HM-4014-3

# αSTEP-One

# OPERATING MANUAL

## ASX66

CE

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.

Only qualified personnel should work with the product.
Use the product correctly after thoroughly reading the “1.2 Safety Precautions” section on page 11.
The product described in this manual has been designed and manufactured for use in general industrial machinery, and must not be used for any other purpose. Oriental Motor Co., Ltd. is not responsible for any damage caused through failure to observe this warning.

## ■ How to Read This Operating Manual

This operating manual explains the handling and safety instructions regarding the αSTEP-One device (motor/driver/controller).
The structure of the manual is outlined below.
Read the appropriate pages during use with reference to the following system configuration diagram.

Structure of Operating Manual

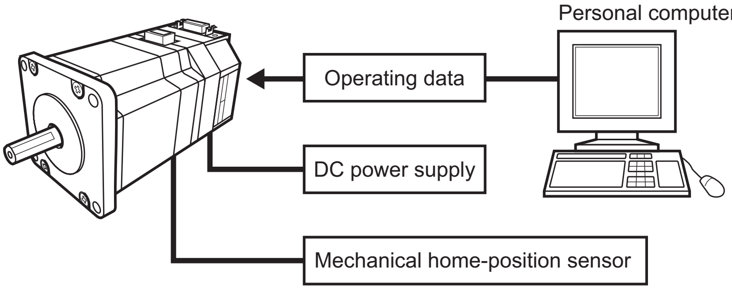
| Chapter | Contents |
|---|---|
| Chapter 1 Before Use | Explains the items you should know before using the device. |
| Chapter 2 Quick Setup | Explains the methods to use the device quickly and easily. |
| Chapter 3 Installation and Connection | Explains the names and functions of parts, the installation, and wiring in compliance with EMC Directives.<br>Explains the methods of connecting sensors, external signals, and power supply, as well as the grounding method, connection examples and control inputs/outputs. |
| Chapter 4 Features | This chapter explains the main functions. |
| Chapter 5 Program Creation and Execution | Explains the methods used to create new programs, edit existing programs and execute programs. |
| Chapter 6 Command List | Explains the keys and commands used in communication between the device and terminal program. |
| Chapter 7 Troubleshooting | Explains the protective functions, procedures for inspection, and troubleshooting/diagnostics. |
| Appendix A Model Number | Describes the model-number format, options, etc. |
| Appendix B Sample Programs | Describes sample programs. |
| Appendix C Daisy Chain Connection Procedure | Describes the procedure for daisy chain connection. |
| Appendix D Timing Charts | Describes the timing charts. |
| Appendix E Command Cross Reference | Describes a brief command cross reference. |
| Appendix F ASCII Data | Describes a listing of common ASCII codes. |
| Appendix G Command Format | Describes the type format of command and parameter. |

## ■ System Configuration

The following items are required in order to use the αSTEP-One device

- A 12 to 48 VDC power supply.
- A device that can operate in a terminal mode (e.g. a PC running Windows Hyper Terminal).
- An RS-232C communications cable between the device and the terminal device.
- The mechanical home seeking function requires home-position sensors.

### Typical Configuration




## ■ Installation Conditions (EN Standard)

The device is to be used as a component within other equipment.
Over voltage category I
Pollution degree 2
Class I equipment

The user is advised to perform the following treatments when conducting product installation and connection.

- This product is designed for use within machinery, so it should be installed within an enclosure.
- For the device's power supply, use a power supply with reinforced insulation on its primary and secondary sides.
- Connect the device to a Protective Earth (PE) for safety. See "3.10 Grounding the Device" on page 32 for details.

## ■ CE Marking

### For EMC Directive

This product has received EMC compliance under the conditions specified in "3.3 Installing and Wiring in Compliance with EMC Directive" on page 18.
The compliance of the final machinery with the EMC Directives 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 testing by the customer of the final machinery.

### Applicable Standards

| | | |
|---|---|---|
| EMI | Emission Tests<br>Radiated Emission Test | EN61000-6-4<br>EN55011 |
| EMS | Immunity Tests<br>Radiation Field Immunity Test<br>Electrostatic Discharge Immunity Test<br>Fast Transient/Burst Immunity Test<br>Conductive Noise Immunity Test | EN61000-6-2<br>IEC61000-4-3<br>IEC61000-4-2<br>IEC61000-4-4<br>IEC61000-4-6 |

### For Low Voltage Directive

This product is not subject to the EC's Low Voltage Directive because its input power-supply voltage is 12 to 48 VDC.

## Table of Contents


**Chapter 1 Before Use ................................ 8**

1.1 Introduction ....... 8
1.2 Safety Precautions ....... 11
1.3 Precautions for use ....... 12

**Chapter 2 Quick Setup ........................... 13**

2.1 Overview ....... 13
2.2 Connecting the Power Supply ....... 13
2.3 Connecting the Personal Computer (PC) ....... 14
2.4 Immediate Motion Creator (IMC) ....... 14
2.5 Making a Positioning Move ....... 14

**Chapter 3 Installation and Connection .. 16**

3.1 Names and Functions of Parts ....... 16
3.2 Installation ....... 17
3.3 Installing and Wiring in Compliance with EMC Directive ....... 18
3.4 Connecting the I/O ....... 20
3.5 Input Signals ....... 21
3.6 Output Signals ....... 24
3.7 Connecting the Device to a Personal Computer ....... 27
3.8 Connection Example ....... 30
3.9 Connecting to the Power Supply ....... 32
3.10 Grounding the Device ....... 32

**Chapter 4 Features ................................. 33**

4.1 Overview ....... 33
4.2 Making the Motor Move ....... 33
4.3 Initial Setting (User Unit) ....... 34
4.4 Motion Types ....... 36
4.5 Enhanced Features ....... 46
4.6 Stopping Motion ....... 50
4.7 Support Functions ....... 51
4.8 Pulse Input Modes ....... 55
4.9 Protective Functions ....... 56

**Chapter 5 Program Creation and Execution ................................ 58**

5.1 Overview of Operation ....... 58
5.2 Communication and Terminal Specifications ....... 59
5.3 Creating a New Sequence ....... 59
5.4 Editing an Existing Sequence ....... 61
5.5 Executing a Sequence ....... 66
5.6 Error Messages Displayed on the Terminal ....... 67

**Chapter 6 Command List ......................... 73**

# : Sequence Comment ....... 81
+, −, *, /, %, &, |, ^, <<, >> ....... 82
/ (Forward slash) : Display Continuously ....... 83
; : Statement Separator ....... 84
@ : Select Device ....... 85
<ESC> : (Escape) Abort Operation(s) ....... 86
a!=b, a<=b, a<b, a=b, a>=b, a>b : Conditional Operators ....... 87
ABORT : Abort Sequence and Motions ....... 88
ABORTLV : ABORT Input Level ....... 89
ALARMLV : ALARM Output Level ....... 90
ALM : Alarm Status and History ....... 91
ALMACT : Alarm Action ....... 92
ALMCLR : Clear Alarm ....... 93
ALMMSG : Alarm Message Action ....... 95
ALMSET : Set User Alarm ....... 96
A to Z : User Variables ....... 97
\ : Global Command ....... 99
BAUD : Communication Baud Rate ....... 100
BREAKL : Break LOOP Block ....... 101
BREAKW : Break WHILE Block ....... 102
CALL : Call Sequence as Subroutine ....... 103
CLEARALL : Clear All Programming (return to factory condition) ....... 104
CLEARPOS : Clear POS[x] Position Array Data ....... 105
CLEARSEQ : Clear sequences ....... 106
CLEARVAR : Clear User-Defined Variables ....... 107
CMODE : Current Mode ....... 108
CONT : Continue Motion ....... 109
COPY : Copy Sequence ....... 111
CRACC : Acceleration Current ....... 112
CRDEC : Deceleration Current ....... 113
CREATEVAR : Create User-Defined Variable ....... 114
CROFFLV : Current Off Input Level ....... 115
CRRUN : Run Current ....... 116
CRSTOP : Stop Current ....... 117
CURRENT : Current On/Off ....... 118
CV : Change Velocity ....... 119
DEL : Delete Sequence ....... 120
DELETEVAR : Delete User-Defined Variable ....... 121
DIR : Sequence Directory ....... 123
DIRINV : Direction Invert ....... 124
DIS : Incremental Motion Distance ....... 125
DISx : Linked Motion Distance or Destination ....... 126

DPP : Distance Per Pulse .... 127
DPR : Distance Per Revolution .... 128
DTMP : Drive Temperature .... 130
DTMPMAX : Maximum Driver Temperature .... 131
DTMPWRN : Drive Warning Temperature .... 132
DVIN : Drive Input Voltage .... 133
DVINSET : Nominal Drive Input Voltage .... 134
ECHO : Communications Echo Control .... 135
EDIT : Edit Sequence .... 136
EHOME : Return to Electrical Home .... 137
ELSE
: Begin ELSE Block: execute if IF is false .... 138
END : End Sequence .... 139
ENDIF : End of IF Block .... 140
ENDL : End of LOOP Block .... 141
ENDLV : END Output Level .... 142
EVx : Configure Event Output .... 143
FILT : Jerk Filter Time Lag .... 144
GA, GB : Electrical Gear Ratio .... 145
HELP : Display Help Information .... 147
HOMELV : HOME Input Level .... 148
HOMEPLV : Home Position Output Level .... 149
HOMETYP : Homing Type .... 150
HSTOP : Hard Stop .... 151
IA : Motor Phase Current Amplitude .... 152
ID : Device ID .... 153
IF : Begin IF Block: execute if true .... 154
IN : General Input Status .... 155
INCABSx : Linked Move Type .... 157
INCROFF
: Current Off Signal Input Assignment .... 158
INHOME : HOME Signal Input Assignment .... 159
INITIO : Initialize I/O .... 160
INITPRM : Initialize Parameters .... 161
INLSN, INLSP
: Limit Switch Negative & Positive Signal Input Assignments .... 162
INMSTOP
: Motor Stop Signal Input Assignment .... 163
INPAUSE : PAUSE Signal Input Assignment .... 164
INPAUSECL
: Pause Clear Signal Input Assignment .... 165
INPSTOP
: Panic Stop Signal Input Assignment .... 166
INSENSOR
: SENSOR Signal Input Assignment .... 167
INSG : System Input Signal Status .... 168
INx : Individual General Input Status .... 169
INxLV : INx Input Level .... 170
IO : Input/Output Status .... 171
KB : Keyboard Input .... 172
KBQ : Keyboard Input (Quiet) .... 173
LDCHK : Estimate Load Parameters .... 174
LF : Load Function .... 176
LG : Load Gravity .... 177
LI : Load Inertia .... 178
LIMN, LIMP : Software Position Limits .... 179
LINKx : Link Control .... 181
LIST : List Sequence Contents .... 182
LISTVAR : Lists All User-Defined Variables .... 183
LOCK : Lock Sequence .... 184
LOOP : Begin Counted LOOP Block .... 185
LSF : Load Static Friction .... 186
MA : Move to Absolute Position .... 187
MAXPOS : Maximum Position Value .... 189
MAXVEL : Maximum Velocity Value .... 190
MAXOVERFLOW : Maximum Overflow .... 191
MCN, MCP : Move Continuously .... 192
MEND : Wait for Motion End .... 193
MGHN, MGHP
: Seek Mechanical Home Position .... 194
MI : Move Incremental Distance .... 195
MIx : Start Linked Index .... 196
MODE : Device Mode .... 198
MOVELV : Move Output Level .... 200
MSTOP : Motor Stop .... 201
MSTOPACT : Motor Stop Action .... 202
MSTOPLV : Motor Stop Input Level .... 203
MTMP : Motor Temperature .... 204
MTMPMAX : Maximum Motor Temperature .... 205
MTMPWRN : Motor Warning Temperature .... 206
N_xxx : User-Defined Numeric Variables .... 207
OFFSET : Home Offset Position .... 208
OLTIME : Overload Time .... 209
OTACT : Overtravel Action .... 210
OTLV : Overtravel Input Level .... 211
OUT : General Output Status .... 212
OUTALARM
: ALARM Signal Output Assignment .... 213
OUTEND : END Signal Output Assignment .... 214
OUTHOMEP
: Home Position Signal Output Assignment .... 215
OUTMBC
: Magnetic Brake Control Signal Output Assignment .... 216
OUTMOVE
: MOVE Signal Output Assignment .... 217

OUTPSTS
: Pause Status Signal Output Assignment ......... 218
OUTRUN : RUN Signal Output Assignment .... 219
OUTSG : System Output Signal Status ............. 220
OUTTEMP
: TEMP Signal Output Assignment................... 221
OUTTEST : I/O Test Utility .............................. 222
OUTx : Individual General Output Control....... 223
OVERFLOW : Position Error Limit.................. 224
OVERVEL : Velocity Limit............................... 225
PAUSE : Pause Motion...................................... 226
PAUSECLR :Clear State of Paused Motion ...... 227
PAUSECLLV : Pause Clear Input Level............ 228
PAUSELV : PAUSE Input Level ....................... 229
PC : Position Command .................................... 230
PCI : Incremental Position Command ............... 231
PE : Position Error............................................. 232
PF : Motor Position............................................ 233
PFI : Incremental Motor Position ...................... 234
POS [x] : Position Array Data............................ 235
PSTOP : Panic Stop ........................................... 236
PSTOPLV : Panic Stop Input Level................... 237
PSTSLV : Pause Status Output Level ................ 238
RC : Motor Shaft Position ................................. 239
REN : Rename Sequence.................................... 240
REPORT : Display System Status ..................... 241
RESET : Reset Device........................................ 242
RET : Sequence Return ..................................... 243
RMODE : Ramp Mode...................................... 244
RUN : Run Sequence......................................... 245
RUNLV : RUN Output Level ............................ 246
S_xxx : User-Defined String Variables.............. 247
SACS : Send ASCII Control String ................... 248
SAS : Send ASCII String................................... 249
SAVEALL : Save All Data ............................... 250
SAVEPOS : Save Position Array Data............... 251
SAVEPRM : Save Parameters ........................... 252
SCHGPOS : Distance After SENSOR Input...... 253
SCHGVR : Velocity After SENSOR Input ........ 254
SENSORACT : SENSOR Input Action............. 255
SENSORLV : SENSOR Input Level ................. 256
SIGALARM
: System ALARM Output Signal ...................... 257
SIGCROFF
: System Current Off Input Signal .................... 259
SIGEND : System END Output Signal ............. 260
SIGHOME : System HOME Input Signal ......... 261
SIGHOMEP
: System Home Position Output Signal ............ 262
SIGLSN
: System Limit Switch Negative Input Signal... 263
SIGLSP
: System Limit Switch Positive Input Signal .... 264
SIGMBC
: System Magnetic Brake Control Output
Signal.............................................................. 265
SIGMOVE : System MOVE Output Signal....... 266
SIGMSTOP
: System Motor Stop Input Signal..................... 267
SIGPAUSE : System PAUSE Input Signal ........ 268
SIGPAUSECL
: System Pause Clear Input Signal .................... 269
SIGPSTOP : System Panic Stop Input Signal.... 270
SIGPSTS
: System Pause Status Output Signal ................ 271
SIGRUN : System RUN Output Signal ............. 272
SIGSENSOR
: System SENSOR Input Signal........................ 273
SIGTEMP : System TEMP Output Signal ......... 274
SLACT : Software Position Limit Control ........ 275
SSTOP : Soft Stop ............................................. 277
STARTACT : START Input Action.................... 278
STARTLV : START Input Level........................ 279
STRSW : Current State at System Start............. 280
TA : Acceleration Time...................................... 281
TALK : Select Device........................................ 282
TD : Deceleration Time ..................................... 283
TEACH : Teach Positions.................................. 284
TEMPLV : TEMP Output Level ........................ 285
TF : Motor Torque ............................................. 286
TIMER : Running Timer ................................... 287
TQFF : Torque Feedforward Control................. 288
TRACE : Sequence Trace Control..................... 289
TU : Torque Utilization...................................... 290
UNLOCK : Unlock Sequence............................ 291
UU : User Units ................................................. 292
VC : Velocity Command.................................... 293
VE : Velocity Error ............................................ 294
VER : Display Firmware Version ...................... 295
VERBOSE : Command Response Control ........ 296
VF : Motor Velocity........................................... 297
VIEW : View Parameter .................................... 298
VR : Running Velocity...................................... 299
VRx : Linked Motion Running Velocity............ 300
VS : Starting Velocity ........................................ 301
WAIT : Wait for Specified Time ........................ 302
WEND : End of WHILE Block ......................... 303

WHILE
: Begin WHILE Block: execute while true ....... 304
Chapter 7 Troubleshooting .................. 305
7.1 Protective Functions and Troubleshooting .... 305
7.2 Inspection .................................................... 307
7.3 Troubleshooting and Corrective Actions ....... 308
Appendix A Model Number ...................... 312
A.1 How to Identify the Product Model ................ 312
Appendix B Sample Programs ................. 313
B.1 Repeated Positioning Operation ................... 313
B.2 Executing Linked Operation .......................... 314
Appendix C Daisy Chain Connection Procedure ............................ 315
C.1 Setting the Unit ID's ..................................... 315
C.2 Daisy Chain Connection Procedure .............. 315
C.3 Daisy Chain Communication Example .......... 316
Appendix D Timing Charts ........................ 317
D.1 Execution of a Sequence .............................. 317
D.2 Stopping Operation ...................................... 318
D.3 CROFF Input and MBC Output .................... 321
D.4 Output .......................................................... 321
D.5 Mechanical Home Seeking ........................... 322
D.6 Teaching Operation ...................................... 323
Appendix E Command Cross Reference ............................. 324
Appendix F ASCII Data ............................. 325
Appendix G Command Format ................. 326

# Chapter 1 Before Use

This chapter explains the items you should know before using the device.

## 1.1 Introduction

This section explains the main features, system configuration and package contents.

### ■ Main Features

The αSTEP-One device is an all in one device which consists of a driver, a controller, and a stepping motor.

- No Misstepping

The motor uses a built-in rotor position sensor.
When the motor is about to misstep due to an overload, the motor control switches to closed loop and operation continues at the motor's maximum torque.

- User Unit

The αSTEP-One device allows any unit, rev, mm, degrees · · ·, and so on to be used as the base amount of travel. The unit of travel depends on the application and can be set easily. The setting of distance and velocity is easy to perform. This should be defined before starting operation. See “4.3 Initial Setting (User Unit)” on page 34 for details.

- Easy Mounting, Easy Wiring, Less Space

The αSTEP-One device has everything contained in its body, you only need to mount the motor to your system, installation is done, and it's ready to move. There is no requirement for installation space for the driver and controller.

- Easy Drive, Fastest Positioning

The αSTEP-One device is a motor having a built-in controller, motion commands go directly to the motor. Also, you do not need to calculate the motion profile. Just command the destination (moving distance) for positioning control. The αSTEP-One device automatically calculates optimum motion profile based on the torque – speed curve of the motor itself. See “4.5 Enhanced Features” on page 46.

- Low Power Dissipation, Low Heat Rise, Low Vibration

The αSTEP-One device can set the driver current to meet with the applied load. This feature reduces energy dissipation, heat rise, and vibration. See “4.5 Enhanced Features” on page 46.

- Immediate Motion Creator (IMC)

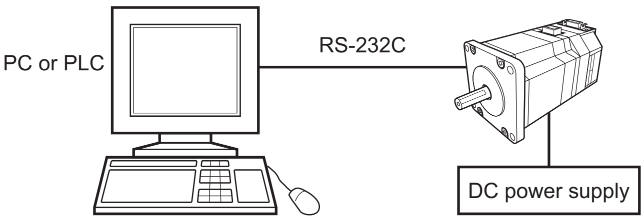
If you install the exclusive GUI tool, IMC, to your computer, just clicking your mouse can do configuration and programming. Of course, for the person who is prefer to use a keyboard and the programming language, the device can be programmed via terminal software on a PC, such as Hyper Terminal. However, IMC will greatly help you to save and load data between a PC and the αSTEP-One. IMC includes a motion creating function, sequence editing function, terminal function, data save/load function, and setting system parameters function. For details of the IMC program, see the IMC installation manual or the tutorial included in IMC.

## System Configuration

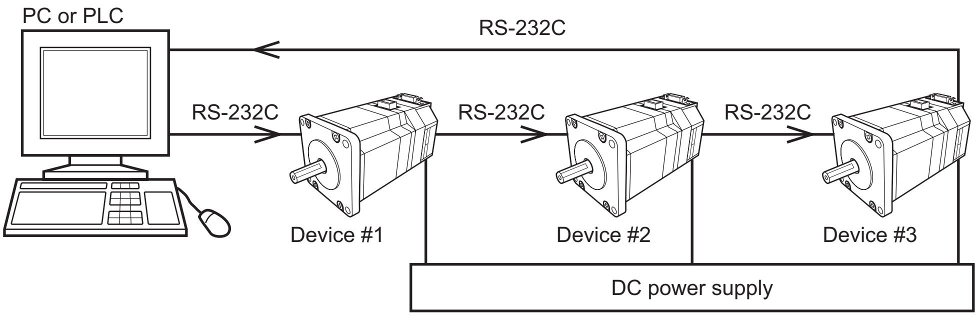
A sample system configuration using the αSTEP-One device is provided below.
A 12 to 48 VDC input power source can be used.

- Communication Mode

Connect to the computer and command the device to run.

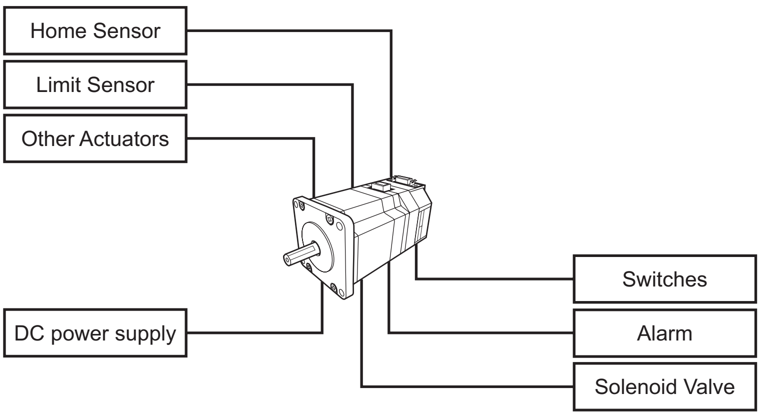



The αSTEP-One devices can be connected by RS-232C in a daisy chain style (Up to 36 axis).



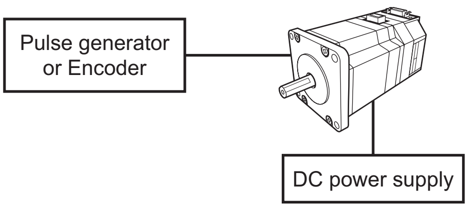

- Sequence Mode

Peripheral sensors and switches can be connected to logic I/O to start/stop programs, and control other devices.




- Pulse Input Mode

The αSTEP-One device can be controlled by external pulse signals.



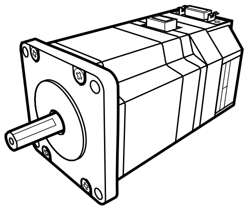
## Checking the Product

Upon opening the package, verify that the items listed below are included.
Report any missing or damaged items to the branch or sales office from which you purchased the product.

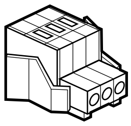
### αSTEP-One Package

Verify the model number of the purchased unit against the number shown on the package label.
Check the model number of the motor and driver against the number shown on the nameplate.

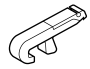
• Motor 1 piece



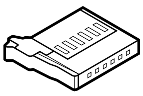
• Power connector 1 piece
734-103/037-000 (WAGO)



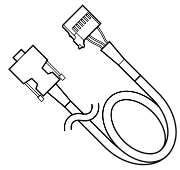
• Wire insert tool 1 piece



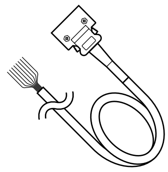
• Communication connector 1 piece
1473562-6 (Tyco Electronics AMP)



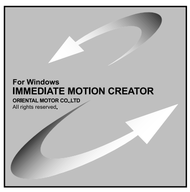
• Communication Cable 1 piece
**ASX66A-2** only



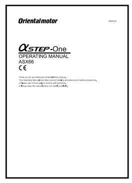
• I/O Cable 1 piece
**ASX66A-2** only



• GUI CD 1 piece
**ASX66A-2** only




• Operating manual 1 copy
**ASX66A-2** only



## 1.2 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.
- Provide a means to hold the moving parts in place for applications involving vertical travel. The motor loses holding torque when the power is shut off, allowing the moving parts to fall and possibly cause injury or damage to equipment.
- When the device’s protective function is triggered, the device will stop and lose its holding torque, possibly causing injury or damage to equipment.
- 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.
- Do not forcibly bend, pull or pinch the cables. Doing so may result in 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.
- Do not turn the CROFF (All windings off) input to “ON” while the device is operating. The device will stop and lose its holding ability, which may result in 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 it's specifications, or injury or damage to equipment may result.
- Keep your fingers and objects out of the openings in the device, or fire or injury may occur.
- Do not touch the device's heat radiating plate during operation or immediately after stopping. The surfaces are hot and may cause a skin burn (s).

Transportation

- Do not hold the device output shaft or 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).
- To prevent the risk of damage to equipment, leave nothing around the device that would obstruct ventilation.
- Provide a cover over the rotating parts (output shaft) of the device to prevent injury.

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 mean inputs to the device to "OFF." Otherwise, the device may start suddenly and cause injury or damage to equipment.
- To prevent bodily injury, do not touch the rotating parts (output shaft) of the device during operation.
- Before moving the device directly (as in the case of manual positioning), confirm that the device CROFF (Motor current off) input is "ON" to prevent injury.
- 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.

## 1.3 Precautions for use

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

### ■ Overhung Load

Always operate the device within the allowable range of overhung load.
Continuing to operate the device under an overhung load exceeding the allowable value may damage the device's bearing (ball bearing). See page 18 for details on the permissible overhung load.

### ■ Surface Temperature of the Motor Case

Be certain the device's surface temperature doesn't exceed 100°C (212°F) during use. Although the device has a protective function for overheating, the device's bearing life (ball bearing) may deteriorate depending on the operating conditions (ambient temperature, operating speed, operating duty, etc.).

### ■ Preventing Electrical Noise

See "3.3 Installing and Wiring in Compliance with EMC Directive" on page 18 for measures with regard to noise.

### ■ Changing Input Voltage

Do not change input voltage during operation. The device reads the initial input voltage to set parameters for optimal performance. Reset the device after input voltage is changed.

### ■ EEPROM Write Cycle

EEPROM is used in the device to store motion programs and parameters. Write cycles are limited to less than 100,000. The number of cycles is indicated each time a save function is used.

# Chapter 2 Quick Setup

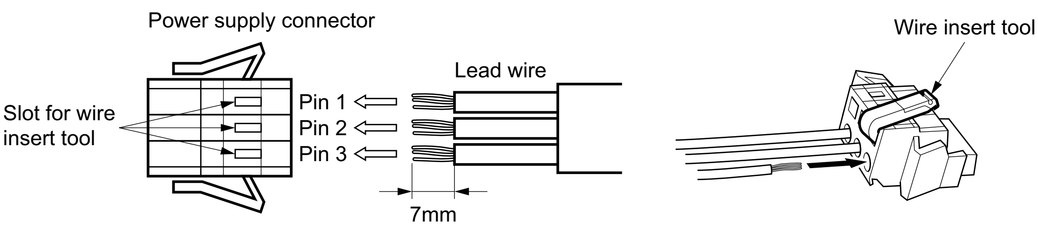
This chapter explains the methods to use the device quickly and easily.

## 2.1 Overview

2.2 Connecting the Power Supply
2.3 Connecting the Personal Computer (PC)
2.4 Immediate Motion Creator (IMC)
2.5 Making a Positioning Move

## 2.2 Connecting the Power Supply

1. Connect cables to the power supply connector.



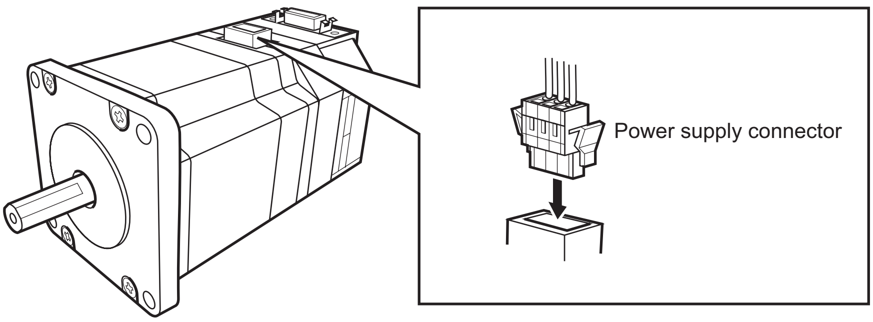

**Note** *The cable is not included in the package. Prepare a cable of a size ranging from AWG28 to 14.*

Configuration of Power-supply Connector

| Pin # | Signal (Normal Mode) | Type | Description |
|---|---|---|---|
| 1 | +VDC | Input | +12 to 48 VDC Power Supply |
| 2 | GND | Input | GND for Power supply |
| 3 | FG | | Frame GND |

**Note**
- *Confirm that each terminal is correct connected. A wrong connection may cause damage.*
- *Use a power supply that has sufficient capacity. (Maximum input current is 4.0 A.)*

2. Connecting power supply
Connect above connector to the αSTEP-One device.



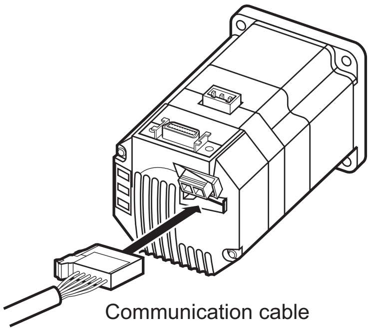
## 2.3 Connecting the Personal Computer (PC)

Connect the PC and the αSTEP-One device by using a communication cable. Use the RS-232C communications port of the PC.

**Note** *Default Communication Speed [bps]: 9600, Data bit: 8, Parity: none, STOP bit: 1, Flow control: none. See “3.7 Connecting the Device to a Personal Computer” on page 27 for details.*

Pin-out table

| Pin # | Signal Name |
|---|---|
| 1 | GND |
| 2 | TX |
| 3 | NC |
| 4 | NC |
| 5 | RX |
| 6 | GND |




**Note**
- *Communication cable is not included in the* **ASX66A-1** *package.*
- *If a sequence is executing and/or a motion is in progress and communications between the αSTEP-One and the PC are interrupted for any reason (Cable cut, program shutdown, etc.), the αSTEP-One will continue to execute the sequence and/or the motion profile. In this case, there is no way to stop the device without removing power to the device. In order to prevent this situation, it is recommended that either the INPSTOP or INMSTOP signals be assigned and connected before starting any motion or executing any sequences.*

## 2.4 Immediate Motion Creator (IMC)

The Immediate Motion Creator (IMC) is included in the **ASX66A-2** package. The IMC includes a motion creating function, sequence editing function, terminal function, data save/load function, and setting system parameters function.
The IMC is recommended for initial operation of the αSTEP-One device. See the IMC installation manual to install IMC to your PC. See the tutorial in the IMC for its use.
In this manual, the αSTEP-One device is operated with terminal software. The same operation is possible when using the IMC terminal function.

**Note** *The IMC is not included in the* **ASX66A-1** *package.*

## 2.5 Making a Positioning Move

Positioning operation is configured by its peak running velocity, distance and other parameters prior to starting motion.
The αSTEP-One device has an auto ramping function. This function does not require the ramping time to be set for acceleration and deceleration. An optimal profile for acceleration and deceleration is automatically created. As an example, this function is used in this section.

**Memo** See “4.5 Enhanced Features (Automatic Ramping)” on page 46 for details.

## ■ Examples

Conditions:

| | |
|---|---|
| Distance | 1 Rev |
| Running velocity | 1 Rev/sec. |
| Load | None |

**Note** *If the load has already been attached to the motor shaft, the load parameters must be entered before executing motion to prevent any unexpected response. See “4.5 Enhanced Features (Load Estimation)” on page 53 for details.*

1. Turn on the device.
   This assumes that the terminal program is already running.

```
-----------------------------
AS-One (ASX66)
Integrated Motor
Software Version: 1.00
Copyright 2004
ORIENTAL MOTOR CO., LTD.
-----------------------------
>
```

2. Enter “RMODE=1”.
   “RMODE” is the parameter to set the ramping mode where the acceleration rate is automatically calculated to make the move time as short as possible.

```
>RMODE=1
 RMODE=1 (Auto)
>
```

3. Enter “DIS=1”.
   “DIS” is the parameter to set distance.

```
>RMODE=1
 RMODE=1 (Auto)
>DIS=1
 DIS=1 Rev
>
```

**Note**
- *Before starting the operation, the αSTEP-One device requires the definition of the User Unit. Factory default is “Rev” and is used for this example. See “4.3 Initial Setting (User Unit)” on page 34 for setting the user unit.*

**Memo** See “4.4 Motion Types” on page 36.

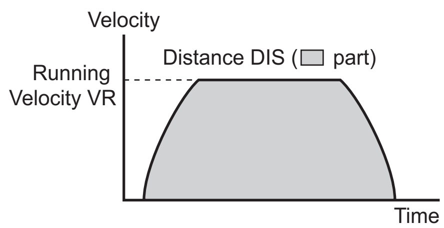
4. Enter “VR=1”.
   “VR” is the parameter to set running velocity.

```
>RMODE=1
 RMODE=1 (Auto)
>DIS=1
 DIS=1 Rev
>VR=1
 VR=1 Rev/sec.
>
```

5. Enter “MI”.
   “MI” is the command to start a positioning move (index motion).
   The device stops after the motion of 1 rotation in 1 second.
   The velocity profile should be like the picture below.

```
>RMODE=1
 RMODE=1 (Auto)
>DIS=1
 DIS=1 Rev
>VR=1
 VR=1 Rev/sec
>MI
>
```

6. Try a different distance and velocity.
   (Example) DIS=10, VR=10




**Note**
- *If you need to stop the device, enter the “ESC” key at anytime.*
- *See “4.6 Stopping Motion” on page 50 for stopping the operation.*

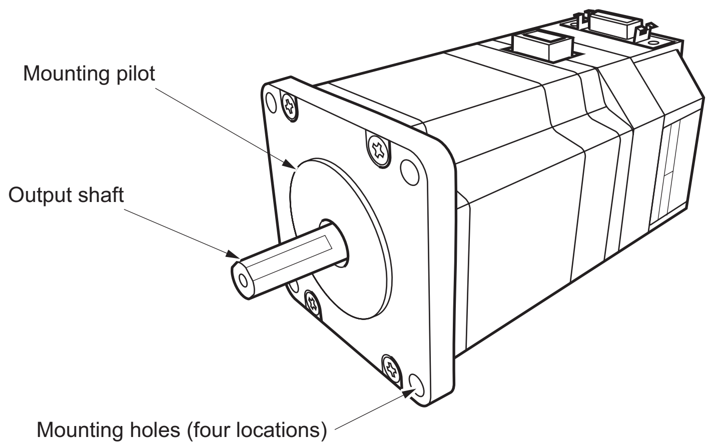
# Chapter 3 Installation and Connection

This chapter explains the names and functions of parts, the installation, and wiring in compliance with EMC Directives of the αSTEP-One device. This chapter also explains the methods of connecting sensors, external signals, and the power supply, as well as the grounding method, connection examples and control inputs/outputs.

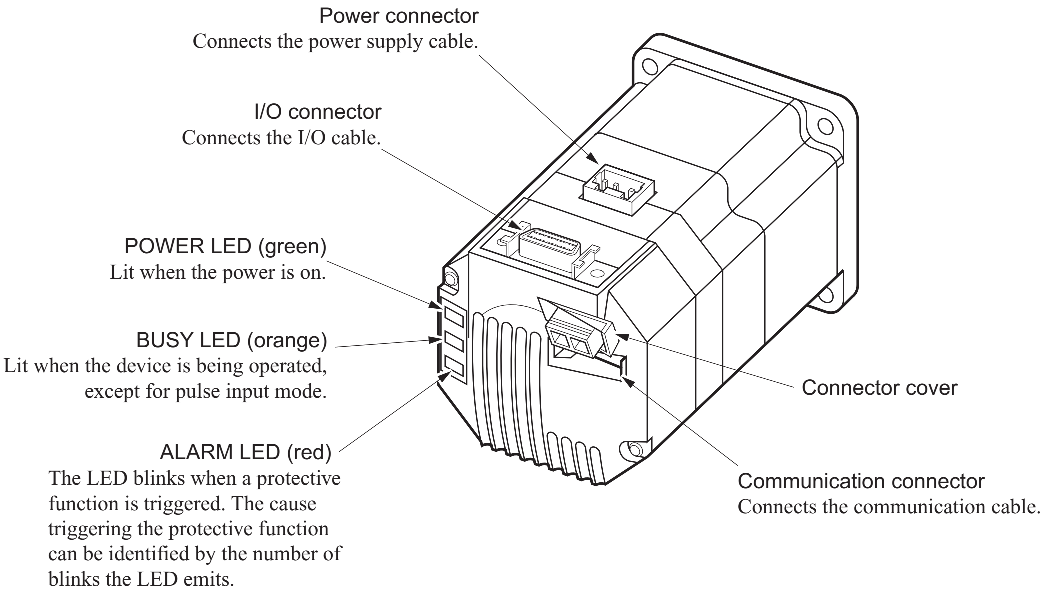
## 3.1 Names and Functions of Parts

This section covers the names and functions of the parts in the device.

### ■ Front Side of Device




### ■ Back Side of Device



## 3.2 Installation

This section covers the environment and method of installing the device, along with load installation.

### ■ Location for Installation

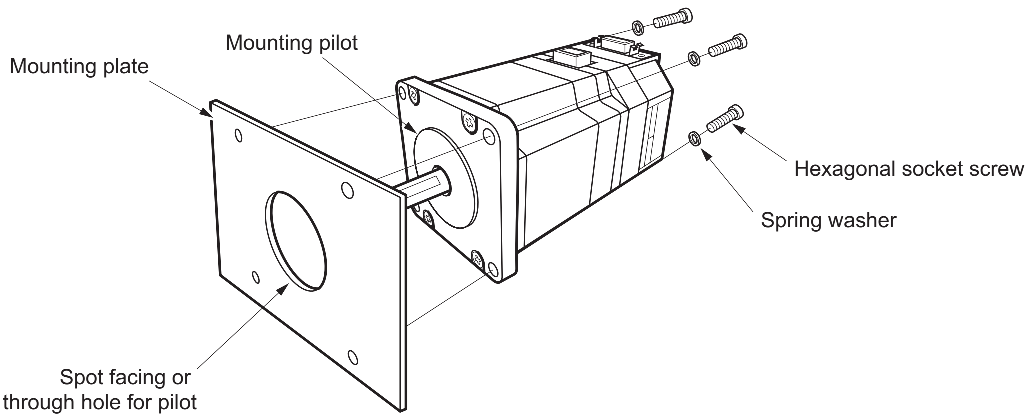
The device is designed and manufactured for installation in equipment.
Install it in a well-ventilated location that provides easy access for inspection. The location must also satisfy the following conditions:

- Inside an enclosure that is installed indoors (provide vent holes)
- Operating ambient temperature 0°C to +40°C (+32°F to +104°F) (non-freezing)
- Operating ambient humidity 85%, maximum (no condensation)
- Area that is free from an explosive nature or toxic gas (such as sulfuric gas) or liquid
- Area not exposed to direct sun
- Area free of an excessive amount of dust, iron particles or the like
- Area not subject to splashing water (storms, water droplets), oil (oil droplets) or other liquids
- Area free of excessive salt
- Area not subject to continuous vibration or excessive shocks
- Area free of excessive electromagnetic noise (from welders, power machinery, etc.)
- Area free of radioactive materials, magnetic fields or vacuum
- 1000 meters (3300 ft.) or lower above sea level

### ■ Installing the Device

- How to Install the Device

1. Insert the mounting pilot on the device's installation surface into the counter bore or through-hole provided on a metal plate.
2. Tighten four bolts (not supplied) so that no gap is left between the device and metal plate. The following conditions are recommended.
   Nominal diameter of bolt: M4
   Effective depth of bolt: 8 mm (0.35 in)
   Tightening torque: 2 N・m (280 oz-in)




- Installing a Load

When connecting a load to the device, align the centers of the device's output shaft and load shaft.

**Note**

- *When coupling the load to the device, pay attention to the centering of the shafts, belt tension, and parallelism of the pulleys. Securely tighten the coupling and pulley set screws.*
- *Be careful not to damage the output shaft or the bearings when installing a coupling or pulley to the device's output shaft.*
- *Do not modify or machine the device's output shaft. Doing so may damage the bearings and destroy the device.*

- Overhung Load and Thrust Load

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

**Note** *Be certain the overhung load and thrust load do not exceed their respective allowable values. Failure to do so may cause fatigue damage to the device's bearing (ball bearing) and output shaft.*

| Frame Size | Device Type | Permissible Overhung Load [N (lb.)] | | | | | Permissible Thrust Load [N(lb.)] |
|---|---|---|---|---|---|---|---|
| | | Distance from the Tip of Motor's Output Shaft [mm (inch)] | | | | | |
| | | 0 (0) | 5 (0.2) | 10 (0.4) | 15 (0.6) | 20 (0.8) | |
| 60 mm sq. (2.36 in sq.) | **ASX66A** | 63 (14.1) | 75 (16.8) | 95 (21) | 130 (29) | 190 (42) | 8.3 (1.8) |

# 3.3 Installing and Wiring in Compliance with EMC Directive

## ■ General

- EMC Directive (89/336/EEC, 92/31EEC)

The αSTEP-One 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.

**Memo** For the EMC Directives, see "CE Marking" on page 3.

## ■ Installing and Wiring

Effective measures must be taken against the EMI that the αSTEP-One device may give to adjacent control-system equipment, as well as the EMS of the αSTEP-One device it self, in order to prevent a serious functional impediment in the machinery.
The use of the following installation and wiring methods will enable the αSTEP-One device to be compliant with the EMC Directive (the aforementioned compliance standards).

- About Power Source

The αSTEP-One device products are of the DC power-input specification.
Use a DC power supply (such as a switching power supply) that is optimally compliant with the EMC Directive.
If a transformer is used in the power supply, be sure to connect a mains filter to the input side of the transformer.

- Connecting Mains Filter for Power Source Line

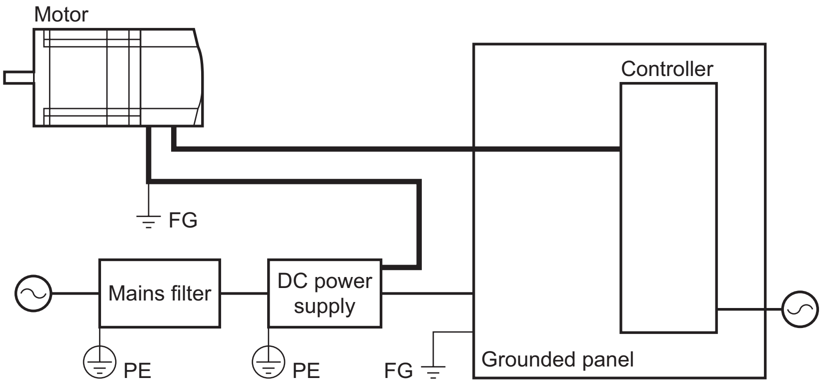
Install a mains filter on the input side of the DC power supply in order to prevent the noise generated within the driver from propagating outside via the DC power-source line.
For mains filters, use 10ESK1 (by CORCOM), ZAG2210-11S (by TDK), or an equivalent. Install the mains filter as close to the 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.

- Grounding Procedure

The cable used to ground the mains filter and FG terminal must be as thick and short to the grounding point as possible so that no potential difference is generated.

- Other Wiring

  - Grounding connections should be made directly to the grounding points so that potential differences will not occur. Connect the device 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 circuit to suppress surge generated by them.
  - Keep cable as short as possible without coiling and bundling extra length.
  - Place the 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** *Do not come close to or touch driver while the power is on.*

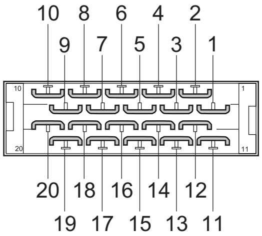
## ■ Precaution about Static Electricity

Static electricity may cause the device to malfunction or suffer damage. Be careful when handling the driver with the power on.

# 3.4 Connecting the I/O

- Connect the limit sensor signals to this connector.
- Connect the general I/O signals to this connector.

The following figure shows the pin arrangement of the connector.




Connector (20 pins)
Viewed from the soldering side.

**Note**

*Product Name*

- *Connector (Sumitomo 3M Limited)*
  *10120-6000EL (Pressure welding type)*
  *10120-3000PE (Soldering type)*
- *Housing (Sumitomo 3M Limited)*
  *10320-52F0-008*

## ■ Signal Table

- Communication Mode (Operation by sending commands to the device directly)

| Pin No. | Signal | Description | Type | Pin No. | Signal | Description | Type |
|---|---|---|---|---|---|---|---|
| 1 | START+ | Start Input | Input | 11 | IN6 | Digital Input | Input |
| 2 | START− | | | 12 | OUT1 | Digital Output | Output |
| 3 | ABORT+ | Abort Input | Input | 13 | OUT2 | Digital Output | Output |
| 4 | ABORT− | | | 14 | OUT3 | Digital Output | Output |
| 5 | EXT V+ | External power-supply terminal (5 VDC) | Input | 15 | OUT4 | Digital Output | Output |
| 6 | IN1 | Digital Input | Input | 16 | GND | GND | |
| 7 | IN2 | Digital Input | Input | 17 | GND | GND | |
| 8 | IN3 | Digital Input | Input | 18 | TX | RS-232C I/F | |
| 9 | IN4 | Digital Input | Input | 19 | RX | RS-232C I/F | |
| 10 | IN5 | Digital Input | Input | 20 | GND | RS-232C I/F | |

**Memo**

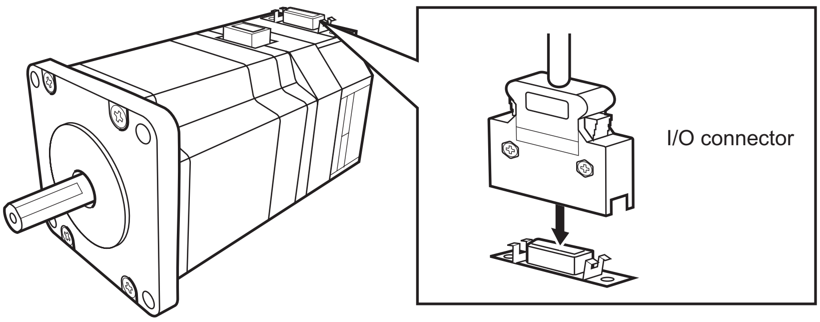
- The following signals can be assigned arbitrarily via program settings. Additionally, the output logic of each can be switched.
  OUTALARM, OUTEND, OUTRUN, OUTMOVE, OUTPSTS, OUTHOMEP, OUTTEMP, OUTMBC.
- The following signals can be assigned arbitrarily via program settings. Additionally, the input logic of each can be switched.
  INPSTOP, INMSTOP, INSENSOR, INPAUSE, INPAUSECL, INLSP, INLSN, INHOME, INCROFF, INALMCLR.

• Pulse Input Mode

| Pin No. | Signal | Description | Type | Pin No. | Signal | Description | Type |
|---|---|---|---|---|---|---|---|
| 1 | PULSE1+ | CW Pulse/Pulse/CHA+ | Input | 11 | N/C | − | − |
| 2 | PULSE1− | CW Pulse/Pulse/CHA− | Input | 12 | END | Positioning complete | Output |
| 3 | PULSE2+ | CCW Pulse/Direction/ CHB+ | Input | 13 | ALAR M | Alarm | Output |
| 4 | PULSE2− | CCW Pulse/Direction/ CHB− | Input | 14 | TEMP | Temperature Limit | Output |
| 5 | EXT V+ | External power supply terminal (5 VDC) | Input | 15 | MBC | Magnetic Brake Control | Output |
| 6 | +LS | +Limit Sensor | Input | 16 | GND | GND | |
| 7 | −LS | −Limit Sensor | Input | 17 | GND | GND | |
| 8 | PSTOP | Emergency Stop | Input | 18 | TX | RS-232C I/F | |
| 9 | CROFF | Current Off | Input | 19 | RX | RS-232C I/F | |
| 10 | ALMCLR | Alarm Clear | Input | 20 | GND | RS-232C I/F | |

## ■ Connection Method

Plug the 20-pin connector into the device's I/O connector.

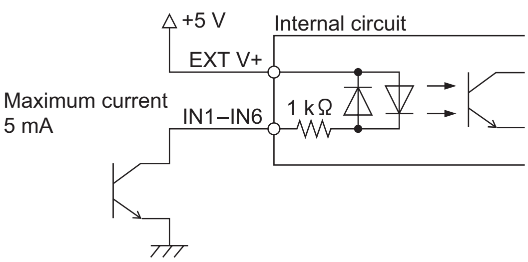


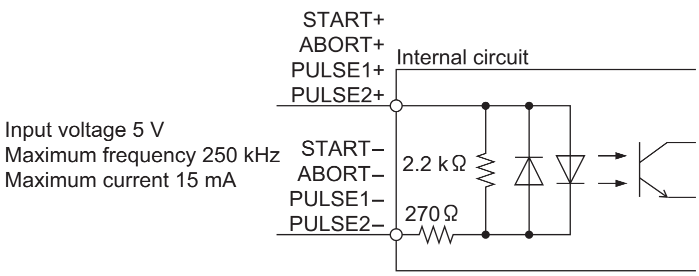

# 3.5 Input Signals

## ■ 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** *Use input signals at 5 VDC±5%.*



- EXT V+

This is a power-source input terminal for the input signals.

- GND

Use them when sharing the input signal power source among two or more drivers.

- START

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

**Note** *This signal cannot be used in pulse input mode.*

- ABORT

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

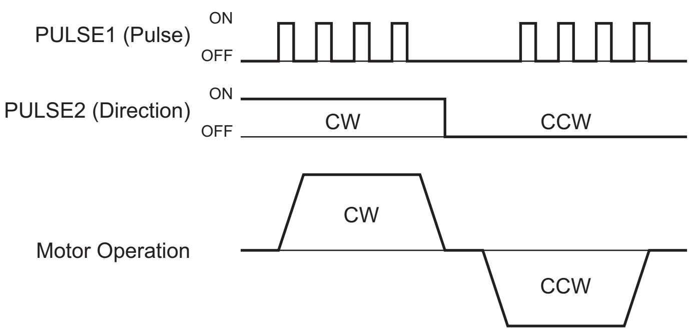
**Note** *This signal cannot be used in pulse input mode.*

- PULSE1/PULSE2

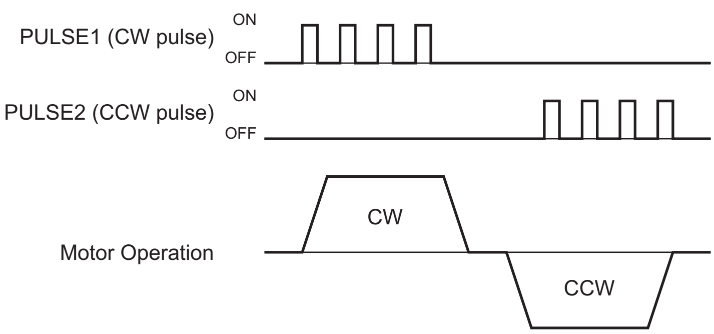
| MODE | PULSE1 | PULSE2 |
|---|---|---|
| 1: 1-pulse input mode | Pulse | Direction |
| 2: 2-pulse input mode | CW pulse | CCW pulse |
| 3: Quad-pulse input mode | Phase A | Phase B |

These signals are used to count input pulses input for pulse mode operation.
The direction of rotation can be changed using the DIRINV command.

1-pulse Input Mode




2-pulse Input Mode



- IN1−IN6 Input

The IN1 through IN6 inputs can be used as input ports for general signals.
The status of each port can be read using an IN command or INx (x=1−6) command.
The general signals assignable to the IN1 through IN6 inputs are listed below.
Panic Stop ......................... INPSTOP
Motion Stop....................... INMSTOP
Sensor................................ INSENSOR
Pause ................................. INPAUSE
Pause Clear........................ INPAUSECL
Limit Switch Positive........ INLSP
Limit Switch Negative ...... INLSN
HOME............................... INHOME
Motor Current OFF ........... INCROFF
Alarm Clear....................... INALMCLR

**Note** *In pulse input mode, the assignment of IN1−IN6 is determine by the device and cannot be changed.* *See “Pulse Input Mode” on page 21.*

**Memo** Inputs that are not configured are set to a general input.

- INPSTOP

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

- INMSTOP

This signal is used to forcibly stop motion. This command does not stop a sequence program.
Set the stopping method using the MSTOPACT command.
Additionally, the input logic can be changed using the MSTOPLV command. (The factory setting of this command is normally open.)
The leading edge of the signal will cause the action.

**Note** *This signal cannot be used in pulse input mode.*

- INSENSOR

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

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

**Note** *This signal cannot be used in pulse input mode.*

- INPAUSE

This signal is used to stop motion temporarily. If the INPAUSE input is turned ON while any motion, motion is stopped and device retains the type of on going operation (positioning, continuous, etc) and remaining distance to the original target position if paused operation is positioning.
If INSTART input is turned ON while paused situation with the sequence stay running by waiting for the end of paused motion, or if simply CONT command is executed, remaining motion will be started.
Additionally, the input logic can be changed using the PAUSELV command. (The factory setting of this command is normally open.)
Only the device operation is paused. The program execution will not stop. The leading edge of the signal will cause the action.

**Note** *This signal cannot be used in pulse input mode.*

- INPAUSECL

This signal clears the on-going operation state that has been paused by the input of a PAUSE signal.
Additionally, the input logic can be changed using the PAUSECLLV command. (The factory setting of this command is normally open.)
The leading edge of the signal will cause the action.

**Note** *This signal cannot be used in pulse input mode.*

- INLSP, INLSN

These signals are used to limit travel range.
The input logic can be changed using the OTLV command. (The factory setting of this command is normally open.)

- INHOME

This signal is used to set the mechanical home position.
Additionally, the input logic can be changed using the HOMELV command. (The factory setting of this command is normally open.)
The leading edge of the signal will start the home seeking.

**Note** *This signal cannot be used in pulse input mode.*

- INCROFF

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

**Note** *Setting CURRENT=1 while CROFF is ON is ignored, CROFF has higher priority.*
*The trailing edge of this signal will set the value of CURRENT to 1.*

- INALMCLR

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

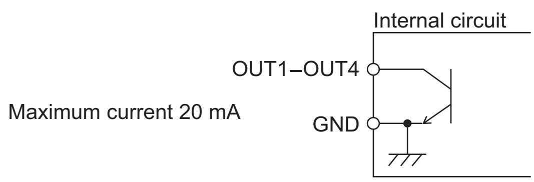
**Note**
- *After input the INALMCLR signal, please wait 200 milliseconds for the system to restore to the normal operation state.*
- *For a description of the protective functions, see "7.1 Protective Functions and Troubleshooting" on page 305.*

## 3.6 Output Signals

### ■ 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.

**Note** *Use output signals at 5−30 VDC, 20 mA max.*



- OUT1−OUT4 Output

The OUT1 through OUT4 outputs can be used as output ports for general signals.
The status of each port can be read using an OUT command or OUTx (x=1−4) command.
The general signals assignable to the OUT1 through OUT 4 outputs are listed below.
Alarm ................................ OUTALARM
End of Motion ................... OUTEND
Sequence Running............. OUTRUN
Motor Moving ................... OUTMOVE
Pause Status....................... OUTPSTS
Home Position ................... OUTHOMEP
Maximum Temperature ..... OUTTEMP
Magnetic Brake Control .... OUTMBC

**Note**
- *The assigned general output signals will be reset by any of the following operations.*
  *When turning on the AC power.*
  *When starting a program by the RUN command or the START input.*
  *When resetting a specific alarm by the ALMCLR command.*
- *In pulse input mode, the assignment of OUT1−OUT4 is determined by the device and cannot be changed. See “Pulse Input Mode” on page 21.*

**Memo** Outputs that are not configured are set as a general output.

- OUTALARM

This signal is output when an alarm is generated by the device’s protective function.
The reason for triggering of the protective function can be identified through the blink count of the alarm LED, or ALARM command.
To reset the ALM output, remove the cause of the alarm and then perform one of the following procedures after ensuring safety:
- Enter an ALMCLR command.
- Turn off the power, wait at least 10 seconds, then turn it back on.
- Input INALMCLR signal.

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

**Memo** For a description of the protective functions, see “7.1 Protective Functions and Troubleshooting” on page 305.

- OUTEND

This signal is output when the device motion is complete, and the rotor position is within ±1.8 degrees of the commanded position.
Additionally, the output logic can be changed using the ENDLV command, (the factory settling of this command is normally open).

- OUTRUN

This signal is output while sequence is running. Whereas the MOVE output turns ON only when the device is moving, the RUN output can also turns ON even if the device has been paused when MEND is used to wait motion end in the sequence.
Additionally, the output logic can be changed using the RUNLV command. (The factory setting of this command is normally open.)
[OFF: Operation completed, ON: Operation is being Executed]

**Note** *This signal cannot be used in pulse input mode.*

- OUTMOVE

This signal is output while the device is being commanded to move.
Additionally, the output logic can be changed using the MOVELV command. (The factory setting of this command is normally open.)
[OFF: No motion command, ON: Motion commanded]

- OUTPSTS

  This signal is output while the device is pausing with PAUSE input signal.
  Additionally, the output logic can be changed using the PSTSLV command. (The factory setting of this command is normally open.)
  [OFF: No PAUSEing, ON: PAUSEing]

- OUTHOMEP

  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 such as EHOME or MA 0 sets this signal to ON.
  Additionally, the output logic can be changed using the HOMEPLV command. (The factory setting of this command is normally open.)
  [OFF: Out of home position, ON: At home position]

- OUTTEMP

  This signal is output under any of followings conditions
  1) MTMP (Motor Case temperature) reaches the MTMPWRN level
  2) DTMP (Driver circuit temperature) reaches the DTMPWRN level
  Additionally, the output logic can be changed using the TEMPLV command. (The factory setting of this command is normally open.)
  [OFF: Below temperature limit, ON: At temperature Limit]

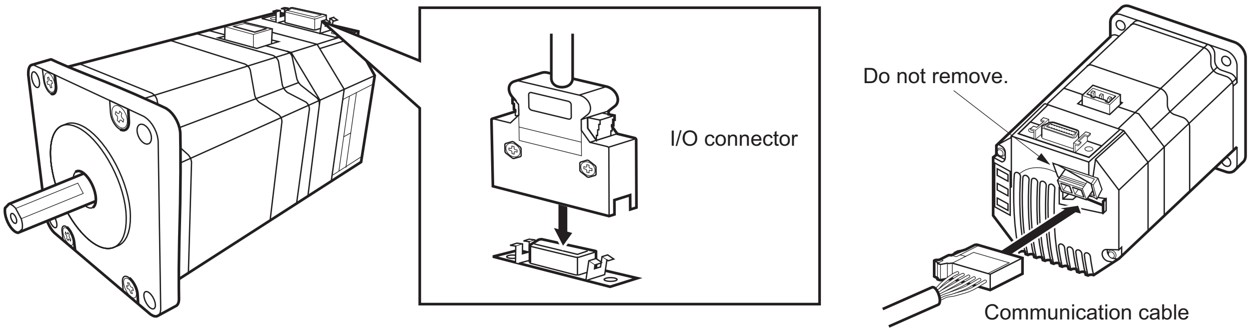
  **Note** *See page 132 and page 206 for DTMPWRN and MTMPWRN.*

- OUTMBC

  This is an electromagnetic brake control signal.
  The MBC output turns OFF when the motor loses its holding torque due to a current cutoff or alarm.
  The host controller should be set so that it detects an MBC output OFF condition and turns ON/OFF the power to the electromagnetic brake, thereby activating it.
  [OFF: Hold electromagnetic brake, ON: Release electromagnetic brake]

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

## 3.7 Connecting the Device to a Personal Computer




- Connection Method

Connect to a PC using a communication cable or I/O cable.
Plug in the optional communication cable to the communications connector of the device, or plug in the I/O cable to the I/O connector of the device.
Do not connect the PC to both the I/O connector terminals and the communication connector at the same time. Doing so may cause the device to respond incorrectly to any commands sent from the terminal.

**Note**

- *Wire the RS-232C signal lines over the shortest possible distance. The maximum distance should be 15 m (49.2 ft.). It is recommended that the signal lines be shielded to protect them from noise interference.*
- *Use this method when connecting only one device. See "Daisy Chain Connection Procedure" on page 315 when two or more devices are connected via a daisy chain.*
- *Do not pull the communication connector cover forcibly because it is attached to the motor case. Do not remove it.*
- *Communication cable and I/O cable is not included in the* **ASX66A-1**.
- *If a sequence is executing and/or a motion is in progress and communications between the αSTEP-One and the PC are interrupted for any reason (Cable cut, program shutdown, etc.), the αSTEP-One will continue to execute the sequence and/or the motion profile. In this case, there is no way to stop the device without removing power to the device. In order to prevent this situation, it is recommended that either the INPSTOP or INMSTOP signals be assigned and connected before starting any motion or executing any sequences.*

• Using the I/O connector

| PC (9-pin COM port) | I/O connector |
|---|---|
| 2 RX | 18 TX |
| 3 TX | 19 RX |
| 5 GND | 20 GND |
| 4 DTR ⎤ * | |
| 6 DSR ⎦ | |
| 7 RTS ⎤ * | |
| 8 CTS ⎦ | |

• Using the communication connector

| PC (9-pin COM port) | Communication connector |
|---|---|
| 2 RX | 2 TX |
| 3 TX | 5 RX |
| 5 GND | 6 GND |
| 4 DTR ⎤ * | |
| 6 DSR ⎦ | |
| 7 RTS ⎤ * | |
| 8 CTS ⎦ | |

**Note** ** Be sure to short pins 4 (DTR) and 6 (DSR), 7 (RTS) and 8(CTS).*

**Memo** Also, any other GND for DC terminal can be used.

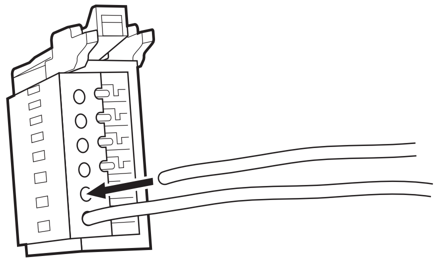
- Communication Specifications

| 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 | Selectable: 9600, 19200, 38400 bps (9600 is default.) |
| Protocol | TTY (CR+LF) |
| Connector specifications | Connector (6 lines, 6 pins) |

**Note** *As the baud rate increases, the maximum allowable length of the RS-232C connection will decrease.*

- Termination Method of Communication Connector

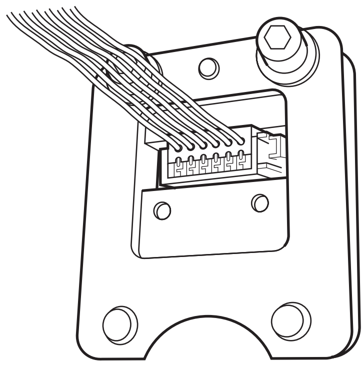
1. Insert each cable into a slot of the connector.
Insert unstripped cables all the way to the end. The outside diameter of applied cable should be 1.0–1.15 mm and the wire diameter should be 0.1–0.5 mm for the included connector.



