

Commander Theory of Operation and Command Reference

Table of contents

Copyright	10
Cautions	10
NPA Commander Online Manual	11
1.0 Introduction and Overview	12
1.1 Product Overview	14
1.2 Specifications	15
2.0 Operating principle	16
2.1 General Connections and Setup	19
2.1.1 Connect Multiple Commander Cores in a System	21
2.1.2 Motor interface	22
2.1.2.1 Stepper	24
2.1.2.2 Digital Servo	26
Encoder Inputs	27
2.1.3 External Connections	29
2.1.3.1 Dedicated Input/Outputs	30
2.1.3.1.1 Mechanical input signals	31
2.1.3.1.1.1 Limit Switches	31
2.1.3.1.1.2 Slowdown switches	32
2.1.3.1.1.3 Home switch	33
2.1.3.1.1.4 Soft-Limits	33
2.1.3.1.2 Emergency stop	35
2.1.3.1.3 External Stop	36
2.1.3.1.4 External Start	38
2.1.3.1.5 Manual pulse generator (MPG) inputs	39
2.1.3.2 High-speed Inputs/Outputs	39
2.1.3.2.1 High-Speed Digital Inputs	40
2.1.3.2.2 High-Speed Digital Output	42
2.1.3.3 General-purpose input/output	45
2.1.4 Motion Profile	46
2.1.5 Power-on settings	50
2.1.6 System settings	50
2.1.6.1 ERC (deflection counter clear)	51
2.1.6.2 INP (in-position)	52
2.1.6.3 Input modes and the polarity of the input/output logic	52
2.1.7 Flash Memory	54
2.2 Single-axis and Multi-axis operations	54
2.2.1 Motor Power	54
2.2.2 Jog operation	55
2.2.3 Stopping	56
2.2.4 Positioning Operation	57
2.2.4.1 Absolute or Incremental	57
2.2.4.2 Individual Position Moves	58
2.2.5 Homing Operation	59
2.2.5.1 Z-move Operation	59
2.3 Interpolation Operation	60
2.3.1 Linear Interpolation	61
2.3.2 Circular Interpolation	62

2.3.2.1 Arc Interpolation	63
2.3.3 Helical or Tangential interpolation	65
2.3.4 Buffered Interpolation operation	66
2.4 On-the-fly-speed/target position change	68
2.4.1 On-the-fly Speed Change	68
2.4.2 On-the-fly Target Position Change (Target position change)	69
2.5 Advanced Functions	70
2.5.1 Joystick operation (Analog input control)	70
2.5.1.1 Direction control	71
2.5.1.2 Speed control	72
2.5.1.3 Position control	73
2.5.2 Manual pulse generator (MPG) operation	74
2.5.2.1 MPG signal input mode	75
2.5.2.2 Electronic gearing the MPG signal	76
2.5.2.3 Encoder Frequency and Impulse Multiplier/Divider	77
2.5.3 StepNLoop Closed Loop Control operation	77
2.6 Status monitoring function	81
2.6.1 Counter Values	81
2.6.1.1 Output Pulse Counter	81
2.6.1.2 Feedback Pulse Counter	82
2.6.1.3 Deviation Counter	82
2.6.2 Axis status register	83
2.6.2.1 Axis Status Acquisition Command	83
2.6.2.2 Pulse Speed register	84
2.6.2.3 Error clear command	84
2.7 Standalone program Specification	85
2.7.1 Program Development	85
2.7.2 Conditional loop control	86
2.7.3 Multi-thread	86
2.7.4 Subroutines	86
2.7.5 Error Handling	87
2.7.6 Standalone Variables	87
2.7.7 Compiling, Saving, Downloading	88
2.7.8 Standalone Control	88
2.7.9 Standalone Status	89
2.7.10 Standalone Run on Boot-Up	89
3.0 Communication Interface	90
3.1 USB Communication	90
3.1.1 Typical USB Setup	91
3.1.2 HID Communication API	91
3.1.2.1 How to use	92
3.1.2.2 GetCommanderHIDnumber	92
3.1.2.3 GetCommanderHIDName	93
3.1.2.4 OpenCommanderHID	94
3.1.2.5 CloseCommanderHID	94
3.1.2.6 SendReceiveCommanderHID	95
3.1.2.7 Communication Error Codes	96
3.2 Serial Communication	97
3.2.1 Communication Port Settings	97
3.2.2 ASCII Protocol	98

3.3 Identification Number	98
3.4 Windows GUI	99
3.4.1 Commander Control Software	101
3.4.2 Nippon Pulse Terminal Utility	101
4.0 ASCII Language Specification	101
4.1 ASCII Command Set	101
4.2 Error Codes	111
4.3 Examples of command control from a PC	112
4.3.1 Start and stop of jog move	112
4.3.2 Position Move	113
4.3.3 Obtaining and clearing the counter value	114
4.3.4 Homing Operation	115
4.3.5 Linear interpolation operation	116
4.3.6 Circular Interpolation Operation	116
4.3.7 On-the-fly speed change	117
4.3.8 Synchronization Function	118
4.3.9 Control of general purpose input/output	119
5.0 Standalone Language Specification	119
5.1 Using A-Script programming interface	120
5.1.1 Program Development	121
5.1.2 Subroutines	121
5.1.3 Error Handling	121
5.1.4 Multi-thread	121
5.1.5 Compiling, Saving, Downloading	121
5.2 Standalone Command Set	122
5.3 Error Codes	125
5.4 Example Standalone Programs	126
5.4.1 Standalone Example Program 1 – Single Thread	126
5.4.2 Standalone Example Program 2 – Single Thread with buffer	127
5.4.3 Standalone Example Program 3 – Single Thread	127
5.4.4 Standalone Example Program 4 – Single Thread	128
5.4.5 Standalone Example Program 5 – Single Thread	128
5.4.6 Standalone Example Program 6 – Single Thread with subroutine	129
5.4.7 Standalone Example Program 7 – Single Thread with error handling	130
5.4.8 Standalone Example Program 8 – Multi-thread	131
5.4.9 Standalone Example Program 9 – Multi-thread with subroutine	131
5.4.10 Standalone Example Program 10 – Multi-thread with subroutine	133
6.0 Dimensions and Connectivity	135
6.1 Dimensions	136
6.2 Connectivity	136
6.2.1 Pin-out	137
6.2.1.1 Pin Assignment Diagram	138
6.2.1.2 Pin Descriptions by Functions	138
6.2.1.3 Recommended Footprint	144
6.2.2 Interface Boards	145
7.0 Electrical Characteristics	145
7.1 Electrical and Thermal Specifications	146
7.2 Handling Precautions	147
Appendix A: Homing Mode Actions	148
A.1 Homing Mode 0	150

A.2 Homing Mode 1	152
A.3 Homing Mode 2	152
A.4 Homing Mode 3	153
A.5 Homing Mode 4	154
A.6 Homing Mode 5	155
A.7 Homing Mode 6	156
A.8 Homing Mode 7	157
A.9 Homing Mode 8	158
A.10 Homing Mode 9	159
A.11 Homing Mode 10	160
A.12 Homing Mode 11	161
A.13 Homing Mode 12	162
Appendix B: Speed Settings	163
B.1 Acceleration/Deceleration Range	164
B.2 Acceleration/Deceleration Range – Positional Move	164
Appendix C - Interpolation	165
C.1 Linear Interpolation	166
C.2 Arc Interpolation	166
C.3 Circular Interpolation	167
C.4 Helical or Tangential interpolation (Circular Interpolation Synchronized with the U-axis)	167
C.5 Additional notes regarding Interpolation functions	167
Revision History	168
CMD Command Reference	169
Firmware Compatibility	171
Commander Operating principle	173
Functional Listing	174
System Setup Settings	175
System Setup	175
Motor Setup	176
Encoder Setup	176
Limits/Safety Setup	177
I/O Setup	178
Basic Motion Commands	179
Motion Settings	180
Axis motion	181
Stop Motion	182
Homing	182
Home modes cross reference - PMX and CMD	183
Advanced Functions	184
Interpolated operations	184
Linear Interpolation	185
Circular Interpolation	186
On-the-fly-speed/target position change	186
Buffer Operation	187
Interpolated Motion Buffer	188
Synchronization Output Buffer	189
Synchronization	189
Joystick	190
StepNLoop	192

Latching	193
Manual Pulse Generator	193
Standalone Programming Commands	194
System Status	195
Environment setting	197
Controller information	198
Service Commands	199
External Connections	200
Dedicated I/O	201
Mechanical input signals	201
Limit Switches	203
Soft-Limits	204
Slowdown switch	205
Home	206
Emergency stop	206
High speed I/O	206
General purpose I/O	207
Summary Error Codes by Controller Family	208
Alphabetical Command Listing	209
Summary Command Reference Table by Controller Family	209
; – Comments	216
ABORT – Immediate Stop	217
ABS – Absolute Coordinate Mode	218
ACC – Acceleration Time	218
AI – Analog Input	220
AO – Analog Output	221
ARC – Circular Interpolation (Arc)	222
ARCT – Circular Interpolation (Helix Arc)	223
BF – Buffer Disable	224
BO – Buffer Enable	225
BSTART, ISTART – Buffer Start	226
BSTAT – Buffer Status	227
BTUNE – Tuning for Motion Buffer	228
CIR – Circular Interpolation	229
CIRT – Circular Interpolation (Helix)	230
CLR – Clear Axis Error	232
D – Deviation Counter	233
DB – Baud Rate Setting	234
DEC – Deceleration Time	235
DELAY – Standalone DELAY Command	237
DI – Digital Input Status	238
DIP – Digital Input Polarity	239
DO – Digital Output Status	240
DOBOOT – Digital output boot-up state	241
DOP – Digital Output Polarity	243
E – Encoder value (axis)	244
EINT – Interpolation Enable	245
EO – Axis Enable Status	246
EOBOOT – Axis Enable Status on Boot-up	248
EP – Encoder value (all)	249

ERC – Deviation Counter Clear Conditions	250
ERCD – Deviation Counter Clear Delay	252
ERCP – Deviation Counter Clear Pulse Width	253
ESTOP – Emergency Stop	254
EXSTA – External Start	255
EXSTP – External Stop	257
GS – Subroutine	258
HOME, H – Homing	259
Home mode Cross reference	262
CMD/PMX Homing Mode 0 [H[axis][+/-]]	264
CMD Homing Mode 1 PMX Mode 4	266
CMD/PMX Homing Mode 2 [ZH[axis][+/-]]	268
CMD Homing Mode 3	269
CMD Homing Mode 4	270
Homing Mode 5	270
CMD Homing Mode 6 PMX Mode 1	271
L[axis][+/-]	272
Homing Mode 7	273
Homing Mode 8	273
Homing Mode 9	274
Homing Mode 10	274
Homing Mode 11	276
Homing Mode 12	276
HSPD – High Speed Set Point	277
I – Interpolation (X, Y, Z Linear Interpolation)	279
I – 4 Axis Vector (X, Y, Z, U Linear Interpolation)	281
IACC – Automatic acceleration for buffered interpolation	282
ID – Identification Number	283
IERR – Alarm/Limit Registration	284
IF/ELSEIF/ELSE/ENDIF – Conditional loop	286
INC – Incremental Coordinate Mode	287
INP – Wait for In Position	288
IO – General-Purpose I/O Status	289
IOBOOT – General-Purpose I/O Status on Boot-up	292
IOCFG – General-Purpose I/O Configuration	294
IOP – General-Purpose I/O Polarity	296
IP – IP Address	297
J – JOG	298
JDEL – Joystick speed change delta	299
JENA – Joystick enable	300
JKA – Jerk Control Acceleration	301
JKD – Jerk Control Deceleration	303
JLIM – Joystick Limits	305
JMAX – Joystick Maximum Voltage	307
JMIN – Joystick Minimum Voltage	308
JSPD – Joystick Max Speed	310
JTOL – Joystick Deadzone	311
LSPD – Low Speed Set Point	313
LT – Latch Enable	315
LTE – Latched Encoder Position	315

LTP – Latched Step Position	316
LTS – Latch Function Status	317
MM – Move Mode	318
MP – Manual Pulse Generator Counter	318
MPD – Manual Pulse Generator Divider	319
MPE – Manual Pulse Generator Enable	321
MPM – Manual Pulse Generator Multiplier	322
MSR – MicroStep Resolution	323
MST – Motor Status Acquisition Command	324
Operators – A-script operators	326
Arithmetic Operators examples	328
Bitwise Operators examples	328
P – Pulse counter value (axis)	330
POL – Polarity	331
PRG/END – Standalone PRG/END Command	333
PP – Pulse counter value (all)	334
PS – Pulse Speed	335
PWM – Pulse Width Modulation Duty Cycle	336
SASTAT – Standalone Program Status	337
SACTRL – Standalone Control	338
SAMAX – Standalone Maximum Lines	339
SAP – Standalone Error Handling	340
SCV – S-Curve Acceleration	341
SDC – Slow Down Configuration	343
SDE – Slow Down Enable/Disable	344
SFL – Soft-limit Enable/Disable	345
SFLN – Negative Soft-limit setting	347
SFLP – Positive Soft-limit setting	348
SFLSD – Soft-limit Stop Mode	349
SFLSO – Soft-limit Source	351
SL – StepNLoop Enable	352
SLA – StepNLoop Attempts	353
SLE – StepNLoop Error Range	354
SLOAD – Standalone Program Autostart	355
SLR – StepNLoop Pulse Conversion	356
SLRA – StepNLoop Automatic Pulse Conversion	357
SLS – StepNLoop status	358
SLT – StepNLoop In Position Range	360
SN – Commander Serial Number	361
SPC – Standalone index line	362
SPL – Standalone line	362
SSPD – On-the-Fly Speed Change	363
SSPDM – Speed Range	364
STOP – Controlled Stop	366
STORE – Save System Settings	367
SSTART – Simultaneous start	368
SUB/ENDSUB – Subroutines	369
SYNC – Synchronization Mode	370
SYNF – Synchronization mode off	373
SYNBO – Sync Position Buffer Enable	374

SYNBF – Sync Position Buffer Disable	374
SYNBSTAT – Buffer Status	375
SYNMAX – Synchronization Window Upper Limit	376
SYNMIN – Synchronization Window Lower Limit	377
SYNO – Synchronization mode on	379
SYNP – Synchronization Position	380
SYNS – Synchronization Status	382
SYNWF – Synchronization Window Off	383
SYNWO – Synchronization Window On	384
T – On-the-Fly Target Position Change	385
V – Variables	386
VEC– Vector Motion Enable	388
VER – Firmware Version	389
WAIT – Standalone wait command	390
WHILE/ENDWHILE – Conditional loop	391
X, Y, Z, U – Positioning Function	392
ZCNT – Z-index pulse counter	393
ZMOVE – Z-move Operation	395
%RST – Soft reset	396
Unpublished	397
%BLD – Enter boot-load mode	397
%DID – SOC TYPE ID	398
%RDATA – Reset the data section of flash	400
REG – p645 Register read/write	401
SA – Standalone Program Line	404
%SBLK – LED Blink bit	404
%SCLK – Set System Clock Frequency	405
%SMAC - MAC address	406
%SRID – Auto ID set	408
%WSN – Write Commander Serial Number	409
Revision History	410
Contact information	410

Copyright

Copyright © 2019-2026 by NIPPON PULSE AMERICA, INC. All Rights Reserved.

First edition, October 2019

Current edition, July 2026

NIPPON PULSE AMERICA, INC. copyrights this document. You may not reproduce or translate into any language in any form any part of this publication without written permission from NIPPON PULSE AMERICA, INC.

NIPPON PULSE AMERICA, INC. makes no representations or warranties regarding the content of this document. We reserve the right to revise this document any time without notice and obligation.

Cautions

- Copying any part of this manual without written approval is prohibited.
- The specifications of this controller may be changed to improve performance or quality without prior notice.
- Although this manual was produced with the utmost care, if you find any points that are unclear, incorrect, or have inadequate descriptions, please let us know. The best way to do this is by clicking the [Page issue?](#) at the bottom of the page where you believe there is an issue. This will send NPA an e-mail with the link to the page, and you can describe the issue that you found and we can correct it quickly.
- We are not responsible for any results that occur from using this controller, regardless of the item above.
- The Commander core is designed for use in commercial apparatus (office machines, communication equipment, measuring equipment, and household appliances). If you use it in any device that may require high quality and reliability, or where faults or malfunctions may directly injure humans or endanger human survival (such as in nuclear power control devices, aviation devices or spacecraft, traffic signals, fire control, or various types of safety devices), we will not be liable for any problem that occurs, even if it was directly caused by the Commander core. Customers must provide their own safety measures to ensure appropriate performance in all circumstances.

Explanation of the descriptions in this manual

- The "X" "Y" "Z" and "U" of terminal names and bit names refer to the X-axis, Y-axis, Z-axis, and U-axis, respectively.
- Terminals with a / (ex. /RST) are negative logic. Their logic cannot be changed. Terminals without a / are positive logic. Their output logic can be changed.
- When describing the bits in registers, "n" refers to the bit position. A "0" means that the bit is in position 0 and that it is prohibited to write to any bit other than "0."

Finally, this bit will always return a "0" when read out.

NPA Commander Online Manual

Commander Theory of Operation

Commander Motion Controller Core (CMD-4CR) is an advanced, self-contained and programmable four-axis hybrid motion controller. The basic design of the Commander offers OEMs the convenience of a standard controller and the technology of a fully engineered controller. As such it exhibits the best qualities of both types of controllers:

- Faster path-to-market
- Flexible design adaptable to a wide range of applications
- No compromise on capabilities and processing speed
- Scalable from prototype to production without software changes
- Intuitive to use
- Cost-effective design



The Commander core architecture facilitates integration into custom hardware. Communication to the Commander core can be established via USB, UART (RS-485), **SPI, I²C, Ethernet, or CAN bus** (Gen 5 feature). It supports downloading multiple standalone programs to the device, enabling fully independent operation without a host.

The Commander core is a CMOS hybrid IC designed to provide the oscillating, high-speed pulses needed to drive stepper motors and digital servomotors (pulse string input types) using various commands. It can offer various types of control over the pulse strings and therefore the motor performance. These include continuous feeding, positioning, and origin return, at a constant speed, and linear or S-curve acceleration/deceleration.

The Commander core controls four axes. It can control the linear interpolation of two to four axes, circular interpolation between any two axes, confirm controller operation status, and output an interrupt with various conditions. It also integrates an interface for servo motor drivers.

These functions can be used with simple commands. The intelligent design philosophy reduces the burden on the CPU. Commander is a robust and powerful controller with numerous operating capabilities.

In this manual:

- [Introduction and Overview](#)
 - o [Product Overview](#)

- o [Specifications](#)
- [Operating Principle](#)
 - o [General Connections and Setup](#)
 - o [Single-axis and multi-axis operations](#)
 - o [Interpolation Operation](#)
 - o [On-the-fly-speed/target position change](#)
 - o [Advanced Functions](#)
 - o [Status monitoring function](#)
 - o [Standalone program specification](#)
- [Communication Interface](#)
 - o [USB Communication](#)
 - o [Serial Communication](#)
 - o [Identification Number](#)
 - o [Windows GUI](#)
- [ASCII Language Specification](#)
 - o [ASCII Command Set](#)
 - o [Error Codes](#)
 - o [Examples of command control from a PC](#)
- [Standalone Language Specification](#)
 - o [Using A-Script programming interface](#)
 - o [Standalone Command Set](#)
 - o [Error Codes](#)
 - o [Example Standalone Programs](#)
- [Dimensions and Connectivity](#)
 - o [Dimensions](#)
 - o [Connectivity](#)
- [Electrical Characteristics](#)
 - o [Electrical and Thermal Specifications](#)
 - o [Handling Precautions](#)
- [Appendix: Homing Mode Actions](#)
- [Appendix: Speed Settings](#)
- [Appendix: Interpolation](#)

https://www.nipponpulse.com	Copyright notice	Revision History	Cautions

1.0 Introduction and Overview

Introduction

Commander motion controller core (CMD-2CR, CMD-4CR, CMD-8CR) is a first-in-class, advanced multi-axis standalone-programmable motion controller hybrid integrated circuit (IC). The Commander core provides original equipment manufacturer (OEMs) the convenience of an off-the-shelf solution combined with the flexibility of a fully customized controller.

The Commander core architecture facilitates integration into custom hardware. Communication to the Commander core can be established via USB, UART (RS-485), SPI, I²C, Ethernet, or CAN bus (Gen 5 feature). It supports downloading multiple standalone programs to the device, enabling independent operation without a host.

The Commander core is a CMOS hybrid IC, specifically engineered to generate precise, high-speed pulse trains required for driving stepper motors and pulse-controlled digital servomotors. Using simple and intuitive commands, it offers precise control of motor behavior, including continuous feed, precise positioning, homing (origin return), constant speed movement, and acceleration/deceleration profiles (linear and S-curve).

The Commander core controls up to eight axes (CMD-8CR, Gen 5), enabling linear interpolation across two to eight axes and circular interpolation between any two axes. It provides comprehensive operational status feedback, supports interrupts under configurable conditions, and integrates seamlessly with servo motor drivers.

These advanced capabilities are accessible through simple, intuitive commands, simplifying integration and operation. Its intelligent design reduces CPU overhead, resulting in a robust and powerful solution with extensive operational flexibility.

Feature	Description
Standalone control separated from PC	Operate independently without a PC, using onboard program execution capabilities.
Programming language (A-Script)	Easy-to-use A-Script language; create programs on a PC, then compile and download to the Commander core.
Operation check by utility software	Easy to use monitoring screens for motor status, I/O status, and system operation status
Compile, write and read standalone program by utility software	Easy to use tools to write, edit and debug software
Joystick operation with analog input (X-axis, Y-axis)	Utilize joystick for manual control of motion
13 types of homing mode	Flexibility in how machine mechanisms are setup and homing operation is initiated
Manual pulse generator operation	
On-the-fly speed change (speed overridden during movement)	Quickly change operation speed on-the-fly (manual)
On-the-fly target position change (target position overridden during movement).	Target position can be updated on-the-fly (manual)
2 to 4-axis linear interpolation	Create a variety of paths utilizing

	interpolation options
2-axis circular and arc interpolation, including 3rd axis for Helical motion	Create circles, arcs or helical motion utilizing x, y and/or z axes
Sync output configuration	
Absolute positioning or incremental positioning can be selected	Flexibility of positioning coordinates as absolute from a home location or incremental
12 inputs and 12 outputs of general-purpose input/output signals	Flexibility in outputs, including configuration of general-purpose I/O, dedicated I/O and high-speed inputs

1.1 Product Overview

Overview

The Commander core is based on the PCL6045BL motion control LSI and an ARM Core processor.

Power

- +3.3VDC Power Input
- 3.3V logic level, 5V tolerant

Communication

- USB 2.0, HID compatible. No driver signature required.
- RS-485 ASCII, selectable 9600, 19200, 38400, 57600, 115200 baud rates available
- I²C bus [1 channel], available for future expansion for interface with external IC
- SPI bus [2 channels], available for future expansion for interface with external IC
- Ethernet, available for future expansion

Control

- Command control from PC through USB or serial communication
- Standalone control allows operation separated from PC, using a BASIC-like programming language (A-SCRIPT)

Utility software

- Allows for operation check by utility software
- Compile, write, and read standalone programs

Motor interface

- Maximum pulse output rate of 6.55M PPS
- Stepper motor interface with pulse, direction, and enable outputs for XYZU
- Servo motor interface with pulse, direction, enable, in-position, servo alarm, and error clear outputs for XYZU

Homing routines

13 different built-in homing routines available

Positioning operations

- Selectable trapezoidal or s-curve acceleration
- Automatic high-speed correction
- Selectable absolute or incremental positioning
- On-the-fly speed change (speed overridden during movement).
- On-the-fly target position change (target position overridden during movement).
- Coordinated motion
 - Linear (XYZU) (2 to 4 axis)
 - Only one linear interpolation set can be executed at a time.
 - Circular/Arc (any two axes)
 - Helix/Tangential (XYZ)
 - Continuous linear/circular/arc coordinated buffered motion for XYZ axis
- Control using manual pulse generator (MPG)
 - Electronic camming
 - Electronic gearing
 - Encoder Frequency and Impulse Multiplier/Divider
 - Manual fine axis control

Inputs/outputs

- With integrated noise filter, to reduce switch bounce
- A/B/Z encoder inputs for XYZU (TTL Compatible)
 - Maximum frequency of 6.5 MHz (26M counts with 4x multiplication)
 - StepNLoop closed-loop position verification algorithm (XYZU)
- Manual pulse generator (MPG) operation
 - A/B pulse inputs for XYZU (TTL Compatible)
 - Maximum frequency of 6.5 MHz (26M counts with 4x multiplication)
- Digital I/O
 - 4 designated high-speed inputs
 - 4 designated high-speed outputs
 - +Limit/-Limit/Home inputs for XYZU
 - Simultaneous start input
 - Emergency stop input
 - 32 configurable I/O
 - Latching input
 - Synchronization Pulse output
- Analog I/O
 - Two 10-bit analog input
 - Two PWM outputs
 - Joystick operation with analog input (X-axis, Y-axis)

Firmware upgrade and downgrade via USB firmware loader.

1.2 Specifications**Specifications**

Item	Description
Number of Axes	4 axes (X, Y, Z, and U)
Positioning control range	-134,217,728 to +134,217,727 (28-bit)
Ramping-down point setting range	0 to 16,777,215 (24-bit)
Number of registers used for setting speeds	Three for each axis (FL, FH, and FA [speed correction])
Speed setting step range	1 to 65,535 (16-bits)
Speed magnification range	Multiply by 0.1 to 100 Multiply by 0.1 = 0.1 to 6,553.5 pps Multiply by 1 = 1 to 65,535 pps Multiply by 100 = 100 to 6,553,500 pps
Acceleration/deceleration characteristics	Selectable Linear or S-curve acceleration/deceleration
Acceleration rate setting range	1 to 65,535 (16-bit)
Deceleration rate setting range	1 to 65,535 (16-bit)
Feed speed automatic correction function	Automatically lowers the feed speed for short distance positioning moves
Manual operation input	Manual pulse generator (MPG) input, push-button switch input
Interpolation functions	Linear interpolation, any 2 to 4 axes (only one set of interpolation moves at a time, i.e, cannot perform an X, Y linear interpolation and a Z, U linear interpolation at the same time); circular interpolation, any 2 axes
Operating temperature range	-40 to +85°C
Power	Single power supply of 3.3 V $\pm$ 10%
Package	150-pin Hybrid IC

2.0 Operating principle

Operating principle

The Commander core has numerous modes of operation. These include single, multi-axis, interpolation, on-the-fly speed or target position changes, status monitoring functions, and I/O operations. Any or all of the different modes of operation can occur simultaneously without affecting the performance of the other operations.

Single-axis operations involve motion of a single axis and does not require interaction with

any other axes. **Multi-axis** operations involve motion of two or more axes but does not require interaction between the axes. **Interpolated** operations involve two or more axes whose movements are interdependent. There are several types of interpolated moves. **Linear interpolation** is when two or more axes work together to move along the hypotenuse between the starting and target coordinates. For linear interpolation, each axis moves in only one direction and start and stop at the same time. Only one interpolation set can be executed at a time. **Circular interpolation** includes two axes working together to move along the outer rim of a circle. For circular interpolation, each axis will change direction and speed as required to allow for a circular path of motion. A popular subset of circular interpolation is **arc interpolation**, where two axes work together to move along the outer rim of a circle for a set number of degrees. **Helical or tangential interpolation** is when circular or arc interpolation of the X and Y-axis is combined with linear interpolation of the center point of the circle and the Z-axis. This allows for movement along a helical motion path.

The Commander core allows all of these different operations to function:

- as independent moves
- as any combination of the above moves
- or **buffered** together to allow for seamless **continuous interpolated** motion.

When operating a motor, it is possible to **jog** a motor at speed until it is told to **stop** or move to a predefined target **position**. The target position can be **absolute** to a 0 (home) point defined during one of the many **homing** processes, or can be **incremental** or relative to the motor's current position. While moving, it may be necessary to change the **target speed**, or while making a positioning move, it may be necessary to change **target position**. The Commander core can seamlessly do this without requiring any of the motors to stop. These operations are referred to as **on-the-fly speed change** or **on-the-fly target position change** respectively. It is also possible to use a **joystick** or **analog input** to control a motor. Using the **MPG** function, the motor can follow a master axis or a cam.

Regardless of the type of operation, the first step is to define your motion profile. The basics of a motion profile are: acceleration/deceleration type (**S-curve** or **linear**/trapezoidal), the initial or **low speed**, the max or **high speed**, and the amount of time needed for **acceleration** and **deceleration**.

There are a number of inputs and outputs available on the Commander core. These can be broken into two categories; dedicated high-speed (hS), and general-purpose. Dedicated

inputs/outputs do not require further interaction from the microprocessor, and usually fall into one of two categories: 1) Inputs/Outputs that automatically cause a fixed reaction, or 2) Inputs/Outputs that automatically cause a reaction that can be configured. Safety inputs such as **end limits**, **slowdown** (near the limit) inputs, **home** (origin) switch, **latching** inputs, and **synchronization** outputs are all dedicated inputs. On the other hand, the **general-purpose** outputs will change state and require the microprocessor to detect and then act on their status. In addition, the microprocessor can request the current status of any input and of the various counters, output pulse, encoder, MPG, etc. along with the full status of each axis.

In this section:

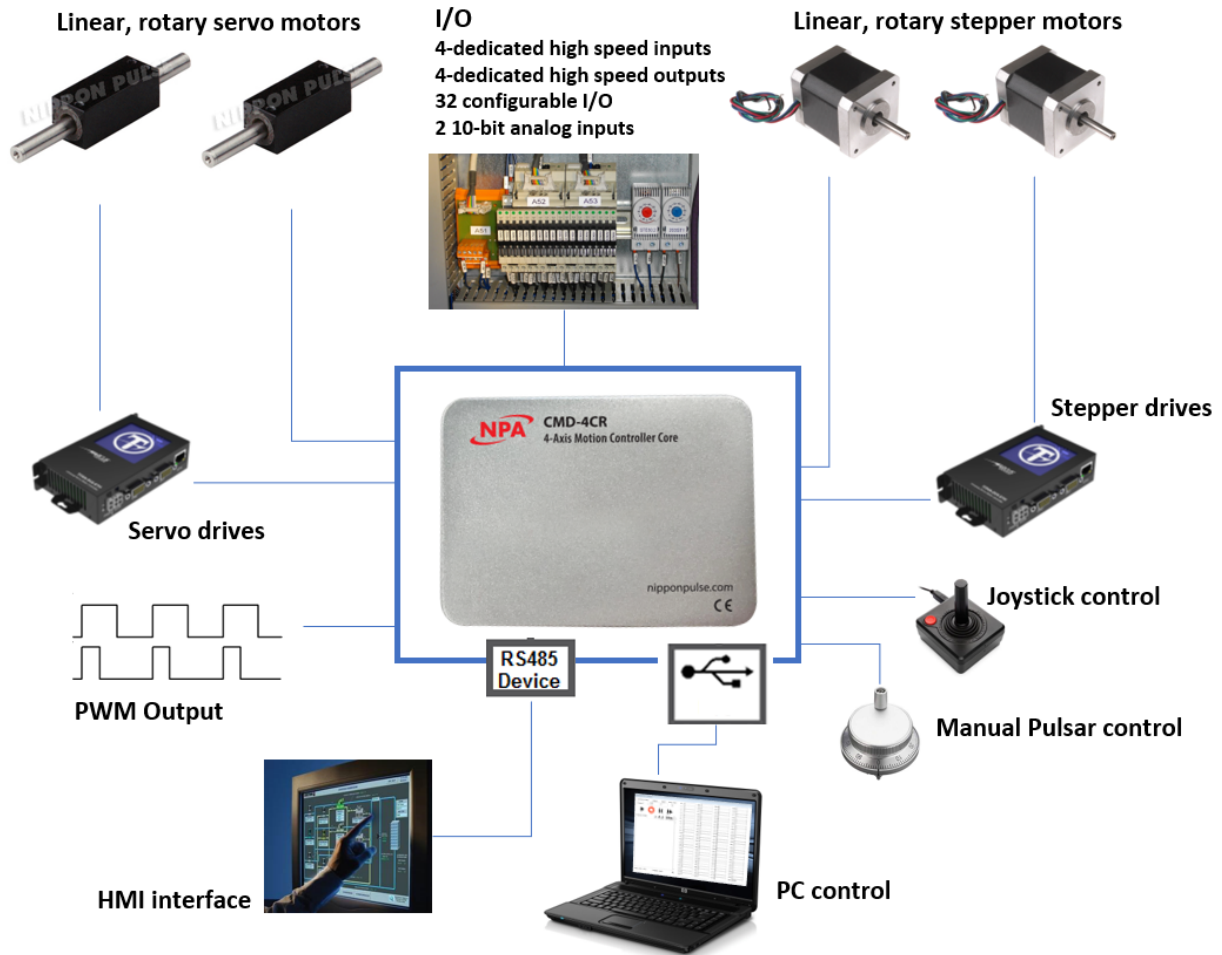
- [General Connections and Setup](#)
 - [Connecting multiple Commander cores in a system](#)
 - [Motor interface](#)
 - [External Connections](#)
 - [Motion Profile](#)
 - [Power-on settings](#)
 - [System settings](#)
 - [Flash Memory](#)
- [Single-axis and Multi-axis operations](#)
 - [Motor Power](#)
 - [Jog operation](#)
 - [Stopping](#)
 - [Positioning operation](#)
 - [Homing operation](#)
- [Interpolation Operation](#)
 - [Linear interpolation](#)
 - [Circular interpolation](#)
 - [Helical or Tangential interpolation](#)
 - [Buffered interpolation operation](#)
- [On-the-fly-speed/target position change](#)
 - [On-the-fly speed change](#)
 - [On-the-fly target position change \(Target position change\)](#)
- [Advanced Functions](#)
 - [Joystick operation \(Analog input control\)](#)
 - [Manual pulse generator \(MPG\) operation](#)
 - [StepNLoop Closed Loop Control operation](#)
- [Status monitoring function](#)

- o [Counter values](#)
- o [Axis status register](#)
- [Standalone program Specification](#)
 - o [Program Development](#)
 - o [Conditional loop control](#)
 - o [Multi-thread](#)
 - o [Subroutines](#)
 - o [Error Handling](#)
 - o [Standalone Variables](#)
 - o [Compiling, Saving, Downloading](#)
 - o [Standalone Control](#)
 - o [Standalone Status](#)
 - o [Standalone Run on Boot-Up](#)

2.1 General Connections and Setup

Connections and Setup

Note: All the commands described in this section are ASCII commands unless called out as standalone commands. ASCII commands are used when communicating over USB or serial communication. Standalone commands are used when writing a standalone program to the Commander core. Not every ASCII command is supported in standalone operations.



In this section:

- [Connecting multiple Commander cores in a system](#)
 - [Identification Number](#)
 - [Synchronization with external trigger](#)
- [Motor interface](#)
 - [Stepper](#)
 - [Digital Servo](#)
 - [Encoder Inputs](#)
- [External Connections](#)
 - [Dedicated Input/Outputs](#)
 - [Mechanical input signals](#)
 - [Limit Switches](#)
 - [Slowdown switches](#)
 - [Home switch](#)
 - [Soft-Limits](#)
 - [Emergency stop](#)
 - [External Start](#)

- [Manual pulse generator \(MPG\) inputs](#)
- [High-speed Inputs/Outputs](#)
 - [High-Speed Digital Inputs](#)
 - [Position Latching Inputs](#)
 - [High-Speed Digital Inputs](#)
 - [High-Speed Digital Output](#)
 - [Synchronization Outputs](#)
 - [High-Speed Digital Outputs](#)
- [General-purpose input/output](#)
 - [Configurable Digital I/O](#)
 - [Analog Input/output](#)
[Joystick Control](#)
 - [Pulse-width modulated Outputs](#)
- [Motion Profile](#)
- [Power-on settings](#)
- [System settings](#)
 - [ERC \(deflection counter clear\)](#)
 - [INP \(in-position\)](#)
 - [Input modes and the polarity of the input/output logic](#)
- [Flash Memory](#)

2.1.1 Connect Multiple Commander Cores in a System

Connect multiple Commander Cores

The Commander core allows for up to 64 devices to be connected to the same host. If multiple devices are connected to the host, the identification number for each device should be unique. It is possible to use an external trigger to allow synchronization across multiple Commander cores, or to start a defined motion profile on one or more Commander cores.

On this page

- [Identification Number](#)
- [Synchronization with external trigger](#)

Identification Number

The Commander core allows for each device to have a unique identification number, allowing up to 64 devices to be connected to the same host. If multiple devices are connected to the host, the identification number for each device should be unique.

By default, all Commander cores are shipped from the factory with identification number 01. The current identification number of a device can be found by reading the **ID** command. Also, the device name will always include the current identification number. For USB communication, the device name will be "CMD-4CR-xx" where xx is the identification number of the device. For serial communication, the "DeviceName" is the identification number of the device.

In order to change the identification number of the device, first store the desired number using the **ID** command. Note that this value must be within the range [00-99].

For example, the command **ID=02** can be sent to change the identification number to 02. To save a modified identification number to the flash memory of the device, use the **STORE** command. The new identification number will not take effect until after a power cycle.

The current version of firmware can be found with the **VER** command.

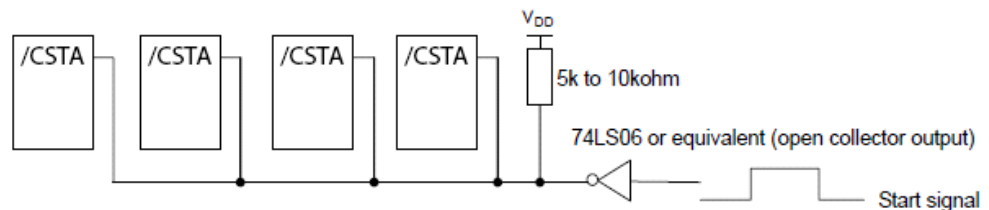
ASCII	ID	STORE	VER
Standalone	-	STORE	-

Synchronization with external trigger

It is possible to use an external trigger to start a defined motion profile. This is done by connecting the **/CSTA** pins as shown in the figure below. The **/CSTA** line needs an external pull-up resistor to VDD.

By default, the external start input is disabled (**EXSTA=0**), which will allow motion to start immediately upon receipt of any motion command.

If the external start input is enabled (**EXSTA=1**), then any motion command sent will wait for the **/CSTA** input to go low before motion is started. The Commander core can buffer up to three movements on a first-in first-out order; they will start one at a time each time the **/CSTA** input goes low.



The logical input of the **/CSTA** input cannot be changed.

ASCII	EXSTA
Standalone	-

JC add section about EXSTP /CSTP

2.1.2 Motor interface

Motor interface

The Commander core is designed to be a master controller for both stepper and digital servomotors and their drivers. There are a few differences between how a stepper and a servo system react and the connections needed between the Commander core and their respective drivers.

The difference between a stepper motor and a servomotor configuration is shown in the following subsections. The design and construction of the motors are also different. Stepper motor operation is synchronized by command pulse signals output from a pulse generator (strictly speaking, it follows the pulses). In contrast, a servomotor will translate the command pulse signals into speed, direction, and distance commands (usually one pulse equals one

encoder pulse); as such the servomotor operation lags behind the command pulses.

In this section:

- [Stepper](#)
- [Digital Servo](#)
- [Encoder Inputs](#)

2.1.2.1 Stepper

Stepper Motor Interface

Since a stepper motor operation is synchronized by command pulse signals output from the Commander core (strictly speaking, it follows the pulses), the stepper driver will need the **pulse** (speed command), **direction** (direction command), and **enable** connected. An encoder connection is not required unless you will be using the **StepNLoop** function;

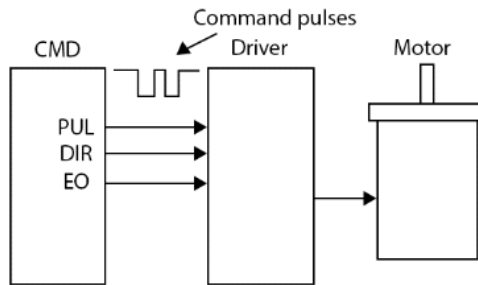
however, there are advantages if an encoder is connected. The **pulse (PUL)** and **direction (DIR)** communicate the speed and direction in real-time to the drive.

The polarity and output mode of these signals is set with the **POL[axis](Bit0~2)**.

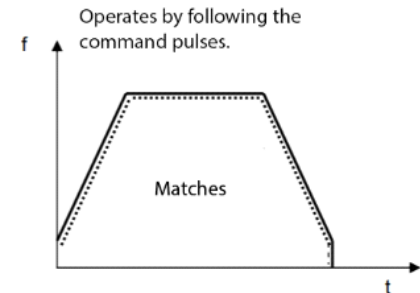
The **enable (EO)** output is most often used to turn the motor excitation circuit ON/OFF. But, if not required by your driver or application, it can be used as a general-purpose output. The polarity of this output is set with the **POL[axis](Bit9)**.

The figure to the right shows connection examples for both open loop and closed loop stepper systems.

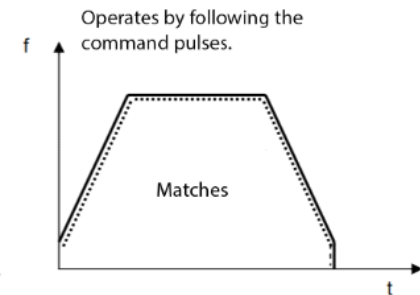
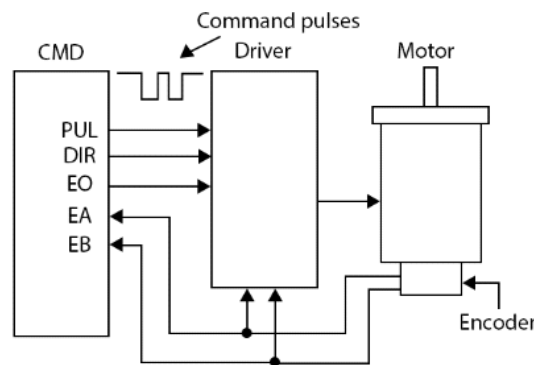
Stepper Motor – Open Loop



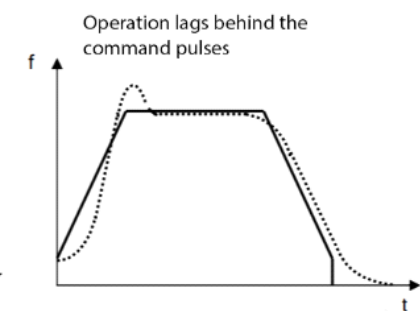
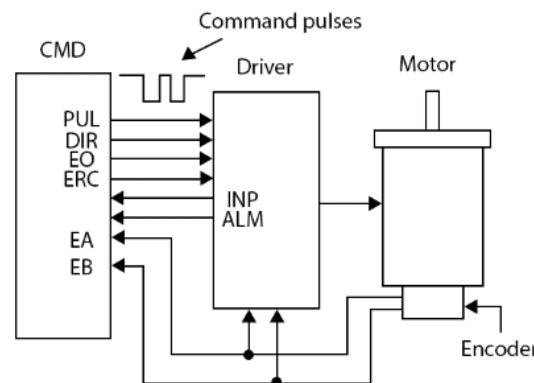
— : Command pulses output from the CMD
 : Actual motor rotation



Stepper Motor – Closed Loop



Servomotor – Closed Loop



ASCII	POL[axis]	EO
Standalone	-	EO

2.1.2.2 Digital Servo

Digital Servo Motor Interface

The servomotor rotation lags behind the command pulses from the Commander core. This means that when the Commander core completes outputting pulses,

the encoder will take some time to return all of the pulses. As such, there is a need for more communication between the Commander core and the servo drive. A servomotor driver will need the

pulse (speed command), **direction** (direction command), **enable**, **INP** (in-position), **ERC** (deflection counter clear) and **ALM** (alarm)

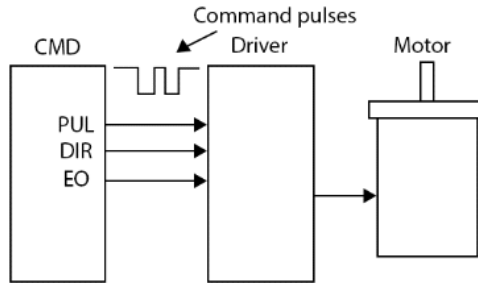
connected. It is also strongly

recommended to connect the encoder output to the Commander core.

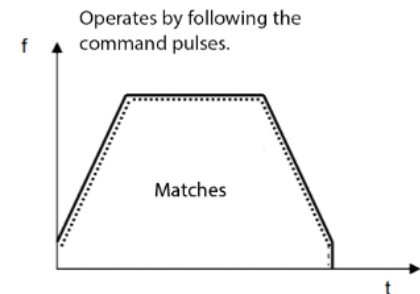
The servomotor driver will send:

An **INP** (in-position) signal is sent from the driver when the motor has reached the requested position based on the pulses sent. The polarity of this input is set with the **POL[axis](Bit7)**. When the **INP[axis]** function is enabled, the operation completion status is delayed until the in-position signal is received from the driver.

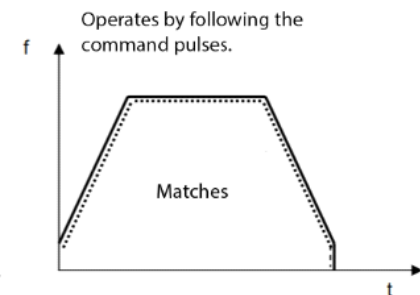
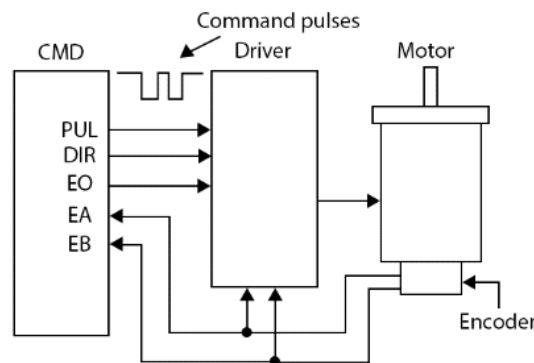
Stepper Motor – Open Loop



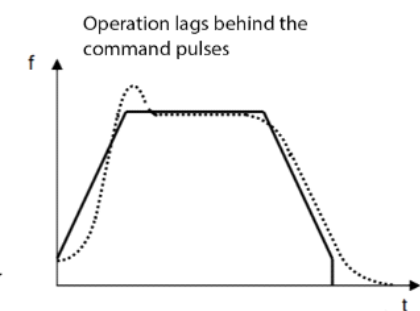
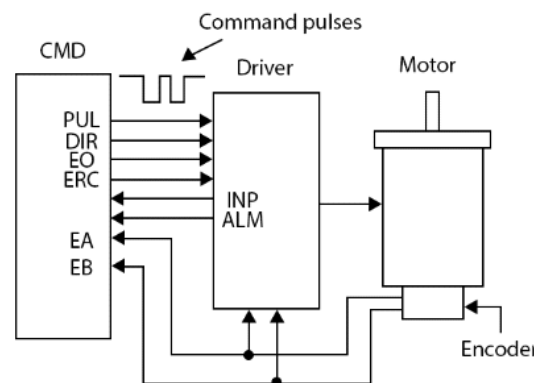
— : Command pulses output from the CMD
 : Actual motor rotation



Stepper Motor – Closed Loop



Servomotor – Closed Loop



An **ALM** (alarm) signal is received when there is an error or abnormality in the servo driver or motor. This can include:

- (1) The deflection amount becomes abnormally large.
- (2) Excess current is flowing through the motor (instantaneous measurement, or at certain intervals)
- (3) There is a temperature error.
- (4) There is a power supply voltage error.

The alarm signal associated with each error varies with the servo driver. Generally, the signal from a servo driver is kept ON until it is cleared. The polarity of this input is set with the **POL[axis](Bit5)**.

As such the **ERC** command can be set up so the Commander core can send an **ERC** (deflection counter clear) signal to clear the error. The polarity of this output is set with the **POL[axis](Bit8)**, while the pulse width is set with the **ERCP** command and delay is set with the **ERCD** command.

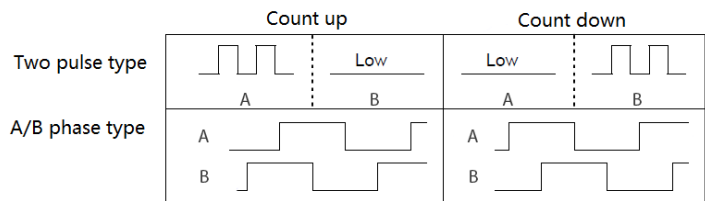
The figure to the right shows connection examples for a servomotor driver.

ASCII	POL[axis]	EO	INP[axis]	ERC[axis]	ERCD[axis]	ERCP[axis]
Standalone	-	EO	-	-	-	-

Encoder Inputs

Encoder Inputs

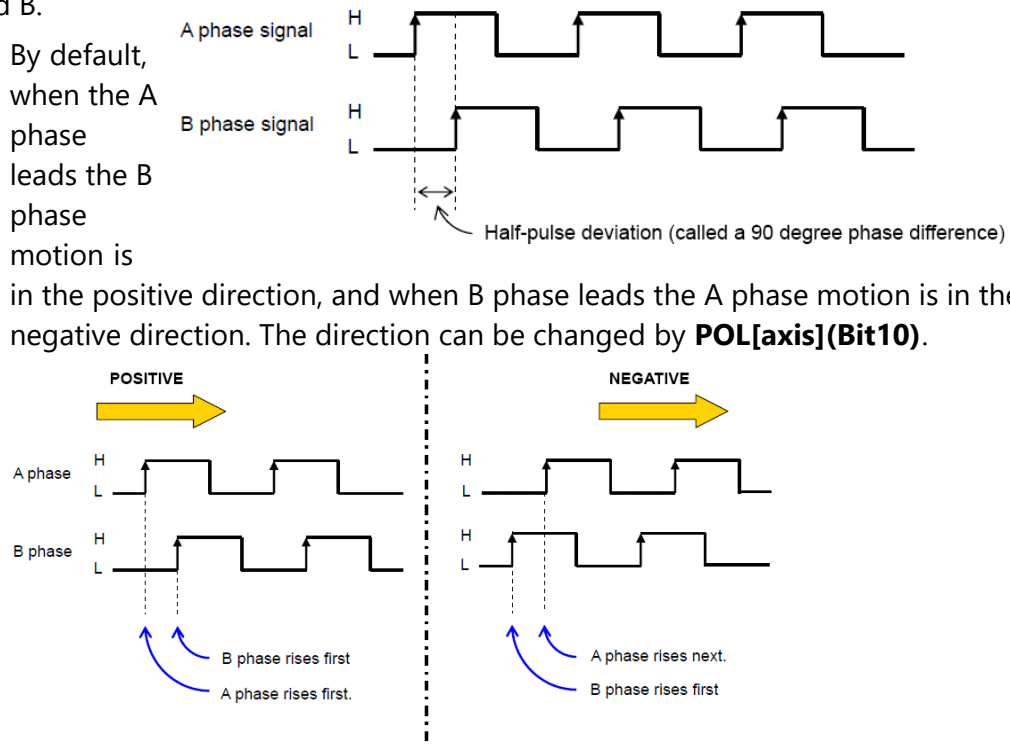
An encoder is a type of pulse generator that can be used to communicate to the Commander core the actual speed, direction, and position of the motor or system it is attached to. The Commander core accepts two different pulse types, which are set using the **POL[axis](Bit11~12)**.



1. Two pulse type (not common) is where pulses are sent to A input to count up and B input to count down. Each pulse input is one count of the encoder.
 - a. This can be used as an up/down counter to track pulses unrelated to the motion of the motor.
2. A/B phase type (common).

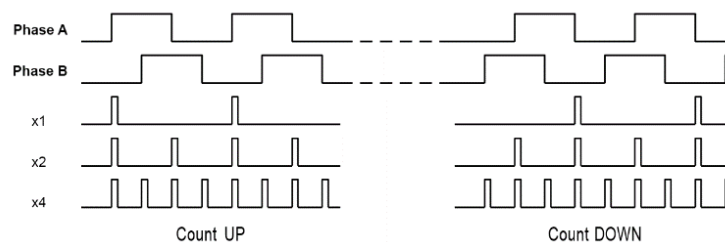
- a. For an A/B phase type, there should be 90° difference between the pulses from A and B.

- b. By default, when the A phase leads the B phase motion is in the positive direction, and when B phase leads the A phase motion is in the negative direction. The direction can be changed by **POL[axis](Bit10)**.



- c. For an A/B phase type you can select x1, x2 or x4 multiplication rate. The figure below shows how the encoder counts would be interpolated for a x1, x2, and x4 multiplication rate on a single encoder.

- i. x1 multiplication – Only the rising edge of the A phase is counted, B phase is not counted and is only used to check the direction of motion.
- ii. x2 multiplication – The rising and falling edge of the A phase is counted, B phase is not counted and is only used to check the direction of motion.
- iii. x4 multiplication – The rising and falling edge of the A and B phases are counted, and used to check the direction of motion.



Maximum frequency input on the encoder line is the maximum number of pulses that can be input to an encoder input, not the counts. Since most encoders' outputs are rated based on a x4 multiplication rate, the output frequency of pulses is most likely ¼ the published count rate output of the encoder. Please check the data sheet for your encoder to confirm.

While an encoder input is not required, there are a number of advantages to connecting one.

- 1) Closed loop control using StepNLoop – StepNLoop is a closed-loop position verification algorithm. When the encoder is placed at the work point, it can be used to eliminate backlash in the system. When the encoder is mounted to the motor, it can help overcome missed steps. Section [2.5.3 StepNLoop Closed Loop Control operation](#)

contains more information regarding StepNLoop.

- 2) Motor Stall detect – When an encoder is connected it is easy to detect if the motor has stalled.
- 3) Position verification – Verify the motor has finished its move and reached the desired position.
- 4) Z-index – the Z-index can improve your homing method. The polarity of Z-index input is set with the **POL[axis](Bit13)**

If an encoder is not connected or needed, the encoder inputs can be used as an up/down counter to track pulses unrelated to the motion of the motor.

ASCII	POL[axis]
Standalone	$\frac{3}{4}$

2.1.3 External Connections

2.1.3 External Connections

The Commander core includes several types of inputs/and outputs which are not directly related to the motion of the motor. Digital I/O is broken down into three categories: 1) **Dedicated I/O** - high speed inputs or outputs that have a dedicated function, which cannot be changed; 2) **high speed I/O** - high speed inputs or outputs with processing times of <200ns that have certain functions associated to them, but can also be used as general purpose inputs or outputs; and 3) **general purpose I/O**, which have no built-in associated functions and are defined by the user.

There are also two analog inputs and two PWM outputs.

In this section

- [Dedicated Input/Outputs](#)
 - o [Mechanical input signals](#)
 - [Limit Switches](#)
 - [Slowdown switches](#)
 - [Home switch](#)
 - [Soft-Limits](#)
 - o [Emergency stop](#)
 - o [External Start](#)
 - o [Manual pulse generator \(MPG\) inputs](#)
- [High-speed Inputs/Outputs](#)
 - o [High-Speed Digital Inputs](#)
 - [Position Latching Inputs](#)
 - [High-Speed Digital Inputs](#)
 - o [High-Speed Digital Output](#)

- [Synchronization Outputs](#)
- [High-Speed Digital Outputs](#)
- [General-purpose input/output](#)
 - o [Configurable Digital I/O](#)
 - o [Analog Input/output](#)
 - [Joystick Control](#)
 - o [Pulse-width modulated Outputs](#)

2.1.3.1 Dedicated Input/Outputs

Dedicated Input/Outputs

The Commander core includes a number of inputs and outputs that have dedicated functions. These include safety-related functions such as **emergency stop** and **limit switches**, informational inputs such as **slowdown** and **home**, and functional inputs such as **external start** and **(MPG) pulsar inputs**.

In this section:

- [Mechanical input signals](#)
 - o [Limit Switches](#)
 - o [Slowdown switches](#)
 - o [Home switch](#)
 - o [Soft-Limits](#)
- [Emergency stop](#)
- [External Start](#)
- [Manual pulse generator \(MPG\) inputs](#)

2.1.3.1.1 Mechanical input signals

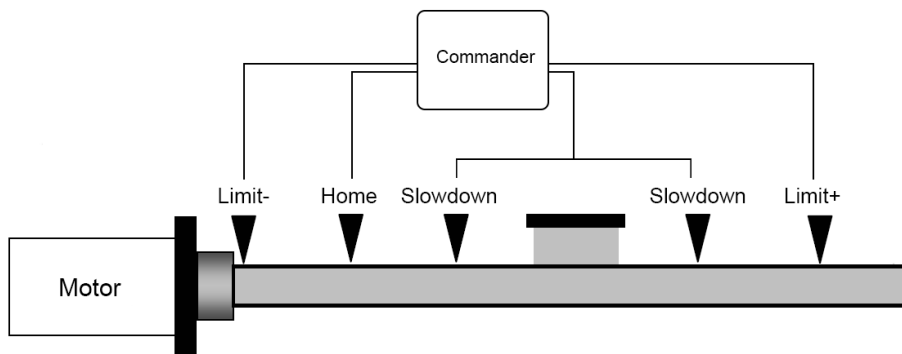
Mechanical input signals

The following four signals can be input for each axis:

- 1) +L (Limit +) When this signal turns ON while operating in the positive direction, the motor stops immediately, or decelerates and stops.
- 2) -L (Limit -) When this signal turns ON while operating in the negative direction, the motor stops immediately, or decelerates and stops.
- 3) SD (Slowdown) Used as a deceleration signal or a deceleration stop signal (set by software).
- 4) H (Home) Input signal for a Homing operation.

The input logic for these signals can be changed with software.

The Commander also has a soft-limit function. The soft-limits can be used in a system where physical limits are impossible, or for setting operation limits within the allowable mechanical stroke limits.



In this section:

- [Limit Switches](#)
- [Slowdown switches](#)
- [Home switch](#)
- [Soft-Limits](#)

2.1.3.1.1.1 Limit Switches

Limit switches

Limit switches, most commonly seen in linear mechanisms to restrict motion to a safe range, are used to identify that the motor has reached its end of travel. If either the positive (**L+**) or negative (**L-**) signal becomes active, the Commander core is instructed to immediately stop all advancement in that direction but will allow movement in the opposite direction. As such it is important to ensure the placement of the positive limit at the end of positive travel and negative limit at the end of negative travel. The polarity of limit inputs is set with the **POL[axis] (Bit3)**.

When designing your limit circuit, here are a few points to keep in mind:

- 1) It is especially important that the motor must stop when the signal is active.
- 2) To prevent nuisance limit faults that can stop your system, ensure that electrical noise does not trigger your limits as active.
- 3) To operate in a fail-safe manner, the signal line must be designed to trigger as active when the cable connecting the limit switch to the Commander core becomes disconnected.

With the above points in mind think about your limit circuit and the active logic level in your system.

Limit error

By default **IERR=0**, which will trigger a limit error **MST[axis](bit8~9)** when a limit switch is set to active. Once the limit error is set, use the **CLR[axis]** command to clear the error in ASCII mode or the **ECLEAR[axis]** command in standalone mode.

When using the end limit during the homing modes, the **ERC** (deflection counter clear) signal can be setup to clear the limit errors that accrue due to homing automatically.

The limit error states can be ignored by setting **IERR=1**. In this case, the motor will still stop when the appropriate switch is triggered; however, it will not enter an error state.

ASCII	CLR	IERR	ERC[axis]	MST[axis]
Standalone	ECLEAR[axis]	¾	¾	MST[axis]

2.1.3.1.1.2 Slowdown switches

Slowdown Switches

The slowdown switch (**SD**) input indicates a point to begin deceleration. If the slowdown switch inputs are active when the motor is operating at HSPD speed and this signal is triggered, the motor will start to decelerate.

The 2-bit **SDE[axis]** command enables the SD inputs and sets the latching function. By default, **SDEX=0** disables the X-axis SD inputs. The inputs can be enabled without a latch function using **SDEX=1**, or enabled with a latch function **SDEX=3**.

Bit	Description	Setting	
0	SD input	0 - Disabled	1 - Enabled
1	SD latching	0 - Disabled	1 - Enabled

Table 2-1 SDE command

The **SDC[axis]** command determines the specific operation of the SD input. By default, **SDC[axis]=0** sets the operation, when triggered, to decelerate the motor to LSPD without stopping. When **SDC[axis]=1**, the operation is set to decelerate and stop when the SD input is triggered.

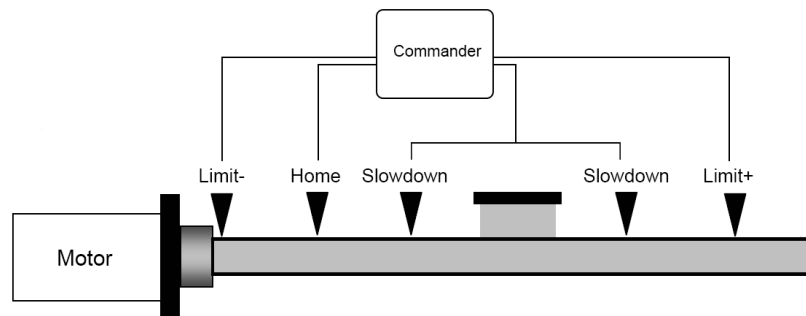
Value	Description
0	Decelerate only
1	Decelerate and stop

Table 2-2 SDC command

Triggering the SD input when **SDC=1** will set the SD stop flag, **MST[axis](bit17)**. The SD stop flag will not prevent further motion, but can only be cleared by using the **CLR[axis]** command in ASCII mode or the **ECLEAR[axis]** command in standalone mode.

The SD input, when triggered, will cause the motor to slowdown regardless of the direction of travel.

The figure below shows an example of a setup using the slowdown inputs. These inputs are mainly used in homing operations. In this case the axis would move at HSPD, trigger the slowdown input, allowing the motor to decelerate to LSPD, before it triggers the home input.



Other uses for the slowdown input are placing SD limits at each end of travel just before the end limits. This allows for two options. The slowdown inputs function as a soft limit (**SDC=1**) that decelerates and stops the motion, allowing the end limits to operate as emergency hard end limits. Or they can just allow the motor to decelerate (**SDC=0**) to LSPD before triggering the end limits. The polarity of the slowdown inputs is set with the **POL[axis](Bit6)**.

ASCII	SDE[axis]	SDC[axis]	POL[axis]	CLR	MST[axis]
Standalone	3/4	3/4	3/4	ECLEAR[axis]	MST[axis]

2.1.3.1.1.3 Home switch

Home Switch

The Home (**H**) switch is used during some homing modes to aid in the detection and location of the zero-position used in **absolute** positioning operations. For more information see homing operation in section [2.2.5 Homing operation](#) and [Appendix A: Homing Mode Actions](#). The polarity of the home input is set with the **POL[axis](Bit4)**.

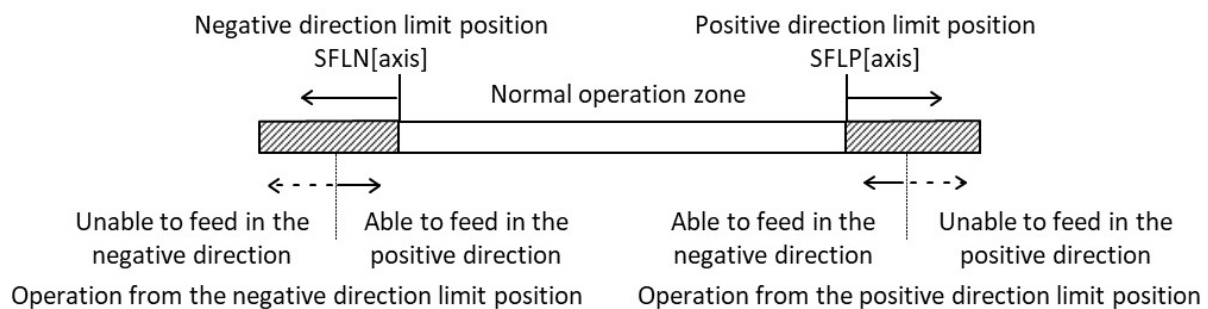
2.1.3.1.1.4 Soft-Limits

Soft-Limits

The soft-limit function uses a comparator for a positive direction limit (**SFLP[axis]**) and a second comparator for a negative direction limit (**SFLN[axis]**) to stop the axis based on the axis position related to the comparator's results and the operation direction. The **SFLSO[axis]** command selects the encoder counter (**1**) or pulse generator counter (**0**) for the soft-limit axis position. It is safest to use the encoder counter if an encoder is available, as it will allow the tracking of the motor even if an outside force moves the motor.

When the soft-limit function is enabled, the axis stops immediately when the motor reaches the soft-limit setting at the low-speed setting (LSPD). At the high-speed setting (HSPD), then stopping is executed based on the setting of the **SFLSD[axis]**.

While deceleration stop is selected as the process when soft-limit triggered (**SFLSD[axis]=1**), when the axis reaches the soft-limit while at high-speed, that axis stops using deceleration at the rate set by **DEC**. When some other process is specified (**SFLSD[axis]=0**), or while in a constant (low) speed, that axis stops immediately. While the axis is at or beyond the soft-limit setting, the axis cannot be moved in the direction in which the soft-limit is enabled. However, it can move in the opposite direction.



Once the parameters are set, the soft-limit function can be enabled by setting **SFL[axis]=1** and disabled by setting **SFL[axis]=0**.

ASCII	SFL[axis]	SFLP[axis]	SFLN[axis]	SFLSO[axis]	SFLSD[axis]
Standalone	—	—	—	—	—

2.1.3.1.2 Emergency stop

Emergency stop

The Commander core has an **emergency stop (/CEMG)** input for dedicated use as an emergency stop signal. When in operation, if the **/CEMG** input goes low or the **ESTOP** command is sent, all axes will stop immediately. No axis can operate while the **/CEMG** signal is low.

The logical input of the **/CEMG** input cannot be changed.

Note: In a normal stop operation, the final pulse width is normal. However, in an emergency stop operation, the final pulse width may not be normal. As such the motor drivers may not recognize the last pulses, and therefore the Commander core internal counter may not match the mechanical position. Therefore, after an emergency stop, it is recommended that you perform a homing operation to match the command and mechanical positions.

Emergency stop error

No axis can operate while the **/CEMG** signal is low.

By default when **IERR=0**, any axis that is operating when the **/CEMG** signal goes low or the **ESTOP** command is sent will set an **EMG** error flag **MST[axis](bit16)** for that axis. In addition, any axis that tries to start operating while the **/CEMG** signal is low will set an **EMG** error flag for that axis. Any axis with a set **EMG** error flag cannot operate until the **EMG** error flag has been cleared. However, if an axis was not moving when the **/CEMG** or **ESTOP** was triggered, the **EMG** error flag will not be set for that axis, and thus that axis can operate immediately after the **/CEMG** input is cleared.

To clear the **EMG** error flag, use the **CLR[axis]** command in ASCII mode or the **ECLEAR[axis]** command in standalone mode.

The **EMG** error flags are ignored by setting **IERR=1**. In this case, all axes will still stop when **/CEMG** is triggered or **ESTOP** command is sent, and all axes can operate again immediately after the **/CEMG** input is cleared.

