.

- When Using Other Software than the **IMC**

  - The Computer:
    Check the baud rate of the computer application that is used to communicate with the **SCX10**, or check the COM port property of Windows if the application does not have the baud rate function.
  - The **SCX10**:
    The USBBAUD is the command used to change the baud rate for the USB connection.  Always set the **SCX10** baud rate first, then set the baud rate on the master to the same baud rate.

## 6.4  Connecting the I/O Signals

Connect the PLC, switch, sensor etc. to the I/O connector (D-sub connector on the front panel of the **SCX10**).

## 6.4.1 Pin Assignments

At the time of shipment, any signals that have specific functions are not assigned to the I/O connector, which functions as general input "IN1 to IN9" and general output "OUT1 to OUT4." As necessary, assign signals and connect accordingly (The connector is not supplied. Provide 15 Pin D-Sub connector separately). Those become "system xxx signal."

### ■ Connector Function Table

See the following pin assignments for a solder type connector.

| Description | Signal | Pin No. |
|---|---|---|
| Input common | IN-COM | 1 |
| General input | IN2 | 2 |
| General input | IN4 | 3 |
| General input | IN6 | 4 |
| General input | IN8 | 5 |
| General output | OUT1 | 6 |
| General output | OUT3 | 7 |
| Output common | OUT-COM | 8 |

| Pin No. | Signal | Description |
|---|---|---|
| 9 | IN1 | General input |
| 10 | IN3 | General input |
| 11 | IN5 | General input |
| 12 | IN7 | General input |
| 13 | IN9 | General input |
| 14 | OUT2 | General output |
| 15 | OUT4 | General output |
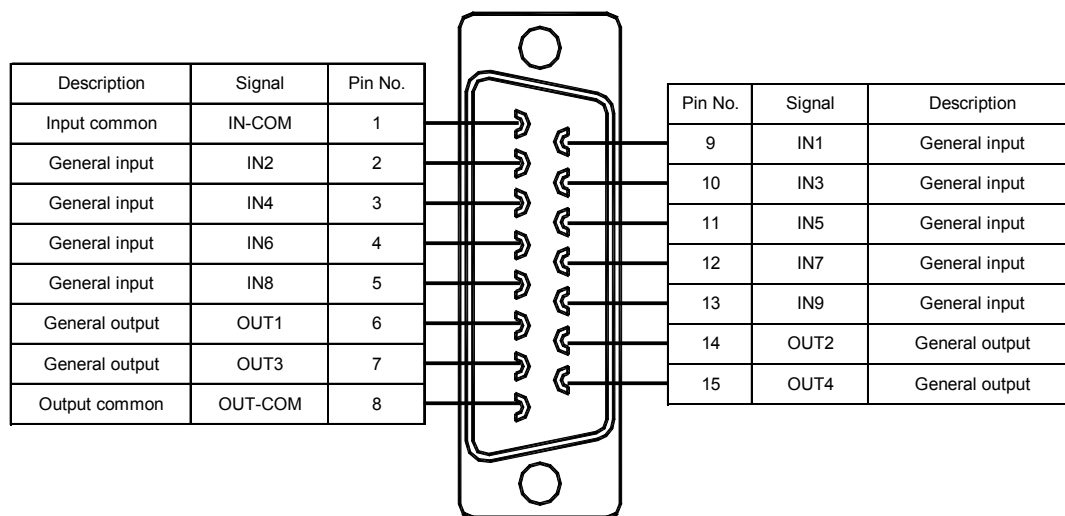
**Memo** : The connector shell is connected to the FG terminal.

## ■ Assigning Signals

Assign necessary signals to the I/O using the provided utility software, **Immediate Motion Creator for CM/SCX Series (IMC)**.
Connect the **SCX10** to a computer and activate the **IMC**. Connect the **SCX10** to a computer, launch the **IMC** and follow the steps below.
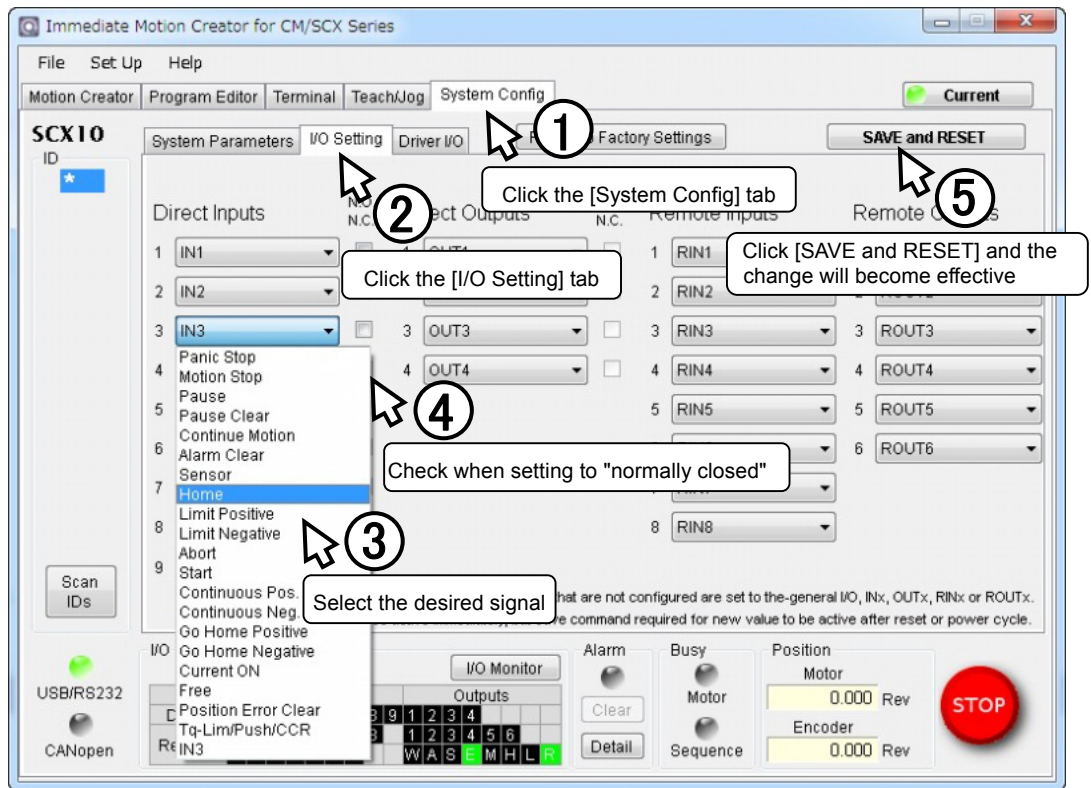


The setting can also be performed by command input, using the terminal mode of **IMC** or general terminal software. See the following chart for the command for assignment and logic level. After executing command, enter "SAVEPRM" and press the Enter key to save the parameter. New value becomes active after reset or power cycle.

- Input

| Signal | Command for Assignment | Command for Logic Level Setting |
|---|---|---|
| PSTOP (panic stop) | INPSTOP | PSTOPLV |
| MSTOP (motor stop) | INMSTOP | MSTOPLV |
| PAUSE (pause motion) | INPAUSE | PAUSELV |
| PAUSECL (pause clear) | INPAUSECL | PAUSECLLV |
| CONT (continue motion) | INCONT | CONTLV |
| ALMCLR (alarm clear) | INALMCLR | ALMCLRLV |
| SENSOR (sensor) | INSENSOR | SENSORLV |
| HOME (home sensor) | INHOME | HOMELV |
| LSP (limit switch positive) | INLSP | OTLV |
| LSN (limit switch negative) | INLSN | OTLV |
| ABORT (abort motion and sequence execution) | INABORT | ABORTLV |
| START (start sequence) | INSTART | STARTLV |
| MCP (move continuously positive) | INMCP | MCPLV |
| MCN (move continuously negative) | INMCN | MCNLV |
| MGHP (move go home positive) | INMGHP | MGHPLV |
| MGHN (move go home negative) | INMGHN | MGHNLV |
| CON (current on) | INCON | CONLV |
| FREE (current off, magnetic brake free) | INFREE | FREELV |
| PECLR (position error clear) | INPECLR | PECLRLV |
| TL (torque limiting/push-motion operation /current cutback release) | INTL | TLLV |

- Output

| Signal | Command for Assignment | Command for Logic Level Setting |
|---|---|---|
| ALARM (alarm) | OUTALARM | ALARMLV |
| END (motion end) | OUTEND | ENDLV |
| RUN (sequence running) | OUTRUN | RUNLV |
| MOVE (motor moving) | OUTMOVE | MOVELV |
| READY (operation ready) | OUTREADY | READYLV |
| LC (limiting condition) | OUTLC | LCLV |
| PSTS (pause status) | OUTPSTS | PSTSLV |
| HOMEP (home position) | OUTHOMEP | HOMEPLV |
| MBFREE (magnetic brake free) | OUTMBFREE | - |

The following example is a command to assign the FREE input to the IN3 and set to "normally closed."

```
>INFREE=3
>FREELV=1   (0: Normally Open, 1: Normally Closed)
>SAVEPRM
>RESET
```

## 6.4.2  Input Signals

### ■ Internal Input Circuit

All input signals of the device are photo coupler inputs. The signal state represents the "ON: Carrying current" or "OFF: Not carrying current" state of the internal photo coupler rather than the voltage level of the signal.
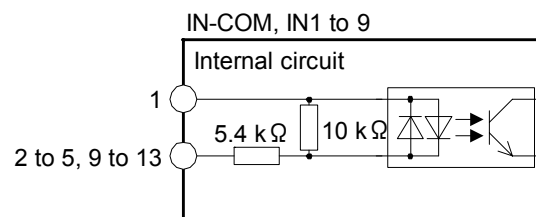


**Note**
- All input signals are "normally open" under the factory setting. When setting the logic level to "normally closed," ON/OFF will be opposite in the description of the following signals.
- Set the voltage between IN-COM and INx to be 4.25 VDC to 26.4 VDC when the photo coupler is ON.

### ■ Signals

- IN1-IN9 Input (unassigned)

The IN1 through IN9 inputs can be used as input ports for general signals when they are not assigned to a specific signal.
The status of each port can be read using an IN command or INx (x=1-9) command, and directly commanded or used in a sequence program.
A sequence program can be selected by the binary value of the general inputs when a START signal is input. See "9.6 Executing a Sequence" on page 125.

- PSTOP (panic stop) Input

This signal is used to forcibly stop motion and the sequence. Also the deviation counter in the driver is cleared for stopping servo motors and $\alpha$STEP products immediately. The action of the motor current and the alarm state after the PSTOP operation is determined by the ALMACT command, either alarm or no alarm.
The leading edge of the signal will cause the action.

- MSTOP (motor stop) Input

This signal is used to forcibly stop motion. This command does not stop a sequence program.
While the motor is operating, when MSTOP input is turned ON, the motion will be stopped as configured by the MSTOPACT command.
The leading edge of the signal will cause the action, either hard stop or soft stop.

- **SENSOR (sensor) Input**

  This signal is used for:
  - Stopping motion during continuous operation.
  - Offset motion on the fly during continuous operation.
  - Secondary home input for better accuracy during the mechanical homing operation.

  Set the operation (hard stop, soft stop, soft stop at fixed distance from SENSOR signal, no action) using the SENSORACT command.
  The leading edge of the signal will cause the action.

- **PAUSE (pause motion) Input**

  This signal is used to stop motion temporarily. If the PAUSE input is turned ON during any motion, motion is stopped and the device retains the motion type (positioning, continuous, etc) and remaining distance to the original target position if the paused operation is a positioning motion.
  If CONT command or input is executed, while in a paused situation, the remaining motion will be started. If START input is turned ON while in a paused situation only during sequence execution, the remaining motion will be started (STARTACT=0).
  Linked motions, return-to-electrical home operation and mechanical home seeking cannot be paused and resumed: PAUSE causes a soft stop, and CONT is ignored.
  If the PAUSECL input is turned ON or the PAUSECLR is commanded, the remaining motion will be canceled. Only the on-going motion is paused. The program execution will not stop.
  The leading edge of the signal will cause the action.

- **PAUSECL (pause clear) Input**

  This signal clears the on-going operation state that has been paused by the input of a PAUSE signal. (The remaining motion is canceled.)
  If this signal is activated while a sequence is running, only remaining portion of the current motion is cleared and the next step of the sequence will be executed, since the PAUSE does not stop the sequence.
  The leading edge of the signal will cause the action.

- **CONT (continue motion) Input**

  If CONT input is executed, while in a paused situation by PAUSE input, the remaining motion will be started.
  The leading edge of the signal will cause the action.

- **LSP (limit switch positive) Input/LSN (limit switch negative) Input**

  These signals are used to limit travel range. The stopping action is determined by the OTACT command.
  If OTACT=0, the system will stop the motor as quickly as possible (hard stop). Also the deviation counter in the driver is cleared for stopping servo motors and $\alpha_{STEP}$ products immediately.
  If OTACT=1, the system will stop the motor by a controlled deceleration over time (soft stop).
  While LSP/LSN input is active, system LSP/LSN signal/status is active.

- **HOME (home sensor) Input**

  This signal is used to set the home position when executing mechanical home seeking operation using sensor etc.
  While HOME input is active, system HOME signal/status is active.

- CON (current on) Input

This signal is used to toggle the motor current: the motor is in an excited state when ON (servo ON in the case of a servo motor), while freeing the motor shaft when OFF.
This signal also controls the power to the MBFREE (magnetic brake free) output. When the CON is on, the MBFREE output is on. When the CON is off, the MBFREE output is off (locked).
The leading edge of this signal will supply the current to the motor.

> **Note**
> - When the CON input is ON, the motor current can be toggled by the CURRENT command. When CON is OFF, the CURRENT command is ignored.
> - If the CON input is not assigned to any input and the CANopen is not active, the motor current at power on is determined by the STRSW setting.
> - If the CON input is assigned to the I/O connector and/or the CANopen is active, the motor current becomes ON only when both CON signals are ON.
> - During the CON input is OFF (motor current is off), PC (position command) is continuously overwritten by PF (position feedback) value. This is to track the actual position.
> - If the operation is made immediately after CON input is ON, position error may occur since the motor current rising time will be late. Allow a time interval according to the timing chart for each driver. Care should be taken especially when using CURRENT command in sequence program, or controlling CURRENT command, CON/COFF terminal or CON in CANopen by the host controller programs. For cases where the driver has READY output (ex. **AR**/**NX** Series driver), connect the READY input for driver on the **SCX10** and set to enable (DREADY=1). The position error will not occur.

- ALMCLR (alarm clear) Input

This signal is used to reset the alarm that has been generated by the system protective function or the driver alarm. Input the ALMCLR signal once after removing the cause that has triggered the protective function.
The leading edge of the signal will cause the action.

> **Note** For a description of the protective functions, see "13 Troubleshooting" on page 367.

- START (start sequence) Input

This signal is used to start the sequence execution as determined by the selected IN1 to IN7. See "9.6 Executing a Sequence" on page 125.
Set the starting method using the STARTACT command.

| STARTACT | Operation |
|---|---|
| 0 | Setting START input from OFF to ON starts sequence execution.<br>　When sequence is running, paused motion is resumed.<br>(Setting START input from ON to OFF does not stop sequence.) |
| 1 | Setting START input from OFF to ON starts sequence execution.<br>Setting START input from ON to OFF aborts the sequence. |

- ABORT (abort motion and sequence execution) Input

This signal is used to terminate a sequence execution and a motion. The motor will decelerate and stop.
The leading edge of the signal will cause the action.

- MCP (move continuously positive) Input/MCN (move continuously negative) Input

This signal is used to cause continuous motion. When the MCP input is detected, continuous operation in the forward direction (+ coordinate direction) will occur. When the MCN input is detected, continuous operation in the negative direction (- coordinate direction) will occur. It is not necessary to define the final position to start motion. The leading edge of the signal will cause the action.

- MGHP (move go home positive) Input/MGHN (move go home negative) Input

This signal is used to start the mechanical home seeking. When the MGHP input is detected, mechanical-home-seeking will be performed in the positive direction. When MGHN input is detected, mechanical-home-seeking will be performed in the negative direction.
The leading edge of the signal will start the home seeking.

- **FREE (current off, magnetic brake free) Input**
  - When the driver has the FREE input: Connect the FREE input of the driver to the FREE output on the driver connector on the **SCX10**. When turning this signal (the FREE input) ON, the motor current will be turned OFF and the electromagnetic brake will be released (The FREE input of the connected driver is turned ON).
  - When the driver has the M.B.FREE input: Connect the M.B.FREE input to the M.B.FREE output on the driver connector on the **SCX10**. When turning this signal (the FREE input) ON, the electromagnetic brake will be released (The M.B.FREE input of the driver is turned ON).
    * The motor current is not controlled.

  The FREE input is also available on the remote I/O (CANopen), and the FREE function will occur when either of those inputs becomes ON. The FREE command can also be used for this function regardless of the state of those inputs.
  While FREE input is active, system FREE signal/status is active.

- **PECLR (position error clear) Input**

  This signal is used to reset the PE (position error) value to zero (0).
  When PECLR input is turned ON, the PC (position command) value is set to equal to PF (feedback position) value. As a result, the PE is reset to zero. Also the deviation counter in the driver is cleared, when the driver alarm is inactive. This function can be used when the motor moved away from the PC such as overload alarm condition.
  The leading edge of the signal will cause the action.

- **TL (torque limiting/push-motion operation/current cutback release) Input**

  This signal is used to perform the following functions.
  - **AR** Series driver: push-motion operation
  - **NX** Series driver: torque limiting
  - **CRK**, **CMK** Series driver: current cutback release

  **Memo** The TL output of the driver connector on the **SCX10** will turn ON and OFF in synchronization with this TL input. To operate the function of this signal, it is required to connect the T-MODE input of the **AR** Series driver, the TL input of the **NX** Series driver, the C.D.INH input of the **CRK** Series driver or the ACDOFF input of the **CMK** Series driver to the TL output on the driver connector in advance.

## 6.4.3  Output Signals

### ■ Internal Output Circuit

All output signals of the device are open-collector outputs.
The signal state represents the "ON: Carrying current" or "OFF: Not carrying current" state of the internal transistor rather than the voltage level of the signal.



OUT1 to 4, OUT-COM
Internal circuit
30 VDC or less
20 mA or less
6, 7, 14, 15
8
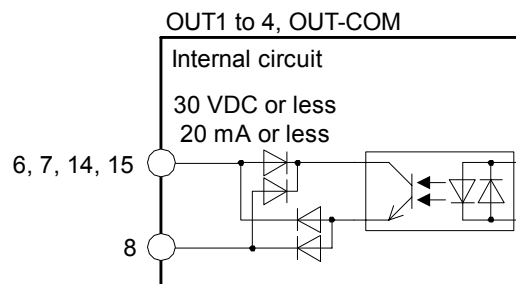
**Note** All input signals are "normally open" under the factory settings. When setting the logic level to "normally closed," ON/OFF will be opposite in the description of the following signals.

## ■ Signals

- ### OUT1-OUT4 Output (unassigned)

  The OUT1 through OUT4 outputs are used as output ports for general signals when they are not assigned to a specific signal. The status of each port is read and toggled using an OUT command or OUTx (x=1-4) command, and can be directly commanded or used in a sequence program.

- ### ALARM (alarm) Output

  When an alarm generates, the ALARM output will change. You can check the cause of the alarm by counting the number of times the ALARM LED blinks or by executing the ALM command.
  To reset the ALARM output, remove the cause of the alarm and then perform one of the following procedures after ensuring safety:
  - Input the ALMCLR signal once or enter the ALMCLR command.
  - Turn off the power, wait at least 10 seconds, and then turn it back on.

  **Memo** For a description of the protective functions, see "13 Troubleshooting" on page 367.

- ### END (motion end) Output

  When in the following condition, END signal will be active. See "8.8 END (motion end) Signal" on page 105.

  | Definition of END Signal (Source) | ENDACT Parameter | DEND Parameter | Typical Motor Type |
  |---|---|---|---|
  | End of pulse | 0 | 0 | Stepping motor |
  | End of pulse and within end area | 0<n (END area) | 0 | Stepping motor with an encoder |
  | Driver END signal | Unrelated | 1 | $\alpha_{STEP}$/servo motor |

- ### RUN (sequence running) Output

  This signal is output during sequence program execution.

  **Memo**
  - When the last command of the sequence program is a motor operation command (e.g. MI), the RUN output will be turned OFF as soon as the command is executed and motion is started.
  - When turning this signal OFF after completing an operation is desired, insert the MEND (Wait for Motion End) command at the end of the sequence program.

- ### MOVE (motor moving) Output

  This signal is output while the motor is moving. Motion commands are not accepted while the signal is ON.

  **Memo** This signal is output while pulses are being outputs and is not related with the MOVE input on the driver connector of the **SCX10**. The MOVE input on the driver connector of the **SCX10** is referred only when the sensor-less home seeking is performing with the **ESMC** controller.

- ### READY (operation ready) Output

  This signal is turned ON when the **SCX10** is ready to operate (other than MOVE, RUN and ALARM status). A sequence program or a motion command (MA, MI, MCP, MCN, MGHP, MGHN, MIx, EHOME, CONT) can be executed.
  Ready status is equal to other than MOVE, RUN and ALARM status.
  If "DREADY" is set to 1, when the driver is ready to operate (the driver's READY output signal is ON) in addition to above condition, this signal will be turned ON (It is required to connect the READY output of the driver to the READY input on the driver connector of the **SCX10** in advance). See the READY input on page 38 for the operation.

- LC (limiting condition) Output

This signal is turned ON under the following conditions.
  - **AR** Series driver: When the motor is in a state of push condition (the position deviation is 1.8 degrees or more) in the normal operating mode, or when the motor torque reaches to the preset value in the current control operating mode.
  - **NX** Series driver: When the motor torque reaches the preset value while the torque limiting function is used (Also when the motor torque reaches 300% of rated torque even while the torque limiting function is not used)
  - **RBK** Series driver: Under current cutback condition
  - **ESMC** controller: While pressing the mechanical home when performing sensor-less mechanical home seeking operation.

**Memo** The LC output will turn ON and OFF in synchronization with the LC input on the driver connector of the **SCX10**. To operate the function of this signal, it is required to connect the TLC output of the **AR** Series/**NX** Series driver, the CD output of the **RBK** Series driver or the T-UP output of the **ESMC** controller to the LC input on the driver connector of the **SCX10** in advance.

- PSTS (pause status) Output

This signal is output while the device is pausing with the PAUSE command or PAUSE input signal and can be cancelled by the CONT, PAUSECL, START (when sequence is running) or ABORT input signals or commands.

- HOMEP (home position) Output

This signal is output when a mechanical home seeking motion is successfully completed. This position is set to the origin (PC=0). Once this signal is ON, stopping on this position by operations such as EHOME or MA 0 sets this signal to ON.

- MBFREE (magnetic brake free) Output

This signal is used to control the electromagnetic brake
The MBFREE output is ON under normal operating condition and the system power is ON (CURRENT=1). The MBFREE output turns OFF when the motor loses its holding torque due to a current cutoff or alarm (CURRENT=0). Configure the circuit so that the holding torque of the electromagnetic brake is generated when this signal is OFF.
The MBFREE output can also be manually controlled with the FREE input signals on the system I/O connector (if assigned) and on the remote I/O (CANopen), as well as the FREE command. If any of those inputs is ON, the state of the FREE function becomes 1, and the MBFREE output becomes ON.
The relationship among the status of CURRENT, FREE and MBFREE is as below.

| CURRENT | FREE | MBFREE |
|---------|------|--------|
| 0 | 0 | 0 (Lock) |
| 0 | 1 | 1 (Free) |
| 1 | 0 | 1 (Free) |
| 1 | 1 | 1 (Free) |

∗ The state of MBFREE output on the I/O connector is always the same as the state of the MBFREE output on the driver connector of the **SCX10**.

**Note** Once the motor has lost its holding torque, the equipment that is attached to the motor shaft may move due to gravity or the presence of a load before the electromagnetic brake generates holding force.

## 6.4.4  Connection Example of I/O

### ■ Current Sink
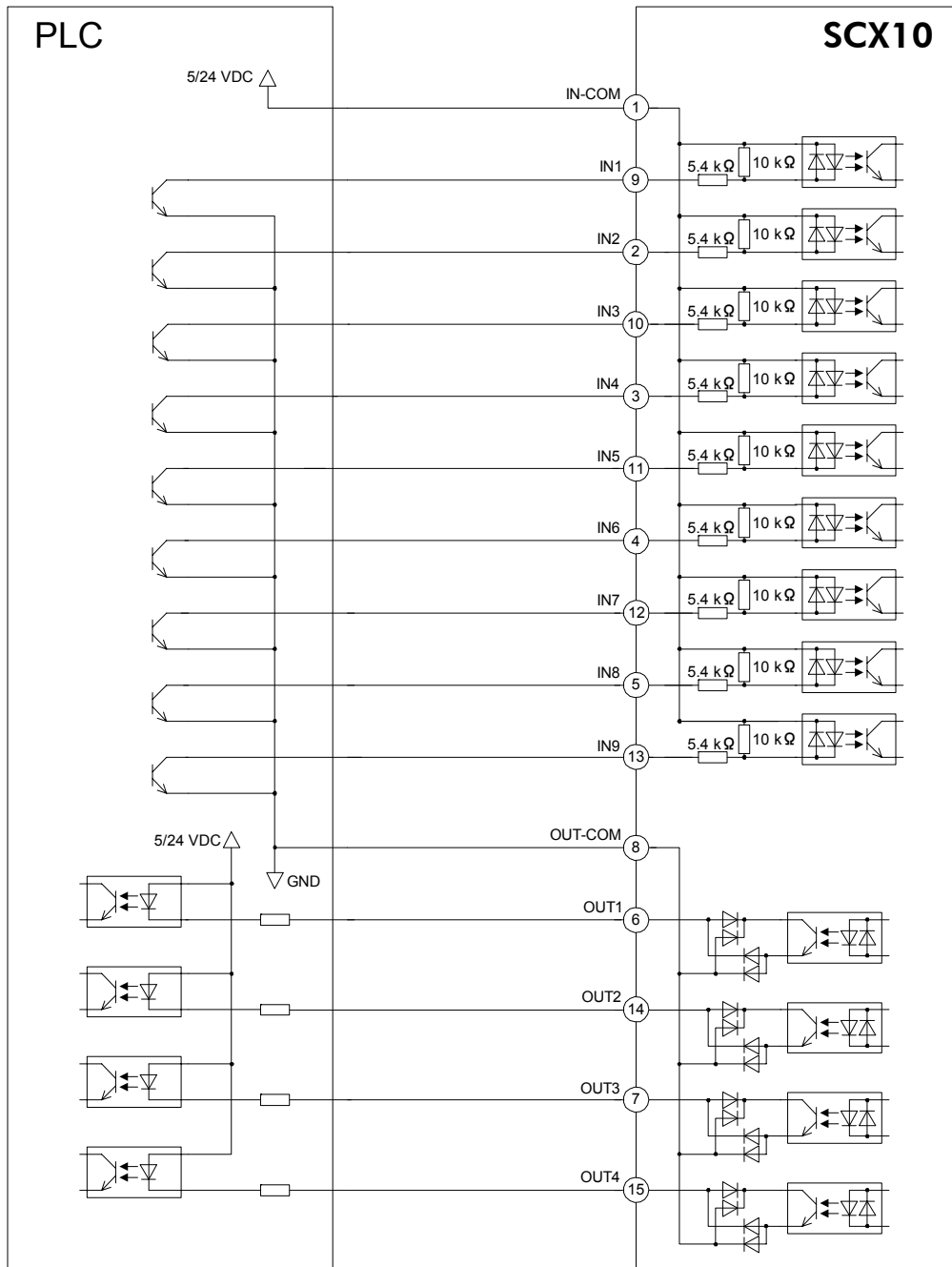
## ■ Current Source
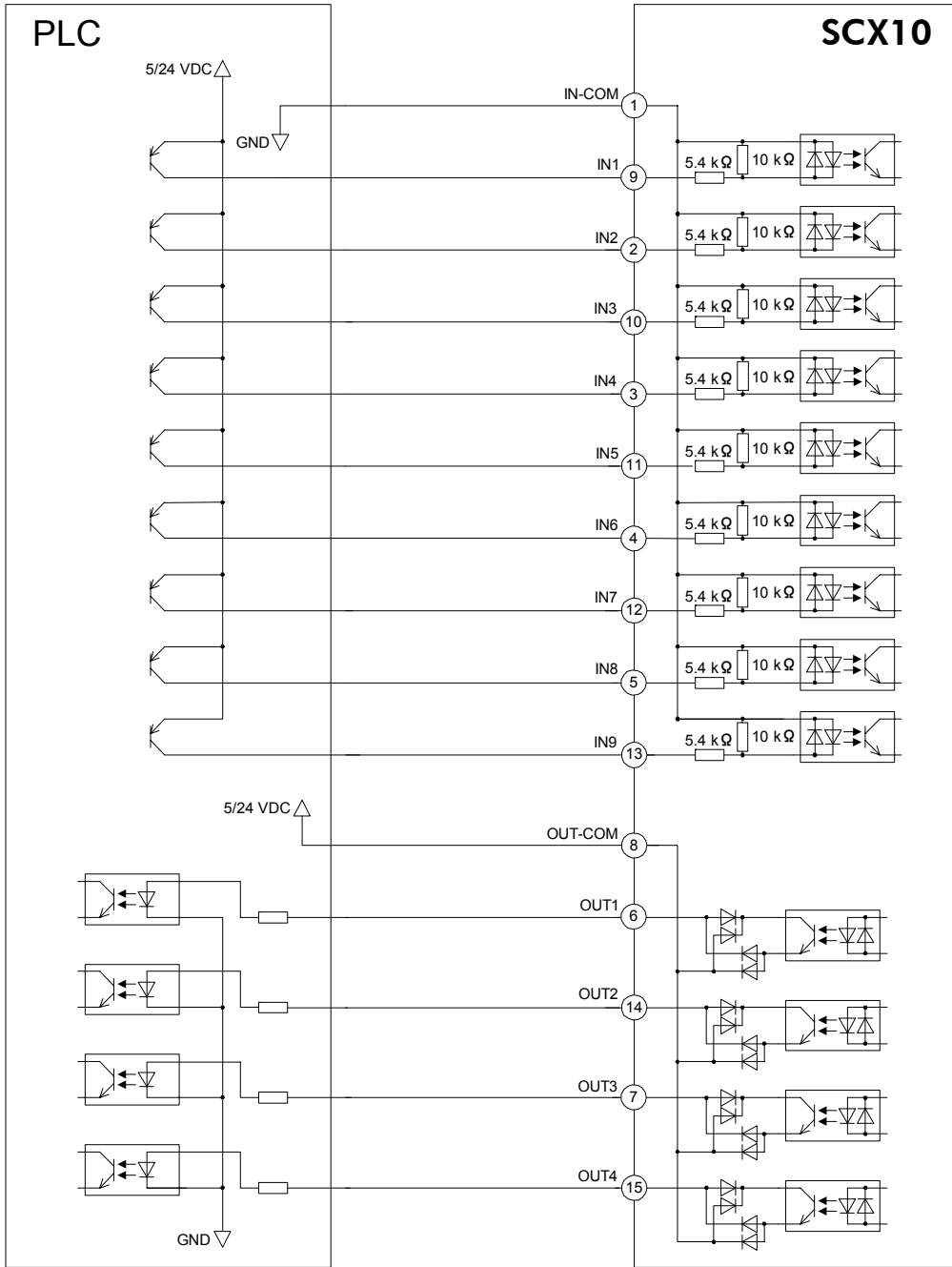
## 6.5  Connecting the Driver

### 6.5.1  I/O Voltage and Logic

Set the voltage and logic switches according to the driver.
Example: Set to 5 V and current source logic

DRIVER

5 V    24 V          SOURCE    SINK

**Memo**  Most of Oriental Motor's driver can be used with either source logic or sink logic. Selection can be made according to your needs.  The switch should be set to match with the selected drivers' logic.

**Note**
- The power to the **SCX10** must be off while setting these switches.
- The **SCX10** supplies the power to the driver to turn the photo couplers on the driver ON or OFF.  Selecting the wrong voltage and/or wrong logic may damage the driver and/or the **SCX10**.
- Set the voltage switch to 5 V for most of Oriental Motor's drivers while the **ESMC** controller requires setting to 24 V.

### 6.5.2  Signal Assignments

#### ■ Pin Assignments and Connector Function Table

Connect the I/O signal cable to the connector while checking the pin numbers in the table provided below. (The connector is not supplied. Provide a 25 Pin D-Sub connector separately.)
Signals are pre-assigned to the driver I/O so as to match with most of Oriental Motor products, while the assignments for the driver functions are shown in "6.5.3 Change of Signal Assignment" on page 35.

Following pin assignments are shown for a solder type connector.

| Driver general I/O when canceling assignment | Description | Signal name | Pin No. |
|---|---|---|---|
| – | Pulse/CW output | PLS/CW+ | 1 |
| – | Direction/CCW output | DIR/CCW+ | 2 |
| – | ASG Differential input | ASG+ | 3 |
| – | BSG Differential input | BSG+ | 4 |
| IN7 | Timing/ZSG Differential input | TIMD+ | 5 |
| – | Ground Connection | GND | 6 |
| OUT2 | Current On output | CON | 7 |
| OUT4 | Resolution Selection output | CS | 8 |
| OUT6 | Magnetic Brake Free output | MBFREE | 9 |
| OUT8 | General output | OUT8 | 10 |
| IN2 | Motion End input | END | 11 |
| IN4 | Limiting Condition input | LC | 12 |
| – | 5/24 V output | 5/24 V OUT | 13 |

| Pin No. | Signal name | Description | Driver general I/O when canceling assignment |
|---|---|---|---|
| 14 | PLS/CW- | Pulse/CW output | – |
| 15 | DIR/CCW- | Direction/CCW output | – |
| 16 | ASG- | ASG Differential input | – |
| 17 | BSG- | BSG Differential input | – |
| 18 | TIMD- | Timing/ZSG Differential input | IN7 |
| 19 | COFF | Current Off output | OUT1 |
| 20 | ACL/DCL | Alarm Clear/Deviation Counter Clear output | OUT3 |
| 21 | FREE | Current Off, Magnetic Brake Free output | OUT5 |
| 22 | TL | Torque Limiting output | OUT7 |
| 23 | ALARM | Alarm input | IN1 |
| 24 | TIMS | Timing/ZSG Single Ended input | IN3 |
| 25 | READY | Operation Ready input | IN5 |

**Memo**  The connector shell is internally connected to the FG terminal.

**Note**
- Be sure that the **SCX10** is not powered ON when the I/O connector is connected to, or disconnected from the driver. The connection or disconnection with power is ON damages the interface circuit in the **SCX10**.
- The 5/24 V output in this connector is supplied only for the purpose of interfacing with the driver as instructed in this manual. Do not use it for other purpose.

## ■ Internal Circuit
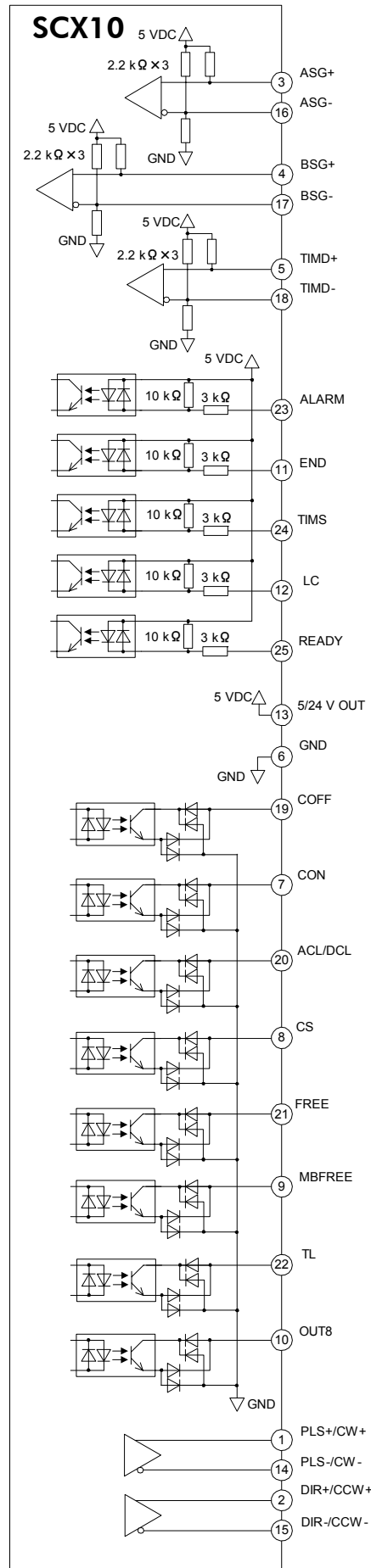
## 6.5.3 Change of Signal Assignment

The signal assignments on the driver connector are required to change in the following cases.

- With the **AR** Series driver (AC and DC power input type), when push-motion operation is used
- With the **NX** Series driver, when torque limiting function is used, when current position reading function is used, or when accurate mechanical home seeking operation using the Z-phase signal (timing signal) etc. is required
- With the **ESMC** controller, when sensor-less mechanical home seeking operation is used or, when current position reading function is used

Change the driver I/O using the "Automatic Setting" function of the provided utility software, **Immediate Motion Creator for CM/SCX Series (IMC)**. With this automatic setting function, the necessary signals will be assigned to the I/O for driver.

| Memo | When the **LSD** driver is used, refer to the instructions indicated as the "**AR** Series" in this operating manual. |

| Note | Push-motion, torque limiting, current reading functions and sensor-less mechanical home seeking are implemented in the firmware Ver.2.00 or later of the **SCX10**. The automatic setting of **IMC** (Ver.2.00 or later) is a function that is available for the firmware Ver.2.00 or later of the **SCX10**. With the Ver.1.07 or older model, only the manual setting and the initialization to the factory setting are available. (The firmware version can be confirmed by [Help] - [Version Information] on the **IMC** or VER command.) |

## ■ Setting Procedure

Connect the **SCX10** to a computer and activate the **IMC**.
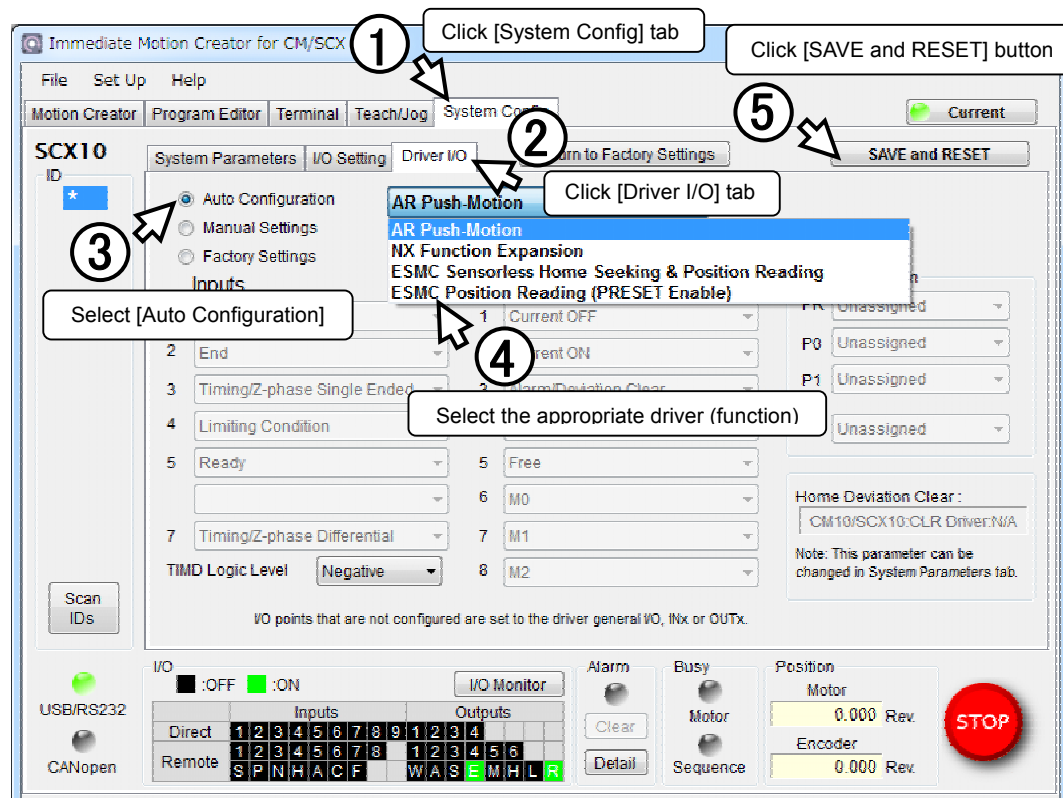
## ■ Automatic Setting Types

The following selections are available for setting.

| Selection | Description | Signals to be Assigned | Signals to be Deleted | Connection Diagram |
|---|---|---|---|---|
| AR Push-Motion | With the **AR** Series driver, when using the push-motion operation function (disabling the resolution switching function) | M0, M1, M2 | MBFREE | page 46 to 47 |
| NX Function Expansion | With the **NX** Series driver, when torque limiting function and/or current position reading function is used, and/or when accurate mechanical home seeking operation is required such as using Z-phase pulse (timing signal) | M0, M1, PRESET, REQ, PR, CK, P0, P1 | COFF, CS, MBFREE | page 60 to 61 |
| ESMC Sensorless Home Seeking & Position Reading | With the **ESMC** controller, when sensor-less mechanical home seeking operation is performed (The current position reading function will also be effective but the PRESET signal that sets the home position at an arbitrary position will not be effective) | REQ, PR, CK, P0, P1, HOME, HMSTOP, MOVE | CON, TL, READY | page 56 to 57 |
| ESMC Position Reading (PRESET Enable) | When using the current position reading function of the **ESMC** controller (The PRESET signal that sets the home position at an arbitrary position will be effective but sensor-less mechanical home seeking operation cannot be used) | PRESET, REQ, PR, CK, P0, P1, MOVE | CON, READY | page 58 to 59 |

**Memo**

- When using other functions than the driver's factory setting such as driver current position reading, torque limiting, push-motion operation, etc., the setting is also required to the driver. Refer to page 100 "8.6 Driver Current Position Reading (**NX** Series driver, **ESMC** controller)" or page 97 "8.5 Torque Limiting/Push-motion Operation (**NX** Series driver, **AR** Series driver)" for details.
- To see the driver I/O assignment by each setting, refer to the wiring diagram in this manual, or the driver I/O setting or I/O status monitor of the **IMC**.
- If the automatic setting is executed, unnecessary signals will be unassigned to assign necessary signals for the selected driver and function.
- When setting to "NX Function Expansion," HOMEDCL will automatically be set to "1" in addition to the driver I/O change. By setting this, when detecting the home position at mechanical home seeking operation, the driver deviation counter will be cleared and thus the motor will stop immediately. (The motor will be able to stop at the home position accurately and reset home position.)  Refer to page 86 "8.2.5 Mechanical Home Seeking, HOMEDCL (deviation counter clear select at home seeking operation)" When setting manually, set "Home Deviation Clear" under [System Config] - [System Parameters] tab of the **IMC**.
- The driver I/O setting can also be done for each signal individually (manual setting). The manual setting is used in special cases such as using the driver functions that is not supported. Refer to "memo" for "6.5.4 Input Signals for Driver" (page.37-) or "6.5.5 Output Signals for Driver" (page.40-).

## 6.5.4 Input Signals for Driver

### ■ Internal Input Circuit

All input signals for the driver except for the encoder inputs are photo coupler inputs.
The signal state represents the "ON: Carrying current" or "OFF: Not carrying current" state of the internal photo coupler rather than the voltage level of the signal.

5/24 VDC or GND, ALARM,
END, TIMS, LC, READY

Internal circuit

10 kΩ    3 kΩ

6 or 13

11, 12, 23 to 25
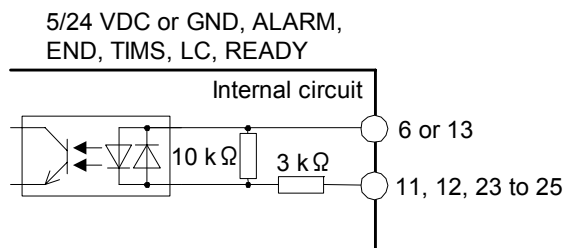
The encoder inputs are line receiver inputs. These inputs can be connected to line driver, open collector and TTL output encoders, since the resistors are configured as shown in the figure.

ASG+, BSG+, ZSG+/TIMD+,
ZSG-, BSG-, ZSG-/TIMD-

5 VDC    Internal circuit

26LV32 or equivalent    2.2 kΩ ×3

(+) 3, 4, 5

(−) 16, 17, 18

GND
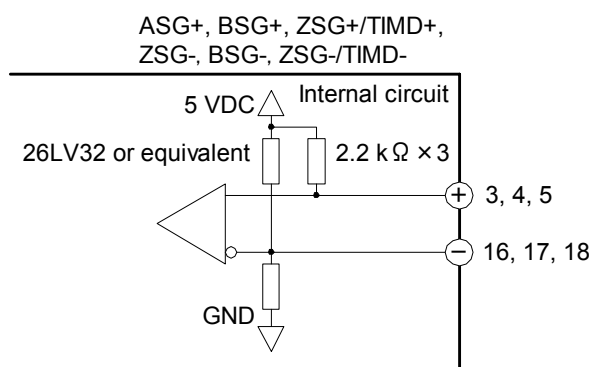
**Note** When these inputs are used, connect the GND on the **SCX10** and the GND on the driver. Otherwise the **SCX10** may be damaged by a potential difference.

### ■ Signals

- ALARM (alarm) Input

This signal is used to input the alarm signal (ALM output) or over heat signal (O.H. output) from the driver. If a driver alarm has occurred, the system ALARM signal/status becomes active (alarm code 6Eh: driver alarm). The motor will decelerate to a stop and the sequence program will also stop. No pulse can be output while an alarm signal input is active, although any command not associated with pulse output can be executed.
  - The input logic is Normally Closed. (The **SCX10** detects the OFF status of driver output as the driver is in an alarm condition.) If there is a selectable switch for the alarm logic on the driver, set it to "Normally Closed."
  - The DALARM parameter enables/disables the use this signal. (The factory setting is inactive.) Set DALARM=1 to enable this input if the driver alarm signal is used.

    **Note** If DALARM is set to 1 and the power on timing of the driver is delayed from the **SCX10**, the driver's alarm output is OFF at the start up, and that is identical to a "driver alarm." The driver alarm status is cleared automatically when the driver is powered on and the alarm output becomes ON, meaning alarm is now OFF. These actions cause that the alarm status history (ALM) records alarm code 6E: Driver Alarm, to be recorded each time the driver is powered on.

- **END (motion end) Input**

  This signal is used to input the END signal output from the driver at the end of operation.
  The signal becomes the system END signal/status, and used for the MEND parameter, END output, and mechanical home seeking. This signal should be connected in order to confirm the definite operation completion when using an $\alpha_{STEP}$ or a servo motor system that has the deviation in the driver. See "8.8 END (motion end) Signal" on page 105.
  - The DEND parameter selects the source of system END status, either driver end signal or the internal end signal. (The factory setting is DEND=0, the use of internal end signal.) Set DEND=1 to select this input if the driver end signal is used.

- **TIMS (timing signal·Z-phase pulse single ended input) Input**

  This signal is used to input the open collector excitation timing signal output of the stepping motor driver (Z-phase signal for the servo motor driver). The TIM signal is an excitation timing output of the stepping motor driver, and is considered ON in fifty (50) or one hundred (100) fixed, evenly spaced locations per motor revolution. The Z-phase signal of the servo motor is turned ON once per revolution of the motor.
  When mechanical home is detected in mechanical home seeking mode, an accurate home position can be found by using this signal with the SENSOR input and/or the HOME input. (See "8.2.5 Mechanical Home Seeking" on page 83.)
  - The combination of the ENC and the TIM parameters selects the source of the system TIM signal, the TIMD and the TIMS signal from the driver or the Z signal from the external encoder. (The factory setting is ENC=0, TIM=1, the use of TIMS.) See "Selection of the Timing Source When Using the Timing Signal" on page 85

- **TIMD (timing signal·Z-phase pulse differential input) Input**

  This signal is used to input the differential excitation timing output signal of the stepping motor driver (Z-phase signal for the servo motor driver).
  The function of this input signal is same as the TIMS, except this terminal is differential input that allows connecting to the differential timing output on the driver.
  - Set TIM=0 to use this signal. (See above.)

- **ASG/BSG (ASG pulse/BSG pulse differential) Inputs**

  These signals are used to input the pulse output signals (ASG, BSG) from the driver that correspond the motor operation.
  The **SCX10** continuously monitors the feedback position (PF), calculated from encoder count (EC) that counts these input pulses. The difference between PC (position command) and PF becomes PE (position error). The PE is used for position confirmation referenced by the ENDACT command. The PE can also be used in sequence programs, and the miss-step detection function of the stepping motor can be configured. See "8.9 Encoder Function" on page 105.
  - The ENC parameter selects the source of the system EC signal, either the driver or the external encoder. (Factory setting is ENC=0, the use of driver encoder.)

- **MOVE (motor moving) Input**

  This signal is used to input the MOVE signal from the driver. It is turned ON while the motor is operating. Using the DSIGMOVE command, you can check the signal status (It can also be used in a sequence). When connecting to the **ESMC** controller, this signal is used when performing sensor-less mechanical home seeking operation.

- **READY (operation ready) Input  (AR/NX Series driver)**

  This signal is used to input the READY signal from the driver.
  If a command for starting the motor operation such as MI or MA etc. is executed while the READY signal is OFF, the pulse signal will output when the READY signal is turned ON. If the waiting time exceeds 3 seconds, a timeout will occur and an alarm will generate (alarm code 6F: driver connection error).
  - Select by the DREADY parameter whether the READY signal from the driver is used or not (The factory setting of the **SCX10** is "DREADY=0," and the READY signal from the driver is not used). Set the parameter to "DREADY=1" when using the READY signal from the driver.

- **PR (position data output ready) Input  (NX Series driver, ESMC controller)**

  This signal is used when the drivers' current position reading function is executed. This signal is used to input the P-OUTR signal from the **NX** Series driver or the OUTR signal from the **ESMC** controller.
  In normal operation, this input functions as another signal that is redundantly assigned to the same pin number (a general input when not assigned). And it automatically switches to the PR input only when executing the current position reading command (ABSREQ and ABSREQPC).

- P0 (position data bit 0) Input (**NX** Series driver, **ESMC** controller)

  This signal is used to read the drivers' current position. This signal is used to input the P-OUT0 signal from the **NX** Series driver or the OUT0 signal from the **ESMC** controller.

  In normal operation, this input functions as another signal that is redundantly assigned to the same pin number (a general input when not assigned). And it automatically switches to the P0 input only when executing the current position reading command (ABSREQ and ABSREQPC).

- P1 (position data bit 1) Input (**NX** Series driver, **ESMC** controller)

  This signal is used to read the drivers' current position. This signal is used to input the P-OUT1 signal from the **NX** Series driver or the OUT1 signal from the **ESMC** controller.

  In normal operation, this input functions as another signal that is redundantly assigned to the same pin number (a general input when not assigned). And it automatically switches to the P1 input only when executing the current position reading command (ABSREQ and ABSREQPC).

- LC (limiting condition) Input

  This signal is used to input the TLC signal of the **AR** Series/**NX** Series driver, the CD signal of the **RBK** Series driver or the T-UP signal of the **ESMC** controller. It will be turned ON under the following conditions.

  - **AR** Series driver: When the motor is in a state of push-motion operating condition (the position deviation is 1.8 degrees or more) while in the normal operating mode, or when the motor torque reaches to the preset value in the current control operating mode.
  - **NX** Series driver: When the motor torque reaches the preset value while the torque limiting function is used
  - **RBK** Series driver: Under current cutback condition
  - **ESMC** controller: While pressing the mechanical home when performing sensor-less mechanical home seeking operation.

  When the LC output is assigned to the I/O connector, the LC output will turn ON and OFF according to this signal. Using the DSIGLC command, you can check the signal status (It can also be used in a sequence).

  **Memo**
  - The warning outputs from the **AR** Series, **NX** Series drivers and the electromagnetic brake control signal output from the **NX** Series driver are not supported. To use these signals, prepare the driver general input by canceling the assignment for one of the inputs (IN1 to IN5) on the driver connector of the **SCX10**, and then connect to it. By using the DINx or DIN command, you can check whether the warning or electromagnetic brake control signal is output or not. It can also be used in a sequence. The factory setting for the electromagnetic brake control signal output of the **NX** Series driver is not enabled. Be sure to assign the electromagnetic brake control signal output by referring to the driver operating manual.
  - The alarm code outputs from the **AR** Series and **NX** Series drivers are not supported. The driver alarm code output AL0, AL1 and AL2, which are assigned to the same terminals as READY, TLC and TIM2 (ZSG2) of the driver respectively, are connected to READY, TLC and TIMS of the **SCX10** respectively. To read the ALARM code, cancel the assignment of READY, LC and TIMS to be the driver general input IN3, IN4 and IN5. Using the DINx or DIN command, you can check the alarm code. They can also be used in a sequence. The factory setting for the alarm code output of the **AR** Series and **NX** Series driver is not enabled. Be sure to assign the alarm code output by referring to the driver operating manual.
  - The positioning near output (NEAR output) from the **NX** Series driver is not supported. A similar function can be performed by setting of ENDACT of the **SCX10**, or combining the command PE that returns the position deviation and an "IF" statement. See "8.8 END (motion end) Signal" on page 105 and "8.9 Encoder Function" on page 106. When using the NEAR output from the driver, see the instructions below.
    The NEAR output of the driver, which is assigned to the same terminal as the ZSG2 output, is connected to TIMS of the **SCX10**. To read the NEAR output, cancel the assignment of the TIMS signal or an unnecessary signal to be the driver general input. Using the DINx or DIN command, you can check whether this signal is output or not. It can also be used in a sequence. The factory setting for the NEAR output of the **NX** Series driver is not enabled. Be sure to assign the NEAR output by referring to the driver operating manual.
  - When assigning or canceling the driver signal, use the provided utility software **Immediate Motion Creator for CM/SCX Series (IMC)**. When selecting "Manual Settings" on [System Config] - [Driver I/O] screen, the driver I/O can be set individually.

## 6.5.5 Output Signals for Driver

### ■ Internal Output Circuit

All output signals for the driver except for the pulse outputs are open-collector outputs.
The signal state represents the "ON: Carrying current" or "OFF: Not carrying current" state of the internal transistor rather than the voltage level of the signal.

COFF, CON, ACL/DCL, CS, FREE,
MBFREE, TL, OUTx, 5/24 VDC or GND

Internal circuit

30 VDC or less
20 mA or less

7 to 10, 19 to 22

6 or 13
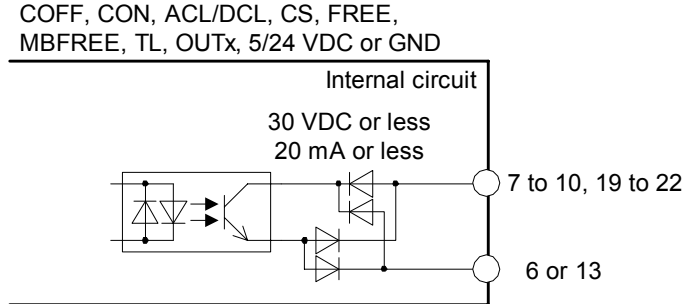
The pulse outputs employ differential outputs. This output can be connected to photocoupler inputs and line receiver inputs.

PLS+/CW+, DIR+/CCW+,
PLS-/CW-, DIR-/CCW-

Internal circuit

4428 or equivalent   30 mA or less
10 Ω
(+) 1, 2

30 mA or less
10 Ω
(−) 14, 15
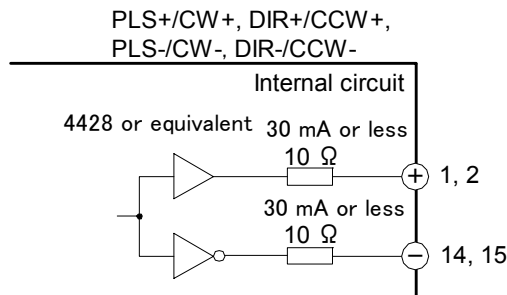
### ■ Signals

- PLS/CW and DIR/CCW Output

These terminals are used to output the pulse and direction signals to the driver.
Both the 1 pulse mode and the 2 pulse mode are supported. In the 1 pulse mode, the motor will rotate in the CW direction with the DIR (direction) output turned ON and in the CCW direction with the DIR output turned OFF. In the 2 pulse mode, pulses for the CW and the CCW are provided separately to each terminal. The output mode may be switched between 1 pulse and 2 pulse modes via the PULSE parameter. (Set PULSE=0 for 2 pulse mode, PULSE=1 for 1 pulse mode. Factory setting of the parameter is PULSE=1.)
Additionally, the output logic can be changed using the PLSINV command.

Pulse output characteristic:
> Rise/fall time: 0.1 μs or less
> Duty cycle: 50% (ON time is fixed to 20μs when frequency is 25 kHz or less)

| Note | If the driver for the 2-phase **CSK** Series or the **UMK** Series is connected to **SCX10**, only 2-pulse mode should be used. With the 1 pulse mode, some pulses may be missed when reversing due to a delay in response on the driver interface. |

1 pulse mode
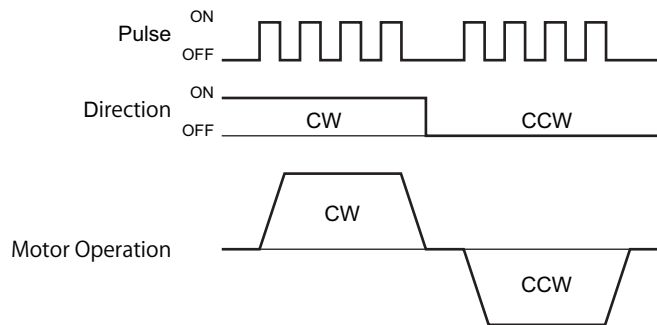


2 pulse mode
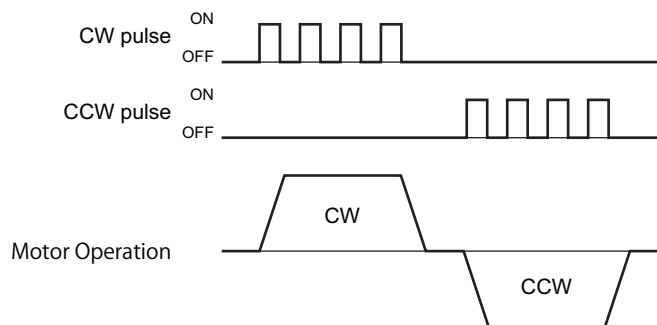


- COFF (current off) Output

  This signal is used to toggle the motor current. The motor current is OFF when this signal is ON. Connect this signal to the C.OFF (current off) or AWO (all windings off) input on the driver, if the driver has either of these inputs.

  This signal can be controlled using the system CON input signal on the system I/O connector (if assigned), the CURRENT command and CANopen communication. When even one of these method commands "Current Off", this signal will be turned ON.

  The STRSW parameter determines the motor current ON or OFF when the device is powered on.

  The function of this signal is exactly the same as the CON output, except the logic is opposite.

- CON (current on) Output  (**AR**/**NX** Series driver)

  This signal is used to toggle the motor current. The motor current is ON when this signal is ON. Connect this signal to the C-ON (current ON) input or S-ON (servo ON) on the driver, if the driver has those inputs.

  This signal can be controlled using the system CON input signal on the system I/O connector (if assigned), the CURRENT command or CANopen communication. When even one of these method commands "Current Off", this signal will be turned OFF.

  If the CON input is not assigned to the I/O connector, the STRSW parameter determines the motor current ON or OFF when the device is powered on.

  The function of this signal is exactly the same as the COFF output, except the logic is opposite.

- ACL/DCL (driver alarm clear/deviation counter clear) Output  (**AR**/**NX** Series driver)

  This signal is used to clear the alarm status and the deviation counter of the driver. Connect this signal to the ACL (alarm clear) or CLR/ALM-RST (deviation counter clear/alarm reset) input terminal on the driver.

  This signal can be controlled using the ALMCLR input signal on the system I/O connector (if assigned), the ALMCLR command or CANopen communication. When the alarm is inactive, this signal functions as the deviation counter clear, and can be controlled using the PECLR input signal on the system I/O connector (if assigned), the PECLR command or CANopen communication.

  This signal is also used to stop servo motors and $\alpha_{STEP}$ products immediately by clearing the deviation counter in the driver. The deviation counter clear output momentarily becomes active when the limit sensors (LSP/LSN signal) are detected (when OTACT=0), the PSTOP is commanded and a mechanical home seeking is performed (when HOMEDCL=1).

- **CS (resolution selection) Output**

  This signal is used to select the motor resolution. Connect this signal to the CS (change step) input terminal of the driver.

  The state of this signal is set when the system power is first turned ON, and remains in that state while the system is power up. The state of CS can be changed using the STRDCS command.

  STRDCS=0: CS output is OFF at system start up

  STRDCS=1: CS output is ON at system start up

- **FREE (current off, magnetic brake free) Output  (AR/NX Series driver, ESMC controller)**

  This signal is used to make the motor shaft free (motor current OFF and release the electromagnetic brake at the same time). Connect this signal to the FREE input terminal on the driver.

  This signal can be controlled using the FREE input signals on the system I/O connector (if assigned) and on the remote I/O (CANopen), as well as the FREE command. If any of those inputs is ON, the state of the FREE function becomes 1 and the FREE output becomes ON. (The FREE input of the connected driver becomes ON. That causes the motor current to be turned OFF and the electromagnetic brake will be released.) The FREE command can always be used to control the FREE output regardless of the states of those inputs.

- **HOME (sensor-less home seeking operation start) Output  (ESMC controller)**

  This signal is used to start sensor-less home seeking operation.

  It becomes effective when the mechanical home seeking operation mode (homing type) is set to "HOMETYP=12." See "8.2.5 Mechanical Home Seeking" in more detail.

- **HMSTOP (sensor-less home seeking operation stop) Output  (ESMC controller)**

  This signal is used to stop sensor-less home seeking operation.

- **REQ (position data transmission request) Output  (NX Series driver, ESMC controller)**

  This signal is used to request the driver to send the data when executing the current position reading command (ABSREQ and ABSREQPC). The PR input, P0 input, P1 input and CK output will become effective when this signal is turned ON.

- **PRESET (reset home position) Output  (NX Series driver, ESMC controller)**

  This signal is used to set the driver's current position to the PRESET position (home position).

- **CK (position data transmission clock) Output  (NX Series driver, ESMC controller)**

  This signal becomes effective when executing the current position reading command (ABSREQ and ABSREQPC). It is the clock signal to request the data.

  In normal operation, this signal does not function and another signal that is redundantly assigned to the same pin number (a general output signal when nothing is assigned) functions. This signal functions only when executing the current position reading command (ABSREQ and ABSREQPC).

- **TL (torque limiting/push-motion operation/current cutback release) Output (AR/NX/CRK/ CMK Series driver)**

  This signal is used to perform the following functions.
  - **AR** Series driver: push-motion operation
  - **NX** Series driver: torque limiting
  - **CRK**, **CMK** Series driver: current cutback release
    (The current cut back release can be performed by turning the TL input signal ON or commanding TL =1.)

- **M0/M1/M2 (data select bit 0/bit 1/bit 2) Output (AR/NX Series driver)**

  These signals are used to set the operation data (when executing the torque limiting/push-motion operation function etc.).

  **Memo** ┇ Only the M0 and M1 are used when performing the torque limiting function with the **NX** Series driver.

- MBFREE (magnetic brake free) Output  (**RK** Series driver)

This signal is used to control the electromagnetic brake. Connect this signal to the M.B.FREE input terminal on the driver. With standard Oriental Motor's stepping motor drivers, when the M.B.FREE input on the driver is ON, power to the electromagnetic is supplied and electromagnetic brake is disengaged (Free).
The MBFREE output is ON under normal operating conditions and the system power is ON (CURRENT=1). The MBFREE output turns OFF when the motor loses its holding torque due to a current cutoff or alarm (CURRENT=0).
The MBFREE can also be manually controlled with the FREE function status. The state of the FREE function can be controlled using the FREE input signals on the system I/O connector (if assigned) and on the remote I/O (CANopen), as well as the FREE command. If any of these inputs is ON, the state of the FREE function becomes 1, and the MBFREE output becomes ON.
The relationship among the status of CURRENT, FREE and MBFREE is as below.

| CURRENT | FREE | MBFREE |
|:---:|:---:|:---:|
| 0 | 0 | 0 (Lock) |
| 0 | 1 | 1 (Free) |
| 1 | 0 | 1 (Free) |
| 1 | 1 | 1 (Free) |

∗ The state of MBFREE output on the driver connector of the **SCX10** is always the same as the state of the MBFREE output on the I/O connector.

**Memo**  The current control mode ON (CCM) function of the **AR** Series driver is not supported. However, if this signal is connected to the general output OUT8, it can be operated to turn "ON and OFF" by the OUT8 command (ON: OUT8=1, OFF: OUT8=0). It can also be used in a sequence.

## 6.5.6 Connection Example of Driver

### ■ AR Series Driver (Current Sink)

## ■ AR Series Driver (Current Source)

**SCX10**  5 VDC

2.2 kΩ×3

**AR**

| SCX10 | | AR |
|---|---|---|
| 3 ASG+ | | ASG+ 3 |
| 16 ASG- | | ASG- 4 |

5 VDC
2.2 kΩ×3
GND

| 4 BSG+ | | BSG+ 5 |
| 17 BSG- | | BSG- 6 |

5 VDC
GND  2.2 kΩ×3

| 5 TIMD+ | | TM1+ 7 |
| 18 TIMD- | | TIM1- 8 |

GND

| | ALM+ 9 |
| 10 kΩ  3 kΩ  ALARM  23 | ALM- 10 |
| 10 kΩ  3 kΩ  END  11 | (WNG+) 11 |
| | (WNG-) 12 |
| 10 kΩ  3 kΩ  TIMS  24 | END+ 13 |
| | END - 14 |
| 10 kΩ  3 kΩ  LC  12 | READY+/(AL0+) 15 |
| | READY-/(AL0-) 16 |
| 10 kΩ  3 kΩ  READY  25 | TLC+/(AL1+) 17 |
| GND | TLC-/(AL1-) 18 |
| 5 VDC  5/24 V OUT  13 | TIM2+/(AL2+) 19 |
| | TIM2-/(AL2-) 20 |

| 6 GND | | GND 2 |
| GND | | GND 21 |

GND

| 5 VDC  COFF  19 | IN-COM 22 |
| CON  7 | C-ON 23 |
| ACL/DCL  20 | CLR/ALM-RST 24 |
| CS  8 | CCM 25 |
| FREE  21 | CS/(T-MODE) 26 |
| MBFREE  9 | -/(M0) 27 |
| TL  22 | RETURN/(M1) 28 |
| OUT8  10 | P-RESET/(M2) 29 |
| | FREE 30 |
| 1 PLS+/CW+ | CW+/PLS+ 31 |
| 14 PLS-/CW- | CW-/PLS- 32 |
| 2 DIR+/CCW+ | CCW+/DIR+ 35 |
| 15 DIR-/CCW- | CCW-/DIR- 36 |

SW Settings
SINK/SOURCE : SOURCE
5V/24V : 5V

## ■ AR Series Driver  Push-motion (Current Sink)

**Note**  The I/Os for driver on the **SCX10** indicated below are set for "AR Push-Motion." The I/Os on the driver are also set for the push-motion operation. See "6.5.3 Change of Signal Assignment" on page 35 and "8.5 Torque Limiting/Push-motion Operation (**NX** Series driver, **AR** Series driver)" on page97.



SW Settings
SINK/SOURCE：SINK
5V/24V　　　 ：5V

## ■ AR Series Driver  Push-motion (Current Source)

| Note | The I/Os for driver on the **SCX10** indicated below are set for "AR Push-Motion." The I/Os on the driver are also set for the push-motion operation. See "6.5.3 Change of Signal Assignment" on page 35 and "8.5 Torque Limiting/Push-motion Operation (**NX** Series driver, **AR** Series driver)" on page97. |
|---|---|



SW Settings
SINK/SOURCE：SOURCE
5V/24V 　　　：5V

## ■ AS/ARL Series Driver (Current Sink)



SW Settings
SINK/SOURCE：SINK
5V/24V      ：5V

## ■ AS/ARL Series Driver (Current Source)



SW Settings
SINK/SOURCE：SOURCE
5V/24V ：5V

## ■ ASC Series Driver (Current Sink)



SCX10

ASC

SW Settings
SINK/SOURCE : SINK
5V/24V      : 5V

## ■ ASC Series Driver (Current Source)



SW Settings
SINK/SOURCE：SORCE
5V/24V　　　：5V

## ■ RK Series Driver (Current Sink)



SW Settings
SINK/SOURCE：SINK
5V/24V　　　：5V

## ■ RK Series Driver (Current Source)



SW Settings
 SINK/SOURCE : SOURCE
 5V/24V      : 5V

## ■ RBK Series Driver (Current Sink)

## ■ RBK Series Driver (Current Source)



SW Settings
SINK/SOURCE：SOURCE
5V/24V　　　：5V

## ■ ESMC Controller  Sensor-less Home Seeking & Position Reading (Current Sink)

**Note**
- The I/Os for driver on the **SCX10** indicated below are set for "ESMC Sensorless Home Seeking & Position Reading."  See "6.5.3 Change of Signal Assignment" on page 35 , "HOME Seeking Pattern: HOMETYP 12 (Sensor-less mode)" on page 87 and "8.6 Driver Current Position Reading (**NX** Series driver, **ESMC** controller)" on page 100.
- For use of the **SCX10**, the **ESMC** controller should be set to the "driver mode." If the **ESMC** controller is used as the "controller mode (factory setting)," unexpected motion may occur due to unmatched I/O assignments.



- 56 -

## ■ ESMC Controller  Sensor-less Home Seeking & Position Reading (Current Source)

**Note**
- The I/Os for driver on the **SCX10** indicated below are set for "ESMC Sensorless Home Seeking & Position Reading."  See "6.5.3 Change of Signal Assignment" on page 35 , "HOME Seeking Pattern: HOMETYP 12 (Sensor-less mode)" on page 87 and "8.6 Driver Current Position Reading (**NX** Series driver, **ESMC** controller)" on page 100.
- For use of the **SCX10**, the **ESMC** controller should be set to the "driver mode." If the **ESMC** controller is used as the "controller mode (factory setting)," unexpected motion may occur due to unmatched I/O assignments.



SW Settings
SINK/SOURCE：SOURCE
5V/24V　　 ：24V

## ■ ESMC Controller  Position Reading (Current Sink)

**Note**
- The I/Os for driver on the **SCX10** indicated below are set for "ESMC Position Reading (PRESET Enable)." The I/Os on the driver are also set for the preset enable. See "6.5.3 Change of Signal Assignment" on page 35 and "8.6 Driver Current Position Reading (**NX** Series driver, **ESMC** controller)" on page 100.
- For use of the **SCX10**, the **ESMC** controller should be set to the "driver mode." If the **ESMC** controller is used as the "controller mode (factory setting)," unexpected motion may occur due to unmatched I/O assignments.

**SCX10** | **ESMC**

| SCX10 pin | Signal | ESMC signal | ESMC pin |
|---|---|---|---|
| 3 | ASG+ | ASG2 | 22 |
| 16 | ASG- | /ASG2 | 23 |
| 4 | BSG+ | BSG2 | 24 |
| 17 | BSG- | /BSG2 | 25 |
| 5 | TIMD+ | | |
| 18 | TIMD- | | |
| 23 | ALARM | ALM | 2 |
| 11 | END/PR | END/OUTR | 4 |
| 24 | TIMS/P0 | TIM/OUT0 | 5 |
| 12 | LC/P1 | MOVE | 3 |
| 25 | MOVE | T-UP/OUT1 | 6 |
| | | OUT-COM | 1 |
| 13 | 5/24 V OUT | | |
| 6 | GND | I/O-GND | 19 |
| | | IN-COM1 | 18 |
| 19 | COFF | C.OFF | 10 |
| 7 | REQ | ACL/CK | 8 |
| 20 | ACL/DCL / CK | REQ | 30 |
| 8 | CS | (HOME)/PRESET | 17 |
| 21 | FREE | HMSTOP | 11 |
| 9 | MBFREE | - | 12 |
| 22 | TL | - | 13 |
| 10 | PRESET | - | 14 |
| | | - | 15 |
| | | - | 16 |
| | | - | 7 |
| | | FREE | 9 |
| 1 | PLS+/CW+ | FP+ | 31 |
| 14 | PLS-/CW- | FP- | 32 |
| 2 | DIR+/CCW+ | RP+ | 34 |
| 15 | DIR-/CCW- | RP- | 35 |

5 VDC
2.2 kΩ×3
GND

24 VDC
10 kΩ  3 kΩ

24 VDC
GND

SW Settings
SINK/SOURCE：SINK
5V/24V　　：24V

## ■ ESMC Controller  Position Reading (Current Source)

**Note**
- The I/Os for driver on the **SCX10** indicated below are set for "ESMC Position Reading (PRESET Enable)." The I/Os on the driver are also set for the preset enable. See "6.5.3 Change of Signal Assignment" on page 35 and "8.6 Driver Current Position Reading (**NX** Series driver, **ESMC** controller)" on page 100.
- For use of the **SCX10**, the **ESMC** controller should be set to the "driver mode." If the **ESMC** controller is used as the "controller mode (factory setting)," unexpected motion may occur due to unmatched I/O assignments.

## ■ NX Series Driver  Function Expansion (Current Sink)

**Note** The I/Os for driver on the **SCX10** indicated below are set for "NX Function Expansion." See "6.5.3 Change of Signal Assignment" on page 35, "8.6 Driver Current Position Reading (**NX** Series driver, **ESMC** controller)" on page 100 and "8.5 Torque Limiting/Push-motion Operation (**NX** Series driver, **AR** Series driver)" on page 97.

## ■ NX Series Driver  Function Expansion (Current Source)

**Note** The I/Os for driver on the **SCX10** indicated below are set for "NX Function Expansion."  See "6.5.3 Change of Signal Assignment" on page 35, "8.6 Driver Current Position Reading (**NX** Series driver, **ESMC** controller)" on page 100 and "8.5 Torque Limiting/Push-motion Operation (**NX** Series driver, **AR** Series driver)" on page 97.

**SCX10**

**NX**

| SCX10 signal | pin | NX signal | pin |
|---|---|---|---|
| ASG+ | 3 | ASG+ | 3 |
| ASG- | 16 | ASG- | 4 |
| BSG+ | 4 | BSG+ | 5 |
| BSG- | 17 | BSG- | 6 |
| TIMD+ | 5 | ZSG1+ | 7 |
| TIMD- | 18 | ZSG1- | 8 |
| | | ALM+ | 9 |
| | | ALM- | 10 |
| ALARM | 23 | WNG+/(MOVE+)/(MBC+) | 11 |
| | | WNG -/(MOVE -)/(MBC-) | 12 |
| END | 11 | END+ | 13 |
| | | END- | 14 |
| TIMS/P1 | 24 | READY+/(AL0+)/P-OUTR+ | 15 |
| | | READY-/(AL0-)/P-OUTR- | 16 |
| LC/P0 | 12 | TLC+/(AL1+)/P-OUT0+ | 17 |
| | | TLC-/(AL1-)/P-OUT0- | 18 |
| READY/PR | 25 | ZSG2+/(NEAR+)/(AL2+)/P-OUT1+ | 19 |
| | | ZSG2-/(NEAR-)/(AL2-)/P-OUT1- | 20 |
| 5/24 V OUT | 13 | | |
| GND | 6 | GND | 2 |
| | | GND | 21 |
| REQ | 19 | IN-COM | 22 |
| CON | 7 | S-ON | 23 |
| ACL/DCL / CK | 20 | CLR/ALM-RST/P-CK | 24 |
| TL | 8 | P-REQ | 25 |
| FREE | 21 | TL | 26 |
| M0 | 9 | M0 | 27 |
| M1 | 22 | M1 | 28 |
| PRESET | 10 | P-RESET | 29 |
| | | FREE | 30 |
| PLS+/CW+ | 1 | CW+/PLS+ | 31 |
| PLS-/CW - | 14 | CW -/PLS- | 32 |
| DIR+/CCW+ | 2 | CCW+/DIR+ | 35 |
| DIR-/CCW - | 15 | CCW-/DIR- | 36 |

5 VDC  2.2 kΩ×3  GND

10 kΩ  3 kΩ

SW Settings
SINK/SOURCE：SOURCE
5V/24V          ：5V

## ■ CRK/CMK Series Driver (Current Sink)

## ■ CRK/CMK Series Driver (Current Source)



SW Settings
SINK/SOURCE : SOURCE
5V/24V : 5V

∗1 **CRK** Series
∗2 **CMK** Series
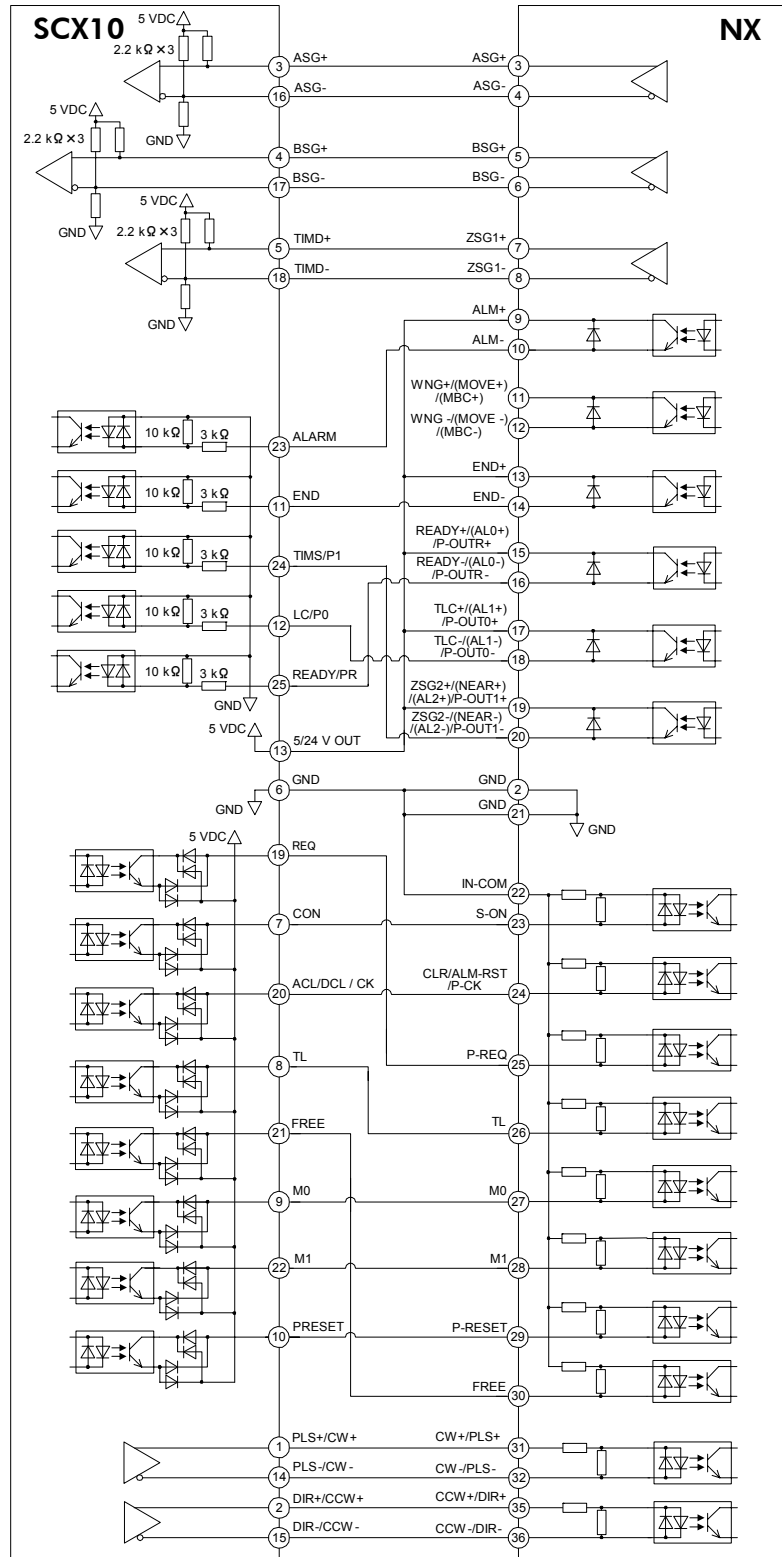
# 6.6 Connecting the RS-232C

The RS-232C connection can be used for all the operations including initial setup, test operation, program creation, I/O configuration and real time monitoring, using general terminal software or supplied utility software as well as user program. Everything you can perform via RS-232C can also be performed via USB, except daisy chain connection.

## ■ Specification

| Item | Description |
|------|-------------|
| Electrical characteristics | In conformance with RS-232C |
| Transmission method | Start-stop synchronous method, NRZ (Non-Return Zero), full-duplex |
| Data length | 8 bits, 1 stop bit, no parity |
| Transmission speed | 9600, 19200, 38400, 57600, 115200 bps (9600 is factory setting.) Selected by the BAUD parameter |
| Protocol | TTY (CR+LF) |

Terminal Specification

- ASCII mode
- VT100 compatible recommended
- Handshake: None
- Transmission CR: C-R
- Word wrap: None
- Local echo: None
- Beep sound: ON

Memo : All commanding to the **SCX10** can be made using general terminal software, such as Windows Hyper Terminal. For the quick start up, supplied utility software, the **Immediate Motion Creator for CM/SCX Series** is recommended. See "6.3 Connecting the USB and Installation of Utility Software" on page 20.

## ■ Connector and Applicable Lead Wire
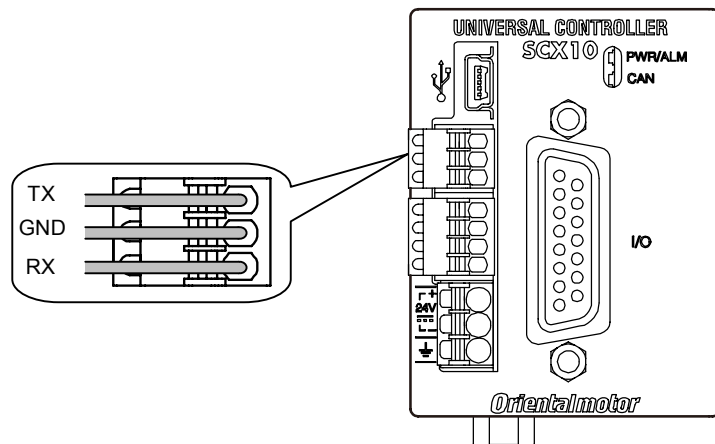


Connector and Applicable Lead Wire

| Connector | FK-MC 0,5/3-ST-2,5 (PHOENIX CONTACT) |
|-----------|--------------------------------------|
| Applicable lead wire size | AWG26 to 20 (0.14 to 0.5 mm$^2$) |

## ■ Connection Method

1. Strip the lead wire insulation by 8 mm.

2. Push the spring (orange) of the connector with a flat-tip screwdriver, to open a terminal port.
   Recommended flat-tip screwdriver: a tip of 2 mm in width, 0.4 mm in thickness

3. Insert the cable while pushing down the flat-tip screwdriver.

4. Release the flat-tip screwdriver. The lead wire will be attached.

## ■ Single Axis Connection

```
        PC
 (9pin COM Port)   SCX10
   ┌──────────┐  ┌──────┐
   │ 2 RX     │──│ TX   │
   │ 5 GND    │──│ GND  │
   │ 3 TX     │──│ RX   │
   ├──────────┤  └──────┘
   │ 4 DTR    │┐
   │ 6 DSR    │┘
   │ 7 RTS    │┐
   │ 8 CTS    │┘
   └──────────┘
```
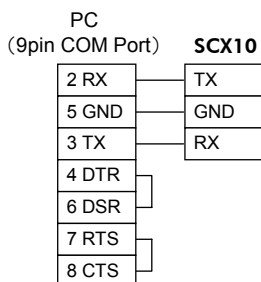
## ■ Daisy Chain Connection

You can connect multiple controllers with either of the two methods shown below.