**Note** *If you use a cable with a different size, use a different connector (Tyco Electronics AMP).*

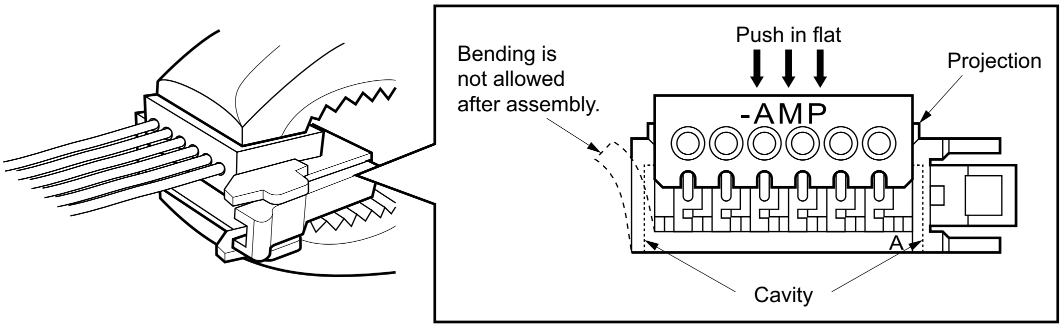
| Outside Diameter | Wire Diameter | Connector |
|---|---|---|
| 0.6–0.9 mm | 0.1–0.5 mm | 3-1473562-6 |
| 0.9–1.0 mm | 0.1–0.5 mm | 1-1473562-6 |
| 1.15–1.35 mm | 0.1–0.5 mm | 2-1473562-6 |
| 1.35–1.6 mm | 0.1–0.5 mm | 4-1473562-6 |

2. Press or clamp the connector into the housing with pliers or the dedicated tool.
Press transparent housing into black housing all the way to the bottom. The dedicated tool is strongly recommended.



**Note**

- *Use pliers wider than the transparent housing and be sure to press evenly. Uneven pressing in the process causes mis-contact.*
- *Make sure surface on both housings come together or the gap is less than 0.25 mm after termination. The four projections (locking device) on the both sides of transparent housing should mate with the cavities in the black housing. A gap between both housings or bending on the side of the black housing may be seen if the latch locking device does not completely mate.*




**Note**

*Dedicated Tool*
*Manufacturer: Tyco Electronics AMP*
*Part Number: 1596114-1*

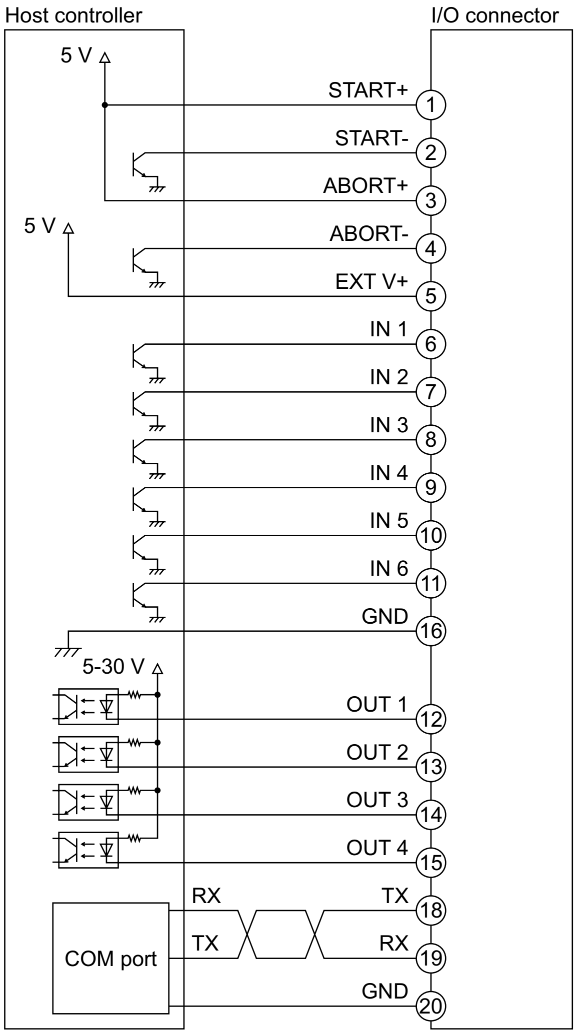
## 3.8 Connection Example

**Note**

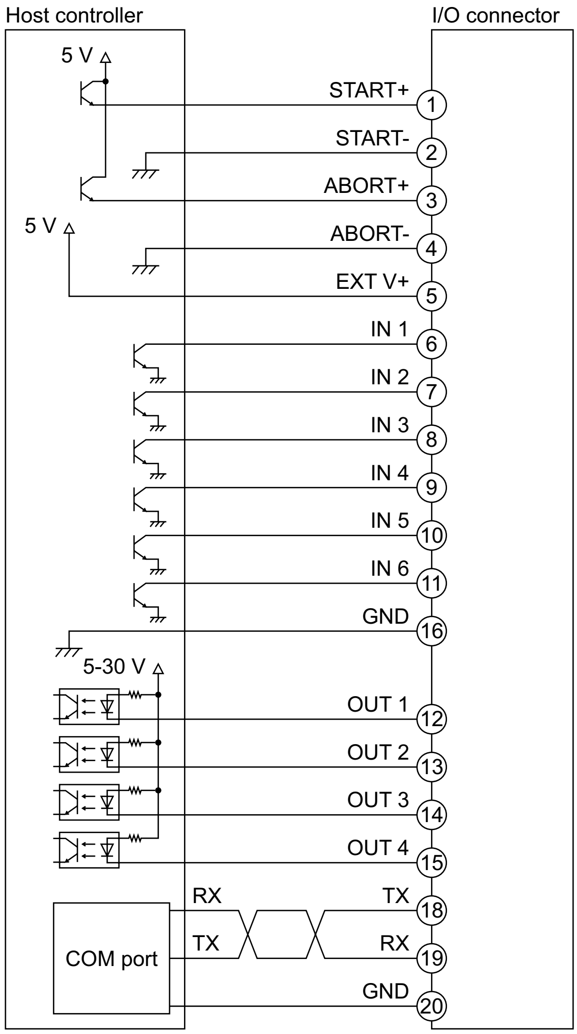
- *Use input signals at 5 VDC±5 %.*
- *Use output signals at 5−30 VDC, 20 mA max.*

### ■ Internal Profiler Mode (MODE=0)

• Example 1



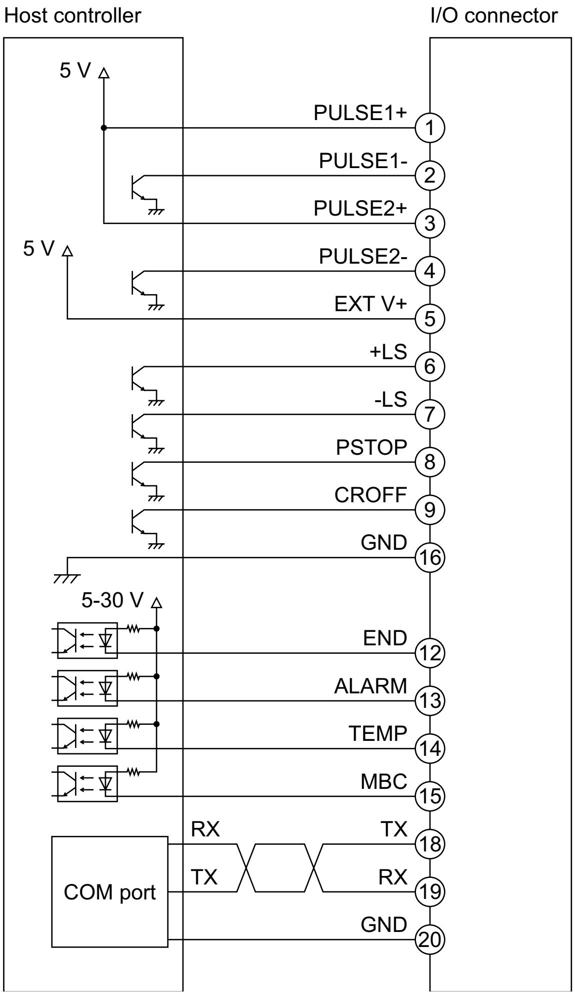

• Example 2



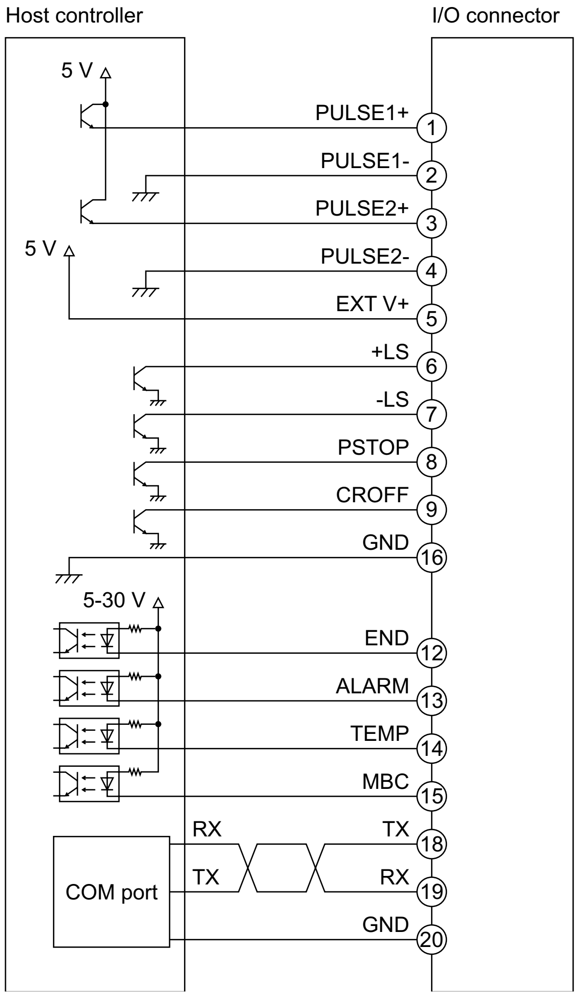

**Memo** The terminal of START and ABORT can be chosen the current sink connection or the current source connection.

## ■ Pulse Input Mode

• Example 1



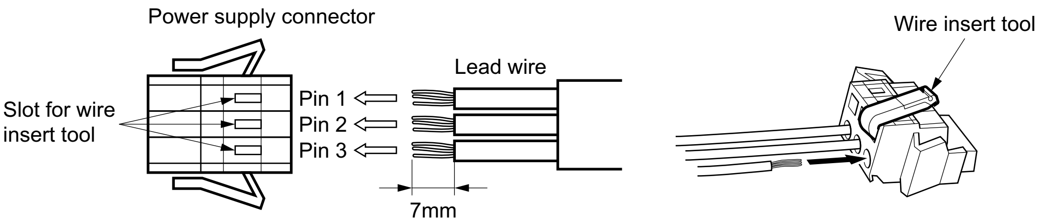

• Example 2




**Memo** The terminal of PULSE1 and PULSE2 can be chosen the current sink connection or the current source connection.

## 3.9 Connecting to the Power Supply

Connect cables to the power supply connector.




**Note** *The cable is not included in the package. Prepare a cable of a size ranging from AWG28 to AWG14.*

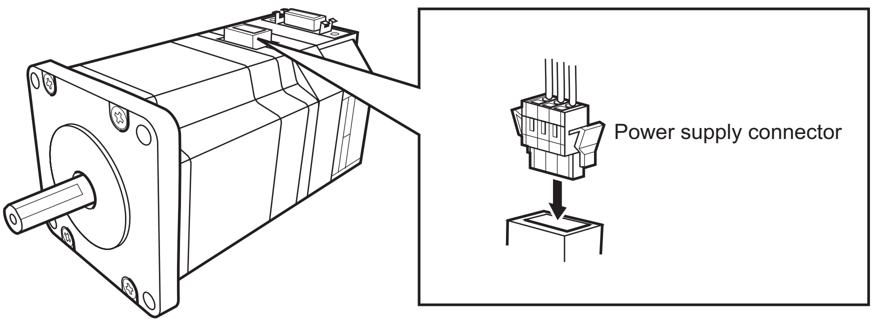
- Configuration of Power-supply Connector

| Pin No | Signal (Normal Mode) | Type | Description |
|---|---|---|---|
| 1 | +VDC | input | +12 to 48 VDC Power Supply |
| 2 | GND | input | GND for Power supply |
| 3 | FG | | Frame GND |

**Note**
- *Confirm that each terminal is correctly connected.*
- *Use a power supply that has sufficient capacity. Maximum input current is 4.0 A.*

- Connecting Power Supply

Connect the above connector to the αSTEP-One device.

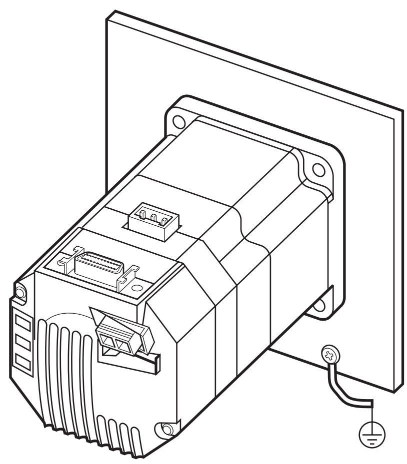



## 3.10 Grounding the Device

Properly ground the device.

### ■ How to Ground

- Connect the front side to the grounded metal plate.
- Use a cable of AWG18 (0.75 mm$^2$) or more diameter.



# Chapter 4 Features

This chapter introduces the main features of the αSTEP-One device.

## 4.1 Overview

The αSTEP-One device is designed to make motion control simple and convenient. At the same time, the system has powerful enhanced features to maximize performance, and support functions to accelerate successful system integration. The following subjects are discussed in the sections which follow:

| | | |
|---|---|---|
| 4.2 | Making the Motor Move | commanding motions, system MODE, and feature availability |
| 4.3 | Initial Setting (User Unit) | configuring the system to operate in distance units that are natural for the application |
| 4.4 | Motion Types | point-to-point, continuous, and home-seeking motions |
| 4.5 | Enhanced Features | automatic ramping, automatic current setting, torque feedforward control, and the jerk limiting filter |
| 4.6 | Stopping Motion | hard stops, soft stops, and system status after stopping |
| 4.7 | Support Functions | teaching positions, estimating load conditions, and testing I/O |
| 4.8 | Pulse Input Modes | controlling motion with externally generated pulses |
| 4.9 | Protective Functions | controlling the system response to alarm conditions |

## 4.2 Making the Motor Move

There are three ways to make the motor move, depending on the MODE parameter:

- In Internal Profiler mode (MODE 0):

  - By configuring motion parameters and sending a motion start command via the serial port
  - By executing a sequence containing motion commands. Sequences can be started via the serial port (using the RUN command), or from the I/O port (using the START input).

Most of the features described in this chapter work only in internal profiler mode. All of this chapter applies to internal profiler mode, except for Section 4.8 (Pulse Input Modes).

- In Pulse Input modes (MODEs 1 to 3):

  - By feeding the system a series of externally generated pulses, similar to a conventional stepping motor system. MODEs 1 to 3 differ only in how they interpret the pulse signals electrically.

Many of the features described in this chapter are not available in pulse input modes. Read Section 4.3 (Initial Settings), then skip ahead to Section 4.8 (Pulse Input Modes). Section 4.7 (Support Functions) contains a discussion of I/O Testing, which is also applicable.

## 4.3 Initial Setting (User Unit)

The αSTEP-One device defines all position and velocity related parameters in terms of "user units".
The number of user units per motor revolution is determined by the DPR parameter (Distance Per Revolution), and the text used for unit information 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 default values, DPR=1 and UU=Rev, assume motions are programmed in revolutions.)
Some examples appear on the following page.

**Note**

- *Changing DPR changes the physical distances and velocities of any previously programmed motions. Generally, DPR should be configured once, and not changed afterward, unless the mechanics of the application also change.*
- *If electronic gearing is used, DPR represents the distance per revolution of the output of the gearhead (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).*

- Example

```
>DPR=10
 DPR=1 (10) Rev
 OVERFLOW, OVERVEL is re-scaled to default
equivalent.
 OVERFLOW=3(30) Rev
 OVERVEL=100(1000) Rev/sec
 Position range = +/- 41943 (419430)
 Velocity range = 0.001 - 83.333 (833.33)
>
```

Maximum distance (MAXPOS)
Maximum Velocity (MAXVEL)

**Note**

*The new value (10) is shown in parenthesis after the active value. The new value will become effective only after saving (with SAVEPRM) and resetting (with RESET, or by cycling power).*

- Parameters for Initial Setting

| Parameter | Parameter Value | Function |
|---|---|---|
| DPR | 0.5 to 51200 (1) | Distance Per Revolution [user unit] {mm, deg, etc.} |
| GA | 1 to 100 (1) | Electric Gear {Numerator} |
| GB | 1 to 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<br>1: Motor rotates counterclockwise for positive distances |

( ): default value

**Memo**

Electronic gearing can 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). 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.

## ■ 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＝Rev, DPR=1
- Rotating table, with 100:1 gear (Desired unit: Degree)
  ∗Distance per motor revolution is less than 0.5.
  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

## ■ Effect on Parameter Range

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

## ■ Automatic Rescale in Parameter Limits

When DPR, GA or GB are changed, OVERFLOW (maximum position error) and OVERVEL (Maximum velocity) are automatically rescaled to values that are the physical equivalent of their factory defaults.　The maximum position error becomes the equivalent of three (3) motor revolutions, and maximum velocity becomes the equivalent of 100 motor revolutions per second.　Automatic rescaling attempts to prevent unexpected alarms caused when new values for DPR, GA or GB are combined with previous values of OVERVEL and OVERFLOW.　If OVERFLOW or OVERVEL had already been configured, they must be reconfigured.

# 4.4 Motion Types

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

## ■ Point-to-Point Motions

Point-to-point motions cause the motor to start moving from one position to another position, using a preset distance or destination. Motions start and stop at zero speed.
The motor accelerates to running velocity VR and continues to move at that velocity, as necessary, until decelerating to the final target position. If linear ramps are used (default), 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. If automatic ramping is used (RMODE=1), the acceleration and deceleration patterns are automatically determined by the system, and VS, TA and TD are ignored.
See “4.5 Enhanced Features (Automatic Ramping)” on page 46.

- Commands and Parameters for Point-to-Point Motions

| Command/ Parameter | Argument/Parameter Value | Function |
|---|---|---|
| MI | None | Start incremental motion, distance DIS |
| MA | −MAXPOS to +MAXPOS | Start absolute motion to the specified destination [user unit]. |
| DIS | −MAXPOS to +MAXPOS (0) | Distance for incremental motion [user unit] |
| VS | 0 to MAXVEL (0.1) | Starting velocity [user unit/sec.] * |
| VR | 0.001 to MAXVEL (1) | Running Velocity [user unit/sec.] |
| TA | 0.001 to 500 (0.5) | Acceleration time [sec.] * |
| TD | 0.001 to 500 (0.5) | Deceleration time [sec.] * |

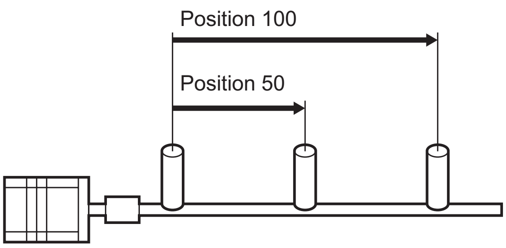
( ): default value
* With automatic ramping (RMODE=1), VS, TA and TD are not required.

**Note**
- *See “4.6 Stopping Motion” on page 50 for information on stopping motions before they finish.*
- *See the description of "Linked Motions" (below) for information on more complex motion profiles.*

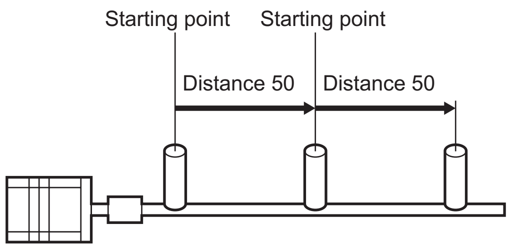
- Point-to-Point Motion Types

Two positioning modes are available for use in the positioning operation: absolute mode and incremental mode.
In absolute mode, the distance from electrical home is set.
In incremental mode, each device destination becomes the starting point for the next movement. This mode is suitable when the same distance is repeatedly used.

• Absolute Mode




• Incremental Mode

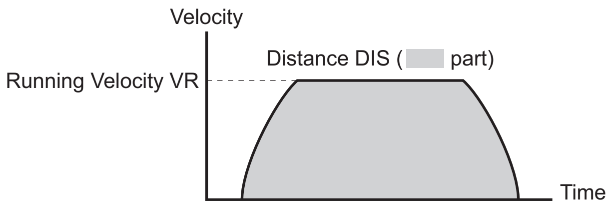


- Example

Conditions:
Ball screw: lead 10 mm (See “4.3 Initial Setting (User Unit)” on page 34.)
Distance: 60 mm (Incremental)
Running Velocity: 5 mm/sec.
Starting Velocity: 1 mm/sec. (for manual ramping)
Acceleration time: 0.5 sec. (for manual ramping)
Deceleration time: 0.5 sec. (for manual ramping)

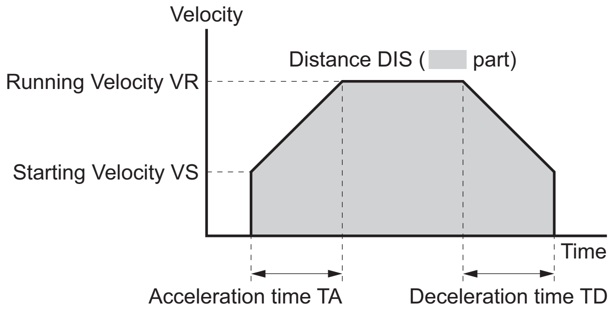
1. Automatic Ramping

```
>RMODE=1
 RMODE=1 [Auto]
>DIS=60
 DIS=60 mm
>VR=5
 VR=5 mm/sec.
>MI
>
```




2. Manual Ramping

```
>RMODE=0
 RMODE=0 [Liner]
>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
>
```




## 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 to +MAXPOS (0) | Distance or destination for link segment 'x' [user unit] |
| VRx (x=0−3) | 0.001 to 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 to MAXVEL (0.1) | Starting velocity [user unit/sec.] |
| TA | 0.001 to 500 (0.5) | Acceleration time [sec.] |
| TD | 0.001 to 500 (0.5) | Deceleration time [sec.] |

( ): default value

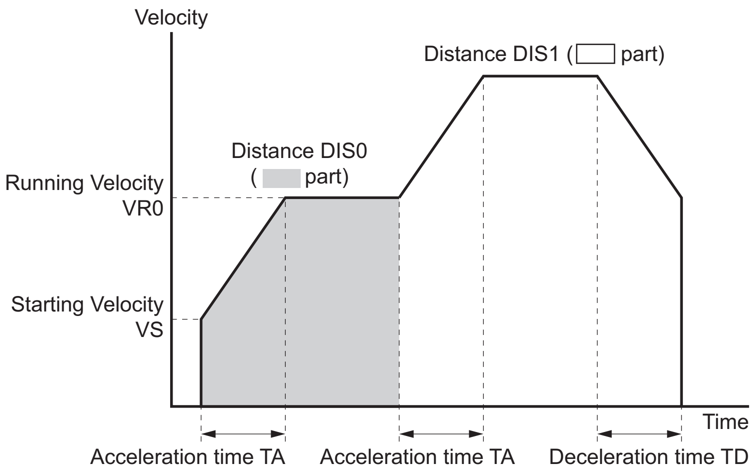
**Note**

- *See "4.6 Stopping Motion" on page 50 for information on stopping motions before they finish.*
- *Acceleration and deceleration times TA and TD are the same for each segment.*
- *Automatic ramping is not supported for linked motions.*
- *Link segments can be absolute or incremental, but all segments must execute in the same direction.*
- *Linked Motions cannot be paused and then continued: PAUSE causes a soft stop, and CONT is ignored.*

- Example

Conditions: (User Units mm)
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
>
```




## 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.
Ramping behavior depends on ramping mode RMODE. If linear ramps are used (RMODE=0), the system changes speed over a fixed time interval. If speed is increasing (away from zero), acceleration time TA is used. If speed is decreasing (toward zero), deceleration time TD is used. (If the motor is stopped when the command is executed, speed changes immediately to starting velocity VS before ramping.) If automatic ramping is used (RMODE=1), the system automatically determines ramp shape: VS, TA and TD are ignored.
Velocity can be changed by setting a new value of running velocity VR and executing a continuous motion again. Direction changes are not allowed: MCN is only permitted after a previous MCN, and MCP is only permitted after a previous MCP.
The SENSOR input can be used to change speed and eventually stop after a predetermined distance: see the example and discussion below.

**Note** *See "4.6 Stopping Motion" on page 50 for information on stopping motions.*

- Commands and Parameters for Continuous Operation

| Command/ Parameter | Argument/ Parameter Value | Function |
|---|---|---|
| MCP | None | Start moving continuously in the positive direction.<br>Change velocity |
| MCN | None | Move continuous in the negative direction.<br>Change velocity |
| VR | 0.001 to MAXVEL (1) | Running velocity [user unit/sec.] |
| VS | 0 to MAXVEL (0.1) | Starting velocity [user unit/sec.] * |
| TA | 0.001 to 500 (0.5) | Acceleration time [sec.] * |
| TD | 0.001 to 500 (0.5) | Deceleration time [sec.] * |
| SENSORACT | 0 to 2 (2) | SENSOR input action<br>0: Hard stop<br>1: Soft stop<br>2: Soft stop at fixed distance from SENSOR signal |
| SCHGPOS | 0 to MAXPOS (0) | Distance from SENSOR input to the stop position [user unit] if SENSORACT=2 |
| SCHGVR | 0.001 to MAXVEL (1) | Velocity after SENSOR input [user unit/sec.] if SENSORACT=2 |

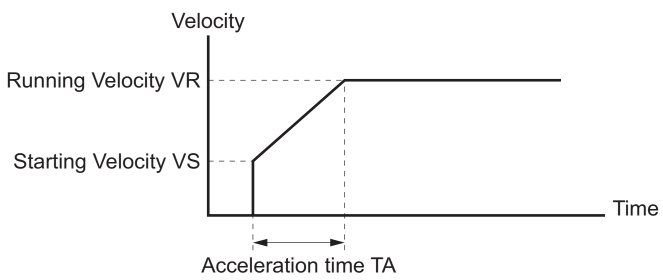
( ): default value

* With automatic ramping (RMODE=1), VS, TA and TD are not used.

- Example

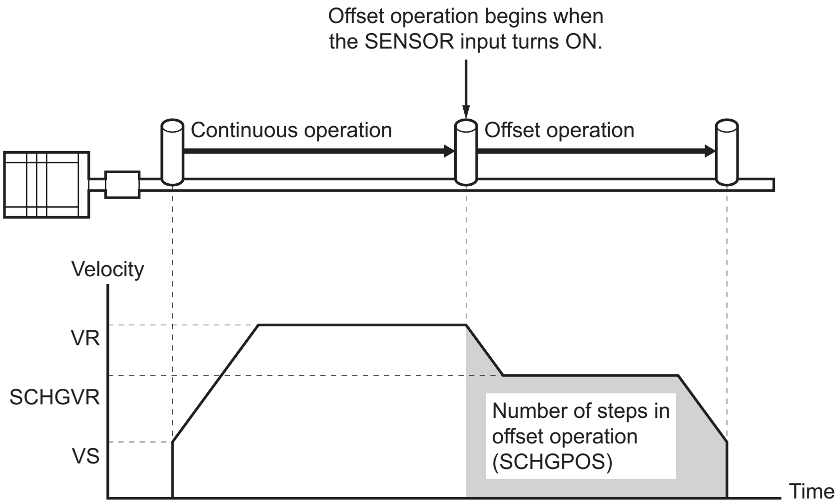
Conditions:
Ball screw: lead 10 mm (See “4.3 Initial Setting (User Unit)” on page 34.)
Running Velocity: 5 mm/sec.
Starting Velocity: 1 mm/sec.
Direction: Positive

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



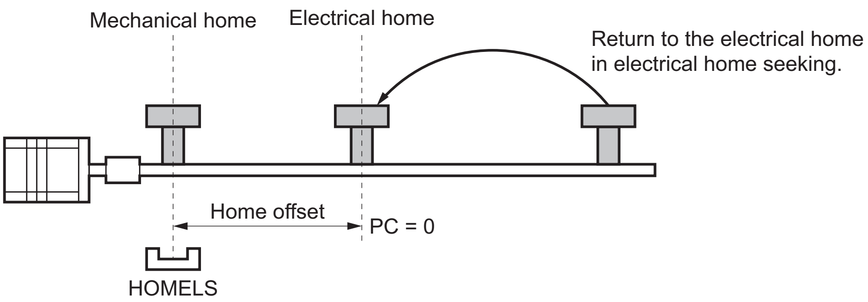
- SENSOR Action

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.
See "4.6 Stopping Motion" on page 50 for information on hard stops and soft stops. The picture below illustrates stopping action when SENSORACT=2.




## Electrical Home and Mechanical Home Seeking

When the αSTEP-One device is started or reset, the position counter (PC) is set to position zero (0). The physical position at which PC=0 is called "electrical home". The electrical home position can be aligned with an external reference signal (or signals) through a process called "mechanical home seeking", in which the system moves until a predefined home input signal pattern has been found, and then moves a predefined distance (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: position counter PC is reset to zero (0).
Position counter PC can also be set to any valid position value by direct assignment, provided the motor is not moving.



## Mechanical Home Seeking

Mechanical home seeking is an operation in which the motor moves in a specific pattern, seeking a valid mechanical home position determined by external (and possibly internal) signals. Twelve 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 43 to 45.
The SENSOR input and internal TIMING signal can be used to increase the repeatability of the final home position. The internal TIMING signal is considered ON in fifty (50) fixed, evenly spaced motor locations; each location has a width of about 0.04 (motor) degrees. If a SENSOR or internal TIMING signal are used, they are ANDed with the designated home position signal to form a valid mechanical home input signal set.

**Note** *See "4.6 Stopping Motion" for information on stopping motions before they are finished.*

- Commands and Parameters for Mechanical Home Seeking

| Command/ Parameter | Argument/Parameter Value | Function |
|---|---|---|
| MGHP | None | Start seeking mechanical home in the + direction |
| MGHN | None | Start seeking mechanical home in the − direction |
| OFFSET | −MAXPOS to +MAXPOS (0) | Offset for mechanical home seeking [user unit] |
| HOMETYP | 0 to 11 (0) | Mechanical home seeking mode: see table below |
| VS | 0 to MAXVEL (0.1) | Starting velocity [user unit/sec.] |
| VR | 0.001 to MAXVEL (1) | Running velocity [user unit/sec.] |
| TA | 0.001 to 500 (0.5) | Acceleration time [sec.] |
| TD | 0.001 to 500 (0.5) | Deceleration time [sec.] |

( ): default value

**Note** *Mechanical home seeking normally uses starting velocity VS for the final approach to the home signal(s). If VS is less than 0.001 motor revolutions/second, the final approach will be at 0.001 motor revolutions/second.*
*If the internal TIMING signal is used, and VS is more than 0.2 motor revolutions/second, the final approach will be at 0.2 motor revolutions/second.*
*The TIMING signal is based on position command (or setpoint), not on position feedback information.*

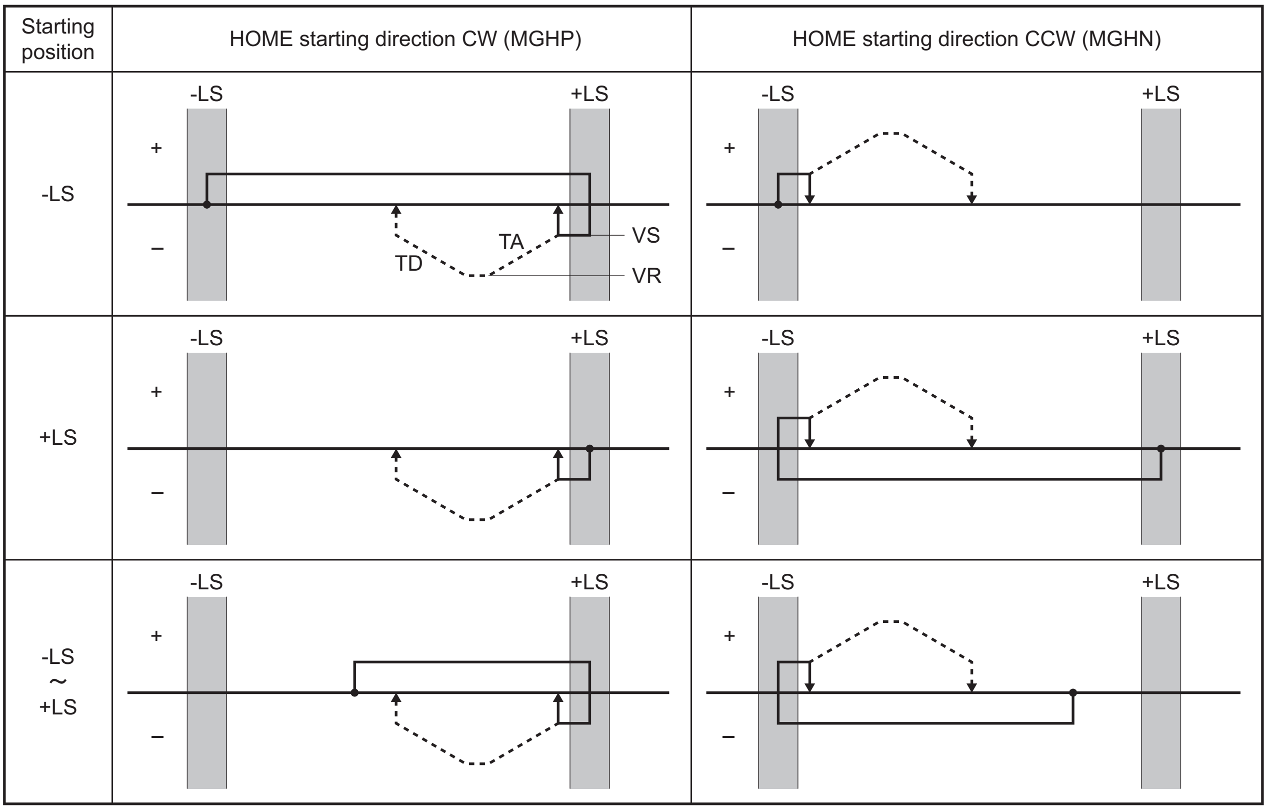
<table>
<tr><th rowspan="2">HOMETYP</th><th colspan="4">Home Position Detector</th><th rowspan="2">Motion Pattern</th></tr>
<tr><th>HOME</th><th>+LS, −LS</th><th>SENSOR</th><th>TIMING</th></tr>
<tr><td>0</td><td rowspan="4">Not used</td><td rowspan="4">Required for valid home</td><td></td><td></td><td rowspan="4">See P. 43</td></tr>
<tr><td>1</td><td></td><td>Required for valid home</td></tr>
<tr><td>2</td><td>Required for valid home</td><td></td></tr>
<tr><td>3</td><td>Required for valid home</td><td>Required for valid home</td></tr>
<tr><td>4</td><td rowspan="8">Required for valid home</td><td rowspan="4">Reverse direction</td><td></td><td></td><td rowspan="4">See P. 44</td></tr>
<tr><td>5</td><td></td><td>Required for valid home</td></tr>
<tr><td>6</td><td>Required for valid home</td><td></td></tr>
<tr><td>7</td><td>Required for valid home</td><td>Required for valid home</td></tr>
<tr><td>8</td><td rowspan="4">Stop: Alarm</td><td></td><td></td><td rowspan="4">See P. 45</td></tr>
<tr><td>9</td><td></td><td>Required for valid home</td></tr>
<tr><td>10</td><td>Required for valid home</td><td></td></tr>
<tr><td>11</td><td>Required for valid home</td><td>Required for valid home</td></tr>
</table>

- Example: Mechanical Home Seeking with HOMETYP=4

Conditions:
Ball screw, lead 10 mm (See “4.3 Initial Setting (User Unit)” on page 34.)
Running velocity: 5 mm/sec.
Starting velocity: 1 mm/sec.
Starting direction: positive
Acceleration time: 0.1 sec.
Deceleration time: 0.1 sec.

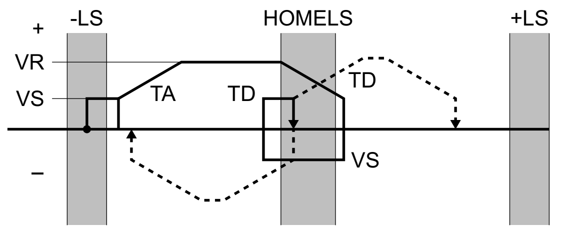
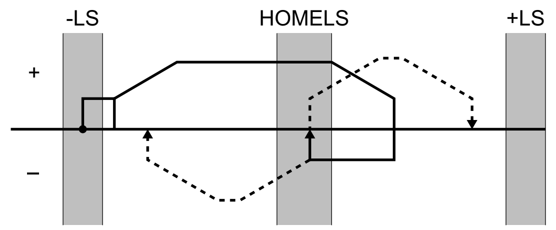
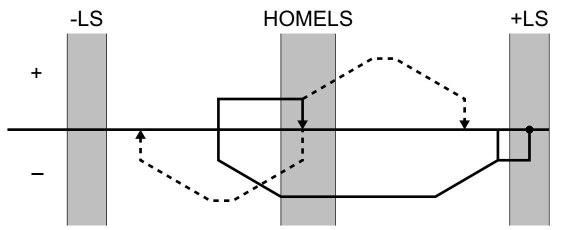
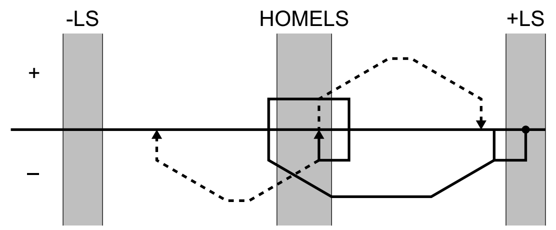
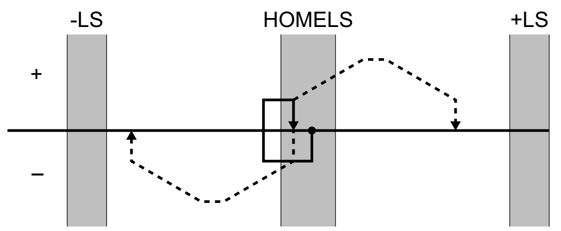
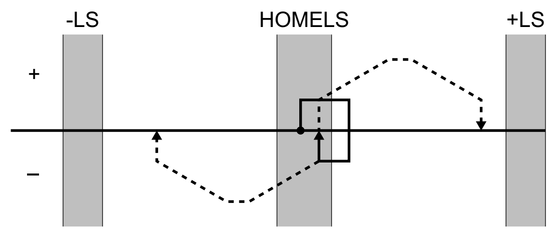
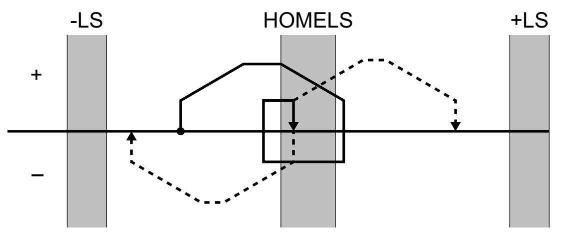
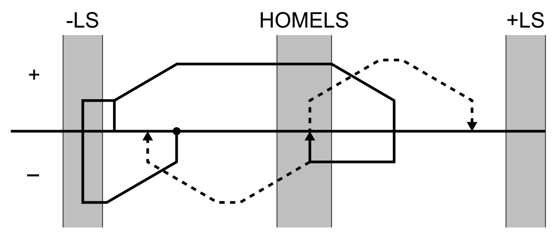
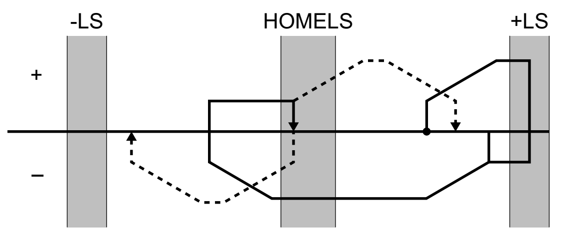
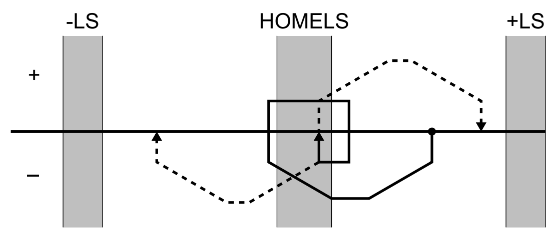
```
>HOMETYP=4
 HOMETYP=4
>VS=1
 VS=1 mm/sec
>VR=5
 VR=5 mm/sec
>TA=0.1
 TA=0.1
>TD=0.1
 TD=0.1
>MGHP
>
```

- HOME Seeking Pattern: HOMETYP 0 to 3



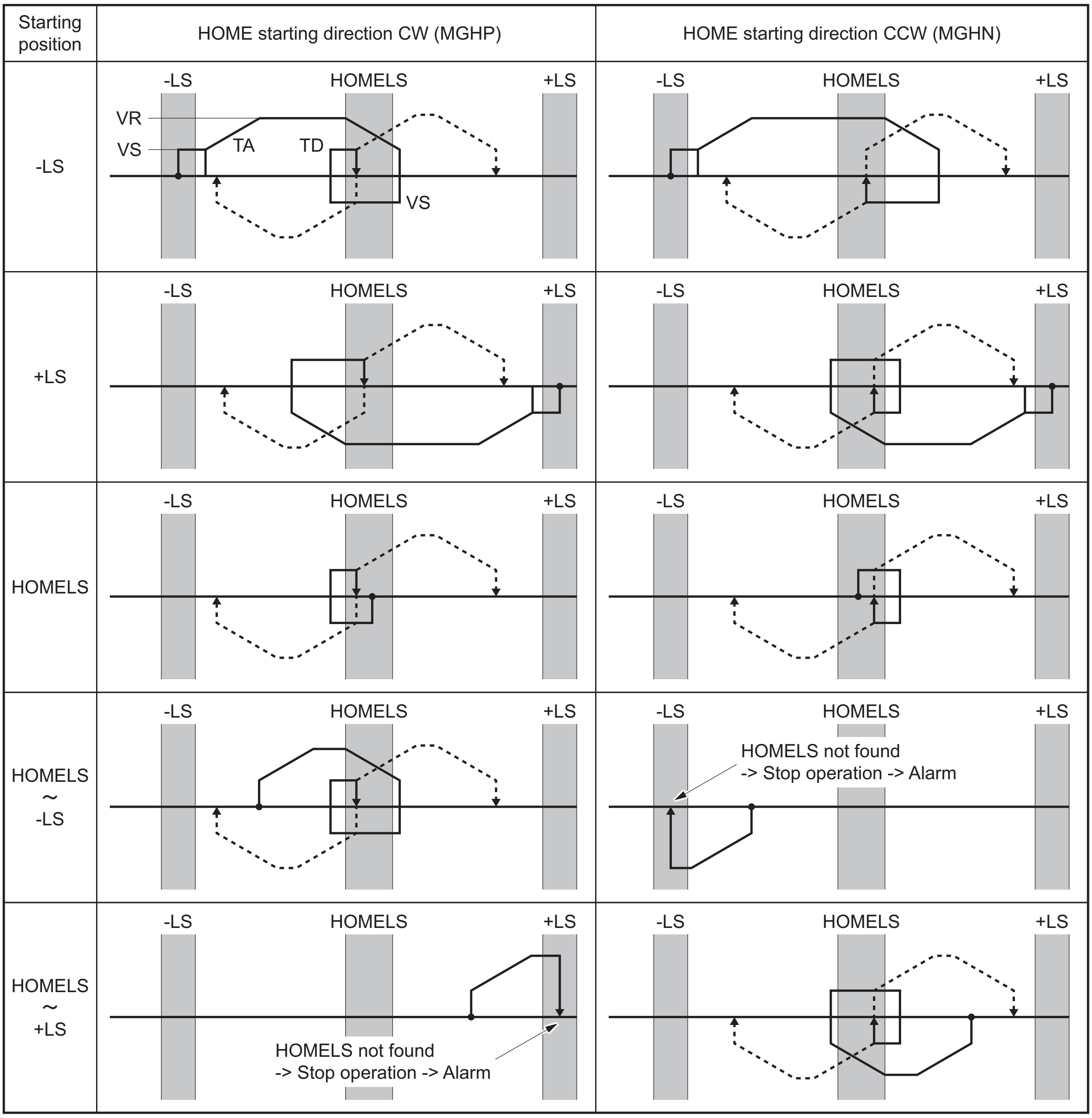

---- is operation with offset.

- HOME Seeking Pattern: HOMETYP 4 to 7

| 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

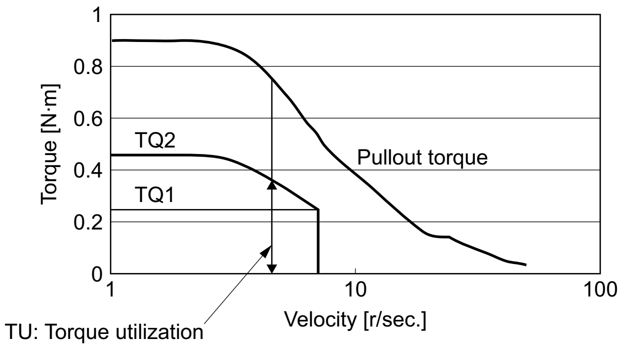


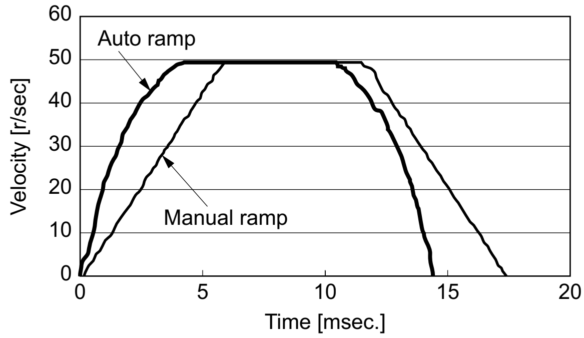

---- is operation with offset.

# 4.5 Enhanced Features

## ■ Automatic Ramping

Automatic ramping attempts to change speeds in the shortest time possible for a given torque utilization. Acceleration and deceleration profiles are automatically determined based on load parameters.
In many applications, constant torque is required for linear ramping. Because maximum motor torque is greatest near zero speed and decreases as speed increases, torque utilization is low at low speed, and higher at high speed. Automatic ramping attempts to maintain the same torque utilization (and thus torque margin) throughout a speed change, by matching acceleration rate to the αSTEP-One torque – velocity characteristics.




＊ TQ1: Torque command with linear ramping
TQ2: Torque command with automatic ramping

**Memo** Ramping mode RMODE cannot be changed while a motion is in progress.

When using linear ramps (RMODE=0), motions start and end with start velocity VS, and accelerate and decelerate over acceleration and deceleration times TA and TD. With automatic ramping (RMODE=1), VS, TA and TD are ignored.

**Note** *For proper operation of automatic ramping, the system must be configured with reasonably accurate load estimates. Estimates can be entered directly if known, or the Load Estimator utility can be used to determine approximate values. See "Load Estimation" later in this chapter for a description of the Load Estimator utility.*
*The system must also have a reasonably stable DC input supply voltage, regulated close to the programmed value for DVINSET. DVINSET is used to determine the nominal torque-velocity characteristics. The αSTEP-One device measures the actual input voltage and sets a warning if the actual value is different by more than ±10 [%] from the DVINSET value.*

- Parameters for Ramp Mode Setting

| Parameters | Parameter Value | Function |
|---|---|---|
| RMODE | 0, 1 (0) | Ramp mode.<br>0: Linear ramp mode<br>1: Automatic ramp mode |
| DVINSET | 12.000−48.000 (24) | Nominal drive input voltage [volts] |
| VR | 0.001 to MAXVEL (1) | Running velocity [user unit/sec.] |
| VS | 0 to VR (0.1) | Starting velocity [user unit/sec.] ＊ |
| TA | 0.001 to 500 (0.5) | Acceleration time [sec.] ＊ |
| TD | 0.001 to 500 (0.5) | Deceleration time [sec.] ＊ |
| LI | 0 to 12000 (0) | Load inertia [g·cm$^2$] |
| LF | 0 to 200 (0) | Load friction [N·cm] |
| LG | −200 to 200 (0) | Load gravity [N·cm] |
| TU | 0 to 100 (50) | Torque utilization [%]. Torque required for motion, as a percent of available torque. |

( ): default value
＊ With automatic ramping (RMODE=1), VS, TA and TD are not required.

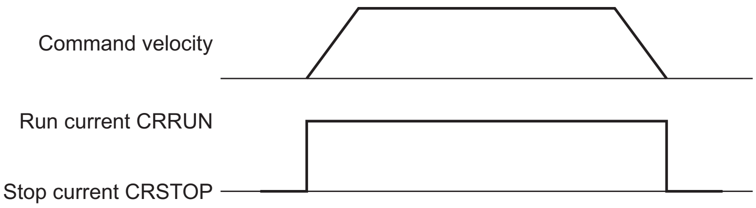
**Note** *Use a power supply that has sufficient capacity.*
*(Voltage: +12 to +48 V, Current: More than 4.0 A)*

## Automatic Current Setting

The Automatic Current Setting feature allows the system to automatically modify motor current according to motion requirements, desired torque utilization, and estimated load conditions, using knowledge of input supply voltage and torque-velocity characteristics.
Torque availability scales with motor current. Usually, torque requirements are higher when accelerating or decelerating than when stopped or running at constant speed. Allowing the system to automatically adjust current to meet anticipated torque requirements can reduce energy consumption and device temperature. It may also reduce vibration and audible noise.
Three types of current setting are available when MODE=0 (internal profiling). In Basic and Manual modes, current settings are fixed by programming parameters, and no automatic modifications are made. In automatic mode, the system automatically adjusts current. (In pulse input modes, only basic current settings are available.)

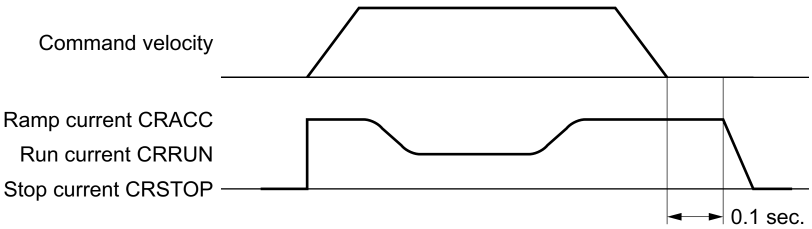
- Basic Current Setting

Motor current is defined by the stop current CRSTOP and run current CRRUN as a percent of rated current. CRSTOP controls current while the motor is stopped, and CRRUN controls current while the motor is moving.



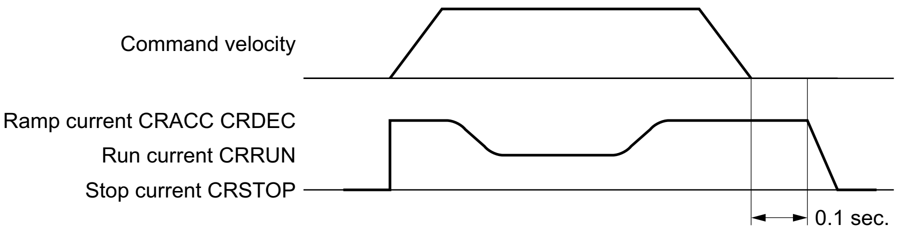

- Manual Current Setting

Motor current is defined by the stop current CRSTOP, run current CRRUN, and acceleration current CRACC, as a percent of rated current. CRSTOP controls current while the motor is stopped, CRRUN controls current while the motor is moving at constant speed, and CRACC controls current while velocity is changing.




- Automatic Current Setting

The αSTEP-One device automatically determines the appropriate value of CRRUN, CRSTOP, CRACC and deceleration current CRDEC each time a motion is started based on the load conditions, available motor torque, and programmed torque utilization TU. CRSTOP controls current while the motor is stopped, CRRUN controls current while the motor is moving at constant speed, CRACC controls current while accelerating, and CRDEC controls current while decelerating. In automatic mode, these parameters are modified by the system, and cannot be changed manually.



**Note**

*Automatic current setting:*

- *For proper operation, the system must be configured with reasonably accurate load estimates. Estimates can be entered directly if known, or the Load Estimator utility can be used to determine approximate values. See "Load Estimation" later in this chapter for a description of the Load Estimator utility.*
- *The system must also have a reasonably stable DC input supply voltage, regulated close to the programmed value for DVINSET. DVINSET is used to determine the nominal torque-velocity characteristics. The αSTEP-One device measures the actual input voltage and sets a warning if the actual value is different by more than ±10 [%] from the DVINSET value.*
- *Current requirements may fall outside parameter ranges, depending on motion and load conditions. If this happens, the relevant current parameters are set to their minimum or maximum values (whichever is closer to the calculated requirement). Warning messages are displayed (if ALMMSG=2). See “4.9 Protective Functions” on page 56 for an explanation of ALMMSG.*
- *CRSTOP is updated when LG, LSF, or TU are changed. CRRUN, CRACC, and CRDEC are updated when motions start, and are not effected by parameter changes until the next motion start.*

- Parameters for Current Setting

| Command | Parameter | Function | Parameter Access | | |
|---|---|---|---|---|---|
| | | | Basic | Manual | Auto |
| CMODE | 0 to 2 (0) | Current setting mode.<br>0: Basic<br>1: Manual<br>2: Automatic | – | – | – |
| CRRUN | 0 to 100 (100) | Run current [% of Rated Current] | R/W | R/W | R |
| CRSTOP | 0 to 100 (50) | Stop current [% of Rated Current] | R/W | R/W | R |
| CRACC | 25 to 100 (100) | • Acceleration and Deceleration Current [% of Rated Current] (Manual mode)<br>• Acceleration Current [% of Rated Current](Automatic mode) | R | R/W | R |
| CRDEC | None | Deceleration Current [% of Rated Current] (Automatic mode only) | R | R | R |
| LI | 0 to 12000 (0) | Load inertia [g·cm$^2$] | R/W | R/W | R/W |
| LF | 0 to 200 (0) | Load friction [N·cm] | R/W | R/W | R/W |
| LG | −200 to 200 (0) | Load gravity [N·cm] | R/W | R/W | R/W |
| LSF | 0 to 200 (0) | Static Load friction [N·cm] | R/W | R/W | R/W |

( ): default value
R/W: Read and Write
R: Read Only

## Torque Feedforward Control (TQFF)

Torque feedforward control attempts to anticipate torque requirements based on estimated load conditions and motion parameters. The system determines how much to advance or retard the motor phase current angle to meet the anticipated torque requirements. If load estimates are reasonable, torque feedforward control may reduce position error and improve performance.

- Parameters for Feedforward Control

| Parameter | Parameter Value | Function |
|---|---|---|
| TQFF | 0 to1 (0) | Set torque feedforward control mode.<br>0: Disabled<br>1: Enabled |
| LI | 0 to 12000 (0) | Load inertia [g·cm$^2$] |
| LF | 0 to 200 (0) | Load friction [N·cm] |
| LG | −200 to 200 (0) | Load gravity [N·cm] |
| LSF | 0 to 200 (0) | Static Load friction [N·cm] |

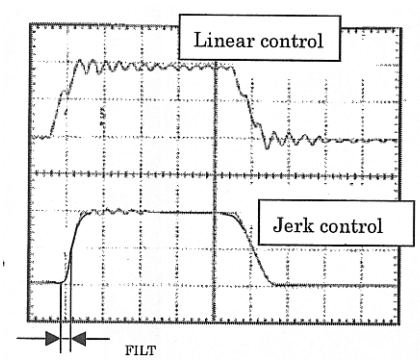
( ): default value

**Note**

- *For proper operation, torque feedforward control requires reasonably accurate load estimates. Estimates can be entered directly if known, or the Load Estimator utility can be used to determine approximate values. See "Load Estimation" later in this chapter for a description of the Load Estimator utility.*
- *Torque feedforward control assumes that the load consists of inertial, frictional and gravity (or other constant torque) components only, and that load parameters are constant. It may not improve performance, and can degrade performance, if these assumptions are not true. Be sure to test the application carefully.*
- *Torque feedforward control can intentionally introduce sudden changes in motor torque as motions are executed. These changes can excite undesirable resonances in the motor and load. The jerk limiting filter can be used to reduce this effect by smoothing torque transitions. (See the description of the jerk limiting filter in the next section.)*

## Jerk Limiting Filter

The jerk limiting filter suppresses jerk (the derivative of acceleration) by smoothing motion profiles and preventing sudden changes in acceleration (or deceleration). The filter may help reduce vibration and audible noise in some applications.
The filter introduces "time lag" (or delay) in the motion profile. Filter performance is controlled by adjusting the filter time lag, FILT. Increasing FILT makes motions smoother, but also introduces more delay.
Changing FILT while a motion is in progress is not allowed.



- Parameters for Jerk Limiting Filter

| Parameter | Parameter Value | Function |
|---|---|---|
| FILT | 0 to 1 (0.003) | Jerk limiting filter lag time [sec.] |

( ): default value

**Note**

- *Setting FILT to 0 disables the filter completely.*
- *With linear ramping (RMODE=0), the system transitions suddenly between zero speed and start speed VS at the beginning and end of motions. The jerk limiting filter smoothes these jumps: VS and the jerk limiting filter work against each other. For the smoothest motion, set VS to zero (0) or a very low value.*

## 4.6 Stopping Motion

Commands and parameters for stopping motion are shown below.

- Commands and Parameters for Stopping Motion

| Command/ Parameter | Argument/ Parameter Value | Function |
|---|---|---|
| SSTOP | None | Soft stop: controlled deceleration over time |
| HSTOP | None | Hard stop: stop as quickly as possible |
| MSTOP | None | Motor stop: Hard or soft stop, determined by MSTOPACT* |
| PSTOP | None | Panic stop: Hard stop, system state after stop is defined by ALMACT* |
| ABORT | None | Soft stop, abort sequence execution* |
| PAUSE | None | Pause motion (soft stop) * |
| MSTOPACT | 0 to 1 (1) | Motor stop action (affects MSTOP command or MSTOP input)<br>0: Hard stop (MSTOP behaves as HSTOP)<br>1: Soft stop (MSTOP behaves as SSTOP) |
| ALMACT | 0 to 2 (2) | Alarm action (affects PSTOP command or PSTOP input)<br>0: No alarm triggered<br>1: Trigger alarm, motor current remains ON<br>2: Trigger alarm, motor current disabled |

( ): default value

* MSTOP, PAUSE, and PSTOP can be assigned to an input port. ABORT has a dedicated input port.

**Note**

- *The ESC key or character stops motion and aborts sequences, similar to ABORT.*
- *A motion that has been stopped with the PAUSE command can be continued (resumed) using the CONT command. See the entries for CONT and PAUSE in Chapter 6.*
- *See "3.5 Input Signals" on page 21 for details on assigning signals to I/O ports.*

# 4.7 Support Functions

## ■ Teaching Positions

The αSTEP-One device 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.
```

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

- the system constantly monitors and displays system position command, in user units
- motor current can be toggled on and off
- if motor current is on, the system can be commanded to move, positively or negatively, in increments of 0.001 user units, or continuously at running velocity VR
- if motor current is off, the system can be repositioned using external force or torque
- at any time, the system position can be stored to any location in the position data array.