ASCII	ESTOP	MST[axis]	CLR[axis]	IERR
Standalone	—	MST[axis]	ECLEAR[axis]	—

2.1.3.1.3 External Stop

Eternal Stop

The **external stop (/CSTP)** input allows motion to be stopped by taking the **/CSTP** input low.

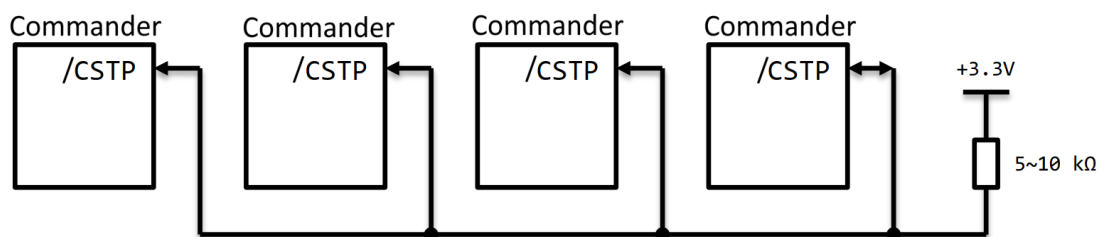
The **EXSTP[axis]** command determines the specific stop operation performed when of the **/CSTP** input is triggered. By default, the external stop input is disabled (**EXSTP[axis]=0**), which will make it so that the **/CSTP** input will have no effect on the motion for that axis. When **EXSTP[axis]=1**, the operation is set to decelerate and stop when the **/CSTP** input is triggered. When **EXSTP[axis]=2**, the operation is set to stop without deceleration when the **/CSTP** input is triggered.

Value	Description
0	Disabled
1	Decelerate and stop
2	Stop immediately

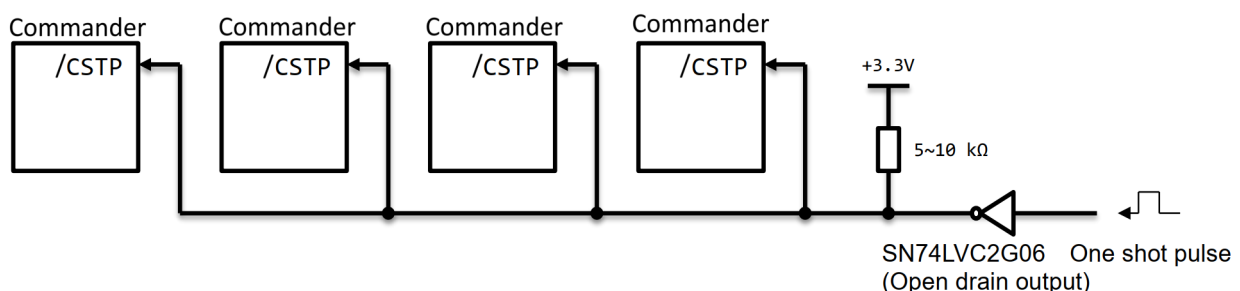
Table 2-3 EXSTP command

There are two ways to trigger **external stop (/CSTP)**:

1. The **CSTP=1** command enables a one-shot pulse with the pulse width of 0.4 μ s to be output from the **/CSTP** pin when the **STOP** command is sent. This one-shot pulse will be input to all Commanders connected to the **/CSTP** pins, which will stop all the axis with **EXSTP[axis]=1 or 2**. The one-shot pulse will be re-input to the Commander that has output the one-shot pulse and will stop all the axis with **EXSTP[axis]=1 or 2** on the sending Commander also.



2. Input a one-shot pulse with the pulse width of 0.2 μ s or more from the outside to **/CSTP** pin. All Commanders connected by the **/CSTP** pins input a one-shot pulse to stop the axis with **EXSTP[axis]=1 or 2**.



The logical input of the **/CSTP** input cannot be changed. **MST[axis].bit20** returns the current state of the **/CSTP** input. When **EXSTP[axis]** is set to **0** for all axis, then **/CSTP** can be used as a general purpose input.

The difference between **/CSTP** and **/CEMG** is that, **/CSTP** is treated as a normal stop command, the same as sending the **STOP** command while **/CEMG** is used to perform an emergency stop on all axes. **/CEMG** will cancel all other commands and motions.

ASCII	MST[axis]	EXSTP[axis]	CSTP	STOP
Standalone	MST[axis]	$\frac{3}{4}$	$\frac{3}{4}$	STOP

2.1.3.1.4 External Start

External Start

The **external start (/CSTA)** input allows motion to be started by taking the **/CSTA** input low. By default, the external start input is disabled (**EXSTA=0**), which will allow motion to start immediately upon receipt of any motion command. If the external start input is enabled (**EXSTA=1**), then any motion command sent will wait for the **/CSTA** input to go low before motion is started. The Commander core can buffer up to three motions, on a first-in first-out order. They will start one at a time each time the **/CSTA** input goes low.

The logical input of the **/CSTA** input cannot be changed. **MST[axis].bit14** returns the current state of the **/CSTA** input.

ASCII	MST[axis]	EXSTA
Standalone	MST[axis]	³ / ₄

2.1.3.1.5 Manual pulse generator (MPG) inputs

Manual Pulse Generator (MPG) Inputs

An MPG is similar to an encoder in that it is a type of pulse generator that can be used to communicate a speed and direction to the Commander core. The difference between the two is that an MPG commands motion, while an encoder reports the motion.

The MPG input can be used for a number of functions such as:

- 1) A manual pulse generator, which allows for fine adjustment of an axis.
- 2) Cam follower, which will allow the axis to follow a cam axis based on its encoder output.
- 3) Electronic gearing, which allows an axis to follow the input commands but at a different rate of speed (faster or slower).
- 4) Encoder Frequency and Impulse Multiplier/Divider

More information on MPG operation can be found in Section [2.5.2 Manual pulse generator \(MPG\) operation](#).

2.1.3.2 High-speed Inputs/Outputs

High-speed Inputs/Outputs

The Commander core has a number of high-speed inputs and outputs that have dedicated functions. These include **position latching inputs** and **synchronization outputs**. If these functions are not needed for your application, they can also be used as **general-purpose inputs or outputs**.

In this section:

- [High-Speed Digital Inputs](#)
 - o [Position Latching Inputs](#)
 - o [High-Speed Digital Inputs](#)
- [High-Speed Digital Output](#)
 - o [Synchronization Outputs](#)
 - o [High-Speed Digital Outputs](#)

2.1.3.2.1 High-Speed Digital Inputs

High-speed Inputs/Outputs

The Commander core provides four (4) high-speed inputs. Each input can independently be set as a **position latching input** or **general-purpose input**. By default, these inputs are set to general-purpose input.

Position Latching Inputs

Any of the four high-speed inputs can be set independently to be position latch inputs by enabling the feature using the **LT[axis]** command for the corresponding axis. This feature will allow the controller to perform a high-speed position capture of both pulse and encoder counters based on a digital input trigger. The timing between digital input trigger being received and the capture of the pulse and encoder position is 150 to 200ns.

Each axis has a designated digital input to perform a high-speed position latch. See corresponding latch input for each axis in the table:

Axis	Input
X	DI1/LTCx
Y	DI2/LTCy
Z	DI3/LTCz
U	DI4/LTCu

Latching inputs

Use the **LT[axis]** command to enable and disable the latch feature. To read the latch status, use the latch status **LTS[axis]** command. Table LTS command return below describes the possible return values of the latch status command.

Value	Condition
0	Latch off
1	Latch on and waiting for latch trigger
2	Latch triggered

LTS command return

Once the latch is triggered, the triggered pulse position can be retrieved using **LTP[axis]** command. Similarly, the triggered encoder position can be retrieved using the **LTE[axis]** command.

When StepNLoop mode is enabled, the position value is invalid.

ASCII	LT[axis]	LTS[axis]	LTP[axis]	LTE[axis]
Standalone	LT[axis]	LTS[axis]	LTP[axis]	LTE[axis]

High-Speed Digital Inputs

Any of the four high-speed inputs can be set independently to be general-purpose inputs by disabling the position latch feature using the **LT[axis]** command for the corresponding axis. These inputs are set to general-purpose by default.

The digital input status of all four available inputs can be read with the **DI** command. Digital input values can also be referenced one bit at a time by using the **DI[1-4]** commands. Note that the indexes are 1-based for the bit references. For example, DI1 refers to bit 0, not bit 1. See table DI command below for details.

Bit	Input	Bit-Wise Command
0	Digital Input 1	DI1
1	Digital Input 2	DI2
2	Digital Input 3	DI3
3	Digital Input 4	DI4

DI command

If a digital input is on, the corresponding bit of the **DI** command is 1. Otherwise, the bit status is 0. The voltage level required to activate a digital input is determined by the polarity setting. The **DIP** command can be used to toggle the polarity of the digital inputs.

ASCII	DI	DI[1-4]	DIP	LT[axis]
Standalone	DI	DI[1-4]	³ / ₄	LT[axis]

2.1.3.2.2 High-Speed Digital Output

High-Speed Digital Output

The Commander core provides four (4) high-speed outputs. Each output can independently be set as a **synchronization output** or a **general-purpose output**. By default, the outputs are set to general-purpose outputs.

Synchronization Outputs

The Commander core provides four high-speed synchronization outputs. This feature allows the controller to perform a high-speed comparison between the encoder or pulse **position counter** and a **set condition (comparator)**, which is set with the **SYNP[axis]** command. When the condition is met, the corresponding synchronization output is enabled. The timing between the condition being met and the synchronization output being triggered is 150 to 200ns.

There are three modes for the synchronization outputs to be enabled, which can be set with the **SYNC[axis]** command:

1. **At position mode** (counter = comparator)
2. **> or < position mode** (counter is either > or < than the comparator)
3. **Continuous mode** (every comparator position)

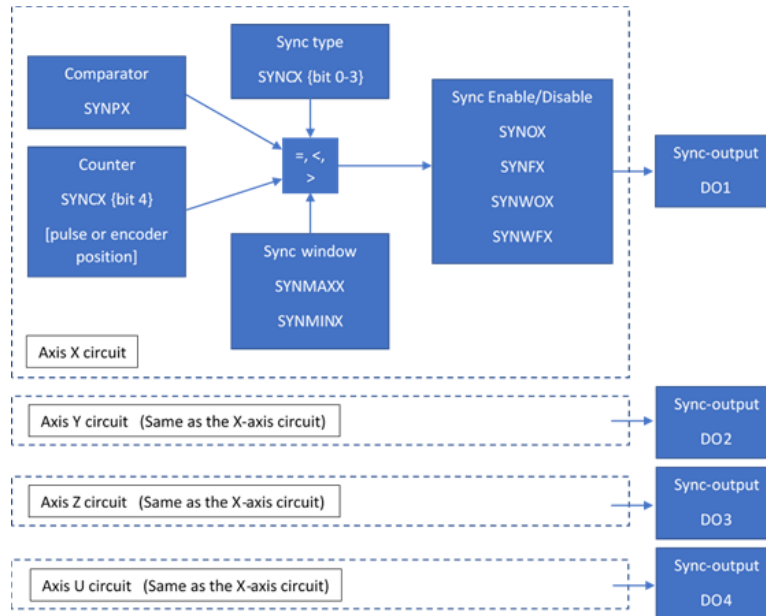
For **continuous mode**, and **at position mode**, the output can be output in both directions, or limited to only one direction (positive or negative) of movement.

For **> or < position mode**, you can select only Counter > Comparator or Counter < Comparator.

Bit	Description		Decimal	
			Pulse Position	Encoder Position
0-3	Synchronization Condition:		From sync source:	
	0001	Trigger when sync position = actual position. Any direction	1	17
	0010	Trigger when sync position = actual position. Positive motion	2	18
	0011	Trigger when sync position = actual position. Negative motion	3	19
	0100	Trigger when sync position > actual position	4	20
	0101	Trigger when sync position < actual position	5	21
	0111	Reserved/not available		
	1000	Trigger every sync position. Any direction	8	24
	1001	Trigger every sync position. Positive motion	9	25
	1010	Trigger every sync position. Negative motion	10	26

4	Synchronization Source:			
	0	Sync position will be compared to pulse position		
	1	Sync position will be compared to encoder position		

Synchronization mode and source (SYNC)



Block Diagram of synchronization function

Each axis has a designated high-speed synchronization output. While synchronization output is enabled for an axis, the corresponding digital output cannot be manually controlled by the user. See corresponding synchronization output for each axis in table synchronization output below.

Axis	Output
X	DO1/SYN Cx
Y	DO2/SYN Cy
Z	DO3/SYN Cz
U	DO4/SYN Cu

Synchronization output

Use the **SYNO[axis]** command to enable and the **SYNF[axis]** command to disable the synchronization feature. To read the synchronization status, use the synchronization status **SYNS[axis]** command (2-bit).

Bit	Condition	Setting	
0	Synchronization function	0 -	1 -

		Disabled	Enabled
1	Synchronization Window	0 - Disabled	1 - Enabled

SYNS command return

In **continuous mode**, there is an option for a synchronization window function. When enabled with the **SYNWO[axis]** command, the synchronization function will run continuously, but only outputs a pulse when the counter is between the synchronization maximum **SYNMAX[axis]** and minimum **SYNMIN[axis]** points. There is no loss of synchronization when outside the synchronization window while continuous mode is enabled. To disable the window function, use the **SYNWF[axis]** command; synchronization will not be lost. To simultaneously disable the window function and continuous sync mode, use the **SYNF[axis]** command.

The **DOP** can be used to toggle the polarity of the digital outputs.

ASCII	SYNO[axis]	SYNWO[axis]	SYNF[axis]	SYNWF[axis]	SYNS[axis]
Standalone	SYNON[axis]	$\frac{3}{4}$	SYNOFF[axis]	$\frac{3}{4}$	SYNSTAT[axis]

ASCII	SYNP[axis]	SYNC[axis]	SYNMAX[axis]	SYNMIN[axis]	DOP
Standalone	SYNPOS[axis]	SYNCFG[axis]	$\frac{3}{4}$	$\frac{3}{4}$	$\frac{3}{4}$

High-Speed Digital Outputs

Any of the four high-speed outputs can be set independently to be general-purpose outputs by disabling the synchronization feature using the **SYNF[axis]** command for the corresponding axis. These outputs are set to general-purpose by default.

The 4-bit **DO** command can be used to set all the digital outputs at once. The **DO** value is a decimal number and must be within the range of 0-15.

Digital outputs can also be set one at a time with the **DO[1-4]** commands. Note that the indexes are 1-based for the bit references. For example, DO1 refers to bit 0, not bit 1. See table DO command below for details.

Bit	Output	Bit-Wise Command
0	Digital Output 1	DO1
1	Digital Output 2	DO2
2	Digital Output 3	DO3
3	Digital Output 4	DO4

Table DO command

If a digital output is turned on, the corresponding bit of the **DO** command is 1. Otherwise, the bit status is 0. The voltage level of the digital output when it is on or off is determined by the polarity setting. The **DOP** command can be used to toggle the polarity of the digital outputs. The initial state of the digital outputs can be defined by setting the 4-bit **DOBOOT** register to

the desired initial digital output value. The **DOBOOT** value must be within the range of 0-15. The value is stored to flash memory once the **STORE** command is issued.

Bit	Output	Setting
0	Digital Output 1	0 - OFF 1 - ON
1	Digital Output 2	0 - OFF 1 - ON
2	Digital Output 3	0 - OFF 1 - ON
3	Digital Output 4	0 - OFF 1 - ON

Table DOBOOT command

ASCII	DO	DO[1-4]	DOP	DOBOOT	SYNF[axis]	STORE
Standalone	DO	DO[1-4]	$\frac{3}{4}$	$\frac{3}{4}$	SYNOFF[axis]	STORE

2.1.3.3 General-purpose input/output

General-purpose input/output

The Commander core has a number of general-purpose inputs and outputs: **configurable digital I/O**, **analog inputs** and **PWM outputs**. There is a built-in function that will allow a **joystick** connected to the analog input to be used for control of the X and Y axes.

- [Configurable Digital I/O](#)
- [Analog Input/output](#)
[Joystick Control](#)
- [Pulse-width modulated Outputs](#)

Configurable Digital I/O

The Commander core has 32 general-purpose configurable digital I/O available. The **IO** command can be used to access the current configurable I/O status. The value must be a 32-bit integer.

Configurable digital I/O can also be accessed one bit at a time using the **IO[1-32]** commands. Note that the indexes are 1-based for the bit references. For example, IO1 refers to bit 0, not bit 1.

In order to set a configurable digital I/O as an input or output, use the **IOCFG** command. If a bit in the 32-bit **IOCFG** value is set to 0, the corresponding configurable digital I/O will be defined as an input. If the bit is 1, the configurable digital I/O will be defined as an output.

The initial state of the configurable digital I/O can be defined by setting the 32-bit **IOBOOT** register to the desired initial configured output value. The **IOBOOT** value will only be applicable to configurable I/O defined as outputs by the **IOCFG** command. The **IOBOOT** value

must be a 32-bit value. The value is stored to flash memory once the **STORE** command is issued. The settings take effect on the next power cycle.

The 2-bit **IOP** command can be used to toggle the polarity of the configurable I/O. See table IOP command below.

Bit	Description	Setting	
0	Logic of input setting	0 - Negative logic Active Low	1 - Positive logic Active High
1	Logic of output setting	0 - Negative logic Active Low	1 - Positive logic Active High

IOP command

ASCII	IO	IO[1-32]	IOBOOT	IOCFG	IOP	STORE
Standalone	IO	IO[1-32]	$\frac{3}{4}$	$\frac{3}{4}$	$\frac{3}{4}$	STORE

Analog Input/output

The Commander core has two 10-bit analog inputs available. The **AI[1-2]** command can be used to read the current analog input value. The return value is in millivolts and can range from 0 to 3300.

The voltage supplied to the analog inputs should stay within the 0V to 3.3V range.

ASCII	AI[1-2]
Standalone	AI[1-2]

Joystick Control

Joystick control is available on the Commander core for the X and Y axis. When this mode is enabled, the pulse speed and direction outputs for the X and Y axes can be controlled by the corresponding analog input. See the Section [2.5.1 Joystick operation \(Analog input control\)](#) for more information.

Pulse-width modulated Outputs

The Commander core has two pulse-width modulated (PWM) outputs available. The PWM outputs are 500Hz and can range from a 1% to 99% on-time duty cycle. The **PWM[1-2]** command is used to set the value, listed as a percentage, and can range from 1 to 99.

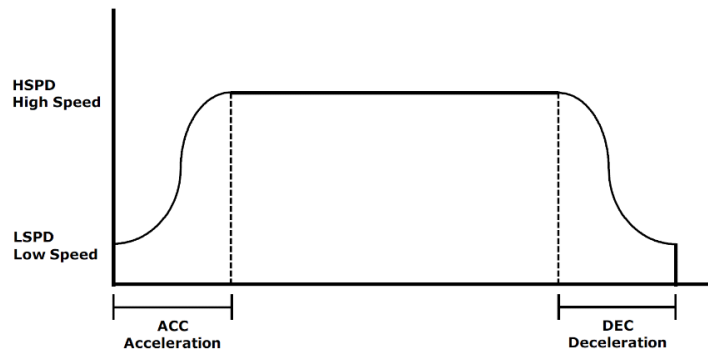
ASCII	PWM[1-2]
Standalone	PWM[1-2]

2.1.4 Motion Profile

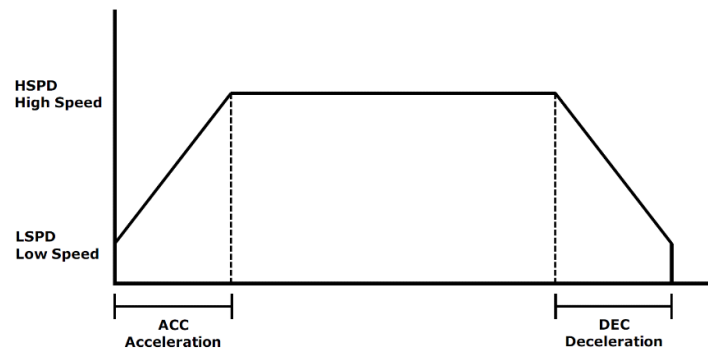
Motion Profile

Any motion profile can be plotted as a trapezoid, where time is the base, Velocity is the height, acceleration rate is the angle, and distance is the area.

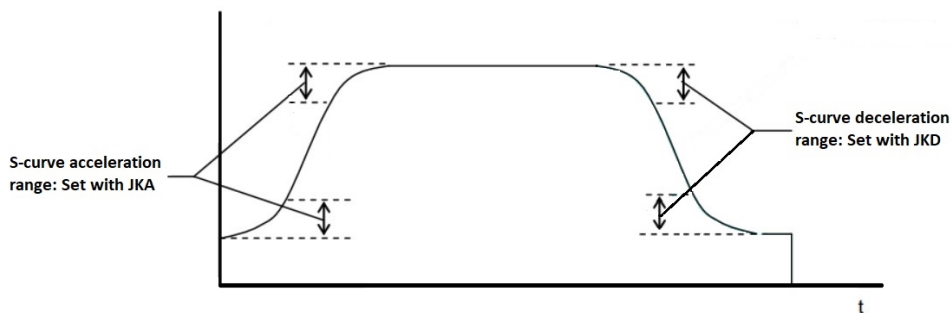
The Commander core uses an S-curve acceleration profile by default, as shown in the figure below. The S-curve acceleration profile reduces jerk, but it requires more power from the motor than a linear acceleration profile when the time for acceleration is the same. Setting **SCV[axis]** to 1 enables an S-curve for the indicated axis.



To reduce the power required from the motor, a linear acceleration profile, as shown in the figure below, is possible. Linear acceleration is used when turning S-curve off by setting **SCV[axis]** to 0.



A balance between the amount of jerk and power required from the motor is achieved by combining the S-curve acceleration with linear acceleration in the center. The **JKA[axis]** command for acceleration and the **JKD[axis]** command for deceleration set the ratio between S-Curve and linear acceleration. They range from 0 to 100, which is the percentage of the S-curve the profile contains. A setting of 0 is the same as a completely linear profile, and a setting of 100 is the same as a complete S-curve profile.



The axis starts moving at the low-speed setting and accelerates to the high-speed setting for a typical move. Once at high-speed, the motor moves at a constant speed until it decelerates from high to low speed and immediately stops.

Both high and low speeds are in pps (pulses/second). Use the commands **HSPD[axis]** and **LSPD[axis]** to set/get individual high-speed and low-speed settings in ASCII and standalone mode. To access the global high-speed and low-speed values, use the ASCII/standalone commands **HSPD** and **LSPD**.

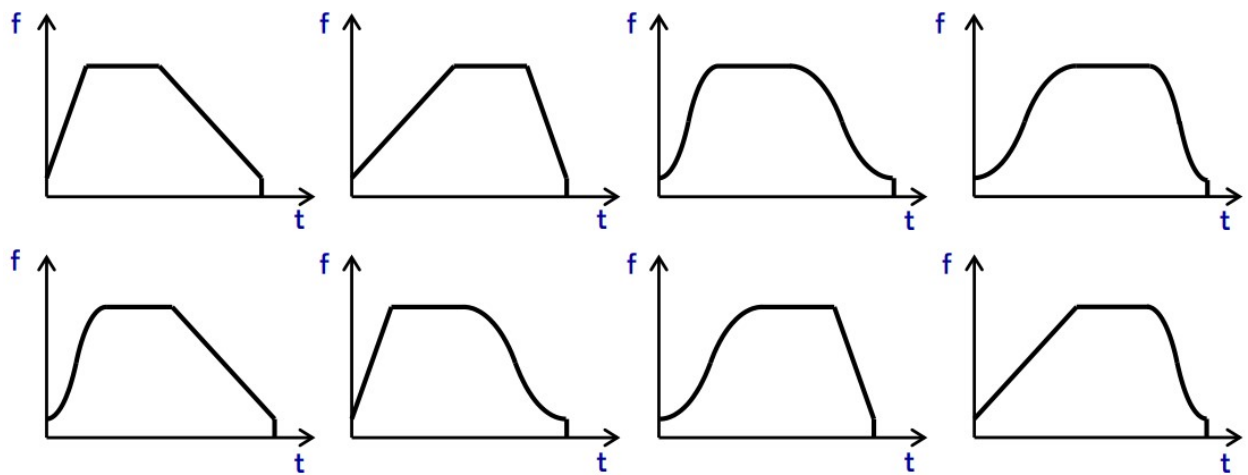
Acceleration and deceleration times are in milliseconds. Use the **ACC[axis]/DEC[axis]**

command to set/get individual acceleration/deceleration values. Use the corresponding ACC or DEC command to access the global acceleration/deceleration value. The following formula can calculate the acceleration/deceleration rate in pps² (pulses/second²).

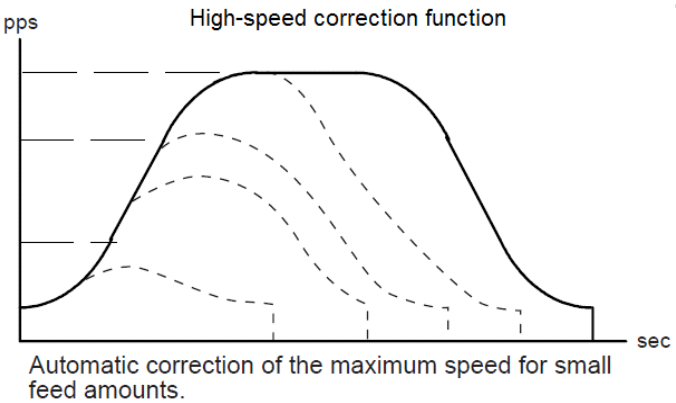
$$\text{Acceleration rate}^3 = \frac{\text{HSPD} - \text{LSPD}}{\text{ACC}/1000}$$

The minimum and maximum acceleration values depend on the high-speed and low-speed settings. Refer to [Appendix B: Speed Settings](#) for details. Deceleration time cannot be greater than 2 times acceleration time.

Below are a few examples of styles of motion profile are possible with setting SCV[axis], ACC[axis], DEC[axis], JKA[axis], JKD[axis]. This allows of maximum control of the requested motion profile. The Profile Builder in the Commander Control Software can assist you in determining the correct settings for the commander to achieve your desired profile.



Sometimes it is not possible to reach the high speed requested by the **HSPD[axis]** command, based on the acceleration, deceleration times and distance commanded. In S-curve mode, the Commander will automatically calculate a new HSPD, only to be used for that move, which allows for the quickest and smoothest possible move for the requested commands.



Individual speed and acceleration settings will take priority over global settings. A global setting will only be used under the following conditions:

- 1) The corresponding individual setting is defined as zero.
- 2) Coordinated motion is being performed.

ASCII	HSP	LSP	AC	DEC	HSPD[axis]	JKA[axis]	JKD[axis]	LSPD[a	ACC[axis]	DEC[axi

	D	D	C		s]	s]		xis]		
Standalone	HSP D	LSP D	ACC	DEC	HSPD[axi s]	-	-	LSPD[a xis]	ACC[axis]	DEC[axi

³ For Deceleration rate replace DEC value for ACC.

2.1.5 Power-on settings

Power-on settings

The Commander core has the ability to set and store the default power-on value for the digital outputs and the standalone programs.

The initial state of the enable outputs can be defined by setting the 4-bit **EOBOOT** register to the desired initial enable output value.

The initial state of the high-speed digital outputs can be defined by setting the 4-bit **DOBOOT** register to the desired initial digital output value. The **DOBOOT** value must be within the range of 0-15.

The initial state of the configurable digital I/O can be defined by setting the **IOBOOT** register to the desired initial configured output value. The **IOBOOT** value will only be applicable to configurable I/O defined as outputs by the **IOCFG** command. The **IOBOOT** value must be a 32-bit value.

Standalone programs can be configured to run on boot-up using the **SLOAD** command.

The values for the power-on settings are stored to flash memory once the **STORE** command is issued. They will take effect on the following power cycle.

ASCII	EOBOOT	DOBOOT	IOBOOT	IOCFG	SLOAD	STORE
Standalone	-	-	-	-	-	STORE

2.1.6 System settings

System settings

The Commander core has the ability to set and store the following system settings:

Command	Description
ERC[axis]	Setting the conditions for output of the ERC (error clear) signal.
ERCD[axis]	Setting the delay time of the ERC signal.
ERCP[axis]	Setting the on-time of the ERC signal.
EXST	Enables the use of the external start signal.
IERR	Disables the error flag for the end-limit and alarm signal.
INP[axis]	Enables the use of the INP signal.
POL[axis]	Sets the input modes and the polarity of the pulse, direction outputs, end limits, home, alarm, SD, INP, ERC, EO, encoder, index, and MPG signals.
SCV[axis]	Enables the use of S-curve acceleration profile
SDC[axis]	Sets the operation when the SD signal is detected.
SDE[axis]	Enables the use of the SD signal.

STORE	Saves the settings of commands to the flash memory.

System settings**In this section**

- [ERC \(deflection counter clear\)](#)
- [INP \(in-position\)](#)
- [Input modes and the polarity of the input/output logic](#)

2.1.6.1 ERC (deflection counter clear)**ERC (deflection counter clear)**

The Commander core can be set up to send an **ERC** (deflection counter clear) signal to clear any external errors from the servo drive. The ERC signal is configured with the following commands:

ERC[axis] command is 2-bit and sets the output condition of the ERC signal. The **ERC[axis]** command can write a new condition or read the condition based on the chart below.

Bit	Condition for Output	Setting
0	Error Stop	0: Disabled (Not used) 1: Enabled (Used)
1	Homing operation complete	0: Disabled (Not used) 1: Enabled (Used)

ERC command

ERCD[axis] command sets the delay time used when a deviation clear signal is sent. The **ERCD[axis]** command can write a new delay or read the condition based on the table ERCD command below.

Value	Delay
0	0 μ s
1	12 μ s
2	1.6 ms
3	104 ms

ERCD command

ERCP[axis] command sets the width of the output deviation clear signal. The **ERCP[axis]** command can write a new pulse width or read the condition based on the table ERCP command below.

Value	Pulse Width
0	12 μ s
1	102 μ s
2	409 μ s

3	1.6 ms
4	13 ms
5	52 ms
6	104 ms

ERCP command

ASCII	ERC[axis]	ERCD[axis]	ERCP[axis]
Standalone	$\frac{3}{4}$	$\frac{3}{4}$	$\frac{3}{4}$

2.1.6.2 INP (in-position)

INP (in-position)

The **INP[axis]** command determines whether the Commander core will use the **INP** (in-position) signal to determine if the motor has reached the requested position. When the **INP[axis]** function is enabled, the operation completion status is delayed until an in-position signal is received. The **INP[axis]** command can be written to set or read the current setting based on the chart below.

Value	Description
0	Disabled (Not used)
1	Enabled (Used)

INP command

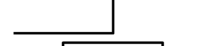
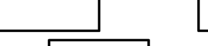
ASCII	INP[axis]
Standalone	$\frac{3}{4}$

2.1.6.3 Input modes and the polarity of the input/output logic

Input modes and the polarity of the input/output logic

The **POL[axis]** command is a 17-bit value used to set or read the input mode and/or input/output logic of various signals. Use table POL command below to see descriptions of each bit setting.

Bit	Description			
0~2	The output modes of command pulse signals			
	Positive direction operation		Negative direction operation	
	PUL output	DIR output	PUL output	DIR output
000		High		Low
001		High		Low
010		Low		High
011		Low		High

100		High	High	
101	PLS  DIR 		PLS  DIR 	
110	PLS  DIR 		PLS  DIR 	
111		Low	Low	
Bit	Description		Setting	
3	Logic of End limit signal (+/-L)		0 Positive logic	1 Negative logic
4	Logic of home signal (H)		0 Negative logic	1 Positive logic
5	Logic of alarm signal (ALM)		0 Negative logic	1 Positive logic
6	Logic of deceleration signal (SD)		0 Negative logic	1 Positive logic
7	Logic of in-position (INP)		0 Negative logic	1 Positive logic
8	Logic of deviation counter clear signal (ERC)		0 Negative logic	1 Positive logic
9	Logic of enable signal (EO)		0 Negative logic	1 Positive logic
10	Encoder input direction		0 Do Not Reverse	1 Reverse
11~12	Specification of the feedback pulse signal		00 x1	01 x2
			10 x4	11 two pulse
13	Logic of Z-index signal		0 Falling Edge	1 Rising Edge
14	MPG input direction		0 Do Not Reverse	1 Reverse
15~16	Specification of the MPG input signal		00 x1	01 x2
			10 x4	11 two pulse

POL command

The polarity can also be set for the digital inputs, digital outputs, and configurable I/O. The **DIP** command can be used to toggle the polarity of the digital inputs and the **DOP** can be used to toggle the polarity of the digital outputs. For these 0 = Negative logic, 1 = Positive logic.

The **IOP** command is 2-bit and can be used to toggle the polarity of the configurable I/O. Bit 0 sets the input logic and Bit 1 sets output logic. See table IOP command below.

Bit	Description	Setting	
0	Logic of input setting	0 Negative logic	1 Positive logic
1	Logic of output setting	0 Negative logic	1 Positive logic

IOP command

ASCII	POL[axis] J	DIP	DOP	IOP
Standalone	$\frac{3}{4}$	$\frac{3}{4}$	$\frac{3}{4}$	$\frac{3}{4}$

2.1.7 Flash Memory

Flash Memory

The Commander core has a built-in flash memory that allows it to store many of the system settings that are loaded on startup.

Once set, using the **STORE** command will save the settings in the table below to Flash memory.

DB	ERC[axis]	INP[axis]	JLIM[number]	MPD[axis]	SDC[axis]	SLOAD[prg]
DIP	ERCD[axis]	IOBOOT	JMAX[axis]	MPE[axis]	SDE[axis]	SLR[axis]
DOBOOT	ERCP[axis]	IOCFG	JMIN[axis]	MPM[axis]	SL[axis]	SLT[axis]
DOP	EXST	IOP	JSPD[axis]	POL[axis]	SLA[axis]	V[50-99]
EINT	ID	JDEL[axis]	JTOL[axis]	SAP	SLE[axis]	ZCNT
EOBOOT	IERR	JENA				

STORE command

2.2 Single-axis and Multi-axis operations

2.2 Single-axis and Multi-axis operations

Single-axis operations involve the moving of a single axis, and do not require interaction with any other axis. **Multi-axis** operations involve the moving of two or more axes; it does not require interaction between the axes. This section includes basic function blocks that are used for these operations.

In this section

- [Motor Power](#)
- [Jog operation](#)
- [Stopping](#)
- [Positioning operation](#)
 - o [Absolute or Incremental](#)
 - o [Individual Position Moves](#)
- [Homing operation](#)
 - [Z-move Operation](#)

2.2.1 Motor Power

Motor Power

The enable output (**EO**) command is most often used to command the driver to enable or disable the current to the motor. The effect of the enable output signals will depend on the characteristics of the motor drive. The polarity of this output needs to match the drive and is set with the **POL[axis](Bit9)**.

The enable output (**EO**) command is a 4-bit value, where a bit value of 1 is enabled and 0 is disabled for the corresponding axis. **EO** has a range of 0-15.

Enable output values can also be referenced one bit at a time by the **EO[1-4]** commands. Note that the indexes are 1-based for the bit references (i.e. EO1 refers to bit 0, not bit 1).

Bit	Output	Bit-Wise Command
0	EOx [X-axis]	EO1
1	EOy [Y-axis]	EO2
2	EOz [Z-axis]	EO3
3	EOu [U-axis]	EO4

EO command

The initial state of the enable outputs can be defined by setting the 4-bit **EOBOOT** register to the desired initial enable output value. The value is stored to flash memory once the **STORE** command is issued.

Bit	Output	Setting
0	EOx [X-axis]	0 - Disabled 1 - Enabled
1	EOy [Y-axis]	0 - Disabled 1 - Enabled
2	EOz [Z-axis]	0 - Disabled 1 - Enabled
3	EOu [U-axis]	0 - Disabled 1 - Enabled

EOBOOT command

ASCII	EO	EO[1-4]	EOBOOT	STORE	POL[axis]
Standalone	EO	EO[1-4]	-	STORE	-

2.2.2 Jog operation

Jog operation

A jog move is used to continuously move the motor without stopping. Use the **J[axis]** **+J[axis]**- command when operating in ASCII mode, and the **JOG[axis]** **+JOG[axis]**- in standalone mode.

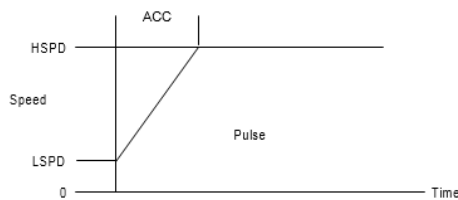
The Commander core will output pulses to move the motor immediately upon receipt of the command. Any number of motors can be moved independently at the same time.

The motor will start moving at the defined **low speed**, then will accelerate as defined by **the**

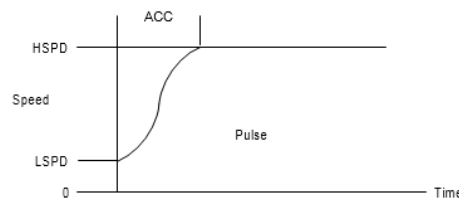
acceleration time and type, to the final defined **high speed**.

Once this move is started, the speed can be adjusted by using the **on-the-fly speed change** command; however, the motor will only stop if a limit input is activated during the move or a **STOP** command is issued.

In case of linear acceleration



In case of S-curve acceleration



If a motion command is sent while the controller is already moving, the command will not be processed. Instead, an error response is returned. See table [4.2 Error Codes](#) for details on error responses.

ASCII	J[axis][+/-]
Standalone	JOG[axis][+/-]

Operating procedure

If there is no change in the previous set data, the operation 1 below is not necessary.

- 1) Set the operating speeds
- 2) Issue the **J** command
- 3) Issue stop command

Operating conditions

- Joystick operation is disabled
- MPG operation is disabled
- Operation is stopped
- The error status is clear

Related commands:

[LSPD](#), [HSPD](#), [ACC](#), [DEC](#), [J](#), [SCV](#), [STOP](#), [ABORT](#), [ESTOP](#), [MST](#)

2.2.3 Stopping

Stopping

When the motor is performing any type of move, motion can be stopped either abruptly or with deceleration. It is recommended to use decelerated stops so that there is less impact on the system. The **ABORT[axis]** command will immediately stop an individual axis. Use the **ABORT** command to immediately stop ALL axes.

To employ deceleration on a stop, use the **STOP[axis]** command to stop an individual axis. Use the **STOP** command to stop ALL axes.

If an interpolation operation is in progress when a **STOP[axis]** or **ABORT[axis]** command is entered, all axes involved in the interpolation operation will stop.

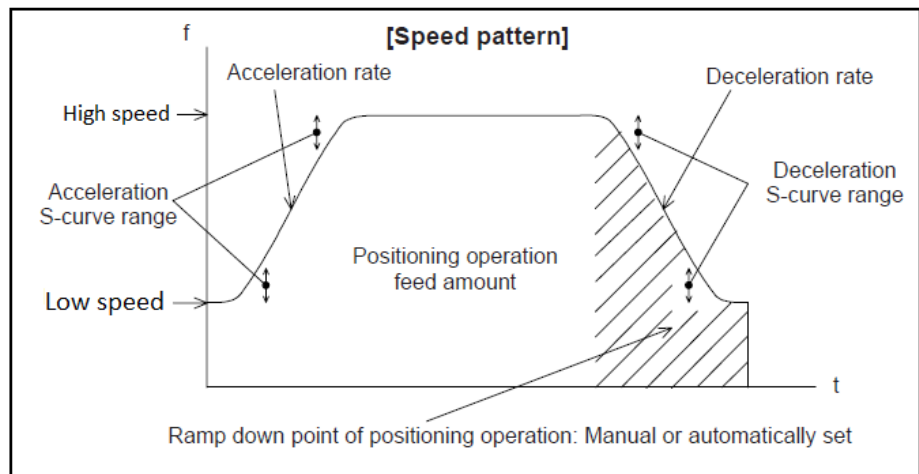
ASCII	ABORT	ABORT[axis]	STOP	STOP[axis]
Standalone	ABORT	ABORT[axis]	STOP	STOP[axis]

2.2.4 Positioning Operation

Positioning Operation

Unlike a Jog operation, a Positioning operation is a move with a specified stop point. The stop point will be either an absolute position in relation to the 0 start point, which was defined during homing, or an incremental position from the motor's current position.

The motor will start moving at the defined low speed, then will accelerate based on the acceleration time and type, to the final defined high speed. Once it gets close to the stop point, it will decelerate as defined by the deceleration time and type to the defined low speed, then the motor will stop.



At any time while the motor is moving, the specified stop point can be adjusted by using the on-the-fly target position change command, and the speed at which it is moving can be adjusted by using the on-the-fly speed change command.

In addition, the motor will stop without completing the defined positioning operation if a limit input is activated during the move or a STOP/ABORT command is issued.

The Commander core can perform positional moves for individual axes, multiple axes, or linear coordinated motion.

In this section

- [Absolute or Incremental](#)
- [Individual Position Moves](#)

2.2.4.1 Absolute or Incremental

Absolute or Incremental

The Commander core can perform positional moves in absolute or incremental mode. For absolute mode, the **ABS** command is used, and for incremental mode the **INC** command is used. Best practice is to set the absolute or incremental mode during system setup. The move mode will remain in absolute or incremental mode until it is changed.

In absolute mode, the axis will move to the specified target position. In incremental mode, the

axis will increase or decrease its current position by the specified target position.

The **MM** command can be used to determine the current move mode of the controller. Table MM command below details the value assignment of the **MM** command.

Value	Setting
0	Absolute Mode
1	Incremental Mode

MM command

ASCII	ABS	INC	MM
Standalone	ABS	INC	-

Operating procedure

If there is no change in the previous set data, the operation 1 and 2 below is not necessary.

- 1) Set the operating speeds
- 2) Set the positioning mode with **ABS** or **INC** command
- 3) Issue a positioning command

Operating conditions

- Joystick motion is disabled
- MPG operation is disabled
- Operation is stopped
- The error status is clear

Related commands

[LSPD](#),

[HSPD](#)

[ACC](#)

[DEC](#)

[X, Y, Z, U](#)

[SCV](#)

[ABS](#)

[INC](#)

[MST](#)

2.2.4.2 Individual Position Moves

Individual Position Moves

For individual axis control use **X**, **Y**, **Z** and **U** commands followed by the target position value. For example, the **X1000** command will move the X-axis to position 1000 if performed in absolute mode. **X1000Y2000** command will move the X-axis to position 1000 and Y-axis to

position 2000 if performed in absolute mode. Any number of axes can be moved independently at the same time. If **EINT=0** then the axes will move independently; however, if **EINT=1** the move will be linearly interpolated. For more information, see section [2.3.1 Linear interpolation](#).

The Commander core will output pulses to move the motor immediately upon receipt of the command.

ASCII	X[target]	Y[target]	Z[target]	U[target]	EINT
Standalone	X[target]	Y[target]	Z[target]	U[target]	EINT

2.2.5 Homing Operation

Homing operation

Home search routines involve moving the motor and using the home, limit, slowdown, or Z-index inputs to determine the zero-reference position (home point) used for absolute positioning. The Commander core has thirteen (13) different built-in home search routines. Since the signal or signals used to define the home point varies depending on the homing mode selected, carefully select a homing mode that best matches the environment and available input signals of your system.

Some homing routines that involve a decelerated stop may result in a final position that is non-zero. The zero-reference position is correctly preserved as the position that was marked when the home trigger condition was detected.

In ASCII mode, use the **H[axis][+/-][mode]** command to perform a home move. Standalone mode will use the **HOME[axis][+/-][mode]** command to perform a homing routine. Please refer to [Appendix A: Homing Mode Actions](#) for a full list of the available homing routines and a diagram of the inputs required and their timing.

When **ERC[axis](bit1)** is set to 1, the **ERC**(deflection counter clear) signal is output when the zero-reference position (home point) is reached. For more information on the ERC timing, see homing operation in [Appendix A: Homing Mode Actions](#).

ASCII	H[axis][+/-][mode]	ERC[axis]
Standalone	HOME[axis][+/-][mode]	-

See also

[Z-move Operation](#)

2.2.5.1 Z-move Operation

Z-move Operation

The Commander core has a built-in move called a Z-Move. When the **ZMOVE[axis][dir]**

command is sent, the motor will move in the direction commanded at **HSPD** until such time as the number of Z-index pulses are detected, as set by the **ZCNT[axis]=[0-15]** command.

The **ZCNT[axis]** command also works with any homing operation that uses the Z-index pulses as part of the homing operation to set the number of Z-index pulses that need to be detected. The range for **ZCNT[axis]** is 0-15. The number of Z-index pulses to be detected will be the number set plus one. **Example:** 0 = stop at the first Z-index detection, 1 = stop at the second Z-index detection, 15 = stop at the 16th Z-index detection.

ASCII	ZMOVE[axis][dir]	ZCNT[axis]
Standalone	ZMOVE[axis][dir]	-

Operating procedure

If there is no change in the previous set data, the operations 1 and 2 below are not necessary.

- 1) Set the operating speeds
- 2) Set the Z-index count numbers and the Z-index input logic
- 3) Issue a homing command

Operating conditions

- Joystick motion is disabled
- MPG operation is disabled
- Operation is stopped
- The error status is clear

Related commands

[LSPD](#), [HSPD](#), [ACC](#), [DEC](#), [H](#), [ZCNT](#), [MST](#)

2.3 Interpolation Operation

Interpolation Operation

Interpolated operations involve two or more axes whose movements are interdependent. There are many types of interpolated moves. **Linear interpolation** is when two or more axes work together to create movement along a straight line. For linear interpolation, each axis moves in only one direction, and start and stop at the same time. **Circular interpolation** is when two axes work together to create movement along the outer rim of a circle. For circular interpolation, each axis will change direction and speed as required to allow for a circular path of movement. A popular subset of circular interpolation is **arc interpolation**, when two axes work together to move along the outer rim of a circle for a set number of degrees. **Helical or tangential interpolation** is when a circular or arc interpolation of the X and Y-axis is combined with linear interpolation of the center point of the circle and the Z-axis. This allows for movement along a helical motion path. The Commander core allows for all of these different moves, independently or in combination. These moves can be **buffered** together to allow for seamless **continuous** motion.

For more details on how the Commander core handles interpolation, refer to [Appendix C -](#)

Interpolation.

In this section

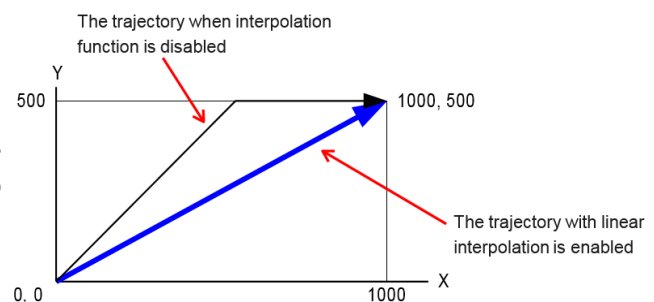
- [Linear interpolation](#)
- [Circular interpolation](#)
 - o [Arc interpolation](#)
- [Helical or Tangential interpolation](#)
- [Buffered interpolation operation](#)

2.3.1 Linear Interpolation

Linear interpolation

The Commander core can perform linear interpolated motion for up to 4 axes. The X, Y, Z, and U commands can be combined into a single move command that consists of up to 4 target positions (one for each axis). Any combination of axes can be used for linear interpolation, and at least two axes must be referenced in order to perform linear interpolation. Only one linear interpolation set can be executed at a time.

By default, interpolation is enabled (**EINT=1**). While interpolation is enabled, linear interpolation operations are performed by writing positioning commands of multiple axes on the same line. There is no dedicated command for linear interpolation. If interpolation is disabled (**EINT=0**), writing positioning commands of multiple axes on the same line will not result in an interpolated move. See the figure to the right.



For example, the **X1000Y1000Z100U800** will move all four axes to the position (1000, 1000, 100, 800). Similarly, the **X1000Y-1000Z500** will move the X, Y, and Z-axis to the position (1000, -1000, 500) while the U axis remains idle.

The Commander core will output pulses to move the motors immediately upon receipt of the command. While performing a linear interpolated motion, any axis not included in the linear interpolated motion can be moved independently at the same time, or included in other interpolated motions, without impacting the linear interpolated motion.

For more details on how the Commander core handles interpolation, refer to [Appendix C - Interpolation](#).

ASCII	X[target]	Y[target]	Z[target]	U[target]	EINT
Standalone	X[target]	Y[target]	Z[target]	U[target]	EINT

Operating procedure

If there is no change in the previous set data, operations 1 to 3 below are not necessary

- 1) Set an operating speed
- 2) Set the positioning coordinate mode with **ABS** or **INC** command
- 3) Enable an interpolation operation with the **EINT** command
- 4) Issue a positioning operation command of multiple axes

Operating conditions

- Joystick motion is disabled
- MPG operation is disabled
- Operation is stopped
- The error status is clear

Related commands

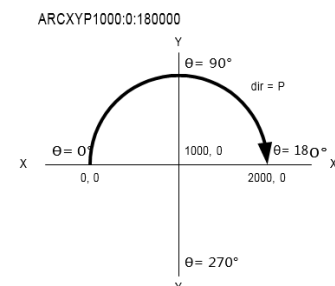
[LSPD](#), [HSPD](#), [ACC](#), [DEC](#), [EINT](#), [X](#), [Y](#), [Z](#), [U](#), [MST](#)

2.3.2 Circular Interpolation

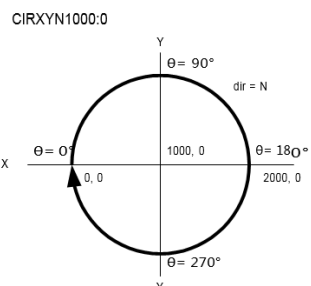
Circular interpolation

Circular interpolation is when two axes work together to cause movement along the outer rim of a circle. The Commander core supports circular interpolation moves using the **CIRP** and **CIRN** commands. This will perform a complete 360° circle using any two axes. When less than 360° of motion is needed, then arc interpolation is used with the **ARCP** and **ARCN** commands. The two functions can be combined when more than 360° of motion is needed. Circles and arcs can be drawn using any two axes.

Trajectory of the circular interpolation operation specifying the end point

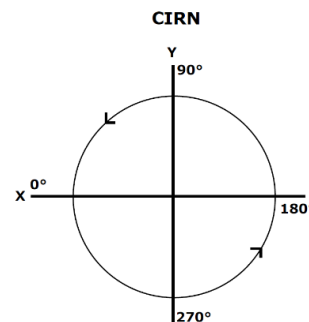
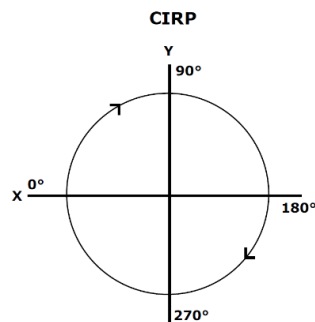


Circular interpolation trajectory drawing a perfect circle



CIR[A1][A2]P[C1]:[C2] – Draw a circle in CW direction where [A1][A2] signifies the selected axes and [C1][C2] signifies the absolute position of the circle center.

CIR[A1][A2]N[C1]:[C2] – Draw a circle in CCW direction where [A1][A2] signifies the selected axes and [C1][C2] signifies the absolute position of the circle center.



For example, the command **CIRXZP1000:1000** will move the X and Z axis in a full circle in the CW direction with the XZ position (1000, 1000) as the circle center.

Circular interpolated moves will use the current axis positions, as well as the specified circle

center, to automatically calculate the circle radius. The maximum allowable radius is 134,216,773 pulses for circular interpolated moves. All circular moves are interpreted as absolute moves.

ASCII	CIR[A1][A2]P[C1]:[C2]	CIR[A1][A2]N[C1]:[C2]
Standalone	CIR[A1][A2]P[C1]:[C2]	CIR[A1][A2]N[C1]:[C2]

Operating procedure

If there is no change in the previous set data, operations 1 to 3 below are not necessary.

- 1) Set an operating speed
- 2) Set the positioning coordinate mode with **ABS** or **INC** command
- 3) Enable the interpolation operation with the **EINT** command
- 4) Issue a circular interpolation operation command

Operating conditions

- Joystick motion is disabled
- MPG operation is disabled
- The operation is stopped
- The error status is clear

Related commands

[LSPD](#), [HSPD](#), [ACC](#), [DEC](#), [EINT](#), [ARC](#), [CIR](#), [MST](#)

See also

[Arc interpolation](#)

2.3.2.1 Arc Interpolation

Arc Interpolation

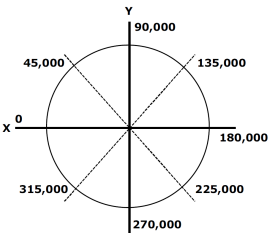
The Commander core supports arc interpolation moves using the **ARCP** and **ARCN** commands.

ARC[A1][A2]P[C1]:[C2]:[θ] – Draw an arc in CW direction where [A1][A2] signifies the selected axes, [C1][C2] signifies the absolute position of the arc center, and θ signifies the absolute arc angle.

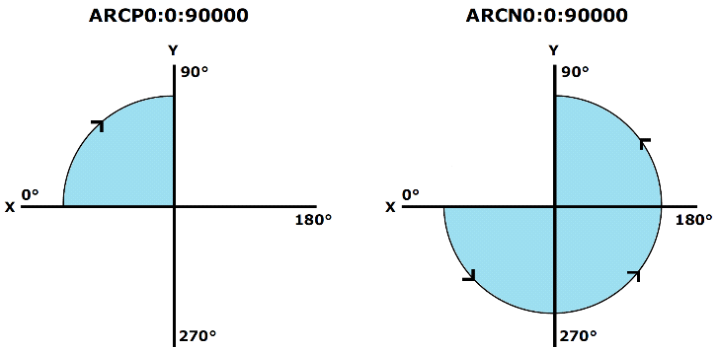
ARC[A1][A2]N[C1]:[C2]:[θ] – Draw an arc in CCW direction where [A1][A2] signifies the selected axes, [C1][C2] signifies the absolute position of the arc center, and θ signifies the absolute arc angle.

An arc interpolated move will use the current axis positions as well as the specified arc center to automatically calculate the radius. The maximum allowable radius is 134,216,773 pulses on

arc interpolated moves. All arc moves are interpreted as absolute moves. The absolute angle (θ) is in units of a millidegree. For example, to move to the absolute angle of 45° , the absolute angle parameter would be 45000. The absolute angle locations can be found in the figure to the right.



The figure below shows the difference between issuing the **ARCXYP0:0:90000** and the **ARCXYN0:0:90000** command if both cases have the same starting point at 0 degrees.



For more details on how the Commander core handles interpolation refer to [Appendix C - Interpolation](#).

ASCII	ARC[A1][A2]P[C1]:[C2]:[θ]	ARC[A1][A2]N[C1]:[C2]:[θ]
Standalone	ARC[A1][A2]P[C1]:[C2]:[θ]	ARC[A1][A2]N[C1]:[C2]:[θ]

Operating procedure

If there is no change in the previous set data, operations 1 to 3 below are not necessary.

- 1) Set an operating speed
- 2) Set the positioning coordinate mode with **ABS** or **INC** command
- 3) Enable the interpolation operation with the **EINT** command
- 4) Issue a circular interpolation operation command

Operating conditions

- Joystick motion is disabled
- MPG operation is disabled
- The operation is stopped
- The error status is clear

Related commands

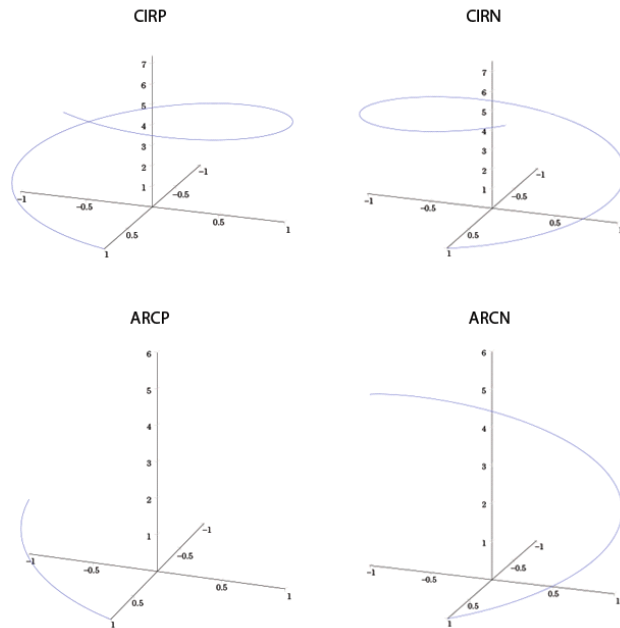
[LSPD](#), [HSPD](#), [ACC](#), [DEC](#), [EINT](#), [ARC](#), [CIR](#), [MST](#)

2.3.3 Helical or Tangential interpolation

Helical or Tangential interpolation

The Commander core is able to combine the linear and circular interpolation operations in order to perform helix or tangential type moves. These moves can only use the X and Y axes for circular interpolation, with the Z axis used for linear coordination.

A move with less than 360° movement for the X and Y axes is made by using the **ARCTP** and **ARCTN** commands. A move with 360° movement for the X and Y axes is made by using the **CIRTP** and **CIRTN** commands. A continuous movement with more than 360° of movement is possible by using the buffer operation to combine the **ARCTP/ARCTN** and **CIRTP/CIRTN** commands.



ARCTP[C1]:[C2]:[θ]:[Z target] – Draw an arc in CW direction where [C1][C2] signifies the absolute position of the arc center, θ signifies the absolute arc angle, and [Z target] signifies the absolute end point.

ARCTN[C1]:[C2]:[θ]:[Z target] – Draw an arc in CCW direction where [C1][C2] signifies the absolute position of the arc center, θ signifies the absolute arc angle, and [Z target] signifies the absolute end point.

CIRTP[C1]:[C2]:[Z target] – Draw a circle in CW direction where [C1][C2] signifies the absolute position of the circle center and [Z target] signifies the absolute end point.

CIRTN[C1]:[C2]:[Z target] – Draw a circle in CCW direction where [C1][C2] signifies the absolute position of the circle center and [Z target] signifies the absolute end point.

Note: that these commands will use the U axis for calculation purposes; therefore, the U axis should NOT be used for motion while using these commands.

For more details on how the Commander core handles interpolation, refer to [Appendix C - Interpolation](#).

ASCII	ARC[A1][A2]P[C1]:[C2]:[θ]	ARC[A1][A2]N[C1]:[C2]:[θ]
Standalone	ARC[A1][A2]P[C1]:[C2]:[θ]	ARC[A1][A2]N[C1]:[C2]:[θ]

ASCII	CIR[A1][A2]P[C1]:[C2]	CIR[A1][A2]N[C1]:[C2]
Standalone	CIR[A1][A2]P[C1]:[C2]	CIR[A1][A2]N[C1]:[C2]

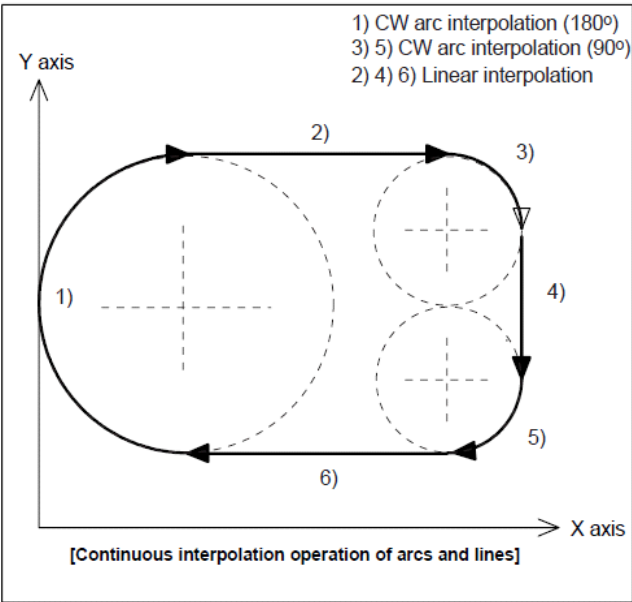
2.3.4 Buffered Interpolation operation

Buffered Interpolation Operation

The Commander core’s buffer mode allows for up to 100 prebuffered interpolated moves. This allows for the creation of unique patterns without any delay between moves. Each move has its own constant speed setting.

For normal operation, commands are written to an operation register.

For buffer operation, the coordinated motion commands are written to the buffer registers instead. There are 100 separate buffer registers to hold coordinated motion commands. When the start buffer motion (**BSTART**) command is sent, the coordinated motion command in Buffer0 is moved to the operation register. Then the coordinated motion command in Buffer1 is moved to Buffer0 and so on. This process is continued until all the buffer registers are empty or the buffer mode is turned off with the (**BF**) command. See table below for details.



Operation register	For normal operation, a command is written in the operation register, and the operation starts immediately.

Buffer0	100 Buffer registers	For buffer operations, coordinated motion commands are written to the buffer registers. When buffer operation starts, these commands are automatically written to the operation register, which automatically starts the operation, and are deleted from the buffer registers. Buffered commands are processed on a first-in first-out basis.
Buffer1		
Buffer2		
⋮		
Buffer9 7		
Buffer9 8		
Buffer9 9		

New Commands	As the buffered commands are moved to the operating register, the buffer registers can be loaded with additional coordinated motion commands.

To turn the buffer mode on and off, use the **BO** and **BF** ASCII commands, respectively. For standalone mode, the **BUFON** and **BUFOFF** commands should be used. The buffered move

operation cannot be used while StepNLoop is enabled.

When buffer mode is enabled, motion will not start until the appropriate command has been sent, even if there are commands in the buffer. This allows the buffer to be fully loaded before any move is processed. The command **BSTART** starts feeding the buffered commands to the operation register.

The status of the buffer operation can be read using the **BSTAT** command. The return value will contain the buffer enable status, buffer start and end positions, and the number of available buffer registers. The return value has the following format:

[Buffer enabled]:[Buffer start]:[Buffer end]:[Available Buffer]

The buffer operation only supports motion of the X, Y, and Z axes. In addition, only linear, circular and arc interpolation moves can be added to the buffer. As such only the linear, circular and arc interpolation commands below are supported during buffered operation.

Linear interpolation

In ASCII mode, a buffered move command will have the syntax **I[X pos]:[Y pos]:[Z pos]:[speed]**. For example, the **I1000:2000:3000:5000** command indicates position (1000, 2000, 3000) at a speed of 5000. If buffered mode is not enabled, the **I** command will not be processed by the controller.

In standalone mode, the buffered move command will have the syntax **X[pos]Y[pos]Z[pos]**, and the speed for the move command will use the current global high-speed setting. If buffered mode is not enabled, these standalone commands will be processed as linear interpolation commands.

Circular and Arc interpolation

The **ARC** and **CIR** commands, described in Section [2.3.2 Circular interpolation](#), can be used when buffer mode is enabled. The syntax for these commands is the same in ASCII and standalone mode.

ASCII	BO	BF	BSTART	BSTAT	I[X]:[Y]:[Z]:[S]
Standalone	BUFON	BUFOFF	ISTART	-	X[pos]Y[pos]Z[pos]

Operating procedure

- 1) Enable buffer operation with **BO** commands.
- 2) **I** command writes an interpolation operation (**CIR** or **ARC** command is also available)
- 3) Buffer operation is started with a **BSTART** command
- 4) Add the interpolation operation while checking the buffer register free space with the **BSTAT** command
- 5) If there is no interpolation operation to be added, the buffer operation needs to be disabled by the **BF** command after completion

Operating conditions

- Joystick motion is disabled
- MPG operation is disabled
- Corrective action is disabled
- The error status is clear

Related commands

[BO](#), [BF](#), [BSTAT](#), [BSTART](#), [I](#), [CIR](#), [ARC](#)

2.4 On-the-fly-speed/target position change

On-the-fly-speed/target Position Change

The Commander core supports the ability to change the moving speed and/or the target position on the fly.

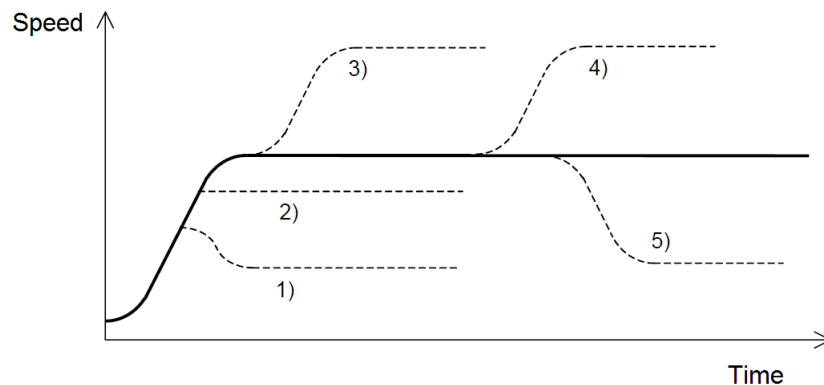
In this section

- [On-the-fly Speed Change](#)
- [On-the-fly Target Position Change \(Target position change\)](#)

2.4.1 On-the-fly Speed Change

On-the-fly Speed Change

This function can change the speed while an axis is in motion, at any point in the motion profile. The figure below shows a few examples of how speed changes are executed.



If the target speed is changed during the acceleration process, and:

- 1) If the newly set value is lower than the pulse rate at the time of the change, deceleration is made to the newly set value.
- 2) If the newly set value is equal to or higher than the pulse rate at the time of the change but less than the previous target, acceleration is made to the newly set value.
- 3) If the newly set value is greater than the previous target, acceleration is made to the preset pulse rate and then to the newly set value.

If the target speed is changed while traveling at the previously set target speed:

- 4) If the new value is higher than the previous target, acceleration is made to the newly set value.
- 5) If the new value is lower than the previous target, deceleration is made to the newly set value.

On-the-fly speed change can be achieved using the **SSPD[axis]=[newspeed]** command. Before executing an on-the-fly speed change, ensure the correct speed range is set with the **SSPDM[axis]** command. For more information on the speed range setting, see [Appendix B: Speed Settings](#) and SSPDM in the command reference.

ASCII	SSPD[axis]=[newspeed]	SSPDM[axis]
Standalone	SSPD[axis]=[newspeed]	SSPDM[axis]

Operating procedure

If there is no change in the previous set data, operation 1 below is not necessary.

- 1) Set the speed range with SSPDM command
- 2) Issue the speed change command

Operating conditions

- While operating at HSPD (not decelerating)
- Buffer operation must disable
- S-curve acceleration/deceleration mode must disable

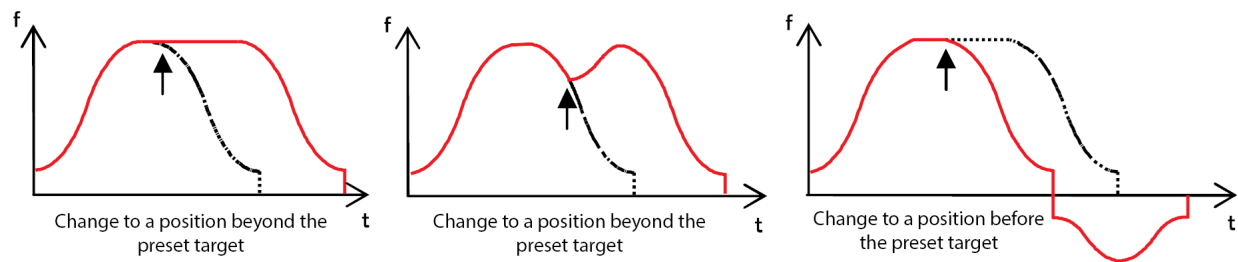
Related commands

[SSPD](#), [SSPDM](#), [MST](#), [PS](#)

2.4.2 On-the-fly Target Position Change (Target position change)

On-the-fly Target Position Change (Target position change)

On-the-fly target position change can be achieved using the **T[axis][newtarget]** command, and will change the final destination of the motor while the motor is moving. If the motor has already passed the new target position, it will reverse direction when the target position change command is issued.