Using only the RS-232C connector

Using RS-232C and CANopen connector
(If this method is used, it is not required to connect two lead wires to one GND terminal on the **SCX10**.)

```
        PC
 (9pin COM Port)   SCX10
   ┌──────────┐  ┌──────┐
   │ 2 RX     │──│ TX   │
   │ 5 GND    │──│ GND  │
   │ 3 TX     │──│ RX   │
   │ 4 DTR    │┐ ├──────┤
   │ 6 DSR    │┘ │ TX   │
   │ 7 RTS    │┐ │ GND  │
   │ 8 CTS    │┘ │ RX   │
   └──────────┘  ├──────┤
                 │ TX   │
                 │ GND  │
                 │ RX   │
                 └──────┘
```

```
        PC
 (9pin COM Port)   SCX10
   ┌──────────┐  ┌ ─ ─ ─ ─┐
   │ 2 RX     │──│ TX     │
   │ 5 GND    │──│ GND    │
   │ 3 TX     │──│ RX     │
   │ 4 DTR    │┐ │        │
   │ 6 DSR    │┘ │ CAN_H  │
   │ 7 RTS    │┐ │ SHIELD │
   │ 8 CTS    │┘ │ CAN_L  │
   └──────────┘  │ GND    │
                 └ ─ ─ ─ ─┘
                 ┌ ─ ─ ─ ─┐
                 │ TX     │
                 │ GND    │
                 │ RX     │
                 │        │
                 │ CAN_H  │
                 │ SHIELD │
                 │ CAN_L  │
                 │ GND    │
                 └ ─ ─ ─ ─┘
                 ┌ ─ ─ ─ ─┐
                 │ TX     │
                 │ GND    │
                 │ RX     │
                 │        │
                 │ CAN_H  │
                 │ SHIELD │
                 │ CAN_L  │
                 │ GND    │
                 └ ─ ─ ─ ─┘
```
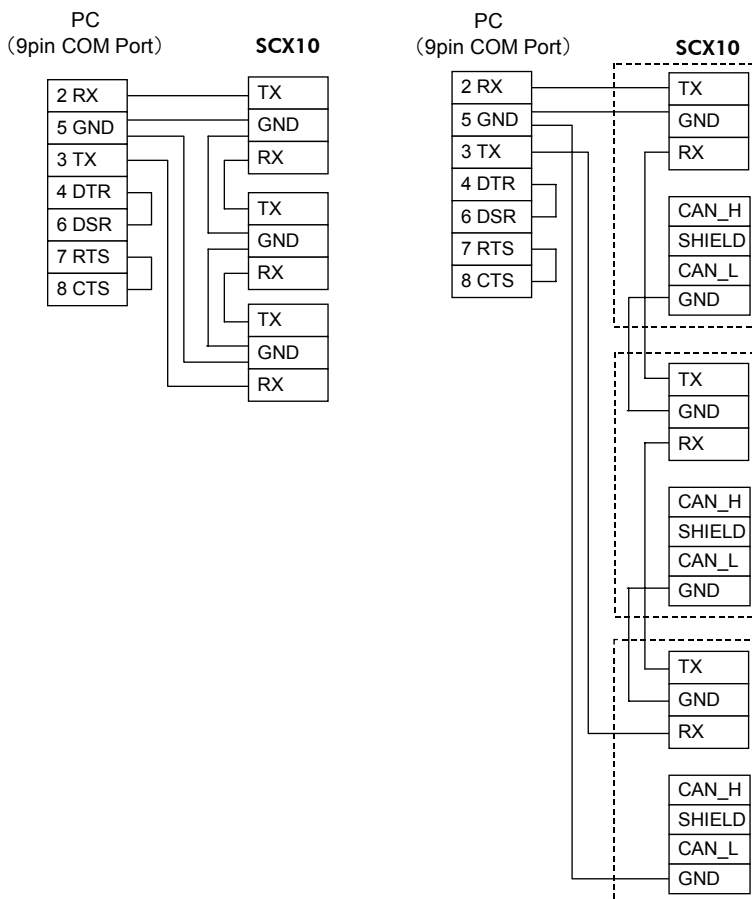
## 6.7  Connecting the CANopen

### ■ Connector, Applicable Lead Wire and Connecting Example of CANopen
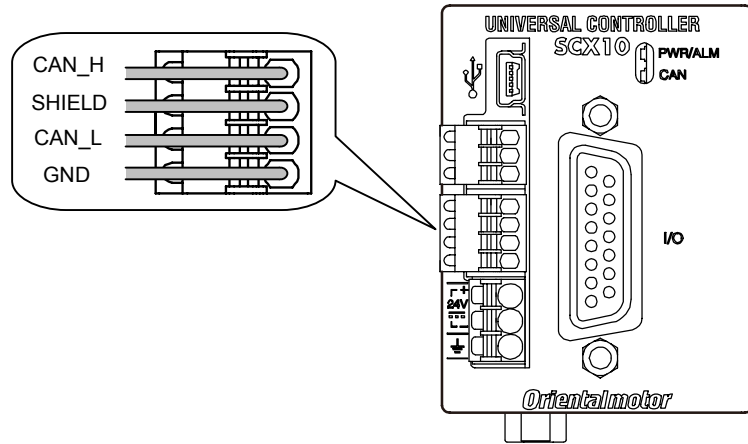


Connector and Applicable Lead Wire

| Connector | FK-MC 0,5/4-ST-2,5 (PHOENIX CONTACT) |
|---|---|
| Applicable lead wire size | AWG26 to 20 (0.14 to 0.5 mm$^2$) |
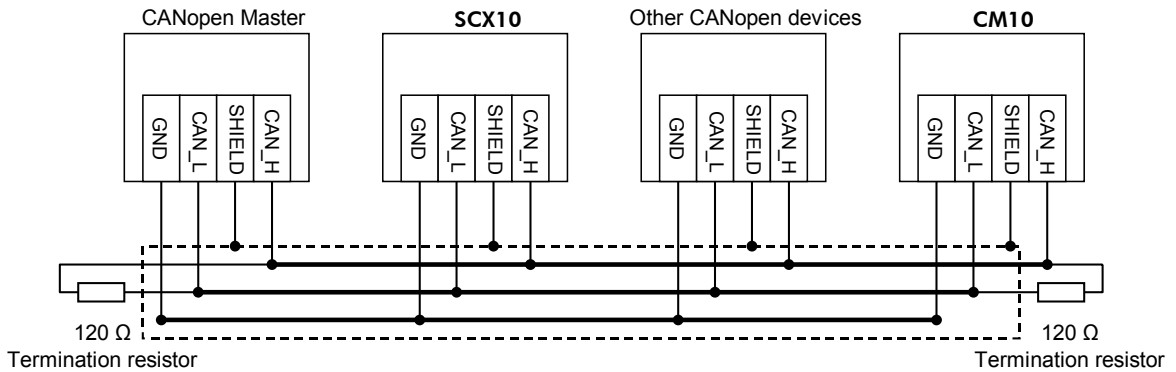


- The **SCX10** can be connected on the same network as other CANopen devices.
- Connect a terminating resistor (120 Ω 1/4 W) on both ends of the line. Termination resistors are not provided.

### ■ Connection Method

1. Strip the lead wire insulation by 8 mm.

2. Push the spring (orange) of the connector with a flat-tip screwdriver, to open a terminal port.
   Recommended flat-tip screwdriver: a tip of 2 mm in width, 0.4 mm in thickness

3. Insert the cable while pushing down the flat-tip screwdriver.

4. Release the flat-tip screwdriver. The lead wire will be attached.

## 6.8 Connecting the External Encoder

Use the terminals and housing provided in the package for the external encoder connection.

### ■ Connector and Lead Wire

| Connector housing | EHR-8 (manufactured by JST) |
|---|---|
| Terminals | BEH-001T-P0.6 (manufactured by JST) |
| Applicable lead wire size | AWG30 to 22 (0.05 to 0.33 mm$^2$) |
| Crimping tool | YC-260R (manufactured by JST) |

### ■ Pin Assignments and Signal Table
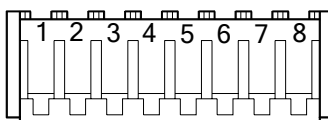


| Pin No. | Signal Name | Description |
|---|---|---|
| 1 | EXTA+ | External encoder ASG input |
| 2 | EXTA- | |
| 3 | EXTB+ | External encoder BSG input |
| 4 | EXTB- | |
| 5 | EXTZ+ | External encoder ZSG input |
| 6 | EXTZ- | |
| 7 | Encoder power + | External encoder power output (+5 V) |
| 8 | Encoder power - | External encoder power output (0 V) |

Note
- This encoder power output should be used for the external encoder only.
- Connect the encoder input power "-" (ground) line to the "Encoder Power -" terminal as instructed below. Do not connect it to the "GND" terminal on the other connectors on the **SCX10**. The **SCX10** has a dedicated encoder power supply and the protection circuit on the "Encoder Power -" line detects the over current. The maximum current is 150 mA.

### ■ Connection Example

- Line Driver Output Encoder Connection
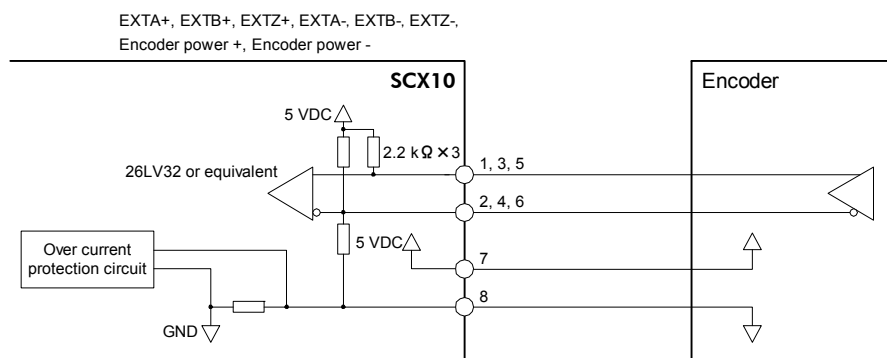
- Open Collector Output Encoder Connection (NPN Type)

EXTA+, EXTB+, EXTZ+,
Encoder power +, Encoder power -

SCX10

Encoder

5 VDC

2.2 kΩ × 3

26LV32 or equivalent

1, 3, 5

2, 4, 6

5 VDC

7

Over current
protection circuit

8

GND
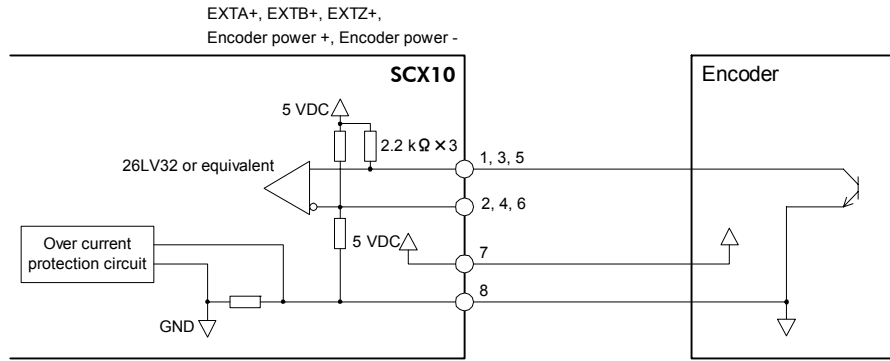
- Open Collector Output Encoder Connection (PNP Type)

Connect a pull-down resistor (470 Ω). The pull-down resistors are not provided.

EXTA+, EXTB+, EXTZ+,
Encoder power +, Encoder power -

SCX10

Encoder

5 VDC

2.2 kΩ × 3

26LV32 or equivalent

1, 3, 5

2, 4, 6

470 Ω

5 VDC

7

Over current
protection circuit

8

GND

- TTL Output Encoder Connection

EXTA+, EXTB+, EXTZ+,
Encoder power +, Encoder power -

SCX10

Encoder

5 VDC

2.2 kΩ × 3

26LV32 or equivalent

1, 3, 5

2, 4, 6

5 VDC

7

Over current
protection circuit

8

GND
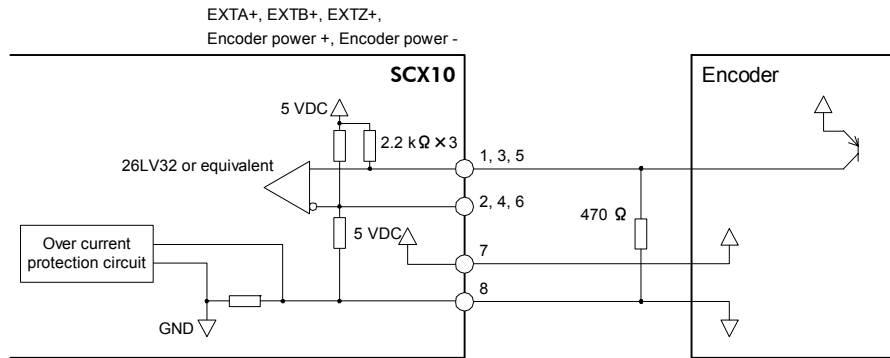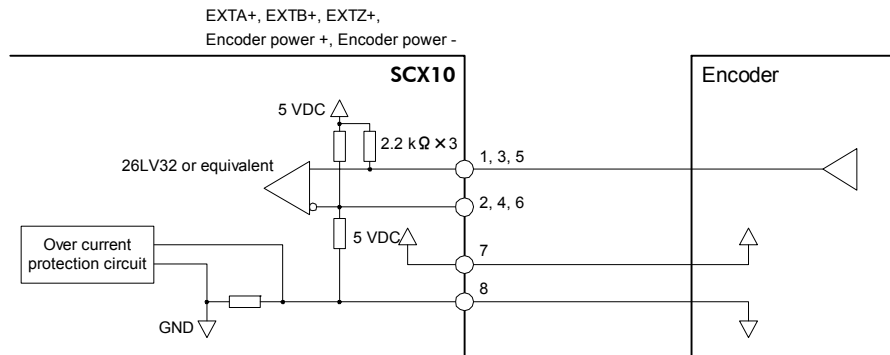
# 7  Start Up (Immediate Command)

This chapter explains the initial set up, how to operate the device immediately from the terminal as well as command format. These are basic skills and are applied to many modes and functions that the **SCX10** has. All users should read this chapter for start up.

> **Memo** | Start Up by command input is explained here, the method for using the supplied utility software (**IMC**) is introduced in a separate STARTUP MANUAL.

## 7.1  Overview

### ■ What is an Immediate Command?

You can operate the motor by sending commands immediately from the master controller such as a computer or PLC via RS-232C, USB or CANopen.
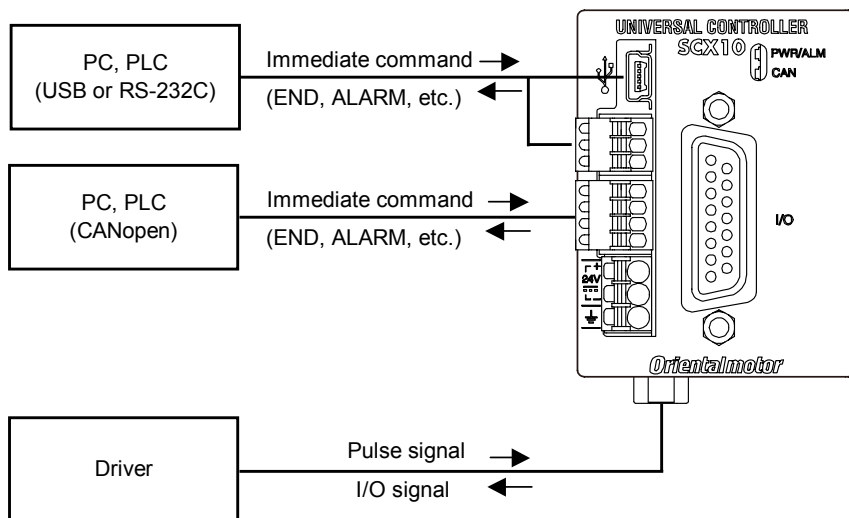


### ■ Contents

7.2  Preparation
7.3  Setting the User Unit
7.4  Making the Motor Move (Immediate Command)
7.5  Optional Settings for Driver
7.6  Command Format

## 7.2  Preparation

1. Make sure the driver is properly connected.  At a minimum, the pulse and direction lines (if the driver is set to one pulse mode) and the CON line (if the driver has this input) need to be connected.

2. Connect a personal computer via USB or RS-232C.
   See "6.3 Connecting the USB and Installation of Utility Software" on page 20 or "6.6 Connecting the RS-232C" on page 64 as necessary.

3. Power ON the **SCX10** and the driver.
   See "6.2 Connecting the Power Supply" on page 17 as necessary.

4. Launch the supplied utility software (**IMC**) or any general terminal software.
   See "6.3 Connecting the USB and Installation of Utility Software" on page 20 as necessary.

   <u>Communication Setting for the general terminal software</u>
   8 bits, 1 stop bit, no parity
   Baud rate: 9600 bps
   ∗ The default USB/RS-232C baud rate of the **SCX10** is 9600 bps.

> **Memo** | If DALARM is set to 1 and the power on timing of the driver is delayed from the **SCX10**, the LED on the **SCX10** becomes red and blinks (means alarm) until the driver power comes on. This action is normal and will be cleared automatically when the driver is powered on. For more detail, see ALM command on page 183.

# 7.3  Setting the User Unit

In the **SCX10**, the actual motion distance of the user application, such as "mm," "inch," "revolution" and "degree" is used, instead of the pulse unit that is commonly used in pulse generators and motor controllers. This is called "user unit" and is set initially by the following steps as with the motor resolution. (Pulse unit can also be used within this user unit system. See following pages.)

- DPR, UU: The number of user units per motor revolution is determined by the DPR parameter (Distance per Revolution), and the text information used for unit is set with UU (User Units). DPR should be configured before programming motions.  DPR can be set to any value between 0.500 and 51200.000 user units per motor revolution, in increments of 0.001.  Choose a meaningful value, appropriate for the application. (The factory settings, DPR=1 and UU=Rev, assuming motions are programmed in revolutions.)  Some examples appear on the following page.

- MR: The MR parameter must also be set according to the motor resolution that is set on the driver. (See the driver operating manual about the driver resolution and the setting.)

- Example

(Using "deg" unit, Motor resolution is 500/rev. )

1. Set the user unit

   Enter "UU=deg" and press the Enter key.

   (A space can be used and replaced with an equal sign between command and parameter.)

   ```
   >UU=deg
    UU=deg
   ```

2. Set the distance per revolution

   Enter "DPR=360," and press the Enter key.

   ```
   >DPR=360
    DPR=1(360) deg
    Position range = +/- 500000(499680)
    Velocity range = 0.001 - 1240(446400)
    Minimum Movable Distance = +/- 0.001(0.36)
   ```

3. Set the motor resolution

   Enter "MR=10000," and press the Enter key.

   ```
   >MR=10000
    MR=1000(10000)
    Position range = +/- 500000(499680)
    Velocity range = 0.001 - 1240(44640)
    Minimum Movable Distance = +/- 0.001(0.036)
   ```

4. Save to the EEPROM

   Enter "SAVEPRM," and press the Enter key.

   ```
   >SAVEPRM
    (EEPROM has been written 5 times)
    Enter Y to proceed, other key to cancel.
   ```

   Enter "Y," and press the Enter key.

   ```
    Enter Y to proceed, other key to cancel.Y
    Saving Parameters........OK.
   ```

5. Reset the system

   Enter "RESET," and press the Enter key.

   ```
   >RESET
    Resetting system.
   ```

   The parameter setting is finished.

   Check if the parameters are properly set.

6. Enter DIS and press the Enter Key

   ```
   >DIS
    DIS=0 deg
   ```

   Distance is now indicated in the user unit, "deg"

7. Enter VR and press the Enter Key

   ```
   >VR
    VR=1 deg/sec
   ```

   Velocity is now indicated in the user unit, "deg/sec"

**Note** The new value is shown in parenthesis after the active value. The new value will become effective only after saving (with SAVEPRM command) and resetting (with RESET command, or by cycling power) the device.

- Parameters for User Unit

| Parameter | Parameter Value | Function |
|---|---|---|
| DPR | 0.500 - 51200.000 (1) | Distance per revolution [user unit] {mm, deg, etc.} |
| MR | 10 - 51200 (1000) | Motor resolution [pulse/rev] |
| GA | 1 - 100 (1) | Electric gear {Numerator} |
| GB | 1 - 100 (1) | Electric gear {Denominator} |
| UU | String (Rev) | User unit text. 20 chars max. Cleared (NULL) by "UU 0" |
| DIRINV | 0, 1 (0) | 0: Motor rotates clockwise for positive distances |
| | | 1: Motor rotates counterclockwise for positive distances |

( ): factory setting

**Note** Changing DPR and MR changes the physical distances and velocities of any previously programmed motions. Generally, DPR and MR should be configured once, and not changed afterward, unless the mechanics of the application also change.

**Memo**
- When setting the output shaft of the motor gearhead or transmission of the equipment to the reference of user unit (one revolution), the motor can be operated by compensating the gear ratio electronically. The electrical gear ratio is set via the GA and GB parameters and the applied gear ratio equals GA/GB. When a 3:1 reduction gearhead is used, set GA=3 and GB=1. (The numerator and the denominator of the electric gear are set to opposite to the mechanical gear. The motor must rotate 3 times as far to complete one revolution at the gearhead output. Therefore, the GA value is 3 to compensate for the gear ratio's reduction in distance and velocity.)
- Electronic gearing can also be used when the distance per revolution is less than 0.5 (less than lower limit value of the DPR), or when the exact ratio cannot be specified (can not be divided) in three decimal places (e.g. if the distance per revolution is 1/3 user unit). Setting DPR=1, electronic gear numerator GA=3 and denominator GB=1 will result in three motor rotations per one user unit, for an effective DPR of exactly 1/3 user unit.
- If electronic gearing is used, DPR represents the distance per revolution of the output of the gear head (or other transmission device).
- The convention for positive vs. negative motion and torque can be changed with DIRINV.
- User unit text can be suppressed by setting UU to 0 (zero).

## ■ Application Examples

- Ball screw, lead 10 mm (Desired unit: mm)
  UU=mm, DPR=10
- Ball screw, lead 10 mm, with 10:1 gear (Desired unit: mm)
  UU=mm, DPR=1
  or UU=mm, DPR=10, GA=10, GB=1
- Ball screw, lead 10 mm, with 3:1 gear (Desired unit: mm)
  ∗ Distance per motor revolution approximately 3.333, exact value cannot be set with DPR alone
  UU=mm, DPR=10, GA=3, GB=1
- Rotating table (Desired unit: Revolution)
  UU=Revolution, DPR=1
- Rotating table, with 100:1 gear (Desired unit: Degree)
  UU=Degree, DPR=360, GA=100, GB=1
- Rotating table, with 3:1 gear (Desired unit: Degree)
  UU=Degree, DPR=120
  or UU=Degree, DPR=360, GA=3, GB=1

- Examples when combining the **SCX10** with actuator products -

• Products which resolution is 0.01 mm when combining with the **ESMC** controller (**EZSII** Series, **EZCII** Series, **EZA** Series)

  UU=mm, DPR=1, MR=100

  (Since the resolution of the actuator/**ESMC** controller is based on one millimeter while the resolution of other motors/drivers are based on one revolution. Therefore, the "distance per revolution" parameter on the **SCX10** works as a "distance per millimeter" for the **ESMC** controller.)

• Products which resolution is 0.01 mm when combining with the **ESMC** controller  (unit: inch)

  UU=inch, DPR=3.937, GA=100, GB=1, MR=100

  (Since 1 inch equals to 25.4 mm, "1 divided by 25.4" (0.0393700…..) is the travel distance of "1 mm" in terms of the user unit. Since DPR can use up to 3 decimal places, an electronic gear is used here in order to increase the accuracy by using as greater number of digits as possible.)

• Products which resolution is 0.001 mm when combining with the **ESMC** controller (**ESR** Series, **SPR4**, **PWA8**)

  UU=mm, DPR=1, MR=1000

• Products which lead is 12 mm when combining with the **EAS** Series

  UU=mm, DPR=1, MR=1000, GA=3, GB=250

• Products which lead is 6 mm when combining with the **EAS** Series

  UU=mm, DPR=1, MR=1000, GA=3, GB=500

• When combining with the **LAS** Series Rack and Pinion Systems (example of **LAS2B500**)

  UU=mm, DPR=9.997 (Multiply the resolution by the travel distance per one pulse for the **LAS2B500** described in the operating manual of the **LAS** Series.), MR=500 (Factory Setting)

  (When using all of five digits in order to increase more accuracy, using the electronic gear, set GA=10 and GB=1, and then set to DPR=99.974.)

• When combining with the **DG/DGII** Series Hollow Rotary Actuators (unit: Rev)

  UU=Rev, GA=18, GB=1 (The electronic gear is an inverse number of the gear ratio 18:1), DPR=1 (Because the electronic gear is used), MR=1000 (Factory Setting)

• When combining with the **DG/DGII** Series Hollow Rotary Actuators (unit: deg)

  UU=deg, GA=18, GB=1, DPR=360, MR=1000

## ■ Parameter Range and Least Input Increment

The maximum position range in user units is –500,000 to +500,000 and the maximum speed range in user units is 2,147,483.647.  Both can be used to three decimal places (0.001).

When the DPR, MR, ER, GA, GB is changed, position and velocity ranges also change due to internal calculation limits and pulse output speed limit.  The new ranges are shown when the DPR, MR, ER, GA, GB is changed.  The values can also be queried independently using the MAXVEL and MAXPOS: The MAXVEL is the maximum value for any velocity-based parameter, and position-based parameters must be between −MAXPOS and +MAXPOS.

## ■ How to Use Pulse Unit

Pulse unit can also be configured in this user unit system.  Set the DPR equal to the MR (motor resolution). Ex. Set the UU=Pulse, MR =1000, DPR=1000.

> **Note**  When DPR=MR, the maximum position that can be commanded is limited to 500,000 (pulses). If a greater maximum position (number of output pulses) is required, set DPR to a lower number and keeping in mind that the number of output pulses is the user unit multiplied by (MR/DPR).
>
> Ex.  Set the UU=Pulse (1000x), MR=1000, DPR=1.
>
> When "500,000" user unit is commanded, 500,000,000 pulses are output. Since the user unit can be used with three decimal places (0.001), commanding the increment of 1 pulse can still be done.

# 7.4 Making the Motor Move (Immediate Command)

## ■ Set the Motor Current ON (Only for STRSW=0)

Enter "CURRENT=1," and press the Enter key.

(A space can be used and replaced with an equal sign between command and parameter.)

```
>CURRENT=1
 CURRENT=1
```

> **Memo** The motor current at system start can be set via the STRSW parameter 1 (ON) or 0 (OFF).
> The factory setting is STRSW=1 (ON).

## ■ Experience the Operation

### 1. Set the move distance

Enter "DIS=360," and press the Enter key.

```
>DIS=360
 DIS=360 deg
```

### 2. Set the running velocity

Enter "VR=360," and press the Enter key.

```
>VR=360
 VR=360 deg/sec
```

### 3. Make the motor move

Enter "MI," and press the Enter key.

The motor starts to move in clockwise direction, and will rotate 1 revolutions in 1 second.

```
>MI
 >
```

### 4. Invert the direction

Enter "DIS= -360" and "MI," both followed by pressing the Enter key.

The motor will rotate 1 revolution in 1 second in reverse direction.

```
>DIS=-360
 DIS=-360 deg
>MI
 >
```

### 5. Change the running velocity

Enter "VR=720" and "MI," both followed by the Enter key.

The motor will rotate in 0.5 seconds.

```
>VR=720
 VR=720 deg/sec
>MI
 >
```

You now know the basics of incremental motion the **SCX10**. See "8.2 Motion Types" on page 77 for a variety of other motions.

# 7.5  Optional Settings for Driver

The **SCX10** can be connected with all Oriental Motor's standard pulse input drivers. Depending on a kind of a driver and necessary function/setting, please set the following parameters.

∗ Setting can be done by using the [System Config] tab screen on the supplied **Immediate Motion Creator for CM/SCX Series** utility software. (A setting by commands can be executed using commercially available terminal software as well.)

## ■ Parameters

The following parameters should be set according to the driver and your needs.

| IMC | Command | Range | Page |
|---|---|---|---|
| Motor Resolution | MR | 10 to 51200 | 290 |
| Pulse Mode | PULSE | 0: 2 pulse mode<br>1: 1 pulse mode | 314 |
| Startup Current | STRSW | 0: Current off at system start<br>1: Current on at system start | 343 |
| Encoder Resolution | ER | 10 to 51200 | 244 |
| Encoder Source | ENC | 0: Not used<br>1: Driver encoder<br>2: External encoder | 237 |
| Driver Alarm | DALARM | 0: Not-use the ALARM signal input<br>1: Use the ALARM signal input | 206 |
| Driver End | DEND | 0: Internal end area<br>1: Driver END signal | 211 |
| TIMD Logic Level<br>(Tab: Driver I/O) | TIMDLV | 0: Positive<br>1: Negative | 365 |
| Driver Ready | DREADY | 0: Disable<br>1: Enable | 229 |
| Home Deviation Clear | HOMEDCL | 0: Not clear driver deviation counter at homing<br>1: Clear driver deviation counter at homing<br>2: Not clear driver deviation counter at homing and the deviation is maintained precisely as the deviation in the **SCX10** will not be cleared at homing | 250<br>86 |

## ■ Example Settings

Example of settings that matches to the each driver with factory settings is shown below. Setting should be changed according to the driver settings and your needs.

| Settings on IMC | Factory Settings | AR | AS/ARL | RK | RBK | ESMC | NX | CRK | CMK |
|---|---|---|---|---|---|---|---|---|---|
| Motor Resolution | 1000 | 1000 | 1000 | 500 | 200 | 100 | 1000 | 500 | 200 |
| Pulse Mode | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Startup Current | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| Encoder Resolution | 1000 | 1000 | 1000 | n/a | n/a | 100 | 1000 | n/a | n/a |
| Encoder Source | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| Driver Alarm | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| Driver End | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| TIMD Logic Level<br>(Tab: Driver I/O) | 1 | AC input: 1<br>DC input: 0 | 0 | n/a | n/a | n/a | 1 | n/a | n/a |
| Driver Ready | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| Home Deviation Clear | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

**Memo**  The **AR** Series driver and the **NX** Series driver are designed so that the motor current at driver start up is OFF. The motor current can be controlled by the **SCX10** using the CURRENT parameter.  While the state of the CURRENT at system start up of the **SCX10** is set to "ON," it can be changed using the STRSW parameter.

## 7.6  Command Format

This section shows the command format.  Case {Upper/Lower} of the character does not a matter unless specified. Decimal point number is accepted in some of the parameters.

**Note** The decimal point is defined as "." (period), and "," (comma) cannot be used.

**Memo** See "Appendix A How to Send Commands Using ASCII Strings" on page 374 for information on ASCII string construction and for a automated communication.

### ■ 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] [Space] [Parameter value (constant)]

• Examples

| Condition | Example | Remarks |
|---|---|---|
| Parameter value is constant | `DIS=1.234, DIS 1.234` | "=" can be replaced with the space |
| Parameter value is variable (Available in sequence only) | `DIS=A` | "=" is required |
| Parameter value is equation (Available in sequence only) | `DIS=A*1.5` | "=" is required |

### ■ Commands

Spacing between command and argument (if needed argument) by at least a space is required.

• Format

[Command] [Space] [Argument]

• Examples

| Condition | Example | Remarks |
|---|---|---|
| No argument | `MI` | - |
| Argument is constant | `MA 1.234` | "=" is not accepted |
| Argument is variable | `MA POS[1]` | "=" is not accepted |
| Argument is string | `RUN Test` | "=" is not accepted |

### ■ Multiple-statement 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. This is applicable in all conditions such as in a sequence program.

• Example

```
>DIS 1.234; VR 3; TA 0.5; TD 0.1; MI
```

**Memo** VERBOSE parameter defines the response display. The following shows some example.

```
VERBOSE=1 (Factory Setting)    VERBOSE=0
>PC                            >PC
 PC=0.123 Rev                   0.123
>DIS=5.678                     >DIS=5.678
 DIS=5.678 Rev                  5.678
>HOMETYP                       >HOMETYP
 HOMETYP=0                      0
```

**Note** The ECHO command defines the echo back ON/OFF for entered ASCII data. Factory setting 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 329 for SAS command).

# 8  Features

This chapter introduces the main features of **SCX10**.

## 8.1  Overview

The **SCX10** device is designed to make motion control simple and convenient.  At the same time, the system has the versatility to adopt various types of operation, powerful features to maximize performance of the motor and driver, and support functions to accelerate successful system integration. The following subjects are discussed in the sections which follow:

### ■  Contents

| | | |
|---|---|---|
| 8.2 | Motion Types | 8.2.1  PTP (Point-to-Point) Motions<br>8.2.2  Linked Motions<br>8.2.3  Continuous Motions<br>8.2.4  Electrical Home Position and Mechanical Home Position<br>8.2.5  Mechanical Home Seeking |
| 8.3 | Stopping Motion and Sequence | variety of stopping types, pausing, and system status after stopping |
| 8.4 | Teaching Positions | teaching and storing positions |
| 8.5 | Torque Limiting/Push-motion Operation (**NX** Series driver, **AR** Series driver) | setting the driver, setting the **SCX10**, performing torque limiting/push-motion operation |
| 8.6 | Driver Current Position Reading (**NX** Series driver, **ESMC** controller) | setting the driver, setting the **SCX10**, reading the driver current position, releasing the absolute position loss alarm/setting the home position |
| 8.7 | Multi Axis Operation | how to configure multi axis operation |
| 8.8 | END (motion end) Signal | definition of END Signal (source), output of END signal, how END signal/status is Internally used in the **SCX10** |
| 8.9 | Encoder Function | monitor position error, detect miss-steps and other encoder functions |
| 8.10 | Math/Logical/Conditional Operators | mathematical/logical operators, conditional operators |
| 8.11 | User Variables | user variables, user-defined variables |
| 8.12 | View and Test Functions | monitoring and testing I/O, monitoring parameters |
| 8.13 | Protective Functions | controlling the system response to alarm conditions |

> **Note**  DPR and MR should be set before making any motions.  See "7.3 Setting the User Unit" on page 70.

## 8.2 Motion Types

The **SCX10** supports three basic types of motion: point-to-point motions, continuous motions, and electrical and mechanical home seeking. Also, linked operation that combines multiple sets of PTP-motion (point to point) is possible. This section explains each of these basic motion types.

## 8.2.1 PTP (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. 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

| Command/Parameter | Argument/Parameter Value | Function |
|---|---|---|
| MA | −MAXPOS - +MAXPOS | Start absolute motion to the specified destination [user unit] |
| MI | None | Start incremental motion, distance DIS |
| DIS | −MAXPOS - +MAXPOS (0) | Distance for incremental motion [user unit] |
| VR | 0.001 - MAXVEL (1) | Running velocity [user unit/sec] |
| VS | 0 - MAXVEL (0.1) | Starting velocity [user unit/sec] |
| TA | 0.001 - 500.000 (0.5) | Acceleration time [sec] |
| TD | 0.001 - 500.000 (0.5) | Deceleration time [sec] |

( ): factory setting

> **Note** See "8.3 Stopping Motion and Sequence" on page 93 for information on stopping motions before they finish.

> **Memo**
> - See the description of "8.2.2 Linked Motions" (below) for information on more complex motion profiles.
> - The minimum output frequency on the **SCX10** is 1 Hz. If the running velocity in a user unit is set equivalent to less than 1 Hz, the actual pulse output frequency becomes 1 Hz.

- Point-to-Point Motion Types

Two positioning modes are available for use in the positioning operation: absolute mode and incremental mode. In the absolute mode, set the target position by the distance from electrical home. For example, if absolute operation is performed from the electrical home toward the target position 100, it will stop after moving a distance of 100. However, if the current position is 30 from the electrical home, it will move a distance of 70 and stop. The operation is performed by the command and argument which shows target position.

    Example: MA 100      (Start absolute move to target position 100)

In incremental mode, set the target position by the distance from the current position. Since each device destination becomes the starting point for the next movement, this mode is suitable when the same distance is repeatedly used. Travel distance is set by the parameter "DIS" in advance, and the operation is performed by MI command.

    Example: DIS 100      (Set the distance to 100)
             MI           (Start move incremental)
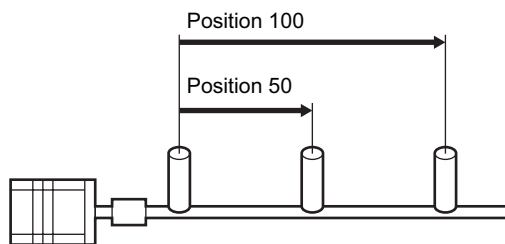
• Absolute Mode                    • Incremental Mode
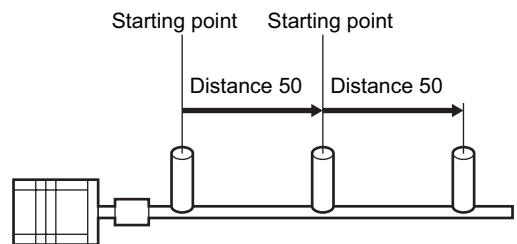


Before starting operation, set VR (running velocity), VS (starting velocity), TA (acceleration time), TD (deceleration time).

- Example

Conditions:
Ball screw: lead 10 mm (See "7.3 Setting the User Unit" on page 70.)
Distance: 60 mm (Incremental)
Running velocity: 5 mm/sec
Starting velocity: 1 mm/sec
Acceleration time: 0.5 sec
Deceleration time: 0.5 sec

```
>DIS=60
 DIS=60 mm
>VR=5
 VR=5 mm/sec
>VS=1
 VS=1 mm/sec
>TA=0.5
 TA=0.5
>TD=0.5
 TD=0.5
>MI
>
```
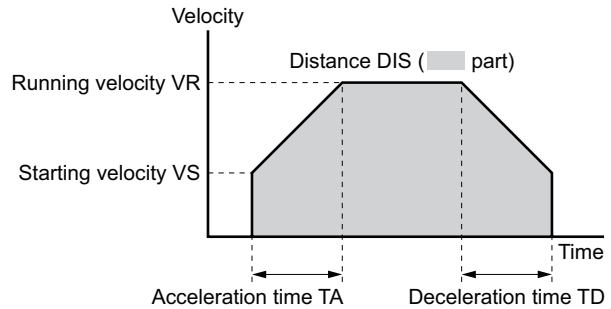


## 8.2.2 Linked Motions

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) | −MAXPOS - +MAXPOS (0) | Distance or destination for link segment 'x' [user unit] |
| VRx (x=0 - 3) | 0.001 - MAXVEL (1) | Running velocity of link segment 'x' [user unit/sec] |
| INCABSx (x=0 - 3) | 0, 1 (1) | Link type for link segment 'x'<br>0: Absolute<br>1: Incremental |
| LINKx (x=0 - 2) | 0, 1 (0) | Link control for link segment 'x'<br>0: segment terminates linked motion<br>1: motion continues with next segment |
| VS | 0 - MAXVEL (0.1) | Starting velocity [user unit/sec] |
| TA | 0.001 - 500.000 (0.5) | Acceleration time [sec] |
| TD | 0.001 - 500.000 (0.5) | Deceleration time [sec] |

( ): factory setting

**Note** See "8.3 Stopping Motion and Sequence" on page 93 for information on stopping motions before they finish.

**Memo**
- Acceleration and deceleration times TA and TD are the same for each segment.
- Link segments can be absolute or incremental, but all segments must execute in the same direction.
- Linked motions cannot be paused and resumed: PAUSE causes a soft stop, and CONT is ignored.
- The minimum output frequency on the **SCX10** is 1 Hz. If the running velocity in a user unit is set equivalent to less than 1 Hz, the actual pulse output frequency becomes 1 Hz.

- Example

Conditions:
Number of linked segments: 2
Link segment 0: Distance: 20 mm, Running velocity: 3 mm/sec
Link segment 1: Distance: 60 mm, Running velocity: 5 mm/sec
Starting velocity: 1 mm/sec

```
>DIS0=20
 DIS0=20 mm
>DIS1=60
 DIS1=60 mm
>VR0=3
 VR0=3 mm/sec
>VR1=5
 VR1=5 mm/sec
>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=1
 VS=1 mm/sec
>MI0
>
```
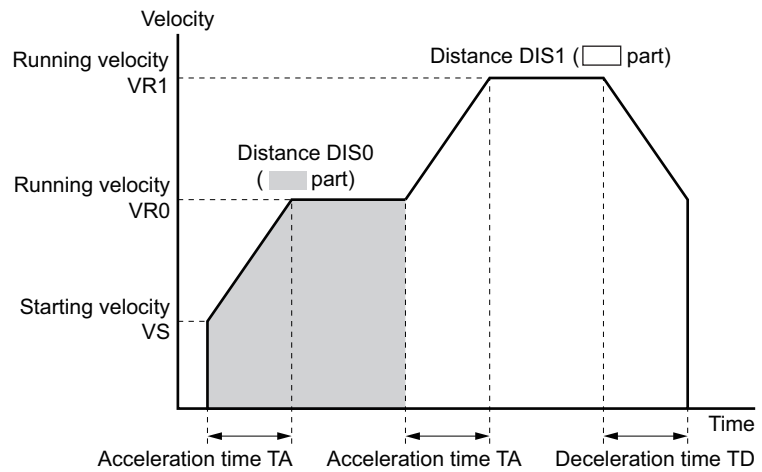


## 8.2.3 Continuous Motions

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 changed by a new (continuous) motion command, a stop command, or input signal.
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 executes.

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

- Commands and Parameters for Continuous Operation

| Command/ Parameter | Argument/ Parameter Value | Function |
| --- | --- | --- |
| MCP | None | Move continuously positive |
| MCN | None | Move continuously negative |
| VR | 0.001 - MAXVEL (1) | Running velocity [user unit/sec] |
| VS | 0 - MAXVEL (0.1) | Starting velocity [user unit/sec] |
| TA | 0.001 - 500.000 (0.5) | Acceleration time [sec] |
| TD | 0.001 - 500.000 (0.5) | Deceleration time [sec] |
| SENSORACT | 0 - 3 (2) | SENSOR input action<br>0: Hard stop<br>1: Soft stop<br>2: Soft stop at fixed distance from SENSOR signal<br>3: No action |
| SCHGPOS | 0 - MAXPOS (0) | Distance from SENSOR input to the stop position [user unit] if SENSORACT=2 |
| SCHGVR | 0.001 - MAXVEL (1) | Velocity after SENSOR input [user unit/sec] if SENSORACT=2 |

( ): factory setting

> **Note** See "8.3 Stopping Motion and Sequence" on page 93 for information on stopping motions before they finish.

> **Memo** The minimum output frequency on the **SCX10** is 1 Hz. If the running velocity in a user unit is set equivalent to less than 1 Hz, the actual pulse output frequency becomes 1 Hz.

- Example

  Conditions:
  Ball screw: lead 10 mm (See "7.3 Setting the User Unit" on page 70.)
  Running velocity: 5 mm/sec
  Starting velocity: 1 mm/sec
  Direction: Positive

```
>VR=5
 VR=5 mm/sec
>VS=1
 VS=1 mm/sec
>TA=0.5
 TA=0.5
>MCP
>
```
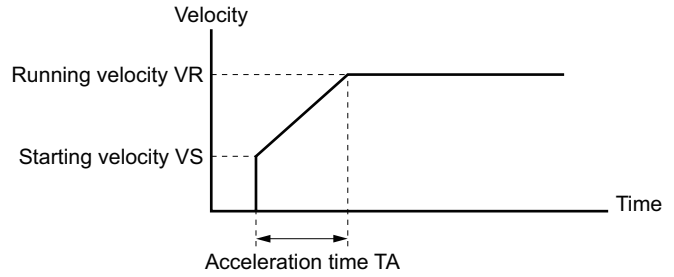


- SENSOR Action

  If the SENSOR input is configured, it 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. If SENSORACT=3, the system does not stop. See "8.3 Stopping Motion and Sequence" on page 93 for information on hard stops and soft stops. The picture below illustrates stopping action when SENSORACT=2.
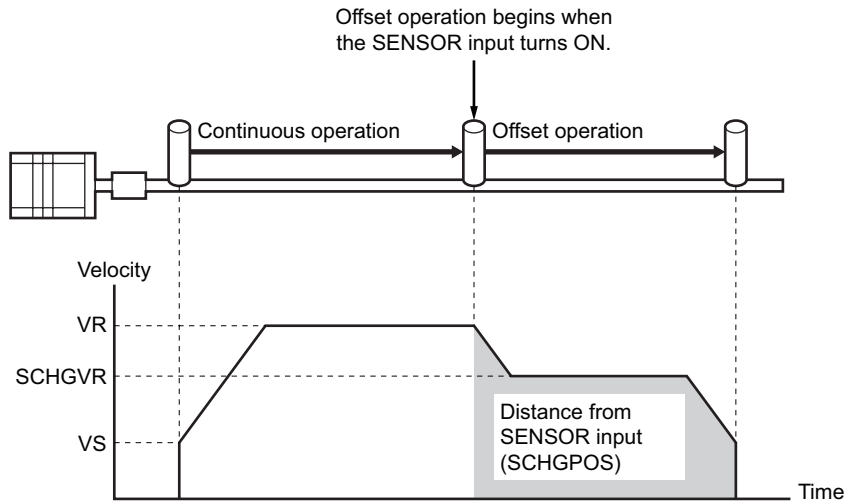
## 8.2.4 Electrical Home Position and Mechanical Home Position

The **SCX10** is operated based on the position command (PC). The position command can be read and indicates the current position.
When the **SCX10** is powered on or reset, PC is set to position zero (0).

- Setting of the Electrical Home Position

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, or then moves a predefined distance (OFFSET) 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: PC is reset to zero (0).
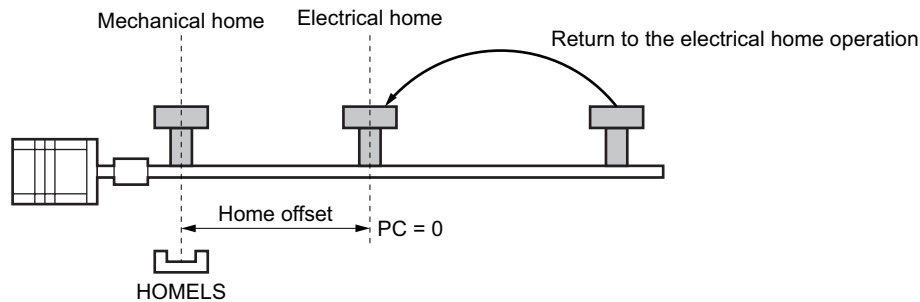


The electrical home position can be set in an arbitrary position regardless of the mechanical home position. When executing the PRESET (reset home position) command, the current position will be set to the electrical home position and reset to "PC=0." Also, if the EHOME (return-to-electrical home operation) command is executed when the electrical home position has never been set, the position of "PC=0" will be set to the electrical home position.

- Function of the Electrical Home Position

Once the electrical home position is set, the software position limits (LIMP, LIMN) will become effective. Whenever returning to the electrical home position, the HOMEP (home position) output of the I/O connector will be turned ON (when assigned).

- Returning to the Electrical Home Position

When returning to the electrical home position from an arbitrary position, use the EHOME (return-to-electrical home operation) command or command "MA 0" (move to the absolute position 0). The EHOME operating pattern is determined by VR (operating speed), VS (starting speed), TA (acceleration time) and TD (deceleration time) as well as other operations.

- Commands and Parameters Related to the Electrical Home Position

| Command/Parameter | Parameter Value | Description |
|---|---|---|
| PC | −MAXPOS - +MAXPOS (0) | Reference and setting of the position command (current position) |
| PRESET | - | Reset of the position command, setting current position to be the electrical home position |
| EHOME | - | Start return-to-electrical home operation (When the electrical home position is not set, setting PC=0 position to be the electrical home position) |
| OUTHOMEP | 0 - 4 (0) | Assign the HOMEP (home position) output to the OUTx of the I/O connector (the parameter 0 is unassigned) |
| HOMEPLV | 0, 1 (0) | Home position output logic (0: Normally open, 1: Normally closed) |
| SIGHOMEP | 0, 1 | Home position output status (0: Not-active, 1: Active (at home)) |
| LIMP | −MAXPOS - +MAXPOS (0) | Setting of software position limit (Positive direction) |
| LIMN | −MAXPOS - +MAXPOS (0) | Setting of software position limit (Negative direction) |
| SLACT | 0, 1(0) | Software position limit enable (0: Disabled, 1: Enabled) |

( ): factory setting

**Note**
- See "8.3 Stopping Motion and Sequence" on page 93 for information on stopping motions before they finish.
- An arbitrary value can be written to PC but avoid a careless change because the reference point of all operation will be changed.
- Return-to-electrical home operation cannot be paused and then resumed: PAUSE causes a soft stop, and CONT is ignored.

**Memo**
- When turning on the power or when resetting, PC is set to zero (PC=0), but the electrical home position will not be set. Similarly, although "PC=0" is commanded at an arbitrary position, the electrical home position will not automatically be set. To use the software position limit and home position output functions without executing mechanical home seeking operation, set the electrical home position using EHOME or PRESET.
- When combining with a driver that has a function to read the current position (**NX** Series driver or **ESMC** controller etc.) and if the reading driver current position/updating internal position (ABSREQPC) is performed, the electrical home position will be set without executing mechanical home seeking operation. See "8.6 Driver Current Position Reading (**NX** Series driver, **ESMC** controller)" on page 100.

## 8.2.5 Mechanical Home Seeking

Mechanical home seeking is an operation in which the motor moves in a specific pattern, seeking a valid mechanical home position determined by external signals. Thirteen patterns are available, differing in their signal requirements and response. See the HOMETYP table (below) and the motion chart for each homing type on pages 89 to 92.

The SENSOR input and the TIM input can be used to increase the repeatability of the final home position. The TIM input is connected to the TIM (excitation timing) output of the stepping motor driver, and is considered ON in fifty (50) or one hundred (100) fixed, evenly spaced locations per motor revolution. If a SENSOR and/or a TIM input are used, they are ANDed with the designated home position signal to form a valid mechanical home input signal set.

- Commands and Parameters for Mechanical Home Seeking

| Command/ Parameter | Argument/Parameter Value | Function |
|---|---|---|
| MGHP | None | Move go home positive |
| MGHN | None | Move go home negative |
| OFFSET | −MAXPOS - +MAXPOS (0) | Offset for mechanical home seeking [user unit] |
| HOMETYP | 0 - 12 (0) | Mechanical home seeking mode: see table "HOME Seeking Pattern" |
| VR | 0.001 - MAXVEL (1) | Running velocity [user unit/sec] |
| VS | 0 - MAXVEL (0.1) | Starting velocity [user unit/sec] |
| TA | 0.001 - 500.000 (0.5) | Acceleration time [sec] |
| TD | 0.001 - 500.000 (0.5) | Deceleration time [sec] |
| HOMEDCL | 0 - 2 (0) | Select the deviation counter clear during mechanical home seeking operation (refer to P.86 in details) |
| SENSORACT | 0 - 3 (2) | SENSOR input action<br>0: Hard stop<br>1: Soft stop<br>2: Soft stop at fixed distance from SENSOR signal<br>3: No action<br>If the SENSOR input, which is used in mechanical home seeking operation, is not used in normal operation, set "SENSORACT=3." |

( ): factory setting

**Note**
- See "8.3 Stopping Motion and Sequence" on page 93 for information on stopping motions before they finish.
- Return-to-electrical home operation cannot be paused and then resumed: PAUSE causes a soft stop, and CONT is ignored.

**Memo**
- Mechanical home seeking normally uses starting velocity VS for the final approach to the home signal(s). If VS=0, the final approach will be internally set to 0.001 user unit. If the TIM signal is used and VS is more than 200Hz pulse output speed, the final approach will be 200 Hz=200/MR "user unit/sec."
- The minimum output frequency on the **SCX10** is 1 Hz. If the running velocity in a user unit is set equivalent to less than 1 Hz, the actual pulse output frequency becomes 1 Hz.
- System END signal is referenced at the end of each motion. An error occurs if the END signal is not found. "See 8.8 END (motion end) Signal" on page 105 in more detail of END signal.
- The TIM (excitation timing) signal of the driver is based on position command (or set point), not on position feedback information.
- The ACL/DCL signal on the driver connector of the **SCX10** is momentarily output when a limit sensor (+LS or -LS) is found during mechanical home seeking, and it clears the deviation counter in the driver for an immediate stop.

- HOME Seeking Pattern

| HOMETYP | Home Position Detector | | | | Mode (Motion Pattern) |
| --- | --- | --- | --- | --- | --- |
| | HOME | +LS, −LS | SENSOR | TIM (excitation timing) | |
| 0 | Not used | Required for valid home | - | - | 2-sensor mode (See p.89) |
| 1 | | | - | Required for valid home | |
| 2 | | | Required for valid home | - | |
| 3 | | | Required for valid home | Required for valid home | |
| 4 | Required for valid home | Reverse direction | - | - | 3-sensor mode (See p.90) |
| 5 | | | - | Required for valid home | |
| 6 | | | Required for valid home | - | |
| 7 | | | Required for valid home | Required for valid home | |
| 8 | | Stop: Alarm | - | - | 1-sensor mode (See p.91) |
| 9 | | | - | Required for valid home | |
| 10 | | | Required for valid home | - | |
| 11 | | | Required for valid home | Required for valid home | |
| 12 | Not used | Not used | - | - | Sensor-less mode (See p.92) ∗ See also p.87. |

- Example: Mechanical Home Seeking with HOMETYP=4

Conditions:
Ball screw, lead 10 mm (See "7.3 Setting the User Unit" on page 70.)
Running velocity: 5 mm/sec
Starting velocity: 1 mm/secStarting direction: positive
Acceleration time: 0.1 sec
Deceleration time: 0.1 sec

```
>HOMETYP=4
 HOMETYP=4
>VR=5
 VR=5 mm/sec
>VS=1
 VS=1 mm/sec
>TA=0.1
 TA=0.1
>TD=0.1
 TD=0.1
>MGHP
>
```

- **Operation and Function at Mechanical Home Seeking**

When mechanical home seeking operation is completed (when the operation for OFFSET is completed if OFFSET is set), the final position will be redefined as the new electrical home position, and the position counter PC will be reset to zero (0). (Refer to the previous section "Electrical home position and mechanical home position") Once the electrical home position is set, the software position limit (LIMP, LIMN) will become effective. Whenever returning to the electrical home position, the HOMEP (home position) output will be turned ON (when assigned).

When connecting to a driver that has a function to read the current position, **SCX10** resets the driver internal position to the home position. (The PRESET output of the driver connector on the **SCX10** is turned ON)

| Note | When connecting to a driver that has a function to read the current position and using the driver current position reading function, assign the PRESET (reset home position) output to the driver connector on the **SCX10**, and set the PRESET input of the driver to be enabled. (Refer to page 100 "8.6 Driver Current Position Reading (**NX** Series driver, **ESMC** controller)" in details)   When the parameter for OFFSET in the driver is set to other than 0 (zero), execute a ABSREQPC command after completing the mechanical home seeking operation or executing PRESET command. EC (encoder count), PF (feedback position) and PC (position command) in the **SCX10** are aligned with the HOME parameter of the driver. However, if setting the electrical home position other than the mechanical home position is required as described above, it is recommended to set the offset value in the **SCX10** (use the OFFSET command) but not in the driver. |
|---|---|
| | (Under "HOMETYP=12," the reset of the home position will automatically be performed in the driver after completing mechanical home seeking operation. Even when using the driver current position reading function, the PRESET signal is not required in mechanical home seeking operation and the PRESET signal from the **SCX10** will not output.) |

- **Selection of the Timing Source When Using the Timing Signal**

The TIM (timing) signal that is used in mechanical home seeking operation can be selected to be either the driver timing signal (Z-phase output signal in case of a servo motor) or Z-phase signal of an external encoder. Also, the timing signal of the driver can be selected either the single-ended signal or differential signal. The ENC (encoder selection) command and TIM (timing signal selection) command are used to select.

TIMING Source Selection

| TIMING Source | ENC | TIM |
|---|---|---|
| Timing signal・Z-phase pulse differential input TIMD | 1 (Driver) | 0 (TIMD/EXTZ) |
| Timing signal・Z-phase pulse single ended input TIMS | Unrelated | 1 (TIMS) |
| External encoder ZSG EXTZ | 2 (External encoder) | 0 (TIMD/EXTZ) |
| No source is selected* | 0 (Not used) | 0 (TIMD/EXTZ) |

∗ If ENC is set to 0 (zero) and TIM is set to 0, the system alarm status will be active when executing MGHP or MGHN command when the selected home seeking type requires the timing signal.

Signal Flow Path

Timing signal・Z-phase pulse differential input    TIMD-------1

**ENC**--------0 (TIMD/EXTZ)

External encoder ZSG                                         EXTZ-------2         **TIM**-----------------→TIMING signal

Timing signal・Z-phase pulse single ended input  TIMS--------------------1 (TIMS)
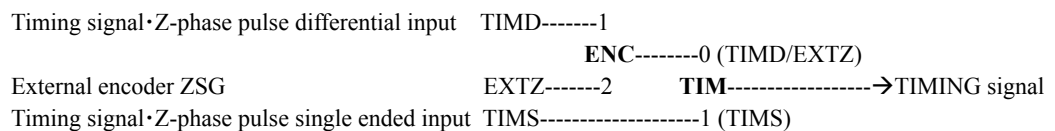
- HOMEDCL (deviation counter clear select at mechanical home seeking operation)

    Select whether the deviation counter of the **SCX10** and/or driver is cleared at the time of detecting the home position, and you can perform an accurate home seeking operation with a servo motor or you can acquire accurate feedback information when using an encoder.

    Setting of HOMEDCL

    | Setting Value | Deviation Counter of the **SCX10** | Deviation Counter of the Driver |
    |---|---|---|
    | 0 | Clear | - |
    | 1 | Clear | Clear |
    | 2 | - | - |

    factory setting: 0 (zero)

    > **Note**
    > - When HOMEDCL is set to 1, ACL/DCL (driver alarm clear/deviation counter clear) output on the **SCX10** is required to be connected to the deviation clear input on the driver (CLR/ALM-RST input if the **NX** Series driver is used).
    > - When HOMEDCL is set to 2, both deviation counters are not reset when limit swiches are found or PSTOP (panic stop) is performed even under other than mechanical home seeking operation. The motor may not stop immediatly depending on the setting of velocity filter in the driver when uing $\alpha_{STEP}$ product.

- Example 1: When using a stepping motor with an encoder (including the $\alpha_{STEP}$, the **ESMC** controller)
    → Set to "HOMEDCL=0."

    When there is no problem for little deviation caused by a load, or when there is no load at the time of mechanical home seeking operation even if accurate deviation is necessary, set to "HOMEDEL=0." When performing mechanical home seeking operation, the deviation will automatically be cleared, then PC (position command) and PF (feedback position) will be identically set to "0 (zero)."
    The motor shaft may turn up to ±3.6° when turning on the power because the first state of the excitation sequence for the stepping motor (excluding the $\alpha_{STEP}$) is pre-determined in the driver. PC will follow PF when executing the motor current OFF, but the motor shaft may shift when executing the motor current ON the next time since the motor is excited to the last excitation sequence at the time of turning the motor current off regardless of the motors' current position. In these cases, PC will not change, but PF will change to the appropriate value. If "HOMEDCL=0" is set, the deviation counter will be cleared when a mechanical home seeking operation is performed, and PC and PF will become the same.
    In case of the $\alpha_{STEP}$, the current shaft position is excited when turning on the power or executing the motor current ON. Therefore, the above-mentioned problem does not occur, but the motor shaft may become free from generated alarms for overload or others. In this case, if the motor shaft will rotate, PF will change while PC will not change. However, if "HOMEDCL=0" is set, the deviation will be cleared when executing mechanical home seeking operation, then PC and PF will become same.

- Example 2: When using a servo motor
    → Set to "HOMEDCL=1."

    Since the servo motor has a deviation counter in the driver, the motor may not stop immediately even if the pulse input is stopped. If "HOMEDCL=1" is set, the driver deviation counter will be cleared simultaneously when detecting the home position sensor. Therefore, it will be able to stop at the position of the home position sensor accurately. The deviation counter in the **SCX10** is also cleared simultaneously, and the driver deviation is reflected to PE (position deviation) of the **SCX10** correctly.

- Example 3: When an accurate deviation is required to be observed with the stepping motor plus encoder (including the $\alpha_{STEP}$, **ESMC** controller driver) while the load is applied to the shaft at mechanical home seeking operation
    → Set to "HOMEDCL=2."

    The stepping motor generates its torque by shifting the rotor position to within 1.8 degree corresponding to the amount of load. When the load is always applied to the shaft, such as a gravity load in vertical axis, the motor excitation position and rotor position are out of alignment even if mechanical home seeking operation was performed. It is an expected condition that the position command (PC) and feedback position (PF) are out of alignment the same exact amount. If "HOMEDCL=2" is set, the deviation is maintained precisely as the deviation in the **SCX10** will not be cleared at mechanical home seeking operation. However, in advance, PC and PF are required to be matched without a load on the motor shaft (the excitation position and rotor position to be the same). Execute PECLR (deviation counter clear) in a no load condition. The PC value will be matched the PF value without a deviation. Then, perform mechanical home seeking operation. When it is difficult to do the above in vertical drive (gravitational operation), refer to the following steps.

When using the $\alpha_{STEP}$ : Once execute the motor current OFF (CURRENT=0), PC and PF will be the same (PC=PF) in no load condition. Then, execute the current ON and perform mechanical home seeking operation.

When the stepping motor other than the $\alpha_{STEP}$ is combined with an encoder: Measure the correct value of the deviation in advance, such as comparing the difference of deviations (PE) between PC and PF in horizontal condition and vertical condition. In the current ON state, execute PECLR (deviation counter clear), overwrite the PF value with the value of the measured deviation added to the current PF value when the motor current is ON. This is required to be done whenever executing the motor current ON if the motor current may be OFF.

**Memo**
- When using a stepping motor without using an encoder or feedback information, the setting of HOMEDCL is unrelated and is not required.
- A servo motor or the $\alpha_{STEP}$ will not misstep. Therefore, there is no need to see the deviation (HOMEDCL=2) in general.
- When HOMETYP is set to use the driver timing signal during mechanical home seeking operation with the $\alpha_{STEP}$ , set HOMEDCL to "0" or "2." If HOMEDCL is set to "1," the motor shaft may be shifted from the timing signal position by the delay with the speed filter in the $\alpha_{STEP}$ driver.

- Sensor-less Mechanical Home Seeking Operation for "HOMETYP=12"

"HOMETYP=12" is sensor-less (push-motion) type mechanical home seeking operation, and can be used only when combining the **SCX10** with the **ESMC** controller. See the following steps to use.

**Memo** This mode is available only when the controller is used in combination with an **EZSII** Series, **EZCII** Series or **EZA** Series actuator.

1. Check the HOME input of the driver is effective

   Setting to HOME in the I/O parameter for "HOME/PRESET switching" is required. The factory setting is the HOME input. Use the **EZT1** teaching pendant or the **EZED2** data editing software for checking and setting.

2. Set the home parameters of the driver

   Set the return direction, home offset, return method, starting speed of return and operating speed of return.

   **Memo** Check that the "Push-motion 1" or "Push-motion 2" is selected as the return method of the driver. The factory setting for the **EZSII** Series, **EZCII** Series and **EZA** Series is "Push-motion 1."

3. Assign the necessary signals in the driver connector on the **SCX10**

   Set to select "ESMC Sensor-less Home Seeking & Position Reading" using the "Auto Configuration" under the [System Config] tab - the [Driver I/O] tab in the provided utility software (**IMC**). Click the [SAVE and RESET] button and the necessary I/O will be set automatically.

   Commands, which are input via the keyboard, are as follows.

   | Command | Parameter Value (Factory Setting) | Description |
   |---|---|---|
   | DINEND | 0 - 5 (2) | Assign the DINx input of the driver connector on the **SCX10** to the END (motion end) input (Connect to the END output of the driver) |
   | DINMOVE | 0 - 5 (0) | Assign the DINx input of the driver connector on the **SCX10** to the MOVE (motor moving) input (Connect to the MOVE output of the driver) |
   | DOUTHMSTOP | 0 - 8 (0) | Assign the DOUTx output of the driver connector on the **SCX10** to the HMSTOP (sensor-less mechanical home seeking operation stop) output (Connect to the HMSTOP input of the driver) |
   | DOUTHOME | 0 - 8 (0) | Assign the DOUTx output of the driver connector on the **SCX10** to the HOME (sensor-less mechanical home seeking operation start) output (Connect to the HOME input of the driver) |

   ∗ Parameter 0 indicates "unassigned" (assignment cancel).

4. Connect the I/O points that have been configured in the step 3 and the points on the **ESMC** controller.

Refer to the connection diagram of "ESMC Sensor-less Home Seeking & Position Reading" on page 56 to page 57.

5. Select "HOMETYP=12" and command mechanical home seeking operation

Both commands for MGHP and MGHN will be the same operation.

(Starting direction and speed of the operation depend on the HOME parameters that were set to the driver in "2.")

• HOME Seeking Pattern: HOMETYP 0 to 3 (2-sensor mode)

| Starting position | HOME starting direction +(CW)    (MGHP) | HOME starting direction −(CCW)    (MGHN) |
|---|---|---|
| -LS |  |  |
| +LS |  |  |
| -LS to +LS |  |  |
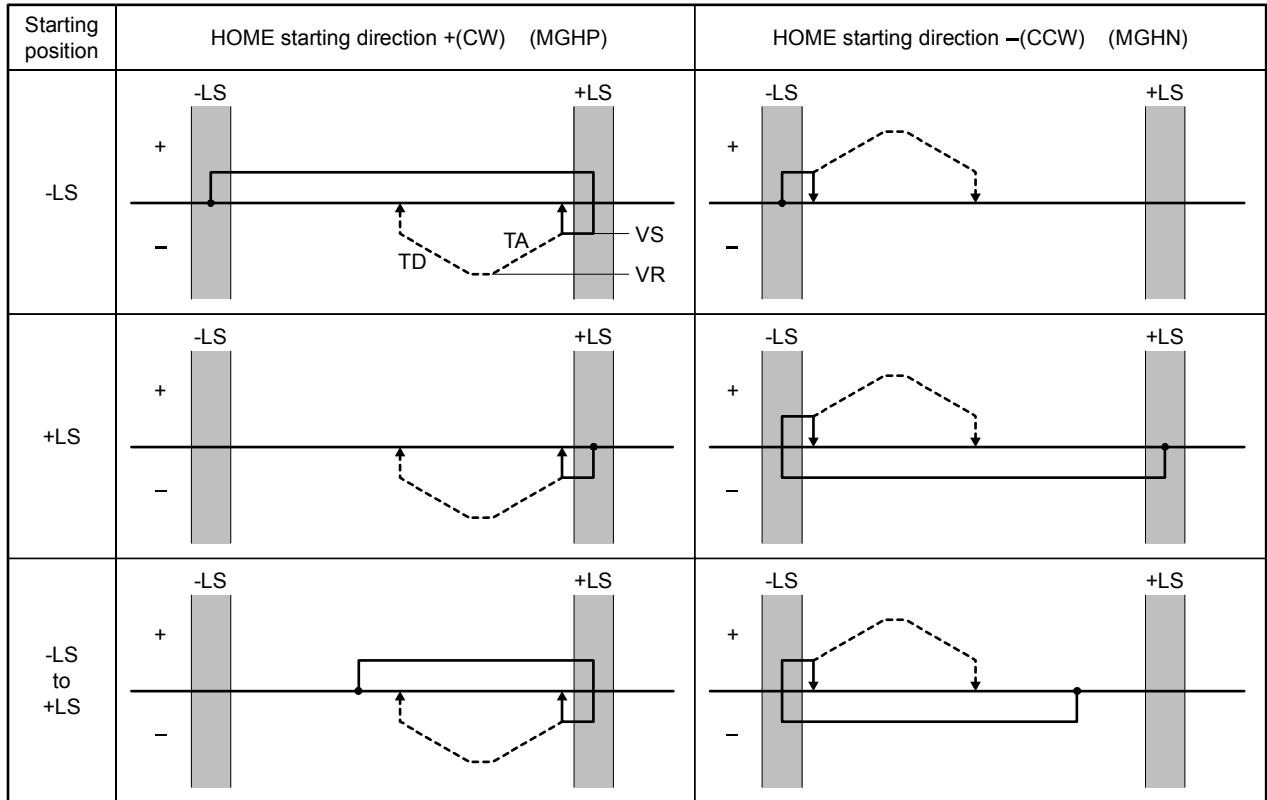
---- is operation with offset.

- HOME Seeking Pattern: HOMETYP 4 to 7 (3-sensor mode)

| Starting position | HOME starting direction +(CW)    (MGHP) | HOME starting direction −(CCW)    (MGHN) |
|---|---|---|
| -LS |  |  |
| +LS |  |  |
| HOMELS |  |  |
| HOMELS to -LS |  |  |
| HOMELS to +LS |  |  |
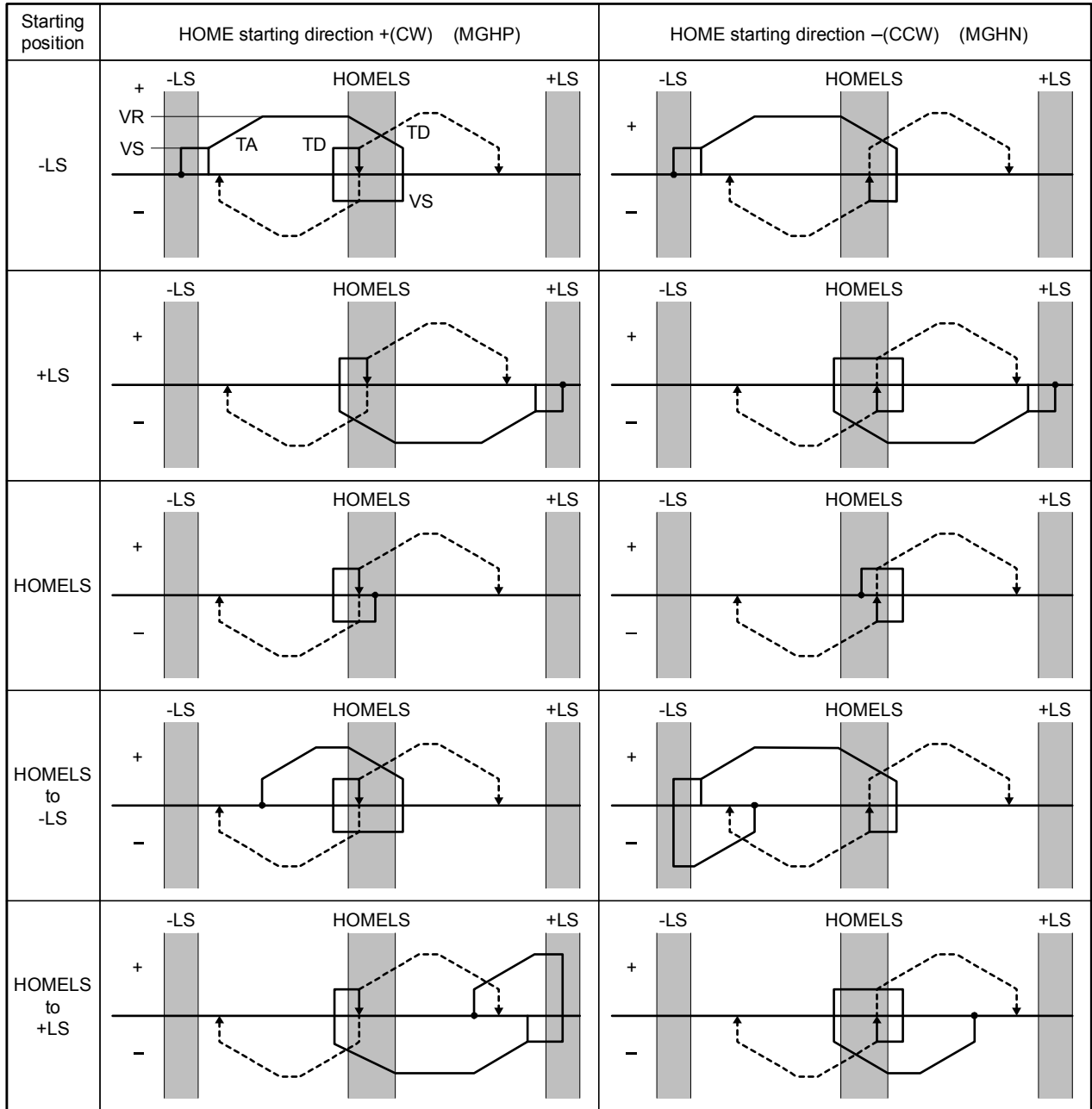
---- is operation with offset.

- HOME Seeking Pattern: HOMETYP 8 to 11 (1-sensor mode)
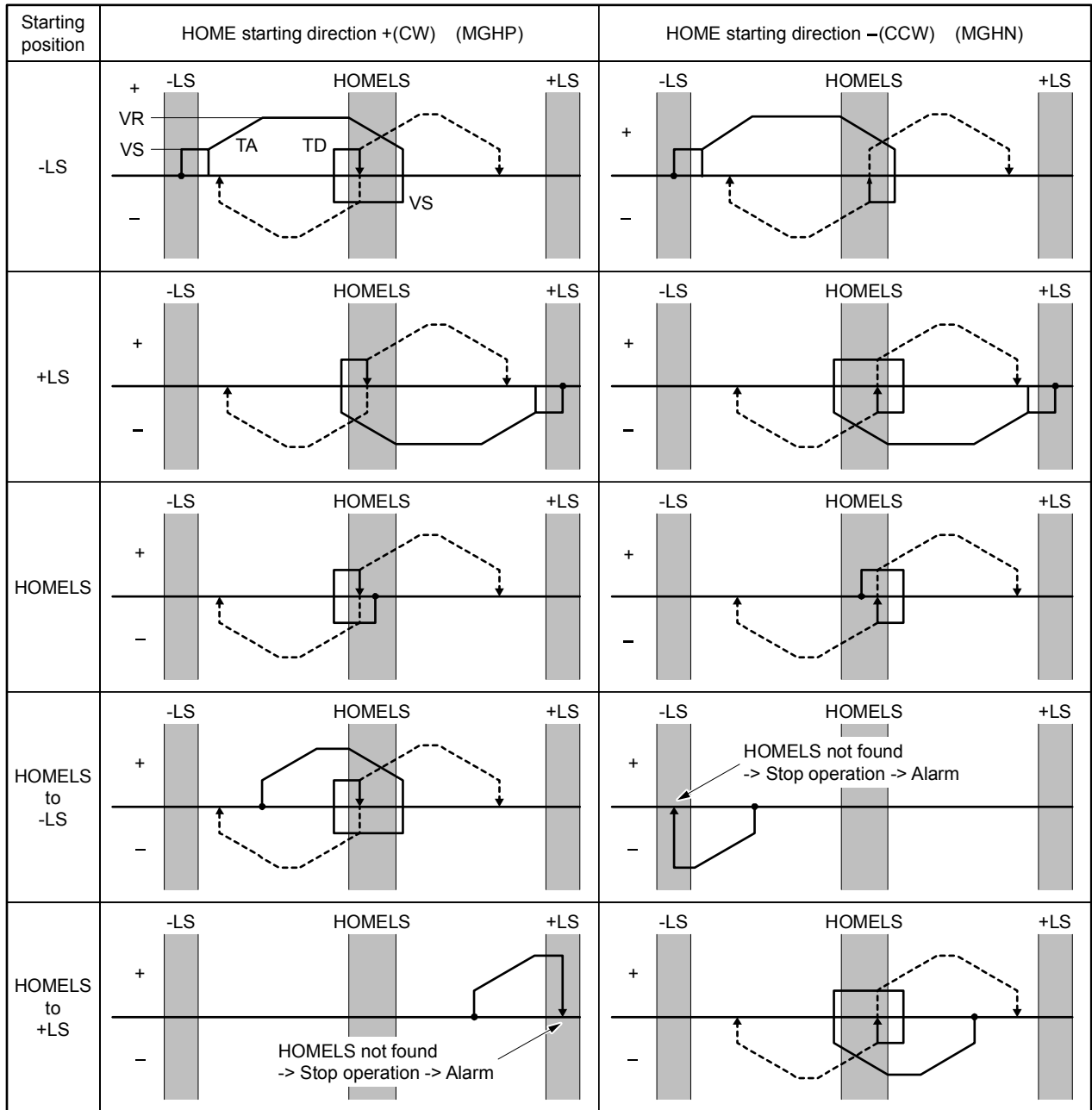


---- is operation with offset.

- HOME Seeking Pattern: HOMETYP 12 (Sensor-less mode)

| Sensor-less home seeking operation type | Starting direction of home seeking operation: Motor side | Starting direction of home seeking operation: Opposite the motor side |
|---|---|---|
| Push-motion1 |  |  |
| Push-motion2 (high speed) |  |  |

--- is operation with offset.

**Memo** The "Push-motion 1" or "Push-motion 2" is set at the **ESMC** controller.

## 8.3  Stopping Motion and Sequence

There are three ways to stop motion and sequence, command via serial communication ports, operate switch or PLC with I/O port, or CANopen communication.  Various stop actions including soft stop, hard stop, stop motion or stop sequence are also available as below.

- Commands and Parameters for Stopping Motion

| Command/ Parameter | Argument/ Parameter Value | Function | Motion | Sequence | Assignable to I/O Port |
|---|---|---|---|---|---|
| SSTOP | None | Soft stop: decelerate and stop according to TD | Yes | - | - |
| HSTOP | None | Hard stop: stop as quickly as possible | Yes | - | - |
| MSTOP | None | Motor stop: Hard or soft stop, determined by MSTOPACT | Yes | - | Yes |
| PSTOP | None | Panic stop: Hard stop, stop also sequence, motor current and alarm state after stop is defined by ALMACT | Yes | Yes | Yes |
| ABORT | None | Abort motion and sequence execution, soft stop | Yes | Yes | Yes |
| PAUSE | None | Pause motion (soft stop) | Yes | - | Yes |
| CONT | None | Continue motion (resume from pause) | Yes | - | Yes |
| START | None | Setting ON to OFF abort sequence (STARTACT=1) When sequence is running, paused motion is resumed (STARTACT=0). | - | Yes | Yes |
| PAUSECLR | None | Pause clear: While a sequence is running, only remaining portion of the paused motion is cleared and the next step of the sequence will be executed. | Yes | - | Yes |
| MSTOPACT | 0 - 1 (1) | Motor stop action (affects MSTOP command or MSTOP input) 0: Hard stop (MSTOP behaves as HSTOP) 1: Soft stop (MSTOP behaves as SSTOP) | - | - | - |
| ALMACT | 0 - 2 (2) | Alarm action (affects PSTOP command or PSTOP input) 0: No alarm triggered, motor current remains ON 1: Trigger alarm, motor current remains ON 2: Trigger alarm, motor current disabled | - | - | - |

( ): factory setting

**Memo**
- The ESC key or character stops motion and aborts sequences, similar to ABORT.
- The SSTOP, HSTOP, MSTOP and PAUSE do not stop the sequences.
- A motion that has been stopped with the PAUSE command or input on the system I/O connector (if assigned) can be continued (resumed) using the CONT command or input or the START input on the system I/O connector (if assigned). See the descriptions for CONT (page 200), STARACT (page 341), PAUSECLR (page 303) and PAUSE (page 302).
- The PAUSECLR command or PAUSECL input on the system I/O connector (if assigned) can be used to clear the remaining motion.  If this signal becomes activated while a sequence is running, only remaining portion of the current motion is cleared and the next step of the sequence will be executed, since the PAUSE does not stop the sequence.
- Linked motions, return-to-electrical home operation and mechanical home seeking cannot be paused and resumed: PAUSE causes a soft stop, and CONT is ignored.
- See "6.4.2 Input Signals" on page 25 for details on assigning signals to I/O ports.

## 8.4 Teaching Positions

The **SCX10** includes a position data array, which can hold up to 100 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[100].

- Example

```
>MA POS[1]        #Start absolute motion to the position specified by POS[1].
 W=POS[100]    #Assign the value of POS[100] to variable W. (in sequence only)
```

Set the position data using the [Teach/Jog] tab screen of the supplied utility software **Immediate Motion Creator for CM/SCX Series (IMC)** or the keyboard operation using any general terminal software.
When setting by using a keyboard, the position array data is 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 constantly monitors and displays system position command, in user units
- motor current and power to the electromagnetic brake (if used) can be toggled on and off
- if motor current is on, the system can be commanded to move, positively or negatively, in increments of jog distance (travel distance), or continuously at running velocity VR
- if motor current is off, the system can be repositioned using external force or torque
- (while motor current is off, the system keeps monitoring position via the encoder inputs ASG and BSG, and position command (PC) is replaced by feedback position (PF). This performs position teaching by hand. Accordingly, position becomes zero (0) if ASG and BSG inputs are not connected to the encoder or the encoder output of motor driver. Teaching by hand cannot be performed.)
- at any time, the system position can be stored to any location in the position data array

Motions use running velocity VR, starting velocity VS, and acceleration and deceleration times TA and TD.

| Note | • 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 when not in the teaching mode. |
| --- | --- |
| | • 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 CON 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 released. |
| B | Move negatively in increments that is set by "D" command (Jog negative) Jog motion in the negative direction (while key pressed) |
| N | Move positively in increments that is set by "D" command (Jog positive) Jog motion in the positive direction (while key pressed) |
| M | Move continuously, in the positive direction (while key pressed). Soft stop when key released. |
| Q | Toggle motor current OFF and ON |
| K | Set key interval detection time [millisecond] |
| D | Set jog distance in user unit (factory setting is 0.001) |
| F | When the driver has the FREE input : Toggle motor current and electromagnetic brake OFF and ON When the driver has the M.B.FREE input : Toggle electromagnetic brake OFF and ON |
| S | Save position array data to EEPROM (same as SAVEPOS) |
| <SPACE> | Hard stop |
| <Enter> | Store current position to a location in the position data array (System will prompt for location, 1-100) |
| <ESC> | Soft stop, terminate teaching |

**Note**
- For freeing the shaft, use the 'F' key when the driver has a FREE input , and use the 'Q' key when the driver has a M.B.FREE input. If the driver has a M.B.FREE input and is equipped with an electromagnetic brake, use both the 'Q' key and the 'F' key for freeing the shaft.
- Be careful when using the 'Q' key when an encoder is not connected. Once the 'Q' key is pressed, the current position is set to 0.

**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').
- Use an English keyboard for most comfortable key locations for the operation.

- Example

```
        *** Teach mode ***

(V)     : Move Cont. Neg.    (M)      : Move Cont. Pos.
(B)     : Jog Negative       (N)      : Jog Positive
(Q)     : Current ON/OFF     (F)      : FREE ON/OFF
(S)     : Save all data to EEPROM
(K)     : Change Key Interval (50-500[msec])
(D)     : Change Jog Distance (0.001-500000 [Rev])
             Minimum Movable Distance : 0.001 [Rev]
<Space> : Immediate Stop
<Enter> : Data entry mode (Input POS number, then <Enter>)
<ESC>   : Exit teach mode


PC=        0.000        ◄──────── Current position
```

After TEACH command

```
        *** Teach mode ***

(V)     : Move Cont. Neg.    (M)      : Move Cont. Pos.
(B)     : Jog Negative       (N)      : Jog Positive
(Q)     : Current ON/OFF     (F)      : FREE ON/OFF
(S)     : Save all data to EEPROM
(K)     : Change Key Interval (50-500[msec])
(D)     : Change Jog Distance (0.001-500000 [Rev])
             Minimum Movable Distance : 0.001 [Rev]
<Space> : Immediate Stop
<Enter> : Data entry mode (Input POS number, then <Enter>)
<ESC>   : Exit teach mode

                                      Set target position data
PC=        15.410    Save to POS[ ◄──  array location: Input "1"
```

After <ENTER> received while teaching

```
        *** Teach mode ***

(V)     : Move Cont. Neg.    (M)      : Move Cont. Pos.
(B)     : Jog Negative       (N)      : Jog Positive
(Q)     : Current ON/OFF     (F)      : FREE ON/OFF
(S)     : Save all data to EEPROM
(K)     : Change Key Interval (50-500[msec])
(D)     : Change Jog Distance (0.001-500000 [Rev])
             Minimum Movable Distance : 0.001 [Rev]
<Space> : Immediate Stop
<Enter> : Data entry mode (Input POS number, then <Enter>)
<ESC>   : Exit teach mode


PC=        15.410    Save to POS[1] Data set OK.
```

After <ENTER> received

```
        *** Teach mode ***

(V)     : Move Cont. Neg.    (M)      : Move Cont. Pos.
(B)     : Jog Negative       (N)      : Jog Positive
(Q)     : Current ON/OFF     (F)      : FREE ON/OFF
(S)     : Save all data to EEPROM
(K)     : Change Key Interval (50-500[msec])
(D)     : Change Jog Distance (0.001-500000 [Rev])
             Minimum Movable Distance : 0.001 [Rev]
<Space> : Immediate Stop
<Enter> : Data entry mode (Input POS number, then <Enter>)
<ESC>   : Exit teach mode


PC=        15.410    Saving POS[ ] Data.......OK.
```

After (S) received

## 8.5 Torque Limiting/Push-motion Operation (NX Series driver, AR Series driver)

For safety operation, the maximum output torque of the servo motor can be limited. Also, push-motion operation can be performed using a $\alpha_{STEP}$.

A driver that has the torque limiting function (the **NX** Series driver) is required when performing torque limiting, and a driver that has the push-motion operation function (**AR** Series driver) is required when performing push-motion operation.

### ■ Setting the Driver

Set the torque limiting value of the driver (position control mode), or set to perform push-motion operation. The data setter **OPX-2A** or data setting software **MEXE02** can be used to set the data.

- When using the **NX** Series driver:
  1. Set the analog input signal parameter [SyS-1-05] to "Disable."
  2. Recycle the power to the driver. (When the 24 VDC is used, also recycle it.)
  3. Set the desired torque limiting value to the operation data No. 0 to 3.

- When using the **AR** Series driver:
  1. Set "push-motion operation" using the application parameter for I/O input mode [APP-2-00].
     (The I/O mode of "positioning operation" under the factory setting will be changed to "push-motion operation," and the I/O terminal function of "CS (resolution selection) input" will be changed to "T-MODE (push-motion operation) input." Also, M0, M1 and M2 of the push-current select input will be enabled.)
  2. When setting the value of the push current other than the factory setting combination, set using the application parameter for push-motion current 0 [APP-2-05] to 7 [APP-2-12].

Memo : With the **NX** Series driver, the **SCX10** has partially overlapped with the analog I/O signals connector (CN6). Select "digital" for setting the torque limiting value. The digital selection of the torque limiting value is three points under the factory setting, but it can be extended to four points if the analog I/O signal is set to disable.

### ■ Setting the SCX10

When performing torque limiting/push-motion operation, when monitoring the operation status or when operating using the I/O terminals by an external signal, it is required to assign the functions to the driver connector on the **SCX10** and/or I/O connector as follows.

- Setting the Driver Connector on the **SCX10**

It is required to assign the TL (torque limiting/push-motion operation) output, M0/ M1/M2* output (operation data selection) and LC (limiting condition) to the driver connector on the **SCX10**.

∗ When using the **NX** Series driver, there is no need to assign the M2 input.

Execute the assignment using the "Auto Configuration" under the [System Config] tab - [Driver I/O] tab in the provided utility software (**IMC**). (See "6.5.3 Change of Signal Assignment" on page 35.)

- When using the **NX** Series driver, select "NX Function Expansion"
- When using the **AR** Series driver, select "AR Push-Motion"

After selecting, click "SAVE and RESET." The necessary I/O signals will automatically be assigned.

Commands, which are executed by operating the keyboard, are as follows.

| Command | Parameter Value (Factory Setting) | Description |
|---------|-----------------------------------|-------------|
| DINLC | 0 - 5 (4) | Assign the driver connector on the **SCX10** DINx to the LC (limiting condition) input. (Connect to the TLC output of the driver) |
| DOUTTL | 0 - 8 (7) | Assign the driver connector on the **SCX10** DOUTx to the TL (torque limiting/push-motion operation) output. (Connect to the TL/T-Mode input of the driver) |
| DOUTM0 | 0 - 8 (0) | Assign the driver connector on the **SCX10** DOUTx to the M0 (data select bit 0) output. (Connect to the M0 input of the driver) |
| DOUTM1 | 0 - 8 (0) | Assign the driver connector on the **SCX10** DOUTx to the M1 (data select bit 1) output. (Connect to the M1 input of the driver) |
| DOUTM2 | 0 - 8 (0) | Assign the driver connector on the **SCX10** DOUTx to the M2 (data select bit 2) output. (Connect to the M2 input of the driver) |

∗ Parameter 0 indicates "unassigned" (assignment cancel).

- Setting the I/O Connector

When commanding torque limiting/push-motion operation using the I/O connector, or when outputting the status if the motor torque reaches the set torque value/if the motor is in push-motion state, it is required to assign the signals. The signals can be set in the [I/O Setting] tab under the [System Config] tab using the provided utility software (**IMC**). Commands, which are set by operating the terminal, are as follows.

| Command | Parameter Value (Factory Setting) | Description |
| --- | --- | --- |
| INTL | 0, 1 - 9 (0) | Assign the I/O connector INx to the TL (torque limiting/push-motion operation) input |
| OUTLC | 0, 1 - 4 (0) | Assign the I/O connector OUTx to the LC (limiting condition) output |

∗ Parameter 0 indicates "unassigned" (assignment cancel).

**Memo** When selecting push-motion operation, the TL (torque limiting/push-motion operation) output of the driver connector on the **SCX10** is enabled. And the CS (resolution selection) output is canceled and disabled.

## ■ Connection

Connect the I/O points that have been configured in the above and the points on the **NX** Series driver or **AR** Series driver. (Refer to the connection diagram for **NX** Series driver on page 60 to page 61 or for **AR** Series driver on page 46 to page 47.)

## ■ Performing Torque Limiting/Push-motion Operation

See the following steps.

- When Performing Torque Limiting Operation

1. Select the desired torque limiting value using the DD (driver operation data) parameter.
2. When setting the TL command (torque limiting) to "1" (TL=1) or turning the TL input of the I/O connector ON while executing positioning operation, the maximum output torque is limited by the set torque limiting value.
   When reaching the set torque limiting value, The LC output of the I/O connector will be turned ON (you can notice also by the DSIGLC command).