**Note**

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

- Key functions, while TEACHing

| Key | Function |
|---|---|
| V | Move continuously, in the negative direction (while key pressed).Soft stop when key released. |
| B | Move negatively in increments of 0.001 user units. |
| N | Move positively in increments of 0.001 user units. |
| M | Move continuously, in the positive direction (while key pressed).Soft stop when key released. |
| Q | Toggle current OFF and ON |
| K | Set key interval detection time [millisecond] |
| S | Save position array data to EEPROM (same as SAVEPOS) |
| <SPACE> | Hard stop |
| <ESC> | Soft stop, terminate TEACHing |
| <Enter> | Store current position to a location in the position data array(System will prompt for location, 1-100) |

**Memo**

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

• Example

After TEACH command

```
        ***Teach mode***

(V)     : Move Cont. Neg.    (M): Move Cont. Pos.
(B)     : Move Incr. -0.001  (N): Move Incr. +0.001
(Q)     :Current ON/OFF      (S):Save all data to EEPROM
(K)     :Change Key Interval (50-500 [msec])
<Space> : Immediate stop
<Enter> :Data entry mode (Input POS number, then <Enter>)
<ESC>   :Exit teach mode

PC=     0.000
```

Current position

After <ENTER> received while teaching

```
        ***Teach mode***

(V)     : Move Cont. Neg.    (M): Move Cont. Pos.
(B)     : Move Incr. -0.001  (N): Move Incr. +0.001
(Q)     :Current ON/OFF         (S):Save all data to
EEPROM
(K)     :Change Key Interval (50-500 [msec])
<Space> : Immediate stop
<Enter> :Data entry mode (Input POS number, then <Enter>)
<ESC>   :Exit teach mode

PC=     15.410  Save to POS [1]
```

Set target position data array location: Input “1”

After <ENTER> received

```
        ***Teach mode***

(V)     : Move Cont. Neg.    (M): Move Cont. Pos.
(B)     : Move Incr. -0.001  (N): Move Incr. +0.001
(Q)     :Current ON/OFF         (S):Save all data to
EEPROM
(K)     :Change Key Interval (50-500 [msec])
<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 <ESC> received

```
        ***Teach mode***

(V)     : Move Cont. Neg.    (M): Move Cont. Pos.
(B)     : Move Incr. -0.001  (N): Move Incr. +0.001
(Q)     :Current ON/OFF         (S):Save all data to
EEPROM
(K)     :Change Key Interval (50-500 [msec])
<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.
End teach mode
>
```

<ESC>

## Load Estimation

Automatic ramping, automatic current setting, and torque feedforward control require reasonably accurate estimates of load parameters to operate properly. Values for these parameters may be entered directly, if known, or the system can estimate them using the load estimation utility, LDCHK.
LDCHK causes the motor to move, back and forth, in a repeating pattern. The system continuously monitors position error and internally converts position error to shaft torque. The system calculates load estimates by comparing shaft torque to velocity, acceleration, and motion direction. If position error is too small to be meaningful, the system may increase velocity or decrease motor current to obtain meaningful data.
The motion pattern starts in the direction of positive motion, and moves no more than 1.1 motor revolutions before returning to its starting position. Speed may increase to 5 motor revolutions per second. The motion pattern terminates when sufficient data has been accumulated (usually in less than a minute). The final estimates are displayed, and may be accepted (by <ENTER>) or rejected (by <ESC>).
If an Escape (<ESC>) is received during the process, the process terminates after returning to the starting position.

**Note** *The motor starts moving as soon as the LDCHK command is executed. Be sure that the system can move safely in the positive direction, at least 1.1 motor revolutions, and reverse direction and return to the starting point.*

- Estimated Load Parameters

| | Inertia | Coulomb Friction | Gravity (or other constant) Load | Static Friction |
|---|---|---|---|---|
| Parameter | LI [$g \cdot cm^2$] | LF [N·cm] | LG [N·cm] | LSF [N·cm] |

- Example

```
>LDCHK
 Start Load Parameter Estimation

 Press <ESC> to exit.

 LI :   LF   :   LG   :   LSF   : Status
 986    8        0        3    Complete

 Press <Enter> to proceed, <ESC> to cancel.

 Load parameters are set.

 LI  =    986 [gcm^2]
 LF  =     8  [Ncm]
 LG  =     0  [Ncm]
 LSF =     3  [Ncm]

 Exit load estimation mode.
>
```

Status:
Measuring: Still estimating
Complete: Finished estimating*

Displayed only when <Enter> is entered

Displayed when estimation terminates

* If one or more calculated estimates are out of range, they will be clipped to the appropriate limit, and a message will appear here.

**Note**

- *The load parameters are not stored to EEPROM automatically; they must be saved using the SAVEPRM command.*
- *Load estimation is not available while a sequence is being executed, or motion is in progress. While estimating, sequences may not be executed and only the PSTOP, +LS, −LS, and CROFF inputs are acknowledged, if they are configured as inputs.*
- *In some applications (e.g. very low loads or low stiffness), the load estimator may not provide accurate results. When using automatic ramping (RMODE=1), automatic current setting (CMODE=2) or torque feedforward control (TQFF=1), check system performance carefully after accepting the load estimates.*

## ■ I/O Test

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

| | | | | | |
|---|---|---|---|---|---|
| OUT1 (1) | OUT2 (2) | OUT3 (3) | OUT4 (4) | MOVE (M) | RUN (R) |
| END (E) | HOMEP (H) | ALARM (A) | PSTS (P) | TEMP (T) | MBC (B) |

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

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

- Example

```
        *** Input Monitor -- Output Simulator ***
Inputs  (1-6) = IN1   IN2   -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  -(SEQ#)  - START   ABORT   -  1 2 3 4
0 0 0 0 0 0  - (   0)  -  0       0     -  0 0 1 0
```

# 4.8 Pulse Input Modes

In pulse input modes, system position is controlled by a pair of dedicated external inputs. Working together, these inputs provide a net pulse count, which the system interprets as a position command and attempts to follow.
MODE determines how the system generates motions. By default, the system uses its internal profiler to generate motion commands (MODE=0). There are also three types of pulse input modes:

| MODE | Motion determined by: | Position Command Signal Pair |
|---|---|---|
| 0 | Internal Profiler | − |
| 1 | Pulse Input (1 pulse) | Pulse, Direction |
| 2 | Pulse Input (2 pulse) | Positive Pulse, Negative Pulse |
| 3 | Pulse Input (Quadrature pulse) | Channel A, Channel B |

To configure the system for a pulse input mode, set MODE to the appropriate value, save parameters (with the SAVEPRM command), and either reset (with RESET) or cycle system power. The system will start in the selected mode.
Many features, commands, and parameters which can be used in MODE 0 (Internal Profiler) are not meaningful and are unavailable in pulse input modes.

**Note**

- *In 2-pulse mode, if the positive pulse signal is active, negative pulses will not be counted. If the negative pulse signal is active, positive pulses will not be counted. Avoid having both signals active at the same time.*
- *See “3.4 Connecting the I/O” for pin assignment.*
- *In pulse input modes, current is set to stop current CRSTOP when no pulses have been detected for 0.1 second, and set to run current CRRUN when the pulse count is changing.*
- *If audible noise or vibration is noticeable at constant velocity, try increasing the jerk limiting filter time lag FILT gradually until performance becomes acceptable. Increasing FILT makes motions smoother, but also increases system response time.*

## ■ Resolution in Pulse Input Modes

In pulse input modes (MODE=1 to 3), an additional parameter is required to determine positions from pulse counts. DPP (Distance Per Pulse) is the number of user units to travel per pulse input received. DPP is used with DPR (Distance Per Revolution), and optional electronic gear factors GA and GB, to determine the physical motion traveled per pulse received. (See "4.3 Initial Setting (User Unit)" earlier in this chapter for a detailed description of DPR, user units, etc.)

- Parameters for Initial Settings (Pulse Input Modes)

| Parameter | Range | Function |
|---|---|---|
| DPR | 0.5 to 51200 (1) | Distance Per Rotation [user unit] (mm, deg, etc.) |
| DPP | 0.001 to DPR/500(0.001) | Distance Per Pulse [user unit] (mm, deg, etc.) |
| GA | 0 to 100 (1) | Electronic Gear {Numerator} |
| GB | 0 to 100 (1) | Electronic 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<br>1: Motor rotates counterclockwise for positive distances |

( ): default value

**Note**

*DPP becomes the position setpoint resolution, expressed in user units per pulse.
To express position setpoint resolution in terms of pulses per revolution, divide DPR by DPP. The result is not required to be an integer. Note that "revolution" refers to the revolution of the output of a (physical or virtual) gearhead, if electronic gearing is used.*

- Example

DPR = 1 user units/revolution, DPP = 0.001 user units/pulse → 1000 pulses/revolution
DPR = 1 user units/revolution, DPP = 0.002 user units/pulse → 500 pulses/revolution
DPR = 1 user units/revolution, DPP = 0.003 user units/pulse → 333.333… pulses/revolution
DPR = 2 user units/revolution, DPP = 0.001 user units/pulse → 2000 pulses/revolution

If finer resolution is needed, try reducing DPP first. If DPP is already set to its minimum (0.001), try increasing DPR. Changing DPR changes the base unit of distance for the system, so User Units (UU) may need to be modified, as well.

- Example

DPR = 10 user units/revolution, DPP = 0.001 user units/pulse → 10000 pulses/revolution
DPR = 100 user units/revolution, DPP = 0.001 user units/pulse → 100000 pulses/revolution

## 4.9 Protective Functions

The αSTEP-One device constantly monitors system conditions to detect potentially harmful conditions, such as overheating, over voltage, and over loading. 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 |
|---|---|---|
| Over position error | Position error exceeds programmed limit OVERFLOW | 0x10 |
| Over load | Maximum permitted torque applied, duration exceeds programmed limit OLTIME | 0x30 |
| Over velocity | Velocity exceeded programmed limit OVERVEL | 0x31 |
| Hardware over travel | Positive or negative position limit signal detected | 0x66 |
| Software over travel | Position outside of programmed positive and negative position limits LIMP and LIMN | 0x67 |
| Panic stop | System executed a panic stop because of a PSTOP input or command | 0x68 |

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

| ALMACT | Action |
|---|---|
| 0 | Continue operation. Alarm OFF. |
| 1 | Abort sequences and stop motion. Motor current ON, Alarm ON. |
| 2 | Abort sequences and stop motion. Motor current OFF, Alarm ON, default) |

**Note** *See “5.6 Error Messages Displayed on the Terminal” on page 67 for details of each alarm condition and system response.*

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

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

**Note**
- *See “5.6 Error Messages Displayed on the Terminal” on page 67 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 = 1 ,  RECORD: 1 2 3 4 5 6 0 0 0 0

 WRN_OVER_LOAD , 5.981 [sec] past.
>
```

**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.*

# Chapter 5 Program Creation and Execution

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

## 5.1 Overview of Operation

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

### ■ Operation from Terminal (Monitor Mode)

Operation from the terminal is available when the device's power is input.
When operating from the terminal, you can create, delete, copy, lock and execute sequences. Additionally, motion can be started, stopped and the status of the device and I/O signals can be monitored.

### ■ Sequence Editing

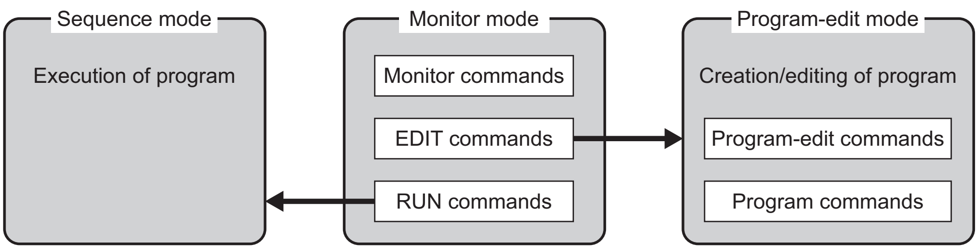
Sequences can be edited by either,
- Editing from the terminal.
- Editing from the IMC.

In this chapter, "Editing from the terminal" is explained. (See IMC tutorial for "Editing from the IMC")
The system enters this mode when "EDIT" is entered from the terminal.
In the sequence-edit mode, you can edit a sequence by changing, inserting or deleting specified lines. You can also perform a syntax check.

### ■ Executing Sequences

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

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.



## 5.2 Communication and Terminal Specifications

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

### Communication Specification

| Item | Description |
|---|---|
| Electrical Characteristics | In conformance with 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 | Selectable: 9600, 19200, 38400 bps (9600 is default.) |
| Protocol | TTY (CR+LF) |
| Connector specifications | Connector (6 lines, 6 pins) |

### Terminal Specifications

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

## 5.3 Creating a New Sequence

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

### Sequence Specifications

| | |
|---|---|
| Maximum number of programmable sequences | 100 sequences (Name is use configurable) |
| Maximum sequence size | 2KB maximum for total compiled sequences<br>4KB maximum for 1 sequence (text + compiled) |
| Sequence execution by external input | START input executes a sequence selected by binary combination of IN1 to IN6. |
| Maximum number of selectable sequences by external input | 64 sequences (0 to 63). Depending on INx assignment. |
| Automatic sequence execution at power up. | 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.<br>**Name may not begin with number, or “N_”, “S_”, “n_”, “s_”.**<br>Using αSTEP-One command 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. |

**Note** *Device memory status can be checked either by the “DIR” command from the terminal or by the “M” command while editing sequence.*

## ■ Example of Creating a New Sequence

1. Connect the device to the terminal.

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

```
>EDIT SAMPLE1
New Sequence
Sequence Name   : SAMPLE1
Sequence Number : 0
Lines   : 0
Bytes   : 0
Bytes Free      : 2048
>>Command: I
( 1)_
```

3. Enter commands and parameters by referring to Chapter 6, “Command List,” 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 operating velocity of 3 rev/sec., with a distance of 5 rev (DPR=1).
Insert a space or equal sign between the command and the parameter. See “Command Format” on page 326 as a reference.

```
>EDIT SAMPLE1
New Sequence
Sequence Name   : SAMPLE1
Sequence Number : 0
Lines   : 0
Bytes   : 0
Bytes Free      : 2048
>>Command: I
( 1) VS=1
( 2) VR=3
( 3) MI 5
( 4) _
```

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

```
Bytes   : 0
Bytes Free      : 2048
>>Command: I
( 1) VS=1
( 2) VR=3
( 3) MI 5
( 4)
>>Command: S
Compiling...OK
Storing.........OK
>>Command: Q
>_
```

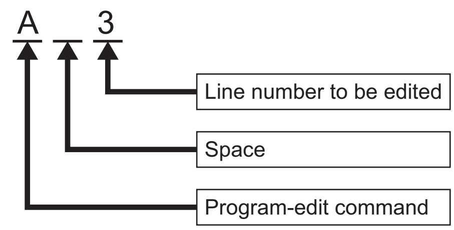
## 5.4 Editing an Existing Sequence

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

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

```
>EDIT PROGRAM1
Sequence Name   : PROGRAM1
Sequence Number : 1
Lines   : 5
Bytes   : 23
Bytes Free      : 2025
>>Command:_
```

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




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

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

## ■ Example of Line Editing

This section explains the steps to edit PROGRAM1 as follows:

• Before editing

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

• After editing

PROGRAM 1
(1) VR 5
(2) PC 12
(3) MA 5 ← Line 3 is changed from MI to MA5.
(4) WAIT10 ← 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      : 2025
>>Command:_
```

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

```
>EDIT PROGRAM1
Sequence Name   : PROGRAM1
Sequence Number : 1
Lines   : 5
Bytes   : 23
Bytes Free      : 2025
>>Command: L
( 1) VR5
( 2) PC12
( 3) MI
( 4) MGHP
( 5) END
>>Command:_
```

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

```
>EDIT PROGRAM1
Sequence Name   : PROGRAM1
Sequence Number : 1
Lines   : 5
Bytes   : 23
Bytes Free      : 2025
>>Command: A 3
( 3) MI_
```

b. Delete “MI” with the Back space key.

```
>EDIT PROGRAM1
Sequence Name   : PROGRAM1
Sequence Number : 1
Lines   : 5
Bytes   : 23
Bytes Free      : 2025
>>Command: A 3
( 3) _
```

c. Enter “MA 5”.

```
>EDIT PROGRAM1
Sequence Name   : PROGRAM1
Sequence Number : 1
Lines   : 5
Bytes   : 23
Bytes Free      : 2025
>>Command: A 3
( 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 program-edit command.

```
>EDIT PROGRAM1
Sequence Name   : PROGRAM1
Sequence Number : 1
Lines   : 5
Bytes   : 23
Bytes Free      : 2025
>>Command: A 3
( 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.

```
>EDIT PROGRAM1
Sequence Name   : PROGRAM1
Sequence Number : 1
Lines   : 5
Bytes   : 23
Bytes Free      : 2025
>>Command: A 3
( 3)MA 5_
>>Command: I 4
( 4)_
```

b. Enter “WAIT 10”.

```
>EDIT PROGRAM1
Sequence Name   : PROGRAM1
Sequence Number : 1
Lines   : 5
Bytes   : 23
Bytes Free      : 2025
>>Command: A 3
( 3) MA 5_
>>Command: I 4
( 4) WAIT 10_
```

c. Press the Enter key.
“WAIT 10” is added to line 4 of PROGRAM1.
You will now insert a new line at line 5.

```
>EDIT PROGRAM1
Sequence Name   : PROGRAM1
Sequence Number : 1
Lines   : 5
Bytes   : 23
Bytes Free      : 2025
>>Command: A 3
( 3) MA 5_
>>Command: I 4
( 4) WAIT 10
( 5) _
```

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

```
>EDIT PROGRAM1
Sequence Name   : PROGRAM1
Sequence Number : 1
Lines   : 5
Bytes   : 23
Bytes Free      : 2025
>>Command: A 3
( 3) MA 5_
>>Command: I 4
( 4) WAIT 10
( 5)
>>Command:_
```

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

```
>EDIT PROGRAM1
Sequence Name   : PROGRAM1
Sequence Number : 1
Lines   : 5
Bytes   : 23
Bytes Free      : 2025
>>Command: A 3
( 3) MA 5_
>>Command: I 4
( 4) WAIT 10
( 5)
>>Command: D 5
>>Command:_
```

## ■ Ending the Edit Session

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

```
Bytes   : 23
Bytes Free      : 2025
>>Command: A 3
( 3) MA 5_
>>Command: I 4
( 4) WAIT 10
( 5)
>>Command: D 5
>>Command: S
Compiling...OK
Storing.........OK
>>Command:_
```

2. Enter "Q" to quit the sequence editor.
   a ">" (command prompt) is displayed.

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

# 5.5 Executing a Sequence

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

## ■ Executing a Sequence from the Terminal

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

## ■ Executing a Sequence from I/O

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

Example of selection

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

**Note**
- *Empty section means OFF.*
- *This selection can also be done with rotary digital switches.*

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

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

# 5.6 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 Displayed during Program Creation

Unknown command: xxxx.

Cause/action: Input at Editor prompt did not match any of the single-character Editor commands (which can be seen by entering 'H' for [H]elp).

Name too long.

Cause/action: Given sequence name exceeds 10 characters

Sequence is locked.

Cause/action: At "R X Y"(rename), "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. (example: 34c)

Invalid editor syntax.

Cause/action: Extra text is found after an editor command.

End line must follow start line

Cause/action: Many Editor commands can take both start and end line numbers ([A]lter, [D]elete, [L]ist, [C]opy, [C]ut…). The start line must be before the end line.

Missing line number argument.

Cause/action: Editor command prompt expecting and did not find a valid line number.

## ■ Error Messages Displayed during Syntax Check

Array index out of range.

Cause/action: Reference to POS[ ] data, index out of range. Can happen in any of MA POS[ ], POS[ ], POS[ ]=, =POS[ ].

Invalid argument.

Cause/action: Argument is invalid for the command. (MA xxx, WAIT xxx, VIEW xxx, etc)

Block depth too deep.

Cause/action: "Blocks" (WHILE−WEND, LOOP−ENDL, IF−ENDIF) can be "nested" inside each other. We permit up to 8 levels of nesting.

BREAKL outside LOOP block.

Cause/action: BREAKL is entered at the outside of LOOP block.

BREAKW outside WHILE block.

Cause/action: BREAKW is entered at the outside of WHILE block.

Conditional expression expected.

Cause/action: IF or WHILE statements require a conditional expression.

Invalid sequence number.

Cause/action: CALL by number detected invalid sequence number, number out of range (0 to 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 operations. |
| Invalid conditional. | |
| 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 operation. |
| Text beyond END. | |
| Cause/action: | (Non-commented) text found beyond END statement. |
| Unknown command or parameter. | |
| Cause/action: | Command or parameter is not found. |
| Unsupported precision. | |
| Cause/action: | Numeric constant specified with too much precision (e.g. 1.2345). |
| Read-only parameter. | |
| Cause/action: | Attempt to modify a read-only parameter (e.g. IA) |
| Parameter cannot be displayed. | |
| Cause/action: | Attempt to "View" a non-viewable parameter (e.g. KB, KBQ). |
| WAIT must be positive. | |
| Cause/action: | Negative number is entered as argument for WAIT. |

## ■ Error Messages Displayed during Program Execution

These are not displayed in multi axis mode.

| | |
|---|---|
| Position error over limit. | |
| Cause/action: | Reduce the inertial load, load torque or command velocity, or increase the acceleration/deceleration times. |
| Current over limit. | |
| Cause/action: | An excessive current has flowed into the power element of the inverter section. |
| Drive temperature over limit. | |
| Cause/action: | Driver temperature is over DTMPMAX.<br>Revise the ventilation condition so that the ambient temperature of the device becomes 50°C (185°F) or below. |
| Drive voltage over limit. | |
| Cause/action: | The main circuit's inverter voltage has exceeded the upper limit.<br>When an alarm has occurred during acceleration/deceleration:<br>Reduce the inertial load or increase the acceleration/deceleration times. |
| Drive voltage under minimum. | |
| Cause/action: | Detected main power OFF. Check the main power. |
| Motor temperature over limit. | |
| Cause/action: | Motor temperature is over MTMPMAX.<br>Revise the ventilation condition so that the ambient temperature of the device becomes 50°C (185°F) or below. |
| Max. permitted torque, duration over limit. | |
| Cause/action: | Over load torque applied with duration over OLTIME.<br>Review the load and acceleration/deceleration rates again. |
| Velocity over limit. | |
| Cause/action: | The velocity has exceeded 6000 r/min or OVERVEL value.<br>Set the speed of the device's output shaft to 5000 r/min or less. |
| EEPROM data corrupt. | |
| Cause/action: | EEPROM data is destroyed. |
| Position feedback signal error (no connection, etc.) | |
| Cause/action: | Sensor is not detected properly (no connection, etc). |
| Rotor not stationary at power up. | |
| Cause/action: | The device's power turned on while the device's output shaft was turning by external force.<br>Make sure the device's output shaft does not turn by external force when the power is input. |
| Both +LS, −LS ON. | |
| Cause/action: | Both the +LS and −LS are ON simultaneously.<br>Check the logic setting for hardware limit sensors Normally Open (N.O.) or Normally Closed (N.C.). |
| LS detected, opposite HOME direction. | |
| Cause/action: | Opposite LS is detected from HOME direction. Connect the +LS and −LS correctly. |
| Abnormal LS status detected on HOME. | |
| Cause/action: | Mechanical home seeking could not be executed correctly.<br>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. |
| TIMING, 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.<br>Reduce the number of nested commands. |
| Attempted to call non-existent sequence. | |
| Cause/action: | Non-existent program is called. Delete the program, then enter it again. |
| Calculation result overflow | |
| Cause/action: | Calculation result over flow. Enter a program name that exists. |
| Parameter out of range | |
| Cause/action: | Parameter exceeds its setting range. |
| Division by Zero detected | |
| Cause/action: | Divide by zero was executed. Revise the program. |
| Attempted to modify PC while moving. | |
| Cause/action: | “PC" command is updated while the device is operating or loses it’s holding torque.<br>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)<br>Display only "sub code" in ALM command. |
| Warning: Max. permitted torque detected | |
| Cause/action: | Over load torque applied with duration over OLTIME<br>Review the load and acceleration/deceleration rates again. |
| Warning: Drive input voltage is out of nominal range | |
| Cause/action: | Check the main power. |
| Warning: Insufficient torque for zero-speed requirement. CRSTOP is set to its limit | |
| Cause/action: | Stop current is insufficient. |
| Warning: Insufficient torque for motion requirement. CRRUN is set to its limit | |
| Cause/action: | Run current is insufficient. |
| Warning: Insufficient torque for acceleration requirement. | |
| Cause/action: | Acceleration current is insufficient. |
| Warning: Insufficient torque for deceleration requirement. | |
| Cause/action: | Deceleration current is insufficient. |

## Error Messages Relating to Monitor Commands

Error: Command or parameter is unknown.

Cause/action: Text entered at the command prompt is not recognized (e.g. “DIV”, “VY”).

Error: Action is not allowed. (Motor is moving)

Cause/action: Command is attempted that is not executable while motor is running.
Attempted to modify a parameter that may not be modified while motor is moving.

Error: Action is not allowed. (Sequence is running)

Cause/action: Command that starts motion is attempted while a sequence is running.

Error: Action is not allowed. (Alarm is ON)

Cause/action: Command is attempted that is not executable while alarm is ON.

Error: Action is not allowed. (Motion or I/O settings incompatible)

Cause/action: One of following situations is detected.

- Motion command attempted while current is OFF.
- MI, MA, EHOME, MCP, MCN execution with RMODE=1, load parameter (LI, LF, LG, LSF, TU) exceeds required motor torque.
- CV command is attempted while MI, MA, EHOME motion with RMODE=1.
- CV command is attempted while decelerating at MI, MA, EHOME motion.
- MIx (Link index) is attempted with RMODE=1 (auto ramp mode).
- 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. “MODE=10”, “VR=−0.1”)

Error: Argument is out of range.

Cause/action: Attempted to execute command, argument is out of range.
(e.g. "MA 5000000000000000")

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) exceeds 10 characters.

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. “VF 10”, “CRDEC=50” (at CMODE=2))

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.

# Chapter 6 Command List

This chapter provides detailed information about each command and parameter.
In the tables below, the commands are grouped by functionality, for quick reference. After the tables, each command or parameter is described in detail, in alphabetical order.

## ■ Table Keys

| | |
|---|---|
| n/a | not applicable |
| MAXVEL | Maximum permissable velocity value, in User Units per second. MAXVEL can be queried directly, see MAXVEL for details. |
| MAXPOS | Maximum permissable 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 |
| SAVE & RESET REQUIRED | n/a : Not applicable, or not required. For parameters, new value becomes active immediately.<br>S : New value active immediately, but save command required for new value to be active after reset or power cycle.<br>SR : Save and Reset (or save and power cycle) required before new value becomes active. |
| MODE | Value(s) of MODE in which this command or parameter is available |
| IN SEQ? | yes : Command or parameter can be used within sequences.<br>− : Command or parameter cannot be used within sequences. |
| h | (after a value) Value is shown in hexadecimal notation |
| UU | User Units. Value shown in units determined by the user. See Section 4.3, "Initial Setting (User Unit)" for more details |
| source<br>target<br>newname | Sequence names or sequence numbers, as appropriate. Names can be Up to 10 letters or numbers, and must start with a letter. |

## Motion Commands

| COMMAND | DESCRIPTION | DEFAULT VALUE | RANGE | SAVE & RESET REQUIRED | MODE | IN SEQ? | PAGE |
|---|---|---|---|---|---|---|---|
| CONT | Continue motion | n/a | n/a | n/a | 0 | yes | 109 |
| CV | Change velocity | n/a | 0.001 to MAXVEL [UU/sec.] | n/a | 0 | yes | 119 |
| EHOME | Return to electrical home | n/a | n/a | n/a | 0 | yes | 137 |
| HSTOP | Hard stop | n/a | n/a | n/a | 0 | yes | 151 |
| MA | Move to absolute position | n/a | −MAXPOS to +MAXPOS [UU] | n/a | 0 | yes | 187 |
| MCN, MCP | Move continuously, negative or positive | n/a | n/a | n/a | 0 | yes | 192 |
| MGHN, MGHP | Seek mechanical home position | n/a | n/a | n/a | 0 | yes | 194 |
| MI | Move incremental distance | n/a | n/a | n/a | 0 | yes | 195 |
| MIx | Start linked index | n/a | (x = 0 to 3) | n/a | 0 | yes | 196 |
| MSTOP | Motor stop | n/a | n/a | n/a | 0 | yes | 201 |
| PAUSE | Pause motion | n/a | n/a | n/a | 0 | yes | 226 |
| PAUSECLR | Clear state of paused motion | n/a | n/a | n/a | 0 | yes | 227 |
| PSTOP | Panic stop | n/a | n/a | n/a | 0 | yes | 236 |
| SSTOP | Soft stop | n/a | n/a | n/a | 0 | yes | 277 |

## Motion Variables

| COMMAND | DESCRIPTION | DEFAULT VALUE | RANGE | SAVE & RESET REQUIRED | MODE | IN SEQ? | PAGE |
|---|---|---|---|---|---|---|---|
| DIS | Incremental motion distance | 0 | −MAXPOS to +MAXPOS [UU] | S | 0 | yes | 125 |
| DISx | Linked motion distance or destination | 0 | (x = 0 to 3)<br>−MAXPOS to +MAXPOS [UU] | S | 0 | yes | 126 |
| INCABSx | Linked move type | 1 | (x = 0 to 3)<br>0 [Absolute] to 1 [Incremental] | S | 0 | yes | 157 |
| LINKx | Link control | 0 | (x = 0 to 2)<br>0 [No Link] to 1 [Link-to-next] | S | 0 | yes | 181 |
| OFFSET | Home offset position | 0 | −MAXPOS to +MAXPOS [UU] | S | 0 | yes | 208 |
| POS[x] | Position array data | 0 | (x = 1 to 100)<br>−MAXPOS to +MAXPOS [UU] | S | 0 | yes | 235 |
| RMODE | Ramp mode | 0 | 0 [Linear profile]<br>1 [Automatic profile] | S | 0 | yes | 244 |
| SCHGPOS | Distance after SENSOR input | 0 | 0 to MAXPOS [UU] | S | 0 | yes | 253 |
| SCHGVR | Velocity after SENSOR input | 1 | 0.001 to MAXVEL [UU/sec.] | S | 0 | yes | 254 |
| TA | Acceleration time | 0.5 | 0.001 to 500.00 [sec.] | S | 0 | yes | 281 |
| TD | Deceleration time | 0.5 | 0.001 to 500.00 [sec.] | S | 0 | yes | 283 |
| VR | Running velocity | 1 | 0.001 to MAXVEL [UU/sec.] | S | 0 | yes | 299 |
| VRx | Linked motion running velocity | 1 | (x = 0 to 3)<br>0.001 to MAXVEL [UU/sec.] | S | 0 | yes | 300 |
| VS | Starting velocity | 0.1 | 0.001 to MAXVEL [UU/sec.] | S | 0 | yes | 301 |

## System Control

| COMMAND | DESCRIPTION | DEFAULT VALUE | RANGE | SAVE & RESET REQUIRED | MODE | IN SEQ? | PAGE |
|---|---|---|---|---|---|---|---|
| ; | Statement separator for multi-statement | n/a | n/a | n/a | 0−3 | yes | 84 |
| <ESC> | (Escape): Abort operation(s) | n/a | n/a | n/a | 0−3 | − | 86 |
| ABORT | Abort sequences and motions | n/a | n/a | n/a | 0 | yes | 88 |
| ALMACT | Alarm action | 2 | 0 [No Alarm]<br>1 [Abort, Current On, Alarm On]<br>2 [Abort, Current Off, Alarm On] | SR | 0−3 | − | 92 |
| ALMCLR | Clear alarm | n/a | n | n/a | 0−3 | − | 93 |
| ALMMSG | Alarm message action | 0 | 0 [No messages]<br>1 [Messages, Alarms Only]<br>2 [Messages, Alarms and Warnings] | S | 0−3 | − | 95 |
| ALMSET | Set user alarm | n/a | n/a | n/a | 0−3 | yes | 96 |
| CLEARALL | Clear all programming (return to factory condition) | n/a | n/a | n/a | 0 | − | 104 |
| CLEARPOS | Clear POS[x] position array data | n/a | n/a | n/a | 0 | − | 105 |
| CMODE | Current mode | 0 | 0 [Basic]<br>1 [Manual]<br>2 [Automatic] | S | 0 | yes | 108 |
| CRACC | Acceleration current (also deceleration current if CMODE=2) | 100 | 25 to 100 [% of Rated Current] | S | 0 | yes | 112 |
| CRDEC | Deceleration current (when CMODE=2 only) | n/a | 25 to 100 [% of Rated Current] | n/a | 0 | yes | 113 |
| CRRUN | Run current | 100 | 0 to 100 [% of Rated Current] | S | 0−3 | yes | 116 |
| CRSTOP | Stop current | 50 | 0 to 100 [% of Rated Current] | S | 0−3 | yes | 117 |
| CURRENT | Current On/Off | 1 | 0 [Motor Current Off]<br>1 [Motor current On] | n/a | 0−3 | yes | 118 |
| DIRINV | Direction Invert | 0 | 0 [positive motion is clockwise]<br>1 [positive motion is counter-clockwise] | SR | 0−3 | yes | 124 |
| DPP | Distance Per Pulse | 0.001 | 0.001 to DPR/500 [UU] | SR | 0−3 | − | 127 |
| DPR | Distance per revolution | 1 | 0.5 to 51200 [UU] | SR | 0−3 | yes | 128 |
| DTMPMAX | Maximum drive temperature | 80 | 0 to 80 [°C] | S | 0−3 | yes | 131 |
| DTMPWRN | Drive warning temperature | 80 | 0 to 80 [°C] | S | 0−3 | − | 132 |
| DVINSET | Nominal drive input voltage | 24 | 12 to 48 [Volts, DC] | S | 0 | − | 134 |
| FILT | Jerk filter time lag | 0.003 | 0 to 1 [sec.] | S | 0−3 | yes | 144 |
| GA, GB | Electronic Gear Ratio | 1 | 1 to 100 | SR | 0−3 | yes | 145 |
| HOMETYP | Homing type | 0 | 0 to 11<br>(See HOMETYP entry　for a full explanation.) | S | 0 | yes | 150 |
| INITPRM | Initialize parameters | n/a | n/a | n/a | 0−3 | − | 161 |
| LF | Load friction | 0 | 0 to 200 [N·cm] | S | 0 | yes | 176 |
| LG | Load gravity | 0 | −200 to 200 [N·cm] | S | 0 | yes | 177 |
| LI | Load inertia | 0 | 0 to 12,000 [g·cm$^2$] | S | 0 | yes | 178 |
| LIMN, LIMP | Software position limits | 0 | −MAXPOS to +MAXPOS [UU] | SR | 0 | yes | 179 |
| LSF | Load static friction | 0 | 0 to 200 [N·cm] | S | 0 | yes | 186 |
| MODE | Device mode | 0 | 0 [Internal Motion Profile]<br>1 [1-Pulse Input Mode]<br>2 [2-Pulse Input Mode]<br>3 [Quadrature Pulse Input Mode] | SR | 0−3 | yes | 198 |
| MSTOPACT | Motor stop action | 1 | 0 [Hard Stop]<br>1 [Soft Stop] | SR | 0 | − | 202 |
| MTMPMAX | Maximum motor temperature | 85 | 0 to 85 [°C] | SR | 0−3 | yes | 205 |
| MTMPWRN | Motor warning temperature | 85 | 0 to 85 [°C] | SR | 0−3 | − | 206 |
| OLTIME | Overload time | 5.0 | 0.5 to 25.000 [sec.] | SR | 0−3 | yes | 209 |
| OTACT | Overtravel action | 0 | 0 [Hard Stop]<br>1 [Soft Stop] | SR | 0 | − | 210 |
| OVERFLOW | Position error limit | 3 | 0.001 to MAXOVERFLOW [UU] | SR | 0−3 | yes | 224 |
| OVERVEL | Velocity limit | 100 | 0.001 to (1.2×MAXVEL) [UU/sec.] | SR | 0−3 | yes | 225 |
| RESET | Reset device | n/a | n/a | n/a | 0−3 | − | 242 |
| SAVEALL | Save all data | n/a | n/a | n/a | 0 | − | 250 |

| COMMAND | DESCRIPTION | DEFAULT VALUE | RANGE | SAVE & RESET REQUIRED | MODE | IN SEQ? | PAGE |
|---|---|---|---|---|---|---|---|
| SAVEPOS | Save position array data | n/a | n/a | n/a | 0 | − | 251 |
| SAVEPRM | Save parameters | n/a | n/a | n/a | 0−3 | − | 252 |
| SENSORACT | SENSOR input action | 2 | 0 [Hard Stop]<br>1 [Soft Stop]<br>2 [Soft stop at fixed distance from SENSOR signal] | SR | 0 | − | 255 |
| SLACT | Software position limit control | 0 | 0 [Disabled]<br>1 [Enabled after homing] | SR | 0 | yes | 275 |
| STARTACT | START input action | 0 | 0 [Start Sequence when set active]<br>1 [Start Sequence when set active, Abort when set inactive] | SR | 0 | − | 278 |
| STRSW | Current state at system start | 1 | 0 [Current Off]<br>1 [Current On] | SR | 0−3 | − | 280 |
| TQFF | Torque feedforward control | 0 | 0 [Disabled]<br>1 [Enabled] | S | 0 | yes | 288 |
| TU | Torque utilization | 50 | 0 to 100 [% Rated Torque] | S | 0 | yes | 290 |
| UU | User units | Rev | Text string, 20 characters maximum | S | 0−3 | − | 292 |

## System Status

| COMMAND | DESCRIPTION | DEFAULT VALUE | RANGE | SAVE & RESET REQUIRED | MODE | IN SEQ? | PAGE |
|---|---|---|---|---|---|---|---|
| DTMP | Drive temperature | n/a | n/a [°C] | n/a | 0−3 | yes | 130 |
| DVIN | Drive input voltage | n/a | n/a [Volts, DC] | n/a | 0−3 | yes | 133 |
| IA | Motor phase current amplitude | n/a | 0.000 to 3.6000 [Amp] | n/a | 0−3 | yes | 152 |
| MAXPOS | Maximum position value | 41493 | n/a [UU] | n/a | 0−3 | − | 189 |
| MAXVEL | Maximum velocity value | 83.333 | n/a [UU/sec.] | n/a | 0−3 | − | 190 |
| MAXOVERFLOW | Maximum OVERFLOW | 600 | n/a [UU] | n/a | 0−3 | − | 191 |
| MTMP | Motor temperature | n/a | n/a [°C] | n/a | 0−3 | yes | 204 |
| PC | Position command | n/a | −MAXPOS to +MAXPOS [UU] | n/a | 0−3 | yes | 230 |
| PCI | Incremental position command | n/a | −MAXPOS to +MAXPOS [UU] | n/a | 0 | yes | 231 |
| PE | Position error | n/a | −MAXPOS to +MAXPOS [UU] | n/a | 0−3 | yes | 232 |
| PF | Motor position | n/a | −MAXPOS to +MAXPOS [UU] | n/a | 0−3 | yes | 233 |
| PFI | Incremental motor position | n/a | −MAXPOS to +MAXPOS [UU] | n/a | 0 | yes | 234 |
| RC | Motor shaft Position | n/a | 0.000 to 359.999 [angular degrees] | n/a | 0−3 | yes | 239 |
| SIGALARM | System ALARM output signal | n/a | 0 [Off]<br>1 [On] | n/a | 0−3 | yes | 257 |
| SIGALMCLR | System ALMCLR input signal | n/a | 0 [Off]<br>1 [On] | n/a | 0−3 | yes | 258 |
| SIGCROFF | System CROFF input signal | n/a | 0 [Off]<br>1 [On] | n/a | 0−3 | yes | 259 |
| SIGEND | System END output signal | n/a | 0 [Off]<br>1 [On] | n/a | 0−3 | yes | 260 |
| SIGHOME | System HOME input signal | n/a | 0 [Off]<br>1 [On] | n/a | 0 | yes | 261 |
| SIGHOMEP | System HOMEP output signal | n/a | 0 [Off]<br>1 [On] | n/a | 0 | yes | 262 |
| SIGLSN | System LSN input signal | n/a | 0 [Off]<br>1 [On] | n/a | 0−3 | yes | 263 |
| SIGLSP | System LSP input signal | n/a | 0 [Off]<br>1 [On] | n/a | 0−3 | yes | 264 |
| SIGMBC | System MBC output signal | n/a | 0 [Off]<br>1 [On] | n/a | 0−3 | yes | 265 |
| SIGMOVE | System MOVE output signal | n/a | 0 [Off]<br>1 [On] | n/a | 0 | yes | 266 |
| SIGMSTOP | System MSTOP input signal | n/a | 0 [Off]<br>1 [On] | n/a | 0 | yes | 267 |
| SIGPAUSE | System PAUSE input signal | n/a | 0 [Off]<br>1 [On] | n/a | 0 | yes | 268 |
| SIGPAUSECL | System PAUSECL input signal | n/a | 0 [Off]<br>1 [On] | n/a | 0 | yes | 269 |

| COMMAND | DESCRIPTION | DEFAULT VALUE | RANGE | SAVE & RESET REQUIRED | MODE | IN SEQ? | PAGE |
|---|---|---|---|---|---|---|---|
| SIGPSTOP | System PSTOP input signal | n/a | 0 [Off]<br>1 [On] | n/a | 0−3 | yes | 270 |
| SIGPSTS | System PSTS output signal | n/a | 0 [Off]<br>1 [On] | n/a | 0 | yes | 271 |
| SIGRUN | System RUN output signal | n/a | 0 [Off]<br>1 [On] | n/a | 0 | yes | 272 |
| SIGSENSOR | System SENSOR input signal | n/a | 0 [Off]<br>1 [On] | n/a | 0 | yes | 273 |
| SIGTEMP | System TEMP output signal | n/a | 0 [Off]<br>1 [On] | n/a | 0−3 | yes | 274 |
| TF | Motor torque | n/a | n/a [Nt·m] | n/a | 0−3 | yes | 286 |
| TIMER | Running timer | n/a | 0 to 500,000.000 [sec.] | n/a | 0−3 | yes | 287 |
| VC | Velocity command | n/a | −MAXVEL to +MAXVEL [UU/sec.] | n/a | 0−3 | yes | 293 |
| VE | Velocity error | n/a | n/a [UU/sec.] | n/a | 0−3 | yes | 294 |
| VF | Motor Velocity | n/a | n/a [UU/sec.] | n/a | 0−3 | yes | 297 |

## I/O

| COMMAND | DESCRIPTION | DEFAULT VALUE | RANGE | SAVE & RESET REQUIRED | MODE | IN SEQ? | PAGE |
|---|---|---|---|---|---|---|---|
| ABORTLV | ABORT input level | 0 | 0 [Normally Open]<br>1 [Normally Closed] | SR | 0 | − | 89 |
| ALARMLV | ALARM output level | 0 | 0 [Normally Open]<br>1 [Normally Closed] | SR | 0−3 | − | 90 |
| ALMCLRLV | ALMCLR input level | 0 | 0 [Normally Open]<br>1 [Normally Closed] | SR | 0−3 | − | 94 |
| CROFFLV | CROFF input level | 0 | 0 [Normally Open]<br>1 [Normally Closed] | SR | 0−3 | − | 115 |
| ENDLV | END output level | 0 | 0 [Normally Open]<br>1 [Normally Closed] | SR | 0−3 | − | 142 |
| EVx | Configure event output | n/a | (x = 1 to 2)<br>(See EVx entry for a full explanation.) | n/a | 0 | yes | 143 |
| HOMELV | HOME input level | 0 | 0 [Normally Open]<br>1 [Normally Closed] | SR | 0 | − | 148 |
| HOMEPLV | HOMEP output level | 0 | 0 [Normally Open]<br>1 [Normally Closed] | SR | 0 | − | 149 |
| IN | General input status | n/a | 0 to 63 | n/a | 0−3 | yes | 155 |
| INALMCLR | ALMCLR signal input assignment | 0 | 0 [unassigned] and 1 to 6 | SR | 0 | − | 156 |
| INCROFF | CROFF signal input assignment | 0 | 0 [unassigned] and 1 to 6 | SR | 0 | − | 158 |
| INHOME | HOME signal input assignment | 0 | 0 [unassigned] and 1 to 6 | SR | 0 | − | 159 |
| INITIO | Initialize I/O | n/a | n/a | n/a | 0 | − | 160 |
| INLSN, INLSP | LSN, LSP signal input assignments | 0 | 0 [unassigned] and 1 to 6 | SR | 0 | − | 162 |
| INMSTOP | MSTOP signal input assignment | 0 | 0 [unassigned] and 1 to 6 | SR | 0 | − | 163 |
| INPAUSE | PAUSE signal input assignment | 0 | 0 [unassigned] and 1 to 6 | SR | 0 | − | 164 |
| INPAUSECL | PAUSECL signal input assignment | 0 | 0 [unassigned] and 1 to 6 | SR | 0 | − | 165 |
| INPSTOP | PSTOP signal input assignment | 0 | 0 [unassigned] and 1 to 6 | SR | 0 | − | 166 |
| INSENSOR | SENSOR signal input assignment | 0 | 0 [unassigned] and 1 to 6 | SR | 0 | − | 167 |
| INSG | System input signal status | 0 | 0 to 4095 | n/a | 0−3 | yes | 168 |
| INx | Individual general input status | 0 | (x = 1 to 6)<br>0 [Off]<br>1 [On] | n/a | 0−3 | yes | 169 |
| INxLV | INx input level | 0 | (x = 1 to 6)<br>0 [Normally Open]<br>1 [Normally Closed] | SR | 0 | − | 170 |
| IO | Input/Output status | n/a | n/a | n/a | 0−3 | − | 171 |
| MOVELV | MOVE output level | 0 | 0 [Normally Open]<br>1 [Normally Closed] | SR | 0 | − | 200 |
| MSTOPLV | MSTOP input level | 0 | 0 [Normally Open]<br>1 [Normally Closed] | SR | 0 | − | 203 |
| OTLV | Overtravel input level | 0 | 0 [Normally Open] | SR | 0−3 | − | 211 |

| COMMAND | DESCRIPTION | DEFAULT VALUE | RANGE | SAVE & RESET REQUIRED | MODE | IN SEQ? | PAGE |
|---|---|---|---|---|---|---|---|
| | | | 1 [Normally Closed] | | | | |
| OUT | General output status | n/a | 0 to 15 | n/a | 0−3 | − | 212 |
| OUTALARM | ALARM signal output assignment | 0 | 0 [unassigned] and 1 to 4 | SR | 0 | − | 213 |
| OUTEND | END signal output assignment | 0 | 0 [unassigned] and 1 to 4 | SR | 0 | − | 214 |
| OUTHOMEP | HOMEP signal output assignment | 0 | 0 [unassigned] and 1 to 4 | SR | 0 | − | 215 |
| OUTMBC | MBC signal output assignment | 0 | 0 [unassigned] and 1 to 4 | SR | 0 | − | 216 |
| OUTMOVE | MOVE signal output assignment | 0 | 0 [unassigned] and 1 to 4 | SR | 0 | − | 217 |
| OUTPSTS | PSTS signal output assignment | 0 | 0 [unassigned] and 1 to 4 | SR | 0 | − | 218 |
| OUTRUN | RUN signal output assignment | 0 | 0 [unassigned] and 1 to 4 | SR | 0 | − | 219 |
| OUTSG | System output signal status | n/a | 0 to 255 | n/a | 0−3 | yes | 220 |
| OUTTEMP | TEMP signal output assignment | 0 | 0 [unassigned] and 1 to 4 | SR | 0 | − | 221 |
| OUTTEST | I/O test utility | n/a | n/a | n/a | 0−3 | − | 222 |
| OUTx | Individual general output control | 0 | (x = 1 to 4)<br>0 [Off]<br>1 [On] | n/a | 0−3 | yes | 223 |
| PAUSECLLV | PAUSECL input level | 0 | 0 [Normally Open]<br>1 [Normally Closed] | SR | 0 | − | 228 |
| PAUSELV | PAUSE input level | 0 | 0 [Normally Open]<br>1 [Normally Closed] | SR | 0 | − | 229 |
| PSTOPLV | PSTOP input level | 0 | 0 [Normally Open]<br>1 [Normally Closed] | SR | 0−3 | − | 237 |
| PSTSLV | PSTS output level | 0 | 0 [Normally Open]<br>1 [Normally Closed] | SR | 0 | − | 238 |
| RUNLV | RUN output level | 0 | 0 [Normally Open]<br>1 [Normally Closed] | SR | 0 | − | 246 |
| SENSORLV | SENSOR input level | 0 | 0 [Normally Open]<br>1 [Normally Closed] | SR | 0 | − | 256 |
| STARTLV | START input level | 0 | 0 [Normally Open]<br>1 [Normally Closed] | SR | 0 | − | 279 |
| TEMPLV | TEMP output level | 0 | 0 [Normally Open]<br>1 [Normally Closed] | SR | 0−3 | − | 285 |

## Monitor Commands

| COMMAND | DESCRIPTION | DEFAULT VALUE | RANGE | SAVE & RESET REQUIRED | MODE | IN SEQ? | PAGE |
|---|---|---|---|---|---|---|---|
| ALM | Alarm status and history | n/a | 00h to F2h | n/a | 0−3 | − | 91 |
| HELP | Display help information | n/a | n/a | n/a | 0−3 | − | 147 |
| LDCHK | Estimate load parameters | n/a | n/a | n/a | 0 | − | 174 |
| REPORT | Display system status | n/a | n/a | n/a | 0−3 | − | 241 |
| TEACH | Teach Positions | n/a | n/a | n/a | 0 | − | 284 |
| TRACE | Sequence trace control | 0 | 0 [Disabled]<br>1 [Enabled] | n/a | 0 | − | 289 |
| VER | Display firmware version | n/a | n/a | n/a | 0−3 | − | 295 |

## Communications

| COMMAND | DESCRIPTION | DEFAULT VALUE | RANGE | SAVE & RESET REQUIRED | MODE | IN SEQ? | PAGE |
|---|---|---|---|---|---|---|---|
| @ | Select device | n/a | *, 0 to 9, A to Z | n/a | 0−3 | − | 85 |
| \ | Global command | n/a | n/a | n/a | 0−3 | − | 99 |
| BAUD | Communication BAUD rate | 0 | 0 [9600 bps]<br>1 [19200 bps]<br>2 [38400 bps] | SR | 0−3 | yes | 100 |
| ECHO | Communications echo control | 1 | 0 [Echo Off]<br>1 [Echo On] | S | 0−3 | − | 135 |
| ID | Device ID | * | *, 0 to 9, A to Z | S | 0−3 | − | 153 |
| TALK | Select device | n/a | *, 0 to 9, A to Z | n/a | 0−3 | − | 282 |
| VERBOSE | Command response control | 1 | 0 [Respond with data only]<br>1 [Respond with data and descriptive text] | S | 0−3 | − | 296 |

## Sequence　Commands

| COMMAND | DESCRIPTION | DEFAULT VALUE | RANGE | SAVE & RESET REQUIRED | MODE | IN SEQ? | PAGE |
|---|---|---|---|---|---|---|---|
| # | Sequence Comment | n/a | n/a | n/a | 0 | yes | 81 |
| BREAKL | Break LOOP block | n/a | n/a | n/a | 0 | yes | 101 |
| BREAKW | Break WHILE block | n/a | n/a | n/a | 0 | yes | 102 |
| CALL | Call sequence as subroutine | n/a | Valid sequence name or number. May be a variable. | n/a | 0 | yes | 103 |
| ELSE | Begin ELSE block: execute if IF is false | n/a | n/a | n/a | 0 | yes | 138 |
| END | End sequence | n/a | n/a | n/a | 0 | yes | 139 |
| ENDIF | End of IF block | n/a | n/a | n/a | 0 | yes | 140 |
| ENDL | End of LOOP block | n/a | n/a | n/a | 0 | yes | 141 |
| IF | Begin IF block: execute if true | n/a | n/a | n/a | 0 | yes | 154 |
| KB | Keyboard Input | n/a | Depends on target variable | n/a | 0 | yes | 172 |
| KBQ | Keyboard Input (quiet) | n/a | Depends on target variable | n/a | 0 | yes | 173 |
| LOOP | Begin counted LOOP block | n/a | 1 to 500,000,000. Must be integer.　To make an infinite loop, omit count.　Count may be a variable. | n/a | 0 | yes | 185 |
| MEND | Wait for motion end | n/a | n/a | n/a | 0 | yes | 193 |
| RET | Sequence Return | n/a | n/a | n/a | 0 | yes | 243 |
| SACS | Send ASCII control string | n/a | Characters to transmit (up to 70). May contain embedded control characters.　Does not append carriage return and line feed. No new prompt. | n/a | 0 | yes | 248 |
| SAS | Send ASCII string | n/a | Characters to transmit (up to 70). Appends carriage return and line feed.　New prompt. | n/a | 0 | yes | 249 |
| VIEW | View parameter | n/a | Valid parameter or variable name | n/a | 0 | yes | 298 |
| WAIT | Wait for specified time | n/a | 0 − 500000.000 [sec.] May be a variable. | n/a | 0 | yes | 302 |
| WEND | End of WHILE block | n/a | n/a | n/a | 0 | yes | 303 |
| WHILE | Begin WHILE block: execute while true | n/a | n/a | n/a | 0 | yes | 304 |

## Math/Logical Operators (In sequences only)

| COMMAND | DESCRIPTION | DEFAULT VALUE | RANGE | SAVE & RESET REQUIRED | MODE | IN SEQ? | PAGE |
|---|---|---|---|---|---|---|---|
| &, \|, ^, <<, >> | AND, OR, XOR, left logic shift, right logic shift | n/a | n/a | n/a | 0 | yes | 82 |
| +, −, *, /, % | Addition, subtraction, multiplication, division, modulo | n/a | n/a | n/a | 0 | yes | 82 |
| a < b | a is smaller than b | n/a | n/a | n/a | 0 | yes | 87 |
| a <= b | a is equal to or smaller than b | n/a | n/a | n/a | 0 | yes | 87 |
| a = b | a is equal to b | n/a | n/a | n/a | 0 | yes | 87 |
| a! = b | a is not equal to b | n/a | n/a | n/a | 0 | yes | 87 |
| a >= b | a is equal to or larger than b | n/a | n/a | n/a | 0 | yes | 87 |
| a > b | a is larger than b | n/a | n/a | n/a | 0 | yes | 87 |

## User Variables

| COMMAND | DESCRIPTION | DEFAULT VALUE | RANGE | SAVE & RESET REQUIRED | MODE | IN SEQ? | PAGE |
|---|---|---|---|---|---|---|---|
| A to Z | User variables | 0 | ±Max. Number | S | 0 | yes | 97 |
| CLEARVAR | Clear user-defined variables | n/a | n/a | S | 0 | − | 107 |
| CREATEVAR | Create user-defined variable | n/a | N_xxx [Numeric Type]<br>S_xxx [String Type] | S | 0 | − | 114 |
| DELETEVAR | Delete user-defined variable | n/a | N_xxx [Numeric Type]<br>S_xxx [String Type] | S | 0 | − | 121 |
| LISTVAR | Lists all user-defined variables | n/a | n/a | n/a | 0 | − | 183 |
| N_xxx | User-defined numeric variables | (0 when created) | ±Max. Number | S | 0 | yes | 207 |
| S_xxx | User-defined string variables | (empty when created) | Text string, 20 characters maximum | S | 0 | yes | 247 |

## Sequence Management

| COMMAND | DESCRIPTION | SYNTAX | MODE | IN SEQ? | PAGE |
|---|---|---|---|---|---|
| CLEARSEQ | Clear sequences | CLEARSEQ | 0 | − | 106 |
| COPY | Copy sequence | COPY source target | 0 | − | 111 |
| DEL | Delete sequence | DEL target | 0 | − | 120 |
| DIR | Sequence Directory | DIR [target] | 0 | − | 123 |
| EDIT | Edit sequence | EDIT [target] | 0 | − | 136 |
| LIST | List sequence contents | LIST target [startline] [endline] | 0 | − | 182 |
| LOCK | Lock Sequence | LOCK target | 0 | − | 184 |
| REN | Rename sequence | REN target newname | 0 | − | 240 |
| RUN | Run sequence | RUN target | 0 | − | 245 |
| UNLOCK | Unlock sequence | UNLOCK target | 0 | − | 291 |

## # : Sequence Comment — Sequence Command

**Execution Mode** Sequence

**Syntax** #commenting text

**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.

**See Also** EDIT, LIST

**Example**

| Command | Description |
|---|---|
| `>LIST 1` | #List sequence 1 |
| `(1) TA=0.5` | #Acceleration Time, seconds |
| `(2) TD=0.5` | #Deceleration Time, seconds |
| `(3) VS=10` | #Starting Velocity, User Units/second |
| `(4) VR=20` | #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 |
| `>` | |

## +, −, *, /, %, &, |, ^, <<, >> — Math/Logical Operators

**Execution Mode** Sequence

**Syntax**
Z = X n Y
X = Numeric Value or Variable
n = mathematical operation
Y = Numeric Value or Variable
Z = Variable

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

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

Note on shift operations: A = B >> C, A = B << C:

If B has a fractional part, the shift operation is treated as a multiply (or divide) by the appropriate power of 2.

Note on Modulo operations: A%B = A − (B * sign (A/B) * floor (|A/B|) )

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

**See Also** A to Z

**Example**

| Command | Description |
|---|---|
| `>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 |
| `>` | |

## / (Forward slash) : Display Continuously — System Control

**Execution Mode** Immediate

**Syntax** (command) /

**Description** 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

Applicable Displayed Commands:
DTMP, DVIN, IA, IN, INSG, INx, IO, MTMP, OUT, OUTSG, OUTx, PC, PCI, PE, PF, PFI, RC, SIGALARM, SIGCROFF, SIGEND, SIGHOME, SIGHOMEP, SIGLSN, SIGLSP, SIGMBC, SIGMOVE, SIGMSTOP, SIGPAUSE, SIGPAUSECL, SIGPSTOP, SIGPSTS, SIGRUN, SIGSENSOR, SIGTEMP, TF, TIMER, VC, VE, VF

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 must be transmitted to the device a second time to cause the motion to cease.

**Caution** **Do not confuse this special command with the division operator, "/".**

**Example**

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

## ; : Statement Separator — System Control

**Execution Mode** Immediate and Program

**Syntax** Command; Command

**Description** The semicolon (;) allows for multiple command statements to be used on a single command line. The maximum number of characters per one line is 80 characters.

**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 mm/second, distance to 2 mm and them perform an index move |
| `VR=10 mm/sec` | #Device response |
| `DIS=2 mm` | #Device response |
| `>` | #Device response |

## @ : Select Device — Communications

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | @id |
| **Range** | id = *, 0 to 9, A to Z |
| **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 default ID (*), communication with the device requires using the @ or TALK commands to establish communication. |
| **See Also** | TALK, 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>@A` | #Talk to Device A |
| `a>MGHP` | #Device A go home |

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

**Execution Mode** Immediate

**Syntax** <ESC> (Escape key or character)

**Description** <ESC> represents an escape key or character (1Bh).
<ESC> will abort motion, decelerating to a stop.
<ESC> will abort an executing sequence.
<ESC> will also abort continuous display of a parameter via the (/) command.
<ESC> will discard any characters on a line and send a carriage return and line feed (CR + LF), and new prompt.

**See Also** ABORT, ALMACT, HSTOP, MSTOP, MSTOPACT, PSTOP, SSTOP, TD

**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 : Conditional Operators    Math/Logical Operators

**Execution Mode** Sequence

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

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

**See Also** IF, WHILE

**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 |
| `>` | |

## ABORT : Abort Sequence and Motions — System Control

**Execution Mode** Immediate (MODE = 0 Only) and Sequence

**Syntax** ABORT

**Description** The ABORT command will stop the execution of a sequence.
Commanding sequence ABORT while the motor is running will stop any sequence execution and cause the motor to come to a stop based on the MSTOPACT (Motor Stop Action) setting.

**See Also** <ESC>, ALMACT, HSTOP, MSTOP, MSTOPACT, PSTOP, SSTOP

**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) END` | #End the sequence |
| `>RUN 9` | #Execute sequence #9 |
| `>ABORT` | #Abort sequence execution and decelerate the motor to a stop |
| `>` | |

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

**Execution Mode** Immediate (MODE = 0 Only)

**Syntax** ABORTLV n

**Range** n = 0: Normally Open
1: Normally Closed

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**Description** Sets the active level for the dedicated ABORT Input.

**See Also** <ESC>, ABORT, ALMACT, HSTOP, MSTOP, MSTOPACT, PSTOP, STARTACT, SSTOP

**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.` | |

```
------------------------------------------
    AS-One (ASX66)
      Integrated Motor
    Software Version: *.**
       Copyright 2004
   ORIENTAL MOTOR CO., LTD.
------------------------------------------
```

| Command | Description |
|---|---|
| `>ABORTLV` | #Confirm ABORT input logic level |
| `ABORTLV=1(1)` | |
| `>` | |

## ALARMLV : ALARM Output Level — I/O

**Execution Mode** Immediate

**Syntax** ALARMLV n

**Range** n = 0: Normally Open
1: Normally Closed

**Initial Value** 0

**SAVEPRM and 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) value.

**Access** READ and WRITE

**Description** Sets the active level of the ALARM output, if used.
The ALARM Output will switch to the opposite state of the normal setting when an alarm condition occurs. For instance, when the ALARMLV=1 (Normally Closed) and the device is in an alarm state, the ALARM output will change to an open level (Normally Open).
Until updated with a new value or the execution of the INITPRM command, the output logic will remain stored in nonvolatile memory.

**See Also** SIGALARM, OUTALARM, OUTSG, ALM, ALMCLR

**Example**

| Command | Description |
|---|---|
| `>ALARMLV=1` | #Set the ALARM Output as Normally Closed |
| ` ALARMLV=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.` | |

```
------------------------------------------
     AS-One (ASX66)
       Integrated Motor
     Software Version: *.**
        Copyright 2004
    ORIENTAL MOTOR CO., LTD.
------------------------------------------
```

| Command | Description |
|---|---|
| `>ALARMLV` | #Query the current ALARMLV setting |
| ` ALARMLV=1 (1)` | #Device response |
| `>` | |

## ALM : Alarm Status and History — Monitor Commands

**Execution Mode** Immediate

**Syntax** ALM

**Range** 00 to F2

**Initial Value** 00

**Access** READ

**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 Chapter 7 "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.

**See Also** ALARMLV, ALMACT, ALMCLR, ALMMSG, ALMSET, CURRENT, OUTALARM

**Example**

| Command | Description |
|---|---|
| >CRSTOP 0 | #Set the STOP current to 0% of the rated current (100%) |
| CRSTOP=0 | #Device response |
| > | |

```
'The motor output shaft is physically moved out of position by hand'
>ALM                  #Query the current ALARM code
  ALARM =30 ,  RECORD : 30 23 9A 23 68 68 66 60 66 66

  ALM_OVER_LOAD , 4.222 [sec] past.