An on-the-fly target position change command will only be valid if the specified axis is performing an individual position move. An interpolated, jogging, or homing move will not accept an on-the-fly target position change command.

In **ABS** mode, the target change will be to the absolute position from the homed zero point. In **INC** mode the target change will be to the incremental position from the original move start point.

ASCII	T[axis][newtarget]
Standalone	-

Operating procedure

There is no predetermined procedure. On-the-fly target position change commands can be issued while the axis is moving.

Operating conditions

- While operating with the set value of HSPD (not decelerating)

Related communication commands

[T](#), [MST](#)

2.5 Advanced Functions

Advanced Functions

The Commander supports a number of built in advanced functions. This includes analog joystick control, and manual pulse generator (MPG) operation.

In this section

- [Joystick operation \(Analog input control\)](#)
 - o [Direction control](#)
 - o [Speed control](#)
 - o [Position control](#)
- [Manual pulse generator \(MPG\) operation](#)
 - o [MPG signal input mode](#)
 - o [Electronic gearing the MPG signal](#)
 - o [Encoder Frequency and Impulse Multiplier/Divider](#)
- [StepNLoop Closed Loop Control operation](#)

2.5.1 Joystick operation (Analog input control)

Joystick operation (Analog input control)

Joystick control is available on the Commander core for the X and Y axes. When this mode is enabled, the pulse speed and direction output for the X and Y axes can be controlled by their corresponding analog input. See the axis to analog input assignment in the table analog inputs below.

Axis	Analog

	Input
X	AI1
Y	AI2

Analog inputs

To enable or disable joystick control for an axis, use the ASCII command **JENA** or the standalone command **JOYENA**. The joystick enable parameter is a 2-bit value. For example, A joystick enable value of 3 means the joystick feature is enabled on both the X and Y axes. If joystick control is enabled, StepNLoop is automatically disabled.

Bit	Description	Setting	
0	Joystick control X-axis	0 - Disabled	1 - Enabled
1	Joystick control Y-axis	0 - Disabled	1 - Enabled

JENA command

For the Commander core to properly translate the analog inputs into a safe pulse speed and direction output for your system, you first must define what is acceptable and safe for your system. This includes defining the relation between voltage and **direction** of travel, and the zero-tolerance (or dead) zone during which there is no output from the Commander core. The **safe speeds** are defined as the maximum speed the analog input can command and maximum rate of speed change. The joystick **positioning** soft limits define where the analog inputs can command a system to move.

ASCII	JENA
Standalone	JOYENA

In this section

- [Direction control](#)
- [Speed control](#)
- [Position control](#)

2.5.1.1 Direction control**Direction Control**

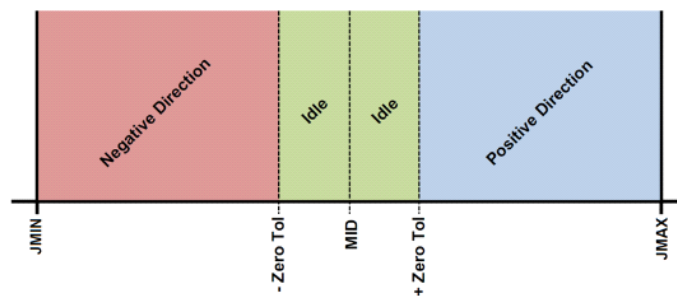
The **JMAX[axis]** command can be used to define the maximum analog input value for the specified axis. This parameter has units of millivolts (mV) and a range of 0 to 3300. This value will define the analog input value in which the maximum joystick speed occurs in the positive direction. This command is only available in ASCII mode.

Similarly, the **JMIN[axis]** command can be used to define the minimum analog input value for the specified axis. This parameter has units of millivolts (mV) and a range of 0 to 3300. This value will define the analog input value in which the maximum joystick speed occurs in the negative direction. This command is only available in ASCII mode.

MID represents the zero-joystick position. There is no command corresponding to *MID*. This is an internally calculated value using the following formula:

$$MID = ((JMAX[axis] - JMIN[axis]) / 2) + JMIN[axis]$$

During joystick operation, analog input of **JMIN[axis]** to *MID* represents negative joystick direction and analog input of *MID* to **JMAX[axis]** represents positive joystick direction. Setting the value of **JMAX[axis]** to be less than the value of **JMIN[axis]** will result in a change of direction polarity during joystick operation.



The ASCII command **JTOL[axis]** can be used to define the zero-tolerance (or dead) zone around the *MID* value. No movement will occur while the analog input value is within the zero-tolerance zone. This parameter has units of millivolts (mV) and a range of 0 to 3300. The zero-tolerance parameter will be on the positive and negative side of the *MID* point to define the zero-tolerance zone. See the figure above for details.

Example: The zero-tolerance zone at **JMAX** = 3,250 mV, **JMIN** = 10 mV, **JTOL** = 50 mV will be

$$MID = ((JMAX[axis] - JMIN[axis]) / 2) + JMIN[axis]$$

- 1580 mV ~ 1680 mV since $((3,250 - 10) \div 2) + 10 = 1,630$ mV
- 1,630 mV $\pm$ 50 mV

ASCII	JMAX[axis]	JMIN[axis]	JTOL[axis]
Standalone	-	-	-

2.5.1.2 Speed control

Speed control

The maximum joystick speed for the X and Y axes is set using the ASCII command **JSPD[axis]**. This parameter will define the speed of the specified axis at the maximum and minimum analog input values.

The maximum allowable speed change (delta) for the X and Y axes is set using the ASCII command **JDEL[axis]**. This parameter will control the acceleration/deceleration of the specified axis while the joystick mode is enabled.

ASCII	JSPD[axis]	JDEL[axis]
Standalone	-	-

2.5.1.3 Position control

Position Control

Joystick control also has soft limits, which are defined by a negative outer limit, negative inner limit, positive inner limit, and a positive outer limit, using the **JLIM[number]** command. The soft limits are in units of pulses.

When moving in a positive direction, as soon as the positive inner limit is crossed, the speed is reduced. Speed will continue to decline as the position moves closer to the outer limit value. If the position reaches the positive outer limit, the joystick speed is set to zero and movement in the positive direction is prohibited. The same behavior is applicable to the negative direction and negative limits.

The figure below represents the relationship between the joystick speed and inner and outer limits.

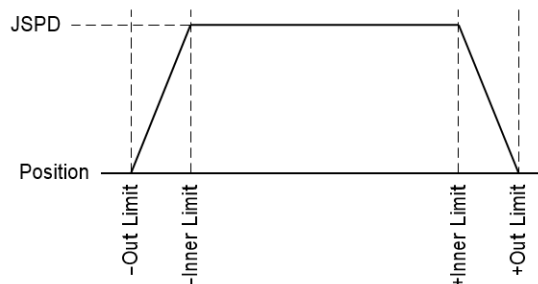


Table **JLIM** command below shows the command assignment of the negative and positive soft limits for joystick control, as well as a summary of the additional joystick commands.

Command	Description
JLIM1	X-axis negative outer limit
JLIM2	X-axis negative inner limit
JLIM3	X-axis positive inner limit
JLIM4	X-axis positive outer limit
JLIM5	Y-axis negative outer limit
JLIM6	Y-axis negative inner limit
JLIM7	Y-axis positive inner limit
JLIM8	Y-axis positive outer limit

JLIM command

This limit value is based on the value "0" of the command pulse counter. If a stop command is issued while a joystick operation is active, it will return an invalid status.

ASCII	JENA	JMAX[axis]	JMIN[axis]	JSPD[axis]	JDEL[axis]	JLIM[number]
Standalone	JOYENA	-	-	-	-	-

Operating procedure

If there is no change in the previous set data, operation 1 below is not necessary.

- 1) Set all the joystick operation values
- 2) Enable joystick operation with JENA command
- 3) Move the axis by operating the joystick
- 4) Disable the joystick operation at the end of the operation

Operating conditions

- Joystick operation is disabled
- MPG operation is disabled
- Operation is stopped
- The error status is clear

Related communications commands

[JSPD](#), [JDEL](#), [JMAX](#), [JMIN](#), [JTOL](#), [JENA](#), [JLIM](#), [AI](#), [MST](#)

2.5.2 Manual pulse generator (MPG) operation

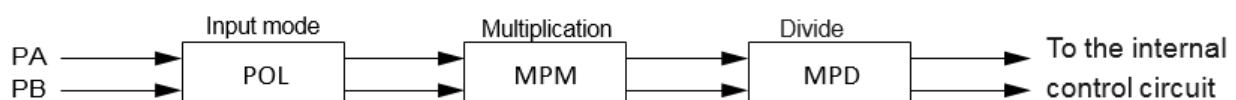
Manual pulse generator (MPG) operation

The MPG input can be used for a number of functions such as:

- 1) A manual pulse generator, which allows for fine adjustment of an axis.
- 2) Cam follower, which will allow the axis to follow a cam axis based on its encoder output.
- 3) Electronic gearing, which allows an axis to follow the input commands but at a different rate of speed (faster or slower).
- 4) Encoder Frequency and Impulse Multiplier/Divider

The Commander core has a dedicated MPG input for each axis. To enable the MPG input, set the **MPE[axis]** command to 1. To disable the MPG input, set the **MPE[axis]** command to 0.

The MPG inputs are controlled by three commands: MPG signal input mode (**POL[axis]**), multiplication (**MPM[axis]**) and division (**MPD[axis]**). These work together to set how the pulses from the MPG input controls the motion of an axis.



In this section

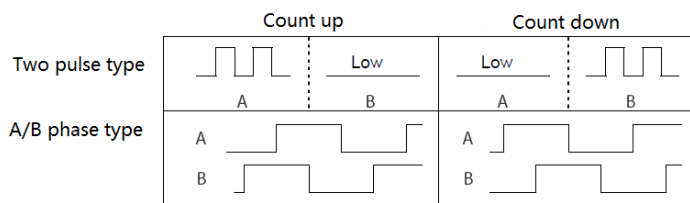
- [MPG signal input mode](#)
- [Electronic gearing the MPG signal](#)
- [Encoder Frequency and Impulse Multiplier/Divider](#)

2.5.2.1 MPG signal input mode

MPG signal input mode

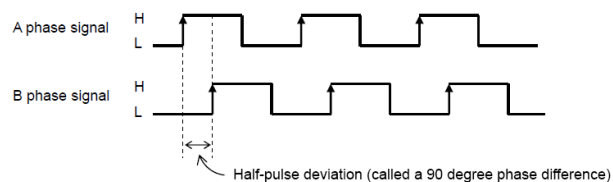
The Commander core can accept two different pulse types, which can be set using the **POL[axis](Bit15~16)** command.

- Two pulse type (not common) is where pulses are sent to A input to count up and B input to count down. Each pulse input is one count of the encoder.

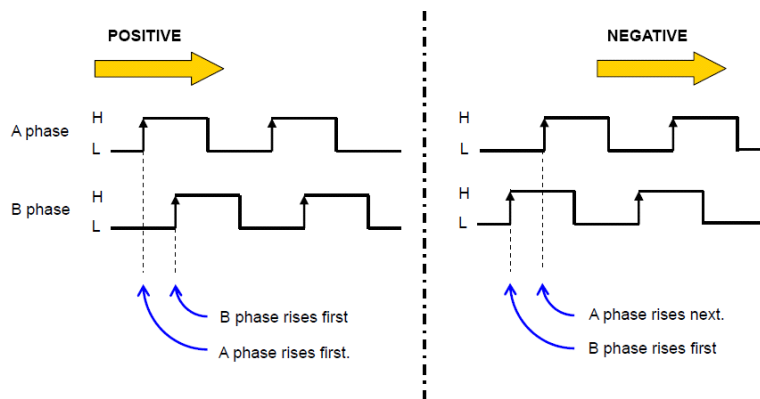


- A/B phase type (common).

- For an A/B phase type, there should be 90° difference between the pulses from A and B.

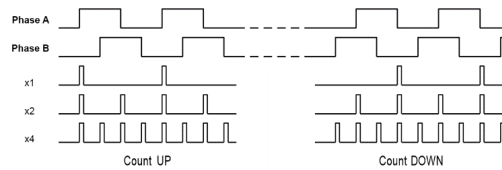


- By default, when the A phase leads the B phase motion is in the positive direction, and when B phase leads the A phase motion is in the negative direction. The direction can be changed by **POL[axis](Bit14)**.



- For an A/B phase type you can select x1, x2 or x4 multiplication rate. The figure below shows how the A/B phase counts would be interpolated for a x1, x2, and x4 multiplication rate.
 - x1 multiplication – Only the rising edge of the A phase is counted, B phase is not counted and is only used to check the direction of motion.

- ii. x2 multiplication – The rising and falling edge of the A phase is counted, B phase is not counted and is only used to check the direction of motion.
- iii. x4 multiplication – The rising and falling edge of the A and B phase is counted, and used to check the direction of motion.



Maximum frequency input on the MPG line is the maximum pulses that can be input to the MPG input, not the counts. Since most manual pulse generator outputs are rated based on a x4 multiplication rate, the output frequency of pulses is most likely ¼ the count rate output. Please check the datasheet for your manual pulse generator to confirm.

2.5.2.2 Electronic gearing the MPG signal

Electronic gearing the MPG signal

After the input mode stage, the MPG pulse signal is converted to counts based on the **POL[axis]** settings. The counts then pass through an electronic gearing stage, which includes a multiplier **MPM[axis]** and divider **MPD[axis]** circuit.

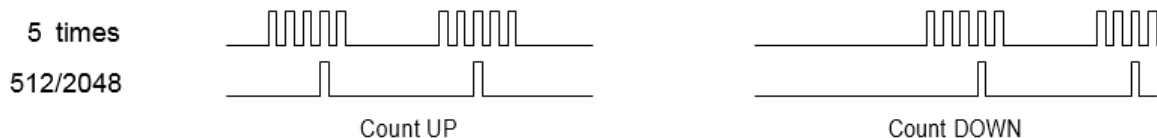
Multiplication (in the case of MPM = 5)



MPM[axis] can be used to access the multiplier. The multiplier can be set to a value from [1 to 32]. A value of 1 will turn off the multiplier.

MPD[axis] is the numerator of the divider circuit. The **MPD[axis]** can be set to a value from [1 to 2048]. The denominator of the divider circuit is fixed as 2048. A value of 2048 will turn the divider off.

Divide (in the case of MPD = 512) 512/2048 = 1/4



The multiplier and divider circuits can be used together to create any ratio. The multiplier and divider use the following equation to determine the number of output counts ⁴:

$$\text{Output counts} = (\text{MPG counts}) * \frac{\text{MPM} * \text{MPD}}{2048}$$

Example: Final ratio that is needed is 100 pulses-in and 125 pulses-out (1.25). By setting **MPM[axis]** to 5, then the input of 100 pulses is 500. since $500/4 = 125$ **MPD[axis] = 512**

(512/2048 = 1/4) the output pulses will be 125.

$$125 = (100) * \frac{5 * 512}{2048}$$

MP[axis] can be written to set an MPG position or read to see the current position. The counter value here is the final counts after the electronic gearing stage.

ASCII	MPD[axis]	MPE[axis]	MPM[axis]	MP[axis]	POL[axis]
Standalone	-	MPGE[axis]	-	-	-

* MPG counts are counts after the input mode setup done by POL[axis].

2.5.2.3 Encoder Frequency and Impulse Multiplier/Divider

Encoder Frequency and Impulse Multiplier/Divider

The MPG input and its corresponding pulse output can be used to multiply or divide an encoder or impulse input. This requires the use of a single axis, the encoder or impulse input is connected to the MPG input, and the processed signal is output on the pulse and direction outputs.

Follow these steps below:

1. Set **POL[axis](Bit14~16)** to match your encoder or impulse input.
See [2.5.2.1 MPG signal input mode](#).
2. Set **MPM[axis]** and **MPD[axis]** (electronic gearing) to achieve the appropriate multiple or derivative of the input signal that is required.
See [2.5.2.2 Electronic gearing the MPG signal](#).
3. Set **POL[axis](Bit0~2)** to match your required encoder or impulse output format.
See [2.1.6.3 Input modes and the polarity of the input/output logic](#).

4.

2.5.3 StepNLoop Closed Loop Control operation

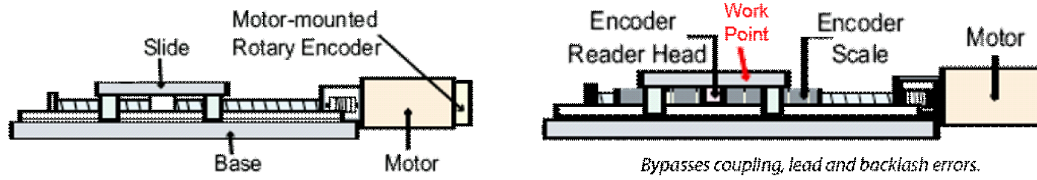
StepNLoop Closed Loop Control Operation

The Commander core features a closed-loop position verification algorithm called StepNLoop. The algorithm requires the use of an incremental encoder.

StepNLoop performs the following operations:

- 1) **Position Verification:** At the end of any targeted move, StepNLoop will perform a correction if the current error is greater than the tolerance value (**SLT[axis]**).
- 2) **Delta Monitoring:** The delta value is the difference between the actual and the target position. When delta exceeds the error range value (**SLE[axis]**), the motor is stopped and the StepNLoop status goes into an error state. Delta monitoring is performed during moves – including homing and jogging. To read the delta value, use the **D[axis]** command.

StepNLoop is commonly used to close a position loop on stepper systems (encoder on back of motor). StepNLoop can also be used for the external position loop on stepper or servo systems, which have a loss of motion due to backlash or hysteresis, as long as the encoder feedback to the Commander core is from the final work-point instead of the motor. See Figures "Encoder at motor" and "Encoder at workpoint" below, as examples.



Encoder at motor

Encoder at work-point

This table outlines how StepNLoop behaves in different scenarios.

Condition	StepNLoop behavior (motor is moving)	StepNLoop behavior (motor is idle)
$\delta \leq \text{SLT}$	Continue to monitor the D[axis]	In Position. No correction is performed.
$\delta > \text{SLT}$ AND $\delta < \text{SLE}$	Continue to monitor the D[axis]	Out of Position. A correction is performed.
$\delta > \text{SLT}$ AND $\delta > \text{SLE}$	Stall Error. Motor stops and signals an error*.	Error Range Error. Motor stops and signals an error*.
Correction Attempt > SLA	N/A	Max Attempt Error. Motor stops and signals an error*.

StepNLoop behavior

Key:

[δ]: Error between the target position and actual position

SLT: Tolerance range

SLE: Error range

SLA: Max. correction attempt

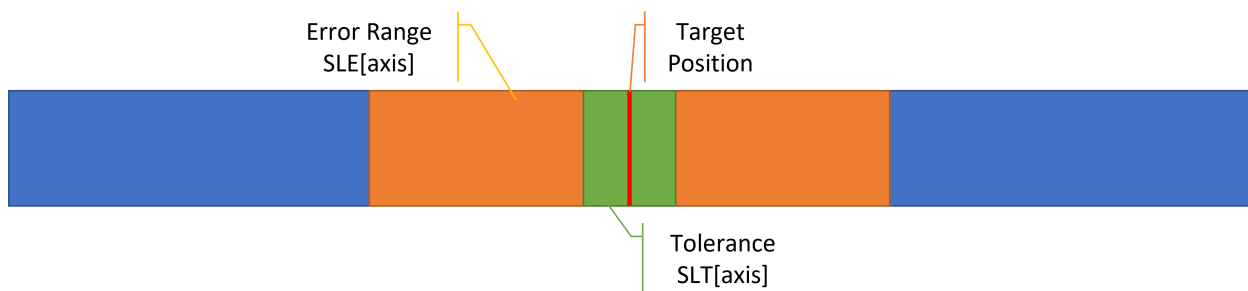
Footnote:

* The error status here is not the main status (**MST**) but the status (**SLS**) of the compensation operation.

In order to perform the StepNLoop Closed Loop Control operation, the following four parameters must be set beforehand. Parameters should be set for each axis.

StepNLoop	Description	Command

Parameter		
StepNLoop Ratio	The ratio between motor pulses and encoder counts. This ratio will depend on the motor type, micro-stepping, encoder resolution, and decoding multiplier. The value must be in the range [0.001, 999.999].	SLR[axis]
Tolerance	The maximum error between target and actual position that is considered "In Position." In this case, no correction is performed. Units are in encoder counts.	SLT[axis]
Error Range	The maximum error between target and actual position that is not considered a serious error. If the error exceeds this value, the motor will stop immediately and go into an error state.	SLE[axis]
Correction Attempt	The maximum number of correction tries that the controller will attempt before stopping and going into an error state. Setting SLA[axis]=0 will allow for infinite number of correction tries.	SLA[axis]



With StepNLoop enabled: When making a move to a target position (Red Line above), if the final position is inside the tolerance range set by **SLT[axis]** (Green area above), then the move is finished successfully. If the move ends inside the error range set by **SLE[axis]** (Orange area above), the StepNLoop process will attempt to move the motor position until it is inside the tolerance range. If the motor position ends outside the error range (Blue area above), or it takes more than the number of times set by Correction Attempt **SLA[axis]**, an error is returned, and the StepNLoop correction process ends.

A convenient way to find the StepNLoop ratio is to set EX=0, PX=0 and move the motor +1000 pulses. The ratio can be calculated by dividing 1000 by the resulting EX value. Note that the value must be positive. If it is not, then the direction polarity must be adjusted. This test should be performed while StepNLoop is disabled.

To enable/disable the StepNLoop feature use the **SL[axis]** command. To read the StepNLoop status, use **SLS[axis]** command. See table SLS command below for a list of the return values.

Value	Description
0	Idle
1	Moving
2	Correcting
3	Stopping

4	Aborting
5	Jogging
6	Homing
7	Z-Homing
8	Correction range error. To clear this error, use CLR command.
9	Correction attempt error. To clear this error, use CLR command.
10	Stall Error. D value has exceeded the correction range value. To clear this error, use CLR command.
11	Limit Error
12	N/A (i.e. StepNLoop is not enabled)

SLS command

While StepNLoop is enabled, position move commands are in term of encoder position. For example, **X1000** means to move the motor to encoder position 1000. This applies to individual as well as interpolated moves.

Additionally, once StepNLoop is enabled, the speed is in encoder speed. For example, **HSPD=1000** when StepNLoop is enabled means that the target high speed is 1000 encoder counts per second. This only applies to individual axis moves.

For linear interpolated movement, the speed is calculated as pulse/sec, NOT encoder counts/sec. The same will apply to the X and Y axis while performing arc/circular interpolated moves.

ASCII	SL[axis]	SLR[axis]	SLT[axis]	SLE[axis]	SLA[axis]	SLS[axis]
Standalone	SL[axis]	-	-	-	-	SLS[axis]

ASCII	D[axis]	CLR[axis]
Standalone	-	-

Operating procedure

If there is no change in previous set data, operation 1 below is not necessary.

- 1) Set each setting value of StepNLoop Closed Loop Control operation.
- 2) Enable the StepNLoop Closed Loop Control operation function with the **SL** command.
- 3) Execute a positioning operation.
- 4) A StepNLoop Closed Loop Control operation is executed after the positioning operation is completed.

Operating conditions

- Joystick operation is invalid
- Buffer operation is invalid
- The error status is clear

Related commands

[SL](#), [SLR](#), [SLT](#), [SLE](#), [SLA](#), [SLS](#)

2.6 Status monitoring function

Status Monitoring Function

The Commander core has a number of registers that can be queried at any time for the current system status.

In this section

- [Counter values](#)
 - o [Output Pulse Counter](#)
 - o [Feedback Pulse Counter](#)
 - o [Deviation Counter](#)
- [Axis status register](#)
 - o [Axis Status Acquisition Command](#)
 - o [Pulse Speed register](#)
 - o [Error clear command](#)

2.6.1 Counter Values

Counter Values

The Commander core has three counters for each axis that can be read to obtain the current number of output pulses, encoder pulses received, and the deviation between the output and input.

In this section

- [Output Pulse Counter](#)
- [Feedback Pulse Counter](#)
- [Deviation Counter](#)

2.6.1.1 Output Pulse Counter

Output Pulse Counter

This counter shows the total number of pulses that have been to the motor driver. It can be

read using the ASCII command **PP**, which returns the pulse count of all 4 axes. The return value has the following format:

[Pulse X]:[Pulse Y]:[Pulse Z]:[Pulse U]

The pulse count of each axis can also be accessed individually. To manually set/get the pulse position of an individual axis, use the **P[axis]** command.

When StepNLoop closed-loop control is enabled, the pulse counter commands return the real-time target position of the motor. When StepNLoop closed-loop control is disabled, the pulse position commands return the step position.

See [2.5.3 StepNLoop Closed Loop Control operation](#) for more information.

2.6.1.2 Feedback Pulse Counter

Feedback Pulse Counter

This counter shows the total number of pulses that have been received on the encoder input. It can be read using ASCII command **EP**, which returns the encoder counts of all 4 axes. The return value has the following format:

[Encoder X]:[Encoder Y]:[Encoder Z]:[Encoder U]

The encoder count of each axis can also be accessed or set individually, by using the **E[axis]** command.

2.6.1.3 Deviation Counter

Deviation Counter

This counter is enabled with StepNLoop operations, and shows the difference between commanded position and the encoder input. To get the deviation counts of an individual axis, use the **D[axis]** command. When the StepNLoop function is disabled, the value of **D[axis]** will be 0.

ASCII	P[axis]	PP	E[axis]	EP	D[axis]
Standalone	P[axis]	-	E[axis]	-	-

Operating procedure

There is no particular condition.

Related commands

[P](#), [PP](#), [E](#), [EP](#), [D](#)

2.6.2 Axis status register

Axis Status Register

The Commander core has two axis-monitoring registers for each axis; these can be read to obtain the current status of each axis.

In this section

- [Axis Status Acquisition Command](#)
- [Pulse Speed register](#)
- [Error clear command](#)

2.6.2.1 Axis Status Acquisition Command

Axis Status Acquisition Command

Motor/axis status register can be read anytime using **MST[axis]** command. The reply contains information about the motor status, including error status (such as end limit stop, alarm stop, and EMG stop). Value of the motor status is relayed as an integer value with the following bit assignment:

Bit	Function	Condition	
0	Accelerating	0: Not Accelerating	1: Accelerating
1	Decelerating	0: Not Decelerating	1: Decelerating
2	Constant Speed	0: Not at Constant Speed	1: At Constant Speed
3	Alarm Signal Input Status	0: OFF	1: ON
4	Positive End Limit Status	0: OFF	1: ON
5	Negative End Limit Status	0: OFF	1: ON
6	Home or Origin Status	0: OFF	1: ON
7	Slowdown Input Status	0: OFF	1: ON
8	Positive End Limit Error*	0: No Error	1: Error
9	Negative End Limit Error*	0: No Error	1: Error
10	Alarm Error*	0: No Error	1: Error
11	In-Position Input Status	0: OFF	1: ON
12	Deviation Counter Clear	0: OFF	1: ON
13	Z-index Input Status	0: OFF	1: ON
14	External Status Input	0: OFF	1: ON
15	EMG Signal Status	0: OFF	1: ON
16	EMG Error Status*	0: No Error	1: Error
17	Stop by Slowdown Detection [□]	0: Not Stopped	1: Stopped
18	Wait for In-Position Input Status	0: Not Waiting	1: Wait

19	Wait for External Start Signal Input	0: Not Waiting	1: Wait
20	External Stop Input ***	0: OFF	1: ON

MST command

Additional information on this register is available in the [Command Reference](#).

Footnote:

* For Bits 8, 9, 10, and 16 (+ end limit, - end limit, alarm, and EMG errors), errors must be canceled with the CLR command before next operation.

□ For Bit 17 (Slowdown Detection), additional operations can be performed with flag ON (1) but the flag will remain ON until canceled by CLR command.

*** bit 20 is valid for firmware version V301 and later version. Older versions will always return a 0 at bit 20.

2.6.2.2 Pulse Speed register

Pulse Speed Register

The current pulse rate register can be read using the command **PS[axis]**. For units, see the table below.

Operation Mode	Speed Units
StepNLoop disabled	Pulse / sec
ALL interpolated moves	Pulse / sec
StepNLoop enabled and non-interpolated move	Encoder counts / sec

PS command units

See section [2.5.3 StepNLoop Closed Loop Control operation](#) for more information.

2.6.2.3 Error clear command

Error Clear Command

Many errors in the Commander core will set flags to alert to the fact that an error occurred. These error flags can be cleared with use of the **CLR** ASCII command. For standalone mode, the **ECLEAR[axis]** command should be used.

This command clears the error flags one axis at a time; there is no command to clear all flags from all axes at once. These commands only clear the error flags, they do not clear the cause of the error.

ASCII	MST[axis]	CLR	PS[axis]
Standalone	MST[axis]	ECLEAR[axis]	PS[axis]

Operation procedure

There is no particular procedure.

Related commands

[MST](#), [CLR](#), [PS](#)

2.7 Standalone program Specification

Standalone Program Specification

The Commander core supports standalone programming, which allows the controller to execute a user-defined program that is stored in its internal memory. The standalone program can be run independently of USB and serial communication, or while communication is active.

In this section

- [Program Development](#)
- [Conditional loop control](#)
- [Multi-thread](#)
- [Subroutines](#)
- [Error Handling](#)
- [Standalone Variables](#)
- [Compiling, Saving, Downloading](#)
- [Standalone Control](#)
- [Standalone Status](#)
- [Standalone Run on Boot-Up](#)

2.7.1 Program Development

Program Development

Commander core utilizes A-Script, a BASIC-like language that helps the programmer quickly write programs for motion control projects, including conditional statements ([IF](#), [ELSE](#), [ELSEIF](#), [ENDIF](#), [WHILE](#), [ENDWHILE](#)), subroutines, and mathematical and logical expressions ([=](#), [!=](#), [<](#), [>](#), [<=](#), [>=](#), [+](#), [-](#), [*](#), [/](#)). Single-line **comments** are available and start with ; and continue until the end of the line. The programming syntax uses a series of standalone commands, listed in section [5.2 Standalone Command Set](#), that allows the programmer to build single and multi-thread programs that can be compiled and run in the controller. By writing lines of code using the set of commands that make up the A-Script language, set-up, control of motion, management of inputs and outputs, and error handling can all be addressed. A basic program consists of a start line (**PRGxx**) and an end line (**END**). All code is written between the PRG and END lines. Use the Utility software to write, compile, download, upload, and execute programs.

2.7.2 Conditional loop control

Conditional Loop Control

A-Script supports a number of conditional statements that will perform different actions for different [conditions](#). A-Script supports the following conditional statements: **IF/ELSEIF/ELSE** loop and **WHILE** loop. Table Conditions below shows the available conditions.

Symbo l	Meaning
=	Equals
!=	Not Equal
>	Greater than
<	Less than
>=	Greater than or Equals
<=	Less than or Equals

Conditions

Examples of using conditional loop control in a program are found in sections [5.4.3 Standalone Example Program 3 – Single Thread](#), through [5.4.10 Standalone Example Program 10 – Multi-thread with subroutine](#).

2.7.3 Multi-thread

Multi-thread

The Commander core utilizes four independent processors that can run simultaneously. Therefore, it is possible to run multiple programs at the same time with or without coordination.

While multiple programs may be running independently, global variable settings can cause unintended consequences between individual programs. As an example, a thread that changes the position mode to incremental could inadvertently cause the position moves of a second thread to respond incorrectly, if it was based on absolute position mode. As such it is recommended that all motion activities be written in one thread, which will prevent unintended interactions between axes, utilizing the other threads to manage I/O and other monitoring operations.

[5.4.8 Standalone Example Program 8 – Multi-thread](#), through [5.4.10 Standalone Example Program 10 – Multi-thread with subroutine](#) contain examples of using multi-threads in a program.

2.7.4 Subroutines

Subroutines

Up to 32 (0-31) subroutines can be called upon. Subroutines are independent program snippets that can be accessed based on the needs of the programmer. Typically, subroutines are used to perform functions that are repeated throughout the operation and can be used in conjunction with logic statements related to motor actions or the status of inputs and outputs.

Sections [5.4.6 Standalone Example Program 6 – Single Thread with subroutine](#), [5.4.9 Standalone Example Program 9 – Multi-thread with subroutine](#), and [5.4.10 Standalone Example Program 10 – Multi-thread with subroutine](#) contain examples of using a subroutine in a program.

Subroutine 31 is used specifically for error handling and is executed whenever an error occurs.

Note: subroutines can be shared by both standalone programs and the ASCII command set. Once a subroutine is written into the flash memory, it can be recalled via ASCII communication using the **GS** command. Standalone programs can also jump to a subroutine using the **GOSUB** command. The subroutines are referenced by their subroutine number [**SUB 0- SUB 31**]. If a subroutine number is not defined, the controller returns with an error.

2.7.5 Error Handling

Error Handling

Subroutine 31 (**SUB 31**) is used specifically for error handling and is executed whenever an error occurs. If **SUB 31** is not defined, the program will cease execution and go into an error state.

How errors are addressed can be coded in SUB 31. Conditional statements that check for specific error conditions can execute additional operations, depending on the error conditions.

Typically, the code within subroutine 31 will contain the standalone command **ECLEAR[axis]** in order to clear the current error. Sections [5.4.7 Standalone Example Program 7 – Single Thread with error handling](#), [5.4.9 Standalone Example Program 9 – Multi-thread with subroutine](#), and [5.4.10 Standalone Example Program 10 – Multi-thread with subroutine](#) contain examples of using subroutine 31 to perform error handling.

The return jump from subroutine 31 are determined by the ASCII command **SAP**. Write a **0** to this setting to have the standalone program jump back to the last performed line. Write a **1** to this setting to have the standalone program jump back to the first line of the program.

Value	Description
0	Return to last excited line
1	Return to first line of program

SAP command

2.7.6 Standalone Variables

Standalone Variables

The Commander core has 100 32-bit signed standalone variables available for general purpose use. They can be used to perform basic calculations and support integer operations. The **V[0-99]** command can be used to access the specified variables. The syntax for all available

[operations](#) can be found in the table below. Note that these operations can only be performed in standalone programming.

Operator	Description	Example
+	Integer Addition	V1=V2+V3
-	Integer Subtraction	V1=V2-V3
*	Integer Multiplication	V1=V2*V3
/	Integer Division (round down)	V1=V2/V3
%	Modulus	V1=V2%5
>>	Bit Shift Right	V1=V2>>2
<<	Bit Shift Left	V1=V2<<2
&	Bitwise AND	V1=V2&7
	Bitwise OR	V1=V2 8
~	Bitwise NOT	V1=~V2

Operations

All variables can be accessed and written with the ASCII **V[0-99]** command. Variables V0-V49 will be initialized to zero on power up, and variables V50 through V99 can be stored to flash memory using the STORE command.

2.7.7 Compiling, Saving, Downloading

Compiling, Saving, Downloading

Standalone programs can be written to the Commander core using the Windows GUI. Refer to the Software Manual for details. Once a standalone program is written by the user, it is then compiled, saved for future reference and downloaded to the Commander core. Each line of written standalone code creates 1-4 assembly lines of code after compilation. See the software manual for more information.

2.7.8 Standalone Control

Standalone Control

The Commander core supports the simultaneous execution of four standalone programs. All programs can be controlled by the **SACTRL[0-3]=[control]** ASCII command and the **SR[0-3]=[control]** standalone command. Program 0 will use command **SACTRL0**, Program 1 will use command **SACTRL1**, and so on. The following [control] assignments can be used for the standalone control command.

Value	Description

0	Stop standalone program
1	Start standalone program
2	Pause standalone program
3	Continue standalone program

SACTRL command

2.7.9 Standalone Status

Standalone Status

The **SASTAT[0-3]** command can be used to determine the current status of the specified standalone program. The table below details the returned values of this command.

Value	Description
0	Idle
1	Running
2	Paused
3	N/A
4	Error

SASTAT command

The **SPC[0-3]** command can also be used to find the current assembled line that the specified standalone program is executing. Note that the return value of the **SPC[0-3]** command is referencing the assembly language line of code and does not directly transfer to the pre-compiled user generated code. The return value can range from [0-1799].

2.7.10 Standalone Run on Boot-Up

Standalone Run on Boot-Up

Standalone can be configured to run on boot-up using the **SLOAD** command. Once this command has been issued, the **STORE** command will be needed to save the setting to flash memory. It will take effect on the following power cycle. See description in the table below for the bit assignment of the **SLOAD** setting.

Bit	Description
0	Standalone Program 0
1	Standalone Program 1

2	Standalone Program 2
3	Standalone Program 3

SLOAD command

ASCII	SACTRL[0-3]	SASTAT[0-3]	SPC[0-3]	GS[0-31]	V[0-99]	SLOAD
Standalone	SR[0-3]	-	-	GOSUB[0-31]	V[0-99]	-

ASCII	SAP	STORE
Standalone	-	STORE

3.0 Communication Interface

Communication Interface

The Commander core has a number of communication methods available. The default method is USB communication. There are also a number of available Serial Communication methods, including UART, I²C, and SPI.

In this section

- [USB Communication](#)
 - [Typical USB Setup](#)
 - [HID Communication API](#)
- [Serial Communication](#)
 - [Communication Port Settings](#)
 - [ASCII Protocol](#)
- [Identification Number](#)
- [Windows GUI](#)
 - [Commander Control Software](#)
 - [Nippon Pulse Terminal Utility](#)

3.1 USB Communication

USB Communication

The Commander core is a HID device and is USB 2.0 compliant. No additional USB drivers are needed to communicate with the Commander core. All USB communication is done using an ASCII command protocol.

In this section

- [Typical USB Setup](#)
- [HID Communication API](#)
 - [How to use](#)

- o [GetCommanderHIDnumber](#)
- o [GetCommanderHIDName](#)
- o [OpenCommanderHID](#)
- o [CloseCommanderHID](#)
- o [SendReceiveCommanderHID](#)
- o [Communication Error Codes](#)

3.1.1 Typical USB Setup

Typical USB Setup

When using a breakout board, the Commander core can be connected to a PC directly via USB or through a USB hub, allowing for multiple units to connect and communicate at once. Nippon Pulse suggest that all USB cables should have noise suppression chokes, also known as ferrite beads, to avoid communication loss or interruption.

When designing your own PCB, please follow the layout guidelines as outlined by the USB 2.0 specifications. D+/D- traces should be the same length, maintain a nominally 90Ω differential impedance, and have a continuous ground plane directly beneath the D+/D- traces and extend at least five times the spacing width to either side of the D+/D- traces.



3.1.2 HID Communication API

HID Communication API

Communication between the PC and Commander core is done using the Windows-compatible DLL API functions shown below. Windows programming languages that use a DLL, such as Visual BASIC, Visual C++, Java, Python, LabVIEW, etc. can be used to communicate with the Commander core.

Typical communication transaction time for sending a command from the PC and getting a reply from the controller using the **SendReceiveCommanderHID** API function is in single-digit milliseconds. This value will vary with the CPU speed and type of command requested.

The DLL file **CMDHidApi.dll** (32bit) or **CMDHidApi64.dll** (64bit) will be needed to access these API functions. These DLL files are not included with this product, please download them from our website.

For HID communication, the following DLL API functions are provided:

Function name	Description
GetCommanderHIDnumber()	Return the number of Commander core devices connected to the host
GetCommanderHIDName(int index, char* name)	Obtain the device name string of a device connected to host

OpenCommanderHID(HANDLE* pHandle, char* name)	Obtain the HANDLE* to operate a device
CloseCommanderHID(HANDLE* pHandle)	Close the specified HANDLE*
SendReceiveCommanderHID(HANDLE* pHandle)	Communication between the host and the device

DLL/API functions

In this section

- [How to use](#)
- [GetCommanderHIDnumber](#)
- [GetCommanderHIDName](#)
- [OpenCommanderHID](#)
- [CloseCommanderHID](#)
- [SendReceiveCommanderHID](#)
- [Communication Error Codes](#)

3.1.2.1 How to use

Use HID Communication API

1. Obtain the number of Commander core HID devices connected to the PC with **GetCommanderHIDnumber**.
2. Get the device name string of the number (index) specified by **GetCommanderHIDName**.
3. Get the HANDLE* to manipulate the Commander core with the device name string specified by **OpenCommanderHID**.
4. Communicate with the Commander core using **SendReceiveCommanderHID** by specifying the acquired HANDLE*.
5. After sending the command string, the reply string will return.
 - a. For the command definition and the returned serial number string, refer to the Command Reference.
6. Close the HANDLE* with **CloseCommanderHID** when finished with all operations.

3.1.2.2 GetCommanderHIDnumber

GetCommanderHIDnumber

Overview

Used to get total number of Commander core HID devices connected to the host. The number of devices is represented in the return value. The function will return 0 if no devices are connected. This function shows that the Commander core is correctly recognized by the host.

Argument

None

Reply

Int type

- 0 No recognized device
- 1 One device was recognized
- η η pcs of device are recognized

Example

```
(1) VC
    int numDev;                                // The number of devices
    numDev = GetCommanderHIDnumber();           // Obtain the number of devices connected
to the host

(2) VB
    Dim numDev as Integer                       ' The number of devices
    numDev = GetCommanderHIDnumber()           ' Obtain the number of devices connected
to the host
```

3.1.2.3 GetCommanderHIDName**GetCommanderHIDName****Overview**

Obtains the device name string of a Commander core HID device by its index number within the detected number of devices. The first detected device is indexed as 0, the second as 1, and so on. Will return 1 if successful, 0 if unsuccessful.

Argument

- int number Specify the index number of the device to get name for.
- char* name Device name

The device name will be "**CMD-4CR-xx**" where **xx** is the Identification Number (**ID**) of the device.

Reply

- Int type
- 0 Send Receive failed
- 1 Send Receive success
- Other (-value) Host communication failed ([3.1.2.7 Communication Error Codes](#))

Example

```
(1) VC
    int result;                                // Function result
    char devName[16];                          // Device name
    result = GetCommanderHIDName(0, devName);   // Obtain the device name

(2) VB
    Public devName As New String("", 15)        ' Device name
    Dim result as Integer                       ' Function result
```

```
result = GetCommanderHIDName(0, devName);    ' Obtain the device name
```

3.1.2.4 OpenCommanderHID

OpenCommanderHID

Overview

Opens a handle to a Commander core HID device by its device name string. Returns 1 if successful, 0 if unsuccessful.

Execute this for restart after disconnecting with the function, CloseCommanderHID.

Argument

HANDLE* pHandle	Device handle
char* name	Device name string

The device name string is acquired by the function, GetCommanderHIDName

Reply

Int type	
0	Send Receive failed
1	Send Receive success
Other (-value)	Host communication failed (3.1.2.7 Communication Error Codes)

Example

```
(1) VC
int result;                                // Function result
HANDLE devHandle;                          // Device handle
char devName[16];                          // Serial number string
result = OpenCommanderHID(devHandle, devName); // Open device, Obtain
handle

(2) VB
Public devHandle As Integer                 ' Device handle
Public devName As New String("", 15)       ' Serial number string
Dim result as Integer                      ' Function result
result = OpenCommanderHID(devHandle, devName); ' Open device, Obtain
handle
```

3.1.2.5 CloseCommanderHID

CloseCommanderHID

Overview

Closes a HANDLE* previously opened on a Commander core HID device. The handle cannot be used for communication once this function is called. Returns 1 if successful, 0 if unsuccessful.

It is recommended that you close all handles when you end the program.

Argument

HANDLE* pHandle Device handle

The device handle is acquired by the function, **OpenCommanderHID**

Reply

Int type

0 Send Receive failed

```
1      Send Receive success
```

Other (-value) Host communication failed ([3.1.2.7 Communication Error Codes](#))

Example

(1) VC

```
int result; // Function result
```

```
HANDLE devHandle;           // Device handle
```

```
result = CloseCommanderHID (devHandle); // Close the handle
```

(2) VB

Public devHandle As Integer ' Device handle

Dim result as Integer ' Function result

```
result = CloseCommanderHID (devHandle); ' Close the handle
```

3.1.2.6 SendReceiveCommanderHID

SendReceiveCommanderHID

Overview

Sends a command to the Commander core HID device through the open path handle, which is specified in the parameter "pHandle" and reads the response from the device. The command to be sent must be passed as a string in parameter "command." The response string received from the device is stored in "reply." Returns 1 if successful, 0 if unsuccessful.

The command and reply parameters must be 63 bytes in length.

Argument

HANDLE* pHandle Device handle

The device handle is acquired by the function, **OpenCommanderHID**

char* command Command

Requires the storage capacity for 63 bytes

```
char* reply      Reply data
```

Requires the storage capacity for 63 bytes

Reply

Int type

0 Send Receive failed

1 Send Receive success

Other (-value) Host communication failed (3.1.2.7 Communication Error

Codes)

Example

```
(1) VC
int result; // Function result
HANDLE devHandle; // Device handle
char cmdStr[64]; // Command
string
char replyStr[64]; // Reply string

strcpy(cmdStr, "PX"); // Specify a
command string
result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr); // Execute the
communication

(2) VB
Public devHandle As Integer ' Device handle
Dim result as Integer ' Function result
Dim cmdStr As New String("", 63) ' Command
string
Dim replyStr As New String("", 63) ' Reply string

cmdStr = PX ' Specify a command
string
result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr); ' Execute the
communication
```

3.1.2.7 Communication Error Codes

SendReceiveCommanderHID

Table API error codes below is a list of possible error codes from the API and their meanings.

Returned Value	Description
-1	API to HID, timeout error
-2	API to HID, communication failed with the device
-3	Invalid function
-4	Device not found
-5	Communication path not found
-6	Communication refused by the device
-7	Invalid handle

-8	Not enough memory
-9	Out of memory
-10	In protect status
-11	Write error
-12	Read out error
-13	Invalid parameter
-14	Open failed
-15	Response was received from the device with an unsupported report ID
-16	Response of incorrect size was received from the device
-255	Unknown error during communication

API error codes

3.2 Serial Communication

Serial Communication

The Commander core has the ability to communicate over a serial communication interface using an ASCII protocol. Various serial interfaces can be

In this section

- [Communication Port Settings](#)
- [ASCII Protocol](#)

3.2.1 Communication Port Settings

Communication Port Settings

The Commander core has the communication port settings shown in table Serial port settings below.

Parameter	Setting
Byte Size	8 bits
Parity	None
Flow Control	None
Stop Bit	1

Serial port settings

Commander core provides the user with the ability to set the desired baud rate of the serial communication. In order to make these changes, first set the desired baud rate by using the **DB** command. By default, the Commander core has a baud rate setting of 9600 bps.

Return Value	Description
1	9600
2	19200
3	38400
4	57600
5	115200

DB command

To write the values to the device's flash memory, use the **STORE** command. The new baud rate will now be written to memory, but will not go into effect until after a complete power cycle.

3.2.2 ASCII Protocol

ASCII Protocol

The following ASCII protocol should be used for sending commands and receiving replies from the Commander core. For details on valid ASCII commands, refer to Section [4.0 ASCII Language Specification](#) or the Command Reference.

Sending Command

ASCII command string in the format of:

@[DeviceName][ASCII Command][CR] '[CR] character has ASCII code 13

Receiving Reply

The response will be in the format of:

[Response][CR] '[CR] character has ASCII code 13

Example

For querying the polarity of the X-axis

Send: @01POLX[CR]

Reply: 7[CR]

For reading the digital input status

Send: @01DI[CR]

Reply: 8[CR]

For performing a store to flash memory

Send: @01STORE[CR]

Reply: OK[CR]

3.3 Identification Number

Identification Number

The Commander core gives each device a unique identification number, allowing up to 64 devices to be connected to the same host. If multiple devices are connected to the host, the

identification number for each device should be unique.

By default, all Commander cores are shipped from the factory with identification number 01. The current identification number of a device can be found by reading the **ID** command. Also, the device name will always include the current identification number. For USB communication the device name will be "CMD-4CR-xx" where xx is the identification number of the device. For serial communication, only the identification number of the device is used.

In order to change the identification number of the device, first store the desired number using the **ID** command. Note that this value must be within the range [00-99].

For example, the command **ID=02** can be sent to change the identification number to 02. To save a modified identification number to the flash memory of the device, use the **STORE** command. The new identification number will not take effect until after a power cycle.

3.4 Windows GUI

Windows GUI

Nippon Pulse has made two Windows GUI programs available for use with the Commander core series at no cost. These are available via download from our website.

1. Commander Control Software
2. Nippon Pulse Terminal Utility

Commander Control Software

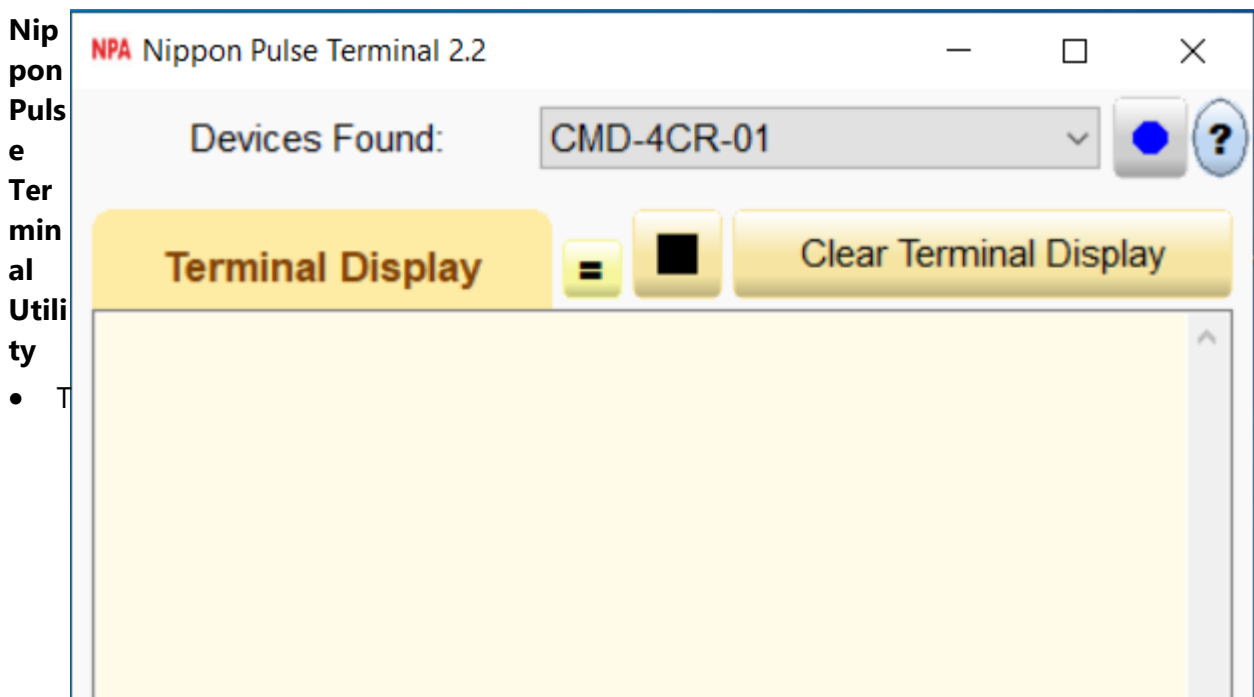
- The Commander Control Software can be downloaded from this link > V202a 

The Commander Control Software is a basic Windows GUI program that serves two main functions:

- (1) Provides a shortcut to every ASCII command to assist in testing and troubleshooting the setup of the Commander core and your own interface code.
- (2) Allows you to test, program, compile, download and debug the controller using the A-script standalone language.



See the [CMD-4CR Software Manual](#) for details.



he Nippon Pulse Terminal Utility can be downloaded from this link > Portable V2.49-P > Installed V2.49

The Nippon Pulse Terminal Utility is intended for advanced users. It works with all Nippon Pulse controllers. It has three main functions:

- (1) Terminal mode allows for quick testing of individual ASCII commands. It also shows total cycle timing of each ASCII command.
- (2) ASCII Programs allows the creation of ASCII programs to test of the proposed code.
- (3) Auto Control Programs provides the ability to write scripts that fully control of the terminal utility, including running of ASCII programs and logging the results.

Terminal mode

ASCII commands can be sent directly to the controller from the Command input box. ASCII commands can be entered one at a time or ASCII commands can be entered as a string of commands. For a string, each command must be separated by a comma.

ASCII program mode

The Nippon Pulse terminal program has the ability to run a list of ASCII commands and compare the reply to an expected reply. This feature supports all ASCII commands for the connected controller, as well as a few additional commands.

This allows for easy testing of command sequences, along with the replies that are expected.

- o If the actual reply matches the expected reply, the command passes.
- o If the actual reply does not match the expected reply, the command fails.
- o The actual reply must match the expected result exactly (including case) in order to pass. Anything else is a fail.
 - Numeric replies have additional options for comparison.
- o Passed commands are displayed normally
- o Failed commands are marked with an exclamation point, and the received and expected results are shown.

Auto Control Programs

This feature is very powerful, with a script format that is very simple and a robust set of commands that allows the following key functions to be performed automatically:

- o Setting and controlling the Nippon Pulse Terminal Program startup defaults.
- o Control of Auto control script.
- o Run ASCII programs.
- o Store results and create Pass/Fail logs based on ASCII programs.
- o Accept user inputs, Store and display Variables, Constants, Date, Time, and ASCII reply.

See the [Nippon Pulse Terminal Program Software Manual](#) for details.

3.4.1 Commander Control Software

Software Utility - Commander Control Software Manual

3.4.2 Nippon Pulse Terminal Utility

4.0 ASCII Language Specification

ASCII Language Specification

Important Note: All the commands described in this section are interactive ASCII commands and are not analogous to standalone commands. Refer to Section [5.0 Standalone Language Specification](#).

For more detailed information on any command, please refer to the Command Reference.

Invalid commands are returned with a "?". Always check for a proper reply when a command is sent. For USB and RS-485 communication, the commands detailed in section [ASCII Command Set](#) are valid.

In this section

- [ASCII Command Set](#)
- [Error Codes](#)
- [Examples of command control from a PC](#)
 - o [Start and stop of jog move](#)
 - o [Position Move](#)
 - o [Obtaining and clearing the counter value](#)
 - o [Homing Operation](#)
 - o [Linear interpolation operation](#)
 - o [Circular Interpolation Operation](#)
 - o [On-the-fly speed change](#)
 - o [Synchronization Function](#)
 - o [Control of general purpose input/output](#)

4.1 ASCII Command Set

ASCII Command Set

Command	R/ W	Description	Reference Chapter
ABORT	W	Immediately stops all motion for all axes	2.2.3
ABORT[axis]	W	Immediately stops motion in [axis]	
ABS	W	Turns on absolute coordination mode	2.2.4.1
ACC	R	Returns current global acceleration value in ms	2.1.4

Command	R/ W	Description	Reference Chapter
ACC=[value]	W	Sets global acceleration value in ms	
ACC[axis]	R	Returns current [axis] acceleration value in ms	
ACC[axis]=[value]	W	Sets [axis] acceleration value in ms	
AI[ch]	R	Returns analog input status for [ch] in mV	2.1.3.3.2
ARC[A1][A2]N[C1]:[C2]:[Θ]	W	Arc interpolation with axes [A1], [A2], centers at [C1], [C2], CCW direction to [Θ]	2.3.2.1
ARC[A1][A2]P[C1]:[C2]:[Θ]	W	Arc interpolation with axes [A1], [A2], centers at [C1], [C2], CW direction to [Θ]	
ARCTN[C1]:[C2]:[Θ]:[Z target]	W	X-Y Arc interpolation, centers at [C1], [C2], CCW direction to [Θ] with Z linear interpolation to [target]	2.3.3
ARCTP[C1]:[C2]:[Θ]:[Z target]	W	XY Arc interpolation, centers at [C1], [C2], CW direction to [Θ] with Z linear interpolation to [target]	
BF	W	Disable buffered move	2.3.4
BO	W	Enable buffered move	2.3.4
BSTART	W	Start processing the buffer (buffer must be enabled)	2.3.4
BSTAT	R	Status of the buffer	2.3.4
CIR[A1][A2]N[C1]:[C2]	W	Circular interpolation with axes [A1], [A2], centers at [C1], [C2], CCW direction	2.3.2
CIR[A1][A2]P[C1]:[C2]	W	Circular interpolation with axes [A1], [A2], centers at [C1], [C2], CW direction	
CIRTN[C1]:[C2]:[Z target]	W	X-Y circular interpolation, centers at [C1], [C2], CCW direction with Z linear interpolation to [target]	2.3.3
CIRTP[C1]:[C2]:[Z target]	W	X-Y circular interpolation, centers at [C1], [C2], CW direction with Z linear interpolation to [target]	
CLR[axis]	W	Clears limit, alarm status bit, and StepNLoop errors for [axis]	2.1.3.1.1.1 2.1.3.1.1.2 2.1.3.1.2 2.5.3 2.6.2.3

Command	R/ W	Description	Reference Chapter
D[axis]	R	Returns StepNLoop value for [axis]	2.5.3 2.6.1.3
DB	R	Returns baud rate	2.1.7
DB=[value]	W	Sets baud rate to [value]	
DEC	R	Returns the current global deceleration value in ms	2.1.4
DEC=[value]	W	Sets global deceleration value in ms	
DEC[axis]	R	Returns current [axis] deceleration value in ms	2.1.4
DEC[axis]=[value]	W	Sets [axis] deceleration value in ms	
DI	R	Returns 4-bits of general-purpose digital inputs	2.1.3.2.1
DI[in]	R	Returns bit status of general digital input [in]	2.1.3.2.1
DIP	R	Returns the digital input polarity	2.1.3.2.1
DIP=[pol]	W	Sets the digital input polarity [pol]	2.1.6.3 2.1.7
DO	R	Returns 4-bits of general-purpose digital outputs.	2.1.3.2.2
DO=[value]	W	Sets 4-bits of general-purpose digital outputs.	
DO[out]	R	Returns bit status of general digital output [out]	2.1.3.2.2
DO[out]=[value]	W	Sets bit of general-purpose digital output [out]	
DOBOOT	R	Returns the DO boot-up state	2.1.3.2.2
DOBOOT=[value]	W	Sets the DO boot-up state	2.1.5 2.1.7
DOP	R	Returns the digital output polarity	2.1.3.2.2
DOP=[pol]	W	Sets the digital output polarity [pol]	2.1.6.3 2.1.7
E[axis]	R	Returns the encoder value of [axis]	2.6.1.2
E[axis]=[value]	W	Sets the encoder value of [axis]	
EINT	R	Returns the interpolation setting	2.1.7
EINT=[value]	W	Enables/disables interpolation	2.2.4.2 2.3.1

Command	R/ W	Description	Reference Chapter
EO	R	Returns 4-bits of enable output value	2.1.2.1
EO=[value]	W	Sets 4-bits of enable output value	2.1.2.2
EO[en]	R	Returns bit of enable output [en]	2.2.1
EO[en]=[value]	W	Sets bit of enable output [en]	
EOBOOT	R	Returns EO boot-up state	2.1.5
EOBOOT=[value]	W	Sets EO boot-up state	2.1.7 2.2.1
EP	R	Returns the encoder value for all 4 axes	2.6.1.2
ERC[axis]	R	Returns the enable status of the deflection counter clear output	2.1.2.2 2.1.3.1.1.1
ERC[axis]=[value]	W	Enables/disables the deflection counter clear output	2.1.6 2.1.6.1 2.1.7 2.2.5
ERCD[axis]	R	Returns the pulse delay set value	2.1.6
ERCD[axis]=[value]	W	Sets the pulse delay value (2-bits)	2.1.6.1 2.1.7 2.1.2.2
ERCP[axis]	R	Returns the pulse width set value	2.1.6
ERCP[axis]=[value]	W	Sets the pulse width value (3-bits)	2.1.6.1 2.1.7 2.1.2.2
ESTOP	W	Performs an emergency stop (same as triggering CEMG)	2.1.3.1.2
EXST	R	Returns the status of the external start feature	2.1.1.2 2.1.6
EXST=[value]	W	Enable/disable the external start feature	2.1.6.2 2.1.7
GS[sub]	W	Call a defined subroutine [sub] (value 0 – 31)	2.7.4
H[axis][+/-][mode]	W	Home [axis] in positive or negative direction using homing mode [mode] (see Appendix A for detailed information on homing modes)	2.2.5 Appendix A
HSPD	R	Returns the global high speed setting	2.1.4
HSPD=[value]	W	Sets the global high speed value	

Command	R/ W	Description	Reference Chapter
HSPD[axis]	R	Returns the high speed setting for [axis]	2.1.4
HSPD[axis]=[value]	W	Sets the high speed value for [axis]	
I[X target]:[Y target]:[Z target]:[speed]	W	Writes an XYZ interpolated move to buffer. Sets each axis move target and desired move speed.	2.3.4
ID	R	Returns the Commander's ID	2.1.1.1
ID=[value]	W	Sets the Commander's ID. Range is 00 to 99. Use STORE command to save.	2.1.7
IERR	R	Returns the ignore limit/alarm error status	2.1.3.1.1.1
IERR=[value]	W	Sets the ignore limit/alarm status	2.1.3.1.2
INC	W	Turns on incremental coordinate mode	2.1.6
INP[axis]	R	Returns enable/disable status of positioning complete signal for [axis]	2.1.7
INP[axis]=[value]	W	Enables/disables the positioning complete signal for [axis]	2.2.4.1
IO	R	Returns the status of all configurable I/O ports (32-bit)	2.1.2.2
IO=[value]	W	Sets the bit status of all configurable outputs	
IO[port]	R	Returns the status of configurable I/O at [port]	2.1.6
IO[port]={0,1}	W	Sets the value of configurable output bit at [port]	
IOBOOT	R	Returns the status of all general purpose I/O at startup (32-bit)	2.1.7
IOBOOT=[value]	W	Sets the value of all configurable outputs at startup	
IOCFG	R	Returns the current configuration of the configurable I/O ports	2.1.3.3.1
IOCFG=[value]	W	Sets the configuration of the configurable I/O ports	
IOP	R	Returns the configurable I/O polarity setting (2-bit logic)	2.1.5
IOP=[0,1]	W	Sets the configurable I/O polarity (2-bit logic)	

Command	R/ W	Description	Reference Chapter
J[axis][+/-]	W	Jogs the motor in the [+] or [-] direction	2.2.2
JDEL[axis]	R	Returns the maximum allowable speed change with joystick control for [axis]	2.1.3.3.2 2.1.7
JDEL[axis]=[value]	W	Sets the maximum allowable speed change with joystick control for [axis]	2.5.1.2
JENA	R	Returns status for joystick control	2.1.3.3.2
JENA=[value]	W	Enables/disables joystick control	2.1.7 2.5.1
JLIM[number]	R	Returns the current operating behavior limit setting of joystick operation	2.1.3.3.2 2.1.7 2.5.1.3
JLIM[number]=[value]	W	Sets the operating behavior limit value of joystick operation	
JMAX[axis]	R	Returns the maximum voltage setting of the input terminal for joystick operation of [axis]	2.1.3.3.2 2.1.7 2.5.1.1
JMAX[axis]=[value]	W	Sets the maximum terminal voltage of [axis] for joystick operation	
JMIN[axis]	R	Returns the minimum voltage setting of the input terminal for joystick operation of [axis]	2.1.3.3.2 2.1.7 2.5.1.1
JMIN[axis]=[value]	W	Sets the minimum terminal voltage of [axis] for joystick operation	
JSPD[axis]	R	Returns the maximum speed setting of [axis] for joystick operations	2.1.3.3.2 2.1.7
JSPD[axis]=[value]	W	Sets the maximum speed of [axis] for joystick operations	2.5.1.2
JTOL[axis]	R	Returns the current dead band around mid-voltage range of [axis] for joystick operation	2.1.3.3.2 2.1.7 2.5.1.1
JTOL[axis]=[value]	W	Sets the dead band range around mid-voltage of [axis] for joystick operation	
LSPD	R	Returns global low speed setting	2.1.4
LSPD=[value]	W	Sets global low speed value	
LSPD[axis]	R	Returns low speed setting for [axis]	2.1.4
LSPD[axis]=[value]	W	Sets low speed value for [axis]	
LT[axis]=[value]	W	Enables/disables latch feature	2.1.3.2.1

Command	R/ W	Description	Reference Chapter
LTE[axis]	R	Returns the previously latched encoder position	2.1.3.2.1
LTP[axis]	R	Returns the previously latched step position	2.1.3.2.1
LTS[axis]	R	Returns the current latch status	2.1.3.2.1
MM	R	Returns the current move mode for moves (absolute or incremental)	2.2.4.1
MP[axis]	R	Returns the current manual pulse generator position	2.1.3.1.4 2.5.2.2
MP[axis]=[values]	W	Sets the manual pulse generator position	
MPD[axis]	R	Returns the divider parameter for manual pulse generator operation	2.1.3.1.4 2.1.7 2.5.2 2.5.2.2
MPD[axis]=[value]	W	Sets the divider parameter for manual pulse generator operation	
MPE[axis]	R	Returns the manual pulse generator enable status	2.1.3.1.4 2.1.7 2.5.2
MPE[axis]=[value]	W	Enable/disable the manual pulse generator	
MPM[axis]	R	Returns the multiplier parameter for manual pulse generator operation	2.1.3.1.4 2.1.7 2.5.2 2.5.2.2
MPM[axis]=[value]	W	Sets the multiplier parameter for manual pulse generator operation	
MST[axis]	R	Returns all motor status, buffer move status, and move mode status for [axis]	2.1.3.1.1.1 2.1.3.1.1.2 2.1.3.1.2 2.6.2.1
P[axis]	R	Returns the position value for [axis]	2.6.1.1
P[axis]=[value]	W	Sets the position value for [axis]	

Command	R/ W	Description	Reference Chapter
POL[axis]	R	Returns polarity parameter for [axis] (see Table 6.9)	2.1.2.1 2.1.2.2
POL[axis]=[pol]	W	Sets the polarity parameter for [axis]	2.1.2.3 2.1.3.1.1.2 2.1.6 2.1.6.3 2.1.7 2.2.1 2.5.2 2.5.2.1
PP	R	Returns current pulse counter value of all 4 axes	2.6.1.1
PS[axis]	R	Returns the current pulse speed for [axis]	2.6.2.2
PWM[1-2]	R	Returns current PWM duty cycle setting (in %)	2.1.3.3.3
PWM[1-2]=[value]	W	Set the PWM duty cycle (in %)	
SACTRL[prgNo]=[0-3]	W	Controls the standalone [prgNo]: stop(0), start(1), pause(2), continue(3)	2.7.8
SAP	R	Returns the return jump line (standalone error handling)	2.7.5
SAP[0-1]	W	Sets the return jump line (standalone error handling)	
SASTAT[0-3]	R	Returns status for standalone program	2.7.9
SCV[axis]	R	Returns S-curve control status for [axis]	2.1.4 2.1.6
SCV[axis]=[0,1]	W	Enable/disable S-curve acceleration for [axis]. Trapezoidal accel/decel used if disabled.	
SDC[axis]	R	Returns the configuration of the Slow Down decel operation for [axis]	2.1.3.1.1.2 2.1.6 2.1.7
SDC[axis]=[value]	W	Sets the configuration for the Slow Down decel (decel only or decel and stop) for [axis]	
SDE[axis]	R	Status of the Slow Down decel operation for [axis]	2.1.3.1.1.2 2.1.6 2.1.7
SDE[axis]=[value]	W	Enable/disable the external deceleration feature for [axis]	

Command	R/ W	Description	Reference Chapter
SL[axis]	R	Returns status of StepNLoop enable for [axis]	2.1.7 2.5.3
SL[axis]=[value]	W	Enable/Disable StepNLoop feature for [axis]	
SLA[axis]	R	Returns maximum attempts to complete StepNLoop operation for [axis]	2.1.7 2.5.3
SLA[axis]=[value]	W	Sets maximum attempts to complete StepNLoop operation for [axis] 0 = infinite number of correction tries.	
SLE[axis]	R	Returns error range of StepNLoop operation for [axis]	2.1.7 2.5.3
SLE[axis]=[value]	W	Sets error range of StepNLoop operation for [axis]	
SLOAD[prg#]	R	Returns the RunOnBoot parameter for [prg#]	2.1.5 2.1.7 2.7.10
SLOAD[prg#]=[value]	W	Sets the RunOnBoot parameter for [prg#]	
SLR[axis]	R	Returns the StepNLoop ratio for [axis] (ppr/cpr)	2.1.7 2.5.3
SLR[axis]=[value]	W	Sets the StepNLoop ratio for [axis] (ppr/cpr)	
SLS[axis]	R	Returns StepNLoop status for [axis]	2.5.3
SLT[axis]	R	Returns StepNLoop tolerance for [axis]	2.1.7 2.5.3
SLT[axis]=[value]	W	Sets StepNLoop tolerance for [axis]	
SPC[prg#]	R	Returns program counter for [prg#]	2.7.9
SSPD[axis]=[value]	W	Performs on-the-fly speed change for [axis]. Buffer must be disabled	2.4.1
SSPDM[axis]	R	Returns the current on-the-fly speed mode	2.4.1
SSPDM[axis]=[value]	W	Sets the on-the-fly speed mode	
STOP	W	Performs ramp down to low speed and stop if motor is moving for all axes	2.2.3
STOP[axis]	W	Performs ramp down to low speed and stop if motor is moving for [axis]	

Command	R/ W	Description	Reference Chapter
STORE	W	Saves parameters to flash memory	2.1.1.1 2.1.3.2.2 2.1.3.3.1 2.1.5 2.1.6 2.1.7 2.2.1 2.7.6
SYNC[axis]	R	Returns SYNC condition and source	2.1.3.2.2
SYNC[axis]=[value]	W	Sets SYNC condition and source [axis]	
SYNF[axis]	W	Disable synchronization window and synchronization output	2.1.3.2.2
SYNMAX[axis]	R	Returns the current maximum value of the Sync Pulse Window for [axis]	2.1.3.2.2
SYNMAX[axis]=[value]	W	Sets the maximum value of the Sync Pulse Window for [axis]	
SYNMIN[axis]	R	Returns the current minimum value of the Sync Pulse Window for [axis]	2.1.3.2.2
SYNMIN[axis]=[value]	W	Sets the minimum value of the Sync Pulse Window for [axis]	
SYNO[axis]	W	Enable synchronization output	2.1.3.2.2
SYNP[axis]	R	Returns the synchronization position value or distance between pulses for continuous sync	2.1.3.2.2
SYNP[axis]=[value]	W	Sets the synchronization position value or distance between pulses for continuous sync	
SYNS[axis]	R	Returns the status of the synchronization feature	2.1.3.2.2
SYNWF[axis]	W	Disables the synchronization window for [axis]	2.1.3.2.2
SYNWO[axis]	W	Enables the synchronization window and synchronization output feature for [axis]	2.1.3.2.2
T[axis][newtarget]	W	Sets the on-the-fly target position change for [axis]	2.4.2
V[var]	R	Returns the standalone [var] value	2.1.7
V[var]=[value]	W	Sets the standalone [var] value	2.7.6

Command	R/ W	Description	Reference Chapter
VER	R	Returns the current firmware version	2.1.1.1
[axis][targetpos]	W	Individual move command for [axis] to [targetpos]	2.2.4.2
X[targetX]Y[targetY] Z[targetZ]U[targetU]	W	Interpolation move command (include all axes that will be interpolated). EINT must be enabled (1)	2.2.4.2 2.3.1
ZCNT[axis]	R	Returns the number of Z-index pulses to count for [axis]. Used for relevant moves (ZMOVE and homing routines)	2.1.7 2.2.5.1
ZCNT[axis]=[value]	W	Sets the Z-index count to be used for relevant moves (ZMOVE and homing routines). Range is 1 – 16.	
ZMOVE[axis][dir]	W	Perform the EZ Count operation for [axis] in [dir] positive (+) or negative (-) direction	2.2.5.1

ASCII commands

4.2 Error Codes

Error Codes

If an ASCII command cannot be processed by the Commander core, the controller will reply with an error code. See below for possible error responses:

Error Code	Description
?Unknown Command: [Command]	The ASCII command is not understood by the Commander core.
?Error Moving	The move command could not be processed. One or more of the required axes may be in error state or in motion.
?Error Alarm	The move command could not be processed. The Alarm signal is in the on state.
?Error +Limit	The move command could not be processed. The + End Limit is in the on state.
?Error -Limit	The move command could not be processed. The - End Limit is in the on state.
?Error ESTOP	The move command could not be processed. The EMG signal is in the on state.
?Error SNL	The move command could not be processed. The StepNLoop control is in error.
?Invalid Index	The Index number is invalid

?Program Error	The command cannot be processed because a standalone program is in error state.
?Sub not Initialized	Call to a subroutine using the GS command is not valid because the specified subroutine has not been defined.
?Buffer Full	Move cannot be added to a buffer register because all registers are full.
?Buffer Disabled	Buffer command is not valid because buffer mode is disabled.
?Buffer Enabled	SSPD command cannot be processed because buffer mode is enabled.
?SSPD Error	SSPD command cannot be processed because the SSPD mode is not defined or is out of range. S-curve should also be disabled.
?Override Failed	Override operation failed

4.3 Examples of command control from a PC

Error Codes

If these examples are unclear, or do not provide helpful content, contact us. Your feedback is essential to us.