- When Performing Push-motion Operation

1. Select the desired motor current value using the DD (driver operation data) parameter.
2. Set the TL command (push-motion operation) to "1" (TL=1) or turn the TL input of the I/O connector ON.
3. Operate CW or CCW direction.
   A load is pressed continuously by selected current value while turning TL ON. The push-force is increased with the deviation and if the push-motion operation is continued and the position deviation becomes 1.8 degrees or more, the LC output of the I/O connector will be turned ON (you can notice also by the DSIGLC command).
4. When the press operation is finished, stop the motion (stop pulses). After that, execute the deviation counter clear (PECLR) or move in the negative direction and return to the position where the motion started.
5. Set the TL command to 0 (zero) or turn the TL input of the I/O connector OFF.

| Command/Parameter | Parameter Value (Factory Setting) | Description |
| --- | --- | --- |
| TL | 0, 1 (0) | Performs torque limiting/push-motion operation while setting to "1." |
| DD | 0 - 3 (0) **NX** Series driver<br>0 - 7 (0) **AR** Series driver | Driver operation data |

**Memo**

- Since pulses are accumulated in the driver during push condition, a prolonged push condition may generate an excessive position deviation alarm of the driver. If the push condition continues for a prolonged period, stop the operation. You can check whether it is the push condition or not using the LC output or DSIGLC command.
- When performing push-motion operation, the LC output may be turned ON during acceleration/deceleration.
- The LC output is turned ON when the motor torque reaches 300% of rated torque even while the torque limiting function is not used.
- Torque limiting/push-motion operation can be performed via CANopen. Assign TL to any bit of the byte [0] in RPDO using the RINTL command in advance. To execute torque limiting/push-motion operation, set DD by the command code and start by the assigned TL. The limiting condition will be output by LC (byte [1], bit [4]) that is pre-assigned in the TPDO.
- TL can be set to 1 (ON) or 0 (OFF) by the command, I/O or CANopen, and the later command will be given priority.
- When setting the driver operation data "DD," the M0, M1 and M2 output of the driver connector on the **SCX10** will be changed as follows.

| DD | M2 | M1 | M0 |
|----|-----|-----|-----|
| 0 | OFF | OFF | OFF |
| 1 | OFF | OFF | ON |
| 2 | OFF | ON | OFF |
| 3 | OFF | ON | ON |
| 4 | ON | OFF | OFF |
| 5 | ON | OFF | ON |
| 6 | ON | ON | OFF |
| 7 | ON | ON | ON |

# 8.6  Driver Current Position Reading (NX Series driver, ESMC controller)

When combining with a driver that has a function to read the current position (**NX** Series driver or **ESMC** controller etc.), the current position data of the driver can be read by a dedicated command. The current position data is indicated by the user unit. The current position in the **SCX10** is automatically updated by the value read.

## ■ Setting the Driver

Connect the battery and set the driver to the status that can read the current position.

- When using the **NX** Series driver, set the switch on the front panel (SW1-3) to ON (enable the absolute function).
- When using the **ESMC** controller, you can use the factory setting, but if you use the preset (reset home position) function, select the PRESET function using the I/O parameter "HOME/PRESET switching." (After the PRESET is selected, the HOME signal cannot be used and sensor-less mechanical home seeking operation cannot be performed.)

> **Note** When selecting the "ESMC Position Reading (PRESET Enable)" on the **SCX10** (as below), be sure to select the PRESET in the HOME/PRESET switching of the **ESMC** controller. If the PRESET is not selected, the **ESMC** controller will start sensor-less mechanical home seeking operation when operating the PRESET on the **SCX10**.

> **Memo** With the **ESMC** controller, if both the current position reading and the sensor-less mechanical home seeking operation are required, select HOME (enable sensor-less mechanical home seeking operation start signal) by "HOME/PRESET switching." (The driver current position reading function can be performed even with this setting.) In sensor-less mechanical home seeking operation, executing the PRESET input is not required because the reset (setting the home position) is automatically performed in the driver when finishing a mechanical home seeking operation.  The PRESET input is required when performing mechanical home seeking operation with external sensors etc. (other than sensor-less mechanical home seeking operation). (The PRESET signal is output from the **SCX10**.)

## ■ Setting the SCX10

To read the driver's current position, it is required to assign the necessary signals to the driver connector on the **SCX10**. Perform the following assignments using the automatic setting under [System Config] tab of the supplied utility software (**IMC**) as needed. (See "6.5.3 Change of Signal Assignment" on page 35.)

- **NX** Series driver: Select "NX Function Expansion"
- **ESMC** controller: Select "ESMC Sensorless Home Seeking & Position Reading" or "ESMC Position Reading (PRESET Enable)."

- When operating by a keyboard and not using the **IMC**, see the followings.
  It is required to assign each signal of the REQ (position data transmission request) output, CK (position data transmission clock) output, PR (position data output ready) input, P0 (position data bit 0) input and P1 (position data bit 1) input to the driver connector on the **SCX10** respectively. Also, it is required to assign the PRESET (reset home position) output signal when using the preset function.

Commands, which are executed via the keyboard, are as follows.
The non-zero parameters indicate the port number of I/O. For example, when assigning the PRESET signal is required, type DOUTPRESET=7. To unassign, type DOUTPRESET=0.

| Command | Parameter Value (Factory Setting) | Description |
|---|---|---|
| DINPR | 0 - 5 (0) | Assign the PR (position data output ready) input to the driver connector on the **SCX10** DINx<br>(Connect to the OUTR output of the driver) |
| DINP0 | 0 - 5 (0) | Assign the P0 (position data bit 0) input to the driver connector on the **SCX10** DINx<br>(Connect to the OUT0 output of the driver) |
| DINP1 | 0 - 5 (0) | Assign the P1 (position data bit 1) input to the driver connector on the **SCX10** DINx<br>(Connect to the OUT1 output of the driver) |
| DOUTCK | 0 - 8 (0) | Assign the CK (position data transmission clock) output to the driver connector on the **SCX10** DOUTx<br>(Connect to the CK input of the driver) |
| DOUTREQ | 0 - 8 (0) | Assign the REQ (position data transmission request) output to the driver connector on the **SCX10** DOUTx<br>(Connect to the REQ input of the driver) |
| DOUTPRESET | 0 - 8 (0) | Assign the PRESET (reset home position) output to the driver connector on the **SCX10** DOUTx<br>(Connect to the PRESET input of the driver) |

∗ Parameter 0 indicates "unassigned" (assignment cancel).

## ■ Connection

Connect the I/O points that have been configured in the above and the points on the **NX** Series driver or **ESMC** controller. (When the driver I/Os are assigned by using the automatic setting of the **IMC**, refer to the connection diagram for **NX** driver on page 60 to page 61 or for **ESMC** controller on page 56 to page 59.)

## ■ Reading the Driver Current Position

**Memo** When turning the power ON for the first time after connecting a battery to the driver, start operating with the next section "Releasing the Absolute Position Loss Alarm/Setting the Home Position."

The driver's current position is read using the following commands. (The operation can be done with the attached utility software, **IMC**. Click the [Absolute System] button under the [Motion Creator] tab.)

| Command/Parameter | Description |
|---|---|
| ABSREQ<br>(Reading driver current position) | Read the driver current position data, driver status code and driver alarm code, and then indicate<br>(The current position data is converted to the user unit and written to PABS, while the driver status code and driver alarm code are written to ABSSTS) |
| ABSREQPC<br>(Reading driver current position /updating internal position) | In addition to the ABSREQ function,<br>・PC (position command) of the **SCX10** is overwritten by the driver current position that was read, PF (feedback position) and EC (encoder count) of the **SCX10** will be updated maintaining the deviation<br>・Set the electrical home position and enable LIMP and LIMN when the software position limit control is set to 1 (SLACT=1) |
| PABS<br>(Driver current position) | Driver current position (User unit) |
| ABSSTS<br>(Driver status code/driver alarm code) | Driver status code (2 digits), driver alarm code (2 digits)<br>example) 1C48<br>　　　1C: driver status code (Hexadecimal)<br>　　　48: driver alarm code (Hexadecimal) |

<Steps>

1. Execute "CURRENT=0" (servo OFF/current OFF)

    CURRENT=0 causes to release any load on the shaft by the motor generating torque to be zero, and the deviation in the **SCX10** is always forced to be zero.

    ∗ The driver current position can be read in servo ON/current ON state. However, when a load is applied, the position read may not be equal to the actual machine position due to the machine deflection or the deviation in the driver. When any position displacement occurs by servo OFF/current OFF such as on the vertical axis, hold the load in position using an electromagnetic brake etc.

2. Execute ABSREQPC

    Read the driver current position and update the position information (PC, PF, EC) in the **SCX10**.

    If only reading the driver current position is required and updating the position information in the **SCX10** is not required, execute ABSREQ.

3. Execute "CURRENT=1" (servo ON/current ON)

    To prevent the position displacement from reading the current position, execute servo ON/current ON immediately after reading the driver current position.

    ∗ Once the ABSREQ or ABSREQPC commands have been executed, the driver current position will be written to PABS, and the driver statue code and driver alarm code will be written to ABSSTS. They can be referred to at anytime.

<Example>

| Command | Description |
|---|---|
| >PC | Confirm the PC value |
| PC=3.05 Rev | 3.05 Rev |
| >PF | Confirm the PF value |
| PF=3.04 Rev | 3.04 Rev |
| >CURRENT=0 | Servo OFF/current OFF  (release of a load, deviation zero) |
| >ABSREQPC | Read driver information, Overwrite PC and PF (EC) |
| PABS=124.35 Rev | Current position |
| Driver Status Code = 1C | Driver status code |
| Driver ALARM Code = 48 | Driver alarm code |
| >PC | Confirm the PC value |
| PC=124.35 Rev | 124.35 Rev (rewritten) |
| >PF | Confirm the PF value |
| PF=124.35 Rev | 124.35 Rev (rewritten) |
| >PABS | Position when executed the current position reading |
| PABS=124.35 Rev | 124.35 Rev |
| >ABSSTS | Driver Status Code/Driver Alarm Code |
| ABSSTS=1C48 | 1C48 (Indicates status code in 2 digits + alarm code in 2 digits) |

**Memo**
- The range of the driver's current position can be read is "-2,147,483.648 to +2,147,483.647," which is the value after converting to the user unit. The range to be written to PC and PF is limited by MAXPOS (maximum position value).
- When only referring to the current position (and not overwriting PC, PF EC), use the ABSREQ (reading driver current position) command. Also, ABSREQ can be used at any time while an error will occur when ABSREQPC is used during pulse generation.
- When executing ABSREQPC, the PC (position command) of the **SCX10** is overwritten by the driver current position that was read. The value for PF (feedback position) and EC (encoder count) of the **SCX10** will be updated maintaining the deviation. However, as PC will always follow PF with "deviation-zero" during servo OFF/current OFF, PC and PF will be updated with the same value.
- The driver's current position reading can be commanded via CANopen. If the data is required to be loaded to the master of CANopen, first command ABSREQPC or ABSREQ, and execute PABS and/or ABSSTS when the ABSDATA (driver current position data ready) output becomes "1." The ABSDATA output can be assigned to the remote output (in TPDO) of CANopen using the **IMC** software  or ROUTABSDATA command.

# ■ Releasing the Absolute Position Loss Alarm/Setting the Home Position

The absolute position loss alarm will generate when turning the power ON for the first time after connecting a battery to the driver. This is because the home position has not yet been set to the driver. When the battery voltage is decreased or the coordinate control range is exceeded, this alarm will generate. Referring to the following steps, release the absolute position loss alarm and set/reset the home position.

| Command/Parameter | Description |
|---|---|
| PRESET<br>(Reset home position) | ・Set PC (position command) to 0 (zero) and the set position will become the electrical home position.<br>　Since PF (feedback position) and EC (encoder count) will follow PC maintaining PE (deviation), they will become the deviation value as a result.<br>・When connecting a driver that has a PRESET input (the **NX** Series driver, **ESMC** controller etc.), the driver current position is reset to the home position. |
| ABSPLSEN<br>(Enable driver operation after absolute position loss alarm release) | Enable mechanical home seeking operation after the absolute position loss alarm of the driver is released. (This command will turn ON the P-REQ output assigned to the driver connector on the **SCX10** for about 1 msec.)<br>∗ This operation is required only when **NX** Series driver is used, while there is no need to do with the **ESMC** controller. |

<Steps>  (Start from the step 3. when turning the power ON first time)

1. Execute ABSREQ and read the cause of the alarm

2. Eliminate the cause of the alarm

3. Execute ALMCLR (alarm clear)

4. Execute ABSPLSEN (enable driver operation after absolute position loss alarm release) *for the **NX** Series driver only

5. Execute mechanical home seeking home operation
   (When mechanical home seeking operation is finished, the PRESET output is automatically output and the home position is set to the driver.)

**Memo**
- When setting the current position to the home position without mechanical home seeking operation, or when performing mechanical home seeking manually, skip the steps 4 and 5, and execute PRESET command.
- If the PRESET command is executed when the parameter for PRESET value (offset value from the home position) on the driver is set to a value other than zero, the current position of the **SCX10** (PC and PF) will not match the driver's current position. In this case, they will be matched by executing the ABSREQPC command. However, if setting the electrical home position other than the mechanical home position is desired, it is recommended to set the offset value in the **SCX10** (adjust "Home Offset" with the **IMC** software or use the OFFSET command) but not in the driver.
- When using the **NX** Series driver, set to "HOMEDCL=1." When finishing mechanical home seeking operation, the ACLDCL signal on the driver connector will be output and the driver deviation counter will be cleared.

# 8.7  Multi Axis Operation

User or the master controller can operate more than one **SCX10** at a time via serial communication.

## ■ RS-232C (Daisy Chain)

- Maximum Nodes: 36
- Address ID: command "ID n"
- Addressing method: Command "@n" or "TALKn" to select device. This makes a logical connection to a specific device, whose ID is "n" in a daisy chain connection.
- Reference section: "6.6 Connecting the RS-232C" on page 64, ID (page 253), @ (page 175), TALK (page 345).

The **SCX10** can also be daisy chained with the **CM10** Series and the *αSTEP*-**One**.  Commands are limited by each device, though they are very similar.

| Note | When a *αSTEP*-**One** is daisy chained, provided utility software, **Immediate Motion Creator for CM/SCX Series** cannot be used. |

- Daisy Chain Communication Example

Call the **SCX10** used for communication using the @ command.

### Example) Connection to a device whose axis number is 1 on the communication line.

When more than one device is connected to be communication line and the device power is first turned on, the terminal program or utility software will not be connected to any of the devices on the communication line. As a result, no prompt (">") is displayed on the screen. The @ command must be used to make a connection to one of the devices on the communication line.

| Command | Description |
|---------|-------------|
| @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. |
| 2>ID | #Query the ID of the connected device. |
| ID=2 | #The axis ID is returned (2). |

| Note | The communications echo control must be ON (ECHO=1) when daisy chain is performed. |

## ■ CANopen

- Maximum Nodes: 127
- Address ID: command "CANID n" via USB or RS-232C
- Addressing method: The master calls the device by following the CANopen specifications.
- Reference section: "6.7  Connecting the CANopen" on page 66.

## ■ USB

- Maximum Nodes: Up to the number of COM ports on the master controller (PC)
- Addressing method: One **SCX10** pairs with one COM port on the master controller (PC).  Command any target **SCX10** through the assigned port, each **SCX10** acts as a single node to one COM port.  No ID is necessary.  If desired, ID can be named by the command, "ID n".

| Memo | • If the computer does not have an extra USB port, a USB hub can be used.<br>• The **IMC** or general terminal software is operated in one-on-one relationship with the COM port. To connect multiple units of the **SCX10** to USB and drive each unit using PC, it is required to switch the setting of the **IMC** or terminal software to the COM port corresponding to each **SCX10**, or to launch the software being an equal number of the connected **SCX10**. |

| Note | Be aware that Windows automatically changes the COM port number when a **SCX10** is replaced. |

## 8.8 END (motion end) Signal

The END (motion end) signal is output when a motion was completed. This signal is used as a trigger to start the next motion or to perform other operations by confirming the completion of the present motion. Customers can use this signal as an output of the END signal or a sequence program, and also this signal is utilized as an internal function of the **SCX10**.

### ■ Definition of END Signal (Source)

There are two methods to generate the END signal/status of the **SCX10**: One is to use the driver END signal and another is to create the END signal internally in the **SCX10**. The source of the END signal/status can be selected from the following three types with a combination of the ENDACT (system end action) parameter and DEND (driver end signal enable) parameter. The factory setting is "End of pulse" (ENDACT=0, DEND=0).

| Definition of END Signal (Source) | ENDACT Parameter | DEND Parameter | Typical Motor Type |
|---|---|---|---|
| End of pulse | 0 | 0 | Stepping motor |
| End of pulse and within end area | 0<n (END area) | 0 | Stepping motor with an encoder |
| Driver END signal | Unrelated | 1 | $\alpha_{STEP}$/servo motor |

**Memo**
- The "End of pulse and within end area" is generally used to confirm that the positioning of the stepping motor has completed without having a loss of synchronism. With this selection, END signal/status is active when PE (position error) is within the ENDACT value. See "8.9 Encoder Function" on page 106. Connecting the encoder to the **SCX10** is required. See "6.8 Connecting the External Encoder" on page 67.
- "Driver END signal" is used to confirm that the driver has completed the operation of the $\alpha_{STEP}$ or a servo motor system that have the position deviation in the driver. See "6.5 Connecting the Driver" on page 33 and connect the driver END signal.
- When "Driver END signal" is chosen, there may be the case where END signal intermittently outputs during motor is moving at very slow speed. This is because the motor has been instantaneously entering the end area of the driver. Adjust the END related parameters in the driver or choose "End of pulse" or "End of pulse and within end area" (DEND=0) to avoid intermittent END signal.

Signal Flow Path
    **SCX10** internal END creation
        End of pulse generation----------- 0
                                                    **ENDACT**---------------- 0
        End of pulse generation
        and motor is within end area--- n (END area)
                                                    **DEND** ----→END status
        Driver END signal-------------------------------------------------- 1
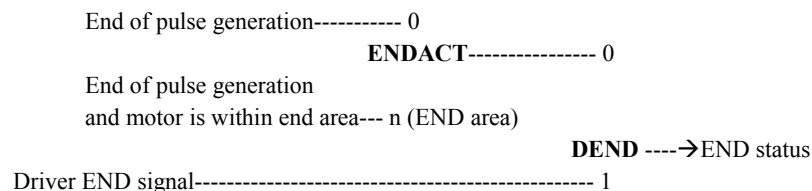
### ■ Output of END Signal, and How END Signal/Status is Internally Used in the SCX10

The END signal/status, which was created and selected in the definition of END signal, is used by the following three methods.

- END (motion end) Signal

The signal is output as the END (motion end) signal of the I/O connector. (See "6.4  Connecting the I/O Signals" on page 23.)  The operation completion can be sent to the host controller by the END signal. The END signal can be checked continuously via CANopen, and also it can be checked using the SIGEND command via serial communication.

- MEND (wait for motion end) Command

The MEND command is a command to be used in a sequence program. This command stops the sequence program until the motor operation completes. The sequence moves on the next one when the **SCX10** detects the END status internally. Therefore, the sequence program that starts the next motion or operation after confirming the present motion completion can be constructed by using the MEND command.

- Mechanical Home Seeking

When performing mechanical home seeking operation (MGHP, MGHN), the motor stops a few times during operation and the **SCX10** checks the END status.

**Memo**
- In the case of the MEND command or mechanical home seeking operation, if the pulse signal output is completed and the END signal is not output in a certain time, an alarm is generated. The alarm status is "Excessive Position Deviation (10h)" when the definition of END is set to the end area, and "Driver Connection Error (6Fh)" when the definition is set to the driver END signal.
- The waiting time until an alarm generates can be set using the ENDWAIT (END wait time) parameter (the factory setting is 6 seconds).
- The cause, that the END does not establish is the position deviation by overload in many cases. For example, the **AR** Series driver has the overload alarm function, and the time to decide whether or not to output the alarm is 5 seconds in the factory setting. The factory setting of ENDWAIT is 6 seconds in order to give the priority to the driver setting, and the driver alarm is generated if an overload is occurred (when the driver alarm is connected and it is set to enable).

# 8.9  Encoder Function

The **SCX10** has an encoder input that is used to check the actual position of the motor or device, or to perform an operation according to the actual position. The PF (feedback position) and PE (position error) are generated by the encoder input and those can directly be observed as figures, or used in a program. The PE can also be used to determine whether the position error is in the allowable range when the operation completes, in order to generate the END status using mainly a stepping motor with an encoder. Using these functions, it is possible to detect loss of synchronism for the stepping motor.

## ■ Connecting an Encoder and Selecting Inputs

There are two encoder inputs, on the driver connector of the **SCX10** and independent encoder connector. Choose one of them by ENC parameter, and connect it by referring "ASG/BSG Input" in "6.5 Connecting the Driver" on page 33 or "6.8 Connecting the External Encoder" on page 67. The factory setting is "Not used" (ENC=0).

| ENC Parameter | Description | Typical Motor Type | Reference Page |
|---|---|---|---|
| 0 | Not used | Stepping motor | - |
| 1 | On driver connector (ASG/BSG, TIMD/TIMS) | $\alpha_{STEP}$/servo motor | 6.5 Connecting the Driver |
| 2 | Independent encoder connector (EXTA, EXTB, EXTZ) | Stepping motor with an encoder/external Encoder | 6.8 Connecting the External Encoder |

**Memo**
- To use the encoder information correctly, it is required to adjust zero positions for the PC (position command) and PF. See "HOMEDCL (deviation counter clear select at mechanical home seeking operation) of 8.2.5 Mechanical Home Seeking" on page 83 and set properly for the conditions of use.
- The "Z" channel (zero position) of the selected encoder input by the ENC parameter is regard as TIMD/EXTZ (timing signal·Z-phase pulse differential input/external encoder ZSG) in the **SCX10**, and can be used to seek the mechanical home position. See "8.2.5 Mechanical Home Seeking" on page 83.

## ■ Parameters and Relations

The **SCX10** counts above selected signals by encoder count (EC), and convert it to user unit and continuously monitored as the feedback position (PF). The difference between PC (position command) and PF becomes the PE (position error). These relationships are shown by the following formula. Be sure to set the ER (encoder resolution) parameter first.

PE=PC-PF
PF=EC/ER∗DPR∗GB/GA

PE: position error (user unit)
PC: position command (user unit)
PF: feedback position (user unit)
EC: encoder count
ER: encoder resolution
DPR: distance per revolution
GA, GB: electrical gear ratio (GA/GB)

## ■ How to Use

- Position confirmation after positioning operation

  When the DEND (driver end signal enable) parameter is set to 0 (internal END signal), the position check using the encoder feedback information (PE) is performed by setting the positioning completion signal range using the ENDACT (system end action).  See "8.8 END (motion end) Signal" on page 105.

- Detection of loss of synchronism for stepping motor

  When using a stepping motor with an encoder, it is possible to check whether loss of synchronism is generated or not by monitoring the PE.
  In the case of the basic step angle 1.8° of 2-phase stepping motor and 0.72° of 5-phase stepping motor, if the offset of the small teeth on the stator and rotor (PE) at motor standstill condition exceeds ±1.8 degrees (±0.9 degrees for the basic step angle 0.9° of 2-phase stepping motor and 0.36° of 5-phase stepping motor), loss of synchronism may be generated because the force to synchronize the motor becomes very unstable condition (the acceptable value varies depending on the operating condition such as operating speed etc.).

  ・How to use the END (motion end) signal: See "8.8  END (motion end) Signal" on page 105.
  If the value exceeding just 1.8 degrees at the motor output shaft is set using the ENDACT command, loss of synchronism can be determined by the END signal or the SIGEND command. (In this case, set "DEND=0" to enable the internal END signal.) For example, if the DPR (distance per resolution) is set to 360, set "ENDACT=1.81."

  ・When programming with the PE (position error):
  Loss of synchronism can be detected using the position error if the END area (value of ENDACT) is required to set individually regardless of loss of synchronism. Programming example to show the result on the terminal is as follows.

| Command | Description |
|---|---|
| ( 1) MI | #Start incremental motion |
| ( 2) MEND | #Wait for motion to end |
| ( 3) IF (PE<-1.8) | #When the PE is smaller than -1.8° |
| ( 4)   SAS MISS-STEP MAY HAVE OCCURRED | #Transmit a message |
| ( 5) ENDIF | #End the IF statement |
| ( 6) IF (PE>1.8) | #When the PE is greater than -1.8° |
| ( 7)   SAS MISS-STEP MAY HAVE OCCURRED | #Transmit a message |
| ( 8) ENDIF | #End the IF statement |

  ∗ As another example, the sample program which urges maintenance of the equipment when a load is increased is introduced in the explanation of the PE command (page 306).

- Perform operation and judgement based on a operation of the other parts for the device.

  By installing an encoder to the different parts other than the motor such as the shaft of the belt conveyor or the float to measure liquid etc., it is possible to operate or stop the motor, to operate according to the EC (encoder count) value, or to determine OK or NG when the EC reaches the set value. Examples are shown below.

  | Command | Description |
  |---|---|
  | `Example to operate the motor when the EC` `becomes the set value.` | |
  | `(  1) WHILE (EC<100);WEND` | #When the EC becomes greater than 100 |
  | `(  2) MI` | #Start incremental motion |
  | | |
  | `Example to operate the motor by the` `motion distance according to the EC value` | |
  | `(  1) DIS=EC/100` | #Assign a value of hundredth part of the EC to the incremental motion distance |
  | `(  2) MI` | #Start incremental motion |

  **Memo**
  - The value of EC can be set such as to set "EC=0" as an initial value. The setting range of EC is "-MAXEC to +MAXEC."
  - The PF can be used instead of the EC, but the user unit becomes in common with the motor and it is affected by the DPR, GA and GB. Use the encoder resolution (ER) to adjust the magnification. And the PE will have no meaning in many cases.

# 8.10 Math/Logical/Conditional Operators

You can use math/logical/conditional operators in a sequence program. For command reference of math/logical/conditional Operators, see "■ Math/Logical/Conditional Operators (In Sequence only)" on page 167.

## ■ Math/Logical Operators

The following math/logical operators can be used in a program:
- + : Addition
- − : Subtraction
- * : Multiplication
- / : Division
- % : Modulo (remainder)
- & : AND (Boolean)
- | : OR (Boolean)
- ^ : XOR (Boolean)
- << : Left arithmetic shift (Shift to left bit)
- >> : Right arithmetic shift (Shift to right bit)

- Example

| Command | Description |
|---|---|
| >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 Command 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 |
| >PC | #Query the PC value |
| PC=10 Rev | #Device response |
| >RUN 1 | #Run sequence #1 |
| >20 | #Device response |
| > | |

## ■ Conditional Operators

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==b : a is equal to b
- a>=b : a is greater than or equal to b
- a>b : a is greater than b

- Example

| 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=1 | #Set the distance to 1 User Unit |
| ( 3) MI | #Move Incrementally |
| ( 4) ENDIF | #End the IF Statement |
| ( 5) END | #End the sequence |
| > | |

# 8.11 User Variables

As opposed to the variables (parameters) which roles are fixed such as PC and PF, there are general purpose numeric variables that can be used in various uses such as calculation etc. Variables that can be named freely and variables that can treat a character string are also provided. For command reference of user variables, see "■ User Variables" on page 168.

## ■ User Variables

- A to Z

General purpose numeric variables.
Upper and lower case are permitted, but 'A' and 'a' reference the same variable.
In immediate mode, A to Z may only be set and queried.
Within a sequence, variables may also be used in the following conditions:
• Targets or arguments for assignments (e.g. A=TIMER; DIS=A)
• Loop Counters (e.g. LOOP Q)
• Conditional Statement Values (e.g. if (VR>X))
• Arguments for a subroutine CALL (e.g. CALL S)
• Parts of Mathematical Expressions
• Targets for interactive data entry commands (e.g. X=KBQ)

- Example

| Command | Description |
|---|---|
| (  1) MCP | #Start continuous motion |
| (  2) WHILE (IN1=0) | #Repeat while IN1=1 |
| (  3)   D=PE-E | #PE: Position error, E: user variable |
| (  4)   D=0.01*D | #D: user variable |
| (  5)   E=E+D | |
| (  6) WEND | #End of WHILE block |
| (  7) SSTOP | #Soft Stop |
| (  8) MEND | #Wait for motion end |
| (  9) IF (E>3) | #If E>3 |
| ( 10)   SAS Load increasing, clean machine. | #Transmit comment |
| ( 11) ENDIF | #End of IF block |

## ■ User-defined Variables

- N_xxx :

General purpose, user-defined numeric variables.
A user-defined variable must be created with CREATEVAR before it can be used. After it has been created, it can be used in the same way as the general purpose variables A to Z, except that it cannot be used as the argument for a CALL statement.

- S_xxx :

General purpose, user-defined string variables.
A user-defined string variable must be created with CREATEVAR before it can be used. After it has been created, it can be used to store or display text.

- Associated commands

CLEARVAR: Clears all user-defined variables from memory.
DELETEVAR: Deletes a specific user-defined variable.
LISTVAR: Lists the names and values of all user-defined variables.

- Example

| Command | Description |
|---|---|
| >CREATEVAR N_COUNTS=0<br> New variable N_COUNTS is added.<br> N_COUNTS=0 | #Create user-defined numeric variable named N_COUNTS |
| >CREATEVAR N_TOTAL=10<br> New variable N_TOTAL is added.<br> N_TOTAL=10 | #Create user-defined numeric variable named N_TOTAL |
| >LIST MAIN | #List sequence MAIN |

```
(  1)  WHILE (N_COUNTS < N_TOTAL)      #N_COUNTS, N_TOTAL user-defined variables
(  2)    MI; MEND                      #Start incremental motion; wait until complete
(  3)    OUT4 = 1                      #Set output 4 on
(  4)    WHILE (IN6=0); WEND           #Wait for input 6 to go off
(  5)    OUT4 = 0                      #Set output 4 off
(  6)    WHILE (IN6=1); WEND           #Wait for input 6 to go on
(  7)    N_COUNTS=N_COUNTS+1           #Increment N_COUNTS by 1
(  8) WEND                             #End of WHILE block
>
```

# 8.12 View and Test Functions

The **SCX10** has various support functions, such as functions to display ON-OFF status and test the connection of I/O, to display the system status (parameters etc.), and to display online HELP letting you know the command names and brief descriptions without reading this manual.

## ■ I/O Test

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 screen.
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) | END (E) | HOMEP (H) | ALARM (A) |
| PSTS (P) | MBFREE (B) | READY (D) | LC (L) | |

A SPACE key sets all outputs to inactive (0).
An <ESC> key or character exits the OUTTEST process.

**Memo**
- 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 ***
Inputs  (1-9) = RUN ABORT IN3 IN4 IN5 –LS +LS HOME PSTOP
Outputs (1-4) = OUT1(1) OUT2(2) END(E) ALARM(A)


  - Use (x) Keys to toggle Outputs.
  - Use <space> to set all outputs to zero.
  - Use <esc> to exit OUTTEST mode.


        I/O Status Monitor
--Inputs--              Outputs
1 2 3 4 5 6 7 8 9 –(SEQ#)– 1 2 3 4
0 0 0 0 0 0 0 0 0 –(  0 )– 0 0 1 0
```

## ■ Signal Status

All I/O status can be reported by SIG__commands.  For example, the SIGHOME replies "1" when an external HOME input signal is ON (active), and replies "0" when it is OFF (not active).

### • Example

| Command | Description |
|---|---|
| >LIST SLIPCHECK | #List sequence SLIPCHECK |
| | |
| (  1)  EHOME | #Return to home position (PC=0) |
| (  2)  MEND | #Wait for motion to complete |
| (  3)  IF (SIGHOME!=1) | #If HOME 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 |
| > | |

While SIG__ commands are generally used in a sequence programs, they are also convenient for monitoring purposes.
See "Continuous Display" below.

Applicable Commands:
SIGPSTOP, SIGMSTOP, SIGABORT, SIGSTART, SIGMCP, SIGMCN, SIGMGHP, SIGMGHN, SIGPAUSE, SIGCONT, SIGPAUSECL, SIGPECLR, SIGALMCLR, SIGSENSOR, SIGHOME, SIGHOMEP, SIGLSP, SIGLSN, SIGCON, SIGFREE, SIGMOVE, SIGRUN, SIGEND, SIGALARM, SIGPSTS, SIGMBFREE, SIGREADY, SIGTL
DSIGACLDCL, DSIGALARM, DSIGEND, DSIGTIMS, DSIGTIMDEXTZ, DSIGCON, DSIGCOFF, DSIGCS, DSIGMBFREE, DSIGFREE, DSIGLC, DSIGMOVE, DSIGM0, DSIGM1, DSIGM2, DSIGTL

> **Note** The display by the SIG_commands represents ON/OFF (active or not active) of signals but not ON/OFF of photocouplers. Take note because it depends on the setting of I/O logic level (LV command). If "1" is received when inquiring "SIGHOME" while setting to "HOMELV=0", "0" will be received when inquiring "SIGHOME" while setting "HOMELV=1."

## ■ Continuous Display

A forward slash character (/) following certain variables causes the system to continuously display the value of those elements utilizing these rules:
· Only one command may be displayed simultaneously
· A space must separate the command from the / character
· This data is updated every 0.15 seconds
· Keyboard <ESC> terminate the display loop

### • Example

| Command | Description |
|---|---|
| >UU=Degrees | #Set the User Units to Degrees |
|  UU=Degrees | |
| >VR=10 | #Set the running velocity to 10 Degrees/second |
|  VR=10 Degrees/sec | |
| >MCP | #Begin moving continuously |
| >PC / | #Continuously display the position command |
| | |
|      72.639 Degrees | #Device response at one moment in time |
| | #<ESC> sent: display terminates |
| > | |

Applicable Displayed Commands:
IN, INSG, INx, OUT, OUTSG, OUTx, IO, RIO, DIO, PC, PCI, PE, PF, PFI, EC,
SIGPSTOP, SIGMSTOP, SIGABORT, SIGSTART, SIGMCP, SIGMCN, SIGMGHP, SIGMGHN, SIGPAUSE, SIGCONT, SIGPAUSECL, SIGPECLR, SIGALMCLR, SIGSENSOR, SIGHOME, SIGHOMEP, SIGLSP, SIGLSN, SIGCON, SIGFREE, SIGMOVE, SIGRUN, SIGEND, SIGALARM, SIGPSTS, SIGMBFREE, SIGREADY, SIGTL
DIN, DINx, DINSG, DOUT, DOUTx, DOUTSG, DSIGACLDCL, DSIGALARM, DSIGEND, DSIGTIMS, DSIGTIMDEXTZ, DSIGCON, DSIGCOFF, DSIGCS, DSIGMBFREE, DSIGFREE, DSIGLC, DSIGMOVE, DSIGM0, DSIGM1, DSIGM2, DSIGTL

The ESC key will cause the termination of the / (forward slash) command execution. While the forward slash command is executing and motion is occurring, the ESC key will first cause the termination of the forward slash command execution. The ESC key must be transmitted to the device a second time to cause the motion to cease.

> **Note**
> • Do not confuse this special command with the division operator, "/."
> • When this command is used in the RS-232C with daisy chain connection, line feeds will occur and the bottom line will indicate the current value.

## ■ System Status

The system status summary including all I/O assignments, active level and status, values of important parameters, current position, and alarm condition are displayed by the REPORT command. The next page will be displayed by pressing the SPACE key. The REPORT command can be an effective tool for troubleshooting problems with the system.

```
/ I/O REPORT /---(NO:Normally Open, NC:Normally Closed)-----------
   IN1(NO) = 0    IN2(NO) = 0    IN3(NO) = 0    IN4(NO) = 0
   IN5(NO) = 0    IN6(NO) = 0    IN7(NO) = 0    IN8(NO) = 0
   IN9(NO) = 0
   OUT1(NO) = 0   OUT2(NO) = 0   OUT3(NO) = 0   OUT4(NO) = 0


/ REMOTE I/O REPORT /-----------------------------------------------
   IN1 = 0    IN2 = 0    IN3 = 0    IN4 = 0
   IN5 = 0    IN6 = 0    IN7 = 0    IN8 = 0
   ABORT = 0    START = 0    MCP = 0    MCN = 0
   MGHN = 0    CON = 0    FREE = 0
   OUT1 = 0    OUT2 = 0    OUT3 = 0    OUT4 = 0
   OUT5 = 0    OUT6 = 0
   END = 0    MOVE = 0    HOMEP = 0    LC = 0    READY = 0


/ DRIVER I/O REPORT /-----------------------------------------------
   ALM(NC) = 1    IN2(NO) = 0    END(NO) = 0    READY(NO) = 0
   LC(NO) = 0    TIMS(NO) = 0    TIMD/EXTZ(TIMDLV=1, EXTZLV=0) = 0
   CON(NO) = 0    ACL/DCL(NO) = 0    REQ(NO) = 0    TL(NO) = 0
   M0(NO) = 0    M1(NO) = 0    PRESET(NO) = 0    FREE(NO) = 0


   Enter [SPACE] to continue, other key to quit.


/PARAMETER REPORT /-----------------------------------------------
   UU = Rev
   STRSW = 0    DPR = 1    MR = 1000    GA = 1    GB = 1
   ER = 1000    DIRINV = 0
   VS = 0.1    VR = 1    TA = 0.5    TD = 0.5    DIS = 0
   LIMP = 0    LIMN = 0    SLACT = 0
   STARTACT = 0    MSTOPACT = 0    SENSORACT = 2    OTACT = 0
   ALMACT = 2    ALMMSG = 0    HOMETYP = 0    HOMEDCL = 1
   INCABS0 = 1    VR0 = 1    DIS0 = 0    LINK0 = 0
   INCABS1 = 1    VR1 = 1    DIS1 = 0    LINK1 = 0
   INCABS2 = 1    VR2 = 1    DIS2 = 0    LINK2 = 0
   INCABS3 = 1    VR3 = 1    DIS3 = 0
   DALARM = 1    DREADY = 1    STRDSC = 0    TIM = 1    ENC = 1
   DEND = 1    ENDACT = 0    MBFREEACT = 0
   PULSE = 1    PLSINV = 0    CANID = 1    CANBAUD = 1


/ POSITION REPORT /-----------------------------------------------
   PC = 0    PF = 0    PE = 0    EC = 0    PABS = 0


/ ALARM HISTORY /-----------------------------------------------
   ALARM = 6E ,    RECORD : 6E 6E 00 00 00 00 00 00 00 00
   ALM_DRIVER_ALARM , 15.123 [sec] past.
   Driver Status Code = 00    Driver ALARM Code = 00
```

## ■ Commands

Type HELP to display the command syntax and brief description. The SPACE key on the keyboard lists the next HELP screen. Any other keyboard key will exit the HELP screen mode.

```
--- Command List ---

    TALK*       : Select unit in multi-unit communications
    @*          : Select unit in multi-unit communications
    MI          : Move Incrementally
    MA          : Move Absolutely (-MAXPOS - +MAXPOS[UU])
    CV          : Change Velocity for Index (0.001 - MAXVEL[UU/sec])
    MCP         : Move Continuous Positive
    MCN         : Move Continuous Negative
    DIS         : Incremental motion distance (-MAXPOS - +MAXPOS[UU])
    VR          : Running velocity (0.001 - MAXVEL[UU/sec])
    VS          : Starting velocity (0 - MAXVEL[UU/sec])
    TA          : Acceleration time (0.001-500.000[sec])
    TD          : Deceleration time (0.001-500.000[sec])
    PSTOP       : Stop immediately, stop sequence, follow ALMACT setting
    HSTOP       : Stop immediately (hard stop)
    MSTOP       : Stop according to MSTOPACT
    SSTOP       : Stop, decelerating (soft stop)
    SCHGPOS     : Distance from SENSOR on MCx (0 - +MAXPOS[UU])
    SCHGVR      : Velocity on SCHGPOS motion (0.001 - MAXVEL[UU/sec])

    Enter [SPACE] to continue, other key to quit.
```

# 8.13 Protective Functions

For some alarm conditions, the action(s) taken when the condition is detected can be controlled by ALMACT, to suit the application

- Alarm Conditions Effected by ALMACT

| Condition | Description | Alarm Code |
|---|---|---|
| Hardware over travel | Positive or negative position limit signal detected | 66h |
| Software over travel | Position outside of programmed positive and negative software position limits LIMP and LIMN | 67h |
| Panic stop | System executed a panic stop because of a PSTOP input or command | 68h |

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

| ALMACT | Action |
|---|---|
| 0 | Motor current ON, alarm OFF. |
| 1 | Motor current ON, alarm ON. |
| 2 | Motor current OFF, alarm ON. (factory setting) |

**Memo** See "9.7 Error Messages Displayed on the Terminal" on page 127 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 (factory setting) |
| 1 | Automatically transmit messages for alarms, but not warnings |
| 2 | Automatically transmit messages for alarms and warnings |

**Memo**
- See "9.7 Error Messages Displayed on the Terminal" on page 127 for message details.
- Warnings are for informational purposes only, and do not effect system operation.

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

- Example

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

 ALM_OVER_LOAD , 3.234 [sec] past.

 WARNING =00 , RECORD: 00 00 00 00 00 00 00 00 00 00

 No warning.
>
```

**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.

# 9 Program Creation and Execution

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

Memo : Although these functions are introduced as keyboard operation using any general terminal software in this chapter, most of the operations can be done by using the [Program Editor] tab screen on the supplied **Immediate Motion Creator for CM/SCX Series** utility software.

## 9.1 Overview

### ■ What is Program Execution?

You create sequences using a computer, save the programs into the built-in memory of the **SCX10**, specify which sequence number to run, and input the start signal to execute the sequence. The program creation is made via USB or RS-232C, the program selection and execution are made via USB, RS-232C, CANopen or I/O selection.
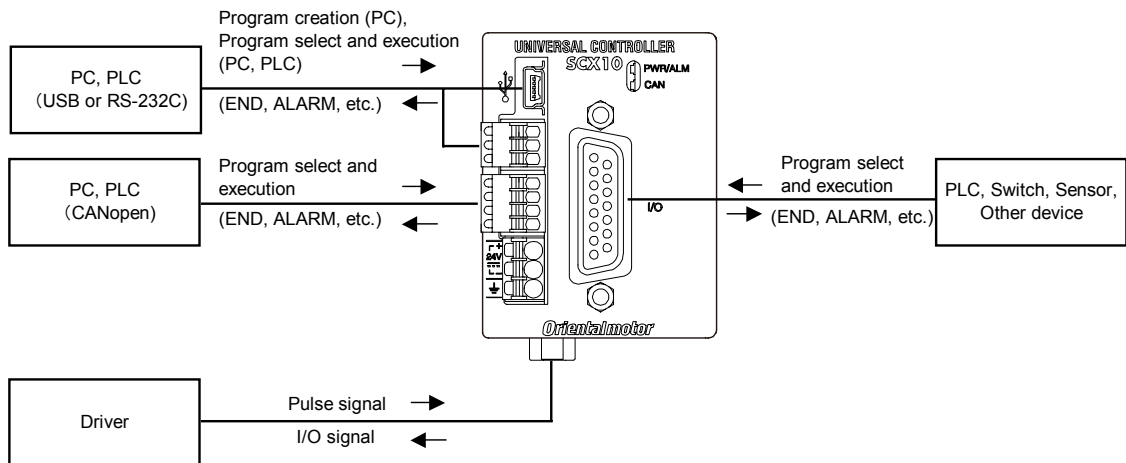


| | Immediate Command (Monitor Mode) | Program Creation (Program Edit Mode) | Program Select and Execution (Sequence Mode) |
|---|:---:|:---:|:---:|
| USB | ✓ | ✓ | ✓ |
| RS-232C | ✓ | ✓ | ✓ |
| CANopen | ✓ | - | ✓ |
| I/O | - | - | ✓ |

### ■ Contents

9.2 Operating Modes
9.3 Preparation
9.4 Creating a New Sequence
9.5 Sample Programs
9.6 Executing a Sequence
9.7 Error Messages Displayed on the Terminal

## 9.2  Operating Modes

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 (Program Edit Mode)

Sequences can be edited by either,
- Editing from the terminal.
- Editing from supplied utility software, the **Immediate Motion Creator for CM/SCX Series (IMC)**.
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 (Sequence Mode)

Sequences can be executed by either,
- Using the "RUN" command from the terminal.
- From the I/O or the CANopen using the "START" and "INx" inputs.

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.
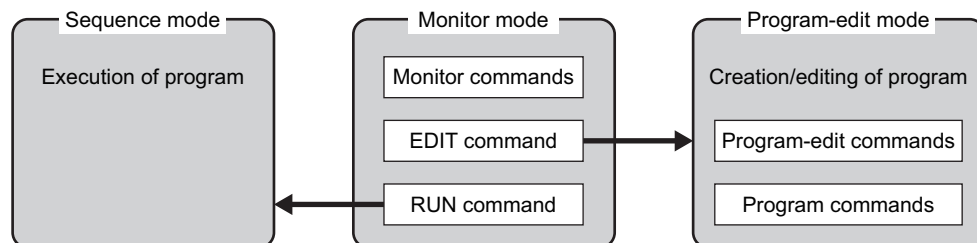
## 9.3  Preparation

1. Connect a personal computer via USB or RS-232C
   See "6.3 Connecting the USB and Installation of Utility Software" on page 20 or "6.6 Connecting the RS-232C" on page 64 as necessary.

2. Power ON the **SCX10** and the driver
   See "6.2 Connecting the Power Supply" on page 19 as necessary.

3. Launch any general terminal software or the supplied utility software (**IMC**)
   See "6.3 Connecting the USB and Installation of Utility Software" on page 20 as necessary.

   Communication Settings for general terminal software
   8 bits, 1 stop bit, no parity
   Baud rate: 9600 bps (first connection)
   ∗ The default USB and RS-232C baud rate of the **SCX10** is 9600 bps. After connecting initially, you can change the baud rate as necessary.

## 9.4  Creating a New Sequence

Programs contain data with which to define device operation, such as the running 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 | 100 sequences (Name is use configurable) |
| Maximum sequence size | Total sequences: 6 kB (compiled)  21 kB (text + compiled) <br> 1 sequence: 6 kB (text)  2 kB (compiled) |
| Sequence execution by external input | START input executes a sequence selected by binary combination of IN1 to IN7. |
| Automatic sequence execution at power up. | Sequence named CONFIG is executed at power up. |
| Sequence program name | 10 characters maximum. <br> 0 to 9, A to Z, a to z, _(underscore) can be used as characters. Upper and lower case are permitted, not case sensitive. <br> **Name cannot begin with number, or "N_," "S_," "n_," "s_."** <br> Using command name and/or parameter names for sequence names can cause confusion, and is not recommended. <br> If sequence is saved by name, system assigns sequence number within 0 to 99. Assigned number is used for selection to start sequence by I/O. |

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

### ■ Sequence Commands

The majority of commands on the **SCX10** can be used in a sequence.  See "12 Command Reference" on page 160.
For sequence specific commands, refer to "■ Sequence Commands" on page 167 and
"■ Math/Logical/Conditional Operators (In Sequence only)" on page 167.

## ■ Example of Creating a New Sequence

**1.** Enter the monitor command "EDIT ∗," and press the Enter key.
(∗ indicates the sequence name or number (optional). Insert a space between "EDIT" and them.)
The system enters the sequence-edit mode, and a message indicating a blank sequence (New sequence) is displayed.
Enter "I" (Insert).
Subsequently, the line number "(1)" is displayed and
you can now create a sequence.

```
>EDIT SAMPLE1

New Sequence

Sequence Name  : SAMPLE1
Sequence Number: 0
Lines:         : 0
Bytes:         : 0
Bytes Free     : 6144

>>Command: I

( 1)_
```

**2.** Enter commands and parameters by referring to "12 Command Reference," to create a program.
The following shows a sample program. This program, SAMPLE1, executes an absolute positioning operation at a starting velocity of 1 rev/sec and running velocity of 3 rev/sec, with the target position of 5 revolution from PC=0. (When setting the user unit (UU) to "rev" and the distance per revolution (DPR) to 1)
Insert a space or equal sign between the command and the parameter. See "7.6 Command Format" on page 75 as a reference. ("=" is not used because the MA is not a parameter)

```
( 1) VS=1
( 2) VR=3
( 3) MA 5
( 4) _
```

**Memo** The semicolon (;) allows for multiple command statements to be used on a single command line. See "7.6 Command Format" on page 75 for more details.

**3.** When the program entry is complete, press the Enter key, enter "S" and press the Enter key to save the program.
Finally, enter "Q" to complete the program and exit edit mode.

```
( 4)

>>Command: S

Compiling...OK
Saving........OK

>>Command: Q
>_
```

**Memo** If "S" is input, syntax check will be executed before saving the program. If a syntax error is found, the error type and the line number that was found an error will be displayed.

## ■ **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 ∗," and press the Enter key.
(∗ indicates the sequence name or number (optional). Insert a space between "EDIT" and them.)
The system enters the sequence-edit mode.

```
>EDIT PROGRAM1

Sequence Name  : PROGRAM1
Sequence Number: 1
Lines:         : 5
Bytes:         : 23
Bytes Free     : 6121


>>Command:_
```

**2.** Enter an editor command and a line number according to the edit operation you wish to perform.
Insert a space between the editor command and the line number. Two or more lines can also be specified depending on the editor command.

A    3

↑ ↑ ↑
— Line number to be edited
— Space
— Program-edit command
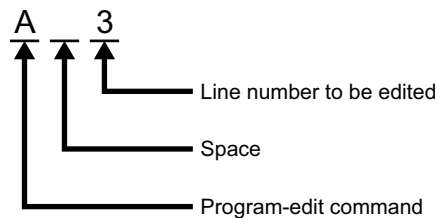
| Command | Description |
|---------|-------------|
| I | Insert |
| A | Alter(Change) |
| D | Delete |
| L | List |
| X | Cut |
| C | Copy |
| P | Paste |
| S | Save, Compile |
| M | Display memory status |
| H | Display help |
| Q | Quit |

**Memo** : The "H" command will show the command help (above list) while editing a sequence.

```
>>Command: H
    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
    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 (0-99)
    S sss     | Save   sequence, by name (10 char max)
    M         | Display memory status
    H         | Display this help reminder
    Q         | Quit sequence editor
```

## ■ **Example of Line Editing**

This section explains the steps to edit PROGRAM1 as follows:

• Before editing

```
PROGRAM1
(1) VR 5
(2) PC 12
(3) MI
(4) MGHP
(5) END
```

• After editing

```
PROGRAM1
(1) VR 5
(2) PC 12
(3) MA 5          ◄—— Line 3 is changed from MI to MA 5.
(4) WAIT 10       ◄—— A WAIT command is added between
                        lines 3 and 5.

                  ◄—— MGHP is deleted.
(5) END
```
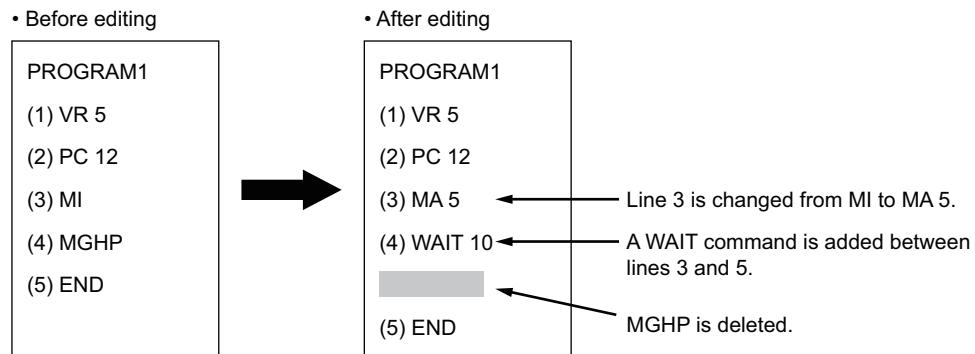
1. Enter "EDIT PROGRAM1" and press the Enter key.
   After the contents of PROGRAM1 are displayed, ">>Command:" is displayed and the monitor waits for editing input.

```
>EDIT PROGRAM1

Sequence Name  : PROGRAM1
Sequence Number: 1
Lines:         : 5
Bytes:         : 23
Bytes Free     : 6121

>>Command:_
```

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

```
>>Command: L

( 1) VR 5
( 2) PC 12
( 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.

```
>>Command:_ A 3
( 3) MI_
```

   b. Delete "MI" with the Backspace key.

```
( 3) _
```

   c. Enter "MA  5."

```
( 3) MA 5_
```

   d. Press the Enter key.
      Line 3 of PROGRAM1 is changed to "MA  5." The command prompt is displayed and the monitor waits for the next editor command.

```
( 3) MA 5_
>>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.

```
>>Command: I 4
( 4)_
```

b. Enter "WAIT  10."

```
( 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.

```
( 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 editor command.

```
( 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 editor command.

```
>>Command: D 5
>>Command:_
```

## ■ Ending the Edit Session

**1.** Enter the command "S" to end the session after saving the edited contents, and then press the Enter key.

The edited contents are saved.

```
>>Command: D 5
>>Command: S

Compiling...OK
Saving........OK

>>Command:_
```

**2.** Enter "Q" to quit the sequence editor.

a ">" (command prompt) is displayed.

```
>>Command: Q
>_
```

# 9.5  Sample Programs

This section provides sample programs.

## ■ Repeated Positioning Operation

- Motion Pattern
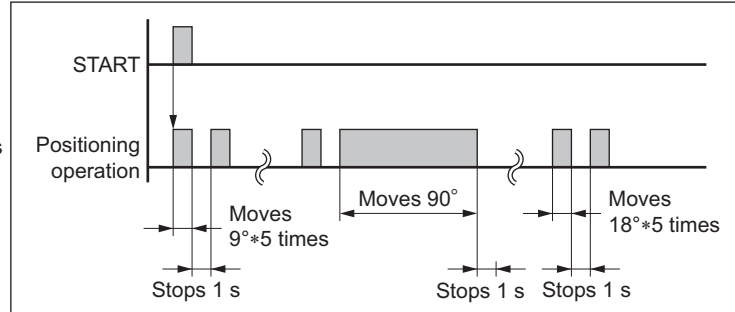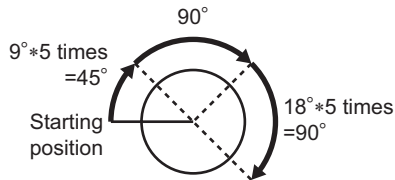


- Main Program

Applicable device: Rotary table

Resolution: 360 deg/rev (DPR=360)
UU: degrees

```
( 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 deg/sec.
The running velocity is set to 360 deg/sec.
Lines 6 through 9 are repeated five times.
The distance is set to 9 degrees.
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 degrees.
Incremental positioning operating 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 degrees.
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.

## ■ **Executing Linked Operation**

- Motion Pattern

Resolution: 10 mm/rev (DPR=10)
UU=mm

| LINKx | Setting Value |
|-------|---------------|
| LINK0 | 1 (linked) |
| LINK1 | 1 (linked) |
| LINK2 | 0 (one-shot) |

Distance: 10 mm
Operating speed: 10 mm/s

Distance: 20 mm
Operating speed: 20 mm/s

Distance: 30 mm
Operating speed: 30 mm/s

Velocity

No.0    No.1    No.2

Time
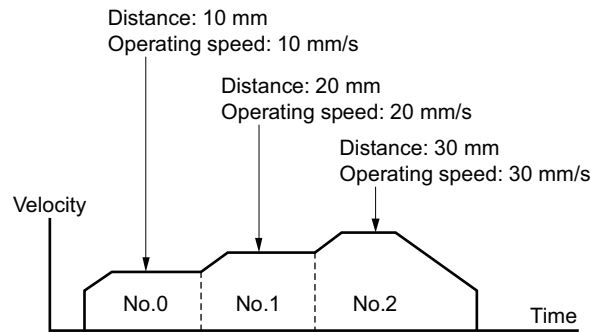
- Main Program

```
(  1)  DIS0=10      The distance for segment number 0 is set to 10 mm.
(  2)  DIS1=20      The distance for segment number 1 is set to 20 mm.
(  3)  DIS2=30      The distance for segment number 2 is set to 30 mm.
(  4)  VR0=10       The operating speed for segment number 0 is set to 10 mm/sec.
(  5)  VR1=20       The operating speed for segment number 1 is set to 20 mm/sec.
(  6)  VR2=30       The operating speed for segment number 2 is set to 30 mm/sec.
(  7)  INCABS0=1    The positioning mode for segment number 0 is set to incremental.
(  8)  INCABS1=1    The positioning mode for segment number 1 is set to incremental.
(  9)  INCABS2=1    The positioning mode for segment number 2 is set to incremental.
( 10)  LINK0=1      Segment number 0 is set to linked.
( 11)  LINK1=1      Segment number 1 is set to linked.
( 12)  LINK2=0      Segment number 2 is set to termination.
( 13)  MI0          Start the operation to start at segment number 0. (Numbers 0 through 2 are linked.)
( 14)  END          The sequence program is ended.
```

## 9.6  Executing a Sequence

You can execute sequences stored in the device's memory. There are two ways to execute a sequence.
To perform this via CANopen, refer to "■ Executing a sequence" on page 141.

### ■ Executing a Sequence from the Terminal

1. Connect the **SCX10** to the terminal by USB or RS-232C. Start the **Immediate Motion Creator for CM/SCX Series** (included) or terminal software.

2. Enter the monitor command "RUN ∗," and press the Enter key.
   (∗ indicates the sequence name or number. Insert a space between "RUN" and them.)
   When the command is entered, the system executes the sequence.

### ■ Executing a Sequence from I/O

1. Assign START and ABORT, as needed input signal to IN1 to IN9 of the system I/O connector.
   Set it using the [System Config] - [I/O setting] tab screens of the provided utility software, **IMC**.

   Memo
   - Assign it to the high number input so that the START input and the inputs used for program selection (see below) do not overlap each other.
   - If terminal operation is preferred rather than using the **IMC**, use INSTART command. (ex. INSTART=5)

2. Connect the START (ABORT) input to the host controller.

3. Assert IN1 to IN7 inputs to select the sequence to execute.
   Sequence is selected by the binary value of IN1 to IN7. (see the chart)  Inputs assigned to other functions (MSTOP, HOME, etc) are read always as OFF. (e.g. INPAUSE=3 means IN3 is always read as OFF.)

Example of Selection                                                    (Empty section means OFF)

| Sequence Number | Input Port | | | | | | |
|---|---|---|---|---|---|---|---|
| | IN1 (1) | IN2 (2) | IN3 (4) | IN4 (8) | IN5 (16) | IN6 (32) | IN7 (64) |
| 0 | | | | | | | |
| 1 | ON | | | | | | |
| 2 | | ON | | | | | |
| 4 | | | ON | | | | |
| 8 | | | | ON | | | |
| 16 | | | | | ON | | |
| 32 | | | | | | ON | |
| 64 | | | | | | | ON |
| 3 | ON | ON | | | | | |
| 63 | ON | ON | ON | ON | ON | ON | |
| 99 | ON | ON | | | | ON | ON |

Memo
- The rotary digital switch may conveniently be used when sequence selection is performed by switches.
- This chart does not mean all seven inputs are always necessary to select sequence number. If the number of sequences to select are limited, such as eight, only IN 1 to IN 4 are necessary, and IN5 to IN9 can be assigned to any system input such as the SENSOR, ABORT and CON.
- The sequence tracing command, "TRACE" is available to check sequence action. When TRACE=1, while the sequence is processing, sequence statements can be displayed as they are executed, one statement at a time.  See TRACE (page 351) in more detail.

## ■ Stop the sequence execution

When stopping the sequence execution is required, use the ABORT input or signal. The START input also has the ABORT function when configured. Set "START Action" to "Level", under [System Config]-[System Parameters] of the **IMC**.

| STARTACT | Operation |
|---|---|
| 0 (edge) | Setting START input from OFF to ON starts sequence execution.<br><br>    When sequence is running, paused motion is resumed.<br>Setting START input from ON to OFF does not stop sequence.<br>ABORT input is required for aborting sequence. |
| 1 (level) | Setting START input from OFF to ON starts sequence execution.<br>Setting START input from ON to OFF aborts the sequence. |

Memo
- If the terminal operation is preferred, use the STARTACT command to configure as above.
- A variety of stopping methods is introduced in "8.3 Stopping Motion and Sequence" on page 93.

## 9.7  Error Messages Displayed on the Terminal

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

### ■ Error Messages for Editing

| | |
|---|---|
| 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: | The sequence name is invalid (Given sequence name exceeds 10 characters, the first character is N, S, n, s, etc.) |
| Sequence is locked. | |
| Cause/action: | At  "D x"(delete), "E x"(edit), "S x"(save) execution, 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. |
| Invalid editor syntax. | |
| Cause/action: | The syntax of the editor command is invalid.  (Extra text is found after an editor command, etc.) |
| End line must follow start line | |
| Cause/action: | While many Editor commands can take both start and end line numbers (Alter, Delete, List, Copy, Cut), the start line number must be before the end line number. |
| Missing argument. | |
| Cause/action: | Editor command prompt expecting and did not find a valid line number. |

### ■ Error Messages for Syntax

| | |
|---|---|
| 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: | Net is too deep. "Blocks" (WHILE−WEND, LOOP−ENDL, IF−ENDIF) can be "nested" inside each other up to 8 levels. |
| 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 99), 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. |

| | |
|---|---|
| **Invalid Operator** | |
| Cause/action: | Math operator not an allowable operator. |
| **Invalid user parameter name.** | |
| Cause/action: | Too many characters or invalid characters are entered as user parameter name. |
| **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 operators. |
| **Conditional expression expected.** | |
| Cause/action: | Missing parenthesis, or miss-spelled parameter names, typo in number, etc. |
| **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 operation.** | |
| Cause/action: | Operation contained elements that are not constants or known parameters. |
| **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. |
| **String argument not allowed here.** | |
| Cause/action: | MA S_name, WAIT S_name, LOOP S_name, etc is detected. |
| **Strings not allowed in conditionals.** | |
| Cause/action: | String entry is detected at conditional expression. |
| **Strings not allowed in operations.** | |
| Cause/action: | String entry is detected at math operator. |
| **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. SIGEND) |
| **Parameter cannot be displayed.** | |
| Cause/action: | Attempt to "View" a non-viewable parameter (e.g. KB, KBQ). |
| **Wait time 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.

| | |
|---|---|
| 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: | Abnormal hardware limit is detected during mechanical home seeking. |
| | Check the hardware limits, installation of HOMELS, wiring, and operation data used for the mechanical home seeking. |
| HOMELS not detected between +LS and –LS on HOME (3 sensor mode). | |
| Cause/action: | Check the hardware limits, installation of HOMELS, wiring, and operation data used for the mechanical home seeking. |
| TIM, SENSOR not detected on HOMELS at HOME | |
| Cause/action: | Check that the SENSOR 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. |
| Calculation result overflow | |
| Cause/action: | Calculation result over flow. |
| 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" command is updated while the device is operating or loses its holding torque. |
| | Execute the PC command while the device is at a standstill in the energized state. |
| Attempted to access non-existent user parameter | |
| Cause/action: | Non-exist variable is accessed. |
| Attempted to write to read-only parameter | |
| Cause/action: | Accessed to read only parameter. (Include prohibit access while motion, etc) |
| 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) |

## ■ 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: | Not feasible (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: | Not feasible (Command that starts motion is attempted while a sequence is running.) |
| Error: Action is not allowed. (Alarm is ON) | |
| Cause/action: | Not feasible (Command is attempted that is not executable while alarm is ON.) |
| Error: Action is not allowed. (Motion or I/O settings incompatible) | |
| Cause/action: | Not feasible (discrepancy of operation, discrepancy of I/O setting, and discrepancy between operation and I/O setting) |
| | One of following situations is detected. |
| | - Motion command attempted while current is OFF. |
| | - CV command is attempted while decelerating at MI, MA, EHOME motion. |
| | - MGHP, MGHN is attempted with VS=0. |
| | - MGHP, MGHN is attempted with HOMETYP=0 to 3 (2 sensor mode) and INLSP=INLSN=0 (±LS not configured). |
| | - MGHP, MGHN is attempted with HOMETYP=4 to 11 (1, 3 sensor mode) and INHOME=0 (HOME not configured). |
| 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 500001," "CURRENT 3") |
| Error: String is too long. | |
| Cause/action: | Length of string entered to user string parameter (S_xxx) exceeds 20 characters. |
| Error: Name is too long. | |
| Cause/action: | Length of string entered as the name of parameter (N_xxx, S_xxx) or sequence name exceeds 10 characters. |
| Error: Unsupported precision. | |
| Cause/action: | Numeric constant specified with precision over 3 decimal places. (e.g. "A=1.2345") |
| | Numeric constant specified with too much precision for its scale. |
| | Supported precision: 3 decimal places within ±500000 |
| | 2 decimal places within ±5000000 |
| | 1 decimal places within ±50000000 |
| | 0 decimal places within ±500000000 |
| Error: Parameter is read-only. | |
| Cause/action: | Attempted to write a read only parameter (e.g. "VC 10") |
| 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: Could not delete previous sequence. | |
|---|---|
| Cause/action: | Copy a sequence "over" another sequence, and could not delete the target sequence (maybe locked). |

| Error: Could not modify sequence. Executing? | |
|---|---|
| Cause/action: | Rename: Sequences cannot be changed. Sequences executing. Delete: Sequences cannot be changed. Sequences executing. Tried to modify sequences in some way, while a sequence was executing. Not permitted. |

| Error: Sequence directory full. | |
|---|---|
| Cause/action: | Copy: Required creating a new sequence, all 100 sequences exist already. Tried to create a sequence, by copying an existing sequence or saving from the editor. No free directory entries available: all 100 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) |

| Error: XXX(###) is out of range. | |
|---|---|
| Cause/action: | Parameter "XXX" is out of range. This may be caused by DPR, GA, GB change, followed by SAVEPRM or SAVEALL, and RESET. |

| Warning: XXX(###) is out of range. | |
|---|---|
| Cause/action: | Parameter "XXX" become out of range after DPR, GA, GB change. |

# 10 Control by CANopen Communication

This chapter explains how to control the **SCX10** using CANopen communication.
In this manual, physical I/O is defined as "Direct I/O" and CANopen I/O is defined as "Remote I/O."
The EDS file can be found on the attached CD-ROM. (File name: SCX10_x_x.eds, under CANopen_EDS folder)

## 10.1 Overview

The **SCX10** uses CANopen CiA 301 protocol and the product is tested and certified by the CiA (CAN in Automation).

- Via CANopen, you can make the motor move either by an immediate command or by program execution. All the functions for CANopen are the same as for USB and RS-232C. Refer to "6.4.2 Input Signals" on page 25, "6.4.3 Output Signals" on page 28 and "8 Features" on page 76 for signals and commands.
- All the settings related to CANopen or program creation are done by the supplied utility software, **Immediate Motion Creator for CM/SCX Series** or a general terminal software via USB or RS-232C. Refer to "■ CANopen settings" in "12 Command Reference" on page 169 for commands related to the CANopen and "9 Program Creation and Execution."
- While both the PDO (Process Data Object) and SDO (Service Data Object) are supported with the **SCX10**, the PDO is mainly explained in this manual.

10.2  Transmission Speed and ID Setting
10.3  LED Indication
10.4  Controlling I/O Message (PDO)
10.5  I/O Message Format (PDO)
10.6  I/O Message Command Code List (PDO)
10.7  Object Dictionary (SDO)

## 10.2 Transmission Speed and ID Setting

First, be sure to set the communication baud rate and the ID. The setting is done by the supplied utility software, **Immediate Motion Creator for CM/SCX Series** via USB or RS-232C. If a general terminal software is used, use the CANBAUD command (page 194) and CANID command (page195).

| Item | Description |
|---|---|
| Protocol | CiA 301 Ver.4.02 compliant |
| Transmission speed | Software setting<br>Selectable: 10 kbps, 20 kbps, 50 kbps, 125 kbps,<br>                 250 kbps, 500 kbps, 800 kbps, 1 Mbps |
| ID setting | Software setting (1 to 127) |

# 10.3 LED Indication

The **SCX10** has bicolor CAN LED to indicate communication statuses including run and error.

## ■ Red: Error

The CANopen error LED indicates the status of the CAN physical layer and errors due to missing CAN messages (sync, guard or heartbeat). If at a given time several errors are present, the error with the highest number is indicated (e.g. if NMT error and sync error occur, the sync error is indicated)

| LED | State | Description |
| --- | --- | --- |
| Off | No error | The device is in working condition |
| Single flash | Warning limit reached | At least one of the error counters of the CAN controller has reached or exceeded the warning level (too many error frames) |
| Double flash | Error control event | A guard event (NMT-slave or NMT-master) or a heartbeat event (heartbeat consumer) has occurred |
| On | Bus off | The CAN controller is bus off |

## ■ Green: Run

The CANopen run LED indicates the status of the CANopen network state machine.
If CAN has not been connected after turn the power on or the device is executing a reset, the CANopen run LED is off.

| LED | State | Description |
| --- | --- | --- |
| Blinking | Preoperational | The device is in a preoperational state |
| Single flash | Stopped | The device is in a stopped state |
| On | Operational | The device is in an operational state |

**Note**  In case there is a conflict between the LED being green versus red, the LED will be turned on red because errors have a higher priority.

## ■ LED Indicator States and Flash Rates

The following indicator states are defined.
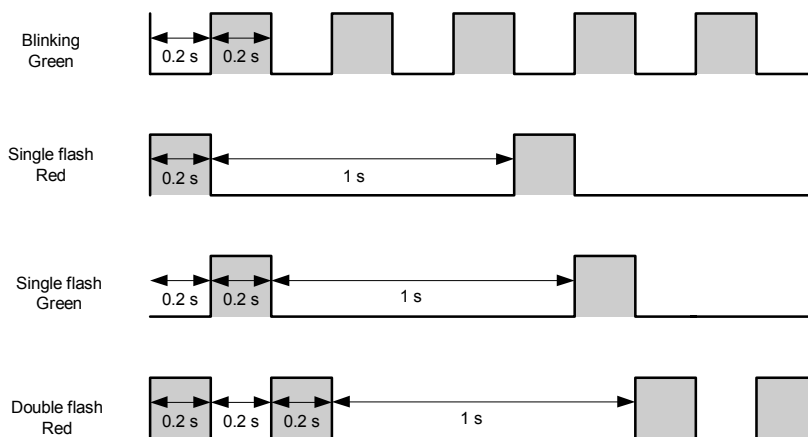
# 10.4 Controlling I/O Message (PDO)

## ■ I/O Message Format

The **SCX10** is controlled via CANopen by the following 8 byte I/O message formats; master to the **SCX10**, and the **SCX10** to a master, respectively.

• RPDO (Receive Process Data Object) : Master to **SCX10**

The RPDO acts as an input of the **SCX10**.

RPDO Mapping (Master → **SCX10**)

|  | Bit [7] | Bit [6] | Bit [5] | Bit [4] | Bit [3] | Bit [2] | Bit [1] | Bit [0] |
|---|---|---|---|---|---|---|---|---|
| Byte [0] | RIN8 | RIN7 | RIN6 | RIN5 | RIN4 | RIN3 | RIN2 | RIN1 |
| Byte [1] | - | FREE | CON | ABORT | MGHN | MCN | MCP | START |
| Byte [2] | COMMAND | | | | | | | |
| Byte [3] | - | TRIG | | | | | | |
| Byte [4] | DATA | | | | | | | |
| Byte [5] | | | | | | | | |
| Byte [6] | | | | | | | | |
| Byte [7] | | | | | | | | |

• TPDO (Transmit Process Data Object) : **SCX10** to Master

The TPDO acts as an output of the **SCX10**.

TPDO Mapping (**SCX10** → Master)

|  | Bit [7] | Bit [6] | Bit [5] | Bit [4] | Bit [3] | Bit [2] | Bit [1] | Bit [0] |
|---|---|---|---|---|---|---|---|---|
| Byte [0] | ALM | WNG | ROUT6 | ROUT5 | ROUT4 | ROUT3 | ROUT2 | ROUT1 |
| Byte [1] | 0 | 1 | READY | LC | HOME_P | MOVE | END | START_R |
| Byte [2] | COMMAND _R | | | | | | | |
| Byte [3] | STATUS | TRIG_R | | | | | | |
| Byte [4] | DATA_R | | | | | | | |
| Byte [5] | | | | | | | | |
| Byte [6] | | | | | | | | |
| Byte [7] | | | | | | | | |

## ■ Using as a Switch

• RPDO (INPUT of **SCX10**)

Each bit in the Byte [0] and Byte [1] acts exactly the same as a physical switch used to control a signal on the I/O connector on the **SCX10**. For example, to make the RIN1 turned ON, set the bit, Byte [0]-Bit [0] in RPDO to "1" (ON). The **SCX10** receives this command and takes it as the "remote input 1 is turned ON."

RPDO (Master → **SCX10**)

|  | Bit [7] | Bit [6] | Bit [5] | Bit [4] | Bit [3] | Bit [2] | Bit [1] | Bit [0] |
|---|---|---|---|---|---|---|---|---|
| Byte [0] | RIN8 | RIN7 | RIN6 | RIN5 | RIN4 | RIN3 | RIN2 | **RIN1** |
| Byte [1] | - | FREE | CON | ABORT | MGHN | MCN | MCP | START |
| Byte [2] | COMMAND | | | | | | | |
| Byte [3] | - | TRIG | | | | | | |
| Byte [4] | DATA | | | | | | | |
| Byte [5] | | | | | | | | |
| Byte [6] | | | | | | | | |
| Byte [7] | | | | | | | | |

In the Byte [1] area, signals that are commonly used are pre-assigned. For example, set the bit, Byte [1]-Bit [5] to 1. The motor current will be turned ON.

RPDO (Master → **SCX10**)

|  | Bit [7] | Bit [6] | Bit [5] | Bit [4] | Bit [3] | Bit [2] | Bit [1] | Bit [0] |
|---|---|---|---|---|---|---|---|---|
| Byte [0] | RIN8 | RIN7 | RIN6 | RIN5 | RIN4 | RIN3 | RIN2 | RIN1 |
| Byte [1] | - | FREE | CON | ABORT | MGHN | MCN | MCP | START |
| Byte [2] | COMMAND | | | | | | | |
| Byte [3] | - | TRIG | | | | | | |
| Byte [4] | DATA | | | | | | | |
| Byte [5] | | | | | | | | |
| Byte [6] | | | | | | | | |
| Byte [7] | | | | | | | | |

Before any motion can occur, the motor current needs to be turned ON. Always keep this bit ON under normal operating conditions.

The functions of all pre-assigned I/Os are the same as the I/O signals on the I/O connector on the **SCX10**. See "6.4.2 Input Signals" on page 25 and "6.4.3 Output Signals" on page 28 - for details of each signal.

- TPDO (OUTPUT of **SCX10**)

Each bit in the Byte [0] and Byte [1] acts exactly the same as a physical switch or an output signal on the I/O connector on the **SCX10** to control a master controller. For example, if the ROUT1 is 1 (ON), that means the remote output ROUT1 on the **SCX10** is turned ON, and the remote input signal 1 of the master controller is turned ON.

TPDO (**SCX10** → Master)

|  | Bit [7] | Bit [6] | Bit [5] | Bit [4] | Bit [3] | Bit [2] | Bit [1] | Bit [0] |
|---|---|---|---|---|---|---|---|---|
| Byte [0] | ALM | WNG | ROUT6 | ROUT5 | ROUT4 | ROUT3 | ROUT2 | ROUT1 |
| Byte [1] | 0 | 1 | READY | LC | HOME_P | MOVE | END | START_R |
| Byte [2] | COMMAND_R | | | | | | | |
| Byte [3] | STATUS | TRIG_R | | | | | | |
| Byte [4] | DATA_R | | | | | | | |
| Byte [5] | | | | | | | | |
| Byte [6] | | | | | | | | |
| Byte [7] | | | | | | | | |

In the same manner, if the READY is 1 (ON), it indicates that the **SCX10** is in the ready status.

TPDO (**SCX10** → Master)

|  | Bit [7] | Bit [6] | Bit [5] | Bit [4] | Bit [3] | Bit [2] | Bit [1] | Bit [0] |
|---|---|---|---|---|---|---|---|---|
| Byte [0] | ALM | WNG | ROUT6 | ROUT5 | ROUT4 | ROUT3 | ROUT2 | ROUT1 |
| Byte [1] | 0 | 1 | READY | LC | HOME_P | MOVE | END | START_R |
| Byte [2] | COMMAND_R | | | | | | | |
| Byte [3] | STATUS | TRIG_R | | | | | | |
| Byte [4] | DATA_R | | | | | | | |
| Byte [5] | | | | | | | | |
| Byte [6] | | | | | | | | |
| Byte [7] | | | | | | | | |

## ■ Assigning Signals

While the RIN1-RIN8 in the RPDO and ROUT1-ROUT6 in the TPDO are used as general I/O, you may also assign any of these I/Os to specific system inputs/outputs such as the ALMCLR (alarm clear), MSTOP (motor stop), MBFREE (magnetic brake free) and PSTS (pause status) in addition to pre-assigned I/Os explained before, and use these signals as the signals on the I/O connector on the **SCX10**.

Example: MSTOP is assigned to RIN1

RPDO (Master → **SCX10**)

|  | Bit [7] | Bit [6] | Bit [5] | Bit [4] | Bit [3] | Bit [2] | Bit [1] | Bit [0] |
|---|---|---|---|---|---|---|---|---|
| Byte [0] | RIN8 | RIN7 | RIN6 | RIN5 | RIN4 | RIN3 | RIN2 | MSTOP |
| Byte [1] | - | FREE | CON | ABORT | MGHN | MCN | MCP | START |
| Byte [2] | COMMAND | | | | | | | |
| Byte [3] | - | TRIG | | | | | | |
| Byte [4] | DATA | | | | | | | |
| Byte [5] | | | | | | | | |
| Byte [6] | | | | | | | | |
| Byte [7] | | | | | | | | |

The signals assignable to remote I/Os are listed below. All assignments are done by the supplied utility software, **Immediate Motion Creator for CM/SCX Series** via a USB or RS-232C connection. If a general terminal software is used, refer to the chart below.

The functions of all system I/Os are the same as the I/O signals on the I/O connector on the **SCX10**. See "6.4.2 Input Signals" on page 25 and "6.4.3 Output Signals" on page 28 - for details of each signal.

|  | Signal | | Command for Assignment |
|---|---|---|---|
| INPUT (RPDO) | ALMCLR | (alarm clear) | RINALMCLR |
| | CONT | (continue motion) | RINCONT |
| | HOME | (home sensor) | RINHOME |
| | LSP | (limit switch positive) | RINLSP |
| | LSN | (limit switch negative) | RINLSN |
| | MSTOP | (motor stop) | RINMSTOP |
| | MGHP | (move go home positive) | RINMGHP |
| | PAUSE | (pause motion) | RINPAUSE |
| | PAUSECL | (pause clear) | RINPAUSECL |
| | PECLR | (position error clear) | RINPECLR |
| | PSTOP | (panic stop) | RINPSTOP |
| | SENSOR | (sensor) | RINSENSOR |
| | TL | (torque limiting/push-motion operation /current cutback release) | RINTL |
| OUTPUT (TPDO) | ABSDATA | (driver current position data ready) | ROUTABSDATA |
| | MBFREE | (magnetic brake free) | ROUTMBFREE |
| | PSTS | (pause status) | ROUTPSTS |
| | RUN | (sequence running) | ROUTRUN |

## ■ Sending Message Commands

Let's make the motor move, starting with an index motion. The following procedure is required to make an incremental move, as same as the communication via the USB/RS-232C. Assume the user unit is set to "Rev" (revolution).

> 1. Set the starting velocity, VS=0.1.    (0.1 Rev/sec)
> 2. Set the running velocity, VR=1.    (1 Rev/sec)
> 3. Set the acceleration time, TA=0.2.    (0.2 Rev/sec)
> 4. Set the deceleration time, TD=0.2.    (0.2 Rev/sec)
> 5. Set the motion distance, DIS=1.    (1 Rev)
> 6. Set the move index command, MI.

- Sending a Message Command

The message command code is formed by 14 bits. The parameter following the command is written in the DATA area.

To set the first parameter VS=0.1, find the command code for a "Write" of the VS from the following "10.6 I/O Message Command Code List (PDO)." "1142h" is found to be the command code for a write of the VS. The command code is written as a hexadecimal number (last digit "h" means hexadecimal).

The parameter (0.1) following the command (VS) is set in the DATA area. The data format is a signed integer and little endian (start from least significant bytes). Since the user unit can be up to three decimal places, you must always multiply the user unit value by 1000. Thus, "0.1" becomes "100." It is 64h.

Set these command code and data into designated areas as follows.

RPDO (Master → **SCX10**)

|         | Bit [7] | Bit [6] | Bit [5] | Bit [4] | Bit [3] | Bit [2] | Bit [1] | Bit [0] |
|---------|---------|---------|---------|---------|---------|---------|---------|---------|
| Byte [0] | RIN8 | RIN7 | RIN6 | RIN5 | RIN4 | RIN3 | RIN2 | RIN1 |
| Byte [1] | - | FREE | CON | ABORT | MGHN | MCN | MCP | START |
| Byte [2] | 1142h | | | | | | | |
| Byte [3] | - | TRIG | | | | | | |
| Byte [4] | 64h | | | | | | | |
| Byte [5] | | | | | | | | |
| Byte [6] | | | | | | | | |
| Byte [7] | | | | | | | | |

∗ The CON needs to be kept ON for motor operation.

The data is set to the **SCX10** when the trigger (TRIG) bit is set to 1 (ON).

RPDO (Master → **SCX10**)

|         | Bit [7] | Bit [6] | Bit [5] | Bit [4] | Bit [3] | Bit [2] | Bit [1] | Bit [0] |
|---------|---------|---------|---------|---------|---------|---------|---------|---------|
| Byte [0] | RIN8 | RIN7 | RIN6 | RIN5 | RIN4 | RIN3 | RIN2 | RIN1 |
| Byte [1] | - | FREE | CON | ABORT | MGHN | MCN | MCP | START |
| Byte [2] | 1142h | | | | | | | |
| Byte [3] | - | TRIG | | | | | | |
| Byte [4] | 64h | | | | | | | |
| Byte [5] | | | | | | | | |
| Byte [6] | | | | | | | | |
| Byte [7] | | | | | | | | |

Note that all the command codes and data need to be kept until the TRIG to be set to 1, since all data is read by the **SCX10** when the trigger state is changed from 0 to 1.

- **Confirming a Message Receipt and Status**

After sending message command, a confirmation of receipt is done by checking the TPDO message.
The TRIG_R (trigger response) indicates that the process is completed (the message is received), and
COMMAND_R areas show what are received. (It is right response that the data of the DATA_R area is 0 when
a write command is issued in the RPDO.)

TPDO (**SCX10** → Master)

|  | Bit [7] | Bit [6] | Bit [5] | Bit [4] | Bit [3] | Bit [2] | Bit [1] | Bit [0] |
|---|---|---|---|---|---|---|---|---|
| Byte [0] | ALM | WNG | ROUT6 | ROUT5 | ROUT4 | ROUT3 | ROUT2 | ROUT1 |
| Byte [1] | 0 | 1 | READY | LC | HOME_P | MOVE | END | START_R |
| Byte [2] | 1142h | | | | | | | |
| Byte [3] | STATUS | TRIG_R | | | | | | |
| Byte [4] | 0h | | | | | | | |
| Byte [5] | | | | | | | | |
| Byte [6] | | | | | | | | |
| Byte [7] | | | | | | | | |

If the STATUS is 1 (ON), that means that there is a process error.

TPDO (**SCX10** → Master)

|  | Bit [7] | Bit [6] | Bit [5] | Bit [4] | Bit [3] | Bit [2] | Bit [1] | Bit [0] |
|---|---|---|---|---|---|---|---|---|
| Byte [0] | ALM | WNG | ROUT6 | ROUT5 | ROUT4 | ROUT3 | ROUT2 | ROUT1 |
| Byte [1] | 0 | 1 | READY | LC | HOME_P | MOVE | END | START_R |
| Byte [2] | 1142h | | | | | | | |
| Byte [3] | STATUS | TRIG_R | | | | | | |
| Byte [4] | 0h | | | | | | | |
| Byte [5] | | | | | | | | |
| Byte [6] | | | | | | | | |
| Byte [7] | | | | | | | | |

Errors occur if the condition is not allowed such as the parameter is out of range or the parameters are sent when
it is not allowed.
After confirming TRIG_R, you may clear all the data from the RPDO, except Byte [0] and Byte [1].

RPDO (Master → **SCX10**)

|  | Bit [7] | Bit [6] | Bit [5] | Bit [4] | Bit [3] | Bit [2] | Bit [1] | Bit [0] |
|---|---|---|---|---|---|---|---|---|
| Byte [0] | RIN8 | RIN7 | RIN6 | RIN5 | RIN4 | RIN3 | RIN2 | RIN1 |
| Byte [1] | - | FREE | CON | ABORT | MGHN | MCN | MCP | START |
| Byte [2] | 0h | | | | | | | |
| Byte [3] | - | TRIG | | | | | | |
| Byte [4] | 0h | | | | | | | |
| Byte [5] | | | | | | | | |
| Byte [6] | | | | | | | | |
| Byte [7] | | | | | | | | |

The **SCX10** is now ready to receive next parameters.

Set other parameters including the VR, TA, TD, DIS by using the same procedure.
Note that the time is in milliseconds. In other words, always multiply by 1000 when seconds are used for the
time unit. For setting TA=0.2, 200 should be set in the DATA area.
After setting all parameters, set the move index command MI, and then set the TRIG to 1.
When the TRIG to 1, the motor will start to move and will rotate one revolution.

## ■ Requesting Current Parameter and Status

Not only can you send commands to the **SCX10**, but you may also ask the **SCX10** about a parameter's value/status.

For instance, if a verifying the current VS value is required, find the command code for a "Read" of the VS from the following "IO Message command code list." "0142h" is found to be the command code for a read of the VS. Set the command code as previously explained.

RPDO (Master → **SCX10**)

|  | Bit [7] | Bit [6] | Bit [5] | Bit [4] | Bit [3] | Bit [2] | Bit [1] | Bit [0] |
|---|---|---|---|---|---|---|---|---|
| Byte [0] | RIN8 | RIN7 | RIN6 | RIN5 | RIN4 | RIN3 | RIN2 | RIN1 |
| Byte [1] | - | FREE | CON | ABORT | MGHN | MCN | MCP | START |
| Byte [2] | | | | 0142h | | | | |
| Byte [3] | - | TRIG | | | | | | |
| Byte [4] | | | | | | | | |
| Byte [5] | | | | 0h | | | | |
| Byte [6] | | | | | | | | |
| Byte [7] | | | | | | | | |

∗ The DATA value is not required in a read command.

Set the TRIG to 1. After setting the TRIG to 1, the **SCX10** will send the value of the requested parameter to the master using TPDO as follows.

TPDO (**SCX10** → Master)

|  | Bit [7] | Bit [6] | Bit [5] | Bit [4] | Bit [3] | Bit [2] | Bit [1] | Bit [0] |
|---|---|---|---|---|---|---|---|---|
| Byte [0] | ALM | WNG | ROUT6 | ROUT5 | ROUT4 | ROUT3 | ROUT2 | ROUT1 |
| Byte [1] | 0 | 1 | READY | LC | HOME_P | MOVE | END | START_R |
| Byte [2] | | | | 0142h | | | | |
| Byte [3] | STATUS | TRIG_R | | | | | | |
| Byte [4] | | | | | | | | |
| Byte [5] | | | | 64h | | | | |
| Byte [6] | | | | | | | | |
| Byte [7] | | | | | | | | |

By using same method as writing the parameter, this is read as VS=0.1.