  WARNING =01 ,  RECORD : 01 00 00 00 00 00 00 00 00 00

  WRN_OVERLOAD , 9.309 [sec] past.
>
```

## ALMACT : Alarm Action — System Control

**Execution Mode** Immediate

**Syntax** ALMACT = n

**Range** n = 0: Continue operations (ALARM OFF)
1: Abort sequences and stop motion. Motor current remains ON (ALARM ON)
2: Abort sequences and stop motion. Turn Motor Current OFF (ALARM ON)

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**Description** Establishes the motor current response and alarm state after a PSTOP operation, or after Over Position Error, Over Load, Over Velocity, or hardware or software overtravel errors.
If ALMACT=0, Sequence operation will continue after a PSTOP command or input.

**See Also** ALARMLV, ALM, ALMCLR, OUTALARM, PSTOP

**Example**

| 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.` | |

```
------------------------------------------
    AS-One (ASX66)
      Integrated Motor
    Software Version: *.**
       Copyright 2004
   ORIENTAL MOTOR CO., LTD.
------------------------------------------
```

| Command | Description |
|---|---|
| `>ALMACT` | #Query new value |
| `>ALMACT=1(1)` | |
| `>` | |

## ALMCLR　: Clear Alarm　　　　System Control

**Execution Mode** Immediate

**Syntax** ALMCLR

**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.

**See Also** ALM, ALARMLV, ALMACT, ALMMSG, ALMSET, OUTALARM, CURRENT

**Example**

| Command | Description |
|---|---|
| >ALM | #Query ALM |
| >ALMCLR | #Clear the alarm condition, if possible. |

```
>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.
>
```

**Note** Before issuing an ALMCLR command, remove the cause of the alarm. If the ALARM condition persists, the drive will enter the ALARM state again. Please see the Troubleshooting section for a description of the causes of specific ALARM codes.
Some alarm conditions cannot be cleared.　Refer to See Chapter 7, "Troubleshooting", to see which conditions can and cannot be cleared.

## ALMCLRLV : ALARM CLEAR Level I/O

**Execution Mode** Immediate

**Syntax** ALMCLRLV = n

**Range** n = 0: Normally Open
1: Normally Closed

**Initial Value** 0

**SAVEPRM and 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) value.

**Access** READ and WRITE

**Description** ALMCLRLV is the active level of the Alarm Clear (ALMCLR) input, if used.

**See Also** ALM, ALARMLV, ALMACT, ALMCLR, INALMCLR, ALMMSG, ALMSET, OUTALARM, CURRENT

**Example**

| Command | Description |
|---|---|
| `>ALMCLRLV=1` | #Set the ALMCLR input as Normally Closed |
| `ALMCLRLV=0 (1)` | #Device response |
| `>SAVEPRM` | #Save the parameter assignments |
| `(EEPROM has written 29 times)` | #Device response |
| `Enter Y to proceed, other key to cancel.` | #Device response |
| `Saving Parameters........OK.` | #Device response |
| `>RESET` | #Establish the saved parameter values |
| `>ALMCLRLV` | #Query the current ALMCLRLV setting |
| `ALMCLRLV =1 (1)` | #Device response |

## ALMMSG : Alarm Message Action — System Control

**Execution Mode** Immediate

**Syntax** ALMMSG n

**Range** n = 0: Do not automatically transmit alarm and warning messages (default)
1: Automatically transmit messages for alarms, but not warnings
2: Automatically transmit messages for alarms and warnings

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**Description** The system can automatically transmit a message when alarms or warnings are detected. ALMMSG controls what types of messages are automatically transmitted.
Warning messages are sent only if the detected warning condition is different from the last reported warning.

**See Also** ALARMLV, ALM, ALMACT, ALMCLR, ALMSET, OUTALARM

**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` | #Device response |
| `Saving Parameters........OK.` | |
| `>RESET` | #Reset the system to establish the new settings |
| `Resetting system.` | |

```
------------------------------------------
    AS-One (ASX66)
      Integrated Motor
    Software Version: *.**
       Copyright 2004
   ORIENTAL MOTOR CO., LTD.
------------------------------------------
>
```

## ALMSET : Set User Alarm — System Control

**Execution Mode** Immediate and Program

**Syntax** ALMSET

**Description** The ALMSET command allows the user to place the device in a forced Alarm State.

**See Also** ALARMLV, ALM, ALMACT, ALMCLR, ALMMSG, OUTALARM

**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 |
| `>RUN CHKINPUT` | #Run sequence CHKINPUT |
| `>Motion succeeded` | #Successful |
| `>RUN CHKINPUT` | #Run again |
| `>Illegal sensor input entry!` | #Sequence aborted |
| `>ALM` | #Check alarm |

```
  ALARM =E0 ,  RECORD : E0 30 23 9A 23 68 68 66 60 66

  ALM_USR_ALARM , 12.887 [sec] past.

  WARNING =01 ,  RECORD : 01 02 01 02 01 02 01 02 01 02

  WRN_OVERLOAD , 745.35 [sec] past.
```

| Command | Description |
|---|---|
| `>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　　User Variables

**Execution Mode**　Immediate and Program

**Syntax**　{A|B. . . Y|Z} = n
In sequence only: {A|B. . . Y|Z} = expression
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)
expression must evaluate to a value within the same range as n, and can be any of:
- constant numeric value
- any variable available to sequences
- math expression

**Initial Value**　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 default (initial) value.

**Access**　READ and WRITE

**Description**　General purpose numeric variables.
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 (CRRUN=CRSTOP+I)
· Targets for interactive data entry commands (X=KBQ)

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.

A sequence will not show the name of the variable (A – Z) when the value is displayed to the terminal. The reason for this operation is to reduce the amount of ASCII information sent out of the device to an external host controller or terminal.
For example:
Sequence 1
( 1) A=2　　#Set the value of variable A
( 2) A　　　#Display the value of A

When sequence 1 executes the device displays the following:
```
>
  2          #Device response to line 2 (shown above)
>
```

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.
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.

**See Also**　CLEARVAR, CREATEVAR, DELETEVAR, N_xxx, POS [x], S_xxx, SAVEALL, SAVEPRM, VIEW, SAS, SACS

**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 |

## \ : Global Command　　Communications

**Execution Mode**　Immediate

**Syntax**　\ (Command)

**Description**　Global command operator. Attaching this operator before the command enables command to all the units. "\ID" is for checking all devices assigned ID numbers active on the daisy chain communication network.

Applicable Commands:
ABORT, CONT, CURRENT, CV, EHOME, HSTOP, ID, MA, MCN, MCP, MGHN, MGHP, MI, MIx, MSTOP, PAUSE, PAUSECLR, PSTOP, RESET, RUN, SSTOP

**See Also**　@, ID, TALK, VERBOSE

**Example**

| Command | Description |
|---|---|
| `2>\ID` | #Send the Global ID query command to all devices |
| `3` | #Device response |
| `1` | |
| `2` | |
| `0` | |
| `2>` | |

**Note**　Support for \CURRENT is limited. \CURRENT=0 and \CURRENT=1 are supported (globally set current ON and OFF, respectfully), but \CURRENT, as a query (no arguments) generates no response.

## BAUD : Communication Baud Rate — Communications

**Execution Mode** Immediate and Sequence

**Syntax** BAUD n

**Range** n = 0: 9600 (bits per second)
1: 19200
2: 38400

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE
READ only in sequences

**Description** Establishes the RS232 Communication BAUD Rate for the device

**See Also** @, ECHO, ID, TALK, VERBOSE

**Example**

| Command | Description |
|---|---|
| `>BAUD 1`<br>`BAUD=0(1) [9600bps]` | #Set the Baud Rate to 19200 Bits per second (bps) |
| `>SAVEPRM`<br>`(EEPROM has been written 21 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>`AS-One (ASX66)`<br>`Integrated Motor`<br>`Software Version: *.**`<br>`Copyright 2004`<br>`ORIENTAL MOTOR CO., LTD.`<br>`------------------------------------------` | #Reset the system to establish the new baud value<br>#NOTE: change baud rate of host system before proceeding! |
| `>BAUD`<br>`BAUD=1(1) [19200bps]` | #Query the Baud Rate<br>#Baud is set as 19200 |

**Note 1** When using a terminal emulator, such as Microsoft® HyperTerminal, a new session is needed for communicating at the higher speed. Within Microsoft® HyperTerminal, follow these procedures to connect at the higher speed.

1) Disconnect the current session
   a. Select the CALL menu then click DISCONNECT
2) Establish the new Baud rate in the properties menu
   a. Select the FILE menu then Properties
   b. Click CONFIGURE
   c. Change the BITS PER SECOND field to the value established with the BAUD command.
   d. Click OK and OK again
   e. Select the CALL menu then click CALL

If the CLEARALL command is issued, the baud rate will be reset to 9600 bps.

**Note 2** 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 10 feet, a high baud rate (greater than 9600 bps) may not operate properly because of communication signal deterioration over the line.
All units in a daisy chain configuration must have the same BAUD setting.

## BREAKL : Break LOOP Block — Sequence Commands

**Execution Mode** Program

**Syntax** BREAKL

**Description** Exits the innermost LOOP block. Often used to exit a LOOP based on the value of a conditional statement.

**See Also** BREAKW, ELSE, ENDIF, ENDL, IF, LOOP, WEND, WHILE

**Example**

| Command | Description |
|---|---|
| `>LIST 7` | #List sequence 7 |
| `(  1) LOOP` | #Loop indefinitely |
| `(  2)   IF (IN2=1)` | #If INPUT2 is 1 (ON), the sequence proceeds to line 3. |
| `(  3)     BREAKL` | #Exit the loop and execute the line after the ENDL command |
| `(  4)   ELSE` | #Branch here if not true |
| `(  5)     SAS HELLO` | #Send HELLO via the ASCII Communication port |
| `(  6)   ENDIF` | #End the IF statement |
| `(  7) ENDL` | #End the loop and return to the beginning of the loop at line 1 |
| `(  8) END` | #End the sequence |
| `>` | |

## BREAKW : Break WHILE Block — Sequence Commands

**Execution Mode** Sequence

**Syntax** BREAKW

**Description** Exits the innermost WHILE block. Often used to exit a WHILE block based on the value of a conditional statement.

**See Also** BREAKL, ELSE, ENDIF, ENDL, IF, LOOP, WEND, WHILE

**Example**

| Command | Description |
|---|---|
| `>LIST 8` | #List sequence 8 |
| `(  1) WHILE (IN1=0)` | #Start WHILE block. Execute lines 2 through 4 while condition is true |
| `(  2)   IF (IN2=1)` | #If IN2 is 1 (ON), execute line 3 |
| `(  3)     BREAKW` | #Exit the WHILE loop and execute the line after the WEND command |
| `(  4)   ENDIF` | #End the IF block |
| `(  5) WEND` | #End the WHILE block, return to line 1 |
| `(  6) END` | #End the sequence |
| `>` | |

## CALL　: Call Sequence as Subroutine　　Sequence Commands

**Execution Mode**　Sequence

**Syntax**　CALL n

**Range**　n = Valid sequence name or number, or variable.

**Description**

Executes a sequence as a subroutine, then returns to the calling sequence.
If n is a variable name (e.g. CALL Q), then Q must be equal to a valid sequence number.
Calling sequences by name can make sequences more readable, but requires an internal name lookup operation.　That operation takes an unpredictable amount of time, which depends on system activity and the number of sequences that have been programmed.
Calling sequences by number is fast and always executes in the same elapsed time, but is less readable.
Calling by variable is just slightly slower than calling by number, and always executes in the same elapsed time.　Calling by variable should only be used if necessary, to avoid calling the wrong (or a nonexistent) sequence.
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.

**See Also**　DIR, RET

**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. |
| `>` | |

## CLEARALL : Clear All Programming (return to factory condition) System Control

**Execution Mode** Immediate (MODE = 0 Only)

**Syntax** CLEARALL

**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** **Use caution when clearing all parameter values, position array data, and sequences. Once the information is cleared it cannot be restored.**
**The CLEARALL command writes to EEPROM. The EEPROM has a nominal expected lifetime of 100,000 write cycles. The CLEARALL command should not be used automatically (i.e. by a host controller) if it could possibly execute at high frequency.**

**See Also** CLEARPOS, CLEARSEQ, CLEARVAR, INITPRM

**Example**

| Command | Description |
|---|---|
| >CLEARALL | #Initialize all parameters, clear all position array data and sequences |

```
>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.
>
```

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

**Execution Mode** Immediate (MODE = 0 Only)

**Syntax** CLEARPOS

**Description** Clears all POS [x] position array data. Position data will set to 0.

**Caution** **Use caution when clearing position array data. Once the data points are cleared, they cannot be restored.**
**The CLEARPOS command writes to EEPROM. The EEPROM has a nominal expected lifetime of 100,000 write cycles. The CLEARPOS command should not be used automatically (i.e. by a host controller) if it could possibly execute at high frequency.**

**See Also** CLEARALL, CLEARSEQ, CLEARVAR, INITPRM, TEACH

**Example**

| Command | Description |
|---|---|
| `>CLEARPOS` | #Clear all position array data to 0 |

```
>CLEARPOS
(EEPROM has been written 13 times)
 Enter Y to proceed, other key to cancel. y
 Clear POS[ ] Data.....OK.
>
```

## CLEARSEQ : Clear sequences — Sequence Management

**Execution Mode** Immediate (MODE = 0 Only)

**Syntax** CLEARSEQ

**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** **Use caution when clearing all sequences. Once the sequences are deleted, they cannot be restored.**
**The CLEARSEQ command writes to EEPROM. The EEPROM has a nominal expected lifetime of 100,000 write cycles. The CLEARSEQ command should not be used automatically (i.e. by a host controller) if it could possibly execute at high frequency.**

**See Also** CLEARALL, CLEARPOS, CLEARVAR, DEL, EDIT

**Example**

| Command | Description |
|---|---|
| >DIR | #List all sequences |

```
  ## Name        TextSize  Locked
  == ==========  ========  ======
   0 <nameless>        10
   1 <nameless>        37

 Total:   2
 Executable memory:    43 bytes used of  2048 bytes total,     2 percent.
 Storage memory:       98 bytes used of 21775 bytes total,     0 percent.
```

| Command | Description |
|---|---|
| >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 |
| > | |

## CLEARVAR : Clear User-Defined Variables — User Variables

**Execution Mode** Immediate

**Syntax** CLEARVAR

**Description** CLEARVAR clears all user-defined variables from memory.

**Caution** **Use caution when clearing all user-defined variables. Once the variables are cleared, they cannot be restored.**

**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 default (initial) value.

**See Also** CLEARALL, CLEARPOS, CLEARSEQ, DELETEVAR, LISTVAR, INITPRM, N_xxx, S_xxx

**Example**

| Command | Description |
|---|---|
| >LISTVAR | #List all user-defined variables |
| >CLEARVAR | #Clear all user-defined variables from memory. |

```
>LISTVAR
   ##  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
 Enter Y to proceed, other key to cancel. y
 All user parameters are deleted.
>
```

## CMODE : Current Mode — System Control

**Execution Mode** Immediate and Program

**Syntax** CMODE = n

**Range** n = 0: Basic
1: Manual
2: Automatic

**Initial Value** 0

**Access** READ and WRITE while the motion is not executing.
READ only while motion is in progress

**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 default (initial) value.

**Description** CMODE selects the system current profile strategy. The motor current can be different while stopped, while accelerating, while running at constant speed, and while decelerating, depending on the value of CMODE.
The current mode can be changed for each motion profile.
CMODE cannot be changed while the motor is moving.

• Basic Setting (CMODE=0)
This is the conventional current setting. Current is defined by the stop current CRSTOP, run and acceleration and deceleration current CRRUN with the ratio (%) to the rated current setting.

• Manual Setting (CMODE=1)
Current is defined by the stop current CRSTOP, acceleration and deceleration current CRACC and current at constant speed CRRUN. If, for instance, the load requirement is low at constant speed (low friction), but high while accelerating or decelerating (high inertia or low ramp times), CRRUN could be set below CRACC, reducing power consumption, heat radiation, and possibly audible noise.

• Automatic Setting (CMODE=2)
Set motor current that meets the torque required to drive the load. The motor current is calculated internally, and the original CRRUN, CRSTOP and CRACC settings are overwritten. The actual values depend on programmed torque utilization (TU), load inertia (LI), load friction and static friction (LF, LSF), and gravity loading (LG). These parameters can be entered directly if known, or estimated by using the LDCHK command.

**Important Interactions** If CMODE = 2, the values of CRSTOP, CRACC, CRRUN, and CRDEC are automatically and continuously updated, depending on motion requirements. If CMODE is later set back to 0 or 1, CRSTOP and CRRUN (and CRACC if CMODE=1) must be reprogrammed with appropriate values. (The previous values, set with CMODE 2 for the previous motion, may not be appropriate for subsequent motions.)

**See Also** CRRUN, CRACC, CRDEC, CRSTOP, LI, LF, LG, LSF, TU

**Example**

| Command | Description |
|---|---|
| `>CMODE 1` | #Set the current control mode to Manual mode |
| `CMODE=1 [Manual]` | #Device response |
| `>CRRUN 75` | #Set constant speed run current to 75% |
| `CRRUN=75` | |
| `>CRACC 100` | #Set acceleration current to 100% |
| `CRACC=100` | #(Will also be used for deceleration, with CMODE=1) |
| `>MCP` | #Start continuous motion, positive direction |
| `>SSTOP` | #Decelerate to a stop |
| `>` | |

## CONT　: Continue Motion　　　　Motion Commands

**Execution Mode** Immediate (MODE = 0 Only) and Sequence

**Syntax** CONT

**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 linear ramping is used (RMODE=0), acceleration and deceleration times TA and TD, and start and running velocities VS and VR determine the motion profile while changing speed.　If auto-ramping is used (RMODE=1), the system automatically determines the motion profile during the speed change.
If the paused motion was a point-to-point index (MI, MA, EHOME), 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.
If the paused motion was a mechanical home seeking operation (MGHP, MGHN), a CONT command restarts the process from the beginning: CONT has the same effect as re-issuing the original MGHx command.
In all cases, the system uses the values of VS, VR, TA and TD in effect at the time the CONT command is executed.
The CONT command has no effect if motion has not been previously PAUSE'd.
If sequences are running, the START input can cause the same action as a CONT command.

**See Also** INPAUSE, INPAUSECL, OUTPSTS, PAUSE, PAUSECLLV, PAUSECLR, PAUSELV, PSTSLV, SIGPAUSE, SIGPAUSECL, SIGPSTS

**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, PAUSECLR, or new motion).
Pause and Continue operations are not supported for Linked Motions (MIx). PAUSE during a Linked Motion causes a soft stop, and subsequent CONT commands are ignored.

**Example**

| Command | Description |
|---|---|
| `>LIST CHKJAM` | #List sequence CHKJAM |
| `(  1) DIS=10; VR=10` | #Set motion parameter |
| `(  2) LOOP` | #Start infinite loop |
| `(  3)   MI` | #Start move incremental |
| `(  4)   WHILE (TF<0.5)` | #Check if over loaded |
| `(  5)     IF (SIGMOVE=0)` | #Check for motion end |
| `(  6)       BREAKW` | #Exit while loop, if so |
| `(  7)     ENDIF` | |
| `(  8)   WEND` | |
| `(  9)   IF (SIGMOVE!=0)` | #Check if moving |
| `( 10)     PAUSE` | #TF>0.5: PAUSE motion |
| `( 11)     WAIT TD` | #Wait for stop, send text, get response |
| `( 12)     SAS System in trouble.` | |
| `( 13)     SACS Enter 1 to continue, other to stop:` | |
| `( 14)     A=KBQ; SACS ^M^J>` | |
| `( 15)     IF (A=1)` | |
| `( 16)       CONT; MEND` | #CONTinue, if A=1 |
| `( 17)     ELSE` | #Otherwise, report stopped |
| `( 18)       SAS Operation stopped.` | |
| `( 19)       RET` | #Return from sequence |
| `( 20)     ENDIF` | |
| `( 21)   ENDIF` | #Send normal message |
| `( 22)   SAS Motion end, goto next.` | |
| `( 23)   WAIT 1` | #Dwell 1 second, loop back to top. |
| `( 24) ENDL` | |
| `>RUN CHKJAM` | #Execute sequence CHKJAM |
| `>Motion end, goto next.` | #Normal message |
| `>Motion end, goto next.` | #Normal message |
| `>System in trouble.` | #Over loaded message |
| `>Enter 1 to continue, other to stop:1` | #Prompt message -> Entry "1" |
| `>Motion end, goto next.` | #Normal message |
| `>Motion end, goto next.` | #Normal message |
| `>System in trouble.` | #Overloaded message |
| `>Enter 1 to continue, other to stop:2` | #Prompt message -> Entry "2" |
| `>Operation stopped.` | #Finished message (Sequence finish) |
| `>` | |

## COPY : Copy Sequence — Sequence Management

**Execution Mode** Immediate

**Syntax** COPY source target

**Range** source and target can be any valid sequence number (0−99) or name (consisting of letters or numbers, 10 character maximum, must start with a letter)

**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.

**See Also** DEL, EDIT, REN

**Example**

| Command | Description |
|---|---|
| `>COPY 1 MASTER` | #Copy Sequence #1 to sequence named MASTER |
| `>COPY REMOTE 2` | #Copy Sequence REMOTE to Sequence #2 |

## CRACC : Acceleration Current — System Control

**Execution Mode** Immediate (MODE = 0 Only) and Sequence

**Syntax** CRACC n

**Range** n = 25 to 100 (integer values), (% of Rated Current)

**Initial Value** 100

**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 default (initial) value.

**Access** READ and WRITE (when CMODE = 1)
READ only (when CMODE = 0 or 2)

**Description** Acceleration and Deceleration current for manual current mode (CMODE=1).
Acceleration current for auto current mode (CMODE=2).
Different CRACC values may be set for each motion profile. The purpose of this command is to adjust the amount of motor current used to accelerate or decelerate the motor. Adjusting the motor current will change the amount of torque available from the motor. Adjusting the motor current may also change vibration in the system.
If CMODE=0 (basic), CRRUN is used for acceleration and deceleration current.

**See Also** CMODE, CRRUN, CRSTOP, CRDEC

**Important Interactions** If CMODE = 2, the values of CRSTOP, CRACC, CRRUN, and CRDEC are automatically and continuously updated, depending on motion requirements. If CMODE is later set back to 0 or 1, CRSTOP and CRRUN (and CRACC if CMODE=1) must be reprogrammed with appropriate values. (The previous values, set with CMODE 2 for the previous motion, may not be appropriate for subsequent motions.)

**Note** Use caution when adjusting the Acceleration and Deceleration motor current. If the motor current is set too low, the motor may not be able to accelerate the load up to speed. The motor may not be able to decelerate the motor from speed to a rest position.
An alarm condition may occur if the motor is too far out of position from the commanded position due to a low motor current setting.

**Example**

| Command | Description |
|---|---|
| `>CMODE 1`<br>`CMODE=1 [Manual]` | #Set the current mode to manual mode |
| `>CRACC 75`<br>`CRACC=75` | #Set the Acceleration and Deceleration motor current to 50% of the rated motor current |
| `>SAVEPRM`<br>`(EEPROM has been written 10 times)`<br>`Enter Y to proceed, other key to cancel. y`<br>`Saving Parameters........OK.` | #Save the input assignments |
| `>RESET`<br>`Resetting system.` | #Establish the saved parameter values |

```
------------------------------------------
    AS-One (ASX66)
      Integrated Motor
    Software Version: *.**
       Copyright 2004
   ORIENTAL MOTOR CO., LTD.
------------------------------------------
>
```

## CRDEC : Deceleration Current — System Control

**Execution Mode** Immediate (MODE = 0 Only) and Sequence

**Syntax** CRDEC

**Range** 25 to 100 (integer values), (% of Rated Current)

**Initial Value** 100

**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 default (initial) value.

**Access** READ Only

**Description** Deceleration current for auto current mode (CMODE=2).
In auto current mode (CMODE=2), the system automatically determines motor current during deceleration. CRDEC reflects this value.
When CMODE=0, deceleration current is controlled by CRRUN: CRDEC is not used.
When CMODE=1, deceleration current is controlled by CRACC: CRDEC is not used.

**See Also** CMODE, CRRUN, CRSTOP, CRACC

**Example**

| Command | Description |
|---|---|
| `>CMODE 2` | #Select auto current mode |
| `CMODE=2 [Auto]` | |
| `>DIS .5` | #Set deceleration time to 0.05 seconds |
| `DIS=0.5 Rev` | |
| `>VR 10` | |
| `VR=10 Rev/sec` | |
| `>TD .05` | |
| `TD=0.05` | |
| `>MI` | #Perform an incremental motion |
| `>CRDEC` | #Check deceleration current |
| `CRDEC=55` | #System set deceleration current to 55% |
| `>TD .025` | #Set a shorter deceleration time: 0.025 seconds |
| `TD=0.025` | |
| `>MI` | #Perform another incremental motion |
| `>CRDEC` | #Check deceleration current again |
| `CRDEC=62` | #System set a higher deceleration current |
| `>` | |

## CREATEVAR : Create User-Defined Variable — User Variables

**Execution Mode** Immediate

**Syntax** CREATEVAR {N_xxx | S_xxx} {value | string}

**Range** xxx = Variable name: 1 to 10 alphanumeric characters
value | string (optional): intial 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.

**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 communication mode.

**See Also** A to Z, CLEARALL, CLEARVAR, DELETEVAR, LISTVAR, N_xxx, S_xxx

**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`<br>`New variable N_DEPTH is added.`<br>`N_DEPTH=0` | #Create user-defined numeric variable named N_DEPTH |
| `>N_DEPTH 10.02`<br>`N_DEPTH=10.02` | #Set user-defined numeric variable value |
| `>CREATEVAR S_LABEL IDLE`<br>`New variable S_LABEL is added.`<br>`S_LABEL=IDLE` | #Create user-defined string variable named S_LABEL, initialize to “IDLE” |
| `>S_LABEL RUNNING`<br>`S_LABEL=RUNNING` | #Set user-defined string variable value |
| `>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
>
```

## CROFFLV : Current Off Input Level — I/O

**Execution Mode** Immediate

**Syntax** CROFFLV = n

**Range** n = 0: Normally Open
1: Normally Closed

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**Description** Sets the active level for the CROFF input, if used.

**See Also** CURRENT, INITIO, IO, SAVEPRM, SIGCROFF

**Example**

| Command | Description |
| --- | --- |
| `>CROFFLV=1`<br>`CROFFLV=0(1)` | #Set the CROFF input logic to Normally Closed |
| `>SAVEPRM`<br>`(EEPROM has been written 14 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>`AS-One (ASX66)`<br>`Integrated Motor`<br>`Software Version: *.**`<br>`Copyright 2004`<br>`ORIENTAL MOTOR CO., LTD.`<br>`------------------------------------------` | #Reset the device to initialize the modified CROFF setting |
| `>CROFFLV`<br>`CROFFLV=1(1)`<br>`>` | #New value is active |

## CRRUN　: Run Current　　　　System Control

**Execution Mode** Immediate and Program

**Syntax** CRRUN n

**Range** n = 0 to 100 (integer values), (% of Rated Current)

**Initial Value** 100 (% of Rated Current)

**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 default (initial) value.

**Access** READ and WRITE (when CMODE=0 or 1)
READ only　(when CMODE=2)

**Description** The motor run current value is set as a percentage of the rated current. The motor current setting takes place immediately.
When CMODE=0, CRRUN controls current while accelerating, running at constant speed, and decelerating.
When CMODE=1, CRRUN controls current while running at constant speed only.
When CMODE=2, the system automatically determines current settings, including CRRUN.　CRRUN cannot be manually changed when CMODE=2.　CRRUN reflects the value selected by the system for constant speed operation.

**See Also** CROFFLV, CMODE, CRACC, CRDEC, CRSTOP, INCROFF

**Example**

| Command | Description |
|---|---|
| `>CRSTOP 25` | #The motor stop current is set to 25% of the maximum applicable current value (rated current) |
| ` CRSTOP=25` | |
| `>CRRUN 50` | #Set the motor run current to 50% of the maximum applicable current value (rated current) |
| ` CRRUN=50` | |
| `>` | |

## CRSTOP　: Stop Current　　System Control

**Execution Mode** Immediate and Program

**Syntax** CRSTOP n

**Range** n = 0 to 100 (integer values), (% of Rated Current)

**Initial Value** 50 (% Rated Current)

**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 default (initial) value.

**Access** READ and WRITE (when CMODE=0 or 1)
READ only when CMODE=2

**Description** The motor stop current value is set as a percentage of the rated current. The motor current setting takes place immediately.

**See Also** CROFFLV, CMODE, CRACC, CRRUN, CRDEC, INCROFF

**Important Interactions** If CMODE = 2, the values of CRSTOP, CRACC, CRRUN, and CRDEC are automatically and continuously updated, depending on motion requirements.　If CMODE is later set back to 0 or 1, immediately set CRSTOP and CRRUN (and CRACC if CMODE=1) to appropriate values. (The previous values, set with CMODE 2 for the previous motion, may not be appropriate for subsequent motions.)

**Example**

| Command | Description |
|---|---|
| >CRSTOP 25<br>CRSTOP=25 | #The motor stop current is set to 25% of the maximum applicable current value (rated current) |
| >CRRUN 98<br>CRRUN=98<br>> | #Set the motor run current to 98% of the maximum applicable current value (rated current) |

## CURRENT : Current On/Off — System Control

**Execution Mode** Immediate and Program

**Syntax** CURRENT n

**Range** n = 0: Motor Current is OFF
1: Motor Current is ON

**Initial Value** 1: Motor current is ON (Motor current at power up can be controlled with STRSW.)

**Access** READ and WRITE

**Description** Enables or disables the motor current.

**See Also** CROFFLV, CMODE, CRACC, CRRUN, CRDEC, CRSTOP, INCROFF, SIGCROFF, STRSW

**Note** When CURRENT=1, the actual amount of current and holding torque is controlled by the current mode (CMODE), and the values (in percent of rated current) of CRSTOP, CRRUN, and (depending on CMODE) CRACC and CRDEC.

**Example**

| Command | Description |
|---|---|
| `>CURRENT 0` | #Turn motor current off. Motor has no holding torque |
| `CURRENT=0` | |
| `>CURRENT 1` | #Turn motor current on. Motor now has holding torque |
| `CURRENT=1` | |
| `>` | |

## CV : Change Velocity — Motion Commands

**Execution Mode** Immediate (MODE = 0 Only) and Sequence

**Syntax** CV n

**Range** n = 0.001 to MAXVEL (User Units/second)

**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).
The CV command can only be used when the motor is accelerating or at running velocity. The CV command is not executable while the motor is decelerating to the final target position. If CV is attempted in communications mode while the motor is decelerating, the device will send out a warning message. If CV is attempted within a sequence while the motor is decelerating, an alarm is set (70h).
CV is only available if RMODE=0 (linear ramp mode).
Changing the running velocity via the CV command will affect the time required to complete the original commanded motion profile.

The are several other ways to change speeds while moving:
- If moving continuously by MCP, set new VR, and execute MCP again.
- If moving continuously by MCN, set new VR, and execute MCN again
- If all motion parameters are known, use linked index motions. Refer to MIx.

Use the SENSOR input with SCHGVR and SCHGPOS

**See Also** DPR, MA, MCN, MCP, MI, MIx, VR, VS, UU, MAXVEL, SCHGVR, SCHGPOS

**Important Interactions** 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`<br>` UU=MM` | #Set User Units (UU) to mm (millimeters) |
| `>VR 3`<br>` VR=3 MM/sec` | #Set the running velocity to 3 mm/second. |
| `>DIS 10`<br>` DIS=10 MM` | #Set the distance to 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 |
| `>` | |

## DEL : Delete Sequence — Sequence Management

**Execution Mode** Immediate

**Syntax** DEL target

**Range** target can be the name or number of any existing sequence.

**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.

**See Also** CLEARALL, CLEARSEQ, COPY, DIR, EDIT, LOCK, UNLOCK

**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  2048 bytes total,   1 percent.
 Storage memory:       87 bytes used of 21775 bytes total,   0 percent.
```

| Command | Description |
|---|---|
| >DEL TEST1 | #Delete the program TEST1 from memory |
| Enter Y to proceed, other key to cancel. y | |
| > | #Device response |

## DELETEVAR : Delete User-Defined Variable — User Variables

**Execution Mode** Immediate

**Syntax** DELETEVAR {N_xxx | S_xxx}

**Range** xxx = Variable name: 1 to 10 alphanumeric characters

**SAVEPRM** The entered value will execute immediately. However, SAVEPRM is required to save the parameter values in nonvolatile memory. Otherwise the parameter value is reset to the last saved value at device power up. If no new parameter value was saved, then the value is set to the default (initial) value.

**Description** Deletes a specific user-defined variable.

**See Also** CLEARALL, CLEARSEQ, CLEARVAR, N_xxx, S_xxx, CREATEVAR

**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
```

| Command | Description |
| --- | --- |
| >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
  ==  ==========  ============
   1              0                   #LOOPS is gone
   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                                #SAVEPRM required to make this
 (EEPROM has been written 17 times)     change permanent
 Enter Y to proceed, other key to cancel. y
 Saving Parameters........OK.
>
```

## DIR : Sequence Directory — Sequence Management

**Execution Mode** Immediate

**Syntax** DIR [target]

**Range** target is optional. If given, it should be a valid sequence number (0−99) or name (up to 10 alpha-numeric characters, starting with a letter).

**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.

**See Also** COPY, EDIT, REN

**Example**

```
>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  2048 bytes total,   34 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  2048 bytes total,   34 percent.
 Storage memory:      2259 bytes used of 21775 bytes total,   10 percent.
>
```

## DIRINV : Direction Invert — System Control

**Execution Mode**
Immediate and Sequence

**Syntax**
DIRINV n

**Range**
n = 0: Motor rotates in the Clockwise (CW) direction for positive distance values
1: Motor rotates in the Counter-Clockwise (CCW) direction for positive distance values

**Initial Value**
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 default (initial) value.

**Access**
READ and WRITE
READ only in sequences

**Description**
Inverts the direction of motor rotation. When using a gearhead, the direction of the gearhead output shaft may rotate in the opposite direction of the motor's rotation.

**See Also**
DIS, MA, MCN, MCP, MGHN, MGHP, MI, EHOME

**Example**

| Command | Description |
|---|---|
| `>DIRINV 1` | #Invert the motor direction |
| `DIRINV 0 (1)` | #Device response |
| `>SAVEPRM` | #Save the parameter assignments |
| ` (EEPROM has been written 21 times)` | |
| ` Enter Y to proceed, other key to cancel. y` | |
| ` Saving Parameters........OK.` | |
| `>RESET` | #Execute a RESET operation to activate the saved parameters |
| ` Resetting system.` | |

```
-------------------------------------------
    AS-One (ASX66)
      Integrated Motor
    Software Version: *.**
       Copyright 2004
   ORIENTAL MOTOR CO., LTD.
-------------------------------------------
```

| Command | Description |
|---|---|
| `>DIS 1000` | #Set the distance value |
| ` DIS=1000` | #Device response |
| `>MI` | #The motor rotates 1000 user units in the CCW direction |
| `>` | |

## DIS : Incremental Motion Distance — Motion Variables

**Execution Mode** Immediate (MODE = 0 Only) and Sequence

**Syntax** DIS n

**Range** n = −MAXPOS to +MAXPOS (User Units)
MAXPOS is determined by the current DPR value and varies when the DPR value is changed.
Query DPR or MAXPOS to determine the range of n.

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**Description** Determines the distance to be moved for the MI (move incremental) command. The sign of DIS determines the direction of motion.

**See Also** CV, DIRINV, DPR, MA, MAXPOS, MI, TA, TD, VS, VR

**Example**

| Command | Description |
|---|---|
| `>DPR` | #Query the DPR value |
| `DPR=1(1) Rev` | #Device response |
| `Position range = +/- 41943(41943)` | #Device response |
| `Velocity range = 0.001 - 83.333(83.333)` | #Device response |
| `>DIS 2000` | #Set distance to 2000 user units in the positive direction |
| `DIS=2000 Rev` | |
| `>MI` | #Execute the Index Move |
| `>DIS -2000` | #Set distance to 2000 user units in the negative direction |
| `DIS=-2000 Rev` | |
| `>MI` | #Execute the Index Move |
| `>` | |

## DISx : Linked Motion Distance or Destination — Motion Variables

**Execution Mode** Immediate (MODE=0 Only) and Sequence

**Syntax** DISx n

**Range** x = 0 to 3 (Linked Motion Profiles defined by DISx, VRx, INCABSx, and LINKx)
n = −MAXPOS to +MAXPOS (User Units)
MAXPOS is determined by the current DPR value and varies when the DPR value is changed.
Query the DPR or MAXPOS commands to determine the range of n.

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**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.

**See Also** INCABSx, MIx, LINKx, VRx

**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" link2 from link3 |
| `LINK2=0` | #Device response |
| `>MI1` | #Start the linked operation motion |
| `>` | |

## DPP : Distance Per Pulse — System Control

**Execution Mode** Immediate

**Syntax** DPP n

**Range** n = 0.001 to DPR/500 (User Units per pulse)

**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) value.

**Access** READ and WRITE

**Description** When the device is used in the Pulse Input Mode (MODE=1, 2, 3), the distance moved per input pulse received is set via the DPP command.
The number of User Units per shaft revolution is set by DPR. The number of pulses per shaft revolution is thus DPR/DPP. (This may also be affected by electronic gearing: refer to GA and GB.)

**See Also** DPR, UU, GA, GB, MODE

**Important Interactions** Because the range of DPP is governed by DPR, consider setting an appropriate value of DPR first.

**Example**

| Command | Description |
|---|---|
| `>UU mm`<br>` UU=mm` | #Set the User Units to mm (millimeters) |
| `>DPR 10.0`<br>` DPR=1(10) mm` | #Set Distance-per-Revolution to 10 mm. |
| `>DPP 0.01`<br>` DPP=0.001(0.01) mm` | #Move 0.01 user units for each input pulse received |
| `>MODE 1`<br>` MODE=0(1)` | #Set the device in 1-Pulse Input Mode |
| `>SAVEPRM`<br>` (EEPROM has been written 14 times)`<br>` Enter Y to proceed, other key to cancel. Y`<br>` Saving Parameters........OK.` | #Save the parameter assignments |
| `>RESET`<br>` Resetting system.` | #Reset the device to establish the saved parameters |

```
------------------------------------------
    AS-One (ASX66)
      Integrated Motor
    Software Version: *.**
       Copyright 2004
   ORIENTAL MOTOR CO., LTD.
------------------------------------------
```

| Command | Description |
|---|---|
| `>DPP`<br>` DPP=0.01(0.01) mm`<br>`>` | #Check programmed value |

## DPR : Distance Per Revolution — System Control

| | |
|---|---|
| **Execution Mode** | Immediate and Sequence |
| **Syntax** | DPR n |
| **Range** | n = 0.500 to 51200.000 (User Units per revolution) |
| **Initial Value** | 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 default (initial) value. |
| **Access** | READ and WRITE<br>READ only in sequences |
| **Description** | The value of DPR sets the distance per revolution in terms of User Units (mm, degrees, etc.).<br>DPR allows programming all distances, positions and velocities in terms of real world units. For instance, a lead screw with a lead of 10 millimeters per revolution may use a DPR value of 10. The User Unit (UU) value could be set to mm (millimeters). Therefore, a distance of 10 mm would equate to a DIS value of 10. The operator is now working in terms of real world units as opposed to motor revolutions or steps.<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, and maximum position error limit MAXOVERFLOW. |
| **See Also** | DPP, GA, GB, UU, MAXPOS, MAXVEL, MAXOVERFLOW |
| **Important Interactions** | The value of DPR effects *all* positions, distances, and velocities. For most applications, an appropriate value for DPR should be set *before* any motions are programmed. Changing DPR later may invalidate some motions because of range limits, and will change all physical shaft motions.<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. |
| **Note** | When DPR, GA, or GB are changed, OVERFLOW (maximum position error) and OVERVEL (maximum velocity) are automatically rescaled. OVERFLOW is set to approximately 3 motor revolutions, and OVERVEL is set to approximately 100 motor revolutions per second. MAXVEL and MAXPOS may change, programmed values for some parameters (e.g. VR, VS) may be out of range with the new scaling. See Section 4.3, "Initial Setting (User Unit)" for more detail. |

**Example**

| Command | Description |
|---|---|
| >UU mm<br> UU=mm | #Set the User Units to mm (millimeters) |
| >DPR 10<br> DPR=1(10) mm<br> OVERFLOW, OVERVEL is re-scaled to default equivalent.<br> OVERFLOW=3(30) mm<br> OVERVEL=100(1000) mm/sec<br> Position range = +/- 41943(419430)<br> Velocity range = 0.001 - 83.333(833.333) | #Set the Distance Per Revolution to 10 User Units, device responds with rescaled limits and new ranges |
| >SAVEPRM<br> (EEPROM has been written 21 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>AS-One (ASX66)<br>Integrated Motor<br>Software Version: *.**<br>Copyright 2004<br>ORIENTAL MOTOR CO., LTD.<br>------------------------------------------ | #Establish the saved parameter values |
| >MAXPOS<br> MAXPOS=419430(419430) mm | #Query the Maximum Position Value |
| >MAXVEL<br> MAXVEL=833.333(833.333) mm/sec<br>> | #Query the Maximum Velocity Value |

## DTMP　: Drive Temperature　　System Status

| | |
|---|---|
| **Execution Mode** | Immediate and Sequence |
| **Syntax** | DTMP |
| **Range** | n/a (Degrees Celsius) |
| **Access** | READ |
| **Description** | DTMP indicates the temperature measured near the device electronics, in degrees Celsius.<br>The system constantly monitors temperature near the electronics (DTMP) and near the motor windings (MTMP).　Either temperature can trigger an alarm or warning if excessive. The alarm limits are set by DTMPMAX and MTMPMAX.　Warning limits are set by DTMPWRN and MTMPWRN, and can be used to trigger an output (TEMP) if these limits are exceeded. |
| **See Also** | / (Forward slash), DTMPMAX, DTMPWRN, MTMP, MTMPMAX, MTMPWRN, OUTTEMP, TEMPLV |

**Example**

| Command | Description |
|---|---|
| >DTMPMAX 50<br> DTMPMAX=50 | #Set the drive overheat temperature protective function to activate at a value of 50 degrees Celsius. |
| >DTMPWRN 45<br> DTMPWRN=45 | #Set the device to trigger a warning when the drive temperature exceeds 45 degrees Celsius |
| >MTMPMAX 55<br> MTMPMAX=55 | #Set the Motor Temperature Maximum value |
| >MTMPWRN 50<br> MTMPWRN=50 | #Set the device to trigger a warning when the motor temperature exceeds 50 degrees Celsius |
| >MTMP<br> MTMP=35 | #Query the current motor temperature |
| >DTMP | #Query the drive temperature value |
| DTMP=42 | #Displays the current drive temperature value |
| >SIGTEMP | #Query temperature warning signal |
| SIGTEMP=0 | #SIGTEMP is zero because both drive and motor are below warning limits |
| > | |

## DTMPMAX : Maximum Driver Temperature — System Control

**Execution Mode** Immediate and Sequence

**Syntax** DTMPMAX n

**Range** n = 0 to 80 (integer values) (Degrees Celsius)

**Initial Value** 80

**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 default (initial) value.

**Access** READ and WRITE
READ only in sequences

**Description** DTMPMAX controls the temperature used to trigger the drive overheat protective function. The system will stop motion, turn motor current off, and indicate an alarm condition (alarm code: 21h) if the temperature at the drive electronics exceeds DTMPMAX.
The system monitors temperature in two locations: near the drive electronics and near the motor windings. The overheat alarm triggers if either temperature exceeds its programmed limit. MTMPMAX sets the temperature limit for the motor windings.
The system can also provide an early warning of elevated drive or motor temperature (via TEMP output), so that actions may be taken to avoid an alarm and shutdown (e.g. reduce motor current, reduce application throughput, etc.). See discussions at DTMPWRN and MTMPWRN.

**See Also** DTMP, DTMPWRN, MTMP, MTMPMAX, MTMPWRN

**Example**

| Command | Description |
|---|---|
| >DTMPMAX 50 | #Set the drive overheat temperature protective function to activate at a value of 50 degrees Celsius. |
| DTMPMAX=50 | |
| >DTMPWRN 45 | #Set the device to trigger a warning when the drive temperature exceeds 45 degrees Celsius |
| DTMPWRN=45 | |
| >MTMPMAX 55 | #Set the Motor Temperature Maximum value |
| MTMPMAX=55 | |
| >MTMPWRN 50 | #Set the device to trigger a warning when the motor temperature exceeds 50 degrees Celsius |
| MTMPWRN=50 | |
| >MTMP | #Query the current motor temperature |
| MTMP=35 | |
| >DTMP | #Query the drive temperature value |
| DTMP=42 | #Displays the current drive temperature value |
| >SIGTEMP | #Query temperature warning signal |
| SIGTEMP=0 | #SIGTEMP is zero because both drive and motor are below warning limits |
| > | |

## DTMPWRN : Drive Warning Temperature — System Control

**Execution Mode** Immediate

**Syntax** DTMPWRN n

**Range** n = 0 to 80 (integer values) (Degrees Celsius)

**Initial Value** 80

**Access** READ and WRITE

**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 default (initial) value.

**Description** DTMPWRN controls the drive temperature threshold used to control the SIGTEMP temperature warning signal, and TEMP output (if used).
The system monitors temperature in two locations: near the drive electronics and near the motor windings. The temperature warning triggers if either temperature exceeds its programmed warning limit. (MTMPWRN sets the temperature warning limit for the motor windings.)
DTMPWRN (and MTMPWRN) can be used to provide an early warning of elevated drive or motor temperature (via TEMP output), so that actions may be taken to avoid an alarm and shutdown (e.g. reduce motor current, reduce application throughput, etc.). The temperature warning feature could also be used in applications that are very temperature sensitive (e.g. motor current could be disabled and machine operation suspended until temperatures had reduced sufficiently).
SIGTEMP reflects the combined temperature warning status of both the motor and drive. SIGTEMP will be zero (0) if both drive and motor are below their limits, and one (1) if either drive or motor are above their limits. SIGTEMP can be monitored over the serial port if the TEMP output is not configured.

**See Also** DTMP, DTMPMAX, MTMP, MTMPMAX, MTMPWRN, OUTTEMP, TEMPLV, SIGTEMP

**Example**

| Command | Description |
|---|---|
| >DTMPMAX 50 | #Set the drive overheat temperature protective function to activate at a value of 50 degrees Celsius. |
| DTMPMAX=50 | |
| >DTMPWRN 45 | #Set the device to trigger a warning when the drive temperature exceeds 45 degrees Celsius |
| DTMPWRN=45 | |
| >MTMPMAX 55 | #Set the Motor Temperature Maximum value |
| MTMPMAX=55 | |
| >MTMPWRN 50 | #Set the device to trigger a warning when the motor temperature exceeds 50 degrees Celsius |
| MTMPWRN=50 | |
| >MTMP | #Query the current motor temperature |
| MTMP=35 | #Device response sent to the terminal |
| >DTMP | #Query the drive temperature value |
| DTMP=42 | #Displays the current drive temperature value |
| >SIGTEMP | #Query temperature warning signal |
| SIGTEMP=0 | #SIGTEMP is zero because both drive and motor are below warning limits |
| > | |

## DVIN　: Drive Input Voltage　　System Status

**Execution Mode** Immediate and Program

**Syntax** DVIN

**Range** n/a

**Resolution Increments** 0.1 (Volts, DC)

**Access** READ

**Description** DVIN indicates the measured drive input voltage.

**Important Interactions**

1) For proper operation, some features require that drive input voltage (DVIN) be close to nominal drive input voltage (DVINSET).　If motor current is set automatically (CMODE=2), or motion profiles are generated automatically (RMODE=1), or torque feedforward is selected (TQFF=1), a warning may be issued if DVIN is more than 10% different from DVINSET.

2) Some internal current control parameters are calculated just after system startup, based on the value of DVIN at that time.　These internal parameters do not change after startup.　If DVIN changes more than 10% from the measured startup value, a warning will be generated, and system performance may degrade. Intentionally changing DC input voltage while operating is not recommended.　Cycle system power or RESET the system after DC input power supply voltage change.

**See Also** DVINSET, ALMMSG

**Example**

| Command | Description |
|---|---|
| `>DVIN` | #Query the drive input voltage |
| `DVIN=24.7` | #Displays the current drive input voltage |
| `>` | |

## DVINSET : Nominal Drive Input Voltage — System Control

**Execution Mode** Immediate (MODE = 0 Only) and Sequence

**Syntax** DVINSET n

**Range** 12.000 to 48.000 (Volts, DC)

**Initial Value** 24

**Access** READ and WRITE
READ only while motion is in progress
READ only in sequences

**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 default (initial) value.

**Description** DVINSET indicates the nominal drive input voltage.
DVINSET should be set as close to the actual input voltage as possible.

**Important Interactions** System performance may degrade if actual drive input voltage varies significantly from the nominal drive input voltage. See discussion at DVIN.

**See Also** DVIN

**Example**

| Command | Description |
|---|---|
| >DVINSET 36<br>DVINSET=36 | #Set the Drive input voltage to 36 volts |
| >SAVEPRM<br>(EEPROM has been written 29 times)<br>Enter Y to proceed, other key to cancel. Y<br>Saving Parameters........OK. | #Save parameter assignments |
| >RESET<br>Resetting system. | #Establish the saved parameter values |

```
------------------------------------------
    AS-One (ASX66)
      Integrated Motor
    Software Version: *.**
       Copyright 2004
   ORIENTAL MOTOR CO., LTD.
------------------------------------------
```

| Command | Description |
|---|---|
| >DVIN<br>DVIN=36.1<br>> | #Query the drive input voltage<br>#Displays the current drive input voltage |

## ECHO : Communications Echo Control — Communications

**Execution Mode** Immediate

**Syntax** ECHO n

**Range** n = 0: OFF, Commands are suppressed and not shown on the terminal
1: ON, Commands are echoes back to the terminal

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**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 a 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.

**See Also** VERBOSE

**Example**

| Command | Description |
|---|---|
| `>VS` | #Query the Starting Velocity |
| ` VS=0.1 mm/sec` | #Device response |
| `>ECHO 0` | #Turn off the ECHO |
| ` ECHO=0` | #Device response |
| `>ECHO=0` | #Query ECHO setting: note actual query text not echoed back (just response) |
| `>VS=0.1 mm/sec` | #Query the Starting Velocity. Again, query doesn't show: just response. |
| `>` | |

## EDIT : Edit Sequence — Sequence Management

**Execution Mode** Immediate

**Syntax** EDIT [target]

**Range** target (optional): any valid sequence number (0–99) or name (consisting of letters and numbers, up to 10 characters, starting with a letter)

**Description** Enters the sequence editor, where sequences can be created or modified.
Every sequence must have a unique number. If [target] is unspecified, or specified as a new name, EDIT automatically assigns the lowest unused sequence number to the new sequence.
The editor uses its own prompt (>>Command:). Editing operations are performed by entering a one character command, and any relevant arguments. The editor commands are listed below: this information is also available by entering 'H' at the editor prompt ([] indicates an optional argument).
The ESCAPE character can also be used to quit the sequence editor.

| Editor Command | Description |
|---|---|
| I [x] | Insert line(s) before line x (end of sequence if no x) |
| A x [y] | Alter line(s) x, or x to y |
| D x [y] | Delete line(s) x, or x to y |
| L [x] [y] | List line(s). All, or x to end, or x to y |
| X x [y] | Cut line(s) to clipboard. x, or x to y |
| 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 |

**Important Interactions** - 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      : 2048` | #Device response |
| `>>Command:` | #<ESC> is sent to exit the Editor |
| `>` | #Back at the main system prompt |

## EHOME　: Return to Electrical Home　　Motion Commands

**Execution Mode** Immediate (MODE = 0 Only) and Sequence

**Syntax** EHOME

**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: EHOME establishes position 0 as the "home" position, and can make software limit checking possible.
When using linear ramps (RMODE=0), the EHOME motion is defined by stop velocity VS, running velocity VR, and acceleration and deceleration times TA and TD. When using automatic ramping (RMODE=1), VR determines the maximum permitted velocity: VS, TA, and TD are not used.
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 negative and positive position limits (LIMN, LIMP), 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.

**See Also** HOMETYP, LIMN, LIMP, MGHN, MGHP, PC, SLACT

**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 — Program Control

**Execution Mode** Sequence

**Syntax** ELSE

**Description** Branches to an alternate operation if the preceding conditional IF statement is not true.

**See Also** IF, ENDIF, WHILE, WEND

**Example**

| Command | Description |
|---|---|
| >LIST 5 | #List sequence 5 |
| ( 1) IF (IN1=1) | #If input #1 is on, then do line 2 |
| ( 2) VR=20 | #Running Velocity=20 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 |
| > | |

## END : End Sequence — Sequence Commands

**Execution Mode** Sequence

**Syntax** END

**Description** The END statement can be used to formally terminate sequence text. END behaves exactly the same as a return statement (RET), but END, if used, must be the last statement in the sequence. Any text following the END statement will cause a error when attempting to save the sequence.

END is provided for compatibility with other Oriental Motor products. Its use is strictly optional: a sequence does not need an END as its last statement.

**See Also** RET

**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 |
| `>` | |

## ENDIF : End of IF Block — Sequence Commands

**Execution Mode** Sequence

**Syntax** ENDIF

**Description** Indicates the completion of a conditional IF statement.

**See Also** IF, ELSE, WHILE, WEND

**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 |
| `>` | |

## ENDL : End of LOOP Block — Sequence Commands

**Execution Mode** Sequence

**Syntax** ENDL

**Description** Terminates the innermost LOOP block

**See Also** LOOP, BREAKL

**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 |
| `>` | |

## ENDLV : END Output Level — I/O

**Execution Mode** Immediate

**Syntax** ENDLV n

**Range** n = 0: Normally Open
1: Normally Closed

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**Description** Establishes the active level of the END output, if used.

**See Also** OUTEND, SIGEND

**Example**

| Command | Description |
|---|---|
| >ENDLV 1 | #Set the END input logic to the Normally Closed logic level |
| ENDLV=0(1) | |
| >SAVEPRM | #Save the parameter assignments |
| (EEPROM has been written 10 times)<br>Enter Y to proceed, other key to cancel. y<br>Saving Parameters........OK. | Device response |
| >RESET | #Establish the saved parameter values |
| Resetting system. | |

```
-------------------------------------------
    AS-One (ASX66)
      Integrated Motor
    Software Version: *.**
       Copyright 2004
   ORIENTAL MOTOR CO., LTD.
-------------------------------------------
```

| Command | Description |
|---|---|
| >ENDLV | #Query the current END input logic level |
| ENDLV=1(1) | #Device response |
| > | |

## EVx : Configure Event Output — I/O

**Execution Mode** Immediate (MODE=0 Only) and Sequence

**Syntax**

EVx OUTy = z m = n
or
EVx 0

**Range**

x: Event channel number; 1 or 2
y: Output number; 1 to 4
z: Output logic level after trigger; 0 or 1
m: Event trigger source
T: Trigger n seconds after motion start; 0.000 to 500.000 (second)
D: Trigger after moving distance n from motion start. n=0.000 to MAXPOS (User Units)
V: Trigger after reaching speed setpoint n. n=0.001 to MAXVEL (User Units/second)

**Description**

Configures events which control outputs on-the-fly. Up to 2 events can be configured and active at the same time, using both event channels 1 and 2
EVx 0 clears (deactivates) the event. Once an event has been configured, it remains active until cleared. Clearing the event does not clear or reset the output itself.
Event checking restarts at the beginning of a motion. The designated output will be set to the designated state when the designated condition has been met. To detect the transition, assure that the designated output is in the opposite state prior to the event occurring.
The output used should not have an assigned system output signal (e.g. if OUTEND=3, do not use output 3 for events). If the output has been assigned to a system output signal, no event-driven transitions will occur on the output.

**Example**

| Command | Description |
|---|---|
| `>EV1 OUT2=1 V=10`<br>` EV1 OUT2=1 V=10` | #Turn on Output#2 when reach speed of 10 User Units/second |
| `>EV2 OUT1=1 T=2`<br>` EV2 OUT1=1 T=2` | #Turn on Output#1 2 seconds after motion starts |
| `>MCP` | #Execute a continuous move in the positive direction |
| `>EV1 0; EV2 0`<br>`>` | #Clear events number 1 and 2 |

## FILT : Jerk Filter Time Lag — System Control

**Execution Mode** Immediate and Sequence

**Syntax** FILT n

**Range** n = 0.000 to 1.000 (seconds)

**Resolution Increments** 0.001

**Initial Value** 0.003

**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 default (initial) value.

**Access** READ and WRITE
READ only while motion is in progress

**Description** Jerk limiting filter time lag, in seconds. The jerk filter modifies the motion profile (all modes), smoothing out any sudden transitions. This may significantly reduce vibration in some applications, especially at the start and end of speed changes.
The jerk limiting filter introduces lag (time delay) into the system response. Higher values of FILT provide more smoothing, but also increase lag. Lower values of FILT provide less smoothing, but also less lag.
Setting FILT to 0 (zero) disables the filter.
This parameter is effective upon entry. It cannot be changed while motion is in progress.

**Important Interactions** The default value of FILT may be appropriate for most applications. The filter effect may not be noticeable.
High values of FILT may significantly increase settling time at the end of motion, and the amount of time that must pass before motions are truly finished.
Setting higher values of FILT may not be advisable in combination with other enhanced functions. For instance, torque-feedforward intentionally introduces sharp transitions to the motion profile, to compensate for expected loading conditions. These sharp transitions then get filtered (reduced) by the jerk filter. The system is working against itself. Likewise, with automatic profiling (RMODE=1), the auto-profiler is trying to find the "quickest" possible motion for the programmed conditions, and then the jerk filter smoothes the result: that may not be desirable.

**See Also** TA, TD, VR, VS

**Example**

| Command | Description |
|---|---|
| `>UU Degrees` | #Set the User Units to Degrees |
| `UU=Degrees` | #Device response |
| `>VR 100` | #Set the running speed to 100 Degrees/second |
| `VR=100 Degrees/sec` | #Device response |
| `>DIS 10` | #Set the distance to 10 Degrees |
| `DIS=10 Degrees` | #Device response |
| `>FILT 0.005` | #Set the FILT value to 0.005 seconds |
| `FILT=0.005` | #Device response |
| `>MI` | #Execute the incremental motion |
| `>` | |

## GA, GB : Electrical Gear Ratio — System Control

| | |
|---|---|
| **Execution Mode** | Immediate and Sequence |
| **Syntax** | GA n<br>GB n |
| **Range** | n = 1 to 100 (integer values) |
| **Initial Value** | 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 default (initial) value. |
| **Access** | READ and WRITE<br>READ only in sequences |
| **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 commands. The applied gear ratio equals GA/GB.<br>For example, if a ball screw with a lead of 10 mm/rev is combined with a 3:1 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. |
| **See Also** | DPR, UU, MAXVEL, OVERVEL |
| **Important Interactions** | GA and GB, along with DPR, determine the value of MAXVEL. When any of these parameters are changed:<br><br>• OVERFLOW (maximum position error) and OVERVEL (maximum velocity) are automatically rescaled. OVERFLOW is set to approximately 3 motor revolutions, and OVERVEL is set to approximately 100 motor revolutions per second. The new values become available after a SAVEPRM and RESET. (These values can be still be modified to suit the application: the automatic rescaling is intended as a convenience.)<br><br>• Programmed velocities VS, VR, VR0−3, and SCHGVR are checked for an out of range condition. A warning message will be transmitted for any velocities that are out of range, with the new values of DPR, GA, and GB.<br><br>See Section 4.3, "Initial Setting (User Unit)" for more detail. |

**Example**

| Command | Description |
|---|---|
| >UU mm<br> UU=mm | #Set User Units to mm (millimeters) |
| >DPR 10<br> DPR=1(10) mm<br> OVERFLOW, OVERVEL is re-scaled to default equivalent.<br> OVERFLOW=3(30) mm<br> OVERVEL=100(1000) mm/sec<br> Position range = +/- 41943(419430)<br> Velocity range = 0.001 - 83.333(833.333) | #Set the distance per revolution to 10 mm: system rescales OVERFLOW, OVERVEL |
| >GA 3<br> GA=1(3)<br> OVERFLOW, OVERVEL is re-scaled to default equivalent.<br> OVERFLOW=3(10) mm<br> OVERVEL=100(333.332) mm/sec | #Set the Gear Ratio Numerator to 3: system rescales again |
| >GB 1<br> GB=1(1)<br> OVERFLOW, OVERVEL is re-scaled to default equivalent.<br> OVERFLOW=3(10) mm<br> OVERVEL=100(333.332) mm/sec | #Set the Gear Ratio Denominator to 1, system rescales again |
| >MAXVEL<br> MAXVEL=83.333(277.777) 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> AS-One (ASX66)<br> Integrated Motor<br> Software Version: *.**<br> Copyright 2004<br> ORIENTAL MOTOR CO., LTD.<br>------------------------------------------- | #Establish the saved parameter values |
| >DPR<br> DPR=10(10) mm<br> Position range = +/- 419430(419430)<br> Velocity range = 0.001 - 833.333(833.333)<br>>GA<br> GA=3(3)<br>>GB<br> GB=1(1)<br>> | #Check new effective values. |

## HELP : Display Help Information — Monitor Commands

**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 | #Display the Help information |

```
--- Commands for internal profiler mode ---

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, forcing ALARM
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 MGHx (-MAXPOS - +MAXPOS[UU])

Enter [SPACE] to continue, other key to quit.   #non-space entered
>
```

## HOMELV : HOME Input Level — I/O

**Execution Mode** Immediate (MODE = 0 Only)

**Syntax** HOMELV n

**Range** n = 0: Normally Open
1: Normally Closed

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**Description** Sets the active level of the HOME input, if used.

**See Also** HOMETYP, INHOME, MGHN, MGHP, SIGHOME

**Example**

| Command | Description |
|---|---|
| `>HOMELV 1` | #Set the HOME input to Normally Closed |
| `HOMELV=0(1)` | |
| `>INHOME 3` | #Assign the HOME input to input #3 |
| `INHOME=0(3)` | |
| `>SAVEPRM` | #Save the parameter assignments |
| `(EEPROM has been written 29 times)` | |
| `Enter Y to proceed, other key to cancel. Y` | |
| `Saving Parameters........OK.` | |
| `>RESET` | #Establish the saved parameter values |
| `Resetting system.` | |

```
------------------------------------------
    AS-One (ASX66)
      Integrated Motor
    Software Version: *.**
       Copyright 2004
   ORIENTAL MOTOR CO., LTD.
------------------------------------------
```

| Command | Description |
|---|---|
| `>HOMELV` | #Query the current HOMEPLV setting |
| `HOMELV=1(1)` | #The device responds with the current setting |
| `>` | |

## HOMEPLV : Home Position Output Level — I/O

**Execution Mode** Immediate (MODE = 0 Only)

**Syntax** HOMEPLV n

**Range** n = 0: Normally Open
1: Normally Closed

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**Description** HOMEPLV sets the logical level of the HOMEP output, if used.

**See Also** HOMETYP, MGHN, MGHP, OUTHOMEP, SIGHOMEP

**Example**

| Command | Description |
|---|---|
| >HOMEPLV 1 | #Set the HOMEP output to Normally Closed |
| HOMEPLV=0(1) | |
| >OUTHOMEP 2 | #Assign the HOMEP output to output #2 |
| OUTHOMEP=0(2) | |
| >SAVEPRM | #Save the parameter assignments |
| (EEPROM has been written 29 times) | |
| Enter Y to proceed, other key to cancel. Y | |
| Saving Parameters........OK. | |
| >RESET | #Establish the saved parameter values |
| Resetting system. | |