See the [Command Reference](#) for more examples.

In this section

- [Start and stop of jog move](#)
- [Position Move](#)
- [Obtaining and clearing the counter value](#)
- [Homing Operation](#)
- [Linear interpolation operation](#)
- [Circular Interpolation Operation](#)
- [On-the-fly speed change](#)
- [Synchronization Function](#)
- [Control of general purpose input/output](#)

4.3.1 Start and stop of jog move

Start and stop of jog move

Summary of operation

Set the speed of the X-axis movement and turn on excitation. After that, start the movement of the X axis in the negative direction. After a while, decelerate and stop the axis.

Program code

```
cmdStr = "LSPDX=500"
```


result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Starting speed of X-axis set to 500 pps

cmdStr = "HSPDX=5000"

result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Moving speed of X-axis set to 5,000 pps

cmdStr = "ACCX=200"

result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Accel time of X-axis set to 200 ms

cmdStr = "DECX=200"

result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Decel time of X-axis set to 200 ms

cmdStr = "EO1=1"

result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Enable X-axis

cmdStr = "JX–"

result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Start moving in the (–) direction

cmdStr = "STOPX"

result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Decelerate and stop

4.3.2 Position Move

Position Move

Summary of operation

Set the global moving speed and turn on the excitation of X and Y axes. Set positioning coordinates to absolute mode. Move the X axis to position –20,000 and Y axis to position 15,000.

Program code

cmdStr = "LSPD=500"

result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Starting speed set to 500 pps (Global)

cmdStr = "HSPD=5000"

result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Moving speed set to 5,000 pps (Global)

cmdStr = "ACC=200"

result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Accel time set to 200 ms
(Global)

cmdStr = "DEC=200"

result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Deceleration time set to 200
ms (Global)

cmdStr = "EO=3"

result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Enable X and Y-axis

cmdStr = "ABS"

result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Set position coordinate
mode to Absolute

cmdStr = "X-20000"

result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Start moving the X-axis to
the position of -20,000

cmdStr = "Y15000"

result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Start moving the Y-axis to
the position of 15,000.

4.3.3 Obtaining and clearing the counter value

Obtaining and Clearing the Counter Value

Summary of operation

Read and display the command pulse and feedback pulse counter on the X-axis. The counter value can be cleared only when the axis is normally stopped (stopped without error). Read the counter value in a timer event.

Program code

cmdStr = "PX"

result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Obtain the command pulse
counter value on X-axis

OutplsX.Text = replyStr 'Display the command
pulse counter value

cmdStr = "EX"

result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Obtain the feedback
pulse counter value on X-axis

FeedbackX.Text = replyStr 'Display the feedback
counter value

cmdStr = "MSTX"

result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Obtain the status value of X-
axis

mStatusX.Text = replyStr 'Display the status
value for X-axis

Execute the above process repeatedly in the timer event

If mStatusX.Text = "0" Then 'Does X-axis stop normally?
 cmdStr = "PX=0"
 result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'X-axis command
 pulse counter clear
 cmdStr = "EX=0"
 result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'X-axis feedback pulse
 counter clear

4.3.4 Homing Operation

Homing Operation

Summary of operation

Set the Y-axis moving speed and enable axis. Then, start the Y-axis homing operation in the negative direction using homing mode 1.

Program Code

```
cmdStr = "LSPDY=500"
result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Starting speed of Y-axis set
to 500 pps
cmdStr = "HSPDY=5000"
result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Moving speed of Y-axis set
to 5,000 pps
cmdStr = "ACCY=200"
result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Accel time of Y-axis set to
200 ms
cmdStr = "DECY=200"
result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Decel time of Y-axis set to
200 ms
cmdStr = "EO2=1"
result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Enable Y-axis
cmdStr = "H-1"
result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Start the homing operation
of Y-axis
```

4.3.5 Linear interpolation operation

Linear Interpolation Operation

Summary of operation

Set global moving speed and enable the X- and Y-axes. Set the positioning coordinate mode to absolute. Execute the linear interpolation operation, moving the X-axis to position -20,000 and the Y-axis to position 15,000.

Program code

```
cmdStr = "LSPD=500"
result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Starting speed set to 500
pps (Global)
cmdStr = "HSPD=5000"
result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Moving speed set to 5,000
pps (Global)
cmdStr = "ACC=200"
result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Accel time set to 200 ms
(Global)
cmdStr = "DEC=200"
result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Decel time set to 200 ms
(Global)
cmdStr = "EO=3"
result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Enable X and Y-axes
cmdStr = "ABS"
result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Set position coordinate
mode to absolute
cmdStr = "EINT=1"
result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Interpolation operation
enabled
cmdStr = "X-20000Y15000"
result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) ' Start interpolation
operation of X and Y axes
```

4.3.6 Circular Interpolation Operation

Circular Interpolation Operation

Summary of operation

Set global moving speed and enable the X and Y axes. Set positioning coordinate mode to

absolute. Execute a circular interpolation move. The center coordinates of circle are (1000, 0) and the rotation direction clockwise.

Program code

```
cmdStr = "LSPD=500"
result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Starting speed set to 500
pps (Global)

cmdStr = "HSPD=5000"
result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Moving speed set to 5,000
pps (Global)

cmdStr = "ACC=200"
result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Accel time set to 200 ms
(Global)

cmdStr = "DEC=200"
result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Decel time set to 200
ms (Global)

cmdStr = "EO=3"
result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Enable X and Y axes

cmdStr = "ABS"
result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Set position coordinate
mode to absolute

cmdStr = "EINT=1"
result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Interpolation operation
enabled

cmdStr = "CIRXYP1000:0"
result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Start interpolation operation
of X and Y axes
```

4.3.7 On-the-fly speed change

On-the-fly Speed Change

Summary of operation

Change the X-axis speed at 50,000 pps to 70,000 pps during motion.

Program code

```
cmdStr = "LSPDX=1000"
result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Starting speed of X axis set
to 500 pps

cmdStr = "HSPDX=50000"
```

```
result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Moving speed of X axis set to 50,000 pps
```

---omitted steps---

```
cmdStr = "JX--"
```

```
result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Start moving in the - direction
```

```
cmdStr = "SSPDM=2"
```

```
result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Double the speed range
```

```
cmdStr = "SSPD=70000"
```

```
result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Update speed to 70,000 pps
```

4.3.8 Synchronization Function

Synchronization Function

Summary of operation

When the X axis moves to position 5,000, turn on synchronization signal DO1/SYNCx.

Program code

```
cmdStr = "LSPDX=100"
```

```
result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Starting speed of X axis set to 100 pps
```

```
cmdStr = "HSPDX=1000"
```

```
result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Moving speed of X axis set to 1,000 pps
```

---omitted steps---

```
cmdStr = "SYNCX=1"
```

```
result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Set X-axis sync condition
```

```
cmdStr = "SYNPX=5000"
```

```
result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Set X-axis comparison data value to 5,000
```

```
cmdStr = "SYNO"
```

```
result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Sync output configuration
```

enabled

```
cmdStr = "X5000"
```

```
result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Move X axis to position
5,000
```

'DO1/SYNCx will be on when
movement is completed

4.3.9 Control of general purpose input/output

Control of General Purpose Input/Output

Summary of operation

Turn ON/OFF general-purpose output 19 in synchronization with general-purpose input 1.

Program code

```
cmdStr = "IOCFG=262144" 'Configure general-purpose
IO19 to output all other IO to input
```

```
result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr)
```

---omitted steps---

```
cmdStr = "IO1"
```

```
result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr) 'Read status of general-
purpose input 1
```

```
If replyStr = "1" Then
```

```
    cmdStr = "IO19=1" 'It is ON
```

```
    result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr)
```

```
Else
```

```
    cmdStr = "IO19=0" 'It is not ON
```

```
    result = SendReceiveCommanderHID(devHandle, cmdStr, replyStr)
```

```
End If
```

Note: The state of the general-purpose input must be read repeatedly

5.0 Standalone Language Specification

Standalone Language Specification

Note: Commands described in this section are standalone language commands and are not analogous to ASCII commands. Refer to Section [4.0 ASCII Language Specification](#) for more information about ASCII commands.

For more information on any command, see [Command Reference](#).

This topic is broken into the following chapters:

- [Using A-Script programming interface](#)
 - [Program Development](#)
 - [Subroutines](#)
 - [Error Handling](#)
 - [Multi-thread](#)
 - [Compiling, Saving, Downloading](#)
- [Standalone Command Set](#)
- [Error Codes](#)
- [Example Standalone Programs](#)
 - [Standalone Example Program 1 – Single Thread](#)
 - [Standalone Example Program 2 – Single Thread with buffer](#)
 - [Standalone Example Program 3 – Single Thread](#)
 - [Standalone Example Program 4 – Single Thread](#)
 - [Standalone Example Program 5 – Single Thread](#)
 - [Standalone Example Program 6 – Single Thread with subroutine](#)
 - [Standalone Example Program 7 – Single Thread with error handling](#)
 - [Standalone Example Program 8 – Multi-thread](#)
 - [Standalone Example Program 9 – Multi-thread with subroutine](#)
 - [Standalone Example Program 10 – Multi-thread with subroutine](#)

5.1 Using A-Script programming interface

Using A-Script Programming Interface

The Commander core supports standalone programming, called A-Script, which allows the controller to execute a user-defined program that is stored in its internal memory. See section [2.7 Standalone program Specification](#) for more information.

In this section

- [Program Development](#)
- [Subroutines](#)
- [Error Handling](#)
- [Multi-thread](#)
- [Compiling, Saving, Downloading](#)

5.1.1 Program Development

Program Development

A-Script is a BASIC-like language that can help the programmer quickly write programs for motion control projects, including conditional statements (IF, ELSE, ELSEIF, ENDIF, WHILE, ENDWHILE), subroutines, and mathematical and logical expressions (=, !=, <, >, <=, >=, +, -, *, /). The programming syntax uses a series of standalone commands (listed later in this section) that allow the programmer to build single and multi-thread programs that can be compiled and run in the controller. By writing lines of code using the set of commands, setup, control of motion, management of inputs and outputs, and error handling can all be addressed. A basic program consists of a start line (PRG xx) and an end line (END). All code is written between the PRG and END lines. Use the utility software to write, compile, download, upload, and execute programs.

5.1.2 Subroutines

Subroutines

Subroutines are program snippets that can be accessed based on the needs of the programmer, and are used in conjunction with logic statements related to motor actions or status of inputs and outputs. Up to 32 (0-31) subroutines can be called upon. See section [2.7.4 Subroutines](#) for more information.

5.1.3 Error Handling

Error Handling

Subroutine 31 is used for error handling. See section [2.7.5 Error Handling](#) for more information.

5.1.4 Multi-thread

Multi-thread

The Commander core utilizes four independent processors that can run simultaneously, making it possible to run multiple programs at the same time with or without coordination. See section [2.7.3 Multi-thread](#) for more information.

5.1.5 Compiling, Saving, Downloading

Compiling, Saving, Downloading

Standalone programs can be written to the Commander core using the Windows GUI. Refer to the Software Manual for details. Once a standalone program is written by the user, it is then compiled, saved for future reference and downloaded to the Commander core. Each line of

written standalone code creates 1-4 assembly lines of code after compilation. See the software manual for more information.

5.2 Standalone Command Set

Command	R/W	Description	Reference Chapter
;	-	Any text following semicolon in the same line appears as a comment	2.7.1
ABORT	W	Immediately stops all motion for all axes	2.2.3
ABORT[axis]	W	Immediately stops motion in [axis]	
ABS	W	Turns on absolute position coordinate mode	2.2.4.1
ACC	R/W	Returns/sets global acceleration value in ms	2.1.4
ACC[axis]	R/W	Returns/sets [axis] acceleration value in ms	
AI[ch]	R	Returns analog input status for [ch] in mV	2.1.3.3.2
ARC[A1][A2]N[C1]:[C2]:[Θ]	W	Arc interpolation with axes [A1], [A2], centers at [C1], [C2], CCW direction to [Θ]	2.3.2.1
ARC[A1][A2]P[C1]:[C2]:[Θ]	W	Arc interpolation with axes [A1], [A2], centers at [C1], [C2], CW direction to [Θ]	
ARCTN[Cx]:[Cy]:[Θ]:[pos]	W	X-Y arc interpolation, centers at [C1], [C2], CCW direction to [Θ] with Z linear interpolation to [pos]	2.3.3
ARCTP[Cx]:[Cy]:[Θ]:[pos]	W	X-Y arc interpolation, centers at [C1], [C2], CW direction to [Θ] with Z linear interpolation to [pos]	
BUFOFF	W	Disable buffered move	2.3.4
BUFON	W	Enable buffered move	2.3.4
CIR[A1][A2]N[C1]:[C2]	W	Circular interpolation with axes [A1], [A2], centers at [C1], [C2], CCW direction	2.3.2
CIR[A1][A2]P[C1]:[C2]	W	Circular interpolation with axes [A1], [A2], centers at [C1], [C2], CW direction	
CIRTN[C1]:[C2]:[pos]	W	X-Y circular interpolation, centers at [C1], [C2], CCW direction with Z linear interpolation to [pos]	2.3.3
CIRTP[C1]:[C2]:[pos]	W	X-Y circular interpolation, centers at [C1], [C2], CW direction with Z linear interpolation to [pos]	
DEC	R/W	Returns/sets the global deceleration value in ms	2.1.4
DEC[axis]	R/W	Returns/sets [axis] deceleration value in ms	
DELAY	W	Sets delay in ms	
DI	R	Return status of digital inputs (4-bit)	2.1.3.2.1
DI[1-4]	R	Get individual bit status of digital inputs. Will return [0,1].	2.1.3.2.1
DO	R/W	Returns/sets 4 bits of general-purpose digital outputs	2.1.3.2.2
DO[1-4]	R/W	Returns/sets bit status of general digital output [1-4]	2.1.3.2.2

Command	R/W	Description	Reference Chapter
E[axis]	R/W	Returns/sets the encoder value of [axis]	2.6.1.2
ECLEAR[axis]	W	Clear any motor status errors.	2.1.3.1.1.1 2.1.3.1.1.2 2.1.3.1.2 2.6.2.3 2.7.5
EINT	R/W	Returns/sets the interpolation Enable/Disable status	2.1.7 2.2.4.2 2.3.1
EO	R/W	Returns/sets 4 bits of enable output value for all axes	2.1.2.1 2.1.2.2
EO[1-4]	R/W	Returns/sets enable output value for [1-4]	2.2.1
GOSUB[0-31]	-	Executes subroutine [0-31]	2.7.4
HOME[axis][+/-][mode]	W	Home [axis] in positive or negative [+/-] direction using homing mode [mode] (see Appendix A: Homing Mode Actions)	2.2.5
HSPD	R/W	Returns/sets the global high-speed setting	2.1.4
HSPD[axis]	R/W	Returns/sets the high-speed setting for [axis]	
IF ELSEIF ELSE ENDIF	-	Executes the instruction following the IF statement when condition is true. Execute the instruction following the next ELSE or ELSEIF statement when the condition is false. When all conditions are false, continue executing program after ENDIF. The following operands are used in the conditional IF statement: =, >, <, >=, <=, !=	2.7.2
INC	W	Turns on incremental coordinate mode	2.2.4.1
IO	R/W	Returns/sets the status of all configurable I/O ports (32-bit)	2.1.3.3.1
IO[index]	R/W	Returns/sets the status of configurable I/O at [index]	2.1.3.3.1
ISTART	W	Start processing the buffer (buffer must be enabled)	2.3.4
JOG[axis][+/-]	W	Jogs the [axis] in the [+] or [-] direction using the HSPD setting	2.2.2
JOYENA	R/W	Returns/sets the status for joystick control	2.1.3.3.2 2.5.1
JOYHS[axis]	R/W	Returns/sets the maximum speed setting of [axis] for joystick operations	2.1.3.3.2 2.5.1.2
LSPD	R/W	Returns/sets global low speed setting	2.1.4
LSPD[axis]	R/W	Returns/sets low speed setting for [axis]	2.1.4
MST[axis]	R	Returns all motor status, buffer move status, and move mode status for [axis]	2.1.3.1.1.1 2.1.3.1.1.2 2.1.3.1.2 2.6.2.1
P[axis]	R/W	Returns/sets the position value for [axis]	2.6.1.1

Command	R/W	Description	Reference Chapter
PRG[0-3] END	-	Defines the first and last line of a standalone program [0-3].	2.7.1
PS[axis]	R	Returns the current pulse speed for [axis]	2.6.2.2
SL[axis]	R/W	Returns/sets enable/disable of StepNLoop for [axis]	2.5.3
SLS[axis]	R	Returns StepNLoop status for [axis]	2.5.3
SR[0-3]	W	Controls standalone program [0-3]. Use command to stop, start, pause, continue.	2.7.8
SSPD[axis]	W	Performs on-the-fly speed change for [axis]. Buffer must be disabled.	2.4.1
SSPDM[axis]	W	Sets the on-the-fly speed mode for [axis]	2.4.1
STOP	W	Performs ramp down to low speed and stop if motor is moving for all axes	2.2.3
STOP[axis]	W	Performs ramp down to low speed and stop if motor is moving for [axis]	
STORE	W	Saves parameters to flash memory	2.1.1.1 2.1.3.2.2 2.1.3.3.1 2.1.5 2.1.6 2.1.7 2.2.1 2.7.6
SUB[0-31] ENDSUB	-	Used to define the first and last line of a Subroutine [0-31]	2.7.4
SYNCFG[axis]	R/W	Returns/sets SYNC condition and source [axis]	2.1.3.2.2
SYNOFF[axis]	W	Disable synchronization window and output [axis]	2.1.3.2.2
SYNON[axis]	W	Enable synchronization output [axis]	2.1.3.2.2
SYNPOS[axis]	R/W	Sets/returns the synchronization position value or distance between pulses for continuous sync [axis]	2.1.3.2.2
SYNSTAT[axis]	W	Returns the status of the synchronization feature [axis]	2.1.3.2.2
V[var]	R/W	Returns/sets the variable [var] value. [var] is number 0-99.	2.7.6
WAIT[axis]	W	Waits for completion of [axis] motion before next command is executed	2.1.1.2 2.6.2.1
WHILE ENDWHILE	-	Continues executing the program after the WHILE statement when the conditional statement is true. Repeat until WHILE condition is false, and then execute program after ENDWHILE. Conditional operands available include: =, >, <, >=, <=, !=	2.7.2
X[target]	W	Individual move command for X axis to [target]	2.2.4.2

Command	R/W	Description	Reference Chapter
Y[target]	W	Individual move command for Y axis to [target]	
Z[target]	W	Individual move command for Z axis to [target]	
U[target]	W	Individual move command for U axis to [target]	
X[pos]Y[pos]Z[pos]U[pos]	W	Interpolation move command (include all axes that will be interpolated). EINT must be enabled (1).	2.2.4.2 2.3.1 2.3.4
ZMOVE[axis][dir]	W	Perform the EZ Count operation. Moves [axis] the number of Z-index pulses specified by ZCNT in [dir] positive (+) or negative (-) before stopping.	2.2.5.1

A-Script standalone commands**5.3 Error Codes**

Error Code	Description	Incorrect Code Example
Unknown Error	Program doesn't recognize command	PRGA
BIT Error	Specified bit does not exist	DO10=1
Token Error	Syntax error	DELAY=V1+200
= Error	Incorrect assignment formula	V2+1=V3
Assignment Error	Invalid value specified	V1= ABS
Operation Argument Error	Invalid command specified	V1=V2+ABS
Error not Number of Var	Invalid specification method	DO=PX
Var Range Error	Exceeds the defined variable range	V200=1
Missing =	Assignment operator '=' cannot be found	HSPD*1000
Error GOSUB	Specified value is outside the range for GOSUB	GOSUB900
Error SUB Number	Subroutine number is not valid	GOSUB DI1
Error SUB	Specified value is outside the range for SUB	SUB5000
Error ENDSUB	ENDSUB is missing	SUB 1 ... ENDSUB
Error PRG	Specified value is outside the range for PRG	PRG200
Error PRG Number	Program number is not valid	PRG X13
Missing END	END for PRG missing	PRG 0 ... END
Missing ENDWHILE	ENDWHILE for WHILE missing	WHILE 1=1 ... ENDWHILE
Missing ENDIF	ENDIF for IF missing	IF 1=1 ... ENDIF
Error	Other syntax errors	V1=V2

A-Script error codes**5.4 Example Standalone Programs****Example Standalone Programs**

The following examples are provided to assist your understanding. Although they were produced with the utmost care, if you find any points that are unclear, wrong, or have inadequate descriptions, please let us know.

For more examples, please refer to the command reference.

In this section

- [Standalone Example Program 1 – Single Thread](#)
- [Standalone Example Program 2 – Single Thread with buffer](#)
- [Standalone Example Program 3 – Single Thread](#)
- [Standalone Example Program 4 – Single Thread](#)
- [Standalone Example Program 5 – Single Thread](#)
- [Standalone Example Program 6 – Single Thread with subroutine](#)
- [Standalone Example Program 7 – Single Thread with error handling](#)
- [Standalone Example Program 8 – Multi-thread](#)
- [Standalone Example Program 9 – Multi-thread with subroutine](#)
- [Standalone Example Program 10 – Multi-thread with subroutine](#)

5.4.1 Standalone Example Program 1 – Single Thread**Standalone Example Program 1 – Single Thread**

Download this program:

[StandAloneExample1.PRG](#)

Summary of operation

Set high and low speed and move the X axis to 1000 and then back to 0.

Program code

PRG 0	;Program start line, program 0
HSPD=20000	;Set high speed to 20,000 pulses/sec
LSPD=1000	;Set low speed to 1,000 pulses/sec
ACC=300	;Set acceleration to 300 msec
EO=1	;Enable the X axis
X1000	;Move X axis to 1,000
WAITX	;Wait for X axis move to complete
X0	;Move X axis to 0
WAITX	;Wait for X axis move to complete
END	;End of program 0

5.4.2 Standalone Example Program 2 – Single Thread with buffer

Standalone Example Program 2 – Single Thread with Buffer

Download this program:

[StandAloneExample2.PRG](#)

Summary of operation

Create a contouring profile for the X axis using buffer move function.

Program code

```

PRG 1                                ;Program start line, program 1
    BUFON                            ;Enable buffer move
; Load Buffer
    HSPD=100                        ;Set high speed to 100 pulses/sec
    X1000Y100Z100                    ;I:1000:100:100:100
    HSPD=200                        ;Set high speed to 200 pulses/sec
    X2000Y100Z100                    ;I:2000:100:100:200
    HSPD=300                        ;Set high speed to 300 pulses/sec
    X3000Y100Z100                    ;I:3000:100:100:300
    HSPD=400                        ;Set high speed to 400 pulses/sec
    X4000Y100Z100                    ;I:4000:100:100:400
    HSPD=500                        ;Set high speed to 500 pulses/sec
    X5000Y100Z100                    ;I:5000:100:100:500
    HSPD=400                        ;Set high speed to 400 pulses/sec
    X3000Y100Z100                    ;I:3000:100:100:400
    HSPD=300                        ;Set high speed to 300 pulses/sec
    X0Y100Z100                      ;I:0:100:100:300
; Start buffered move
    ISTART                          ;BSTART
    DELAY=100                       ;Delay 0.1 seconds
    WAITX                           ;Wait until X-axis motion has stopped
BUFOFF                              ;Disable buffer move
END                                 ;End of program 1

```

5.4.3 Standalone Example Program 3 – Single Thread

Standalone Example Program 3 – Single Thread

Download this program:

[StandAloneExample3.PRG](#)

Summary of operation

Move the X axis back and forth indefinitely between position 1000 and 0.

Program code

```
PRG 1                ;Program start line, program 1
    HSPD=20000        ;Set the high speed to 20,000 pulses/sec
    LSPD=1000         ;Set the low speed to 1000 pulses/sec
    ACC=300           ;Set the acceleration to 300 msec
    EO=1              ;Enable the motor power
    WHILE 1=1         ;Forever loop
        X0            ;Move to zero
        WAITX         ;Wait for X-axis move to complete
        X1000         ;Move to 1000
        WAITX         ;Wait for X-axis move to complete
    ENDWHILE          ;Go back to WHILE statement
END                  ;End of program 1
```

5.4.4 Standalone Example Program 4 – Single Thread

Standalone Example Program 4 – Single Thread

Download this program:

[StandAloneExample4.PRG](#)

Summary of operation

Move the X axis back and forth 10 times between position 1000 and 0.

Program code

```
PRG 1                ;Program start line, program 1
    HSPD=20000        ;Set high speed to 20,000 pulses/sec
    LSPD=1000         ;Set low speed to 1,000 pulses/sec
    ACC=300           ;Set acceleration to 300 msec
    EO=1              ;Enable the X-axis
    V1=0              ;Set variable 1 to 0
    WHILE V1<10       ;Loop while variable 1 is less than 10
        X0            ;Move X-axis to zero (0)
        WAITX         ;Wait for X-axis move to complete
        X1000         ;Move X-axis to 1,000
        WAITX         ;Wait for X-axis move to complete
        V1=V1+1       ;Increment variable 1 by 1
    ENDWHILE          ;Go back to WHILE statement
END                  ;End of program 1
```

5.4.5 Standalone Example Program 5 – Single Thread

Standalone Example Program 5 – Single Thread

Download this program:

[StandAloneExample5.PRG](#)**Summary of operation**

Move the X axis back and forth between position 1000 and 0 only if the digital input 1 is turned on.

Program code

```

PRG 2                                ;Program start line, program 2
    HSPD=20000                       ;Set high speed to 20,000 pulses/sec
    LSPD=1000                        ;Set low speed to 1000 pulses/sec
    ACC=300                          ;Set acceleration to 300 msec
    EO=1                             ;Enable the X-axis
    WHILE 1=1                        ;Forever loop
        IF DI1=1                    ;If digital input 1 is on, execute the following commands
            X0                       ;Move X-axis to zero
            WAITX                    ;Wait for X-axis move to complete
            X1000                    ;Move X-axis to 1,000
            WAITX                    ;Wait for X-axis move to complete
        ENDIF                       ;End of the IF loop
    ENDWHILE                         ;Go back to WHILE Statement
END                                  ;End of program 2

```

5.4.6 Standalone Example Program 6 – Single Thread with subroutine

Standalone Example Program 6 – Single Thread with subroutine

Download this program:

[StandAloneExample6.PRG](#)

Summary of operation

Using a subroutine, increment the X axis by 1000 whenever the DI1 rising edge is detected.

Program code

```

PRG 0                                ;Program start line, Program 0
    HSPD=20000                       ;Set high speed to 20,000 pulses/sec
    LSPD=1000                        ;Set low speed to 1000 pulses/sec
    ACC=300                          ;Set acceleration to 300 msec
    ABS                              ;Set move mode to absolute mode
    EO=1                             ;Enable the X-axis
    V1=0                             ;Set variable 1 to 0
    WHILE 1=1                        ;Forever loop
        IF DI1=1                    ;If digital input 1 is on, execute the following commands
            GOSUB 1                 ;Execute subroutine 1
        ENDIF                       ;End of the IF loop
    ENDWHILE                         ;Go back to WHILE statement
END                                  ;End of program 0

```

```

SUB 1                                ;Subroutine start line, subroutine 1
  XV1                                ;Move X-axis to V1 target position
  WAITX                              ;Wait for X-axis move to complete
  V1=V1+1000                         ;Increment V1 by 1,000
  WHILE DI1=1                        ;Wait until the DI1 is turned off (prevent multiple
increments)
    ENDWHILE                          ;Go back to WHILE statement
ENDSUB                              ;End of subroutine 1

```

5.4.7 Standalone Example Program 7 – Single Thread with error handling

Standalone Example Program 7 – Single Thread with error handling

Download this program:

[StandAloneExample7.PRG](#)

Summary of operation

If digital input 1 is on, move to position 2000. If digital input 2 is on, move to position 4000. If digital input 3 is on, move to 6000. If digital input 4 is on, home the motor in negative direction, mode 0. Use digital output 1 to indicate that the motor is moving or not moving. Also, create an error handling subroutine utilizing SUB 31 that clears the error and then returns the motor to position zero.

Program code

```

PRG 2                                ;Start program 2
  HSPD=15000                         ;Set high speed to 15,000 pulses/sec
  LSPD=500                           ;Set low speed to 500 pulses/sec
  ACC=300                            ;Set acceleration to 300 msec
  EO=1                               ;Enable the X-axis
  WHILE 1=1                          ;Forever loop
    IF DI1=1                          ;If digital input 1 is on
      X2000                          ;Move X-axis to 1000
    ELSEIF DI2=1                     ;If digital input 2 is on
      X4000                          ;Move X-axis to 4000
    ELSEIF DI3=1                     ;If digital input 3 is on
      X6000                          ;Move X-axis to 6000
    ELSEIF DI4=1                     ;If digital input 4 is on
      HOMEX-0                        ;Home X-axis with mode 0 in the negative direction
    ENDIF                            ;End of the IF loop
    V1=MSTX                          ;Store the X-axis status to variable 1
    V2=V1&7                          ;Get first 3 bits
    IF V2!=0                          ;If one of first 3 bits is high (X-axis moving)
      DO1=1                          ;Turn on digital output 1
    ELSE                              ;Else if first 3 bits are low (X-axis idle)
      DO1=0                          ;Turn off digital output 1
    ENDIF                            ;End of the IF loop
  ENDWHILE                          ;Go back to WHILE statement

```

```

END                                ;End of program 2
SUB 31                             ;Subroutine start line, subroutine 31 (error handling
subroutine)
    ECLEARX                        ;Clear the current error in X-axis
    X0                             ;Return X-axis to position 0
    DO2=1                          ;Turn on digital output 2
ENDSUB                             ;End of subroutine 31

```

5.4.8 Standalone Example Program 8 – Multi-thread

Standalone Example Program 8 – Multi-thread

Download this program:

[StandAloneExample8.PRG](#)

Summary of operation

Program 0 will continuously move the motor between positions 0 and 1000. Simultaneously, program 1 will control the status of program 0 using digital inputs.

Program code

```

PRG 0                               ;Start program 0
    HSPD=20000                      ;Set speed to 20,000 pulses/sec
    LSPD=500                        ;Set speed to 500 pulses/sec
    ACC=500                          ;Set acceleration to 500 msec
    WHILE 1=1                       ;Forever loop
        X0                          ;Move X-axis to position 0
        WAITX                       ;Wait for X-axis move to complete
        X1000                      ;Move X-axis to position 1,000
        WAITX                      ;Wait for X-axis move to complete
    ENDWHILE                        ;Go back to WHILE statement
END                                 ;End program 0
PRG 1                               ;Start program 1
    WHILE 1=1                       ;Forever loop
        IF DI1=1                   ;If digital input 1 is triggered
            ABORTX                 ;Stop movement in X-axis
            SR0=0                  ;Stop program 0
        ELSE                        ;If digital input 1 is not triggered
            SR0=1                  ;Run program 0
        ENDIF                      ;End of the IF loop
    ENDWHILE                        ;Go back to WHILE statement
END                                 ;End program 1

```

5.4.9 Standalone Example Program 9 – Multi-thread with subroutine

Standalone Example Program 9 – Multi-thread with subroutine

Download this program:[StandAloneExample9.PRG](#)**Summary of operation**

Program 0 starts program 1 and program 2, after which it calls subroutine 1, which homes axis X to the negative limit. Then Program 0 continuously moves the X-axis between positions 0 and 1000. Simultaneously, program 1 is monitoring the X-axis status. Program 1 uses digital output 1 to indicate that the motor is moving or not moving. Program 2 flashes digital output 3, on for 3 seconds, then off for 3 seconds. An error handling subroutine utilizing SUB 31 turns on digital output 2 clears the error and returns the motor to position zero, then waits 1.5 seconds, after which it turns digital output 2 off.

Program code

```

PRG 0                                ;Start of program 0
    SR1=1                            ;Start program 1 running
    SR2=1                            ;Start program 2 running
    SCVX=1                           ;Enable S-Curve for axis X
    ABS                              ;Set move mode to absolute mode
    HSPD=1000                        ;Set high speed to 1,000 pulses/sec
    LSPD=100                         ;Set low speed to 100 pulses/sec
    ACC=200                          ;Set acceleration to 200 msec
    DEC=200                          ;Set deceleration to 200 msec
    EO=1                             ;Enable X-axis
    GOSUB 1                          ;Go to Subroutine 1
    WHILE 1=1                        ;Forever loop
        X0                           ;Move X-axis to position 0
        WAITX                         ;Wait for X-axis move to complete
        DELAY=500                    ;Delay 0.5 seconds
        X1000                        ;Move X-axis to position 1000
        WAITX                         ;Wait for X-axis move to complete
        DELAY=500                    ;Delay 0.5 seconds
    ENDWHILE                         ;Go back to WHILE statement
END                                  ;End program 0
PRG 1                                ;Start of program 1
    WHILE 1=1                        ;Forever loop
        V1=MSTX                      ;Store the X-axis status to variable 1
        V2=V1&7                      ;Get first 3 bits
        IF V2!=0                     ;If one of first 3 bits is high (X-axis moving)
            DO1=1                    ;Turn on digital output 1
        ELSE                          ;Else if first 3 bits are low (X-axis idle)

```

```

DO1=0           ;Turn off digital output 1
ENDIF           ;End of the IF loop
ENDWHILE        ;Go back to WHILE statement
END             ;End program
PRG 2           ;Start of program 2
WHILE 1=1       ;Forever loop
DO3=0           ;Turn off digital output 3
DELAY=3000      ;Delay 3 seconds
DO3=1           ;Turn on digital output 3
DELAY=3000      ;Delay 3 seconds
ENDWHILE        ;Go back to WHILE statement
END             ;End program
SUB 1           ;Subroutine start line, subroutine 1
HOMEX-6         ;Home X-axis with mode 6 in the negative direction
WAITX          ;Wait for X-axis move to complete
ENDSUB          ;End of subroutine 1
SUB 31          ;Subroutine start line, subroutine 31 (error handling
subroutine)
DO2=1           ;Turn on digital output 2
ECLEARX        ;Clear the current error in X-axis
X0             ;Return X-axis to position 0
DELAY=1500      ;Delay 1.5 seconds
DO2=0           ;Turn off digital output 2
ENDSUB          ;End of subroutine 31

```

5.4.10 Standalone Example Program 10 – Multi-thread with subroutine

Standalone Example Program 10 – Multi-thread with subroutine

Download this program:

[StandAloneExample10.PRG](#)

Summary of operation

Program 1 will home X and Y axes, then will continuously move the X and Y axes and control a laser pointer to draw a circle, line, then an arc. Simultaneously, program 3 will control the status of general-purpose digital outputs to control LEDs. Subroutine 1 turns off a laser pointer, Subroutine 2 turns on the laser pointer, Subroutine 7 enables and homes the X and Y axes, and Subroutine 31 handles errors.

Program code

```
PRG 1           ;Start program 1
```

GOSUB 1	;Go to subroutine 1, Turn off laser pointer
GOSUB 7	;Go to subroutine 7, Enable and Home X/Y
WHILE 1=1	;Forever loop
; Draw Circle	
X85610Y562530	;Move to start point for Circle
WAITX	;Wait for X-axis move to complete
GOSUB 2	;Go to subroutine 2, Turn on laser pointer
CIRXYP487659:562431	;Moves XY in a CW circle around points
X487659 Y562431	
WAITX	;Wait for X-axis move to complete
GOSUB 1	;Go to subroutine 1, Turn off laser pointer
; Draw Line	
X100000Y100000	;Move to start point of Line
WAITX	;Wait for X-axis move to complete
GOSUB 2	;Go to subroutine 2, Turn on laser pointer
X909000Y642390	;Linear XY move from X100000 Y100000 to X909000
Y642390	
WAITX	;Wait for X-axis move to complete
GOSUB 1	;Go to subroutine 1, Turn off laser pointer
; Draw Arc	
X430589Y20623	;Move to start point for Arc
WAITX	;Wait for X-axis move to complete
GOSUB 2	;Go to subroutine 2, Turn on laser pointer
ARCXYN288899:267403:95000	;270deg arc of XY CCW around points X288899
Y267403	
WAITY	;Wait for Y-axis move to complete
GOSUB 1	;Go to subroutine 1, Turn off laser pointer
ENDWHILE	;Go back to WHILE statement
END	;End program 1
PRG 3	;Start program 3
WHILE 1 = 1	;Forever loop
IO=64	;General-purpose output 7 on.
DELAY=700	;Delay 0.7 seconds
IO=80	;General-purpose output 5 and 7 on.
DELAY=700	;Delay 0.7 seconds
IO=84	;General-purpose output 3, 5, and 7 on.
DELAY=700	;Delay 0.7 seconds
IO=85	;General-purpose output 1, 3, 5, and 7 on.
DELAY=700	;Delay 0.7 seconds
V30=0	; Set variable 30 to 0.
WHILE V30 < 10	;Loop while variable 30 is less than 10
IO = 85	;General-purpose output 1, 3, 5, and 7 on.
DELAY=50	;Delay 0.05 seconds
IO = 0	;All general-purpose outputs off.
DELAY=50	;Delay 0.05 seconds
V30=V30+1	;Increment variable 30 by 1

ENDWHILE	;Go back to WHILE statement
ENDWHILE	;Go back to WHILE statement
END	;End Program 3
SUB 1	;Subroutine start line, subroutine 1
DO2=0	;DO 2 off to turn off Laser pointer
DELAY=10	;Delay 0.01 seconds to stabilize laser
ENDSUB	;End of subroutine 1
SUB 2	;Subroutine start line, subroutine 2
DO2=1	;DO 2 on to turn on Laser pointer
DELAY=10	;Delay 0.01 seconds to stabilize laser
ENDSUB	;End of subroutine 2
SUB 7	;Subroutine start line, subroutine 7
; Enable motors	
HSPD=15000	;Set high speed to 15,000 pulses/sec
LSPD=500	;Set low speed to 500 pulses/sec
ACC=100	;Set acceleration to 100 msec
EO=3	;Enable X and Y axis
; Home axis	
HOMEX-6	;Home X-axis with mode 6 in the negative direction
WAITX	;Wait for X-axis move to complete
HOMEY-6	;Home Y-axis with mode 6 in the negative direction
WAITY	;Wait for Y-axis move to complete
; Move X/Y to Start Position	
HSPD=35000	;Set high speed to 15,000 pulses/sec
X500000Y500000	;Move X and Y-axis to X500,000 Y500,000
WAITX	;Wait for X-axis move to complete
ENDSUB	;End of subroutine 7
SUB 31	;Subroutine start line, subroutine 31 (error
handling)	
SR1=0	;Stop program 1
SR3=0	;Stop program 3
ABORT	;Abort motion of all axes
ECLEARX	;Clear the current error in X-axis
ECLEARY	;Clear the current error in Y-axis
IO=0	;All general-purpose outputs off.
DO2=0	;DO 2 off to turn off Laser pointer
ENDSUB	;End of subroutine 31

6.0 Dimensions and Connectivity

Dimensions and Connectivity

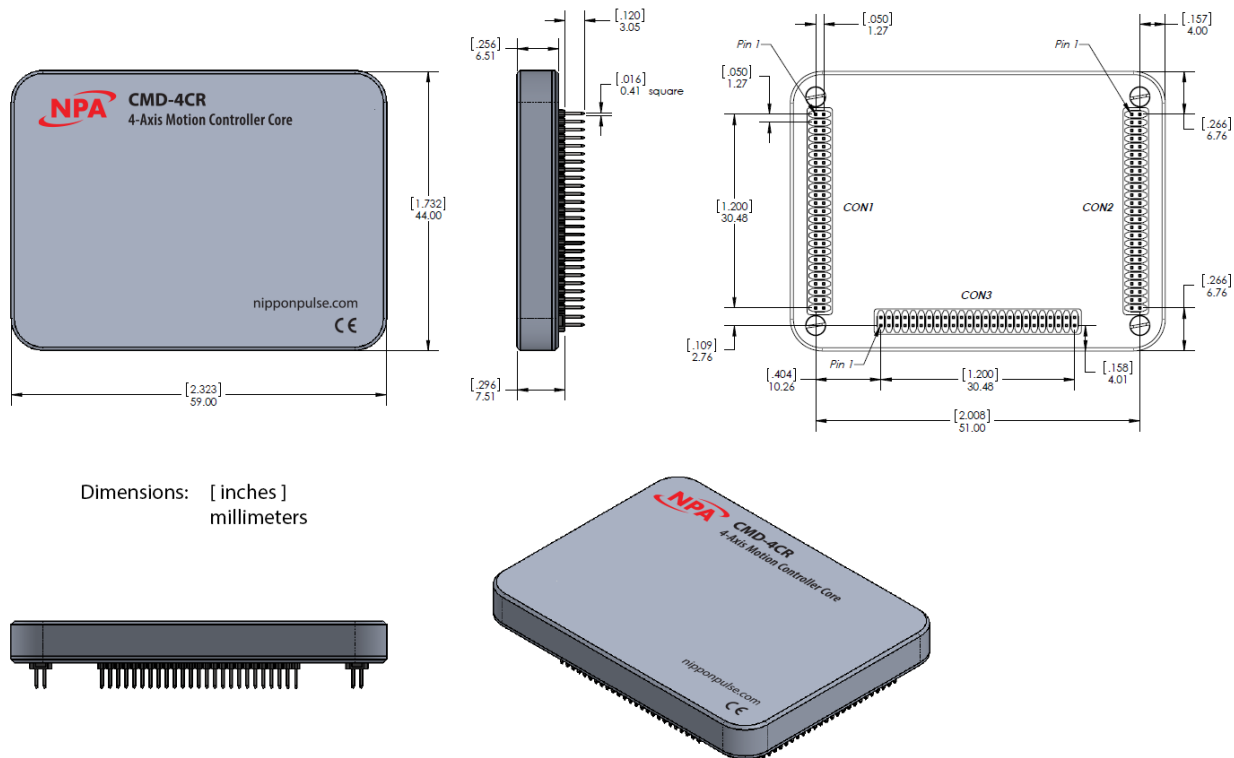
In this section

- [Dimensions](#)
- [Connectivity](#)
 - [Pin-out](#)
 - [Pin Assignment Diagram](#)
 - [Pin Descriptions by Functions](#)
 - [Recommended Footprint](#)
 - [Interface Boards](#)
 - [Development Kit](#)

6.1 Dimensions

Dimensions

All dimensions in mm. Tolerance ± 0.1



External dimensions between generation 4 and generation 5 are the same.

6.2 Connectivity

Connectivity

The Commander CMD-2CR and CMD-4CR cores uses 3 x 50-pin connectors with 1.27mm pitch to allow access to all signals. The CMD-8CR uses 4 x 50-pin connectors with 1.27mm pitch to allow access to all signals.

In this manual – when viewing from the bottom, with the pins closest to the edge, left, right and bottom – we will refer to the left set of pins as CON1, the right set as CON2, and the bottom set as CON3. For the CMD-8CR the the set of pins at the top are CON4. For the

purpose of simplicity, in this manual, all pin numbers are referred to by CON number then pin number in the set. For example, pin 3 on CON1 will be referred to as pin 1-3, whereas pin 25 on CON3 will be referred to as pin 3-25.

Recommended Mating Connector Information

Through hole

Manufacturer: Samtec Inc.

Part Number: SFMC-125-01-L-D

Through hole

Manufacturer: Samtec Inc.

Part Number: FLE-125-01-G-DV-A-K

Other compatible 1.27mm connectors can be used. Figure in Section [6.2.1 Pin-out](#) shows the Pin 1 locations.

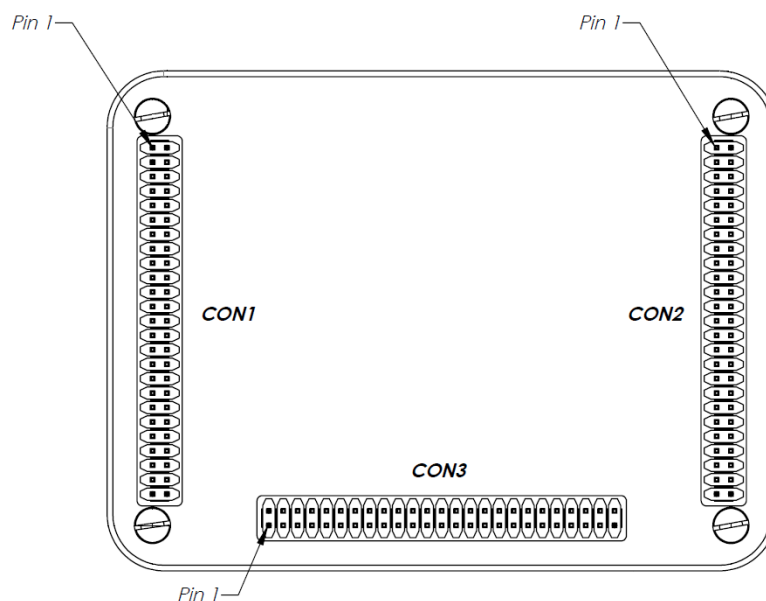
In this section

- [Pin-out](#)
 - o [Pin Assignment Diagram](#)
 - o [Pin Descriptions by Functions](#)
 - o [Recommended Footprint](#)
- [Interface Boards](#)
 - [Development Kit](#)

6.2.1 Pin-out

Pin-out

The unit as viewed from the underside.



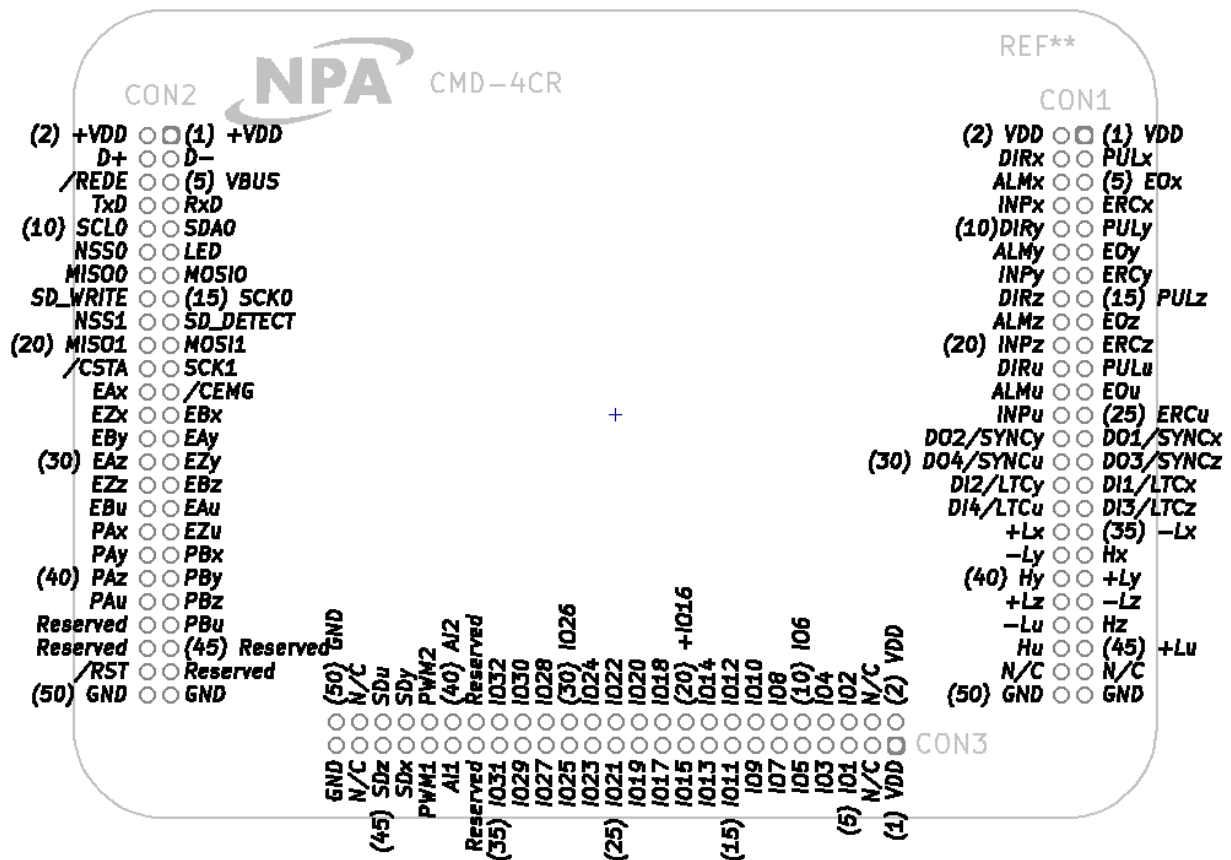
In this section

- [Pin Assignment Diagram](#)
- [Pin Descriptions by Functions](#)
- [Recommended Footprint](#)

6.2.1.1 Pin Assignment Diagram

Pin Assignment Diagram

As viewed from the top, looking through the device. (As laid out on PCB.) You can see how the pins are numbered by looking at CON1 as an example: pin 1 is on the top right, and pin 2 is next to it on the top left; then pin 3 is below pin 1, and on and on. This graphic shows pins 1 and 2, then includes the pin numbers of the multiples of five (5, 10, 15...) for ease of use.



6.2.1.2 Pin Descriptions by Functions

Pin Descriptions by Functions

Pins sorted by function:

Function	Signal Name	Terminal Number	Input/Output	Logic	Descr
Power					

Function	Signal Name	Terminal Number	Input/Output	Logic	Description
Ground	GND	1-49, 1-50, 2-49, 2-50, 3-49, 3-50	Power Source		Common Ground Make
Voltage	VDD	1-1, 1-2, 2-1, 2-2, 3-1, 3-2	Power Source		Supply Voltage The al +3.3 V Make
Reset	/RST	2-48	Input	Negative	Input can be Common hold /
Power Indicator	LED	2-11	Output	Negative	LED power show
Motion					
Pulse	PULx PULy PULz † PULu † PUL5 ‡ PUL6 ‡ PUL7 ‡ PUL8 ‡	1-3 1-9 1-15 † 1-21 † 4-1 ‡ 4-13 ‡ 4-25 ‡ 4-37 ‡	Output	Negative#	Output contro When select Output directi When select Output directi When select Output differe The out softwa

Function	Signal Name	Terminal Number	Input/Output	Logic	Describe
Direction	DIRx DIRy DIRz † DIRu † DIR5 ‡ DIR6 ‡ DIR7 ‡ DIR8 ‡	1-4 1-10 1-16 † 1-22 † 4-2 ‡ 4-14 ‡ 4-26 ‡ 4-38 ‡	Output	Negative#	Output control direction. When select Output direction. When select Output direction. When select Output direction. The output software
Enable	EOx EOy EOz † EOu † EO5 ‡ EO6 ‡ EO7 ‡ EO8 ‡	1-5 1-11 1-17 † 1-23 † 4-3 ‡ 4-15 ‡ 4-27 ‡ 4-39 ‡	Output	Negative#	Output driver. Output software
In-Position	INPx INPy INPz † INPu † INP5 ‡ INP6 ‡ INP7 ‡ INP8 ‡	1-8 1-14 1-20 † 1-26 † 4-6 ‡ 4-18 ‡ 4-30 ‡ 4-42 ‡	Input U	Negative#	Input from software
Servo Alarm	ALMx ALMy ALMz † ALMu † ALM5 ‡ ALM6 ‡ ALM7 ‡ ALM8 ‡	1-6 1-12 1-18 † 1-24 † 4-4 ‡ 4-16 ‡ 4-28 ‡ 4-40 ‡	Input U	Negative#	Input signal immediate stop. The input software

Function	Signal Name	Terminal Number	Input/Output	Logic	Descr
Error Clear	ERCx ERCy ERCz † ERCu † ERC5 † ERC6 † ERC7 † ERC8 †	1-7 1-13 1-19 † 1-25 † 4-5 † 4-17 † 4-29 † 4-41 †	Output	Negative#	Output count as a p width A LEVI
Safety					
Simultaneous Start	/CSTA	2-22	Input/Output	Negative	Input/ simult Synch
Simultaneous stop	/CSTP	2.21	Input/Output	Negative	Input/ simult
E-Stop	/CEMG	2-23	Input U	Negative	Input this sig If this operat operat
Limits	+Lx +Ly +Lz † +Lu † +L5 † +L6 † +L7 † +L8 †	1-36 1-39 1-42 † 1-45 † 4-11 † 4-23 † 4-35 † 4-47 †	Input U	Negative#	Input (+) dir while directi imme stop. The in softwa
	-Lx -Ly -Lz † -Lu † -L5 † -L6 † -L7 † -L8 †	1-35 1-38 1-41 † 1-44 † 4-10 † 4-22 † 4-34 † 4-46 †	Input U	Negative#	Input (-) dire while motio imme stop. The in softwa
Slow Down	SDx SDy SDz † SDu †	3-43 3-44 3-45 † 3-46 †	Input U	Negative#	Input signal or LAT The in softwa

Function	Signal Name	Terminal Number	Input/Output	Logic	Descr
Home	Hx	1-37	Input U	Negative#	Input Used 1 (edge The in softwa
	Hy	1-40			
	H _z †	1-43 †			
	H _u †	1-46 †			
	H5 †	4-12 †			
	H6 †	4-24 †			
	H7 †	4-36 †			
	H8 †	4-48 †			
Feedback					
Encoder	EAx, EBx	2-24, 2-25	Input U		Input contro the en differ positiv (-) pul When signals of the count The co using
	EAy, EBy	2-27, 2-28			
	EAz, EBz †	2-30, 2-31 †			
	E _{Au} , E _{Bu} †	2-33, 2-34 †			
	EA5, EB5 †	4-7, 4-8 †			
	EA6, EB6 †	4-19, 4-20 †			
	EA7, EB7 †	4-31, 4-32 †			
	EA8, EB8 †	4-43, 4-44 †			
Index	EZx	2-26	Input U	Negative#	Input using mode. each t EZ sig precis The in softwa
	EZy	2-29			
	EZ _z †	2-32 †			
	EZ _y †	2-35 †			
	EZ5 †	4-9 †			
	EZ6 †	4-21 †			
	EZ7 †	4-33 †			
	EZ8 †	4-45 †			
Input/Output					
Digital I/O	DI1/LTCx	1-31	Input U	Negative#	High-s functio and pu The in softwa
	DI2/LTCy	1-32			
	DI3/LTCz †	1-33 †			
	DI4/LTCu †	1-34 †			
	D01/SYNcx	1-27	Output	Positive	High-s synchr will ou synchr The in softwa
	D02/SYNcy	1-28			
	D03/SYNcz †	1-29 †			
	D04/SYNcu †	1-30 †			
	D05/SYNc5 †	1-47 †			
	D06/SYNc6 †	1-48 †			
	D07/SYNc7 †	4-49 †			
	D08/SYNc8 †	4-50 †			

Function	Signal Name	Terminal Number	Input/Output	Logic	Description
	IO1 to IO32	3-5 to 3-36	Input/Output		General purpose I/O
Analog	AI1	3-39	Input		Analog Input
	AI2	3-40	Input		Analog Input
	AO1	3-37	Output		Analog Output
	AO2	3-38	Output		Analog Output
	PWM1	3-41	Output		PWM Output
	PWM2	3-42	Output		PWM Output
MPG	PAx, PBx PAy, PBy PAz, PBz + PAu, PBu +	2-36, 2-37 2-38, 2-29 2-40, 2-41 + 2-42, 2-43 +	Input U		Input pulses for motor position feedback. When used, the count of the pulses is used for the count. The count is used for the motor position feedback.
Communication					
USB	D+ D- VBUS	2-4 2-3 2-5	Input/Output Input/Output Input		USB D+ USB D- USB VBUS
RS485	/REDE RxD TxD	2-6 2-7 2-8	Output Input Output		Receive comm RXD s TXD s
I ² C	SDA SCL	3.47 3.48	Input/Output Output		SDA [I ² C] SCL [I ² C]
SPI	NSS0 MOSI0 MISO0 SCK0	2-12 2-13 2-14 2-15			Second Main c Main i Serial
CAN	CAN_Tx CAN_Rx	2-18 2-20			These firmwa conne
Clock					
Clock Bus	Clock+ Clock-	2-17 2-19	Input/Output		
Clock Select	CLK_SEL	2.47	Input		
No Connection					

Function	Signal Name	Terminal Number	Input/Output	Logic	Descr
Reserved		2-9, 2-10, 2-16, and 2-44 to 2-46	N/C		These firmwa conne

Note 1: All pins shown are common between the CMD-2CR, CMD-4CR, and CMD-8CR, unless otherwise noted. Pins marked with † are only used on CMD-4CR and CMD-8CR while pins marked with ‡ are only used on the CMD-8CR.

Note 2: "Input U" refers to an input with a pull-up resistor. The internal pull-up resistance (40 k to 240 k-ohms) is only used to keep a terminal from floating. If you want to use the Commander core with an open collector system, an external pull-up resistor (5k to 10 k-ohms) is required.

As a noise prevention measure, pull up unused terminals to VDD using an external resistor (5 k to 10k-ohms), or connect them directly to VDD.

Note 3: "Input/Output *" refers to a terminal with a pull-up resistor. The internal pull-up resistor (40 k to 240 k-ohms) is only used to keep a terminal from floating. If it is connected in a wired OR circuit, an external pull-up resistor (5 k to 10 k-ohms) is required.

As a noise prevention measure, pull up unused terminals to VDD using an external resistor (5 k to 10 k-ohms).

Note 4: If an output terminal is not being used, leave it open.

Note 5: "Positive" refers to positive logic. "Negative" refers to negative logic. "#" means that the logic can be changed using software. The logic shown refers only to the initial status of the terminal.

Note 6: Use the POL[axis] command to select an output signal.

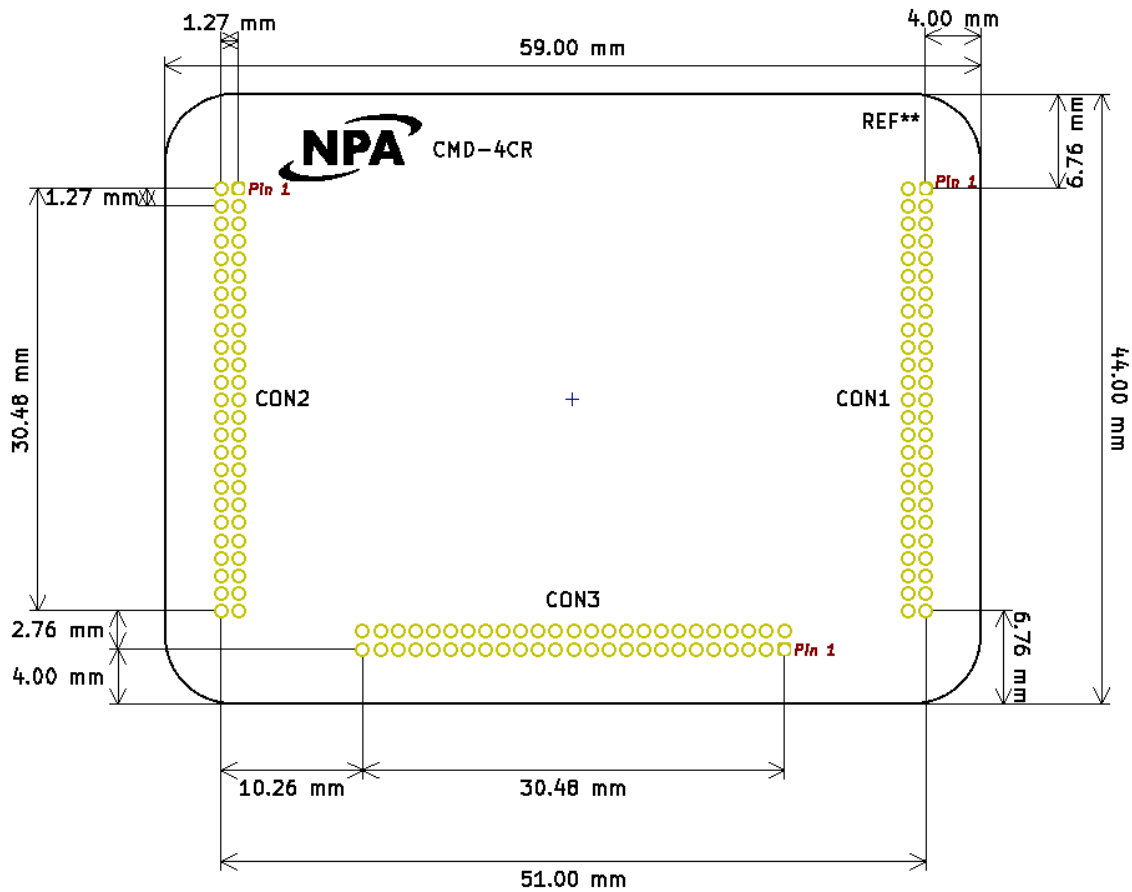
Note 7: When a deceleration stop (Slow Down) is selected, keep the input signal ON until an axis stops.

Note 8: Home input is synchronized with output pulses, sampled and controlled by a change of sampling result. Therefore, keep Home sensor ON for longer than feed amount for one pulse.