## ■ Set the TRIG back to "0" for the next command execution.I/O Command and Message Command

In some cases, the same command can be set by either I/O command or message command.

- **Using an I/O Command**

For an example of an I/O command, the MCP (move continuously positive) can be commanded by just setting the bit, Byte [1]-Bit [1] to 1 (ON). The motor starts to move as soon as MCP is set to 1.

RPDO (Master → **SCX10**)

|          | Bit [7] | Bit [6] | Bit [5] | Bit [4] | Bit [3] | Bit [2] | Bit [1] | Bit [0] |
|----------|---------|---------|---------|---------|---------|---------|---------|---------|
| Byte [0] | RIN8    | RIN7    | RIN6    | RIN5    | RIN4    | RIN3    | RIN2    | RIN1    |
| Byte [1] | -       | FREE    | CON     | ABORT   | MGHN    | MCN     | MCP     | START   |
| Byte [2] | COMMAND |         |         |         |         |         |         |         |
| Byte [3] | -       | TRIG    |         |         |         |         |         |         |
| Byte [4] | DATA    |         |         |         |         |         |         |         |
| Byte [5] |         |         |         |         |         |         |         |         |
| Byte [6] |         |         |         |         |         |         |         |         |
| Byte [7] |         |         |         |         |         |         |         |         |

To stop the motion, set the ABORT bit, Byte [1]-Bit [4] to 1 (ON). The MSTOP, PAUSE and PSTOP commands can also be used if they are assigned to any of the RIN1-RIN8.

- **Using a Message Command**

For an example of using a message command, set the command code for the MCP "1C12h."
∗ Note that digit "C" in hexadecimal is defined as "1100" in binary.
∗ The data area is blank (set to zero), since MCP does not require a parameter.

RPDO (Master → **SCX10**)

|          | Bit [7] | Bit [6] | Bit [5] | Bit [4] | Bit [3] | Bit [2] | Bit [1] | Bit [0] |
|----------|---------|---------|---------|---------|---------|---------|---------|---------|
| Byte [0] | RIN8    | RIN7    | RIN6    | RIN5    | RIN4    | RIN3    | RIN2    | RIN1    |
| Byte [1] | -       | FREE    | CON     | ABORT   | MGHN    | MCN     | MCP     | START   |
| Byte [2] | 1C12h   |         |         |         |         |         |         |         |
| Byte [3] | -       | TRIG    |         |         |         |         |         |         |
| Byte [4] | 0h      |         |         |         |         |         |         |         |
| Byte [5] |         |         |         |         |         |         |         |         |
| Byte [6] |         |         |         |         |         |         |         |         |
| Byte [7] |         |         |         |         |         |         |         |         |

The motor starts to rotate in positive direction when TRIG bit is set to 1.

RPDO (Master → **SCX10**)

|          | Bit [7] | Bit [6] | Bit [5] | Bit [4] | Bit [3] | Bit [2] | Bit [1] | Bit [0] |
|----------|---------|---------|---------|---------|---------|---------|---------|---------|
| Byte [0] | RIN8    | RIN7    | RIN6    | RIN5    | RIN4    | RIN3    | RIN2    | RIN1    |
| Byte [1] | -       | FREE    | CON     | ABORT   | MGHN    | MCN     | MCP     | START   |
| Byte [2] | 1C12h   |         |         |         |         |         |         |         |
| Byte [3] | -       | TRIG    |         |         |         |         |         |         |
| Byte [4] | 0h      |         |         |         |         |         |         |         |
| Byte [5] |         |         |         |         |         |         |         |         |
| Byte [6] |         |         |         |         |         |         |         |         |
| Byte [7] |         |         |         |         |         |         |         |         |

Set the TRIG bit back to "0" for the next command execution.

**Memo**  Various message commands can also be used to stop the motion instead of using I/O command such as the ABORT as explained above. See "■ Motion Commands" on page 161. Set the command code of each stop command and then set the TRIG bit to 1.

# ■ Executing a Sequence

There are two ways to make a sequence execute, one is to select a sequence using remote inputs and the other one is using a message command.

- Using I/O Commands (Remote Inputs)

Set the program number (in decimal format) in Byte [0] area in binary format.
For instance, if the program number 3 needs to be executed, set the program number as below.

RPDO (Master → **SCX10**)

|          | Bit [7] | Bit [6] | Bit [5] | Bit [4] | Bit [3] | Bit [2] | Bit [1] | Bit [0] |
|----------|---------|---------|---------|---------|---------|---------|---------|---------|
| Byte [0] | RIN8    | RIN7    | RIN6    | RIN5    | RIN4    | RIN3    | RIN2    | RIN1    |
| Byte [1] | -       | FREE    | CON     | ABORT   | MGHN    | MCN     | MCP     | START   |
| Byte [2] | COMMAND | | | | | | | |
| Byte [3] | -       | TRIG    | | | | | | |
| Byte [4] | DATA | | | | | | | |
| Byte [5] | | | | | | | | |
| Byte [6] | | | | | | | | |
| Byte [7] | | | | | | | | |

When the START bit is set to 1, the program number 3 is executed.

RPDO (Master → **SCX10**)

|          | Bit [7] | Bit [6] | Bit [5] | Bit [4] | Bit [3] | Bit [2] | Bit [1] | Bit [0] |
|----------|---------|---------|---------|---------|---------|---------|---------|---------|
| Byte [0] | RIN8    | RIN7    | RIN6    | RIN5    | RIN4    | RIN3    | RIN2    | RIN1    |
| Byte [1] | -       | FREE    | CON     | ABORT   | MGHN    | MCN     | MCP     | START   |
| Byte [2] | COMMAND | | | | | | | |
| Byte [3] | -       | TRIG    | | | | | | |
| Byte [4] | DATA | | | | | | | |
| Byte [5] | | | | | | | | |
| Byte [6] | | | | | | | | |
| Byte [7] | | | | | | | | |

After the program is started, set the START bit back to "0" for the next command execution.

Note
- When any of the RIN1-RIN8 is assigned to specific system inputs such as the ALMCLR (alarm clear) and MSTOP (motor stop), these bits (inputs) cannot be used to select the program number. The value of an assigned input is always 0.
- Try to avoid a use of the same input both in the program and for the program selection. For example, If the program is written so that it refers RIN1 in the beginning of the sequence, you must clear RIN1 status right after executing the program. (Otherwise, the **SCX10** reads RIN1 as ON in the program if the RIN is set to 1 at the program selection.) In order to avoid it, the method that performs a program selection using the message command shown as follows is effective.

- Using a Message Command

  Set the command code for run sequence, the RUN (1C05h) in COMMAND area, and set program number 3 (03h) in the DATA area as below.

  RPDO (Master → **SCX10**)

  |          | Bit [7] | Bit [6] | Bit [5] | Bit [4] | Bit [3] | Bit [2] | Bit [1] | Bit [0] |
  |----------|---------|---------|---------|---------|---------|---------|---------|---------|
  | Byte [0] | RIN8    | RIN7    | RIN6    | RIN5    | RIN4    | RIN3    | RIN2    | RIN1    |
  | Byte [1] | -       | FREE    | CON     | ABORT   | MGHN    | MCN     | MCP     | START   |
  | Byte [2] | 1C05h   |         |         |         |         |         |         |         |
  | Byte [3] | -       | TRIG    |         |         |         |         |         |         |
  | Byte [4] | 03h     |         |         |         |         |         |         |         |
  | Byte [5] |         |         |         |         |         |         |         |         |
  | Byte [6] |         |         |         |         |         |         |         |         |
  | Byte [7] |         |         |         |         |         |         |         |         |

When the TRIG bit is set to 1, program number 3 is executed.

  RPDO (Master → **SCX10**)

  |          | Bit [7] | Bit [6] | Bit [5] | Bit [4] | Bit [3] | Bit [2] | Bit [1] | Bit [0] |
  |----------|---------|---------|---------|---------|---------|---------|---------|---------|
  | Byte [0] | RIN8    | RIN7    | RIN6    | RIN5    | RIN4    | RIN3    | RIN2    | RIN1    |
  | Byte [1] | -       | FREE    | CON     | ABORT   | MGHN    | MCN     | MCP     | START   |
  | Byte [2] | 1C05h   |         |         |         |         |         |         |         |
  | Byte [3] | -       | TRIG    |         |         |         |         |         |         |
  | Byte [4] | 03h     |         |         |         |         |         |         |         |
  | Byte [5] |         |         |         |         |         |         |         |         |
  | Byte [6] |         |         |         |         |         |         |         |         |
  | Byte [7] |         |         |         |         |         |         |         |         |

Set the TRIG bit back to "0" for the next command execution.

# 10.5 I/O Message Format (PDO)

## ■ RPDO Mapping: Master → SCX10

|          | Bit [7] | Bit [6] | Bit [5] | Bit [4] | Bit [3] | Bit [2] | Bit [1] | Bit [0] |
|----------|---------|---------|---------|---------|---------|---------|---------|---------|
| Byte [0] | RIN8    | RIN7    | RIN6    | RIN5    | RIN4    | RIN3    | RIN2    | RIN1    |
| Byte [1] | -       | FREE    | CON     | ABORT   | MGHN    | MCN     | MCP     | START   |
| Byte [2] | COMMAND |         |         |         |         |         |         |         |
| Byte [3] | -       | TRIG    |         |         |         |         |         |         |
| Byte [4] | DATA    |         |         |         |         |         |         |         |
| Byte [5] |         |         |         |         |         |         |         |         |
| Byte [6] |         |         |         |         |         |         |         |         |
| Byte [7] |         |         |         |         |         |         |         |         |

| Signals | Description | Range |
|---------|-------------|-------|
| RIN1 to RIN8 | 1) Remote General Inputs<br>All inputs can be assigned to system remote input signals. (PSTOP, MSTOP, MGHP, PAUSE, CONT, PAUSECL, ALMCLR, SENSOR, HOME, +LS, -LS, PECLR, TL).<br>2) Sequence Number Select<br>If RINx is assigned to a system remote input signal, RINx is 0. | 0: Not Active<br>1: Active |
| START | Start Sequence<br>When the signal level is changed from 0 to 1, start the sequence whose sequence number (0-99) is determined by the selected RIN1 to RIN7. (Edge trigger start)<br>Set the starting method using the STARTACT command. | <STARTACT=0><br>0: No Action<br>1: Start Sequence<br><STARTACT=1><br>0: Abort Sequence<br>1: Start Sequence |
| MCP | Move Continuously Positive | 0: No Action<br>1: Start Motion Positive |
| MCN | Move Continuously Negative | 0: No Action<br>1: Start Motion Negative |
| MGHN | Move Go Home Negative<br>(HOMETYP command determines the type of home seeking motion.) | 0: No Action<br>1: Go Home |
| ABORT | Abort Motion and Sequence Execution | <Motion><br>0: No Action<br>1: Stop Motions<br><Sequence Program><br>0: No Action<br>1: Abort |
| CON | Motor Current ON/OFF | 0: Motor Current OFF<br>1: Motor Current ON |
| FREE | Motor Current OFF, Magnetic Brake Free | 0: Normal Condition<br>1: Motor Shaft Free |
| COMMAND | Command Code for Parameter, Monitor or Maintenance Command | See I/O Message Command Code List in detail. |
| TRIG | Trigger for Handshake<br>When parameter for reading and writing parameter and when monitor and maintenance command are transmitted, this bit is set from 0 to 1 to indicate data is ready for loading | 0: No Action<br>1: Execute |
| DATA | Data for Parameter Writing or Argument of a Command. | See I/O Message Command Code List in detail. |

## ■ TPDO Mapping: SCX10 → Master

| | Bit [7] | Bit [6] | Bit [5] | Bit [4] | Bit [3] | Bit [2] | Bit [1] | Bit [0] |
|---|---|---|---|---|---|---|---|---|
| Byte [0] | ALM | WNG | ROUT6 | ROUT5 | ROUT4 | ROUT3 | ROUT2 | ROUT1 |
| Byte [1] | 0 | 1 | READY | LC | HOME_P | MOVE | END | START_R |
| Byte [2] | COMMAND_R | | | | | | | |
| Byte [3] | STATUS | TRIG_R | | | | | | |
| Byte [4] | DATA_R | | | | | | | |
| Byte [5] | | | | | | | | |
| Byte [6] | | | | | | | | |
| Byte [7] | | | | | | | | |

| Signals | Description | Range |
|---|---|---|
| ROUT1 to ROUT6 | Remote General Outputs<br>All outputs can be assigned to system output signals.<br>(RUN, PSTS, MBFREE, ABSDATA) | 0: Not Active<br>1: Active |
| WNG | Warning | 0: No Warning<br>1: Warning Occurred |
| ALM | Alarm | 0: No Alarm<br>1: Alarm Occurred |
| START_R | Echo of START Input Signal | 0: START input signal is 0<br>1: START input signal is 1 |
| END | Motion End | When ENDACT=0, DEND=0<br>·0: Pulse Generating<br>·1: End of Pulse Generating<br><br>When ENDACT>0 (End area), DEND=0<br>·0: Pulse Generating or not in END Area<br>·1: End of Pulse and within END Area<br><br>When DEND=1 (ENDACT: Unrelated)<br>·0: Driver END Signal Inactive<br>·1: Driver END Signal Active |
| MOVE | Motor Moving | 0: Stopped<br>1: Moving (pulses are generating or sensor-less mechanical home seeking is in a operation) |
| HOME_P | Home Position | 0: Not in HOME Position<br>1: At HOME Position |
| LC | Limiting Condition<br>- **AR** Series driver: When the motor is in a state of push condition (the position deviation is 1.8 degrees or more) in the normal operating mode, or when the motor torque reaches to the preset value in the current control operating mode.<br>- **NX** Series driver: When the motor torque reaches the preset value while the torque limiting function is used.<br>- **RBK** Series driver: Under current cutback condition<br>- **ESMC** controller: While pressing the mechanical home when performing sensor-less mechanical home seeking operation. | 0: Not in Limiting Condition<br>1: Limiting Condition |
| READY | Operation Ready<br>- Possible to execute sequence program<br>- Possible to start motion command (MA, MI, MCP, MCN, MGHP, MGHN, MIx, EHOME, CONT)<br>∗ This bit is 1 when RUN, MOVE and ALM outputs are 0. | 0: Not Ready<br>1: Ready |
| COMMAND_R | Echo of Command Code | See I/O Message Command Code List in detail. |
| TRIG_R | Echo of Trigger | 0: Not yet Process<br>1: Processing Completed |

| STATUS | If sequence program cannot be executed, this bit  is 1. | 0: Normal<br>1: Process Error |
|---|---|---|
| DATA_R | Result of Read Parameter, Monitor or Maintenance Command | See I/O Message Command Code list in detail. |

# 10.6 I/O Message Command Code List (PDO)

## ■ Format

Data formats for user unit and time: For user unit, always multiply the user unit by 1000, since the decimal point is not used in message commands. The unit of time is in millisecond.

Ex. Incremental motion distance=12.7mm (user unit): Set the value, 12700 in the command message.
Running velocity=10mm/sec (user unit): Set the value, 10000 in the command message.

The MAXPOS and MAXVEL in the chart below are also 1000 times greater than the formula in the command description and shown on the terminal window.

Acceleration time=1 sec: Set the value, 1000 in the command message.

## ■ Motion Data

**Memo** : Refer to the "12 Command Reference" and "8 Features" for the detail of commands.

| Command Code | | Description | Range | Factory Setting | Command |
|---|---|---|---|---|---|
| Read | Write | | | | |
| 0140h | 1140h | Acceleration Time | 1 to 500,000 [msec] | 500 | TA |
| 0141h | 1141h | Deceleration Time | 1 to 500,000 [msec] | 500 | TD |
| 0142h | 1142h | Starting Velocity | 0 to MAXVEL [UU/sec] | 100 | VS |
| 0160h | 1160h | Mechanical Home Seeking Mode | 0 to 12 | 0 | HOMETYP |
| 0164h | 1164h | Offset for Mechanical Home Seeking | -MAXPOS to +MAXPOS [UU] | 0 | OFFSET |
| 0380h | 1380h | Driver Operation Data | 0 to 7 | 0 | DD |
| 0400h | 1400h | Distance for Incremental Motion | -MAXPOS to +MAXPOS [UU] | 0 | DIS |
| 0401h | 1401h | Position Array Data No.1 | -MAXPOS to +MAXPOS [UU] | 0 | POS[1] |
| : | : | : | : | : | : |
| 0464h | 1464h | Position Array Data No.100 | -MAXPOS to +MAXPOS [UU] | 0 | POS[100] |
| 0480h | 1480h | Running Velocity | 1 to MAXVEL [UU/sec] | 1000 | VR |
| 04A0h | 14A0h | Distance from SENSOR Input to the Stop Position | 0 to MAXPOS [UU] | 0 | SCHGPOS |
| 04A1h | 14A1h | Velocity after SENSOR Input | 1 to MAXVEL [UU/sec] | 1000 | SCHGVR |
| 0500h | 1500h | Distance or Destination for Link Segment '0' | -MAXPOS to +MAXPOS [UU] | 0 | DIS0 |
| 0501h | 1501h | Distance or Destination for Link Segment '1' | -MAXPOS to +MAXPOS [UU] | 0 | DIS1 |
| 0502h | 1502h | Distance or Destination for Link Segment '2' | -MAXPOS to +MAXPOS [UU] | 0 | DIS2 |
| 0503h | 1503h | Distance or Destination for Link Segment '3' | -MAXPOS to +MAXPOS [UU] | 0 | DIS3 |
| 0510h | 1510h | Running Velocity of Link Segment '0' | 1 to MAXVEL [UU/sec] | 1000 | VR0 |
| 0511h | 1511h | Running Velocity of Link Segment '1' | 1 to MAXVEL [UU/sec] | 1000 | VR1 |
| 0512h | 1512h | Running Velocity of Link Segment '2' | 1 to MAXVEL [UU/sec] | 1000 | VR2 |
| 0513h | 1513h | Running Velocity of Link Segment '3' | 1 to MAXVEL [UU/sec] | 1000 | VR3 |
| 0520h | 1520h | Link Type for Link Segment '0' | 0: Absolute<br>1: Incremental | 1 | INCABS0 |
| 0521h | 1521h | Link Type for Link Segment '1' | 0: Absolute<br>1: Incremental | 1 | INCABS1 |
| 0522h | 1522h | Link Type for Link Segment '2' | 0: Absolute<br>1: Incremental | 1 | INCABS2 |
| 0523h | 1523h | Link Type for Link Segment '3' | 0: Absolute<br>1: Incremental | 1 | INCABS3 |
| 0530h | 1530h | Link Control for Link Segment '0' | 0: No link<br>1: Link | 0 | LINK0 |
| 0531h | 1531h | Link Control for Link Segment '1' | 0: No link<br>1: Link | 0 | LINK1 |
| 0532h | 1532h | Link Control for Link Segment '2' | 0: No link<br>1: Link | 0 | LINK2 |
| 0900h | 1900h | Running Timer | 0 to 500,000,000 [msec] | - | TIMER |

## ■ Motion Commands

| Command Code | Description | Range | Command |
|---|---|---|---|
| 1C00h | Soft Stop | n/a | SSTOP |
| 1C01h | Hard Stop | n/a | HSTOP |
| 1C02h | Panic Stop | n/a | PSTOP |
| 1C03h | Motor Stop | n/a | MSTOP |
| 1C04h | Abort Motion and Sequence Execution | n/a | ABORT |
| 1C05h | Run Sequence (Sequence number is set in DATA.) | 0 to 99 | RUN |
| 1C10h | Start Incremental Motion, Distance DIS | n/a | MI |
| 1C11h | Start Absolute Motion to the Specified Destination | 1 to 100 | MA [POSx] |
| 1C12h | Move Continuously Positive | n/a | MCP |
| 1C13h | Move Continuously Negative | n/a | MCN |
| 1C14h | Move Go Home Positive | n/a | MGHP |
| 1C15h | Move Go Home Negative | n/a | MGHN |
| 1C16h | Start Linked Motion at Link Segment 'x' | 0 to 3 | MI[x] |
| 1C17h | Start Return-to-electrical Home Operation | n/a | EHOME |
| 1C18h | Pause Motion | n/a | PAUSE |
| 1C19h | Continue Motion | n/a | CONT |
| 1C1Ah | Pause Clear | n/a | PAUSECLR |
| 1C1Bh | Change Velocity (Change velocity is set in DATA.) | 0.001 to MAXVEL [UU/sec] | CV |
| 1C1Ch | Position Error Clear | - | PECLR |
| 1C1Dh | Reading Driver Current Position | - | ABSREQ |
| 1C1Eh | Reading Driver Current Position/Updating Internal Position | - | ABSREQPC |
| 1C1Fh | Torque Limiting/Push-motion Operation/Current Cutback Release | 0, 1 | TL |

n/a: Not Applicable.

## ■ Monitor Commands

| Command Code | Description | Range | Command |
|---|---|---|---|
| 2040h | Alarm Status | 0 to 255 | ALM |
| 2063h | Position Command | -MAXPOS to +MAXPOS [UU] | PC |
| 2064h | Velocity Command | -MAXVEL to +MAXVEL [UU/sec] | VC |
| 2066h | Feedback Position | -MAXPOS to +MAXPOS [UU] | PF |
| 206Ah | General Input Status | 0 to 511 | IN |
| 206Bh | Driver Current Position | -2,147,483.648 to +2,147,483.647 [UU] | PABS |
| 206Ch | Driver Status Code/Driver Alarm Code | (driver-dependent) | ABSSTS |
| 2080h | Position Error | -MAXPOS to +MAXPOS [UU] | PE |
| 2081h | Encoder Count | -MAXEC to +MAXEC | EC |
| 2091h | System Signal Input Status | 0 to 2,096,127 | INSG |
| 2092h | General Output Status | 0 to 15 | OUT |
| 2093h | System Signal Output Status | 0 to 1,983 | OUTSG |
| 20A0h | Remote General Input Status | 0 to 255 | RIN |
| 20A2h | Remote General Output Status | 0 to 63 | ROUT |
| 20B0h | Driver General Input Status | 0 to 127 | DIN |
| 20B1h | Driver System Signal Input Status | 0 to 127 | DINSG |
| 20B2h | Driver General Output Status | 0 to 255 | DOUT |
| 20B3h | Driver System Signal Output Status | 1 to 16,382 | DOUTSG |

### ■ Maintenance Command

| Command Code | Description | Range | Command |
|---|---|---|---|
| 30C0h | Alarm Clear | 0: Not Reset<br>1: Reset | ALMCLR |
| 30C5h | Reset Home Position | - | PRESET |
| 30CAh | Enable Driver Operation after Absolute Position Loss Alarm Release | - | ABSPLSEN |

| Note | The minimum output frequency on the **SCX10** is 1 Hz. If the running velocity in a user unit is set equivalent to less than 1 Hz, the actual pulse output frequency becomes 1 Hz. |
|---|---|

# 10.7 Object Dictionary (SDO)

| Index (hex) | Sub Index (Hex) | Name | Data Type | Access | Factory Setting | Comment |
|---|---|---|---|---|---|---|
| 1000 | 00 | Device Type | UNSIGNED32 | R | 00000000h | |
| 1001 | 00 | Error Register | UNSIGNED8 | R | - | generic error (bit 0) only |
| 1008 | 00 | Manufacturer Device Name | Visible String | R | **CM10** **SCX10** | |
| 1009 | 00 | Manufacturer Hardware Version | Visible String | R | 1 | |
| 100A | 00 | Manufacturer Software Version | Visible String | R | 2 | |
| 100C | 00 | Guard Time | UNSIGNED16 | R W | 0 | msec |
| 100D | 00 | Life Time Factor | UNSIGNED8 | R W | 0 | |
| 1010 | | Store Parameters | | | | Signature "save" |
| | 00 | Largest Subindex Supported | UNSIGNED8 | R | 2 | |
| | 01 | Save all Parameters | UNSIGNED32 | R W | - | |
| | 02 | Save Communication Parameters | UNSIGNED32 | R W | - | |
| 1011 | | Restore Default Parameters | UNSIGNED32 | | - | Signature "load" |
| | 00 | Largest Subindex Supported | UNSIGNED8 | R | 2 | |
| | 01 | Restore all Default Parameters | UNSIGNED32 | R W | - | |
| | 02 | Restore Communication Default Parameters | UNSIGNED32 | R W | - | |
| 1014 | 00 | COB-ID Emergency Object | UNSIGNED32 | R | Node-ID+0 0000080h | |
| 1017 | 00 | Producer Heartbeat Time | UNSIGNED16 | R W | 0 | |
| 1018 | | Identity Object | | | | |
| | 00 | Number of Entries | UNSIGNED8 | R | 2 | |
| | 01 | Vendor ID | UNSIGNED32 | R | 000002BEh | Oriental Motor (Europa) GmbH |
| | 02 | Product Code | UNSIGNED32 | R | | **CM10**: 5001 **SCX10**: 5002 |
| 1400 | | Receive PDO Communication Parameter | | | | |
| | 00 | Largest Sub-index Supported | UNSIGNED8 | R | 2 | |
| | 01 | COB-ID Used by PDO | UNSIGNED32 | R | Node-ID+200h | |
| | 02 | Transmission type | UNSIGNED8 | R | 254 | Manufacturer specific (immediately) |
| 1600 | | Receive PDO Mapping Parameter | | | - | |
| | 00 | Number of Mapped Application Objects in PDO | UNSIGNED8 | R | 3 | |
| | 01 | PDO Mapping for the nth Application to be Mapped | UNSIGNED32 | R | 2E000110h | INP (Index=2E00h, Sub Index= 01h, 16bit) |
| | 02 | PDO Mapping for the nth Application to be Mapped | UNSIGNED32 | R | 2E000210h | AID (Index=2E00h, Sub Index= 02h, 16bit) |
| | 03 | PDO Mapping for the nth Application to be Mapped | UNSIGNED32 | R | 2E000320h | DATA (Index=2E00h, Sub Index =03h, 32bit) |
| 1800 | | Transmit PDO Communication Parameter | | | | |
| | 00 | Largest Sub-index Supported | UNSIGNED8 | R | 5 | |
| | 01 | COB-ID Used by PDO | UNSIGNED32 | R W | Node-ID+180h | Bit 31 is 1, if inhibit time and/or event timer is changed. |
| | 02 | Transmission Type | UNSIGNED8 | R | 254 | Manufacturer specific (Event: COS, Time over) |
| | 03 | Inhibit Time | UNSIGNED16 | R W | 0 | The value is defined as multiple of 100 µsec. |
| | 05 | Event Timer | UNSIGNED16 | R W | 0 | The event timer elapses as multiple of 1 msec. |

| Index (hex) | Sub Index (Hex) | Name | Data Type | Access | Factory Setting | Comment |
|---|---|---|---|---|---|---|
| 1A00 | | Transmit PDO Mapping Parameter | | | - | |
| | 00 | Number of Mapped Application Objects in PDO | UNSIGNED8 | R | 3 | |
| | 01 | PDO Mapping for the nth Application to be Mapped | UNSIGNED32 | R | 2E100110h | OUTP (Index=E10h, Sub Index=01h, 16bit) |
| | 02 | PDO Mapping for the nth Application to be Mapped | UNSIGNED32 | R | 2E100210h | ID_R (Index=2E10h, Sub Index=02h, 16bit) |
| | 03 | PDO Mapping for the nth Application to be Mapped | UNSIGNED32 | R | 2E100320h | DATA_R (Index=2E10h, Sub Index=03h, 32bit) |
| 2040 | 00 | Alarm Status | UNSIGNED8 | R | 0 | ALM |
| 2063 | 00 | Position Command | SIGNED32 | RW | - | PC (-MAXPOS to +MAXPOS) UU |
| 2064 | 00 | Velocity Command | SIGNED32 | R | - | VC (-MAXVEL to +MAXVEL) UU/sec |
| 2066 | 00 | Feedback Position | SIGNED32 | RW | - | PF (-MAXPOS to +MAXPOS) UU |
| 206A | 00 | General Input Status | UNSIGNED32 | R | | IN (0 to 511) |
| 206B | 00 | Driver Current Position | SIGNED32 | R | - | PABS (-2,147,483.648 to 2,147,483.647) |
| 206C | 00 | Driver Status Code/Driver Alarm Code | UNSIGNED16 | R | - | ABSSTS |
| 2080 | 00 | Position Error | SIGNED32 | R | - | PE (-MAXPOS to +MAXPOS) UU |
| 2081 | 00 | Encoder Count | SIGNED32 | RW | - | EC (-MAXEC to +MAXEC) |
| 2091 | 00 | System Signal Input Status | UNSIGNED32 | R | - | INSG (0 to 2,096,127) |
| 2092 | 00 | General Output Status | UNSIGNED32 | R | - | OUT (0 to 15) |
| 2093 | 00 | System Signal Output Status | UNSIGNED32 | R | - | OUTSG (0 to 1,983) |
| 20A0 | 00 | Remote General Input Status | UNSIGNED8 | R | - | RIN (0 to 255) |
| 20A2 | 00 | Remote General Output Status | UNSIGNED8 | R | - | ROUT (0 to 63) |
| 20B0 | 00 | Driver General Input Status | UNSIGNED8 | R | - | DIN (0 to 127) |
| 20B1 | 00 | Driver System Signal Input Status | UNSIGNED32 | R | - | DINSG (0 to 127) |
| 20B2 | 00 | Driver General Output Status | UNSIGNED8 | R | - | DOUT (0 to 255) |
| 20B3 | 00 | Driver System Signal Output Status | UNSIGNED32 | R | - | DOUTSG (1 to 16,382) |
| 20C0 | 00 | Alarm Clear | UNSIGNED8 | W | - | ALMCLR |
| 2140 | 00 | Acceleration Time | UNSIGNED32 | RW | 500 | TA (1 to 500,000) msec |
| 2141 | 00 | Deceleration Time | UNSIGNED32 | RW | 500 | TD (1 to 500,000) msec |
| 2142 | 00 | Starting Velocity | UNSIGNED32 | RW | 100 | VS (0 to MAXVEL) UU/sec |
| 2160 | 00 | Mechanical Home Seeking Mode | UNSIGNED8 | RW | 0 | HOMETYP (0 to 12) |
| 2164 | 00 | Offset for Mechanical Home Seeking | SIGNED32 | RW | 0 | OFFSET (-MAXPOS to +MAXPOS) UU |
| 2380 | 00 | Driver Operation Data | UNSIGNED8 | RW | 0 | DD (0 to 7) |
| 2400 | 00 | Distance for Incremental Motion | SIGNED32 | RW | 0 | DIS (-MAXPOS to +MAXPOS) UU |
| 2401 | | Absolute Position | | | | |
| | 00 | Number of Entries | UNSIGNED8 | R | 100 | |
| | 01 | Position Array Data No.1 | SIGNED32 | RW | 0 | POS[1] (-MAXPOS to +MAXPOS) UU |
| | : | | SIGNED32 | RW | 0 | : |
| | 64 | Position Array Data No.100 | SIGNED32 | RW | 0 | POS[100] (-MAXPOS to +MAXPOS) UU |
| 2480 | 00 | Running Velocity | UNSIGNED32 | RW | 1000 | VR (1 to MAXVEL) UU/sec |
| 24A0 | 00 | Distance from SENSOR Input to the Stop Position | UNSIGNED32 | RW | 0 | SCHGPOS (0 to MAXPOS) UU |
| 24A1 | 00 | Velocity after SENSOR Input | UNSIGNED32 | RW | 1000 | SCHGVR (1 to MAXVEL) UU/sec |

| Index (hex) | Sub Index (Hex) | Name | Data Type | Access | Factory Setting | Comment |
|---|---|---|---|---|---|---|
| 2500 | | Linked Motion Distance or Destination | SIGNED32 | RW | | |
| | 00 | Number of Entries | UNSIGNED8 | R | 4 | |
| | 01 | Distance or Destination for Link Segment '0' | SIGNED32 | RW | 0 | DIS0 (-MAXPOS to +MAXPOS) UU |
| | 02 | Distance or Destination for Link Segment '1' | SIGNED32 | RW | 0 | DIS1 (-MAXPOS to +MAXPOS) UU |
| | 03 | Distance or Destination for Link Segment '2' | SIGNED32 | RW | 0 | DIS2 (-MAXPOS to +MAXPOS) UU |
| | 04 | Distance or Destination for Link Segment '3' | SIGNED32 | RW | 0 | DIS3 (-MAXPOS to +MAXPOS) UU |
| 2510 | | Linked Motion Running Velocity | | | | |
| | 00 | Number of Entries | UNSIGNED8 | R | 4 | |
| | 01 | Running Velocity of Link Segment '0' | SIGNED32 | RW | 1000 | VR0 (1 to MAXVEL) UU/sec |
| | 02 | Running Velocity of Link Segment '1' | SIGNED32 | RW | 1000 | VR1 (1 to MAXVEL) UU/sec |
| | 03 | Running Velocity of Link Segment '2' | SIGNED32 | RW | 1000 | VR2 (1 to MAXVEL) UU/sec |
| | 04 | Running Velocity of Link Segment '3' | SIGNED32 | RW | 1000 | VR3 (1 to MAXVEL) UU/sec |
| 2520 | | Linked Move Type | | | | |
| | 00 | Number of Entries | UNSIGNED8 | R | 4 | |
| | 01 | Link Type for Link Segment '0' | BOOLEAN | RW | 1 | INCABS0 (0:Absolute, 1:Incremental) |
| | 02 | Link Type for Link Segment '1' | BOOLEAN | RW | 1 | INCABS1 (0:Absolute, 1:Incremental) |
| | 03 | Link Type for Link Segment '2' | BOOLEAN | RW | 1 | INCABS2 (0:Absolute, 1:Incremental) |
| | 04 | Link Type for Link Segment '3' | BOOLEAN | RW | 1 | INCABS3 (0:Absolute, 1:Incremental) |
| 2530 | | Link Control | | | | |
| | 00 | Number of Entries | UNSIGNED8 | R | 3 | |
| | 01 | Link Control for Link Segment '0' | BOOLEAN | RW | 0 | LINK0 (0:No Link, 1:Link-to-next) |
| | 02 | Link Control for Link Segment '1' | BOOLEAN | RW | 0 | LINK1 (0:No Link, 1:Link-to-next) |
| | 03 | Link Control for Link Segment '2' | BOOLEAN | RW | 0 | LINK2 (0:No Link, 1:Link-to-next) |
| 2900 | 00 | Running Timer | UNSIGNED32 | RW | - | TIMER (0 to 500,000,000) msec |
| 2E00 | | Command | | | | |
| | 00 | Number of Entries | UNSIGNED8 | R | 3 | |
| | 01 | INP | UNSIGNED16 | RW | - | CANINP |
| | 02 | CANID | UNSIGNED16 | RW | - | CANID |
| | 03 | DATA | UNSIGNED32 | RW | - | CANDATA |
| 2E10 | | Response | | | | |
| | 00 | Number of Entries | UNSIGNED8 | R | 3 | |
| | 01 | OUTP | UNSIGNED16 | R | - | CANOUTP |
| | 02 | CANID_R | UNSIGNED16 | R | - | CANID_R |
| | 03 | DATA_R | UNSIGNED32 | R | - | CANDATA_R |

# 11 Timing Charts

## ■ Power Input



| DC24 V input |
| OUTx output — 1 s max. |
| CON output (Driver connector) — 200 ms max. |
| MBFREE output (I/O and driver connector) — 200 ms max. |
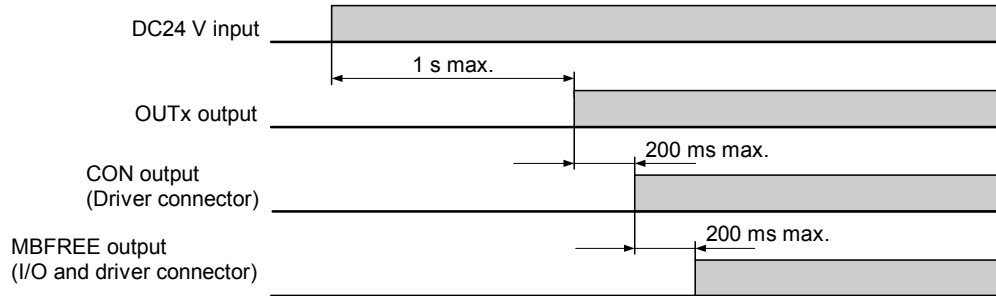
## ■ Driver Alarm Clear and Deviation Counter Clear



ALMCLR input
PECLR input
(I/O connector)

ALMCLR : 110 ms±5 ms
PECLR : 0.5 ms min.

2 ms max.

ACL/DCL output
(Driver connector)

## ■ General Output



1 ms min.

OUTx (x=1-4) output condition has occrred

5 ms max.          5 ms max.
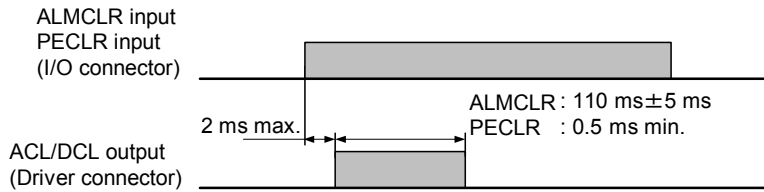
OUTx output

## ■ Selection and Execution of a Sequence



2 ms min.

START input

Sequence selection | General input | Sequence selection

IN1-IN9 input

2 ms max.

RUN output
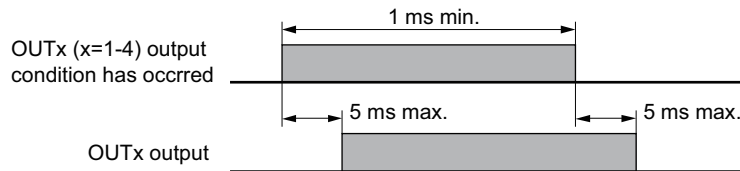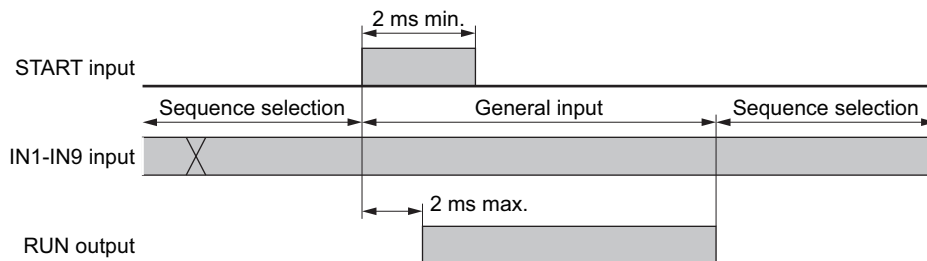
## ■ Execution and Stopping a Sequence (START, ABORT, RUN, MOVE)
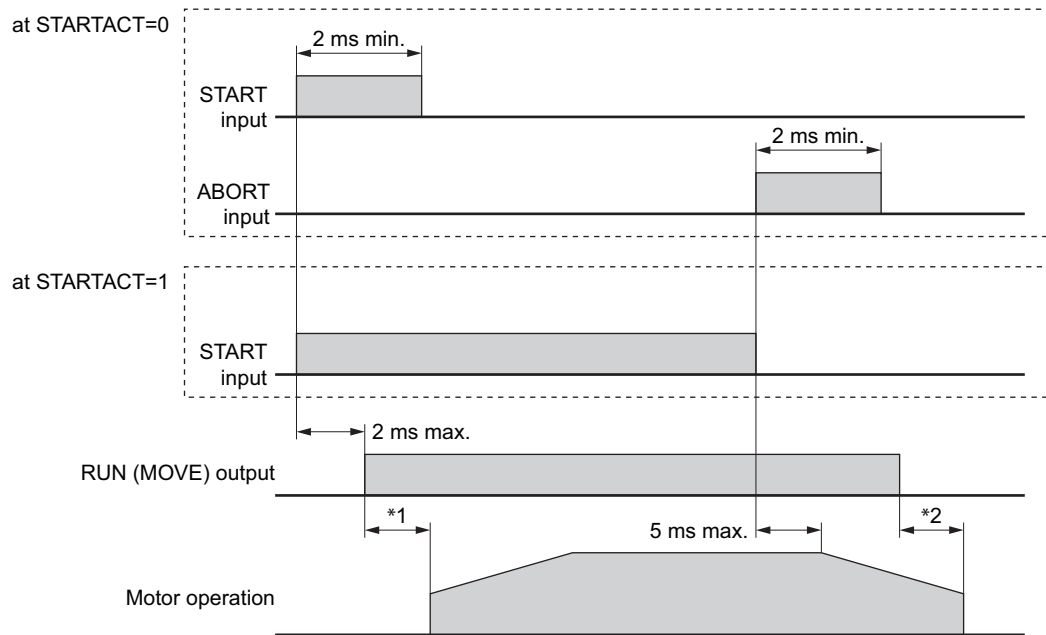


*1 Depend on the program.
*2 Depend on the load condition and settling time at stop.

## ■ Pausing Index Operation (PAUSE, PSTS)
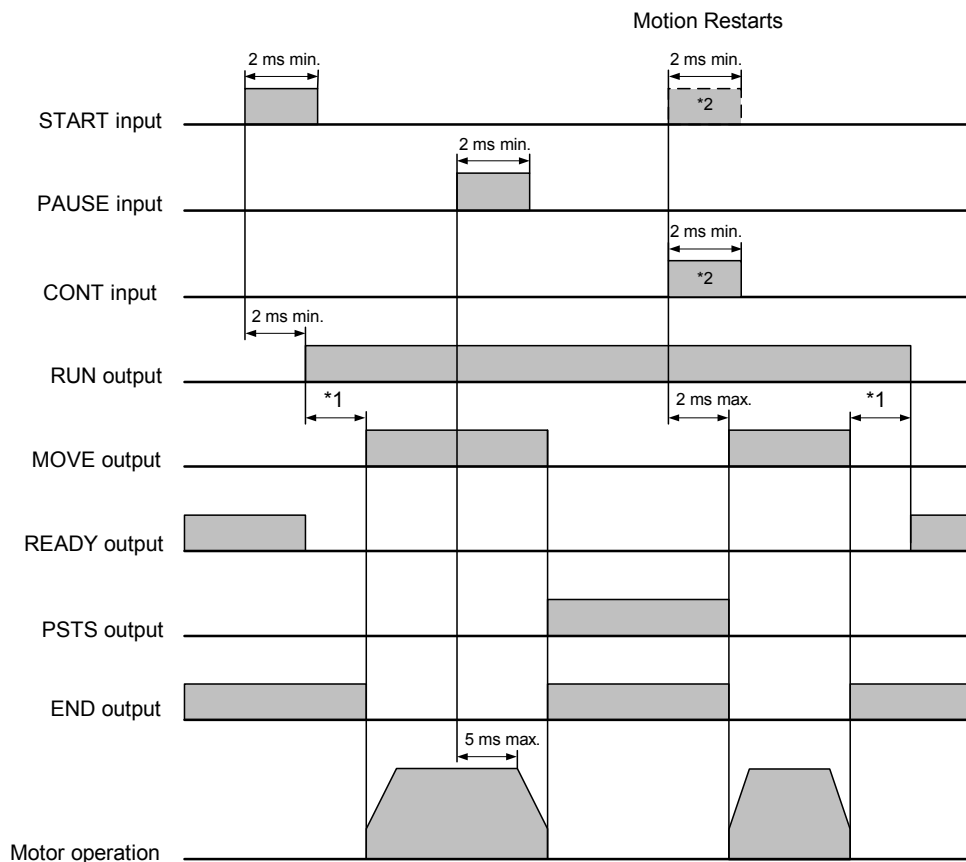


*1 Depend on the program.
*2 CONT input always resumes the paused motion while START input resumes the paused motion only when sequence is running and STARTACT is set to 0. Either one of them is used.

## ■ When the PSTOP Input is Turned ON
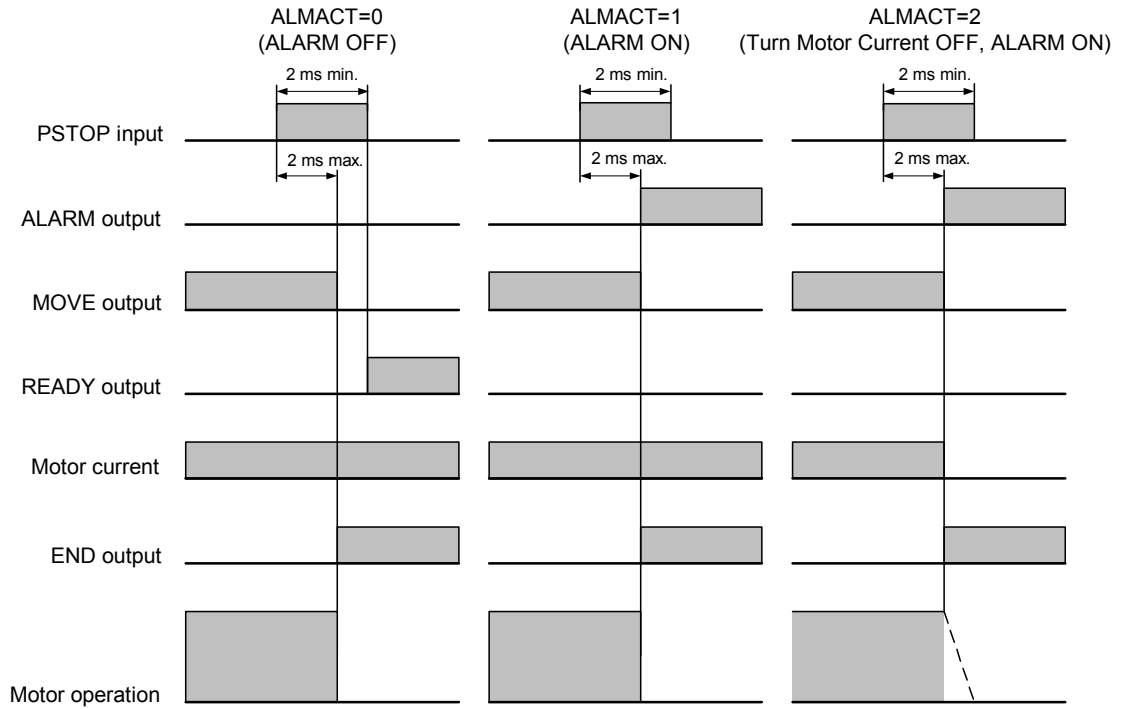


## ■ When the MSTOP Input is Turned ON
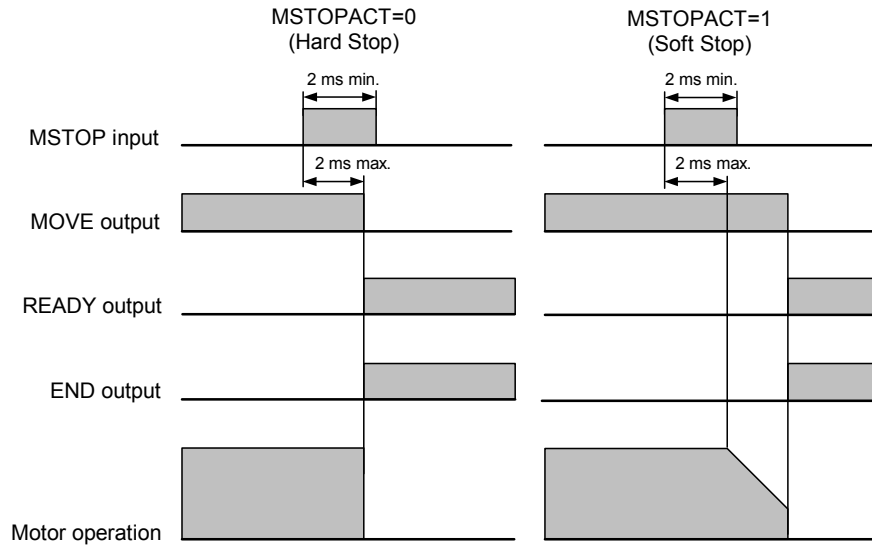
## ■ When the (+LS, −LS) Input is Used
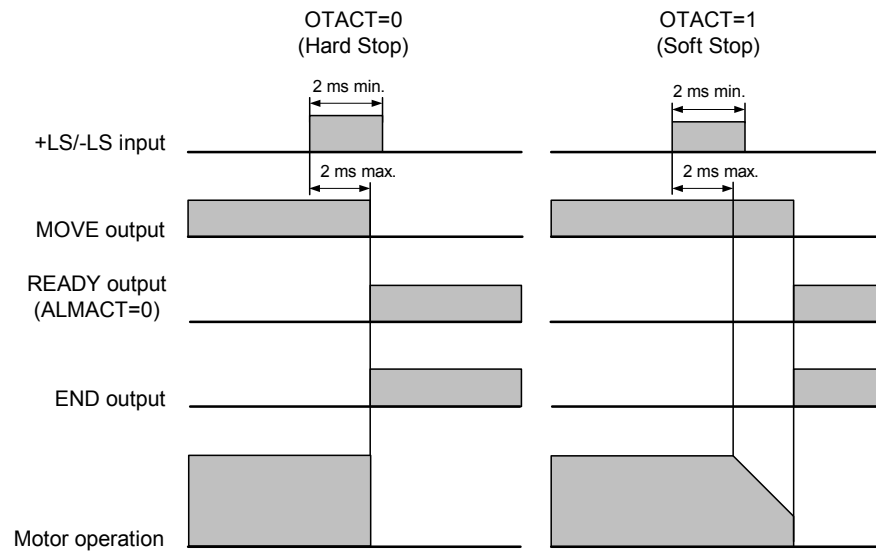


## ■ When the SENSOR Input is Used
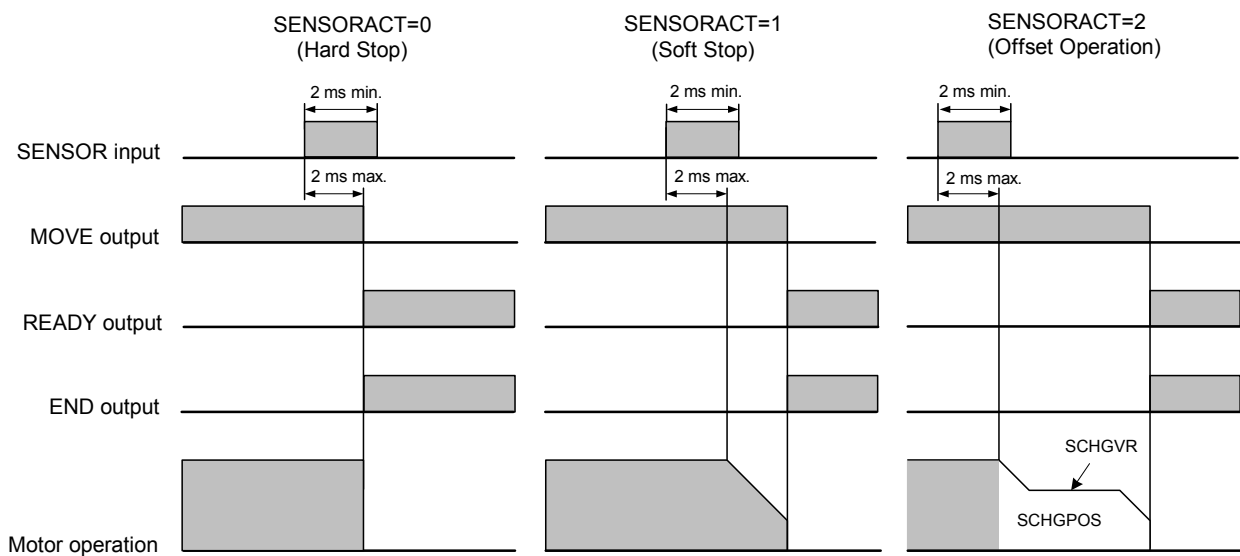


If SENSORACT=3, even when the SENSOR input is detected, the motor does not stop.  Set SENSORACT=3 when using the SENSOR input only in the return-to-mechanical home operation.

## ■ When an ALARM is Occurred

| | ALMACT=2<br>(At Current OFF Alarm [Free Run Stop]) | ALMACT=1<br>(At Current ON Alarm) |
|---|---|---|
| ALARM Occur | | |
| | 5 ms max. | 5 ms max. |
| ALARM output | | |
| MOVE output | | |
| READY output | | |
| Motor current | | |
| END output | | |
| Motor operation | | |
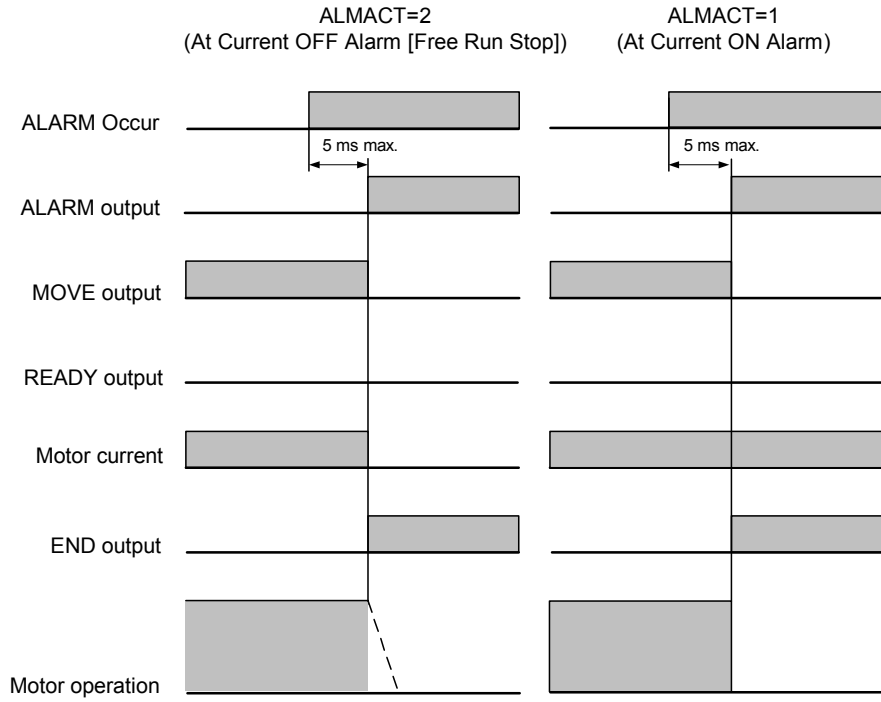
## ■ Operation for Motor Excitation and Electromagnetic Brake
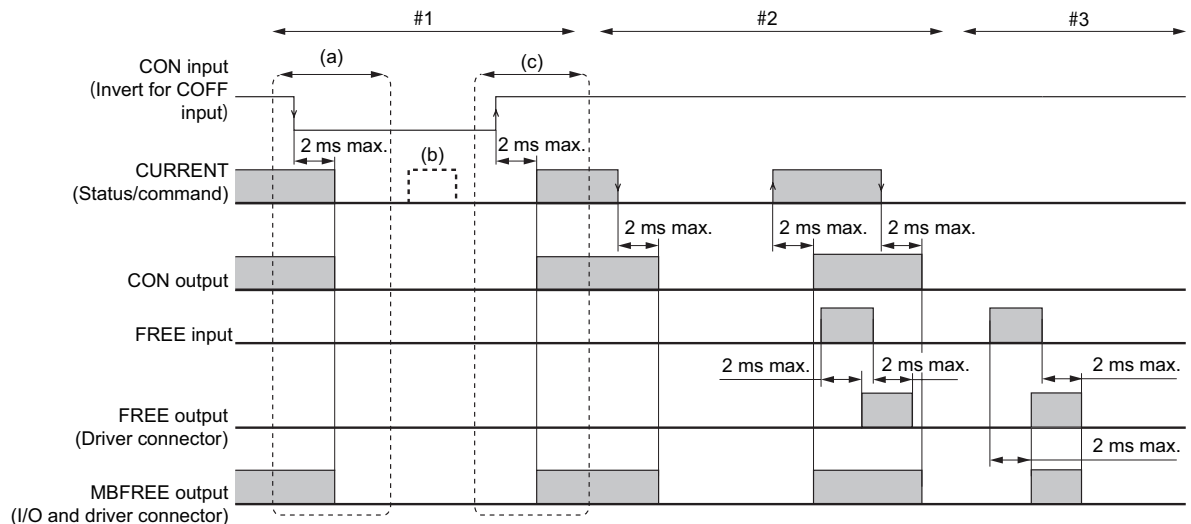


Operation using the CON input and CURRENT command

### #1 Operation using the CON input

(a) The falling edge of the CON input (the leading edge of the COFF input) is detected in excitation state.
  CURRENT=1→0, the CON output is OFF, the MBFREE output is OFF (magnetic brake lock)

(b) When the CON input is OFF (the COFF input is ON)
  Writing to the CURRENT command is invalid. (read only)

(c) The leading edge of the CON input (the falling edge of the COFF input) is detected in non-excitation state.
  CURRENT=0→1, the CON output is ON, the MBFREE output is ON (magnetic brake free)

### #2 Operation using the CURRENT command when the CON input is ON (the COFF input is OFF)

CURRENT=1→0, the CON output is OFF, the MBFREE output is OFF (magnetic brake lock)
CURRENT=0→1, the CON output is ON, the MBFREE output is ON (magnetic brake free)

Operation using the FREE input and CURRENT command

### #2 Operation using the FREE input at CURRNET=1

When the leading edge of the FREE input is detected, the FREE output is ON (non-excitation state)
When the falling edge of the FREE input is detected, the FREE output is OFF, the MBFREE output is ON (magnetic brake free)

### #3 Operation using the FREE input at CURRNET=0

When the leading edge of the FREE input is detected, the FREE output is ON, the MBFREE output is ON (magnetic brake free)
When the falling edge of the FREE input is detected, the FREE output is OFF, the MBFREE output is OFF (magnetic brake lock)

## ■ **Mechanical Home Seeking**

(Except "HOMETYP=12")

- When the Mechanical Home Seeking Operation is Completed without Passing the HOME Input.
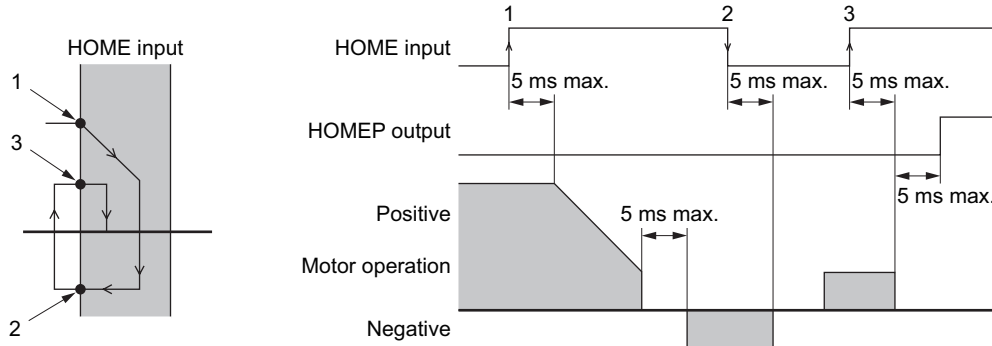


- When the Mechanical Home Seeking Operation is Completed with Passing the HOME Input Once.
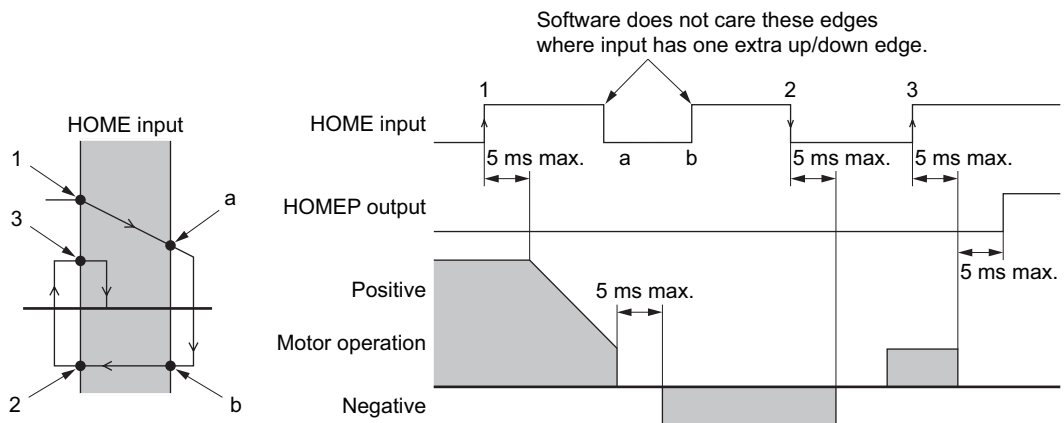


- Stopping Operation

Using HOME and SENSOR
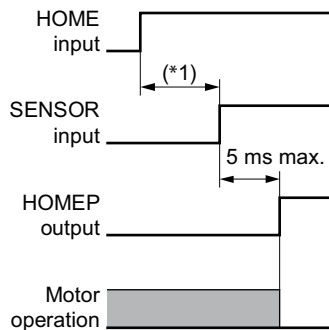
Using HOME and TIM signal
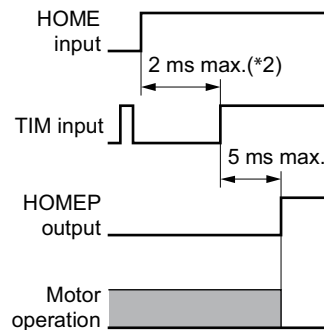
Using HOME, SENSOR and TIM signal
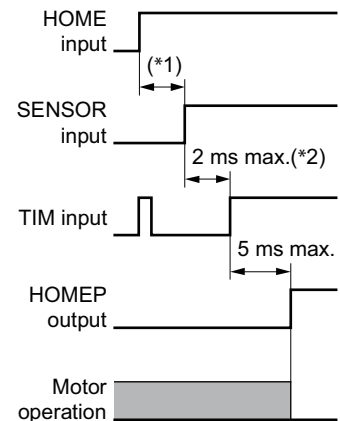


*1 Depends on the HOME and SENSOR positions.
*2 Depends on the SENSOR, rotor position and VS.

## ■  Teaching Operation

- When Holding the Key Down

  $\boxed{\text{M}}$  : CW scan input key on query keyboard.

Motor operation

RS-232C/USB input

500 ms max.          150 ms

Motor operation

RS-232C/USB input

500 ms          150 ms
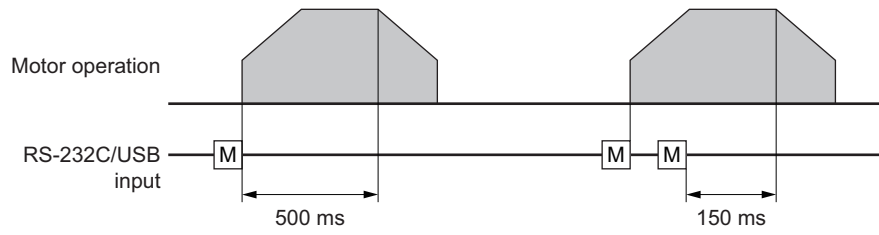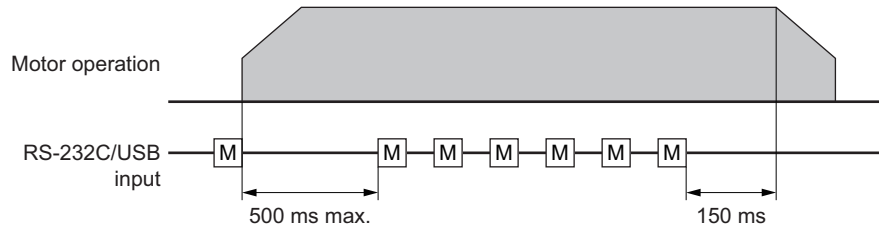
- To Stop Motion Immediately

Motor operation

RS-232C/USB input

SP: Space bar
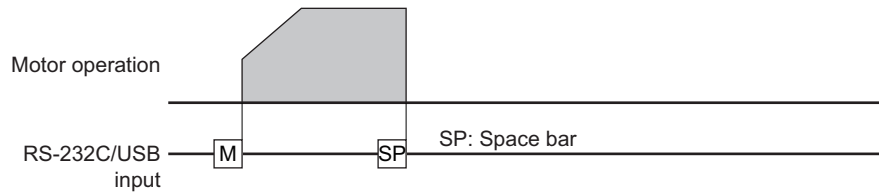
# 12 Command Reference

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 of Contents

## 12.1 Command List

### ■ Table Keys

| | | |
|---|---|---|
| n/a | not applicable | |
| MAXVEL | Maximum permissible velocity value, in user units per second. MAXVEL can be queried directly, see MAXVEL for details. | |
| MAXPOS | Maximum permissible position or distance value, in user units. MAXPOS can be queried directly, see MAXPOS for details. | |
| Max. Number | Maximum permitted numeric value. Max. Number depends on precision: higher precision requires lower numeric range.<br>Max. Number = 500000000    500 million, no decimal places<br>         50000000.0    50 million, one decimal place<br>         5000000.00    5 million, two decimal places<br>         500000.000    500 thousand, three decimal places | |
| Continuous Display | yes :<br><br><br><br><br>- : | A forward slash following certain variables causes the system to continuously display the value of those elements. (Example: If "PF /" is entered, the feedback position displays continuously.) See "■Continuous Display" on page 112.<br>/ parameter cannot be used. |
| SAVE & RESET REQUIRED | n/a :<br>S :<br><br><br>R :<br>SR : | For parameters, new value becomes active immediately.<br>New value becomes active immediately, but save command required for new value to be active after reset or power cycle.<br>Reset or power cycle required before new value becomes active.<br>Save and Reset (or save and power cycle) required before new value becomes active. |
| Immediate? | yes :<br>read :<br>- : | Command or parameter can be used in immediately.<br>Command or parameter can be used in immediately. (Read only)<br>Command or parameter cannot be used in immediately. |
| IN sequences? | yes :<br>read :<br>- : | Command or parameter can be used within sequences.<br>Command or parameter can be used within sequences. (Read only)<br>Command or parameter cannot be used within sequences. |
| CANopen? | yes :<br>read :<br>- : | Command or parameter can be used via CANopen.<br>Command or parameter can be used via CANopen. (Read only)<br>Command or parameter cannot be used via CANopen. |
| Commands not Allowed | MOVE<br>RUN<br>MVRO | Command is not accepted while the motor is moving.<br>Command is not accepted while the sequence is running.<br>Command is not accepted while the motor is moving. (Read only) |
| h | (after a value) Value is shown in hexadecimal notation | |
| UU | User units. Value shown in units determined by the user. See "7.3 Setting the User Unit" on page 70 for more details. | |

Memo ⋮ See "7.6 Command Format" on page 75 for verifying rules when commanding.

## ■ Motion Commands

| COMMAND | DESCRIPTION | FACTORY SETTING | RANGE | CONTINUOUS DISPLAY | SAVE & RESET REQUIRED | Immediate? | IN sequences? | CANopen? | Commands not Allowed | PAGE |
|---|---|---|---|---|---|---|---|---|---|---|
| CONT | Continue Motion | n/a | n/a | - | n/a | yes | yes | yes | - | 200 |
| CV | Change Velocity | n/a | 0.001 to MAXVEL | - | n/a | yes | yes | yes | - | 205 |
| EHOME | Start Return-to-electrical Home Operation | n/a | n/a | - | n/a | yes | yes | yes | MOVE | 235 |
| HSTOP | Hard Stop | n/a | n/a | - | n/a | yes | yes | yes | - | 252 |
| MA | Start Absolute Motion to the Specified Destination | n/a | -MAXPOS to +MAXPOS | - | n/a | yes | yes | yes | MOVE | 276 |
| MCP, MCN | Move Continuously Positive, Move Continuously Negative | n/a | n/a | - | n/a | yes | yes | yes | - | 282 |
| MGHP, MGHN | Move Go Home Positive, Move Go Home Negative | n/a | n/a | - | n/a | yes | yes | yes | MOVE | 285 |
| MI | Start Incremental Motion, Distance DIS | n/a | n/a | - | n/a | yes | yes | yes | MOVE | 287 |
| MIx (x=0 to 3) | Start Linked Motion at Link Segment 'x' | n/a | n/a | - | n/a | yes | yes | yes | MOVE | 288 |
| MSTOP | Motor Stop | n/a | n/a | - | n/a | yes | yes | yes | - | 291 |
| PAUSE | Pause Motion | n/a | n/a | - | n/a | yes | yes | yes | - | 302 |
| PAUSECLR | Pause Clear | n/a | n/a | - | n/a | yes | yes | yes | - | 303 |
| PSTOP | Panic Stop | n/a | n/a | - | n/a | yes | yes | yes | - | 313 |
| SSTOP | Soft Stop | n/a | n/a | - | n/a | yes | yes | yes | - | 340 |

## ■ Motion Variables

| COMMAND | DESCRIPTION | FACTORY SETTING | RANGE | CONTINUOUS DISPLAY | SAVE & RESET REQUIRED | Immediate? | IN sequences? | CANopen? | Commands not Allowed | PAGE |
|---|---|---|---|---|---|---|---|---|---|---|
| DIS | Distance for Incremental Motion | 0 | -MAXPOS to +MAXPOS | - | S | yes | yes | yes | - | 220 |
| DISx (x=0 to 3) | Distance or Destination for Link Segment 'x' | 0 | -MAXPOS to +MAXPOS | - | S | yes | yes | yes | - | 221 |
| INCABSx (x=0 to 3) | Link Type for Link Segment 'x' | 1 | 0: Absolute 1: Incremental | - | S | yes | yes | yes | - | 256 |
| LINKx (x=0 to 2) | Link Control for Link Segment 'x' | 0 | 0: No Link 1: Link-to-Next | - | S | yes | yes | yes | - | 271 |
| OFFSET | Offset for Mechanical Home Seeking | 0 | -MAXPOS to +MAXPOS | - | S | yes | yes | yes | - | 294 |
| POS[x] (x=1 to 100) | Position Array Data | 0 | -MAXPOS to +MAXPOS | - | S | yes | yes | yes | - | 311 |
| SCHGPOS | Distance from SENSOR Input to the Stop Position | 0 | 0 to MAXPOS | - | S | yes | yes | yes | - | 333 |
| SCHGVR | Velocity after SENSOR Input | 1 | 0.001 to MAXVEL | - | S | yes | yes | yes | - | 334 |
| TA | Acceleration Time | 0.5 | 0.001 to 500.000 [s] | - | S | yes | yes | yes | - | 344 |
| TD | Deceleration Time | 0.5 | 0.001 to 500.000 [s] | - | S | yes | yes | yes | - | 346 |
| VR | Running Velocity | 1 | 0.001 to MAXVEL | - | S | yes | yes | yes | - | 359 |
| VRx (x = 0 to 3) | Running Velocity of Link Segment 'x' | 1 | 0.001 to MAXVEL | - | S | yes | yes | yes | - | 360 |
| VS | Starting Velocity | 0.1 | 0 to MAXVEL | - | S | yes | yes | yes | - | 361 |

# ■ System Control

| COMMAND | DESCRIPTION | FACTORY SETTING | RANGE | CONTINUOUS DISPLAY | SAVE & RESET REQUIRED | Immediate? | IN sequences? | CANopen? | Commands not Allowed | PAGE |
|---|---|---|---|---|---|---|---|---|---|---|
| ; | Statement Separator for Multi-statement | n/a | n/a | - | n/a | yes | yes | - | - | 174 |
| <ESC> | (Escape): Abort Operation(s) | n/a | n/a | - | n/a | yes | - | - | - | 176 |
| ABORT | ABORT Motion and Sequence Execution | n/a | n/a | - | n/a | yes | yes | yes | - | 178 |
| ABSPLSEN | Enable Driver Operation after Absolute Position Loss Alarm Release | n/a | n/a | - | n/a | yes | yes | yes | - | 179 |
| ABSREQ | Reading Driver Current Position | n/a | n/a | - | n/a | yes | yes | yes | - | 180 |
| ABSREQPC | Reading Driver Current Position/Updating Internal Position | n/a | n/a | - | n/a | yes | yes | yes | MOVE | 181 |
| ALMACT | ALARM Action | 2 | 0: Current On, Alarm Off<br>1: Current On, Alarm On<br>2: Current Off, Alarm On | - | SR | yes | - | - | - | 184 |
| ALMCLR | ALARM Clear | n/a | n/a | - | n/a | yes | - | yes | - | 185 |
| ALMMSG | ALARM Message Action | 0 | 0: No Messages<br>1: Messages, Alarms only<br>2: Messages, Alarms and Warnings | - | S | yes | - | - | - | 186 |
| ALMSET | Set User ALARM | n/a | n/a | - | n/a | yes | yes | - | - | 187 |
| CLEARALL | Return to Factory Condition | n/a | n/a | - | n/a | yes | - | - | MOVE RUN | 196 |
| CLEARPOS | Clear POS[x] Position Array Data | n/a | n/a | - | n/a | yes | - | - | MOVE RUN | 197 |
| CURRENT | Current On/Off | 0: **CM10-1**, **5**<br>1: **CM10-2**, **3**, **4**, **SCX10** | 0: Motor Current Off<br>1: Motor Current On | - | n/a | yes | yes | yes | - | 204 |
| DD | Driver Operation Data | 0 | 0 to 3: Torque Limiting<br>0 to 7: Push-motion Operation | - | S | yes | yes | yes | - | 207 |
| DIRINV | Direction Invert | 0 | 0: Positive Motion is Clockwise<br>1: Positive Motion is Counter-clockwise | - | SR | yes | read | - | - | 219 |
| DPR | Distance per Revolution | 1 | 0.500 to 51200.000 | - | SR | yes | read | - | - | 227 |
| ENC | Encoder Selection | 0: **CM10-2**, **SCX10**<br>1: **CM10-1**, **3**, **4**, **5** | 0: Not Used<br>1: Driver Encoder<br>2: External Encoder | - | SR | yes | read | - | - | 237 |
| ENDACT | System End Action | 0 | 0: End of Pulse Generation<br>0.001 to (+MAXPOS/2): END Area | - | SR | yes | read | - | - | 240 |
| ENDWAIT | END wait time | 6 | 0.1 to 40.0 | - | SR | yes | - | - | - | 243 |
| ER | Encoder Resolution | 100: **CM10-3**<br>200: **CM10-2**<br>1000: **CM10-1**, **4**, **5**, **SCX10** | 10 to 51200 | - | SR | yes | read | - | - | 244 |
| FREE | Current Off, Magnetic Brake Free | 0 | 0: Normal Condition<br>1: Motor Shaft Free | - | n/a | yes | yes | - | - | 246 |
| GA, GB | Electrical Gear Ratio | 1 | 1 to 100 | - | SR | yes | read | - | - | 247 |
| HOMEDCL | Select the Deviation Counter Clear during Mechanical Home Seeking Operation | 0: **CM10-1**, **2**, **3**, **4**, **SCX10**<br>1: **CM10-5** | 0: Clear **CM10/SCX10**<br>1: Clear **CM10/SCX10** and Driver<br>2: Not Clear | - | SR | yes | read | - | - | 250 |

| COMMAND | DESCRIPTION | FACORY SETTING | RANGE | CONTINUOUS DISPLAY | SAVE & RESET REQUIRED | Immediate? | IN sequences? | CANopen? | Commands not Allowed | PAGE |
|---|---|---|---|---|---|---|---|---|---|---|
| HOMETYP | Mechanical Home Seeking Mode | 0 | 0 to 11: **CM10-1**, **2**, **4**, **5**<br>0 to 12: **CM10-3**, **SCX10** | - | S | yes | yes | yes | MVRO | 251 |
| INITPRM | Initialize Parameters | n/a | n/a | - | R | yes | - | - | MOVE RUN | 260 |
| LIMP, LIMN | Setting of Software Position Limits (Positive Direction, Negative Direction) | 0 | -MAXPOS to +MAXPOS | - | SR | yes | read | - | - | 269 |
| MBFREEACT | Magnetic Brake Free Action | 0: **CM10-1**, **2**, **3**, **4**, **5**<br>1: **SCX10** | 0: Driver Alarm is Unrelated<br>1: MBFREE Outputs on Both the Driver Connector on the **CM10/SCX10** and the I/O Connector Become Active When a Driver Alarm is Active (Electromagnetic Brake is Locked) | - | SR | yes | - | - | - | 281 |
| MR | Motor Resolution | 100: **CM10-3**<br>200: **CM10-2**<br>1000: **CM10-1**, **4**, **5**, **SCX10** | 10 to 51200 | - | SR | yes | read | - | - | 290 |
| MSTOPACT | Motor Stop Action | 0 | 0: Hard Stop<br>1: Soft Stop | - | SR | yes | - | - | - | 292 |
| OTACT | Overtravel Action | 0 | 0: Hard Stop<br>1: Soft Stop | - | SR | yes | - | - | - | 295 |
| PECLR | Position Error Clear | n/a | n/a | - | n/a | yes | yes | yes | MOVE | 307 |
| PRESET | Reset Home Position | n/a | n/a | - | n/a | yes | yes | yes | MOVE | 312 |
| PULSE | Pulse Output Mode | 1 | 0: 2PULSE Mode<br>1: 1PULSE Mode | - | SR | yes | - | - | - | 314 |
| RESET | Reset Device | n/a | n/a | - | n/a | yes | - | - | - | 317 |
| SAVEALL | Save All Data | n/a | n/a | - | n/a | yes | - | - | MOVE RUN | 330 |
| SAVEPOS | Save Position Array Data | n/a | n/a | - | n/a | yes | - | - | MOVE RUN | 331 |
| SAVEPRM | Save Parameters | n/a | n/a | - | n/a | yes | - | - | MOVE RUN | 332 |
| SENSORACT | SENSOR Input Active | 2 | 0: Hard Stop<br>1: Soft Stop<br>2: Soft Stop at Fixed Distance from SENSOR Signal<br>3: No Action | - | SR | yes | - | - | - | 335 |
| SLACT | Software Position Limit Enable | 0 | 0: Disabled<br>1: Enabled after Homing | - | SR | yes | read | - | - | 338 |
| STARTACT | START Input Action | 0 | 0: Start Sequence When Set Active<br>1: Start Sequence When Set Active, Abort When Set Inactive | - | SR | yes | - | - | - | 341 |
| STRDCS | Driver Step Angle at System Start | 0 | 0: CS Output Off at System Start<br>1: CS Output On at System Start | - | SR | yes | read | - | - | 342 |
| STRSW | Current State at System Start | 0: **CM10-1**, **5**<br>1: **CM10-2**, **3**, **4**, **SCX10** | 0: Current Off at System Start<br>1: Current On at System Start | - | SR | yes | - | - | - | 343 |
| TL | Torque Limiting /Push-motion Operation /Current Cutback Release | 0 | 0: OFF<br>1: ON | - | n/a | yes | yes | yes | - | 350 |

| COMMAND | DESCRIPTION | FACTORY SETTING | RANGE | CONTINUOUS DISPLAY | SAVE & RESET REQUIRED | Immediate? | IN sequences? | CANopen? | Commands not Allowed | PAGE |
|---|---|---|---|---|---|---|---|---|---|---|
| UU | User Units | Rev: **CM10-1**, **2**, **4**, **5**, **SCX10** mm: **CM10-3** | ASCII Characters, 20 Characters Maximum, Except ";" and "@" | - | S | yes | - | - | - | 354 |

## ■ System Status

| COMMAND | DESCRIPTION | FACTORY SETTING | RANGE | CONTINUOUS DISPLAY | SAVE & RESET REQUIRED | Immediate? | IN sequences? | CANopen? | Commands not Allowed | PAGE |
|---|---|---|---|---|---|---|---|---|---|---|
| ABSSTS | Driver Status Code/Driver Alarm Code | n/a | n/a | - | n/a | read | read | read | - | 182 |
| EC | Encoder Count | 0 | -MAXEC to +MAXEC | yes | n/a | yes | yes | yes | MV RO | 232 |
| MAXEC | Maximum Encoder Count | 500000000: **CM10-1**, **4**, **5**, **SCX10** 100000000: **CM10-2** 50000000: **CM10-3** | - | - | n/a | read | - | - | - | 278 |
| MAXPOS | Maximum Position Value | 500000 | n/a | - | n/a | read | - | - | - | 279 |
| MAXVEL | Maximum Velocity Value | 1240: **CM10-1**, **4**, **5**, **SCX10** 6200: **CM10-2** 12400: **CM10-3** | n/a | - | n/a | read | - | - | - | 280 |
| PABS | Driver Current Position | n/a | -2,147,483.648 to 2,147,483.647 | - | n/a | read | read | read | - | 301 |
| PC | Position Command | 0 | -MAXPOS to +MAXPOS | yes | n/a | yes | yes | yes | MV RO | 304 |
| PCI | Incremental Position Command | n/a | -2*MAXPOS to +2*MAXPOS | yes | n/a | read | read | - | - | 305 |
| PE | Position Error | n/a | -MAXPOS to +MAXPOS | yes | n/a | read | read | read | - | 306 |
| PF | Feedback Position | 0 | -MAXPOS to +MAXPOS | yes | n/a | yes | yes | yes | MV RO | 308 |
| PFI | Incremental Feedback Position | n/a | -2*MAXPOS to +2*MAXPOS | yes | n/a | read | read | - | - | 309 |
| TIM | Select Timing Input Signal | 1 | 0: Use the TIMD/EXTZ Input 1: Use the TIMS Input | - | SR | yes | read | - | - | 348 |
| TIMER | Running Timer | n/a | 0.000 to 500000.000 | yes | n/a | yes | yes | yes | - | 349 |
| VC | Velocity Command | n/a | -MAXVEL to +MAXVEL | yes | n/a | read | read | read | - | 355 |

12 Command Reference

## ■ I/O

| COMMAND | DESCRIPTION | FACORY SETTING | RANGE | CONTINUOUS DISPLAY | SAVE & RESET REQUIRED | Immediate? | IN sequences? | CANopen? | Commands not Allowed | PAGE |
|---|---|---|---|---|---|---|---|---|---|---|
| DALARM | Driver ALARAM Signal Enable | 0: **SCX10** 1: **CM10-1**, **2**, **3**, **4**, **5** | 0: Not-use the DALARM Signal Input 1: Use the DALARM Signal Input | - | SR | yes | read | - | - | 206 |
| DEND | Driver END Signal Enable | 0: **CM10-2**, **SCX10** 1: **CM10-1**, **3**, **4**, **5** | 0: Internal End Area 1: Driver END Signal | - | SR | yes | read | - | - | 211 |
| DIN | Driver General Input Status | n/a | 0 to 127 | yes | n/a | read | read | read | - | 212 |
| DINSG | Driver System Signal Input Status | n/a | 0 to 127 | yes | n/a | read | read | read | - | 213 |
| DINx (X=1 to 7) | Individual Driver General Input Status | n/a | 0: Not Active 1: Active | yes | n/a | read | read | - | - | 214 |
| DINxxx | Driver System Signal Input Assignment | See Page 215 for DINxxx Command | See Page 215 for DINxxx Command | - | SR | yes | - | - | - | 215 |
| DIO | Driver I/O Status | n/a | 0: Not Active 1: Active | - | n/a | read | - | - | - | 217 |
| DOUT | Driver General Output Control | n/a | 0 to 255 | yes | n/a | yes | yes | read | - | 222 |
| DOUTSG | Driver System Signal Output Status | n/a | 1 to 16382 | yes | n/a | read | read | read | - | 223 |
| DOUTx (x=1 to 8) | Individual Driver General Output Control | 0 | 0: Not Active 1: Active | yes | n/a | yes | yes | - | - | 224 |
| DOUTxxx | Driver System Signal Output Assignment | See Page 225 for DOUTxxx Command | See Page 225 for DOUTxxx Command | - | SR | yes | - | - | - | 225 |
| DREADY | Driver READY Signal Enable | 0: **CM10-2**, **3**, **4**, **SCX10** 1: **CM10-1**, **5** | 0: Disable 1: Enable | - | SR | yes | read | - | - | 229 |
| DSIGxxx | Status for Driver System Input Signal/Driver System Output Signal | n/a | 0: Not Active 1: Active | yes | n/a | read | read | - | - | 230 |
| EVx (x=1, 2) | Configure Event Output | n/a | See Page 245 for EVx Command | - | n/a | yes | yes | - | - | 245 |
| IN | General Input Status | n/a | 0 to 511 | yes | n/a | read | read | read | - | 255 |
| INITDIO | Initialize Driver I/O | n/a | n/a | - | SR | yes | - | - | - | 257 |
| INITIO | Initialize I/O | n/a | n/a | - | SR | yes | - | - | - | 258 |
| INSG | System Signal Input Status | n/a | 0 to 2096127 | yes | n/a | read | read | read | - | 262 |
| INx (x=1 to 9) | Individual General Input Status | n/a | 0: Not Active 1: Active | yes | n/a | read | read | - | - | 263 |
| INxxx | System Signal Input Assignment | 0 | 0: Unassigned 1 to 9: Assigned | - | SR | yes | - | - | - | 264 |
| IO | Input/Output Status | n/a | n/a | yes | n/a | yes | - | - | - | 266 |
| OUT | General Output Status | 0 | 0 to 15 | yes | n/a | yes | yes | read | - | 296 |
| OUTSG | System Signal Output Status | n/a | 0 to 1983 | yes | n/a | read | read | read | - | 297 |
| OUTTEST | I/O Test Utility | n/a | n/a | - | n/a | yes | - | - | MOVE RUN | 298 |
| OUTx (x=1 to 4) | Individual General Output Control | 0 | 0: Not Active 1: Active | yes | n/a | yes | yes | - | - | 299 |
| OUTxxx | System Signal Output Assignment | 0 | 0: Unassigned 1 to 4: Assigned | - | SR | yes | - | - | - | 300 |
| PLSINV | Pulse Output Invert | 0 | 0: Positive Logic 1: Negative Logic | - | SR | yes | - | - | - | 310 |
| SIGxxx | Status for System Input Signal/System Output Signal | n/a | 0: Not Active 1: Active | yes | n/a | read | read | - | - | 336 |
| xxxLV | System Input Level/System Output Level | See Page 365 for xxxLV Command | See Page 365 for xxxLV Command | - | SR | yes | - | - | - | 365 |

- 165 -

## ■ Monitor Commands

| COMMAND | DESCRIPTION | FACORY SETTING | RANGE | CONTINUOUS DISPLAY | SAVE & RESET REQUIRED | Immediate? | IN sequences? | CANopen? | Commands not Allowed | PAGE |
|---|---|---|---|---|---|---|---|---|---|---|
| ALM | Alarm Status and History | n/a | n/a | - | n/a | yes | - | yes | - | 183 |
| HELP | Display Help Information | n/a | n/a | - | n/a | yes | - | - | - | 249 |
| REPORT | Display System Status | n/a | n/a | - | n/a | yes | - | - | - | 316 |
| TEACH | Teach Positions | n/a | n/a | - | n/a | yes | - | - | MOVE RUN | 347 |
| TRACE | Sequence Trace Control | 0 | 0: Trace is Disabled 1: Trace is Enabled | - | n/a | yes | - | - | - | 351 |
| VER | Display Firmware Version | n/a | n/a | - | n/a | yes | - | - | - | 356 |