```
------------------------------------------
    AS-One (ASX66)
      Integrated Motor
    Software Version: *.**
       Copyright 2004
   ORIENTAL MOTOR CO., LTD.
------------------------------------------
```

| Command | Description |
|---|---|
| >HOMEPLV | #Query the current HOMEPLV setting |
| HOMEPLV=1(1) | #The device responds with the current setting |
| > | |

## HOMETYP : Homing Type **System Control**

**Execution Mode** Immediate (MODE = 0 Only) and Sequence

**Syntax** HOMETYP n

**Range** n = 0 to 11 (integer values)

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE
READ only while motion is in progress

**Description** HOMETYP configures system operation when seeking a mechanical home position with the MGHN or MGHP commands. Mechanical home seeking reacts to various inputs differently, depending on HOMETYP, and according to the following table:

<table>
<tr><th rowspan="2">HOMETYP</th><th colspan="4">Home Position Indicator Signals</th></tr>
<tr><th>HOME</th><th>LSN, LSP</th><th>SENSOR</th><th>TIMING</th></tr>
<tr><td>0</td><td rowspan="4">not used</td><td rowspan="4">Required for valid home</td><td></td><td></td></tr>
<tr><td>1</td><td></td><td>Required for valid home</td></tr>
<tr><td>2</td><td>Required for valid home</td><td></td></tr>
<tr><td>3</td><td>Required for valid home</td><td>Required for valid home</td></tr>
<tr><td>4</td><td rowspan="8">Required for valid home</td><td rowspan="4">Reverse direction</td><td></td><td></td></tr>
<tr><td>5</td><td></td><td>Required for valid home</td></tr>
<tr><td>6</td><td>Required for valid home</td><td></td></tr>
<tr><td>7</td><td>Required for valid home</td><td>Required for valid home</td></tr>
<tr><td>8</td><td rowspan="4">Stop: Alarm</td><td></td><td></td></tr>
<tr><td>9</td><td></td><td>Required for valid home</td></tr>
<tr><td>10</td><td>Required for valid home</td><td></td></tr>
<tr><td>11</td><td>Required for valid home</td><td>Required for valid home</td></tr>
</table>

In HOMETYP 0–3, a limit sensor (LSN or LSP) is used as the primary home indicator.
In HOMETYP 4–11, the HOME input is used as the primary home indicator.
If SENSOR is used, the SENSOR input and the home indicator must both be active to establish the HOME position.
If TIMING is used, the motor must reach a current phase angle of zero (0) degrees while the home input is active to establish the HOME position. (Current phase angle reaches zero degrees fifty (50) times per motor revolution.)
If SENSOR and TIMING are both used, a TIMING angle must be reached while both the SENSOR input and home indicator are active, to establish a HOME position.

**See Also** INHOME, INLSN, INLSP, MGHN, MGHP, OFFSET, OUTHOMEP, INSENSOR

**Note** The SENSOR input can also be used to stop or modify continuous motions MCN and MCP. The action caused by a SENSOR input while executing MCN or MCP is determined by SENSORACT. SENSORACT does not affect the use of the SENSOR input while seeking mechanical home with MGHN or MGHP.

**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 — Motion Commands

**Execution Mode** Immediate (MODE = 0 Only) and Sequence

**Syntax** HSTOP

**Description** HSTOP stops the motor as quickly as possible.
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. The actual distance traveled during a Hard Stop depends on velocity, load, and current settings.**

**See Also** <ESC>, ABORT, MSTOP, MSTOPACT, PSTOP, SSTOP

**Note** HSTOP should be used with care. 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 |
| `>` | |

## IA : Motor Phase Current Amplitude — System Status

**Execution Mode** Immediate and Sequence

**Syntax** IA

**Range** 0.000 to 3.600 (Amps)

**Access** READ

**Description** Displays the motor current magnitude, in amps.
Nominal motor current is affected by CURRENT (which enables or disables current), and current settings CRSTOP, CRRUN, and possibly CRACC and CRDEC (depending on current mode CMODE).
Actual motor current (displayed by IA) can differ from nominal motor current. The system attempts to maintain actual current at the nominal current level, but may not be able to do so all of the time. In particular, at high speeds, actual current will tend to reduce. (The speed at which actual current begins to reduce depends on input voltage).
IA does not reflect an individual phase current, and it is not an RMS current. It is comparable to the peak amplitude of a sine wave. If the currents through the two motor phases are $Ix$ and $Iy$, then:

$$IA = \sqrt{Ix^2 + Iy^2}$$

**See Also** CRRUN, CRSTOP, CRACC, CRDEC, CMODE, CURRENT, DVIN

**Example**

| Command | Description |
|---|---|
| `>CRSTOP 50` | #Set the motor Stop current to 50% |
| `CRSTOP=50` | #Device response |
| `>IA` | #Display the motor current amplitude |
| `IA=1.608` | #Device response |
| `>` | |

## ID　: Device ID　　　　Communications

**Execution Mode** Immediate

**Syntax** ID n

**Range** n = *, 0 to 9, and A to Z (upper or lower case, not case sensitive)

**Initial Value** *

**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 default (initial) value.

**Access** READ and WRITE

**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 default value (*) 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.
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.

**See Also** @, \ (BACKSLASH), ECHO, TALK, VERBOSE, BAUD

**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 |
| `*.** / Date  Sep.1.2004` | #Version response |
| `C>` | #Prompt from device with ID=C |

## IF : Begin IF Block: execute if true — Sequence Commands

**Execution Mode** Sequence

**Syntax** IF (element1 {Conditional Operator} element2)

**Description** Conditional test and branch operation

Parenthesis 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

IF statements must be followed (at some point) by a corresponding ENDIF statement, forming an IF "block". An ELSE statement may appear within the IF block.

When executed, the conditional expression is evaluated. If it evaluates to TRUE, sequence processing proceeds to the statement following the IF. If it evaluates to FALSE, sequence processing proceeds to the statement following the next ELSE (if used) or ENDIF (if ELSE is not used).

Block structures (IF−ENDIF, WHILE−WEND, LOOP−ENDL) may be nested, to eight (8) levels deep.

**See Also** ELSE, ENDIF, WHILE, BREAKW, WEND, LOOP, BREAKL, ENDL

**Example**

| Command | Description |
|---|---|
| `>LIST 7` | #List sequence 7 |
| `(  1) IF (PC>25000)` | #Compare position to 25000 user units |
| `(  2)    SSTOP` | #If true, soft stop… |
| `(  3)    MEND` | #Wait for motion to finish |
| `(  4)    SAS End of motion` | #Transmit "End of motion". Finished. |
| `(  5) ELSE` | #Otherwise… |
| `(  6)    IF (SIGTEMP=1)` | #Check SIGTEMP temperature warning |
| `(  7)       SSTOP` | #If true, soft stop… |
| `(  8)       MEND` | #wait for motion to end |
| `(  9)       CURRENT 0` | #Turn current off |
| `( 10)       SAS Cooling` | #Transmit "Cooling". Done. |
| `( 11)    ENDIF` | #Close inner IF block |
| `( 12) ENDIF` | #Close outer IF block |
| `>` | |

## IN : General Input Status — I/O

**Execution Mode** Immediate and Sequence

**Syntax** IN

**Range** 0 to 63 (integer values)

**Access** READ

**Description** The IN command displays the current status of all the general purpose Inputs, as one integer number.
The general purpose inputs contribute to the value of IN as follows:

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

For example, if IN=14 then Input #2 (2) is ON, Input #3 (4) is ON and Input#4 is ON (8). (2+4+8=14)
To check the status of a single general input, use the INx command.

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

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

**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 |
| `(  8)   WEND` | #End the WHILE loop |
| `(  9) ENDIF` | #End the IF block |
| `>` | |

## INALMCLR : ALARM CLEAR Input I/O

**Execution Mode** Immediate

**Syntax** INALMCLR = n

**Range** n = 0 to 6

**Initial Value** 0 (Unassigned to an input)

**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) value.

**Access** READ and WRITE

**Description** INALMCLR assigns the ALMCLR (Alarm Clear) system input signal to one of the input pins. This signal can be assigned to any of the 6 inputs when MODE=0. ALMCLR is assigned to Input 5 in pulse input mode (MODE=1 to 3); INALMCLR cannot be read or written in those modes.

The active level of the ALMCLR input is determined by ALMCLRLV.

The ALMCLR command performs the same function as an ALMCLR input.

**See Also** ALM, ALARMLV, ALMACT, ALMCLR, ALMCLRLV, ALMMSG, ALMSET, IN, INITIO, INSG, INx, INxLV, IO

**Interactions** Modifies: REPORT
Modified by: CLEARALL, INITIO, VERBOSE

**Example**

| Command | Description |
|---|---|
| `>INALMCLR=4` | #Assign the ALMCLR input to Input # 4 |
| `INALMCLR=0 (4)` | #Device response |
| `>SAVEPRM` | #Save the parameter assignments |
| `(EEPROM has written 29 times)` | #Device response |
| `Enter Y to proceed, other key to cancel.` | #Device response |
| `Saving Parameters........OK.` | #Device response |
| `>RESET` | #Establish the saved parameter values |

## INCABSx : Linked Move Type — Motion Variables

**Execution Mode** Immediate (MODE = 0 Only) and Sequence

**Syntax** INCABSx n

**Range** x = 0 to 3 (Linked Motion Profiles defined by DISx, INCABSx, VRx)
n = 0: Absolute
1: Incremental

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**Description** INCABSx determines whether DISx represents a distance or an absolute destination for linked index (MIx) motion commands.

**See Also** DISx, VRx, LINKx, MIx, TA, TD, VS

**Important Interactions** Each of the four links can be incremental or absolute. Incremental and absolute can be used in combination, but all links executed together must move in the same direction.
- For incremental links, motion direction is determined by the arithmetic sign of DIS.
- For absolute links, motion direction is determined by the motor position at the start of that motion link. Generally, absolute links are not recommended when the motor position before linked operation cannot be predicted.

**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" link2 from link3 |
| `LINK2=0` | #Device response |
| `>MI1` | #Start the linked operation motion |
| `>` | |

## INCROFF : Current Off Signal Input Assignment — I/O

**Execution Mode** Immediate (MODE = 0 Only)

**Syntax** INCROFF n

**Range** n = 0 to 6

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**Description** INCROFF assigns the CROFF (Motor Current Off) system input signal to one of the input pins.
This signal can be assigned to any of the 6 inputs when MODE=0. CROFF is assigned to Input 4 in pulse input modes (MODE=1 to 3) and INCROFF cannot be read or written in those modes.
If configured, this input causes motor current to turn OFF when the signal becomes active (CURRENT=0). When the input goes to the inactive state, motor current may turn ON (CURRENT=1). Alarm conditions may prevent current from turning on.
The active level of the CROFF signal is determined by CROFFLV.
If CROFF becomes active while a motion is in progress, the motor freewheels (no torque). No attempt to stop is made: current is immediately turned off and no motor torque is available.

**See Also** SIGCROFF, CROFFLV, INSG, CURRENT

**Example**

| Command | Description |
|---|---|
| >INCROFF 2 | #Assign the CROFF input to Input # 2 |
| INCROFF=0(2) | |
| >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. | |

```
-------------------------------------------
    AS-One (ASX66)
      Integrated Motor
    Software Version: *.**
       Copyright 2004
   ORIENTAL MOTOR CO., LTD.
-------------------------------------------
```

| Command | Description |
|---|---|
| >INCROFF | #Confirm new values |
| INCROFF=2(2) | |
| > | |

## INHOME : HOME Signal Input Assignment I/O

**Execution Mode** Immediate (MODE = 0 Only)

**Syntax** INHOME n

**Range** n = 0 to 6

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**Description** INHOME assigns the HOME (Home position indicator) system input signal to one of the input pins.
This signal can be assigned to any of the 6 inputs when MODE=0. It is not used in pulse input modes (MODE=1 to 3) and INHOME cannot be read or written in those modes.
If configured, this input indicates "at home position" when active.
The active level of the HOME input is determined by HOMELV

**See Also** SIGHOME, HOMELV, HOMETYP, INSG, HOMETYP, MGHN, MGHP

**Example**

| Command | Description |
|---|---|
| `>INHOME 3` | #Assign INPUT number 3 as the HOME input |
| `INHOME=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.` | |

```
-------------------------------------------
    AS-One (ASX66)
      Integrated Motor
    Software Version: *.**
       Copyright 2004
   ORIENTAL MOTOR CO., LTD.
-------------------------------------------
```

| Command | Description |
|---|---|
| `>INHOME` | #Confirm new value |
| `INHOME=3(3)` | |
| `>` | |

## INITIO : Initialize I/O — I/O

**Execution Mode** Immediate (MODE = 0 Only)

**Syntax** INITIO

**Description** Cancels all Input or Output assignments.
All system input signal assignment values (INCROFF, INHOME, etc) and all system output signal assignment values (OUTEND, OUTHOMEP, etc) are set to zero (0), unassigned.
All Inputs and Outputs are reset for general purpose use.
The command must be confirmed before it executes. SAVEPRM and RESET are then required for this command to take effect. 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.).

**See Also** IN, INSG, INx, IO, OUT, OUTSG, OUTTEST, OUTx,

**Example**

| Command | Description |
|---|---|
| >INITIO | #Reset the current IO assignment to factory settings |
| Enter Y to proceed, other key to cancel. Y | #Device response |
| 5(0) | |
| 0(0) | |
| 0(0) | |
| 0(0) | |
| 4(0) | |
| 1(0) | |
| 3(0) | |
| 2(0) | |
| 6(0) | |
| 0(0) | |
| 0(0) | |
| 0(0) | |
| 0(0) | |
| 0(0) | |
| 0(0) | |
| 0(0) | |
| 0(0) | |
| All I/O configurations are set to factory default. | #Device response |
| Execute SAVEPRM then RESET to activate new settings. | #Device response |
| > | |

## INITPRM : Initialize Parameters — System Control

**Execution Mode** Immediate

**Syntax** INITPRM

**Description** Reprograms all parameters to the original factory default setting.
Execute a RESET command after INITPRM to activate the default settings.
INITPRM cannot be executed while the motor is moving or a sequence is executing.

**Caution** **When parameters are initialized to factory default settings, all previous values are lost.**

**The INITPRM command writes to EEPROM. The EEPROM has a nominal expected lifetime of 100,000 write cycles. The INITPRM command should not be used automatically (i.e. by a host controller) if it could possibly execute at high frequency.**

**See Also** CLEARALL, CLEARPOS, CLEARSEQ, INITIO

**Example**

| Command | Description |
|---|---|
| `>INITPRM` | #Reset all of the motion parameters to default values |
| `(EEPROM has been written 45 times)` | |
| `Enter Y to proceed, other key to cancel. y` | #Once confirmed, memory overwritten, old values lost. |
| `Initializing Parameters..OK.` | |
| `>RESET` | #Reset required to activate new factory default settings. |
| `Resetting system.` | |

```
------------------------------------------
    AS-One (ASX66)
      Integrated Motor
    Software Version: *.**
       Copyright 2004
   ORIENTAL MOTOR CO., LTD.
------------------------------------------
>
```

#Ready

## INLSN, INLSP : Limit Switch Negative & Positive Signal Input Assignments I/O

**Execution Mode** Immediate (MODE = 0 Only)

**Syntax** INLSN n
INLSP n

**Range** n = 0 to 6

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**Description** INLSN and INLSP assign the LSN (Limit sensor: negative) and LSP (Limit sensor: positive) system input signals to input pins.
These signals can be assigned to any of the 6 inputs when MODE=0. LSN is assigned to Input 2 and LSP is assigned to Input 1 in pulse input modes (MODE=1 to 3); INLSN and INLSP cannot be read or written in those modes.
If assigned, these signals act as position limits (end-of-travel indicators). If either input is detected in the active state, the hardware overtravel action is triggered. Motion stops. If MODE=0, stop action (hard stop or soft stop) is configured with OTACT. In Modes 1−3, stop action is always hard stop. Further actions are determined by ALMACT.
If both inputs become active simultaneously, an alarm is triggered (Alarm 60h). Motion stops and sequences abort.
The active level of both the LSN and LSP inputs is determined by OTLV.
The system responds to limit sensors differently during mechanical home seeking: refer to "Mechanical Home Seeking" in Chapter 4 for more detail. For HOMETYP values 0 to 3, LSN and LSP must both be assigned to inputs before mechanical home seeking is possible (MGHN, MGHP).

**See Also** ALM, ALMACT, ALMCLR, INSG, IO, OTLV, SIGLSN, SIGLSP

**Example**

| Command | Description |
|---|---|
| `>INLSN 1` | #Sets the LSN input to input #1 |
| `INLSN=0(1)` | #Device response |
| `>INLSP 2` | #Set the LSP input to input #2 |
| `INLSP=0(2)` | #Device response |
| `>SAVEPRM` | #Save the parameter assignments |
| `(EEPROM has been written 29 times)` | #Device response |
| `Enter Y to proceed, other key to cancel. Y` | #Device response |
| `Saving Parameters........OK.` | |
| `>RESET` | #Establish the saved parameter values |
| `Resetting system.` | |

```
-------------------------------------------
    AS-One (ASX66)
      Integrated Motor
    Software Version: *.**
       Copyright 2004
   ORIENTAL MOTOR CO., LTD.
-------------------------------------------
```

| Command | Description |
|---|---|
| `>INLSN` | #Confirm the LSN and LSP assignments |
| `INLSN=1(1)` | |
| `>INLSP` | |
| `INLSP=2(2)` | |
| `>` | |

## INMSTOP : Motor Stop Signal Input Assignment I/O

**Execution Mode** Immediate (MODE = 0 Only)

**Syntax** INMSTOP n

**Range** n = 0 to 6

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**Description** INMSTOP assigns the MSTOP (Motor Stop) system input signal to one of the input pins.
This signal can be assigned to any of the 6 inputs when MODE=0. It is not used in pulse input modes (MODE=1 to 3) and INMSTOP cannot be read or written in those modes.
If configured, the motor will stop when the signal becomes active. Stop behavior (soft stop or hard stop) is determined by MSTOPACT.
The active level of the MSTOP input is determined by MSTOPLV. Motion can be started, even with the MSTOP input active: the transition from inactive to active state triggers MSTOP action.
An MSTOP input does not abort sequences.
The MSTOP command performs the same function as an MSTOP input.

**See Also** IO, MSTOP, MSTOPACT, MSTOPLV, SIGMSTOP, INSG

**Example**

| Command | Description |
|---|---|
| `>INMSTOP 1`<br>` INMSTOP=0(1)` | #Assign the MSTOP input to Input # 1 |
| `>SAVEPRM`<br>` (EEPROM has been written 2 times)`<br>` Enter Y to proceed, other key to cancel. Y`<br>` Saving Parameters........OK.` | #Save the parameter assignments |
| `>RESET`<br>` Resetting system.` | #Establish the saved parameter values |

```
------------------------------------------
    AS-One (ASX66)
      Integrated Motor
    Software Version: *.**
       Copyright 2004
   ORIENTAL MOTOR CO., LTD.
------------------------------------------
```

| Command | Description |
|---|---|
| `>INMSTOP`<br>` INMSTOP=1(1)`<br>`>` | #Confirm new value |

## INPAUSE : PAUSE Signal Input Assignment I/O

**Execution Mode** Immediate (MODE = 0 Only)

**Syntax** INPAUSE n

**Range** n = 0 to 6

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**Description** INPAUSE assigns the PAUSE (Pause Motion) system input signal to one of the input pins.
This signal can be assigned to any of the 6 inputs when MODE=0. It is not used in pulse input modes (MODE=1 to 3) and INPAUSE cannot be read or written in those modes.
If configured, the motor will stop when the signal becomes active (soft stop), but stay prepared to resume the same motion, if a CONT (continue) command is executed. A START input will also resume the motion. A PAUSECL (Pause Clear) input or PAUSECLR command clears the pause condition, effectively "forgetting" the remainder of the previously paused motion.
The active level of the PAUSE input is determined by PAUSELV. Motion can be started, even with the PAUSE input active: the transition from inactive to active state triggers PAUSE action.

A PAUSE input does not pause or suspend sequences.

While motion is PAUSE'd, the system output signal PSTS (Pause Status) is true (1), and, if configured, the PSTS output is active. The signal becomes false and the output inactive if the motion is continued or the pause condition cleared.

The PAUSE command performs the same function as a PAUSE input.

**See Also** CONT, IN, INxLV, INPAUSECL, IO, PAUSE, PAUSECLLV, PAUSELV, OUTPSTS, SIGPAUSE, SIGPAUSECL

**Example**

| Command | Description |
|---|---|
| >INPAUSE 6 | #Assign the PAUSE input to Input # 6 |
| INPAUSE=0(6) | |
| >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. | |

```
-------------------------------------------
    AS-One (ASX66)
      Integrated Motor
    Software Version: *.**
       Copyright 2004
   ORIENTAL MOTOR CO., LTD.
-------------------------------------------
```

| Command | Description |
|---|---|
| >INPAUSE | #Confirm new value |
| INPAUSE=6(6) | |
| > | |

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

**Execution Mode** Immediate (MODE = 0 only)

**Syntax** INPAUSECL n

**Range** n = 0 to 6

**Initial Value** 0 (Unassigned to a general input)

**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) value.

**Access** READ and WRITE

**Description** INPAUSECL assigns the PAUSECL (Clear Pause) system input signal to one of the input pins.
This signal can be assigned to any of the 6 inputs when MODE=0. It is not used in pulse input modes (MODE=1 to 3) and INPAUSECL cannot be read or written in those modes.
If configured, a PAUSE condition will be cleared when the signal becomes active. If a motion had been PAUSE'd (by a PAUSE input or PAUSE command), the remainder of that motion is "forgotten": the motion cannot be continued. When a PAUSE condition is cleared, system output signal SIGPSTS becomes false (0). If configured, the PSTS output becomes inactive.
The active level of the PAUSE input is determined by PAUSELV. Motion can be started, even with the PAUSE input active: the transition from inactive to active state triggers PAUSE action.

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

**Example**

| Command | Description |
|---|---|
| `>INPAUSECL 4`<br>` INPAUSECL=0(4)` | #Assign the PAUSECL input to Input #4 |
| `>SAVEPRM`<br>` (EEPROM has been written 2 times)`<br>` Enter Y to proceed, other key to cancel. Y`<br>` Saving Parameters........OK.` | #Save the parameter assignments |
| `>RESET`<br>` Resetting system.` | #Establish the saved parameter value |

```
-------------------------------------------
     AS-One (ASX66)
       Integrated Motor
     Software Version: *.**
        Copyright 2004
    ORIENTAL MOTOR CO., LTD.
-------------------------------------------
```

| Command | Description |
|---|---|
| `>INPAUSECL`<br>` INPAUSECL=4(4)`<br>`>` | #Confirm new value |

## INPSTOP : Panic Stop Signal Input Assignment I/O

**Execution Mode** Immediate (MODE = 0 Only)

**Syntax** INPSTOP n

**Range** n = 0 to 6

**Initial Value** 0 (Unassigned to a general input)

**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) value.

**Access** READ and WRITE

**Description** INPSTOP assigns the PSTOP (Panic Stop) system input signal to one of the input pins.
This signal can be assigned to any of the 6 inputs when MODE=0. PSTOP is assigned to Input 4 in pulse input modes (MODE=1 to 3); INPSTOP cannot be read or written in those modes.

If configured, the system will stop the motor as quickly as possible (hard stop) when the PSTOP signal becomes active, and possibly set an alarm or disable motor current (depending on the value of ALMACT).

The active level of the PSTOP input is determined by PSTOPLV.

The PSTOP command performs the same function as a PSTOP input.

**See Also** <ESC>, ABORT, ALMACT, HSTOP, INMSTOP, IN, INxLV, INSG, IO, MSTOPACT, MSTOPLV, PSTOP, PSTOPLV, SIGPSTOP, SSTOP

**Note** The actual distance traveled between executing a PSTOP and actually stopping depends on velocity, load, and current settings.

**Example**

| Command | Description |
|---|---|
| `>INPSTOP 2`<br>`INPSTOP=0(2)` | #Assign the PSTOP input to Input #2 |
| `>SAVEPRM`<br>`(EEPROM has been written 2 times)`<br>`Enter Y to proceed, other key to cancel. Y`<br>`Saving Parameters........OK.` | #Save the parameter assignments |
| `>RESET`<br>`Resetting system.` | #Establish the saved parameter value |

```
-----------------------------------------
    AS-One (ASX66)
      Integrated Motor
    Software Version: *.**
       Copyright 2004
   ORIENTAL MOTOR CO., LTD.
-----------------------------------------
```

| Command | Description |
|---|---|
| `>INPSTOP`<br>`INPSTOP=2(2)`<br>`>` | #Confirm new value |

## INSENSOR : SENSOR Signal Input Assignment I/O

**Execution Mode** Immediate (MODE = 0 Only)

**Syntax** INSENSOR n

**Range** n = 0 to 6

**Initial Value** 0 (Unassigned to a general input)

**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) value.

**Access** READ and WRITE

**Description** INSENSOR assigns the SENSOR (Sensor) system input signal to one of the input pins.
This signal can be assigned to any of the 6 inputs when MODE=0. It is not used in pulse input modes (MODE=1 to 3) and INSENSOR cannot be read or written in those modes.
If configured, a Sensor action will be triggered when the signal becomes active, if the motor is moving continuously (MCP or MCN). The action may be a hard stop, a soft stop, or a change in motion, depending on the value of SENSORACT. Refer to "Continuous Motions" in Section 4.4 for more information.
The SENSOR input has no affect on index motions (MI, MA, EHOME).
The SENSOR input has a special use during mechanical home seeking. For some types of homing operations, INSENSOR must be assigned. See the HOMETYP entry, and "Mechanical Home Seeking" in Chapter 4, for more information.
The active level of the SENSOR input is determined by SENSORLV.

**See Also** IN, INxLV, IO, MGHN, MGHP, SCHGPOS, SCHGVR, SENSORACT, SENSORLV

**Example**

| Command | Description |
|---|---|
| `>INSENSOR 3`<br>`INSENSOR=0(3)` | #Assign the SENSOR input to Input # 3 |
| `>SAVEPRM`<br>`(EEPROM has been written 2 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>`AS-One (ASX66)`<br>`Integrated Motor`<br>`Software Version: *.**`<br>`Copyright 2004`<br>`ORIENTAL MOTOR CO., LTD.`<br>`-------------------------------------------` | #Establish the saved parameter value |
| `>INSENSOR`<br>`INSENSOR=3(3)`<br>`>` | #Confirm new value |

## INSG : System Input Signal Status I/O

**Execution Mode** Immediate and Sequence

**Syntax** INSG

**Range** 0 to 4095 (integer values)

**Access** READ

**Description** The INSG command displays the current status of all the system input signals, as one integer number. The system input signals contribute to the value of INSG as follows:

| Bit Location | Signal | Contribution to INSG if active |
|---|---|---|
| Bit 0 | START | 1 |
| Bit 1 | ABORT | 2 |
| Bit 2 | PSTOP | 4 |
| Bit 3 | MSTOP | 8 |
| Bit 4 | LSP | 16 |
| Bit 5 | LSN | 32 |
| Bit 6 | HOME | 64 |
| Bit 7 | SENSOR | 128 |
| Bit 8 | PAUSE | 256 |
| Bit 9 | PAUSECL | 512 |
| Bit 10 | CROFF | 1024 |
| Bit 11 | ALMCLR | 2048 |

INSG is the sum of the contribution of all active signals:
If INSG=2, the ABORT signal is active, and all other signals are inactive.
If INSG=192, the HOME (64) and SENSOR (128) signals are active (64+128=192), and all other signals are inactive.

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

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

**Example**

| Command | Description |
|---|---|
| `>INSG` | #Query the current Input Signal Value |
| `INSG=1024` | #Device response: the CROFF signal is active |

## INx : Individual General Input Status I/O

**Execution Mode** Immediate and Sequence

**Syntax** INx

**Range** x = 0 to 6
0: Not Active
1: Active

**Initial Value** 0

**Access** READ

**Description** INx returns the state of General Purpose Input "x".
The active level of each General Purpose Input is determined by INxLV.
If the input has been assigned to a system input signal, then it is no longer "General Purpose". INx for these inputs will always return 0 (Not Active). Use the INSG command to check the status of the system input signals.

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

**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 |
| > | |

## INxLV : INx Input Level — I/O

**Execution Mode** Immediate (MODE = 0 Only)

**Syntax** INxLV n

**Range**
x = 0 to 6
n = 0: Normally Open
1: Normally Closed

**Initial Value** n = 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 default (initial) value.

**Access** READ and WRITE

**Description** INxLV establishes the active level of General Purpose Input x.
If input x has been assigned to a system input signal, then INxLV has no affect: the active level assigned to the signal is used.

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

**Example**

| Command | Description |
|---|---|
| >IN3LV<br>IN3LV=0(0) | #Query the active level for General Purpose Input 3 |
| >IN3LV=1<br>IN3LV=0(1) | #Set INPUT #3 to the Normally Closed logic level |
| >SAVEPRM<br>(EEPROM has been written 2 times)<br>Enter Y to proceed, other key to cancel. Y<br>Saving Parameters........OK. | #Save the parameter assignments |
| >RESET<br>Resetting system. | #Establish the saved parameter values |

```
------------------------------------------
    AS-One (ASX66)
      Integrated Motor
    Software Version: *.**
       Copyright 2004
   ORIENTAL MOTOR CO., LTD.
------------------------------------------
```

| Command | Description |
|---|---|
| >IN3LV<br>IN3LV=1(1)<br>> | #Confirm the active level for General Purpose Input 3 |

## IO : Input/Output Status I/O

**Execution Mode** Immediate

**Syntax** IO

**Description** IO displays the current status of General Purpose Inputs and Outputs and system input signals and system output signals.
Values are reported as 0:Inactive or 1:Active.
For inputs and outputs that have been assigned to a system input or output signal, the signal state is shown.
When MODE=0, a START input can start a sequence, determined by the binary value of IN. This value is shown in the I/O response under (SEQ#), and is the number of the sequence that would start if a START signal became active in this I/O state.
In the example below, Input 1 and Output 4 remain General Purpose: all other I/O have been assigned to system signals. PSTOP (Panic Stop) is asserted, and ALARM output is set. General Purpose Input #1 is active, so IN=1, and Sequence 1 would start if the alarm condition were cleared and START became active.

**See Also** ABORTLV, ALARMLV, CROFFLV, ENDLV, HOMELV, HOMEPLV, INCROFF, INHOME, INITIO, INLSN, INLSP, INMSTOP, INPAUSE, INPAUSECL, INPSTOP, INSENSOR, IN, INxLV, MOVELV, MSTOPLV, OTLV, OUTALARM, OUTEND, OUTHOMEP, OUTMBC, OUTMOVE, OUTPSTS, OUTRUN, OUTTEMP, PAUSECLLV, PAUSELV, PSTOPLV, PSTSLV, RUNLV, SENSORLV, STARTLV, TEMPLV

**Example**

| Command | Description |
|---|---|
| >IO | #Display the IO status |
| Inputs (1-6) = IN1 SENSOR HOME PSTOP -LS +LS | #Device response |
| Outputs (1-4) = MOVE RUN ALARM OUT4 | |

```
 --Inputs---                     Outputs
 1 2 3 4 5 6 -(SEQ#)- START ABORT - 1 2 3 4
 1 0 0 1 0 0 -(  1 )-   0      0    - 0 0 1 0
>
```

## KB : Keyboard Input — Sequence Commands

**Execution Mode** Sequence

**Syntax** *variable* = KB

**Range** *variable* refers to any numeric variable which sequences can write to.
Actual permitted range depends on *variable*

**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*.
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).
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.
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.
For similar operation without prompting, see KBQ (Keyboard Input Quiet).
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.

**See Also** KBQ, SAS, SACS, VIEW

**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 |

**Note** 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.

## KBQ : Keyboard Input (Quiet) — Sequence Commands

**Execution Mode** Sequence

**Syntax** *variable* = KBQ

**Range** *variable* refers to any numeric variable which sequences can write to.
Actual permitted range depends on *variable*

**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.

**See Also** KB, SAS, SACS, VIEW

**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 20 User Units |

**Note** 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.

## LDCHK : Estimate Load Parameters — Monitor Commands

**Execution Mode** Immediate (MODE = 0 Only)

**Syntax** LDCHK

**Description** LDCHK (Load Check) starts a load estimation process.
Several system features require reasonable estimates of load conditions for proper operation:
• Torque feedforward (TQFF=1)
• Automatic profiling (RMODE=1)
• Automatic current level setting (CMODE=2)

These features depend on reasonable estimates of load inertia (LI), load friction (LF), load static friction (LSF) and gravity (or other constant) torque (LG). See Section 4.5, "Enhanced Features", for more information on these features.
These parameters can be entered directly, if known. LDCHK is provided for those applications where measuring or calculating reasonable estimates is difficult.
LDCHK attempts to estimate these load parameters by carefully monitoring actual motion, while directing the motor in a "windshield wiper" pattern. The position error during this motion pattern is carefully monitored, and used to deduce estimates of the required values. If position error is insignificant, velocity may be increased, or current decreased, in an attempt to produce enough position error for reasonable estimation.
The maximum speed attempted by LDCHK is 5 revolution/second, at the motor shaft.
The motion pattern terminates if LDCHK finds reasonable values, or if an ESCAPE character is detected (in which case LDCHK aborts). Motion may also terminate if an alarm condition occurs, or if it unable to develop significant position error.
If LDCHK finds estimates and terminates motion normally, a carriage return, linefeed, or keyboard "Enter" will store the estimates into their respective parameters LI, LF, LSF, LG. ESCAPE will abort the process, leaving the parameters unchanged.
The new values of the load estimates become effective immediately, but SAVEPRM is required to permanently change the parameter values.
The LDCHK function is not available while a sequence is executing or motion is occurring. While this feature is in use, sequences are not available to start, and only PSTOP, LSN, LSP, CROFF inputs (if configured) are active.
LDCHK cannot be executed while motion is in progress or sequences are executing.
See "Load Estimation" in Section 4.7 for more information about load estimation

**Caution** **LDCHK causes the motor to move.**

**See Also** LF, LG, LI, LSF

**Note** LDCHK estimates the physical load parameters effective at the rotor shaft. If a gearhead is mounted to the motor, these are estimates of the torques and inertia present at the *input* to the gearhead, and will be reduced from the physical torques and inertia effective at the *output* of the gearhead. LF, LSF, and LG will be reduced from the gearhead output torques by the gear ratio (R), and LI will be reduced from the inertia effective at the output of the gearhead by a factor of $R^2$.
If electronic gearing is used (GA not equal to GB), it has no effect on the estimation process. The motion pattern is defined in terms of the motor shaft.

**Example**

| Command | Description |
|---|---|
| >LDCHK | #Execute the load check function |

```
>LDCHK
 Start Load Parameter Estimation

  <ESC> : Abort, exit.

    LI  :  LF  :  LG  :  LSF : Status
   937     8     0      0  Complete

 <Enter> to proceed, <ESC> to cancel.

 Load parameters are set.

 LI  =   937 [gcm^2]
 LF  =     8 [Ncm]
 LG  =     0 [Ncm]
 LSF =     0 [Ncm]

 Exit load estimation mode.
>LI
 LI=937
>
```

Description:

- After "937 8 0 0 Complete": #Estimates updated here
- After "<Enter> to proceed, <ESC> to cancel.": #Motion Stops. Accept with <CR>
- After "Load parameters are set.": #System reports final values.
- After ">LI": #Back to prompt. Query load inertia
- After "LI=937": #LI has been updated by LDCHK.

## LF : Load Function — System Control

**Execution Mode** Immediate (MODE = 0 Only) and Sequence

**Syntax** LF n

**Range** n = 0 to 200 (Integer values only) (N·cm)

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**Description** LF is the estimated load friction torque, in Newton-Centimeters (N·cm). LF represents the frictional torque expected while the motor is moving, opposing the direction of motion.
LF is used to calculate feedforward torque (when TQFF=1), to calculate motion profiles (when RMODE=1), and to calculate current requirements (when CMODE=2). When these features are used, LF effects the performance of the system, and a reasonable estimate is recommended.
LF can be programmed directly, or the system can attempt to find a reasonable estimate automatically, using the Load Estimation procedure (LDCHK).

**See Also** LDCHK, LG, LI, LSF

**Note** LF represents the friction torque effective at the rotor shaft. If a gearhead is mounted to the motor, LF represents the friction torque present at the *input* to the gearhead, and will be reduced from the friction torque effective at the *output* of the gearhead by the gear ratio.
Because load estimates are effective at the rotor shaft, electronic gearing does not effect the estimates.

**Example**

| Command | Description |
|---|---|
| `>LF 5`<br>` LF=5` | #Set the Load Friction Torque value to 5 N·cm |
| `>LG 20`<br>` LG=20` | #Set the Load Gravity Torque value to 20 N·cm |
| `>LI 500`<br>` LI=500` | #Set the Load Inertia value to 500 g·cm$^2$ |
| `>LSF 3`<br>` LSF=3` | #Set the Load Static Friction to 3 N·cm |
| `>SAVEPRM`<br>` (EEPROM has been written 62 times)`<br>` Enter Y to proceed, other key to cancel. y`<br>` Saving Parameters........OK.` | #Save all parameter assignments |
| `>RESET`<br>` Resetting system.`<br>`------------------------------------------`<br>`AS-One (ASX66)`<br>`Integrated Motor`<br>`Software Version: *.**`<br>`Copyright 2004`<br>`ORIENTAL MOTOR CO., LTD.`<br>`------------------------------------------` | #Establish the saved parameter values |
| `>LF`<br>` LF=5`<br>`>` | #Confirm new value of load friction |

## LG : Load Gravity — System Control

**Execution Mode** Immediate (MODE = 0 Only) and Sequence

**Syntax** LG n

**Range** n = −200 to 200 (Integer values only) (N·cm)

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**Description** LG is the estimated load gravity torque (or other constant torque), in Newton-Centimeters (N·cm). LG represents the load torque expected at all times (motor moving or motor stopped). Positive values of LG represent constant load torques "pushing down", or tending to move the motor in the negative direction. Negative values of LG represent constant load torques "pushing up", or tending to move the motor in the positive direction.
LG is used to calculate feedforward torque (when TQFF=1), to calculate motion profiles (when RMODE=1), and to calculate current requirements (when CMODE=2). When these features are used, LG effects the performance of the system, and a reasonable estimate is recommended.
LG can be programmed directly, or the system can attempt to find a reasonable estimate automatically, using the Load Estimation procedure (LDCHK).

**See Also** LDCHK, LF, LI, LSF, DIRINV

**Interactions** Modifies: Motion Profile
Modified by: LDCHK, Verbose

**Note** LG represents the constant torque effective at the rotor shaft. If a gearhead is mounted to the motor, LG represents the constant torque present at the *input* to the gearhead, and will be reduced from the constant torque effective at the *output* of the gearhead by the gear ratio.
Because load estimates are effective at the rotor shaft, electronic gearing does not effect the estimates.

**Example**

| Command | Description |
|---|---|
| `>LF 5`<br>` LF=5` | #Set the Load Friction Torque value to 5 N·cm |
| `>LG 20`<br>` LG=20` | #Set the Load Gravity Torque value to 20 N·cm |
| `>LI 500`<br>` LI=500` | #Set the Load Inertia value to 500 g·cm$^2$ |
| `>LSF 3`<br>` LSF=3` | #Set the Load Static Friction to 3 N·cm |
| `>SAVEPRM`<br>` (EEPROM has been written 62 times)`<br>` Enter Y to proceed, other key to cancel. y`<br>` Saving Parameters........OK.` | #Save all parameter assignments |
| `>RESET`<br>` Resetting system.`<br>`-----------------------------------------`<br>`    AS-One (ASX66)`<br>`      Integrated Motor`<br>`    Software Version: *.**`<br>`      Copyright 2004`<br>`   ORIENTAL MOTOR CO., LTD.`<br>`-----------------------------------------` | #Establish the saved parameter values |
| `>LG`<br>` LG=20`<br>`>` | #Confirm new value of load gravity |

## LI : Load Inertia — System Control

**Execution Mode** Immediate (MODE = 0 Only) and Sequence

**Syntax** LI n

**Range** n = 0 to 12000 (Integer values only) ($g \cdot cm^2$)

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**Description** LI is the estimated load inertia, represented at the motor shaft, in gram-centimeters$^2$. LI effects the torque required for acceleration and deceleration.
LI is used to calculate feedforward torque (when TQFF=1), to calculate motion profiles (when RMODE=1), and to calculate current requirements (when CMODE=2). When these features are used, LI effects the performance of the system, and a reasonable estimate is recommended.
LI can be programmed directly, or the system can attempt to find a reasonable estimate automatically, using the Load Estimation procedure (LDCHK).
The inertia of the motor rotor itself should not be included in LI: the system accounts for rotor inertia automatically.

**See Also** LDCHK, LF, LG, LSF

**Note** LI represents the load inertia effective at the rotor shaft. If a gearhead is mounted to the motor, LI represents the load inertia present at the input to the gearhead, and will be reduced from the inertia effective at the output of the gearhead by the gear ratio squared.
Because load estimates are effective at the rotor shaft, electronic gearing does not effect the estimates.

**Example**

| Command | Description |
|---|---|
| `>LF 5`<br>`LF=5` | #Set the Load Friction Torque value to 5 N·cm |
| `>LG 20`<br>`LG=20` | #Set the Load Gravity Torque value to 20 N·cm |
| `>LI 500`<br>`LI=500` | #Set the Load Inertia value to 500 $g \cdot cm^2$ |
| `>LSF 3`<br>`LSF=3` | #Set the Load Static Friction to 3 N·cm |
| `>SAVEPRM`<br>`(EEPROM has been written 62 times)`<br>`Enter Y to proceed, other key to cancel. y`<br>`Saving Parameters........OK.` | #Save all parameter assignments |
| `>RESET`<br>`Resetting system.`<br>`----------------------------------------------------`<br>`AS-One (ASX66)`<br>`Integrated Motor`<br>`Software Version: *.**`<br>`Copyright 2004`<br>`ORIENTAL MOTOR CO., LTD.`<br>`----------------------------------------------------` | #Establish the saved parameter values |
| `>LI`<br>`LI=500`<br>`>` | #Confirm new value of load inertia |

## LIMN, LIMP : Software Position Limits — System Control

| | |
|---|---|
| **Execution Mode** | Immediate (MODE = 0 Only) and Sequence |
| **Syntax** | LIMN n: Minimum permitted position<br>LIMP n: Maximum permitted position |
| **Range** | n = −MAXPOS to +MAXPOS (User Units) |
| **Initial Value** | 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 default (initial) value. |
| **Access** | READ and WRITE<br>READ only in sequences |
| **Description** | When SLACT=1, software position limits LIMN and LIMP are enforced, provided the system has completed a homing action (EHOME, MGHP, MGHN).<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>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 LIMN and LIMP.)<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 the action defined by ALMACT will trigger.<br>For continuous motions (MCN, MCP), 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. MCN or MCP can be executed, if the motor would move in the direction of the operational range. |
| **See Also** | SLACT, PC, MGHP, MGHN, EHOME, ALM, ALMACT, OTACT |
| **Note** | If LIMN=LIMP=0, software position limit checking is disabled, even if SLACT=1. LIMN and LIMP 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 instead of LSN, LSP. |
| HOMETYP=8 | |
| >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. | |
| ------------------------------------------ | |
| AS-One (ASX66) | |
| Integrated Motor | |
| Software Version: *.** | |
| Copyright 2004 | |
| 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 detected. | #Detected limit |
| >PC | |
| PC=10.001 Rev | #Checked PC |
| > | #Just over LIMP |

## LINKx : Link Control — Motion Variables

**Execution Mode** Immediate (MODE=0 Only) and Sequence

**Syntax** LINKx n

**Range**
x = 0 to 2 (Linked Motion Profiles defined by DISx, INCABSx, VRx)
n = 0: Segment (x) terminates motion
1: Link segment (x) to segment (x+1)

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**Description** LINKx control whether linked motion segment x is linked to the next segment, or not.
If LINKx=0, the motion segment defined by DISx and VRx will terminate.
If LINKx=1, the motion segment defined by DISx and VRx will not terminate: motion will proceed to motion segment (x+1).

**See Also** DISx, INCABSx, MIx, TA, TD, VRx, VS

**Example**

| Command | Description |
|---|---|
| `>VR0 5` | #Set the velocity for link segment #0 to 5 user units/s |
| `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　　Sequence Management

**Execution Mode**　Immediate (MODE = 0 Only)

**Syntax**　LIST target [start line] [end line]

**Range**　target can be the name or number of any existing sequence
[start line] is an optional line number.
[end line] is an optional line number, if [startline] is specified.　If given, it must not be less than [start line].

**Description**　LIST lists the contents of a stored sequence.
If [start line] and [end line] are not specified, the entire sequence is listed.
If [start line] is specified, output starts with line [start line].
If [end line] is specified, output ends after line [end line].

**See Also**　DIR, EDIT

**Example**

| Command | Description |
|---|---|
| `>LIST TEMPCHECK 6 11` | #List sequence TEMPCHECK, from line 6 through 11 |

```
(  6)   IF (SIGTEMP=1)
(  7)      SSTOP
(  8)      MEND
(  9)      CURRENT 0
( 10)      SAS Cooling
( 11)   ENDIF
>
```

#Partial contents of sequence TEMPCHECK

## LISTVAR : Lists All User-Defined Variables — User Variables

**Execution Mode** Immediate (MODE=0 Only)

**Syntax** LISTVAR

**Description** LISTVAR lists the names and values of all user-defined variables (String – S_xxx and Numeric – N_xxx)

**See Also** CLEARVAR, CREATEVAR, N_xxx, S_xxx

**Example**

| Command | Description |
|---|---|
| >LISTVAR | #List all user-defined variables |

```
   ##  N_name      Numeric Data
   ==  ==========  ============
    1  PRICE        0
    2  QUANTITY     0
    3  LOT          32
    4  SERIAL      4583274
    5  SIZE         0
    6  LENGTH      106
    7  WIDTH       60
    8  WEIGHT      0.95
    9             0
    10            0
   ##  S_name      String Data
   ==  ==========  ====================
    1  NAME         ASOne66
    2  SERIES      VEXTA STEP
    3  STATUS      Same day shipping OK
    4  MESSAGE
    5  UNIT        Kilogram
    6  COUNTRY     USA
    7
    8
    9
   10
>
```

#List for numeric user-defined variables

#Variables created but not assigned a value show 0

#Empty (available) slots have no name.
#List for string user-defined variables

## LOCK : Lock Sequence — Sequence Management

**Execution Mode** Immediate (MODE = 0 Only)

**Syntax** LOCK target

**Range** target can be the name or number of any existing sequence

**Description** LOCK prevents changes to a sequence.
A locked sequence cannot be deleted, renamed, or overwritten (by COPY or EDIT).
A locked sequence can still be loaded into the editor (with the EDIT command), but any changes must be saved to a new location.
A locked sequence can be unlocked with the UNLOCK command.
The sequence directory listing (DIR command) shows the lock status for all sequences.

**See Also** DEL, DIR, EDIT, UNLOCK

**Note** 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  2048 bytes total,    2 percent.
  Storage memory:      77 bytes used of 21775 bytes total,    0 percent.
>
```

## LOOP : Begin Counted LOOP Block — Sequence Commands

**Execution Mode** Sequence

**Syntax** LOOP [n]

**Range** n = 1 to 500,000,000 (integer values), loop count

**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 (LOOP−ENDL, IF−ENDIF, WHILE−WEND) can be nested up to 8 levels deep.

**See Also** BREAKL, ENDL, WHILE, WEND

**Example**

| Command | Description |
|---|---|
| `>LIST 27` | #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 |
| `>` | |

## LSF : Load Static Friction — System Control

**Execution Mode** Immediate (MODE = 0 Only) and Sequence

**Syntax** LSF n

**Range** n = 0 to 200 (Integer values only) (N·cm)

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**Description** LSF is the estimated load static friction torque, in Newton-Centimeters (N·cm). LSF represents the frictional torque expected while the motor is stopped.
LSF is used to calculate feedforward torque (when TQFF=1) and to calculate current requirements (when CMODE=2). When these features are used, LSF effects the performance of the system, and a reasonable estimate is recommended.
LSF can be programmed directly, or the system can attempt to find a reasonable estimate automatically, using the Load Estimation procedure (LDCHK).

**See Also** LDCHK, LF, LG , LI

**Note** LSF represents the static friction torque effective at the rotor shaft. If a gearhead is mounted to the motor, LSF represents the friction torque present at the *input* to the gearhead, and will be reduced from the static friction torque effective at the *output* of the gearhead by the gear ratio.
Because load estimates are effective at the rotor shaft, electronic gearing does not effect the estimates.

**Example**

| Command | Description |
|---|---|
| `>LF 5`<br>` LF=5` | #Set the Load Friction Torque value to 5 N·cm |
| `>LG 20`<br>` LG=20` | #Set the Load Gravity Torque value to 20 N·cm |
| `>LI 500`<br>` LI=500` | #Set the Load Inertia value to 500 g·cm$^2$ |
| `>LSF 3`<br>` LSF=3` | #Set the Load Static Friction to 3 N·cm |
| `>SAVEPRM`<br>` (EEPROM has been written 62 times)`<br>` Enter Y to proceed, other key to cancel. y`<br>` Saving Parameters........OK.` | #Save all parameter assignments |
| `>RESET`<br>` Resetting system.` | #Establish the saved parameter values |

```
-----------------------------------------
    AS-One (ASX66)
      Integrated Motor
    Software Version: *.**
       Copyright 2004
   ORIENTAL MOTOR CO., LTD.
-----------------------------------------
```

| Command | Description |
|---|---|
| `>LSF`<br>` LSF=3`<br>`>` | #Confirm new value of load static friction |

## MA : Move to Absolute Position — Motion Commands

**Execution Mode** Immediate (MODE = 0 Only) and Sequence

**Syntax** MA n

**Range** n = −MAXPOS to +MAXPOS (User Units)
In immediate mode, 'n' can be a constant or any POS [x] position array variable.
In a sequence, 'n' can be a constant or any variable which can be read within a sequence.

**Description** MA starts a point-to-point motion to position “n”.
Motion velocity is determined by running velocity (VR). Start velocity (VS), acceleration time (TA), and deceleration time (TD) are effective when linear ramps are used (RMODE=0). When automatic ramping is used (RMODE=1), the system automatically determines profile shape: VS, TA and TD are ignored.
If RMODE=0, speed may be changed while the motion is in progress, using the Change Velocity command (CV). The CV command is not permitted while RMODE=1.
If the motion finishes successfully, the position setpoint (PC) should equal 'n'.
Some combinations of effective distance, speeds and acceleration and deceleration times are not feasible. For instance: if VR is very high, and TA and TD are very long, but the effective distance is very short, the system could cover too much distance accelerating to velocity VR over time TA. The system monitors for these conditions, and starts decelerating early if necessary. (Under these conditions, peak speed will be less than VR, and acceleration and deceleration times will be less than TA and TD.) The system is careful to preserve the actual motion distance, and the effective acceleration and deceleration *rates*.
MA is not accepted while the motor is moving, when current is off, or when the system has an active alarm condition. An attempt to execute MA while the motor is moving causes an error message in immediate mode, and causes an alarm and sequence termination (alarm code: A0h) if executed from a sequence.

**See Also** DPR, MCN, MCP, MI, PC, RMODE, TEACH, UU, MEND, CV

**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 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 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. | |
| > | |

## MAXPOS : Maximum Position Value — System Status

**Execution Mode** Immediate

**Syntax** MAXPOS

**Range** n/a (User Units)

**Initial Value** 41943.000 (User Units)

**Access** READ

**Description** MAXPOS (Maximum Position) is the largest permitted value for position-related parameter entry. Position related parameters (DIS, PC, OFFSET, etc.) 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), and is automatically updated when DPR is changed. The new value becomes effective after SAVEPRM and RESET; both active and future values of MAXPOS are shown when MAXPOS is queried.

**See Also** DPR, GA, GB, MAXVEL, MAXOVERFLOW

**Example**

| Command | Description |
|---|---|
| >UU in.<br> UU=in. | #Set the User Units to in. (inches) |
| >DPR 10<br> DPR=1(10) in.<br> OVERFLOW, OVERVEL is re-scaled to default equivalent.<br> OVERFLOW=3(30) in.<br> OVERVEL=100(1000) in./sec<br> Position range = +/- 41943(419430)<br> Velocity range = 0.001 - 83.333(833.333) | #Set the Distance Per Revolution to 10 User Units: device responds with ranges, active and (future). |
| >SAVEPRM<br> (EEPROM has been written 68 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>AS-One (ASX66)<br>Integrated Motor<br>Software Version: *.**<br>Copyright 2004<br>ORIENTAL MOTOR CO., LTD.<br>------------------------------------------ | #Establish the saved parameter values |
| >DPR<br> DPR=10(10) in.<br> Position range = +/- 419430(419430)<br> Velocity range = 0.001 - 833.333(833.333) | #Confirm the new DPR setting |
| >MAXPOS<br> MAXPOS=419430(419430) in. | #Query the Maximum Position Value |
| >MAXVEL<br> MAXVEL=833.333(833.333) in./sec<br>> | #Query the Maximum Velocity Value |

## MAXVEL : Maximum Velocity Value — System Status

**Execution Mode** Immediate

**Syntax** MAXVEL

**Range** n/a (User Units/Second)

**Initial Value** 83.333 (User Units/Second)

**Access** READ

**Description** MAXVEL (Maximum Velocity) is the largest permitted value for velocity-related parameter entry. Velocity related parameters (VS, VR, OVERVEL, etc.) must be less than or equal to MAXVEL.
MAXVEL is determined by DPR (Distance per Revolution), and electronic gearing parameters GA and GB. 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 when MAXVEL is queried.

**See Also** DPR, GA, GB, MAXPOS, MAXOVERFLOW

**Example**

| Command | Description |
|---|---|
| `>UU in.`<br>` UU=in.` | #Set the User Units to in. (inches) |
| `>DPR 10`<br>` DPR=1(10) in.`<br>` OVERFLOW, OVERVEL is re-scaled to default equivalent.`<br>` OVERFLOW=3(30) in.`<br>` OVERVEL=100(1000) in./sec`<br>` Position range = +/- 41943(419430)`<br>` Velocity range = 0.001 - 83.333(833.333)` | #Set the Distance Per Revolution to 10 User Units: device responds with ranges, active and (future). |
| `>SAVEPRM`<br>` (EEPROM has been written 68 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>`AS-One (ASX66)`<br>`Integrated Motor`<br>`Software Version: *.**`<br>`Copyright 2004`<br>`ORIENTAL MOTOR CO., LTD.`<br>`-------------------------------------------` | #Establish the saved parameter values |
| `>DPR`<br>` DPR=10(10) in.`<br>` Position range = +/- 419430(419430)`<br>` Velocity range = 0.001 - 833.333(833.333)` | #Confirm the new DPR setting |
| `>MAXPOS`<br>` MAXPOS=419430(419430) in.` | #Query the Maximum Position Value |
| `>MAXVEL`<br>` MAXVEL=833.333(833.333) in./sec`<br>`>` | #Query the Maximum Velocity Value |

## MAXOVERFLOW : Maximum Overflow — System Status

**Execution Mode** Immediate

**Syntax** MAXOVERFLOW

**Range** n/a (User Units)

**Initial Value** 600.000 (User Units)

**Access** READ

**Description** MAXOVERFLOW (Maximum value of position error limit OVERFLOW) indicates the permitted value for OVERFLOW.
MAXOVERFLOW is determined by DPR (Distance per Revolution), and is automatically updated when DPR is changed. The new value becomes effective after SAVEPRM and RESET; both active and future values of MAXOVERFLOW are shown when MAXOVERFLOW is queried.

**See Also** DPR, GA, GB, MAXPOS, MAXVEL, OVERFLOW

**Example**

| Command | Description |
|---|---|
| `>UU deg`<br>` UU=deg` | #Set the User Units to deg (degrees) |
| `>DPR 360`<br>` DPR=1(360) deg`<br>` OVERFLOW, OVERVEL is re-scaled to default equivalent.`<br>` OVERFLOW=3(1080) deg`<br>` OVERVEL=100(36000) deg/sec`<br>` Position range = +/- 41943(499680)`<br>` Velocity range = 0.001 - 83.333(30000)` | #Set the Distance Per Revolution to 360 degrees. OVERFLOW and OVERVEL are automatically rescaled, new ranges are calculated. |
| `>MAXOVERFLOW`<br>` MAXOVERFLOW=600(216000) deg` | #Query the Maximum Overflow Value |
| `>OVERFLOW 45`<br>` OVERFLOW=3(45) deg`<br>`>` | #Set OVERFLOW to 45 degrees. |

## MCN, MCP : Move Continuously — Motion Commands

**Execution Mode** Immediate (MODE = 0 Only) and Sequence

**Syntax**
MCN
MCP

**Description**
MCN and MCP start continuous motions, with no defined final position. MCN starts moving in the negative direction, and MCP starts moving in the positive direction.
Motion velocity is determined by running velocity (VR). Start velocity (VS), acceleration time (TA) and deceleration time (TD) are effective when linear ramps are used (RMODE=0). When automatic ramping is used (RMODE=1), the system automatically determines profile shape: VS and TA are ignored.
Motion continues until the system is commanded to stop or an alarm condition occurs.
Velocity can be changed while a continuous motion is in progress, by changing the value of VR and re-issuing the MCN or MCP command. If linear ramps are used (RMODE=0), ramp time will be TA if speed is increasing (away from zero) and TD if speed is decreasing (toward zero). The direction cannot be changed: MCN cannot be issued while an MCP 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.
MCN and MCP 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.

**See Also**
<ESC>, ABORT, DIRINV, DPR, PSTOP, INLSN, INLSP, INMSTOP, INPAUSE, INxLV, LIMN, LIMP, MSTOPACT, MSTOPLV, PAUSE, TA, TD, UU, VR, VS

**Note**
MCN and MCP start continuous motions, but do not wait for motion to end. Other commands can be issued in immediate mode or executed by a sequence while the motion is running, although most motion commands cannot be executed until the motion is complete.
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 — Sequence Commands

**Execution Mode** Sequence

**Syntax** MEND

**Description** MEND suspends sequence processing until motion is complete.
Most motion commands *start* motions, but do not wait for motion to complete. Other operations can be performed while the motor is moving. MEND provides a simple way of synchronizing sequence execution with the end of a motion. When the motion completes (or if no motion is in progress), sequence execution proceeds to the statement following MEND.
MEND is equivalent to WHILE (SIGMOVE=1); WEND
Most motion commands cannot be executed while another motion is in progress. To avoid errors, sequences should be designed to assure that each motion is complete before proceeding to another motion.

**See Also** SIGMOVE, SIGEND, WHILE, WEND, IF, ENDIF

**Example**

| Command | Description |
|---|---|
| >LIST MOVETIME | |
| ( 1) VS .1; VR 8; TA .05; TD .05 | #Set motion parameters |
| ( 2) DIS 4 | #Set distance DIS is 5. |
| ( 3) T=0; Z=TIMER; MI | #T is zero, Z= start time. Move Incremental |
| ( 4) WHILE (VC<VR); WEND; T=TIMER-Z | #Wait for velocity to reach peak. Calc time. |
| ( 5) SACS ACTUAL ACCELERATION TIME: | #Message (last two chars are ^ and space) |
| ( 6) T | #Transmit acceleration time |
| ( 7) MEND; T=TIMER-Z | #Wait for motion end, capture elapsed time |
| ( 8) SACS TOTAL MOVE TIME:^ | #Message (last two chars are ^ and space) |
| ( 9) T | #Transmit motion time. |
| >FILT 0.025; RUN MOVETIME | #Set filter time lag to 25 msec, run |
| FILT=0.025 | |
| >ACTUAL ACCELERATION TIME: 0.127 | #First message |
| >TOTAL MOVE TIME: 0.644 | #Second message |
| >FILT 0.002; RUN MOVETIME | #Reduce filter lag time, run again |
| FILT=0.002 | |
| >ACTUAL ACCELERATION TIME: 0.055 | #Acceleration time much shorter |
| >TOTAL MOVE TIME: 0.555 | #Total motion time much shorter |
| > | |

## MGHN, MGHP : Seek Mechanical Home Position — Motion Commands

**Execution Mode** Immediate (MODE = 0 Only) and Sequence

**Syntax**
MGHN
MGHP

**Description**
MGHN and MGHP start motion patterns, attempting to find a mechanical home position which links position zero (PC=0) to an application reference signal. MGHN starts moving in the negative direction, and MGHP starts moving in the positive direction.
The process may involve moving in both directions before concluding. MGHN and MGHP differ in starting direction, and in direction upon final approach to the designated home signal (final approach is in the same direction as starting direction).
The actual motion pattern and signal requirements are determined by HOMETYP. Depending on HOMETYP, one or more of system input signals LSN, LSP, and HOME must be assigned to an input, before executing MGHN or MGHP. 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 "Mechanical Home Seeking" in Section 4.4 for more information.
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.
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.
Software position limits LIMN and LIMP 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 (LIMN and LIMP not both 0), the limits are enabled after successful completion of a homing process.
Auto ramping (RMODE=1) is not available during home seeking. The system uses linear ramping, even if RMODE=1.
MGHN and MGHP 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.

**See Also** DIRINV, INHOME, INLSN, INLSP, HOMETYP, HOMELV, PC, OFFSET, OUTHOMEP, OUTSG, SIGHOMEP

**Note**
MGHN and MGHP start the home seeking process, but do not wait for the process to end. Other commands can be issued in immediate mode or executed by a sequence while the process is running, although motion commands cannot be executed until the process is complete.
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, LSN, LSP |
| `>MGHP` | #Start seeking home, positive direction |
| `>SIGMOVE` | #Check MOVE signal (after motion) |
| `SIGMOVE=0` | #MOVE is OFF |
| `>SIGHOMEP` | #Check HOMEP signal |
| `SIGHOMEP=1` | #HOMEP is ON (Home is found, success) |
| `>PC` | #Check position command PC |
| `>PC=0 mm` | #Automatically zeroed when homing succeeded. |
| `>` | |

## MI : Move Incremental Distance — Motion Commands

**Execution Mode** Immediate (MODE = 0 Only) and Sequence

**Syntax** MI

**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 when linear ramps are used (RMODE=0). When automatic ramping is used (RMODE=1), the system automatically determines profile shape: VS, TA and TD are ignored.
If RMODE=0, speed may be changed while the motion is in progress, using the Change Velocity command (CV). The CV command is not permitted while RMODE=1.
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.