6.2.1.3 Recommended Footprint

Recommended Footprint

TODO



6.2.2 Interface Boards

Interface Boards

Nippon Pulse provides off-the-shelf interface board designs specifically for the Commander core. These provide access to all the features and technology of the Commander core.

Development Kit

A development kit that includes a Commander core, an interface PCB, related cables, and software (via download) is available. Every available feature and signal on the Commander core is available through the development kit (**CMD-4CR-EV**), which can function as the launchpad for your custom motion solution. [Contact Nippon Pulse](#) for information about the development kit.

7.0 Electrical Characteristics

Electrical Characteristics

Unless otherwise stated all specifications are based on and were tested at 25 °C and $V_{dd}=3.3V$.

In this section

- [Electrical and Thermal Specifications](#)
- [Handling Precautions](#)

- o [Design precautions](#)
- o [Precautions for transporting and storing Commander cores](#)
- o [Precautions for installation](#)
- o [Other precautions](#)

7.1 Electrical and Thermal Specifications

Electrical and Thermal Specifications

Parameter	Min	Typ	Max	Units
Power				
Main Power Input (V_{DD})	+3	+3.3	+3.6	V
		250	1000	mA
VBUS (USB Sense)	0	+5	+5.8	V
I/O				
Digital Outputs	0		V_{DD}	V
	-6		6	mA
Digital Output High Voltage	$V_{DD} - 0.6$		V_{DD}	V
Digital Output Low Voltage	0	0.4	0.6	V
Digital Inputs	-0.3		V_{DD}	V
	-90		20	uA
Digital Input High Voltage	$V_{DD} - 0.6$		5.8	V
Digital Input Low Voltage	0		0.6	V
Analog Inputs	0		V_{DD}	V
			1.5	mA
ADC resolution		12		bit
Analog Outputs	0		V_{DD}	V
			2.0	mA
DAC resolution		12		bit
Thermal				
Operating Temperature	-40		+85	°C
Storage Temperature	-40		+100	°C
Flash Memory				
Endurance (Write/Erase Cycles)	20K	100K		Cycles
Retention (25°C, 1K Cycles)	10	100		Years

7.2 Handling Precautions

Handling Precautions

- [Design Precautions](#)
- [Precautions for transporting and storing Commander cores](#)
- [Precautions for installation](#)
- [Other precautions](#)

Design precautions

- 1) Never exceed the absolute maximum ratings, even for a very short time.
- 2) Take precautions against the influence of heat in the environment, keeping the temperature around the Commander core as cool as possible.
- 3) Please note that ignoring the following may result in latching up and may cause overheating and smoke.
 - a. Do not apply a voltage greater than the absolute maximum rating voltage. Please consider the voltage drop timing when turning the power ON/OFF.
 - b. Be careful not to introduce external noise into the Commander core.
 - c. Hold the unused input terminals to +3.3V or GND level.
 - d. Do not short-circuit the outputs.
 - e. Protect the LSI from inductive pulses caused by electrical sources that generate large voltage surges, and take appropriate precautions against static electricity.
- 4) Provide external circuit protection components so that overvoltage caused by noise, voltage surges, or static electricity is not fed to the Commander core.

Precautions for transporting and storing Commander cores

- 1) Always handle carefully and keep in their packages. Throwing or dropping may damage them.
- 2) Do not store in a location exposed to water droplets or direct sunlight.
- 3) Do not store in a location where corrosive gases are present, or in excessively dusty environments.
- 4) Store in an anti-static storage container, and make sure that no physical load is placed on the Commander cores.

Precautions for installation

- 1) In order to prevent damage caused by static electricity, pay attention to the following:
 - a. Make sure to ground all equipment, tools, and jigs that are present at the work site.
 - b. Ground the work desk surface using a conductive mat or similar apparatus (with an appropriate resistance factor). Do not allow work on a metal surface, which can cause a rapid change in the electrical charge due to extremely low resistance.
 - c. When using a vacuum device, provide anti-static protection using a conductive rubber pickup tip. Anything that comes in contact with the leads should have as high a resistance as possible.

- d. When using a pincer that may make contact with any terminals, use an anti-static model. Do not use a metal pincer, if possible.
 - e. Store unused Commander cores in a PC-board storage box that protects against static electricity, and make sure there is adequate clearance between the Commander cores. Never directly stack them on each other, as it may cause friction that can develop an electrical charge.
- 2) Operators must wear wrist straps that are grounded through approximately 1M-ohm of resistance.
 - 3) Use low voltage soldering devices and make sure the tips are grounded.
 - 4) Do not store or use Commander cores near high-voltage electrical fields, such those produced by a CRT.
 - 5) Please avoid soldering in a soaking method so as to prevent a dramatic change of temperature to the package and damage to the device.

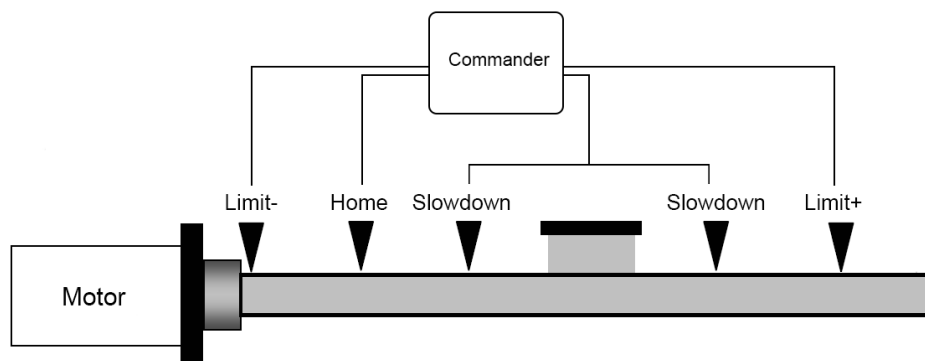
Other precautions

- 1) The package is made from flame retardant material but may burn. When baked or burned, it may generate gases or fire. Do not use it in the proximity of ignition sources or flammable objects.
- 2) The Commander core is designed for use in commercial apparatus (office machines, communication equipment, measuring equipment, and household appliances). If you use it in any device that may require high quality and reliability, or where faults or malfunctions may directly affect human injury/survival (such as in nuclear power control devices, aviation devices or spacecraft, traffic signals, fire control, or various types of safety devices), we will not be liable for any problem that occurs, even if it was directly caused by the Commander core. Customers must provide their own safety measures to ensure appropriate performance in all circumstances.

Appendix A: Homing Mode Actions

Appendix A: Homing Mode Actions

Commander core has 13 homing search methods, utilizing home limit, end of travel limit, slowdown inputs, or Z-index pulse, or combinations of these to complete a homing operation. Homing is used to establish a zero or home position to use as a reference for absolute motion actions. Some homing methods will stop in a non-zero position, using the position at which the home, end of travel, or Z-index pulse was triggered as the zero-reference position.



The chart below indicates which type of limits are used in each mode, along with home

approach speed, direction and end-point status.

● indicates input required, ○ indicates input optional.

Homing Mode	Home	End Limits	Z-index	Slow Down	Approaches Home at		Direction to Trigger Zero	Ending Point
					High Speed	Low Speed		
Mode 0	●			○	H		Dir of Home	Non-Zero
Mode 1	●			○		L	Dir of Home	Zero
Mode 2	●		●	○		L	Dir of Home	Zero
Mode 3	●		●	○	H		Dir of Home	Non-Zero
Mode 4	●		●	○		L	Reverse Dir	Zero
Mode 5	●		●	○	H		Reverse Dir	Non-Zero
Mode 6		●		○	H		Reverse Dir	Zero
Mode 7		●	●	○		L	Reverse Dir	Zero
Mode 8		●	●	○	H		Reverse Dir	Non-Zero
Mode 9	●			○	H		Reverse Dir	Zero
Mode 10	●		●	○	H		Reverse Dir	Zero
Mode 11	●		●	○	H		Dir of Home	Zero
Mode 12		●	●	○	H		Dir of Home	Zero

Homing inputs

Consider the system mechanics carefully when setting up limit switches, using caution to set the limits for the appropriate axis movement and considering the deceleration distance when establishing end of travel limits relative to the hard stop positions.

Note: For homing modes 9 – 12, encoder feedback is required. When using a stepper motor and homing with modes 9 – 12, an encoder is required and the command pulse count and encoder pulse count must be equal.

More information on the homing modes can be found in Section 9-5-1 of the PCL6045BL users manual.

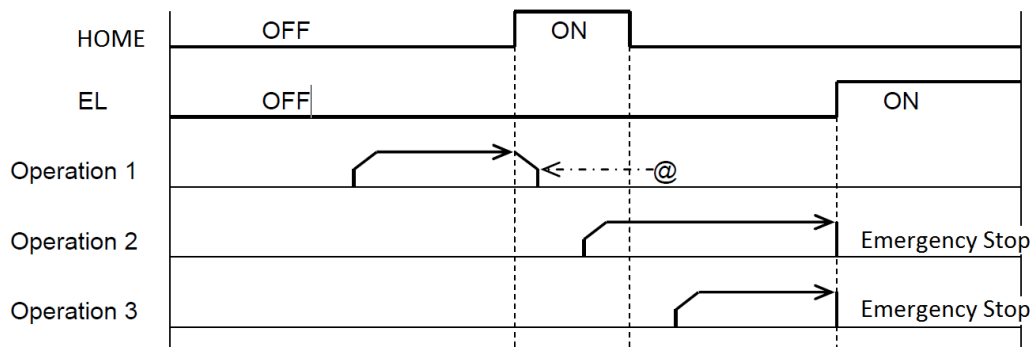
A.1 Homing Mode 0

Homing Mode 0

Homing mode 0 is available [with](#) or [without](#) a slowdown input. It operates differently when the SD input is used.

Homing mode 0 with Slow Down (SD) disabled. This homing method works with a home limit switch input and sets home when the motion stops after the home limit switch is encountered.

Homing Mode	Home Limit	End Limits	Z-index	Slow Down	Approaches Home at		Direction to Trigger Zero	Ending Point
					High Speed	Low Speed		
Mode 0	•			○	H		Dir of Home	Non-Zero



Operation 1:

- Homing is started toward home switch, accelerate from LSPD to HSPD
- Deceleration starts when the home (ORG) input point is detected, zero position is set.
- Homing is complete when deceleration stops.
- ERC pulse is sent.

Operation 2:

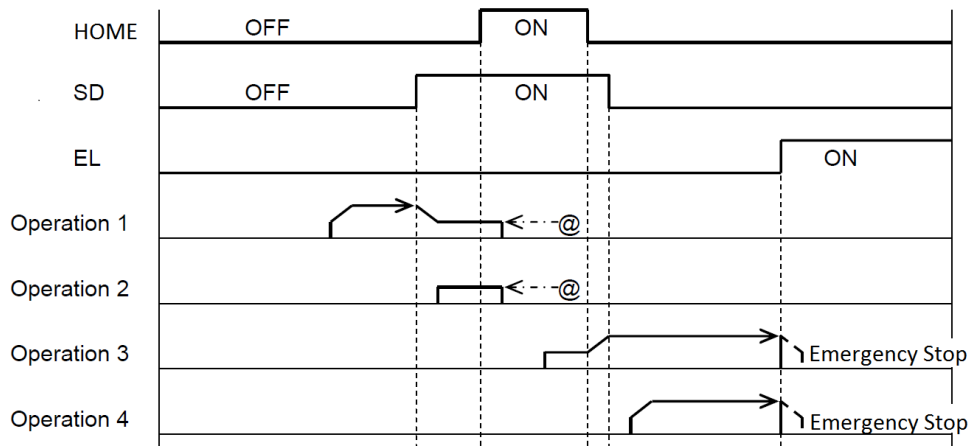
- Homing is started while home input is activated.
- Acceleration from LSPD to HSPD.
- End limit is encountered, immediate stop.

Operation 3:

- Homing is started moving away from the home input switch.
- Acceleration from LSPD to HSPD.
- End limit is encountered, immediate stop.

Homing mode 0 with Slow Down (SD) enabled. In this homing mode, an input will first trigger a deceleration to a lower speed point and then stops and sets home as the home limit switch is reached.

Homing Mode	Home Limit	End Limits	Z-index	Slow Down	Approaches Home at		Direction to Trigger Zero	Ending Point
					High Speed	Low Speed		
Mode 0	•			•		L	Dir of Home	Non-Zero



Operation 1:

- A) Homing is started toward home switch, accelerate from LSPD to HSPD.
- B) Deceleration starts to LSPD when the SD input point is detected, continue towards home.
- C) Stop when home input is detected, homing complete.
- D) ERC pulse is sent.

Operation 2:

- A) Homing is started while SD input is activated.
- B) Continue traveling at LSPD towards home input.
- C) Stop when home input is detected, zero position is set, homing complete.
- D) ERC pulse is sent.

Operation 3:

- A) Homing is started while SD input is activated and is moving away from the home input switch.
- B) Travel at LSPD until SD input is deactivated, then accelerate from LSPD to HSPD.
- C) End limit is encountered, immediate stop.

Operation 4:

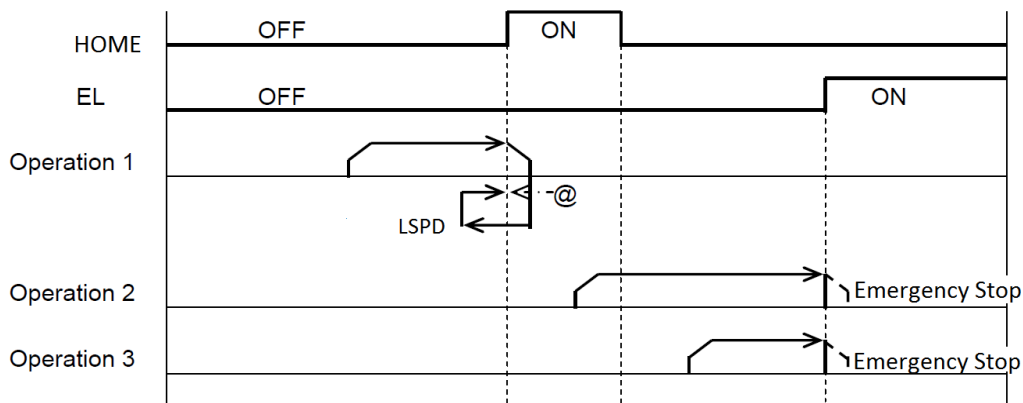
- A) Homing is started moving away from the home input switch and SD limit is deactivated.
- B) Acceleration from LSPD to HSPD.
- C) End limit is encountered, immediate stop.

A.2 Homing Mode 1

Homing Mode 1

Homing mode 1 decelerates when it first encounters the home limit switch, reverses direction to back off the limit switch, and then approaches the home limit switch again at a lower speed point and sets home when the home limit switch is encountered a second time.

Homing Mode	Home Limit	End Limits	Z-index	Slow Down	Approaches Home at		Direction to Trigger Zero	Ending Point
					High Speed	Low Speed		
Mode 1	•			○		L	Dir of Home	Zero



Operation 1:

- Homing operation is started, accelerate from LSPD to HSPD.
- Deceleration starts when home limit is encountered.
- Reverse direction at move at LSPD towards home limit.
- Move at LSPD until home limit is turned off and reverse direction.
- Operation stops when home limit is encountered again, zero position is set.
- ERC pulse is sent.

Operation 2:

- Homing is started while home input is activated.
- Acceleration from LSPD to HSPD.
- End limit is encountered, immediate stop.

Operation 3:

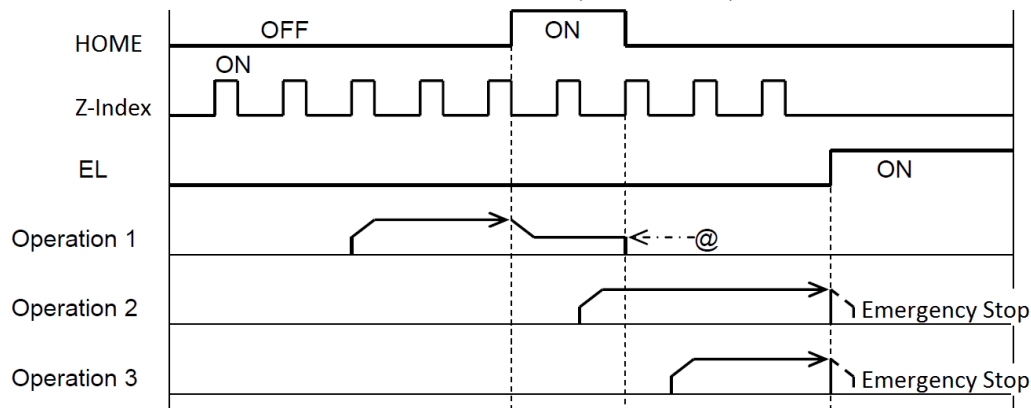
- Homing is started moving away from the home input switch.
- Acceleration from LSPD to HSPD.
- End limit is encountered, immediate stop.

A.3 Homing Mode 2

Homing Mode 2

Homing mode 2 utilizes both a home limit switch and the Z-index pulse of the encoder. Once the home limit switch is encountered, the motor slows to a lower speed point and sets home as it encounters the Z-index pulse.

Homing Mode	Home Limit	End Limits	Z-index	Slow Down	Approaches Home at		Direction to Trigger Zero	Ending Point
					High Speed	Low Speed		
Mode 2	•		•	○		L	Dir of Home	Zero



Operation 1:

- A) Homing operation is started, accelerate from LSPD to HSPD.
- B) Deceleration starts when home limit is encountered.
- C) Continue moving at LSPD until specified number of Z-index pulses are counted (ZCNT), zero position is set.
- D) Operation stops and is completed.
- E) ERC pulse is sent.

Operation 2:

- A) Homing is started while home input is activated.
- B) Acceleration from LSPD to HSPD.
- C) End limit is encountered, immediate stop.

Operation 3:

- A) Homing is started moving away from the home input switch.
- B) Acceleration from LSPD to HSPD.
- C) End limit is encountered, immediate stop.

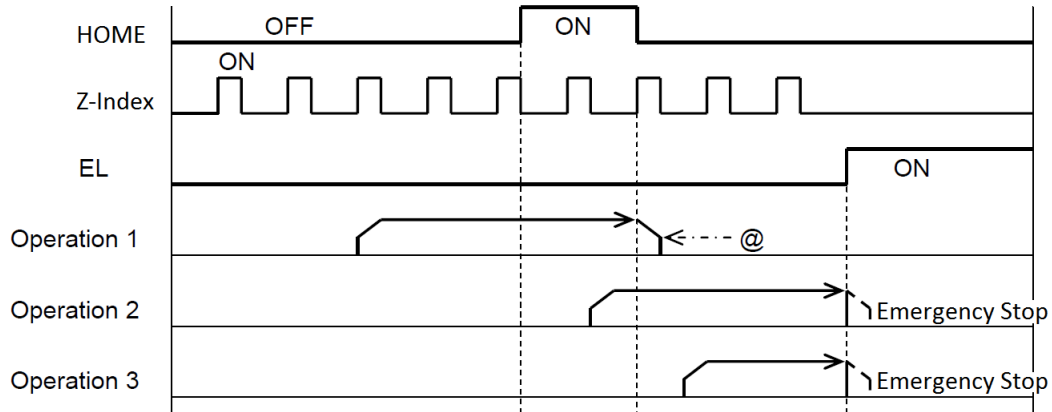
A.4 Homing Mode 3

Homing Mode 3

Homing mode 3 uses both the home limit switch and Z-index pulse. The deceleration to the home set point occurs after the move passes through the home limit switch and encounters the Z-index pulse.

Homing Mode	Home Limit	End Limits	Z-index	Slow Down	Approaches Home at	Direction to Trigger	Ending Point

					High Speed		Zero	
Mode 3	•		•	○	H		Dir of Home	Non-Zero



Operation 1:

- Homing operation is started, accelerate from LSPD to HSPD.
- Continue moving at HSPD after home limit is encountered.
- Decelerate when specified number of Z-index pulses are counted (ZCNT), zero position is set.
- Operation decelerates and stops.
- ERC pulse is sent.

Operation 2:

- Homing is started while home input is activated.
- Acceleration from LSPD to HSPD.
- End limit is encountered, immediate stop.

Operation 3:

- Homing is started moving away from the home input switch.
- Acceleration from LSPD to HSPD.
- End limit is encountered, immediate stop.

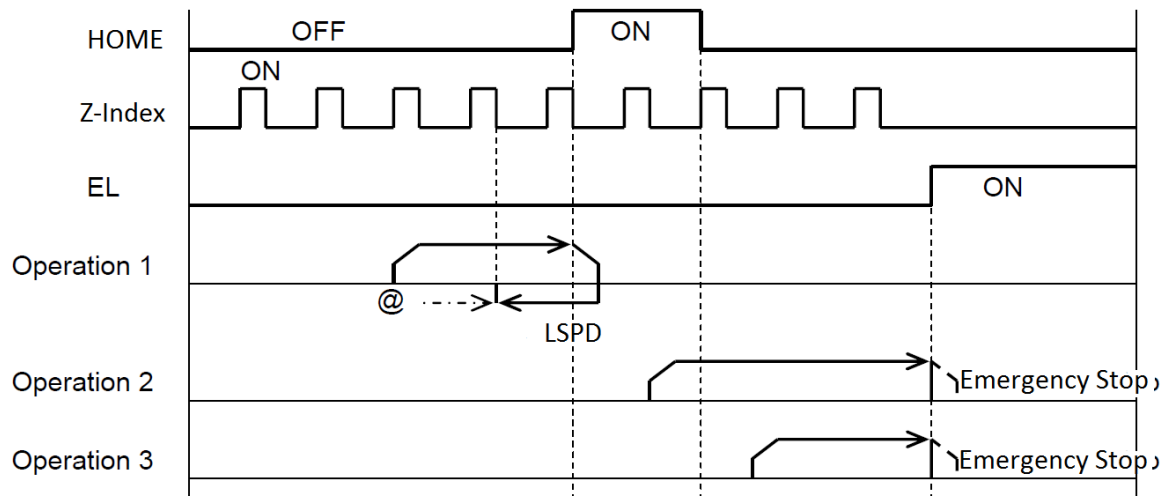
A.5 Homing Mode 4

Homing Mode 4

For homing mode 4, the motor decelerates upon reaching the home limit switch and reverses direction upon reaching the lower speed point. Home is set when the Z-index pulse is encountered.

Homing Mode	Home Limit	End Limits	Z-index	Slow Down	Approaches Home at		Direction to Trigger Zero	Ending Point
					High Speed	Low Speed		

Mode 4	•		•	○		L	Reverse Dir	Zero



Operation 1:

- Homing operation is started, accelerate from LSPD to HSPD.
- Begin deceleration after home limit is encountered.
- Reverse direction at LSPD after deceleration complete.
- Continue moving at LSPD until specified number of Z-index pulses are counted (ZCNT).
- Operation stops and is completed; zero position is set.
- ERC pulse is sent.

Operation 2:

- Homing is started while home input is activated.
- Acceleration from LSPD to HSPD.
- End limit is encountered, immediate stop.

Operation 3:

- Homing is started moving away from the home input switch.
- Acceleration from LSPD to HSPD.
- End limit is encountered, immediate stop.

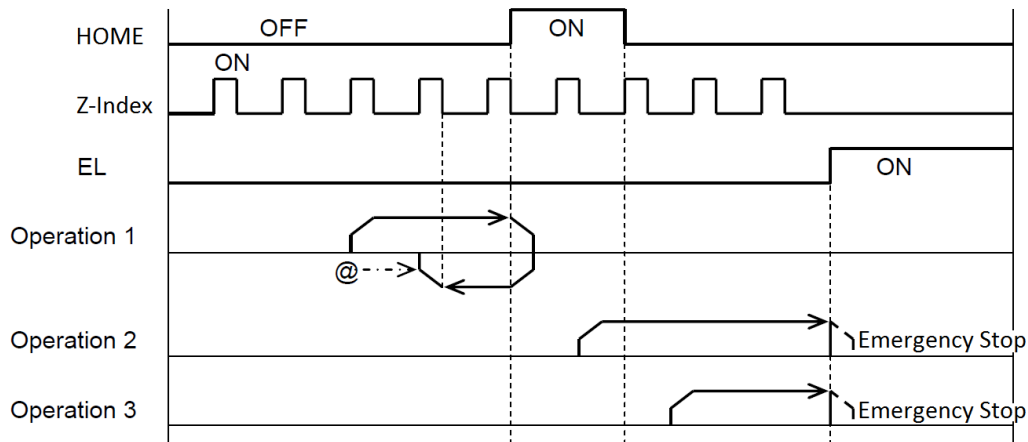
A.6 Homing Mode 5

Homing Mode 5

In homing mode 5, the home limit switch initiates a full reversal from high speed in the initial direction to high speed in the reverse direction. Homing is completed when the Z-index pulse is encountered and the motion comes to a stop.

Homing Mode	Home Limit	End Limits	Z-index	Slow Down	Approaches Home at		Direction to Trigger Zero	Ending Point
					High Speed	Low Speed		

Mode 5	•		•	○	H		Reverse Dir	Non-Zero



Operation 1:

- Homing operation is started, accelerate from LSPD to HSPD.
- Begin deceleration to LSPD after home limit is encountered.
- Reverse direction and accelerate to HSPD.
- Continue moving at HSPD until specified number of Z-Index pulses are counted (ZCNT), zero position is set.
- Operation decelerates, stops and is completed.
- ERC pulse is sent.

Operation 2:

- Homing is started while home input is activated.
- Acceleration from LSPD to HSPD.
- End limit is encountered, immediate stop.

Operation 3:

- Homing is started moving away from the home input switch.
- Acceleration from LSPD to HSPD.
- End limit is encountered, immediate stop.

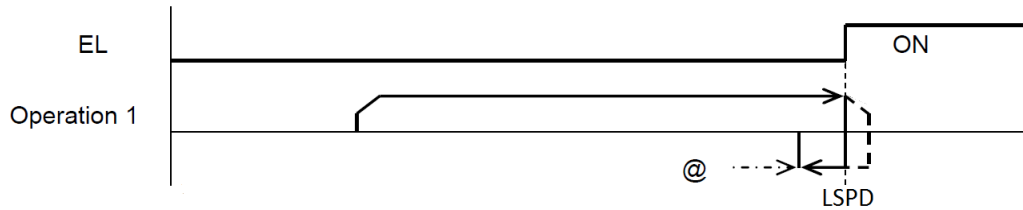
A.7 Homing Mode 6

Homing Mode 6

Homing mode 6 uses the end of travel limit switch to trigger home location. The end limit is encountered and the motor reverses off the end of travel limit at a lower speed and sets home position.

Homing Mode	Home Limit	End Limits	Z-index	Slow Down	Approaches Home at		Direction to Trigger Zero	Ending Point
					High Speed	Low Speed		

Mode 6		•		○	H		Reverse Dir	Zero

**Operation 1:**

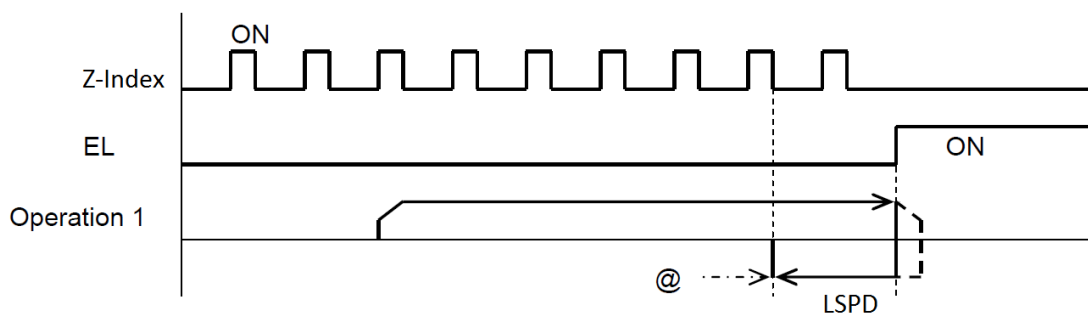
- A) Homing operation is started, accelerate from LSPD to HSPD.
- B) Immediate stop when the end of travel limit switch is encountered.
- C) Reverse direction at LSPD until end limit switch turns off.
- D) Operation stopped immediately and homing operation is completed, zero position is set.
- E) ERC pulse is sent.

Note: When utilizing the end travel limits, the ERC command will clear the end-of-travel limit flag in the direction of the home during the homing operation. Refer to **ERC** command in the Command Reference for further details.

A.8 Homing Mode 7**Homing Mode 7**

Homing mode 7 is similar to [homing mode 6](#), except it reverses until it locates the Z-index pulse after encountering the end of travel limit switch. Once the Z-index is reached, homing is completed.

Homing Mode	Home Limit	End Limits	Z-index	Slow Down	Approaches Home at		Direction to Trigger Zero	Ending Point
					High Speed	Low Speed		
Mode 7		•	•	○		L	Reverse Dir	Zero

**Operation 1:**

- A) Homing operation is started, accelerate from LSPD to HSPD.
- B) Immediate stop when the end of travel limit switch is encountered.
- C) Reverse direction at LSPD until specified number of Z-Index pulses are counted (ZCNT).
- D) Operation stopped immediately and homing operation is completed, zero position is set.
- E) ERC pulse is sent.

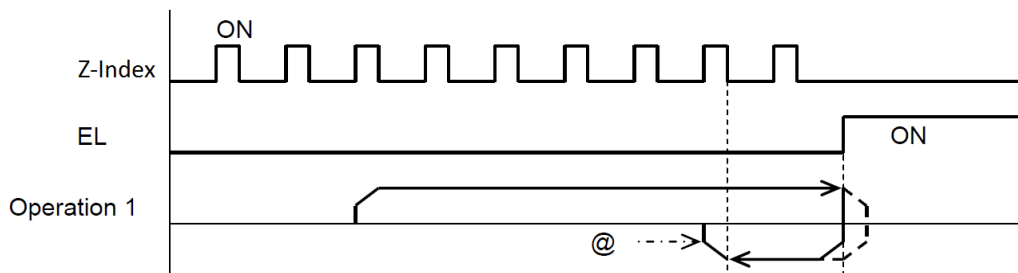
Note: When utilizing the end travel limits, the ERC command will clear the end-of-travel limit flag in the direction of the home during the homing operation. Refer to **ERC** command in the Command Reference for further details.

A.9 Homing Mode 8

Homing Mode 8

In homing mode 8, the end of travel limit switch triggers an immediate stop and then full reversal at high speed until Z-index is encountered and zero position is set. At this point the motor decelerates and stops.

Homing Mode	Home Limit	End Limits	Z-index	Slow Down	Approaches Home at		Direction to Trigger Zero	Ending Point
					High Speed	Low Speed		
Mode 8		•	•	○	H		Reverse Dir	Non-Zero



Operation 1:

- A) Homing operation is started, accelerate from LSPD to HSPD.
- B) Immediate stop when the end of travel limit switch is encountered.
- C) Reverse direction accelerating from LSPD to HSPD and move until specified number of Z-Index pulses are counted (ZCNT).
- D) As the specified number of Z-index pulses is encountered, zero position is set and deceleration to LSPD begins.
- E) Homing operation is completed by deceleration and stop.
- F) ERC pulse is sent.

Note: When utilizing the end travel limits, the ERC command will clear the end-of-travel limit flag in the direction of the home during the homing operation. Refer to **ERC** command in the Command Reference for further details.

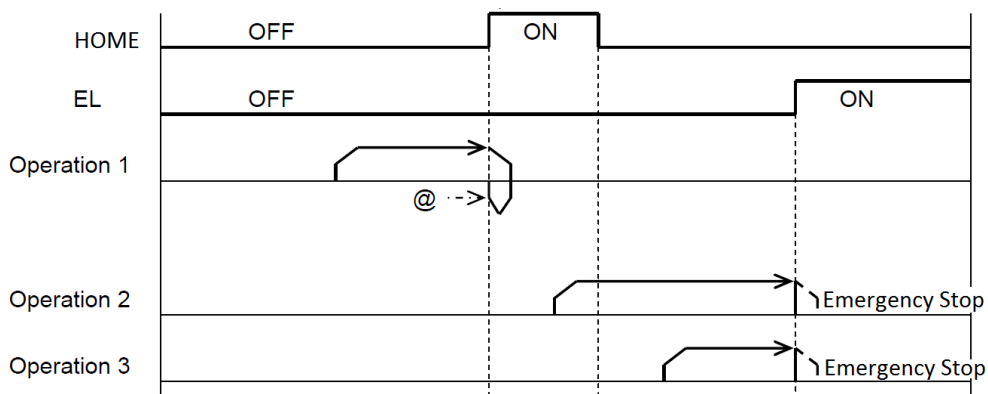
A.10 Homing Mode 9

Homing Mode 9

Homing Mode 9 uses the home limit switch to initiate a deceleration, then continues with a quick index back to the position where the home limit was encountered.

- This is similar to [Homing mode 0](#) except it returns to the encoder 0 point, after the home limit is encountered. If your system does not have an encoder, or the encoder pulse count is different from the motor pulse count, it is suggested that you use [Homing mode 0](#), as Homing mode 9 will not function properly.

Homing Mode	Home Limit	End Limits	Z-index	Slow Down	Approaches Home at		Direction to Trigger Zero	Ending Point
					High Speed	Low Speed		
Mode 9	•			○	H		Reverse Dir	Zero



Operation 1:

- Homing operation is started, accelerate from LSPD to HSPD.
- Deceleration begins as the home limit switch is encountered and zero position is set.
- Begin zero-point return operation back to home position at the end of deceleration, accelerate from LSPD to HSPD.
- Homing operation is completed when the 0 position is reached.
- ERC pulse is sent.

Operation 2:

- Homing is started while home input is activated.
- Acceleration from LSPD to HSPD.
- End limit is encountered, immediate stop.

Operation 3:

- Homing is started moving away from the home input switch.
- Acceleration from LSPD to HSPD.
- End limit is encountered, immediate stop.

Note: For homing modes 9 – 12, encoder feedback is required. When using a stepper motor and homing with modes 9 – 12, an encoder is required and the command pulse count and encoder pulse count must be equal.

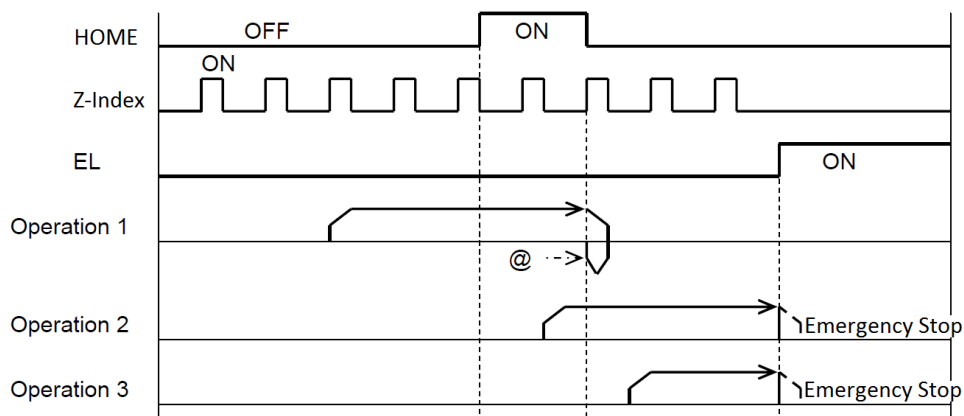
A.11 Homing Mode 10

Homing Mode 10

Homing mode 10 uses both the home limit switch and Z-index pulse. Deceleration occurs after the move passes through the home limit switch and encounters the Z-index pulse. The operation continues with a quick index back to the position where the Z-index pulse was encountered.

- This is similar to [Homing mode 3](#) except it returns to the encoder 0 point, after the home limit is encountered. If your system does not have an encoder, or the encoder pulse count is different from the motor pulse count, it is suggested that you use [Homing mode 3](#) instead, as Homing mode 10 will not function properly.

Homing Mode	Home Limit	End Limits	Z-index	Slow Down	Approaches Home at		Direction to Trigger Zero	Ending Point
					High Speed	Low Speed		
Mode 10	•		•	○	H		Reverse Dir	Zero



Operation 1:

- Homing operation is started, accelerate from LSPD to HSPD.
- Continue moving at HSPD as the home limit switch is encountered.
- Deceleration begins when specified number of Z-index pulses is encountered (ZCNT), zero set.
- At end of deceleration, a zero-point return operation begins to the position where Z-index pulse was encountered, accelerating from LSPD to HSPD and decelerating and stopping at 0 position.
- Homing is completed when the 0 position is reached.
- ERC pulse is sent.

Operation 2:

- A) Homing is started while home input is activated.
- B) Acceleration from LSPD to HSPD.
- C) End limit is encountered, immediate stop.

Operation 3:

- A) Homing is started moving away from the home input switch.
- B) Acceleration from LSPD to HSPD.
- C) End limit is encountered, immediate stop.

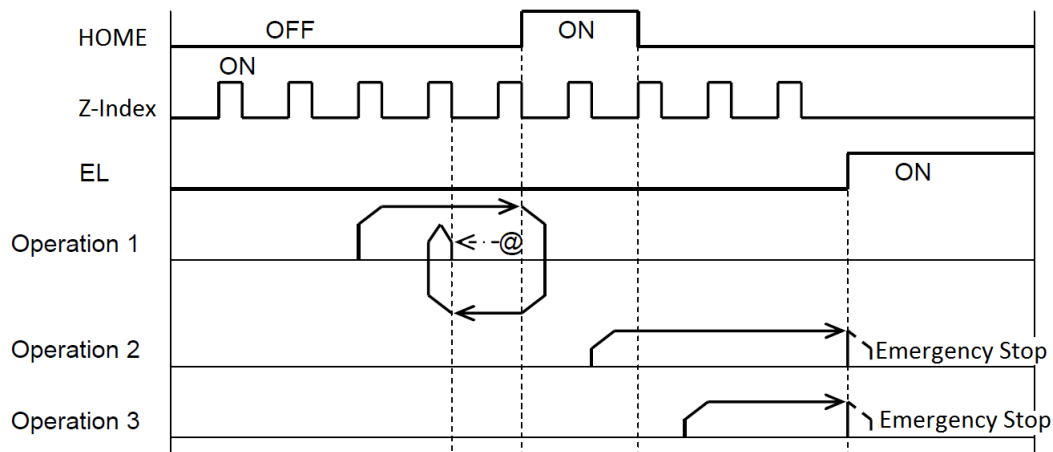
Note: For homing modes 9 – 12, encoder feedback is required. When using a stepper motor and homing with modes 9 – 12, an encoder is required and the command pulse count and encoder pulse count must be equal.

A.12 Homing Mode 11**Homing Mode 11**

In homing mode 11, the home limit switch is used to initiate a deceleration. The operation then continues with a fast move in the reverse direction until the Z-index pulse is encountered, where a quick index reverses motion back to the position where the Z-index pulse was encountered.

- This is the similar to [Homing mode 5](#) except it returns to the encoder 0 point, after the home limit is encountered. If your system does not have an encoder, or the encoder pulse count is different from the motor pulse count, it is suggested that you use [Homing mode 5](#) instead, as Homing mode 11 will not function properly.

Homing Mode	Home Limit	End Limits	Z-index	Slow Down	Approaches Home at		Direction to Trigger Zero	Ending Point
					High Speed	Low Speed		
Mode 11	•		•	○	H		Dir of Home	Zero



Operation 1:

- A) Homing operation is started, accelerate from LSPD to HSPD, then decelerate as the home limit switch is encountered.
- B) Reverse direction, accelerating from LSPD to HSPD, then decelerate when specified number of Z-indexpulses is encountered (ZCNT), zero position is set.
- C) Start a zero-point return move to the position where the Z-index pulse was encountered, accelerating from LSPD to HSPD then decelerating and stopping at 0 position.
- D) ERC pulse is sent.

Operation 2:

- A) Homing is started while home input is activated.
- B) Acceleration from LSPD to HSPD.
- C) End limit is encountered, immediate stop.

Operation 3:

- A) Homing is started moving away from the home input switch.
- B) Acceleration from LSPD to HSPD.
- C) End limit is encountered, immediate stop.

Note: For homing modes 9 – 12, encoder feedback is required. When using a stepper motor and homing with modes 9 – 12, an encoder is required and the command pulse count and encoder pulse count must be equal.

A.13 Homing Mode 12

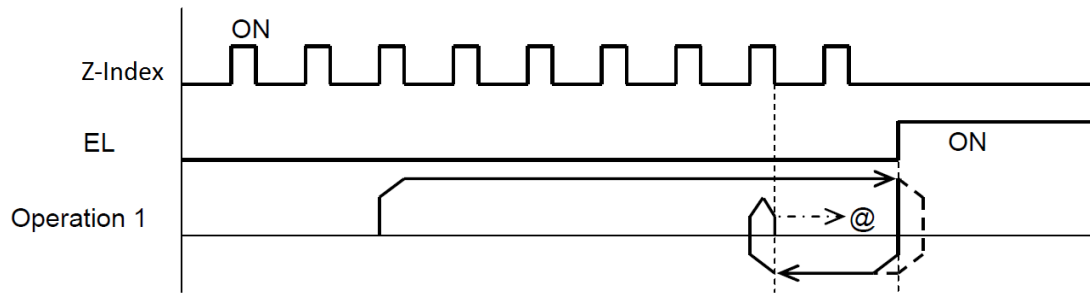
Homing Mode 12

Homing mode 12 is similar to homing mode 11, except it utilizes the end of travel limit to initiate an immediate stop before it makes a fast move back to the Z-index pulse, where a quick index reverses motion back to the position where the Z-index pulse was encountered.

- This is the similar to [Homing mode 8](#) except it returns to the encoder 0 point, after the home limit is encountered. If your system does not have an encoder, or the encoder pulse count is different from the motor pulse count, it is suggested that you use [Homing mode 8](#)

as this home mode will not function properly.

Homing Mode	Home Limit	End Limits	Z-index	Slow Down	Approaches Home at		Direction to Trigger Zero	Ending Point
					High Speed	Low Speed		
Mode 12		•	•	○	H		Dir of Home	Zero



Operation 1:

- Homing operation is started, accelerate from LSPD to HSPD.
- Immediate stop on encountering end of travel limit switch.
- Reverse direction, accelerating from LSPD to HSPD.
- Deceleration begins when specified number of Z-index pulses is encountered (ZCNT), zero position is set.
- Start a zero-point return move to the position where the Z-index pulse was encountered, accelerating from LSPD to HSPD then decelerating and stopping at 0 position.
- Homing is completed when the 0 position is reached.
- ERC pulse is sent.

Note: For homing modes 9 – 12, encoder feedback is required. When using a stepper motor and homing with modes 9 – 12, an encoder is required and the command pulse count and encoder pulse count must be equal.

Appendix B: Speed Settings

Appendix B: Speed Settings

HSPD value [PPS] †	Speed Window [SSPDM]	Min. LSPD value	Min. ACC [ms]	δ	Max ACC setting [ms]
1 - 65K	0, 1	1	2	50	$((\text{HSPD} - \text{LSPD}) / \delta) \times 1000$
65K - 130K	2	2	1	100	
130K - 325K	3	5	1	200	

325K - 650K	4	10	1	800	
650K - 1.3M	5	20	1	1500	
1.3M - 3.2M	6	50	1	3800	
3.2M - 6.55M	7	100	1	7500	

Speed settings

† If StepNLoop is enabled, the HSPD range values need to be transposed from PPS (pulse/sec) to EPS (encoder counts/sec) using the following formula: **EPS = PPS / Step-N-Loop Ratio**

In this section

- [Acceleration/Deceleration Range](#)
- [Acceleration/Deceleration Range – Positional Move](#)

B.1 Acceleration/Deceleration Range

B.1 Acceleration/Deceleration Range

The allowable acceleration/deceleration values depend on the **LSPD** and **HSPD** settings. The minimum accel/decel setting for a given high speed and low speed is shown in the table above. The maximum accel/decel setting for a given high speed and low speed can be calculated using the formula:

$$\text{Max ACC} = ((\text{HS} - \text{LS}) / \delta) \times 1000 [\text{ms}]$$

Note: The ACC parameter will be automatically adjusted if the value exceeds the allowable HSPD value range.

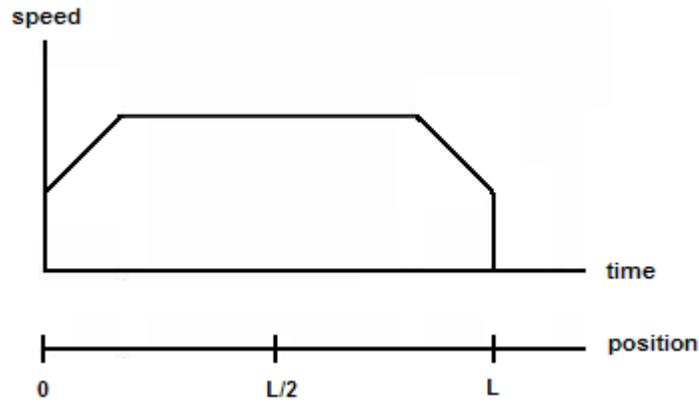
Examples:

- a) If **HSPD** = 20,000 pps, **LSPD** = 10,000 pps:
 - a. Min acceleration allowable: **1 ms**
 - b. Max acceleration allowable: $((20,000 - 10,000) / 50) \times 1,000 \text{ ms} = \mathbf{200,000 \text{ ms}}$ (200 sec)
- b) If **HSPD** = 900,000 pps, **LSPD** = 9,000 pps:
 - a. Min acceleration allowable: **1 ms**
 - b. Max acceleration allowable: $((900,000 - 9,000) / 1500) \times 1000 \text{ ms} = \mathbf{594,000 \text{ ms}}$ (594 sec)

B.2 Acceleration/Deceleration Range – Positional Move

B.2 Acceleration/Deceleration Range – Positional Move

When dealing with positional moves, the controller automatically calculates the appropriate acceleration and deceleration based on the following rules:



- 1) ACC vs. DEC 1: If the theoretical position where the controller begins deceleration is less than $L/2$, the acceleration value is used for both ramp up and ramp down.
- 2) ACC vs. DEC 2: If the theoretical position where the controller begins constant speed is greater than $L/2$, the acceleration value is used for both ramp up and ramp down.
- 3) Triangle Profile: If either (1) or (2) occur, the velocity profile becomes triangle. Maximum speed is reached at $L/2$.

Appendix C - Interpolation

Appendix C - Interpolation

Interpolation consists of generating data points between given coordinate axis positions and using this data to generate a path in space (linear, arc, circle, or helix) related to the coordinated axes.

With linear interpolation, these points establish a straight-line path through space of the associated axes. With Commander, an interpolation causes the axes to move simultaneously from the start to the end of the command. The interpolation calculates individual axis velocities to drive the axes along the desired path at the given feed rate. Thousands of intermediate coordinate points along the path are calculated between the start point and the end point of the move.

Commander core utilizes linear interpolation (linear coordinated motion between two to four axes), arc and circular Interpolation with any two axes, and helical Interpolation with X, Y and Z axes (arc and circle (X, Y) in combination with linear interpolation (Z-axis)). Interpolation operation can be executed up to the maximum pulse frequency.

In this section

- [Linear Interpolation](#)
- [Arc Interpolation](#)
- [Circular Interpolation](#)
- [Helical or Tangential interpolation \(Circular Interpolation Synchronized with the U-axis\)](#)
- [Additional notes regarding Interpolation functions](#)

C.1 Linear Interpolation

C.1 Linear Interpolation

To execute a linear interpolated move, specify the endpoint coordinates and the desired linear interpolation operation. The endpoint is specified as an incremental number of pulses from the current position on each axis, or the desired absolute position coordinates.

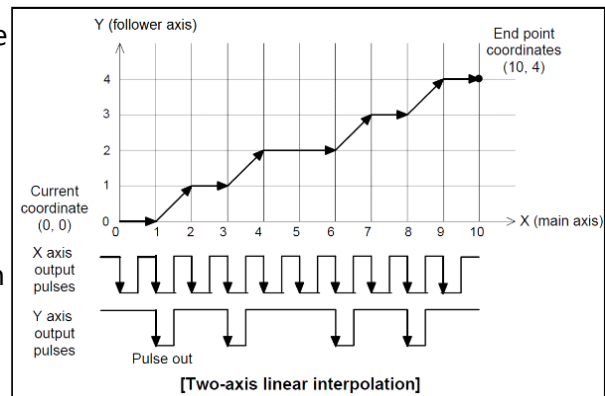
Commander automatically identifies the axis with the larger feed distance as the main axis, and the other axes as the secondary axes. The main axis is supplied a pulse rate tied to the specified feed rate, while the secondary axes are supplied with a reduced pulse rate based on the interpolation calculations.

Precision of linear interpolation

As shown in the figure on the right, linear interpolation executes an interpolation from the current coordinates to the end coordinates provided. The positional precision of a specified line during linear interpolation will be ± 0.5 LSB throughout the interpolation range.

"LSB" refers to the minimum feed unit for the system. It corresponds to the resolution of the mechanical system. (Size of the cells in the figure on the right.)

See the PCL6045BL manual, section 9.8.5 for more information.

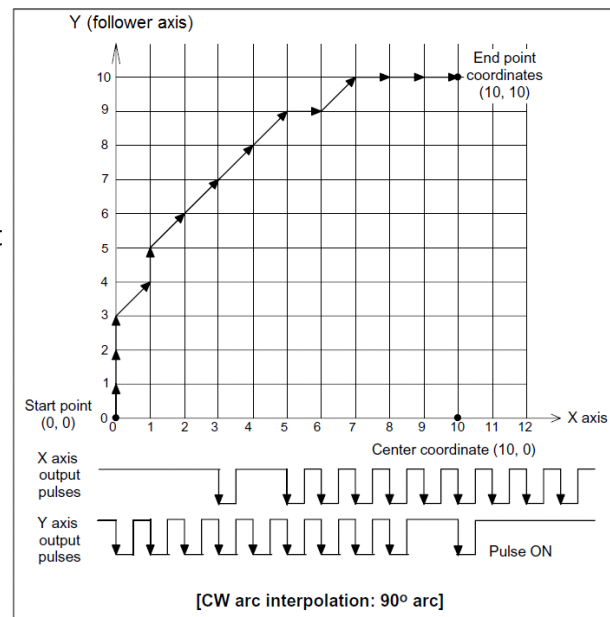


C.2 Arc Interpolation

C.2 Arc Interpolation

Executing an arc interpolation from the current position as the starting point requires the center coordinates and endpoint of the arc in degrees. Select either a CW or CCW arc interpolation direction and enter the arc center coordinates (incremental or absolute) and arc travel endpoint in degrees. The CW arc interpolation draws an arc from the current coordinates to the endpoint coordinates in a clockwise direction, using the center coordinates as the arc center. The CCW arc interpolation draws an arc in a counterclockwise direction.

The figure to the right is an example of drawing an arc with the X and Y axes in a CW arc interpolation operation



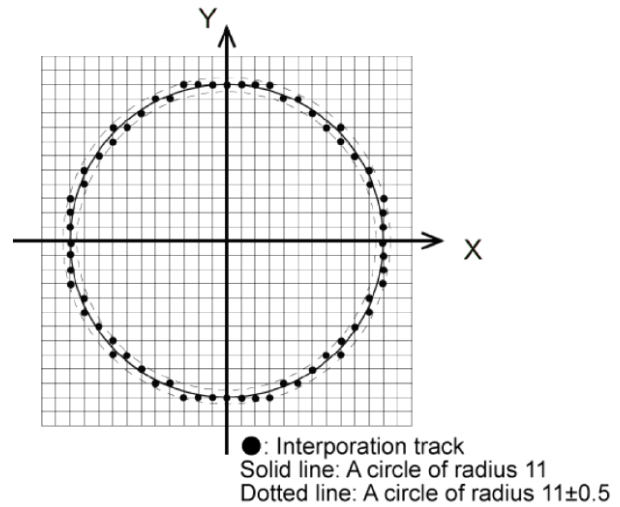
See the PCL6045BL manual section 9.8.8 for more information.

C.3 Circular Interpolation

C.3 Circular Interpolation

Executing circular interpolation from the current position as the starting point requires the center coordinates of the circle. Select either a CW or CCW circular interpolation direction and enter the circle center coordinates (incremental or absolute). The circular interpolation function draws a circle from the current position to the end coordinate, moving CW or CCW. The positional deviation from the specified curve is ± 0.5 LSB. The LSB corresponds to the resolution of the mechanical system (size of the cells in the figure).

The figure on the right is an example of how a simple circle with a radius of 11 units is created.

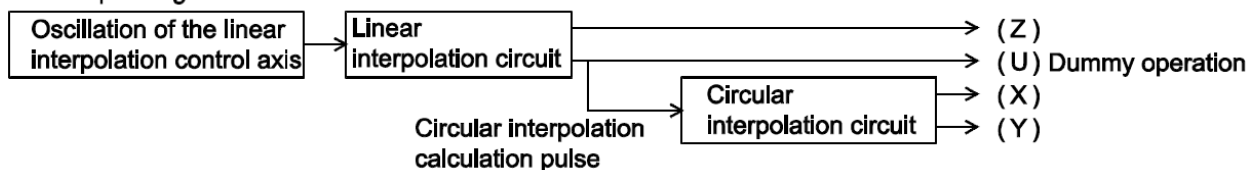


C.4 Helical or Tangential interpolation (Circular Interpolation Synchronized with the U-axis)

C.4 Helical or Tangential interpolation (Circular Interpolation Synchronized with the U-axis)

The Commander core allows a combination of circular or arc interpolation of X and Y axes with linear interpolation of the Z-axis to provide a helical path of motion. The U axis is used in the background as a virtual axis to aid in the performance of this function. This feature is used in applications that may require circular interpolation between the X and Y axes, and to adjust the angle of a jig toward an arc tangent point with the Z axis. Since the U axis plays a role in these moves, it cannot be used for any other purposes.

<Conceptual figure>



C.5 Additional notes regarding Interpolation functions

C.5 Additional notes regarding Interpolation functions

Acceleration/deceleration operations

Acceleration and deceleration (linear and S-curve) can be used with linear, arc, and circular interpolation operations.

Error stop

If any of the axes performing interpolated movements stops with an error, all of the axes performing interpolation will stop.

SD input

When SD input is enabled and turns ON for any axis during an interpolation operation, all axes will decelerate or decelerate and stop.

Idling control

If any axis is in idling range, none of the axes performing interpolation will accelerate.

Correction function

When a direction is changed by switching of quadrants during circular interpolation, backlash correction and slip correction control cannot be used.

Revision History

Date and Revision	Firmware Compatibility	Changes Made
October 2019 Version 1.0	V126BL	New Document
November 2019 Version 2.0	V127BL	Updated Table 6-44 Pins by function to agree with Note 2 on page 106. ERCP add mode 7, level output. PWM polarity corrected. MPGM, MPGD, and SCV support added to standalone. BF and BSTART command replay corrected. High-speed correction function added. An added calculation to get acc/dec rate. Expanded information on-the-fly speed change acc/dec rate.
December 2019 Version 2.1	V127BL	Updating SSPDM command information and HSPD operation related to SSPDM. SYNS command operation update.
February 2020 Version 2.2	V127BL	Table 2-12 MOSI0 changes to pin 2-13, Added notes regarding Linear Interpolation.
March 2021 Version 3.0	V201BL	Move manual to online electronic version. Adjustment of formatting to match new version. Update of Commander to RTOS. Corrected mistake in POL command in section 2.1.6.3 Corrected wording of INP function in section 2.1.6.2 Appendix C changed chapter numbers.
March 2022 Version 3.1	V201BL	Corrected wording of Note1 in section 6.2.1.2 Added links for each command to the command

		reference Added JKA, JKD commands to section 2.1.4 Updated Standalone Program Example 9, added links to download all Standalone Example Programs
April 2022 Version 3.2	V201BL	Added cross Links to Command Reference
November 2023 Version 3.3	V205BL	Added download links for software.
March 2034 Version 3.4	V205.2BL	Updates links
July 2025 Version 4.0	V301BL	Update manual for Commander Generation 5. 6.1 Dimensions 7.1 Electrical and Thermal Specifications

Last Build: 7/9/2026 6:29 PM

CMD Command Reference

Command Reference

The NPA Command Reference includes a comprehensive listing of all commands related to the **Commander** series Nippon Pulse controller products currently available. The manual is broken into the following sections;



Commands by

- [Associated functions](#)
- [Alphabetical listing](#) which has detailed information on each command.

- [Error codes](#)
 - [A-Script](#)
- Appendix
- [Home mode](#)



Introduction and Overview of the Commander

Commander motion controller core (CMD-4CR) is a first in class advanced 4 axes, standalone-programmable motion controller hybrid IC. The Commander core design provides OEMs the convenience of an off-the-shelf controller and the technology of a designed-from-scratch controller. As such it exhibits the best qualities of both types of controllers:

- Faster path-to-market
- Flexible design that fits a large range of applications
- No sacrifice to controller capabilities and processing speed
- Seamlessly scales from prototype to production with no software changes required
- Simple to use
- Cost-effective

The Commander core architecture allows for easy integration into custom hardware.

Communication to the Commander core can be established over USB, Serial (RS-485, I²C, SPI) or Ethernet. It is possible to download four standalone programs to the device and have it run independently of a host.

The Commander core is a CMOS hybrid IC designed to provide the oscillating, high-speed pulses needed to drive stepper motors and digital servomotors (pulse string input types) using various commands. It can offer various types of control over the pulse strings and therefore the motor performance. These include continuous feeding, positioning, and origin return, at a constant speed, and linear or S-curve acceleration/deceleration.

The Commander core controls four axes. It can control the linear interpolation of two to four axes, circular interpolation between any two axes, confirm controller operation status, and output an interrupt with various conditions. It also integrates an interface for servo motor drivers.

These functions can be used with simple commands. The intelligent design philosophy reduces the burden on the CPU. Commander is a robust and powerful controller with numerous operating capabilities. This table summarizes a variety of capabilities and features available in the Commander core software:

Feature	Description
Standalone control separated from PC	Operate motion without PC using standalone functionality
Programming language for stand-alone program with utility software	A-Script programming language, develop program on PC, compile and download to Commander
Operation check by utility software	Easy to use monitoring screens for motor status, I/O status, and system operation status

Compile, write and read standalone program by utility software	Easy to use tools to write, edit and debug software
Joystick operation with analog input (X-axis, Y-axis)	Utilize joystick for manual control of motion
13 types of homing mode	Flexibility in how machine mechanisms are set-up and homing operation is initiated
Manual pulse generator operation	
On-the-fly speed change (speed overridden during movement)	Quickly change operation speed manually on-the-fly
On-the-fly target position change (target position overridden during movement).	Target position can be updated manually on-the-fly
2 to 4-axis linear interpolation	Create a variety of paths utilizing interpolation options
2-axis circular and arc interpolation, including 3rd axis for Helical motion	Create circles, arcs or even create helical motion utilizing x, y, z axis
Sync output configuration	
Absolute positioning or incremental positioning can be selected	Flexibility of positioning coordinates as absolute from a home location or incremental
12 inputs and 12 outputs of general-purpose input/output signals	Flexibility in outputs, including configuration of general-purpose I/O, dedicated I/O and high-speed inputs

Firmware Compatibility

Firmware Compatibility

Commander Firmware Version History:

V122: Commander re-branded CMD-4CR (changed from PMX-4EX-CR)

V123: Continuations Sync mode added. SYNC[axis] bit 3 = 1

V124: Add feature SyncWindow to enable Continuations Sync only between two points

– Added commands: SYNWF[axis], SYNWO[axis], SYNMAX[axis], SYNMIN[axis]

V125: Fix issue with Continuations Sync mode

V126: Firmware fixes.

– PMX_4JX_OPTION version made the default standard. ID for PMX_4JX_OPTION is now CMD-4CR. This makes the following options standard and enabled for all CMD-4CR:

- o Servo Motor Interface
- o Latch Function
- o Sync Functions
- o 13 Homing modes
- o Joystick Functions
- o PWM Functions

V127 – Fix SSPD[axis] not working when S-curve enabled. PWM1, PWM2 now have same phase. Remapped MPE and MPG. Added code for compiled SCV. BF and BSTART now return error if buffer is off.

V128 – Add Sync Buffer Support. New commands: SYNBO – SYNC Buffer on Enabled, SYNBF – SYNC Buffer off Disabled. SYNBSTAT (Read Only)Returns sync buffer status (is it enabled,current

count in sync buffer, max number of entries available in sync buffer);

- o In sync buffer mode SYNPN(axis) command loads the sync buffer

V129 – improvements to StepnLoop operations. Added ability to make Homing Moves at Low speed (mode bit 4 =1) or HighSpeed (mode bit 4 =0).

Starting With V200 of the Commander Firmware, the commander was moved from a Single-Threaded bare metal system to a Multi-Threaded Real Time Operating System.

V200 – First version with RTOS, Ported from V127 and does not support StandAlone (A-Script) language. Support for Hardware Serial Number was added.

V201 – StandAlone is fully Supported, All features from V129 were implemented. Added improvements to on-the-fly, full support S-curve during on-the-fly, and SSPDM is now only required if changing speed levels, See Appendix Speed Settings. Support for soft-reset was added (SFLP[axis] SFLN[axis]). Jerk control for S-Curve was added (JKA[axis] JKD[axis]). CLR command now can be sent without axis information and clear errors from all axes.

V202 – Fixed known Bug with CMD-4EX-SA with SCV, POL, I/O, and Motor Driver Micro-steps commands. Added 4 axis in buffer mode (Vector Motion Mode).

V203 – Fixed known Bug with Standalone, Buffering, and USB communication.

V204 – Fixed known Bug from V203 with linear interpolation move running at 1pps, and UART, RS485 communication. Added new service commands.

V205 – Fixed bugs with buffering and circular interpolation moves. Fixed known Bugs from 204 with V0 - V4 being set with moves. Added additional checks to catch invalid inputs related to HSPD, LSPD, ACC, DEC, SCV, MPE, LT, SCV, SL, SFL, and SFLSD . Standardize error messages and expand replies to be more helpful. The CMD-4EX-SA version now includes the default settings for the CMD-4EX-SA series of controllers. The CMD-4CR version is updated to V205.1 This fixes an issue where the stored POL[axis] was over written by the default on power-cycle. V205.2 fixed an issue where PS[axis] was returning the correct speed values, but PS was reporting -1 for axis Y and incorrect values for axis Z and U.

V206 – Fixed known Bug from V205. Test for valid motion setting now check when starting move instead of as commands entered. Added speed improvements for standalone programs.

V207 – Fixed known Bug from V206. Added speed improvements for standalone programs. Fixed bug where ABORT[axis] at times would not stop all axis involved in linear interpolation move. Increased stability of StepnLoop, Abort, and Standalone programming.

V300 of the Commander Firmware is only supports Gen 5 commander hardware.

V300 – First version for Gen 5 commander hardware. Ported from V207 and does not support StandAlone (A-Script) language.

V300.1 – Fixed UART to support CR end of line. Compiler directive support for CMD2CR and CMD8CR models.

V300.2 – Added code to support EXSTP (external stop). Support for SN (Serial Number) added. Fixed encoder register, circle command, limit polarity.

V300.3 – Changes to MST command to support EXSTP.

V300.4 – Changes on board LED to provide blue heartbeat. Changes to motion buffer to support acceleration and deceleration with direction change.

V300.5 – Motion buffer fix to support first move in negative direction.

V300.6 – Major update to Motion Buffer. Functions related to the Motion Buffer have moved to MotionBuffer.cpp. The Data Structure has changed CommanderStruct.hpp. Added calcRampDownPoint() to p646_Motion_Control.cpp Added new Command BTUNE, RMG can now return rmg in addition to writing to asic. Motion buffer reviews every buffer move and sets

the correct profile for each move. Motion Buffer now has its own thread.

V300.7 – Analog Out added. Fixed RDP for buffer motion.

V300.8 – Support for StandAlone (A-Script) language added.

V301BL – First production release. Addition of boot-loader.

V302BL – Fixed known Bug with STORE that data was overwritten on startup.

Commander Operating principle

Commander Operating Principle

The Commander core has many modes of operation. These include single, multi-axis, interpolation, on-the-fly speed or target position changes, status monitoring functions, and I/O operations. Any or all of the different modes of operation can occur simultaneously without affecting the performance of the other processes.

Single-axis operations involve motion of a single axis and does not require interaction with any other axes.

Multi-axis operations involve motion of two or more axes but does not require interaction between the axes.

Interpolated operations involve two or more axes whose movements are interdependent. There are several types of interpolated moves.

Linear interpolation is when two or more axes work together to move along the hypotenuse between the starting and target coordinates. For linear interpolation, each axis moves in only one direction and start and stop at the same time. Only one interpolation set can be executed at a time.

Circular interpolation includes two-axes working together to move along the outer rim of a circle. For circular interpolation, each axis will change direction and speed as required to allow for a circular path of motion.

A popular subset of circular interpolation is **arc interpolation** where two axes work together to move along the outer rim of a circle for a set number of degrees.

Helical or tangential interpolation is when circular or arc interpolation of the X and Y-axis is combined with linear interpolation of the center point of the circle and the Z-axis. This allows for movement along a helical motion path.

The Commander core allows for all of these different moves:

- as independent moves
- any combination of the above moves
- or **buffered** together to allow for seamless **continuous interpolated** motion.

When operating a motor, it is possible to **jog** a motor at speed until it is told to **stop** or move to a predefined target **position**. The target position can be **absolute** to a 0 (home) point defined during one of the many **homing** processes or can be **incremental** or relative to the motor's current position. While moving it may be necessary to change the **target speed**, or while making a positioning move, it may be necessary to change **target position**. The Commander core can seamlessly do this without requiring any of the motors to stop. These operations are referred to as **on-the-fly speed change** or **on-the-fly target position change** respectively. It is also possible to use a **joystick** or **analog input** to control a motor. Using the **MPG** function, the motor can follow a master axis or a cam.

Regardless of the type of operation, the first step is to define your motion profile. The basics of a motion profile are: acceleration/deceleration type (**S-curve** or **Linear**/trapezoidal), the initial or **low speed**, the max or **high speed**, and the amount of time needed for **acceleration** and **deceleration**.

There are a number of inputs and outputs available on the Commander core. These can be broken up into two categories; dedicated high-speed (hS) and general-purpose. Dedicated inputs/outputs do not require further interaction from the microprocessor, and usually fall into one of two categories: 1) Inputs/Outputs that automatically cause a reaction that is fixed, and 2) Inputs/Outputs that automatically cause a reaction that can be configured. Safety inputs such as **end limits**, **slowdown** (near the limit) inputs, **home** (origin) switch, **latching** inputs, and **synchronization** outputs are all dedicated inputs. On the other hand, the **general-purpose** outputs will change state and require the microprocessor to detect and then act on their status. In addition, the microprocessor can request the current status of any input and of the various counters, output pulse, encoder, MPG, etc. along with the full status of each axis.