## ■ Communications

| COMMAND | DESCRIPTION | FACORY SETTING | RANGE | CONTINUOUS DISPLAY | SAVE & RESET REQUIRED | Immediate? | IN sequences? | CANopen? | Commands not Allowed | PAGE |
|---|---|---|---|---|---|---|---|---|---|---|
| \ | Global Command | n/a | [command] | - | n/a | yes | - | - | - | 173 |
| @ | Select Device | n/a | *, 0 to 9, A to Z | - | n/a | yes | - | - | - | 175 |
| BAUD | RS-232C BAUD Rate | 0 | 0: 9600 bps 1: 19200 bps 2: 38400 bps 3: 57600 bps 4: 115200 bps | - | SR | yes | read | - | - | 190 |
| ECHO | Communications Echo Control | 1 | 0: Echo Off 1: Echo On | - | S | yes | - | - | - | 233 |
| ID | Device ID | * | *, 0 to 9, A to Z | - | S | yes | - | - | - | 253 |
| TALK | Select Device | n/a | *, 0 to 9, A to Z | - | n/a | yes | - | - | - | 345 |
| USBBAUD | USB BAUD Rate | 0 | 0: 9600 bps 1: 19200 bps 2: 38400 bps 3: 57600 bps 4: 115200 bps | - | SR | yes | read | - | - | 353 |
| VERBOSE | Command Response Control | 1 | 0: Respond with Data only 1: Respond with Data and Descriptive Text | - | S | yes | - | - | - | 357 |

## ■ Sequence Commands

| COMMAND | DESCRIPTION | FACORY SETTING | RANGE | CONTINUOUS DISPLAY | SAVE & RESET REQUIRED | Immediate? | IN sequences? | CANopen? | Commands not Allowed | PAGE |
|---|---|---|---|---|---|---|---|---|---|---|
| # | Sequence Comment | n/a | Alphanumeric Characters | - | n/a | - | yes | - | - | 171 |
| BREAKL | Break LOOP Block | n/a | n/a | - | n/a | - | yes | - | - | 191 |
| BREAKW | Break WHILE Block | n/a | n/a | - | n/a | - | yes | - | - | 192 |
| CALL | Call Sequence as Subroutine | n/a | Valid Sequence Name or Number, or Variable | - | n/a | - | yes | - | - | 193 |
| ELSE | Begin ELSE Block: execute if IF is false | n/a | n/a | - | n/a | - | yes | - | - | 236 |
| END | Motion End | n/a | n/a | - | n/a | - | yes | - | - | 239 |
| ENDIF | End of IF Block | n/a | n/a | - | n/a | - | yes | - | - | 241 |
| ENDL | End of LOOP Block | n/a | n/a | - | n/a | - | yes | - | - | 242 |
| IF | Begin IF Block: execute if IF is true | n/a | Conditional Expression | - | n/a | - | yes | - | - | 254 |
| KB | Keyboard Input | n/a | -Max.Number to +Max.Number | - | n/a | - | yes | - | - | 267 |
| KBQ | Keyboard Input (Quiet) | n/a | -Max.Number to +Max.Number | - | n/a | - | yes | - | - | 268 |
| LOOP | Begin Counted LOOP Block | n/a | 1 to Max.Number | - | n/a | - | yes | - | - | 275 |
| MEND | Wait for Motion End | n/a | n/a | - | n/a | - | yes | - | - | 284 |
| RET | Sequence Return | n/a | n/a | - | n/a | - | yes | - | - | 318 |
| SACS | Send ASCII Control String | n/a | String: A Series of ASCII Characters or Control Codes, Maximum 70 Characters, Except @ | - | n/a | - | yes | - | - | 328 |
| SAS | Send ASCII String | n/a | String: A Series of ASCII Characters, Maximum 70 Characters, Except @ | - | n/a | - | yes | - | - | 329 |
| VIEW | View Parameter | n/a | Valid Parameter or Variable Name | - | n/a | - | yes | - | - | 358 |
| WAIT | Wait for Specified Time | n/a | 0.0 to 500000.0 | - | n/a | - | yes | - | - | 362 |
| WEND | End of WHILE Block | n/a | n/a | - | n/a | - | yes | - | - | 363 |
| WHILE | Begin WHILE Block | n/a | Conditional Expression | - | n/a | - | yes | - | - | 364 |

## ■ Math/Logical/Conditional Operators (In Sequence only)

| COMMAND | DESCRIPTION | FACORY SETTING | RANGE | CONTINUOUS DISPLAY | SAVE & RESET REQUIRED | Immediate? | IN sequences? | CANopen? | Commands not Allowed | PAGE |
|---|---|---|---|---|---|---|---|---|---|---|
| +, -, *, /, % | Addition, Subtraction, Multiplication, Division, Modulo | n/a | n/a | - | n/a | - | yes | - | - | 172 |
| &, \|, ^, <<, >> | AND, OR, XOR, Left Arithmetic Shift, Right Arithmetic Shift | n/a | n/a | - | n/a | - | yes | - | - | 172 |
| a < b | a is smaller than b | n/a | n/a | - | n/a | - | yes | - | - | 177 |
| a <= b | a is equal to or smaller than b | n/a | n/a | - | n/a | - | yes | - | - | 177 |
| a = b, a = = b | a is equal to b | n/a | n/a | - | n/a | - | yes | - | - | 177 |
| a > b | a is greater than b | n/a | n/a | - | n/a | - | yes | - | - | 177 |
| a >= b | a is equal to or greater than b | n/a | n/a | - | n/a | - | yes | - | - | 177 |
| a! = b | a is not equal to b | n/a | n/a | - | n/a | - | yes | - | - | 177 |

## ■ User Variables

| COMMAND | DESCRIPTION | FACTORY SETTING | RANGE | CONTINUOUS DISPLAY | SAVE & RESET REQUIRED | Immediate? | IN sequences? | CANopen? | Commands not Allowed | PAGE |
|---|---|---|---|---|---|---|---|---|---|---|
| A to Z | User Variables | 0 | -Max. Number to +Max.Number | - | S | yes | yes | - | - | 188 |
| CLEARVAR | Clear User-defined Variables | n/a | n/a | - | S | yes | - | - | RUN | 199 |
| CREATEVAR | Create User-defined Variable | n/a | N_xxx [Numeric Type] or S_xxx [String Type] | - | S | yes | - | - | RUN | 202 |
| DELETEVAR | Delete User-defined Variable | n/a | N_xxx [Numeric Type] or S_xxx [String Type] | - | S | yes | - | - | RUN | 209 |
| LISTVAR | Lists all User-defined Variables | n/a | n/a | - | n/a | yes | - | - | - | 273 |
| N_xxx | User-defined Numeric Variables | 0 when created | -Max.Number to +Max.Number | - | S | yes | yes | - | - | 293 |
| S_xxx | User-defined String Variables | empty when created | Text String, 20 Characters Maximum | - | S | yes | yes | - | - | 327 |

## ■ Sequence Management

| COMMAND | DESCRIPTION | FACTORY SETTING | RANGE | CONTINUOUS DISPLAY | SAVE & RESET REQUIRED | Immediate? | IN sequences? | CANopen? | Commands not Allowed | PAGE |
|---|---|---|---|---|---|---|---|---|---|---|
| CLEARSEQ | Clear Sequences | n/a | n/a | - | n/a | yes | - | - | RUN | 198 |
| COPY | Copy Sequence | n/a | Valid Sequence Name or Number | - | n/a | yes | - | - | RUN | 201 |
| DEL | Delete Sequence | n/a | Valid Sequence Name or Number | - | n/a | yes | - | - | RUN | 208 |
| DIR | Sequence Directory | n/a | n/a or Valid Sequence Name or Number | - | n/a | yes | - | - | - | 218 |
| EDIT | Edit Sequence | n/a | Valid Sequence Name (Consisting of Letters, Numbers and Underscore, up to 10 Characters, the First Character is not N, S, n, s) or Number (0-99) | - | n/a | yes | - | - | RUN | 234 |
| LIST | List Sequence Contents | n/a | Valid Sequence Name or Number (Optional: Start and End Line Number) | - | n/a | yes | - | - | - | 272 |
| LOCK | Lock Sequence | n/a | Valid Sequence Name or Number | - | n/a | yes | - | - | RUN | 274 |
| REN | Rename Sequence | n/a | Valid Sequence Name or Number | - | n/a | yes | - | - | RUN | 315 |
| RUN | Sequence Running | n/a | Valid Sequence Name or Number | - | n/a | yes | - | yes | RUN | 326 |
| UNLOCK | Unlock Sequence | n/a | Valid Sequence Name or Number | - | n/a | yes | - | - | RUN | 352 |

■ **CANopen Setting**

| COMMAND | DESCRIPTION | FACTORY SETTING | RANGE | CONTINUOUS DISPLAY | SAVE & RESET REQUIRED | Immediate? | IN sequences? | CANopen? | Commands not Allowed | PAGE |
|---|---|---|---|---|---|---|---|---|---|---|
| CANBAUD | CANopen BAUD Rate | 1 | 0: 10 kbps<br>1: 20 kbps<br>2: 50 kbps<br>3: 125 kbps<br>4: 250 kbps<br>5: 500 kbps<br>6: 800 kbps<br>7: 1 Mbps | - | SR | yes | - | - | - | 194 |
| CANID | CANopen Node ID | 1 | 1 to 127 | - | SR | yes | - | - | - | 195 |
| INITRIO | Initialize Remote I/O | n/a | n/a | - | SR | yes | - | - | - | 261 |
| RIN | Remote General Input Status | n/a | 0 to 255 | yes | n/a | read | read | read | - | 319 |
| RINx<br>(x=1 to 8) | Individual Remote General Input Status | n/a | 0: Not Active<br>1: Active | yes | n/a | read | read | - | - | 320 |
| RINxxx | Remote System Signal Input Assignment | 0 | 0: Unassigned<br>1 to 8: Assigned | - | SR | yes | - | - | - | 321 |
| RIO | Remote I/O Status | n/a | n/a | yes | n/a | yes | - | - | - | 322 |
| ROUT | Remote General Output Control | 0 | 0 to 63 | yes | n/a | yes | yes | read | - | 323 |
| ROUTx<br>(x=1 to 6) | Individual Remote General Output Control | 0 | 0: Not Active<br>1: Active | yes | n/a | yes | yes | - | - | 324 |
| ROUTxxx | Remote System Signal Output Assignment | 0 | 0: Unassigned<br>1 to 6: Assigned | - | SR | yes | - | - | - | 325 |

## 12.2 I/O Signal and Command Structure

This table represents the corresponding relations for a series of command groups that are used when assigning or displaying I/O signals. Immediate commands that have same functions as the I/O signals are also shown with the corresponding relations.

| Immediate Command | I/O Assignment | CANopen Remote I/O Assignment | Action | Logic Level | Signal Status | Description |
|---|---|---|---|---|---|---|
| ABORT | INABORT | (Default Assignment) | - | ABORTLV | SIGABORT | Abort Motion and Sequence Execution |
| ALMCLR | INALMCLR | RINALMCLR | - | ALMCLRLV | SIGALMCLR | Alarm Clear |
| CONT | INCONT | RINCOMT | - | CONTLV | SIGCONT | Continue Motion |
| CURRENT | INCON | (Default Assignment) | - | CONLV | SIGCON | Current ON |
| FREE | INFREE | (Default Assignment) | - | FREELV | SIGFREE | Current OFF, Magnetic Brake Free |
| MCP | INMCP | (Default Assignment) | - | MCPLV | SIGMCP | Move Continuously Positive |
| MCN | INMCN | (Default Assignment) | - | MCNLV | SIGMCN | Move Continuously Negative |
| MGHP | INMGHP | RINMGHP | - | MGHPLV | SIGMGHP | Move Go Home Positive |
| MGHN | INMGHN | (Default Assignment) | - | MGHNLV | SIGMGHN | Move Go Home Negative |
| MSTOP | INMSTOP | RINMSTOP | MSTOPACT | MSTOPLV | SIGMSTOP | Motor Stop |
| PAUSE | INPAUSE | RINPAUSE | - | PAUSELV | SIGPAUSE | Pause Motion |
| PAUSECLR | INPAUSECL | RINPAUSECL | - | PAUSECLLV | SIGPAUSECL | Pause Clear |
| PECLR | INPECLR | RINPECLR | - | PECLRLV | SIGPECLR | Position Error Clear |
| PSTOP | INPSTOP | RINPSTOP | ALMACT | PSTOPLV | SIGPSTOP | Panic Stop |
| RUN | INSTART | (Default Assignment) | STARTACT | STARTLV | SIGSTART | Start Sequence |
| TL | INTL | RINTL | - | TLLV | SIGTL | Torque Limiting/Push-motion Operation /Current Cutback Release |
| - | INHOME | RINHOME | - | HOMELV | SIGHOME | Home Sensor |
| - | INLSP | RINLSP | OTACT | OTLV | SIGLSP | Limit Switch Positive |
| - | INLSN | RINLSN | OTACT | OTLV | SIGLSN | Limit Switch Negative |
| - | INSENSOR | RINSENSOR | SENSORACT | SENSORLV | SIGSENSOR | Sensor |
| - | OUTALARM | (Default Assignment) | - | ALARMLV | SIGALARM | Alarm |
| - | OUTEND | (Default Assignment) | ENDACT | ENDLV | SIGEND | Motion End |
| - | OUTHOMEP | (Default Assignment) | - | HOMEPLV | SIGHOMEP | Home Position |
|  | OUTLC | (Default Assignment) | - | LCLV | SIGLC | Limiting Condition |
| - | OUTMBFREE | ROUTMBFREE | MBFREEACT | - | SIGMBFREE | Magnetic Brake Free |
| - | OUTMOVE | (Default Assignment) | - | MOVELV | SIGMOVE | Motor Moving |
| - | OUTPSTS | ROUTPSTS | - | PSTSLV | SIGPSTS | Pause Status |
| - | OUTREADY | (Default Assignment) | - | READYLV | SIGREADY | Operation Ready |
| - | OUTRUN | ROUTRUN | - | RUNLV | SIGRUN | Sequence Running |

## 12.3  Command Description

The details of commands are explained in alphabetical order.

### # : Sequence Comment

| | |
|---|---|
| **Execution Mode** | Sequence |
| **Syntax** | #Commenting Text |
| **Range** | Alphanumeric Characters (not case sensitive), space and sign (! " # $ % & ' ( ) * + - . / : ; < = > ? @ [ \ ] ^ _ ) |
| **See Also** | EDIT, LIST |
| **Description** | 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. |
| | 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. |
| **Example** | Command                Description |
| | ```
>LIST 1          #List sequence 1

(1) TA=0.5       #Acceleration time, seconds
(2) TD=0.5       #Deceleration time, seconds
(3) VS=1         #Starting velocity, user units/second
(4) VR=2         #Running velocity, user units/second
(5) DIS=10       #Distance of the move equals 10 user units
(6) MI           #Begin the index move
(7) END          #End the sequence
>
``` |

## +, -, *, /, %, &, |, ^, <<, >>

| Execution Mode | Sequence |
|---|---|
| **Syntax** | Z = X n Y<br>X = Numeric Value or Variable<br>n = Math Operator<br>Y = Numeric Value or Variable<br>Z = Variable |
| **See Also** | A to Z |
| **Description** | The following math operators can be used in a program:<br>+ : Addition<br>- : Subtraction<br>* : Multiplication<br>/ : Division<br>% : Modulo (remainder)<br>& : AND (boolean)<br>\| : OR (boolean)<br>^ : XOR (boolean)<br><< : Left arithmetic shift (shift to left bit)<br>>> : Right arithmetic shift (shift to right bit)<br><br>Division by zero (0) or numeric overflow will cause an Alarm condition, stopping motion and halting sequence operation.<br><br>Note on Modulo operations: A%B = A - (B * sign (A/B) * floor (\|A/B\|) ) |

| Example | Command | Description |
|---|---|---|
| | `>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 command 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 |
| | `>PC` | #Query the PC value |
| | ` PC=10 Rev` | #Device response |
| | `>RUN 1` | #Run sequence 1 |
| | `>20` | #Device response |
| | `>` | |

## \ : Global Command

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | \Command |
| **See Also** | @, ID, TALK, VERBOSE |
| **Description** | Global command operator. Attaching this operator before the command enables command to all the units.<br><br>"\ID" is for checking all devices assigned ID numbers currently active on the daisy chain communication network.<br><br>Applicable Commands:<br>ABORT, CONT, CURRENT, CV, EHOME, HSTOP, ID, MA, MCP, MCN, MGHP, MGHN, MI, MIx, MSTOP, PAUSE, PAUSECLR, PSTOP, RESET, RUN, SSTOP |

| **Example** | Command | Description |
|---|---|---|
| | 2>\ID | #Send the global ID query command to all devices |
| | 3 | #Device response |
| | 1 | |
| | 2 | |
| | 0 | |
| | 2> | |

## ;  : Statement Separator

| | |
|---|---|
| **Execution Mode** | Immediate and Sequence |
| **Syntax** | Command; Command |
| **Description** | The semicolon (;) allows for multiple command statements to be used on a single command line.<br><br>The maximum number of characters per one line is 80 characters. |
| **Note** | 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. |

| **Example** | Command | Description |
|---|---|---|
| | >UU mm | #Set the user units to mm (millimeters) |
| | UU=mm | #Device response |
| | >VR 10;DIS 2;MI | #Set the running velocity to 10, distance to 2 and then perform an index move |
| | VR=10 mm/sec | #Device response |
| | DIS=2 mm | #Device response |
| | > | |

## @  : Select Device

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | @id |
| **Range** | id = *, 0 to 9 and A to Z (not case sensitive) |
| **See Also** | TALK, ID, \ (BACKSLASH) |
| **Description** | Makes a logical connection to a specific device in a multiple device, e.g. daisy chain configuration. That device can then be uniquely addressed and programmed. If the device ID is anything other than the default ID (*), communication with the device requires using the @ or TALK commands to establish communication |
| **Note** | Each device used in a daisy chain communication configuration requires a unique device ID. |
| **Example** | 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)

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | <ESC> (Escape Key or Character (1Bh)) |
| **See Also** | ABORT, ALMACT, HSTOP, MSTOP, MSTOPACT, PSTOP, SSTOP, TD |
| **Description** | <ESC> represents an escape key or character (1Bh).<br><br><ESC> will abort motion, decelerating to a stop.<br><br><ESC> will abort an executing sequence.<br><br><ESC> will also abort continuous display of a parameter via the (/) command.<br><br>The function to display the sequence progress by the TRACE command is canceled.<br><br><ESC> will discard any characters on a line and send a carriage return and line feed (CR + LF), and new prompt. |

| **Example** | Command | Description |
|---|---|---|
| | >UU mm | #Set the user units to mm (millimeters) |
| |  UU=mm | #Device response |
| | >VR 10 | #Set the running velocity to 10 mm/second |
| |  VR=10 mm/sec | #Device response |
| | >MCN | #Move continuously in the negative rotation direction |
| | > | #<ESC> received, motion begins decelerating to a stop |
| | > | #New prompt |

## a!=b, a<=b, a<b, a=b, a==b, a>=b, a>b : Conditional Operators

| | |
|---|---|
| **Execution Mode** | Sequence |
| **See Also** | IF, WHILE |
| **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==b : a is equal to b |
| | • a>=b : a is greater than or equal to b |
| | • a>b : a is greater than b |

| **Example** | Command | Description |
|---|---|---|
| | ( 1) DIS=0;VS=1;VR=10;TA=0;TD=0.1 | #Set motion parameters |
| | ( 2) LOOP | #Start infinite loop |
| | ( 3)    IF (IN6= =1) | #If input 6 is active |
| | ( 4)       DIS=1 | #Distance equals 1 user units |
| | ( 5)       MI | #Move incremental |
| | ( 6)       MEND | #Wait for motion to end |
| | ( 7)       DIS=0 | #Distance equals 0 user units |
| | ( 8)    ENDIF | #End of IF block |
| | ( 9)    IF (IN7= =1) | #If input 7 is active |
| | ( 10)      BREAKL | #Exit the loop and execute the line after the ENDL |
| | ( 11)   ENDIF | command |
| | ( 12) ENDL | #Terminate the LOOP |
| | ( 13) END | #End the sequence |

## ABORT : ABORT Motion and Sequence Execution

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | ABORT |
| **See Also** | <ESC>, ALMACT, HSTOP, MSTOP, MSTOPACT, PSTOP, SSTOP |
| **Description** | The ABORT command stops the motion and the execution of a sequence. If the motor is running, the motor decelerates to start velocity VS over deceleration time TD, and then stops completely. |
| | The ABORT function may also be executed via the ABORT input on the I/O connector if assigned and/or the CANopen remote I/O. When detailed explanation is required, see "6.4 Connecting the I/O signals" on page 23 for I/O connector, and see "10.4 Controlling I/O Message (PDO)" on page 134 for CANopen remote I/O. |

| **Example** | Command | Description |
|---|---|---|
| | >LIST 9 | #List sequence 9 |
| | ( 1) TA=0.5 | #Acceleration time, seconds |
| | ( 2) TD=0.1 | #Deceleration time, seconds |
| | ( 3) VR=20 | #Set the running velocity to 20 user units/second |
| | ( 4) MCP | #Move continuously in the positive direction |
| | ( 5) MEND | #Wait for stop to complete |
| | ( 6) END | #End the sequence |
| | >RUN 9 | #Execute sequence 9 |
| | >ABORT | #Abort sequence execution and decelerate the motor to a stop |
| | > | |

## ABSPLSEN  : Enable Driver Operation after Absolute Position Loss Alarm Release

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | ABSPLSEN |
| **See Also** | PRESET, ALMCLR, MGHP, MGHN, DSIGxxx (DSIGREQ), DOUTxxx (DOUTREQ) |
| **Description** | When the **NX** Series driver is used in the absolute system, executing this command after releasing the absolute position loss alarm will enable mechanical home seeking operation. This command will turn ON the P-REQ output assigned to the driver connector on the **CM10**/**SCX10** for about 1 msec. |
| | When setting the home position (by executing the PRESET command) without performing a mechanical home seeking operation, the ABSPLSEN command is not required to be executed. With the **ESMC** controller, the ABSPLSEN command is not required to execute even when mechanical home seeking operation is performed. |

**Example**

| Command | Description |
|---|---|
| >ALMCLR | #Release the driver absolute position loss alarm |
| >ABSPLSEN | #Operation enabled |
| >HOMETYP 4<br>  HOMETYP=4 | #Set the pattern of mechanical home seeking operation. Use HOME, LSP and LSN |
| >MGHP | #Start in the positive direction of mechanical home seeking operation |

## ABSREQ : Reading Driver Current Position

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | ABSREQ |
| **See Also** | PABS, ABSSTS, PRESET, ABSREQPC, DINxxx (DINPR, DINP0, DINP1), DOUTxxx (DOUTREQ, DOUTCK), ROUTxxx (ROUTABSDATA), DSIGxxx (DSIGREQ) |
| **Description** | This command reads and indicates the current position data, driver status code and driver alarm code from the driver that has the current position output function (the **NX** Series driver, **ESMC** controller etc.). |
| | The driver current position is converted to the user unit and written to the PABS, and driver status code and driver alarm code are written to the ABSSTS (driver status code/driver alarm code) parameters. |
| | The ABSREQ command can be used at any time regardless of the motor operation. |
| | Assign the (PR, P0, P1, REQ and CK) I/Os to the driver connector on the **CM10**/**SCX10** when using this command. |
| **Memo** | ・The range of the driver current position that can be read is "-2,147,483.648 to +2,147,483.647," which is the value after converting to the user unit. |
| | ・When updating the position command (PC) with the driver current position is required, please use the ABSREQPC command. |

| **Example** | Command | Description |
|---|---|---|
| | ```
>PC
 PC=0 Rev
>ABSREQ
 PABS=124.35 Rev
 Driver Status Code = 00
 Driver ALARM Code = 00
>PC
 PC=0 Rev
``` | #Confirm the PC value<br>#PC=0<br>#Read current position, driver status and driver alarm<br>#Current position<br>#Driver status code<br>#Driver alarm code<br>#Confirm the PC value<br>#PC=0 (no rewrite) |

## ABSREQPC : Reading Driver Current Position/Updating Internal Position

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | ABSREQPC |
| **Commands not Allowed** | MOVE |
| **See Also** | PABS, ABSSTS, PRESET, ABSREQ, DINxxx (DINPR, DINP0, DINP1), DOUTxxx (DOUTREQ, DOUTCK), ROUTxxx (ROUTABSDATA), DSIGxxx (DSIGREQ) |
| **Description** | This command reads and indicates the current position data, driver status code and driver alarm code from the driver that has the current position output function (the **NX** Series driver, **ESMC** controller etc.), and it also updates the value of PC (position command). (The Value of PF (feedback position) follows PC while maintaining position error.) |
| | The current position data will be converted to the user unit and written to the PABS (driver current position) parameter. The driver status and alarm will be written to the ABSSTS (driver status code/driver alarm code) parameter. When the electrical home is set and the software position limit control is set to 1 (SLACT=1), LIMP and LIMN (software position limits) will be enabled. |
| | Assign the (PR, P0, P1, REQ and CK) I/Os to the driver connector on the **CM10**/**SCX10** when using this command. |
| **Note** | ・When only referring to the current position, use the ABSREQ (reading driver current position) command. The current position can be referred to using ABSREQPC (reading driver current position/updating position command), but PC and PF (EC) will be overwritten. Also, an error will occur when ABSREQPC is used during pulse generating. |
| | ・Use the ABSREQPC command in the current-off/servo-off status. If the ABSREQPC command is executed in the current-on/servo-on status, the value may be the wrong value. Refer to "8.6 Driver Current Position Reading (**NX** Series driver, **ESMC** controller)" on page 100. |
| **Memo** | The range of the driver current position can be read is "-2,147,483.648 to +2,147,483.647," which is the value after converting to the user unit. The range to be written to PC is limited by MAXPOS (maximum position value). |
| **Example** | Command            Description |

| Command | Description |
|---|---|
| >PC | #Confirm the PC value |
| PC=0 Rev | #PC=0 (no rewrite) |
| >PF | #Confirm the PF value |
| PF=0 Rev | #PF=0 (no rewrite) |
| >ABSREQPC | #Read driver information and overwrite PC, PF and EC |
| PABS=124.35 Rev | #Current position |
| Driver Status Code = 1C | #Driver status code |
| Driver ALARM Code = 48 | #Driver alarm code |
| >PC | #Confirm the PC value |
| PC=124.35 Rev | #PC=124.35 (rewritten) |
| >PF | #Confirm the PF value |
| PF=124.35 Rev | #PF=124.35 (rewritten) |

## ABSSTS  : Driver Status Code/Driver Alarm Code

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | ABSSTS |
| **Access** | READ |
| **See Also** | ABSREQ, ABSREQPC, PABS, ROUTxxx (ROUTABSDATA) |
| **Description** | This is a variable to which the driver status code and driver alarm code acquired by ABSREQ or ABSREQPC is written.<br><br>When referring to the status code or alarm code from the host controller, refer to ABSSTS after executing the ABSREQ (reading driver current position) command or ABSREQPC (reading driver current position/updating internal position) command.<br><br>When reading ABSSTS via CANopen, execute the ABSREQ command or ABSREQPC command first and then execute ABSSTS after confirming that the ABSDATA output was 1 (ON) by remote I/O of CANopen.<br><br>When PABS and ABSSTS can be referred to if the ABSDATA output is assigned, the ABSDATA output will be 1 (ON).<br><br>ABSSTS cannot be read under the following conditions:<br>· Current position has not been read yet since the power is ON.<br>· Data is being read<br>· Although the data was read, a range that could be written was exceeded. |

| **Example** | Command | Description |
|---|---|---|
| | >ABSREQPC<br> PABS=124.35 Rev<br> Driver Status Code = 1C<br> Driver ALARM Code = 48<br>>ABSSTS<br> ABSSTS=1C48 | #After reading the current position, driver status and driver alarm, overwrite PC and PF (EC)<br>#Current position<br>#Driver status code<br>#Driver alarm code<br>#Driver status code, driver alarm code |

## ALM : Alarm Status and History

| | |
|---|---|
| **Execution Mode** | Immediate and CANopen (Recent Alarm Only) |
| **Syntax** | ALM |
| **See Also** | xxxLV (ALARMLV), ALMACT, ALMCLR, ALMMSG, ALMSET, CURRENT, OUTxxx (OUTALARM), DALARM, INxxx (INALMCLR), RINxxx (RINALMCLR), DOUTxxx (DOUTACLDCL) |
| **Description** | The ALM command displays the current alarm code, history of the last 10 alarm and warning issues, a brief alarm code description, and the elapsed time for the latest alarm code and warning message. |
| | See "13 Troubleshooting" for a list of all ALARM codes and causes. |
| | The current ALM code is overwritten upon device power up or reset. The alarm history is automatically saved in EEPROM. |
| **Memo** | If DALARM=1, the "driver alarm" may occur each time the **CM10/SCX10** and driver are powered ON, depending on the power on timing between the **CM10/SCX10** and the driver, and it will be recorded on alarm history. |
| | (The alarm output of the driver is negative logic = OFF during an alarm condition. If the power on timing of driver is later than the **CM10/SCX10**, the alarm output is OFF at the start up, and that is identical to "driver alarm." The driver alarm status is cleared automatically when the driver alarm output becomes ON, meaning alarm OFF.) |

| **Example** | Command | Description |
|---|---|---|
| | ```
>ALM
  ALARM =68 ,  RECORD : 68 00 00 00 00 00 00 00 00 00

  ALM_PSTOP , 67.156 [sec] past.

  WARNING =00 ,  RECORD : 00 00 00 00 00 00 00 00 00 00

  No warning.
>
``` | #Query the current ALARM code |

## ALMACT : ALARM Action

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | ALMACT=n |
| **Range** | n = 0: Motor Current Remains ON (ALARM OFF)<br>   1: Motor Current Remains ON (ALARM ON)<br>   2: Turn Motor Current OFF (ALARM ON) |
| **Factory Setting** | 2 |
| **SAVEPRM & RESET** | Required to execute any changes made to the parameter value and to save 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 factory setting. |
| **Access** | READ and WRITE |
| **See Also** | xxxLV (ALARMLV), DALARM, ALM, ALMCLR, OUTxxx (OUTALARM), PSTOP, INxxx (INALMCLR, INPSTOP), RINxxx (RINPSTOP) |
| **Description** | Establishes the action of the motor current and alarm state after a PSTOP operation, or hardware or software overtravel errors. |
| **Memo** | The ALMACT is effective only in limited types of alarms as above. See "13.1 Protective Functions and Troubleshooting" on page 367. |
| **Example** | Command |

| Command | Description |
|---|---|
| `>ALMACT=1` | #Set the ALMACT to 1 |
| ` ALMACT=2(1)` | |
| `>SAVEPRM` | #Save the parameter assignments |
| ` (EEPROM has been written 10 times)` | |
| ` Enter Y to proceed, other key to cancel. y` | |
| ` Saving Parameters........OK.` | |
| `>RESET` | #Establish the saved parameter values |
| ` Resetting system.` | |
| `-------------------------------------------` | |
| `          CM10-*` | |
| `          Controller Module` | |
| `          Software Version: *.**` | |
| `          Copyright 2010` | |
| `          ORIENTAL MOTOR CO., LTD.` | |
| `-------------------------------------------` | |
| `>ALMACT` | #Query new value |
| `>ALMACT=1(1)` | |
| `>` | |

## ALMCLR : ALARM Clear

| | |
|---|---|
| **Execution Mode** | Immediate and CANopen |
| **Syntax** | ALMCLR |
| **See Also** | INxxx (INALMCLR), RINxxx (RINALMCLR), DALARM, ALM, xxxLV (ALARMLV), ALMACT, ALMMSG, ALMSET, OUTxxx (OUTALARM), CURRENT |
| **Description** | 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. The ALMCLR function may also be executed via the ALMCLR input on the I/O connector and/or the CANopen remote I/O if assigned. When detailed explanation is required, see "6.4 Connecting the I/O signals" on page 23 for I/O connector, and see "10.4 Controlling I/O Message (PDO)" on page 134 for CANopen remote I/O. |
| **Note** | Before issuing an ALMCLR command, remove the cause of the alarm. If the ALARM condition persists, the **CM10**/**SCX10** will enter the ALARM state again. Please see the troubleshooting section for a description of the causes of specific ALARM codes.  Some alarm conditions cannot be cleared.  Refer to see "13 Troubleshooting" to see which conditions can and cannot be cleared. |
| **Memo** | If the system alarm status is active and it is caused by a driver alarm , the system alarm status becomes inactive when the driver alarm becomes inactive. |
| | If the system also has an alarm by anything other than a driver alarm at the same time, the system alarm status will remain active even after the driver alarm becomes inactive. |

| **Example** | Command | Description |
|---|---|---|
| | ```
>ALM
  ALARM =68 ,  RECORD : 68 68 66 60 66 66 60 68 66 66

  ALM_PSTOP , 3.062 [sec] past.

  WARNING =00 ,  RECORD : 00 00 00 00 00 00 00 00 00 00

  No warning.
>ALMCLR
>ALM
  ALARM =00 ,  RECORD : 68 68 66 60 66 66 60 68 66 66

  No alarm.

  WARNING =00 ,  RECORD : 00 00 00 00 00 00 00 00 00 00

  No warning.
>
``` | #Query ALM

#Clear the alarm condition, if possible |

## ALMMSG : ALARM Message Action

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | ALMMSG=n |
| **Range** | n = 0: No Messages<br>1: Messages, Alarms only<br>2: Messages, Alarms and Warnings |
| **Factory Setting** | 0 |
| **SAVEPRM** | 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 factory setting. |
| **Access** | READ and WRITE |
| **See Also** | xxxLV (ALARMLV), ALM, ALMACT, ALMCLR, ALMSET, OUTxxx (OUTALARM), INxxx (INALMCLR), DALARM, DSIGxxx (DSIGALARM) |
| **Description** | The system can automatically transmit a message when alarms or warnings are detected. ALMMSG controls what types of messages are automatically transmitted.<br><br>Warning messages are sent only if the detected warning condition is different from the last reported warning. |

| **Example** | Command | Description |
|---|---|---|
| | >ALMMSG=1 | #Set the ALMMSG to messaging alarm only |
| | ALMMSG=1 [Alarm] | |
| | >SAVEPRM | #Save the parameter assignments |
| | (EEPROM has been written 10 times) | #Device response |
| | Enter Y to proceed, other key to cancel. Y | |
| | Saving Parameters........OK. | |

## ALMSET  : Set User ALARM

| | |
|---|---|
| **Execution Mode** | Immediate and Sequence |
| **Syntax** | ALMSET |
| **See Also** | xxxLV(ALARMLV), ALM, ALMACT, ALMCLR, ALMMSG, OUTxxx (OUTALARM), SIGxxx (SIGALARM) |
| **Description** | The ALMSET command allows the user to place the device in a forced alarm state. |

| **Example** | Command | Description |
|---|---|---|
| | >LIST CHKINPUT | #List sequence CHKINPUT |
| | | |
| | (  1) DIS 10; VR 1 | #Set distance to 10, run velocity to 1 |
| | (  2) MI | #Start incremental motion |
| | (  3) WHILE (SIGMOVE=1) | #While system is moving… |
| | (  4)   IF (IN1=1) | #If general purpose input #1 is active |
| | (  5)     SAS Illegal sensor input entry! | #Transmit a message |
| | (  6)     SSTOP | #Stop motion |
| | (  7)     MEND | #Wait for stop to complete |
| | (  8)     ALMSET | #Force an alarm: sequence halts. |
| | (  9)   ENDIF | #Terminate IF block |
| | ( 10) WEND | #Terminate WHILE loop |
| | ( 11) SAS Motion succeeded | #Send a success message |
| | >ALMMSG=1 | |
| | ALMMSG=1 [Alarm] | |
| | >RUN CHKINPUT | #Run sequence CHKINPUT |
| | >Motion succeeded | #Successful |
| | >RUN CHKINPUT | #Run again |
| | >Illegal sensor input entry! | #Sequence aborted |
| | >ALMSET command detected. | #Check alarm |
| | >ALM | |
| | ALARM =E0 ,  RECORD : E0 30 23 9A 23 68 68 66 60 66 | |
| | | |
| | ALM_USR_ALARM , 12.887 [sec] past. | |
| | | |
| | WARNING =00 ,  RECORD : 00 00 00 00 00 00 00 00 00 00 | |
| | | |
| | No warning | |
| | >SIGALARM | #Query the ALARM status signal |
| | SIGALARM=1 | #The device is in an ALARM state |
| | >ALMCLR | #Clear the alarm |
| | > | #Device response |

## A to Z : User Variables

| | |
|---|---|
| **Execution Mode** | Immediate and Sequence |
| **Syntax** | A=n (in sequence only : expression)<br><br>:<br><br>Z=n<br><br>Upper and lower case are permitted, but 'A' and 'a' reference the same variable. There are 26 variables. |
| **Range** | n = -Maximum Number to +Maximum Number<br><br>expression must evaluate to a value within the same range as n, and can be any of:<br>- constant numeric value<br>- any variable available to sequences<br>- math expression |
| **Factory Setting** | 0 |
| **SAVEPRM** | 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 factory setting. |
| **Access** | READ and WRITE |
| **See Also** | CLEARVAR, CREATEVAR, DELETEVAR, N_xxx, POS[x], S_xxx, SAVEALL, SAVEPRM, VIEW, SAS, SACS |
| **Description** | General purpose numeric variables.<br><br>In immediate mode, A to Z may only be set and queried.<br><br>Within a sequence, variables may also be used in the following conditions:<br>· Targets or arguments for assignments (e.g. A=TIMER; DIS=A)<br>· Loop Counters (e.g. LOOP Q)<br>· Conditional Statement Values (e.g. if (VR>X))<br>· Arguments for a subroutine CALL (e.g. CALL S)<br>· Parts of Mathematical Expressions (e.g. VR=VS+C)<br>· Targets for interactive data entry commands (e.g. X=KBQ)<br><br>User variables may be saved by issuing the SAVEPRM (Save all parameter values) command while in immediate mode. If the variables values are not saved upon the next RESET or power cycle of the product, the variables will be cleared to the value of zero.<br><br>A sequence will not show the name of the variable (A to 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.<br><br>For example:<br>Sequence 1<br>`( 1) A=2`    #Set the value of variable A<br>`( 2) A`        #Display the value of A<br><br>When sequence 1 executes the device displays the following:<br>`>RUN 1`<br>` 2`                #Device response to line 2 (shown above)<br>`>`<br><br>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.<br>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. |

**Example**

| Command | Description |
|---------|-------------|
| >B 0.1 | #Set the variable B to a value of 0.1 |
|  B=0.1 | #Device response |
| >LIST 1 | #List sequence 1 |
|  |  |
| (  1)  A=KB | #Query the user for the value of the variable A via the serial port |
| (  2)  LOOP A | #Use A as a loop count |
| (  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 Rev |  |
| >RUN 1 | #Run sequence 1 |
| >? 4 | #Prompt the user for the value of A |
|  | #Motion executes 4 times |

## BAUD  : RS-232C BAUD Rate

| | |
|---|---|
| **Execution Mode** | Immediate and Sequence |
| **Syntax** | BAUD=n |
| **Range** | n =  0: 9600 (bps)<br>       1: 19200<br>       2: 38400<br>       3: 57600<br>       4: 115200 |
| **Factory Setting** | 0 |
| **SAVEPRM & RESET** | Required to execute any changes made to the parameter value and to save 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 factory setting. |
| **Access** | READ and  WRITE<br>READ only in Sequences |
| **See Also** | @, ECHO, ID, TALK, VERBOSE |
| **Description** | Establishes the RS-232C communication baud rate for the device. |
| **Note** | The default RS-232C baud rate of the **CM10/SCX10** is 9600 bps, same as the default baud rate of a general Windows computer. If the baud rate on the computer or the **CM10/SCX10** is changed, the baud rate must also be changed on the other.  (Always set the **CM10/SCX10** baud rate first, then set the baud rate of the computer.)<br><br>Check the baud rate of  the computer always set the **CM10**/**SCX10** baud rate first, then set the baud rate of the computer application that is used to communicate with the **CM10**/**SCX10**, or check the COM port property of windows if the application does not have a baud rate function. (The supplied utility software, **IMC** is set to 9600 bps when it is installed and no change is required for initial connection to the **CM10**/**SCX10**.)<br><br>When daisy chaining several devices, a higher baud rate reduces the amount of time required for communicating with each device on the chain. However, when using a daisy chain longer than 30 m (10 feet), a high baud rate (greater than 9600 bps) may not operate properly because of communication signal deterioration over the line.<br>All units in a daisy chain configuration must have the same BAUD setting. |

| **Example** | Command | Description |
|---|---|---|
| | >BAUD 1<br> BAUD=0(1) [9600bps(19200bps)]<br>>SAVEPRM<br> (EEPROM has been written 21 times)<br> Enter Y to proceed, other key to cancel. y<br> Saving Parameters........OK.<br>>RESET<br> Resetting system.<br>----------------------------------------<br>          CM10-*<br>          Controller Module<br>          Software Version: *.**<br>          Copyright 2010<br>          ORIENTAL MOTOR CO., LTD.<br>----------------------------------------<br>>BAUD<br> BAUD=1(1) [19200bps(19200bps)] | #Set the Baud Rate to 19200 Bits per second (bps)<br>#Save the parameter assignments<br><br><br><br>#Reset the system to establish the new baud value<br>#NOTE: change baud rate of host system before proceeding!<br><br><br><br><br><br>#Query the Baud Rate<br>#Baud is set as 19200 |

## BREAKL : Break LOOP Block

| | |
|---|---|
| **Execution Mode** | Sequence |
| **Syntax** | BREAKL |
| **See Also** | BREAKW, ELSE, ENDIF, ENDL, IF, LOOP, WEND, WHILE |
| **Description** | Exits the innermost LOOP block. Often used to exit a LOOP based on the value of a conditional statement. |

**Example**

| Command | Description |
|---|---|
| >LIST 7 | #List sequence 7 |
| | |
| ( 1) LOOP | #Loop indefinitely |
| ( 2)   IF (IN2=1) | #If input 2 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 |
| ( 8) END | #End the sequence |
| > | |

## BREAKW  : Break WHILE Block

| | |
|---|---|
| **Execution Mode** | Sequence |
| **Syntax** | BREAKW |
| **See Also** | BREAKL, ELSE, ENDIF, ENDL, IF, LOOP, WEND, WHILE |
| **Description** | Exits the innermost WHILE block. |
| | Often used to exit a WHILE block based on the value of a conditional statement. |

**Example**

| Command | Description |
|---|---|
| >LIST 8 | #List sequence 8 |
| | |
| (  1)  MCP | #Move continuously (positive) |
| (  2)  WHILE (IN1=0) | #Start WHILE block. Execute lines 3 through 5 while condition is true |
| (  3)    IF (IN2=1) | #If input 2 is 1 (ON), execute line 4 |
| (  4)      BREAKW | #Exit the WHILE loop and execute the line after the WEND command |
| (  5)    ENDIF | #End the IF block |
| (  6)  WEND | #End the WHILE block, return to line 2 |
| (  7)  SSTOP | #Slow down and stop the motor |
| (  8)  END | #End the sequence |

# CALL : Call Sequence as Subroutine

| | |
|---|---|
| **Execution Mode** | Sequence |
| **Syntax** | CALL n |
| **Range** | n = Valid Sequence Name or Number, or Variable |
| **See Also** | DIR, RET |
| **Description** | Executes a sequence as a subroutine, then returns to the calling sequence. |
| | If target 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. |
| | 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." |
| | If the target sequence does not exist, an alarm is triggered, and all sequence processing stops. |

| **Example** | 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 user units |
| | `(  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 command in sequence 1 is line 3 |
| | `>` | and is the next line to execute after the subroutine sequence 2 completes executing. |

## CANBAUD : CANopen BAUD Rate

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | CANBAUD=n |
| **Range** | n = 0: 10 (kbps)<br>1: 20 (kbps)<br>2: 50 (kbps)<br>3: 125 (kbps)<br>4: 250 (kbps)<br>5: 500 (kbps)<br>6: 800 (kbps)<br>7: 1 (Mbps) |
| **Factory Setting** | 1 |
| **SAVEPRM & RESET** | Required to execute any changes made to the parameter value and to save 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 factory setting. |
| **Access** | READ and WRITE |
| **See Also** | CANID |
| **Description** | Establishes the CANopen communication baud rate for the device. |
| **Note** | If the CANopen baud rate on the computer or the **CM10**/**SCX10** is changed, the baud rate must also be changed on the other. |

| Example | Command | Description |
|---|---|---|
| | ```>CANBAUD=1```<br>``` CANBAUD=0(1) [10kbps(20kbps)]```<br>```>SAVEPRM```<br>``` (EEPROM has been written 21 times)```<br>``` Enter Y to proceed, other key to cancel. y```<br>``` Saving Parameters........OK.```<br>```>RESET```<br>``` Resetting system.```<br>```------------------------------------------```<br>```           CM10-*```<br>```           Controller Module```<br>```           Software Version: *.**```<br>```           Copyright 2010```<br>```           ORIENTAL MOTOR CO., LTD.```<br>```------------------------------------------```<br>```>CANBAUD```<br>``` CANBAUD=1(1) [20kbps(20kbps)]``` | #Set the baud rate to 20 kbits per second (kbps)<br>#Save the parameter assignments<br><br><br>#Reset the system to establish the new baud value<br>#NOTE: change baud rate of host system before proceeding!<br><br>#Query the baud rate<br>#Baud is set as 20 kbps |

# CANID : CANopen Node Address

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | CANID=n |
| **Range** | n = 1 to 127 |
| **Factory Setting** | 1 |
| **SAVEPRM & RESET** | Required to execute any changes made to the parameter value and to save 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 factory setting. |
| **Access** | READ and WRITE |
| **See Also** | CANBAUD |
| **Description** | Sets the CANopen node address. |

| **Example** | Command | Description |
|---|---|---|
| | `>CANID=10` | #Set CANID=10 |
| | ` CANID=1(10)` | |
| | `>SAVEPRM` | #Save the parameter assignments |
| | ` (EEPROM has been written 10 times)` | #Device response |
| | ` Enter Y to proceed, other key to cancel. y` | |
| | ` Saving Parameters........OK.` | |
| | `>RESET` | #Establish the saved parameter values |
| | ` Resetting system.` | |
| | `-------------------------------------------` | |
| | `          CM10-*` | |
| | `          Controller Module` | |
| | `          Software Version: *.**` | |
| | `          Copyright 2010` | |
| | `          ORIENTAL MOTOR CO., LTD.` | |
| | `-------------------------------------------` | |
| | `>CANID` | #CANID request |
| | ` CANID=10(10)` | |
| | `>` | |

## CLEARALL  : Return to Factory Condition

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | CLEARALL |
| **Commands not Allowed** | MOVE, RUN |
| **See Also** | CLEARPOS, CLEARSEQ, CLEARVAR, INITPRM |
| **Description** | Clears all parameters, POS[x] position array data and all sequences. The CLEARALL command will clear all of the input and output assignments. |
| **Caution** | **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. The number of times for accumulated write cycles is displayed when executing.** |
| **Note** | Once the information is cleared, it cannot be restored. |
| | Since the baud rate and ID restore to the factory settings when executing CLEARALL, the communication cannot be performed if these settings have been changed. Note this point. |
| | A locked sequence will be cleared by CLEARALL: the lock status offers no protection for these operations. |

| **Example** | Command | Description |
|---|---|---|
| | ```
>CLEARALL
 (EEPROM has been written 12 times)
 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

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | CLEARPOS |
| **Commands not Allowed** | MOVE, RUN |
| **See Also** | CLEARALL, CLEARSEQ, CLEARVAR, TEACH |
| **Description** | Clears all POS[x] position array data. Position data will set to 0. |
| **Caution** | **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. The number of times for accumulated write cycles is displayed when executing.** |
| **Note** | Once the data points are cleared, they cannot be restored. |

| **Example** | Command | Description |
|---|---|---|
| | ```>CLEARPOS``` <br> ```(EEPROM has been written 13 times)``` <br> ``` Enter Y to proceed, other key to cancel. y``` <br> ``` Clear POS[ ] Data.....OK.``` <br> ```>``` | #Clear all position array data to 0 |

## CLEARSEQ  : Clear Sequences

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | CLEARSEQ |
| **Commands not Allowed** | RUN |
| **See Also** | CLEARALL, CLEARPOS, CLEARVAR, DEL, EDIT |
| **Description** | 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. |
| **Caution** | **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. The number of times for accumulated write cycles is displayed when executing.** |
| **Note** | Once the sequences are deleted, they cannot be restored.<br><br>A locked sequence will be cleared by CLEARSEQ: the lock status offers no protection for these operations. |

| **Example** | Command | Description |
|---|---|---|
| | `>DIR` | #List all sequences |

```
  ##  Name       TextSize  Locked
  ==  ==========  ========  ======
   0  <nameless>  10
   1  <nameless>  37

  Total:   2
  Executable memory:     43 bytes used of  6144 bytes total,     1 percent.
  Storage memory:        98 bytes used of 21775 bytes total,     0 percent.
>CLEARSEQ                                   #Delete all sequences from memory
(EEPROM has been written 14 times)
 Enter Y to proceed, other key to cancel. y  #Device response sent to the terminal
 Clearing.................OK.                 #Device response sent to the terminal
>DIR                                         #List all sequences

No Sequence(s) found.
>
```

## CLEARVAR  : Clear User-defined Variables

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | CLEARVAR |
| **SAVEPRM** | 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 factory setting. |
| **Commands not Allowed** | RUN |
| **See Also** | CLEARALL, CLEARPOS, CLEARSEQ, DELETEVAR, LISTVAR, INITPRM, N_xxx, S_xxx |
| **Description** | CLEARVAR clears all user-defined variables from memory. |
| **Caution** | **The CLEARVAR command writes to EEPROM. The EEPROM has a nominal expected lifetime of 100,000 write cycles.  The CLEARVAR command should not be used automatically (i.e. by a host controller) if it could possibly execute at high frequency. The number of times for accumulated write cycles is displayed when executing.** |
| **Note** | Once the variables are cleared, they cannot be restored. |

| **Example** | Command | Description |
|---|---|---|
| | `>LISTVAR` | #List all user-defined variables |

```
    ##   N_name      Numeric Data
    ==   ==========  ============
     1   LOOPS       10
     2               0
     3               0
     4               0
     5               0
     6               0
     7               0
     8               0
     9               0
    10               0
    ##   S_name      String Data
    ==   ==========  ====================
     1   LABEL       OMUSA
     2
     3
     4
     5
     6
     7
     8
     9
    10
>CLEARVAR                                    #Clear all user-defined variables from
 Enter Y to proceed, other key to cancel. y  memory
 All user parameters are deleted.
 >
```

## CONT : Continue Motion

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | CONT |
| **See Also** | PAUSE, PAUSECLR, INxxx (INPAUSE, INPAUSECL), OUTxxx (OUTPSTS) |
| **Description** | 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. If the paused motion was a point-to-point index (MI, MA), the former destination becomes the destination for the resumed motion. |
| | If the paused motion was a continuous motion, the former direction is assumed for the continued motion. Acceleration and deceleration times TA and TD, and start and running velocities VS and VR determine the motion profile while changing speed. |
| | The CONT command has no effect if motion has not been previously PAUSE'd. |
| | The CONT function may also be executed via the CONT input on the I/O connector and/or the CANopen remote I/O if assigned. When detailed explanation is required, see "8.3 Stopping Motion and Sequence" on page 93. |
| | If "STARTACT=0" is set, the START input can cause the same action as the CONT command, while sequences are running. |
| **Note** | 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, or new motion). Linked motions, return-to-electrical home operation and mechanical home seeking cannot be paused and resumed: PAUSE causes a soft stop, and CONT is ignored. |
| **Example** | Command | Description |

```
>LIST WATCHPAUSE            #List sequence WATCHPAUSE

(  1) MA X                  #Start motion, to position in variable 'X'
(  2) WHILE (PC!=X)         #While position command 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
>
```

## COPY : Copy Sequence

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | COPY source target |
| **Range** | source and target can be any valid sequence number (0-99) or name (consisting of letters, numbers or underscore, 10 characters maximum, must start with a letter except n, s, N, S, with no distinction of a capital letter or small letter.) |
| **Commands not Allowed** | RUN |
| **See Also** | DEL, EDIT, REN |
| **Description** | 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. |

| **Example** | Command | Description |
|---|---|---|
| | >COPY 1 MASTER | #Copy sequence 1 to sequence named MASTER |
| | >COPY MASTER 2 | #Copy sequence MASTER to sequence 2 |

## CREATEVAR : Create User-defined Variable

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | CREATEVAR {N_xxx \| S_xxx} {value \| string} |
| **Range** | xxx = Variable Name: 1 to 10 Alphanumeric Characters |
| | value \| string (optional): initial numeric value (N_xxx) or string value (S_xxx). If empty, N_xxx variables are initialized to 0 and S_xxx variables are initially empty. |
| **SAVEPRM** | The new value takes effect immediately. However, SAVEPRM is required to save the parameter values in nonvolatile memory. If SAVEPRM is not executed after a variable has been created, that variable will not exist after a RESET or power cycle. |
| **Commands not Allowed** | RUN |
| **See Also** | A to Z, CLEARALL, CLEARVAR, DELETEVAR, LISTVAR, N_xxx, S_xxx, INITPRM |
| **Description** | Create a user-defined variable. A numeric variable (N_xxx) has a numeric value, while a string variable (S_xxx) can store a string of up to 20 characters. |
| | 10 variables are allowed for each type, numeric and string. Numeric type variable must start with "N_," and string type variable must start with "S_." |
| | Variables are initialized as they are created. If no initialization constant is present, numeric variables (N_xxx) are automatically initialized to 0, and string variables (_xxx) are automatically initialized as "empty." |
| | In order to avoid "careless" creation by variable access, new variable creation requires this command, and new variables cannot be created in a sequence. New variables can be created only in immediate mode. |
| **Note** | Using user-defined variables can make sequences more readable, but accessing these variables requires an internal name lookup operation. |
| | That operation takes an unpredictable amount of time, which depends on system activity and the number of user-defined variables that have been created. |
| | For applications with tight timing requirements, consider using general purpose variables A to Z instead. |

| Example | Command | Description |
|---------|---------|-------------|
| | >CREATEVAR N_DEPTH | #Create user-defined numeric variable named N_DEPTH |
| | New variable N_DEPTH is added. | |
| | N_DEPTH=0 | |
| | >N_DEPTH 10.02 | #Set user-defined numeric variable value |
| | N_DEPTH=10.02 | |
| | >CREATEVAR S_LABEL IDLE | #Create user-defined string variable named S_LABEL, initialize to "IDLE" |
| | New variable S_LABEL is added. | |
| | S_LABEL=IDLE | |
| | >S_LABEL RUNNING | #Set user-defined string variable value |
| | S_LABEL=RUNNING | |
| | >LISTVAR | #List all user-defined variables |

```
    ##   N_name      Numeric Data
    ==   ==========  ============
     1   DEPTH       10.02
     2               0
     3               0
     4               0
     5               0
     6               0
     7               0
     8               0
     9               0
    10               0
    ##   S_name      String Data
    ==   ==========  ====================
     1   LABEL       RUNNING
     2
     3
     4
     5
     6
     7
     8
     9
    10
  >
```

## CURRENT : Current On/Off

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | CURRENT=n |
| **Range** | n = 0: Motor Current is OFF<br>1: Motor Current is ON |
| **Factory Setting** | 0: If the STRSW is set to zero(0)  (**CM10-1**, **5**)<br>1: If the STRSW is set to 1 (**CM10-2**, **3**, **4**, **SCX10**) |
| **Access** | READ and WRITE |
| **See Also** | STRSW, INxxx (INCON), xxxLV (CONLV), SIGxxx (SIGCON) |
| **Description** | Enables or disables the motor current. |
| **Note** | If the operation is made immediately after Current ON is commanded, position error may occur. Allow a time interval according to the timing chart for each driver. Care should be taken especially when using CURRENT command in sequence program, or controlling CURRENT command, CON/COFF terminal or CON in CANopen by the host controller programs. |
| **Memo** | If the CON input is assigned to the I/O connector and/or the CANopen is active, the CURRENT command is available only when all active CON inputs are ON.<br><br>If the CON input is not assigned to the I/O connector and CANopen is not active, the CURRENT status at power on is determined by the STRSW setting. If the CON input is assigned to the I/O connector and/or the CANopen is active, the CURRENT status (motor current) at power on is determined by those inputs.<br><br>The "Current OFF" always has higher priority than "Current ON" among CON in system input, CON in remote (CANopen) input and CURRENT command.<br><br>During CURRENT is 0 (motor current is off), PC (position command) is continuously overwritten by PF (position feedback) value. This is to track the actual position. |

| **Example** | Command | Description |
|---|---|---|
| | >CURRENT 0<br> CURRENT=0 | #Turn motor current OFF. Motor has no holding torque |
| | >CURRENT 1<br> CURRENT=1<br>> | #Turn motor current ON. Motor now has holding torque |

## CV : Change Velocity

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | CV=n |
| **Range** | n = 0.001 to MAXVEL (user units/second)<br><br>(In sequences, the maximum value is further limited by "Max. Number.") |
| **See Also** | DPR, MA, MCP, MCN, MI, MIx, VR, VS, UU, MAXVEL, SCHGVR, SCHGPOS |
| **Description** | 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).<br><br>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 error message. If CV is attempted within a sequence while the motor is decelerating, an alarm is set (70h).<br><br>CV is only available.<br><br>Changing the running velocity via the CV command will affect the time required to complete the original commanded motion profile.<br><br>There are several other ways to change speeds while moving:<br>- If moving continuously by MCP, set new VR, and execute MCP again.<br>- If moving continuously by MCN, set new VR, and execute MCN again.<br>- If all motion parameters are known, use linked index motions. Refer to MIx.<br>- Use the SENSOR input with SCHGVR and SCHGPOS. |
| **Memo** | If successful, a CV command modifies running velocity VR. The new value of VR will be "n" (the argument to the CV command). |

| **Example** | Command | Description |
|---|---|---|
| | >UU mm | #Set user units (UU) to mm (millimeters) |
| | UU=mm | |
| | >VR 3 | #Set the running velocity to 3 mm/second |
| | VR=3 mm/sec | |
| | >DIS 10 | #Set the distance to 10 mm |
| | DIS=10 mm | |
| | >MI | #Start the index move |
| | >CV 5 | #Change the running velocity to 5 mm/second |
| | >MSTOP | #Stop motion |
| | >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=5 | #Set the starting velocity, UU/second |
| | ( 4) VR=10 | #Set the running velocity, UU/second |
| | ( 5) DIS=100 | #Set the distance, UU |
| | ( 6) MI | #Execute an index move |
| | ( 7) WHILE (IN3=0) | #While input 3 is OFF, wait |
| | ( 8) WEND | #If Input 3 is OFF to back to line 7, otherwise go to line 8 |
| | ( 9) CV 15 | #Change the running velocity of the index move to 15 UU/second |
| | ( 10) SAS SPEED CHANGE | #Transmit ASCII string |
| | ( 11) END | #End the program |
| | > | |

## DALARM  : Driver ALARM Signal Enable

| | |
|---|---|
| **Execution Mode** | Immediate and Sequence |
| **Syntax** | DALARM=n |
| **Range** | n = 0: Do not use the ALARM signal input on the driver connector of the **CM10**/**SCX10**<br>    1: Use the ALARM signal input on the driver connector of the **CM10**/**SCX10** |
| **Factory Setting** | 0: **SCX10**<br>1: **CM10-1**, **2**, **3**, **4**, **5** |
| **SAVEPRM & RESET** | Required to execute any changes made to the parameter value and to save 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 factory setting. |
| **Access** | READ and WRITE<br>READ only in Sequences |
| **See Also** | DINxxx (DINALARM), DSIGxxx (DSIGALARM) |
| **Description** | DALARM command enables the use of the ALARM signal input on the driver connector of the **CM10**/**SCX10**.  If this function is active and a driver alarm has occurred, the system ALARM signal/status becomes active (alarm code 6Eh: driver alarm).<br><br>The alarm status automatically becomes inactive when the driver alarm becomes inactive. |

**Example**

| Command | Description |
|---|---|
| >DALARM=0 | #Set DALARM=0 |
| DALARM=1(0) [Enable(Disable)] | |
| >SAVEPRM | #Save the parameter assignments |
| (EEPROM has been written 80 times) | |
| Enter Y to proceed, other key to cancel. Y | |
| Saving Parameters........OK. | |
| >RESET | #Establish the saved parameter values |
| Resetting system. | |
| ----------------------------------------- | |
|       CM10-* | |
|       Controller Module | |
|       Software Version: *.** | |
|       Copyright 2010 | |
|       ORIENTAL MOTOR CO., LTD. | |
| ----------------------------------------- | |
| >DALARM | #Confirm the new assignment |
| DALARM=0(0) [Disable(Disable)] | |
| > | |

## DD : Driver Operation Data

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | DD=n |
| **Range** | n = 0 to 3 (torque limiting) <br> 0 to 7 (push-motion operation) |
| **Factory Setting** | 0 |
| **SAVEPRM** | 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 factory setting. |
| **Access** | READ, WRITE |
| **See Also** | TL, DOUTxxx (DOUTM0, DOUTM1, DOUTM2), DSIGxxx (DSIGM0, DSIGM1, DSIGM2) |
| **Description** | This is a parameter to set the driver operation data such as the push-motion operation current, torque limiting value etc. <br><br> Assign the following signals to the driver connector on the **CM10**/**SCX10**. <br> ・Torque limiting: M0, M1 <br> ・Push-motion operation: M0, M1,  M2 <br><br> M0, M1 and M2 that were assigned to the driver connector on the **CM10**/**SCX10** will vary as follows by DD value. |

| DD | M2 | M1 | M0 |
|---|---|---|---|
| 0 | OFF | OFF | OFF |
| 1 | OFF | OFF | ON |
| 2 | OFF | ON | OFF |
| 3 | OFF | ON | ON |
| 4 | ON | OFF | OFF |
| 5 | ON | OFF | ON |
| 6 | ON | ON | OFF |
| 7 | ON | ON | ON |

| **Example** | Command | Description |
|---|---|---|
| | >DD=1 <br> DD=1 | #Assign the data 1 (M0 is set to ON, and M1 and M2 are set to OFF) |
| | >TL=1 <br> TL=1 | #Torque limiting operation or push-motion operation is enabled |

## DEL  : Delete Sequence

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | DEL target |
| **Range** | target can be the name or number of any existing sequence. |
| **Commands not Allowed** | RUN |
| **See Also** | CLEARALL, CLEARSEQ, COPY, DIR, EDIT, LOCK, UNLOCK |
| **Description** | Deletes a sequence from EEPROM. The system will request confirmation of the DEL action. |
| | A deleted sequence cannot be recovered. |
| | If the sequence is locked, it cannot be deleted.  Use the UNLOCK command to unlock the sequence before deleting. |
| | Sequences cannot be deleted while any sequence is running. |
| **Note** | To delete all sequences see the CLEARSEQ command. |
| **Example** | Command                                                          Description |

```
>DIR                                        #Display the stored programs


   ##  Name         TextSize  Locked
   ==  ==========   ========  ======
    0  test1               8
    1  <nameless>         32
  Total:   2
  Executable memory:     27 bytes used of  6144 bytes total,   0 percent.
  Storage memory:        87 bytes used of 21775 bytes total,   0 percent.
>DEL TEST1                                  #Delete program TEST1 from memory
 Enter Y to proceed, other key to cancel. y  #Device response
>
```

## DELETEVAR : Delete User-defined Variable

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | DELETEVAR {N_xxx | S_xxx} |
| **Range** | xxx = Variable Name: 1 to 10 Alphanumeric Characters |
| **SAVEPRM** | 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 factory setting. |
| **Commands not Allowed** | RUN |
| **See Also** | CLEARALL, CLEARSEQ, CLEARVAR, N_xxx, S_xxx, CREATEVAR |
| **Description** | Deletes a specific user-defined variable. |

| **Example** | Command | Description |
|---|---|---|
| | >LISTVAR | #List user-defined variables |

```
   ##  N_name      Numeric Data
   ==  ==========  ============
    1  LOOPS       10
    2              0
    3              0
    4              0
    5              0
    6              0
    7              0
    8              0
    9              0
   10              0
   ##  S_name      String Data
   ==  ==========  ====================
    1  LABEL       OM USA
    2
    3
    4
    5
    6
    7
    8
    9
   10
>DELETEVAR N_LOOPS                               #Delete user-defined numeric variable
 Enter Y to proceed, other key to cancel. Y
 Variable N_LOOPS is deleted.
>LISTVAR
```

```
        ##   N_name      Numeric Data              #N_LOOPS is gone
        ==   ==========  ============
         1               0
         2               0
         3               0
         4               0
         5               0
         6               0
         7               0
         8               0
         9               0
        10               0
        ##   S_name      String Data
        ==   ==========  ====================
         1   LABEL       OM USA
         2
         3
         4
         5
         6
         7
         8
         9
        10                                         #SAVEPRM required to make this
>SAVEPRM                                            change permanent
 (EEPROM has been written 17 times)
 Enter Y to proceed, other key to cancel. y
 Saving Parameters........OK.
 >
```

## DEND : Driver END Signal Enable

| | |
|---|---|
| **Execution Mode** | Immediate and Sequence |
| **Syntax** | DEND=n |
| **Range** | n = 0: Internal End Area<br>    1: Driver END Signal |
| **Factory Setting** | 0: **CM10-2**, **SCX10**<br>1: **CM10-1**, **3**, **4**, **5** |
| **SAVEPRM & RESET** | Required to execute any changes made to the parameter value and to save 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 factory setting. |
| **Access** | READ and WRITE<br>READ only in Sequences |
| **See Also** | DINxxx (DINEND), DSIGxxx (DSIGEND), ENDACT, MEND |
| **Description** | DEND command selects the source of the END signal, either driver end signal or the internal end signal. The selected signal becomes the system END signal/status, and used for the MEND command, END output, and mechanical home seeking.<br>See "8.8 END (motion of end) Signal" on page 105. |
| **Memo** | The internal end signal is generated by end area and/or end of pulse generation. (See ENDACT.)<br><br>When DEND is set to 1, there may be the case where END signal intermittently outputs during motor is moving at slow speed. This is because the END signal of the driver is intermittently outputting. Set DEND to 0 and use the internal END signal of **CM10**/**SCX10** to avoid intermittent END signal. |

| **Example** | Command | | Description |
|---|---|---|---|
| | ```>DEND=0``` | | #Set DEND=0 |
| | ` DEND=1(0) [Enable(Disable)]` | | |
| | `>SAVEPRM` | | |
| | ` (EEPROM has been written 80 times)` | | |
| | ` Enter Y to proceed, other key to cancel. Y` | | |
| | ` Saving Parameters........OK.` | | |
| | `>RESET` | | |
| | ` Resetting system.` | | |
| | `------------------------------------------` | | |
| | `          CM10-*` | | |
| | `          Controller Module` | | |
| | `          Software Version: *.**` | | |
| | `          Copyright 2010` | | |
| | `          ORIENTAL MOTOR CO., LTD.` | | |
| | `------------------------------------------` | | |
| | `>DEND` | | #Confirm the new assignment |
| | ` DEND=0(0) [Disable(Disable)]` | | |
| | `>` | | |

## DIN  : Driver General Input Status

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | DIN |
| **Range** | 0 to 127 (integer values)<br>/: real time monitor  (immediate mode only) |
| **Access** | READ |
| **See Also** | DIO, DINx, DOUT, DOUTx, DINxxx (DINALARM), INITDIO, REPORT |
| **Description** | The DIN command displays the current status of all the general purpose inputs on the driver connector of the **CM10**/**SCX10**, as one integer number.<br><br>The general purpose inputs on the driver connector of the **CM10**/**SCX10** contribute to the value of DIN as follows: |

| DINx | Contribution to DIN If Active |
|---|---|
| DIN7 | 64 |
| DIN6 | 32 |
| DIN5 | 16 |
| DIN4 | 8 |
| DIN3 | 4 |
| DIN2 | 2 |
| DIN1 | 1 |

For example, if the driver general input 2 (2), driver general input 3 (4) and driver general input 4 (8) are active, while all other signals are not active, "DIN=14" is set (2+4+8=14).

When inputting DIN, "DIN=14" is replied.

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

∗ If an input is assigned to a system driver input signal (ALARM, END, etc) the DIN command will always read that particular input OFF or 0. Use the DINSG command to read the status of the assigned system input signals on the driver connector of the **CM10**/**SCX10**.

| **Example** | Command | Description |
|---|---|---|
| | >DIN | #Query the status of the general inputs |
| | DIN=20 | #Device response indicating input 3 and 5 are ON |
| | > | |

## DINSG  : Driver System Signal Input Status

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | DINSG |
| **Range** | 0 to 127 (integer values)<br>/: real time monitor  (immediate mode only) |
| **Access** | READ |
| **See Also** | DIN, DOUTSG, DIO, DSIGxxx |

| **Description** | The DINSG command displays the current status of all of the system driver inputs, as one integer number. The system driver inputs contribute to the value of the DINSG as follows: |
|---|---|

| Bit Location | DIN | Contribution to DIN If Active |
|---|---|---|
| Bit 6 | READY | 64 |
| Bit 5 | LC | 32 |
| Bit 4 | MOVE | 16 |
| Bit 3 | TIMD/EXTZ | 8 |
| Bit 2 | TIMS | 4 |
| Bit 1 | END | 2 |
| Bit 0 | ALARM | 1 |

DINSG is the sum of the contribution of all active signals.

For example, if the ALARM (1) and TIMS (4) signals are active, while all other signals are not active, "DINSG=5" is set (1+4=5).

When inputting DIN, "DIN=5" is replied.

∗ When checking the status of a single driver system input signal use the DSIGxxx command.

∗ Be careful not to confuse DINSG with DIN (driver general input status).  DIN reports the status of general purpose inputs (those inputs which are not assigned to a specific signal) on the driver connector of the **CM10/SCX10**.

| **Example** | Command | | Description |
|---|---|---|---|
| | ```
>DIO
 Inputs  (1-7) = ALM IN2 END READY LC TIMS TIMD/EXTZ
 Outputs (1-8) = CON ACL/DCL REQ CS OUT5 OUT6 PRESET FREE

 --Inputs--        --Outputs--
 1 2 3 4 5 6 7 - - 1 2 3 4 5 6 7 8
 1 0 0 0 1 1 0 - - 0 0 0 0 0 0 0 0
>DINSG
 DINSG=5
``` | | #Display DIO status<br>#Device response<br><br><br><br><br><br>#Check DINSG<br>#Device response: the ALM signal, the TIMS signal and LC signal are active |

## DINx : Individual Driver General Input Status

| | |
|---|---|
| **Execution Mode** | Immediate and Sequence |
| **Syntax** | DINx   (INx is a signal: IN1 to IN7) |
| **Range** | n =  0: Not Active<br> 1: Active<br>/: real time monitor  (immediate mode only) |
| **Access** | READ |
| **See Also** | DIN, DIO, DINSG |
| **Description** | DINx returns the state of general purpose input "x" on the driver connector of the **CM10**/**SCX10**.<br><br>If the input on the driver connector has been assigned to a system input signal, such as ALARM input, then it is no longer a "general purpose" input. DINx for these inputs will always return 0 (not active).<br><br>Use the DSIGxxx command to check the status of the system input signals on the driver connector of the **CM10**/**SCX10**. |
| **Example** | Command                                Description |

```
>DIN6                   #Query the status of the individual driver general input 6
 DIN6=1
>
```

## DINxxx : Driver System Signal Input Assignment

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | DINxxx=n<br><br>"xxx" represents the signal name to be assigned, and "n" represents the assigned terminal number ("that" becomes the INn general when unassigned). |
| **SAVEPRM & RESET** | Required to execute any changes made to the parameter value and to save 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 factory setting. |
| **Access** | READ and WRITE (READ only for the command with the parameter range = 0) |
| **See Also** | DSIGxxx, DIO, INITDIO, CLEARALL, DIN, DINSG, DINx and "See Also" column in the chart below. |
| **Description** | Assign the driver system input signal to the INn of the driver connector of the **CM10**/**SCX10**. The driver system signal input assignment is released by "DINxxx=0" and it becomes the driver general input INn. When executing the INITDIO command, the parameter restores to the factory setting. |

| Command | Signal | Description | Factory Setting (n) | Range (n) | See Also |
|---|---|---|---|---|---|
| DINALARM | ALARM | Alarm | 1 | 0, 1: **CM10-1**, **2**, **3**, **4**, **5**<br>0 to 5: **SCX10** | DALARM |
| DINEND | END | Motion End | 0: **CM10-2**<br>2: **SCX10**<br>3: **CM10-1**, **3**, **4**, **5** | 0: **CM10-2**<br>0, 3: **CM10-1**, **3**, **4**, **5**<br>0 to 5: **SCX10** | DEND |
| DINLC | LC | Limiting Condition | 0: **CM10-4**<br>4: **SCX10**<br>5: **CM10-1**, **2**, **3**, **5** | 0: **CM10-4**<br>0, 5: **CM10-1**, **2**, **3**, **5**<br>0 to 5: **SCX10** | LC |
| DINMOVE | MOVE | Motor Moving | 0: **CM10-1**, **2**, **4**, **5**, **SCX10**<br>4: **CM10-3** | 0: **CM10-2**, **4**<br>0, 2: **CM10-1**, **5**<br>0, 4: **CM10-3**<br>0 to 5: **SCX10** | - |
| DINP0 | P0 | Position Data Bit 0 | 0: **CM10-1**, **2**, **4**, **SCX10**<br>5: **CM10-5**<br>6: **CM10-3** | 0: **CM10-2**, **4**<br>0, 5: **CM10-1**, **5**<br>0, 6: **CM10-3**<br>0 to 5: **SCX10** | ABSREQ,<br>ABSREQPC |
| DINP1 | P1 | Position Data Bit 1 | 0: **CM10-1**, **2**, **4**, **SCX10**<br>5: **CM10-3**<br>6: **CM10-5** | 0: **CM10-2**, **4**<br>0, 5: **CM10-3**<br>0, 6: **CM10-1**, **5**<br>0 to 5: **SCX10** | ABSREQ,<br>ABSREQPC |
| DINPR | PR | Position Data Output Ready | 0: **CM10-1**, **2**, **4**, **SCX10**<br>3: **CM10-3**<br>4: **CM10-5** | 0: **CM10-2**, **4**<br>0, 3: **CM10-3**<br>0, 4: **CM10-1**, **5**<br>0 to 5: **SCX10** | ABSREQ,<br>ABSREQPC |
| DINREADY | READY | Operation Ready | 0: **CM10-2**, **3**, **4**<br>4: **CM10-1**, **5**<br>5: **SCX10** | 0: **CM10-2**, **3**, **4**<br>0, 4: **CM10-1**, **5**<br>0 to 5: **SCX10** | DREADY |
| DINTIMDEXTZ | TIMD | Timing Signal·Z-phase Pulse Differential Input | 7 | 0, 7 | TIM, ENC |
| DINTIMS | TIMS | Timing Signal·Z-phase Pulse Single Ended Input | 3: **SCX10**<br>6: **CM10-1**, **2**, **3**, **4**, **5** | 0 to 5: **SCX10**<br>0, 6: **CM10-1**, **2**, **3**, **4**, **5** | TIM, ENC |

| Example | Command | Description |
|---|---|---|
| | >DINALARM=0 | #Unassign the ALARM input |
| | DINALARM=1(0) | |
| | >SAVEPRM | #Save the parameter assignments |
| | (EEPROM has been written 2 times) | |
| | Enter Y to proceed, other key to cancel. Y | |
| | Saving Parameters........OK. | |
| | >RESET | #Establish the saved parameter value |
| | Resetting system. | |
| | ------------------------------------------ | |
| |       CM10-* | |
| |       Controller Module | |
| |       Software Version: *.** | |
| |       Copyright 2010 | |
| |       ORIENTAL MOTOR CO., LTD. | |
| | ------------------------------------------ | |
| | >DINALARM | #Confirm new value |
| | DINALARM=0(0) | |
| | > | |

## DIO : Driver I/O Status

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | DIO |
| **Range** | n = 0: Not Active<br>1: Active<br>/: real time monitor |
| **Access** | READ |
| **See Also** | DIN, DOUT, DINx, DOUTx, DSIGxxx, DINSG, DOUTSG, IO, RIO |
| **Description** | DIO displays the current status of general purpose inputs and outputs on the driver connector of the **CM10**/**SCX10** and assigned system input/output signals on the driver connector of the **CM10**/**SCX10**.<br><br>∗ For TIMD/EXTZ, the status of the signal selected by the ENC command (0: External encoder ZSG, 1: Timing signal・Z-phase pulse differential input) is displayed |

**Example**

| Command | Description |
|---|---|
| `>DIO` | #Display the DIO status |
| ` Inputs  (1-7) = ALM IN2 END READY LC TIMS TIMD/EXTZ` | |
| ` Outputs (1-8) = CON ACL/DCL REQ CS OUT5 OUT6 PRESET FREE` | #Device response |
| | |
| ` --Inputs--        --Outputs--` | |
| ` 1 2 3 4 5 6 7 - - 1 2 3 4 5 6 7 8` | |
| ` 0 0 1 1 0 0 1 - - 1 0 0 0 0 0 0 0` | |
| `>` | |

## DIR  : Sequence Directory

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | DIR target |
| **Range** | target is optional. If given, it should be a valid sequence number or name |
| **See Also** | COPY, EDIT, REN |
| **Description** | 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. |
| **Example** | Command             Description |

```
>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  6144 bytes total,   11 percent.
  Storage memory:      2259 bytes used of 21775 bytes total,   10 percent.
>
>DIR RESYNC        #List directory information for one sequence only


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

  Executable memory:    690 bytes used of  6144 bytes total,   11 percent.
  Storage memory:      2259 bytes used of 21775 bytes total,   10 percent.
>
```

## DIRINV : Direction Invert

| | |
|---|---|
| **Execution Mode** | Immediate and Sequence |
| **Syntax** | DIRINV n |
| **Range** | n = 0: Motor rotates in the Clockwise (CW) direction for positive distance values.<br>    1: Motor rotates in the Counter-Clockwise (CCW) direction for positive distance values. |
| **Factory Setting** | 0 |
| **SAVEPRM & RESET** | Required to execute any changes made to the parameter value and to save 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 factory setting. |
| **Access** | READ and WRITE<br>READ only in Sequences |
| **See Also** | DIS, MA, MCP, MCN, MGHP, MGHN, MI, EHOME |
| **Description** | Inverts the direction of motor rotation.<br><br>∗ When using a gearhead, the direction of the gearhead output shaft may rotate in the opposite direction of the motor's rotation. |

| **Example** | Command | Description |
|---|---|---|
| | ```
>DIRINV 1
 DIRINV 0(1)
>SAVEPRM
 (EEPROM has been written 21 times)
 Enter Y to proceed, other key to cancel. y
 Saving Parameters........OK.
>RESET
 Resetting system.
-------------------------------------------
          CM10-*
          Controller Module
          Software Version: *.**
          Copyright 2010
          ORIENTAL MOTOR CO., LTD.
-------------------------------------------
>DIS 1000
 DIS=1000 rev
>MI
>
``` | #Invert the motor direction<br>#Device response<br>#Save the parameter assignments<br><br><br><br><br><br>#Execute a RESET operation to activate the saved parameters<br><br><br><br><br><br><br><br><br>#Set the distance value<br>#Device response<br>#The motor rotates 1000 in the CCW direction |

## DIS : Distance for Incremental Motion

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | DIS=n |
| **Range** | n = -MAXPOS to +MAXPOS (user units) |
| **Factory Setting** | 0 |
| **SAVEPRM** | 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 factory setting. |
| **Access** | READ and WRITE |
| **See Also** | MI, MAXPOS, TA, TD, VS, VR, CV, DIRINV, DPR, MA |
| **Description** | Determines the distance to be moved for the MI (move incremental) command. The sign of DIS determines the direction of motion. |

| **Example** | Command | Description |
|---|---|---|
| | ```
>DPR
 DPR=1(1) Rev
 Position range = +/- 500000(500000)
 Velocity range = 0.001 - 2480(2480)
>DIS 2000
 DIS=2000 Rev
>MI
>DIS -2000
 DIS=-2000 Rev
>MI
>
``` | #Query the DPR value<br>#Device response<br>#Device response<br>#Device response<br>#Set distance to 2000 user units in the positive direction<br>#Execute the index move<br>#Set distance to 2000 user units in the negative direction<br>#Execute the index move |

## DISx : Distance or Destination for Link Segment 'x'

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | DISx=n  (x is a number of linked segments: x=0 to 3.) |
| **Range** | n = -MAXPOS to +MAXPOS (user units) |
| **Factory Setting** | 0 |
| **SAVEPRM** | 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 factory setting. |
| **Access** | READ and WRITE |
| **See Also** | INCABSx, MIx, LINKx, VRx |
| **Description** | 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. |

| **Example** | Command | Description |
|---|---|---|
| | >UU in | #Set user units to in. (inches) |
| | UU=in | #Device response |
| | >VR1 5 | #Set the velocity for linked move 1 to 5 user units/s |
| | VR1=5 in/sec | #Device response |
| | >DIS1 10 | #Set the distance for linked move 1 to 10 user units |
| | DIS1=10 in | #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 10 | #Linked move 2 velocity equals 10 user units/s |
| | VR2=10 in/sec | #Device response |
| | >INCABS2 0 | #Set the move type for linked motion 2 to absolute |
| | INCABS2=0 [ABS] | #Device response |
| | >DIS2 20 | #Linked move 2: destination is position 20 user units |
| | DIS2=20 in | #Device response |
| | >LINK2 0 | #"Unlink" link 2 from link 3 |
| | LINK2=0 | #Device response |
| | >MI1 | #Start the linked operation motion |
| | > | |

# DOUT  : Driver General Output Control

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | DOUT=n |
| **Range** | n = 0 to 255 (integer values)<br>/: real time monitor  (immediate mode only) |
| **Access** | READ and WRITE<br>READ only in CANopen |
| **See Also** | DIO, DOUTx, DIN, DINx, DSIGxxx, DOUTSG, REPORT |
| **Description** | The DOUT command displays or controls the current status of all the general purpose outputs on the driver connector of the **CM10**/**SCX10**, as one integer number.<br><br>The general purpose outputs on the driver connector of the **CM10**/**SCX10** contribute to the value of DOUT as follows: |

| DOUTx | Contribution to DOUT If Active |
|---|---|
| DOUT8 | 128 |
| DOUT7 | 64 |
| DOUT6 | 32 |
| DOUT5 | 16 |
| DOUT4 | 8 |
| DOUT3 | 4 |
| DOUT2 | 2 |
| DOUT1 | 1 |

For example, if the driver general output 2 (2), driver general output 3 (4) and driver general output 4 (8) are active, while all other signals are not active, "DOUT=14" is set (2+4+8=14). When inputting DOUT, "DOUT=14" is replied.

And when inputting "DOUT=14," the driver general output 2 (2), driver general output 3 (4) and driver general output 4 (8) become active regardless of the present output status.

∗ To check or control  the status of a single general output, use the DOUTx command.

∗ The DOUT value always indicates the internal status of the all driver general output signals DOUT1 to DOUT8.  For the output signal assigned the system output signal (CON, FREE etc.), the DOUT command recognizes the signal as inactive or zero (except when the signal has become active by the DOUT, DOUTx command).  Use the DOUTSG command in order to refer to the status of all output signals assigned the specific function.

| **Example** | Command | Description |
|---|---|---|
| | ```
>DIO
 Inputs  (1-7) = ALM IN2 END READY LC TIMS TIMD/EXTZ
 Outputs (1-8) = CON ACL/DCL REQ CS OUT5 OUT6 PRESET FREE

 --Inputs--      --Outputs--
 1 2 3 4 5 6 7 - - 1 2 3 4 5 6 7 8
 0 0 0 0 0 0 0 - - 0 0 1 0 0 1 0 0
>DOUT
 DOUT=32
>DOUT=16
 DOUT=16
>DIO
 Inputs  (1-7) = ALM IN2 END READY LC TIMS TIMD/EXTZ
 Outputs (1-8) = CON ACL/DCL REQ CS OUT5 OUT6 PRESET FREE

 --Inputs--      --Outputs--
 1 2 3 4 5 6 7 - - 1 2 3 4 5 6 7 8
 0 0 0 0 0 0 0 - - 1 0 0 0 1 0 0 0
>
``` | #Display the DIO status<br>#Device response<br><br><br><br><br>#Check DOUT Status<br>#DOUT6 is active<br>#DOUT5 to be active<br>#Device response |

## DOUTSG  : Driver System Signal Output Status

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | DOUTSG |
| **Range** | 1 to 16382 (integer values) |
| | /: real time monitor  (immediate mode only) |
| **Access** | READ |
| **See Also** | DOUT, DINSG, DIO, DSIGxxx, REPORT |
| **Description** | The DOUTSG command displays the current status of all the system driver outputs, as one integer number. |

The driver outputs contribute to the value of DOUTSG as follows:

| Bit Location | Signal Name of the Driver Connector on the **CM10/SCX10** | Contribution to DOUTSG If Active |
|---|---|---|
| Bit 13 | M2 | 8192 |
| Bit 12 | M1 | 4096 |
| Bit 11 | M0 | 2048 |
| Bit 10 | TL | 1024 |
| Bit 9 | REQ | 512 |
| Bit 8 | HMSTOP | 256 |
| Bit 7 | PRESET | 128 |
| Bit 6 | HOME | 64 |
| Bit 5 | MBFREE | 32 |
| Bit 4 | FREE | 16 |
| Bit 3 | CS | 8 |
| Bit 2 | ACL/DCL | 4 |
| Bit 1 | COFF | 2 |
| Bit 0 | CON | 1 |

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

For example, if the CON (1) and CS (8) signals are active, while all other signals are not active, "DOUTSG=9" is set (1+8=9). When inputting DOUTSG, "DOUTSG=9" is replied.

∗ When checking the status of a single driver system output signal, use the DSIGxxx command.

∗ The DOUTSG value always indicates the status of the all driver system output signals.  Note that the signals, which are not output actually due to not assigned to the driver connector terminal, are also counted in the DOUTSG value.