**See Also** DPR, DIS, MA, RMODE, TA, TD, UU, VR, VS, CV

**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 QCOUNTS` | #List sequence QCOUNTS |
| `(  1) RMODE=0` | #Set for linear ramps |
| `(  2) TA=0.1; TD=0.1` | #Acceleration and deceleration times to 0.1 |
| `(  3) VS=0; VR=10;` | #Program start and running speeds |
| `(  4) DIS=1.0` | #Set distance to 1 user unit. |
| `(  5) OUT3=0; OUT4=0` | #Set general purpose outputs 3 and 4 inactive |
| `(  6) LOOP Q` | #Loop, loop count given by variable Q |
| `(  7)   MI` | #Start moving incrementally, distance DIS |
| `(  8)   MEND; PC` | #Wait for motion to end, then send the value of PC |
| `(  9)   OUT4=1` | #Set Output 4 active |
| `( 10)   WAIT 1` | #Wait 1 second |
| `( 11)   OUT4=0` | #Set Output 4 inactive |
| `( 12) ENDL` | #Endo of loop |
| `( 13) MA 0` | #Start moving to absolute position 0 |
| `( 14) MEND; PC` | #Wait for motion to end, then send value of PC |
| `( 15) OUT3=1` | #Set Output 3 active |
| `>PC` | #Display position command before starting |
| `PC=0 mm` | |
| `>Q` | #Display value of variable Q |
| `Q=5` | |
| `>run QCOUNTS` | #Run Qcounts. Should do 5 incremental motions |
| `>1` | #Output of PC after each incremental motion completes |
| `>2` | |
| `>3` | |
| `>4` | |
| `>5` | |
| `>0` | #Output of PC after absolute move completes |
| `>` | |

## MIx　: Start Linked Index　　Motion Commands

**Execution Mode** Immediate (MODE = 0 Only) and Sequence

**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

**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 (INCABS=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.

**See Also** DISx, DPR, INCABSx, LINKx, MIx, TA, TD, UU, VRx, VS

**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 an MIx command is attempted which would result in both positive and negative motion, the MIx command is rejected. (An error message is generated in immediate mode. In a sequence, alarm 0x70h is set, and sequence processing terminates.)
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.

**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" link2 from link3 |
| `LINK2=0` | #Device response |
| `>MI1` | #Start the linked operation motion |
| `>` | |

## MODE　: Device Mode　　System Control

**Execution Mode**　Immediate

**Syntax**　MODE n

**Range**　n =　0: Internal Motion Profiler
1: Pulse Input, 1-pulse (Step and Direction Signals from an external pulse generator)
2: Pulse Input, 2-pulse (Positive and Negative pulses from an external pulse generator)
3: Pulse Input, quadrature (Quadrature pulses from e.g. an external encoder)

**Initial Value**　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 default (initial) value.

**Access**　READ and WRITE

**Description**　MODE determines the motion generation strategy for the device.
When MODE=0 (the default), motions are generated by the device's internal motion profile generator using motion parameters and commands.　Sequences are operational.　System input and output signal assignments may be customized for the application.
When MODE=1, 2, or 3, motions follow an external pulse train.　(MODEs 1, 2, and 3 differ only in how they interpret the pulse input signals.)　The system cannot generate its own motions: many motion-related commands are unavailable.　Sequences are unavailable.　System input and output signal assignments are fixed.
See Section 3.4 – □ in "Installation and Connection" for more information on device connections for Pulse Input Drive Mode.

**See Also**　DPP, DPR

**Example**

| Command | Description |
|---|---|
| >UU degrees<br> UU=degrees | #Set User Units to "degrees" |
| >DPR 360<br> DPR=1(360) degrees | #Set the device to 360 "degrees" per revolution |
| OVERFLOW, OVERVEL is re-scaled to default equivalent.<br> OVERFLOW=3(1080) degrees<br> OVERVEL=100(36000) degrees/sec<br> Position range = +/- 41943(499680)<br> Velocity range = 0.001 - 83.333(30000) | #System auto-rescales some limits, and shows new position and velocity ranges. |
| >DPP .05<br> DPP=0.001(0.05) degrees | #Set the distance per pulse to 0.05 "degrees" |
| >MODE 1<br> MODE=0(1) | #Select 1-Pulse Input Drive Mode |
| >SAVEPRM<br> (EEPROM has been written 73 times)<br> Enter Y to proceed, other key to cancel. Y<br> Saving Parameters........OK. | #Save parameters |
| >RESET<br> Resetting system.<br>-------------------------------------------<br>AS-One (ASX66)<br>Integrated Motor<br>Software Version: *.**<br>Copyright 2004<br>ORIENTAL MOTOR CO., LTD.<br>------------------------------------------- | #Reset the system, make new settings effective. |
| >DPR<br> DPR=360(360) degrees<br> Position range = +/- 499680(499680)<br> Velocity range = 0.001 - 30000(30000)<br>>DPP<br> DPP=0.05(0.05) degrees | #Confirm new settings |
| >MODE<br> MODE=1(1)<br>> | #New MODE=1 |

## MOVELV : Move Output Level — I/O

**Execution Mode** Immediate (MODE = 0 Only)

**Syntax** MOVELV n

**Range** n = 0: Normally Open
1: Normally Closed

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**Description** MOVELV sets the active level of the MOVE output, if used.

**See Also** INITIO, IO, OUT, OUTMOVE, OUTTEST, OUTSG, SIGMOVE

**Example**

| Command | Description |
|---|---|
| >MOVELV 1 | #Set the MOVE output to Normally Closed |
| MOVELV=0(1) | |
| >OUTMOVE 2 | #Assign the MOVE output to output #2 |
| OUTMOVE=0(2) | |
| >SAVEPRM | #Save the parameter assignments |
| (EEPROM has been written 29 times) | |
| Enter Y to proceed, other key to cancel. Y | |
| Saving Parameters........OK. | |
| >RESET | #Establish the saved parameter values |
| Resetting system. | |

```
------------------------------------------
    AS-One (ASX66)
      Integrated Motor
    Software Version: *.**
       Copyright 2004
   ORIENTAL MOTOR CO., LTD.
------------------------------------------
```

| Command | Description |
|---|---|
| >MOVELV | #Confirm the current MOVELV setting |
| MOVELV=1(1) | |
| > | |

## MSTOP : Motor Stop — Monitor Commands

**Execution Mode** Immediate (MODE = 0 Only) and Sequence

**Syntax** MSTOP

**Description** MSTOP causes the motor to stop. 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, if is assigned. See the INMSTOP command for more information.
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.**

**See Also** <ESC>, ABORT, HSTOP, INMSTOP, MSTOPACT, MSTOPLV, PSTOP, SSTOP

**Example**

| Command | Description |
|---|---|
| `>MSTOPACT` | #Check MSTOPACT |
| ` MSTOPACT=1(1)` | #MSTOPACT: Soft Stop |
| `>VR 10` | #Set the running velocity to 10 User Units/Rev |
| ` 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 — System Control

**Execution Mode** Immediate (MODE = 0 Only)

**Syntax** MSTOPACT n

**Range** n = 0: Hard Stop (stop as quickly as possible)
1: Soft Stop (controlled deceleration over time)

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**Description** MSTOPACT establishes the motor action upon activation of the MSTOP input and the MSTOP command.

If MSTOPACT=0, the MSTOP input and command stop the motor as quickly as possible (hard stop). MSTOP behaves exactly the same as HSTOP.

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.**

**See Also** INMSTOP, MSTOPLV, SIGMSTOP, MSTOP, HSTOP, SSTOP

**Example**

| Command | Description |
|---|---|
| >MSTOPACT | #Check the MSTOPACT setting |
| MSTOPACT=1(1) | #Set for soft stop action |
| >RMODE 0 | #RMODE=0: use linear ramps |
| RMODE=0 [Linear] | |
| >VS 0; VR 4.25 | #Set start velocity 0, run velocity 4.25 RPS |
| VS=0 Rev/sec | |
| VR=4.25 Rev/sec | #Acceleration time 0.05, Deceleration time 0.025 |
| >TA 0.05; TD 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 |
| > | |

## MSTOPLV : Motor Stop Input Level — I/O

**Execution Mode** Immediate (MODE = 0 Only)

**Syntax** MSTOPLV n

**Range** n = 0: Normally Open
1: Normally Closed

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**Description** MSTOPLV sets the active level of the MSTOP input, if used.

**See Also** INMSTOP, SIGMSTOP, MSTOP, MSTOPACT

**Example**

| Command | Description |
|---|---|
| `>MSTOPLV 1`<br>`MSTOPLV=0(1)` | #Set the MSTOP input to Normally Closed |
| `>OUTMOVE 4`<br>`OUTMOVE=0(4)` | #Assign the MSTOP input to output 4 |
| `>MSTOPACT 0`<br>`MSTOPACT=1(0)` | #Configure MSTOPACT so that MSTOP causes a hard stop |
| `>SAVEPRM`<br>`(EEPROM has been written 29 times)`<br>`Enter Y to proceed, other key to cancel. Y`<br>`Saving Parameters........OK.` | #Save the parameter assignments |
| `>RESET`<br>`Resetting system.` | #Establish the saved parameter values |

```
------------------------------------------
    AS-One (ASX66)
      Integrated Motor
    Software Version: *.**
       Copyright 2004
   ORIENTAL MOTOR CO., LTD.
------------------------------------------
```

| Command | Description |
|---|---|
| `>MSTOPLV`<br>`MSTOPLV=1(1)`<br>`>` | #Confirm the current MSTOPLV setting |

## MTMP : Motor Temperature System Status

**Execution Mode** Immediate and Sequence

**Syntax** MTMP

**Range** n/a (Degrees Celsius)

**Access** READ

**Description** MTMP indicates the temperature measured near the motor windings, in degrees Celsius.
The system constantly monitors temperature near the electronics (DTMP) and near the motor windings (MTMP). Either temperature can trigger an alarm or warning if excessive. The alarm limits are set by DTMPMAX and MTMPMAX. Warning limits are set by DTMPWRN and MTMPWRN, and can be used to trigger an output (TEMP) if these limits are exceeded.

**Caution** **Motor temperature depends on operating conditions. Use caution when handling the motor as the case temperature may be very hot.**

**See Also** / (Forward slash), DTMP, DTMPMAX, DTMPWRN, MTMPMAX, MTMPWRN, OUTTEMP, TEMPLV

**Example**

| Command | Description |
|---|---|
| `>MTMPWRN; DTMPWRN` | #Check temperatures which trigger warnings |
| `MTMPWRN=85` | #Temp warning if MTMP exceeds 85 C |
| `DTMPWRN=80` | #Temp warning if DTMP exceeds 80 C |
| `>MTMP; DTMP` | #Check actual temperatures, motor and drive |
| `MTMP=35` | #Motor at 35 C |
| `DTMP=43` | #Drive electronics at 43 C |
| `>SIGTEMP` | #Check temp warning signal |
| `SIGTEMP=0` | #Signal is inactive: motor and drive temps OK. |
| `>` | |

## MTMPMAX : Maximum Motor Temperature — System Control

**Execution Mode** Immediate and Sequence

**Syntax** MTMPMAX n

**Range** n = 0 to 85 (integer values) (degrees Celsius)

**Initial Value** 85

**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 default (initial) value.

**Access** READ and WRITE
READ only in sequences

**Description** MTMPMAX controls the temperature used to trigger the motor overheat protective function. The system will stop motion, turn motor current off, and indicate an alarm condition (alarm code: 26h) if the temperature at the motor windings exceeds MTMPMAX.
The system monitors temperature in two locations: near the drive electronics and near the motor windings. The overheat alarm triggers if either temperature exceeds its programmed limit. MTMPMAX sets the temperature limit for the motor windings.
The system can also provide an early warning of elevated drive or motor temperature (via TEMP output), so that actions may be taken to avoid an alarm and shutdown (e.g. reduce motor current, reduce application throughput, etc.). See discussions at DTMPWRN and MTMPWRN.

**See Also** DTMP, DTMPMAX, DTMPWRN, MTMP, MTMPWRN

**Example**

| Command | Description |
|---|---|
| >DTMPMAX 50<br>DTMPMAX=50 | #Set the drive overheat temperature protective function to activate at a value of 50 degrees Celsius. |
| >DTMPWRN 45<br>DTMPWRN=45 | #Set the device to trigger a warning when the drive temperature exceeds 45 degrees Celsius |
| >MTMPMAX 55<br>MTMPMAX=55 | #Set the Motor Temperature Maximum value |
| >MTMPWRN 50<br>MTMPWRN=50 | #Set the device to trigger a warning when the motor temperature exceeds 50 degrees Celsius |
| >MTMP | #Query the current motor temperature |
| MTMP=35 | #Device response sent to the terminal |
| >DTMP | #Query the drive temperature value |
| DTMP=42 | #Displays the current drive temperature value |
| >SIGTEMP | #Query temperature warning signal |
| SIGTEMP=0 | #SIGTEMP is zero because both drive and motor are below warning limits |
| > | |

## MTMPWRN : Motor Warning Temperature — System Control

**Execution Mode** Immediate

**Syntax** MTMPWARN n

**Range** n = 0 to 85 (Integer values only) (degrees Celsius)

**Initial Value** 85

**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 default (initial) value.

**Access** READ and WRITE

**Description** MTMPWRN controls the motor winding temperature threshold used to control the SIGTEMP temperature warning signal, and TEMP output (if used).
The system monitors temperature in two locations: near the drive electronics and near the motor windings. The temperature warning triggers if either temperature exceeds its programmed warning limit. (MTMPWRN sets the temperature warning limit for the motor windings.)
MTMPWRN (and DTMPWRN) can be used to provide an early warning of elevated drive or motor temperature (via TEMP output), so that actions may be taken to avoid an alarm and shutdown (e.g. reduce motor current, reduce application throughput, etc.). The temperature warning feature could also be used in applications that are very temperature sensitive (e.g. motor current could be disabled and machine operation suspended until temperatures had reduced sufficiently).
SIGTEMP reflects the combined temperature warning status of both the motor and drive. SIGTEMP will be zero (0) if both drive and motor are below their limits, and one (1) if either drive or motor are above their limits. SIGTEMP can be monitored over the serial port if the TEMP output is not configured.

**See Also** DTMP, DTMPMAX, DTMPWRN, MTMP, MTMPMAX, OUTTEMP, TEMPLV, SIGTEMP

**Example**

| Command | Description |
|---|---|
| `>DTMPMAX 50` | #Set the drive overheat temperature protective function to activate at a value of 50 degrees Celsius. |
| `DTMPMAX=50` | |
| `>DTMPWRN 45` | #Set the device to trigger a warning when the drive temperature exceeds 45 degrees Celsius |
| `DTMPWRN=45` | |
| `>MTMPMAX 55` | #Set the Motor Temperature Maximum value |
| `MTMPMAX=55` | |
| `>MTMPWRN 50` | #Set the device to trigger a warning when the motor temperature exceeds 50 degrees Celsius |
| `MTMPWRN=50` | |
| `>MTMP` | #Query the current motor temperature |
| `MTMP=35` | #Device response sent to the terminal |
| `>DTMP` | #Query the drive temperature value |
| `DTMP=42` | #Displays the current drive temperature value |
| `>SIGTEMP` | #Query temperature warning signal |
| `SIGTEMP=0` | #SIGTEMP is zero because both drive and motor are below warning limits |
| `>` | |

## N_xxx : User-Defined Numeric Variables — User Variables

**Execution Mode** Immediate (MODE=0 Only) and Sequence

**Syntax** N_xxx n

**Range** N_xxx can be the name of any existing numeric user-defined variable
n = −(Maximum Number) to +(Maximum Number)

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**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:
LOOP N_COUNT
DIS = N_LONGMOVE
… 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:

- Targets or arguments for assignments (e.g. N_TIME=TIMER; DIS=N_LONGMOVE)
- Loop Counters (e.g. LOOP N_COUNT)
- Conditional Statement Values (e.g. if (VR>N_NOMINAL))
- Parts of Mathematical Expressions (N_SPEED=N_SPEED+N_INCREMENT)
- Targets for interactive data entry commands (N_DISTANCE=KBQ)

Refer to the description of A to Z for more information on general variable use.

**See Also** A to Z, CREATEVAR, DELETEVAR, LISTVAR, S_xxx

**Example**

| Command | Description |
|---|---|
| `>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 : Home Offset Position — Motion Variables

**Execution Mode** Immediate (MODE = 0 Only) and Sequence

**Syntax** OFFSET n

**Range** n = −MAXPOS to +MAXPOS (User Units)

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**Description** OFFSET is the distance to be moved as the last step of a mechanical home seeking operation (MGHN, MGHP).
After the home seeking operation has established a valid home signal (or signal combination: see 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 default value of OFFSET is zero (0): the origin is established at the position where a valid home I/O signal pattern is found. Use OFFSET if the natural system origin differs from the home I/O signal location.

**See Also** HOMETYP, MGHN, MGHP

**Example**

| Command | Description |
|---|---|
| >HOMETYP 6 | #Use HOME and SENSOR. LSx causes reversal. |
| HOMETYP=6 | |
| >OFFSET -30 | #OFFSET origin −30 degree from HOME+SENSOR inputs |
| >OFFSET=-30 deg | |
| >MGHP | #Seek mechanical home, approach from the positive direction. |
| >SIGHOME | #AFTER operation complete: check HOME input |
| SIGHOME=0 | #Input is inactive. We have moved away from the signal. |
| >SIGHOMEP | #Check HOMEP output |
| SIGHOMEP=1 | #Signal is active. We are at PC=0 after a valid homing operation, |
| >PC | #Check position command PC. |
| PC=0 | #Origin. Expected position count after home. |
| >MA 30 | #Absolute move to 30 degrees |
| >PC | #AFTER motion completes… check PC |
| PC=30 deg | #PC is 30 degrees |
| >SIGHOME | #Check Home input |
| SIGHOME=1 | #Active. Home input and origin are separated by OFFSET |
| > | |

## OLTIME : Overload Time **System Control**

**Execution Mode** Immediate

**Syntax** OLTIME n

**Range** n = 0 to 25.000 (seconds)

**Initial Value** 5.0000

**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) value.

**Access** READ and WRITE
READ only in sequences

**Description** OLTIME is the amount of time the motor can be in an OVERLOAD condition (position error is greater than 1.8 degree at rotor shaft), before the system triggers an alarm action based on ALMACT (alarm code: 30h).

**See Also** ALM, ALMACT, ALMCLR, ALMMSG

**Example**

| Command | Description |
|---|---|
| >OLTIME 0.5<br> OLTIME=0.5(5) | #Set OLTIME to it's minimum |
| >SAVEPRM<br> (EEPROM has been written 29 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>AS-One (ASX66)<br>Integrated Motor<br>Software Version: *.**<br>Copyright 2004<br>ORIENTAL MOTOR CO., LTD.<br>------------------------------------------ | #Establish the saved parameter values |
| >OLTIME<br> OLTIME=0.5(0.5)<br>> | #Confirm the new setting |

## OTACT : Overtravel Action — System Control

**Execution Mode** Immediate

**Syntax** OTACT n

**Range** n = 0: Hard Stop (stop as quickly as possible)
1: Soft Stop (controlled deceleration over time)

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**Description** OTACT establishes the stop action taken if MODE=0, when the system detects an overtravel input signal (LSN or LSP) or when position exceeds position limits set with LIMN and LIMP.
If OTACT=0, the system will stop the motor as quickly as possible (hard stop). Stop action is exactly the same as HSTOP.
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.
In Modes 1−3, OTACT has no affect. Overtravel inputs always cause a hard stop.
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 overtravel detectors.**

**See Also** HSTOP, SSTOP, ALMACT, SIGLSP, SIGLSN, INLSN, INLSP, OTLV, LIMN, LIMP, SLACT

**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 0`<br>` OTACT=0(0)` | #Set the overtravel action to Hard Stop. |
| `>ALMACT 2`<br>` ALMACT=2(2)` | #Set Alarm Action to 2 (stop, alarm, current off) |
| `>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.` | #Establish the saved parameter values |

```
------------------------------------------
    AS-One (ASX66)
      Integrated Motor
    Software Version: *.**
       Copyright 2004
   ORIENTAL MOTOR CO., LTD.
------------------------------------------
```

| Command | Description |
|---|---|
| `>MGHP`<br>`>` | #Seek home, start in positive direction (if successful, LIMN and LIMP position limits become active) |

## OTLV : Overtravel Input Level — I/O

**Execution Mode** Immediate

**Syntax** OTLV n

**Range** n = 0: Normally Open
1: Normally Closed

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**Description** OTLV is the active level of the over travel limit inputs, LSN and LSP, if used.

**See Also** INLSN, INLSP, SIGLSN, SIGLSP

**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 0`<br>` OTACT=0(0)` | #Set the overtravel action to Hard Stop. |
| `>ALMACT 2`<br>` ALMACT=2(2)` | #Set Alarm Action to 2 (stop, alarm, current off) |
| `>OTLV 1`<br>` OTLV=0(1)` | #Set the over-travel limit inputs to Normally Closed |
| `>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.` | #Establish the saved parameter values |

```
------------------------------------------
    AS-One (ASX66)
      Integrated Motor
    Software Version: *.**
       Copyright 2004
   ORIENTAL MOTOR CO., LTD.
------------------------------------------
```

| Command | Description |
| --- | --- |
| `>OTLV`<br>` OTLV=1(1)`<br>`>` | #Confirm new value for OTLV |

## OUT : General Output Status I/O

**Execution Mode** Immediate and Sequence

**Syntax** OUT

**Range** n = 0 to 15 (integer values)

**Access** READ and WRITE

**Description** OUT displays or sets the value of all the general purpose outputs, as one integer number.
The general purpose outputs contribute to the value of OUT as follows:

| OUTx | Contribution to OUT if active |
|---|---|
| OUT4 | 8 |
| OUT3 | 4 |
| OUT2 | 2 |
| OUT1 | 1 |

For example, if OUT=10 then OUT2 #2 (2) is ON, and Output#4 is ON (8). (2+8=10)
To check or change the status of a single general output, use the OUTx command.
All general purpose outputs are in the inactive (OFF) state immediately following system startup.

**Caution** **All outputs are OFF when device power is off.**

**See Also** INITIO, IO, IN, OUTTEST, OUTSG, OUTx, REPORT

**Important Interactions** If an output is assigned to a system output signal (OUTHOMEP, OUTALARM, OUTEND, etc) the OUT command will not effect or reflect the electrical I/O port state. The port is always controlled by its assigned signal. Use the OUTSG command to read the status of the assigned system output signals.

**Example**

| Command | Description |
|---|---|
| >IO | #Check IO Status |
| Inputs (1-6) = -LS IN2 IN3 IN4 IN5 +LS<br>Outputs (1-4) = ALARM OUT2 OUT3 OUT4 | #Response: Note that ALARM has been assigned to Output #1. Outputs 2–4 are general purpose. |
| --Inputs--- Outputs<br>1 2 3 4 5 6 -(SEQ.#)- START ABORT - 1 2 3 4<br>0 0 0 0 0 0 -( 0 )- 0 0 - 0 0 0 0 | #All outputs reported off. |
| >OUT 15 | #Set OUT to 15 (all outputs on) |
| OUT=15 | |
| >IO | #Check IO Status again. |
| Inputs (1-6) = -LS IN2 IN3 IN4 IN5 +LS<br>Outputs (1-4) = ALARM OUT2 OUT3 OUT4 | |
| --Inputs--- Outputs<br>1 2 3 4 5 6 -(SEQ#)- START ABORT - 1 2 3 4<br>0 0 0 0 0 0 -( 0 )- 0 0 - 0 1 1 1 | #All outputs are on… expect Output #1. Output 1 active state cannot be effected by OUT |
| > | |

## OUTALARM : ALARM Signal Output Assignment I/O

**Execution Mode** Immediate (MODE = 0 Only)

**Syntax** OUTALARM n

**Range** n = 0 to 4 (integer values only)

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**Description** OUTALARM is the number of the system output assigned to system output signal ALARM. If OUTALARM is zero (0), the ALARM signal is not assigned to a system output.
The ALARM output is set to its active state when the system detects an alarm condition.
The ALARM output is set to its inactive state if the alarm condition is explicitly cleared with the ALMCLR command. Not all alarm conditions can be cleared.
See Section 7.1, "Protective Functions and Troubleshooting", for details on alarm conditions.
In pulse input modes (MODE=1−3), OUTALARM is not available. The ALARM signal is automatically assigned to Output 2.
The active level of the ALARM output is controlled by ALARMLV.

**See Also** SIGALARM, ALARMLV, OUTSG, ALM, ALMCLR

**Example**

| Command | Description |
|---|---|
| `>OUTALARM` | #Check ALARM assignment |
| `OUTALARM=1(1)` | #Assigned to Output #1 |
| `>OUTALARM 3` | #Change the ALARM output 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.` | |

```
-------------------------------------------
    AS-One (ASX66)
      Integrated Motor
    Software Version: *.**
       Copyright 2004
   ORIENTAL MOTOR CO., LTD.
-------------------------------------------
```

| Command | Description |
|---|---|
| `>OUTALARM` | #Confirm the new assignment |
| `OUTALARM=3(3)` | |
| `>` | |

## OUTEND : END Signal Output Assignment — I/O

**Execution Mode** Immediate (MODE = 0 Only)

**Syntax** OUTEND n

**Range** n = 0 to 4

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**Description** OUTEND is the output port assigned to system output signal END (Motion End). If OUTEND is zero (0), the END signal is not assigned to an output.
The END output is set to its active state when the system is not moving and the position error is within +/−1.8 degree (measured at the rotor shaft).
The END output is set to its inactive state if the motor is moving, or if position error is outside of the +/−1.8 degree range.
In pulse input modes (MODE=1−3), OUTEND is not available. The END output is automatically assigned to Output 1.
The active level of the END output is controlled by ENDLV.

**See Also** SIGEND, ENDLV, OUTSG

**Example**

| Command | Description |
|---|---|
| `>OUTEND 2` | #Assign the END output assignment to Output #2 |
| `OUTEND=0(2)` | |
| `>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.` | |

```
-------------------------------------------
     AS-One (ASX66)
       Integrated Motor
     Software Version: *.**
        Copyright 2004
    ORIENTAL MOTOR CO., LTD.
-------------------------------------------
```

| Command | Description |
|---|---|
| `>OUTEND` | #Confirm the new assignment |
| `OUTEND=2(2)` | |
| `>` | |

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

**Execution Mode** Immediate (MODE = 0 Only)

**Syntax** OUTHOMEP n

**Range** n = 0 to 4

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**Description** OUTHOMEP is the output port assigned to system output signal HOMEP (Home Position).
If OUTHOMEP is zero (0), the HOMEP signal is not assigned to an output.
The HOMEP output is set to its active state after a successful homing operation (EHOME, MGHN, MGHP), when the system position command is set to zero (PC=0.000). After that, it is set to its active state whenever the position command is zero.
The HOMEP output is set to its inactive state until a homing operation completes. After that, it is set to its inactive state whenever the position command (PC) is not exactly 0.000.
In pulse input modes (MODE=1−3), OUTHOMEP and the HOMEP signal are not available.
The active level of the HOMEP output is controlled by HOMEPLV.

**See Also** SIGHOMEP, HOMEPLV, OUTSG

**Example**

| Command | Description |
|---|---|
| `>OUTHOMEP 1` | #Assign the HOMEP output assignment to Output #1 |
| `OUTHOMEP=0(1)` | |
| `>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.` | |

```
-------------------------------------------
    AS-One (ASX66)
      Integrated Motor
    Software Version: *.**
       Copyright 2004
   ORIENTAL MOTOR CO., LTD.
-------------------------------------------
```

| Command | Description |
|---|---|
| `>OUTHOMEP` | #Confirm the new assignment |
| `OUTHOMEP=1(1)` | |
| `>` | |

## OUTMBC : Magnetic Brake Control Signal Output Assignment I/O

**Execution Mode** Immediate (MODE = 0 Only)

**Syntax** OUTMBC n

**Range** n = 0 to 4

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**Description** OUTMBC is the output port assigned to system output signal MBC (Magnetic Brake Control).
If OUTMBC is zero (0), the MBC signal is not assigned to an output.
The MBC output is set to its active state when motor current is off (CURRENT=0).
The MBC output is set to its inactive state when motor current is on (CURRENT=1).
In pulse input modes (MODE=1−3), OUTMBC is not available. The MBC output is automatically assigned to Output 4.
The active level of the MBC output is fixed at Normally Closed, and cannot be changed. The electrical state of the I/O port when CURRENT=0 is identical to the state when DC power is off.

**See Also** SIGMBC, OUTSG, CURRENT

**Example**

| Command | Description |
|---|---|
| `>OUTMBC 4` | #Assign the MBC output assignment to Output #4 |
| `OUTMBC=0(4)` | |
| `>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.` | |

```
------------------------------------------
    AS-One (ASX66)
      Integrated Motor
    Software Version: *.**
       Copyright 2004
   ORIENTAL MOTOR CO., LTD.
------------------------------------------
```

| Command | Description |
|---|---|
| `>OUTMBC` | #Confirm the new assignment |
| `OUTMBC=4(4)` | |
| `>` | |

## OUTMOVE : MOVE Signal Output Assignment I/O

**Execution Mode** Immediate (MODE = 0 Only)

**Syntax** OUTMOVE n

**Range** n = 0 to 4

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**Description** OUTMOVE is the output port assigned to system output signal MOVE (Motor Moving).
If OUTMOVE is zero (0), the MOVE signal is not assigned to an output.
The MOVE output is set to its active state while the motor is executing a motion command.
The MOVE output is set to its inactive state when the motor is not executing a motion command.
In pulse input modes (MODE=1−3), OUTMOVE and the MOVE signal are not available.
The active level of the MOVE output is controlled by MOVELV.

**See Also** SIGMOVE, MOVELV, OUTSG

**Example**

| Command | Description |
|---|---|
| `>OUTMOVE 3` | #Assign the MOVE output assignment to Output #3 |
| `OUTMOVE=0(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.` | |

```
-------------------------------------------
    AS-One (ASX66)
      Integrated Motor
    Software Version: *.**
       Copyright 2004
   ORIENTAL MOTOR CO., LTD.
-------------------------------------------
```

| Command | Description |
|---|---|
| `>OUTMOVE` | #Confirm the new assignment |
| `OUTMOVE=3(3)` | |
| `>` | |

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

**Execution Mode** Immediate (MODE = 0 Only)

**Syntax** OUTPSTS n

**Range** n = 0 to 4

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**Description** OUTPSTS is the output port assigned to system output signal PSTS (Pause Status). If OUTPSTS is zero (0), the PSTS signal is not assigned to an output.
The PSTS output is set to its active state when a motion has been paused, by PAUSE command or PAUSE input.
The PSTS output is set to its inactive state when motion has not been paused, when a paused motion has been resumed with by a CONT command or START input, and when a paused motion has been cleared by a PAUSECL (Pause Clear) input or PAUSECLR command.
In pulse input modes (MODE=1−3), OUTPSTS and the PSTS signal are not available.
The active level of the PSTS output is controlled by PSTSLV.

**See Also** SIGPSTS, PSTSLV, OUTSG, PAUSE, PAUSECLR, CONT

**Example**

| Command | Description |
|---|---|
| >OUTPSTS 2 | #Assign the PSTS output assignment to Output #2 |
| OUTPSTS=0(2) | |
| >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. | |

```
------------------------------------------
    AS-One (ASX66)
      Integrated Motor
    Software Version: *.**
       Copyright 2004
   ORIENTAL MOTOR CO., LTD.
------------------------------------------
```

| Command | Description |
|---|---|
| >OUTPSTS | #Confirm the new assignment |
| OUTPSTS=2(2) | |
| > | |

## OUTRUN : RUN Signal Output Assignment — I/O

**Execution Mode** Immediate (MODE = 0 Only)

**Syntax** OUTRUN n

**Range** n = 0 to 4

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**Description** OUTRUN is output port assigned to system output signal RUN (Sequence Running). If OUTRUN is zero (0), the RUN signal is not assigned to an output.
The RUN output is set to its active state while the sequences are executing.
The RUN output is set to its inactive state when sequences are not executing.
In pulse input modes (MODE=1−3), OUTRUN and the RUN signal are not available.
The active level of the RUN output is controlled by RUNLV.

**See Also** SIGRUN, RUNLV, OUTSG

**Example**

| Command | Description |
|---|---|
| >OUTRUN 1<br>OUTRUN=0(1) | #Assign the RUN output assignment to Output #1 |
| >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>AS-One (ASX66)<br>Integrated Motor<br>Software Version: *.**<br>Copyright 2004<br>ORIENTAL MOTOR CO., LTD.<br>----------------------------------------- | #Establish the saved parameter values |
| >OUTRUN<br>OUTRUN=1(1)<br>> | #Confirm the new assignment |

## OUTSG : System Output Signal Status I/O

**Execution Mode** Immediate and Sequence

**Syntax** OUTSG n

**Range** n = 0 to 255

**Initial Value** 0

**Access** READ

**Description** OUTSG displays the current status of all the system output signals, as one integer number.
The system output signals contribute to the value of OUTSG as follows:

| Bit Location | Signal | Contribution to OUTSG if active |
|---|---|---|
| Bit 0 | MOVE | 1 |
| Bit 1 | RUN | 2 |
| Bit 2 | END | 4 |
| Bit 3 | HOMEP | 8 |
| Bit 4 | ALARM | 16 |
| Bit 5 | PSTS | 32 |
| Bit 6 | TEMP | 64 |
| Bit 7 | MBC | 128 |

OUTSG is the sum of the contribution of all active signals:
- If OUTSG=2, the RUN signal is active, and all other signals are inactive.
- If OUTSG=192, the TEMP (64) and MBC (128) signals are active (64+128=192), and all other signals are inactive.

Be careful not to confuse OUTSG with OUT (Output Status). OUT reports the status of General Purpose Outputs (those outputs which are not assigned to a signal). OUTSG reports the status of system output signals.
System output signals are always maintained in their appropriate state, even in the signals are not assigned to outputs.

**See Also** SIGMOVE, SIGRUN, SIGEND, SIGHOMEP, SIGALARM, SIGPSTS, SIGTEMP, OUT

**Example**

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

## OUTTEMP : TEMP Signal Output Assignment I/O

**Execution Mode** Immediate (MODE = 0 Only)

**Syntax** OUTTEMP n

**Range** n = 0 to 4

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**Description** OUTTEMP is the output port assigned to system output signal TEMP (Temperature Warning).
If OUTTEMP is zero (0), the TEMP signal is not assigned to an output.
The TEMP output is set to its active state if drive electronics temperature DTMP exceeds drive temperature warning limit DTMPWRN, or if motor winding temperature exceeds temperature warning limit MTMPWRN.
The TEMP output is set to its inactive state when both DTMP and MTMP are below their respective warning levels.
In pulse input modes (MODE=1−3), OUTTEMP is not available. The TEMP output is automatically assigned to Output 3.
The active level of the TEMP output is controlled by TEMPLV.

**See Also** SIGTEMP, TEMPLV, OUTSG, MTMP, MTMPWRN, DTMP, DTMPWRN

**Example**

| Command | Description |
|---|---|
| `>OUTTEMP 4` | #Assign the TEMP output assignment to Output #4 |
| `OUTTEMP=0(4)` | |
| `>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.` | |

```
------------------------------------------
    AS-One (ASX66)
      Integrated Motor
    Software Version: *.**
       Copyright 2004
   ORIENTAL MOTOR CO., LTD.
------------------------------------------
```

| Command | Description |
|---|---|
| `>OUTTEMP` | #Confirm the new assignment |
| `OUTTEMP=4(4)` | |
| `>` | |

## OUTTEST : I/O Test Utility I/O

**Execution Mode** Immediate

**Syntax** OUTTEST

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

| | | | |
|---|---|---|---|
| OUT1 | 1 | OUT2 | 2 |
| OUT3 | 3 | OUT4 | 4 |
| MOVE | M | RUN | R |
| END | E | HOMEP | H |
| ALARM | A | PSTS | P |
| TEMP | T | MBC | B |

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.

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

**Example**

Command / Description

```
>OUTTEST                                              #Start the OUTTEST process

        *** Input Monitor -- Output Simulator ***

 Inputs  (1-6) = IN1 IN2 -LS +LS HOME PSTOP           #Assignments and toggle keys
 Outputs (1-4) = OUT1(1) OUT2(2) END(E) ALARM(A)      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 -(SEQ#)- START ABORT - 1 2 3 4           #Active (1) or inactive (0) states
 0 0 0 0 1 0 -(  0  )-   0     0   - 0 0 1 0           shown here
>                                                     #Escape entered: OUTTEST
                                                      ends
```

## OUTx : Individual General Output Control — I/O

**Execution Mode** Immediate and Sequence

**Syntax** OUTx n

**Range** x = 1 to 4
n = 0: Not Active
1: Active

**Description** OUTx controls the state of General Purpose Output 'x'.
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 OUTSG to check the status of assigned system output signals.

**See Also** INITIO, OUT, OUTSG, OUTTEST

**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. |
| `>` | |

## OVERFLOW : Position Error Limit — System Control

**Execution Mode** Immediate and Sequence

**Syntax** OVERFLOW n

**Range** n = 0.001 to MAXOVERFLOW (User Units)

**Initial Value** 3 (User Units)

**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) value.

**Access** READ and WRITE
READ only in sequences

**Description** OVERFLOW is the position error limit. If the position error (PE) exceeds OVERFLOW, an alarm can be triggered (alarm code:10h). Alarm action is determined by ALMACT.

**See Also** ALM, ALMACT, DPR, GA, GB, PE, OVERVEL, MAXPOS, MAXOVERFLOW

**Important Interactions** If the distance per revolution (DPR) or electronic gearing factors (GA, GB) are changed, the system automatically rescales OVERFLOW and OVERVEL to values that are physically equivalent to their factory defaults. OVERFLOW is set to the equivalent of 3 revolutions, and OVERVEL is set to the equivalent of 100 revolutions per second. (The rescaling helps prevent nuisance alarms when DPR, GA and GB are programmed). The largest possible value for OVERFLOW (MAXOVERFLOW) is also changed when DPR is changed. Set DPR, GA and GB before programming OVERFLOW and OVERVEL. See Section 4.3, "Initial Setting (User Unit)", for more detail.

**Example**

| Command | Description |
|---|---|
| `>UU deg`<br>`UU=deg` | #Set user units to degrees (deg) |
| `>DPR 360`<br>`DPR=1(360) deg`<br>`OVERFLOW, OVERVEL is re-scaled to default equivalent.` | #Change Distance-per-revolution to 360 (degrees) |
| `OVERFLOW=3(1080) deg` | #OVERFLOW rescaled to ~ 3 revs |
| `OVERVEL=100(36000) deg/sec` | #OVERVEL rescaled to ~ 100 RPS |
| `Position range = +/- 41943(499680)` | #New position range |
| `Velocity range = 0.001 - 83.333(30000)` | #New velocity range |
| `>OVERFLOW 45`<br>`OVERFLOW=3(45) deg` | #Set OVERFLOW to 45 degrees. |
| `>OVERVEL 7200`<br>`OVERVEL=100(7200) deg/sec` | #Set OVERVEL to 7200 deg/sec. |
| `>SAVEPRM`<br>`(EEPROM has been written 86 times)`<br>`Enter Y to proceed, other key to cancel. Y`<br>`Saving Parameters........OK.` | #Save changes |
| `>RESET`<br>`Resetting system.` | #Reset to make new values active |

```
------------------------------------------
    AS-One (ASX66)
      Integrated Motor
    Software Version: *.**
       Copyright 2004
   ORIENTAL MOTOR CO., LTD.
------------------------------------------
```

| Command | Description |
|---|---|
| `>OVERFLOW`<br>`OVERFLOW=45(45) deg`<br>`>` | #Confirm new value of OVERFLOW |

## OVERVEL : Velocity Limit — System Control

**Execution Mode** Immediate and Sequence

**Syntax** OVERVEL n

**Range** n = 0.001 to (1.2 * MAXVEL) (User Units/second)

**Initial Value** 100 (User Units/second)

**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) value.

**Access** READ and WRITE
READ only in sequences

**Description** OVERVEL is the velocity limit. If the velocity (VF) exceeds OVERVEL, an alarm can be triggered (alarm code: 31h). Alarm action is determined by ALMACT.
OVERVEL should not be confused with the maximum possible velocity entry, MAXVEL.

**See Also** ALM, ALMACT, DPR, GA, GB, VF, OVERFLOW, MAXVEL

**Important Interactions** If the distance per revolution (DPR) or electronic gearing factors (GA, GB) are changed, the system automatically rescales OVERFLOW and OVERVEL to values that are physically equivalent to their factory defaults. OVERFLOW is set to the equivalent of 3 revolutions, and OVERVEL is set to the equivalent of 100 revolutions per second. (The rescaling helps prevent nuisance alarms when DPR, GA and GB are programmed). The largest possible value for OVERVEL (1.2 * MAXVEL) is also changed when DPR, GA or GB is changed. Set DPR, GA and GB before programming OVERFLOW and OVERVEL. See Section 4.3, "Initial Setting (User Unit)", for more detail.

**Example**

| Command | Description |
|---|---|
| `>UU deg`<br>` UU=deg` | #Set user units to degrees (deg) |
| `>DPR 360`<br>` DPR=1(360) deg`<br>` OVERFLOW, OVERVEL is re-scaled to default equivalent.` | #Change Distance-per-revolution to 360 (degrees) |
| ` OVERFLOW=3(1080) deg` | #OVERFLOW rescaled to ~ 3 revs |
| ` OVERVEL=100(36000) deg/sec` | #OVERVEL rescaled to ~100 RPS |
| ` Position range = +/- 41943(499680)` | #New position range |
| ` Velocity range = 0.001 - 83.333(30000)` | #New velocity range |
| `>OVERFLOW 45`<br>` OVERFLOW=3(45) deg` | #Set OVERFLOW to 45 degrees. |
| `>OVERVEL 7200`<br>` OVERVEL=100(7200) deg/sec` | #Set OVERVEL to 7200 deg/sec. |
| `>SAVEPRM`<br>` (EEPROM has been written 86 times)`<br>` Enter Y to proceed, other key to cancel. Y`<br>` Saving Parameters........OK.` | #Save changes |
| `>RESET`<br>` Resetting system.` | #Reset to make new values active |

```
-------------------------------------------
    AS-One (ASX66)
      Integrated Motor
    Software Version: *.**
       Copyright 2004
   ORIENTAL MOTOR CO., LTD.
-------------------------------------------
```

| Command | Description |
|---|---|
| `>OVERVEL`<br>` OVERVEL=7200(7200) deg/sec`<br>`>` | #Confirm new value of OVERVEL |

## PAUSE : Pause Motion — Motion Commands

**Execution Mode** Immediate (MODE = 0 Only) and Sequence

**Syntax** PAUSE

**Description** PAUSE interrupts a motion, stopping the motor by controlled deceleration (soft stop). See SSTOP for details on the velocity profile during deceleration.
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.
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.
The system remains in a "paused" state, until motion is continued (see CONT), or the state is explicitly cleared (with a PAUSECL input or PAUSECLR command), or another motion command is executed.
If no motion is in process when a PAUSE command is issued, the PAUSE command has no effect.
Motions may also be paused by assigning system input signal PAUSE to an input. Operation after PAUSE becomes active is identical to issuing a PAUSE command.

**See Also** SSTOP, CONT, INITIO, INPAUSE, INPAUSECL, OUTPSTS, OUTSG, PAUSECLLV, PAUSECLR, PAUSELV, PSTSLV, SIGPAUSE, SIGPAUSECL, SIGPSTS

**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 PAUSECLR command, or new motion).
Pause and Continue operations are not supported for Linked Motions (MIx). PAUSE during a Linked Motion causes a soft stop, and subsequent CONT commands are ignored.

**Example**

| Command | Description |
|---|---|
| `>LIST CHKJAM` | #List sequence CHKJAM |
| `(  1) DIS=10; VR=10` | #Set motion parameter |
| `(  2) LOOP` | #Start infinite loop |
| `(  3)   MI` | #Start move incremental |
| `(  4)   WHILE (TF<0.5)` | #Check if over loaded |
| `(  5)     IF (SIGMOVE=0)` | #Check for motion end |
| `(  6)       BREAKW` | #Exit while loop, if so |
| `(  7)     ENDIF` | |
| `(  8)   WEND` | |
| `(  9)   IF (SIGMOVE!=0)` | #Check if moving |
| `( 10)     PAUSE` | #TF>0.5: PAUSE motion |
| `( 11)     WAIT TD` | #Wait for stop, send text, get response |
| `( 12)     SAS System in trouble.` | |
| `( 13)      SACS Enter 1 to continue, other to stop:` | |
| `( 14)     A=KBQ; SACS ^M^J>` | |
| `( 15)     IF (A=1)` | |
| `( 16)       CONT; MEND` | #CONTinue, if A=1 |
| `( 17)     ELSE` | |
| `( 18)       SAS Operation stopped.` | #Otherwise, report stopped |
| `( 19)       RET` | #Return from sequence |
| `( 20)     ENDIF` | |
| `( 21)   ENDIF` | |
| `( 22)   SAS Motion end, goto next.` | #Send normal message |
| `( 23)   WAIT 1` | #Dwell 1 sec., loop back to top. |
| `( 24) ENDL` | |
| `>RUN CHKJAM` | #Execute sequence CHKJAM |
| `>Motion end, goto next.` | #Normal message |
| `>Motion end, goto next.` | #Normal message |
| `>System in trouble.` | #Over loaded message |
| `>Enter 1 to continue, other to stop:1` | #Prompt message -> Entry "1" |
| `>Motion end, goto next.` | #Normal message |
| `>Motion end, goto next.` | #Normal message |
| `>System in trouble.` | #Overloaded message |
| `>Enter 1 to continue, other to stop:2` | #Prompt message -> Entry "2" |
| `>Operation stopped.` | #Finished message (Sequence finish) |
| `>` | |

## PAUSECLR :Clear State of Paused Motion — Motion Commands

**Execution Mode** Immediate (MODE=0 Only) and Sequence

**Syntax** PAUSECLR

**Description** PAUSECLR clears the state of any paused motion.

Any remaining motion is “forgotten”, and the previously paused motion cannot be continued (the CONT command will be ignored). If motion was paused, the PSTS signal will be cleared by the PAUSECLR command.

If motion was not paused when a PAUSECLR command is issued, the PAUSECLR command has no effect.

Paused motions may also be cleared by assigning system input signal PAUSECL to an input. Operation after PAUSECL becomes active is identical to issuing a PAUSECLR command.

**See Also** SSTOP, CONT, INITIO, INPAUSE, INPAUSECL, OUTPSTS, OUTSG, PAUSECLLV, PAUSE, PAUSELV, PSTSLV, SIGPAUSE, SIGPAUSECL, SIGPSTS

**Example**

| Command | Description |
|---|---|
| `>LIST RESUME` | #List sequence RESUME |
| `(  1) IF (CURRENT=1)` | #If motor current applied |
| `(  2)   CONT` | #Continue previous motion |
| `(  3) ELSE` | #Otherwise (no motor current) |
| `(  4)   PAUSECLR` | #Clear pause status |
| `(  5) ENDIF` | |
| `>` | |

## PAUSECLLV : Pause Clear Input Level I/O

**Execution Mode** Immediate (MODE = 0 Only)

**Syntax** PAUSECLLV n

**Range** n = 0: Normally Open
1: Normally Closed

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**Description** PAUSECLLV is the active level of the Pause Clear (PAUSECL) input, if used.

**See Also** INPAUSECL, SIGPAUSECL

**Example**

| Command | Description |
|---|---|
| >INPAUSECL 1<br> INPAUSECL 0(1) | #Assign the PAUSECL input to input #1 |
| >PAUSECLLV=1<br> PAUSECLLV=0(1) | #Set the PAUSECL input logic level to Normally Closed |
| >SAVEPRM<br> (EEPROM has been written 14 times)<br> Enter Y to proceed, other key to cancel. Y<br> Saving Parameters........OK. | #Save the parameter assignments |
| >RESET<br> Resetting system. | #Establish the saved parameter values |

```
------------------------------------------
    AS-One (ASX66)
      Integrated Motor
    Software Version: *.**
       Copyright 2004
   ORIENTAL MOTOR CO., LTD.
------------------------------------------
```

| Command | Description |
|---|---|
| >PAUSECLLV<br> PAUSECLLV=1(1)<br>> | #Confirm the new value of PAUSECLLV |

## PAUSELV : PAUSE Input Level I/O

**Execution Mode** Immediate (MODE = 0 Only)

**Syntax** PAUSELV n

**Range** n = 0: Normally Open
1: Normally Closed

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**Description** PAUSELV is the active level of the Pause (PAUSE) input, if used.

**See Also** INPAUSE, SIGPAUSE

**Example**

| Command | Description |
|---|---|
| `>INPAUSE 1`<br>` INPAUSE 0(1)` | #Assign the PAUSE input to input #1 |
| `>PAUSELV=1`<br>` PAUSELV=0(1)` | #Set the PAUSE input logic level to Normally Closed |
| `>SAVEPRM`<br>` (EEPROM has been written 14 times)`<br>` Enter Y to proceed, other key to cancel. Y`<br>` Saving Parameters........OK.` | #Save the parameter assignments |
| `>RESET`<br>` Resetting system.` | #Establish the saved parameter values |

```
-------------------------------------------
    AS-One (ASX66)
      Integrated Motor
    Software Version: *.**
       Copyright 2004
   ORIENTAL MOTOR CO., LTD.
-------------------------------------------
```

| Command | Description |
|---|---|
| `>PAUSELV`<br>` PAUSELV=1(1)`<br>`>` | #Confirm the new value of PAUSELV |

## PC : Position Command — System Status

**Execution Mode** Immediate and Sequence

**Syntax** PC n

**Range** n = −MAXPOS to +MAXPOS (User Units)

**Initial Value** 0

**Access** READ and WRITE
READ only while motion is in progress

**Description** PC is the position command (or setpoint), 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). 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 (MODE=0), or by external pulse counts (MODE=1−3).
- 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, MGHN, MGHP).

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.

**See Also** DPR, GA, GB, EHOME, MA, MGHN, MGHP, MI, PCI, PE, PF, PFI, MAXPOS

**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 — System Status

**Execution Mode** Immediate (MODE = 0 Only) and Sequence

**Syntax** PCI

**Range** −MAXPOS to +MAXPOS (User Units)

**Access** READ

**Description** PCI is the change in position command PC since the last motion started.
PCI is continuously updated by the system.
PCI is set to zero (0) at system startup. PCI is undefined immediately after a mechanical home seeking operation completes (MGHN, MGHP).

**See Also** DPR, GA, GB, PC, PE, PF, PFI

**Example**

| Command | Description |
|---|---|
| `>LIST AREAOUT2` | |
| `(  1) DIS 100; VR=10` | #Set distance, velocity |
| `(  3) MI` | #Start move incremental |
| `(  4) SAS Motion started` | #Send message #1 |
| `(  5) WHILE (PCI<30)` | #Wait for PCI to reach 30 |
| `(  6) WEND` | |
| `(  7) SAS Passed 30mm` | #Send message #2 |
| `(  8) WHILE (PCI<60)` | #Wait for PCI 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 AREAOUT2` | #Start sequence |
| `>Motion started` | #Message #1 |
| `>Passed 30mm` | #Message #2 |
| `>Passed 60mm` | #Message #3 |
| `>Reached target` | #Message #4 |
| `>` | |

## PE : Position Error — System Status

**Execution Mode** Immediate and Sequence

**Syntax** PE

**Range** −MAXPOS to +MAXPOS (User Units)

**Access** READ

**Description** PE is the position error, or the difference between commanded position (PC) and actual position (PF), in user units. PE = PC – PF.

PE is continuously updated by the system, and can be used to monitor the systems response to load conditions.

The system uses position error information to detect two alarm conditions:

An OVERFLOW alarm (alarm 10h) occurs if position error ranges outside a programmable maximum (+/−OVERFLOW).

A Warning is logged if position error ranges outside +/−1.8 degrees (measured at the motor shaft). A warning message will be automatically transmitted if ALMMSG=2. An OVERLOAD alarm (alarm 30h) occurs if the position error stays outside this range for longer than a programmable maximum time period (OLTIME).

**See Also** DPR, GA, GB, PC, PCI, PF, PFI, OVERFLOW, OLTIME, ALMMSG

**Example**

| Command | Description |
|---|---|
| >LIST CHECKLOAD | #List sequence CHECKLOAD |
| ( 1) MCP | #Start continuous motion, positive |
| ( 2) WHILE (IN1=0) | #While Input 1 is off. |
| ( 3) D=PE-E | #Capture position error, and… |
| ( 4) D=0.01*D | #…Form a simple moving… |
| ( 5) E=E+D | #…average in variable E. |
| ( 6) WEND | #End of WHILE block |
| ( 7) SSTOP | #When IN1=1: Start soft stop |
| ( 8) MEND | #Wait for motion to stop |
| ( 9) IF (E>3) | #E = averaged position error |
| ( 10) SAS Load increasing, clean machine. | #if high, send reminder |
| ( 11) ENDIF | #End of IF block |
| > | |

## PF : Motor Position **System Status**

**Execution Mode** Immediate and Sequence

**Syntax** PF

**Range** −MAXPOS to +MAXPOS (User Units)

**Access** READ

**Description** PF is the actual motor position, measured by the internal position sensor.
PF is continuously updated by the system.
PF can deviate from the commanded position PC, depending on load conditions. The difference between PC and PF is the position error PE.
PF cannot be set directly, but does 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).

**See Also** DPR, GA, GB, PC, PCI, PE, PFI

**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 Motor Position — System Status

**Execution Mode** Immediate (MODE = 0 Only) and Sequence

**Syntax** PFI

**Range** −MAXPOS to +MAXPOS (User Units)

**Access** READ

**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. PFI is undefined immediately after a mechanical home seeking operation completes (MGHN, MGHP).

**See Also** DPR, GA, GB, PC, PCI, PE, PF

**Example**

| Command | Description |
|---|---|
| `>LIST 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 |
| `>` | |

## POS [x] : Position Array Data — Motion Variables

**Execution Mode** Immediate (MODE = 0 Only) and Sequence

**Syntax** POS [x] n

**Range** x = 1 to 100
n = −MAXPOS to +MAXPOS

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**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.:
MA POS [7]
…will start an absolute motion to the position stored in POS [7].
POS [x] data may be entered directly if known, or positions can be interactively found and stored using the TEACH function. See Section 4.7, "Support Functions" for more information on the TEACH function.
All POS [x] data can be cleared (initialized to zero) with the CLEARPOS command.

**See Also** MA, PC, TEACH, CLEARPOS, CLEARALL

**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] |
| `>` | |

## PSTOP : Panic Stop — Motion Commands

**Execution Mode** Immediate (MODE = 0 Only) and Sequence

**Syntax** PSTOP

**Description** PSTOP stops the motor as quickly as possible (hard stop), and then takes the alarm action determined by ALMACT, which may involve setting an alarm (alarm 68h), aborting sequences, and possibly disabling motor current.
The PSTOP command operates independently of the motor stop action setting (MSTOPACT) .
The PSTOP function may also be executed via the PSTOP input. See the INPSTOP command to assign the PSTOP input.

There are several different ways to stop the motor. See the Motor Stop Command list for more information.

**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.**

**See Also** <ESC>, MA, MCN, MCP, MGHN, MGHP, MI, EHOME, SSTOP, HSTOP, MSTOP, INPSTOP, ALMACT, ABORT

**Example**

| 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 |
| `>` | |

## PSTOPLV : Panic Stop Input Level — I/O

| | |
|---|---|
| **Execution Mode** | Immediate |
| **Syntax** | PSTOPLV n |
| **Range** | n = 0: Normally Open<br>1: Normally Closed |
| **Initial Value** | 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 default (initial) value. |
| **ACCESS** | READ and WRITE |
| **Description** | PSTOPLV is the active level of the Panic Stop (PSTOP) input, if used. |
| **See Also** | INPSTOP, SIGPSTOP |

**Example**

| Command | Description |
|---|---|
| `>INPSTOP 1`<br>` INPSTOP 0(1)` | #Assign the PSTOP input to input #1 |
| `>PSTOPLV=1`<br>` PSTOPLV=0(1)` | #Set the PSTOP input logic level to Normally Closed |
| `>SAVEPRM`<br>` (EEPROM has been written 14 times)`<br>` Enter Y to proceed, other key to cancel. Y`<br>` Saving Parameters........OK.` | #Save the parameter assignments |
| `>RESET`<br>` Resetting system.` | #Establish the saved parameter values |

```
 ------------------------------------------
     AS-One (ASX66)
       Integrated Motor
     Software Version: *.**
        Copyright 2004
    ORIENTAL MOTOR CO., LTD.
 ------------------------------------------
```

| Command | Description |
|---|---|
| `>PSTOPLV`<br>` PSTOPLV=1(1)`<br>`>` | #Confirm the new value of PSTOPLV |

## PSTSLV : Pause Status Output Level — I/O

**Execution Mode** Immediate (MODE = 0 Only)

**Syntax** PSTSLV n

**Range** n = 0: Normally Open
1: Normally Closed

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**Description** PSTSLV is the active level of the Pause Status (PSTS) output, if used.

**See Also** OUTPSTS, SIGPSTS

**Example**

| Command | Description |
| --- | --- |
| `>OUTPSTS 4`<br>`OUTPSTS 0(4)` | #Assign the PSTS output to output #4 |
| `>PSTSLV=1`<br>`PSTSLV=0(1)` | #Set the PSTS output logic level to Normally Closed |
| `>SAVEPRM`<br>`(EEPROM has been written 14 times)`<br>`Enter Y to proceed, other key to cancel. Y`<br>`Saving Parameters........OK.` | #Save the parameter assignments |
| `>RESET`<br>`Resetting system.` | #Establish the saved parameter values |

```
-------------------------------------------
    AS-One (ASX66)
      Integrated Motor
    Software Version: *.**
       Copyright 2004
   ORIENTAL MOTOR CO., LTD.
-------------------------------------------
```

| Command | Description |
| --- | --- |
| `>PSTSLV`<br>`PSTSLV=1(1)`<br>`>` | #Confirm the new value of PSTSLV |

## RC : Motor Shaft Position — System Status

**Execution Mode** Immediate and Sequence

**Syntax** RC

**Range** 0 to 359.999 (Angular Degrees)

**Access** READ

**Description** RC is the actual motor shaft position, in units of mechanical degrees, within one revolution of the motor shaft.
RC provides an alternative way to check motor position, which may be helpful in some cases (e.g. if system resolution is fairly low, RC may be used to detect small or slow motions).
RC is independent of direction inversion (DIRINV), and always advances counterclockwise. RC rolls over from 359.999 degrees to 0.000 degrees.
RC is based on position sensor information. At system startup, RC is calculated relative to the nearest zero (0) sensor angle, in the clockwise direction. Zero sensor angles occur at fifty (50) evenly-spaced locations per motor rotation, so RC should not be considered an "absolute" motor angle.

**See Also** DPR, PC, UU

**Example**

| Command | Description |
|---|---|
| `>RC`<br>` RC=184.014` | #Check motor position, mechanical degrees. |
| `>RESET`<br>`Resetting system.` | #Reset the system |

```
------------------------------------------
     AS-One (ASX66)
       Integrated Motor
     Software Version: *.**
        Copyright 2004
    ORIENTAL MOTOR CO., LTD.
------------------------------------------
```

| Command | Description |
|---|---|
| `>RC`<br>` RC=4.204`<br>`>` | #Check RC again. Now almost 180 degrees different, because of RESET. Actual motion only 0.19 degrees. |

## REN : Rename Sequence — Editor Command

**Execution Mode** Immediate (MODE = 0 Only)

**Syntax** REN target newname

**Range** 'target' must be the name or number of an existing sequence.
'newname' must be a valid sequence name (consisting of letters or numbers, 10 character maximum, must start with a letter).

**Description** REN renames an existing sequence. The new name must be unique.
REN can also be used to name a sequence which was created by number only, and has no name.
'target' cannot be renamed if it is locked.

**See Also** COPY, DIR, EDIT, DEL, LOCK, UNLOCK

**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  2048 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  2048 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 — Monitor Commands

**Execution Mode** Immediate

**Syntax** REPORT

**Description** REPORT display a system status summary.
The REPORT command can be an effective tool for troubleshooting problems with the system. The REPORT command displays the status and active level of all of the system's inputs an outputs, the values of important parameters, the value of position command PC, and the alarm and warning history.

**See Also** ALM, DIR, IO

**Example**

| Command | Description |
|---|---|
| >REPORT | #Get a system status summary: sample result follows |

```
/ I/O REPORT /---(NO:Normally Open,NC:Normally Close)-------------

   START(NO) = 0   ABORT(NO) = 0   -LS(NO) = 0   HOME(NO) = 0
   CROFF(NC) = 1   +LS(NO) = 0   IN5(NO) = 0   IN6(NO) = 0
   MOVE(NO) = 0   RUN(NO) = 0   END(NO) = 1   ALARM(NC) = 0

/ PARAMETER REPORT /----------------------------------------------

   STRSW = 1   DPR = 360   GA = 1   GB = 1
   DIRINV = 0 (+DIS Value = CW Rotation)
   CRRUN = 100   CRSTOP = 50   FILT = 0.003
   VS = 0   VR = 3600   TA = 0.5   TD = 0.5   DIS = 360
   LIMP = 0   LIMN = 0
   OLTIME = 5   OVERFLOW = 1080

/ POSITION REPORT /-----------------------------------------------

   PC = 136.652 [deg]

/ ALARM HISTORY /-------------------------------------------------

   ALARM = 00 ,  RECORD : 31 30 23 98 9E 70 30 00 00 00
   No alarm.
>
```

## RESET : Reset Device — System Control

**Execution Mode** Immediate

**Syntax** RESET

**Description**

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 (MBC) 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, Current Off (CROFF) input, if used, and STRSW setting), current is enabled, and begins to increase from 0 to CRSTOP value.
7) Outputs (other than MBC ) 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 MBC output is configured, the MBC output is set to its closed, conducting state when current reaches CRSTOP.
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, 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.**
**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.**

**See Also** STRSW, CRSTOP, INCROFF, CROFFLV, VERBOSE, SAVEPRM, SAVEPOS, SAVEALL

**Example**

| Command | Description |
|---|---|
| >ALMACT 1<br>ALMACT=2(1) | #New value of Alarm Action: stop, alarm, keep current on. Active (Future) values |
| >ALMMSG 2<br>ALMMSG=2 [Alarm+Warning] | #New value for Alarm Messaging: Transmit message for alarm and warning. |
| >SAVEPRM<br>(EEPROM has been written 95 times)<br>Enter Y to proceed, other key to cancel. Y<br>Saving Parameters........OK. | #Save all parameters |
| >RESET<br>Resetting system.<br>------------------------------------------<br>AS-One (ASX66)<br>Integrated Motor<br>Software Version: *.**<br>Copyright 2004<br>ORIENTAL MOTOR CO., LTD.<br>------------------------------------------ | #Reset the system to make new value of ALMACT active (ALMMSG change was effective immediately). Reset messages follow. |
| >ALMACT; ALMMSG<br>ALMACT=1(1)<br>ALMMSG=2 [Alarm+Warning]<br>> | #Confirm new values of ALMACT and ALMMSG. |

## RET : Sequence Return — Sequence Commands

**Execution Mode** Sequence

**Syntax** RET

**Description** RET terminates processing of the current sequence.
If the sequence was CALL'ed from another sequence, the original sequence will resume at the statement following the CALL statement. If the sequence was started with the START input or the RUN command, RET terminates all sequence execution.
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).