Functional Listing

Functional Listing

This chapter summarizes the commands (ASCII and Standalone) for the **Commander** and **Performax** series according to the following functional groups:

- **System Setup Settings** – Basic settings that are needed for the controller to operate correctly in your environment
 - o **System Setup** – Setting needed to communicate with the controller
 - o **Motor Setup** – Settings to interface with the motor driver
 - o **Encoder Setup** – Settings to configure a feedback system
 - o **Limits/Safety Setup** – Settings to configure the safety-related inputs
 - o **I/O Setup** – Settings related to inputs and outputs on the controller
- **Basic Motion Commands** – Basic motion-related commands
 - o **Motion settings** – Commands associated with defining the motion profile
 - o **Axis motion** – Commands associated with a single/multi-axis motion
 - o **Stop motion** – Commands associated with stopping motion
 - o **Homing** – Commands associated with the homing process
- **Advanced Functions** – Advanced functions
 - o **Coordinated/Interpolated motion** – Commands associated with coordinated/interpolated motion
 - Linear interpolation
 - Arc and arc tangent interpolation
 - Circular and helical interpolation
 - o **Buffer control Functions** – Commands associated with buffered operations
 - o **Joystick operation** – Commands associated with the configuration of analog inputs for X and Y axes control
 - o **StepNLoop/closed-loop Function** – Commands associated with StepNLoop function
 - o **Synchronization Function** – Commands associated with synchronization output functions
 - o **Latching Function** – Commands associated with latching input functions

- o [Manual Pulse Generator](#) function – Commands associated with configuration and operation of the MPG functions
- [System Status commands](#) – Commands associated with system status
- [Standalone commands](#) – Commands associated with standalone program operations
- Controller information
- **Service commands** – Commands for internal use only

Commands associated with more than one group are listed more than once.

System Setup Settings

System Setup Settings

Before operating your controller, it is necessary to set up the basic system settings. This section outline these commands. Depending on your system, not all settings are needed.

The controller's flash memory stores these settings. They are loaded automatically when the controller is powered up and stored to the flash memory by using the [STORE](#) command — review Section [2.1.7 Flash Memory](#) of the Commander manual for more information about the flash memory.

- [System Setup](#) – Settings needed to communicate with the controller
- [Motor Setup](#) – Settings to interface with the motor driver
- [Encoder Setup](#) – Settings to configure a feedback system
- [Limits/Safety Setup](#) – Settings to configure the safety-related inputs
- [I/O Setup](#) – Settings related to inputs and outputs on the controller

System Setup

System Setup

Nippon Pulse controllers support many different methods of communication, including USB, Serial, and Ethernet. There are settings for each communication method. For more information on the settings, see the following sections in the Commander manual:

- [General Connections and Setup](#)
- [USB Communication](#)
- [Serial Communication](#)

Command		Description	Available with:					Link
ASCII	Standalone		CMD-	PMX-	PMX-4ET	PMX-2ED	PMX-2EX	
DB	--	Baud rate	●	●		●	●	DB
EXST	--	External start enable/disable	●					EXST
ID	--	Commander-4CR identification number	●			●	●	ID
DN	--	Device name		●		●	●	
IERR	--	Ignore alarm/limit flags enable/disable	●	●	●	●	●	IERR
IP	--	IP address setting	●		●			IP
POL	--	Input and logic configuration	●			●	●	POL

PO	--			•	•			
STORE	--	Save system configuration settings to flash memory	•	•	•	•	•	STORE
VER	--	Return firmware version number	•	•	•	•	•	VER
%RST	--	Soft Reset	•					%RST
%SN	--	Hardware Serial Number	•					%SN

The black dots indicate the command is available for the listed controller family.

Motor Setup

Motor Setup

The Commander core is designed to be a master controller for both stepper and digital servomotors and their drivers. There are a few differences between how a stepper and a servo system react and the connections needed between the Commander core and their respective drivers. There are settings needed for setup of both motor types. For more information on the settings, see the following sections in the Commander manual:

- [Motor interface](#)
- Stepper [2.1.2.1 Stepper](#)
- Servo [2.1.2.2 Digital Servo](#)

Command		Description	Available with:					Link
ASCII	Standalone		CMD-	PMX-	PMX-4ET	PMX-2ED	PMX-2EX	
EO	EO	Axis enable/disable	•	•	•	•	•	EO
EOBOOT	--	Enable status set-up on boot-up	•	•	•	•	•	EOBOOT
ERC	--	Deflection counter enable/disable	•					ERC
ERCD	--	Error signal pulse delay	•					ERCD
ERCP	--	Error signal pulse width	•					ERCP
INP	--	In-position enable/disable	•					INP
POL	--	Input and logic configuration	•			•	•	POL
PO	--			•	•			

The black dots indicate the command is available for the listed controller family.

Encoder Setup

Encoder Setup

An encoder attached to a motor or system is a pulse counter that communicates the actual speed, direction, and position to the Commander core. The Commander core has four encoder inputs. For synchronous mode operations such as StepNLoop, the motor outputs and encoder inputs must be connected to the same axis. It is also possible to operate the encoder input in an asynchronous mode where it monitors an unrelated process elsewhere in the system. For more information on the settings needed to set up the encoder inputs, see the following section in the Commander manual:

- Encoder inputs [2.1.2.3 Encoder Inputs](#)

A related function is [StepNLoop](#).

Command		Description	Available with:					Link
ASCII	Standalone		CMD-	PMX-	PMX-4ET	PMX-2ED	PMX-2EX	
POL	--	Input and logic configuration	●			●	●	POL
PO	--			●	●			

The black dots indicate the command is available for the listed controller family.

Limits/Safety Setup

Limits/Safety Setup

The Commander core includes a number of inputs and outputs that have dedicated functions.

These include safety-related functions such as **emergency stop** and **limit switches**, informational inputs such as **slowdown** and **home**, and functional inputs such as **external start** and **(MPG) pulsar inputs**. For more information on the settings required to set up the dedicated inputs, see the following sections in the Commander manual:

- Dedicated Inputs [2.1.3.1 Dedicated Input/Outputs](#)
- Mechanical inputs [2.1.3.1.1 Mechanical input signals](#)
- Limit Switches [2.1.3.1.1.1 Limit Switches](#)
- Slowdown switches [2.1.3.1.1.2 Slowdown switches](#)
- Home switch [2.1.3.1.1.3 Home switch](#)
- Software inputs [2.1.3.1.1.4 Soft-Limits](#)

A related function is [Manual Pulse Generator](#).

Command		Description	Available with:					Link
ASCII	Standalone		CMD-	PMX-	PMX-4ET	PMX-2ED	PMX-2EX	
CLR	ECLEAR	Clears alarm and limit flags	●	●	●	●	●	CLR
ERC	--	Deflection counter enable/disable	●					ERC
ESTOP	--	Emergency stop	●					ESTOP
EXST	--	External start enable/disable	●					EXST
IERR	--	Ignore alarm/limit flags enable/disable	●	●	●	●	●	IERR
MST	MST	Axis status acquisition	●	●	●	●	●	MST
POL	--	Input and logic configuration	●			●	●	POL
PO	--			●	●			
SDC	--	Slow down deceleration signal configuration	●					SDC
SDE	--	Slow down deceleration signal enable/disable	●					SDE
SFL	--	Soft-limit Enable/Disable	●					SFL
SFLN	--	Negative Soft-limit setting	●					SFLN
SFLP	--	Positive Soft-limit setting	●					SFLP

SFLSD	--	Soft-limit Stop Mode	•					SFLSD
SFLSO	--	Soft-limit Source	•					SFLSO
STORE	--	Save system configuration settings to flash memory	•	•	•	•	•	STORE

The black dots indicate the command is available for the listed controller family.

I/O Setup

I/O Setup

The Commander core has a number of high-speed inputs and outputs that have dedicated functions. These include **position latching inputs** and **synchronization outputs**. If these functions are not needed for your application, they can also be used as **general-purpose inputs or outputs**.

The Commander core has a number of general-purpose inputs and outputs: **configurable digital I/O, analog inputs** and **PWM outputs**. There is a built-in function that will allow a **joystick** connected to the analog input to be used for control of the X and Y axes.

For more information on the settings required to set up the inputs and outputs, see the following sections in the Commander manual:

- High-speed Inputs/Outputs [2.1.3.2 High-speed Inputs/Outputs](#)
- High-Speed Inputs [2.1.3.2.1 High-Speed Digital Inputs](#)
- [Position Latching Inputs](#)
- High-Speed Outputs [2.1.3.2.2 High-Speed Digital Output](#)
- [Synchronization Outputs](#)
- General-purpose input/output [2.1.3.3 General-purpose input/output](#)
- Configurable Digital I/O [2.1.3.3.1 Configurable Digital I/O](#)
- Analog Input/output [2.1.3.3.2 Analog Input/output](#)
- Pulse-width modulated Outputs [2.1.3.3.3 Pulse-width modulated Outputs](#)

A related function is [Manual Pulse Generator](#).

Command		Description	Available with:						Link
ASCII	Standalone		CMD-	PMX-	PMX-4ET	PMX-2ED	PMX-2EX		
AI	AI	Analog Input	●	●		●	●	AI	
DI	DI	Digital input status	●	●	●	●	●	DI	
DIP	--	Digital input polarity setting	●	●	●			DIP	
DO	DO	Digital output status	●	●	●	●	●	DO	
DOBOOT	--	Digital output boot-up state	●	●	●	●	●	DOBOOT	
DOP	DOP	Digital output polarity setting	●	●	●			DOP	
IO	IO	General-purpose I/O status	●					IO	
IOBOOT	IOBOOT	General-purpose I/O boot-up state	●					IOBOOT	
IOCFG	IOCFG	General-purpose I/O settings at boot-up	●					IOCFG	
IOP	IOP	General-purpose I/O logic settings	●					IOP	

LT	LT	Enable/disable for latch function	•	•				LT
LTE	LTE	Previously latched encoder position	•					LTE
LT[axis]E				•				
LTP	LTP	Previously latched step position value	•					LTP
LT[axis]P				•				
LTS	LTS	Current latch status	•					LTS
LT[axis]S				•				
PWM	PWM	Pulse width modulation duty cycle	•					PWM
STORE	--	Save system configuration settings to flash memory	•	•	•	•	•	STORE
SYNC	SYNCFG	Comparison configurations for synchronization	•					SYNC
SYN[axis]C				•	•			
SYNF	SYNOFF	Comparator synchronization disabled	•					SYNF
SYN[axis]F				•	•			
SYNMAX	--	Maximum value of sync pulse	•					SYNMAX
SYNMIN	--	Minimum value of sync pulse	•					SYNMIN
SYNO	SYNON	Comparator synchronization enabled	•					SYNO
SYN[axis]O				•	•			
SYNP	SYNPOS	Comparator data settings	•					SYNP
SYN[axis]P				•	•			
SYNS	SYNSTAT	Comparator status	•					SYNS
SYN[axis]S				•	•			
SYNWF		Disable synchronization window	•					SYNWF
SYNWO		Enable synchronization window	•					SYNWO

The black dots indicate the command is available for the listed controller family.

Basic Motion Commands

Basic Motion Commands

Basic motion-related commands can be found in this section. a majority of applications will only require the commands found in this section. For more information on the settings, see the following sections in the Commander manual:

- [2.1.4 Motion Profile](#)
- [2.2 Single-axis and Multi-axis operations](#)
- **Motion settings** – Commands associated with defining the motion profile
- **Axis motion** – Commands associated with a single/multi-axis motion
- **Stop motion** – Commands associated with stopping motion
- **Homing** – Commands associated with the homing process

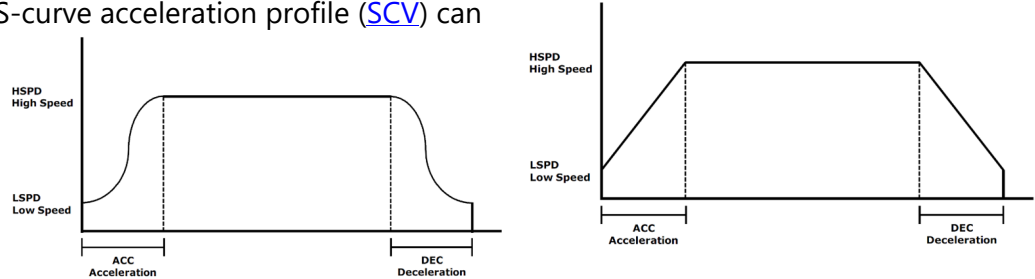
Motion Settings

Motion Settings

Any given motion profile can be plotted as a trapezoid, where Time is the base, Velocity is the height, acceleration rate is the angle and Distance is the area.

By default, the Nippon Pulse controllers use a linear acceleration profile as shown below. To reduce jerk, an S-curve acceleration profile ([SCV](#)) can also be used.

The percentage of S-curve can also be adjusted ([JKA](#), [JKD](#)).



For a typical

move, the axis will start moving at the low-speed setting ([LSPD](#)) and accelerate ([ACC](#)) to the high-speed setting ([HSPD](#)). Once at high-speed, the motor will move at a constant speed until it decelerates ([DEC](#)) from high-speed to low-speed and immediately stops.

The motion profile can be performed as an absolute move ([ABS](#)) from a [homed](#) location or incremental move ([INC](#)) from the current position. The [MM](#) command can be used to determine the current move mode of the controller.

For more information on the settings, see the following sections in the Commander manual:

- [Operating principle](#)
- [Motion Profile](#)
- [Absolute or Incremental](#)
- [Appendix B: Speed Settings](#)

Command		Description	Available with:						Link
ASCII	Standalone		CMD-4CR	PMX-4EX	PMX-4ET	PMX-2ED	PMX-2EX		
ABS	ABS	Absolute position coordinate mode	●	●	●	●	●	ABS	
ACC	ACC	Acceleration time in ms	●	●	●	●	●	ACC	
DEC	DEC	Deceleration time in ms	●	●	●	●	●	DEC	
HSPD	HSPD	High speed setting	●			●	●	HSPD	
HS	HSPD			●	●				
INC	INC	Incremental position coordinate mode	●	●	●	●	●	INC	
JKA	--	Jerk control for acceleration in %	●					JKA	
JKD	--	Jerk control for deceleration in %	●					JKD	
LSPD	LSPD	Low speed setting	●			●	●	LSPD	
LS	LSPD			●	●				
MM	--	Position coordinate mode status	●		●	●	●	MM	
SCV	SCV	S-curve acceleration enable/disable	●	●	●	●	●	SCV	

The black dots indicate the command is available for the listed controller family.

Axis motion

Axis motion

The **enable (EO)** output is most often used to turn the motor excitation circuit ON/OFF. The initial state of the enable outputs can be defined by setting the 4-bit **EOBOOT** register to the desired initial enable output value.

A jog move (**J**) is used to continuously move the motor without stopping. Unlike a Jog operation, a **Positioning** operation is a move with a specified stop point. The stop point will be either an **absolute** position in relation to the 0 start point that was defined during **homing**, or an **incremental** position from the motor's current position.

The Commander core supports the ability to change the moving speed and/or the target position on the fly. On-the-fly speed change (**SSPD**) will change the speed while an axis is in motion, at any point in the motion profile. On-the-fly target position change can be achieved using the **T** command, and will change the final destination of the motor while the motor is moving.

The speed range is set with the **SSPDM** command.

For more information on the settings, see the following sections in the Commander manual:

- [Output Pulse Counter](#)
- [Feedback Pulse Counter](#)
- [Stepper](#)
- [Digital Servo](#)
- [Motor Power](#)
- [Jog operation](#)
- [Positioning operation](#)
-
-

Command		Description	Available with:						Link
ASCII	Standalone		CMD-4CR	PMX-4EX	PMX-4ET	PMX-2ED	PMX-2EX		
E	E	Encoder value (axis)	●	●	●	●	●	E	
EO	EO	Axis enable/disable	●	●	●	●	●	EO	
EOBOOT	--	Enable status set-up on boot-up	●	●	●	●	●	EOBOOT	
EP	--	Encoder value (all)	●	●	●			EP	
J	JOG	Jog command	●	●	●	●	●	J	
P	P	Pulse counter value (axis)	●	●	●	●	●	P	
PP	PP	Pulse counter value (all)	●	●	●			PP	
PS	PS	Current operation speed value	●	●	●	●	●	PS	
SSPD	SSPD	On-the-fly speed change setting	●	●	●	●	●	SSPD	
SSPDM	SSPDM	Speed range setting	●	●	●	●	●	SSPDM	
T	--	On-the-fly target position setting	●	●	●	●	●	T	
X, Y, Z, U	X, Y, Z, U	Axis move command	●	●	●			X, Y, Z, U	

X, Y	X, Y					•	•	U

The black dots indicate the command is available on the listed controller family.

Stop Motion

Stop Motion

When the motor is performing any type of move, motion can be stopped abruptly ([ABORT](#)) or with deceleration ([STOP](#)). It is recommended to use decelerated stops so that there is less impact on the system. If an interpolation operation is in progress when a **STOP** or **ABORT** command is entered, all axes involved in the interpolation operation will stop.

The Commander series has an **emergency stop (/CEMG)** input for dedicated use as an emergency stop signal. When in operation, if the **/CEMG** input goes low or the [ESTOP](#) command is sent, all axes will stop immediately. No axis can operate while the /CEMG signal is low.

Stopping

Command		Description	Available with:						Link
ASCII	Standal one		CMD-	PMX-	PMX-	4FT PMX-	2ED PMX-	2EX	
ABORT	ABORT	Stop all axis motion	•	•	•	•	•		ABORT
ESTOP	--	Emergency stop	•						ESTOP
STOP	STOP	Decelerate and stop	•	•	•	•	•		STOP

The black dots indicate the command is available on the listed controller family.

Homing

Homing

Homing is a built in process by which the controller will move the motor while using the home, limit, slowdown, or Z-index inputs to determine the zero-reference position (home point). This point is then used as the reference for all subsequent moves.

The **Performax** series has five (5) different built-in home search routines. The **Commander** series has thirteen (13) different built-in home search routines.

Since the signal or signals used to define the home point varies depending on the homing mode selected, carefully select a homing mode that best matches the environment and available input signals of your system. Refer to [Home modes](#) for more information on each homing

mode.

2.2.4 Positioning operation

Command		Description	Available with:						Link
ASCII	Standalone		CMD-	PMX-	PMX-4ET	PMX-2ED	PMX-2EX		
H[axis][+/-] [mode]	HOME	Homing	•	•	•			HOME, H	
H[axis][+/-]	HOME[axis] [+/-]					•	•		
HL[axis][+/-]	HLHOME[axis] [+/-]					•	•		
L[axis][+/-]	LHOME[axis] [+/-]					•	•		
Z[axis][+/-]	ZOME[axis][+/-]					•	•		
ZH[axis][+/-]	ZHOME[axis] [+/-]					•	•		
HCA	--	Home correction amount				•	•		
LCA[axis]	--	Limit correction amount				•	•		
ZCNT	--	Z-index pulse counter	•					ZCNT	
ZMOVE[axis] [+/-]	ZMOVE[axis] [+/-]	Initiate Z-index count move	•					ZMOVE	

The black dots indicate the command is available on the listed controller family.

Cross reference for PMX home modes and CMD home modes.

Home modes cross reference - PMX and CMD

Home modes cross reference - PMX and CMD

With the release of the commander series of controllers the number and types of possible built-in homing modes is greatly increased. This is a cross reference which shows the corresponding homing mode for the commander that most closely matches the PMX homing modes.

2EX/2ED	4ET/4EX	CMD	
H[axis][+/-] >	H[axis][+/-]0	Mode 0	
L[axis][+/-] >	H[axis][+/-]1	Mode 6	(Note for 2EX and 3ED LCA is a value for how far away from the Limit the axis will move.
ZH[axis][+/-] >	H[axis][+/-]2	Mode 2	(Note for CMD ZCNT is added to go a set number of Z counts before setting Zero. others set Zero at first Z count.
Z[axis][+/-] >	H[axis][+/-]3	ZMOVE	(Note for CMD ZCNT is set to go a set number of Z counts before setting Zero. others set Zero at first Z count.
When using ZMOVE, the Counter is not			

automatically set to Zero

HL[axis][+/-] > H[axis][+/-]4 Mode 1 (Note for 2EX and 3ED HCA is a value for how far away from the Home sensor, before returning at low speed.

0 renv3 = 0

//Origin Return Operation bit 0000

//Standard homing using the home input only at high speed. Decelerated stop.

1 renv3 = 6

//Origin Return Operation bit 0110

//Home using the limit input. Immediate stop.

2 renv3 = 2

//Origin Return Operation bit 0010

//Home using the home input and z-index.

other renv3 = 1

//Origin Return Operation bit 0001

//Home using the home input at high and low speed.

Advanced Functions

Advanced Functions

In this Section

- [Interpolated Operations](#)
 - [Linear Interpolation](#)
 - [Circular Interpolation](#)
- [On-the-fly Speed/Target Position Change](#)
- [Buffer Operation](#)
 - [Interpolated Motion Buffer](#)
 - [Synchronization Output Buffer](#)
- [Synchronization](#)
- [Joystick](#)
- [StepNLoop](#)
- [Latching](#)
- [Manual Pulse Generator](#)

Interpolated operations

Interpolated operations

Interpolated operations involve two or more axes whose movements are interdependent.

Interpolation consists of generating data points between given coordinate axis positions and using this data to generate a path in space (linear, arc, circle, or helix) related to the coordinated axes. There are many types of interpolated moves. [Linear interpolation](#) is when two or more axes works together to cause movement along a straight line. For linear interpolation, each axis moves in only one direction, and start and stop at the same time.

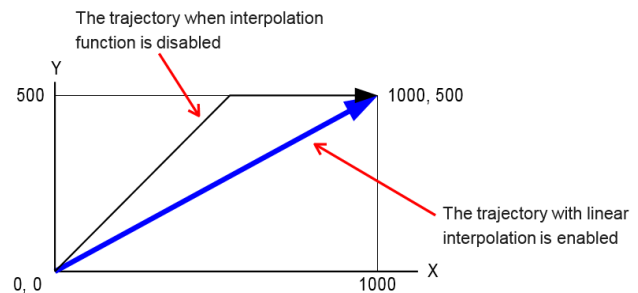
Circular interpolation is when two axes work together to cause movement along the outer rim of a circle. For circular Interpolation, each axis will change direction and speed as required to allow for a circular path of movement. A popular subset of circular interpolation is **arc interpolation**, when two axes work together to move along the outer rim of a circle for a set number of degrees. **Helix or tangential interpolation** is when a circular or arc interpolation of the X and Y-axis is combined with linear interpolation of the center point of the circle and the Z-axis. This allows for movement along a helical motion path. These moves can be **buffered** together to allow for seamless **continuous** motion.

Linear Interpolation

Linear interpolation

Linear interpolation is when two or more axes works together to cause movement along a straight line. For linear interpolation, each axis moves in only one direction, and start and stop at the same time. The **X**, **Y**, **Z**, and **U** commands can be combined into a single move command. Any combination of axes can be used for linear interpolation, and at least two axes must be referenced in order to perform linear interpolation. Only one linear interpolation set can be executed at a time.

By default, interpolation is enabled (**EINT=1**). While interpolation is enabled, linear interpolation operations are performed by writing positioning commands of multiple axes on the same line. There is no dedicated command for linear interpolation. If interpolation is disabled (**EINT=0**), writing positioning commands of multiple axes on the same line will not result in an interpolated move.



Command		Description	Available with:						Link
ASCII	Standalone		CMD-	PMX-	PMX-4ET	PMX-2ED	PMX-2EX		
EINT	EINT	Interpolation enable	●					EINT	
IX:Y:Z: [speed]		Interpolation move within buffer	●	●	●			I	
IX:Y						●	●		
IACC		acceleration setting for buffered interpolation moves		●	●			IACC	
XYZU	XYZU	Interpolation move used with EINT	●					XYZU	
XYZU	XYZU	Interpolated move		●	●				
XY	XY	Interpolated move				●	●		

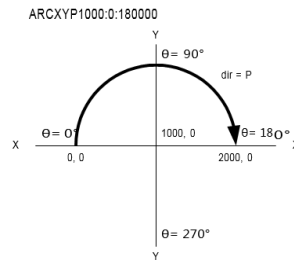
The black dots indicate the command is available on the listed controller family.

Circular Interpolation

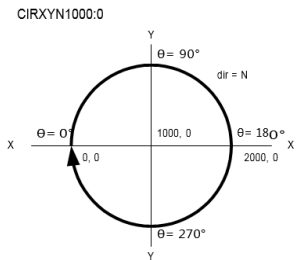
Circular Interpolation

Circular interpolation is when two axes work together to cause movement along the outer rim of a circle. The circular interpolation moves using the CIRP and CIRN commands will perform a complete 360° circle using any two axes. When less than 360° of motion is needed, then arc interpolation is used with the ARCP and ARCN commands. The two functions can be combined when more than 360° of motion is needed.

Trajectory of the circular interpolation operation specifying the end point



Circular interpolation trajectory drawing a perfect circle



Command		Description	Available with:						Link
ASCII	Standalone		CMD-4CR	PMX-4EX	PMX-4ET	PMX-2ED	PMX-2EX		
ARC	ARC	Arc interpolation	•					ARC	
ARCN, ARCP	ARCN, ARCP			•	•				
ARCT	ARCT	Arc interpolation with helix	•					ARC T	
CIR	CIR	Circular Interpolation	•					CIR	
CIRN, CIRP	CIRN, CIRP			•	•				
CIRT	CIRT	acceleration setting for buffered interpolation moves	•					CIR T	

The black dots indicate the command is available on the listed controller family.

On-the-fly-speed/target position change

On-the-fly-speed/target position change

The Commander core supports the ability to change the moving speed and/or the target position on the fly. For more information on the settings, see the following sections in the Commander manual:

- [On-the-fly speed change](#)
- [On-the-fly target position change \(Target position change\)](#)

Command		Description	Available with:					Link
ASCII	Standalone		CMD-	PMX-	PMX-4ET	PMX-2ED	PMX-2EX	
SSPD	SSPD	On-the-fly speed change setting	•	•	•	•	•	
SSPDM	SSPDM	Speed range setting	•	•	•	•	•	
T	--	On-the-fly target position setting	•	•	•	•	•	

The black dots indicate the command is available on the listed controller

Buffer Operation

Buffer Operation

There are two buffering loops available in the Nippon Pulse controllers: one for interpolated motion (available on all 4-axis controllers) and one for synchronization output (available on the Commander series of controllers). These loops are used to pre-buffer commands, allowing for continuous, high speed operation. The buffers are independent from each other and can be used one at a time or simultaneously.

For normal operation, commands are written to an operation register one at a time, and executed immediately.

For buffer operation, the supported commands are written to the buffer registers instead. These commands are then moved from the buffer to the operation register in a first-in-first-out order automatically. This process is continued until all the buffer registers are empty or the buffer mode is turned off.

Command		Description	Available with:						Link
ASCII	Standalone		CMD-	PMX-	PMX-4ET	PMX-2ED	PMX-2EX		
Interpolated Motion Buffer									
BF	BUFOFF	Disable buffer operation	●	●	●			BF	
BO	BUFON	Enable buffer operation	●	●	●			BO	
BSTART	ISTART	Start buffer operation	●					BSTART	
BSTAT	—	Buffer status	●					BSTAT	
I	XYZ, XY	Buffered linear interpolation move settings	●	●	●	●	●	I	
IACC	—	Acceleration setting for buffered interpolation moves		●	●			IACC	
Synchronization Output Buffer									
SYNBF	—	Disable Sync Position Buffer	●					SYNBF	
SYNBO	—	Enable Sync Position Buffer	●					SYNBO	
SYNBSTAT	—	Sync position buffer status	●					SYNCSTAT	

The black dots indicate the command is available on the listed controller family.

Interpolated Motion Buffer

Interpolated Motion Buffer

The interpolated motion buffer mode allows for prebuffering of interpolated moves. This allows for the creation of unique patterns without any delay between moves. Each move has its own constant speed setting.

For normal operation, commands are written to an operation register one at a time, and executed immediately.

For buffer operation, the coordinated motion commands are written to the buffer registers instead.

When the start buffer motion (**BSTART**) command is sent, the coordinated motion command in Buffer0 is moved to the operation register. Then the coordinated motion command in Buffer1 is moved to Buffer0 and so on, in a first-in-first-out order. This process is continued until all the buffer registers are empty or the buffer mode is turned off with the (**BF**) command.

To turn the buffer mode on and off, use the **BO** and **BF** commands, respectively. The buffered move operation cannot be used while StepNLoop is enabled.

When buffer mode is enabled, motion will not start until the appropriate command has been sent, even if there are commands in the buffer. This allows the buffer to be fully loaded before any move is processed. The command **BSTART** starts feeding the buffered commands to the operation register.

The status of the buffer operation can be read using the **BSTAT** command. The return value will contain the buffer enable status, buffer start and end positions, and the number of available buffer registers.

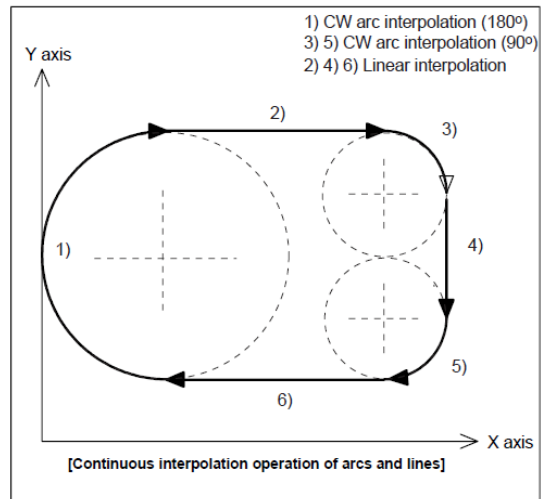
The buffer operation only supports motion of the X, Y, and Z-axis. In addition, only linear, circular and arc interpolation moves can be added to the buffer. As such only the linear, circular and arc interpolation commands below are supported during buffered operation.

Linear interpolation

In ASCII mode, a buffered move command will have the syntax **I[X pos]:[Y pos]:[Z pos]:[speed]**. For example, the **I1000:2000:3000:5000** command indicates position (1000, 2000, 3000) at a speed of 5000. If buffered mode is not enabled, the **I** command will not be processed by the controller.

In standalone mode, the buffered move command will have the syntax **X[pos]Y[pos]Z[pos]**, and the speed for the move command will use the current global high-speed setting. If buffered mode is not enabled, these standalone commands will be processed as linear interpolation commands.

Circular and Arc interpolation



The [ARC](#) and [CIR](#) commands can be used when buffer mode is enabled. The syntax for these commands is the same in ASCII and standalone mode.

Synchronization Output Buffer

Synchronization Output Buffer

The synchronization output buffer mode allows for prebuffering of synchronization target position. This allows for automatic high speed reloading of the the synchronization target position register when in the synchronization at position mode.

For normal operation, a target position is written to the register. The next target position sent overwrites the previous position.

For buffer operation, when synchronization output buffer mode is turned on with the ([SYNFO](#)) the first target positron ([SYNP](#)) is written to the register and supplicant target positrons sent are written to the buffer registers. When the synchronization output conditions are met (counter = target positron), the next target positron in the buffer is written to the register automatically on a first-in-first-out order. This process is continued until all the buffer registers are empty or the synchronization output buffer mode is turned off with the ([SYNBF](#)) command.

Synchronization

Synchronization

The Commander core provides four high-speed synchronization outputs. This feature allows the controller to perform a high-speed comparison between the encoder or pulse **position counter** and a **set condition (comparator)**, which is set with the [SYNP\[axis\]](#) command.

When the condition is met, the corresponding synchronization output is enabled. The timing between the condition being met and the synchronization output being triggered is 150 to 200ns.

There are three modes for the synchronization outputs to be enabled, which can be set with the [SYNC\[axis\]](#) command:

1. **At position mode** (counter = comparator)
2. **> or < position mode** (counter is either > or < than the comparator)
3. **Continuous mode** (every comparator position)

For **continuous mode**, and **at position mode**, the output can be output in both directions, or limited to only one direction (positive or negative) of movement.

For **> or < position mode**, you can select only Counter > Comparator or Counter < Comparator.

Each axis has a designated high-speed synchronization output. While synchronization output is enabled for an axis, the corresponding digital output cannot be manually controlled by the user. See corresponding synchronization output for each axis in table synchronization output below.

Axi s	Output
X	DO1/SYNC x

Y	DO2/SYNC y
Z	DO3/SYNC z
U	DO4/SYNC u

Synchronization output

Use the [SYNO\[axis\]](#) command to enable and the [SYNF\[axis\]](#) command to disable the synchronization feature. To read the synchronization status, use the synchronization status [SYNS\[axis\]](#) command (2-bit).

Command		Description	Available with:					Link
ASCII	Standalone		CMD-	PMX-	PMX-4ET	PMX-2ED	PMX-2FX	
SYNC	SYNCFG	Comparison configurations for synchronization	•					SYNC
SYN[axis]C				•	•			
SYNF	SYNOFF	Comparator synchronization disabled	•					SYNF
SYN[axis]F				•	•			
SYNMAX		Maximum value of sync pulse	•					SYNMAX
SYNMIN		Minimum value of sync pulse	•					SYNMIN
SYNO	SYNON	Comparator synchronization enabled	•					SYNO
SYN[axis]O				•	•			
SYNP	SYNPOS	Comparator data settings	•					SYNP
SYN[axis]P				•	•			
SYNS	SYNSTAT	Comparator status	•					SYNS
SYN[axis]T		Sync output pulse width		•	•			
SYNWF		Disable synchronization window	•					SYNWF
SYNWO		Enable synchronization window	•					SYNWO

The black dots indicate the command is available on the listed controller family.

Joystick

Joystick

Joystick control is available on the Commander core for the X and Y-axis. When this mode is enabled, the pulse speed and direction output for the X and Y axis can be controlled by their corresponding analog input. See the axis to analog input assignment in the table below.

Axis	Analog Input
X	AI1
Y	AI2

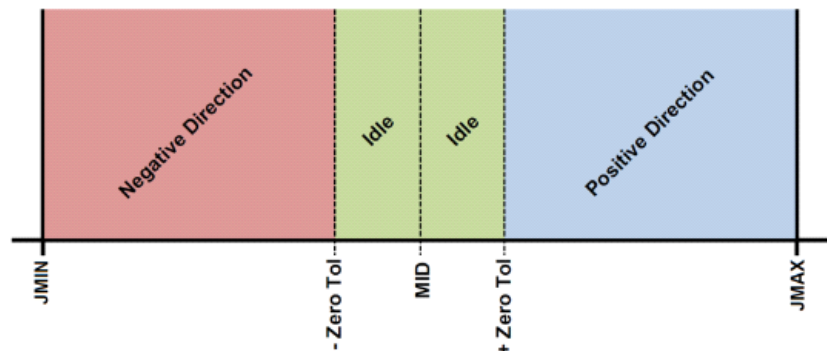
To enable or disable joystick control for an axis, use the ASCII command [JENA](#) or the standalone command [JOYENA](#).

For the Commander core to properly translate the analog inputs into a safe pulse speed and direction output for your system, you first must define what is acceptable and safe for your system. This includes defining the relation between voltage and **direction** of travel, and the zero-tolerance (or dead) zone during which there is no output from the Commander core. The **safe speeds** are defined as the maximum speed the analog input can command and maximum rate of speed change. The joystick **positioning** soft limits define where the analog inputs can command a system to move.

Direction control

The [JMAX\[axis\]](#) command can be used to define the maximum analog input value for the specified axis. Similarly, the [JMIN\[axis\]](#) command can be used to define the minimum analog input value for the specified axis. *MID* represents the zero-joystick position. There is no command corresponding to *MID*. This is an internally calculated value using the following formula:

$$MID = ((JMAX[axis] - JMIN[axis]) / 2) + JMIN[axis]$$



During joystick operation, analog input of [JMIN\[axis\]](#) to *MID* represents negative joystick direction and analog input of *MID* to [JMAX\[axis\]](#) represents positive joystick direction. Setting the value of [JMAX\[axis\]](#) to be less than the value of [JMIN\[axis\]](#) will result in a change of direction polarity during joystick operation.

The ASCII command [JTOL\[axis\]](#) can be used to define the zero-tolerance (or dead) zone around the *MID* value. No movement will occur while the analog input value is within the zero-tolerance zone. The zero-tolerance parameter will be on the positive and negative side of the *MID* point to define the zero-tolerance zone. See the figure to the right for details.

Speed control

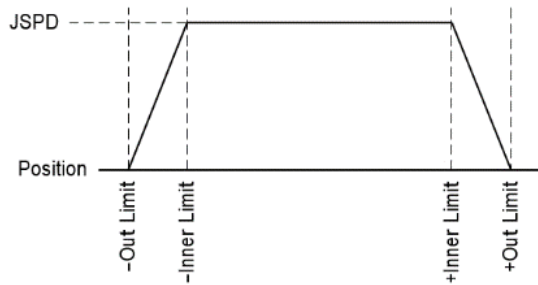
The maximum joystick speed for the X and Y-axes is set using the ASCII command [JSPD\[axis\]](#). For standalone mode, the [JOYHS\[axis\]](#) command should be used. This parameter will define the speed of the specified axis at the maximum and minimum analog input values.

The maximum allowable speed change (delta) for the X and Y axes is set using the ASCII command [JDEL\[axis\]](#). For standalone mode, the [JOYDEL\[axis\]](#) command is used. This parameter will control the acceleration/deceleration of the specified axis while the joystick mode is enabled.

Position control

Joystick control also has soft limits, which are defined by a negative outer limit, negative inner limit, positive inner limit, and a positive outer limit, using the **JLIM[number]** command.

When moving in a positive direction, as soon as the positive inner limit is crossed, the speed is reduced. Speed will continue to decline as the position moves closer to the outer limit value. If the position reaches the positive outer limit, the joystick speed is set to zero and movement in the positive direction is prohibited. The same behavior is applicable to the negative direction and negative limits. The figure to the right represents the relationship between the joystick speed and inner and outer limits.



2.1.3.3.2 Analog Input/output

Command		Description	Available with:					Link
ASCII	Standalone		CMD-					
AI		Analog Input	•					AI
JDEL		Joystick speed delta	•					JDEL
JENA	JOYENA	Joystick enable/disable	•					JENA
JE								
JF								
JO								
JLIM		Joystick limits	•					JLIM
JL								
JMAX		Joystick maximum voltage setting	•					JMAX
JMIN		Joystick minimum voltage setting	•					JMIN
JSPD	JOYHS	Joystick maximum speed setting	•					JSPD
JTOL		Joystick dead band range	•					JTOL
JV		Joystick speed, delta, and tolerance configuration						

The black dots indicate the command is available on the listed controller family.

StepNLoop

StepNLoop

Command	Description	Available with:	Link

ASCII	Standalone		CMD-					
SL	SL	StepNLoop operation enable/disable	•					SL
SLA		Retry count for StepNLoop operation	•					SLA
SLE		Error range for StepNLoop operation	•					SLE
SLR		Pulse conversion value for StepNLoop	•					SLR
			•					
SLS	SLS	StepNLoop operation status	•					SLS
SLT		Position range for StepNLoop operation	•					SLT

The black dots indicate the command is available on the listed controller family.

Latching

Latching

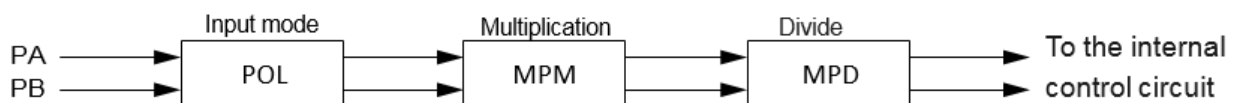
Manual Pulse Generator

Manual Pulse Generator

The MPG input can be used for a number of functions such as:

- 1) A manual pulse generator, which allows for fine adjustment of an axis.
- 2) Cam follower, which will allow the axis to follow a cam axis based on its encoder output.
- 3) Electronic gearing, which allows an axis to follow the input commands but at a different rate of speed (faster or slower).

The Commander core has a dedicated MPG input for each axis. To enable the MPG input, set the [MPE\[axis\]](#) command to 1. To disable the MPG input, set the [MPE\[axis\]](#) command to 0. The MPG inputs are controlled by three commands: MPG signal input mode ([POL\[axis\]](#)), multiplication ([MPM\[axis\]](#)) and division ([MPD\[axis\]](#)). These work together to set how the pulses from the MPG input controls the motion of an axis.



[MP\[axis\]](#) can be written to set an MPG position or read to see the current position. The counter value here is the final counts after the electronic gearing stage.

[2.1.3.1.4 Manual pulse generator \(MPG\) inputs](#)

[2.5.2 Manual pulse generator \(MPG\) operation](#)

Command	Description	Available with:	Link

ASCII	Standalone		CMD-	PMX-	PMX-4ET	PMX-2ED	PMX-2EX	
MPD	--	Manual pulse generator division ratio	•					MPD
MPE	--	Manual pulse generator enable/disable	•					MPE
MPM	--	Manual pulse generator multiplier	•					MPM
MP	--	Manual pulse generator counter	•					MP

The black dots indicate the command is available on the listed controller family.

Standalone Programming Commands

Standalone Programming Commands

The Commander core supports standalone programming, which allows the controller to execute a user-defined program that is stored in its internal memory. The standalone program can be run independently of USB and serial communication, or while communication is active.

Commander core utilizes A-Script, a BASIC-like language that helps the programmer quickly write programs for motion control projects, including conditional statements, subroutines, and mathematical expressions. The programming syntax uses a series of standalone commands that allows the programmer to build single and multi-thread programs that can be compiled and run in the controller. By writing lines of code using the set of commands that make up the A-Script language, set-up, control of motion, management of inputs and outputs, and error handling can all be addressed. A basic program consists of a start line (**PRGxx**) and an end line (**END**). All code is written between the PRG and END lines. Use the Utility software to write, compile, download, upload, and execute programs.

For more information on A-Script, how the programs are formatted and to see example code, see the Commander Manual Section [2.7 Standalone program Specification](#) and Section [5.0 Standalone Language Specification](#).

Command		Description	Available with:					Link
ASCII	Standalone		CMD- 256	PMX- 256	PMX- 4ET	PMX- 2ED	PMX- 2EX	
--	;	Use in-line with code to add comments	•	•	•	•	•	;
--	DELAY	Wait a prescribed amount of time before next command is executed	•	•	•	•	•	DELAY
GS	GOSUB	Go to sub routine	•	•	•	•	•	GS
--	IF, ELSEIF, ELSE, ENDIF	Conditional loop IF, ELSEIF, ELSE/ENDIF	•	•	•	•	•	IF
--	PRG, END	Program begin and end lines	•	•	•	•	•	PRG/END

SA	--	Compiled program line		•	•	•	•	SA
SACTRL	SR	Standalone program control	•					SACTRL
SAMAX	--	Maximum number of standalone program lines						SAMAX
SAP	--	Error handling setting	•	•	•			SAP
SASTAT	--	Standalone program status	•	•	•	•	•	SASTAT
SLOAD	--	Activation of program on power-up	•	•	•	•	•	SLOAD
SPC	--	Current program counter	•	•	•	•	•	SPC
--	SUB, ENDSUB	Subroutine begin and end lines	•	•	•	•	•	SUB
V	V	Standalone variable	•	•	•	•	•	V
--	WAIT	Delay next command until current command is complete	•	•	•	•	•	WAIT
--	WHILE, ENDWHILE	Conditional loop WHILE/ENWHILE	•	•	•	•	•	WHILE
--	+, -, *, /, <, >, =, !=	Mathematical operands and conditionals	•	•	•	•	•	

System Status

System Status

The Commander core allows for each device to have a unique identification number, allowing up to 64 devices to be connected to the same host. If multiple devices are connected to the host, the identification number for each device should be unique.

By default, all Commander cores are shipped from the factory with identification number 01.

The current identification number of a device can be found by reading the **ID** command. Also, the device name will always include the current identification number. For USB communication the device name will be "CMD-4CR-xx" where xx is the identification number of the device. For serial communication, the "DeviceName" is only the identification number of the device.

In order to change the identification number of the device, first store the desired number using the **ID** command. Note that this value must be within the range [00-99].

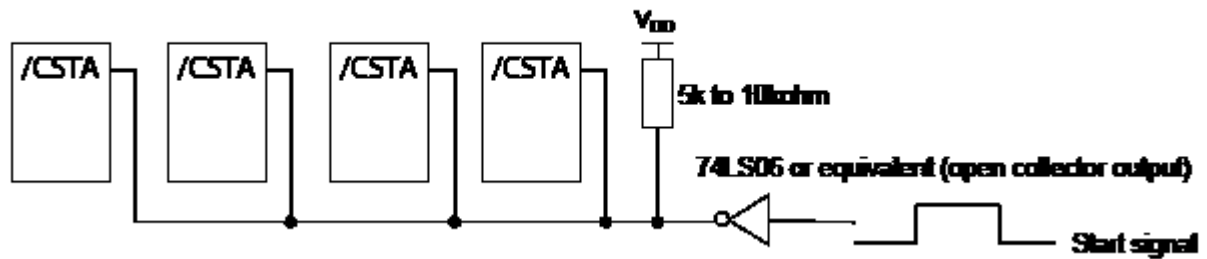
For example, the command **ID=02** can be sent to change the identification number to 02. To save a modified identification number to the flash memory of the device, use the **STORE** command. The new identification number will not take effect until after a power cycle.

The current version of firmware can be found with the **VER** command.

It is possible to use an external trigger to start a defined motion profile. This is done by connecting the /CSTA pins as shown in Figure 2-1. The /CSTA line needs an external pullup

resistor to VDD.

To start simultaneously from an external circuit, connect the CMDs as follows.



By default, the external start input is disabled (**EXST=0**), which will allow motion to start immediately upon receipt of any motion command. If the external start input is enabled (**EXST=1**) then any motion command sent will wait for the **/CSTA** input to go low, before motion is started. The Commander core can buffer up to 3 motions on a first-in first-out order; they will start one at a time each time the **/CSTA** input goes low. The logical input of the **/CSTA** input cannot be changed.

Command		Description	Available with:					Link
ASCII	Standalone		CMD-	PMX-	PMX-4ET	PMX-2ED	PMX-2EX	
CLR	ECLEAR	Clears alarm and limit flags	●	●	●	●	●	CLR
DB		Baud rate	●	●		●	●	DB
D		StepNLoop deviation counter value	●					D
DX				●	●	●	●	
DN		Device name		●		●	●	
E		Encoder value (axis)	●	●	●	●	●	E
EDEC		Unique global deceleration status		●	●	●	●	
EP		Encoder value (all)	●	●	●			EP
ERC		Deflection counter enable/disable	●					ERC
ERCD		Error signal pulse delay	●					ERCD
ERCP		Error signal pulse width	●					ERCP
EXST		External start enable/disable	●					EXST
ID		Commander-4CR identification number	●			●	●	ID
IERR		Ignore alarm/limit flags enable/disable	●	●	●	●	●	IERR
INP		In-position enable/disable	●					INP
LCA		Global limit correction value				●	●	
MST	MST	Axis status acquisition	●	●	●	●	●	MST
P	P	Pulse counter value (axis)	●	●	●	●	●	P
PE		Position values for all encoders		●	●			
EP			●					EP
POL		Input and logic configuration	●			●	●	POL

PO				•	•			
PP		Pulse counter value (all)	•	•	•			PP
PS	PSX	Current operation speed value	•	•	•	•	•	PS
PWM		Pulse width modulation duty cycle	•					PWM
REG		Registers	•					REG
SDC		Slow down deceleration signal configuration	•					SDC
SDE		Slow down deceleration signal enable/disable	•					SDE
STORE		Save system configuration settings to flash memory	•	•	•	•	•	STORE
VER		Returned firmware version number	•	•	•	•	•	VER

Environment setting

Environment setting

Command		Description	Available with:					Link
ASCII	Standalone		CMD-	PMX-	PMX-4ET	PMX-2ED	PMX-2EX	
CLR	ECLEAR	Clears alarm and limit flags	●	●	●	●	●	CLR
DB		Baud rate	●	●		●	●	DB
D		StepNLoop deviation counter value	●					D
DX				●	●	●	●	
DN		Device name		●		●	●	
E		Encoder value (axis)	●	●	●	●	●	E
EDEC		Unique global deceleration status		●	●	●	●	
EP		Encoder value (all)	●	●	●			EP
ERC	~	Deflection counter enable/disable	●					ERC
ERCD	~	Error signal pulse delay	●					ERCD
ERCP	~	Error signal pulse width	●					ERCP
EXST		External start enable/disable	●					EXST
ID		Commander-4CR identification number	●			●	●	ID
IERR		Ignore alarm/limit flags enable/disable	●	●	●	●	●	IERR
INP	~	In-position enable/disable	●					INP
LCA		Global limit correction value				●	●	
MST	MST	Axis status acquisition	●	●	●	●	●	MST
P	P	Pulse counter value (axis)	●	●	●	●	●	P
PE		Position values for all encoders		●	●			
EP			●					EP

POL	~	Input and logic configuration	•			•	•	POL
PO				•	•			
PP		Pulse counter value (all)	•	•	•			PP
PS	PSX	Current operation speed value	•	•	•	•	•	PS
PWM		Pulse width modulation duty cycle	•					PWM
REG		Registers	•					REG
SDC	~	Slow down deceleration signal configuration	•					SDC
SDE		Slow down deceleration signal enable/disable	•					SDE
STORE		Save system configuration settings to flash memory	•	•	•	•	•	STORE
VER		Returned firmware version number	•	•	•	•	•	VER

Controller information

Controller information

The Commander core allows for each device to have a unique identification number, allowing up to 64 devices to be connected to the same host. If multiple devices are connected to the host, the identification number for each device should be unique.

By default, all Commander cores are shipped from the factory with identification number 01.

The current identification number of a device can be found by reading the **ID** command. Also, the device name will always include the current identification number. For USB communication the device name will be "CMD-4CR-xx" where xx is the identification number of the device. For serial communication, the "DeviceName" is only the identification number of the device.

In order to change the identification number of the device, first store the desired number using the **ID** command. Note that this value must be within the range [00-99].

For example, the command **ID=02** can be sent to change the identification number to 02. To save a modified identification number to the flash memory of the device, use the **STORE** command. The new identification number will not take effect until after a power cycle.

The current version of firmware can be found with the **VER** command.

Command		Description	Available with:						Link
ASCII	Standalone		CMD-	PMX-	PMX-4ET	PMX-2ED	PMX-2EX		

ID		Commander-4CR identification number	•			•	•	ID
VER		Returned firmware version number	•	•	•	•	•	VER

Service Commands

Service Commands

The following commands are unpublished and intended for use only in Nippon Pulse software, or by Nippon Pulse personal. Service commands are only supported via ASCII commands.

Command	Description	Available with:	Link
ASCII		CMD-4CR	
%BLD	Places the commander and DLL into boot-load mode	•	%BLD
%DID	Returns the SOC type ID	•	%DID
%RST	Resets (soft power cycle) the commander	•	%RST
%SBLK	Sets or clears the blink bit for LED output	•	%SBLK
%SN	Returns the SOC UUID ID (Hardware serial number)	•	%SN
%SRID	Auto set of device ID with reset	•	%SRID
REG[RD/WR][axis] [reg]	Read / Write the register of the p645 motion controller	•	REG
%SMAC	Set read device MAC address	•	% SMAC

External Connections

External Connections

The motion controller includes several types of inputs/and outputs which are not directly related to the motion of the motor. Digital I/O is broken down into three categories:

- 1) **Dedicated I/O** – high speed inputs or outputs which have a dedicated function which cannot be changed;
- 2) **High speed I/O** – high speed inputs or outputs with processing times of <200ns that have certain functions associated to them, but can also be used as general purpose inputs or outputs; and
- 3) **General purpose I/O**, which have no built-in associated functions and are defined by the user.

There are also analog inputs, analog outputs and PWM outputs.

Command		Description	Available with:					Link
ASCII	Standalone		CMD-	PMX-	PMX-4ET	PMX-2ED	PMX-2EX	
AI		Analog Input	•	•		•	•	AI
DI		Dedicated input status	•	•	•	•	•	DI
DIP		Digital input polarity setting	•	•	•			DIP
DO	DO	General-purpose output status	•	•	•	•	•	DO
DOBOOT		Digital output boot-up state	•	•	•	•	•	DOBOOT
DOP		Digital output polarity setting	•	•	•			DOP
IO	IO	General-purpose I/O status	•					IO
IOBOOT		General-purpose I/O boot-up state	•					IOBOOT
IOCFG		General-purpose I/O settings at boot-up	•					IOCFG
IOP		General-purpose I/O logic settings	•					IOP
LT		Enable/disable for latch function	•	•				LT
LTE		Previously latched encoder position	•					LTE
LT[axis]E				•				
LTP		Previously latched step position value	•					LTP
LT[axis]P				•				
LTS		Current latch status	•					LTS

LT[axis]S				•				

Dedicated I/O

Dedicated I/O

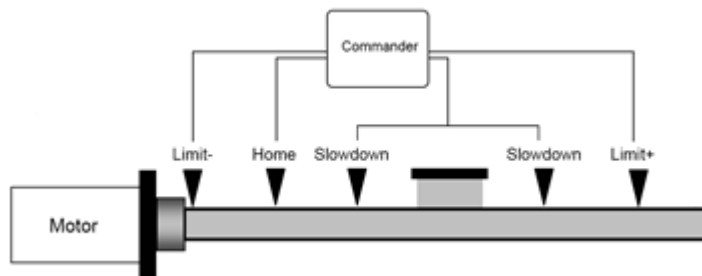
Motion controllers includes a number of inputs and outputs that have dedicated functions. These include safety-related functions such as **emergency stop** and **limit switches**, informational inputs such as **slowdown** and **home**, and functional inputs such as **external start** and **(MPG) pulsar inputs**.

Mechanical input signals

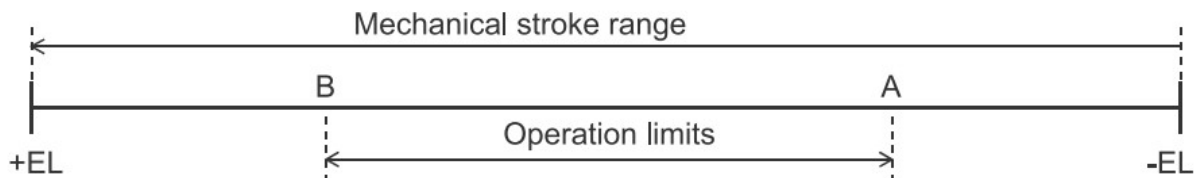
The following four signals can be input for each axis:

- 1) +L (Limit +) When this signal turns ON while operating in the positive direction, the motor stops immediately, or decelerates and stops.
- 2) -L (Limit -) When this signal turns ON while operating in the negative direction, the motor stops immediately, or decelerates and stops.
- 3) SD (Slowdown) Used as a deceleration signal or a deceleration stop signal by a software setting.
- 4) H (Home) Input signal for a Homing operation.

The input logic for these signals can be changed with software.



The Commander also has a soft-limit function. The Soft-limits can be used in a system where physical limits are impossible or for setting operation limits within the allowable mechanical stroke limits.



Command		Description	Available with:					Link
ASCII	Standalone		CMD-4CR	PMX-4EX	PMX-4ET	PMX-2ED	PMX-2EX	
CLR	ECLEAR	Clears alarm and limit flags	•	•	•	•	•	CLR
DB		Baud rate	•	•		•	•	DB
D		StepNLoop deviation counter value	•					D
DX				•	•	•	•	
DN		Device name		•		•	•	
E		Encoder value (axis)	•	•	•	•	•	E
EDEC		Unique global deceleration status		•	•	•	•	
EP		Encoder value (all)	•	•	•			EP
ERC		Deflection counter enable/disable	•					ERC
ERCD		Error signal pulse delay	•					ERCD
ERCP		Error signal pulse width	•					ERCP
EXST		External start enable/disable	•					EXST
ID		Commander-4CR identification number	•			•	•	ID

IERR		Ignore alarm/limit flags enable/disable	•	•	•	•	•	IERR
INP		In-position enable/disable	•					INP
LCA		Global limit correction value				•	•	
MST	MST	Axis status acquisition	•	•	•	•	•	MST
P	P	Pulse counter value (axis)	•	•	•	•	•	P
PE		Position values for all encoders		•	•			
EP			•					EP
POL		Input and logic configuration	•			•	•	POL
PO				•	•			
PP		Pulse counter value (all)	•	•	•			PP
PS	PSX	Current operation speed value	•	•	•	•	•	PS
PWM		Pulse width modulation duty cycle	•					PWM
REG		Registers	•					REG
SDC		Slow down deceleration signal configuration	•					SDC
SDE		Slow down deceleration signal enable/disable	•					SDE
STORE		Save system configuration settings to flash memory	•	•	•	•	•	STORE

Limit Switches

Limit switches, most commonly seen in linear mechanisms to restrict motion to a safe range, are used to identify that the motor has reached its end of travel. If either the positive (**L+**) or negative (**L-**) signal becomes active, the controller is instructed to immediately stop all advancement in that direction but will allow movement in the opposite direction. As such it is important to ensure the placement of the positive limit at the end of positive travel and negative limit at the end of negative travel. The polarity of limit inputs is set with the **POL[axis](Bit3)**.

When designing your limit circuit, here are a few points to keep in mind:

- 1) It is especially important that the motor must stop when the signal is active.
- 2) To prevent nuisance limit faults that can stop your system, ensure that electrical noise does not trigger your limits as active.
- 3) To operate in a fail-safe manner, the signal line must be designed to trigger as active when the cable connecting the limit switch to the Commander core becomes disconnected.

With the above points in mind, think about your limit circuit and the active logic level in your system.

Limit error

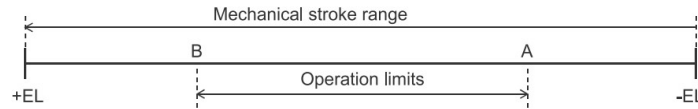
By default **IERR=0**, which will trigger a limit error **MST[axis](bit8~9)** when a limit switch is set to active. Once the limit error is set, use the **CLR[axis]** command to clear the error in ASCII mode or the **ECLEAR[axis]** command in standalone mode. **ECLEAR[axis]** only works in SUB 31 as part of the standalone error handling function. See section **2.7.5 Error Handling** for more information.

When using the end limit during the homing modes, the **ERC** (deflection counter clear) signal can be setup to clear the limit errors that accrue due to homing automatically.

The limit error states can be ignored by setting **IERR=1**. In this case, the motor will still stop when the appropriate switch is triggered; however, it will not enter an error state.

Soft-Limits

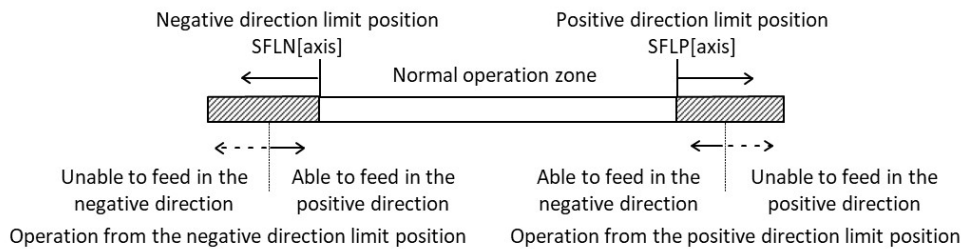
End limits are frequently used to detect the mechanical stroke limits. The Commander also has a soft-limit function. The soft-limits can be used in a system where physical limits are impossible, or for setting operation limits within the allowable mechanical stroke limits.



The soft-limit function uses a comparator for a positive direction limit ([SFLP\[axis\]](#)) and a second comparator for a negative direction limit ([SFLN\[axis\]](#)) to stop the axis based on the axis position related to the comparator's results and the operation direction. The [SFLSO\[axis\]](#) command selects the encoder counter (**1**) or pulse generator counter (**0**) for the soft-limit axis position. It is safest to use the encoder counter, if an encoder is available, as it will allow the tracking of the motor even if an outside force moves the motor.

When the soft-limit function is enabled, the axis stops immediately when the motor reaches the soft-limit setting at the low-speed setting (LSPD). At the high-speed setting (HSPD), then stopping is executed based on the setting of the [SFLSD\[axis\]](#).

While deceleration stop is selected as the process when soft-limit triggered ([SFLSD\[axis\]=1](#)), when the axis reaches the soft-limit while at high-speed, that axis stops using deceleration at the rate set by [DEC](#). When some other process is specified ([SFLSD\[axis\]=0](#)), or while in a constant (low) speed, that axis stops immediately. While the axis is at or beyond the soft-limit setting, the axis can not be moved in the direction in which the soft-limit is enabled. However, it can move in the opposite direction.



Once the parameters are set, the soft-limit function can be enabled by setting [SFL\[axis\]=1](#) and disabled by setting [SFL\[axis\]=0](#).

Slowdown switch

The slowdown switch (**SD**) input indicates a point to begin deceleration. If the slowdown switch inputs are active when the motor is operating at HSPD speed and this signal is triggered, the motor will start to decelerate.

The 2-bit **SDE[axis]** command enables the SD inputs and sets the latching function. By default, **SDEX=0** disables the X-axis SD inputs. The inputs can be enabled without a latch function using **SDEX=1**, or enabled with a latch function **SDEX=3**.

Bit	Description	Setting	
0	SD input	0 – Disabled	1 – Enabled
1	SD latching	0 – Disabled	1 – Enabled

Table 2-3 SDE command

The **SDC[axis]** command determines the specific operation of the SD input. By default, **SDC[axis]=0** sets the operation, when triggered, to decelerate the motor to LSPD without stopping. When **SDC[axis]=1**, the operation is set to decelerate and stop when the SD input is triggered.

Value	Description
0	Decelerate only
1	Decelerate and stop

Table 2-4 SDC command

Triggering the SD input when **SDC=1** will set the SD stop flag, **MST[axis](bit17)**. The SD stop flag will not prevent further motion, but can only be cleared by using the **CLR[axis]** command in ASCII mode or the **ECLEAR[axis]** command in standalone mode. **ECLEAR[axis]** only works in SUB 31 as part of the standalone error handling function. See section 2.7.5 Error Handling for more information.

The SD input, when triggered, will cause the motor to slowdown regardless of the direction of travel.

Figure 2-7 shows an example of a setup using the slowdown inputs. These inputs are mainly used in homing operations. In this case the axis would move at HSPD, trigger the slowdown input, allowing the motor to decelerate to LSPD, before it triggers the home input.

Other uses for the slowdown input are placing SD limits at each end of travel just before the end limits. This allows for two options. The slowdown inputs function as a soft limit (**SDC=1**) that decelerates and stops the motion, allowing the end limits to operate as emergency hard end limits. Or they can just allow the motor to decelerate (**SDC=0**) to LSPD before triggering the end limits. The polarity of the slowdown inputs is set with the **POL[axis](Bit6)**.

Home

Emergency stop

The Commander core has an **emergency stop (/CEMG)** input for dedicated use as an emergency stop signal. When in operation, if the **/CEMG** input goes low or the **ESTOP** command is sent, all axes will stop immediately. No axis can operate while the **/CEMG** signal is low.

The logical input of the **/CEMG** input cannot be changed.

Note: In a normal stop operation, the final pulse width is normal. However, in an emergency stop operation, the final pulse width may not be normal. As such the motor drivers may not recognize the last pulses, and therefore the Commander core internal counter may not match the mechanical position. Therefore, after an emergency stop, it is recommended that you perform a homing operation to match the command and mechanical positions.

Emergency stop error

No axis can operate while the **/CEMG** signal is low.

By default when **IERR=0**, any axis that is operating when the **/CEMG** signal goes low or the **ESTOP** command is sent will set an **EMG** error flag **MST[axis](bit16)** for that axis. In addition, any axis that tries to start operating while the **/CEMG** signal is low will set an **EMG** error flag for that axis. Any axis with a set **EMG** error flag cannot operate until the **EMG** error flag has been cleared. However, if an axis was not moving when the **/CEMG** or **ESTOP** was triggered, the **EMG** error flag will not be set for that axis, and thus that axis can operate immediately after the **/CEMG** input is cleared.

To clear the **EMG** error flag, use the **CLR[axis]** command in ASCII mode or the **ECLEAR[axis]** command in standalone mode. **ECLEAR[axis]** only works in SUB 31 as part of the standalone error handling function. See section 2.7.5 Error Handling for more information.

The **EMG** error flags are ignored by setting **IERR=1**. In this case, all axes will still stop when **/CEMG** is triggered or **ESTOP** command is sent; and all axes can operate again immediately after the **/CEMG** input is cleared.

High speed I/O

High speed I/O

Command		Description	Available with:					Link
ASCII	Standalone		CMD-	PMX-	PMX-4ET	PMX-2ED	PMX-2EX	
AI		Analog Input	•	•		•	•	AI
DI		Dedicated input status	•	•	•	•	•	DI
DIP		Digital input polarity setting	•	•	•			DIP
DO	DO	General-purpose output status	•	•	•	•	•	DO
DOBOOT		Digital output boot-up state	•	•	•	•	•	DOBOOT
DOP		Digital output polarity setting	•	•	•			DOP
IO	IO	General-purpose I/O status	•					IO
IOBOOT		General-purpose I/O boot-up state	•					IOBOOT
IOCFG		General-purpose I/O settings at boot-up	•					IOCFG

IOP		General-purpose I/O logic settings	•					IOP
LT		Enable/disable for latch function	•	•				LT
LTE		Previously latched encoder position	•					LTE
LT[axis]E				•				
LTP		Previously latched step position value	•					LTP
LT[axis]P				•				
LTS		Current latch status	•					LTS
LT[axis]S				•				

General purpose I/O

General purpose I/O

Command		Description	Available with:					Link
ASCII	Standalone		CMD-	PMX-	PMX-4ET	PMX-2ED	PMX-2EX	
AI		Analog Input	•	•		•	•	AI
DI		Dedicated input status	•	•	•	•	•	DI
DIP		Digital input polarity setting	•	•	•			DIP
DO	DO	General-purpose output status	•	•	•	•	•	DO
DOBOOT		Digital output boot-up state	•	•	•	•	•	DOBOOT
DOP		Digital output polarity setting	•	•	•			DOP
IO	IO	General-purpose I/O status	•					IO
IOBOOT		General-purpose I/O boot-up state	•					IOBOOT
IOCFG		General-purpose I/O settings at boot-up	•					IOCFG
IOP		General-purpose I/O logic settings	•					IOP
LT		Enable/disable for latch function	•	•				LT
LTE		Previously latched encoder position	•					LTE
LT[axis]E				•				
LTP		Previously latched step position value	•					LTP
LT[axis]P				•				
LTS		Current latch status	•					LTS
LT[axis]S				•				

Summary Error Codes by Controller Family

Summary Error Codes by Controller Family

Description	Error Code	Used with:				
		CMD-4CR	PMX-4EX	PMX-4ET	PMX-2ED	PMX-2EX
ASCII command is not recognized	?Unknown Command: [Command]	•				
	?[Command]		•	•	•	•
Move command could not be processed. One of more of the required axes may be in an error state or in motion	?Error Moving	•				
Move command could not be processed. Alarm signal is in the on state	?Error Alarm	•				
	?ALARM		•	•		
Move command could not be processed; (+) End Limit is in the on state or (-) End Limit is in the on state	?Error +Limit	•				
	?Error -Limit	•				
	?LIMIT		•	•		
Move command could not be processed. EMG signal is in the on state	?Error ESTOP	•				
Move command could not be processed. StepNLoop control is in error state	?Error SNL	•				
Index number is invalid	?Invalid Index	•				
Command cannot be processed; a standalone program is in an error state	?Program Error	•				
Call to subroutine using the GS command is not valid, specified subroutine has not been defined	?Sub not initialized	•	•	•	•	•
Move can not be added to a buffer register; all registers are full	?Buffer Full	•				
	?BUFFER FULL		•	•		
Buffer command is not valid; buffer mode is disabled	?Buffer Disabled	•				
	?BUFOFF			•		
SSPD command cannot be processed; buffer mode is enabled	?Buffer Enabled	•				
	?BUFON			•		
SSPD command cannot be processed; SSPD mode is not defined, out of range, or S-curve is enabled	?SSPD Error	•	•			
	?SSPD mode not initialized			•	•	•
	?Bad SSPD Command			•		
	?Speed out of range			•	•	•
	?S-curve on			•	•	•
T[] command is invalid because a target position move is not in operation	?ABS/INC is not in operation				•	•
A move command has been issued while the axis is in StepNLoop error	?Clear SNL Error				•	•
Override operation failed	?Override Failed					
An attempt to set the timer register while it is running has been made	?Timer Running		•	•		
A command has been issued while in an error state	?ERRORED		•			
	?StatError			•		
A circle or arc interpolation move has been issued while the controller is in incremental mode	?In Incremental mode		•	•		
The index for the command sent to the controller is not valid.	?Index out of range		•	•	•	•
A move or position change command is sent while the controller is outputting pulses	?PULSING		•	•		
	?Moving				•	•
On-the-fly speed change attempted during an interpolated move	?ICommandOn				•	•
Invalid parameter input	?Invalid Answer				•	•
Low speed parameter is out of range	?Low speed out of range				•	•
The setting for SFLN is larger or equal to SFLP	?SoftLimit values not valid	•				
The setting for SFLSO is not set to a valid input for Soft-	?SoftLimit source not valid	•				

limits						
The value set for SYNMIN was larger than the value set for SYNMAX	?SYNMIN > SYNMAX	•				

Alphabetical Command Listing

Alphabetical Command Listing

This section lists all the commands in alphabetical order, along with detailed definitions and examples. The definition of each command includes the following items:

- A explanation of the operation or task the command preforms.
- **Applies to:** List of witch Nippon Pulse controllers the command is available on, along with an Identification of the command name for ASCII mode and if applicable Standalone mode.