∗ Be careful not to confuse DOUTSG with DOUT (driver general output control).  DOUT reports the status of general purpose outputs (those outputs which are not assigned to a specific signal) on the driver connector of the **CM10**/**SCX10**.

| **Example** | Command | | Description |
|---|---|---|---|

```
>DIO                                                    #Display the DIO status
 Inputs  (1-7) = ALM IN2 END READY LC TIMS TIMD/EXTZ    #Device response
 Outputs (1-8) = CON ACL/DCL REQ CS OUT5 OUT6 PRESET FREE

 --Inputs--        --Outputs--
 1 2 3 4 5 6 7 - - 1 2 3 4 5 6 7 8
 1 0 0 0 0 0 0 - - 1 0 1 0 1 0 0 1
>DOUTSG                                                 #Check DOUTSG
 DOUTSG=529                                             status
```

## DOUTx  : Individual Driver General Output Control

| | |
|---|---|
| **Execution Mode** | Immediate and Sequence |
| **Syntax** | DOUTx=n        (OUTx is signal name: OUT1 to OUT8)<br><br>∗ "=n" is required only when controlling. |
| **Range** | n = 0: Not Active<br>        1: Active<br><br>/: real time monitor  (immediate mode only) |
| **Factory Setting** | 0 |
| **Access** | READ and WRITE |
| **See Also** | INIDTIO, DOUT, DOUTSG, DIO, DIN,  DINSG, DINx, DSIGxxx, REPORT |
| **Description** | DOUTx displays or controls the state of general purpose output 'x' on the driver connector of the **CM10**/**SCX10**.<br><br>If the output on the driver connector has been assigned to a specific system output signal such as CS, then it is no longer a "general purpose" output. DOUTx for these outputs will always return 0 (not active). Use the DSIGxxx command to check the status of the assigned system output signal on the driver connector of the **CM10**/**SCX10**. |
| **Memo** | Even the output signal assigned the system output signal can become active (1) as the general driver output status using the DOUTx command.  For example, even when the OUT1 of the driver connector is assigned to CON, DOUT1=1 can be commanded.  However, the actual output terminal of the driver connector that is assigned to CON will not become active.  Note that the DOUT (driver general output control) value becomes 1 at this time. |

| **Example** | Command | Description |
|---|---|---|
| | ```
>DOUT5=1
 DOUT5=1
>DIO
 Inputs   (1-7) = ALM IN2 END READY LC TIMS TIMD/EXTZ
 Outputs (1-8) = CON ACL/DCL REQ CS OUT5 OUT6 PRESET FREE

 --Inputs--        --Outputs--
 1 2 3 4 5 6 7 - - 1 2 3 4 5 6 7 8
 0 0 1 1 0 0 1 - - 1 0 0 0 1 0 0 0
>DOUT5=0
 DOUT5=0
>DIO
 Inputs   (1-7) = ALM IN2 END READY LC TIMS TIMD/EXTZ
 Outputs (1-8) = CON ACL/DCL REQ CS OUT5 OUT6 PRESET FREE

 --Inputs--        --Outputs--
 1 2 3 4 5 6 7 - - 1 2 3 4 5 6 7 8
 0 0 1 1 0 0 1 - - 1 0 0 0 0 0 0 0
``` | #Driver general output 5 active<br>#Display DIO status<br>#Device response<br><br><br><br><br><br>#Driver general output 5 inactive<br>#Display DIO status<br>#Device response |

# DOUTxxx : Driver System Signal Output Assignment

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | DOUTxxx=n<br><br>"xxx" represents the signal name to be assigned, and "n" represents the assigned terminal number ("that" becomes the OUTn general output when unassigned). |
| **SAVEPRM & RESET** | Required to execute any changes made to the parameter value and to save 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 factory setting. |
| **Access** | READ and WRITE (READ only for the command with the parameter range = 0) |
| **See Also** | DSIGxxx (except: ACLDCL, HMSTOP, HOME), DIO, INITDIO, DOUT, DOUTSG, DOUTx, and "See Also" column in the chart below. |
| **Description** | Assign the driver system output signal to the OUTn of the driver connector of the **CM10**/**SCX10**.  The driver system signal output assignment is released by "DOUTxxx=0" and it becomes the driver general output OUTn.  When executing the INITDIO command, the parameter restores to the factory setting. |

| Command | Signal | Description | Factory Setting (n) | Range (n) | See Also |
|---|---|---|---|---|---|
| DOUTACLDCL | ACL/DCL | Driver Alarm Clear /Deviation Counter Clear | 0: **CM10-2**<br>2: **CM10-1**, **3**, **4**, **5**<br>3: **SCX10** | 0: **CM10-2**<br>0, 2: **CM10-1**, **3**, **4**, **5**<br>0 to 8: **SCX10** | ALMCLR, PECLR |
| DOUTCK | CK | Position Data Transmission Clock | 0: **CM10-1**, **2**, **4**, **SCX10**<br>2: **CM10-3**, **5** | 0: **CM10-2**, **4**<br>0, 2: **CM10-1**, **3**, **5**<br>0 to 8: **SCX10** | ABSREQ, ABSREQPC |
| DOUTCOFF | COFF | Current OFF | 0: **CM10-1**, **5**<br>1: **CM10-2**, **3**, **4**, **SCX10** | 0: **CM10-1**, **5**<br>0, 1: **CM10-2**, **3**, **4**<br>0 to 8: **SCX10** | DOUTCON, CURRENT, FREE, DOUTFREE |
| DOUTCON | CON | Current ON | 0: **CM10-2**, **3**, **4**<br>1: **CM10-1**, **5**<br>2: **SCX10** | 0: **CM10-2**, **3**, **4**<br>0, 1: **CM10-1**, **5**<br>0 to 8: **SCX10** | DOUTCOFF, CURRENT, FREE, DOUTFREE |
| DOUTCS | CS | Resolution Selection | 0: **CM10-3**, **5**<br>4: **CM10-1**, **2**, **4**, **SCX10** | 0: **CM10-3**, **5**<br>0, 4: **CM10-1**, **2**, **4**<br>0 to 8: **SCX10** | STRDCS |
| DOUTFREE | FREE | Current  OFF, Magnetic Brake  Free | 0: **CM10-2**, **4**<br>5: **SCX10**<br>8: **CM10-1**, **3**, **5** | 0: **CM10-2**, **4**<br>0, 8: **CM10-1**, **3**, **5**<br>0 to 8: **SCX10** | FREE, CURRENT |
| DOUTHMSTOP | HMSTOP | Sensor-less Home Seeking Operation Stop | 0: **CM10-1**, **2**, **4**, **5**, **SCX10**<br>5: **CM10-3** | 0: **CM10-1**, **2**, **4**, **5**<br>0, 5: **CM10-3**<br>0 to 8: **SCX10** | HOMETYP, ABORT, PSTOP, HSTOP, MSTOP, SSTOP, PAUSE |
| DOUTHOME | HOME | Sensor-less Home Seeking Operation Start | 0: **CM10-1**, **2**, **4**, **5**, **SCX10**<br>7: **CM10-3** | 0: **CM10-1**, **2**, **4**, **5**<br>0, 7: **CM10-3**<br>0 to 8: **SCX10** | HOMETYP, MGHP, MGHN |
| DOUTMBFREE | MBFREE | Magnetic Brake Free | 0: **CM10-1**, **2**, **3**, **4**, **5**<br>6: **SCX10** | 0: **CM10-1**, **2**, **3**, **4**, **5**<br>0 to 8: **SCX10** | FREE, CURRENT, OUTMBFREE, ROUTMBFREE |
| DOUTM0 | M0 | Data Select Bit 0 | 0: **CM10-1**, **2**, **3**, **4**, **SCX10**<br>5: **CM10-5** | 0: **CM10-2**, **3**, **4**<br>0, 5: **CM10-1**, **5**<br>0 to 8: **SCX10** | DD, TL |

| Command | Signal | Description | Factory Setting (n) | Range (n) | See Also |
|---|---|---|---|---|---|
| DOUTM1 | M1 | Data Select Bit 1 | 0: **CM10-1**, **2**, **3**, **4**, **SCX10**<br>6: **CM10-5** | 0: **CM10-2**, **3**, **4**<br>0, 6: **CM10-1**, **5**<br>0 to 8: **SCX10** | DD, TL |
| DOUTM2 | M2 | Data Select Bit 2 | 0 | 0: **CM10-2**, **3**, **4**, **5**<br>0, 7: **CM10-1**<br>0 to 8: **SCX10** | DD, TL |
| DOUTPRESET | PRESET | Reset Home Position | 0: **CM10-1**, **2**, **3**, **4**, **SCX10**<br>7: **CM10-5** | 0: **CM10-2**, **4**<br>0, 7: **CM10-1**, **3**, **5**<br>0 to 8: **SCX10** | PRESET, ABSREQ, ABSREQPC |
| DOUTREQ | REQ | Position Data Transmission Request | 0: **CM10-1**, **2**, **4**, **SCX10**<br>3: **CM10-3**, **5** | 0: **CM10-2**, **4**<br>0, 3: **CM10-1**, **3**, **5**<br>0 to 8: **SCX10** | ABSREQ, ABSREQPC, ABSPLSEN |
| DOUTTL | TL | Torque Limiting /Push-motion Operation /Current Cutback Release | 0: **CM10-1**, **2**, **3**, **4**<br>4: **CM10-5**<br>7: **SCX10** | 0: **CM10-2**, **3**, **4**<br>0, 4: **CM10-1**, **5**<br>0 to 8: **SCX10** | TL |

| **Example** | Command | Description |
|---|---|---|
| | `>DOUTACLDCL=0` | #Unassign the ACL/DCL output |
| | ` DOUTACLDCL=2(0)` | |
| | `>SAVEPRM` | #Save the parameter assignments |
| | ` (EEPROM has been written 80 times)` | |
| | ` Enter Y to proceed, other key to cancel. Y` | |
| | ` Saving Parameters........OK.` | |
| | `>RESET` | #Establish the saved parameter values |
| | ` Resetting system.` | |
| | `-------------------------------------------` | |
| | `        CM10-*` | |
| | `        Controller Module` | |
| | `        Software Version: *.**` | |
| | `        Copyright 2010` | |
| | `        ORIENTAL MOTOR CO., LTD.` | |
| | `-------------------------------------------` | |
| | `>DOUTACLDCL` | #Confirm the new assignment |
| | ` DOUTACLDCL=0(0)` | |
| | `>` | |

## DPR  : Distance per Revolution

| | |
|---|---|
| **Execution Mode** | Immediate and Sequence |
| **Syntax** | DPR=n |
| **Range** | n =  0.5 00 to 51200.000 (user units per revolution) |
| **Factory Setting** | 1 |
| **SAVEPRM & RESET** | Required to execute any changes made to the parameter value and to save 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 factory setting. |
| **Access** | READ and WRITE<br>READ only in Sequences |
| **See Also** | GA, GB, UU, MR, MAXPOS, MAXVEL |
| **Description** | The value of DPR sets the distance per revolution in terms of user units (mm, degrees, etc.).<br><br>DPR allows programming all distances, positions and velocities in terms of real world units.<br><br>For instance, a lead screw with a lead of 10 millimeters per revolution may use a DPR value of 10.  When setting the DPR parameter, concurrently be sure to set the user unit by the UU command and the motor resolution by the MR command.<br><br>Once they are set, all of operation parameters such as the distance, target position, velocity and others can be commanded by user units (this example is mm, mm/sec).  (The motor resolution or step angle no longer be used.)<br>DPR also effects the minimum and maximum numeric range of many variables. In particular, it effects position range limit MAXPOS, velocity range limit MAXVEL. |
| **Note** | If DPR is changed, MAXPOS (maximum position=position range), MAXVEL (maximum velocity=velocity range) or minimum travel distance (minimum movable distance) may automatically be changed. When executing the DPR command, these pre-change and post-change values are shown.  Position parameters (DIS, DISx, OFFSET, SCHGPOS, LIMP, LIMN, ENDACT) and velocity parameters (VS, VR, VRx, SCHGVR) are automatically checked whether or not to be within the MAXPOS (maximum position) and MAXVEL (maximum velocity) respectively, and if they are outside the range, the warning message will be sent.  Note that the position/velocity parameters in the sequence program and the position array data POS[x] will not be checked. Check whether the required resolution is obtained. The minimum movable distance, which is the actual travel distance, can be checked by the TEACH command.<br><br>If DPR is changed, the actual travel distance or velocity is changed.  Check whether the present settings of the position/velocity parameters or those values in the program are appropriate.<br><br>If electronic gearing is used (GA/GB≠1), DPR reflects the distance moved, in user units, at the output of a hypothetical gear train with ratio GA/GB.  The actual motor shaft (rotor shaft) will rotate GA/GB times this distance. |

| Example | Command | Description |
|---|---|---|
| | >UU mm | #Set the user units to mm (millimeters) |
| |  UU=mm | |
| | >DPR 10 | #Set the distance per revolution to 10 |
| |  DPR=1(10) mm | user units, device responds with |
| |  Position range = +/- 500000(500000) | rescaled limits and new ranges |
| |  Velocity range = 0.001 - 2480(24800) | |
| |  Minimum Movable Distance = +/- 0.001(0.001) | |
| | >SAVEPRM | #Save the parameter assignments |
| |  (EEPROM has been written 21 times) | |
| |  Enter Y to proceed, other key to cancel. y | |
| |  Saving Parameters........OK. | |
| | >RESET | #Establish the saved parameter values |
| |  Resetting system. | |
| | ------------------------------------------- | |
| |       CM10-* | |
| |       Controller Module | |
| |       Software Version: *.** | |
| |       Copyright 2010 | |
| |       ORIENTAL MOTOR CO., LTD. | |
| | ------------------------------------------- | |
| | >MAXPOS | #Query the maximum position value |
| |  MAXPOS=500000(500000) mm | |
| | >MAXVEL | #Query the maximum velocity value |
| |  MAXVEL=24800(24800) mm/sec | |
| | | |
| | > | |

## DREADY : Driver READY Signal Enable

| | |
|---|---|
| **Execution Mode** | Immediate and Sequence |
| **Syntax** | DREADY=n |
| **Range** | n = 0: Disable<br>    1: Enable |
| **Factory Setting** | 0: **CM10-2**, **3**, **4**, **SCX10**<br>1: **CM10-1**, **5** |
| **SAVEPRM & RESET** | Required to execute any changes made to the parameter value and to save 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 factory setting. |
| **Access** | READ and WRITE<br><br>READ only in Sequences |
| **See Also** | DINxxx (DINREADY), DSIGxxx (DSIGREADY), OUTxxx (OUTREADY) |
| **Description** | Set "enable" or "disable" for the READY (driver operation ready) input signal of the driver connector on the **CM10**/**SCX10**.<br><br>When set to "enable":<br><br>1. When operation is commanded starting, check the driver READY (operation ready). When it is ON, operation (pulse generation) is started immediately. When it is OFF, operation will be started after turning ON. If it is not turned ON after three seconds, an alarm will generate (6Fh=driver connection error).<br><br>2. When the **CM10** or **SCX10** is ready to operate (other than MOVE, RUN and ALARM status) while the driver is ready to operate, the READY signal on the I/O connector (including remote I/O) will be output.<br><br>When set to "disable":<br><br>When the **CM10** or **SCX10** is ready to operate (other than MOVE, RUN and ALARM status), the READY signal on the I/O connector (including remote I/O) will be output. |

| **Example** | Command | Description |
|---|---|---|
| | >DREADY=0<br> DREADY=1(0) [Enable(Disable)]<br>>SAVEPRM<br> (EEPROM has been written 80 times)<br> Enter Y to proceed, other key to cancel. Y<br> Saving Parameters........OK.<br>>RESET<br> Resetting system.<br>------------------------------------------<br>        CM10-*<br>        Controller Module<br>        Software Version: *.**<br>        Copyright 2010<br>        ORIENTAL MOTOR CO., LTD.<br>------------------------------------------<br>>DREADY<br> DREADY=0(0) [Disable(Disable)]<br>> | #Change to "DREADY=0"<br><br>#Save the parameter assignments<br><br><br><br>#Establish the saved parameter value<br><br><br><br><br><br><br><br><br>#Confirm new value |

## DSIGxxx : Status for Driver System Input Signal/Driver System Output Signal

| | |
|---|---|
| **Execution Mode** | Immediate and Sequence |
| **Syntax** | DSIGxxx      ("xxx" indicate the signal name on the driver connector. of the **CM10**/**SCX10**. ex: DSIGALARM) |
| **Range** | 0: "xxx" input/output is not active<br>1: "xxx" input/output is active<br>/: real time monitor (immediate mode only) |
| **Access** | READ |
| **See Also** | DINxxx/DOUTxxx, DIO, and "See Also" column in the chart below. |
| **Description** | DSIGxxx is the status of system "xxx" input/output (included EXTZ input signal of the external encoder connector) signal on the driver connector of the **CM10**/**SCX10**. DSIGxxx becomes "0 (zero)" when not active, while DSIGxxx becomes "1" when active. Only the internal status of system "xxx" driver output signal is shown even when it is not assigned. |

&lt;Input&gt;

| Command | Signal | Description | Controller | See Also |
|---|---|---|---|---|
| DSIGALARM | ALARM | Alarm | **CM10-1**, **2**, **3**, **4**, **5**, **SCX10** | ALARM, DALARM |
| DSIGEND | END | Motion End | **CM10-1**, **3**, **4**, **5**, **SCX10** | END, DEND, MEND, ENDACT |
| DSIGLC | LC | Limiting Condition | **CM10-1**, **2**, **3**, **5**, **SCX10** | TL |
| DSIGMOVE | MOVE | Motor Moving | **CM10-1**, **3**, **5**, **SCX10** | - |
| DSIGREADY | READY | Operation Ready | **CM10-1**, **5**, **SCX10** | DREADY |
| DSIGTIMDEXTZ | TIMD | Timing Signal·Z-phase Pulse Differential Input | **CM10-1**, **2**, **3**, **4**, **5**, **SCX10** | TIM, ENC |
| DSIGTIMS | TIMS | Timing Signal·Z-phase Pulse Single Ended Input | **CM10-1**, **2**, **3**, **4**, **5**, **SCX10** | TIM, ENC |

DSIGTIMDEXTZ is the status of system TIMD/EXTZ (timing) input signal.

The DSIGTIMDEXTZ indicates the timing signal TIMD on the driver connector of the **CM10**/**SCX10** or the Z signal on the encoder connector, as selected by the ENC parameter.

Signal Flow Path

Timing signal·Z-phase pulse differential input  TIMD-----1

                                 **ENC**--------DSIGTIMDEXTZ

External encoder Z                   EXTZ-----2

&lt;Output&gt;

| Command | Signal | Description | Controller | See Also |
|---|---|---|---|---|
| DSIGCOFF | COFF | Current OFF | **CM10-1**, **2**, **3**, **4**, **5**, **SCX10** | CURRENT |
| DSIGCON | CON | Current ON | **CM10-1**, **2**, **3**, **4**, **5**, **SCX10** | CURRENT |
| DSIGCS | CS | Resolution Selection | **CM10-1**, **2**, **4**, **SCX10** | STRDCS |
| DSIGFREE | FREE | Current OFF, Magnetic Brake Free | **CM10-1**, **3**, **5**, **SCX10** | FREE, CURRENT |
| DSIGMBFREE | MBFREE | Magnetic Brake Free | **SCX10** | FREE, CURRENT |
| DSIGM0 | M0 | Data Select Bit 0 | **CM10-1**, **5**, **SCX10** | DD, TL |
| DSIGM1 | M1 | Data Select Bit 1 | **CM10-1**, **5**, **SCX10** | DD, TL |
| DSIGM2 | M2 | Data Select Bit 2 | **CM10-1**, **5**, **SCX10** | DD, TL |
| DSIGTL | TL | Torque Limiting/Push-motion Operation /Current Cutback Release | **CM10-1**, **5**, **SCX10** | TL |

| Example | Command | Description |
|---|---|---|
| | ( 1) CURRENT 1 | #Motor current ON |
| | ( 2) WHILE (DSIGREADY!=1);WEND | #Wait for driver READY signal ON |
| | ( 3) EHOME | #Move to position zero |
| | ( 4) MEND | #Wait for motion to complete |

| | ( 1) CURRENT 1 | #Motor current ON |
|---|---|---|
| | ( 2) WHILE (DSIGREADY!=1);WEND | #Wait for driver READY signal ON |
| | ( 3) EHOME | #Move to position zero |
| | ( 4) MEND | #Wait for motion to complete |

## EC : Encoder Count

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | EC |
| **Range** | -MAXEC to +MAXEC<br>/: real time monitor  (immediate mode only) |
| **Factory Setting** | 0 |
| **Access** | READ, WRITE<br>READ only while motion is in progress. |
| **See Also** | PF, ENC, ER, MR, PC, PE, ENDACT |
| **Description** | EC is the value of the encoder counter, created from the A and B signals in the encoder. The EC is converted into user unit and used as the PF (feedback position) value.<br><br>EC get changed when PF is changed. Changing PC also affects EC since PF get changed when PC is changed.<br><br>The driver encoder signal can be used with the **CM10-1**, **3**, **4**, **5** and **SCX10**.  The ENC command selects use of either driver encoder or an external encoder. See "8.9 Encoder Function" on page 106 for more details. |
| **Example** | Command                    Description<br>>EC              #Query the encoder count<br> EC=-2147483647   #Device response |

# ECHO : Communications Echo Control

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | ECHO=n |
| **Range** | n = 0: OFF, Commands are suppressed and not shown on the terminal<br>   1: ON, Commands are echoes back to the terminal |
| **Factory Setting** | 1 |
| **SAVEPRM** | 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 factory setting. |
| **Access** | READ and WRITE |
| **See Also** | VERBOSE |
| **Description** | 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. The ECHO command defines the device's echo back setting (ON/OFF) for the user entered ASCII data on the terminal. If ECHO=0 (OFF), the device will send no response for the entered ASCII data to the terminal. |
| | 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. |

| Example | Command | Description |
|---|---|---|
| | `>ECHO 1` | #Turn ON the ECHO |
| | ` ECHO=1` | #Device response |
| | `>VS` | #Query the starting velocity |
| | ` VS=0.1 rev/sec` | #Device response |
| | `>ECHO 0` | #Turn OFF the ECHO |
| | ` ECHO=0` | #Device response |
| | `>VS=0.1 rev/sec` | #Query the starting velocity. Again, query doesn't show: just |
| | `>` | response. |

## EDIT  : Edit Sequence

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | EDIT target |
| **Range** | target (optional): any valid sequence number (0-99) or name (consisting of letters, numbers or underscore, 10 characters maximum, must start with a letter except n, s, N, S, with no distinction of a capital letter or small letter.) |
| **Commands not Allowed** | RUN |
| **See Also** | DIR, COPY, DEL, REN, LOCK, UNLOCK |
| **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 sequence program cannot be edited when it is locked.  When an edit is required, edit the sequence program after releasing the lock of the sequence program using the UNLOCK command.

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 |
| 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 (0-99) |
| S sss | Save sequence, by name (10 char max) |
| M | Display memory status |
| H | Display this help reminder |
| Q | Quit sequence editor |

| | |
|---|---|
| **Note** | ・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     : 6144 | #Device response |
| | >>Command: | #<ESC> is sent to exit the editor |
| | > | #Back to the main system prompt |

# EHOME  : Start Return-to-electrical Home Operation

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | EHOME |
| **Commands not Allowed** | MOVE |
| **See Also** | HOMETYP, LIMP, LIMN, MGHP, MGHN, PC, SLACT |
| **Description** | EHOME starts an absolute motion to position 0 (PC=0) |
| | The motion caused by EHOME is equivalent to an "MA 0" command (move to position zero), but there is one important difference: If the electrical home has not been set, EHOME establishes position 0 as the "home" position, and can make software limit checking possible. |
| | The EHOME motion is defined by stop velocity VS, running velocity VR, and acceleration and deceleration times TA and TD. |
| | EHOME will not execute while the motor is moving. The motor must come to a stop before an EHOME operation will execute. The EHOME function will not execute while the MOVE output is ON. |
| | Software position limit checking is configured by setting appropriate values for positive and negative software position limits (LIMP, LIMN), requesting software position limit checking (setting SLACT=1), and establishing a valid home position.  Software position limit checking does not start until a valid home position has been established. If limit sensors and a home input are used, this can be done with mechanical home seeking (using MGHP or MGHN).  If an application uses some other means to establish home, EHOME is required as part of the process of enabling software position limit checking. |
| **Note** | EHOME is a "starting" command for return-to-electrical home operation.  Therefore, just after commanding (without waiting for the end of motion), the prompt will be displayed with the immediate command and the sequence will proceed to next command (line) with the sequence execution. |
| | The commands except motion commands can be executed during motion, but all of the motion commands cannot be executed until the motion completes.  To check the motion completion, execute the SIGMOVE command or SIGEND command, and monitor.  It is convenient to use the MEND (wait for motion end) command in the sequence program. |

**Example**

| Command | Description |
|---|---|
| >EHOME | #Initiates the motor moving to the EHOME (PC=0) location |
| >LIST 6 | #List use entered sequence 6 |
| | |
| (  1)  EHOME | #Execute an EHOME operation |
| (  2)  MEND | #Wait for motion to end |
| (  3)  DIS=10 | #Distance equals 10 user units |
| (  4)  MI | #Move incremental |
| (  5)  MEND | #Wait for motion to end |
| (  6)  END | #End the sequence |
| > | |

## ELSE  : Begin ELSE Block: execute if IF is false

| | |
|---|---|
| **Execution Mode** | Sequence |
| **Syntax** | ELSE |
| **See Also** | IF, ENDIF, WHILE, WEND |
| **Description** | Branches to an alternate operation if the preceding conditional IF statement is not true. |
| **Example** | |

| Command | Description |
|---|---|
| >LIST 5 | #List sequence 5 |
| | |
| (  1) IF (IN1=1) | #If input #1 is ON, then do line 2 |
| (  2)   VR=2 | #Running velocity=2 user units/second |
| (  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 |
| > | |

## ENC  : Encoder Selection

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence |
| **Syntax** | ENC=n |
| **Range** | n =  0: Not Used<br>        1: Driver Encoder<br>        2: External Encoder |
| **Factory Setting** | 0: **CM10-2**, **SCX10**<br>1: **CM10-1**, **3**, **4**, **5** |
| **SAVEPRM & RESET** | Required to execute any changes made to the parameter value and to save 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 factory setting. |
| **Access** | READ and WRITE<br>READ only in Sequences |
| **See Also** | EC, PF, PE, TIM |
| **Description** | Select the source of EC (encoder count) signal, either external encoder or the driver encoder.  The EC is then converted into user unit and used as the PF (feedback position) value. |

 Signal Flow Path

Driver Encoder  A, B-------1

$$\textbf{ENC}\text{---}\rightarrow \text{EC} \text{ --}\rightarrow \text{PF value}$$

External Encoder A, B------2

∗ When ENC is set to zero (0), the EC (encoder count) value will always be zero. ENC=0 is used when a indication of PF=0 is desired such as when using the utility software and always indicates the PF value. (If ENC=1 or 2, even though an encoder is not connected, the PF can be nonzero when PC is intentionally changed.) While the A and B signals in the encoder is used for the PF, the Z  (zero position) signal is used as the TIMING signal for a mechanical home seeking.  The driver timing signal TIMD and the Z signal in the external encoder are selected by the ENC parameter, and then the selected output is sent to TIM parameter.

TIMING Source Selection

| TIMING Source | ENC | TIM |
|---|---|---|
| Timing signal・Z-phase pulse differential input<br>TIMD | 1<br>(Driver) | 0<br>(TIMD/EXTZ) |
| Timing signal・Z-phase pulse single ended input<br>TIMS | Unrelated | 1<br>(TIMS) |
| External encoder ZSG<br>EXTZ | 2<br>(External Encoder) | 0<br>(TIMD/EXTZ) |
| No source is selected* | 0<br>(Not Used) | 0<br>(TIMD/EXTZ) |

 ∗ If ENC is set to 0 (zero) and TIM is set to 0, the system alarm status will be active when executing MGHP or MGHN command when the home seeking type uses the timing signal.

Signal Flow Path

Timing signal・Z-phase pulse differential input    TIMD------1

ENC--------0 (TIMD/EXTZ)

External encoder ZSG                              EXTZ------2        **TIM**--------------→TIMING signal

Timing signal・Z-phase pulse single ended input  TIMS------------------1 (TIMS)

| Example | Command | Description |
|---|---|---|
| | >ENC=1 | #Changing the ENC |
| | ENC=0(1) [Not use(Driver)] | #Device response |
| | >SAVEPRM | #Save the parameter assignments |
| | (EEPROM has been written 21 times) | #Device response |
| | Enter Y to proceed, other key to cancel. y | |
| | Saving Parameters........OK. | |
| | >RESET | #Establish the saved parameter |
| | Resetting system. | values |
| | -------------------------------------------- | |
| |        CM10-* | |
| |        Controller Module | |
| |        Software Version: *.** | |
| |        Copyright 2010 | |
| |        ORIENTAL MOTOR CO., LTD. | |
| | -------------------------------------------- | |
| | >ENC | #Query the ENC setting |
| | ENC=1(1) [Driver(Driver)] | #Device response |

## END : Motion End

| | |
|---|---|
| **Execution Mode** | Sequence |
| **Syntax** | END |
| **See Also** | RET |
| **Description** | The END statement can be used to formally terminate sequence text.<br><br>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.<br><br>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. |

| Example | 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 |
| | > | |

## ENDACT  : System End Action

| | |
|---|---|
| **Execution Mode** | Immediate and Sequence |
| **Syntax** | ENDACT=n |
| **Range** | n = 0: End of Pulse Generation<br><br>Greater than 0.001 to Max.pos/2: End Area    (end of pulse generation AND motor is within end area) |
| **Factory Setting** | 0 |
| **SAVEPRM & RESET** | Required to execute any changes made to the parameter value and to save 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 factory setting. |
| **Access** | READ and WRITE<br>READ only in Sequences |
| **See Also** | DEND, MEND, OUTxxx (OUTEND), SIGxxx (SIGEND), PE |
| **Description** | If n is set to zero, the internal end status only references the end of pulse generation.<br><br>If n is set to nonzero, the internal end status references both the end of pulse generation and the end area.<br><br>Set ENDACT to the END range that is "allowable PE (position error) absolute value."  PE is the error between PC (commanded position) and PF (feedback position).<br><br>Example) When setting "ENDACT=1," the system is determined as an internal END status when pulse output is completed and the PE is in  "-1<PE<1."<br><br>See "END (motion end) Signal" on page 105. |

| **Example** | Command | Description |
|---|---|---|
| | >ENDACT=0.01 | #Set the ENDACT |
| | ENDACT=0.001(0.01) | #Device response |
| | >SAVEPRM | #Save the parameter assignments |
| | (EEPROM has been written 21 times) | #Device response |
| | Enter Y to proceed, other key to cancel. y | |
| | Saving Parameters........OK. | |
| | >RESET | #Establish the saved parameter values |
| | Resetting system. | |
| | ----------------------------------------- | |
| | CM10-* | |
| | Controller Module | |
| | Software Version: *.** | |
| | Copyright 2010 | |
| | ORIENTAL MOTOR CO., LTD. | |
| | ----------------------------------------- | |

## ENDIF  : End of IF Block

| | |
|---|---|
| **Execution Mode** | Sequence |
| **Syntax** | ENDIF |
| **See Also** | IF, ELSE, WHILE, WEND |
| **Description** | Indicates the completion of innermost conditional IF statement. |

| **Example** | 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) ENDIF` | #End of IF block |
| | `(  6) END` | #End of sequence: optional |
| | `>` | |

## ENDL  : End of LOOP Block

| | |
|---|---|
| **Execution Mode** | Sequence |
| **Syntax** | ENDL |
| **See Also** | LOOP, BREAKL |
| **Description** | Terminates the innermost LOOP block |

| **Example** | Command | Description |
|---|---|---|
| | `>LIST 5` | #List sequence 5 |
| | | |
| | `(  1) DIS=1` | #Distance equals 1 user unit |
| | `(  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 |
| | `>` | |

## ENDWAIT : END wait time

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | ENDWAIT=n |
| **Range** | n = 0.1 to 40.0 (seconds) |
| **Factory Setting** | 6 |
| **SAVEPRM & RESET** | Required to execute any changes made to the parameter value and to save 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 factory setting. |
| **Access** | READ and WRITE |
| **See Also** | DEND, ENDACT, |

| | |
|---|---|
| **Description** | ENDWAIT sets the waiting time to establish the END status after the pulse signal output is completed (depend on ENDACT setting) when a mechanical home seeking operation or a motion with MEND command used in the sequence is executing. If the END status does not establish within the time that is set by ENDWAIT, an alarm is generated according to the following table. |

| DEND Parameter | ENDACT Parameter | Alarm |
|---|---|---|
| 0 | 0 | n/a |
| 0 | 0<n (END area) | Excessive Position Deviation  10h |
| 1 | Unrelated | Driver Connection Error  6Fh |

See "8.8 END (motion end) Signal" on page 105 in more detail of END signal.

| **Example** | Command | | Description |
|---|---|---|---|
| | `>ENDWAIT=3` | | #Set the ENDACT |
| | ` ENDWAIT=6(3) sec` | | #Device response |
| | `>SAVEPRM` | | #Save the parameter assignments |
| | ` (EEPROM has been written 72 times)` | | #Device response |
| | ` Enter Y to proceed, other key to cancel. y` | | |
| | ` Saving Parameters........OK.` | | |
| | `>RESET` | | #Establish the saved parameter values |
| | ` Resetting system.` | | |
| | `-------------------------------------------` | | |
| | `    CM10-*` | | |
| | `    Universal Controller` | | |
| | `    Software Version: 2.xx` | | |
| | `    Copyright 2010` | | |
| | `    ORIENTAL MOTOR CO., LTD.` | | |
| | `-------------------------------------------` | | |
| | `>` | | |

## ER : Encoder Resolution

| | |
|---|---|
| **Execution Mode** | Immediate and Sequence |
| **Syntax** | ER=n |
| **Range** | n = 10 to 51200 |
| **Factory Setting** | 100: **CM10-3**<br>200: **CM10-2**<br>1000: **CM10-1**, **4**, **5**, **SCX10** |
| **SAVEPRM &<br>RESET** | Required to execute any changes made to the parameter value and to save 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 factory setting. |
| **Access** | READ and WRITE<br>READ only in Sequences |
| **See Also** | EC, PF, MR, PC, PE |
| **Description** | Set the encoder resolution. |
| **Note** | If ER is changed, MAXPOS (maximum position=position range) may be changed.  When executing the ER command, the pre-change and post-change values of MAXPOS are displayed.  Position parameters (DIS, DISx, OFFSET, SCHGPOS, LIMP, LIMN, ENDACT) are automatically checked whether or not to be within the MAXPOS (maximum position), and if they are outside the range, the warning message will be sent.  Note that the position parameters in the program and the position array data POS[x] will not be checked. |

| **Example** | Command | Description |
|---|---|---|
| | ```
>ER=1000
 ER=100(1000)
 Position range = +/- 500000(500000)
 Velocity range = 0.001 - 1240(1240)
>SAVEPRM
 (EEPROM has been written 21 times)
 Enter Y to proceed, other key to cancel. y
 Saving Parameters........OK.
>RESET
 Resetting system.
-------------------------------------
            CM10-*
            Controller Module
            Software Version: *.**
            Copyright 2010
            ORIENTAL MOTOR CO., LTD.
-------------------------------------
>ER
 ER =1000(1000)
``` | #Set the encoder resolution<br>#Device response<br><br><br>#Save the parameter assignments<br>#Device response<br><br><br>#Establish the saved parameter values<br>#Device response |

## EVx  : Configure Event Output

| | |
|---|---|
| **Execution Mode** | Immediate and Sequence |
| **Syntax** | EVx  OUTy=z m=n <br> or <br> EVx  0 |
| **Range** | x: Event Channel Number; 1 or 2 <br> y: Output Number; 1 to 4 (general purpose output 'OUT1 to OUT4' on the I/O connector) <br> z: Output Logic Level after Trigger; 0 (OFF) or 1 (ON) <br> m: Event Trigger Source (T, D, V) <br>  T: Trigger n seconds after motion start; n=0 to 500 (second) <br>  D: Trigger after moving distance n from motion start; n=-MAXPOS to +MAXPOS (user units) <br>  V: Trigger after reaching speed set point n; n=0.001 to MAXVEL (user units/second) |
| **Access** | READ and WRITE |
| **See Also** | OUTx, OUTTEST |
| **Description** | During a motion, the event is recognized to be occurred when either one of the position, velocity or time has reached the specified value, the specified general output signal is changed to the desired level.  It works either by the immediate command or sequence execution. <br><br> Up to 2 events can be configured and active at the same time, using both event channels 1 and 2. <br><br> The setting of the event is not released when the operation has completed, and it is continued until being cleared.  To clear the event, input "EVx 0. "Even if the event has been disappeared or the setting of the event has been cleared, the output will not return automatically.  To set the output status or to return to the status before the event was occurred, use the OUTx command etc. <br><br> Event checking restarts at the beginning of a motion. <br><br> To detect the transition, assure that the designated output is in the opposite state prior to the event occurring. <br><br> If the output has been assigned to a system output signal, no event-driven transitions will occur on the output. |
| **Example** | Command            Description |

```
>OUT1=0;OUT2=0        #Set the output to the opposite direction of the event occurring
 OUT1=0
 OUT2=0
>EV1 OUT2=1 V=10      #Turn ON output 2 when reach speed of 10 user units/second
 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
>OUT1=0;OUT2=0        #Reset output#1 and #2
 OUT1=0
 OUT2=0
>EV1 0; EV2 0         #Clear events number 1 and 2
>
```

## FREE  : Current OFF, Magnetic Brake Free

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | FREE=n |
| **Range** | n =  0: Normal Condition<br>        1: Motor Shaft Free |
| **Factory Setting** | 0 |
| **Access** | READ and WRITE |
| **See Also** | INxxx (INFREE), SIGxxx (SIGFREE), xxxLV (FREELV), CURRENT |
| **Description** | The FREE command is used to control the state of the FREE signal that is tied with the FREE and MBFREE output on the driver connector of the **CM10**/**SCX10** and the MBFREE output on the I/O connector (if assigned). |

The FREE function may also be executed via the FREE input on the I/O connector if assigned and/or the CANopen remote I/O. When detailed explanation is required, see "6.4 Connecting the I/O signals" on page 23 for I/O connector, and see "10.4 Controlling I/O Message (PDO)" on page 134 for CANopen remote I/O.

The FREE command can be used to control the state of the FREE function regardless of the states of those inputs.

Additionally, when the state of FREE function becomes 1, the PC (position command) value will be set equal to the PF (position feedback) value. This function is utilized when positioning without motor current such as when teaching positions is required.

- If the driver has a "FREE" input (**CM10-1**, **3**, **5**, **SCX10** with applicable drivers):

The FREE output on the driver connector of the **CM10**/**SCX10** is used. When the state of the FREE function is set to 1, the motor current will be turned OFF and the electromagnetic brake will be released (The FREE input of the connected driver is turned ON).

- If the driver has a "M.B.FREE" input (**SCX10** with applicable drivers):

The MBFREE output on the driver connector of the **SCX10** is used. When the state of the FREE is set to 1, the electromagnetic brake will be released. (The M.B.FREE does not affect the motor current.) (Additionally, the MBFREE output turns OFF when the motor loses its holding torque due to a current cutoff or alarm.)

Be sure that the M.B.FREE input on the driver is enabled prior to controlling the FREE function status if controlling an electromagnetic brake is desired. Example: With the **RK** Series Five-phase stepping motor and driver unit, the brake function switch is located on the front panel. The factory setting is the "power-failure position-holding mode" and the M.B.FREE input is disabled.

- If the driver does not have either of the above inputs and an external electromagnetic brake is used (**CM10-2**, **4**, **SCX10**):

The MBFREE output on the I/O connector can be used. When the state of the FREE function is set to 1, the electromagnetic brake will be released. (Additionally, the MBFREE output turns OFF when the motor loses its holding torque due to a current cutoff or alarm.)

| **Example** | Command | Description |
|---|---|---|
| | >FREE=1<br>  FREE=1 | #Turn motor current OFF and release electro magnetic brake. Motor has no holding torque |
| | >FREE=0<br>  FREE=0 | #Back to normal |

## GA, GB : Electrical Gear Ratio (GA/GB)

| | |
|---|---|
| **Execution Mode** | Immediate and Sequence |
| **Syntax** | GA=n {Numerator} <br> GB=n {Denominator} |
| **Range** | n = 1 to 100 (integer values) |
| **Factory Setting** | 1 |
| **SAVEPRM & RESET** | Required to execute any changes made to the parameter value and to save 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 factory setting. |
| **Access** | READ and WRITE <br> READ only in Sequences |
| **See Also** | DPR, UU, MAXVEL |
| **Description** | The distance and velocity of the motor may be adjusted to compensate for a gearhead or gear assembly. The gear ratio is set via the GA and GB parameter. The applied gear ratio equals GA/GB. The numerator and the denominator of the electric gear are set to opposite to the mechanical gear. <br><br> For example, if a ball screw with a lead of 10 mm/rev is combined with a 3:1 reduction gearhead, the following parameters would be used: <br> UU=mm  #User units <br> DPR=10  #Distance per rev <br> GA=3      #Electronic gear ratio numerator value <br> GB=1      #Electronic gear ratio denominator value <br><br> The motor must rotate 3 times as far to complete one revolution at the gearhead output. Therefore the GA value is 3 to compensate for the gear ratio's reduction in distance and velocity. <br><br> Electronic gearing can also be used when the distance per revolution is less than 0.5, or when the exact ratio cannot be specified in three decimal places (e.g. if the distance per revolution is 1/3 user unit). <br><br> Setting DPR=1, electronic gear numerator GA=3 and denominator GB=1 will result in three motor rotations per one user unit, for an effective DPR of exactly 1/3 user unit. |
| **Note** | If GA and/or GB are changed, MAXPOS (maximum position=position range), MAXVEL (maximum velocity=velocity range) or minimum travel distance (minimum movable distance) may automatically be changed. When executing the GA and/or GB command, these pre-change and post-change values are shown. Position parameters (DIS, DISx, OFFSET, SCHGPOS, LIMP, LIMN, ENDACT) and velocity parameters (VS, VR, VRx, SCHGVR) are automatically checked whether or not to be within the MAXPOS (maximum position) and MAXVEL (maximum velocity) respectively, and if they are outside the range, the warning message will be sent.  Note that the position/velocity parameters in the program and the position array data POS[x] will not be checked.  Check whether the required resolution is obtained. The minimum movable distance, which is the actual travel distance, can be checked by the TEACH command. <br><br> If GA and/or GB are changed, the actual travel distance or velocity is changed.  Check whether the present settings of the position/velocity parameters or those values in the program are appropriate. |

| Example | Command | Description |
|---|---|---|
| | >UU mm<br> UU=mm | #Set user units to mm (millimeters) |
| | >DPR 10<br> DPR=1(10) mm<br> Position range = +/- 500000(500000)<br> Velocity range = 0.001 - 2480(24800)<br> Minimum Movable Distance = +/- 0.001(0.001) | #Set the distance per revolution to 10 mm |
| | >GA 3<br> GA=1(3) | #Set the electrical gear ratio numerator to 3: system rescales again |
| | >GB 1<br> GB=1(1) | #Set the electrical gear ratio denominator to 1, system rescales again |
| | >MAXVEL<br> MAXVEL=2480(8266.666) mm/sec | #Query the MAXVEL value |
| | >SAVEPRM<br> (EEPROM has been written 28 times)<br> Enter Y to proceed, other key to cancel. y<br> Saving Parameters........OK. | #Save the parameter assignments |
| | >RESET<br> Resetting system.<br> -------------------------------------------<br>          CM10-*<br>          Controller Module<br>          Software Version: *.**<br>          Copyright 2010<br>          ORIENTAL MOTOR CO., LTD.<br> ------------------------------------------- | #Establish the saved parameter values |
| | >DPR<br> DPR=10(10) mm<br> Position range = +/- 500000(500000)<br> Velocity range = 0.001 - 8266.666(8266.666)<br> Minimum Movable Distance = +/- 0.001(0.001) | #Check new effective values |
| | >GA<br> GA=3(3) | |
| | >GB<br> GB=1(1)<br> > | |

## HELP : Display Help Information

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | HELP |
| **Description** | 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. |
| **Example** | Command                                                       Description |

```
>HELP

   --- Command List ---

   TALK*      : Select unit in multi-unit communications
   @*         : Select unit in multi-unit communications
   MI         : Move Incrementally
   MA         : Move Absolutely (-MAXPOS - +MAXPOS[UU])
   CV         : Change Velocity for Index (0.001 - MAXVEL[UU/sec])
   MCP        : Move Continuous Positive
   MCN        : Move Continuous Negative
   DIS        : Incremental motion distance (-MAXPOS - +MAXPOS[UU])
   VR         : Running velocity (0.001 - MAXVEL[UU/sec])
   VS         : Starting velocity (0 - MAXVEL[UU/sec])
   TA         : Acceleration time (0.001-500.000[sec])
   TD         : Deceleration time (0.001-500.000[sec])
   PSTOP      : Stop immediately, stop sequence, follow ALMACT setting
   HSTOP      : Stop immediately (hard stop)
   MSTOP      : Stop according to MSTOPACT
   SSTOP      : Stop, decelerating (soft stop)
   SCHGPOS    : Distance from SENSOR on MCx (0 - MAXPOS[UU])
   SCHGVR     : Velocity on SCHGPOS motion (0.001 - MAXVEL[UU/sec])

   Enter [SPACE] to continue, other key to quit.    #[SPACE] entered

   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)
                   (-MAXPOS - +MAXPOS[UU])
   VR0        : (-VR3) Velocity for linked index 'x' (x=0-3)
                   (0.001 - MAXVEL[UU/sec])
   INCABS0    : (-INCABS3) Set positioning mode for index 'x' (x=0-3)
                   (0:Absolute/1:Incremental)
   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
   EHOME      : Move to position 0
   MGHP       : Find Home, start in Positive direction
   MGHN       : Find Home, start in Negative direction
   OFFSET     : Distance from HOME on Might (-MAXPOS - +MAXPOS[UU])

   Enter [SPACE] to continue, other key to quit.    #non-space entered
>
```

## HOMEDCL : Select the Deviation Counter Clear During Mechanical Home Seeking Operation

| | |
|---|---|
| **Execution Mode** | Immediate and Sequence |
| **Syntax** | HOMEDCL=n |
| **Range** | n = 0: Clear Deviation Counter in **CM10**/**SCX10** at Homing<br>    1: Clear Deviation Counter in both **CM10**/**SCX10** and Driver at Homing<br>    2: Not Clear Deviation Counter neither **CM10**/**SCX10** nor Driver at Homing |
| **Factory Setting** | 0: **CM10-1**, **2**, **3**, **4**, **SCX10**<br>1: **CM10-5** |
| **SAVEPRM &<br>RESET** | Required to execute any changes made to the parameter value and to save 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). |
| **Access** | READ and WRITE<br>READ only in Sequences |
| **See Also** | HOMETYP |
| **Description** | HOMEDCL establishes the ACL/DCL output action at the end of mechanical home seeking.<br><br>If the HOMEDCL is set to 1, the ACL/DCL signal is momentarily output to the driver when a mechanical home position is found.<br><br>If a servo motor is used, set the HOMEDCL to 1. The deviation counter in the driver is cleared to perform an immediate stop, and that causes accurate homing.<br><br>If the $\alpha_{STEP}$ products is used and the HOMETYP is set so that the timing signal is used for a mechanical home seeking, set the HOMEDCL to 0. Setting the HOMEDCL to 1 may cause a position deviation from the timing signal at the final approach due to a delay in the velocity filter of the $\alpha_{STEP}$ driver.<br><br>When an accurate deviation is required to be seen with the stepping motor plus encoder (including the $\alpha_{STEP}$, **ESMC** controller) while the load is applied to the shaft at mechanical home seeking operation, set to "HOMEDCL=2." See "HOMEDCL (deviation counter select at mechanical home seeking operation" on page 86. |

| Setting Value | Deviation Counter of the **CM10**/**SCX10** | Deviation Counter of the Driver |
|:---:|:---:|:---:|
| 0 | Clear | - |
| 1 | Clear | Clear |
| 2 | - | - |

| **Example** | Command | Description |
|---|---|---|

```
>HOMEDCL=1                                      #Set the HOMEDCL to 1
 HOMEDCL=0(1)
>SAVEPRM                                         #Save the parameter assignments
 (EEPROM has been written 10 times)
 Enter Y to proceed, other key to cancel. y
 Saving Parameters........OK.
>RESET                                           #Establish the saved parameter values
 Resetting system.
------------------------------------------
         CM10-*
         Controller Module
         Software Version: *.**
         Copyright 2010
         ORIENTAL MOTOR CO., LTD.
------------------------------------------
>HOMEDCL                                         #Query new value
>HOMEDCL=1(1)
>
```

## HOMETYP  : Mechanical Home Seeking Mode

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | HOMETYP=n |
| **Range** | n = 0 to 11 (0): **CM10-1**, **2**, **4**, **5**<br>0 to 12 (0): **CM10-3**, **SCX10** |
| **Factory Setting** | 0 |
| **SAVEPRM** | 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 factory setting. |
| **Access** | READ and WRITE<br>READ only while motion is in progress |
| **See Also** | HOMEDCL, INxxx (INHOME, INLSP, INLSN, INSENSOR), RINxxx (RINHOME, RINLSP, RINLSN, RINSENSOR), MGHP, MGHN, OFFSET, OUTxxx (OUTHOMEP) |
| **Description** | HOMETYP configures system operation when seeking a mechanical home position with the MGHP or MGHN commands. Mechanical home seeking reacts to various inputs differently, depending on HOMETYP, and according to the following table: |

| HOMETYP | Home Position Indicator Signals | | | |
|---|---|---|---|---|
| | HOME | +LS, -LS | SENSOR | TIMING |
| 0 | not used | Required for valid home | - | - |
| 1 | | | - | Required for valid home |
| 2 | | | Required for valid home | - |
| 3 | | | Required for valid home | Required for valid home |
| 4 | Required for valid home | Reverse direction | - | - |
| 5 | | | - | Required for valid home |
| 6 | | | Required for valid home | - |
| 7 | | | Required for valid home | Required for valid home |
| 8 | | Stop: Alarm | - | - |
| 9 | | | - | Required for valid home |
| 10 | | | Required for valid home | - |
| 11 | | | Required for valid home | Required for valid home |
| 12 | not used | not used | not used | not used |

See "8.2.5 Mechanical Home Seeking" on page 83 for details.

| | |
|---|---|
| **Memo** | SENSORACT does not affect the use of the SENSOR input while seeking mechanical home with MGHP or MGHN. |

| **Example** | Command | Description |
|---|---|---|
| | >HOMETYP 6 | #Use HOME and SENSOR |
| | HOMETYP=6 | #LSx causes reversal |
| | >MGHP | #Seek mechanical home, approach from the positive direction. Home determined by |
| | > | HOME and SENSOR both active |

## HSTOP  : Hard Stop

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | HSTOP |
| **See Also** | <ESC>, ABORT, MSTOP, MSTOPACT, PSTOP, SSTOP,  PAUSE |
| **Description** | HSTOP stops the motor as quickly as possible. This command does not stop a sequence program. |
| | The HSTOP command operates independently of the motor stop action setting (MSTOPACT). |
| **Caution** | **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.** |
| **Note** | At high speeds, or with high inertial loads, HSTOP may cause an alarm condition. |

| **Example** | Command | Description |
|---|---|---|
| | >MCP | #Move the motor continuously in the positive direction |
| | >HSTOP | #Stop the motor as quickly as possible |
| | > | |

## ID : Device ID

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | ID=n |
| **Range** | n = *, 0 to 9 and A to Z (upper or lower case, not case sensitive) |
| **Factory Setting** | * |
| **SAVEPRM** | 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 factory setting. |
| **Access** | READ and WRITE |
| **See Also** | @, \ (BACKSLASH), ECHO, TALK, VERBOSE, BAUD |
| **Description** | ID sets a device address identifier, for serial communications in a multi-axis daisy chain configuration. |
| | In a daisy chain configuration, each device must have a unique ID. That ID (along with other Communications parameters) should be configured before inserting the device into the daisy chain, using single-axis communications. |
| | The factory setting (*) signifies "no ID." The system is configured for single-axis operation. |
| | When a device has an ID that is not "*," it must be specifically addressed before it will process commands or transmit information. Addressing the device can be accomplished in two ways: |
| | - Use the TALK command: TALKid (note no space between TALK and id) will signal that the device with ID=id (and no other) should respond to communications |
| | - Use the @ command prefix: @id, (note no space between @ and id) will also select the device with ID=id, similar to TALK. |
| | When a device has been selected, it remains selected. The device changes its command line prompt to show its ID. If a device with ID=A is selected, the prompt changes from ">" to "A>." All commands will be processed by that device, until another TALK command or @ prefix is sent with a different ID. |
| | When a device is selected, and its ID is changed, the device remains selected (even with the new ID). The new prompt should return immediately, and communications can continue. |
| | Because devices with a non-* ID must be addressed before communicating, these devices will not transmit any sign-on information or prompts after a power cycle or reset. Use TALK or @ to call the device. |
| | To return a device to the default single-axis configuration, select the device, and then set ID=*. |
| | If a device's ID is not known, connect for single-axis communications, and use \ID. Backslash (\) is a "global" selector: all units will respond. If the "unknown" device is the only connected device, the missing ID should be revealed. |
| **Note** | It is usually most efficient to fully configure each device in stand-alone, single-axis mode. Configure the device ID last. Issue the SAVEPRM command, reset the system, and confirm proper ID and operation before inserting a device into the daisy chain. |

| **Example** | Command | Description |
|---|---|---|
| | `>ID` | #System has default prompt... |
| | ` ID=*` | #...and ID |
| | `>ID C` | #Set ID to 'C' |
| | ` ID=C` | |
| | `C>SAVEPRM` | #Save parameters |
| | ` (EEPROM has been written 36 times)` | |
| | ` Enter Y to proceed, other key to cancel. Y` | |
| | ` Saving Parameters........OK.` | |
| | `C>RESET` | #Reset the system |
| | ` Resetting system.` | #Note no sign-on banner or prompt |
| | `@CVER` | #Address C, query version |
| | `CM10-*/*.**/Mar  9 2010` | #Version response |
| | `C>` | #Prompt from device with ID=C |

# IF  : Begin IF Block: execute if IF is true

| | |
|---|---|
| **Execution Mode** | Sequence |
| **Syntax** | IF (element1 {Conditional Operator} element2) |
| **Range** | Conditional Expression |
| **See Also** | ELSE, ENDIF, WHILE, BREAKW, WEND, LOOP, BREAKL, ENDL |
| **Description** | Executes the conditional branching of an IF statement. Parentheses are required.<br><br>Element1 and element2 may be any numeric variable available to sequences, or any numeric constant within the range -(Maximum Number) to +(Maximum Number). (If POS[x] is used, refer to "POS[x]" on page 311.)<br><br>Valid conditional operators are:<br>=, ==   : Equal to<br>!=      : Not equal to<br><       : Less than<br><=      : Less than or equal to<br>>       : Greater than<br>>=      : Greater than or equal to<br><br>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.<br><br>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).<br><br>Block structures (IF-ENDIF, WHILE-WEND, LOOP-ENDL) may be nested, to eight (8) levels deep. |

| Example | Command | Description |
|---|---|---|
| | >LIST 8 | #List sequence 8 |
| | ( 1) MCP | #Move continuously (positive) |
| | ( 2) WHILE (IN1=0) | #Start WHILE block. Execute lines 3 through 5 while condition is true |
| | ( 3)   IF (IN2=1) | #If IN2 is 1 (ON), execute line 4 |
| | ( 4)     BREAKW | #Exit the WHILE loop and execute the line after the WEND command |
| | ( 5)   ENDIF | #End the IF block |
| | ( 6) WEND | #End the WHILE block, return to line 2 |
| | ( 7) SSTOP | #Slow down and stop the motor |
| | ( 8) END | #End the sequence |

## IN : General Input Status

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | IN |
| **Range** | 0 to 511 (integer values) <br> /: real time monitor  (immediate mode only) |
| **Access** | READ |
| **See Also** | IO, INx, xxxLV (INxLV), OUT, OUTx, INxxx, INSG, INITIO, REPORT, OUTTEST |
| **Description** | The IN command displays the current status of all the general purpose inputs, as one integer number. <br><br> The general purpose inputs contribute to the value of IN as follows: |

| INx | Contribution to IN If Active |
|---|---|
| IN9 | 256 |
| IN8 | 128 |
| IN7 | 64 |
| IN6 | 32 |
| IN5 | 16 |
| IN4 | 8 |
| IN3 | 4 |
| IN2 | 2 |
| IN1 | 1 |

For example, if the general input 2 (2), general input 3 (4) and  general input 4 (8) are active, while all other signals are not active, "IN=14" is set (2+4+8=14).

When inputting IN, "IN=14" is replied.


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

∗ If an input is assigned to a system input signal (HOME, LSN, LSP etc.) the IN command will always read that particular input OFF or 0.  Use the INSG command in order to refer to the status of all input signals assigned the specific function.

| **Example** | 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 |
| | `(  9)    WEND` | #End the WHILE loop |
| | `( 10)  ENDIF` | #End the IF block |
| | `>` | |

## INCABSx : Link Type for Link Segment 'x'

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | INCABSx=n      ( x is a number of linked segments: x=0 to 3.) |
| **Range** | n =  0: Absolute<br>      1: Incremental |
| **Factory Setting** | 1 |
| **SAVEPRM** | 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 factory setting. |
| **Access** | READ and WRITE |
| **See Also** | DISx, VRx, LINKx, MIx, TA, TD, VS |
| **Description** | INCABSx determines whether DISx represents a distance or an absolute destination for linked index (MIx) motion commands. |
| **Note** | Each of the four links can be incremental or absolute.<br><br>Incremental and absolute can be used in combination, but all links executed together must move in the same direction.<br>- For incremental links, motion direction is determined by the arithmetic sign of DIS.<br><br>- For absolute links, motion direction is determined by the motor position at the start of that motion link.<br><br>Generally, absolute links are not recommended when the motor position before linked operation cannot be predicted. |

| **Example** | Command | Description |
|---|---|---|
| | ```
>UU in
 UU=in
>VR1 5
 VR1=5 in/sec
>DIS1 10
 DIS1=10 in
>INCABS1 1
 INCABS1=1 [INC]
>LINK1 1
 LINK1=1
>VR2 10
 VR2=10 in/sec
>INCABS2 0
 INCABS2=0 [ABS]
>DIS2 20
 DIS2=20 in
>LINK2 0
 LINK2=0
>MI1
>
``` | #Set user units to in. (inches)<br>#Device response<br>#Set the velocity for linked move 1 to 5 user units/second<br>#Device response<br>#Set the distance for linked move 1 to 10 user units<br>#Device response<br>#Set the move type for linked motion 1 to incremental<br>#Device response<br>#Enable the linked operation for motion 1<br>#Device response<br>#Linked move 2 velocity equals 10 user units/second<br>#Device response<br>#Set the move type for linked motion 2 to absolute<br>#Device response<br>#Linked move 2: destination is position 20 user units<br>#Device response<br>#"Unlink" link 2 from link 3<br>#Device response<br>#Start the linked operation motion |

## INITDIO : Initialize Driver I/O

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | INITDIO |
| **SAVEPRM & RESET** | Required to execute any changes made to the parameter value and to save 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 factory setting. |
| **See Also** | DINxxx, DOUTxxx, INITPRM, CLEARALL |
| **Description** | Cancel all driver input or driver output assignments. All driver system input signal assignment values and all driver system output signal assignment values are set to zero (0), unassigned.<br><br>When having executed the INITDIO command accidentally, execute the RESET command without executing the SAVEPRM command. It returns to the setting before having executed the INITDIO command. |
| **Caution** | **The INITDIO command writes to EEPROM. The EEPROM has a nominal expected lifetime of 100,000 write cycles. The INITDIO command should not be used automatically (i.e. by a host controller) if it could possibly execute at high frequency. The number of times for accumulated write cycles is displayed when executing.** |

| **Example** | Command | Description |
|---|---|---|
| | ```
>INITDIO
 Enter Y to proceed, other key to cancel. Y

1(1)
3(3)
6(6)
7(7)
0(0)
5(5)
4(4)
4(0)
5(0)
6(0)
0(0)
1(1)
2(2)
0(4)
8(8)
0(0)
0(0)
0(0)
3(0)
7(0)
2(0)
4(0)
5(0)
6(0)
0(0)
 All driver I/O configurations are set to factory default.
 Execute SAVEPRM then RESET to activate new settings.
>SAVEPRM
 (EEPROM has been written 21 times)
 Enter Y to proceed, other key to cancel. y
 Saving Parameters........OK.
>RESET
 Resetting system.
-------------------------------------------
         CM10-*
         Controller Module
         Software Version: *.**
         Copyright 2010
         ORIENTAL MOTOR CO., LTD.
-------------------------------------------
>
``` | #Reset the current DIO assignment to factory settings<br><br>#Device response<br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>#Save the assignments<br>#Device response<br><br><br>#Establish the saved parameter values |

## INITIO  : Initialize I/O

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | INITIO |
| **SAVEPRM & RESET** | Required to execute any changes made to the parameter value and to save 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 factory setting. |
| **See Also** | INxxx, OUTxxx, OUTTEST, INITRIO, INITPRM, CLEARALL |
| **Description** | Cancels all input or output assignments. All system input signal assignment values and all system output signal assignment values are set to zero (0), unassigned.<br><br>All inputs and outputs are reset for general purpose use. If the command is executed accidentally, RESET without SAVEPRM. The old I/O assignments remain effective until SAVEPRM and RESET execute. INITIO does not change any signal level assignments (e.g. HOMELV, etc.). |
| **Caution** | **The INITIO command writes to EEPROM. The EEPROM has a nominal expected lifetime of 100,000 write cycles.  The INITIO command should not be used automatically (i.e. by a host controller) if it could possibly execute at high frequency.  The number of times for accumulated write cycles is displayed when executing.** |

**Example**

| Command | Description |
|---|---|
| >INITIO<br> Enter Y to proceed, other key to cancel. Y<br>1(0)<br>0(0)<br>5(0)<br>0(0)<br>0(0)<br>0(0)<br>0(0)<br>0(0)<br>0(0)<br>0(0)<br>0(0)<br>0(0)<br>7(0)<br>0(0)<br>0(0)<br>0(0)<br>0(0)<br>0(0)<br>0(0)<br>0(0)<br>9(0)<br>1(0)<br>0(0)<br>0(0)<br>0(0)<br>0(0)<br>0(0)<br>0(0)<br>0(0)<br>0(0) | #Reset the current IO assignment to factory settings<br>#Device response |

```
 All I/O configurations are set to factory    #Device response
default.
 Execute SAVEPRM then RESET to activate new    #Save the parameter assignments
settings.                                      #Device response
>SAVEPRM
 (EEPROM has been written 21 times)            #Establish the saved parameter values
 Enter Y to proceed, other key to cancel. y
 Saving Parameters........OK.
>RESET
 Resetting system.
-------------------------------------------
        CM10-*
        Controller Module
        Software Version: *.**
        Copyright 2010
        ORIENTAL MOTOR CO., LTD.
-------------------------------------------
>
```

## INITPRM  : Initialize Parameters

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | INITPRM |
| **RESET** | Reset required before new value becomes active. |
| **Commands not Allowed** | MOVE, RUN |
| **See Also** | CLEARALL, CLEARPOS, CLEARSEQ, CLEARVAR, INITIO, INITDIO, INITRIO |
| **Description** | When executing the INITPRM command, the all parameters except position array data (POS[x]) and sequence programs restore to the factory setting.  The all parameters contain the user variables, user-defined variables, all I/O setting and communication settings. |
| **Caution** | **The INITPRM command writes to EEPROM. The EEPROM has a nominal expected lifetime of 100,000 write cycles.  The INITPRM command should not be used automatically (i.e. by a host controller) if it could possibly execute at high frequency.  The number of times for accumulated write cycles is displayed when executing.** |
| **Note** | Since the baud rate and ID restore to the factory settings when executing INITPRM, the communication cannot be performed if these settings have been changed. Note this point. |
| **Example** | Command |

| Command | Description |
|---|---|
| `>INITPRM`<br>` (EEPROM has been written 45 times)`<br>` Enter Y to proceed, other key to cancel. y`<br>` Initializing Parameters..OK.`<br>`>RESET`<br>` Resetting system.`<br>`------------------------------------------`<br>`          CM10-*`<br>`          Controller Module`<br>`          Software Version: *.**`<br>`          Copyright 2010`<br>`          ORIENTAL MOTOR CO., LTD.`<br>`------------------------------------------`<br>`>` | #Reset all of the motion parameters to factory settings<br>#Once confirmed, memory overwritten, old values lost<br>#Reset required to activate new factory default settings |

## INITRIO : Initialize Remote I/O

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | INITRIO |
| **SAVEPRM & RESET** | Required to execute any changes made to the parameter value and to save 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 factory setting. |
| **See Also** | RINxxx, ROUTxxx, INITIO, INITPRM, CLEARALL |
| **Description** | Cancels all remote input or remote output assignments. All remote input signal assignment values and all remote output signal assignment values are set to zero (0), unassigned.<br><br>All remote inputs and outputs are reset for general purpose use. If the command is executed accidentally, RESET without SAVEPRM. The old I/O assignments remain effective until SAVEPRM and RESET execute. |
| **Caution** | **The INITRIO command writes to EEPROM. The EEPROM has a nominal expected lifetime of 100,000 write cycles. The INITRIO command should not be used automatically (i.e. by a host controller) if it could possibly execute at high frequency. The number of times for accumulated write cycles is displayed when executing.** |

| **Example** | Command | Description |
|---|---|---|
| | >INITRIO | #Reset the current RIO assignment to factory settings |
| | Enter Y to proceed, other key to cancel. Y | |
| | 0(0) | |
| | 0(0) | #Device response |
| | 0(0) | |
| | 0(0) | |
| | 1(0) | |
| | 0(0) | |
| | 0(0) | |
| | 0(0) | |
| | 0(0) | |
| | 0(0) | |
| | 0(0) | |
| | 7(0) | |
| | 0(0) | |
| | 0(0) | |
| | 0(0) | |
| | 4(0) | |
| | 0(0) | |
| | All I/O configurations are set to factory default. | #Device response |
| | Execute SAVEPRM then RESET to activate new settings. | |
| | > | |

## INSG  : System Signal Input Status

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | INSG |
| **Range** | 0 to 2096127 (integer values)<br>/: real time monitor  (immediate mode only) |
| **Access** | READ |
| **See Also** | IN, xxxLV, OUTSG, IO, SIGxxx, REPORT, OUTTEST |
| **Description** | The INSG command displays the current status of all the system input signals, as one integer number.<br>The system input signals contribute to the value of INSG as follows: |

| Bit Location | Signal | Contribution to INSG If Active |
|---|---|---|
| Bit 20 | CONT | 1048576 |
| Bit 19 | TL | 524288 |
| Bit 18 | PECLR | 262144 |
| Bit 17 | MGHN | 131072 |
| Bit 16 | MGHP | 65536 |
| Bit 15 | MCN | 32768 |
| Bit 14 | MCP | 16384 |
| Bit 13 | FREE | 8196 |
| Bit 12 | CON | 4096 |
| Bit 11 | ALMCLR | 2048 |
| Bit 10 | - | - |
| Bit 9 | PAUSECL | 512 |
| Bit 8 | PAUSE | 256 |
| Bit 7 | SENSOR | 128 |
| Bit 6 | HOME | 64 |
| Bit 5 | LSN | 32 |
| Bit 4 | LSP | 16 |
| Bit 3 | MSTOP | 8 |
| Bit 2 | PSTOP | 4 |
| Bit 1 | ABORT | 2 |
| Bit 0 | START | 1 |

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

For example, if the HOME (64) and SENSOR (128) signals are active, while all other signals are not active, "INSG=192" is set (64+128=192). When inputting INSG, "INSG=192" is replied.

* When checking the status of a single system input signal use the SIGxxx command.

∗ Be careful not to confuse INSG with IN (general input status).  IN reports the status of general purpose inputs (those inputs which are not assigned to a signal).

| **Example** | Command | Description |
|---|---|---|
| | >INSG | #Query the current input signal value |
| | INSG=52834 | #Device response: the CON signal and TL signal are active |

## INx : Individual General Input Status

| | |
|---|---|
| **Execution Mode** | Immediate and Sequence |
| **Syntax** | INx    (INx is a signal name: IN1 to IN9) |
| **Range** | n =  0: Not Active<br>        1: Active<br>/: real time monitor  (immediate mode only) |
| **Access** | READ |
| **See Also** | IN, INSG, INxLV, IO, OUTTEST, OUTx, SIGxxx, REPORT |
| **Description** | INx returns the state of general purpose input INx.<br><br>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 SIGxxx command to check the status of the individual system input signal. |

| **Example** | Command | Description |
|---|---|---|
| | >LIST JOG | #List sequence named "JOG" |
| | | |
| | (  1) TA= 0.1; TD=0.1; VS=0; VR=5 | #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 |
| | > | |

## INxxx : System Signal Input Assignment

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | INxxx=n ("xxx" represents the signal name to be assigned, and "n" represents the assigned terminal number ("n" becomes the INn input at the time of general input) |
| **Range** | n = 0 to 9 |
| **Factory Setting** | 0 (unassigned) |
| **SAVEPRM & RESET** | Required to execute any changes made to the parameter value and to save 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 factory setting. |
| **Access** | READ and WRITE |
| **See Also** | xxxLV (except: LSP/LSN), SIGxxx, IO, INITIO, CLEARALL, INSG, INx, OUTTEST, REPORT, RINxxx, and "See Also" column in the chart below. |
| **Description** | Assign the system input signal to the INn of the I/O connector of the **CM10**/**SCX10**. The system signal input assignment is released by "INxxx=0" and it becomes the general input INn. When executing the INITIO command, the parameter restores to the factory setting. |

| Command | Signal | Description | See Also |
|---|---|---|---|
| INABORT | ABORT | Abort Motion and Sequence Execution | ABORT, <ESC> |
| INALMCLR | ALMCLR | Alarm Clear | ALMCLR |
| INCON | CON | Current ON | CON, CURRENT, STRSW |
| INCONT | CONT | Continue Motion | PAUSE, PAUSECLR, STARTACT |
| INFREE | FREE | Current OFF, Magnetic Brake Free | FREE, CURRENT |
| INHOME | HOME | Home Sensor | HOMETYP, MGHP, MGHN |
| INLSP/INLSN | LSP/LSN | Limit Switch Positive /Limit Switch Negative | ALM, ALMACT, ALMCLR, OTLV |
| INMCP/INMCN | MCP/MCN | Move Continuously Positive /Move Continuously Negative | - |
| INMGHP/INMGHN | MGHP/MGHN | Move Go Home Positive /Move Go Home Negative | MGHP, MGHN, HOMETYP |
| INMSTOP | MSTOP | Motor Stop | MSTOP, MSTOPACT |
| INPAUSE | PAUSE | Pause Motion | PAUSE, CONT, PAUSECLR, INPAUSECL, |
| INPAUSECL | PAUSECL | Pause Clear | PAUSE, PAUSECLR, CONT, OUTPSTS |
| INPECLR | PECLR | Position Error Clear | PECLR, PC, PE, PF |
| INPSTOP | PSTOP | Panic Stop | PSTOP, ABORT, <ESC>, ALMACT, HSTOP, MSTOP, SSTOP, PAUSE |
| INSENSOR | SENSOR | Sensor | SENSORACT, MGHP, MGHN, SCHGPOS, SGHGVR |
| INSTART | START | Start Sequence | STARTACT |
| INTL | TL | Torque Limiting /Push-motion Operation /Current Cutback Release | TL, DOUTTL |

| Example | Command | Description |
|---|---|---|
| | >INABORT 2 | #Assign the ABORT signal to input 2 |
| | INABORT=0(2) | #Save the parameter assignments |
| | >SAVEPRM | |
| | (EEPROM has been written 2 times) | |
| | Enter Y to proceed, other key to cancel. Y | |
| | Saving Parameters........OK. | |
| | >RESET | #Establish the saved parameter value |
| | Resetting system. | |
| | -------------------------------------------- | |
| | CM10-* | |
| | Controller Module | |
| | Software Version: *.** | |
| | Copyright 2010 | |
| | ORIENTAL MOTOR CO., LTD. | |
| | -------------------------------------------- | |
| | >INABORT | #Confirm new value |
| | INABORT=2(2) | |
| | > | |

## IO  : Input/Output Status

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | IO |
| **Range** | /: real time monitor  (immediate mode only) |
| **See Also** | xxxLV, INxxx, OUTxxx, IN, OUT, OUTTEST, INx, OUTx, SIGxxx, DIO, RIO, REPORT |
| **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.<br><br>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.<br><br>In the example below, input 1 and input 7 to 9, output 4 remains general purpose: all other I/O have been assigned to system signals. General purpose input 1 is active, so IN=1, and sequence 1 would start if the alarm condition were cleared and START became active. |

| **Example** | Command | Description |
|---|---|---|
| | `>IO` | #Display IO status |
| | ` Inputs  (1-9) = IN1 SENSOR HOME PSTOP -LS +LS IN7 IN8 IN9` | #Device response |
| | ` Outputs (1-4) = END RUN ALARM OUT4` | |
| | | |
| | ` --Inputs--              Outputs` | |
| | ` 1 2 3 4 5 6 7 8 9 -(SEQ#)- 1 2 3 4` | |
| | ` 1 1 0 0 0 0 0 0 0 -(  1 )- 1 0 0 0` | |
| | ` >` | |

# KB : Keyboard Input

| | |
|---|---|
| **Execution Mode** | Sequence |
| **Syntax** | *variable* = KB |
| **Range** | *variable* refers to any numeric *variable* which sequences can write to.<br>Actual permitted range depends on *variable*  -Max.Num to +Max.Num |
| **See Also** | KBQ, SAS, SACS, VIEW |
| **Description** | KB transmits a data entry prompt over the serial port, accepts a numeric value from the serial port, and assigns that value to *variable*.<br><br>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).<br><br>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.<br><br>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.<br><br>Sequence execution is effectively suspended while waiting to receive a valid numeric value.<br><br>For similar operation without prompting, see KBQ (Keyboard Input Quiet).<br><br>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.  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. |
| **Memo** | In a daisy chain configuration (ID other than *), all output from sequence commands is suppressed unless the device has been previously addressed (via TALK or @). The KB and KBQ commands will not receive input unless the device has been previously addressed. |
| **Example** | |

| Command | Description |
|---|---|
| `>LIST 9` | #List sequence 9 |
| `( 1) VR=10` | #Set running velocity |
| `( 2) SACS How far do you want to go` | #Prompt user to enter desired distance |
| `( 3) DIS=KB` | #Output ? and wait for new value |
| `( 4) DIS` | #Distance equals the entry value (KB) |
| `( 5) MI` | #Execute an index move of DIS user units |
| `( 6) MEND` | #Wait for motion to end. |
| `>RUN 9` | #Execute sequence 9 |
| `>How far do you want to go? 20` | #Line 2 text, and numeric entry from line 3 |
| `20` | #The distance value is displayed |
| `>` | #Motor moves 20 user units |

## KBQ : Keyboard Input (Quiet)

| | |
|---|---|
| **Execution Mode** | Sequence |
| **Syntax** | *variable* = KBQ |
| **Range** | *variable* refers to any numeric *variable* which sequences can write to. |
| | Actual permitted range depends on *variable* -Max.Num to +Max.Num |
| **See Also** | KB, SAS, SACS, VIEW |
| **Description** | KBQ accepts a numeric value from the serial port, and assigns that value to *variable*. |
| | 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. |
| | 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. |
| | Sequence execution is effectively suspended while waiting to receive a valid numeric value. |
| | 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. |
| | |
| | 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. 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. |
| **Memo** | In a daisy chain configuration (ID other than *), all output from sequence commands is suppressed unless the device has been previously addressed (via TALK or @). The KB and KBQ commands will not receive input unless the device has been previously addressed. |
| **Example** | |

| Command | Description |
|---|---|
| `>LIST 10` | #List sequence 10 |
| | |
| `(  1) VR=10` | #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? -37.5` | #Line 2 text, and numeric entry from line 3 |
| `Moving :-37.5` | #Exact output of lines 4 and 5 |
| | Motor moves -37.5 user units |

## LIMP, LIMN  : Setting of Software Position Limits (Positive Direction, Negative Direction)

| | |
|---|---|
| **Execution Mode** | Immediate and Sequence |
| **Syntax** | LIMP n: Maximum Permitted Position<br>LIMN n: Minimum Permitted Position |
| **Range** | n =  -MAXPOS to +MAXPOS (user units) |
| **Factory Setting** | 0 |
| **SAVEPRM & RESET** | Required to execute any changes made to the parameter value and to save 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 factory setting. |
| **Access** | READ and WRITE<br>READ only in Sequences |
| **See Also** | SLACT, PC, MGHP, MGHN, EHOME, ALM, ALMACT, OTACT |
| **Description** | When SLACT=1, software position limits LIMP and LIMN are enforced, provided the electrical home is set by a homing action (EHOME, MGHP, MGHN). |

The action of software position limits depends on the type of operation as following chart.

| | Continuous Motion (MCP/MCN) | Index Motion (MI) | Absolute Motion (MA) | Homing Motion* (MGHP/MGHN/EHOME) |
|---|---|---|---|---|
| Software limit action | Motion stops when exceeding software limits | Motion does not start if target position is out of software limits | | Software limits are disabled during motion |
| Stopping motion | Immediate stop or deceleration stop depending on OTACT parameter | - | - | - |
| Condition after above action | Activate ALARM (67h) and/or disable motor current depending on ALMACT parameter | | Display error message | - |

∗ A software position limit alarm may be triggered after a homing operation if PC=0 is not between LIMP and LIMN.