**See Also** ABORT, CALL, END

**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 |
| `>` | |

## RMODE : Ramp Mode — Motion Variables

**Execution Mode** Immediate (MODE = 0 Only) and Sequence

**Syntax** RMODE n

**Range** n = 0: Linear Acceleration Profile
1: Automatic Profile

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE
READ only while motion is in progress

**Description** RMODE selects the ramp mode for motion profiles.
When RMODE=0, the system changes speed linearly, over acceleration and deceleration times TA and TD.
When RMODE=1 (automatic profiling), the system automatically calculates a profile designed to maintain torque utilization TU. At lower speeds, the profile will be more aggressive, and at higher speeds, the profile will be less aggressive: the system attempts to match acceleration and deceleration rates to the available torque, which decreases as speed increases. Start speed VS, and acceleration and deceleration times TA and TD are not used when RMODE=1.
Automatic profiling depends on knowledge of load conditions. Load inertia (LI), load friction (LF), and gravity loading (LG) should be configured to reasonably accurate values before automatic profiling is used. The Load Estimation facility (LDCHK) can be used to automatically estimate these load conditions.
For accurate torque information, automatic profiling also depends on a reasonable value for nominal system input voltage, DVINSET. DVINSET should be within 10% of actual system input voltage DVIN.
Automatic profiling calculates available torque based on available current. Current, in turn, depends on current mode CMODE. Automatic profiling uses the current settings CRACC, CRRUN, and CRDEC, as appropriate, depending on current mode CMODE.
Automatic profiling will usually generate a motion profile which reaches the same running speed as linear profiling. Under some cases (high friction or gravity loading), automatic profiling may use a lower running speed, to avoid excessive torque requirements.

**See Also** CRACC, CRRUN, LDCHK, LF, LG, LI, TU, DVINSET

**Example**

| Command | Description |
|---|---|
| `>RMODE 1`<br>` RMODE=1 [Auto]` | #Set auto profiler mode |
| `>TU 75`<br>` TU=75` | #Set torque utilization: use 75% of available torque |
| `>LI 2000`<br>` LI=2000` | #Set load inertia to 2000 g·cm$^2$ |
| `>LF 5`<br>` LF=5` | #Set load friction to 5 N·cm |
| `>LG 10`<br>` LG=10` | #Set load gravity to 10 N·cm |
| `>VR 200`<br>` VR=200 mm/sec` | #Set velocity to 200 mm/second (User Units are mm) |
| `>DIS 125`<br>` DIS=125 mm` | #Set distance to 125 mm |
| `>MI`<br>`>` | #Start incremental motion: profile automatically generated. |

## RUN : Run Sequence — Sequence Commands

**Execution Mode** Immediate (MODE = 0 Only)

**Syntax** RUN target

**Range** 'target' must be the name or number of an existing sequence.

**Description**
RUN starts execution of a sequence.
Sequences can also be started with the dedicated START input.
Sequences cannot be started if the system has an active alarm condition.
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.
RUN cannot be used inside sequences. To execute one sequence from within another, use the CALL command.

**Important Interactions** Sequences cannot be edited while a sequence is executing. The system prevents the editor from starting.

**See Also** EDIT, DIR, ABORT, <ESC>

**Example**

| Command | Description |
|---|---|
| `>LIST MAIN` | #List sequence MAIN |
| `(  1) VR 10`<br>`(  2) LOOP 10`<br>`(  3)    MI`<br>`(  4)    CALL WATCHER`<br>`(  5) ENDL`<br>`(  6) MA 0`<br>`(  7) CALL WATCHER` | #Sequence listing. |
| `>RUN MAIN` | #Run sequence MAIN |
| `>SIGRUN` | #Commands can still be executed, while… |
| ` SIGRUN=1` | #…sequences execute (SIGRUN=1). |
| `>` | |

## RUNLV : RUN Output Level — I/O

**Execution Mode** Immediate (MODE = 0 Only)

**Syntax** RUNLV n

**Range** n = 0: Normally Open
1: Normally Closed

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**Description** RUNLV is the active level of the Sequence Running (RUN) output, if used.

**See Also** OUTRUN, SIGRUN

**Example**

| Command | Description |
|---|---|
| >OUTRUN 3 | #Assign the RUN output to output #3 |
| OUTRUN 0(3) | |
| >RUNLV=1 | #Set the RUN output logic level to Normally Closed |
| RUNLV=0(1) | |
| >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. | |

```
------------------------------------------
    AS-One (ASX66)
      Integrated Motor
    Software Version: *.**
       Copyright 2004
   ORIENTAL MOTOR CO., LTD.
------------------------------------------
```

| Command | Description |
|---|---|
| >RUNLV | #Confirm the new value of RUNLV |
| RUNLV=1(1) | |
| > | |

## S_xxx : User-Defined String Variables — User Variables

**Execution Mode** Immediate (MODE = 0 Only) and Sequence

**Syntax** S_xxx text

**Range** S_xxx can be the name of any existing user-defined string variable
text = 20 characters maximum

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**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 extra carriage return, linefeed or reprompting.

**See Also** CLEARVAR, CREATEVAR, DELETEVAR, LISTVAR, SAS, SACS, VIEW

**Example**

| Command | Description |
|---|---|
| `>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 motion. |
| `>Fourth Quadrant` | |
| `>First Quadrant` | |
| `>First Quadrant` | |
| `>Second Quadrant` | |
| `>Third Quadrant` | #…etc. |
| `>` | |

## SACS : Send ASCII Control String — Sequence Commands

**Execution Mode** Sequence

**Syntax** SACS string

**Range** string : a series of ASCII characters or control codes, maximum 70 characters.

**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.
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.
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.)
There are several exceptions and extensions to the normal ASCII interpretation of control codes:
- ^@ (ASCII value NULL, binary 0) is not supported.
- ^, followed by a space, transmits one space character (this permits leading or trailing space characters in the output)
- ^^ transmits a single caret (^). ^^^ transmits an ASCII CTRL-^, 1Eh.

Other commands and comments cannot follow an SACS command on the same line: they will be considered part of the ASCII string.
For other control codes and their usual interpretations, see Appendix F.

**See Also** KB, KBQ, SAS, VIEW

**Example**

| Command | Description |
|---|---|
| `>LIST REFRESH` | #List Sequence "REFRESH" |
| `(  1) # VT-100 EMULATION` | #Comments, in sequence |
| `(  2) # 1) CLEAR DISPLAY` | #Transmitted control codes below cause |
| `(  3) # 2) HOME CURSOR` | VT−100 displays to clear screen and "home" |
| `(  4) # 3) TRANSMIT PREFERRED PROMPT` | the cursor. Then: transmit a custom prompt |
| `(  5) SACS ^[[2J^[[H-->` | |
| `>RUN REFRESH` | |
| `>` | |
| `-->` | |

**Note** 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.

## SAS : Send ASCII String — Sequence Commands

**Execution Mode** Sequence

**Syntax** SAS string

**Range** string : a series of ASCII characters, maximum 70 characters.

**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.
Other commands and comments cannot follow an SAS command on the same line: they will be considered part of the ASCII string.

**See Also** SACS, VIEW, KB, KBQ

**Example**

| 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
```

**Note** 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.

## SAVEALL : Save All Data — System Control

**Execution Mode** Immediate (MODE = 0 Only)

**Syntax** SAVEALL

**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.
The EEPROM has a nominal expected lifetime of 100,000 write cycles, which should be sufficient for almost all applications. The SAVEALL and SAVEPRM commands should not be used automatically (i.e. by a host controller) if they could possibly execute at high frequency. Saving once per day, for instance, yields a nominal expected lifetime of almost 275 years. Saving once per minute reduces the expected lifetime to about 70 days, and is certainly not recommended.
The system keeps a counter of how many times EEPROM has been written (by SAVExxx commands, CLEARxxx commands, or INITxxx commands). This counter is displayed each time any of these commands is executed, for reference.

**Caution** **The 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.**

**See Also** CLEARALL, RESET, SAVEPOS, SAVEPRM, INITPRM, CLEARPOS

**Example**

| Command | Description |
|---|---|
| `>SAVEALL` | #Save all parameters, variables, and position array data. |
| ` (EEPROM has been written 100 times)` | |
| ` Enter Y to proceed, other key to cancel. y` | #System requires confirmation |
| ` Saving Parameters........OK.` | |
| ` Saving POS[ ] Data.......OK.` | |
| `>` | #Note RESET required before some new settings take effect. |

## SAVEPOS : Save Position Array Data — System Control

**Execution Mode** Immediate (MODE = 0 Only)

**Syntax** SAVEPOS

**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.
The EEPROM has a nominal expected lifetime of 100,000 write cycles, which should be sufficient for almost all applications. The SAVEPOS command should not be used automatically (i.e. by a host controller) if it could possibly execute at high frequency. Saving once per day, for instance, yields a nominal expected lifetime of almost 275 years. Saving once per minute reduces the expected lifetime to about 70 days, and is certainly not recommended.
The system keeps a counter of how many times EEPROM has been written (by SAVExxx commands, CLEARxxx commands, or INITxxx commands). This counter is displayed each time any of these commands is executed, for reference.

**Caution**

**The 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.**

**See Also** CLEARALL, CLEARPOS, RESET, SAVEALL, TEACH, POS [x]

**Example**

Command

```
>TEACH
       *** Teach mode ***

 (V)    : Move Cont. Neg.    (M)    : Move Cont. Pos.
 (B)    : Move Incr. -0.001   (N)    : Move Incr. +0.001
 (Q)    : Current ON/OFF     (S)    : Save all data to EEPROM
 (K)    : Change Key Interval (50-500[msec])
 <Space> : Immediate Stop
 <Enter> : Data entry mode (Input POS number, then <Enter>)
 <ESC>   : Exit teach mode

 PC=       12.349    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.
>
```

Description

#Enter TEACH mode to "learn" target positions.

#<Enter>, this PC saved to POS[23]
#<ESC>: exit TEACH

#SAVEPOS, to be sure all POS[x] data restore after restart.

## SAVEPRM : Save Parameters — System Control

**Execution Mode** Immediate

**Syntax** SAVEPRM

**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 SAVEPRM 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.
The EEPROM has a nominal expected lifetime of 100,000 write cycles, which should be sufficient for almost all applications. The SAVEPRM command should not be used automatically (i.e. by a host controller) if it could possibly execute at high frequency. Saving once per day, for instance, yields a nominal expected lifetime of almost 275 years. Saving once per minute reduces the expected lifetime to about 70 days, and is certainly not recommended.
The system keeps a counter of how many times EEPROM has been written (by SAVExxx commands, CLEARxxx commands, or INITxxx commands). This counter is displayed each time any of these commands is executed, for reference.

**Caution**
**The SAVEPRM command writes to EEPROM. The EEPROM has a nominal expected lifetime of 100,000 write cycles. The SAVEALL command should not be used automatically (i.e. by a host controller) if it could possibly execute at high frequency.**

**See Also** CLEARALL, SAVEALL, SAVEPOS, RESET, INITPRM

**Example**

| Command | Description |
|---|---|
| >VR | #Check value of running velocity |
| VR=1 Rev/sec | #Default value, 1 RPS |
| >VR 10 | #Set running velocity 10 |
| VR=10 Rev/sec | |
| >RESET | #Reset the system |
| Resetting system. | |
| ------------------------------------------ | |
| AS-One (ASX66) | |
| Integrated Motor | |
| Software Version: *.** | |
| Copyright 2004 | |
| 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 startup value |
| (EEPROM has been written 14 times) | |
| Enter Y to proceed, other key to cancel. Y | #Confirm SAVEPRM |
| Saving Parameters........OK. | |
| >RESET | #Reset the system again |
| Resetting system. | |
| ------------------------------------------ | |
| AS-One (ASX66) | |
| Integrated Motor | |
| Software Version: *.** | |
| Copyright 2004 | |
| ORIENTAL MOTOR CO., LTD. | |
| ------------------------------------------ | |
| >VR | #Check value again |
| VR=10 Rev/sec | #OK. We have new startup value |
| > | |

## SCHGPOS : Distance After SENSOR Input — Motion Variables

**Execution Mode** Immediate (MODE = 0 Only) and Sequence

**Syntax** SCHGPOS n

**Range** n = 0 to MAXPOS (User Units)

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**Description** SCHGPOS is the distance used for SENSOR offset operation.
SENSOR offset operation allows the system to stop a continuous motion (MCN, MCP) a specified distance after a SENSOR input is detected. The system will change speed to SCHGVR, and eventually decelerate to a stop after the designated distance.
SENSORACT must be 2 (offset operation), and the system input signal SENSOR must be assigned to an input before offset operations can be used.

**See Also** INSENSOR, SCHGVR, SENSORACT, SENSORLV, MAXPOS

**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　　Motion Variables

**Execution Mode**　Immediate (MODE = 0 Only) and Sequence

**Syntax**　SCHGVR n

**Range**　n = 0.001 to MAXVEL (User Units/second)

**Initial Value**　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 default (initial) value.

**Access**　READ and WRITE

**Description**　SCHGVR is the velocity used for SENSOR offset operation.
SENSOR offset operation allows the system to stop a continuous motion (MCN, MCP) a specified distance after a SENSOR input is detected.　The system will change speed to SCHGVR, and eventually decelerate to a stop after the designated distance.
SENSORACT must be 2 (offset operation), and the system input signal SENSOR must be assigned to an input before offset operations can be used.

**See Also**　INSENSOR, SCHGPOS, SENSORACT, SENSORLV, MAXVEL

**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 — System Control

**Execution Mode** Immediate (MODE=0 Only)

**Syntax** SENSORACT n

**Range** n = 0: Hard Stop (stop as quickly as possible)
1: Soft Stop (controlled deceleration over time)
2: Soft stop at fixed distance from SENSOR signal

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**Description** SENSORACT establishes the stop action taken when the system detects a SENSOR input signal while executing a continuous motion (MCN, MCP).
If SENSORACT=0, the system will stop the motor as quickly as possible (hard stop). Stop action is exactly the same as HSTOP.
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.
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.
SENSOR input behavior is different during home seeking operations (MGHN, MGHP). SENSORACT does not affect the use of the SENSOR input during home seeking.

**See Also** INSENSOR, SCHGPOS, SCHGVR, SENSORLV

**Example**

| Command | Description |
|---|---|
| >INSENSOR 4 | #Assign SENSOR signal to Input 4 |
| INSENSOR=0(4) | |
| >SENSORLV 1 | #Set SENSOR active level to Normally Closed |
| SENSORLV=0(1) | |
| >SENSORACT 1 | #Set SENSORACT to 1: Soft Stop |
| SENSORACT=2(1) | |
| >SAVEPRM | #Save parameters |

```
>INSENSOR 4
 INSENSOR=0(4)
>SENSORLV 1
 SENSORLV=0(1)
>SENSORACT 1
 SENSORACT=2(1)
>SAVEPRM
 (EEPROM has been written 107 times)
 Enter Y to proceed, other key to cancel. y
 Saving Parameters........OK.
>RESET
Resetting system.
-------------------------------------------
    AS-One (ASX66)
      Integrated Motor
    Software Version: *.**
       Copyright 2004
   ORIENTAL MOTOR CO., LTD.
-------------------------------------------
>
```

#Reset to activate new settings

## SENSORLV : SENSOR Input Level — I/O

**Execution Mode** Immediate (MODE = 0 Only)

**Syntax** SENSORLV n

**Range** n = 0: Normally Open
1: Normally Closed

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**Description** SENSORLV is the active level of the SENSOR input, if used.

**See Also** INSENSOR, SIGSENSOR

**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. | #Reset to activate new settings |

```
------------------------------------------
    AS-One (ASX66)
      Integrated Motor
    Software Version: *.**
       Copyright 2004
   ORIENTAL MOTOR CO., LTD.
------------------------------------------
>
```

## SIGALARM : System ALARM Output Signal — System Status

**Execution Mode** Immediate and Sequence

**Syntax** SIGALARM

**Range** 0: No alarm
1: Active alarm condition

**Initial Value** 0

**Access** READ

**Description** SIGALARM is the system alarm signal.
SIGALARM is continuously updated by the system. If the system has an active alarm condition, SIGALARM will be one (1), otherwise it will be zero (0).
If MODE=0, SIGALARM may be assigned to any of the four system outputs, using OUTALARM. (In MODE 1−3 it is assigned to Output 2, and cannot be changed.) The active level of the output can be set with ALARMLV.
Although SIGALARM is available within sequences, it is not really meaningful there. If an alarm is detected, all sequence processing is aborted: sequences can never detect SIGALARM=1.

**See Also** OUTALARM, ALARMLV, ALMACT, OUTSG

**Example**

| Command | Description |
|---|---|
| `>ALMSET` | #Set the device in an alarm condition |
| `>SIGALARM` | #Query the status of the ALARM signal |
| `SIGALARM=1` | #Alarm condition present |
| `>ALMCLR` | #Clear the alarm condition |
| `>SIGALARM` | #Query the status of the ALARM signal again |
| `SIGALARM=0` | |
| `>` | |

## SIGALMCLR　: Functional ALMCLR Signal　　System Status

| | |
|---|---|
| **Execution Mode** | Immediate and Program |
| **Syntax** | SIGALMCLR |
| **Range** | 0: OFF<br>1: ON |
| **Initial Value** | 0 |
| **Access** | READ |
| **Description** | Display the status of the internal ALMCLR (Device ALARM CLEAR) signal. Allows the user to read the active level of the assigned input and the device condition. |
| **See Also** | ALARMLV, INSG, OUTALARM, OUTSG |
| **Interactions** | Modified by: INITIO, VERBOSE |
| **Example** | |

| Command | Description |
|---|---|
| `>SIGALMCLR` | #Query the functional status of the ALMCLR input |
| `SIGALARM=1` | #Device response when input is active |

## SIGCROFF　: System Current Off Input Signal — System Status

**Execution Mode** Immediate and Sequence

**Syntax** SIGCROFF

**Range** 0: CROFF input inactive
1: CROFF input active

**Initial Value** 0

**Access** READ

**Description** SIGCROFF is the system external Current Off (CROFF) input signal state.
SIGCROFF is continuously updated by the system, and reflects the state of the Current Off (CROFF) input, if used. If CROFF has not been assigned to an input, SIGCROFF is always zero (0).
If MODE=0, SIGCROFF may be assigned to any of the six system inputs, using INCROFF. (In MODE 1−3 it is assigned to Input 4, and cannot be changed.)　The active level of the input can be set with CROFFLV.
SIGCROFF does not reflect the actual state of motor current.　For instance, if current has been explicitly disabled (by CURRENT=0), but the CROFF input is unassigned or inactive, SIGCROFF is zero (0).

**See Also** INCROFF, CROFFLV, INSG, CURRENT

**Example**

| Command | Description |
| --- | --- |
| `LOOP` | #Start infinite loop |
| `  WHILE (SIGCROFF=1); WEND` | #While CROFF (current off) active, repeat this line |
| `  IF (PC!=0)` | #If Position command moved off 0… |
| `    MA 0` | #… Move back to 0 |
| `    MEND` | #… and wait for motion to complete |
| `  ENDIF` | #End IF block |
| `ENDL` | #end LOOP block |

## SIGEND : System END Output Signal — System Status

**Execution Mode** Immediate and Sequence

**Syntax** SIGEND

**Range** 0: END inactive.
1: END active.

**Initial Value** 0

**Access** READ

**Description** SIGEND is the system END signal, active while stopped, and inactive while moving. END will also become inactive if the motor is stopped, but outside +/−1.8 degrees of intended position (measured at the rotor shaft).
SIGEND is continuously updated by the system.
If MODE=0, SIGEND may be assigned to any of the four system outputs, using OUTEND. (In MODE 1−3 it is assigned to Output 1, and cannot be changed.) The active level of the output can be set with ENDLV.

**See Also** OUTEND, ENDLV, OUTSG

**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` | #Set Run velocity and Deceleration time… maybe aggressive? |
| `VR=20 Rev/sec` | |
| `TD=0.005` | |
| `>RUN SETTLETIME` | #Run sequence SETTLETIME |
| `>0.027` | #System took ~27 milliseconds to settle after motion profile finished |
| `>` | |

## SIGHOME　: System HOME Input Signal　　System Status

**Execution Mode**　Immediate (MODE = 0 Only) and Sequence

**Syntax**　SIGHOME

**Range**　0: HOME input inactive
1: HOME input active

**Initial Value**　0

**Access**　READ

**Description**　SIGHOME is the system external Home Position (HOME) input signal state.
SIGHOME is continuously updated by the system, and reflects the state of the HOME input, if used. If HOME has not been assigned to an input, SIGHOME is always zero (0).
SIGHOME can be active, even if the system is not in its home position (PC=0).　SIGHOME simply reflects the state of the HOME input.
If MODE=0, SIGHOME may be assigned to any of the six system inputs, using INHOME. (In MODE 1−3 SIGHOME is unavailable.)　The active level of the input can be set with HOMELV.

**See Also**　INHOME, HOMELV, INSG, HOMETYP, MGHP, MGHN, EHOME

**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 |
| `>` | |

## SIGHOMEP : System Home Position Output Signal — System Status

**Execution Mode** Immediate (MODE = 0 Only) and Sequence

**Syntax** SIGHOMEP

**Range** 0: Away from HOME position
1: At HOME position

**Initial Value** 0

**Access** READ

**Description** SIGHOMEP is the system HOMEP signal, active while on a valid HOME position, and inactive otherwise. SIGHOMEP is always 0, until the system has successfully executed a homing operation (EHOME, MGHN, MGHP). After a successful homing operation, SIGHOMEP=1 when position command PC=0, and SIGHOME=0 otherwise.
SIGHOMEP is continuously updated by the system.
If MODE=0, SIGHOMEP may be assigned to any of the four system outputs, using OUTHOMEP. (In MODE 1−3, SIGHOMEP is unavailable.) The active level of the output can be set with HOMEPLV.

**See Also** OUTHOMEP, HOMEPLV, OUTSG, MGHP, MGHN, EHOME, PC

**Example**

| Command | Description |
|---|---|
| `------------------------------------------` `AS-One (ASX66)` `Integrated Motor` `Software Version: *.**` `Copyright 2004` `ORIENTAL MOTOR CO., LTD.` `------------------------------------------` | #System Startup Banner |
| `>PC` | #Check position command PC |
| `PC = 0 Rev` | #Zero (typical for startup) |
| `>SIGHOMEP` | #Check SIGHOMEP |
| `SIGHOMEP=0` | #Zero (0): not at home. |
| `>EHOME` | #EHOME: goto 0 |
| `>SIGHOMEP` | #Check SIGHOMEP again |
| `SIGHOMEP=1` | #Now 1, because of EHOME |
| `>PC 22` | #Set PC to 22 |
| `PC=22 Rev` | |
| `>SIGHOMEP` | #Check SIGHOMEP |
| `SIGHOMEP=0` | #Not at home |
| `>MA 0` | #Absolute move, to PC=0 |
| `>SIGHOMEP` | #Check HOMEP |
| `SIGHOMEP=1` | #OK: this is the new home |
| `>` | |

## SIGLSN : System Limit Switch Negative Input Signal — System Status

**Execution Mode** Immediate and Sequence

**Syntax** SIGLSN

**Range** 0: Negative Limit Sensor inactive
1: Negative Limit Sensor active

**Initial Value** 0

**Access** READ

**Description** SIGLSN is the system external Negative Limit Sensor (LSN) input signal state.
SIGLSN is continuously updated by the system, and reflects the state of the LSN input, if used. If LSN has not been assigned to an input, SIGLSN is always zero (0).
If MODE=0, SIGLSN may be assigned to any of the six system inputs, using INLSN. (In MODE 1−3 it is assigned to Input 2, and cannot be changed.) The active level of the input can be set with OTLV.

**See Also** INLSN, OTLV, OTACT, INSG, HOMETYP, MGHP, MGHN

**Example**

| Command | Description |
|---|---|
| >LIST FIXLIMITS | #List sequence FIXLIMITS |
| ( 1) IF (SIGLSN=1) | #If SIGLSN=1, negative limit sensor active |
| ( 2) MCP | #Start moving continuously, positive direction |
| ( 3) WHILE (SIGLSN=1); WEND | #While the sensor is still active, wait |
| ( 4) ENDIF | #End IF block |
| ( 5) IF (SIGLSP=1) | #If SIGLSP=1, positive limit sensor active |
| ( 6) MCN | #Start moving continuously, negative direction |
| ( 7) WHILE (SIGLSP=1); WEND | #While the sensor is still active, wait |
| ( 8) ENDIF | #End IF block |
| ( 9) SSTOP | #Stop the motor (soft stop) |
| ( 10) MEND | #Wait for stop to finish |
| >ALM | #Check alarm |
| ALARM =66 , RECORD : 66 70 E0 06 00 00 00 00 00 00 | |
| ALM_OVER_POS_M , 8.109 [sec] past. | |
| WARNING =00 , RECORD : 00 00 00 00 00 00 00 00 00 00 | |
| No warning. | |
| >ALMCLR | #Limit sensor alarm: clear it. |
| >RUN FIXLIMITS | #Run sequence FIXLIMITS to get back within limits |
| >SIGLSN; SIGLSP | #Check limits |
| SIGLSN=0 | #Negative… |
| SIGLSP=0 | #…and positive limit sensors inactive. Recovered. |
| > | |

## SIGLSP : System Limit Switch Positive Input Signal — System Status

**Execution Mode** Immediate and Sequence

**Syntax** SIGLSP

**Range** 0: Positive Limit Sensor inactive
1: Positive Limit Sensor active

**Initial Value** 0

**Access** READ

**Description** SIGLSP is the system external Positive Limit Sensor (LSP) input signal state.
SIGLSP is continuously updated by the system, and reflects the state of the LSP input, if used. If LSP has not been assigned to an input, SIGLSP is always zero (0).
If MODE=0, SIGLSP may be assigned to any of the six system inputs, using INLSP. (In MODE 1−3 it is assigned to Input 1, and cannot be changed.) The active level of the input can be set with OTLV.

**See Also** INLSP, OTLV, OTACT, INSG, HOMETYP, MGHP, MGHN

**Example**

| Command | Description |
|---|---|
| `>LIST FIXLIMITS` | #List sequence FIXLIMITS |
| `(  1) IF (SIGLSN=1)` | #If SIGLSN=1, negative limit sensor active |
| `(  2)   MCP` | #Start moving continuously, positive direction |
| `(  3)   WHILE (SIGLSN=1); WEND` | #While the sensor is still active, wait |
| `(  4) ENDIF` | #End IF block |
| `(  5) IF (SIGLSP=1)` | #If SIGLSP=1, positive limit sensor active |
| `(  6)   MCN` | #Start moving continuously, negative direction |
| `(  7)   WHILE (SIGLSP=1); WEND` | #While the sensor is still active, wait |
| `(  8) ENDIF` | #End IF block |
| `(  9) SSTOP` | #Stop the motor (soft stop) |
| `( 10) MEND` | #Wait for stop to finish |
| `>ALM` | #Check alarm |

```
  ALARM =66 ,  RECORD : 66 70 E0 06 00 00 00 00 00 00

  ALM_OVER_POS_M , 8.109 [sec] past.

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

  No warning.
```

| Command | Description |
|---|---|
| `>ALMCLR` | #Limit sensor alarm 66: clear it. |
| `>RUN FIXLIMITS` | #Run sequence FIXLIMITS to get back within limits |
| `>SIGLSN; SIGLSP` | #Check limits |
| `SIGLSN=0` | #Negative… |
| `SIGLSP=0` | #…and positive limit sensors inactive. Recovered. |
| `>` | |

## SIGMBC : System Magnetic Brake Control Output Signal — System Status

**Execution Mode** Immediate and Sequence

**Syntax** SIGMBC

**Range** 0: Magnetic Brake Control OFF, Motor Current ON
1: Magnetic Brake Control ON, Motor Current OFF

**Initial Value** 0

**Access** READ

**Description** SIGMBC is the system Magnetic Brake Control (MBC) signal.
SIGMBC is zero (0) while motor current is enabled, and one (1) while motor current is disabled.
SIGMBC is continuously updated by the system.
If MODE=0, SIGMBC may be assigned to any of the four system outputs, using OUTMBC. (In MODE 1−3 it is assigned to Output 4, and cannot be changed.)
The active level of the output is fixed at Normally Closed, and cannot be changed. Electrically, the output state when SIGMBC=1 is the same as the output state when DC power is off.

**See Also** OUTMBC, OUTSG

**Example**

| Command | Description |
|---|---|
| `>CURRENT; SIGMBC` | #Check CURRENT and SIGMBC |
| `CURRENT=0` | #Current is disabled |
| `SIGMBC=1` | #Magnetic Brake Control on |
| `>CURRENT 1; SIGMBC` | #Enable Current, check SIGMBC |
| `CURRENT=1` | #Current is enabled |
| `SIGMBC=0` | #Magnetic Brake Control off |
| `>PSTOP; CURRENT; SIGMBC` | #Force an alarm condition, check again |
| `CURRENT=0` | #Current is off |
| `SIGMBC=1` | #Magnetic Brake Control on. |
| `>` | |

## SIGMOVE : System MOVE Output Signal — System Status

**Execution Mode** Immediate (MODE = 0 Only) and Sequence

**Syntax** SIGMOVE

**Range** 0: Motor is not executing a motion command
1: Motor is executing a motion command

**Initial Value** 0

**Access** READ

**Description** SIGMOVE is the system MOVE signal, active (1) while the system is executing any motion, and inactive (0) otherwise.
SIGMOVE is updated based on commanded position (PC). When the system is changing PC (intentionally moving), SIGMOVE=1, and SIGMOVE=0 otherwise. The motor may actually move, because of loading conditions, even if the system is not executing a motion. Similarly, the motor may still be physically moving after a commanded motion is complete. Use SIGEND to detect these conditions.
SIGMOVE is continuously updated by the system.
If MODE=0, SIGMOVE may be assigned to any of the four system outputs, using OUTMOVE. (In MODE 1−3, SIGMOVE is unavailable.) The active level of the output can be set with MOVELV.

**See Also** OUTMOVE, MOVELV, OUTSG, SIGEND

**Example**

| Command | Description |
|---|---|
| >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 |
| > | |

## SIGMSTOP : System Motor Stop Input Signal — System Status

**Execution Mode** Immediate (MODE = 0 Only) and Sequence

**Syntax** SIGMSTOP

**Range** 0: MSTOP input inactive
1: MSTOP input active

**Initial Value** 0

**Access** READ

**Description** SIGMSTOP is the system external Motor Stop (MSTOP) input signal state.
The MSTOP input stops motion when activated. Stop behavior is determined by MSTOPACT. MSTOP does not prevent motion: motions can be started while MSTOP is active.
SIGMSTOP is continuously updated by the system, and reflects the state of the MSTOP input, if used. If MSTOP has not been assigned to an input, SIGMSTOP is always zero (0).
If MODE=0, SIGMSTOP may be assigned to any of the six system inputs, using INMSTOP. (SIGMSTOP is not available in MODEs 1−3.) The active level of the input can be set with MSTOPLV.

**See Also** INMSTOP, MSTOPLV, MSTOPACT, INSG

**Example**

| Command | Description |
|---|---|
| `>LIST GO` | #List sequence GO |
| `(  1) WHILE (SIGMSTOP=1); WEND` | #Hold off, while SIGMSTOP is active |
| `(  2) MI` | #Start incremental motion |
| `(  3) MEND` | Wait for motion to end. |
| `>` | |

## SIGPAUSE : System PAUSE Input Signal — System Status

**Execution Mode** Immediate (MODE = 0 Only) and Sequence

**Syntax** SIGPAUSE

**Range** 0: PAUSE input not asserted
1: PAUSE input asserted

**Initial Value** 0

**Access** READ

**Description** SIGPAUSE reflects the state of the external Motion Pause (PAUSE) input.
SIGPAUSE will be active (1) when the PAUSE signal has been assigned to an input (INPAUSE != 0) and the PAUSE input is asserted. If PAUSE has not been assigned to an input, SIGPAUSE is always zero (0).
The PAUSE input is used to interrupt a motion and decelerate to a stop. The motion can be resumed later using the Continue command (CONT), if sequences are executing, asserting the START input. The Pause Clear (PAUSECL) input or PAUSECLR command can be used to abandon the remainder of a PAUSE'd motion.
If MODE=0, SIGPAUSE may be assigned to any of the six system inputs, using INPAUSE. (In MODE 1−3, SIGPAUSE is unavailable.) The active level of the input can be set with PAUSELV.

**See Also** INPAUSE, PAUSELV, INSG, PAUSE, PAUSECLR, CONT

**Example**

| Command | Description |
|---|---|
| `>LIST WATCHPAUSE` | #List sequence WATCHPAUSE |
| `(  1) MA X` | #Start motion, to position in variable 'X' |
| `(  2) WHILE (PC != X)` | #While commanded position still not 'X' |
| `(  3)   IF (SIGPAUSE=1)` | #If PAUSE input detected |
| `(  4)     WHILE (SIGPAUSE=1); WEND` | #Wait for PAUSE input to clear |
| `(  5)     CONT` | #Resume motion |
| `(  6)   ENDIF` | #End of IF block |
| `(  7) WEND` | #End of WHILE block |
| `>` | |

## SIGPAUSECL : System Pause Clear Input Signal — System Control

**Execution Mode** Immediate (MODE = 0 Only) and Sequence

**Syntax** SIGPAUSECL

**Range** 0: PAUSECL input not asserted
1: PAUSECL input asserted

**Initial Value** 0

**Access** READ

**Description** SIGPAUSECL reflects the state of the external Pause Clear (PAUSECL) input.
SIGPAUSECL will be active (1) when the PAUSECL signal has been assigned to an input (INPAUSECL != 0) and the PAUSECL input is asserted. If PAUSECL has not been assigned to an input, SIGPAUSECL is always zero (0).
The Pause Clear (PAUSECL) input causes a previously paused motion to be abandoned.
If MODE=0, SIGPAUSECL may be assigned to any of the six system inputs, using INPAUSECL. (In MODE 1−3, SIGPAUSECL is unavailable.) The active level of the input can be set with PAUSECLLV.

**See Also** INPAUSECL, PAUSECLLV, INSG, PAUSE, PAUSECLR, CONT

**Example**

| Command | Description |
|---|---|
| `>LIST DIRSWITCH` | #List sequence DIRSWITCH |
| `(  1) IF (SIGPAUSECL=1)` | #Re-use PAUSECL as direction select, if I/O budget tight |
| `(  2)    DIS=D` | #If asserted, distance = +D |
| `(  3) ELSE` | |
| `(  4)    DIS=-D` | #Otherwise, distance = −D |
| `(  5) ENDIF` | #End of IF block |
| `(  6) MI` | #Start incremental motion |
| `(  7) MEND` | #Wait for motion end |
| `>D 1` | |
| ` D=1` | |
| `>RUN DIRSWITCH` | |

## SIGPSTOP : System Panic Stop Input Signal — System Status

**Execution Mode** Immediate and Sequence

**Syntax** SIGPSTOP

**Range**
0: PSTOP input inactive
1: PSTOP input active

**Initial Value** 0

**Access** READ

**Description**
SIGPSTOP is the system external Panic Stop (PSTOP) input signal state.
The PSTOP input stops motion as quickly as possible when activated (hard stop), and then takes action defined by ALMACT. While PSTOP is active, motions cannot be started.
SIGPSTOP is continuously updated by the system, and reflects the state of the PSTOP input, if used. If PSTOP has not been assigned to an input, SIGPSTOP is always zero (0).
If MODE=0, SIGPSTOP may be assigned to any of the six system inputs, using INPSTOP. (In MODE 1−3 it is assigned to Input 3, and cannot be changed.) The active level of the input can be set with PSTOPLV.

**See Also** INPSTOP, PSTOPLV, INSG, PSTOP, ALMACT

**Example**

| Command | Description |
|---|---|
| `(  1) IF (SIGPSTOP=1)` | #Check PSTOP before moving |
| `(  2)   SAS NoGo: STOP input active.` | #If active, transmit message… |
| `(  3)   RET` | #…and return (No move, avoid an alarm.) |
| `(  4) ENDIF` | #End of IF block |
| `(  5) MI` | #Start incremental motion |
| `(  6) MEND` | #Wait for motion to end. |
| `>` | |

## SIGPSTS : System Pause Status Output Signal — System Status

**Execution Mode** Immediate (MODE = 0 Only) and Sequence

**Syntax** SIGPSTS

**Range** 0: Motion not paused.
1: Motion paused.

**Initial Value** 0

**Access** READ

**Description** SIGPSTS is the system Pause Status (PSTS) signal, active when a motion has been paused, and inactive otherwise.
SIGPSTS is continuously updated by the system. When motion is paused (by PAUSE input or PAUSE command), PSTS becomes active. It remains active until motion is continued (by CONT command, or START input if sequences are executing), or cleared (by PAUSECL input or PAUSECLR command), or another motion is started.
If MODE=0, SIGPSTS may be assigned to any of the four system outputs, using OUTPSTS. (In MODE 1−3, SIGPSTS is unavailable.) The active level of the output can be set with PSTSLV.

**See Also** OUTPSTS, PSTSLV, OUTSG, PAUSE, PAUSECLR,CONT

**Example**

| Command | Description |
|---|---|
| `>LIST PULSEOUT` | #List sequence PULSEOUT |
| `(  1) MA 0` | #Start absolute move to position 0 |
| `(  2) WHILE (PC != 0)` | #While we aren't there yet |
| `(  3)   IF (SIGPSTS=1)` | #If we are "paused" (by PAUSE input) |
| `(  4)     A=TIMER % 0.250` | #Variable A = TIMER modulo 1: ramp from 0 to 0.249 |
| `(  5)     IF (A>=0.125)` | #Toggle OUT4 based on value of A |
| `(  6)       OUT4=1` | #A>0.125, OUT4=1 |
| `(  7)     ELSE` | |
| `(  8)       OUT4=0` | #A<0.125, OUT4=0 |
| `(  9)     ENDIF` | #End IF block. Results in 4 "blinks" per second |
| `( 10)   ELSE` | |
| `( 11)     OUT4=0` | #If not paused: OUT4=0 |
| `( 12)   ENDIF` | #End IF block |
| `( 13) WEND` | #End WHILE block |
| `>` | |

## SIGRUN : System RUN Output Signal — System Status

**Execution Mode** Immediate (MODE = 0 Only) and Sequence

**Syntax** SIGRUN

**Range** 0: Sequences not executing
1: Sequences executing

**Initial Value** 0

**Access** READ

**Description** SIGRUN is the system RUN signal, active (1) while the system is executing any sequence, and inactive (0) otherwise.
SIGRUN is continuously updated by the system.
SIGRUN can be polled from the serial port to check for sequence completion. Because SIGRUN is always one (1) when sequences are executing, it has no real utility in sequences.
If MODE=0, SIGRUN may be assigned to any of the four system outputs, using OUTRUN. (In MODE 1−3, SIGRUN is unavailable.) The active level of the output can be set with RUNLV.

**See Also** OUTRUN, RUNLV, OUTSG, RUN, ABORT

**Example**

| Command | Description |
|---|---|
| `>RUN GOHOME` | #Run sequence GOHOME |
| `>SIGRUN` | #Host system periodically polls SIGRUN to test for completion |
| `SIGRUN=1` | #Sequence is still running |
| `>SIGRUN` | |
| `SIGRUN=1` | #Sequence is still running |
| `>SIGRUN` | |
| `SIGRUN=1` | #Sequence is still running |
| `>SIGRUN` | |
| `SIGRUN=1` | #Sequence is still running |
| `>SIGRUN` | |
| `SIGRUN=1` | #Sequence is still running |
| `>SIGRUN` | |
| `SIGRUN=0` | #Sequence is finished |
| `>RUN CYCLE` | #Run sequence CYCLE |
| `>` | |

## SIGSENSOR : System SENSOR Input Signal — System Status

**Execution Mode** Immediate (MODE = 0 Only) and Sequence

**Syntax** SIGSENSOR

**Range** 0: SENSOR input inactive
1: SENSOR input active

**Initial Value** 0

**Access** READ

**Description** SIGSENSOR is the system external Sensor (SENSOR) input signal state.
SIGSENSOR is continuously updated by the system, and reflects the state of the SENSOR input, if used. If SENSOR has not been assigned to an input, SIGSENSOR is always zero (0).
If used, the SENSOR input can stop continuous motions MCN and MCP: the actual stopping behavior is determined by SENSORACT. The SENSOR input can also affect homing operations, depending on HOMETYP: refer to the HOMETYP entry for more detail.
If MODE=0, SIGSENSOR may be assigned to any of the six system inputs, using INSENSOR. (In MODE 1−3 SIGSENSOR is unavailable.) The active level of the input can be set with SENSORLV.

**See Also** INSENSOR, SENSORLV, SENSORACT, INSG

**Example**

| Command | Description |
|---|---|
| `>SENSORACT` | #Check Sensor Action (SENSORACT) |
| ` SENSORACT=0(0)` | #0: Hard stop when sensor detected. |
| `>LIST SIMPLEHOME` | #List sequence SIMPLEHOME |
| | |
| `(  1) VR 0.1` | #Running velocity to 0.1 |
| `(  2) MCP` | #Move continuous, positive direction |
| `(  3) MEND` | #Wait for motion to stop re: SENSOR |
| `(  4) IF (SIGSENSOR=1)` | #If SENSOR still active (we stopped at right location) |
| `(  5)   PC=0` | #Set position command PC to 0. This is "home" |
| `(  6) ELSE` | |
| `(  7)   ALMSET` | #Sensor no longer active, motion overshot. Force alarm. |
| `(  8) ENDIF` | #End of IF block |
| `>` | |

## SIGTEMP : System TEMP Output Signal — System Status

**Execution Mode** Immediate and Sequence

**Syntax** SIGTEMP

**Range** 0: No temperature warning
1: Temperature warning

**Initial Value** 0

**Access** READ

**Description** SIGTEMP is the system TEMP signal, active when drive or motor temperatures are above warning levels, and inactive otherwise.
SIGTEMP is continuously updated by the system. If drive electronics temperature DTMP exceeds drive temperature warning level DTMPWRN, or motor winding temperature MTMP exceeds motor temperature warning level MTMPWRN, SIGTEMP will be one (1). Otherwise, SIGTEMP will be zero (0).
If MODE=0, SIGTEMP may be assigned to any of the four system outputs, using OUTTEMP. (In MODE 1−3 it is assigned to Output 3, and cannot be changed.) The active level of the output can be set with TEMPLV.

**See Also** OUTTEMP, OUTSG, TEMPLV, MTMP, MTMPWRN, DTMP, DTMPWRN

**Example**

| Command | Description |
|---|---|
| `>LIST MAINACTION` | #List sequence MAINACTION |
| `(  1) LOOP 10` | #Repeat 10 times |
| `(  2)   MI` | #Start incremental motion |
| `(  3)   MEND` | #Wait for motion to end |
| `(  4)   WHILE (IN6=1); WEND` | #Wait for IN6 to become 0 |
| `(  5)   WHILE (IN6=0); WEND` | #Wait for IN6 to become 1 again |
| `(  6) ENDL` | #End LOOP block |
| `(  7) MA 0` | #Start absolute motion to position 0 |
| `(  8) MEND` | #Wait for motion stop |
| `(  9) IF (SIGTEMP=1)` | #If SIGTEMP=1, DTMP or MTMP getting high |
| `( 10)   SACS ^M^JHigh Temp.^G` | #Transmit control string, then 'beep' |
| `( 11) ENDIF` | |
| `>` | |

## SLACT　: Software Position Limit Control　　　System Control

**Execution Mode** Immediate (MODE = 0 Only) and Sequence

**Syntax** SLACT n

**Range** n = 0: Software position limits are disabled
1: Software position limits are enabled after homing

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE
Read only in sequences

**Description** SLACT enables or disables software position limit action.
When SLACT=1, software position limits LIMN and LIMP are enforced, provided the system has completed a homing action (EHOME, MGHP, MGHN).
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.
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 LIMN and LIMP.)
For absolute or incremental index moves (MA, MI), limit checking is performed before motion starts.
If the final target position is outside the range, the motion will not occur, and the action defined by ALMACT will trigger.
For continuous motions (MCN, MCP), any out of range condition is detected only as it happens.
If the system is outside the software position limits, motions may still be started.　After any alarm is cleared, MI or MA can be executed if their destination would bring the motor within limits.　MCN or MCP can be executed, if the motor would move in the direction of the operational range.

**See Also** LIMP, LIMN, PC, MGHP, MGHN, EHOME, ALM, ALMACT

**Note** If LIMN=LIMP=0, software position limit checking is disabled, even if SLACT=1.　LIMN and LIMP should be set to appropriate values before enabling software position limit checking.

**Example**

| Command | Description |
| --- | --- |
| `>LIMP 10`<br>`LIMP=0(10) Rev` | #Positive position limit: 10 rev |
| `>LIMN -10`<br>`LIMN=0(-10) Rev` | #Negative position limit: 10 rev |
| `>SLACT 1`<br>`SLACT=0(1)` | #Enable position limit checking |
| `>INHOME 1`<br>`INHOME=0(1)` | #Assign HOME input to input 1 |
| `>HOMETYP 8`<br>`HOMETYP=8` | #Select HOME type 8 |
| `>ALMMSG 2`<br>`ALMMSG=2 [Alarm+Warning]` | #Enable automatic transmission of alarm and warning messages |
| `>SAVEPRM`<br>`(EEPROM has been written 62 times)`<br>`Enter Y to proceed, other key to cancel. Y`<br>`Saving Parameters........OK.` | #Save new configuration information |
| `>RESET`<br>`Resetting system.` | #Reset the system to make new settings active |

```
------------------------------------------
    AS-One (ASX66)
      Integrated Motor
    Software Version: *.**
       Copyright 2004
   ORIENTAL MOTOR CO., LTD.
------------------------------------------
```

| Command | Description |
| --- | --- |
| `>MGHP` | #Find home, start in positive dir. |
| `>SIGHOMEP`<br>`SIGHOMEP=1` | #Check HOME input after operation completes: active. |
| `>MCP` | #Start continuous motion |
| `>Over travel: software position limit detected.` | #Alarm at position limit |
| `>PC`<br>`PC=10.001 Rev`<br>`>` | #Check position command<br>#System stopped, just past limit |

## SSTOP : Soft Stop — Motion Commands

**Execution Mode** Immediate (MODE = 0 Only) and Sequence

**Syntax** SSTOP

**Description** SSTOP stops the motor with a controlled deceleration.
If RMODE=0 (linear ramps), the motor decelerates to start velocity VS over deceleration time TD, and then stops completely.
If RMODE=1 (automatic profiling), the system determines the deceleration profile based on load settings and torque utilization.

**See Also** TD, HSTOP, MSTOP, MSTOPACT, PSTOP, ABORT, RMODE

**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 — System Control

**Execution Mode** Immediate (MODE = 0 Only)

**Syntax** STARTACT n

**Range** n = 0: START input starts sequence execution when asserted.
1: START input starts sequence execution when asserted, and aborts sequence execution and motion when cleared.

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**Description** STARTACT determines the action associated with the dedicated START input. (internal profiler mode, MODE=0 only). START can be configured to start sequences only (STARTACT=0), or to act as a toggle (STARTACT=1): starting sequences when set to its active level and aborting sequences (and motions) when set to its inactive level.

**See Also** <ESC>, ABORT, STARTLV

**Example**

| Command | Description |
|---|---|
| >STARTACT 1 | #Set the START input action to level detect |
| STARTACT=0(1) | |
| >SAVEPRM | #Save new settings |
| (EEPROM has been written 62 times) | |
| Enter Y to proceed, other key to cancel. Y | |
| Saving Parameters........OK. | |
| >RESET | #Reset to activate new settings |
| Resetting system. | |

```
-----------------------------------------
     AS-One (ASX66)
       Integrated Motor
     Software Version: *.**
        Copyright 2004
    ORIENTAL MOTOR CO., LTD.
-----------------------------------------
```

| Command | Description |
|---|---|
| >STARTACT | #Confirm new value. |
| STARTACT=1(1) | |
| > | |

## STARTLV : START Input Level **I/O**

**Execution Mode** Immediate (MODE = 0 Only)

**Syntax** STARTLV n

**Range** n = 0: Normally Open
1: Normally Closed

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**Description** Sets the active level of the dedicated START input.

**See Also** STARTACT

**Example**

| Command | Description |
|---|---|
| >STARTLV 1<br>STARTLV=0(1) | #Set the START input logic to the Normally Closed logic level |
| >SAVEPRM | #Save the parameter assignments |
| (EEPROM has been written 10 times)<br>Enter Y to proceed, other key to cancel. Y<br>Saving Parameters........OK. | #Device response |
| >RESET<br>Resetting system. | #Establish the saved parameter values |

```
------------------------------------------
    AS-One (ASX66)
      Integrated Motor
    Software Version: *.**
       Copyright 2004
   ORIENTAL MOTOR CO., LTD.
------------------------------------------
```

| Command | Description |
|---|---|
| >STARTLV<br>STARTLV=1(1)<br>> | #Confirm START input logic level |

## STRSW : Current State at System Start — System Control

**Execution Mode** Immediate

**Syntax** STRSW n

**Range** n = 0: Motor current off at system start
1: Motor current on at system start

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**Description** STRSW enables or disables motor current immediately after system start.
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 (current level determined by CRSTOP).

**See Also** CURRENT, CRRUN, CRSTOP

**Example**

| Command | Description |
|---|---|
| >STRSW 0<br> STRSW=1 (0) [Current ON at start up(Current OFF 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. | #Reset to activate new settings |

```
-------------------------------------------
    AS-One (ASX66)
      Integrated Motor
    Software Version: *.**
       Copyright 2004
   ORIENTAL MOTOR CO., LTD.
-------------------------------------------
```

| Command | Description |
|---|---|
| >CURRENT<br> CURRENT=0<br>> | #CURRENT=0 after restart |

## TA : Acceleration Time — Motion Variables

**Execution Mode** Immediate (MODE = 0 Only) and Sequence

**Syntax** TA n

**Range** n = 0.001 to 500.000 (seconds)

**Initial Value** 0.500

**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 default (initial) value.

**Access** READ and WRITE

**Description** TA is the time used to accelerate the motor (increase velocity away from zero), when linear ramps are used (RMODE=0).
TA affects the initial ramp time for:
- MA (Move Absolute)
- MI (Move Incremental)
- MCN and MCP (Move Continuously, negative and positive)
- MGHN and MGHP (Seek home, start negative and positive)
- 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)
- MCN and MCP (Move Continuously, negative and positive)
- MIx (Linked index)

If speeds are decreasing (toward zero), deceleration time TD determines ramp time.
When RMODE=1 (automatic profiling), TA is ignored: the system determines ramp times based on load estimates and torque utilization.

**See Also** CV, EHOME, MA, MCN, MCP, MGHN, MGHP, MI, MIx, RMODE, TD

**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 — Communications

**Execution Mode** Immediate

**Syntax** TALK id

**Range** id = *, 0 to 9, A to Z (not case sensitive)

**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.
No space is permitted between TALK and id.

**See Also** @, 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>TALKX` | #Talk to Device X |
| `x>MGHP` | #Device A go home |
| `x>` | |

## TD : Deceleration Time — Motion Variables

**Execution Mode** Immediate (MODE = 0 Only) and Sequence

**Syntax** TD n

**Range** n = 0.001 to 500.000 (seconds)

**Initial Value** 0.500

**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 default (initial) value.

**Access** READ and WRITE

**Description** TD is the time used to decelerate the motor (decrease velocity toward zero), when linear ramps are used (RMODE=0).
TD affects the final ramp time for:
- MA (Move Absolute)
- MI (Move Incremental)
- MCN and MCP (Move Continuously, negative and positive)
- MGHN and MGHP (Seek home, start negative and positive)
- EHOME (Return to PC=0)
- SSTOP (Soft Stop)
- MSTOP (Motor Stop, if MSTOPACT=1)
- ABORT (Abort sequences and motions)
- <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)
- MCN and MCP (Move Continuously, negative and positive)
- MIx (Linked index)

If speeds are increasing (away from zero), acceleration time TA determines ramp time.
When RMODE=1 (automatic profiling), TD is ignored: the system determines ramp times based on load estimates and torque utilization.

**See Also** CV, EHOME, MA, MCN, MCP, MGHN, MGHP, MI, MIx, RMODE, TA

**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　　　Monitor Commands

**Execution Mode**　Immediate (MODE = 0 Only)

**Syntax**　TEACH

**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. The motor can also be externally positioned after toggling current off.
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 Section 4.7.

**See Also**　POS [x]

**Example**

| Command | Description |
|---|---|
| >TEACH | #Start the TEACH process |

```
       *** Teach mode ***

 (V)     : Move Cont. Neg.     (M)     : Move Cont. Pos.
 (B)     : Move Incr. -0.001   (N)     : Move Incr. +0.001
 (Q)     : Current ON/OFF      (S)     : Save all data to EEPROM
 (K)     : Change Key Interval (50-500[msec])
 <Space> : Immediate Stop
 <Enter> : Data entry mode (Input POS number, then <Enter>)
 <ESC>    : Exit teach mode

 PC=       23.416
```

## TEMPLV : TEMP Output Level I/O

**Execution Mode** Immediate

**Syntax** TEMPLV n

**Range** n = 0: Normally Open
1: Normally Closed

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**Description** TEMPLV is the active level of the Temperature Warning (TEMP) output, if used.

**See Also** OUTTEMP, SIGTEMP

**Example**

| Command | Description |
|---|---|
| `>OUTTEMP 3`<br>` OUTTEMP 0(3)` | #Assign the TEMP output to output #3 |
| `>TEMPLV=1`<br>` TEMPLV=0(1)` | #Set the TEMP output logic level to Normally Closed |
| `>SAVEPRM`<br>` (EEPROM has been written 14 times)`<br>` Enter Y to proceed, other key to cancel. Y`<br>` Saving Parameters........OK.` | #Save the parameter assignments |
| `>RESET`<br>` Resetting system.` | #Establish the saved parameter values |