While every command is supported in ASCII mode, not every command is available in standalone mode.

For simplicity purpose, the rest of the definition will only reference the ASCII command. Unless otherwise noted, the syntax, reply, and examples between the ASCII and Standalone and between different controllers will be the same with only the identification name changing.

- **Syntax:** Characteristics of the command for both Read and Write as applicable
- **Reply:** Expected reply
- **Notes:** Exceptions or special instructions or warnings associated with this command
- **See Also:** Similar commands associated with the specific operation

Summary Command Reference Table by Controller Family

Summary Command Reference Table by Controller Family

The chart below is a collection of all controller commands associated with the current Nippon Pulse products PMX and CMD families. Similar commands used for the same function by different controller families are grouped together. The black dots indicate the command is available on the listed controller family.

Function	Command Syntax		Available with:					
	ASCII	Standalone	CMD-4CR	PMX-4EX	PMX-4ET	PMX-2ED	PMX-2EX	
Separator for comments in Standalone program code		;	•	•	•	•	•	
Stop all axes immediately, buffer mode disabled	ABORT	ABORT	•	•	•	•	•	
Stop an individual axis immediately, buffer mode disabled	ABORT[axis]	ABORT[axis]	•	•	•	•	•	
Set absolute position coordinate mode	ABS	ABS	•	•	•	•	•	
Read global acceleration time setting (ms)	ACC	ACC	•	•	•	•	•	
Set global acceleration time setting (ms)	ACC=[value]	ACC=[value]	•	•	•	•	•	
Read axis acceleration time setting (ms)	ACC[axis]	ACC[axis]	•	•	•	•	•	
Set axis acceleration time setting (ms)	ACC[axis]=[value]	ACC[axis]=[value]	•	•	•	•	•	
Read analog input status (mV)	AI[1-2]	AI[1-2]	•	•		•	•	

Arc interpolation move CCW direction	ARC[A1][A2]N[C1]:[C2]:[Θ]		•				
	ARCN[X]:[Y]:[Θ]	ARCN[X]:[Y]:[Θ]		•	•		
Arc interpolation move CW direction	ARC[A1][A2]P[C1]:[C2]:[Θ]		•				
	ARCN[X]:[Y]:[Θ]	ARCN[X]:[Y]:[Θ]		•	•		
XY arc interpolation CCW with Z-axis linear interpolation move to target	ARCTN[C1]:[C2]:[Θ]:[target]	ARCTN[C1]:[C2]:[Θ]:[target]	•				
XY arc interpolation CW with Z-axis linear interpolation move to target	ARCTP[C1]:[C2]:[Θ]:[target]	ARCTP[C1]:[C2]:[Θ]:[target]	•				
Disable buffer mode	BF	BUFOFF	•	•	•		
Enable buffer mode	BO	BUFON	•	•	•		
Begin processing the buffer	BSTART	ISTART	•				
Buffer status	BSTAT		•				
Circular interpolation in the CCW direction	CIR[A1][A2]N[C1]:[C2]		•				
	CIRN[X]:[Y]	CIRN[X]:[Y]		•	•		
Circular Interpolation in the CCW Direction	CIR[A1][A2]P[C1]:[C2]		•				
	CIRP[X]:[Y]	CIRP[X]:[Y]		•	•		
XY circular interpolation in the CCW direction with Z-axis linear interpolation move to target	CIRTN[C1]:[C2]:[target]	CIRTN[C1]:[C2]:[target]	•				
XY circular interpolation in the CW direction with Z-axis linear interpolation move to target	CIRTP[C1]:[C2]:[target]	CIRTP[C1]:[C2]:[target]	•				
Clear axis alarm, limits, or StepNLoop error	CLR[axis]	ECLEAR[axis]	•	•	•	•	•
Returns the StepNLoop deviation counter for the specified axis	D[axis]		•				
	DX[axis]			•	•	•	•
Read baud rate of controller	DB	DB	•	•		•	•
Set baud rate of controller	DB=[value]	DB=[value]	•	•		•	•
Read global deceleration time setting (ms)	DEC	DEC	•	•	•	•	•
Set global deceleration time setting (ms)	DEC=[value]	DEC=[value]	•	•	•	•	•
Read axis deceleration time setting (ms)	DEC[axis]	DEC[axis]	•	•	•	•	•
Set axis deceleration time setting (ms)	DEC[axis]=[value]	DEC[axis]=[value]	•	•	•	•	•
Set delay time (ms), delays time next operation starts		DELAY	•	•	•	•	•
Reads status of all general-purpose digital inputs	DI	DI	•	•	•	•	•
Reads status of specific general-purpose digital input	DI[1-8]	DI[1-8]	•	•	•	•	•
Reads the digital input polarity	DIP		•	•	•		
Sets the digital input polarity	DIP=[pol]		•	•	•		
Read device name or identification number	DN			•		•	•
	ID		•				
Set device name or identification number	DN=[value]			•		•	•
	ID=[value]		•				
Reads the status of all general-purpose digital outputs	DO	DO	•	•	•	•	•
Sets the status of all general-purpose digital outputs	DO=[value]	DO=[value]	•	•	•	•	•
Reads the status of a specific digital output	DO[1-8]	DO[1-8]	•	•	•	•	•
Sets the status of a specific digital output	DO[1-8]=[value]	DO[1-8]=[value]	•	•	•	•	•
Reads the configuration of all general purpose digital outputs on boot-up	DOBOOT		•	•	•	•	•
Sets the configuration of all general purpose digital outputs on boot-up	DOBOOT=[value]		•	•	•	•	•
Reads the digital output polarity	DOP		•	•	•		
Sets the digital output polarity	DOP=[pol]		•	•	•		
Returns the StepNLoop deviation counter for the specified axis	DX[axis]			•	•	•	•
Reads encoder value of specific axis	E[axis]	E[axis]	•	•	•	•	•
Sets the encoder value of a specific axis	E[axis]=[value]	E[axis]=[value]	•	•	•	•	•
Reads the unique deceleration status	EDEC			•	•	•	•

Enables/disables the unique deceleration status	EDEC=[0-1]			•	•	•	•	•
Reads the interpolation enable status	EINT		•					
Enables/disables interpolation function	EINT=[value]		•					
Reads enable status of all axes	EO	EO	•	•	•	•	•	•
Sets enable status of all axes	EO=[value]	EO=[value]	•	•	•	•	•	•
Reads enable status of specific axis	EO[no]	EO[no]	•	•	•	•	•	•
Sets enable status of specific axis	EO[no]=[value]	EO[no]=[value]	•	•	•	•	•	•
Reads boot-up enable status of all axes	EOBOOT		•	•	•	•	•	•
Sets enable status for all axes on boot-up	EOBOOT=[value]		•	•	•	•	•	•
Reads the enable output polarity	EOP			•	•			
Sets the enable output polarity	EOP=[value]			•	•			
Reads encoder values of all axis	EP		•					
	PE			•				
Reads the enable status of the deflection counter clear output for a specific axis	ERC[axis]		•					
Sets the enable status of the deflection counter clear output for a specific axis	ERC[axis]=[value]		•					
Reads the pulse delay set value	ERCD		•					
Sets the pulse delay value (2-bits)	ERCD=[value]		•					
Reads the pulse width set value	ERCP		•					
Sets the pulse width value (3-bits)	ERCP=[value]		•					
Initiates emergency stop (same as triggering /CMEG)	ESTOP		•					
Reads the status of the external start enable	EXST		•					
Enables/disables the external start function	EXST=[value]		•					
Initiates a defined subroutine (0 – 31)	GS[0-31]	GOSUB [0-31]	•	•	•	•	•	•
Homing operation	H[axis][+/-][mode]	HOME[axis][+/-][mode]	•	•	•	•	•	•
	H[axis][+/-]	HOME[axis][+/-]		•	•	•	•	•
	HL[axis][+/-]	HLHOME[axis][+/-]		•	•	•	•	•
	L[axis][+/-]	LHOME[axis][+/-]		•	•	•	•	•
	Z[axis][+/-]	ZOME[axis][+/-]		•	•	•	•	•
	ZH[axis][+/-]	ZHOME[axis][+/-]		•	•	•	•	•
Reads global home correction amount	HCA						•	•
Sets global home correction amount value	HCA=[value]						•	•
Reads home correction amount for a specific axis	HCA[axis]						•	•
Sets home correction amount value for a specific axis	HCA[axis]=[value]						•	•
Read global high-speed setting	HSPD	HSPD	•				•	•
	HS			•	•			
Set global high-speed value	HSPD=[value]	HSPD=[value]	•				•	•
	HS=[value]			•	•			
Read high speed setting for a specific axis	HSPD[axis]	HSPD[axis]	•				•	•
	HS[axis]			•	•			
Set the high-speed setting for a specific axis	HSPD[axis]=[value]	HSPD[axis]=[value]	•				•	•
	HS[axis]=[value]			•	•			
Setting for buffered interpolation move	I[X target]:[Y target]:[Z target]:[speed]		•	•	•			
XY Linear interpolated move	I[X target]:[Y target]						•	•
Read automatic acceleration setting for buffered interpolated moves	IACC			•	•			
Set the automatic acceleration setting used for buffered interpolated moves	IACC=[0-1]			•	•			
Read controller ID	ID		•				•	•
Set controller ID	ID=[value]		•					
Read the ignore limit/alarm status	IERR		•	•	•	•	•	•
Set the ignore limit/alarm status	IERR=[1-0]		•	•	•	•	•	•
Conditional loop commands of IF loops in standalone program		IF, ELSEIF, ELSE ENDIF	•	•	•	•	•	•

Set incremental coordinate mode	INC	INC	•	•	•	•	•
Read enable/disable status of positioning complete signal for [axis]	INP[axis]		•				
Set enable/disable status of positioning complete signal for [axis]	INP[axis]=[value]		•				
Read the status of all configurable I/O ports (32-bit)	IO		•				
Set the bit status for all configurable outputs	IO=[value]		•				
Read the status of the configurable I/O at [port]	IO[port]		•				
Set the value of the configurable output bit at [port]	IO[port]=[0-1]		•				
Read the status of all general-purpose I/O at startup (32-bit)	IOBOOT		•				
Set the value of all configurable outputs at startup	IOBOOT=[value]		•				
Read the current configuration of all configurable I/O ports	IOCFG		•				
Set the configuration of all configurable I/O ports	IOCFG=[value]		•				
Read the configurable I/O polarity setting (2-bit logic)	IOP		•				
Set the configurable I/O polarity (2-bit logic)	IOP=[0-1]		•				
Read the device IP address	IP				•		
Set the device IP address	IP=[value]				•		
Jog [axis] in the positive (+) or negative (-) direction	J[axis][+/-]	JOG[axis][+/-]	•	•	•	•	•
Read the allowable speed change for joystick control on [axis]	JDEL[axis]	JOYDEL[axis]	•				
Set the allowable speed change for joystick control on [axis]	JDEL[axis]=[value]	JOYDEL[axis]=[value]	•				
Read joystick control enable configuration	JENA	JOYENA	•				
	JE			•			
	JF					•	•
	JO					•	•
Set joystick control enable configuration setting	JENA=[value]	JOYENA=[value]	•				
	JE=[value]			•			
Read the limit setting for joystick control	JLIM[number]		•				
	JL[number]			•		•	•
Set the limit configuration for joystick control	JLIM[number]=[value]		•				
	JL[number]=[value]			•		•	•
Read the maximum voltage setting for joystick operation for [axis]	JMAX[axis]		•	•			
Set the maximum voltage setting for joystick operation for [axis]	JMAX[axis]=[value]		•	•			
Read the minimum voltage setting for joystick operation for [axis]	JMIN[axis]		•	•			
Set the minimum voltage setting for joystick operation for [axis]	JMIN[axis]=[value]		•	•			
Read the maximum speed setting for joystick operation for [axis]	JSPD[axis]	JOYHS[axis]	•				
Set the maximum speed value for joystick operation for [axis]	JSPD[axis]=[value]	JOYHS[axis]=[value]	•				
Read the dead band tolerance around mid-voltage range for [axis]	JTOL[axis]		•				
Set the dead band tolerance around mid-voltage range for [axis]	JTOL[value]		•				
Read the joystick speed, delta, and tolerance configuration assignments	JV[number]	JOYHS[axis] JOYDEL[axis]		•		•	•
Set the joystick speed, delta, and tolerance configuration assignments	JV[number]=[value]	JOYHS[axis]=[value] JOYDEL[axis]=[value]		•		•	•
Homing using the limit inputs in the [+] or [-]	L[axis][+/-]	LHOME[axis][+/-]				•	•

direction								
Read the global limit correction amount value	LCA						•	•
Set the global limit correction value	LCA=[value]						•	•
Read the limit correction value for [axis]	LCA[axis]						•	•
Set the limit correction value for [axis]	LCA[axis]=[value]						•	•
Read the global low speed parameter	LSPD	LSPD	•				•	•
	LS			•	•			
Set the global low speed parameter	LSPD=[value]	LSPD=[value]	•				•	•
	LS=[value]			•	•			
Read the low speed parameter for [axis]	LSPD[axis]	LSPD[axis]	•				•	•
	LS[axis]			•	•			
Set the low speed parameter for [axis]	LSPD[axis]=[value]	LSPD[axis]=[value]	•				•	•
	LS[axis]=[value]			•	•			
Set the enable/disable for the latch function for [axis]	LT[axis]=[value]		•	•				
Read the previously latched encoder position value for [axis]	LTE[axis]		•					
	LT[axis]E			•	•			
Read the previously latched step position value for [axis]	LTP[axis]		•					
	LT[axis]P			•	•			
Read the current latch status for [axis]	LTS[axis]		•					
	LT[axis]S			•	•			
Read the current position coordinate mode setting	MM		•		•	•	•	•
Read the current manual pulse generator position for [axis]	MP[axis]		•					
Set the manual pulse generator position value for [axis]	MP[axis]=[value]		•					
Read the current divider parameter of the manual pulse generator for [axis]	MPD[axis]	MPGD[axis]	•					
Set the current divider parameter of the manual pulse generator for [axis]	MPD[axis]=[value]	MPDG[axis]=[value]	•					
Read the manual pulse generator enable status for [axis]	MPE[axis]		•					
Enable/disable the manual pulse generator for [axis]	MPE[axis]=[value]		•					
Read the current multiplier parameter of the manual pulse generator for [axis]	MPM[axis]	MPGM[axis]	•					
Set the multiplier parameter value of the manual pulse generator for [axis]	MPM[axis]=[value]	MPGM[axis]=[value]	•					
Read all motor status, buffer move status, and move mode status for [axis]	MST[axis]	MST[axis]	•	•	•	•	•	•
Read the current position value for [axis]	P[axis]	P[axis]	•	•	•	•	•	•
Set the current position value for [axis]	P[axis]=[value]	P[axis]=[value]	•	•	•	•	•	•
Reads encoder values of all axis	PE			•	•			
	EP		•					
Read the polarity parameter for specified [axis]	POL[axis]		•				•	•
	PO[axis]			•	•			
Set the polarity parameter for specified [axis]	POL[axis]=[value]		•				•	•
	PO[axis]=[value]			•	•			
Read the pulse counter for all 4 axes	PP		•	•	•			
Standalone program beginning and ending lines. All program code must be between PRG [0-3] and END		PRG [0-3] END	•	•	•	•	•	•
Read current pulse speed for [axis]	PS[axis]	PSX	•	•	•	•	•	•
Returns current PWM duty cycle in %	PWM[1-2]		•					
Set the current PWM duty cycle in %	PWM[1-2]=[value]		•					
Read the value of a specific [reg] for [axis]	REGRD[axis]Ox[reg]		•					
Write a value to a specific [reg] for [axis]	REGWR[axis]Ox[reg] =[value]		•					

Read the return to zero enable status used during homing	RZ					•	•
Set the return to zero enable status used during homing	RZ=[value]					•	•
Read the specified standalone [line]	SA[line]			•	•	•	•
Set the [value] of the specified standalone [line]	SA[line]=[value]			•	•	•	•
Controls the standalone [prgNo], stop(0), start(1), pause(2), and continue(3)	SACTRL[prgNo]=[0-3]		•				
	SR[prgNo]=[value]	SR[prgNo]=[value]		•	•	•	•
Read the return jump line for standalone program error handling	SAP		•	•	•		
Set the return jump line for standalone program error handling	SAP[0-1]		•	•	•		
Read standalone program operation status	SASTAT[0-3]		•	•	•	•	•
Read S-curve acceleration status for [axis]	SCV[axis]	SCV[axis]	•	•	•	•	•
Set S-curve acceleration for [axis] Trapezoidal acceleration used if S-curve disabled.	SCV[axis]=[0-1]		•				
	SCVX=[0-1]	SCVX=[0-1]		•	•	•	•
Read slow down signal operation status for [axis]	SDC[axis]		•				
Set the configuration for slow down signal operation for [axis] (decel only, or decel and stop)	SDC[axis]=[value]		•				
Read status of slow down operation for axis	SDE[axis]		•				
Enable/disable slow down operation for [axis]	SDE[axis]=[value]		•				
Read enable status for StepNLoop operation for [axis]	SL[axis]	SL[axis]	•	•	•	•	•
Enable/disable StepNLoop operation for [axis]	SL[axis]=[value]	SL[axis]=[value]	•	•	•	•	•
Read maximum number of attempts to complete a StepNLoop operation for [axis]	SLA[axis]		•	•	•	•	•
Set number of attempts allowed to complete a StepNLoop operation for [axis]	SLA[axis]=[value]		•	•	•	•	•
Read error range for StepNLoop operation for [axis]	SLE[axis]		•	•	•	•	•
Set error range for StepNLoop operation	SLE[axis]=[value]		•	•	•	•	•
Read StepNLoop ratio for [axis] (ppr/cpr)	SLR[axis]		•	•	•	•	•
Set StepNLoop ratio for [axis] (ppr/cpr)	SLR[axis]=[value]		•	•	•	•	•
Read StepNLoop status for [axis]	SLS[axis]	SLS[axis]	•	•	•	•	•
Read tolerance setting for StepNLoop operation for [axis]	SLT[axis]		•	•	•	•	•
Set the tolerance value for StepNLoop operation for [axis]	SLT[axis]=[value]		•	•	•	•	•
Read the run on power-up status for [prgNo]	SLOAD[prgNo]		•	•	•	•	•
Set the run on power-up status for [prgNo]	SLOAD[prgNo]=[value]		•	•	•	•	•
Read current program counter for [prgNo]	SPC[prgNo]		•	•	•	•	•
Controls the standalone [prgNo], stop(0), start(1), pause(2), and continue(3)	SR[prgNo]=[value]	SR[prgNo]=[value]		•	•	•	•
	SACTRL[prgNo]=[0-3]		•				
Initiates on-the-fly speed change for [axis], buffer must be disabled	SSPD[axis]=[value]	SSPD[axis]=[value]	•	•	•	•	•
Read the current on-the-fly speed range mode for [axis]	SSPDM[axis]	SSPDM[axis]	•	•	•	•	•
Set the on-the-fly speed range for [axis]	SSPDM[axis]=[value]	SSPDM[axis]=[value]	•	•	•	•	•
Initiates a ramp down to low speed and stop for all axes	STOP	STOP	•	•	•	•	•
Initiates a ramp down to low speed and stop for [axis]	STOP[axis]	STOP[axis]	•	•	•	•	•
Saves parameters to flash memory	STORE		•	•	•	•	•
Standalone program sub routine start and end line		SUB [0-31] ENDSUB	•	•	•	•	•
Read sync condition and source for [axis]	SYNC[axis]	SYNCFG[axis]	•				
	SYN[axis]C			•	•		
Set sync condition and source for [axis]	SYNC[axis]=[value]		•				
	SYN[axis]C=[1-3]			•	•		

Disable sync window and sync output for [axis]	SYNF[axis]	SYNOFF[axis]	•				
	SYN[axis]F			•	•		
Read maximum value of the sync pulse window for [axis]	SYNMAX[axis]		•				
Set maximum value of the sync pulse window for [axis]	SYNMAX[axis]=[value]		•				
Read minimum value of the sync pulse window for [axis]	SYNMIN[axis]		•				
Set minimum value of the sync pulse window for [axis]	SYNMIN[axis]=[value]		•				
Enable synchronization output for [axis]	SYNO[axis]	SYNON[axis]	•				
	SYN[axis]O			•	•		
Read synchronization position value or distance between pulses for continuous sync for [axis]	SYNP[axis]	SYNPOS[axis]	•				
	SYN[axis]P			•	•		
Set synchronization position value or distance between pulses for continuous sync for [axis]	SYNP[axis]=[value]		•				
	SYN[axis]P=[value]			•	•		
Read status of synchronization operation for [axis]	SYNS[axis]	SYNSTAT[axis]	•				
Read sync output pulse width time(ms) for [axis]. Only applicable if sync output configuration is set to 1	SYN[axis]T	SYNTIME[axis]		•	•		
Set sync output pulse width time(ms) for [axis]. Only applicable if sync output configuration is set to 1.	SYN[axis]T=[value]			•	•		
Disable the synchronization window for [axis]	SYNWF[axis]		•				
Enable the synchronization window for [axis]	SYNWO[axis]		•				
Initiates on-the-fly target position change for [axis]	T[axis][value]		•	•	•	•	•
Read the communication time out parameter in ms	TOC	TOC		•	•	•	•
Set the communication time out parameter in ms	TOC=[value]	TOC=[value]		•	•	•	•
Read the timer register value in ms	TR			•	•		
Set the timer register value in ms	TR=[value]			•	•		
Read standalone variable [no]	V[no]	V[no]	•	•	•	•	•
Set standalone variable [no] to [value]	V[no]=[value]	V[no]=[value]	•	•	•	•	•
Read the controller firmware version	VER		•	•	•	•	•
Sets [axis] to wait until the current move is complete before executing the next line of code.		WAIT[axis]	•	•	•	•	•
Conditional statements used in program loops		WHILE ENDWHILE	•	•	•	•	•
Perform individual motion in X and /or Y axis	X[target X] Y[target Y]					•	•
Perform linear interpolation motion with X and Y axes	X[target X]Y[target Y]					•	•
Perform individual motion in X, Y, Z, and/or U axes	X[target X] Y[target Y] Z[target Z] U[target U]	X[target X] Y[target Y] Z[target Z] U[target U]	•	•	•		
Perform linear interpolation motion with X, Y, Z, and/or U axes. Include all axes that will be interpolated. EINT must be enabled (1)	X[targetX]Y[targetY] Z[targetZ]U[targetU]		•				
Perform homing function for [axis] in the positive [+] or negative [-] direction using the Z-index pulse only	Z[axis][+/-]	ZOME[axis][+/-]				•	•
Read number of Z-index pulse to count for [axis]	ZCNT[axis]		•				
Set number of Z-index pulse to count for [axis]	ZCNT[axis]=[value]		•				
Perform homing function for [axis] in the positive [+] or negative [-] direction using the home and Z-index pulse.	ZH[axis][+/-]	ZHOME[axis][+/-]				•	•
Perform the EZ count operation for [axis] in the positive [+] or negative [-] direction	ZMOVE[axis][+/-]		•			•	

; – Comments**; – Comments**

The commands described here are only available when writing a standalone program.

The (;) symbol tells the A-Script compiler to ignore the text following it, or the comment. Comments are brief explanatory notes added to code for the benefit of those reading it.

Standalone ;

Syntax:**Write:**

; [comment]

Where: [comment] is any text

Reply:

There is no reply for this Command

Example:**Write:**

HSPD=20000 ;Set high speed to 20,000 pulses/sec * ; tells
the compiler the text that follows is a comment and to ignore it .

Notes:

- Commenting Guidelines
- The following table provides general good programming guidelines for what types of comments can precede a section of code. These are suggestions; A-Script does not enforce rules for adding comments. Write what works best, both for you and for anyone else who reads your code.

Comment type	Comment description
Purpose	Describes what the code does (not how it does it)
Assumptions	Lists each variable, or other element accessed by the procedure
Effects	Lists each affected variable, control, or other setting, and the effect it has (only if it is not obvious)
Inputs	Specifies the purpose of the argument
Returns	Explains the values returned by the subroutine

See also:[Standalone Programming Commands](#)**Commander Manual**[2.7.1 Program Development](#)**ABORT – Immediate Stop****ABORT – Immediate Stop**

This command is used to stop axis operation immediately. This command is write-only. It is possible to stop each axis separately, or all axes at once.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII/Standalone	ABORT	<i>global abort</i>
	ABORT[axis]	<i>individual axis abort</i>

Syntax:**Write:**

ABORT[axis]

Where: [axis] is set to X, Y, Z, U, or not specified to affect all axis

Reply:

Command is accepted and executed

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

ABORTX	<i>*Stop X axis immediately</i>
ABORT	<i>*Stop all axes immediately</i>

Notes:

- If an interpolation operation is in progress when a STOP[axis] or ABORT[axis] command is entered, all axes involved in the interpolation operation will stop.
- Abort of X, Y, or Z axis disables buffer operation. Use BO command to turn buffer mode back on.
- Abort disables MPG (pulser) operation.
- Abort of the X or Y axis disables Joystick operation, and the Joystick settings are reset.

See also:

[Stop Motion](#)
[ESTOP, STOP](#)

Commander Manual[2.2.3 Stopping](#)**ABS – Absolute Coordinate Mode****ABS – Absolute Coordinate Mode**

Set the position coordinate mode to absolute. Absolute position will be used for motion commands based on a zero-reference location set by a homing routine.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII/Standalone ABS

Syntax:

Write: ABS

Reply:

Command is accepted and executed

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

ABS

** Positioning coordinate mode is set to absolute*

Notes:

None

See also:

[Motion Settings](#), [Homing](#)
[INC](#), [MM](#)

Commander Manual[2.2.4 Positioning operation](#)[2.2.4.1 Absolute or Incremental](#)**ACC – Acceleration Time****ACC – Acceleration Time**

Returns or sets the acceleration time in milliseconds. Acceleration time can be set as a global value, or set for each individual axis.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII/Standalone ACC *global acceleration setting*

ACC[axis] *individual axis acceleration setting*

Syntax:

Read:

ACC[axis]

Where: [axis] is set to X, Y, Z, U, or not specified to read global value

Write:

ACC[axis]=[acc]

Where: [axis] is set to X, Y, Z, U, or not specified to read global value
[acc] is the acceleration time in ms

Reply:

Command is accepted and executed

Read:

Returns the ACC parameter entered (acceleration time in ms).

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read: ACCX ** Returns acceleration time set for the X axis in ms*

Write: ACC=300 ** Sets acceleration time to 300 ms for all axes.*

Notes:

- If [DEC](#) is set to 0, then [ACC](#) setting is used for Acceleration and Deceleration time.
- Acceleration time should be no less than one half of deceleration time.
- Acceleration is time entered in milliseconds. Acceleration rate in pps² (pulses/second²) can be calculated with the following formula:

$$\text{Acceleration rate} = \frac{\text{HSPD} - \text{LSPD}}{\text{ACC}/1000}$$

- The minimum and maximum acceleration values depend on the high-speed and low-speed settings. Refer to [Speed Settings](#) for details.
- Individual acceleration settings will take priority over the global acceleration settings. The global acceleration setting will only be used when one of the following conditions is true:
 1. The corresponding individual acceleration setting is defined as zero
 2. A coordinated motion is being performed

In the example below, the settings will set both the global speed settings and the individual speed setting for the X axis.

HSPD=10000
 HSPDX=2000
 LSPD=300
 ACCX=500
 ACC=300
 DEC=300

Once a move command is issued in this example, the global speed settings for the low speed and deceleration, and the individual X-axis speed settings for the high speed and acceleration, will be used for motion.

See also:

[Motion Settings](#)

[DEC](#), [HSPD](#), [JKA](#), [JKD](#), [LSPD](#), [PS](#), [SCV](#), [SSPDM](#)

Commander Manual

[2.1.4 Motion Profile](#)

[B.1 Acceleration/Deceleration Range](#)

[B.2 Acceleration/Deceleration Range – Positional Move](#)

AI – Analog Input

AI – Analog Input

Obtains the voltage value in mV of an analog input. This is a read-only command.

Applies to: [CMD-4CR](#), [CMD-4EX-SA](#)

ASCII/Standalone

AI[ch]

Syntax:

Read:

AI[ch]

Where: [ch] – specifies the input channel

– *Applies to:* [CMD-4CR](#), [CMD-4EX-SA](#), [PMX-2ED-SA](#), [PMX-2EX](#)

- 2 analog 10-bit inputs [1-2]

- Analog input terminal for X-axis = 1, Y-axis = 2

– *Applies to:* [PMX-4EX-SA](#)

- 8 analog 10-bit inputs [1-8]

- Analog input terminal for X-axis = 1, Y-axis = 2, Z-axis = 3, U-axis = 4

Reply

Command is accepted and executed

Read:

Returns the current voltage (voltage in mV) [range 0- 3300]

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read: AI1 * *returns the value in mV of the voltage on the analog input for the X-axis*
Reply: 2500 * *signifies 2.5V (2500mV) on Analog input 1*

Notes:

None

See also:

[I/O Setup](#),
[JSPD](#), [JDEL](#), [JMAX](#), [JMIN](#), [JTOL](#), [JENA](#), [JLIM](#), [DI](#), [DIP](#), [DO](#), [DOBOOT](#), [DOP](#), [IO](#), [IOBOOT](#), [IOCFG](#), [IOP](#)

Commander Manual

[2.1.3.3.2 Analog Input/output](#)

AO – Analog Output

AO – Analog Output

Sets the voltage value in mV of an analog output. This is a write only command.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII/Standalone AO[ch]=[ao]

Syntax:

Read:

AO[ch]=[ao]

Where: [ch] – specifies the output 2 analog 10-bit outputs [1-2]
 [ao] – specifies the output voltage in mV (0-3,300)

Reply

Command is accepted and executed

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Write: AO2=500 * *sets the analog output value to 500 mV for analog output 2*

Notes:

- This command is not fully working yet Leave as "OUT of DATE" status

See also:

[AI](#)

Commander Manual

[2.1.3.3.2 Analog Input/output](#)

ARC – Circular Interpolation (Arc)

ARC – Circular Interpolation (Arc)

The ARC command is a write-only command that starts a 2-axes arc interpolated motion. Arcs are similar to a 2-axis circular interpolation operation. The difference between them is that the endpoint position of the arc interpolation is arbitrarily set on the trajectory of the circular interpolation, thus creating an arc. An arc can be either the CW or CCW direction. All ARC moves are interpreted as absolute moves.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII/Standalone ARC

The Commander series allows arcs to be drawn using any 2 axes.

Syntax:

Applies to: CMD-4CR, CMD-4EX-SA

Write:

ARC[A1][A2]P[C1]:[C2]:[Θ] **for CW motion*
ARC[A1][A2]N[C1]:[C2]:[Θ] **for CCW motion*

Where: [A1][A2] specifies axis 1 and axis 2 (X, Y, Z, U),
 N or P specifies direction of motion, with N = CCW, and
 P = CW,
 [C1]:[C2] specifies the arc coordinate center of [A1],[A2] (32-bit
 value),
 [Θ] specifies the end of position of the arc (unit =
 0.001 degrees, 0-360,000)

Reply

Command is accepted and executed

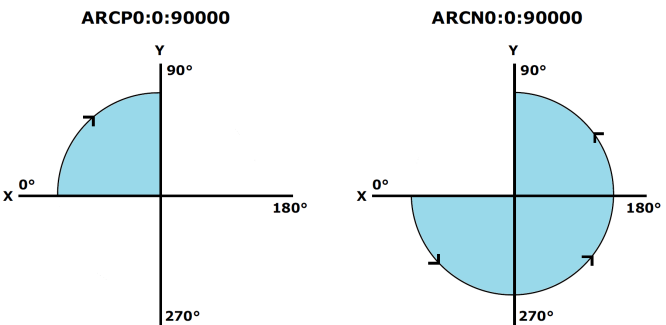
Write:

OK

Command cannot be processed

An error message is returned. See

[Error Codes](#)



Example:

Applies to: CMD-4CR, CMD-4EX-SA

Write: ARCXZP0:100:90000 **Arc interpolation of X and Z axis, center X = 0, Y = 100, 90-degree arc in the CW direction*

Notes:

- Other conditions:
 - Joystick operation is disabled
 - Manual pulse generator (MPG) operation is disabled
 - Current operations are stopped
 - Error status is off

See also:

[Coordinated Motion](#)
[ARCT](#), [CIR](#), [CIRT](#)

Commander Manual

[2.3 Interpolation Operation](#)
[2.3.2.1 Arc interpolation](#)
[2.3.4 Buffered interpolation operation](#)

ARCT – Circular Interpolation (Helix Arc)**ARCT – Circular Interpolation (Helix Arc)**

The ARCT command is similar to the ARC command, with the addition of interpolation with a third axis to perform a helical or tangential type movement. ARCT is a write-only command and can only be used for an arc interpolation between the X and Y axes, with the Z axis used for linear coordination. Arcs can be drawn in either the CW or CCW direction. All ARCT moves are interpreted as absolute moves.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII/Standalone ARCT *global high speed setting*

Syntax:**Write:**

ARCTN[C1]:[C2]:[Θ]:[Z target] ** for CCW motion*

ARCTP[C1]:[C2]:[Θ]:[Z target] ** for CW motion*

Where: N is CCW motion,
 P is CW motion
 [C1]:[C2] specifies the arc coordinate center of X, Y axes (32-bit value),
 [Θ] specifies the end of position of the arc (unit = 0.001 degrees, 0-360,000),
 [Z target] specifies the target position for the Z-axis (32-bit value)

Reply

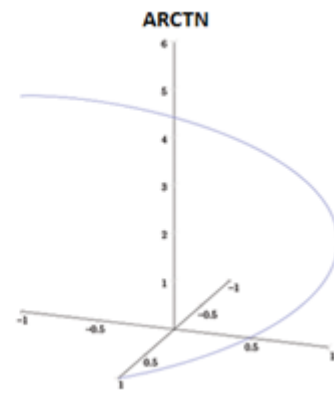
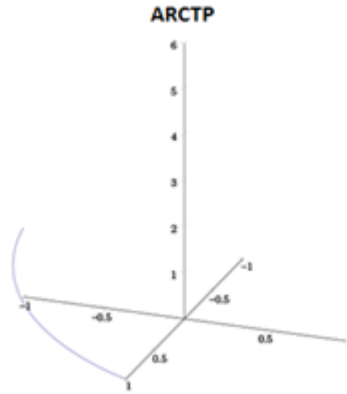
Command is accepted and executed

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)



Example:

Write:

ARCTP1000:0:180000:1000 * Arc interpolation of X and Y axis, center X = 1000, Y = 0, 180-degree arc with movement along the Z-axis to absolute position of 1000

Notes:

- Other conditions:
 - Joystick operation is disabled
 - Manual pulse generator (MPG) operation is disabled
 - Current operations are stopped
 - Error status is off

See also:

[Coordinated Motion](#)
[ARC](#), [CIR](#), [CIRT](#)

Commander Manual

[2.3 Interpolation Operation](#)
[2.3.3 Helical or Tangential interpolation](#)
[2.3.4 Buffered interpolation operation](#)

BF – Buffer Disable

BF – Buffer Disable

Write-only command that disables the buffer operation.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII	BF
Standalone	BUFOFF

Syntax:

Write:

BF

Reply:

Command is accepted and executed

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Write: BF ** Enables the buffer operation*

Notes:

None

See also:

[Buffer Operation](#)

[BO](#), [I](#), [BSTAT](#), [BSTART](#), [IACC](#)

Commander Manual

[2.3.4 Buffered interpolation operation](#)

BO – Buffer Enable

BO – Buffer Enable

Write-only command that enables the buffer operation.

- 3-axis buffer mode is available in all 4axis and 8 axis versions of the Commander.
- 4-axis vector (linear) buffer mode is available in all 4axis and 8 axis versions of the commander with V202 firmware or newer.
- 2-axis vector (linear) buffer mode is available in all Gen 5 versions of the commander with V301 firmware or newer.
- 6-axis and 8-axis vector

Applies to: CMD-4CR, CMD-4EX-SA

ASCII BO

Standalone BUFON

Syntax:

Applies to: CMD-4CR, CMD-4EX-SA

Write:

BO

Applies to: CMD-4CR, CMD-4EX-SA with V202 or newer firmware

Write:

BO=[buffer mode]

Where: [buffer mode] specifies the buffer mode 1 = 3 axis buffer mode, and 2 = 4 axis in vector (linear) buffer mode, for backwards compatibility, when only BO is sent, 3 axis buffer

mode is enabled.

Reply:

Command is accepted and executed

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Write: BO * Enables the 3 axis buffer operation

| Applies to: CMD-4CR, CMD-4EX-SA with V202 or newer firmware

Write:

BO=2 * Enables the 4 axis buffer operation

Notes:

- In 3-axis buffer mode - axis X,Y, and Z are used and supports, [ARC](#), [ARCT](#), [CIR](#), [CIRT](#), and the 3 axis version of the [I](#) command. In this mode axis U is not available for any motion.
- In 4 axis vector (linear) buffer mode only the 4 axis version of the [I](#) command can be buffered.
- With V202 firmware for the CMD-4CR and CMD-4EX-SA a 4 axis in vector (linear) buffer mode, was added. To access the new functionality add BO=2 to enable 4 axis in vector (linear) buffer mode.
- BO still works as originally designed as well as BO=1 for 3 axis buffer mode.
- BF must be called between moving from 3 to 4 axis or vice versa.
- In 4 axis vector (linear) buffer mode only the 4 axis version of the [I](#) command can be buffered.
-
- BF, BSTAT, and BSTART all operate the same for the 3 axis and 4 axis version of the buffer command.

See also:

[Buffer Operation](#)

[BF](#), [I](#), [BSTAT](#), [BSTART](#), [IACC](#)

Commander Manual

[2.3.4 Buffered interpolation operation](#)

BSTART, ISTART – Buffer Start

BSTART, ISTART – Buffer Start

Write-only command that initiates the buffer operations, starting any commands that are loaded in the buffer locations.

| Applies to: CMD-4CR, CMD-4EX-SA

ASCII

BSTART

Standalone ISTART

Syntax:

Write:

BSTART

Reply:

Command is accepted and executed

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Write: BSTART

** Execution of the buffered commands begins*

Notes:

None

See also:

[Buffer Operation](#)

[BO](#), [BF](#), [I](#), [BSTAT](#), [IACC](#)

Commander Manual

[2.3.4 Buffered interpolation operation](#)

BSTAT – Buffer Status

BSTAT – Buffer Status

A read-only command that returns the current status of the buffer operation.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII/Standalone

BSTAT

Syntax:

Read:

BSTAT

Reply

Command is accepted and executed

Read:

Four integers in the format "%d:%d:%d:%d" [buffer enabled]:[buffer start]:
[buffer end]:[available buffer]

Where: [buffer enabled] – indicates the enabled or disabled state (0 = disabled, 1 = enabled),
[buffer start] – indicates the starting step number of the buffer

(0-99)
[buffer end] – indicates the ending step number of the buffer

(0-99)
[available buffer] – indicates the number of available buffer positions (1-100)

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read: BSTAT ** returns the buffer status*

Reply: 1:0:34:66 ** signifies buffer is enabled, starting at position 0, ending at position 34, with 66 positions available*

Notes:

None

See also:

[Buffer Operation](#)

[BO](#), [BF](#), [I](#), [BSTART](#), [IACC](#)

Commander Manual

[2.3.4 Buffered interpolation operation](#)

BTUNE – Tuning for Motion Buffer

BTUNE – Tuning for Motion Buffer

The BTUNE command is a motion buffer command that starts a 2-axis circular interpolation motion. Circles can be drawn using 2 axes in either the CW or CCW direction. All circular moves are interpreted as absolute moves.

Applies to: V300+ CMD-4CR, CMD-4EX-SA

ASCII

BTUNE

Syntax:

Applies to: CMD-4CR, CMD-4EX-SA

Write:

CIR[A1][A2]P[C1]:[C2] *for CW motion*

CIR[A1][A2]N[C1]:[C2] *for CCW motion*

Where: [A1][A2] specifies axis 1 and axis 2 (X, Y, Z, U),
N is CCW motion, P is CW motion,
[C1]:[C2] specifies the circle coordinate center of [A1],[A2] (32-bit value)

Reply

Command is accepted and executed

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Applies to: CMD-4CR, CMD-4EX-SA

Write:

CIRXYP1000:0 * *Circular interpolation of X and Y axis, center X = 1000, Y = 0
move in CW direction*

Applies to: PMX-4EX-SA, PMX-4ET-SA

Write:

CIRN1000:0 * *Circular interpolation of X and Y axis, center X = 1000, Y
= 0 move in CCW direction*

Notes:

- Any axis can be used for circular interpolation operations.
- Other conditions:
 - Joystick operation is disabled
 - Manual pulse generator (MPG) operation is disabled
 - Current operations are stopped
 - Error status is off

See also:

[Coordinated Motion](#)

[ARC](#), [ARCT](#), [CIRT](#)

Commander Manual

[2.3 Interpolation Operation](#)

[2.3.2 Circular interpolation](#)

[2.3.4 Buffered interpolation operation](#)

CIR – Circular Interpolation

CIR – Circular Interpolation

The CIR command is a write-only command that starts a 2-axis circular interpolation motion. Circles can be drawn using 2 axes in either the CW or CCW direction. All circular moves are interpreted as absolute moves.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII/Standalone

CIR

Syntax:

Applies to: CMD-4CR, CMD-4EX-SA

Write:

CIR[A1][A2]P[C1]:[C2] *for CW motion*
 CIR[A1][A2]N[C1]:[C2] *for CCW motion*

Where: [A1][A2] specifies axis 1 and axis 2 (X, Y, Z, U),
 N is CCW motion, P is CW motion,
 [C1]:[C2] specifies the circle coordinate center of [A1],[A2] (32-bit value)

Reply

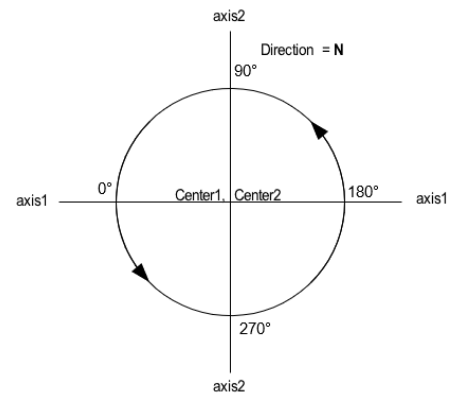
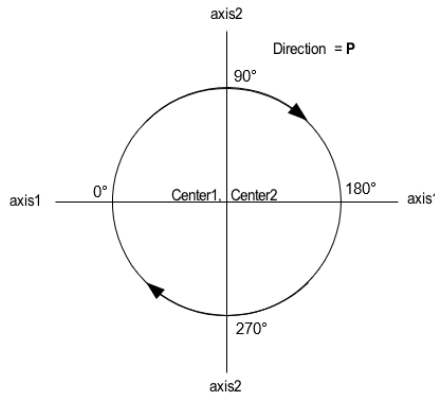
Command is
 accepted and executed

Write:

OK

Command
 cannot be processed

An error
 message is returned.
 See [Error Codes](#)



Example:

Applies to: CMD-4CR, CMD-4EX-SA

Write:

CIRXYP1000:0 * *Circular interpolation of X and Y axis, center X = 1000, Y = 0
 move in CW direction*

Notes:

- Any axis can be used for circular interpolation operations.
- Other conditions:
 - Joystick operation is disabled
 - Manual pulse generator (MPG) operation is disabled
 - Current operations are stopped
 - Error status is off

See also:

[Coordinated Motion](#)
[ARC](#), [ARCT](#), [CIRT](#)

Commander Manual

[2.3 Interpolation Operation](#)
[2.3.2 Circular interpolation](#)
[2.3.4 Buffered interpolation operation](#)

CIRT – Circular Interpolation (Helix)

CIRT – Circular Interpolation (Helix)

The CIRT command is similar to the CIR command, with the addition of interpolation with a

third axis to perform a helical or tangential type movement. CIRT is a write-only command and can only be used for a circular interpolation between the X and Y axes, with the Z axis used for linear coordination. Circles can be drawn in either the CW or CCW direction. All CIRT moves are interpreted as absolute moves.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII/Standalone

CIRT

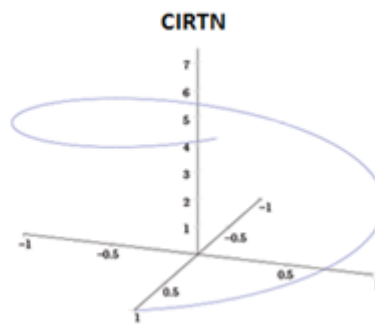
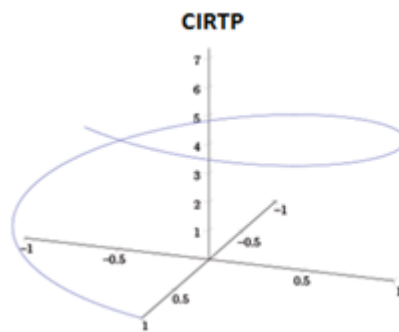
Syntax:

Write:

CIRTN[C1]:[C2]:[Z target] ** for CCW motion*

CIRTP[C1]:[C2]:[Z target] ** for CW motion*

Where: N is CCW motion, P is CW motion,
 [C1]:[C2] specifies the circle coordinate center of X, Y axes (32-bit value),
 [Z target] specifies the target position for the Z-axis (32-bit value)



Reply

Command is accepted and executed

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Write: CIRTP1000:0:1000 ** Circular interpolation of X and Y axis, center X = 1000, Y = 0, with movement along the Z axis to absolute position of 1000*

Notes:

- Other conditions:
 - Joystick operation is disabled
 - Manual pulse generator (MPG) operation is disabled
 - Current operations are stopped
 - Error status is off

See also:

[Coordinated Motion](#), [Circular interpolation](#)
[ARC](#), [ARCT](#), [CIR](#)

Commander Manual

[2.3 Interpolation Operation](#)
[2.3.3 Helical or Tangential interpolation](#)
[2.3.4 Buffered interpolation operation](#)

CLR – Clear Axis Error**CLR – Clear Axis Error**

Write-only command to clear the axis error status. Error statuses can be cleared per axis, or clear all axis errors at once*. The command only clears the error status and does not correct the cause of the error.

*For Commander series firmware V201+ there is an ability to clear all axis errors in a single command.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII	CLR[axis]
Standalone	ECLEAR[axis]

Syntax:**Write:**

CLR[axis]
 ECLEAR[axis]

Where: [axis] specifies the axis (X,Y,Z,U) to clear error status. If no axis is specified errors from all axis are cleared.

Reply

Command is accepted and executed

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Write: CLRX * *clears error status in the X axis*
 CLR * *clears error status for all axes (CMD V201+ only)*

Notes:

- Commander firmware V201+ has the ability to clear errors from all axes in a single command.
- If an error flag accrues while IERR is enabled (IERR=1), and IERR is changed to off (IERR=0).

The error flags will activate, and a CLR must be sent to clear them.

See also:

[Limits/Safety Setup](#),
[IERR](#), [MST](#), [PS](#)

Commander Manual

[2.1.3.1.1.1 Limit Switches](#)
[2.1.3.1.1.2 Slowdown switches](#)
[2.1.3.1.2 Emergency stop](#)

D – Deviation Counter

D – Deviation Counter

Read-only command that returns the current value (CMD series 32-bit value, PMX series 28-bit value) of the deviation counter for the specified axis. This command is related to the StepNLoop function.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII D[axis] *StepNLoop delta value of axis*

Applies to: CMD-2ED-SA, CMD-2EX-SA

ASCII DX[axis] *StepNLoop delta value of axis*

Syntax:

Read:

D[axis] *CMD series*

DX[axis] *PMX series*

Where: [axis] is set to X, Y, Z, U

Reply:

Command is accepted and executed

Read:

Returns the value of the [axis] deviation counter (CMD series 32-bit value, PMX series 28-bit value)

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:

DX * Returns the deviation counter value of X-axis

Notes:

- None

See also:

[StepNLoop](#)
[EP, E, PP, P](#)

Commander Manual

[Identification Number](#)

DB – Baud Rate Setting

DB – Baud Rate Setting

Sets the desired baud rate of the serial communication port in bps.

Applies to: CMD-4CR, CMD-4EX-SA, CMD-2ED-SA, CMD-2EX

ASCII DB

Syntax:

Read:

DB

Write:

DB=[rate]

Where: [rate] Value from 1 – 5 based on: 1 = 9600 (default setting) 2 = 19200 3 = 38400 4 = 57600 5 = 115200

Reply:

Command is accepted and executed

Read:

Returns the value set (1-5)

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:

DB ** Returns value for baud rate*

Write:DB=3 * *Sets baud rate to 38400 bps***Notes:**

- To write the value to the device's flash memory, use the STORE command. After a complete power cycle, the new baud rate will be written to memory.
- Note that until a power cycle is completed, the settings will not take effect.
- The default RS-485 communication port settings are:

Parameter	Setting
Baud Rate	9600 bps
Byte Size	8 bits
Parity	None
Flow Control	None
Stop Bit	1

See also:[System Setup](#)[VER, ID, IP](#)**Commander Manual**[Identification Number](#)**DEC – Deceleration Time****DEC – Deceleration Time**

Sets the deceleration time or acquires the current set value. Deceleration time is available as a global value for all axes and a value that can be set for each axis

Applies to: CMD-4CR, CMD-4EX-SA

ASCII/Standalone

DEC

global acceleration setting

DEC[axis]

*individual axis acceleration setting***Syntax:****Read:**

DEC[axis]

Where: [axis] is set to X, Y, Z, U, or not specified to read global value

Write:

DEC[axis]=[dec]

Where: [axis] is set to X, Y, Z, U, or not specified to read global value
[dec] is the deceleration time in ms (0 – 65,535)

Reply:

Command is accepted and executed

Read:

Returns the DEC parameter entered (deceleration time in ms).

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:

DECY * Returns deceleration time set for the Y axis in ms

Write:

DEC=300 * Sets deceleration time to 300 ms for all axes.

Notes:

- If [DEC](#) is set to 0, then [ACC](#) setting is used for Acceleration and Deceleration time.
- Deceleration time should be equal to or less than 2 times acceleration time.
- Deceleration is time entered in milliseconds. Deceleration rate in pps² (pulses/second²) can be calculated with the following formula.

$$\text{Deceleration rate} = \frac{\text{HSPD} - \text{LSPD}}{\text{ACC}/1000}$$

- The minimum and maximum deceleration values depend on the high-speed and low-speed settings. Refer to [Speed Settings](#) for details.
- Individual deceleration settings will take priority over the global deceleration settings. A global deceleration setting will only be used when, either one of the following conditions is true:
 1. The corresponding individual deceleration setting is defined as zero.
 2. A coordinated motion is being performed.

In the example below, the settings will set both the global speed settings as well as the individual speed setting for the X axis.

HSPD=10000
 HSPDX=2000
 LSPD=300
 ACCX=500
 ACC=300
 DEC=300

Once a move command is issued in this example, the global speed settings for the low speed and deceleration, and the individual X-axis speed settings for the high speed and acceleration will be used for motion.

See also:

[Motion Settings](#)

[ACC](#), [HSPD](#), [JKA](#), [JKD](#), [LSPD](#), [PS](#), [SCV](#), [SSPDM](#)

Commander Manual

[2.1.4 Motion Profile](#)

[B.1 Acceleration/Deceleration Range](#)

[B.2 Acceleration/Deceleration Range – Positional Move](#)

[2.1.3.1.1.2 Slowdown switches](#)

DELAY – Standalone DELAY Command

DELAY – Standalone DELAY Command

The commands described here are only available when writing a standalone program.

In standalone programs, this command is used to set a delay in milliseconds. Assigned value is a 32-bit unsigned integer or a variable.

Applies to: CMD-4CR, CMD-4EX-SA, CMD-2ED-SA, CMD-2EX

Standalone DELAY

Syntax:

Write:

DELAY=[timems]

Where: [timems] is the delay time in ms (0 – 65,535) Can be an unsigned integer or a variable.

Reply:

Command is accepted and executed

No Reply

Example:

Write:

DELAY=300 ** Sets delay time to 300 ms.*

DELAY=V1 ** Sets delay time to value assigned to variable V1.*

Notes:

- None

See also:

[Standalone Programming Commands](#)

Commander Manual

[2.1.1.1 Identification Number](#)

DI – Digital Input Status**DI – Digital Input Status**

Read-only command that returns the input status of all digital inputs or return status individually.

Applies to: CMD-4CR, CMD-4EX-SA, CMD-2ED-SA, CMD-2EX

ASCII/Standalone DI *all digital input status*
 DI[input] *individual digital input status*

Syntax:**Read:**

DI[input]

Where: [input] is number (CMD series 1-4, PMX series 1-8) for input, or not specified to read all input values.

Reply:

Command is accepted and executed

Read:

Returns status (CMD series 0-15, PMX series 0-255) of all inputs based on each bit designated to an input, and the value of that bit is 1 if on, or 0 if off.

Bit	Description	Bit-Wise Command
0	Digital Input 1	DI1
1	Digital Input 2	DI2
2	Digital Input 3	DI3
3	Digital Input 4	DI4
4	Digital Input 5	DI5
5	Digital Input 6	DI6
6	Digital Input 7	DI7
7	Digital Input 8	DI8

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:**Read:**

DI * Returns value for all digital inputs
 Ex. Return a value of 12, or binary 1100, indicates inputs 3 and 4
 are on and all other inputs are off.

DI3 * Returns the status of digital input 3

Notes:

- None

See also:

[I/O Setup](#),
[AI](#), [DIP](#), [DO](#), [DOBOOT](#), [DOP](#), [IO](#), [IOBOOT](#), [IOCFG](#), [IOP](#)

Commander Manual

[2.1.3.2.1 High-Speed Digital Inputs](#)

DIP – Digital Input Polarity**DIP – Digital Input Polarity**

Read/write command that toggles the polarity of inputs.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII DIP=[pol]

Syntax:**Read:**

DIP

Write:

DIP=[pol]

Where: [pol] is set to 0 for negative logic (active low), 1 for positive logic (active high).

Reply:

Command is accepted and executed

Read:

Returns the value set (0,1).

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:**Read:**

DIP ** Returns value for input polarity setting*

Write:

DIP=1 ** Set input polarity to 1.*

Notes:

- To write the value to the device's flash memory, use the STORE command.

See also:

[I/O Setup](#),
[AI](#), [DI](#), [DO](#), [DOBOOT](#), [DOP](#), [IO](#), [IOBOOT](#), [IOCFG](#), [IOP](#)

Commander Manual

[2.1.3.2.1 High-Speed Digital Inputs](#)

DO – Digital Output Status

DO – Digital Output Status

Read/write command that returns/sets the output status of all digital outputs or return/sets statuses individually.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII/Standalone	DO	<i>all digital output status</i>
	DO[output]	<i>individual digital output status</i>

Syntax:

Read:

DO[output]

Where: [output] is number (CMD series 1-4, PMX series 1-8) for output, or not specified to read all output values.

Write:

DO[output]=[value]

Where: [output] is number (CMD series 1-4, PMX series 1-8) for output, or not specified to read all output values.
 [value] for individual output is 1 if on, or 0 if off.
 for all outputs is number (CMD series 0-15, PMX series 0-255) for setting output based on each bit designated to an output, and the value of that bit is 1 if on, or 0 if off. (see chart below)

Reply:

Command is accepted and executed

Read:

Returns status (CMD series 0-15, PMX series 0-255) of all outputs based on each bit designated to an output, and the value of that bit is 1 if on, or 0 if

off.

Bit	Description	Bit-Wise Command
0	Digital Output 1	DO1
1	Digital Output 2	DO2
2	Digital Output 3	DO3
3	Digital Output 4	DO4
4	Digital Output 5	DO5
5	Digital Output 6	DO6
6	Digital Output 7	DO7
7	Digital Output 8	DO8

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:

D0 ** Returns value for all Digital Outputs
Ex. Return a value of 12, or binary 1100, indicates outputs 3 and 4 are on and all other outputs are off.*

D03 ** Returns the status of Digital Output 3*

Write:

D0=11 ** Since 11 is binary 1011, digital outputs 1, 2, and 4 are turned on, all other outputs are turned off.*

D04=1 ** Sets the status of Digital Output 4 to on*

Notes:

- None

See also:

[I/O Setup](#),
[AI](#), [DI](#), [DIP](#), [DOBOOT](#), [DOP](#), [IO](#), [IOBOOT](#), [IOCFG](#), [IOP](#)

Commander Manual

[2.1.3.2.2 High-Speed Digital Output](#)

DOBOOT – Digital output boot-up state

DOBOOT – Digital output boot-up state

Read/write command that returns/sets the boot-up output status of all digital outputs.

Applies to: CMD-4CR, CMD-4EX-SA, CMD-2ED-SA, CMD-2EX

ASCII

DOBOOT

Syntax:**Read:**

DOBOOT

Write:

DOBOOT=[value]

Where: [value] is number (CMD series 0-15, PMX series 0-255) for setting output, based on each bit designated to an output, and the value of that bit is 1 if on, or 0 if off. (see chart below)

Reply:*Command is accepted and executed***Read:**

Returns status (CMD series 0-15) of all outputs based on each bit designated to an output, and the value of that bit is 1 if on, or 0 if off.

Bit	Description
0	Digital Output 1
1	Digital Output 2
2	Digital Output 3
3	Digital Output 4
4	Digital Output 5
5	Digital Output 6
6	Digital Output 7
7	Digital Output 8

Write:

OK

*Command cannot be processed*An error message is returned. See [Error Codes](#)**Example:****Read:**

DOBOOT

** Returns value for all outputs*

Ex. Return a value of 9, or binary 1001, indicates that digital output 1 and 4 will be set on and all other outputs set to off when the controller boots-up.

Write:

DOBOOT=14

** Since 14 is binary 1110, digital output 2,3 and 4 will be set on and all other outputs set to off when the controller boots up.*

Notes:

- None

See also:[I/O Setup](#)[AI](#), [DI](#), [DIP](#), [DO](#), [DOP](#), [IO](#), [IOBOOT](#), [IOCFG](#), [IOP](#)**Commander Manual**[2.1.3.2.2 High-Speed Digital Output](#)**DOP – Digital Output Polarity****DOP – Digital Output Polarity**

Read/write command that returns or sets the polarity of outputs.

Applies to: [CMD-4CR](#), [CMD-4EX-SA](#)

ASCII DOP

Syntax:**Read:**

DOP

Write:

DOP=[pol]

Where: [pol] is set to 0 for negative logic(active low), 1 for positive logic (active high).

Reply:*Command is accepted and executed***Read:**

Returns the value set (0, 1).

Write:

OK

*Command cannot be processed*An error message is returned. See [Error Codes](#)**Example:****Read:**DOP ** Returns value for output polarity setting***Write:**DOP=1 ** Set output polarity to 1 (positive logic [active high]).***Notes:**

- Output polarity can not be changed while any of the High-speed digital outputs are sent to synchronization mode on. (SYNO[axis])
- To write the value to the device's flash memory, use the STORE command.

See also:

[I/O Setup](#),
[AI](#), [DI](#), [DIP](#), [DO](#), [DOBOOT](#), [IO](#), [IOBOOT](#), [IOCFG](#), [IOP](#)

Commander Manual[2.1.3.2.2 High-Speed Digital Output](#)**E – Encoder value (axis)****E – Encoder value (axis)**

Read/write command that returns or sets the feedback counter value (28-bit value)

Applies to: [CMD-4CR](#), [CMD-4EX-SA](#), [CMD-2ED-SA](#), [CMD-2EX](#)

ASCII/Standalone

E[axis]

Syntax:**Read:**

E[axis]

Where: [axis] is set to X, Y, Z, U

Write:

E[axis]=[value]

Where: [axis] is set to X, Y, Z, U,
 [value] specifies the encoder counter value (-134,217,728 to
 +134,217,727)

Reply:

Command is accepted and executed

Read:

Returns the value of the [axis] feedback counter.

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:**Read:**

EX ** Returns feedback counter value of X axis*

Write:

EY=0 ** Sets the Y-axis encoder counter to 0*

Notes:

- Valid encoder counter values are 28-bit numbers between -134,217,728 to +134,217,727

See also:

[Axis motion](#),
[D](#), [EP](#), [PP](#), [P](#)

Commander Manual

[2.6.1.2 Feedback Pulse Counter](#)

EINT – Interpolation Enable**EINT – Interpolation Enable**

Read/write command that returns or enables/disables interpolation operation of all axes.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII/Standalone

EINT

Syntax:**Read:**

EINT

Write:

EINT=[value]

Where: [value] specifies if interpolation operation is enabled/disabled (0 – 1)
0: Interpolation operation for XYZU axes is disabled
1: Interpolation operation for XYZU axes is enabled

Reply

Command is accepted and executed

Read:

Returns status of interpolation operation – enabled (1), or disabled (0)

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read: EINT ** Returns enable/disabled status of interpolation operation*

Write: EINT=1 ** Enables interpolation operation*

Notes:

None

See also:[Linear Interpolation](#)[X, Y, Z, U, 2-axis, 3-axis, 4-axis](#)**Commander Manual**[2.3.2 Circular interpolation](#)**EO – Axis Enable Status****EO – Axis Enable Status**

Read/Write command that returns/sets the enable status of all axes (1-4) at once or return/sets status individually.

Applies to: CMD-4CR, CMD-4EX-SA, CMD-2ED-SA, CMD-2EX

ASCII/Standalone

EO

Return/sets status for all axes

EO[1-4]

*Return/sets status for an axis individually***Syntax:***All axis at once***Read:** EO**Write:** EO=[value]

Where: [value] is an integer [(0-3) for 2-axis, (0-15) for 4-axis] that represents the bit configuration of the enable outputs.

See table in the Notes section for more details.

Each axis individually:

Applies to: CMD-4CR, CMD-4EX-SA

Read: EO[1-4]

Where: [1-4] specifies the axis (1 = X, 2 = Y, 3 = Z, 4 = U)

Write: EO[1-4]=[value]

EO=[value]

Where: [1-4] specifies the axis (1 = X, 2 = Y, 3 = Z, 4 = U)
[value] indicates/sets the enable status (0 = disabled, 1 = enabled)

Applies to: CMD-2ED-SA, CMD-2EX

Read: EO[1-2]

Where: [1-2] specifies the axis (1 = X, 2 = Y)

Write: EO[1-2]=[value]

Where: [1-2] specifies the axis (1 = X, 2 = Y)
 [value] indicates/sets the enable status (0 = disabled, 1 = enabled)

Reply:

Command is accepted and executed

Read:

EO returns an integer [(0-3) for 2-axis, (0-15) for 4-axis] that represents the bit configuration of the enable outputs.

EO[1-4] returns the status of a single enable, With a value of 1 indicating on, and 0 off.

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:

EO * Returns enable status for all axes

Ex. Return a value of 9, or binary 1001, indicates axes X and U are enabled, 2 and 3 are disabled.

EO3 * Returns the status of Z-axis enable

Write:

EO4=1 * Enables U axis

EO=11 * Since 11 is binary 1011, axes X, Y, and U are enabled, Z-axis is disabled.

Notes:

- It is strongly recommended not to disable any axis while moving.
- [Table for CMD-2ED-SA, CMD-2EX](#)

Bit	Axis Designation	Disabled, Enabled		Bit-Wise Command
0	X-axis	0	1	EO1
1	Y-axis	0	1	EO1

- [Table for CMD-4CR, CMD-4EX-SA](#)

Bit	Axis Designation	Disabled, Enabled		Bit-Wise Command
0	X-axis	0	1	EO1
1	Y-axis	0	1	EO2
2	Z-axis	0	1	EO3
3	U-axis	0	1	EO4

- While the EO command can be used to enable or disable the current to the motor, the effect of the enable output signals will depend on the characteristics of the motor drive.

See also:[Motor Setup](#)[J, X, Y, Z, U, EOBOOT](#)**Commander Manual**[2.1.2.1 Stepper](#)[2.1.2.2 Digital Servo](#)[2.2.1 Motor Power](#)**EOBOOT – Axis Enable Status on Boot-up****EOBOOT – Axis Enable Status on Boot-up**

Read/write command that returns/sets the enable status of each axes.

[EOBOOT – Axis Enable Status on Boot-up](#)[Applies to: CMD-2CR, CMD-4CR, CMD-4EX-SA, CMD-8CR](#)

ASCII

EOBOOT

Syntax:**Read:** EOBOOT**Write:** EOBOOT=[value]

Where: [value] is an integer [(0-3) for 2-axis, (0-15) for 4-axis, (0-255) for 8-axis] that represents the bit configuration of the boot enable output status.

See table in the Notes section for more details.

Reply:*Command is accepted and executed***Read:**

EOBOOT returns an integer [(0-3) for 2-axis, (0-15) for 4-axis] that represents the bit configuration of the enable outputs.

Write:

OK

*Command cannot be processed*An error message is returned. See [Error Codes](#)**Example:****Read:** EOBOOT** Returns boot enable output status for all axes*** Ex. Return a value of 9, or binary 1001, indicates axes X and U are enabled, 2 and 3 are disabled on boot up.*

Write: EOBOOT=11 * *Since 11 is binary 1011, sets axes X, Y, and U as enabled, Z-axis as disabled on boot up.*

Notes:

- [Table for CMD-2CR](#)

Bit	Axis Designation	Disabled, Enabled	
0	X-axis	0	1
1	Y-axis	0	1

- [Table for CMD-4CR, CMD-4EX-SA](#)

Bit	Axis Designation	Disabled, Enabled	
0	X-axis	0	1
1	Y-axis	0	1
2	Z-axis	0	1
3	U-axis	0	1

- [Table for CMD-8CR](#)

Bit	Axis Designation	Disabled, Enabled	
0	X-axis	0	1
1	Y-axis	0	1
2	Z-axis	0	1
3	U-axis	0	1
4	A-axis	0	1
5	B-axis	0	1
6	C-axis	0	1
7	D-axis	0	1

See also:

[Motor Setup](#)
[J, X, Y, Z, U, EO](#)

Commander Manual

[2.2.1 Motor Power](#)

EP – Encoder value (all)

EP – Encoder value (all)

Read-only command that returns the current encoder counter values for all axes at once.