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.  MCP or MCN can be executed, if the motor would move in the direction of the operational range.

| | |
|---|---|
| **Note** | If LIMP=LIMN=0, software position limit checking is disabled, even if SLACT=1.  LIMP and LIMN should be set to appropriate values before enabling software position limit checking. |

| Example | Command | Description |
|---|---|---|
| | >LIMP 10 | #Set positive motion limit |
| |  LIMP=0(10) Rev | |
| | >LIMN -10 | #Set negative motion limit |
| |  LIMN=0(-10) Rev | |
| | >SLACT 1 | #Set software limit enable |
| |  SLACT=0(1) | |
| | >INHOME 1 | #Configure HOME input only |
| |  INHOME=0(1) | |
| | >HOMETYP 8 | #Set Home type. Use software limit |
| |  HOMETYP=8 | instead of LSP, LSN |
| | >SAVEPRM | |
| |  (EEPROM has been written 2 times) | |
| |  Enter Y to proceed, other key to cancel. y | #"y" entered to proceed |
| |  Saving Parameters........OK. | |
| | >RESET | #Reset device to activate changes |
| |  Resetting system. | |
| | ------------------------------------------- | |
| |         CM10-* | |
| |         Controller Module | |
| |         Software Version: *.** | |
| |         Copyright 2010 | |
| |         ORIENTAL MOTOR CO., LTD. | |
| | ------------------------------------------- | |
| | >LIMP | #Confirm settings |
| |  LIMP=10(10) Rev | |
| | >LIMN | |
| |  LIMN=-10(-10) Rev | |
| | >SLACT | |
| |  SLACT=1(1) | |
| | >ALMMSG 2 | #Enable alarm messages |
| |  ALMMSG=2 [Alarm+Warning] | |
| | >MGHP | #Start seek mechanical home |
| | >SIGHOMEP | #MGHP finished, check HOMEP signal |
| |  SIGHOMEP=1 | |
| | >MCP | #Move continuously, positive |
| | >Over travel: software position limit | #Limit detected |
| | detected. | |
| | >PC | #Check PC |
| |  PC=10.001 Rev | #Just over LIMP |
| | > | |

- 270 -

## LINKx : Link Control for Link Segment 'x'

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | LINKx=n ( x is a number of linked segments: x=0 to 2.) |
| **Range** | n = 0: Segment (x) Terminates Motion<br>1: Link Segment (x) to Segment (x+1) |
| **Factory Setting** | 0 |
| **SAVEPRM** | 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 factory setting. |
| **Access** | READ and WRITE |
| **See Also** | DISx, INCABSx, MIx, TA, TD, VRx, VS |
| **Description** | LINKx control whether linked motion segment x is linked to the next segment, or not.<br><br>If LINKx=0, the motion segment defined by DISx and VRx will terminate.<br><br>If LINKx=1, the motion segment defined by DISx and VRx will not terminate: motion will proceed to motion segment (x+1). |

| **Example** | Command | Description |
|---|---|---|
| | >VR0 5 | #Set the velocity for link segment 0 to 5 user units/second |
| | VR0=5 in./sec | #Device response |
| | >DIS0 10 | #Set the distance for link segment 0 to 10 user units |
| | DIS0=10 in. | #Device response |
| | >INCABS0 1 | #Set the move type for link segment 0 to incremental |
| | INCABS0=1 [INC] | #Device response |
| | >LINK0 1 | #Enable the link between link segments 1 and 2 |
| | LINK0=1 | #Device response |
| | >VR1 10 | #Link segment 1 velocity equals 10 user units/s |
| | VR1=10 in./sec | #Device response |
| | >DIS1 20 | #Link segment 1 distance equals 20 user units |
| | DIS1=20 in. | #Device response |
| | >INCABS1 0 | #Set the move type for link segment 1 to absolute |
| | INCABS1=0 [ABS] | #Device response |
| | >LINK1 0 | #Unlink segment 1 from segment 2 |
| | LINK1=0 | #Device response |
| | >MI0 | #Start the linked operation motion |

## LIST : List Sequence Contents

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | LIST target [startline] [endline] |
| **Range** | target can be the name or number of any existing sequence<br>[startline] is an optional line number.<br>[endline] is an optional line number, if [startline] is specified.  If given, it must not be less than [startline]. |
| **See Also** | DIR, EDIT |
| **Description** | LIST lists the contents of a stored sequence.<br><br>If [startline] and [endline] are not specified, the entire sequence is listed.<br><br>If [startline] is specified, output starts with line [startline].<br><br>If [endline] is specified, output ends after line [endline].<br><br>If [startline] and [endline] are specified, the sequence between [startline] and [endline] is listed. |
| **Example** | Command                     Description |

```
>LIST PROGRAM10 2 5        #List sequence PROGRAM10, from line 2 through 5

(  2) LOOP 5               #Partial contents of sequence PROGRAM10
(  3)    MI
(  4)    MEND
(  5)    WAIT 1.0
>
```

# LISTVAR : Lists all User-defined Variables

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | LISTVAR |
| **See Also** | CLEARVAR, CREATEVAR, N_xxx, S_xxx |
| **Description** | LISTVAR lists the names and values of all user-defined variables (String – S_xxx and Numeric – N_xxx) |
| **Example** | |

| Command | Description |
|---|---|
| >LISTVAR | #List all user-defined variables |

```
   ##  N_name      Numeric Data        #List for numeric user-defined variables
   ==  ==========  =============
   1   PRICE       0                    #Variables created but not assigned a value
   2   QUANTITY    0                    show 0
   3   LOT         32
   4   SERIAL      4583274
   5   SIZE        0
   6   LENGTH      106
   7   WIDTH       60
   8   WEIGHT      0.95
   9               0                    #Empty (available) slots have no name
   10              0
   ##  S_name      String Data          #List for string user-defined variables
   ==  ==========  ====================
   1   NAME        IIM
   2   STATUS      Same day shipping OK
   3   MESSAGE
   4   UNIT        Kilogram
   5   COUNTRY     USA
   6
   7
   8
   9
   10
>
```

## LOCK  : Lock Sequence

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | LOCK target |
| **Range** | target can be the name or number of any existing sequence. |
| **Commands not Allowed** | RUN |
| **See Also** | DEL, DIR, EDIT, UNLOCK |
| **Description** | LOCK prevents changes to a sequence.<br><br>A locked sequence cannot be deleted, renamed, or overwritten (by COPY or EDIT).<br><br>A locked sequence can be unlocked with the UNLOCK command.<br><br>The sequence directory listing (DIR command) shows the lock status for all sequences. |
| **Caution** | **A locked sequence will be cleared by CLEARSEQ or CLEARALL: the lock status offers no protection for these operations.** |

| **Example** | Command | Description |
|---|---|---|
| | >LOCK PROG1 | #Lock the sequence named PROG1 from deletion |
| | >DEL PROG1 | #Attempt to delete the PROG1 sequence |
| | Error: Sequence is locked. | #Device's response, unable to delete PROG1 |
| | >DIR | #Query the directory sequence |

```
  ##  Name        TextSize  Locked
  ==  ==========  ========  ======
   0  PROG1             37  Locked

  Total:   1
  Executable memory:     32 bytes used of  6144 bytes total,    1 percent.
  Storage memory:        77 bytes used of 21775 bytes total,    0 percent.
>
```

## LOOP  : Begin Counted LOOP Block

| | |
|---|---|
| **Execution Mode** | Sequence |
| **Syntax** | LOOP n |
| **Range** | n = 1 to Max.Num (integer values), Loop Count |
| **See Also** | BREAKL, ENDL, WHILE, WEND |
| **Description** | LOOP begins a "loop block" structure, which must be terminated later in the sequence by a corresponding ENDL (end loop) command. |
| | 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). |
| | 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. |
| | Block structures (IF-ENDIF , LOOP-ENDL, WHILE-WEND) can be nested up to 8 levels deep. |
| **Example** | Command                Description |

```
>LIST 27              #List sequence 27

(  1) DIS=1           #Distance equals 1 user unit
(  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
>
```

## MA : Start Absolute Motion to the Specified Destination

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | MA n |
| **Range** | n = -MAXPOS to +MAXPOS (user units)<br>In immediate mode, 'n' can be a constant or any POS [x] position array variable.<br>In a sequence, 'n' can be a constant or any variable which can be read within a sequence. |
| **Commands not Allowed** | MOVE |
| **See Also** | DPR, MCP, MCN, MI, PC,  TEACH, UU, MEND, CV |
| **Description** | MA starts a point-to-point motion to position "n."<br><br>Motion velocity is determined by running velocity (VR). Start velocity (VS), acceleration time (TA), and deceleration time (TD) are effective. The 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.'<br><br>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.<br><br>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. |
| **Note** | 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 motion commands cannot be executed until the motion is complete.<br><br>To check that motion is finished, monitor SIGMOVE or SIGEND.  In a sequence, the MEND command provides a convenient way to suspend sequence processing until motion is finished. |

**Example**

| Command | Description |
|---|---|
| >LIST MOVEABS | |
| | |
| (  1) PC=0 | #Set PC=0 |
| (  2) TA=0.1; TD=0.1 | #Set ramp times |
| (  3) VS=0; VR=10 | #Set velocities |
| (  4) LOOP | |
| (  5)   SAS Position 1 | #Message-1 |
| (  6)   MA 0.25 | #Move to 0.25 user unit |
| (  7)   MEND; WAIT 1 | |
| (  8)   SAS Position 2 | #Message-2 |
| (  9)   MA 0.75 | #Move to 0.75 user unit |
| ( 10)   MEND; WAIT 1 | |
| ( 11)   SAS Position 3 | #Message-3 |
| ( 12)   MA 0.5 | #Move to 0.5 user unit |
| ( 13)   MEND; WAIT 1 | |
| ( 14)   SAS Position 4 | #Message-4 |
| ( 15)   MA 0.75 | #Move to 0.75 user unit |
| ( 16)   MEND; WAIT 1 | |
| ( 17)   SAS Position 5 | #Message-5 |
| ( 18)   MA 1.0 | #Move to 1.0 user unit |
| ( 19)   MEND | |
| ( 20)   SAS End Session. Go to next. | #Message-6 |
| ( 21)   WAIT 2 | |
| ( 22) ENDL | |
| >RUN MOVEABS | |
| >Position 1 | #Message-1 |
| >Position 2 | |
| >Position 3 | |
| >Position 4 | |
| >Position 5 | #Message-5 |
| >End Session. Go to next. | #Message-6 |
| >Position 1 | |
| >Position 2 | |
| >Position 3 | |
| >Position 4 | |
| >Position 5 | |
| >End Session. Go to next. | |
| > | |

## MAXEC : Maximum Encoder Count

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | MAXEC |
| **Factory Setting** | 500000000: **CM10-1**, **4**, **5**, **SCX10** |
| | 100000000: **CM10-2** |
| | 50000000: **CM10-3** |
| **Access** | READ |
| **See Also** | EC, MAXPOS, ER |
| **Description** | MAXEC is the largest permitted value for encoder count (EC).  EC must be between -MAXEC and +MAXEC. |
| | MAXEC is determined by MAXPOS (maximum position value) and ER (encoder resolution). |

| **Example** | Command | Description |
|---|---|---|
| | >MAXEC | #Query the maximum encoder count |
| |  MAXEC=500000000(500000000) | |
| | >MAXPOS | #Query the maximum position value |
| |  MAXPOS=500000(500000) Rev | |
| | >ER 100 | #Set the encoder resolution |
| |  ER=1000(100) | |
| |  Position range = +/- 500000(500000) | |
| |  Velocity range = 0.001 - 1240(1240) | |
| | >saveprm | #Save the parameter assignments |
| |  (EEPROM has been written 14 times) | |
| |  Enter Y to proceed, other key to cancel. y | |
| |  Saving Parameters........OK. | |
| | >reset | #Establish the saved parameter values |
| |  Resetting system. | |
| | ------------------------------------------ | |
| |     CM10-* | |
| |     Controller Module | |
| |     Software Version: *.** | |
| |     Copyright 2010 | |
| |     ORIENTAL MOTOR CO., LTD. | |
| | ------------------------------------------ | |
| | >MAXEC | #Query the maximum encoder count |
| |  MAXEC=50000000(50000000) | |
| | > | |

## MAXPOS : Maximum Position Value

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | MAXPOS |
| **Factory Setting** | 500000 (user units) |
| **Access** | READ |
| **See Also** | DPR, GA, GB, MAXVEL |

| | |
|---|---|
| **Description** | MAXPOS (maximum position) is the largest permitted value for position-related parameter entry. Position related parameters (DIS, DISx, PC, OFFSET, SCHGPOS, LIMP, LIMN, ENDACT) must be between –MAXPOS and +MAXPOS. |
| | MAXPOS also defines the limit for absolute motions from initial starting position.  If the system moves outside of –MAXPOS to +MAXPOS, the position command (PC) is reset to zero (0).  The new zero position is located exactly at the former –MAXPOS or +MAXPOS position. |
| | MAXPOS is determined by DPR (distance per revolution), GA and GB (electric gear ratio), MR (motor resolution), ER (encoder resolution), and is automatically updated when these parameters are changed. Both active and future values of MAXPOS are shown as "position range" when MAXPOS is queried, and the new value becomes effective after SAVEPRM and RESET are performed.  At the same time, the values of the position parameters (DIS, DISx, OFFSET, SCHGPOS, LIMP, LIMN, ENDACT) are automatically checked whether or not to be within the MAXPOS (maximum position), and if they are outside the range, the warning message will be sent.  Note that the position parameters in the program and the position array data POS[x] will not be checked.  The new values will become effective after executing save and reset. |

| **Example** | Command | Description |
|---|---|---|
| | `>UU in.` | #Set the user units to in. (inches) |
| | ` UU=in.` | |
| | `>MR 10000` | #Set the motor resolution to 1000 user |
| | ` MR=1000(10000)` | units: device responds with ranges, |
| | ` Position range = +/- 500000(214748)` | active and (future) |
| | ` Velocity range = 0.001 - 1240(124)` | |
| | ` Minimum Movable Distance = +/- 0.001(0.001)` | |
| | `>SAVEPRM` | #Save the parameter assignments |
| | ` (EEPROM has been written 68 times)` | |
| | ` Enter Y to proceed, other key to cancel. y` | |
| | ` Saving Parameters........OK.` | |
| | `>RESET` | #Establish the saved parameter values |
| | ` Resetting system.` | |
| | `------------------------------------------` | |
| | `          CM10-*` | |
| | `          Controller Module` | |
| | `          Software Version: *.**` | |
| | `          Copyright 2010` | |
| | `          ORIENTAL MOTOR CO., LTD.` | |
| | `------------------------------------------` | |
| | `>DPR` | #Confirm the new DPR setting |
| | ` DPR=10(10) in.` | |
| | ` Position range = +/- 500000(500000)` | |
| | ` Velocity range = 0.001 - 24800(24800)` | |
| | ` Minimum Movable Distance = +/- 0.001(0.001)` | |
| | `>MAXPOS` | #Query the maximum position value |
| | ` MAXPOS=500000(500000) in.` | |
| | `>MAXVEL` | #Query the maximum velocity value |
| | ` MAXVEL=24800(24800) in./sec` | |
| | `>` | |

## MAXVEL  : Maximum Velocity Value

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | MAXVEL |
| **Factory Setting** | 1240: **CM10-1**, **4**, **5**, **SCX10**<br>6200: **CM10-2**<br>12400: **CM10-3** |
| **Access** | READ |
| **See Also** | DPR, GA, GB, MAXPOS |
| **Description** | MAXVEL (maximum velocity) is the largest permitted value for velocity-related parameter entry. Velocity related parameters (VS, VR, etc.) must be less than or equal to MAXVEL.<br><br>MAXVEL is determined by DPR (distance per revolution), and electronic gearing parameters GA, GB and MR. It is automatically updated when any of these values are changed.  The new value becomes effective after SAVEPRM and RESET; both active and future values of MAXVEL are shown as "velocity range" when MAXVEL is queried.  At the same time, the values of the velocity parameters (VS, VR, VRx, SCHGVR) are automatically checked whether or not to be within the MAXVEL (maximum velocity), and if they are outside the range, the warning message will be sent.  Note that the velocity parameters in the sequence program will not be checked.  The new values will become effective after executing save and reset.<br><br>Formula of MAXVEL is as follows,<br>MAXVEL=1,240,000$*$DPR$*$GB/(MR$*$GA)<br> Ex. DPR=1, GA=1, GB=1, MR=1000<br>     MAXVEL=1240 |

| **Example** | Command | Description |
|---|---|---|
| | ```
>UU in.
 UU=in.
>DPR 10
 DPR=1(10) in.
 Position range = +/- 500000(500000)
 Velocity range = 0.001 - 2480(24800)
 Minimum Movable Distance = +/- 0.001(0.001)
>SAVEPRM
 (EEPROM has been written 68 times)
 Enter Y to proceed, other key to cancel. y
 Saving Parameters........OK.
>RESET
 Resetting system.
------------------------------------------
         CM10-*
         Controller Module
         Software Version: *.**
         Copyright 2010
         ORIENTAL MOTOR CO., LTD.
------------------------------------------
>DPR
 DPR=10(10) in.
 Position range = +/- 500000(500000)
 Velocity range = 0.001 - 24800(24800)
 Minimum Movable Distance = +/- 0.001(0.001)
>MAXPOS
 MAXPOS=500000(500000) in.
>MAXVEL
 MAXVEL=24800(24800) in./sec
>
``` | #Set the user units to in. (inches)<br><br>#Set the distance per revolution to 10 user units: device responds with ranges, active and (future)<br><br><br><br><br>#Save the parameter assignments<br><br><br><br>#Establish the saved parameter values<br><br><br><br><br><br><br><br><br><br>#Confirm the new DPR setting<br><br><br><br><br>#Query the maximum position value<br><br>#Query the maximum velocity value |

## MBFREEACT : Magnetic Brake Free Action

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | MBFREEACT=n |
| **Range** | n = 0: Driver alarm is unrelated.<br><br>1: MBFREE outputs on both the driver connector of the **CM10**/**SCX10** and the I/O connector become inactive when a driver alarm is active (electromagnetic brake is locked). |
| **Factory Setting** | 0: **CM10-1**, **2**, **3**, **4**, **5**<br>1: **SCX10** |
| **SAVEPRM & RESET** | Required to execute any changes made to the parameter value and to save 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). |
| **Access** | READ and WRITE |
| **See Also** | OUTxxx (OUTMBFREE), SIGxxx (SIGMBFREE), ROUTxxx (ROUTMBFREE), DOUTxxx (DOUTMBFREE), DSIGxxx (DSIGMBFREE), DALARM |
| **Description** | The MBFREEACT is used to select the action of the magnetic brake during a driver alarm condition.<br><br>If the MBFREEACT is set to 0, a driver alarm does not affect the MBFREE outputs both on the driver connector of the **CM10**/**SCX10** and I/O connector.<br><br>If the MBFREEACT is set to 1, the MBFREE outputs on both the driver connector of the **CM10**/**SCX10** and the I/O connector become inactive (The electromagnetic brake is locked.) when a driver alarm is active. This setting is used if the automatic current off function on the driver is set to ON (motor current becomes OFF during an alarm).<br><br>∗ If the DALARM is set to 0, driver alarm signal has no effect on the MBFREE outputs and the MBFREE loses its function. |

| **Example** | Command | Description |
|---|---|---|
| | >MBFREEACT=0<br> MBFREEACT=1(0)<br>>SAVEPRM<br> (EEPROM has been written 10 times)<br> Enter Y to proceed, other key to cancel. y<br> Saving Parameters........OK.<br>>RESET<br> Resetting system.<br>------------------------------------------<br>          CM10-*<br>          Controller Module<br>          Software Version: *.**<br>          Copyright 2010<br>          ORIENTAL MOTOR CO., LTD.<br>------------------------------------------<br>>MBFREEACT<br> MBFREEACT=0(0)<br>> | #Set the MBFREEACT to 0<br><br>#Save the parameter assignments<br><br><br><br><br>#Establish the saved parameter values<br><br><br><br><br><br><br><br><br>#Query new value |

## MCP, MCN : Move Continuously Positive, Move Continuously Negative

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | MCP, MCN |
| **See Also** | <ESC>, ABORT, DIRINV, DPR, PSTOP, INxxx (INLSP, INLSN, INMSTOP, INPAUSE), xxxLV (INxLV, MSTOPLV), LIMP, LIMN, MSTOPACT, PAUSE, TA, TD, UU, VR, VS |
| **Description** | MCP and MCN start continuous motions, with no defined final position.  MCP starts moving in the positive direction, and MCN starts moving in the negative direction.<br><br>Motion velocity is determined by running velocity (VR). Start velocity (VS), acceleration time (TA) and deceleration time (TD) are effective.<br><br>Motion continues until the system is commanded to stop or an alarm condition occurs.<br><br>Velocity can be changed while a continuous motion is in progress, by changing the value of VR and re-issuing the MCP or MCN command.  The direction cannot be changed: MCP cannot be issued while an MCN motion is active, or vise versa. These conditions cause an error message if attempted at the command prompt, and an alarm (alarm code: A0h) if attempted in a sequence.<br><br>MCP and MCN cannot be used while other motions are in progress (e.g. MI, MA, EHOME), 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.<br><br>The MCP/MCN function may also be executed via the MCP/MCN input on the I/O connector if assigned and/or the CANopen remote I/O. When detailed explanation is required, see "6.4 Connecting the I/O signals" on page 23 for I/O connector, and see "10.4 Controlling I/O Message (PDO)" on page 134 for CANopen remote I/O. |
| **Note** | MCP and MCN 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 motion commands cannot be executed until the motion is complete.<br><br>To check that motion is finished, monitor SIGMOVE or SIGEND.  In a sequence, the MEND command provides a convenient way to suspend sequence processing until motion is finished. |

| Example | Command | Description |
|---|---|---|

```
>LIST VCHANGE

(  1) TA 0.5; TD 0.5; VR 1
(  2) MCP                                      #Move continuously (positive)
(  3) LOOP
(  4)    IF (IN1=1)
(  5)      VR=VR+1; MCP                        #Increase speed
(  6)      SAS Increase speed by 1 rev/sec     #Send message 1
(  7)      WAIT TA
(  8)      WHILE (IN1=1); WEND
(  9)    ENDIF
( 10)    IF (IN2=1)
( 11)      IF (VR!=1)
( 12)        VR=VR-1; MCP                      #Decrease speed
( 13)        SAS Decrease speed by 1 rev/sec   #Send message 2
( 14)        WAIT TD
( 15)        WHILE (IN2=1); WEND
( 16)      ELSE
( 17)        SSTOP                             #Soft stop
( 18)        SAS Reached endpoint, End Process #Send message 3
( 19)        RET
( 20)      ENDIF
( 21)    ENDIF
( 22) ENDL
>RUN VCHANGE
>Increase speed by 1 rev/sec                   #Message 1
>Increase speed by 1 rev/sec                   #Message 1
>Increase speed by 1 rev/sec                   #Message 1
>Decrease speed by 1 rev/sec                   #Message 2
>Decrease speed by 1 rev/sec                   #Message 2
>Decrease speed by 1 rev/sec                   #Message 2
>Reached endpoint, End Process                 #Message 3: stopped
>
```

## MEND : Wait for Motion End

| | |
|---|---|
| **Execution Mode** | Sequence |
| **Syntax** | MEND |
| **See Also** | SIGxxx (SIGMOVE, SIGEND), WHILE, WEND, IF, ENDIF |
| **Description** | MEND suspends sequence processing until motion is complete. |
| | 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. |
| | MEND is equivalent to WHILE (SIGMOVE=1); WEND |
| | All 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. |
| | MEND refers to the system END signal at the end of each motion. An alarm condition (alarm code: 6Fh) occurs if the END signal is not found. |

| **Example** | Command | Description |
|---|---|---|

```
>LIST MOVEABS

(  1)  PC=0                              #Set PC=0
(  2)  TA=0.1; TD=0.1                    #Set ramp times
(  3)  VS=0; VR=10                       #Set velocities
(  4)  LOOP
(  5)    SAS Position 1                  #Message-1
(  6)    MA 0.25                         #Move to 0.25 user unit
(  7)    MEND; WAIT 1
(  8)    SAS Position 2                  #Message-2
(  9)    MA 0.75                         #Move to 0.75 user unit
( 10)    MEND; WAIT 1
( 11)    SAS Position 3                  #Message-3
( 12)    MA 0.5                          #Move to 0.5 user unit
( 13)    MEND; WAIT 1
( 14)    SAS Position 4                  #Message-4
( 15)    MA 0.75                         #Move to 0.75 user unit
( 16)    MEND; WAIT 1
( 17)    SAS Position 5                  #Message-5
( 18)    MA 1.0                          #Move to 1.0 user unit
( 19)    MEND
( 20)    SAS End Session. Go to next.    #Message-6
( 21)    WAIT 2
( 22)  ENDL
>RUN MOVEABS
>Position 1                              #Message-1
>Position 2
>Position 3
>Position 4
>Position 5                              #Message-5
>End Session. Go to next.               #Message-6
>Position 1
>Position 2
>Position 3
>Position 4
>Position 5
>End Session. Go to next.
>
```

## MGHP, MGHN : Move Go Home Positive, Move Go Home Negative

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | MGHP (move go home positive)<br>MGHN (move go home negative) |
| **Commands not Allowed** | MOVE |
| **See Also** | DIRINV, INxxx (INHOME, INLSP, INLSN), RINxxx (RINHOME, RINLSP, RINLSN), HOMETYP, xxxLV (HOMELV), PC, OFFSET, OUTxxx (OUTHOMEP), OUTSG, SIGxxx (SIGHOMEP) |
| **Description** | MGHP and MGHN start motion patterns, attempting to find a mechanical home position which links position zero (PC=0) to an application reference signal.<br><br>MGHP starts moving in the positive direction, and MGHN starts moving in the negative direction. (When HOMETYP=12 is selected, direction is determined by the driver setting.) The process may involve moving in both directions before concluding. MGHP and MGHN 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).<br><br>The actual motion pattern and signal requirements are determined by HOMETYP. Depending on HOMETYP, one or more of system input signals LSP, LSN, and HOME must be assigned to an input, before executing MGHP or MGHN. If the signal requirements are not met, the home process will not start, and an error message will be sent (immediate mode) or an alarm will be set (Sequence: alarm code 70h). See HOMETYP in this chapter, and "8.2.5 Mechanical Home Seeking" on page 83 for more information.<br><br>The velocities and acceleration and deceleration times used for the home seeking process are determined by start velocity VS and run velocity VR, and acceleration and deceleration times TA and TD, at the time the process starts.<br><br>If the home process completes successfully, the position command (PC) is set to zero (0) and system output signal SIGHOMEP is set to one (1). If configured, the HOMEP output becomes active.<br><br>It is possible to set the position that has moved from the mechanical home to an electrical home using the OFFSET command.<br><br>Software position limits LIMP and LIMN are disabled while the homing process is active. If the system has been configured to used software position limits (SLACT=1) and the limits have been configured (LIMP and LIMN not both 0), the limits are enabled after successful completion of a homing process.<br><br>MGHP and MGHN cannot be used while other motions are in progress (e.g. MI, MA, EHOME), 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.<br><br>The MGHP/MGHN function may also be executed via the MGHP/MGHN input on the I/O connector if assigned and/or the CANopen remote I/O. When detailed explanation is required, see "6.4 Connecting the I/O signals" on page 23 for I/O connector, and see "10.4 Controlling I/O Message (PDO)" on page 134 for CANopen remote I/O. |
| **Note** | MGHP and MGHN 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.<br><br>To check that motion is finished, monitor SIGMOVE or SIGEND. In a sequence, the MEND command provides a convenient way to suspend sequence processing until motion is finished. |

| **Example** | Command | Description |
|---|---|---|
| | >INHOME<br> INHOME=1(1) | #Check HOME input configuration |
| | >VS 1<br> VS=1 mm/sec | #Set start velocity VS to 1 mm/second |
| | >VR 20<br> VR=20 mm/sec | #Set run velocity VR to 20 mm/second |
| | >HOMETYP 4<br> HOMETYP=4 | #Use HOME, LSP, LSN |
| | >MGHP | #Start seeking home, positive direction |
| | >SIGMOVE<br> SIGMOVE=0 | #Check MOVE signal (after motion)<br>#MOVE is OFF |
| | >SIGHOMEP<br> SIGHOMEP=1 | #Check HOMEP signal<br>#HOMEP is ON (home is found, success) |
| | >PC<br> PC=0 mm<br>> | #Check position command PC<br>#Automatically zeroed when homing succeeded |

## MI  : Start Incremental Motion, Distance DIS

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | MI |
| **Commands not Allowed** | MOVE |
| **See Also** | DPR, DIS, MA, TA, TD, UU, VR, VS, CV |
| **Description** | MI starts a point-to-point incremental motion. |
| | 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. |
| | The speed may be changed while the motion is in progress, using the change velocity command (CV).  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. |
| | 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. |
| **Note** | 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. |
| | To check that motion is finished, monitor SIGMOVE or SIGEND.  In a sequence, the MEND command provides a convenient way to suspend sequence processing until motion is finished. |

**Example**

| Command | Description |
|---|---|
| >LIST UPANDDOWN | #List sequence UPANDDOWN |
| | |
| (  1) VS 0.1 | #Start velocity: 0.1 |
| (  2) VR 10 | #Run velocity: 10 |
| (  3) DIS 150 | #Distance: 150 |
| (  4) TA 1 | #Going up: long acceleration time, compared to… |
| (  5) TD 0.1 | #…short deceleration time |
| (  6) MI | #Start incremental motion |
| (  7) MEND | #Wait for motion to finish |
| (  8) TA 0.1 | #Going down: short acceleration time, compared to… |
| (  9) TD 1 | #...long deceleration time |
| ( 10) MA 0 | #Start absolute motion, back to 0 |
| ( 11) MEND | #Wait for motion to complete |
| > | |

## MIx : Start Linked Motion at Link Segment 'x'

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | MIx |
| **Range** | 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 |
| **Commands not Allowed** | MOVE |
| **See Also** | DISx, DPR, INCABSx, LINKx, MIx, TA, TD, UU, VRx, VS |
| **Description** | 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: |
| | - INCABSx determines whether segment 'x' is an absolute motion segment (INCABSx=0, move to a destination) or an incremental motion segment (INCABSx=0, move by a distance). |
| | - DISx is the destination (INCABSx=0) or distance (INCABSx=1) of segment 'x'. |
| | - VRx is the running speed for the segment 'x'. |
| | 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). (Note: There is no LINK3.) |
| | 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. |
| | 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. |
| | 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. |
| **Note** | 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 70h is set, and sequence processing terminates.) |
| | 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. |
| | MIx 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. |
| | To check that motion is finished, monitor SIGMOVE or SIGEND. In a sequence, the MEND command provides a convenient way to suspend sequence processing until motion is finished. |

| | Command | Description |
|---|---|---|
| **Example** | >UU in | #Set user units to in. (inches) |
| | UU=in | #Device response |
| | >VR1 5 | #Set the velocity for linked move 1 to 5 user units/second |
| | VR1=5 in/sec | #Device response |
| | >DIS1 10 | #Set the distance for linked move 1 to 10 user units |
| | DIS1=10 in | #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 10 | #Linked move #2 velocity equals 10 user units/second |
| | VR2=10 in/sec | #Device response |
| | >INCABS2 0 | #Set the move type for linked motion 2 to absolute |
| | INCABS2=0 [ABS] | #Device response |
| | >DIS2 20 | #Linked move 2: destination is position 20 user units |
| | DIS2=20 in | #Device response |
| | >LINK2 0 | #"Unlink" link 2 from link 3 |
| | LINK2=0 | #Device response |
| | >MI1 | #Start the linked operation motion |
| | > | |

- 289 -

## MR : Motor Resolution

| | |
|---|---|
| **Execution Mode** | Immediate and Sequence |
| **Syntax** | MR=n |
| **Range** | n = 10 to 51200 |
| **Factory Setting** | 100: **CM10-3**<br>200: **CM10-2**<br>1000: **CM10-1**, **4**, **5**, **SCX10** |
| **SAVEPRM & RESET** | Required to execute any changes made to the parameter value and to save 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 factory setting. |
| **Access** | READ and WRITE<br>READ only in Sequences |
| **See Also** | DPR, UU, EC, ER, PC, PE |
| **Description** | Sets the motor resolution (pulse/rev).<br><br>This is required when setting the user unit (DPR, UU). |
| **Note** | If MR is changed, MAXPOS (maximum position=position range), MAXVEL (maximum velocity=velocity range) or minimum travel distance (minimum movable distance) may automatically be changed.<br><br>When executing the MR command, these pre-change and post-change values are shown.  Position parameters (DIS, DISx, OFFSET, SCHGPOS, LIMP, LIMN, ENDACT) and velocity parameters (VS, VR, VRx, SCHGVR) are automatically checked whether or not to be within the MAXPOS (maximum position) and MAXVEL (maximum velocity) respectively, and if they are outside the range, the warning message will be sent.  Note that the position/velocity parameters in the sequence program and the position array data POS[x] will not be checked. Check whether the required resolution is obtained. The minimum movable distance, which is the actual travel distance, can be checked by the TEACH command.<br><br>If MR is changed, the actual travel distance or velocity is changed.  Check whether the present settings of the position/velocity parameters or those values in the program are appropriate. |

**Example**

| Command | Description |
|---|---|
| >MR=51200 | #Set the motor resolution |
| MR=500(51200)<br> Position range = +/- 500000(41943) Velocity range = 0.001 - 2480(24.218)<br> Minimum Movable Distance = +/- 0.001(0.001) | #Device response |
| >SAVEPRM | #Save the parameter assignments |
| (EEPROM has been written 21 times) | #Device response |
| Enter Y to proceed, other key to cancel. y | #Device response |
| Saving Parameters........OK.<br>>RESET<br> Resetting system.<br>-------------------------------------<br>         CM10-*<br>         Controller Module<br>         Software Version: *.**<br>         Copyright 2010<br>         ORIENTAL MOTOR CO., LTD.<br>------------------------------------- | |
| >MR | #Confirm new value |
| MR =51200(51200)<br> Position range = +/- 41943(41943)<br> Velocity range = 0.001 - 24.218(24.218)<br> Minimum Movable Distance = +/- 0.001(0.001) | |

# MSTOP : Motor Stop

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | MSTOP |
| **See Also** | <ESC>, ABORT, HSTOP, INxxx (INMSTOP), MSTOPACT, xxxLV (MSTOPLV), PSTOP, SSTOP |
| **Description** | MSTOP causes the motor to stop. This command does not stop a sequence program. |
| | Stop action can be a soft stop with controlled deceleration, or a hard stop (as quickly as possible), depending on motor stop action (MSTOPACT). |
| | The MSTOP function may also be executed via the MSTOP input on the I/O connector and/or the CANopen remote I/O if assigned. When detailed explanation is required, see "6.4 Connecting the I/O Signals" on page 23 for I/O connector, and see "10.4 Controlling I/O Message (PDO)" on page 134 for CANopen remote I/O. |
| | If the MSTOP (sequence, assignment of the I/O signal input) is used to stop the motion, all of the stop action can be changed at a time using MSTOPACT. MSTOP (command or input) can be used with MSTOPACT in a multi-axis setting, if multiple devices need to be stopped, but some devices need to soft stop and some need to hard stop. |
| **Caution** | **Ensure the MSTOPACT is set properly prior to asserting the MSTOP input or executing the MSTOP command.** |
| | **If MSTOPACT=0, the MSTOP command will attempt to cause the motor to stop rotating immediately. Use caution when stopping a high speed load using the MSTOP command. The actual distance traveled during a Motor Stop depends on velocity, load, and current settings.** |

**Example**

| Command | Description |
|---|---|
| >MSTOPACT | #Check MSTOPACT |
| MSTOPACT=1(1) | #MSTOPACT: soft stop |
| >VR 10 | #Set the running velocity to 10 user units/second |
| VR=10 Rev/sec | |
| >MCP | #Start the motor moving in the positive direction |
| >MSTOP | #Stop the motor based on MSTOPACT setting |
| > | |

## MSTOPACT  : Motor Stop Action

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | MSTOPACT=n |
| **Range** | n =  0: Hard Stop (stop as quickly as possible)<br>        1: Soft Stop (controlled deceleration over time) |
| **Factory Setting** | 0 |
| **SAVEPRM & RESET** | Required to execute any changes made to the parameter value and to save 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 factory setting. |
| **Access** | READ and WRITE |
| **See Also** | INxxx (INMSTOP), xxxLV (MSTOPLV), SIGxxx (SIGMSTOP), MSTOP, HSTOP, SSTOP |
| **Description** | MSTOPACT establishes the motor action upon activation of the MSTOP (motor stop) input and the MSTOP command.<br><br>If MSTOPACT=0, the MSTOP input and command stop the motor as quickly as possible (hard stop). MSTOP behaves exactly the same as HSTOP.<br><br>If MSTOPACT=1, the MSTOP input and command stop the motor by controlled deceleration (soft stop). MSTOP behaves exactly the same as SSTOP. |
| **Caution** | **Ensure the MSTOPACT is set properly prior to asserting the MSTOP input or executing the MSTOP command.** |
| **Example** | |

| Command | Description |
|---|---|
| >MSTOPACT | #Check the MSTOPACT setting |
| MSTOPACT=1(1) | #Set for soft stop action |
| >VS 0; VR 4.25 | #Set start velocity 0, run velocity 4.25 RPS |
| VS=0 Rev/sec | |
| VR=4.25 Rev/sec | |
| >TA 0.05; TD 0.025 | #Acceleration time 0.05, Deceleration time 0.025 |
| TA=0.05 | |
| TD=0.025 | |
| >MCP | #Start continuous motion, positive direction |
| >VC | #Check velocity command |
| VC=4.25 Rev/sec | #Velocity has reached running speed |
| >MSTOP | #Stop: will be a soft stop because MSTOPACT is 1 |
| > | |

# N_xxx  : User-defined Numeric Variables

| | |
|---|---|
| **Execution Mode** | Immediate and Sequence |
| **Syntax** | N_xxx=n        xxx=Variable Name: 1 to 10 Alphanumeric Characters (N_xxx can be the name of any existing user-defined numeric variable) |
| **Range** | n =  -Maximum Number to +Maximum Number |
| **Factory Setting** | 0 |
| **SAVEPRM** | 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 factory setting. |
| **Access** | READ and WRITE |
| **See Also** | A to Z, CREATEVAR, DELETEVAR, LISTVAR, S_xxx |
| **Description** | General purpose, user-defined numeric variables. A user-defined variable must be created with CREATEVAR before it can be used.  After it has been created, it can be used in the same way as the general purpose variables A to Z, except that it cannot be used as the argument for a CALL statement.  (CALL N_xxx attempts to call a sequence named N_xxx, not sequence number N_xxx.) |
| | User-defined variables have names, to increase readability. They allow constructs such as:<br>LOOP N_COUNT<br>DIS = N_LONGMOVE<br>  - which help to make the variable's context and purpose clear. |
| | Using user-defined variables in a sequence is slightly slower than using general purpose variable A to Z, because the system requires extra time to search for the variable by name before accessing it.  This may be important in applications with very tight timing requirements. |
| | In immediate mode, user-defined variables may only be set and queried. |
| | Within a sequence, user-defined variables may also be used in the following conditions:<br>・Targets or arguments for assignments (e.g. N_TIME=TIMER; DIS=N_LONGMOVE)<br>・Loop Counters (e.g. LOOP N_COUNT)<br>・Conditional Statement Values (e.g. IF (VR>N_NOMINAL))<br>・Parts of Mathematical Expressions (N_SPEED=N_SPEED+N_INCREMENT)<br>・Targets for interactive data entry commands (N_DISTANCE=KBQ) |
| | Refer to the description of A to Z for more information on general variable use. |

| **Example** | Command | Description |
|---|---|---|
| | ```>CREATEVAR N_COUNTS=0```<br>  ```New variable N_COUNTS is added.```<br>  ```N_COUNTS=0``` | #Create user-defined numeric variable named N_COUNTS |
| | ```>CREATEVAR N_TOTAL=10```<br>  ```New variable N_TOTAL is added.```<br>  ```N_TOTAL=10``` | #Create user-defined numeric variable named N_TOTAL |
| | ```>LIST MAIN``` | #List sequence MAIN |
| | ```(  1)  WHILE (N_COUNTS < N_TOTAL)``` | #N_COUNTS, N_TOTAL user-defined variables |
| | ```(  2)    MI; MEND``` | #Start incremental motion; wait until complete |
| | ```(  3)    OUT4 = 1``` | #Set output 4 ON |
| | ```(  4)    WHILE (IN6=0); WEND``` | #Wait for input 6 to go OFF |
| | ```(  5)    OUT4 = 0``` | #Set output 4 OFF |
| | ```(  6)    WHILE (IN6=1); WEND``` | #Wait for input 6 to go on |
| | ```(  7)    N_COUNTS=N_COUNTS+1``` | #Increment N_COUNTS by 1 |
| | ```(  8)  WEND``` | #End of WHILE block |
| | ```>``` | |

## OFFSET  : Offset for Mechanical Home Seeking

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | OFFSET=n |
| **Range** | n =  -MAXPOS to +MAXPOS (user units) |
| **Factory Setting** | 0 |
| **SAVEPRM** | 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 factory setting. |
| **Access** | READ and WRITE |
| **See Also** | HOMETYP, MGHP, MGHN |
| **Description** | OFFSET is the distance to be moved as the last step of a mechanical home seeking operation (MGHP, MGHN). |
| | After the home seeking operation has established a valid home signal (or signal combination: see HOMETYP), the motor moves by the OFFSET 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 OFFSET motion has start velocity VS, running velocity VR, and acceleration and deceleration times TA and TD. |
| | The factory setting 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. |

| **Example** | Command | Description |
|---|---|---|
| | ```
>HOMETYP 6
 HOMETYP=6
>OFFSET -30
 OFFSET=-30 deg
>MGHP
>SIGHOME
 SIGHOME=0
>SIGHOMEP
 SIGHOMEP=1
>PC
 PC=0
>MA 30
>PC
 PC=30 deg
>SIGHOME
 SIGHOME=1
>
``` | #Use HOME and SENSOR. LSx causes reversal<br><br>#OFFSET origin -30 degree from HOME+SENSOR inputs<br><br>#Seek mechanical home, approach from the positive direction<br>#AFTER operation complete: check HOME input<br>#Input is inactive.  We have moved away from the signal<br>#Check HOMEP output<br>#Signal is active. We are at PC=0 after a valid homing operation<br>#Check position command PC<br>#Origin. Expected position count after home<br>#Absolute move to 30 degrees<br>#After motion completes… check PC<br>#PC is 30 degrees<br>#Check home input<br>#Active. Home input and origin are separated by OFFSET |

## OTACT  : Overtravel Action

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | OTACT=n |
| **Range** | n =  0: Hard Stop (stop as quickly as possible)<br>        1: Soft Stop (controlled deceleration over time) |
| **Factory Setting** | 0 |
| **SAVEPRM & RESET** | Required to execute any changes made to the parameter value and to save 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 factory setting. |
| **Access** | READ and WRITE |
| **See Also** | HSTOP, SSTOP, ALMACT, SIGxxx (SIGLSP, SIGLSN), INxxx (INLSP, INLSN), xxxLV (OTLV), LIMP, LIMN, SLACT |
| **Description** | OTACT establishes the stop action taken, when the system detects an over travel input signal (LSP or LSN) or when position exceeds position limits set with LIMP and LIMN.<br><br>If OTACT=0, the system will stop the motor as quickly as possible (hard stop). Also the ACL/DCL signal on the driver connector of the **CM10**/**SCX10** is momentarily output for stopping servo motors and $\alpha_{STEP}$ products immediately.<br><br>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.<br><br>Action after stop (alarm or no alarm, current on or off) is controlled by ALMACT. |
| **Caution** | **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.** |
| **Example** | |

| Command | Description |
|---|---|
| >INLSN 1<br> INLSN=0(1) | #Assign the negative direction limit sensor to input 1 |
| >INLSP 6<br> INLSP=0(6) | #Assign the positive direction limit sensor to input 6 |
| >OTACT 1<br> OTACT=0(1) | #Set the over travel action to hard stop |
| >ALMACT 1<br> ALMACT=2(1) | #Set Alarm Action to 1 (alarm, current on) |
| >LIMN -50<br> LIMN=0(-50) Rev | #Set negative position limit(typically inside hardware limit) |
| >LIMP 50<br> LIMP=0(50) Rev | #Set positive position limit (typically inside hardware limit) |
| >SLACT 1<br> SLACT=0(1) | #Enable software limit checking (after home operation) |
| >SAVEPRM<br> (EEPROM has been written 80 times)<br> Enter Y to proceed, other key to cancel. Y<br> Saving Parameters........OK. | #Save the parameter assignments |
| >RESET<br> Resetting system.<br>------------------------------------------<br>        CM10-*<br>        Controller Module<br>        Software Version: *.**<br>        Copyright 2010<br>        ORIENTAL MOTOR CO., LTD.<br>------------------------------------------ | #Establish the saved parameter values |
| >MGHP<br> > | #Seek home, start in positive direction (if successful, LIMP and LIMN position limits become active) |

## OUT  : General Output Status

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | OUT=n<br><br>∗ "=n" is required only when controlling. |
| **Range** | n =  0 to 15 (integer values)<br><br>/: real time monitor  (immediate mode only) |
| **Factory Setting** | 0 |
| **Access** | READ and WRITE<br>READ only in CANopen |
| **See Also** | INITIO, IO, IN, OUTTEST, OUTxxx, OUTSG, OUTx, REPORT, OUTTEST |
| **Description** | OUT displays or sets the value of all the general purpose outputs, as one integer number.<br><br>The general purpose outputs contribute to the value of OUT as follows:<br><br><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><br>For example, if the general output 2 (2) and general output 4 (8)are active, while all other signals are not active, "OUT=10" is set (2+8=10). When inputting OUT, "OUT=10" is replied.<br><br>And when inputting "OUT=10," the general output 2 (2) and general output 4 (8) become active regardless of the present output status.<br><br>∗ To check or control the status of a single general output, use the OUTx command.<br><br>∗ The OUT value always indicates the internal status of the all general output signals OUT1 to OUT4.  For the output signal assigned the system output signal (HOMEP, ALARM, END etc.), the OUT command recognizes the signal as inactive or zero (except when the signal has become active by the OUT, OUTx command).  Use the OUTSG command in order to refer to the status of all output signals assigned the specific functions.<br><br>∗ All general purpose outputs are in the inactive (OFF) state immediately following system startup. |
| **Note** | All outputs are OFF when device power is off. |
| **Example** | Command ... Description |

| Command | Description |
|---|---|
| ```>IO```<br>``` Inputs  (1-9) = -LS IN2 IN3 IN4 IN5 +LS IN7 IN8 IN9```<br>``` Outputs (1-4) = ALARM OUT2 OUT3 OUT4```<br><br>``` --Inputs--            Outputs```<br>``` 1 2 3 4 5 6 7 8 9 -(SEQ#)- 1 2 3 4```<br>``` 0 0 0 0 0 0 0 0 0 -(  0 )- 0 0 0 0```<br>```>OUT 15```<br>``` OUT=15```<br>```>IO```<br>``` Inputs  (1-9) = -LS IN2 IN3 IN4 IN5 +LS IN7 IN8 IN9```<br>``` Outputs (1-4) = ALARM OUT2 OUT3 OUT4```<br><br>``` --Inputs--            Outputs```<br>``` 1 2 3 4 5 6 7 8 9 -(SEQ#)- 1 2 3 4```<br>``` 0 0 0 0 0 0 0 0 0 -(  0 )- 0 1 1 1```<br>```>``` | #Check IO status<br>#Response: Note that ALARM has been assigned to output 1. outputs 2 to 4 are general purpose.<br>#All outputs reported off.<br>#Set OUT to 15 (all outputs on)<br>#Check IO status again<br><br><br><br>#All outputs are on… expect output 1. Output 1 active state cannot be effected by OUT |

## OUTSG  : System Signal Output Status

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | OUTSG |
| **Range** | n = 0 to 1983<br>/: real time monitor  (immediate mode only) |
| **Access** | READ |
| **See Also** | SIGxxx, OUT, xxxLV, IO, REPORT, OUTTEST |
| **Description** | OUTSG displays the current status of all the system output signals, as one integer number.<br><br>The system output signals contribute to the value of OUTSG as follows: |

| Bit Location | Signal | Contribution to OUTSG If Active |
|---|---|---|
| Bit 10 | ABSDATA | 1024 |
| Bit 9 | LC | 512 |
| Bit 8 | READY | 256 |
| Bit 7 | MBFREE | 128 |
| Bit 6 | - | - |
| Bit 5 | PSTS | 32 |
| Bit 4 | ALARM | 16 |
| Bit 3 | HOMEP | 8 |
| Bit 2 | END | 4 |
| Bit 1 | RUN | 2 |
| Bit 0 | MOVE | 1 |

OUTSG is the sum of the contribution of all active signals:
For example, if the END (4) and MBFREE (128) signals are active, while all other signals are not active, "OUTSG=132" is set (4+128=132). When inputting OUTSG, "OUTSG=132" is replied.

∗ When checking the status of a single system output signal, use the SIGxxx command.

∗ The OUTSG value always indicates the status of the all system output signals.  Note that the signals, which are not output actually due to not assigned to the I/O connector terminal, are also counted in the OUTSG value.

∗ Be careful not to confuse OUTSG with OUT (general output status).  OUT reports the status of general purpose outputs (those outputs which are not assigned to a signal).

| **Example** | Command | Description |
|---|---|---|
| | >OUTSG | #Query the status of the system output signals |
| | OUTSG=3 | #OUTSG equals 3, when sequence is running or motor is moving |
| | > | |

## OUTTEST  : I/O Test Utility

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | OUTTEST |
| **Commands not Allowed** | MOVE, RUN |
| **See Also** | IN, INx, INSG, SIGxxx, OUT, OUTx, OUTSG, IO, REPORT, xxxLV |
| **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: |

| | | | | |
|---|---|---|---|---|
| OUT1 | 1 | OUT2 | 2 |
| OUT3 | 3 | OUT4 | 4 |
| MOVE | M | RUN | R |
| END | E | HOMEP | H |
| ALARM | A | PSTS | P |
| MBFREE | B | READY | D |
| LC | L | | |

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

An ESCAPE key or character exits the OUTTEST process.

OUTTEST is not permitted while a sequence is running, while a motion is in progress, or if the system is in an alarm state.

| **Example** | Command | Description |
|---|---|---|
| | >OUTTEST | #Start the OUTTEST process |
| | ``` *** Input Monitor -- Output Simulator *** ``` | |
| | ``` Inputs (1-9) = IN1 IN2 -LS +LS HOME PSTOP IN7 IN8 IN9 Outputs(1-4) = OUT1(1) OUT2(2) END(E) ALARM(A) ``` | #Assignments and toggle keys shown here |
| | ``` - Use (x) keys to toggle Outputs. - Use <space> to set all outputs to zero. - Use <esc> to exit OUTTEST mode. ``` | |
| | ``` I/O Status Monitor --Inputs---            Outputs 1 2 3 4 5 6 7 8 9 -(SEQ#)- 1 2 3 4 0 0 0 0 1 0 0 0 0 -( 0  )- 0 0 1 0 > ``` | #Active (1) or inactive (0) states shown here #Escape entered: OUTTEST ends |

## OUTx : Individual General Output Control

| | |
|---|---|
| **Execution Mode** | Immediate and Sequence |
| **Syntax** | OUTx=n        (OUTx is a signal name: OUT1 to OUT4) |
| | * "=n" is required only when controlling. |
| **Range** | n = 0: Not Active |
| | 1: Active |
| | /: real time monitor  (immediate mode only) |
| **Factory Setting** | 0 |
| **Access** | READ and WRITE |
| **See Also** | INITIO, OUT, OUTSG, OUTTEST |
| **Description** | OUTx displays or controls the state of general purpose OUTx. |
| | If the output has been assigned to a system output signal, then it is no longer "general purpose." OUTx for these outputs has no affect on the output pins. Use SIGxxx to check the status of assigned system output signals. |
| **Memo** | Even the output signal assigned the system output signal can become active (1) as the general output status using the OUTx command.  For example, even when the OUT1 of the I/O connector is assigned to ALARM, OUT1=1 can be commanded.  However, the actual output terminal of the I/O connector that is assigned to ALARM will not become active.  Note that the OUT (general output control) value becomes 1 at this time. |
| **Example** | 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
>
```

## OUTxxx : System Signal Output Assignment

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | OUTxxx=n |
| | "xxx" represents the signal name to be assigned, and "n" represents the assigned terminal number ("that" becomes the OUTn general output when unassigned). |
| **Range** | n = 0 to 4 |
| **Factory Setting** | 0 (unassigned) |
| **SAVEPRM & RESET** | Required to execute any changes made to the parameter value and to save 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 factory setting. |
| **Access** | READ and WRITE |
| **See Also** | xxxLV, SIGxxx, INITIO, CLEARALL, IO, OUT, OUTSG, OUTx, OUTTEST, REPORT, ROUTxxx, and "See Also" column in the chart below. |
| **Description** | Assign the system output signal to the OUTn of the I/O connector of the **CM10**/**SCX10**. The system signal output assignment is released by "OUTxxx=0" and it becomes the general output OUTn. When executing the INITIO command, the parameter restores to the factory setting. |

| Command | Signal | Description | See Also |
|---|---|---|---|
| OUTALARM | ALARM | Alarm | ALM, ALMCLR |
| OUTEND | END | Motion End | END, DEND, ENDACT |
| OUTHOMEP | HOMEP | Home Position | MGHN, MGHP, EHOME |
| OUTLC | LC | Limiting Condition | DINLC |
| OUTMBFREE | MBFREE | Magnetic Brake Free | FREE, CURRENT |
| OUTMOVE | MOVE | Motor Moving | MEND |
| OUTPSTS | PSTS | Pause Status | PAUSE, PAUSECLR, CONT |
| OUTREADY | READY | Operation Ready | DREADY |
| OUTRUN | RUN | Sequence Running | RUN |

**Example**

| Command | Description |
|---|---|
| >OUTALARM | #Check ALARM assignment |
|  OUTALARM=1(1) | #Assigned to output 1 |
| >OUTALARM 3 | #Change the ALARM signal assignment to output 3 |
|  OUTALARM=1(3) | |
| >SAVEPRM | #Save the parameter assignments |
|  (EEPROM has been written 80 times) | |
|  Enter Y to proceed, other key to cancel. Y | |
|  Saving Parameters........OK. | |
| >RESET | #Establish the saved parameter values |
|  Resetting system. | |
| ------------------------------------------- | |
|         CM10-* | |
|         Controller Module | |
|         Software Version: *.** | |
|         Copyright 2010 | |
|         ORIENTAL MOTOR CO., LTD. | |
| ------------------------------------------- | |
| >OUTALARM | #Confirm the new assignment |
|  OUTALARM=3(3) | |
| > | |

## PABS : Driver Current Position

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | PABS |
| **Range** | -2,147,483.648 to +2,147,483.647 |
| **Access** | READ |
| **See Also** | ABSREQ, ABSREQPC, ABSSTS, ROUTxxx (ROUTABSDATA) |
| **Description** | This is a variable to which the driver current position acquired by ABSREQ or ABSREQPC is written. The variable unit is the user unit. |
| | When referring to the driver current position from the host controller, refer to PABS after executing the ABSREQ (reading driver current position) command or ABSREQPC (reading driver current position/updating internal position) command. |
| | When reading PABS via CANopen, execute the ABSREQ command or the ABSREQPC command first and then execute PABS command after confirming that the ABSDATA output has become 1 (ON) via the remote I/O of CANopen. PABS can be referred to if the ABSDATA output is assigned and the ABSDATA output is 1 (ON). |
| | PABS cannot be read under the following conditions:<br>・Current position has not been read yet since the power is ON.<br>・Data is being read<br>・Although the data was read, a range that could be written was exceeded. |
| **Memo** | The range of the driver current position can be read is "-2,147,483.648 to +2,147,483.647," which is the value after converting to the user unit. |
| **Example** | |

| Command | Description |
|---|---|
| ```
>ABSREQ
 PABS=124.35 Rev
 Driver Status Code = 00
 Driver ALARM Code = 00
>PABS
 PABS=124.35 Rev
``` | Read current position, status and alarm<br>Current position<br>Driver status code<br>Driver alarm code<br><br>Current position |

## PAUSE  : Pause Motion

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | PAUSE |
| **See Also** | SSTOP, CONT, INITIO, INxxx (INPAUSE, INPAUSECL), OUTxxx (OUTPSTS), OUTSG, xxxLV (PAUSECLLV, PAUSELV, PSTSLV), SIGxxx (SIGPAUSE, SIGPAUSECL, SIGPSTS), PAUSECLR |
| **Description** | PAUSE interrupts a motion, stopping the motor by controlled deceleration (soft stop). The applicable motion includes incremental motion (MI), absolute motion (MA) and continuous motion (MCP, MCN). This command does not stop a sequence program. See SSTOP for details on the velocity profile during deceleration. |
| | A motion that has been stopped with the PAUSE can be continued (resumed) using the CONT command or input. If  START input is turned ON while in a paused situation only during sequence execution, the remaining motion will be started (STARTACT=0). |
| | Linked motions, return-to-electrical home operation and mechanical home seeking cannot be paused and resumed: PAUSE causes a soft stop, and CONT is ignored. |
| | 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. Use the PAUSECLR command to clear the remaining motion and not restarting the motion. |
| | The system remains in a "paused" state, until motion is continued (see CONT), or the state is explicitly cleared (with a PAUSECL input), or another motion command is executed. |
| | 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. |
| | If no motion is in process when a PAUSE command is issued, the PAUSE command has no effect. |
| | The PAUSE function may also be executed via the PAUSE input on the I/O connector and/or the CANopen remote I/O if assigned. When detailed explanation is required, see "6.4 Connecting the I/O signals" on page 23 for I/O connector, and see "10.4 Controlling I/O Message (PDO)" on page 134 for CANopen remote I/O. |
| **Note** | 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, PAUSECL input , or new motion). Linked motions, return-to-electrical home Operation and mechanical home seeking cannot be paused and resumed: PAUSE causes a soft stop, and CONT is ignored. |

| **Example** | Command | Description |
|---|---|---|
| | >MCP | #Move continuously (positive) |
| | >PAUSE | #Pause motion |
| | >CONT | #Resume motion |
| | > | |

## PAUSECLR : Pause Clear

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | PAUSECLR |
| **See Also** | PAUSE, CONT, INxxx (INPAUSECL), RINxxx (RINPAUSECL), SIGxxx (SIGPAUSECL), OUTxxx (OUTPSTS) |
| **Description** | PAUSECLR clears the on-going operation state that has been paused by the input of a PAUSE signal or a PAUSE command.  Any remaining motion is canceled. |
| | If the PAUSECLR is commanded while the sequence is running, only remaining portion of the current motion is cleared and the next step of the sequence will be executed, since the PAUSE does not stop the sequence. The PAUSECLR function may also be executed via the PAUSECL input on the I/O connector and/or the CANopen remote I/O if assigned. When detailed explanation is required, see "6.4 Connecting the I/O signals" on page 23 for I/O connector, and see "10.4 Controlling I/O Message (PDO)" on page 134 for CANopen remote I/O. |
| **Note** | A motion that has been stopped with the PAUSE command or input can be continued (resumed) using the CONT command or input, while the PAUSECLR command or PAUSECL input clears remaining motion. See the entries for CONT, PAUSECLR and PAUSE in "6.4.2 Input Signals," "8.3 Stopping Motion and Sequence." |
| **Example** | Command          Description |

```
>MCP          #Move continuously (positive)
>PAUSE        #Pause motion
>PAUSECLR     #Clear paused motion
>
```

## PC : Position Command

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | PC=n |
| **Range** | n = -MAXPOS to +MAXPOS (user units) <br> /: real time monitor (immediate mode only) |
| **Factory Setting** | 0 |
| **Access** | READ and WRITE <br> READ only while motion is in progress |
| **See Also** | EHOME, MA, MGHP, MGHN, MI, PCI, PE, PF, PFI, MAXPOS |

**Description**

PC is the position command (or set point), in user units.

PC is the position that the system has been instructed to go to. The actual motor position is maintained as PF=Position Feedback (when an encoder is connected). The difference between PC and PF is the position error, PE.

PC is set to zero (0) at system startup.

PC is continuously updated by the system:

- In normal operations, PC is updated by the internal motion profiler.

- If current is off, PC is continuously set to actual position PF (to maintain zero position error while the system is freewheeling).

- PC is automatically set to zero (0) after successful completion of a home seeking operation (EHOME, MGHP, MGHN).

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.

**Example**

| Command | Description |
|---|---|
| >LIST ORIGIN | #List sequence named "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 degrees |
| ( 4) MA 0 | #Go to position zero (PC=0), 45 degrees away from HOME input location |
| > | |

## PCI : Incremental Position Command

| | |
|---|---|
| **Execution Mode** | Immediate and Sequence |
| **Syntax** | PCI |
| **Range** | -2*MAXPOS to +2*MAXPOS<br>/: real time monitor  (immediate mode only) |
| **Access** | READ |
| **See Also** | PC, PE, PF, PFI |
| **Description** | PCI is the change in position command PC (position command) since the last motion started.<br><br>PCI is continuously updated by the system.<br><br>PCI is set to zero (0) at system startup and the start of motion.  PCI is undefined immediately after a mechanical home seeking operation completes (MGHP, MGHN). |

**Example**

| Command | Description |
|---|---|
| >LIST AREAOUT2 | #List sequence AREAOUT2 |
| | |
| (  1) DIS 100; VR=10 | #Set distance, velocity |
| (  2) MI | #Start move incremental |
| (  3) SAS Motion started | #Send message 1 |
| (  4) WHILE (PCI<30) | #Wait for PCI to reach 30 |
| (  5) WEND | |
| (  6) SAS Passed 30mm | #Send message 2 |
| (  7) WHILE (PCI<60) | #Wait for PCI to reach 60 |
| (  8) WEND | |
| (  9) SAS Passed 60mm | #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 30mm | #Message 2 |
| >Passed 60mm | #Message 3 |
| >Reached target | #Message 4 |
| > | |

## PE : Position Error

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | PE |
| **Range** | -MAXPOS to +MAXPOS<br>/: real time monitor  (immediate mode only) |
| **Access** | READ |
| **See Also** | ENDACT, PC, PCI, PF, PFI |
| **Description** | PE is the position error, the difference between position command (PC) and actual position (PF), in user unit.<br><br>  $PE = PC - PF$.<br><br>PE is continuously updated by the system, and can be used to monitor the systems response to load conditions. The PE command is used for position confirmation referenced by the ENDACT command and/or a user program.<br><br>When using a stepping motor with an encoder, the status for loss of synchronism can be checked by monitoring PE.  See "8.9 Encoder Function" on page 106, "HOMEDCL (deviation counter clear select at mechanical home seeking operation" on page 86. |

**Example**

| Command | Description |
|---|---|
| (  1) MI | #Start incremental motion |
| (  2) MEND | #Wait for motion to end |
| (  3) IF (PE<-1.8) | #When the PE is smaller than -1.8° |
| (  4)   SAS MISS-STEP MAY HAVE OCCURRED | #Transmit a message |
| (  5) ENDIF | #End the IF statement |
| (  6) IF (PE>1.8) | #When the PE is greater than -1.8° |
| (  7)   SAS MISS-STEP MAY HAVE OCCURRED | #Transmit a message |
| (  8) ENDIF | #End the IF statement |

## PECLR : Position Error Clear

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | PECLR |
| **Commands not Allowed** | MOVE |
| **See Also** | INxxx (INPECLR), xxxLV (PECLRLV), SIGxxx (SIGPECLR), RINxxx (RINPECLR), EC, PC, PE, PF, ENDACT |
| **Description** | PECLR command resets the PE (position error) value to zero (0). |
| | When PECLR command is executed, the PC value is set to equal to PF value. As a result, the PE is reset to zero. When the PC value was differed from the PF value for any cause, the error can be cleared with this command. |
| | Concurrently, the ACL/DCL signal on the driver connector of the **CM10**/**SCX10** is momentary turned ON (when the driver alarm is not generated) and the deviation counter in the driver is cleared. Therefore the error in the **CM10**/**SCX10** and in the driver will be matched (when the driver has a deviation counter clear input and it is connected). |
| | When using a stepping motor with an encoder, once the PECLR command is executed in a condition that no external turning force is applied to the motor shaft, the accurate deviation between the stator and rotor (depends on the size of load) can always be checked by monitoring PE.  See "HOMEDCL (deviation counter clear select at mechanical home seeking operation" on page 86. |
| | The PECLR function may also be executed via the PECLR input on the I/O connector and/or the CANopen remote I/O if assigned. When detailed explanation is required, see "6.4 Connecting the I/O signals" on page 23 for I/O connector, and see "10.4 Controlling I/O Message (PDO)" on page 134 for CANopen remote I/O. |

| **Example** | Command | Description |
|---|---|---|
| | >PC | #Query the position counter |
| | PC=1000 | #Device response |
| | >PF | #Query the position feedback |
| | PF=1234 | #Device response |
| | >PE | #Query the position error |
| | PE=-234 | #Device response |
| | >PECLR | #PECLR command |
| | >PC | #Query the position counter |
| | PC=1234 | #Device response |
| | >PF | #Query the position feedback |
| | PF=1234 | #Device response |
| | >PE | #Query the position error |
| | PE=0 | #Device response |

## PF  : Feedback Position

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | PF=n |
| **Range** | n = −MAXPOS to +MAXPOS (user units) |
| | /: real time monitor  (immediate mode only) |
| **Factory Setting** | 0 |
| **Access** | READ, WRITE |
| | READ only while motion is in progress. |
| **See Also** | EC, ER, PC, PCI, PE, PFI, ENC, ENDACT |
| **Description** | PF is the actual motor position, measured by the position sensor in the motor or the external encoder. (selected by ENC) |
| | PF is generated by EC (encoder count) by the following formula. |
| | $\quad$ PF=EC/ER$*$DPR$*$GB/GA |
| | $\quad\quad$ $*$ Note that PF is in user units where EC (encoder count) is actual number of pulses. |
| | PF is continuously updated by the system. |
| | PF can deviate from the PC (position command), depending on load conditions.  The difference between PC and PF is the position error PE, and used for position confirmation referenced by the ENDACT command and/or a user program. |
| | PF get changed when PC is changed.  For example, if PC=0 and PF=0.001 with some constant load, setting PC=10 adjusts PF to 10.001 (exact value may vary with load and any small shaft motion). |
| **Note** | When the motor has no internal position sensor or external encoder, or when setting "ENC=0" (no use), the PF value is always zero. |

| **Example** | Command | Description |
|---|---|---|
| | >LIST AREAOUT3 | |
| | ( 1) DIS 100; VR=10 | #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<30) | #Wait for PF to reach 30 |
| | ( 6) WEND | |
| | ( 7) SAS Passed 30mm | #Send message 2 |
| | ( 8) WHILE (PF<60) | #Wait for PF to reach 60 |
| | ( 9) WEND | |
| | ( 10) SAS Passed 60mm | #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 30mm | #Message 2 |
| | >Passed 60mm | #Message 3 |
| | >Reached target | #Message 4 |
| | > | |

## PFI : Incremental Feedback Position

| | |
|---|---|
| **Execution Mode** | Immediate and Sequence |
| **Syntax** | PFI |
| **Range** | −2*MAXPOS to +2*MAXPOS (user units) |
| | /: real time monitor (immediate mode only) |
| **Access** | READ |
| **See Also** | PC, PCI, PE, PF |
| **Description** | PFI is the difference between actual motor position PF and the value of position command PC at the beginning of the last motion. |
| | PFI is continuously updated by the system. |
| | PFI is set to zero (0) at system startup and the start of motion. PFI is undefined immediately after a mechanical home seeking operation completes (MGHP, MGHN). |
| | See also PF. |

| **Example** | Command | Description |
|---|---|---|
| | `>LIST AREAOUT4` | #List sequence AREAOUT4 |
| | | |
| | `(  1) DIS 100; VR=10` | #Set distance, velocity |
| | `(  2) MI` | #Start move incremental |
| | `(  3) SAS Motion started` | #Send message 1 |
| | `(  4) WHILE (PFI<30)` | #Wait for PFI to reach 30 |
| | `(  5) WEND` | |
| | `(  6) SAS Passed 30mm` | #Send message 2 |
| | `(  7) WHILE (PFI<60)` | #Wait for PFI to reach 60 |
| | `(  8) WEND` | |
| | `(  9) SAS Passed 60mm` | #Send message 3 |
| | `( 10) MEND` | #Wait for motion end |
| | `( 11) SAS Reached target` | #Send message 4 |
| | `( 12) END` | |
| | `>` | |
| | `>RUN AREAOUT4` | #Start sequence |
| | `>Motion started` | #Message 1 |
| | `>Passed 30mm` | #Message 2 |
| | `>Passed 60mm` | #Message 3 |
| | `>Reached target` | #Message 4 |
| | `>` | |

## PLSINV : Pulse Output Invert

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | PLSINV=n |
| **Range** | n = 0: Positive Logic<br>     1: Negative Logic |
| **Factory Setting** | 0 |
| **SAVEPRM & RESET** | Required to execute any changes made to the parameter value and to save 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 factory setting. |
| **Access** | READ and WRITE |
| **See Also** | PULSE |
| **Description** | Invert the pulse output logic. |

| **Example** | Command | Description |
|---|---|---|
| | `>PLSINV=0` | #Set pulse output invert |
| | `PLSINV=0(1)` | #Device response |
| | `>SAVEPRM` | #Save the parameter assignments |
| | `(EEPROM has been written 21 times)` | #Device response |
| | `Enter Y to proceed, other key to cancel. y` | #Device response |
| | `Saving Parameters........OK.` | |
| | `>RESET` | |
| | `Resetting system.` | |
| | `--------------------------------------` | |
| | `        CM10-*` | |
| | `        Controller Module` | |
| | `        Software Version: *.**` | |
| | `        Copyright 2010` | |
| | `        ORIENTAL MOTOR CO., LTD.` | |
| | `--------------------------------------` | |
| | `>PLSINV` | #Confirm the new value |
| | `PLSINV=1(1)` | |

# POS[x] : Position Array Data

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | POS[x]=n        (x is a number of position array data: x=1 to 100.) |
| **Range** | n = -MAXPOS to +MAXPOS |
| **Factory Setting** | 0 |
| **SAVEPOS** | The new value takes effect immediately. However, SAVEPOS 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 factory setting. |
| **Access** | READ and WRITE |
| **See Also** | MA, PC, TEACH, CLEARPOS, CLEARALL, SAVEPOS, SAVEALL |
| **Description** | The POS[x] variables provide an array of 100 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.: <br> MA POS[7] <br> …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 "8.4 Teaching Positions" on page 94 for more information. |
| | All POS[x] data can be cleared (initialized to zero) with the CLEARPOS command. |
| **Note** | POS[x] command cannot be used in the IF statement or the WHILE statement. |
| | Use after substituting POS[x] for general variable or user defined variable. |
| | Wrong) WHILE (PC!=POS[1]) |
| | Correct) A=POS[1] <br>              WHILE (PC!=A) |
| **Example** | Command                         Description |

```
>POS[1]                #Query the value established for POS [1]
 POS[1]=1.12
>MA POS[1]             #Move to POS[1]
>PC                    #When motion is finished, query the position command value
 PC=1.12              #Moved as expected, PC=POS[1]
>POS[2] 2.36           #Set POS [2] to 2.36 user units
 POS[2]=2.36
>MA POS[2]             #Move to POS[2]
>PC                    #When motion is finished, query the position command value
 PC=2.36              #Moved as expected, PC=POS[2]
>
```

## PRESET : Reset Home Position

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | PRESET |
| **Commands not Allowed** | MOVE |
| **See Also** | DOUTxxx (DOUTPRESET), PC, PF, OUTxxx (OUTHOMEP), SLACT, LIMP, LIMN, ABSPLSEN |
| **Description** | When the PRESET command is executed, PC (position command) is set to zero. This position will be the electrical home. PF (feedback position) and EC (encoder count) will follow maintaining PE (error), and as the result, they will be the value of the error. At the same time, when using a driver that has a preset (reset home position) function, the home position of the driver will also be reset (The PRESET output assigned to the driver connector on the **CM10**/**SCX10** will be turned ON for about 6 ms). When software position limit control is set to 1 (SLACT=1), LIMP and LIMN (software position limits) will be enabled.<br><br>・When the driver that has a PRESET input (the driver has a current position reading function)<br><br>　Set the home position to the driver by the PRESET command for using the current position reading function.<br><br>　The assignment of the PRESET output to the driver connector on the **CM10**/**SCX10** is required. When the PRESET output is not assigned, the home position of the driver will not reset, though PC will be set to 0 (zero) and this position will be the electrical home.<br><br>・When the driver does not have a PRESET input<br><br>　PC is set to zero, and this position will be the electrical home. |
| **Caution** | **With the ESMC controller, the HOME input and PRESET input are assigned to the same pin. The factory setting is the HOME input. Change the driver setting to the PRESET input from the HOME input before using the PRESET command.** |
| **Note** | If the PRESET command is executed when the parameter for PRESET (reset home position) of the **ESMC** controller is set to the value other than zero, the PC value will not match the driver's current position. In this case, they will be matched by executing the ABSREQPC (driver current position reading/updating internal position) command. However, if setting the electrical home position other than the mechanical home position is required as described above, it is recommended to set the offset value in the **CM10**/**SCX10** (use the OFFSET command) but not in the driver. |
| **Example** | Command　　　　　　　　　　　　Description |

```
>PRESET                   #Set the current position to the home position
>PC                       #Confirm the PC value
 PC=0 Rev                 #PC=0
>SIGHOMEP                 #Confirm whether the current position was set to the electrical home
 SIGHOMEP=1               #The current position was set to the electrical home
>ABSREQ                   #Reading the driver's current position, driver status and driver alarm
 PABS=0 Rev               #Current position
 Driver Status Code = 00  #Driver status code
 Driver ALARM Code = 00   #Driver alarm code
```
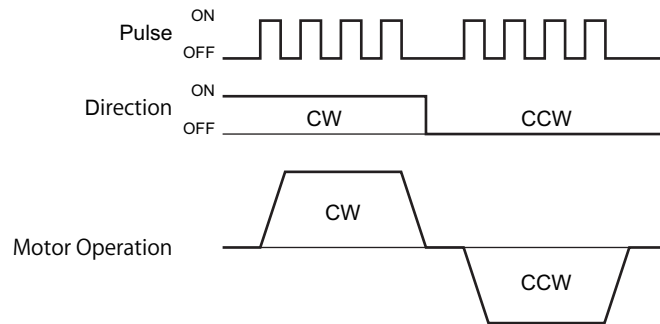
# PSTOP  : Panic Stop

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | PSTOP |
| **See Also** | <ESC>, MA, MCP, MCN, MGHP, MGHN, MI, EHOME, SSTOP, HSTOP, MSTOP, INxxx (INPSTOP), ALMACT, ABORT |
| **Description** | PSTOP stops the motor as quickly as possible (hard stop) and stop sequence, and then takes the alarm action determined by ALMACT, which may involve setting an alarm (alarm 68h), aborting sequences, and possibly disabling motor current. Also the ACL/DCL signal on the driver connector of the **CM10**/**SCX10** is momentarily output for stopping servo motors and $\alpha$*STEP* products immediately. <br><br> The PSTOP function may also be executed via the PSTOP input on the I/O connector and/or the CANopen remote I/O if assigned. When detailed explanation is required, see "6.4 Connecting the I/O Signals" on page 23 for I/O connector, and see "10.4 Controlling I/O Message (PDO)" on page 134 for CANopen remote I/O. |
| **Caution** | **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.** |
| **Example** | Command  Description |

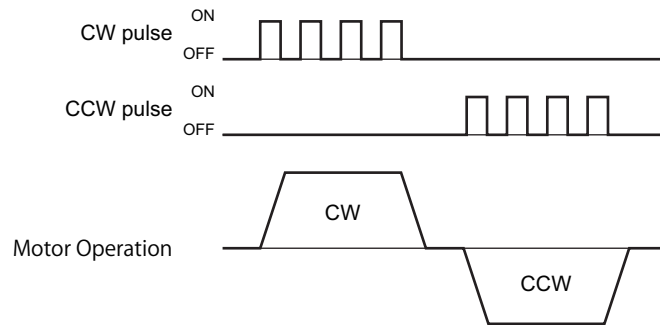| Command | Description |
|---|---|
| >VR 4 | #Set the velocity to 4 mm/second |
| VR=4 mm/sec | #Device response |
| >MCP | #Move continuously in the positive direction |
| >PSTOP | #Stop the motor as quickly as possible |
| > | |

## PULSE : Pulse Output Mode

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | PULSE=n |
| **Range** | n = 0: 2 Pulse Mode<br>    1: 1 Pulse Mode |
| **Factory Setting** | 1 |
| **SAVEPRM & RESET** | Required to execute any changes made to the parameter value and to save 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 factory setting. |
| **Access** | READ and WRITE |
| **See Also** | PLSINV, DIRINV |

**Description**

Set the pulse output mode. Check the pulse input mode of the driver, and set the driver and the **CM10**/**SCX10** to the same pulse input mode.

1 pulse mode



2 pulse mode



**Example**

| Command | Description |
|---|---|
| >PULSE=0 | #Set pulse output invert |
| PULSE=1(0) | #Device response |
| >SAVEPRM | #Save the parameter assignments |
| (EEPROM has been written 21 times) | #Device response |
| Enter Y to proceed, other key to cancel. y | |
| Saving Parameters.......OK. | |
| >RESET | |
| Resetting system. | |
| -------------------------------------- | |
| CM10-* | |
| Controller Module | |
| Software Version: *.** | |
| Copyright 2010 | |
| ORIENTAL MOTOR CO., LTD. | |
| -------------------------------------- | |
| >PULSE | #Confirm the new value |
| PULSE=0(0) | |

# REN  : Rename Sequence

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | REN target newname |
| **Range** | 'target' must be the name or number of an existing sequence.<br>'newname' must be a valid sequence name (consisting of letters, numbers or underscore, 10 characters maximum, must start with a letter except n, s, N, S, with no distinction of a capital letter or small letter.) |
| **Commands not Allowed** | RUN |
| **See Also** | COPY, DIR, EDIT, DEL, LOCK, UNLOCK |
| **Description** | REN renames an existing sequence.  The new name must be unique.<br><br>REN can also be used to name a sequence which was created by number only, and has no name.<br><br>'target' cannot be renamed if it is locked.  Change the name after releasing the lock of the sequence program using the UNLOCK command. |

| **Example** | Command | Description |
|---|---|---|

```
>DIR                                 #Check the names of all sequences


   ##   Name        TextSize  Locked
   ==   ==========  ========  ======
    0   PROG1             37
    1   MOVE1             24

  Total:   2
  Executable memory:     4 bytes used of  6144 bytes total,    0 percent.
  Storage memory:       18 bytes used of 21775 bytes total,    0 percent.
>REN PROG1 PROG2                     #Rename PROG1 to the new name of PROG2
>DIR


   ##   Name        TextSize  Locked
   ==   ==========  ========  ======
    0   PROG2             37
    1   MOVE1             24

  Total:   2
  Executable memory:     4 bytes used of  6144 bytes total,    0 percent.
  Storage memory:       18 bytes used of 21775 bytes total,    0 percent.
>REN PROG2 MOVE1                     #Can't rename if new name exists already

Error: Sequence already exists.
>
```

## REPORT  : Display System Status

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | REPORT |
| **See Also** | ALM, IO, OUTTEST |
| **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 inputs and outputs, the values of important parameters, the value of position command PC, and the alarm and warning history. |
| | The report is displayed on two pages, and pressing the space key to go to the next page. |
| **Example** | Command                Description |

```
>REPORT              #Get a system status summary: sample result follows
/ I/O REPORT /---(NO:Normally Open, NC:Normally Closed)-----------
  IN1(NO) = 0    IN2(NO) = 0    IN3(NO) = 0    IN4(NO) = 0
  IN5(NO) = 0    IN6(NO) = 0    IN7(NO) = 0    IN8(NO) = 0
  IN9(NO) = 0
  OUT1(NO) = 0    OUT2(NO) = 0    OUT3(NO) = 0    OUT4(NO) = 0

/ REMOTE I/O REPORT /----------------------------------------------
  IN1 = 0    IN2 = 0    IN3 = 0    IN4 = 0
  IN5 = 0    IN6 = 0    IN7 = 0    IN8 = 0
  ABORT = 0    START = 0   MCP = 0    MCN = 0
  MGHN = 0    CON = 0    FREE = 0
  OUT1 = 0    OUT2 = 0    OUT3 = 0    OUT4 = 0
  OUT5 = 0    OUT6 = 0
  END = 0    MOVE = 0    HOMEP = 0    LC = 0    READY = 0

/ DRIVER I/O REPORT /----------------------------------------------
  ALM(NC) = 1    IN2(NO) = 0    END(NO) = 0    READY(NO) = 0
  LC(NO) = 0    TIMS(NO) = 0    TIMD/EXTZ(TIMDLV=1, EXTZLV=0) = 0
  CON(NO) = 0    ACL/DCL(NO) = 0    REQ(NO) = 0    TL(NO) = 0
  M0(NO) = 0    M1(NO) = 0    PRESET(NO) = 0    FREE(NO) = 0

  Enter [SPACE] to continue, other key to quit.

 /PARAMETER REPORT /-----------------------------------------------
  UU = Rev
  STRSW = 0    DPR = 1    MR = 1000    GA = 1    GB = 1
  ER = 1000    DIRINV = 0
  VS = 0.1    VR = 1    TA = 0.5    TD = 0.5    DIS = 0
  LIMP = 0    LIMN = 0    SLACT = 0
  STARTACT = 0    MSTOPACT = 0    SENSORACT = 2    OTACT = 0
  ALMACT = 2    ALMMSG = 0    HOMETYP = 0    HOMEDCL = 1
  INCABS0 = 1    VR0 = 1    DIS0 = 0    LINK0 = 0
  INCABS1 = 1    VR1 = 1    DIS1 = 0    LINK1 = 0
  INCABS2 = 1    VR2 = 1    DIS2 = 0    LINK2 = 0
  INCABS3 = 1    VR3 = 1    DIS3 = 0
  DALARM = 1    DREADY = 1    STRDSC = 0    TIM = 1    ENC = 1
  DEND = 1    ENDACT = 0    MBFREEACT = 0
  PULSE = 1    PLSINV = 0    CANID = 1    CANBAUD = 1

/ POSITION REPORT /------------------------------------------------
  PC = 0    PF = 0    PE = 0    EC = 0    PABS = 0

/ ALARM HISTORY /-------------------------------------------------
  ALARM = 6E ,   RECORD : 6E 6E 00 00 00 00 00 00 00 00
  ALM_DRIVER_ALARM , 15.123 [sec] past.
  Driver Status Code = 00    Driver ALARM Code = 00
```

## RESET  : Reset Device

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | RESET |
| **See Also** | STRSW, VERBOSE, SAVEPRM, SAVEPOS, SAVEALL |
| **Description** | RESET resets the device. |

**Description** (continued)

RESET resets the device.

Performing a RESET operation is similar to cycling power, but may respond quicker.

Several events occur when the device is reset:

1) Motor current is disabled, and the Magnetic Brake Control (MBFREE) output, if configured, is set to its open, non-conducting state. 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.

2) The system transmits a message: "Resetting system."

3) All outputs are set to an open (non-conducting) state.

4) The parameters and position array data saved in EEPROM are established.  Any parameter or position array data that was not saved is lost. (Use SAVEPRM to save parameter data, SAVEPOS to save position array data, or SAVEALL to save both, if desired, before issuing a RESET command.)

5) Alarm conditions are checked, and the alarm code is updated accordingly.

6) If motor current is permitted (depending on alarm state, if used, and STRSW setting, CON on the remote I/O and CON input on the I/O connector), current is enabled.

7) Outputs (other than MBFREE ) are set to appropriate states.

8) The immediate mode command prompt is transmitted (>).  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.

9) If current is enabled, and the MBFREE output is configured, the MBFREE output is set to its closed.

10) Inputs are read and appropriate actions taken.

11) If no alarm is set, no sequences are running, and a sequence named CONFIG exists, the CONFIG sequence will begin running automatically.

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.

| **Caution** | **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.** |
|---|---|
| **Note** | When the device is reset, any parameter or position array data that was not saved is lost. Use SAVEPRM to save parameter data, SAVEPOS to save position array data, or SAVEALL to save both, if desired, before issuing a RESET command. |

**Example**

| Command | Description |
|---|---|
| `>ALMACT 1`<br>` ALMACT=2(1)`<br>`>ALMMSG 2`<br>` ALMMSG=2 [Alarm+Warning]`<br>`>SAVEPRM`<br>` (EEPROM has been written 95 times)`<br>` Enter Y to proceed, other key to cancel. Y`<br>` Saving Parameters........OK.`<br>`>RESET`<br>` Resetting system.`<br>`-----------------------------------------`<br>`        CM10-*`<br>`        Controller Module`<br>`        Software Version: *.**`<br>`        Copyright 2010`<br>`        ORIENTAL MOTOR CO., LTD.`<br>`-----------------------------------------`<br>`>ALMACT; ALMMSG`<br>` ALMACT=1(1)`<br>` ALMMSG=2 [Alarm+Warning]`<br>`>` | #New value of alarm action: alarm, keep current on. Active (future) values<br>#New value for alarm messaging: transmit message for alarm and warning<br>#Save all parameters<br><br><br><br>#Reset the system to make new value of ALMACT active (ALMMSG change was effective immediately). Reset messages follow<br><br><br><br><br><br><br><br>#Confirm new values of ALMACT and ALMMSG |

# RET : Sequence Return

| | |
|---|---|
| **Execution Mode** | Sequence |
| **Syntax** | RET |
| **See Also** | ABORT, CALL, END |
| **Description** | RET terminates processing of the sequence that is currently running as a subroutine. |
| | If the sequence that contains "RET" was CALL'ed from another sequence, the original sequence will resume at the statement following the CALL statement.  If the sequence that contains "RET" was started with the START input or the RUN command, RET terminates all sequence execution. |
| | To unconditionally terminate all sequence processing, use ABORT. |
| | 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). |

| **Example** | Command | Description |
|---|---|---|
| | ```>LIST MAIN``` | #List sequence MAIN |
| | ```(  1) VR 10``` | #Set running velocity to 10 |
| | ```(  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 5``` | #……Change speed to 5 |
| | ```(  4)     MEND``` | #……Wait for motion to end |
| | ```(  5)     RET``` | #      and return to caller |
| | ```(  6)   ENDIF``` | #…End of IF block |
| | ```(  7) WEND``` | #End of WHILE block |
| | ```>``` | |

# RIN : Remote General Input Status

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | RIN |
| **Range** | 0 to 255 |
| | /: real time monitor  (immediate mode only) |
| **Access** | READ |
| **See Also** | RIO, RINx, ROUT, ROUTx, RINxxx, INITRIO, REPORT |
| **Description** | The RIN command displays the current status of all the general purpose remote inputs, as one integer number. |

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

| RINx | Contribution to RIN If Active |
|---|---|
| RIN8 | 128 |
| RIN7 | 64 |
| RIN6 | 32 |
| RIN5 | 16 |
| RIN4 | 8 |
| RIN3 | 4 |
| RIN2 | 2 |
| RIN1 | 1 |

For example, if the remote general input 2 (2), remote general input 3 (4) and remote general input 4 (8) are active, while all other signals are not active, "RIN=14" is set (2+4+8=14).

When inputting RIN, "RIN=14" is replied.

∗ To check the status of a single remote general input, use the RINx command.

∗ If remote input is assigned to a system input signal (HOME, LSP, LSN, etc) the RIN command will always show that input as OFF or 0. Inputs which have been assigned to system input signals do not affect RIN.

| **Example** | Command | Description |
|---|---|---|

```
>RIN                      #Query the status of the remote general inputs
 RIN=32                   #Device response indicating remote input 6 is ON
>
>LIST 8                   #List sequence 8

(  1) SAS PRESS START     #Notify user to press start
(  2) IF (RIN=18)         #If remote inputs 2 and 5 are ON then,
(  3)   MGHN              #Go home in the negative direction
(  4) ELSE                #If the value of RIN does not equal 18, then
(  5)   WHILE (RIN=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
>
```

## RINx : Individual Remote General Input Status

| | |
|---|---|
| **Execution Mode** | Immediate and Sequence |
| **Syntax** | RINx    (RINx is a signal name: RIN1 to RIN8) |
| **Range** | n =  0: Not Active<br>    1: Active<br><br>/: real time monitor  (immediate mode only) |
| **Access** | READ |
| **See Also** | RIN, RIO, ROUT, ROUTSG, ROUTx, REPORT |
| **Description** | RINx returns the state of the remote general purpose input "x."<br><br>If the input has been assigned to a system input signal, then it is no longer "general purpose."  RINx for these inputs will always return 0 (not active). |
| **Example** | Command                    Description<br><br>>RIN1                       #Query the status of the individual remote general input<br> RIN1=0 |

# RINxxx  : Remote System Signal Input Assignment

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | RINxxx=n |
| | "xxx" represents the signal name to be assigned, and "n" represents the assigned terminal number ("that" becomes the RINn remote general input when unassigned). |
| **Range** | n =  0 to 8 |
| **Factory Setting** | 0 (unassigned) |
| **SAVEPRM & RESET** | Required to execute any changes made to the parameter value and to save 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 factory setting. |
| **Access** | READ and WRITE |
| **See Also** | SIGxxx, RIO, INITRIO, CLEARALL, RINx, REPORT and "See Also" column in the chart below. |
| **Description** | Assign the system input signal to the RINn on the remote I/O.  The remote system signal input assignment is released by "RINxxx=0" and it becomes the remote general input RINn.  When executing the INITRIO command, the parameter restores to the factory setting. |

| Command | Signal | Description | See Also |
|---|---|---|---|
| RINALMCLR | ALMCLR | Alarm Clear | ALMCLR |
| RINCONT | CONT | Continue Motion | PAUSE, PAUSECLR |
| RINHOME | HOME | Home Sensor | HOMETYP, MGHP, MGHN |
| RINLSP/RINLSN | LSP/LSN | Limit Switch Positive /Limit Switch Negative | ALM, ALMACT, ALMCLR |
| RINMGHP | MGHP | Move Go Home Positive | MGHP, HOMETYP |
| RINMSTOP | MSTOP | Motor Stop | MSTOP, MSTOPACT |
| RINPAUSE | PAUSE | Pause Motion | PAUSE, CONT, PAUSECLR, RINPAUSECL, ROUTPSTS |
| RINPAUSECL | PAUSECL | Pause Clear | PAUSECLR, CONT, PAUSE, RINPAUSE, SIGPSTS, ROUTPSTS |
| RINPECLR | PECLR | Position Error Clear | PECLR, EC, PC, PE, PF |
| RINPSTOP | PSTOP | Panic Stop | PSTOP, ABORT, <ESC>, ALMACT, HSTOP, MSTOPLV, SSTOP, PAUSE |
| RINSENSOR | SENSOR | Sensor | SENSORACT, MGHP, MGHN, SCHGPOS, SCHGVR |
| RINTL | TL | Torque Limiting /Push-motion Operation /Current Cutback Release | TL, DOUTTL |

| | |
|---|---|
| **Memo** | The xxxLV does not invert the logic level of RINxxx. |

| **Example** | Command | Description |
|---|---|---|
| | >RINALMCLR 3 | #Assign remote input number 3 as the ALMCLR signal |
| | RINALMCLR=0(3) | |
| | >SAVEPRM | #Save the parameter assignments |
| | (EEPROM has been written 2 times) | |
| | Enter Y to proceed, other key to cancel. Y | |
| | Saving Parameters........OK. | |
| | >RESET | #Establish the saved parameter values |
| | Resetting system. | |
| | ------------------------------------------ | |
| | CM10-* | |
| | Controller Module | |
| | Software Version: *.** | |
| | Copyright 2010 | |
| | ORIENTAL MOTOR CO., LTD. | |
| | ------------------------------------------ | |
| | >RINALMCLR | #Confirm new value |
| | RINALMCLR=3(3) | |
| | > | |

## RIO : Remote I/O Status

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | RIO |
| **Range** | /: real time monitor  (immediate mode only) |
| **See Also** | RINxxx, ROUTxxx, RIN, ROUT, RINx, ROUTx, DIO, IO, REPORT |
| **Description** | RIO displays the current status of general purpose remote inputs and remote outputs and system remote input signals and system remote 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, general purpose input 1 is active, so IN=1, and sequence 1 would start if the alarm condition were cleared and START became active. |

| **Example** | Command | Description |
|---|---|---|
| | `>RIO` | #Display the RIO status |
| | ` Inputs  (1-8) = IN1 SENSOR IN3 IN4 IN5 IN6 IN7 IN8` | #Device response |
| | ` Outputs (1-6) = MBFREE OUT2 OUT3 OUT4 OUT5 OUT6` | |
| | | |
| | ` --Inputs--           --Outputs--` | |
| | ` 1 2 3 4 5 6 7 8 -(SEQ#)- 1 2 3 4 5 6` | |
| | ` 1 1 0 0 0 0 0 0 -(  1 )- 0 0 0 0 0 0` | |
| | ` >` | |

## ROUT : Remote General Output Control

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | ROUT=n<br><br>∗ "=n" is required only when controlling |
| **Range** | n = 0 to 63 (integer values)<br>/: real time monitor  (immediate mode only) |
| **Factory Setting** | 0 |
| **Access** | READ and WRITE<br>READ only in CANopen |
| **See Also** | RINSG, ROUTSG, RIN, ROUTxxx, RIO |

**Description**

ROUT displays or sets the value of all the general purpose remote outputs, as one integer number.

The general purpose remote outputs contribute to the value of ROUT as follows:

| ROUTx | Contribution to ROUT If Active |
|---|---|
| ROUT6 | 32 |
| ROUT5 | 16 |
| ROUT4 | 8 |
| ROUT3 | 4 |
| ROUT2 | 2 |
| ROUT1 | 1 |

For example, if the remote general output 2 (2) and remote general output 4 (8) are active, while all other signals are not active, "ROUT=10" is set (2+4=10).

When inputting ROUT, "ROUT=10" is replied.


∗ To check or control the status of a single remote general output, use the ROUTx command.

∗ The ROUT value always indicates the internal status of the all remote general output signals ROUT1 to ROUT8.  For the output signal assigned the system output signal (RUN, MBFREE etc.), the ROUT command recognizes the signal as inactive or zero (except when the signal has become active by the ROUT, ROUTx command).

∗ All general purpose outputs are in the inactive (OFF) state immediately following system startup.