```
-------------------------------------------
     AS-One (ASX66)
       Integrated Motor
     Software Version: *.**
        Copyright 2004
    ORIENTAL MOTOR CO., LTD.
-------------------------------------------
```

| Command | Description |
|---|---|
| `>TEMPLV`<br>` TEMPLV=1(1)`<br>`>` | #Confirm the new value of TEMPLV |

## TF : Motor Torque — System Status

**Execution Mode** Immediate and Sequence

**Syntax** TF

**Range** n/a (N·m)

**Access** READ

**Description** TF is the approximate amount of torque generated by the motor, at the motor shaft.
The motor torque varies, depending on motor current and position error. Motor torque TF has the same sign as position error PE: positive torque tends to move the motor in a positive direction. Changing the definition of positive direction (with DIRINV) also changes the definition of positive torque direction.

**See Also** IA, PE, DIRINV

**Example**

| Command | Description |
|---|---|
| `>PE` | #Check position error |
| `PE=0.002 Rev` | |
| `>TF` | #Check motor torque |
| `TF=0.34` | #Positive, approximately 0.34 N·m |
| `>` | |

## TIMER : Running Timer — System Status

**Execution Mode** Immediate and Sequence

**Syntax** TIMER n

**Range** n = 0 to 500000.000 (seconds)

**Initial Value** 0

**Access** READ and WRITE

**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.

**See Also** ALM, WAIT

**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 |
| `>` | |

## TQFF : Torque Feedforward Control — System Control

**Execution Mode** Immediate (MODE = 0 Only) and Sequence

**Syntax** TQFF n

**Range** n = 0: Disabled
1: Enabled

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE
READ only while motion is in progress

**Description** TQFF enables or disables torque feedforward control.
When torque feedforward control is enabled, the system uses load estimates and profile requirements to anticipate torque requirements. The system then determines how much to advance or retard the phase current angle, based on current magnitude, to generate the required torque. If load estimates are reasonable, this may reduce position error and improve performance.
Torque feedforward should not be used until reasonable values have been entered for load inertia (LI), load friction (LF), load static friction (LSF), and gravity or other constant loading (LG).
Start velocity VS replicates some of the functionality of torque feedforward, attempting to get the system moving earlier by jumping to some low speed. If load estimates are known well, torque feedforward provide the same functionality, and VS can be set to zero (0).
See Section 4.5, "Enhanced Features", for more information on torque feedforward control.

**See Also** LF, LG, LI, LSF, VS

**Example**

| Command | Description |
|---|---|
| >LG | #Check estimate of gravity (or other constant) load |
| LG=20 | #LG has been set to 20 N·cm |
| >PE | #Check position error |
| PE=0.472 deg | #Position error is almost 0.5 degree |
| >TQFF 1 | #Enable torque feedforward control, try to compensate for load |
| TQFF=1 | |
| >PE | #Check position error again |
| PE=0.091 deg | #Position error has been reduced to less than 0.1 degree |
| > | |

## TRACE : Sequence Trace Control — Monitor Commands

**Execution Mode** Immediate (MODE=0 Only)

**Syntax** TRACE n

**Range** n = 0: Trace is disabled
1: Trace is enabled

**Initial Value** 0

**Access** READ and WRITE

**Description** TRACE enables or disables tracing of sequence statements.
When sequence tracing is enabled (TRACE=1), sequence statements are displayed as they are executed, one statement at time, surrounded by "curly braces" { and }.

**See Also** RUN , ABORT, LIST

**Note** Enabling sequence tracing alters sequence timing, because of the time required to transmit the trace information. Sequences execute slower when TRACE=1.

**Example**

| Command | Description |
|---|---|
| `>LIST TOGGLEATVR` | #List sequence TOGGLEATVR |
| `(  1) LOOP 3` | #List output… |
| `(  2)   MI` | |
| `(  3)   WHILE (VC!=VR); WEND` | |
| `(  4)   OUT4=1-OUT4` | |
| `(  5)   MEND` | |
| `(  6) ENDL` | |
| `>TRACE 1` | #Enable Tracing |
| `>RUN TOGGLEATVR` | #Run sequence TOGGLEATVR |
| `>{ LOOP 3 }` | #First executing statement, surrounded by { } |
| `{ MI }` | #Next statement |
| `{ WHILE (VC!=VR) }` | #Next statement: note NOT the entire line |
| `{ WEND }` | #End WHILE block… |
| `{ WHILE (VC!=VR) }` | #…back to WHILE statement |
| `{ OUT4=1-OUT4 }` | #WHILE test failed, proceed beyond WEND |
| `{ MEND }` | #Wait for motion end |
| `{ ENDL }` | #End LOOP block, back to top-of-loop |
| `{ MI }` | #Actual to-of-loop is first statement within loop |
| `{ WHILE (VC!=VR) }` | #Repeat… |
| `{ WEND }` | |
| `{ WHILE (VC!=VR) }` | |
| `{ OUT4=1-OUT4 }` | |
| `{ MEND }` | |
| `{ ENDL }` | |
| `{ MI }` | |
| `{ WHILE (VC!=VR) }` | |
| `{ WEND }` | |
| `{ WHILE (VC!=VR) }` | |
| `{ OUT4=1-OUT4 }` | |
| `{ MEND }` | |
| `{ ENDL }` | #Loop count exhausted, sequence is finished. |

## TU : Torque Utilization — Motion Variables

**Execution Mode** Immediate (MODE = 0 Only) and Sequence

**Syntax** TU n

**Range** n = 0 to 100 (Integer values only) (% of available torque)

**Initial Value** 50

**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 default (initial) value.

**Access** READ and WRITE

**Description** TU controls torque utilization for automatic current control (CMODE=2) and automatic profiling (RMODE=1).
These modes attempt to maintain a constant ratio between torque used and available torque.
Increasing TU reduces current (CMODE=2), and makes profiling more aggressive (RMODE=1).
Decreasing TU increases current (CMODE=2), and makes profiling less aggressive (RMODE=1).

**See Also** RMODE, CMODE

**Interactions** Modified: CRACC
Modified by: CRSTOP, CRRUN, VERBOSE

**Example**

| Command | Description |
|---|---|
| `>LG` | #Check load gravity estimate |
| `LG=10` | #10 N·cm |
| `>CMODE 2` | #Select auto-current |
| `CMODE=2 [Auto]` | |
| `>TU` | #Check torque utilization TU |
| `TU=50` | #50% of available torque |
| `>CRSTOP` | #Check stop current, automatically selected |
| `CRSTOP=25` | #Set to 25%. |
| `>TU 25` | #Change torque utilization to 25% available current |
| `TU=25` | |
| `>CRSTOP` | #Check stop current again |
| `CRSTOP=41` | #Stop current increased to 41%, load requires 25%*41%=~10% full-scale torque |
| `>` | |

## UNLOCK　: Unlock Sequence　　Sequence Management

**Execution Mode**　Immediate (MODE=0 Only)

**Syntax**　UNLOCK target

**Range**　target can be the name or number of any existing sequence

**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.

**See Also**　DIR, EDIT, LOCK

**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. |
| `>` | |

## UU : User Units — System Control

**Execution Mode** Immediate

**Syntax** UU UserUnitName

**Range** UserUnitName = ASCII Characters, 10 characters maximum
0 (Clear string)

**Initial Value** Rev

**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 default (initial) value.

**Access** READ and WRITE

**Description** UU defines the units displayed with position− and velocity− related parameters, when the system is responding verbosely (VERBOSE=1).　Position-related values are displayed in terms of user units; velocity-related values are displayed in terms of user units per second.
When VERBOSE=0, only values are displayed: the UU unit information is suppressed.
Changing UU has no affect on actual motion.
Setting UU to 0 (digit zero)

**See Also** DPR

**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 System Status

**Execution Mode** Immediate and Sequence

**Syntax** VC

**Range** n/a (User Units/second)

**Initial Value** 0.000

**Access** READ

**Description** VC is the instantaneous velocity command, or set-point.
This value is controlled by the system's motion profiler in internal profiler mode (MODE=0), and by the input pulse rate in pulse input modes (MODE=1−3). 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, and is tracked in Feedback Velocity (VF).

**See Also** VE, VF

**Example**

| Command | Description |
|---|---|
| `>VR` | #Check target running velocity VR |
| `VR=17.5 mm/sec` | #17.5 mm/second |
| `>TA` | #Check acceleration time TA |
| `TA=20` | #20 seconds |
| `>MCP` | #Start moving continuously, positive direction |
| `>VC` | #Check commanded velocity, while accelerating |
| `VC=0.892 mm/sec` | #Slowly increasing toward VR |
| `>VC` | |
| `VC=1.614 mm/sec` | |
| `>VC` | |
| `VC=2.293 mm/sec` | |
| `>VC` | |
| `VC=3.007 mm/sec` | |
| `>VC` | |
| `VC=3.721 mm/sec` | |
| `>` | |

## VE : Velocity Error — System Status

**Execution Mode** Immediate and Sequence

**Syntax** VE

**Range** n/a (User Units/second)

**Access** READ

**Description** VE is the velocity error, or the difference between commanded velocity (VC) and actual velocity (VF), in user units per second. VE = VC – VF.
VE is continuously updated by the system, and can be used to monitor the systems response to load conditions.

**See Also** VC, VF

**Example**

| Command | Description |
|---|---|
| `>VR 4` | #Set the running velocity to 4 mm/second |
| `VR=4 mm/sec` | |
| `>MCP` | #Move continuously in the positive direction |
| `>VE` | #Display the velocity error |
| `VE=0.406 mm/sec` | |
| `>VE /` | #Continuously display the velocity error |
| `0.203 mm/sec` | #Typical response |

## VER　: Display Firmware Version　　Monitor Commands

**Execution Mode**　Immediate

**Syntax**　VER

**Description**　VER displays the system's firmware version information.

**Example**

| Command | Description |
|---|---|
| `>VER` | #Display the firmware version |
| `*.** / Date  Sep.1.2004` | #Typical response |
| `>` | |

## VERBOSE : Command Response Control — Communications

**Execution Mode** Immediate

**Syntax** VERBOSE n

**Range** n = 0: Respond with data only
1: Respond with data and descriptive text

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**Description** VERBOSE controls the amount of information that the system transmits in response to commands.
When VERBOSE=1 (the default), extra information is transmitted to establish the context of the response. VERBOSE=1 is preferred for human communications.
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.
The examples below show the differences in several responses.

**See Also** ECHO

**Example**

| Command | Description |
|---|---|
| `>VERBOSE` | #Check VERBOSE setting |
| ` VERBOSE=1` | #VERBOSE=1: extra text |
| `>PC` | #Check position setpoint |
| ` 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 setpoint |
| `1.5` | #Only value returned |
| `>VR` | #Check running velocity |
| `1` | #Only value returned |
| `>ALMMSG` | #Check ALMMSG |
| `2` | #Only value returned |
| `>` | |

## VF : Motor Velocity — System Status

**Execution Mode** Immediate and Sequence

**Syntax** VF

**Range** n/a (User Units/second)

**Access** READ

**Description** VF is the actual motor velocity, in user units per second. The sign indicates the direction of travel. VF is continuously updated by the system.
VF can deviate from the commanded velocity VC, depending on load conditions. The difference between VC and VF is the velocity error VE (VE = VC − VF).

**See Also** VC, VE

**Example**

| Command | Description |
|---|---|
| `>VR 9.3` | #Set the running velocity to 9.3 mm/second. |
| `VR=9.3 mm/sec` | |
| `>MCP` | #Move continuously in the positive direction |
| `>VF` | #Display the motor velocity value |
| `VF=9.218 mm/sec` | |
| `>VF /` | #Continuously display the motor velocity |
| `9.388 mm/sec` | #Typical response |

## VIEW : View Parameter — Communications

**Execution Mode** Sequence

**Syntax** VIEW element

**Range** 'element' can be the name of any parameter or variable available in sequences.

**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.

**See Also** KB, KBQ, SACS, SAS, VERBOSE

**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 |

**Note** 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.

## VR : Running Velocity — Motion Variables

**Execution Mode** Immediate (MODE = 0 Only) and Sequence

**Syntax** VR n

**Range** n = 0.001 to MAXVEL (User Units/second)

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**Description** VR is the running velocity for motions when the internal profiler is used (MODE=0). VR specifies the peak target speed for the motion, in user units per second.
VR is always positive: the direction for the motion is determined by start vs. end positions (for point to point motions), or by choice of positive or negative motion command (MCN vs. MCP, MGHN vs. MGHP).

**See Also** CV, EHOME, MA, MCN, MCP, MGHN, MGHP, MI, MAXVEL

**Important Interactions** The Change Velocity (CV) command overwrites VR with the value designated in the CV command.

**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=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 : Linked Motion Running Velocity — Motion Variables

**Execution Mode** Immediate (MODE = 0 Only) and Sequence

**Syntax** VRx n

**Range** x = 0 to 3: Linked motion segment
n = 0.001 to MAXVEL (User Units/second)

**Initial Value** 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 default (initial) value.

**Access** READ and WRITE

**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.
VRx is always positive: the direction for the motion segment is determined by the start and end positions for the entire linked index.

**See Also** INCABSx, MIx, LINKx, MAXVEL

**Example**

| Command | Description |
|---|---|
| `>VR1 5`<br>`VR1=5 in./sec` | #Set the velocity for linked motion segment #1 to 5 user units/second |
| `>DIS1 10`<br>`DIS1=10 in.` | #Set the distance for linked motion segment #1 to 10 user units |
| `>INCABS1 1`<br>`INCABS1=1 [INC]` | #Set the move type for linked motion segment #1 to incremental |
| `>LINK1 1`<br>`LINK1=1` | #Enable the linked between segments #1 and #2 |
| `>VR2 10`<br>`VR2=10 in./sec` | #Linked motion segment #2 velocity equals 10 user units/second |
| `>DIS2 20`<br>`DIS2=20 in.` | #Linked motion segment #2 distance equals 20 user units |
| `>INCABS2 0`<br>`INCABS2=0 [ABS]` | #Set the move type for linked motion segment #2 to absolute |
| `>LINK2 0`<br>`LINK2=0` | #Unlink segment #2 from segment #3 |
| `>MI1` | #Start the linked motion, with segment 1 |

## VS : Starting Velocity — Motion Variables

| | |
|---|---|
| **Execution Mode** | Immediate (MODE = 0 Only) and Sequence |
| **Syntax** | VS n |
| **Range** | n = 0.001 to MAXVEL (User Units/second) |
| **Initial Value** | 0.100 |
| **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 default (initial) value. |
| **Access** | READ and WRITE |
| **Description** | VS is the starting velocity for motions, when linear ramps are used (RMODE=0).<br>When RMODE=0, 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>Speed changes between zero (0) speed and VS is instantaneous. (Note that this is a velocity command, and not actual motor velocity: the motor cannot physically change speeds instantaneously). The sudden change in speed may or may not be desirable. In applications with high static friction, VS may help the system start or finish motions better. VS might also be used to avoid any very low resonant speed.<br>When RMODE=1 (automatic ramping), the system determines acceleration and deceleration profiles automatically, and VS is ignored.<br>VS is also used as the running velocity for MGHN and MGHP with HOMETYP=0−3, and used as the velocity for final HOME input detection with any of HOMETYP value. See "Mechanical Home Seeking" in Section 4.4 for more information on home operations. |
| **See Also** | EHOME, MA, MCN, MCP, MGHN, MGHP, MI, MIx, MAXVEL, RMODE |
| **Important Interactions** | As the jerk filter time lag (FILT) increases, VS has less effect on actual performance. The sudden transitions between VS and zero speed are smoothed by the filter. |

**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 — Sequence Commands

**Execution Mode** Sequence

**Syntax** WAIT n

**Range** n = 0.001 to 500000.000 (seconds)

**Description** WAIT causes sequence execution to wait for the indicated time, before proceeding to the next statement.

**See Also** KB, KBQ, TIMER, MEND

**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` | #Wait for motion to finish |
| `(  9)   MEND` | #Turn output 4 on |
| `( 10)   OUT4 1` | #Wait before proceeding, wait time in variable Q |
| `( 11)   WAIT Q` | #Turn output 4 off |
| `( 12)   OUT4 0` | #End of LOOP block |
| `( 13) ENDL` | |
| `>Q 0.5` | |
| ` Q=0.5` | |
| `>RUN TENTIMES` | |

## WEND : End of WHILE Block — Sequence Commands

**Execution Mode** Sequence

**Syntax** WEND

**Description** WEND terminates the innermost WHILE block in a sequence.
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.

**See Also** ENDIF, ENDL, IF, LOOP, WHILE, BREAKW

**Example**

| Command | Description |
|---|---|
| `>LIST CHKJAM` | #List sequence CHKJAM |
| `(  1) DIS=10; VR=10` | #Set motion parameter |
| `(  2) LOOP` | #Start infinite loop |
| `(  3)   MI` | #Start move incremental |
| `(  4)   WHILE (TF<0.5)` | #Check if over loaded |
| `(  5)     IF (SIGMOVE=0)` | #Check for motion end |
| `(  6)       BREAKW` | #Exit while loop, if so |
| `(  7)     ENDIF` | #End of IF block |
| `(  8)   WEND` | #End of WHILE block |
| `(  9)   IF (SIGMOVE!=0)` | #Check if moving |
| `( 10)     PAUSE` | #TF>0.5: PAUSE motion |
| `( 11)     WAIT TD` | #Wait for stop, send text, get response |
| `( 12)     SAS System in trouble.` | |
| `( 13)     SACS Enter 1 to continue, 0 to stop:` | |
| `( 14)     A=KBQ; SACS ^M^J>` | |
| `( 15)     IF (A=1)` | |
| `( 16)       CONT; MEND` | #CONTinue, if A=1 |
| `( 17)     ELSE` | #Otherwise, report stopped |
| `( 18)       SAS Operation stopped.` | |
| `( 19)       RET` | #Return from sequence |
| `( 20)     ENDIF` | |
| `( 21)   ENDIF` | |
| `( 22)   SAS Motion end, goto next.` | #Send normal message |
| `( 23)   WAIT 1` | |
| `( 24) ENDL` | #Dwell 1 sec., loop back to top. |
| `>` | |

## WHILE : Begin WHILE Block: execute while true — Sequence Commands

**Execution Mode**
Sequence

**Syntax**
WHILE (element1 {Conditional Operator} element2)

**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.
Parenthesis 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, forming a WHILE "block". BREAKW statements may appear within the WHILE block, terminating iteration and breaking out of the block.
When executed, the conditional expression is evaluated. If it evaluates to TRUE, sequence processing proceeds to the statement following the WHILE. If it evaluates to FALSE, sequence processing proceeds to the statement following the closing WEND statement.
The conditional expression is evaluated at the beginning of the block only, once per iteration. If the expression evaluates to TRUE when the WHILE statement executes, the contents of the WHILE block will be executed. The expression will be reevaluated at the next iteration: it is not tested during execution of the enclosed block statements.
Block structures (IF−ENDIF, WHILE−WEND, LOOP−ENDL) may be nested, to eight (8) levels deep.

**See Also**
IF, LOOP, WEND, BREAKW

**Example**

| Command | Description |
|---|---|
| `>LIST RUNTIMEOUT` | #List sequence RUNTIMEOUT |
| | |
| `(  1) IF (IN6=1)` | #If Input 6 is asserted… |
| `(  2)    MCN` | #Start moving continuously, negative direction |
| `(  3) ELSE` | #Otherwise… |
| `(  4)    MCP` | #Start moving continuously, positive direction |
| `(  5) ENDIF` | #End of IF block |
| `(  6) TIMER=0` | #Reset running TIMER to 0 |
| `(  7) WHILE (IN5=1)` | #Begin WHILE block: execute while Input 5 is asserted |
| `(  8)    IF (SIGTEMP=1)` | #If temperature warning |
| `(  9)       ALMSET` | #…set alarm (will automatically abort sequence and motion) |
| `( 10)    ENDIF` | #End of IF block |
| `( 11)    IF (TIMER > 5.0)` | #If TIMER greater than 5 seconds… |
| `( 12)       BREAKW` | #BREAK out of WHILE block: next statement follows WEND |
| `( 13)    ENDIF` | #End of IF block |
| `( 14) WEND` | #End of WHILE block: back to WHILE and reevaluate |
| `( 15) SSTOP` | #Start Soft Stop |
| `( 16) MEND` | #Wait for motion to end |
| `>` | |

# Chapter 7 Troubleshooting

This chapter explains the system's protective functions and procedures for troubleshooting alarm conditions.

## 7.1 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 many cause the system to disable motor current and lose holding torque. The system should be used in a way that prevents personal injury or damage to equipment if an alarm condition occurs.
- When an alarm occurs, determine and correct the cause of the alarm before attempting to restore normal operation. Some alarms can be cleared with the ALMCLR command; others require resetting the system or cycling input power. (A few alarms indicate serious system malfunction, and cannot be cleared.) The cause of the alarm should always be corrected before attempting to clear the system alarm state.

### ■ Types of Protective Functions and Check Methods

⚠ Warning

*The device has protective functions to protect itself from rising ambient temperatures, poor connections, abnormal input power and other similar conditions.*
*When a protective function is triggered, the ALARM LED on the back side of 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.*

Types of protective functions

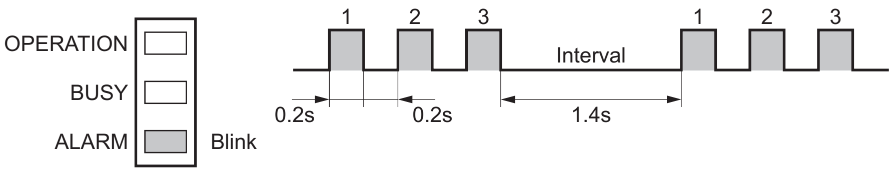
<table>
<tr><th>Protective Function</th><th>Description</th><th>Alarm Code</th><th>ALARM LED Blinks</th><th>System Action</th><th>ALMCLR Effect</th></tr>
<tr><td>No alarm</td><td>No alarm</td><td>0x00</td><td>OFF</td><td>Normal Operation</td><td>−</td></tr>
<tr><td>Stack overflow</td><td>Sequence memory "stack" exhausted</td><td>0x90</td><td rowspan="10">1</td><td rowspan="10">Motion and sequence processing stop.</td><td rowspan="10">Clears alarm</td></tr>
<tr><td>Sequence reference error</td><td>Attempt to call a non-existing sequence as a subroutine</td><td>0x94</td></tr>
<tr><td>Calculation over flow</td><td>Sequence calculation result exceeded numerical limits</td><td>0x98</td></tr>
<tr><td>Parameter range error</td><td>Attempt to set a parameter to a value outside its range</td><td>0x99</td></tr>
<tr><td>Zero division</td><td>Attempt to divide by zero</td><td>0x9A</td></tr>
<tr><td>PC command execution error</td><td>Attempt to modify position counter PC while a motion was in process</td><td>0x9D</td></tr>
<tr><td>User variable reference error</td><td>Attempt to access a non-existing user-defined variable</td><td>0x9E</td></tr>
<tr><td>Parameter write error</td><td>Attempt to change a parameter under invalid conditions (e.g. if prohibited while moving)</td><td>0x9F</td></tr>
<tr><td>Motion while in motion</td><td>Attempt to execute a motion while an incompatible motion is in progress</td><td>0xA0</td></tr>
<tr><td>User alarm</td><td>ALMSET command intentionally executed</td><td>0xE0</td></tr>
<tr><td>Driver overheat</td><td>Drive temperature exceeds programmed limit DTMPMAX</td><td>0x21</td><td rowspan="4">2</td><td rowspan="2">Motor current disabled (no holding torque)</td><td rowspan="4">Clears alarm if condition corrected</td></tr>
<tr><td>Motor overheat</td><td>Motor temperature exceeds programmed limit MTMPMAX</td><td>0x26</td></tr>
<tr><td>Over load</td><td>Maximum permitted torque applied, duration exceeds programmed limit OLTIME</td><td>0x30</td><td rowspan="2">Defined by ALMACT setting</td></tr>
<tr><td>Over velocity</td><td>Velocity exceeded programmed limit OVERVEL</td><td>0x31</td></tr>
</table>

| Protective Function | Description | Alarm Code | ALARM LED Blinks | System Action | ALMCLR Effect |
|---|---|---|---|---|---|
| Over voltage | DC input voltage out of specification (high) | 0x22 | 3 | Motor current disabled (no holding torque) | Clears alarm if condition corrected |
| Low voltage | DC input voltage out of specification (low) | 0x23 | | | |
| Over position error | Position error exceeds programmed limit OVERFLOW | 0x10 | 4 | Defined by ALMACT setting | Clears alarm if condition corrected |
| Over current | Excessive current detected in the motor windings | 0x20 | 5 | Motor current disabled (no holding torque) | No effect |
| Panic stop | System executed a panic stop because of a PSTOP input or command | 0x68 | 6 | Defined by ALMACT setting | Clears alarm |
| LS logic error | Positive and negative position limit signals on simultaneously | 0x60 | 7 | Motion and sequence processing stop | Clears alarm |
| LS connected in reverse | Positive or negative position limit signal detected opposite home seeking direction | 0x61 | | | |
| HOME operation failed | Unstable or unexpected position limit signal detected while seeking home position | 0x62 | | | |
| HOMELS not found | No HOME input detected between position limit signals while seeking home position | 0x63 | | | |
| TIM, SENSOR signal error | Timing position or SENSOR input expected with HOME input: not found | 0x64 | | | |
| Hardware over travel | Positive or negative position limit signal detected | 0x66 | | Defined by ALMACT setting | |
| Software over travel | Position outside of programmed positive and negative position limits | 0x67 | | | |
| LS detection during home offset motion | Positive or negative position limit signal detected while moving to OFFSET position after homing | 0x6A | | Motion and sequence processing stop | |
| Motion parameter error | Attempt to execute motion with incompatible motion parameters | 0x70 | | | |
| Sensor error during motion | Position feedback sensor error detected while motor is moving | 0x28 | 8 | Motor current disabled (no holding torque) | No effect |
| Sensor error | Position feedback sensor malfunction | 0x42 | | | |
| Motor movement during startup | Motor was moving (driven by external torque) while system was starting | 0x43 | | | |
| EEPROM error | User data in non-volatile EEPROM memory is corrupt | 0x41 | 9 | Motor current disabled (no holding torque) | No effect |
| System error | System detected unexpected internal logic state | 0xF0 | ON | Motor current disabled (no holding torque) | No effect |
| Memory error | Internal memory access error | 0xF1 | | | |
| Sequence internal error | Sequence code invalid or corrupt | 0xF2 | | | |

- How to check protective functions

The type of protective function that has been activated can be checked using the following two methods:

1) Count how many times the ALARM LED blinks on the back side of the device.
An example of the ALARM LED's blinking cycle is shown in the figure below.
Example: Overvoltage protection




2) Check the alarm code using the ALM command.

- Clearing alarm conditions

Before clearing alarm conditions, always correct the cause of the alarm.
To clear an alarm condition, perform one of the following:
- Enter an ALMCLR command, for alarm conditions that ALMCLR can clear (refer to table above).
- Enter a RESET command (see the RESET entry in Chapter 6 for details of a system reset).
- Turn off the power, wait for the green POWER LED to turn off, then turn power back on.

# 7.2 Inspection

Periodically inspect the device for the items listed below.
If the device appears or sounds abnormal, or operates poorly, discontinue use and contact your nearest Oriental Motor office.

## ■ During Inspection

- Are any of the device mounting screws loose?
- Check for any unusual noises in the device bearings (ball bearings) or other moving parts.
- Are the device output shaft and load shaft out of alignment?
- Is the power supply connector loose?

# 7.3 Troubleshooting and Corrective Actions

If device operation is not normal , check this section and take appropriate action. If operation is still not normal, contact your nearest Oriental Motor office.

**Memo** Perform failure diagnosis using the following methods:
- Check the alarm code using the ALM command.
- Count how many times the ALARM LED blinks.

| Phenomenon | Alarm Code | ALARM LED Blinks | Protective Function | Description | Action |
|---|---|---|---|---|---|
| Motion and sequence execution stop | 0x90 | 1 | Stack overflow | Sequence memory "stack" exhausted | Restructure sequences to reduce the number of nested blocks or subroutine calls |
| | 0x94 | | Sequence reference error | Attempt to call a non-existing sequence as a subroutine | Revise the CALL statement or rename the intended target sequence |
| | 0x98 | | Calculation overflow | Sequence calculation result exceeded numerical limits | Check math operations, make sure they cannot overflow |
| | 0x99 | | Parameter range error | Attempt to set a parameter to a value outside its range | Make sure all assignments stay within defined limits |
| | 0x9A | | Zero division | Attempt to divide by zero | Check division operations, test divisor for zero before division |
| | 0x9D | | PC 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 |
| | 0x9E | | User variable reference error | Attempt to access a non-existing user-defined variable | Make sure the target user-defined variable exists: use the correct name in sequence |
| | 0x9F | | Parameter write error | Attempt to change a parameter under invalid conditions (e.g. if prohibited while moving) | Make sure that:<br>- CMODE and RMODE are not changed while moving<br>- CRRUN and CRSTOP are not changed while CMODE=2<br>- CRACC is not changed unless CMODE=1 |
| | 0xA0 | | Motion while in motion | Attempt to execute a motion while an incompatible motion is in progress | Make sure motions are not started before a previous motion is complete. Use MEND, poll SIGMOVE, or monitor the MOVE output to detect motion complete. |
| | 0xE0 | | User alarm | ALMSET command intentionally executed | If a user alarm was not expected, check sequence programming for inappropriate ALMSET command(s) |

| Phenomenon | Alarm Code | ALARM LED Blinks | Protective Function | Description | Action |
|---|---|---|---|---|---|
| Motion and sequence execution stop | 0x60 | 7 | LS logic error | Positive and negative position limit signals on simultaneously | - 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.). |
| | 0x61 | | LS connected in reverse | Positive or negative position limit signal detected opposite home seeking direction | |
| | 0x62 | | HOME operation failed | Unstable or unexpected position limit signal detected while seeking home position | |
| | 0x63 | | HOME not found | No HOME input detected between position limit signals while seeking home position | Check HOME sensor wiring and connections |
| | 0x64 | | TIM, SENSOR signal error | Timing 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 Timing position while HOME input active. Make sure HOME and other required input(s) can be active at the same location. |
| | 0x6A | | LS detected during home offset motion | Positive or negative position limit signal detected while moving to OFFSET position after homing | Make sure that the OFFSET distance, measured from the HOME signal position, does not trigger a limit sensor |
| | 0x70 | | Motion parameter error | Attempt to execute motion with incompatible motion parameters | - 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. |

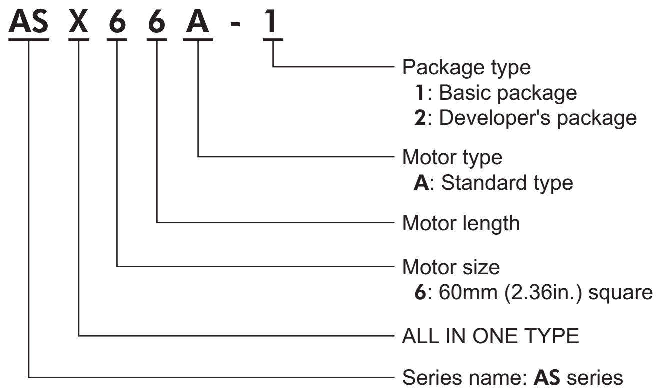
| Phenomenon | Alarm Code | ALARM LED Blinks | Protective Function | Description | Action |
|---|---|---|---|---|---|
| Motion and sequence execution stop.<br><br>Motor may or may not have holding torque, depending on ALMACT. | 0x30 | 2 | Over load | Maximum permitted torque applied, duration exceeds programmed limit OLTIME | - Reduce load.<br>- Increase current.<br>- Reduce running velocity.<br>- Increase acceleration or deceleration times.<br>- Reconsider the value of OLTIME |
| | 0x31 | | Over velocity | Velocity exceeded programmed limit OVERVEL | - Reduce running velocity<br>- Reconsider the value of OVERVEL |
| | 0x10 | 4 | Over position error | Position error exceeds programmed limit OVERFLOW | - Reduce load.<br>- Increase current.<br>- Reduce running velocity.<br>- Increase acceleration or deceleration times.<br>- Reconsider the value of OVERFLOW |
| | 0x68 | 6 | 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). |
| | 0x66 | 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.). |
| | 0x67 | | 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. |

| Phenomenon | Alarm Code | ALARM LED Blinks | Protective Function | Description | Action |
|---|---|---|---|---|---|
| The motor lacks holding torque. | 0x21 | 2 | Drive overheat | Drive temperature exceeds programmed limit DTMPMAX | - Reduce motion duty cycle.<br>- Reduce current.<br>- Increase ventilation.<br>- Reduce ambient temperature. |
| | 0x26 | | Motor overheat | Motor temperature exceeds programmed limit MTMPMAX | |
| | 0x22 | 3 | Overvoltage | DC input voltage out of specification (high) | Check power supply.<br>Can also occur while slowing a large inertial load (regenerative braking). Reduce load inertia or increase deceleration time. |
| | 0x23 | | Low voltage | DC input voltage out of specification (low) | Check power supply. |
| | 0x20 | 5 | Overcurrent | Excessive current detected in the motor windings | Contact Oriental Motor to arrange for inspection or repair. |
| | 0x28 | 8 | Sensor error during motion | Position feedback sensor error detected while motor is moving | Contact Oriental Motor to arrange for inspection or repair. |
| | 0x42 | | Sensor error | Position feedback sensor malfunction | |
| | 0x43 | | Rotor movement during startup | Motor was moving (driven by external torque) while system was starting | Make sure that the motor shaft is not moving when applying power or resetting the system |
| | 0x41 | 9 | EEPROM error | User data in non-volatile EEPROM memory is corrupt | Contact Oriental Motor to arrange for inspection or repair. |
| | 0xF0 | ON | System error | System detected unexpected internal logic state | |
| | 0xF1 | | Memory error | Internal memory access error | |
| | 0xF2 | | Sequence internal error | Sequence code invalid or corrupt | |

# Appendix A Model Number

Model-number format.

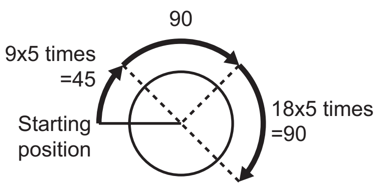
## A.1 How to Identify the Product Model

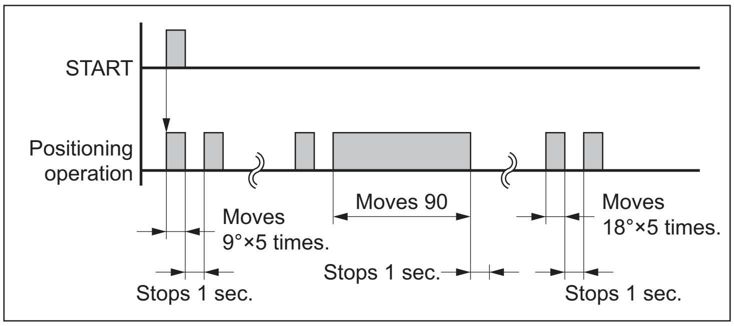


# Appendix B　Sample Programs

This chapter provides sample programs.

## B.1 Repeated Positioning Operation




### ■ Main Program

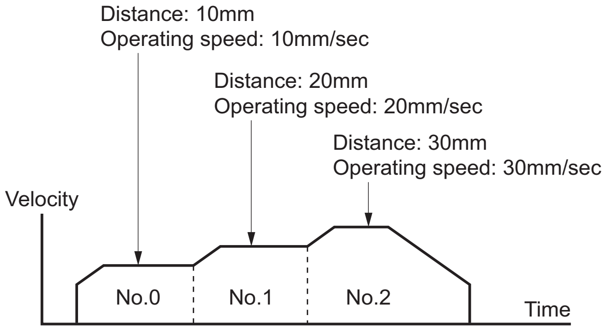
Applicable device:
Resolution: 360 deg/rev (DPR=360)
UU: Degrees

```
( 1) TA 0.1         The acceleration time is set to 0.1sec.
( 2) TD 0.1         The deceleration time is set to 0.1sec.
( 3) VS=10          The starting velocity is set to 10 deg/sec.
( 4) VR=360         The running velocity is set to 360 deg/sec.
( 5) LOOP 5         Lines 6 through 9 are repeated five times.
( 6)   DIS=9        The distance is set to 9 degrees.
( 7)   MI           Incremental positioning operation is executed.
( 8)   MEND         The program waits until the motion is ended.
( 9)   WAIT 1       The program waits 1 sec.
(10) ENDL           The LOOP statement is ended.
(11) DIS=90         The distance is set to 90 degrees.
(12) MI             Incremental positioning operating is executed.
(13) MEND           The program waits until the motion is ended.
(14) WAIT 1         The program waits 1 sec.
(15) LOOP5          Lines 16 through 19 are repeated five times.
(16)   DIS=18       The distance is set to 18 degrees.
(17)   MI           Incremental positioning operation is executed.
(18)   MEND         The program waits until the motion is ended.
(19)   WAIT 1       The program waits 1 sec.
(20) ENDL           The LOOP statement is ended.
(21) END            The program is ended.
```

## B.2 Executing Linked Operation

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

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




| | |
|---|---|
| `(1) DIS 0=10` | The distance for operation number 0 is set to 10 mm. |
| `(2) DIS 1=20` | The distance for operation number 1 is set to 20 mm. |
| `(3) DIS 2=30` | The distance for operation number 2 is set to 30 mm. |
| `(4) VR 0=10` | The operating speed for operation number 0 is set to 10 mm/sec. |
| `(5) VR 1=20` | The operating speed for operation number 1 is set to 20 mm/sec. |
| `(6) VR2=30` | The operating speed for operation number 2 is set to 30 mm/sec. |
| `(7) INCABS 0=1` | The positioning mode for operation number 0 is set to incremental. |
| `(8) INCABS 1=1` | The positioning mode for operation number 1 is set to incremental. |
| `(9) INCABS 2=1` | The positioning mode for operation number 2 is set to incremental. |
| `(10) LINK 0=1` | Operation number 0 is set to linked. |
| `(11) LINK 1=1` | Operation number 1 is set to linked. |
| `(12) LINK 2=0` | Operation number 2 is set to one shot linked. |
| `(13) MI 0` | Start the operation to start at operation number 0. (Numbers 0 through 2 are linked.) |
| `(14) END` | The program is ended. |

# Appendix C Daisy Chain Connection Procedure

This chapter describes the procedure used to connect two or more devices via a daisy chain (up to 35 devices).

## C.1 Setting the Unit ID's

Set the axis number for each device using the ID command (driver axis setting: 0 to 9, A to Z). When setting axis numbers, connect the axes to the RS-232C communication port (CN1) one by one before implementing daisy chain connections. Do not use duplicate axis numbers.

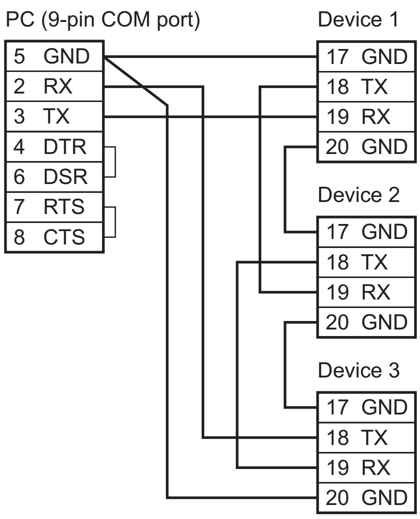
Example) Setting 1 as an ID

| | |
|---|---|
| `>ID=1` | The driver's device's axis number is set to 1. |
| `ID=1` | (Echo back) |
| `@1 SAVEPRM` | Talk to the device and save the parameter |
| `(EEPROM has written 104 times)` | |
| `Enter Y to proceed, other key to cancel. Y` | Input 'Y' |
| `Saving Parameters........OK.` | |
| `1>` | |

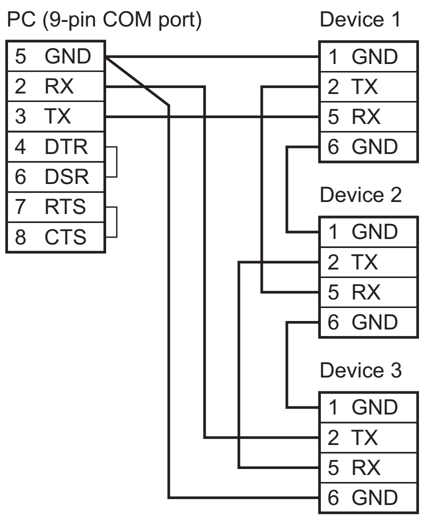
## C.2 Daisy Chain Connection Procedure

Use the RS-232C communication pins (TX, RX, GND) of the I/O connector or communication connector. Two examples of connecting three drivers via a daisy chain is shown below.

• Using the I/O connector




• Using the communication connector




**Note**

- *The maximum distance between drivers when using a daisy chain connection should be 15 m (49.2 feet).*
- *Wire the RS-232C signal lines over the shortest possible distance. It is recommended that the signal lines be shielded to protect them from noise interference.*
- *Be sure to short pins 4 (DTR) and 6 (DSR) on the PC together and pins 7 (RTS) and 8 (CTS) on the PC together.*

# C.3 Daisy Chain Communication Example

Call the specific device used for communication via the @command. When the power is turned on, the communication device is set to the one whose axis number is 0.

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

When the power is turned on, the communication device is set to the one whose axis number is 0.
As a result, a prompt (“>”) is not output.

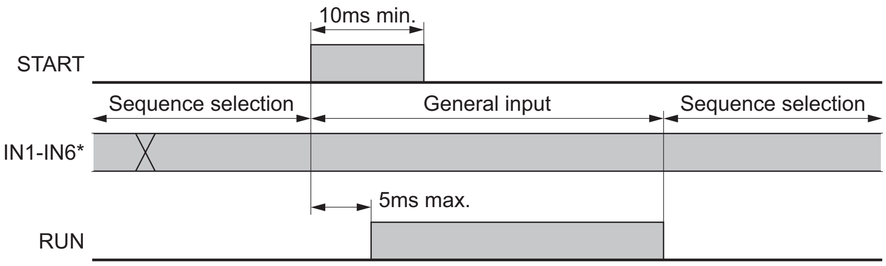
| | |
|---|---|
| @1 | : Executing a “@1” command connects device 1. |
| 1> | : A prompt (“1>”) is output. |
| @2 | : Connect to device 2. |
| 2> | : A prompt (“2>”) is output. |
| 2>ID | : Query the ID of the connected device. |
| ID2 | : The axis ID is returned (2). |

# Appendix D Timing Charts

This chapter includes timing charts that describe the operation of the αSTEP-One device.

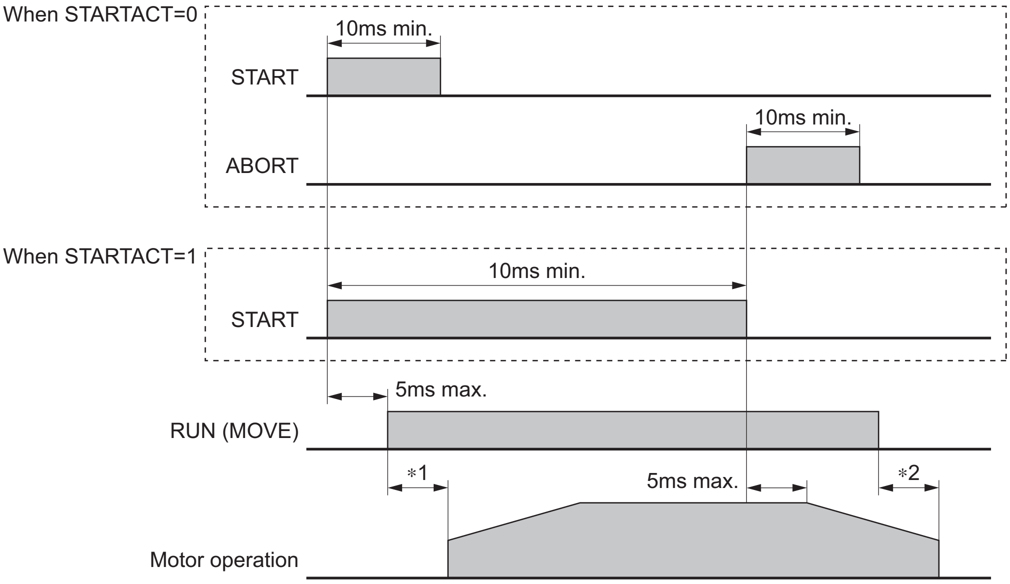
## D.1 Execution of a Sequence

### ■ Selection and Execution of a Sequence




* Only inputs that are not assigned are read.
Inputs assigned to another function are always read as “0”.

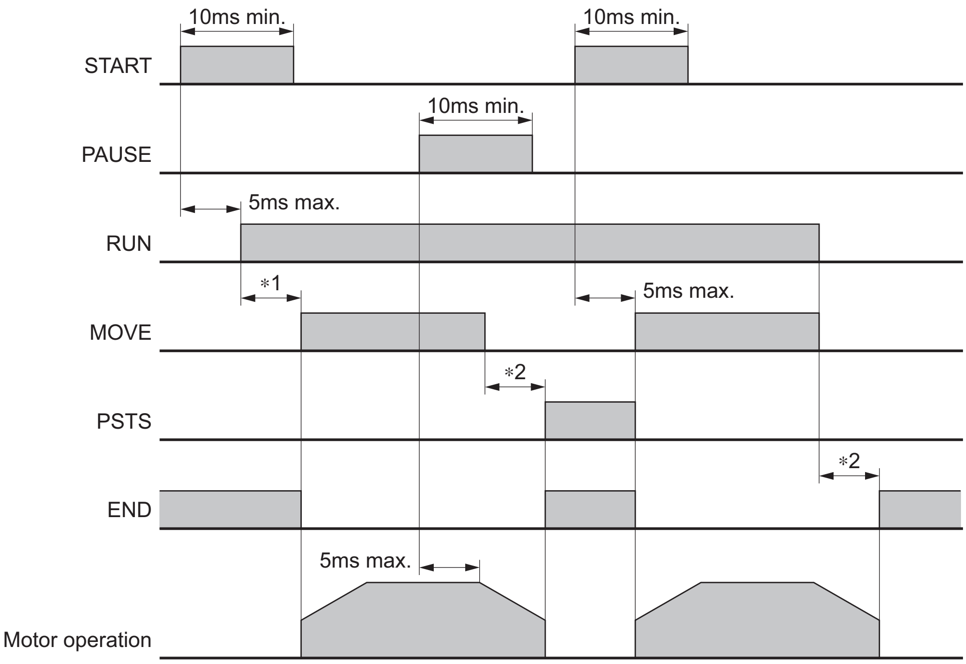
### ■ Execution and Stopping a Sequence (START, ABORT, RUN, MOVE)




*1 Depends on the sequence.
*2 Depends on the velocity filter, load condition and settling time at stop.

# D.2 Stopping Operation

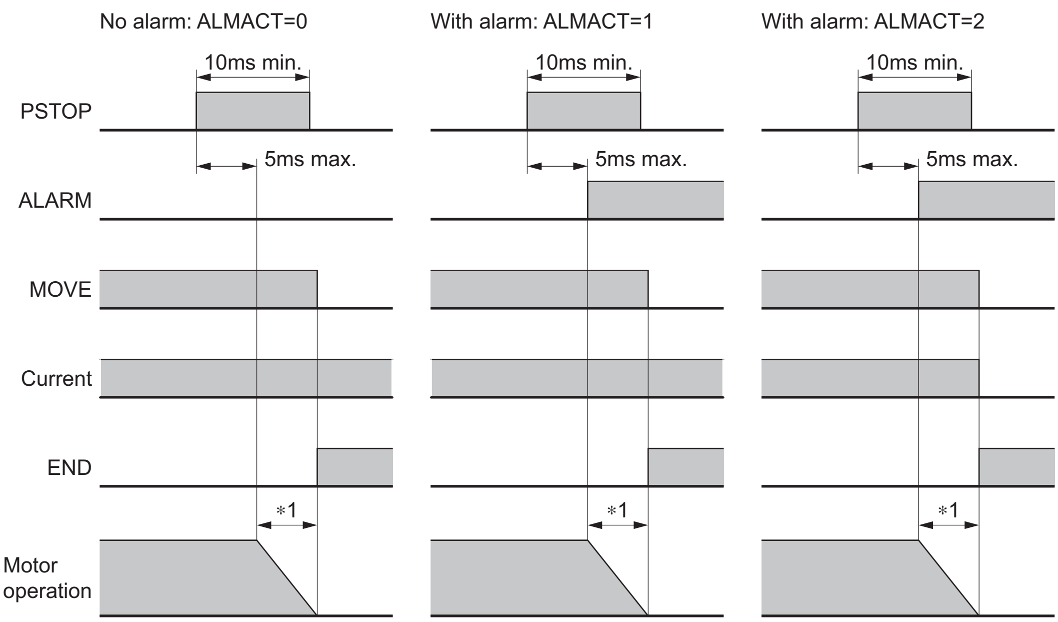
## ■ Pausing Index Operation (PAUSE, PSTS)




*1 Depends on the program.
*2 Depends on the velocity filter, load condition and settling time at stop.

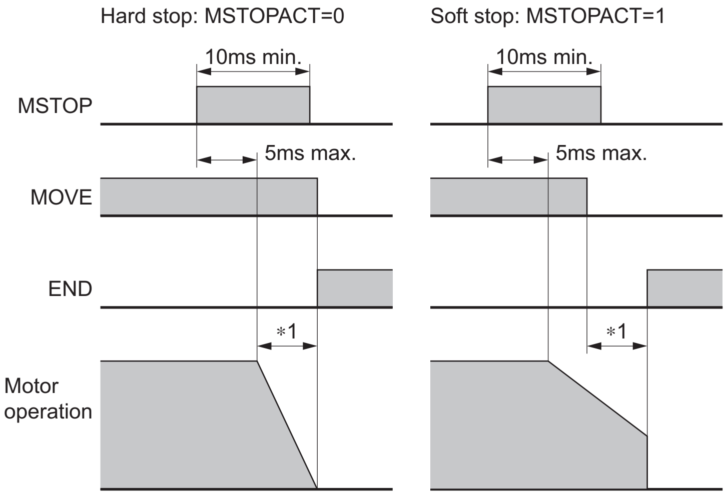
**Note** *The START input will clear a PAUSEd state.*

## ■ When the PSTOP Input is Turned ON



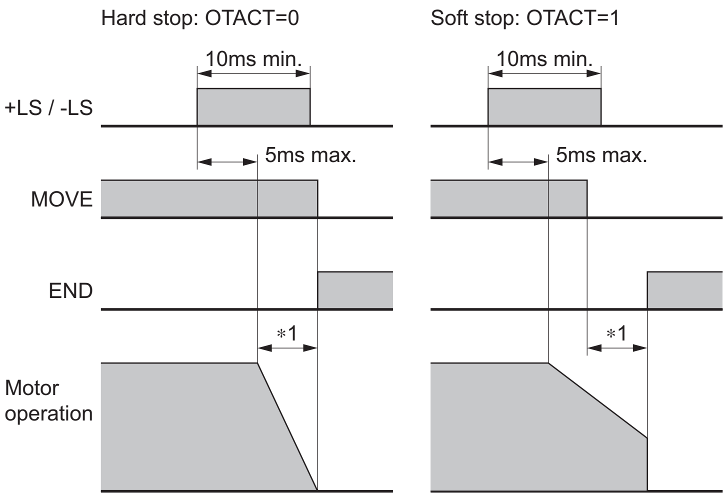

*1 Depends on the velocity filter, load condition and settling time at stop.

## ■ When the MSTOP Input is Turned ON



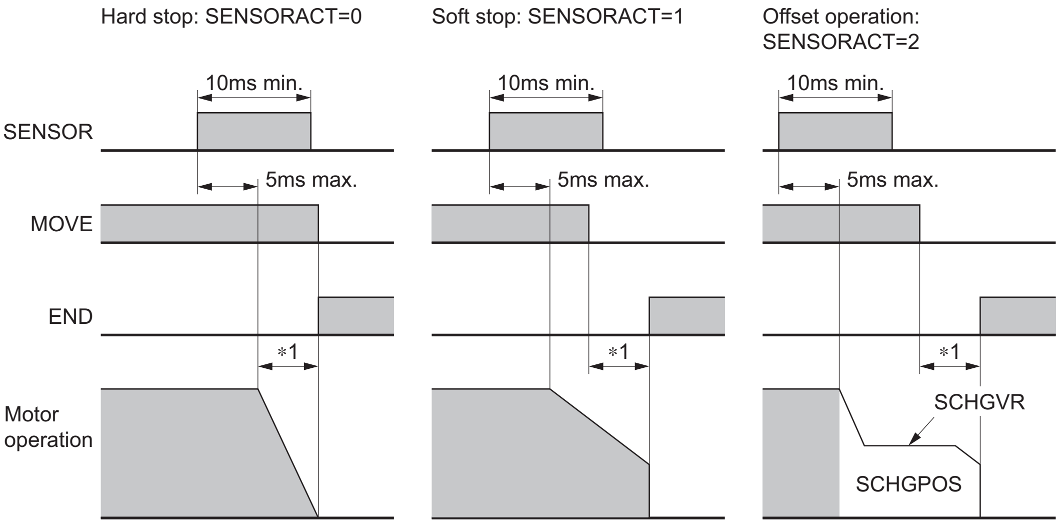

∗1 Depends on the velocity filter, load condition and settling time at stop.

## ■ When the (+LS, −LS) Input is Used



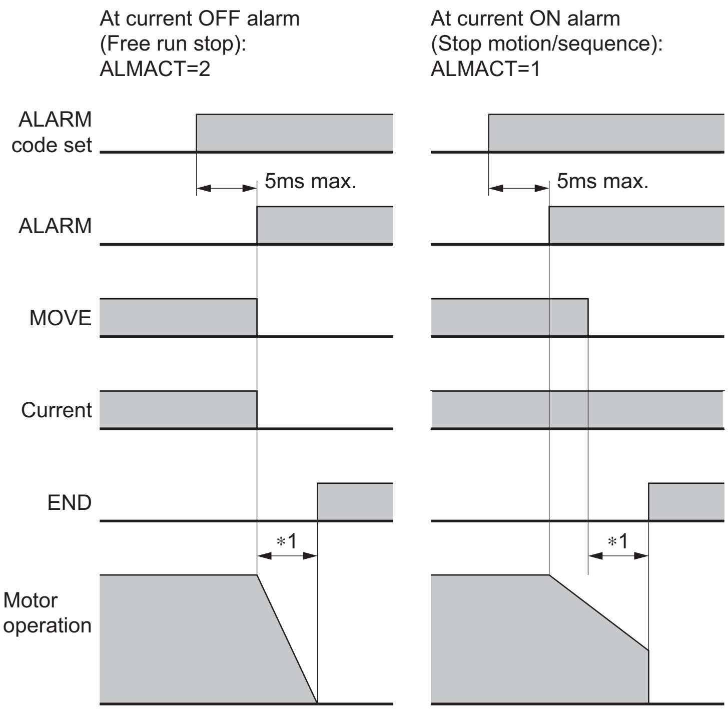

∗1 Depends on the velocity filter, load condition and settling time at stop.

## When the SENSOR Input is Used



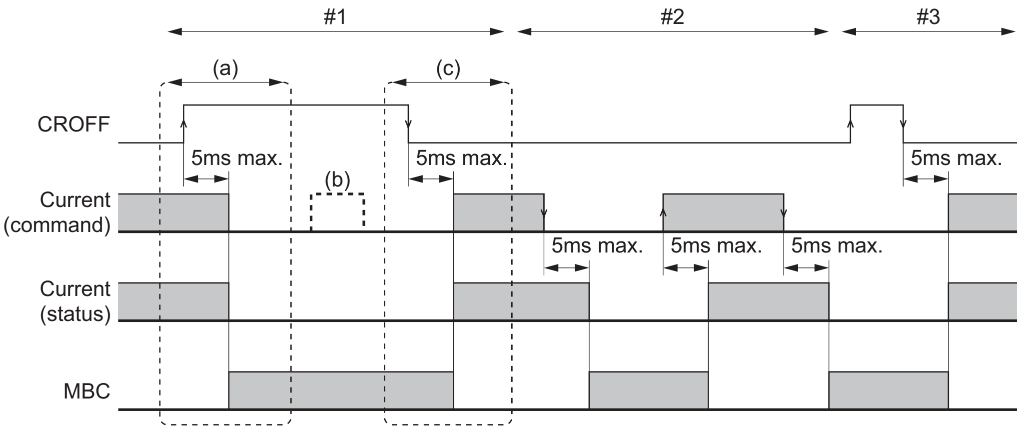

*1 Depends on the velocity filter, load condition and settling time at stop.

## When the ALARM is Occurred




*1 Depends on the velocity filter, load condition and settling time at stop.

## D.3 CROFF Input and MBC Output



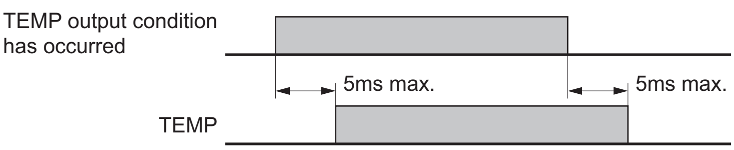

1. The order for priority: CROFF＞CURRENT
   (a) When the device is in an excited state and the active edge of CROFF is detected.
   → CURRENT=1→0、remove the excitation state、MBC output goes active
   (b) When the CROFF input is active.
   Writing to the CURRENT command is invalid. (read only)
   (c) When the non-active edge of the CROFF input is detected.
   → CURRENT=0→1、return to an excitation state、MBC output goes non-active

2. Excitation state ON/OFF by CURRENT command
   When the CROFF input is non-active.
   CURRENT=1→0, remove the excitation state, MBC output is active
   CURRENT=0→1, return to an excitation state, MBC output is non-active

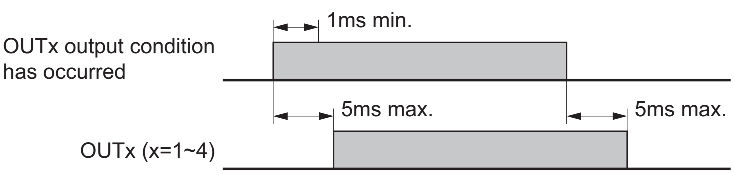
3. CROFF input with a non-excitation state. (CURRENT=0)
   Nothing occurs if the active edge or active level of the CROFF input is detected while CURRENT=0.
   When the non-active edge of CROFF input is detected.
   → CURRENT=0→1、return to excitation state、MBC output goes non-active

## D.4 Output

- TEMP output

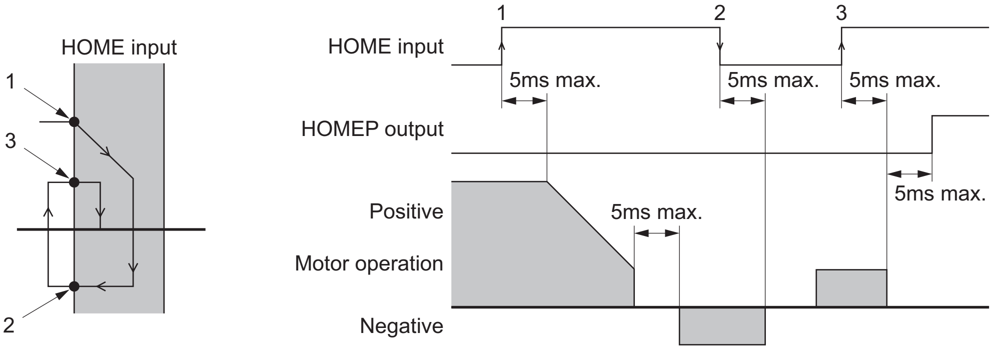



- OUT1−OUT4 output

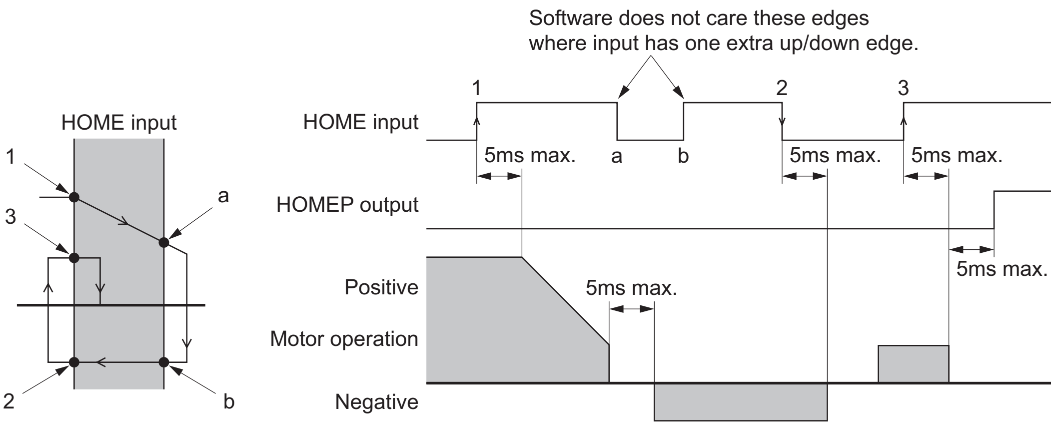


## D.5 Mechanical Home Seeking

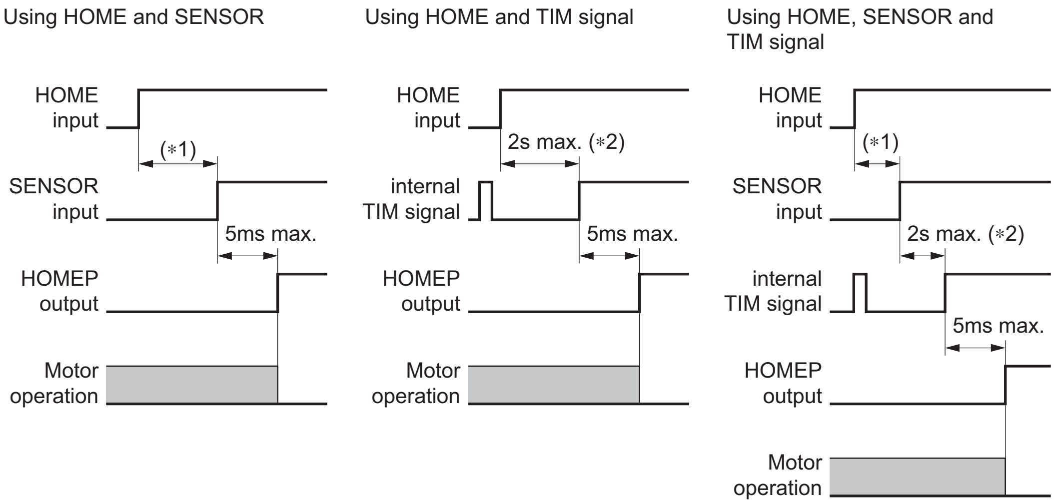
- When the mechanical home seeking operation is completed without passing the HOME input.




- When the mechanical home seeking operation is completed with passing the HOME input once.




- Stopping operation

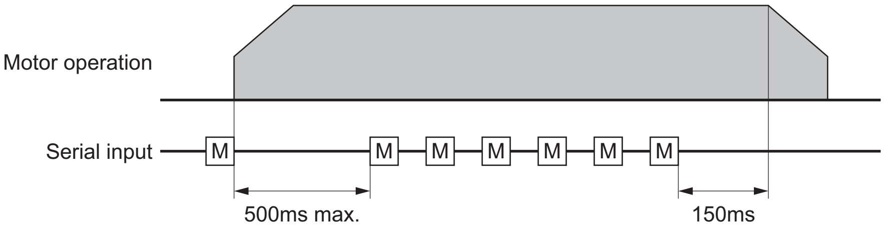


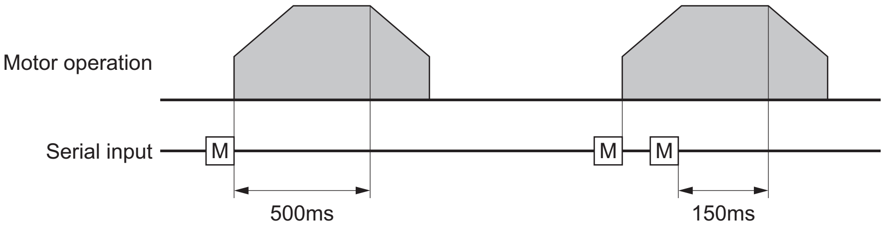

∗1 Depends on the HOMELS and SENSOR position
∗2 Depends on the SENSOR, rotor position and VS. The minimum value is limited to 0.2 [Rev/sec.]

## D.6 Teaching Operation

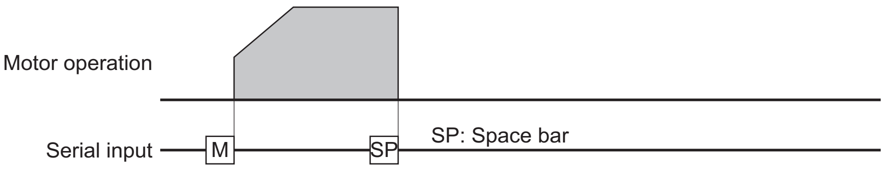
- When holdinf the key down

M : CW scan input key on qwerty keyboard.




- To stop motion immediately



# Appendix E Command Cross Reference

This chapter provides a brief command cross reference between the αSTEP-One to other Oriental Motor controller products.

| Command Description | Product | | | |
|---|---|---|---|---|
| | **ASX66** | **AS Plus** | **SC8800 (E)** | **EMP400** |
| Move continuous CW | MCP | MCP | MC, H± | SCAN1/2 |
| Move continuous CCW | MCN | MCN | MC, H± | SCAN1/2 |
| Move absolute | MA | MA | MA | ABS1/2 |
| Distance for incremental moves | DIS (I) | DIS (I) | D, H± | D |
| Move time | − | − | MT | − |
| Velocity filter | VFIL | VFIL | − | RAMP |
| Report | REPORT | REPORT | R | R |
| Send ASCII string | SAS | SAS | SAS | − |
| Distance scaling | DPR,<br>GEAR1,<br>GEAR2,<br>UU | GEAR1,<br>GEAR2 | DSCALE | UNIT |

# Appendix F　ASCII Data

This chapter explains ASCII data available in the αSTEP-One device.

## ■ Abbreviation for ASCII Data

For convenience, following expressions are used in this manual.

| | Description | ASCII Hex (Dec) |
|---|---|---|
| [BEL] | BEL1 | 07h (7) |
| [BS] | Back Space | 08h (8) |
| [LF] | Line Feed | 04h (10) |
| [CR] | Carriage Return | 0Dh (13) |
| [SP] | SPace | 20h (32) |
| [ESC] | ESCape | 1Bh (27) |
| [EOL] | End Of Line<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 [69h] | Input data buffer count is under 80: | Echo back entered value, store into input buffer. |
| | Input data buffer count is 80: | Send [BEL]. |
| [EOL] | Start parsing commands. Clear input buffer. | |
| [BS] | Any data exist in input buffer: | Send [BS] [SP] [BS].<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. | |

# Appendix G Command Format

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

**Memo** See "ASCII Data" on page 325 for abbreviation for ASCII data.

## ■ Parameters

An "=" between a parameter and parameter value is required. If the parameter value is a constant, a space can be used instead of an "=".

- Format

[Parameter] [=] [Parameter value]
[Parameter] [SP] [Parameter value (constant)]

- Examples

| Condition | Example |
|---|---|
| Parameter value is constant | DIS=1.234, DIS 1.234 |
| Parameter value is variable | DIS=A (Available in sequence only) |
| Parameter value is equation | DIS=A*1.5 (Available in sequence only) |

## ■ Commands

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

- Format

[Command] [SP] [Argument]

- Examples

| Condition | Example |
|---|---|
| No parameter | MI |
| Parameter is constant | MA 1.234 |
| Parameter is variable | MA POS [1] |
| Parameter is string | RUN Test |

## ■ 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 f characters on a one line is 80.

- Example

>DIS 1.234; VR 3; TA 0.5; TD 0.1; MI [EOL]

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

| VERBOSE=1 (default) | VERBOSE=0 |
|---|---|
| >PC<br>PC=0.123 Rev<br>>DIS=5.678<br>DIS=5.678 REV<br>>HOMETYP<br>HOMETYP=1 [2 sensor , TIM=OFF, SENSOR=OFF] | >PC<br>0.123<br>>DIS=5.678<br>>HOMETYP<br>0 |

**Note** *Also the ECHO command defines the echo back ON/OFF for entered ASCII data. Default is ECHO=1 (ON) echo back. If ECHO=0 (OFF) , there will no reply for the entered ASCII data. Display of parameter readout or SAS command from sequence is not affected by ECHO=, they are always displayed (See page 249 for SAS command).*

• Please contact your nearest Oriental Motor office for further information.


ORIENTAL MOTOR U.S.A. CORP.
Technical Support Tel:(800)468-3982
8:30 A.M. to 5:00 P.M., P.S.T. (M-F)
7:30 A.M. to 5:00 P.M., C.S.T. (M-F)
E-mail: techsupport@orientalmotor.com
www.orientalmotor.com

ORIENTAL MOTOR (EUROPA) GmbH
Headquarters and Düsseldorf Office
Tel:0211-52067-00 Fax:0211-52067-099
Munich Office
Tel:089-3181225-00 Fax:089-3181225-25
Hamburg Office
Tel:040-76910443 Fax:040-76910445

ORIENTAL MOTOR (UK) LTD.
Tel:01256-347090 Fax:01256-347099

ORIENTAL MOTOR (FRANCE) SARL
Tel:01 47 86 97 50 Fax:01 47 82 45 16

ORIENTAL MOTOR ITALIA s.r.l.
Tel:02-93906346 Fax:02-93906348

TAIWAN ORIENTAL MOTOR CO.,LTD.
Tel:(02)8228-0707 Fax:(02)8228-0708

SINGAPORE ORIENTAL MOTOR PTE LTD
Tel:(6745)7344 Fax:(6745)9405

ORIENTAL MOTOR (MALAYSIA) SDN. BHD.
Tel:(03)22875778 Fax:(03)22875528

ORIENTAL MOTOR (THAILAND) CO.,LTD.
Tel:66-2-254-6113 Fax:66-2-254-6114

INA ORIENTAL MOTOR CO.,LTD.
KOREA
Tel:(032)822-2042~3 Fax:(032)819-8745

ORIENTAL MOTOR CO.,LTD.
Headquarters Tokyo, Japan
Tel:(03)3835-0684 Fax:(03)3835-1890


Printed on Recycled Paper