**Example**

| Command | Description |
|---|---|

```
>RIO                                              #Display the RIO status
 Inputs  (1-8) = IN1 IN2 IN3 IN4 IN5 IN6 IN7 IN8  #Device response
 Outputs (1-6) = OUT1 OUT2 OUT3 OUT4 OUT5 OUT6

 --Inputs--             --Outputs--
 1 2 3 4 5 6 7 8 -(SEQ#)- 1 2 3 4 5 6
 0 0 0 0 0 0 0 0 -(  0 )- 0 0 0 0 0 0
>ROUT                                             #Check ROUT status
 ROUT=0
>ROUT=15                                          #Set ROUT to 15
 ROUT=15
>RIO                                              #Display the RIO status
 Inputs  (1-8) = IN1 IN2 IN3 IN4 IN5 IN6 IN7 IN8  #Device response
 Outputs (1-6) = OUT1 OUT2 OUT3 OUT4 OUT5 OUT6

 --Inputs--             --Outputs--
 1 2 3 4 5 6 7 8 -(SEQ#)- 1 2 3 4 5 6
 0 0 0 0 0 0 0 0 -(  0 )- 1 1 1 1 0 0
```

## ROUTx  : Individual Remote General Output Control

| | |
|---|---|
| **Execution Mode** | Immediate and Sequence |
| **Syntax** | ROUTx=n        (ROUTx is a signal name: ROUT1 to ROUT6)<br><br>∗ "=n" is required only when controlling. |
| **Range** | n =  0: Not Active<br>　　1: Active<br>/: real time monitor  (immediate mode only) |
| **Factory Setting** | 0 |
| **See Also** | INITRIO, ROUT, ROUTSG, RIO, RIN |
| **Description** | ROUTx displays or controls the state of remote general purpose output 'x'. If the remote output has been assigned to a system output signal such as RUN and MBFREE, then it is no longer "general purpose." ROUTx for these outputs has no affect on the output levels. |
| **Memo** | Even the output signal assigned the system output signal can become active (1) as the remote general output status using the ROUTx command.  For example, even when the ROUT1 is assigned to PSTS, ROUT1=1 can be commanded.  However, the PSTS signal will not become active.  Note that the ROUT (remote general output control) value becomes 1 at this time. |
| **Example** | Command |

Command | Description

```
>LIST 8

(  1) VS=0;VR=1;TA=0.001;TD=0.001
(  2) DIS=10;PC=0
(  3) ROUT=0
(  4) MI
(  5) LOOP                          #Start infinite loop
(  6)   IF (PC>=2)                  #If PC is equal to or greater than 2
(  7)     ROUT1=1                   #ROUT1 is active
(  8)   ENDIF                       #End of IF block
(  9)   IF (PC>=4)                  #If PC is equal to or greater than 4
( 10)     ROUT2=1                   #ROUT2 is active
( 11)   ENDIF                       #End of IF block
( 12)   IF (PC>=6)                  #If PC is equal to or greater than 6
( 13)     ROUT3=1                   #ROUT3 is active
( 14)   ENDIF                       #End of IF block
( 15)   IF (PC>=8)                  #If PC is equal to or greater than 8
( 16)     ROUT4=1                   #ROUT4 is active
( 17)   ENDIF                       #End of IF block
( 18)   IF (PC==DIS)                #If PC is equal to DIS
( 19)     ROUT5=1                   #ROUT5 is active
( 20)     BREAKL                    # Exit the loop and execute the line after the ENDL
( 21)   ENDIF                       command
( 22) ENDL                         # End of IF block
( 23) MEND                         #Terminate the LOOP
                                    #Wait for stop to complete
```

## ROUTxxx : Remote System Signal Output Assignment

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | ROUTxxx=n |
| **Range** | n = 0 to 6 |
| **Factory Setting** | 0 (unassigned) |
| **SAVEPRM & RESET** | Required to execute any changes made to the parameter value and to save 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 factory setting. |
| **Access** | READ and WRITE |
| **See Also** | OUTxxx (except: ABSDATA), SIGxxx, RIO, INITRIO, CLEARALL, ROUT, ROUTx, REPORT and "See Also" column in the chart below. |
| **Description** | Assign the system output signal to the ROUTn of the remote I/O.  The remote system signal output assignment is released by "ROUTxxx=0" and it becomes the remote general output ROUTn.<br><br>When executing the INITRIO command, the parameter restores to the factory setting. |

| Command | Signal | Description | See Also |
|---|---|---|---|
| ROUTABSDATA | ABSDATA | Driver Current Position Data Ready | ABSREQ, ABSREQPC, PABS, ABSSTS |
| ROUTMBFREE | MBFREE | Magnetic Brake Free | CURRENT, FREE |
| ROUTPSTS | PSTS | Pause Status | PAUSE, PAUSECLR, CONT |
| ROUTRUN | RUN | Sequence Running | RUN |

| **Example** | Command | Description |
|---|---|---|
| | ```>ROUTRUN 3```<br>```ROUTRUN=0(3)```<br>```>SAVEPRM```<br>``` (EEPROM has been written 2 times)```<br>``` Enter Y to proceed, other key to cancel. Y```<br>``` Saving Parameters........OK.```<br>```>RESET```<br>``` Resetting system.```<br>```------------------------------------------```<br>```           CM10-*```<br>```           Controller Module```<br>```           Software Version: *.**```<br>```           Copyright 2010```<br>```           ORIENTAL MOTOR CO., LTD.```<br>```------------------------------------------```<br>```>ROUTRUN```<br>``` ROUTRUN=3(3)```<br>```>``` | #Assign remote input number 3 as the RUN signal<br>#Save the parameter assignments<br><br><br><br>#Establish the saved parameter values<br><br><br><br><br><br><br><br><br>#Confirm new value |

## RUN  : Run Sequence

| | |
|---|---|
| **Execution Mode** | Immediate and CANopen |
| **Syntax** | RUN target |
| **Range** | 'target' must be the name or number of an existing sequence. |
| **Commands not Allowed** | RUN |
| **See Also** | EDIT, DIR, ABORT, <ESC>, INxxx (INSTART) |
| **Description** | RUN starts execution of a sequence.<br><br>Sequences can also be started with the dedicated START input if assigned on the I/O connector and/or the CANopen remote I/O. When detailed explanation is required, see "6.4 Connecting the I/O signals" on page 23 for I/O connector, and see "10.4 Controlling I/O Message (PDO)" on page 134 for CANopen remote I/O. Sequences cannot be started if the system has an active alarm condition.<br><br>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.<br><br>RUN cannot be used inside sequences.  To execute one sequence from within another, use the CALL command. |
| **Memo** | Sequences cannot be edited while a sequence is executing.  The system prevents the editor from starting. |
| **Example** | Command                Description |

```
>LIST DIS200          #List sequence DIS200

(  1) DIS=200          #Sequence listing
(  2) VS=0;VR=10
(  3) TA=0.5;TD=1
(  4) MI
(  5) MEND
>RUN DIS200            #Run sequence DIS200
>SIGRUN               #Commands can still be executed, while…
 SIGRUN=1             #…sequences execute (SIGRUN=1)
>
```

# S_xxx : User-defined String Variables

| | |
|---|---|
| **Execution Mode** | Immediate and Sequence |
| **Syntax** | S_xxx=text     xxx = 1 to 10 Alphanumeric Characters (S_xxx can be the name of any existing user-defined string variable.) |
| **Range** | text = 20 Characters Maximum |
| **Factory Setting** | Text is Empty. |
| **SAVEPRM** | 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 factory setting. |
| **Access** | READ and WRITE |
| **See Also** | CLEARVAR, CREATEVAR, DELETEVAR, LISTVAR, SAS, SACS, VIEW, N_xxx |
| **Description** | General purpose, user-defined string variables. |
| | A user-defined string variable must be created with CREATEVAR before it can be used.  After it has been created, it can be used to store or display text. |
| | Within sequences, the VIEW command can be used to display string contents without entering repeated carriage returns, linefeeds or by reprompting. |

**Example**

| Command | Description |
|---|---|
| `>CREATEVAR S_QUAD` | |
| ` NEW variable S_QUAD is added.` | |
| ` S_QUAD=` | |
| `>LIST MAIN` | #List contents of sequence MAIN |
| | |
| `(  1) LOOP` | #Start infinite loop |
| `(  2)   WAIT 0.05` | #Wait 50 milliseconds |
| `(  3)   MI` | #Start incremental motion |
| `(  4)   MEND` | #Wait for motion to end |
| `(  5)   CALL QUADRANT` | #Call sequence QUADRANT as subroutine |
| `(  6)   S_QUAD` | #Show contents of user-defined string variable S_QUAD |
| `(  7) ENDL` | #End of LOOP block |
| `>LIST QUADRANT` | #List contents of sequence QUADRANT |
| | |
| `(  1) R=PC%360` | #R = position command, modulo 360 |
| `(  2) IF (R>=270)` | |
| `(  3)   S_QUAD=Fourth Quadrant` | #Assign text to S_QUAD, depending on quadrant |
| `(  4)   RET` | |
| `(  5) ENDIF` | |
| `(  6) IF (R>=180)` | |
| `(  7)   S_QUAD=Third Quadrant` | |
| `(  8)   RET` | |
| `(  9) ENDIF` | |
| `( 10) IF (R>= 90)` | |
| `( 11)   S_QUAD=Second Quadrant` | |
| `( 12)   RET` | |
| `( 13) ENDIF` | |
| `( 14) S_QUAD=First Quadrant` | |
| `>DIS=75; RUN MAIN` | #Set incremental distance to 75, run sequence MAIN |
| ` DIS=75 Rev` | |
| `>Third Quadrant` | #Sequence MAIN transmits contents of S_QUAD after each |
| `>Fourth Quadrant` | motion |
| `>First Quadrant` | |
| `>First Quadrant` | |
| `>Second Quadrant` | |
| `>Third Quadrant` | #…etc. |
| `>` | |

## SACS  : Send ASCII Control String

| | |
|---|---|
| **Execution Mode** | Sequence |
| **Syntax** | SACS string |
| **Range** | string : a series of ASCII characters or control codes, maximum 70 characters. |
| **See Also** | KB, KBQ, SAS, VIEW |
| **Description** | 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. (A space between SACS command and ASCII string is necessary.)<br><br>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.<br><br>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.  (SAS, send ASCII string, automatically appends a carriage return + linefeed pair, and may be easier to use in some applications.)<br><br>There are several exceptions and extensions to the normal ASCII interpretation of control codes:<br><br>- ^@: a caret followed by @ (ASCII value NULL, binary 0) is not supported.<br><br>- ^ : a caret followed by a space, transmits one space character (this permits leading or trailing space characters in the output)<br><br>- ^^: two carets transmit a single caret (^).<br><br>- ^^^: three carets transmits an ASCII CTRL-^, 1Eh.<br><br>- ^: only a caret followed by nothing does nothing.<br><br>Other commands and comments cannot follow an SACS command on the same line: they will be considered part of the ASCII string.<br><br>For other control codes and their usual interpretations, see "Appendix A." |
| **Memo** | In a daisy chain configuration (ID other than *), all output from sequence commands is suppressed unless the device has been previously addressed (via TALK or @).<br><br>"@" cannot be used in an ASCII string. |

| **Example** | Command | Description |
|---|---|---|
| | `>LIST REFRESH` | #List Sequence "REFRESH" |
| | `(  1) # VT-100 EMULATION`<br>`(  2) #  1) CLEAR DISPLAY`<br>`(  3) #  2) HOME CURSOR`<br>`(  4) #  3) TRANSMIT PREFERRED PROMPT`<br>`(  5) SACS ^[[2J^[[H-->`<br>`>RUN REFRESH`<br>`>` | #Comments, in sequence<br>#Transmitted control codes below cause VT-100 displays to clear screen and "home" the cursor. Then: transmit a custom prompt |

## SAS  : Send ASCII String

| | |
|---|---|
| **Execution Mode** | Sequence |
| **Syntax** | SAS string |
| **Range** | string : a series of ASCII characters, maximum 70 characters. |
| **See Also** | SACS, VIEW, KB, KBQ |
| **Description** | 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. (A space between SACS command and ASCII string is necessary.) |
| | Other commands and comments cannot follow an SAS command on the same line: they will be considered part of the ASCII string. |
| **Memo** | In a daisy chain configuration (ID other than *), all output from sequence commands is suppressed unless the device has been previously addressed (via TALK or @). The KB and KBQ commands will not receive input unless the device has been previously addressed. |
| | "@" cannot be used in an ASCII string. |
| **Example** | Command | Description |

| 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:` | |
| `>1.125` | |
| `>Distance: 0` | |

## SAVEALL  : Save All Data

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | SAVEALL |
| **Commands not Allowed** | MOVE, RUN |
| **See Also** | CLEARALL, RESET, SAVEPOS, SAVEPRM, INITPRM, CLEARPOS |
| **Description** | SAVEALL saves the all current parameter settings and position array data to nonvolatile memory (EEPROM).  SAVEALL is the equivalent of SAVEPRM followed by SAVEPOS. |
| | SAVEALL 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. |
| | SAVEALL requires confirmation. A 'y' (not case sensitive) must be sent before the operation proceeds: any other character aborts the operation. |
| | Many parameters do not become effective until they are saved and the system is restarted.  These parameters will have a SAVEPRM & RESET entry in this chapter, and the system displays their values in active(future) form: the active value is displayed first, and the (future) value, displayed second, will only become effective if the parameter is saved and the system restarted. |

```
>DALARM=0
 DALARM=1 (0) [Enable(Disable)]
             |   |
             |   |----Active value
             |--------Future value
```

| | |
|---|---|
| **Caution** | **The SAVEALL command writes to EEPROM. The EEPROM has a nominal expected lifetime of 100,000 write cycles.  The SAVEALL command should not be used automatically (i.e. by a host controller) if it could possibly execute at high frequency.** |

| **Example** | Command | Description |
|---|---|---|
| | `>SAVEALL`<br>`(EEPROM has been written 100 times)`<br>`Enter Y to proceed, other key to cancel. y`<br>`Saving Parameters........OK.`<br>`Saving POS[ ] Data.......OK.`<br>`>` | #Save all parameters, variables, and position array data<br>#System requires confirmation<br><br>#Note RESET required before some new settings take effect. |

## SAVEPOS  : Save Position Array Data

| Execution Mode | Immediate |
|---|---|
| Syntax | SAVEPOS |
| Commands not Allowed | MOVE, RUN |
| See Also | CLEARALL, CLEARPOS, RESET, SAVEALL, SAVEPRM, INITPRM, TEACH, POS[x] |
| Description | SAVEPOS saves the all position array data (POS[x]) to nonvolatile memory (EEPROM). |
| | SAVEPOS affects the values of the position array data (following a power cycle or RESET command).  The saved values become the initial values after a restart.  The position array data can be modified and used freely in an application without saving, but the last saved values will become active when the system is restarted. |
| | SAVEPOS requires confirmation. A 'y' (not case sensitive) must be sent before the operation proceeds: any other character aborts the operation. |
| Caution | **The SAVEPOS command writes to EEPROM. The EEPROM has a nominal expected lifetime of 100,000 write cycles.  The SAVEPOS command should not be used automatically (i.e. by a host controller) if it could possibly execute at high frequency.** |

| Example | Command | Description |
|---|---|---|
| | ```
      *** Teach mode ***

(V)     : Move Cont. Neg.     (M)     : Move Cont. Pos.
(B)     : Jog Negative        (N)     : Jog Positive
(Q)     : Current ON/OFF      (F)     : FREE ON/OFF
(S)     : Save all data to EEPROM
(K)     : Change Key Interval (50-500[msec])
(D)     : Change Jog Distance (0.001-500000 [Rev])
          Minimum Movable Distance : 0.001 [Rev]
<Space> : Immediate Stop
<Enter> : Data entry mode (Input POS number, then <Enter>)
<ESC>   : Exit teach mode

PC=        0.000    Save to POS[23]     Data set OK.
End teach mode

>SAVEPOS
 (EEPROM has been written 104 times)
 Enter Y to proceed, other key to cancel. Y
 Saving POS[ ] Data.......OK.
>
``` | #Enter TEACH mode to "learn" target positions<br><br>#<Enter>, this PC saved to POS[23]<br>#<ESC>: exit TEACH<br>#SAVEPOS, to be sure all POS[x] data restore after restart |

## SAVEPRM : Save Parameters

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | SAVEPRM |
| **Commands not Allowed** | MOVE, RUN |
| **See Also** | CLEARALL, SAVEALL, SAVEPOS, RESET, INITPRM |
| **Description** | SAVEPRM saves the all current parameter settings to nonvolatile memory (EEPROM). |
| | 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 SAVEALL entry in this chapter. |
| | SAVEPRM requires confirmation. A 'y' (not case sensitive) must be sent before the operation proceeds: any other character aborts the operation. |
| | Many parameters do not become effective until they are saved and the system is restarted. These parameters will have a SAVEPRM & RESET entry in this chapter, and the system displays their values in active(future) form: the active value is displayed first, and the (future) value, displayed second, will only become effective if the parameter is saved and the system restarted. |

```
>DALARM=0
 DALARM=1(0) [Enable(Disable)]
          |  |
          |  |_____Active value
          |_____Future value
```

| | |
|---|---|
| **Caution** | **The SAVEPRM command writes to EEPROM. The EEPROM has a nominal expected lifetime of 100,000 write cycles. The SAVEALL command should not be used automatically (i.e. by a host controller) if it could possibly execute at high frequency.** |

| **Example** | Command | Description |
|---|---|---|

```
>VR                                          #Check value of running velocity
 VR=1 Rev/sec                                #Factory setting, 1 RPS
>VR 10                                       #Set running velocity 10
 VR=10 Rev/sec
>RESET                                       #Reset the system
 Resetting system.
-------------------------------------------
          CM10-*
          Controller Module
          Software Version: *.**
          Copyright 2010
          ORIENTAL MOTOR CO., LTD.
-------------------------------------------
>VR                                          #Check value of running velocity
 VR=1 Rev/sec                                #Didn't stick! Still default 1 RPS
>VR 10                                       #Set back to 10 RPS
 VR=10 Rev/sec
>SAVEPRM                                     #SAVEPRM: this will become new
 (EEPROM has been written 14 times)          startup value
 Enter Y to proceed, other key to cancel. Y  #Confirm SAVEPRM
 Saving Parameters........OK.
>RESET                                       #Reset the system again
 Resetting system.
-------------------------------------------
          CM10-*
          Controller Module
          Software Version: *.**
          Copyright 2010
          ORIENTAL MOTOR CO., LTD.
-------------------------------------------
>VR                                          #Check value again
 VR=10 Rev/sec                               #OK. We have new startup value
>
```

## SCHGPOS : Distance from SENSOR Input to the Stop Position

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | SCHGPOS=n |
| **Range** | n = 0 to MAXPOS (user units) |
| **Factory Setting** | 0 |
| **SAVEPRM** | 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 factory setting. |
| **Access** | READ and WRITE |
| **See Also** | INxxx (INSENSOR), RINxxx (RINSENSOR), MCP, MCN, SCHGVR, SENSORACT, xxxLV (SENSORLV), MAXPOS |
| **Description** | SCHGPOS is the distance used for SENSOR offset operation. |
| | SENSOR offset operation allows the system to stop a continuous motion (MCP, MCN) 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. |
| | SENSORACT must be 2 (offset operation), and the system input signal SENSOR must be assigned to an input before offset operations can be used. |

| **Example** | Command | Description |
|---|---|---|
| | >LIST REGISTER | #List sequence "REGISTER" |
| | ( 1) SCHGPOS 10; SCHGVR 5 | #Set sensor offset distance to 10, and speed to 5 |
| | ( 2) VR 10; MCP | #Set running velocity to 10, move continuous (positive) |
| | ( 3) WHILE (SIGMOVE=1) | #While moving |
| | ( 4)   IF (PCI>50) | #If we have moved more than 50… |
| | ( 5)     ALMSET | #Too far. Force an alarm |
| | ( 6)     RET | #Return: not required. Alarm will abort sequences |
| | ( 7)   ENDIF | #End IF block |
| | ( 8) WEND | #End WHILE block |
| | > | |

## SCHGVR  : Velocity after SENSOR Input

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | SCHGVR=n |
| **Range** | n =  0.001 to MAXVEL (user units/second) |
| **Factory Setting** | 1 |
| **SAVEPRM** | 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 factory setting. |
| **Access** | READ and WRITE |
| **See Also** | INxxx (INSENSOR), RINxxx (RINSENSOR), MCP, MCN, SCHGPOS, SENSORACT, xxxLV (SENSORLV), MAXVEL |
| **Description** | SCHGVR is the velocity used for SENSOR offset operation. |
| | SENSOR offset operation allows the system to stop a continuous motion (MCP, MCN) 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. |
| | SENSORACT must be 2 (offset operation), and the system input signal SENSOR must be assigned to an input before offset operations can be used. |

| **Example** | Command | Description |
|---|---|---|
| | `>LIST REGISTER` | #List sequence "REGISTER" |
| | `(  1) SCHGPOS 10; SCHGVR 5` | #Set sensor offset distance to 10, and speed to 5 |
| | `(  2) VR 10; MCP` | #Set running velocity to 10, move continuous (positive) |
| | `(  3) WHILE (SIGMOVE=1)` | #While moving |
| | `(  4)   IF (PCI>50)` | #If we have moved more than 50… |
| | `(  5)     ALMSET` | #Too far. Force an alarm |
| | `(  6)     RET` | #Return: not required. Alarm will abort sequences |
| | `(  7)   ENDIF` | #End IF block |
| | `(  8) WEND` | #End WHILE block |
| | `>` | |

## SENSORACT : SENSOR Input Action

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | SENSORACT=n |
| **Range** | n = 0: Hard Stop (stop as quickly as possible)<br>       1: Soft Stop (controlled deceleration over time)<br>       2: Soft Stop at Fixed Distance from SENSOR Signal<br>       3: No Action |
| **Factory Setting** | 2 |
| **SAVEPRM & RESET** | Required to execute any changes made to the parameter value and to save 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 factory setting. |
| **Access** | READ and WRITE |
| **See Also** | INxxx (INSENSOR), RINxxx (RINSENSOR), MCP, MCN, SCHGPOS, SCHGVR, xxxLV (SENSORLV) |
| **Description** | SENSORACT establishes the stop action taken when the system detects a SENSOR input signal while executing a continuous motion (MCP, MCN).<br><br>・If SENSORACT=0, the system will stop the motor as quickly as possible (hard stop). Stop action is exactly the same as HSTOP.<br><br>・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.<br><br>・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.<br><br>・If SENSORACT=3, even when the SENSOR input is detected, the motor does not stop. Set when using the SENSOR input only in the return-to-mechanical home operation.<br><br>SENSORACT does not affect the use of the SENSOR input during home seeking. |

| **Example** | Command | Description |
|---|---|---|
| | `>INSENSOR 4`<br>` INSENSOR=0(4)` | #Assign SENSOR signal to input 4 |
| | `>SENSORLV 1`<br>` SENSORLV=0(1)` | #Set SENSOR active level to normally closed |
| | `>SENSORACT 1`<br>` SENSORACT=2(1)` | #Set SENSORACT to 1: soft stop |
| | `>SAVEPRM`<br>` (EEPROM has been written 107 times)`<br>` Enter Y to proceed, other key to cancel. y`<br>` Saving Parameters........OK.` | #Save parameters |
| | `>RESET`<br>` Resetting system.`<br>`------------------------------------------`<br>`          CM10-*`<br>`          Controller Module`<br>`          Software Version: *.**`<br>`          Copyright 2010`<br>`          ORIENTAL MOTOR CO., LTD.`<br>`------------------------------------------` | #Reset to activate new settings |
| | `>SENSORACT`<br>` SENSORACT=1(1)` | #Confirm the new value |

## SIGxxx : Status for System Input Signal/System Output Signal

| | |
|---|---|
| **Execution Mode** | Immediate and Sequence |
| **Syntax** | SIGxxx        ("xxx" indicate the signal name on the I/O connector.  ex: SIGABORT) |
| **Range** | 0: "xxx" input/output is not active. <br> 1: "xxx" input/output is active. <br> /: real time monitor (immediate mode only) |
| **Access** | READ        (SIGABORT, SIGSTART: in immediate mode only) |
| **See Also** | INxxx/OUTxxx, IO, xxxLV (except: LSP/LSN, MBFREE), INSG/OUTSG, OUTTEST, REPORT, and "See Also" column in the chart below. |
| **Description** | SIGxxx is the status of system "xxx" input/output (included remote input/output) signal. <br><br> SIGxxx becomes "0 (zero)" when not active, while SIGxxx becomes "1" when active. <br><br> Even if the output signals are not assigned to the output connector, you can check the internal status, using the SIGxxx commands of the output signals. |

\<Input\>

| Command | Signal | Description | See Also |
|---|---|---|---|
| SIGABORT | ABORT | Abort Motion and Sequence Execution | ABORT |
| SIGALMCLR | ALMCLR | Alarm Clear | ALMCLR, RINALMCLR |
| SIGCON | CON | Current ON | CURRENT |
| SIGFREE | FREE | Current OFF, Magnetic Brake Free | FREE |
| SIGHOME | HOME | Home Sensor | HOMETYP, MGHP, MGHN, EHOME, RINHOME |
| SIGCONT | CONT | Continue Motion | PAUSE, PAUSECLR |
| SIGLSP/SIGLSN | LSP/LSN | Limit Switch Positive /Limit Switch Negative | OTLV, OTACT, HOMETYP, MGHP, MGHN, RINLSP/RINLSN |
| SIGMCP/SIGMCN | MCP/MCN | Move Continuously Positive /Move Continuously Negative | MCP/MCN |
| SIGMGHP/SIGMGHN | MGHP/MGHN | Move Go Home Positive /Move Go Home Negative | MGHP/MGHN |
| SIGMSTOP | MSTOP | Motor Stop | MSTOPACT, RINMSTOP |
| SIGPAUSE | PAUSE | Pause Motion | PAUSE, RINPAUSE, PAUSECLR, CONT |
| SIGPAUSECL | PAUSECL | Pause Clear | PAUSECLR, RINPAUSECL, PAUSE, CONT |
| SIGPECLR | PECLR | Position Error Clear | PECLR, RINPECLR, EC, PC, PE, PF |
| SIGPSTOP | PSTOP | Panic Stop | PSTOP, RINPSTOP, ALMACT |
| SIGSENSOR | SENSOR | Sensor | RINSENSOR, SENSORACT |
| SIGSTART | START | Start Sequence | STARTACT |
| SIGTL | TL | Torque Limiting /Push-motion Operation /Current Cutback Release | TL, RINTL |

\<Output\>

| Command | Signal | Description | See Also |
|---|---|---|---|
| SIGALARM | ALARM | Alarm | ALMACT |
| SIGEND | END | Motion End | DEND, ENDACT |
| SIGHOMEP | HOMEP | Home Position | MGHP, MGHN, EHOME, PC |
| SIGLC | LC | Limiting Condition | TL |
| SIGMBFREE | MBFREE | Magnetic Brake Free | ROUTMBFREE |
| SIGMOVE | MOVE | Motor Moving | SIGEND |
| SIGPSTS | PSTS | Pause Status | ROUTPSTS, PAUSE, CONT |
| SIGREADY | READY | Operation Ready | OUTMOVE, MEND |
| SIGRUN | RUN | Sequence Running | ROUTRUN, RUN, ABORT |

**Example**

| Command | Description |
|---|---|
| `>LIST SETTLETIME` | #List sequence SETTLETIME |
| `(  1) MI` | #Start incremental motion |
| `(  2) MEND` | #Wait for motion profile complete (SIGMOVE=0) |
| `(  3) Z=TIMER` | #Store TIMER value |
| `(  4) WHILE (SIGEND=0)` | #While SIGEND=0... |
| `(  5)    T=TIMER-Z` | # ... make variable T be elapsed time |
| `(  6) WEND` | #End of WHILE block |
| `(  7) T` | #Display T: time between SIGMOVE=0 and SIGEND=0 |
| `>VR 20; TD 0.005` | |
| ` VR=20 Rev/sec` | #Set Run velocity and Deceleration time... maybe |
| ` TD=0.005` | aggressive? |
| `>RUN SETTLETIME` | #Run sequence SETTLETIME |
| `>0.027` | #System took ~27 milliseconds to settle after motion |
| `>` | profile finished |
| | |
| `>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 |
| | |
| `>LIST GOHOME` | #List sequence GOHOME |
| `(  1) SAS Home Requested` | #Transmit "Home Requested" |
| `(  2) IF (SIGMOVE=1)` | #If motion in progress |
| `(  3)    SAS System moving, please wait...` | #Transmit wait message |
| `(  4)    MEND` | #Wait for motion to finish |
| `(  5) ENDIF` | #End IF block |
| `(  6) SAS Returning to home position.` | #Transmit returning message |
| `(  7) EHOME` | #Move to position zero |
| `(  8) MEND` | #Wait for motion to complete |
| `(  9) SAS At home position.` | #Transmit finished message |
| `>` | |

## SLACT : Software Position Limit Enable

| | |
|---|---|
| **Execution Mode** | Immediate and Sequence |
| **Syntax** | SLACT n |
| **Range** | n = 0: Software position limits are disabled.<br>    1: Software position limits are enabled after homing. |
| **Factory Setting** | 0 |
| **SAVEPRM & RESET** | Required to execute any changes made to the parameter value and to save in nonvolatile memory.<br>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 factory setting. |
| **Access** | READ and WRITE<br>READ only in Sequences |
| **See Also** | LIMP, LIMN, PC, MGHP, MGHN, EHOME, ALM, ALMACT |
| **Description** | SLACT enables or disables software position limit action.<br><br>When SLACT=1, software position limits LIMP and LIMN are enforced, provided the system has completed a homing action (EHOME, MGHP, MGHN).<br><br>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 defined by OTACT.<br><br>Software limit checking is disabled while a homing operation is in process (MGHP, MGHN, EHOME). (A software position limit alarm may be triggered after a homing operation if PC=0 is not between LIMP and LIMN.)<br><br>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 in the case of absolute motion (MA), the action defined by ALMACT will trigger.<br><br>For continuous motions (MCP, MCN), any out of range condition is detected only as it happens.<br>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. MCP or MCN can be executed, if the motor would move in the direction of the operational range. |
| **Memo** | If LIMP=LIMN=0, software position limit checking is disabled, even if SLACT=1. LIMP and LIMN should be set to appropriate values before enabling software position limit checking. |

| Example | Command | Description |
|---|---|---|
| | ```
>LIMP 10
 LIMP=0(10) Rev
>LIMN -10
 LIMN=0(-10) Rev
>SLACT 1
 SLACT=0(1)
>INHOME 1
 INHOME=0(1)
>HOMETYP 8
 HOMETYP=8
>ALMMSG 2
 ALMMSG=2 [Alarm+Warning]
>SAVEPRM
 (EEPROM has been written 62 times)
 Enter Y to proceed, other key to cancel. Y
 Saving Parameters........OK.
>RESET
 Resetting system.
-------------------------------------------
          CM10-*
          Controller Module
          Software Version: *.**
          Copyright 2010
          ORIENTAL MOTOR CO., LTD.
-------------------------------------------
>MGHP
>SIGHOMEP
 SIGHOMEP=1
>MCP
>Over travel: software position limit
detected.
>PC
 PC=10.001 Rev
>
``` | #Positive position limit: 10 rev<br><br>#Negative position limit: 10 rev<br><br>#Enable position limit checking<br><br>#Assign HOME input to input 1<br><br>#Select HOME type 8<br><br>#Enable automatic transmission of alarm and warning messages<br>#Save new configuration information<br><br><br><br><br>#Reset the system to make new settings active<br><br><br><br><br><br><br><br><br><br>#Find home, start in positive dir.<br>#Check HOME input after operation completes: active<br>#Start continuous motion<br>#Alarm at position limit<br><br>#Check position command<br>#System stopped, just past limit |

## SSTOP : Soft Stop

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | SSTOP |
| **See Also** | TD, HSTOP, MSTOP, MSTOPACT, PSTOP, ABORT |
| **Description** | SSTOP stops the motor with a controlled deceleration. The motor decelerates to start velocity VS over deceleration time TD, and then stops completely. |
| | This command does not stop a sequence program, though the motor stops even in the sequence. |

| **Example** | Command | Description |
|---|---|---|
| | >TD 1.0 | #Set the deceleration time to 1.0 second. |
| | TD=1.0 | #Device response |
| | >VS 2 | #Set the starting velocity to 2 mm/second |
| | VS=2 mm/sec | #Device response |
| | >VR 4 | #Set the running velocity to 4 mm/second |
| | VR=4 mm/sec | #Device response |
| | >MCP | #Move continuously in the positive direction |
| | >SSTOP | #Slow down and stop the motor |
| | >DIS 10 | #Distance equals 10 mm |
| | DIS=10 mm | #Device response |
| | >MI | #Move incremental |
| | >SSTOP | #Slow down and stop the motor |
| | > | |

## STARTACT : START Input Action

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | STARTACT=n |
| **Range** | n = 0: START input starts sequence execution when asserted<br><br>While the sequence is running, START input also resumes the motion that has been paused by PAUSE command or input<br><br>1: START input starts sequence execution when asserted, and aborts sequence execution and motion when cleared. |
| **Factory Setting** | 0 |
| **SAVEPRM & RESET** | Required to execute any changes made to the parameter value and to save 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 factory setting. |
| **Access** | READ and WRITE |
| **See Also** | <ESC>, ABORT, xxxLV (STARTLV) |
| **Description** | STARTACT determines the action associated with the dedicated START input.<br><br>・STARTACT=0<br><br>START can be configured only to start sequences only.<br><br>If START input is turned ON while in a paused situation while the sequence continues running by waiting for the end of paused motion, the remaining motion will be started.<br><br>・STARTACT=1<br><br>START can be configured to act as a toggle: starting sequences when set to its active level and aborting sequences (and motions) when set to its inactive level. |

| **Example** | Command | Description |
|---|---|---|
| | `>STARTACT 1`<br>` STARTACT=0(1)`<br>`>SAVEPRM`<br>` (EEPROM has been written 62 times)`<br>` Enter Y to proceed, other key to cancel. Y`<br>` Saving Parameters........OK.`<br>`>RESET`<br>` Resetting system.`<br>`------------------------------------------`<br>`        CM10-*`<br>`        Controller Module`<br>`        Software Version: *.**`<br>`        Copyright 2010`<br>`        ORIENTAL MOTOR CO., LTD.`<br>`------------------------------------------`<br>`>STARTACT`<br>` STARTACT=1(1)`<br>`>` | #Set the START input action to level detect<br><br>#Save new settings<br><br><br><br><br>#Reset to activate new settings<br><br><br><br><br><br><br><br><br>#Confirm new value |

## STRDCS : Driver Step Angle at System Start

| | |
|---|---|
| **Execution Mode** | Immediate and Sequence |
| **Syntax** | STRDCS=n |
| **Range** | n = 0: CS Output Off at System Start<br>1: CS Output On at System Start |
| **Factory Setting** | 0 |
| **SAVEPRM & RESET** | Required to execute any changes made to the parameter value and to save 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 factory setting. |
| **Access** | READ and WRITE<br>READ only in Sequences |
| **See Also** | DOUTxxx (DOUTCS), DSIGxxx (DSIGCS) |
| **Description** | The STRDCS sets the CS output level of the driver connector on the **CM10**/**SCX10** at system power on.<br>This is to select the motor resolution, by CS (change step = motor resolution selection) function in the driver. |

| **Example** | Command | Description |
|---|---|---|
| | ```
>STRDCS=1
 STRDCS=0(1)
>SAVEPRM
 (EEPROM has been written 21 times)
 Enter Y to proceed, other key to cancel. y
 Saving Parameters........OK.
>RESET
 Resetting system.
-------------------------------------------
          CM10-*
          Controller Module
          Software Version: *.**
          Copyright 2010
          ORIENTAL MOTOR CO., LTD.
-------------------------------------------
>STRDCS
 STRDCS=1(1)
``` | #Set the driver step angle at system start to level detect<br>#Save new settings<br><br><br><br>#Reset to activate new settings<br><br><br><br><br><br><br><br>#Confirm new value |

## STRSW : Current State at System Start

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | STRSW=n |
| **Range** | n = 0: Motor Current Off at System Start<br>        1: Motor Current On at System Start |
| **Factory Setting** | 0: **CM10-1**, **5**<br>1: **CM10-2**, **3**, **4**, **SCX10** |
| **SAVEPRM & RESET** | Required to execute any changes made to the parameter value and to save 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 factory setting. |
| **Access** | READ and WRITE |
| **See Also** | CURRENT, INxxx (INCON) |
| **Description** | STRSW enables or disables motor current immediately after system start.<br><br>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. If STRSW=1, the system supplies current to the motor after a successful startup. |
| **Memo** | If the CON input is not assigned to the I/O connector and CANopen is not active, the CURRENT status at power on is determined by the STRSW setting. If the CON input is assigned to the I/O connector and/or the CANopen is active, the CURRENT status (motor current) at power on is determined by those inputs. |

| **Example** | Command | Description |
|---|---|---|
| | `>STRSW 0`<br>` STRSW=1 (0) [Current ON at start up(Current OFF`<br>`at start up)]` | #Configure for CURRENT=0 at start up |
| | `>SAVEPRM`<br>` (EEPROM has been written 10 times)`<br>` Enter Y to proceed, other key to cancel. Y`<br>` Saving Parameters........OK.` | #Save new settings |
| | `>RESET`<br>` Resetting system.`<br>`------------------------------------------`<br>`          CM10-*`<br>`          Controller Module`<br>`          Software Version: *.**`<br>`          Copyright 2010`<br>`          ORIENTAL MOTOR CO., LTD.`<br>`------------------------------------------` | #Reset to activate new settings |
| | `>CURRENT`<br>` CURRENT=0`<br>`>` | #CURRENT=0 after restart |

## TA  : Acceleration Time

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | TA=n |
| **Range** | n =  0.001 to 500.000 (seconds) |
| **Factory Setting** | 0.5 |
| **SAVEPRM** | 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 factory setting. |
| **Access** | READ and WRITE |
| **See Also** | CV, EHOME, MA, MCP, MCN, MGHP, MGHN, MI, MIx, TD |

| | |
|---|---|
| **Description** | TA affects the initial ramp time for: |

  - MA (move absolute)

  - MI (move incremental)

  - MCP and MCN (move continuously positive and move continuously negative)

  - MGHP and MGHN (move go home  positive and move go home negative)

  - EHOME (return to PC=0)


TA also affects the time required to change speeds, when speeds are increasing (in an absolute sense), for the following motion types:

  - CV (change velocity)

  - MCP and MCN (move continuously positive and move continuously positive)

  - MIx (linked index)


If speeds are decreasing (toward zero), deceleration time TD determines ramp time.

| **Example** | Command | Description |
|---|---|---|
| | >LIST UPANDDOWN | #List sequence UPANDDOWN |
| | ( 1) VS 0.1 | #Start velocity: 0.1 |
| | ( 2) VR 10 | #Run velocity: 10 |
| | ( 3) DIS 150 | #Distance: 150 |
| | ( 4) TA 1 | #Going up: long acceleration time, compared to… |
| | ( 5) TD 0.1 | #…short deceleration time |
| | ( 6) MI | #Start incremental motion |
| | ( 7) MEND | #Wait for motion to finish |
| | ( 8) TA 0.1 | #Going down: short acceleration time, compared to… |
| | ( 9) TD 1 | #…long deceleration time. |
| | ( 10) MA 0 | #Start absolute motion, back to 0 |
| | ( 11) MEND | #Wait for motion to complete. |
| | > | |

## TALK : Select Device

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | TALKid |
| **Range** | id = ⋆, 0 to 9 and A to Z |
| **See Also** | @, ID, \ (BACKSLASH) |
| **Description** | TALK makes a logical connection to a specific device in a multiple device, e.g. daisy chain configuration. That device can then be uniquely addressed and programmed. If the device ID is anything other than the default ID (⋆), communication with the device requires using the @ or TALK commands to establish communication.<br><br>No space is permitted between TALK and id. |
| **Note** | Each device used in a daisy chain communication configuration requires a unique device ID. |

| **Example** | Command | Description |
|---|---|---|
| | `0>MGHP` | #Device 0 go home |
| | `0>TALKA` | #Talk to Device A |
| | `A>MGHP` | #Device A go home |
| | `A>` | |

## TD : Deceleration Time

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | TD=n |
| **Range** | n =  0.001 to 500.000 (seconds) |
| **Factory Setting** | 0.5 |
| **SAVEPRM** | 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 factory setting. |
| **Access** | READ and WRITE |
| **See Also** | CV, EHOME, MA, MCP, MCN, MGHP, MGHN, MI, MIx, TA |
| **Description** | TD is the time used to decelerate the motor (decrease velocity toward zero). |

TD affects the final ramp time for:

- MA (move absolute)

- MI (move incremental)

- MGHP and MGHN (move go home positive and move go home negative)

- EHOME (return to PC=0)

- SSTOP (soft stop)

- MSTOP (motor stop, if MSTOPACT=1)

- ABORT (abort motion and sequence execution)

- <ESC> (ESCAPE character: equivalent to ABORT)

TD also affects the time required to change speeds, when speeds are decreasing (in an absolute sense), for the following motion types:

- CV (change velocity)

- MCP and MCN (move continuously positive and move continuously negative)

- MIx (Linked index)

If speeds are increasing (away from zero), acceleration time TA determines ramp time.

| **Example** | Command | Description |
|---|---|---|
| | >LIST UPANDDOWN | #List sequence UPANDDOWN |
| | ( 1) VS 0.1 | #Start velocity: 0.1 |
| | ( 2) VR 10 | #Run velocity: 10 |
| | ( 3) DIS 150 | #Distance: 150 |
| | ( 4) TA 1 | #Going up: long acceleration time, compared to… |
| | ( 5) TD 0.1 | #…short deceleration time |
| | ( 6) MI | #Start incremental motion |
| | ( 7) MEND | #Wait for motion to finish |
| | ( 8) TA 0.1 | #Going down: short acceleration time, compared to… |
| | ( 9) TD 1 | #…long deceleration time. |
| | ( 10) MA 0 | #Start absolute motion, back to 0 |
| | ( 11) MEND | #Wait for motion to complete. |
| | > | |

## TEACH : Teach Positions

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | TEACH |
| **Commands not Allowed** | MOVE, RUN |
| **See Also** | POS[x] |
| **Description** | 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. If the encoder is used, the motor can also be externally positioned after toggling the motor current off and power to the electromagnetic brake on (if used). |
| | 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. |
| | For a full explanation of the TEACH utility, refer to "8.4 Teaching Positions" on page 94. |

| **Example** | Command | Description |
|---|---|---|

```
>TEACH                                                    #Start the TEACH process

          *** Teach mode ***

 (V)     : Move Cont. Neg.     (M)      : Move Cont. Pos.
 (B)     : Jog Negative        (N)      : Jog Positive
 (Q)     : Current ON/OFF      (F)      : FREE ON/OFF
 (S)     : Save all data to EEPROM
 (K)     : Change Key Interval (50-500[msec])
 (D)     : Change Jog Distance (0.001-500000 [Rev])
             Minimum Movable Distance : 0.001 [Rev]
 <Space> : Immediate Stop
 <Enter> : Data entry mode (Input POS number, then <Enter>)
 <ESC>   : Exit teach mode

 PC=        0.000
```

## TIM  : Select Timing Input Signal

| | |
|---|---|
| **Execution Mode** | Immediate and Sequence |
| **Syntax** | TIM=n |
| **Range** | n =  0: Use the TIMDEXTZ Input Signal for Mechanical home Seeking<br>        1: Use the TIMS Input Signal for Mechanical Home Seeking |
| **Factory Setting** | 1 |
| **SAVEPRM & RESET** | Required to execute any changes made to the parameter value and to save 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 factory setting. |
| **Access** | READ and WRITE<br>READ only in Sequences |
| **See Also** | DINxxx (DINTIMDEXTZ, DINTIMS), DSIGxxx (DSIGTIMDEXTZ, DSIGTIMS), ENC, HOMETYP |
| **Description** | Select the source of the TIMING signal for mechanical home seeking.<br><br>The Z signal on an external encoder is used with TIMD.  The use of the driver timing signal TIMD and the external encoder signal is selected by using the ENC parameter. |

TIMING Source Selection

| TIMING Source | ENC | TIM |
|---|---|---|
| Timing signal·Z-phase pulse differential input<br>TIMD | 1<br>(Driver) | 0<br>(TIMD/EXTZ) |
| Timing signal·Z-phase pulse single ended input<br>TIMS | Unrelated | 1<br>(TIMS) |
| External encoder ZSG<br>EXTZ | 2<br>(External Encoder) | 0<br>(TIMD/EXTZ) |
| No source is selected* | 0<br>(Not Used) | 0<br>(TIMD/EXTZ) |

∗ If ENC is set to zero (0) and TIM is set to 0, the system alarm status will be active when executing the MGHP or MGHN command when the home seeking type uses the timing signal .

Signal Flow Path

Timing signal·Z-phase pulse differential input    TIMD-------1

$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ **ENC**--------0 (TIMD/EXTZ)

External encoder ZSG $\qquad\qquad\qquad\qquad\qquad$ EXTZ-------2     **TIM**----------------→TIMING signal

Timing signal·Z-phase pulse single ended input  TIMS------------------1 (TIMS)

| **Example** | Command | Description |
|---|---|---|

```
>TIM=1
 TIM=0(1)
>SAVEPRM
 (EEPROM has been written 21 times)
 Enter Y to proceed, other key to cancel. y
 Saving Parameters........OK.
>RESET
 Resetting system.
-----------------------------------------
         CM10-*
         Controller Module
         Software Version: *.**
         Copyright 2010
         ORIENTAL MOTOR CO., LTD.
-----------------------------------------
>TIM
 TIM=1(1)
>
```

#Changing the TIM
#Device response
#Save the parameter assignments
#Device response


#Establish the saved parameter values


#Query the TIM setting
#Device response

## TIMER : Running Timer

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | TIMER=n |
| **Range** | n = 0.000 to 500000.000 (seconds) |
| | /: real time monitor (immediate mode only) |
| **Access** | READ and WRITE |
| **See Also** | ALM, WAIT |
| **Description** | TIMER is a running timer, counting seconds. |
| | TIMER is set to zero (0.000) at system start, and counts up from that time, with millisecond resolution. |
| | TIMER overflows at 500,000 seconds (about 5.8 days), and is restarted from zero. |
| | TIMER can be set to any value within its range, for synchronization. |

**Example**

| Command | Description |
|---|---|
| >LIST WATCH | #List sequence WATCH |
| | |
| ( 1) T=TIMER+60 | #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 |
| > | |

## TL  : Torque Limiting/Push-motion Operation/Current Cutback Release

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | TL=n |
| **Range** | n =  0: OFF<br>　　　1: ON |
| **Factory Setting** | 0 |
| **Access** | READ, WRITE |
| **See Also** | DD, DOUTxxx (DOUTTL, DOUTM0, DOUTM1, DOUTM2, DOUTCS), INxxx (INTL), DINxxx (DINLC), OUTxxx (OUTLC), SIGxxx (SIGTL, SIGLC), DSIGxxx (DSIGTL, DSIGLC, DSIGM0, DSIGM1, DSIGM2) |
| **Description** | ・When using a driver that has a torque limiting function (**NX** Series driver etc.), the torque will be limited according to the DD setting while the parameter is set to 1.<br><br>・When using a driver that has a push-motion operation function (**AR** Series driver etc.), the push-motion operation will be performed at the current according to the DD setting while the parameter is set to 1.<br><br>・When using a driver that has a current cutback release function (**CRK/CMK** Series driver etc.), the current cutback will be released while the parameter is set to 1.<br><br>　(While the parameters are set to 1 in each case, the TL output assigned to the driver connector on the **CM10**/**SCX10** will turn ON.)<br><br>This command is effective under the following conditions (the command and driver combination).<br><br>Torque limiting: The driver needs to have the TL input.<br><br>Push-motion operation: The driver needs to have the T-MODE input.<br><br>Current cutback release: The driver needs to have the current cut back release input.<br><br>Also, it is required to assign the each following signals to the driver connector on the **CM10**/**SCX10**.<br>・Torque limiting: TL, M0, M1<br>・Push-motion operation: TL, M0, M1, M2 (Connect TL to the T-MODE terminal of the driver)<br>・Current cutback release: TL (Connect TL to the current cutback release signal input of the driver)<br><br>The TL  function may also be executed via the TL input on the I/O connector and/or the CANopen remote I/O if assigned. When detailed explanation is required, see "6.4 Connecting the I/O signals" on page 23 for I/O connector, and see "10.4 Controlling I/O Message (PDO)" on page 134 for CANopen remote I/O.<br><br>When the LC input is assigned to the driver connector on the **CM10**/**SCX10** and the LC output is assigned to the I/O connector, the LC output will be turned ON when torque reaches to preset value/becomes push-motion condition (position error is 1.8 degree or more).  (DSIGLC command provides the same function.) |
| **Caution** | **With the AR Series driver, the CS input and T-MODE input are assigned to the same pin. The factory setting is the CS input. Change the driver setting to the T-MODE input from the CS input before performing push-motion operation.** |

| **Example** | Command | Description |
|---|---|---|
| | ```Torque Limiting/Push-motion Operation```<br>`>DD=1`<br>`>TL=1`<br>`>TL=0` | <br>#Assign data 1 (M0 is set to ON, M1 M2 to OFF)<br>#Torque limiting operation or push-motion operation is enabled<br>#Release |
| | ```Current Cutback Release```<br>`>TL=1`<br>`>TL=0` | <br>#Execute the current cutback release<br>#Release |

## UNLOCK  : Unlock Sequence

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | UNLOCK target |
| **Range** | Target can be the name or number of any existing sequence. |
| **Commands not Allowed** | RUN |
| **See Also** | DIR, EDIT, LOCK |
| **Description** | UNLOCK unlocks a sequence that has been previously locked with the LOCK command. |
| | A locked sequence cannot be deleted, renamed, or overwritten (by COPY or EDIT). |
| | The sequence directory listing (DIR command) shows the lock status for all sequences. |

| **Example** | 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 |
| | `>` | |

## USBBAUD : USB BAUD Rate

| | |
|---|---|
| **Execution Mode** | Immediate and Sequence |
| **Syntax** | USBBAUD= n |
| **Range** | n =  0: 9600 (bps)<br>　　1: 19200<br>　　2: 38400<br>　　3: 57600<br>　　4: 115200 |
| **Factory Setting** | 0 |
| **SAVEPRM & RESET** | Required to execute any changes made to the parameter value and to save 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 factory setting. |
| **Access** | READ and WRITE<br>READ only in Sequences |
| **See Also** | @, ECHO, ID, TALK, VERBOSE, BAUD |
| **Description** | Establishes the USB Communication BAUD Rate for the device |
| **Note** | The USB on the **CM10**/**SCX10** talks to the virtual COM port on the computer.<br><br>The default USB baud rate of the **CM10**/**SCX10** is 9600 bps, same as the default baud rate of a general Windows computer. If the baud rate on the computer or the **CM10**/**SCX10** is changed, the baud rate must also be changed on the other. (Always set the **CM10**/**SCX10** baud rate first, then set the baud rate of the computer.)<br><br>Check the baud rate of  the computer application that is used to communicate with the **CM10**/**SCX10**, or check the COM port property of Windows if the application does not have a baud rate function.  (The supplied utility software, **IMC** is set to 9600 bps when it is installed and no change is required for initial connection to the **CM10**/**SCX10**.) |

| **Example** | Command | Description |
|---|---|---|
| | ```>USBBAUD 1```<br>` USBBAUD=0(1) [9600bps(19200bps)]`<br>`>SAVEPRM`<br>` (EEPROM has been written 21 times)`<br>` Enter Y to proceed, other key to cancel. y`<br>` Saving Parameters........OK.`<br>`>RESET`<br>` Resetting system.`<br>`----------------------------------------`<br>`         CM10-*`<br>`         Controller Module`<br>`         Software Version: *.**`<br>`         Copyright 2010`<br>`         ORIENTAL MOTOR CO., LTD.`<br>`----------------------------------------` | #Set the baud rate to 19200 bits per second (bps)<br>#Save the parameter assignments<br><br><br><br>#Reset the system to establish the new baud value<br>#NOTE: change baud rate of host system before proceeding! |
| | ```>USBBAUD```<br>` UABBAUD=1(1) [19200bps(19200bps)]` | #Query the baud rate<br>#Baud is set as 19200 |

## UU  : User Units

| Execution Mode | Immediate |
|---|---|
| **Syntax** | UU=User Unit Name |
| **Range** | User Unit Name =  ASCII Characters, 20 Characters Maximum, Except ";" and "@" |
| **Factory Setting** | Rev: **CM10-1**, **2**, **4**, **5**, **SCX10**<br>mm: **CM10-3** |
| **SAVEPRM** | 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 factory setting. |
| **Access** | READ and WRITE |
| **See Also** | DPR, MR |
| **Description** | UU defines the units displayed with position- and velocity- related parameters, when the system is responding verbosely (VERBOSE=1).  Setting UU with DPR (travel distance per one motor revolution) and MR (motor resolution), the position and velocity can be set in units such as mm, inch or degree, which users actually use.  Position-related values are displayed in terms of user units; velocity-related values are displayed in terms of user units per second.<br><br>When VERBOSE=0, only values are displayed: the UU unit information is suppressed.<br><br>Changing UU has no affect on actual motion.<br><br>Setting UU to 0 (digit zero) causes null and the user unit text is cleared. |
| **Example** | Command | Description |
|---|---|---|
| | >UU | #Check user unit text |
| |  UU=Rev | #Still default, 'Rev' |
| | >VR | #Check running velocity |
| |  VR=1 Rev/sec | #Velocity shown in Rev/sec |
| | >UU mm | #Set user unit text to mm (millimeters) |
| |  UU=mm | |
| | >VR 10 | #Set the running velocity to 10 mm/second |
| |  VR=10 mm/sec | |
| | >VS 1 | #Set the starting velocity to 1 mm/second |
| |  VS=1 mm/sec | |
| | >DIS 100 | #Set the distance value to 100 mm |
| |  DIS=100 mm | |
| | >MI | #Start incremental motion |
| | > | |

## VC : Velocity Command

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | VC |
| **Range** | -MAXVEL to +MAXVEL |
| | /: real time monitor  (immediate mode only) |
| | (In sequences, the maximum value is further limited by "Max. Number".) |
| **Access** | READ |
| **See Also** | VR, VS |
| **Description** | VC is the instantaneous velocity command, or set-point. |
| | The sign reflects the motion direction. |
| | VC reflects the velocity that the system is supposed to be running at.  The actual shaft velocity may vary from VC. |

| **Example** | Command | Description |
|---|---|---|
| | ( 1) DIS=1000;VS=0;VR=70;TA=10;TD=10 | |
| | ( 2) MI | |
| | ( 3) WHILE (PC!=DIS) | #When the PC is not DIS |
| | ( 4)    A=0 | |
| | ( 5)    IF (VC>=17.5) | #When the VC is 17.5 or more |
| | ( 6)      A=1 | |
| | ( 7)    ENDIF | |
| | ( 8)    IF (VC>=35) | #When the VC is 35 or more |
| | ( 9)      A=3 | |
| | (10)    ENDIF | |
| | (11)    IF (VC>=52.5) | #When the VC is 52.5 or more |
| | (12)      A=7 | |
| | (13)    ENDIF | |
| | (14)    IF (VC==VR) | #When the VC is VR |
| | (15)      A=15 | |
| | (16)    ENDIF | |
| | (17)    OUT=A | |
| | (18) WEND | |

## VER  : Display Firmware Version

| Execution Mode | Immediate |
|---|---|
| Syntax | VER |
| Description | VER displays the system's firmware version information. |

| Example | Command | | Description |
|---|---|---|---|
| | >VER | | #Display the firmware version |
| | CM10-1 / 2.02 / Sep  6 2011 | | #Typical response (product name / firm ware version number / firm ware) |
| | > | | |

## VERBOSE  : Command Response Control

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | VERBOSE=n |
| **Range** | n =  0: Respond with Data only<br>    1: Respond with Data and Descriptive Text |
| **Factory Setting** | 1 |
| **SAVEPRM** | 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 factory setting. |
| **Access** | READ and WRITE |
| **See Also** | ECHO |
| **Description** | VERBOSE controls the amount of information that the system transmits in response to commands.<br>・When VERBOSE=1 (the default), extra information is transmitted to establish the context of the response. VERBOSE=1 is preferred for human communications.<br>・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. |

**Example**

| Command | Description |
|---|---|
| >VERBOSE | #Check VERBOSE setting |
| VERBOSE=1 | #VERBOSE=1: extra text |
| >PC | #Check position set point |
| PC=1.5 Rev | #Response includes "PC=," value, and user units ("rev") |
| >VR | #Check running velocity |
| VR=1 Rev/sec | #Response includes "VR=," value, and "rev/sec" |
| >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 |
| 1.5 | #Only value returned |
| >VR | #Check running velocity |
| 1 | #Only value returned |
| >ALMMSG | #Check ALMMSG |
| 2 | #Only value returned |
| > | |

## VIEW  : View Parameter

| | |
|---|---|
| **Execution Mode** | Sequence |
| **Syntax** | VIEW element |
| **Range** | 'element' can be the name of any parameter or variable available in sequences. |
| **See Also** | KB, KBQ, SACS, SAS, VERBOSE |
| **Description** | VIEW transmits the value of a parameter or variable without any extra characters. |
| | 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.  The VIEW command only transmits the numeric value, permitting tighter control of the response. |
| **Memo** | In a daisy chain configuration (ID other than *), all output from sequence commands is suppressed unless the device has been previously addressed (via TALK or @). The KB and KBQ commands will not receive input unless the device has been previously addressed. |
| **Example** | |

| 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 POSITION: | #Send ASCII string "POSITION:" with trailing space |
| ( 4) VIEW PF | #Display value of actual position: no extra text |
| >RUN SAYPOS | #Run sequence SAYPOS |
| >POSITION: | #SAS: results in new line, new prompt |
| >14.655 | #First PF: results in new line, new prompt |
| >POSITION: 14.655 | #SACS output, VIEW output: no new line, no new prompt |

# VR : Running Velocity

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | VR=n |
| **Range** | n =  0.001 to MAXVEL (user units/second)<br><br>(In sequences, the maximum value is further limited by "Max. Number".) |
| **Factory Setting** | 1 |
| **SAVEPRM** | 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 factory setting. |
| **Access** | READ and WRITE |
| **See Also** | CV, EHOME, MA, MCP, MCN, MGHP, MGHN, MI, MAXVEL, VS, TA, TD |
| **Description** | VR is the running velocity for motions. VR specifies the peak target speed for the motion, in user units per second.<br><br>VR is always positive. The rotation direction is set by the travel distance (DIS) for incremental motion, and it is set by the position relation between the starting position and target position (MA) for absolute motion.  For continuous operation and return-to-mechanical home operation, set the rotation by the individual motion commands (MCP, MCN, MGHN, MGHP) |
| **Memo** | The change velocity (CV) command overwrites VR with the value designated in the CV command.<br><br>The minimum output frequency on the **CM10**/**SCX10** is 1 Hz. If the running velocity in a user unit is set equivalent to less than 1 Hz, the actual pulse output frequency becomes 1 Hz. |
| **Example** | Command       Description |

```
>VR                    #Check running velocity
 VR=5 Rev/sec
>EHOME                 #Return to position 0 (PC=0)
>VC                    #Check velocity set-point VC
 VC=5 Rev/sec          #VC has reached VR, acceleration finished
>CV 7.5                #Change motion speed to 7.5
>VC                    #Check velocity set-point VC
 VC=7.5 Rev/sec        #VC has reached new speed target 7.5
>VR                    #Check running velocity
 VR=7.5 Rev/sec        #VR now 7.5, overwritten by CV command
>
```

## VRx  : Running Velocity of Link Segment 'x'

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | VRx=n        (x is a number of linked segments: x=0 to 3.) |
| **Range** | n =  0.001 to MAXVEL (user units/second)<br><br>(In sequences, the maximum value is further limited by "Max. Number".) |
| **Factory Setting** | 1 |
| **SAVEPRM** | 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 factory setting. |
| **Access** | READ and WRITE |
| **See Also** | INCABSx, MIx, LINKx, MAXVEL |
| **Description** | VRx is the running velocity for linked motion segment 'x'. VRx specifies the peak target speed for the segment, in user units per second.<br><br>VRx is always positive: the direction for the motion segment is determined by DISx (travel distance/target position) parameter. |
| **Memo** | The minimum output frequency on the **CM10**/**SCX10** is 1 Hz. If the running velocity in a user unit is set equivalent to less than 1 Hz, the actual pulse output frequency becomes 1 Hz. |
| **Example** | Command |

| Command | Description |
|---|---|
| >VR1 5 | #Set the velocity for linked motion segment 1 to 5 user units/second |
|  VR1=5 in./sec | |
| >DIS1 10 | #Set the distance for linked motion segment 1 to 10 user units |
|  DIS1=10 in. | |
| >INCABS1 1 | #Set the move type for linked motion segment 1 to incremental |
|  INCABS1=1 [INC] | |
| >LINK1 1 | #Enable the linked between segments 1 and 2 |
|  LINK1=1 | |
| >VR2 10 | #Linked motion segment 2 velocity equals 10 user units/second |
|  VR2=10 in./sec | |
| >DIS2 20 | #Linked motion segment 2 distance equals 20 user units |
|  DIS2=20 in. | |
| >INCABS2 0 | #Set the move type for linked motion segment 2 to absolute |
|  INCABS2=0 [ABS] | |
| >LINK2 0 | #Unlink segment 2 from segment 3 |
|  LINK2=0 | |
| >MI1 | #Start the linked motion, with segment 1 |

## VS : Starting Velocity

| | |
|---|---|
| **Execution Mode** | Immediate, Sequence and CANopen |
| **Syntax** | VS=n |
| **Range** | n = 0 to MAXVEL (user units/second)<br><br>∗ Although a value of "0" can be set, the minimum value of VS is limited to 1Hz at pulse output. If a user unit value equivalent to less than 1Hz is entered, the VS is internally set to 1Hz. This is to avoid a "zero speed" or "no motion" condition when seeking the mechanical home position because VS = 0.<br><br>(In sequences, the maximum value is further limited by "Max. Number.") |
| **Factory Setting** | 0.1 |
| **SAVEPRM** | 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 factory setting. |
| **Access** | READ and WRITE |
| **See Also** | EHOME, MA, MCP, MCN, MGHP, MGHN, MI, MIx, MAXVEL, VR, TA, TD, VRx |
| **Description** | VS is the starting velocity for all motion types.<br><br>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.<br><br>When a motion is started, speed changes between zero (0) speed and VS instantaneously. (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.<br><br>VS is also used as the running velocity for MGHP and MGHN with HOMETYP=0-3, and used as the velocity for final HOME input detection with HOMETYP=0-11 value. See "8.2.5 Mechanical Home Seeking" on page 83 for more information on home operations. |
| **Example** | Command       Description |

```
>LIST FINDHOME              #List sequence FINDHOME

(   1)  VS 0.25             #For Home operation: set low starting velocity
(   2)  VR 4                #Set running velocity
(   3)  MGHP                #Start seeking home: positive direction first
(   4)  MEND                #Wait for homing operation to complete
(   5)  VS 0                #Set start velocity to 0 for normal operation
(   6)  VR 10               #Set running velocity to 10 for normal operation
>
```

## WAIT  : Wait for Specified Time

| | |
|---|---|
| **Execution Mode** | Sequence |
| **Syntax** | WAIT n |
| **Range** | n =  0.0 to 500000.0 (seconds) |
| **See Also** | KB, KBQ, TIMER, MEND |
| **Description** | WAIT causes sequence execution to wait for the indicated time, before proceeding to the next statement. |

| **Example** | 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 0.1 | #Start incremental motion (distance DIS) |
| | ( 8)   MI | |
| | ( 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 0.5 | |
| | Q=0.5 | |
| | >RUN TENTIMES | |

## WEND  : End of WHILE Block

| Execution Mode | Sequence |
|---|---|
| **Syntax** | WEND |
| **See Also** | ENDIF, ENDL, IF, LOOP, WHILE, BREAKW |
| **Description** | WEND terminates the innermost WHILE block in a sequence.<br><br>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. |

| Example | Command | Description |
|---|---|---|
| | >CREATEVAR N_COUNTS=0<br> New variable N_COUNTS is added.<br> N_COUNTS=0 | #Create user-defined numeric variable named N_COUNTS |
| | >CREATEVAR N_TOTAL=10<br> New variable N_TOTAL is added.<br> N_TOTAL=10 | #Create user-defined numeric variable named N_TOTAL |
| | >LIST MAIN | #List sequence MAIN |
| | (  1) OUT=4 | |
| | (  2) WHILE (N_COUNTS < N_TOTAL) | #N_COUNTS, N_TOTAL user-defined variables |
| | (  3)   MI; MEND | #Start incremental motion; wait until complete |
| | (  4)   OUT4 = 1 | #Set output 4 on |
| | (  5)   WHILE (IN6=0); WEND | #Wait for input 6 to go off |
| | (  6)   OUT4 = 0 | #Set output 4 off |
| | (  7)   WHILE (IN6=1); WEND | #Wait for input 6 to go on |
| | (  8)   N_COUNTS=N_COUNTS+1 | #Increment N_COUNTS by 1 |
| | (  9) WEND | #End of WHILE block |
| | > | |

## WHILE  : Begin WHILE Block

| | |
|---|---|
| **Execution Mode** | Sequence |
| **Syntax** | WHILE (Conditional Expression)    Conditional Expression: element1 {Conditional Operator} element2 |
| **Range** | Conditional Expression |
| **See Also** | WEND, BREAKW, IF, LOOP |
| **Description** | WHILE begins a conditional iterative block. |
| | Statements between the opening WHILE statement and the closing WEND statement execute while the conditional expression is true. If it evaluates to FALSE, sequence processing proceeds to the statement following the closing WEND statement. The conditional expression is evaluated at the beginning of the block only, once per iteration, and it is not tested during execution of the enclosed block statements. |
| | 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 |
| | WHILE statements must be followed (at some point) by a corresponding WEND statement. |
| | When executed, the conditional expression is evaluated.  If it evaluates to TRUE, sequence processing proceeds to the statement following the WHILE. |
| | Block structures (IF-ENDIF, WHILE-WEND, LOOP-ENDL) may be nested, to eight (8) levels deep (nest). |

| **Example** | Command | Description |
|---|---|---|
| | ```
>CREATEVAR N_COUNTS=0
 New variable N_COUNTS is added.
 N_COUNTS=0
>CREATEVAR N_TOTAL=10
 New variable N_TOTAL is added.
 N_TOTAL=10
>LIST MAIN

(  1)  OUT=4
(  2)  WHILE (N_COUNTS < N_TOTAL)
(  3)    MI; MEND
(  4)    OUT4 = 1
(  5)    WHILE (IN6=0); WEND
(  6)    OUT4 = 0
(  7)    WHILE (IN6=1); WEND
(  8)    N_COUNTS=N_COUNTS+1
(  9)  WEND
>
``` | #Create user-defined numeric variable named N_COUNTS<br><br>#Create user-defined numeric variable named N_TOTAL<br><br>#List sequence MAIN<br><br><br>#N_COUNTS, N_TOTAL user-defined variables<br>#Start incremental motion; wait until complete<br>#Set output 4 on<br>#Wait for input 6 to go off<br>#Set output 4 off<br>#Wait for input 6 to go on<br>#Increment N_COUNTS by 1<br>#End of WHILE block |

## xxxLV : System Input Level/System Output Level

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | xxxLV=n    ("xxx" signal input/output assignment) |
| **Range** | < Except: EXTZLV, TIMDLV><br>n = 0: Normally Open<br>    1: Normally Closed<br><br><EXTZLV, TIMDLV><br>n = 0: Positive Logic<br>    1: Negative Logic |
| **Factory Setting** | < Except: TIMDLV> 0<br><TIMDLV><br>0: **CM10-4**<br>1: **CM10-1**, **2**, **3**, **5**, **SCX10**<br>Set to 0 (zero) when the **CM10-1** is combined with the **AR** Series driver DC power input type. |
| **SAVEPRM &<br>RESET** | Required to execute any changes made to the parameter value and to save 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 factory setting. |
| **Access** | READ and WRITE |
| **See Also** | INxxx (Except: INx, EXTZ, TIMD)/OUTxxx, SIGxxx (Except: INx, EXTZ, TIMD), INITDIO, IO, IN/OUT, INSG/OUTSG, INx/OUTx, OUTTEST, and "See Also" column in the chart below. |
| **Description** | Sets the active level of the system "xxx" input/output signal on the I/O connector and driver connector, if used.<br><br>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. |

<Input>

| Command | Signal | Description | See Also |
|---|---|---|---|
| ABORTLV | ABORT | Abort Motion and Sequence Execution | ABORT, <ESC>, ALMACT, HSTOP, MSTOP, MSTOPACT, PSTOP, STARTACT, SSTOP |
| ALMCLRLV | ALMCLR | Alarm Clear | ALMCLR, ALM, ALARMLV, ALMACT, ALMMSG, ALMSET, OUTALARM |
| CONLV | CON | Current ON | CURRENT |
| CONTLV | CONT | Continue Motion | PAUSE, PAUSECL |
| EXTZLV | EXTZ | External Encoder ZSG | DINTIMDEXTZ, DINTIMS, DSIGTIMDEXTZ, ENC, HOMETYP, TIM |
| FREELV | FREE | Current OFF, Magnetic Brake Free | FREE |
| HOMELV | HOME | Home Sensor | HOMETYP, MGHP, MGHN |
| MCPLV/MCNLV | MCP/MCN | Move Continuously Positive /Move Continuously Negative | MCP/MCN |
| MGHPLV/MGHNLV | MGHP/MGHN | Move Go Home Positive /Move Go Home Negative | MGHP/MGHN |
| MSTOPLV | MSTOP | Motor Stop | MSTOP, MSTOPACT |
| OTLV | LSP/LSN | Limit Switch Positive /Limit Switch Negative | - |
| PAUSECLLV | PAUSECL | Pause Clear | PAUSECLR |
| PAUSELV | PAUSE | Pause Motion | PAUSE |
| PECLRLV | PECLR | Position Error Clear | PECLR, EC, PC, PE, PF, RINPECLR |
| PSTOPLV | PSTOP | Panic Stop | PSTOP |
| SENSORLV | SENSOR | Sensor | SENSORACT, RINSENSOR |
| STARTLV | START | Start Sequence | STARTACT |
| TIMDLV | TIMD | Timing Signal·Z-phase Pulse Differential Input | DINTIMDEXTZ, DINTIMS, DSIGTIMDEXTZ, ENC, HOMETYP, TIM |

| Command | Signal | Description | See Also |
|---------|--------|-------------|----------|
| TLLV | TL | Torque Limiting /Push-motion Operation /Current Cutback Release | TL |

<General Input>

| Command | Signal | Description | See Also |
|---------|--------|-------------|----------|
| INxLV (x=1 to 9) | INx | Individual General Input Status | - |

<Output>

| Command | Signal | Description | See Also |
|---------|--------|-------------|----------|
| ALARMLV | ALARM | Alarm | ALARM, ALM, ALMCLR |
| ENDLV | END | Motion End | END |
| HOMEPLV | HOMEP | Home Position | HOMETYP, MGHP, MGHN |
| LCLV | LC | Limiting Condition | LC |
| MOVELV | MOVE | Motor Moving | - |
| PSTSLV | PSTS | Pause Status | - |
| READYLV | READY | Operation Ready | - |
| RUNLV | RUN | Sequence Running | - |

| **Example** | Command | Description |
|-------------|---------|-------------|
| | >ABORTLV 1 | #Set the ABORT input logic to the Normally Closed logic level |
| | ABORTLV=0(1) | |
| | >SAVEPRM | #Save the parameter assignments |
| | (EEPROM has been written 10 times) | #Device response |
| | Enter Y to proceed, other key to cancel. y | |
| | Saving Parameters........OK. | |
| | >RESET | #Establish the saved parameter values |
| | Resetting system. | |
| | ----------------------------------------- | |
| | CM10-* | |
| | Controller Module | |
| | Software Version: *.** | |
| | Copyright 2010 | |
| | ORIENTAL MOTOR CO., LTD. | |
| | ----------------------------------------- | |
| | >ABORTLV | #Confirm ABORT input logic level |
| | ABORTLV=1(1) | |
| | > | |

# 13  Troubleshooting

This chapter explains the system's protective functions and procedures for troubleshooting alarm conditions.

## 13.1 Protective Functions and Troubleshooting

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 some of them 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 the user application and the device itself.
When a protective function is triggered, the ALARM LED on the device blinks and the ALARM 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 the 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 front side of the device.
   An example of the ALARM LED's blinking cycle is shown in the figure below.

   Example: Hardware over travel



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 (see the RESET entry on page 317 in "12 Command Reference" for details of a system reset).
- Turn off the power, wait for a few seconds after the ALARM LED is turned off, then turn the power back on.

Note | If an alarm occurs when the motor is running, clear the alarm condition after the deceleration time (TD) that is set. The motor may restart if the alarm condition is cleared within the deceleration time.

# 13.2 Types of Protective Functions and Alarms

| Phenomenon | Alarm Code | ALARM LED Blinks | ALMCLR Effect | Protective Function | Description | Action |
|---|---|---|---|---|---|---|
| Motion and sequence execution stop | 32h | 1 | Clears alarm | Out of position range | The PABS value exceeded the coordinate control range (-2,147,483.648 to +2,147,483.647). | Check that PABS is in the range. |
| | 90h | | | Stack overflow | Sequence memory "stack" exhausted | Restructure sequences to reduce the number of nested blocks or subroutine calls |
| | 94h | | | Sequence reference error | Attempt to call a non-existing sequence as a subroutine | Revise the CALL statement or rename the intended target sequence |
| | 98h | | | Calculation overflow | Sequence calculation result exceeded numerical limits | Check math operators, make sure they cannot overflow |
| | 99h | | | Parameter range error | Attempt to set a parameter to a value outside its range | Make sure all assignments stay within defined limits |
| | 9Ah | | | Zero division | Attempt to divide by zero | Check division operations, test divisor for zero before division |
| | 9Dh | | | PC command 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 |
| | 9Eh | | | 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 |
| | 9Fh | | | Parameter write error | Attempt to change a parameter under invalid condition | Check if you tried to write a parameter, which is not allowed to write to, during operation. |
| | A0h | | | 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. |
| | E0h | | | User alarm | ALMSET command intentionally executed | If a user alarm was not expected, check sequence programming for inappropriate ALMSET command(s) |
| | 10h | 4 | | Excessive Position Deviation | When performing the MEND command or mechanical home seeking operation, the END signal was not output in the time set by ENDWAIT. | If "DEND=0," check whether the overload was occurred or ENDACT (END range) was too small. If "DEND=1," check whether the driver END signal is connected, the overload was occurred or the END signal range of the driver was too small. |

| Phenomenon | Alarm Code | ALARM LED Blinks | ALMCLR Effect | Protective Function | Description | Action |
|---|---|---|---|---|---|---|
| Motion and sequence execution stop | 60h | 7 | Clears alarm | LS logic error | Positive and negative position limit signals on simultaneously | - Check limit sensors and wiring.<br>- Check input signal configuration.<br>- Check the logic setting for limit sensors (OTLV): Normally open (N.O.) or Normally closed (N.C.). |
| | 61h | | | LS connected in reverse | Positive or negative position limit signal detected opposite home seeking direction | |
| | 62h | | | HOME operation failed | Unstable or unexpected position limit signal detected while seeking home position | |
| | 63h | | | HOMELS not found | No HOME input detected between position limit signals while seeking home position | Check HOME sensor wiring and connections |
| | 64h | | | TIM, SENSOR signal error | TIM position or SENSOR input expected with HOME input: not found | Selected mechanical home seeking operation (see HOMETYP) requires a valid SENSOR input and/or a valid TIM position while HOME input active. Make sure HOME and other required input(s) can be active at the same location. |
| | 6Ah | | | 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 |
| | 6Eh | | | Driver alarm | Driver alarm signal is active | Check the driver and see the operating manual of the driver. |
| | 6Fh | | | Driver connection error | The command was canceled due to no response from the driver during executing the command or before executing the command | Be sure to check if driver and the **CM10/SCX10** are connected securely. |
| | 70h | | | Motion parameter error | Attempt to execute motion with incompatible motion parameters | - Make sure current is enabled (CURRENT=1).<br>- Home seeking: make sure required inputs are configured.<br>- Linked indexing: make sure all linked segments execute in the same direction. |

| Phenomenon | Alarm Code | ALARM LED Blinks | ALMCLR Effect | Protective Function | Description | Action |
|---|---|---|---|---|---|---|
| Motion and sequence execution stop.<br><br>Motor may or may not have holding torque, depending on ALMACT. | 68h | 6 | Clears alarm | Panic stop | System executed a panic stop because of a PSTOP input or command | If a panic stop was unexpected:<br>- Check PSTOP input configuration.<br>- Check sequence programming for inappropriate PSTOP command(s). |
| | 66h | 7 | | Hardware over travel | Positive or negative position limit signal detected | - Check motion parameters.<br>- Make sure home position is correct.<br>- Check limit sensors and wiring.<br>- Check input signal configuration.<br>- Check the logic setting for limit sensors (OTLV): Normally open (N.O.) or Normally closed (N.C.). |
| | 67h | | | Software over travel | Position outside of programmed positive and negative position limits | - Check motion parameters.<br>- Check software position limits.<br>- Make sure home position is correct. |
| The motor lacks holding torque. | 41h | 9 | No effect | EEPROM error | User data in non-volatile EEPROM memory is corrupt | Contact Oriental Motor to arrange for inspection or repair. |
| | F0h | ON | | System error | System detected unexpected internal logic state | |
| | F1h | | | Memory error | Internal memory access error | |
| | F2h | | | Sequence internal error | Sequence code invalid or corrupt | |

# 14  Inspection

It is recommended that periodic inspections be conducted after each operation of the device.
If an abnormal condition is noted, discontinue any use and contact your nearest office.

## ■ During Inspection

- Are any of the device mounting screws loose?
- Is there any peeling of the tape fastener located between the device and the driver?
- Are there any strange smells or appearances in the device?

**Note** The device uses semiconductor elements, so be extremely careful when handling it. Static electricity may damage the device.

# 15 Specifications

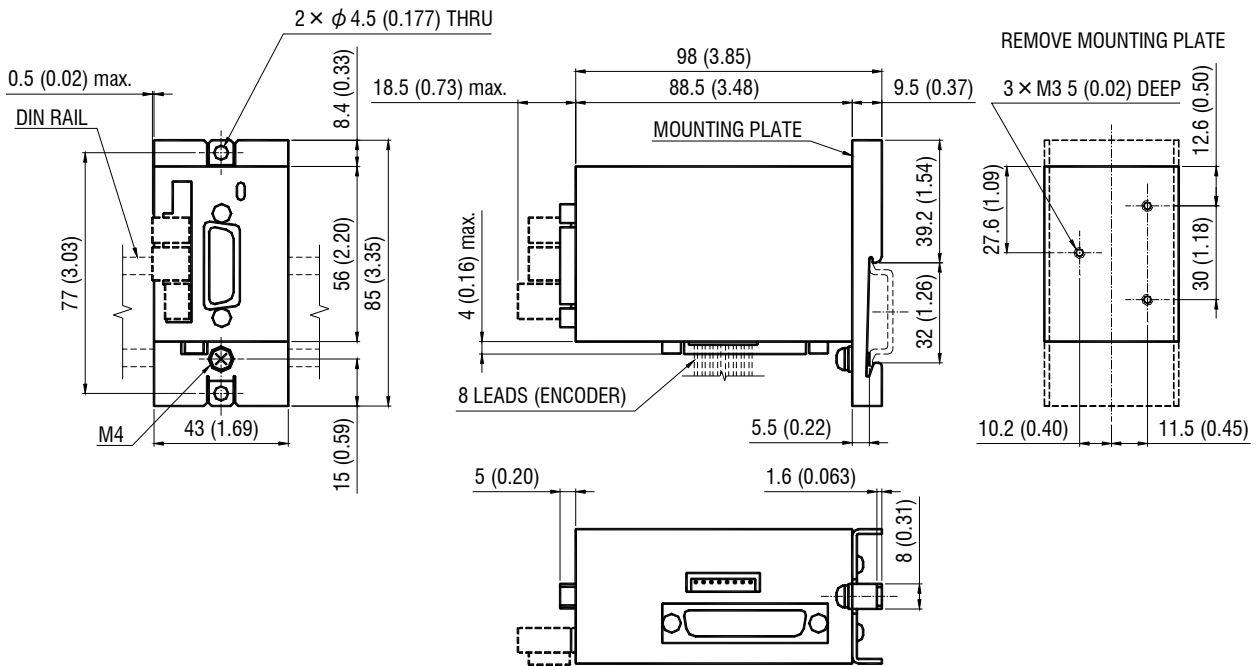| | Operation Mode | Immediate/Stored program |
|---|---|---|
| Programs | Number of Programs | 100 |
| | Size | Total sequences: 6 kB (compiled)  21 kB (text + compiled)<br>1 sequence: 6 kB (text)  2 kB (compiled) |
| | Programming Method | **Immediate Motion Creator for CM/SCX Series** (supplied software)<br>or General terminal software |
| | Function Example | Subroutines, Math/Logical/Conditional operators, User variables |
| Control | Number of Control Axis | Single axis |
| | Control Modes | Incremental motion<br>Absolute motion<br>Move continuously<br>JOG motion<br>Mechanical home seeking operation<br>Link motion (incremental motion・absolute motion・move continuously) |
| | Starting Velocity | 0 to 1.24 MHz (1 Hz increments) |
| | Speed Range | 1 Hz to 1.24 MHz (1 Hz increments) |
| | Acceleration and Deceleration Time | 0.001 to 500 sec (0.001 sec increments) |
| | Position Range | -2,147,483,648 to +2,147,483,647 pulses |
| | Mode for Mechanical Home Seeking | 3 sensor mode, 2 sensor mode, 1 sensor mode<br>(+LS, -LS, HOME, SENSOR, TIM) |
| | Features | User unit, Teaching positions, Multi axis operation,<br>Driver deviation counter, Protective functions |
| Driver Interface | Pulse Output | 1 pulse/2 pulse mode<br>Rise/fall time: 0.1 μs or less<br>Duty cycle: 50% (ON is fixed to 20 μs when frequency is 25 kHz or less)<br>Line driver output (line receiver input/photo-coupler input compatible) |
| | Input | 5 signals  Photo-coupler input<br>Input voltage 4.25 -26.4 V  Input resistance 3 kΩ<br>Built-in 5/24 VDC power supply   Sink logic/Source logic compatible |
| | Output | 8 signals  Photo-coupler open-collector outputs<br>30 VDC  20 mA or less<br>Built-in 5/24 VDC power supply   Sink logic/Source logic compatible |
| | Encoder Input | A-phase, B-phase, Z-phase/Timing signal   Max. frequency  1 MHz<br>Differential  (AM26LV32 or equivalent) |
| External Encoder Input | | A-phase, B-phase, Z-phase/Timing signal   Max. frequency  1 MHz<br>Differential  (AM26LV32 or equivalent)<br>Line-driver, Open collector and TTL compatible<br>Built-in 5 VDC power supply |
| I/O | Input | 9 signals (configurable)  Photo-coupler inputs<br>Input voltage 4.25-26.4 V Input resistance 5.4 kΩ |
| | Output | 4 signals (configurable) Photo-coupler open-collector outputs<br>30 VDC  20 mA or less |
| Serial Communication | USB | USB2.0 compatible (virtual COM port)  USB mini-B terminal<br>9600, 19200, 38400, 57600, 115200 bps (9600 bps is factory setting.) |
| | RS-232C | Start-stop synchronous method, NRZ (non-return zero), full-duplex<br>8 bits, 1 stop bit, no parity<br>9600, 19200, 38400, 57600, 115200 bps (9600 bps is factory setting.)<br>Daisy chain compatible (up to 36 axis) |
| | CANopen | CiA 301 Ver4.02 compliant<br>10 kbps, 20 kbps, 50 kbps, 125 kbps, 250 kbps, 500 kbps, 800 kbps, 1 Mbps<br>Certified by CiA (CiA201208-301V402/20-0155) |
| Power Input | Voltage | 24 VDC ± 10% |
| | Current | 0.26 A |
| Mass | | 0.33 kg (0.73lb) |

## ■ General Specification

| Operation Environment | Degree of protection | IP20 |
|---|---|---|
| | Ambient temperature | 0 to +50 °C (+32 to +122 °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 | −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 | −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 |

## ■ Dimensions

unit: mm (inch)

• **SCX10**

# Appendix A   How to Send Commands Using ASCII Strings

## ■ 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 | 0Ah (10) |
| [CR] | Carriage Return | 0Dh (13) |
| [SP] | SPace | 20h (32) |
| [ESC] | ESCape | 1Bh (27) |
| [EOL] | End Of Line<br>Any of the following combinations<br>[CR]<br>[LF]<br>[CR] [LF]<br>[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 [7Ah] | 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].<br>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. | |

## ■ Command Format

Following shows the command format.

Case (Upper/Lower) of the character is not a matter unless specified. Decimal point number is accepted in some of the parameters.

### • Parameters

No '=' (only spaces) can be accepted only for constant (immediate) value entry.

(Format) [Parameter] [=(2Dh)] [Parameter value] [EOL]
　　[Parameter] [SP] [Parameter value (constant)] [EOL]

■ Examples

| Condition | Example | Notes |
|-----------|---------|-------|
| Parameter value is constant | `DIS=1.234[EOL]`<br>`DIS 1.234[EOL]` | '=' can be replaced with [SP] |
| Parameter value is variable | `DIS=A[EOL]` | '=' is required |
| Parameter value is equation | `DIS=A*1.5[EOL]` | '=' is required |

### • Commands

Spacing between command and parameter by at least 1 [SP] is required. '=' is not accepted.

(Format) [Command] [SP] [Argument] [EOL]

■ Examples

| Condition | Example | Notes |
|-----------|---------|-------|
| No argument | `MI[EOL]` | - |
| Argument is constant | `MA 1.234[EOL]` | '=' is not accepted |
| Argument is variable | `MA POS[1][EOL]` | '=' is not accepted |
| Argument is string | `RUN Test[EOL]` | '=' is not accepted |

# Appendix B  TIPS

The **SCX10** has useful functions that may not be found immediately. This section shows references that might conveniently be used.

## ■ Motion Command

- Changing velocity during motion
  - If moving for positioning by MI or MA, use the command CV to set the new desired speed.
  - If moving continuously by MCP, set the new VR, and execute the MCP command again.
  - If moving continuously by MCN, set the new VR, and execute the MCN command again.
  - Set SENSORACT=2.  Moving continuously by MCP or MCP command, the system changes velocity to SCHGVR, and stops at a distance SCHGPOS after the position at which the SENSOR signal was set.
  - If all motion parameters are known, use linked index motions.  Refer to the MIx command.

- Moving to an absolute position
  - Command MA with desired absolute destination.

- Making home position an offset from valid home signal
  - Use OFFSET command with desired offset value.

- Stopping with a position offset from sensor position
  - Set SENSORACT=2.  Moving continuously by MCP or MCP command, the system changes velocity to SCHGVR, and stops at a distance SCHGPOS after the position at which the SENSOR signal was set.
  - Set SENSORACT=2, SCHGPOS and SCHGVR with desired offset position and velocity.

- Stopping motion or sequence while running on the fly by keyboard
  - Press ESC key.

- Verifying that the motor has not missed any steps
  - Using sequence program, check PE parameter during or at end of motion, or verify END status at end of motion.
    (Applicable when encoder is used, or driver has encoder output or END output.)

- Outputting signal when distance, velocity, time reaches to desired value
  - Set EVx with desired output port and desired parameter. (No program is required)

## ■ System Control

- Clearing all contents (parameters, POS[x] and programs) that have been written and return to factory setting
  - Use the CLEARALL command.

- Inverting all directions at once
  - Set DIRINV followed by a desired direction, and save and reset.

- Compensating mechanical gear ratio, or handling indivisible DPR parameter
  - Set electric gear parameters, GA (numerator) and GB (denominator) opposite to mechanical gear ratio.

- Setting the motor current ON or OFF when the system is powered ON
  - Set STRSW parameter with desired state.

- Measuring time elapsed from system startup or distance between intended times
  - Use the TIMER command, or set TIMER to zero to start measuring time.

- Measuring current velocity while operating
  - Use the VC command.

## ■ Maintenance

- Viewing I/O status all at once
  - Use the IO command for I/O connector signals, the DIO command for driver connector on the **SCX10** signals and the RIO command for remote I/Ox (CANopen) signals.

- Viewing individual input state, viewing and toggling individual output state
  - Use the INx and OUTx commands for I/O connector signals, the DINx and DOUTx commands for driver connector on the **SCX10** signals, and the RINx and ROUTx commands for remote I/Ox (CANopen) signals.

- Toggling output state on the fly by keyboard (with viewing I/O status)
  - Type "OUTTEST."

- Viewing the list of commands during communication
  - Type "HELP", press space key for next page.

- Checking the present alarm condition and history
  - Type "ALM."

- Clearing an alarm condition
  - Type "ALMCLR", or type "RESET" or cycle input power.

- Checking I/O assignments and status, values of important parameters, current position and alarm condition all at once
  - Type "REPORT", press space key for next page.

- Viewing varying parameter in real time
  - Attach " /" (space + forward slash) after command.  (Example: "PC /")

- Displaying sequence statements as they are executed in real time
  - Set TRACE=1.

## ■ Communication

- Using multiple statements in a line
  - Set one command/parameter and use separator ';' and set other command/parameter and repeat. (Example: DIS=10;MI…)  ∗ It can be used within a sequence.

- Facilitate key typing for parameter entry
  - Press space key instead of '='.

- Commanding to all devices at once
  - Attach "\" (back slash) before command.  (Examples: \MI, \ABORT)

- Eliminating extra information (i.e., parameter name) on response to reduce time required to respond
  - Set VERBOSE=0.  (recommended for automatic control)

- Showing only response from **SCX10** and eliminating echo (contents that were sent from host) on operating screen
  - Set ECHO=0.

- Decreasing communication delay on RS-232C daisy chain connection
  - Set BAUD parameter to grater value to increase the transmission rate.

- Using limit switches and home sensor via CANopen for a "sensor motion" or a mechanical home seeking
  - Assign sensor signals using RINSENSOR, RINLSP, RINLSN commands.
    (A delay may occur depending on a condition.)

## ■ **Sequence Program**

- Commenting inside a sequence
  - Use the'#' followed by a commenting text.

- Temporarily disable specific command lines in a sequence
  - Insert a '#' before command line.

- Using user variable, either numeric value or text string in a sequence
  - Set desired variable with CREATEVAR command.

- Using math/logical operators in program
  - The following operations can be used.
    + : Addition, - : Subtraction, * : Multiplication, / : Division, % : Modulo (remainder), & : AND (Boolean), | : OR (Boolean), ^ : XOR (Boolean), << : Left arithmetic shift (Shift to left bit),
    \>> : Right arithmetic shift (Shift to right bit)

- Using subroutines
  - Use CALL command to call a sequence program as a subroutine.- Write RET in the last of the sequence program to be used as the subroutine and call it using the CALL command from the main program

- Automatically asking operator or machine to provide and accepting numeric value to be entered
  - Use KB or KBQ.
    (Example: Set DIS=KB, DIS becomes numeric value that operator typed when it was asked)

- Suspending sequence processing until motion is complete
  - Insert the MEND command between the motion start command and next command

- Display the text on the screen, or control the display by sending the ASCII control code
  - Write the text following the SAS and/or SACS commands, the SACS command can describe the ASCII control code

- Protecting from modifying sequence
  - Use LOCK command followed by sequence name.

- Run a sequence automatically when the system is powered ON
  - Use the name CONFIG as a sequence name.

- Unauthorized reproduction or copying of all or part of this Operating 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 patent-related problem arising in connection with the use of any information, circuit, equipment or device described in the 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 office.
- *Orientalmotor* 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 trademarks or registered 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.