[Applies to: CMD-4CR, CMD-4EX-SA](#)

ASCII EP

Syntax:

Read:

EP

Reply:

Command is accepted and executed

Read:

Four integers in the format "%d:%d:%d:%d" [axisx]:[axisy]:[axisz]:[axisu]

Where: [axisx] – indicates the encoder position for axis X
 [axisy] – indicates the encoder position for axis Y
 [axisz] – indicates the encoder position for axis Z
 [axisu] – indicates the encoder position for axis U

Example:

Read:

EP ** Returns the current encoder counter values of all axes*
Ex. Return value of 1000:0:542:-2000, indicates
Axis X encoder = 1000
Axis Y encoder = 0
Axis Z encoder = 542
Axis U encoder = -2000

Notes:

- Valid encoder counter values are 28-bit numbers between -134,217,728 and +134,217,727

See also:

[System Status and Settings](#)

[D](#), [E](#), [PP](#), [P](#)

Commander Manual

[2.6.1.2 Feedback Pulse Counter](#)

ERC – Deviation Counter Clear Conditions

Write sets the conditions to output a deviation counter clear signal. Read returns the current setting.

ERC – Deviation Counter Clear Conditions

Applies to: CMD-4CR, CMD-4EX-SA

ASCII

ERC

Syntax:

Read:

ERC[axis]

Where: [axis] is set to X, Y, Z, U, or not specified to read global value

Write:

ERC[axis]=[erc]

Where: [axis] is set to X, Y, Z, U

[erc]	sets the use/do not use condition (0-3)
	0: Do not use
	1: Use – Output the signal in case of an error stop
	2: Use – Output the signal in the case of a homing operation
	3: Use – Output the signal for both an error stop and a homing operation

Reply:

Command is accepted and executed

Read:

Returns the value set for use/do not use condition (0-3).

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:

ERCY ** Returns the setting for use/do not use condition of the Y axis*

Write:

ERCZ=1 ** Sets Z-axis deviation counter clear signal to "Output in case of an error stop"*

Notes:

- ERC[axis] command is 2-bit and sets the output condition of the ERC signal. The ERC[axis] command can write a new condition or read the condition based on the chart below.

Bit	Condition for Output	Setting
0	Error Stop	0: Disabled (Not used) 1: Enabled (Used)
1	Homing operation complete	0: Disabled (Not used) 1: Enabled (Used)

See also:

[Motor Setup](#), [Limits/Safety Setup](#),
[ERCD](#), [ERCP](#), [INP](#), [POL](#), [SDC](#), [SDE](#), [EXST](#), [IERR](#), [STORE](#)

Commander Manual

[2.1.2.2 Digital Servo](#)

2.1.3.1.1.1 Limit Switches

ERCD – Deviation Counter Clear Delay

Sets the delay time used between when an error is triggered and when the deviation counter clear signal is turned ON. This setting allows time between when the error is sent from a servo driver and when the servo driver can accept a error clear command.

ERCD – Deviation Counter Clear Delay

Applies to: CMD-4CR, CMD-4EX-SA

ASCII ERCD[axis]

Syntax:

Read:

ERCD[axis]

Where: [axis] is set to X, Y, Z, U

Write:

ERCD[axis]=[delay]

Where: [axis] is set to X, Y, Z, U

[delay] sets the OFF delay time (0-3)

0: 0 us

1: 12 us

2: 1.6 ms

3: 104 ms

Reply:

Command is accepted and executed

Read:

Re
turns the
value set
for delay
time (0-3).

Error

ERC

ERCD

ERCP

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:

ERCDY

* Returns the setting for the OFF delay time for the Y-axis

Write:

ERECZ=3 * Sets Z-axis deviation counter clear signal OFF delay time to 104 ms

Notes:

- See picture to understand interaction of error signal, ERC signal and the ERCD and ERCP settings.

See also:

[Motor Setup](#)

[ERC](#), [ERCP](#), [INP](#), [POL](#), [SDC](#), [SDE](#), [EXST](#), [IERR](#), [STORE](#)

Commander Manual

[2.1.2.2 Digital Servo](#)

ERCP – Deviation Counter Clear Pulse Width

ERCP – Deviation Counter Clear Pulse Width

Sets the output pulse width of the deviation counter clear signal.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII ERCP[axis]

Syntax:

Read:

ERCP[axis]

Where: [axis] is set to X, Y, Z, U

Write:

ERCP[axis]=[outpls]

Where: [axis] is set to X, Y, Z, U,

[outpls] Set the output pulse width (0-7)

0: 12 us

1: 102 us

2: 409 us

3: 1.6 ms

4: 13 ms

5: 52 ms

6: 104 ms

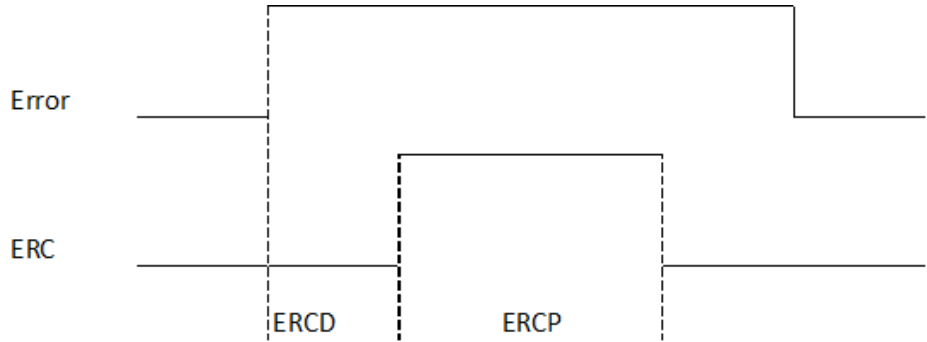
7: Level output

Reply:

Command is accepted and executed

Read:

Re
turns the
value set
for
output
pulse
width (0-7).



Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:

ERCPY
axis

* Returns the setting for the output pulse width for the Y-axis

Write:

ERCPZ=5
width to 52 ms

* Sets Z-axis deviation counter clear signal output pulse width to 52 ms

Notes:

- See picture to understand interaction of error signal, ERC signal and the ERCD and ERCP settings.

See also:

[Motor Setup](#)

[ERC](#), [ERCD](#), [INP](#), [POL](#), [SDC](#), [SDE](#), [EXST](#), [IERR](#), [STORE](#)

Commander Manual

[2.1.2.2 Digital Servo](#)

ESTOP – Emergency Stop

ESTOP – Emergency Stop

ESTOP is the software equivalent of triggering the [/CEMG](#) input. While the /CEMG input is low or the ESTOP command is received, all moving axes stop immediately, and no axis can start moving. By default, any axis stopped or prevented from moving by ESTOP or /CEMG triggers an EMG error MST[axis](bit16). The software must clear this error before the motion can resume. This command is write-only.

Applies to: CMD-2CR, CMD-4CR, CMD-4EX-SA, CMD-8CR

ASCII ESTOP

Syntax:

Write:

ESTOP

Reply

Command is accepted and executed

Write:

OK ** Status becomes an EMG error*

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Write: ESTOP ** Stops all axes immediately and EMG error flag is set*

Notes:

- ESTOP turns off the buffered move.
- The final pulse width may not be as expected in an emergency stop operation. As such, the motor drivers may not recognize the last pulses, so the Commander internal counter may not count these pulses. Therefore, after an emergency stop, the best practice is to perform a home routine to match the command position with the mechanical position.
- By default, **IERR=0** triggers an **EMG** error **MST[axis](bit16)** for the axes moving when the **/CEMG** input is low, or the **ESTOP** command is received. No axis can operate while the **/CEMG** signal is low, and an axis cannot operate until clearing the EMG error. An EMG error trigger for any axis that tries to operate while the **/CEMG** signal is low. However, if an axis was not moving when the **/CEMG** or **ESTOP** was triggered, the EMG error is not set for that axis, and thus that axis can operate immediately after the **/CEMG** input goes high.
- By setting **IERR=1**, the EMG error state is ignored. The motor stops when the appropriate switch is triggered or the command sent; however, it does not enter an error state.
- To clear the EMG error, use the **CLR[axis]** command to clear the error in ASCII mode or the **ECLEAR[axis]** command in standalone mode.

See also:

[Stop Motion](#), [Limits/Safety Setup](#),
[ABORT](#), [STOP](#)

Commander Manual

[2.1.3.1.2 Emergency stop](#)

EXSTA – External Start

EXSTA – External Start

This command enables/disables the external start functionality. This is applicable to all move

commands.

Applies to: CMD-2CR, CMD-4CR, CMD-4EX-SA, CMD-8CR
Firmware: V301+

ASCII EXSTA

Syntax:

Read:

EXSTA

Write:

EXSTA=[value]

Where: [value] is set to:

0: External start disabled, or
 1: External start enabled

Reply:

Command is accepted and executed

Read:

Returns enable/disable status, 0 = disabled, 1= enabled.

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:

EXSTA ** Returns the external start status*

Write:

EXSTA=0 ** Disables the external start function*

Notes:

- When set to 1 (enabled) /CSTA must be set to 0 (low) to start motion for any axis on the Commander.
- When set to 0 (disabled) motion will start immediately. The default setting is 0 (disabled).
- Before Version 301 the ASCII command for external start was **EXST**. V301 and newer versions added the **EXSTP** (external stop) command and changed **EXST** (external start) to **EXSTA**.

See also:

[System Setup](#), [Limits/Safety Setup](#),
[ERC](#), [ERCD](#), [ERCP](#), [INP](#), [POL](#), [SDC](#), [SDE](#), [IERR](#), [STORE](#)

Commander Manual[2.1.1 Connect Multiple Commander Cores in a System](#)[2.1.1.2 Synchronization with external trigger](#)[2.1.3.1.4 External Start](#)**EXSTP – External Stop****EXSTP – External Stop**

This command enables/disables the external stop functionality. This is applicable to all move commands.

Applies to: CMD-2CR, CMD-4CR, CMD-4EX-SA, CMD-8CR

Firmware: V301+

ASCII EXSTP

Syntax:**Read:**

EXSTP

Write:

EXSTP=[value]

Where: [value] is set to:

- | |
|--|
| 0: External stop disabled, |
| 1: External Decelerate-stop enabled (same as STOP command), or |
| 2: External stop immediately enabled (same as ABORT) |

Reply:

Command is accepted and executed

Read:

Returns enable/disable status, 0 = disabled, 1= enabled.

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:**Read:**

EXSTP * *Returns the external stop status*

Write:

EXSTP=0 * *Disables the external stop function*

Notes:

- Added with V301 firmware.

See also:

[System Setup](#), [Limits/Safety Setup](#),
[ERC](#), [ERCD](#), [ERCP](#), [INP](#), [POL](#), [SDC](#), [SDE](#), [IERR](#), [STORE](#)

Commander Manual

[2.1.1.2 Synchronization with external trigger](#)
[2.1.3.1.3 External Start](#)

GS – Subroutine**GS – Subroutine**

Write-only command that executes an A-Script subroutine.

Applies to: CMD-4CR, CMD-4EX-SA, CMD-2ED-SA, CMD-2EX

ASCII	GS
Standalone	GOSUB

Syntax:**Write:**

GS[subNo]
 GOSUB[subNo]

Where: [subNo] specifies the subroutine number to execute (0 – 31)

Reply:

Command is accepted and executed

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:**Write:**

GS5 ** A-Script subroutine 5 will be executed*

Notes:

- None

See also:

[Standalone Programming Commands](#)
[SACTRL](#), [SAP](#), [SASTAT](#), [SLOAD](#), [V](#), [SPC](#)

Commander Manual

[2.1.1.1 Identification Number](#)

HOME, H – Homing

HOME, H – Homing

Starts one of the built-in homing sequences. **Homing** is a built in process by which the controller will move the motor while using the home, limit, slowdown, or Z-index inputs to determine the zero-reference position (home point). This point is then used as the reference for all subsequent moves.

The **Performax** series has five (5) different built-in home search routines.

See [Home modes – Performax](#) for more information of the homing routines available on the **Performax** series

The **Commander** series has thirteen (13) different built-in home search routines.

See [Home modes – Commander](#) for more information of the homing routines available on the **Commander** series

See [Home mode Cross reference](#) for a cross reference between the **Performax** and **Commander** series homing modes.

Homing Mode Actions

This section will review the actions of each homing mode for the **Performax** and **Commander** series controllers.

Performax

The **Performax** series has five (5) different built-in home search routines.

The charts below indicates which type of limits are used in each mode, along with home approach speed and direction and end-point status. ● indicates input required, ○ indicates input optional.

Applies to: CMD-2ED-SA, CMD-2EX-SA

The chart below indicates which type of limits are used in each mode, along with home approach speed and direction and end-point status. ● indicates input required.

Homing Mode		Home	End Limits	Z-index	Slow Down	Approaches Home at		Direction to Trigger Zero	End Point
ASCII	Standalone					High Speed	Low Speed		
H[axis][+/-]	HOME[axis][+/-]	●				H		Dir of Home	Non-
L[axis][+/-]	LHOME[axis][+/-]		●				L	Dir of Home	Ze

ZH[axis][+/-]	ZHOME[axis][+/-]	•		•			L	Dir of Home	Ze
Z[axis][+/-]	ZOME[axis][+/-]			•		H		Dir of Home	Non-
HL[axis][+/-]	HLOME[axis][+/-]	•	•				L	Reverse Dir	Ze

ASCII [mode][axis][dir]
 Standalone [mode][axis][dir]

Syntax:

Write:

ASCII

[mode][axis][dir]

Standalone

[mode][axis][dir]

Where: [mode] specifies the homing to be used [ASCII] (H, L, ZH, Z, HL), [Standalone] (HOME, LHOME, ZHOME, ZOME, HLOME) [axis] specifies the axis to be homed (X, Y, Z, U) [dir] specifies the direction used for homing (+/-)

Reply:

Command is accepted and executed

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Write:

HX+ * Starts moving the X axis in the + using the Home Input only
 mode

ZHZ- * Starts moving the Z axis in the – using the Home Input and Z-index mode

Notes:

See detailed information on inputs, timing, for each homing mode, see its individual page.

Reply:

Command is accepted and executed

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Write:

HX+ * Starts moving the X axis in the + using the Home Input only
mode

ZHZ- * Starts moving the Z axis in the – using the Home Input and Z-
index mode

Applies to: CMD-4CR, CMD-4EX-SA, CMD-2ED-SA, CMD-2EX

ASCII H
Standalone HOME

Syntax:

Write:

ASCII

H[axis][dir][mode]

Each target potion must be separated

by the ":" character

Standalone

HOME[axis][dir][mode]

There is so separation

between each target potion

Where: [axis] specifies the axis to be homed (X, Y, Z, U)
[dir] specifies the speed range according to the chart below:

[rmg]	HSPD Value			Minimum LSPD Value	Magnification
1	1	–	65,000	1	1
2	65,000	–	130,000	2	2
3	130,000	–	325,000	5	5
4	325,000	–	650,000	10	10
5	650,000	–	1,300,000	20	20
6	1,300,000	–	3,200,000	50	50
7	3,200,000	–	6,000,000	100	100

Reply:

Command is accepted and executed

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:

SSPDMX ** returns the speed range value code (rmg – see Chart above)*

Write:

SSPDM=5 ** Sets the speed range value to 20x magnification (rmg=5)*

Notes:

- The **SSPDM[axis]** setting will limit the maximum speed available for the **HSPD[axis]** according to the speed range table listed above. If the SSPDM is set for a lower speed range than the value for the HSPD command, the actual high-speed operation will be limited to the top value of the set range.
- The speed range must be set prior to issuing the SSPD command if the new speed value is in a different range than what is already set. Otherwise, an ?Error SSPD will be indicated.

See also:

Home mode Cross reference

[Axis Motion](#)

[ZMOVE, ZCNT](#)

Commander Manual

[2.2.4 Positioning operation](#)

Home mode Cross reference

Home mode Cross reference

PMX-2EX/2ED	PMX-4ET/4EX	CMD	
H[axis][+/-]	H[axis][+/-]0	Mode 0	Note for 2EX and 2ED LCA is a value for how far away from the Limit the axis v
L[axis][+/-]	H[axis][+/-]1	Mode 6	Note for CMD ZCNT is added to go a set number of Z counts before setting Z
ZH[axis][+/-]	H[axis][+/-]2	Mode 2	Note for CMD ZCNT is set to go a set number of Z counts before setting Zero
Z[axis][+/-]	H[axis][+/-]3	ZMOVE	When using ZMOVE, the Counter is not automatically set to Zero
HL[axis][+/-]	H[axis][+/-]4	Mode 1	Note for 2EX and 2ED HCA is a value for how far away from the Home sensor,

Homing Mode Actions

This section will review the actions of each homing mode for the **Performax** and **Commander** series controllers.

Performax

The **Performax** series has five (5) different built-in home search routines.

The chart below indicates which type of limits are used in each mode, along with home approach speed and direction and end-point status. • indicates input required.

Homing Mode	Home	End Limits	Z-index	Slow Down	Approaches Home at		Direction to Trigger Zero	Ending Point
					High Speed	Low Speed		
H[axis][+/-]	•				H		Dir of Home	Non-Zero
L[axis][+/-]		•				L	Dir of Home	Zero

ZH[axis] [+/-]	•		•			L	Dir of Home	Zero
Z[axis][+/-]			•		H		Dir of Home	Non-Zero
HL[axis] [+/-]	•	•				L	Reverse Dir	Zero

Homing Mode	Home	End Limits	Z-index	Slow Down	Approaches Home at		Direction to Trigger Zero	Ending Point
					High Speed	Low Speed		
Mode 0	•				H		Dir of Home	Non-Zero
Mode 1		•				L	Dir of Home	Zero
Mode 2	•		•			L	Dir of Home	Zero
Mode 3			•		H		Dir of Home	Non-Zero
Mode 4	•	•				L	Reverse Dir	Zero

Commander

The **Commander** series has thirteen (13) different built-in home search routines.

Commander core utilizes 13 homing search methods, home limit, end of travel limit, or Z-index pulse, or combinations to complete a homing operation. Homing is used to establish a zero point, or home, as a reference for absolute motion actions. Some homing methods stop in a non-zero position, using the position at which the home, end of travel, or Z-index pulse triggers as the zero-reference position. By default, the homing modes move using the high-speed operation. Low-speed versions are available by adding 16 to the homing mode.

The chart below indicates which type of limits are used in each mode, along with home approach speed and direction and end-point status. • indicates input required, ○ indicates input optional.

Homing Mode	Home	End Limits	Z-index	Slow Down	Approaches Home at		Direction to Trigger Zero	Ending Point	High Speed Homing	S H
					High Speed	Low Speed				
Mode 0	•			○	H		Dir of Home	Non-Zero	0	
Mode 1	•			○		L	Dir of Home	Zero	1	
Mode 2	•		•	○		L	Dir of Home	Zero	2	
Mode 3	•		•	○	H		Dir of Home	Non-Zero	3	
Mode 4	•		•	○		L	Reverse Dir	Zero	4	
Mode 5	•		•	○	H		Reverse Dir	Non-Zero	5	
Mode 6		•		○	H		Reverse Dir	Zero	6	
Mode 7		•	•	○		L	Reverse Dir	Zero	7	
Mode 8		•	•	○	H		Reverse Dir	Non-Zero	8	
Mode 9	•			○	H		Reverse Dir	Zero	9	
Mode 10	•		•	○	H		Reverse Dir	Zero	10	
Mode 11	•		•	○	H		Dir of Home	Zero	11	
Mode 12		•	•	○	H		Dir of Home	Zero	12	

Consider the system mechanics carefully when setting up limit switches, using precaution to set the limits for the appropriate axis movement and consider the deceleration distance when establishing end of travel limits relative to the hard stop positions.

NOTE: For homing modes 9 – 12, encoder feedback is required. When using a stepper motor and homing

with modes 9 – 12, an encoder is required and the command pulse count and encoder pulse count must be equal.

More information on the homing modes can be found in Section 9-5-1 of the PCL6045BL users manual.

CMD/PMX Homing Mode 0 [H[axis][+/-]]

Home Mode 0 is available [with Slow Down \(SD\)](#) switch or [without SD](#) switch.

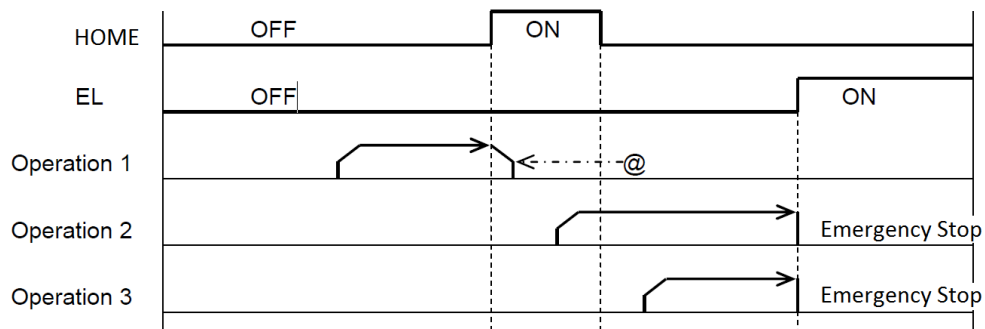
This Home Mode is available with [low-speed operation mode 16](#) on the Commander series.

Homing mode 0 without Slow Down (SD) disabled.

Applies to: CMD-4CR, CMD-4EX-SA, PMX-4EX-SA, PMX-4ET-SA, PMX-2ED-SA, PMX-2EX

This homing method works with a home limit switch input and sets home when the motion stops after the home limit switch is encountered.

Series	Homing Mode	Home Limit	End Limits	Z-index	Slow Down	Approaches Home at		Direction to Trigger Zero	Ending Point
						High Speed	Low Speed		
CMD all PMX-4EX-SA PMX-4ET-SA	Mode 0	•			o	H		Dir of Home	Non-Zero
PMX-2ED-SA PMX-2EX-SA	H[axis][+/-]	•				H		Dir of Home	Non-Zero



Operation 1:

- A) Homing is started toward home switch, accelerate from LSPD to HSPD
- B) Deceleration starts when the home (ORG) input point is detected, zero position is set.
- C) Homing is complete when deceleration stops.
- D) ERC pulse is sent. (CMD only)

Operation 2:

- A) Homing is started while home input is activated.
- B) Acceleration from LSPD to HSPD.
- C) End limit is encountered, immediate stop.

Operation 3:

- A) Homing is started moving away from the home input switch.
- B) Acceleration from LSPD to HSPD.
- C) End limit is encountered, immediate stop.

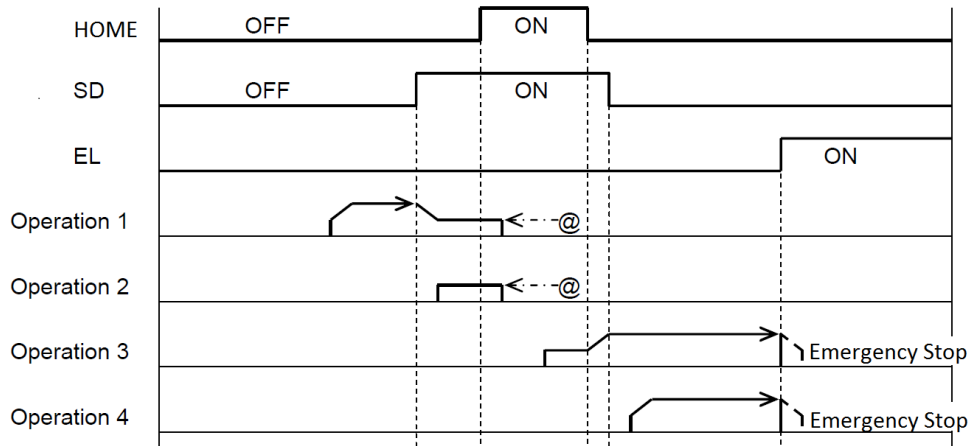
More information on the homing modes can be found in Section 9-5-1 of the PCL6045BL users manual.

Homing mode 0 with Slow Down (SD) enabled.

Applies to: CMD-4CR, CMD-4EX-SA

In this homing mode, an input will first trigger a deceleration to a lower speed point and then stops and sets home as the home limit switch is reached.

Homing Mode	Home Limit	End Limits	Z-index	Slow Down	Approaches Home at		Direction to Trigger Zero	Ending Point
					High Speed	Low Speed		
Mode 0	•			•		L	Dir of Home	Non-Zero



Operation 1:

- A) Homing is started toward home switch, accelerate from LSPD to HSPD
- B) Deceleration starts to LSPD when the SD input point is detected, continue towards home.
- C) Stop when home input is detected, homing complete.
- D) ERC pulse is sent.

Operation 2:

- A) Homing is started while SD input is activated.
- B) Continue traveling at LSPD towards home input.
- C) Stop when home input is detected, zero position is set, homing complete.
- D) ERC pulse is sent.

Operation 3:

- D) Homing is started while SD input is activated and is moving away from the home input switch.
- E) Travel at LSPD until SD input is deactivated, then accelerate from LSPD to HSPD.
- F) End limit is encountered, immediate stop.

Operation 4:

- A) Homing is started moving away from the home input switch and SD limit is deactivated.
- B) Acceleration from LSPD to HSPD.
- C) End limit is encountered, immediate stop.

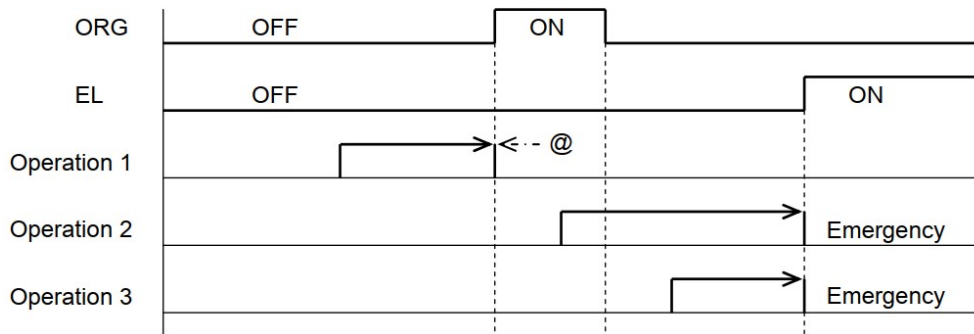
More information on the homing modes can be found in Section 9-5-1 of the PCL6045BL users manual.

Homing mode 16 (Homing mode 0 at low-speed)

Applies to: CMD-4CR, CMD-4EX-SA V129+

This homing method works with a home limit switch input and sets home when the motion stops after the home limit switch is encountered.

Homing Mode	Home Limit	End Limits	Z-index	Slow Down	Approaches Home at		Direction to Trigger Zero	Ending Point
					High Speed	Low Speed		
Mode 16	•			•		L	Dir of Home	Non-Zero



Operation 1:

- A) Homing is started toward home switch, at LSPD.
- B) Motion stops when the home (ORG) input point is detected, zero position is set.
- C) Homing is complete when motion stops.
- D) ERC pulse is sent. (CMD only)

Operation 2:

- A) Homing is started while home input is activated.
- B) Motion is at LSPD.
- C) End limit is encountered, immediate stop.

Operation 3:

- A) Homing is started moving away from the home input switch.
- B) Motion is at LSPD.
- C) End limit is encountered, immediate stop.

More information on the homing modes can be found in Section 9-5-1 of the PCL6045BL users manual.

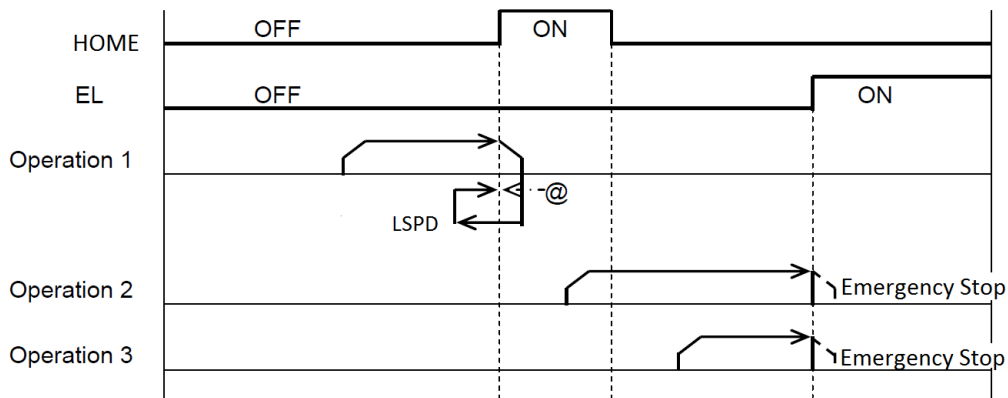
CMD Homing Mode 1 PMX Mode 4

This Home Mode is available with [low-speed operation mode 17](#) on the Commander series.

Applies to: CMD-4CR, CMD-4EX-SA

This homing method decelerates when it first encounters the home limit switch, reverses direction to back off the limit switch, and then approaches the home limit switch again at a lower speed point and sets home when the home limit switch is encountered a second time.

Homing Mode	Home Limit	End Limits	Z-index	Slow Down	Approaches Home at		Direction to Trigger Zero	Ending Point
					High Speed	Low Speed		
Mode 1	●			○		L	Dir of Home	Zero



Operation 1:

- A) Homing operation is started, accelerate from LSPD to HSPD.
- B) Deceleration starts when home limit is encountered.

- C) Reverse direction at move at LSPD towards home limit.
- D) Move at LSPD until home limit is turned off and reverse direction.
- E) Operation stops when home limit is encountered again, zero position is set.
- F) ERC pulse is sent.

Operation 2:

- A) Homing is started while home input is activated.
- B) Acceleration from LSPD to HSPD.
- C) End limit is encountered, immediate stop.

Operation 3:

- A) Homing is started, moving away from the home input switch.
- B) Acceleration from LSPD to HSPD.
- C) End limit is encountered, immediate stop.

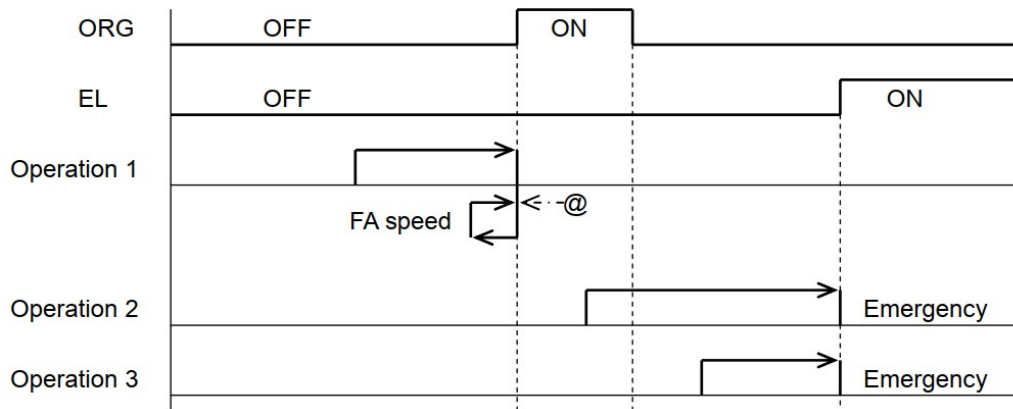
More information on the homing modes can be found in Section 9-5-1 of the PCL6045BL users manual.

Homing mode 17, Homing mode 1 at low-speed

Applies to: CMD-4CR, CMD-4EX-SA V129+

This homing method stops when it first encounters the home limit switch, reverses direction to back off the limit switch, and then approaches the home limit switch again at a lower speed point and sets home when the home limit switch is encountered a second time.

Homing Mode	Home Limit	End Limits	Z-index	Slow Down	Approaches Home at		Direction to Trigger Zero	Ending Point
					High Speed	Low Speed		
Mode 17	●			○		L	Dir of Home	Zero



Operation 1:

- A) Homing operation is started at LSPD
- B) Motion stops when home limit is encountered.
- C) Reverse direction at move at LSPD towards home limit.
- D) Move at LSPD until home limit is turned off and reverse direction.
- E) Operation stops when home limit is encountered again, zero position is set.
- F) ERC pulse is sent.

Operation 2:

- A) Homing is started while home input is activated.
- B) Motion starts at LSPD.
- C) End limit is encountered, immediate stop.

Operation 3:

- A) Homing is started moving away from the home input switch.
- B) Motion starts at LSPD.
- C) End limit is encountered, immediate stop.

More information on the homing modes can be found in Section 9-5-1 of the PCL6045BL users manual.

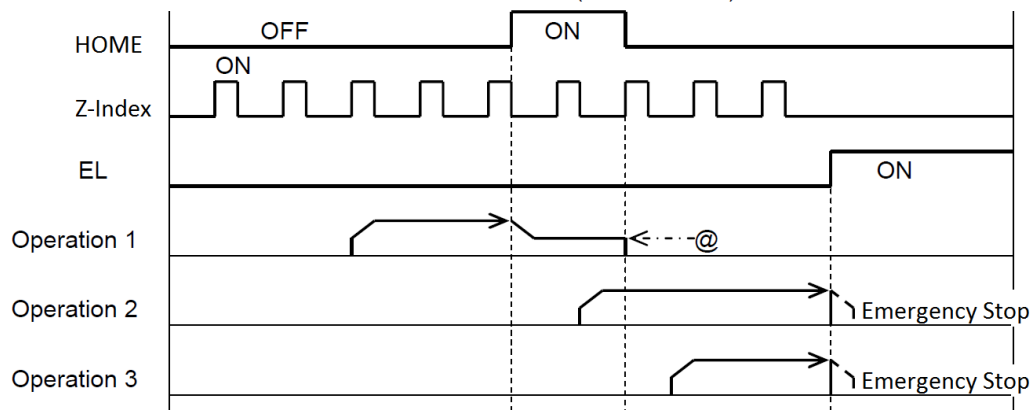
CMD/PMX Homing Mode 2 [ZH[axis]] [+/-]

This Home Mode is available with [low-speed operation a mode 18](#) on the Commander series.

Applies to: CMD-4CR, CMD-4EX-SA

Homing method 2 utilizes both a home limit switch and the Z-index pulse of the encoder. Once the home limit switch is encountered, the motor slows to a lower speed point and sets home as it encounters the Z-index pulse.

Series	Homing Mode	Home Limit	End Limits	Z-index	Slow Down	Approaches Home at		Direction to Trigger Zero	Ending Point
						High Speed	Low Speed		
	Mode 2	●		●	○		L	Dir of Home	Zero
PMX-2ED-SA PMX-2EX-SA	ZH[axis][+/-]								



Operation 1:

- Homing operation is started, accelerate from LSPD to HSPD.
- Deceleration starts when home limit is encountered.
- Continue moving at LSPD until specified number of Z-Index pulses are counted ([ZCNT](#)), zero position is set.
- Operation stops and is completed.
- ERC pulse is sent.

Operation 2:

- Homing is started while home input is activated.
- Acceleration from LSPD to HSPD.
- End limit is encountered, immediate stop.

Operation 3:

- Homing is started moving away from the home input switch.
- Acceleration from LSPD to HSPD.
- End limit is encountered, immediate stop.

More information on the homing modes can be found in Section 9-5-1 of the PCL6045BL users manual.

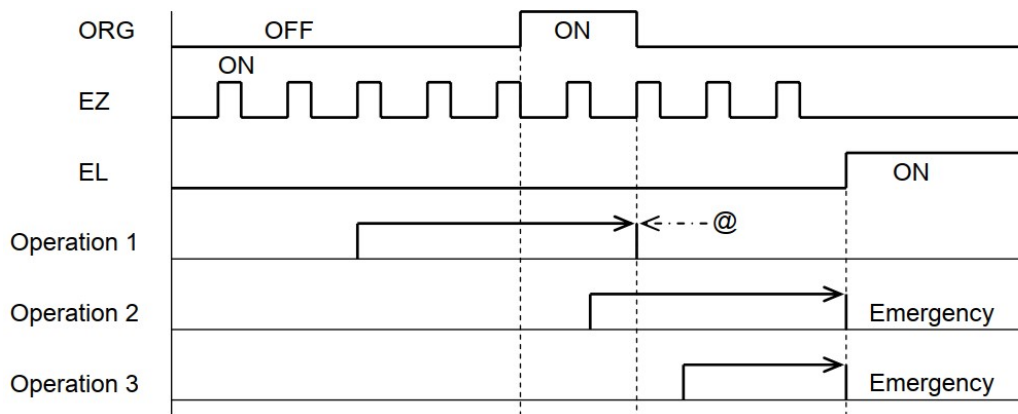
Homing mode 18, Homing mode 2 at low-speed

Applies to: CMD-4CR, CMD-4EX-SA V129+

Homing method 18 utilizes both a home limit switch and the Z-index pulse of the encoder. Once the home limit switch is encountered, the motor slows to a lower speed point and sets home as it encounters the Z-index pulse.

Homing Mode	Home Limit	End Limits	Z-index	Slow Down	Approaches Home at		Direction to Trigger Zero	Ending Point
					High	Low		

					Speed	Speed		
Mode 18	•		•	○		L	Dir of Home	Zero



Operation 1:

- A) Homing operation is started, at LSPD.
- B) Motion continues at LSPD when home limit is encountered.
- C) Continue moving at LSPD until specified number of Z-Index pulses are counted ([ZCNT](#)), zero position is set.
- D) Operation stops and is completed.
- E) ERC pulse is sent.

Operation 2:

- A) Homing is started while home input is activated.
- B) Motion starts at LSPD.
- C) End limit is encountered, immediate stop.

Operation 3:

- A) Homing is started moving away from the home input switch.
- B) Motion starts at LSPD.
- C) End limit is encountered, immediate stop.

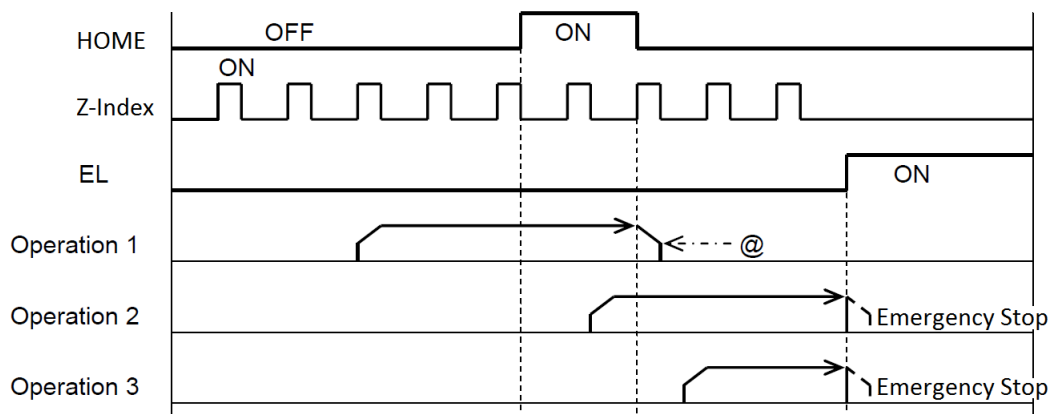
More information on the homing modes can be found in Section 9-5-1 of the PCL6045BL users manual.

CMD Homing Mode 3

Homing Mode 3

Homing mode 3 uses both the home limit switch and Z-index pulse. The deceleration to the home set point occurs after the move passes through the home limit switch and encounters the Z-index pulse.

Homing Mode	Home Limit	End Limits	Z-index	Slow Down	Approaches Home at		Direction to Trigger Zero	Ending Point
					High Speed	Low Speed		
Mode 3	•		•	○	H		Dir of Home	Non-Zero



Operation 1:

- A) Homing operation is started, accelerate from LSPD to HSPD.
- B) Continue moving at HSPD after home limit is encountered.
- C) Decelerate when specified number of Z-Index pulses are counted ([ZCNT](#)), zero position is set.

- D) Operation decelerates and stops.
- E) ERC pulse is sent.

Operation 2:

- A) Homing is started while home input is activated.
- B) acceleration from LSPD to HSPD.
- C) End limit is encountered, immediate stop.

Operation 3:

- A) Homing is started moving away from the home input switch.
- B) acceleration from LSPD to HSPD.
- C) End limit is encountered, immediate stop.

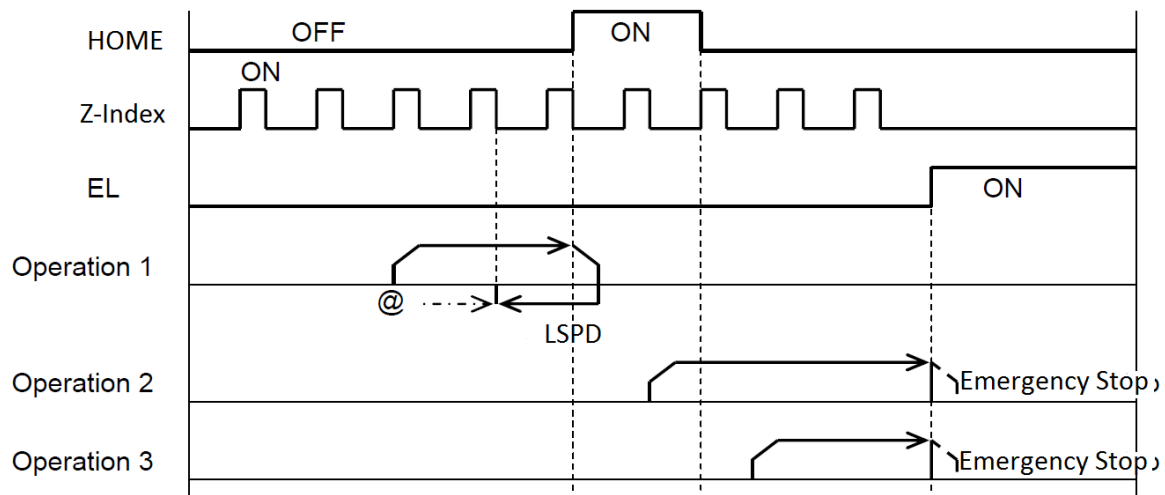
More information on the homing modes can be found in Section 9-5-1 of the PCL6045BL users manual.

CMD Homing Mode 4

Homing Mode 4

For homing mode 4, the motor decelerates upon reaching the home limit switch and reverses direction upon reaching the lower speed point. Home is set when the Z-index pulse is encountered.

Homing Mode	Home Limit	End Limits	Z-index	Slow Down	Approaches Home at		Direction to Trigger Zero	Ending Point
					High Speed	Low Speed		
Mode 4	●		●	○		L	Reverse Dir	Zero



Operation 1:

- A) Homing operation is started, accelerate from LSPD to HSPD.
- B) Begin deceleration after home limit is encountered.
- C) Reverse direction at LSPD after deceleration complete.
- D) Continue moving at LSPD until specified number of Z-Index pulses are counted (ZCNT).
- E) Operation stops and is completed; zero position is set.
- F) ERC pulse is sent.

Operation 2:

- A) Homing is started while home input is activated.
- B) acceleration from LSPD to HSPD.
- C) End limit is encountered, immediate stop.

Operation 3:

- A) Homing is started moving away from the home input switch.
- B) acceleration from LSPD to HSPD.
- C) End limit is encountered, immediate stop.

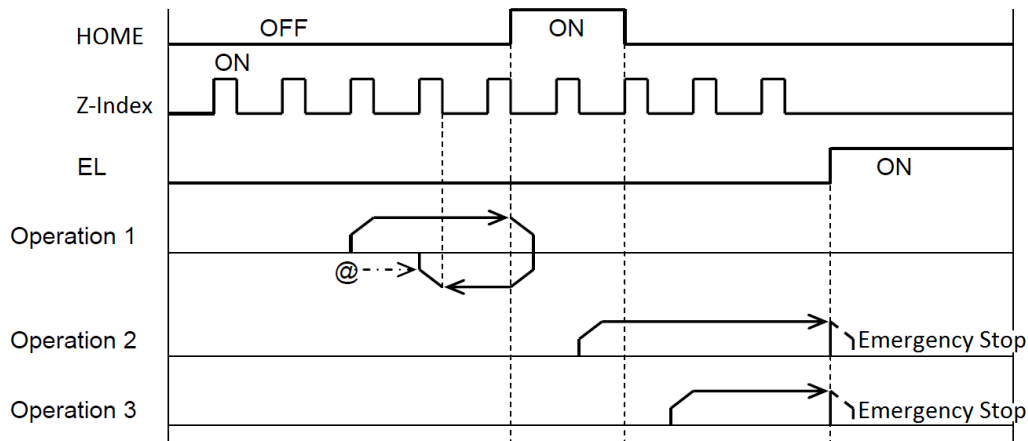
More information on the homing modes can be found in Section 9-5-1 of the PCL6045BL users manual.

Homing Mode 5

Homing Mode 5

In homing mode 5, the home limit switch initiates a full reversal from top speed in the initial direction to top speed in the reverse direction. Homing is completed when the Z-index pulse is encountered and the motion comes to a stop.

Homing Mode	Home Limit	End Limits	Z-index	Slow Down	Approaches Home at		Direction to Trigger Zero	Ending Point
					High Speed	Low Speed		
Mode 6		•		○	H		Reverse Dir	Zero



Operation 1:

- Homing operation is started, accelerate from LSPD to HSPD.
- Begin deceleration to LSPD after home limit is encountered.
- Reverse direction and accelerate to HSPD.
- Continue moving at HSPD until specified number of Z-Index pulses are counted (ZCNT), zero position is set.
- Operation decelerates, stops and is completed.
- ERC pulse is sent.

Operation 2:

- Homing is started while home input is activated.
- acceleration from LSPD to HSPD.
- End limit is encountered, immediate stop.

Operation 3:

- Homing is started moving away from the home input switch.
- acceleration from LSPD to HSPD.
- End limit is encountered, immediate stop.

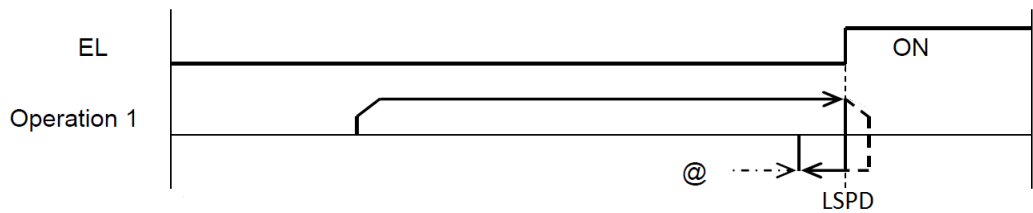
More information on the homing modes can be found in Section 9-5-1 of the PCL6045BL users manual.

CMD Homing Mode 6 PMX Mode 1

Applies to: CMD-4CR, CMD-4EX-SA, PMX-4EX-SA, PMX-4ET-SA, PMX-2ED-SA, PMX-2EX

Homing mode 6 uses the end of travel limit switch to trigger home location. The end limit is encountered and the motor reverses off the end of travel limit at a lower speed and sets home position.

Series	Homing Mode	Home Limit	End Limits	Z-index	Slow Down	Approaches Home at		Direction to Trigger Zero	Ending Point
						High Speed	Low Speed		
CMD all	Mode 6		•		○	H		Reverse Dir	Zero
PMX-4EX-SA PMX-4ET-SA	Mode 1		•						
PMX-2ED-SA PMX-2EX-SA	L[axis][+/-]		•						



Operation 1:

- A) Homing operation is started, accelerate from LSPD to HSPD.
- B) Immediate stop when the end of travel limit switch is encountered.
- C) Reverse direction at LSPD until end limit switch turns off.
- D) Operation stopped immediately and homing operation is completed, zero position is set.
- E) ERC pulse is sent.

Note: When utilizing the end travel limits, the ERC command will clear the end-of-travel limit flag in the direction of the home during the homing operation. Refer to **ERC** command in the Command Reference for further details.

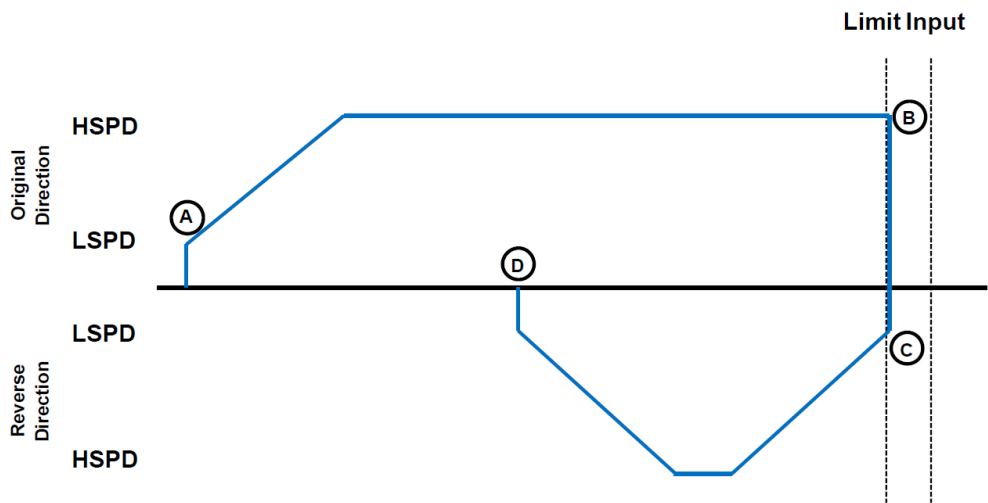
More information on the homing modes can be found in Section 9-5-1 of the PCL6045BL users manual.

L[axis][+/-]

Applies to: PMX-2ED-SA, PMX-2EX

Homing mode 6 uses the end of travel limit switch to trigger home location. The end limit is encountered and the motor reverses off the end of travel limit at a lower speed and sets home position.

Series	Homing Mode	Home Limit	End Limits	Z-index	Slow Down	Approaches Home at		Direction to Trigger Zero	Ending Point
						High Speed	Low Speed		
PMX-2ED-SA PMX-2EX-SA	L[axis][+/-]		•			H		Reverse Dir	Zero



Operation 1:

- A) Homing operation is started, accelerate from LSPD to HSPD.
- B) Immediate stop when the end of travel limit switch is encountered.
- C) Reverse direction at HSPD by the amount defined by the limit correction amount (LCA).
- D) Operation stopped immediately and homing operation is completed, zero position is set.

See also:

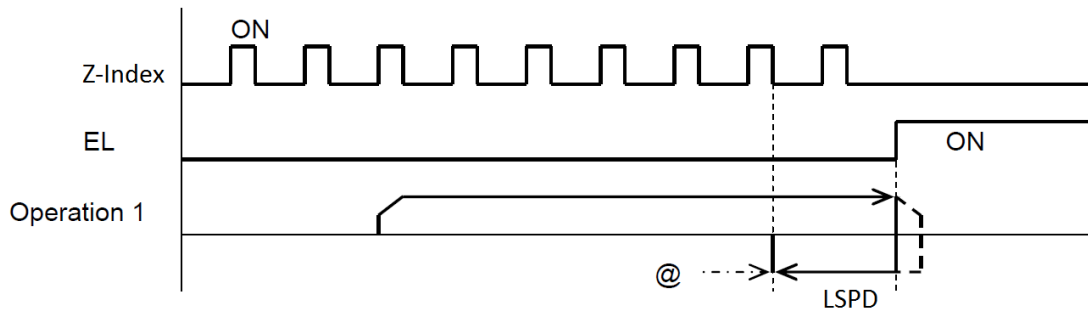
[Homing](#)
[LCA](#)

Homing Mode 7

Homing Mode 7

Homing mode 7 is similar to homing mode 6, except it reverses until it locates the Z-index pulse after encountering the end of travel limit switch. Once the Z-index is reached, homing is completed.

Homing Mode	Home Limit	End Limits	Z-index	Slow Down	Approaches Home at		Direction to Trigger Zero	Ending Point
					High Speed	Low Speed		
Mode 7		•	•	○		L	Reverse Dir	Zero



Operation 1:

- Homing operation is started, accelerate from LSPD to HSPD.
- Immediate stop when the end of travel limit switch is encountered.
- Reverse direction at LSPD until specified number of Z-Index pulses are counted (ZCNT).
- Operation stopped immediately and homing operation is completed, zero position is set.
- ERC pulse is sent.

Note: When utilizing the end travel limits, the ERC command will clear the end-of-travel limit flag in the direction of the home during the homing operation. Refer to **ERC** command in the Command Reference for further details.

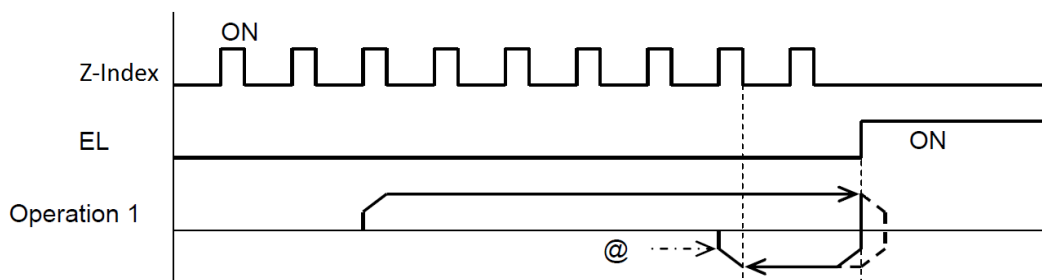
More information on the homing modes can be found in Section 9-5-1 of the PCL6045BL users manual.

Homing Mode 8

Homing Mode 8

In homing mode 8, the end of travel limit switch triggers an immediate stop and then full reversal at top speed until Z-index is encountered and zero position is set. At this point the motor decelerates and stops.

Homing Mode	Home Limit	End Limits	Z-index	Slow Down	Approaches Home at		Direction to Trigger Zero	Ending Point
					High Speed	Low Speed		
Mode 8		•	•	○	H		Reverse Dir	Non-Zero



Operation 1:

- Homing operation is started, accelerate from LSPD to HSPD.
- Immediate stop when the end of travel limit switch is encountered.
- Reverse direction accelerating from LSPD to HSPD and move until specified number of Z-Index pulses are counted (ZCNT).
- As the specified number of Z-index pulses is encountered, zero position is set and deceleration to LSPD begins.
- Homing operation is completed by deceleration and stop.
- ERC pulse is sent.

Note: When utilizing the end travel limits, the ERC command will clear the end-of-travel limit flag in the direction of

the home during the homing operation. Refer to **ERC** command in the Command Reference for further details.

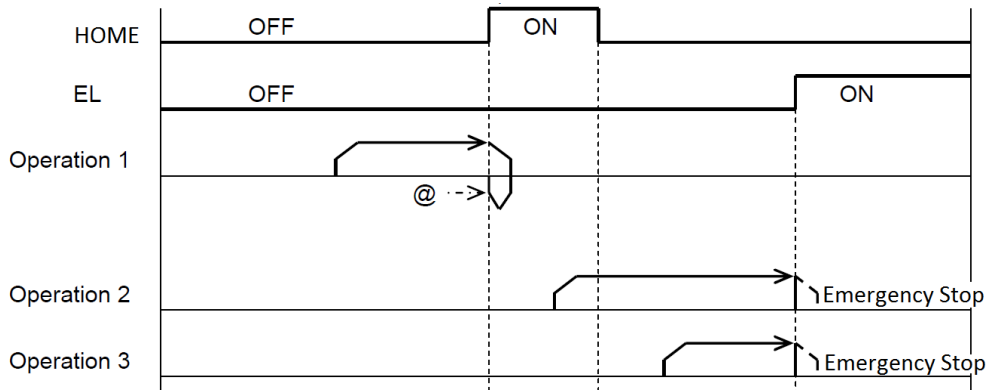
More information on the homing modes can be found in Section 9-5-1 of the PCL6045BL users manual.

Homing Mode 9

Homing Mode 9

Homing Mode 9 uses the home limit switch to initiate a deceleration, then continues with a quick index back to the position where the home limit was encountered.

Homing Mode	Home Limit	End Limits	Z-index	Slow Down	Approaches Home at		Direction to Trigger Zero	Ending Point
					High Speed	Low Speed		
Mode 9	●			○	H		Reverse Dir	Zero



Operation 1:

- Homing operation is started, accelerate from LSPD to HSPD.
- Deceleration begins as the home limit switch is encountered and zero position is set.
- Begin zero-point return operation back to home position at the end of deceleration, accelerate from LSPD to HSPD.
- Homing operation is completed when the 0 position is reached.
- ERC pulse is sent.

Operation 2:

- Homing is started while home input is activated.
- acceleration from LSPD to HSPD.
- End limit is encountered, immediate stop.

Operation 3:

- Homing is started moving away from the home input switch.
- acceleration from LSPD to HSPD.
- End limit is encountered, immediate stop.

NOTE: For homing modes 9 – 12, encoder feedback is required. When using a stepper motor and homing with modes 9 – 12, an encoder is required and the command pulse count and encoder pulse count must be equal.

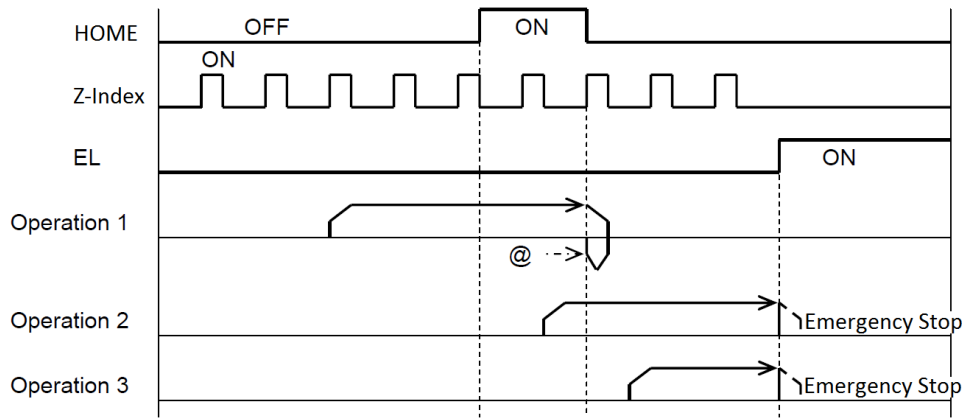
More information on the homing modes can be found in Section 9-5-1 of the PCL6045BL users manual.

Homing Mode 10

Homing Mode 10

Homing mode 10 is similar to homing mode 3 as it uses both the home limit switch and Z-index pulse. Deceleration occurs after the move passes through the home limit switch and encounters the Z-index pulse. The operation continues with a quick index back to the position where the Z-index pulse was encountered.

Homing Mode	Home Limit	End Limits	Z-index	Slow Down	Approaches Home at		Direction to Trigger Zero	Ending Point
					High Speed	Low Speed		
Mode 10	●		●	○	H		Reverse Dir	Zero



Operation 1:

- A) Homing operation is started, accelerate from LSPD to HSPD.
- B) Continue moving at HSPD as the home limit switch is encountered.
- C) Deceleration begins when specified number of Z-index-pulses is encountered (ZCNT), zero set.
- D) At end of deceleration, a zero-point return operation begins to the position where Z-index pulse was encountered, accelerating from LSPD to HSPD and decelerating and stopping at 0 position.
- E) Homing is completed when the 0 position is reached.
- F) ERC pulse is sent.

Operation 2:

- A) Homing is started while home input is activated.
- B) acceleration from LSPD to HSPD.
- C) End limit is encountered, immediate stop.

Operation 3:

- A) Homing is started moving away from the home input switch.
- B) acceleration from LSPD to HSPD.
- C) End limit is encountered, immediate stop.

NOTE: For homing modes 9 – 12, encoder feedback is required. When using a stepper motor and homing with modes 9 – 12, an encoder is required and the command pulse count and encoder pulse count must be equal.

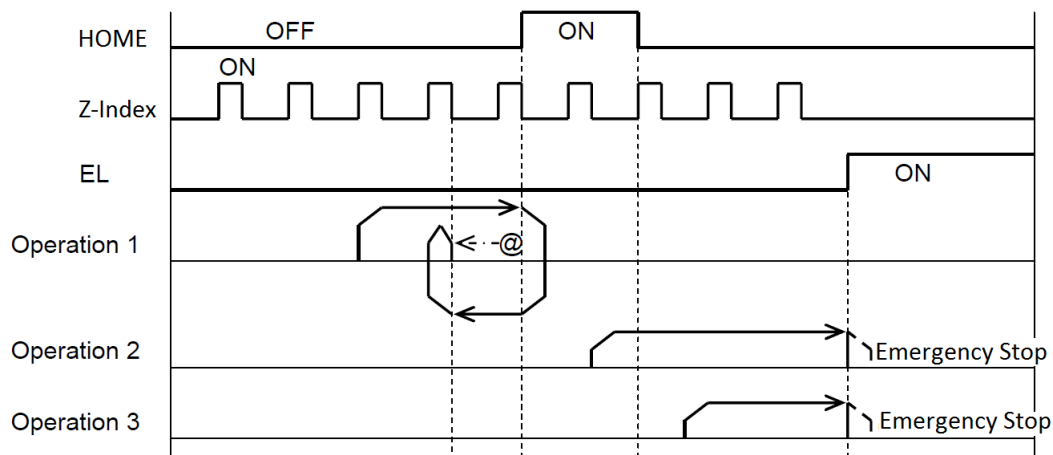
More information on the homing modes can be found in Section 9-5-1 of the PCL6045BL users manual.

Homing Mode 11

Homing Mode 11

In homing mode 11, the home limit switch is used to initiate a deceleration. The operation then continues with a fast move in the reverse direction until the Z-index pulse is encountered, where a quick index reverses motion back to the position where the Z-index pulse was encountered.

Homing Mode	Home Limit	End Limits	Z-index	Slow Down	Approaches Home at		Direction to Trigger Zero	Ending Point
					High Speed	Low Speed		
Mode 11	•		•	○	H		Dir of Home	Zero



Operation 1:

- Homing operation is started, accelerate from LSPD to HSPD, then decelerate as the home limit switch is encountered.
- Reverse direction, accelerating from LSPD to HSPD, then decelerate when specified number of Z-index-pulses is encountered (ZCNT), zero position is set.
- Start a zero-point return move to the position where the Z-index pulse was encountered, accelerating from LSPD to HSPD then decelerating and stopping at 0 position.
- ERC pulse is sent.

Operation 2:

- Homing is started while home input is activated.
- acceleration from LSPD to HSPD.
- End limit is encountered, immediate stop.

Operation 3:

- Homing is started moving away from the home input switch.
- acceleration from LSPD to HSPD.
- End limit is encountered, immediate stop.

NOTE: For homing modes 9 – 12, encoder feedback is required. When using a stepper motor and homing with modes 9 – 12, an encoder is required and the command pulse count and encoder pulse count must be equal.

More information on the homing modes can be found in Section 9-5-1 of the PCL6045BL users manual.

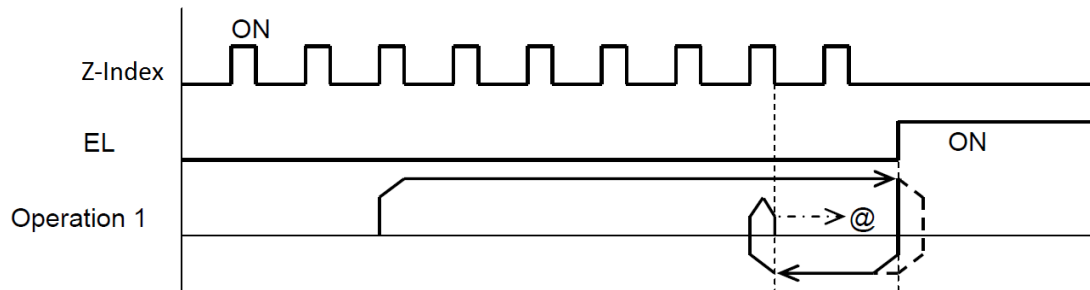
Homing Mode 12

Homing Mode 12

Homing mode 12 is similar to homing mode 11, except it utilizes the end of travel limit to initiate an immediate stop before it makes a fast move back to the Z-index pulse, where a quick index reverses motion back to the position where the Z-index pulse was encountered.

Homing Mode	Home Limit	End Limits	Z-index	Slow Down	Approaches Home at		Direction to Trigger Zero	Ending Point
					High Speed	Low Speed		

Mode 12		•	•	○	H		Dir of Home	Zero



Operation 1:

- A) Homing operation is started, accelerate from LSPD to HSPD.
- B) Immediate stop on encountering end of travel limit switch.
- C) Reverse direction, accelerating from LSPD to HSPD.
- D) Deceleration begins when specified number of Z-index-pulses is encountered (ZCNT), zero position is set.
- E) Start a zero-point return move to the position where the Z-index pulse was encountered, accelerating from LSPD to HSPD then decelerating and stopping at 0 position.
- F) Homing is completed when the 0 position is reached.
- G) ERC pulse is sent.

NOTE: For homing modes 9 – 12, encoder feedback is required. When using a stepper motor and homing with modes 9 – 12, an encoder is required and the command pulse count and encoder pulse count must be equal.

More information on the homing modes can be found in Section 9-5-1 of the PCL6045BL users manual.

HSPD – High Speed Set Point

HSPD – High Speed Set Point

Sets the high moving speed at the time of starting axis motion, or reads the current set value. The moving start speed can be set as a global value, or set for each axis.

Applies to: CMD-4CR, CMD-4EX-SA, CMD-2ED-SA, CMD-2EX

ASCII/Standalone	HSPD	<i>global high speed setting</i>
	HSPD[axis]	<i>individual axis high speed setting</i>
ASCII	HS	<i>global high speed setting</i>
	HS[axis]	<i>individual axis high speed setting</i>
Standalone	HSPD	<i>global high speed setting</i>
	HSPD[axis]	<i>individual axis high speed setting</i>

For all examples below, replace HSPD with HS for ASCII mode when working with a PMX-4EX-SA and PMX-4ET-SA.

Syntax:

Read:

HSPD[axis]

HS[axis]

** For CMD-4EX and CMD-4ET ASCII mode only*

Where: [axis] is set to X, Y, Z, U, or not specified to read global value

Write:

HSPD[axis]=[hspd]

HS[axis]=[hspd]

** For CMD-4EX and CMD-4ET ASCII mode only*

Where: [axis] is set to X, Y, Z, U, or not specified to write global value
 [hspd] is motion high speed (pps) with range of 1 – 6,553,500,
 * *[hspd] must be set higher than the LSPD value.*

Reply

Command is accepted and executed

Read:

Value of HSPD, range of 1 – 6,553,500

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read: HSPDX * *returns the value set for HSPD for the X axis*

Write: HSPDZ=50000 * *sets motion high speed for Z axis to 50,000 pps*

Notes:

- **HSPD** must be larger than 0.
- **HSPD** and **HSPD[axis]** must be a positive value. If a negative value is entered, the absolute value is used. Example entering HSPD=-1000 will set HSPD to 1000.
- **HSPD** and **HSPD[axis]** must be greater than or equal to **LSPD** and **LSPD[axis]**.
- Individual high-speed settings will take priority over the global high-speed settings. A global high-speed setting will only be used when either of the following conditions is true:
 1. The corresponding individual high-speed setting is defined as zero.
 2. An interpolated motion is being performed.

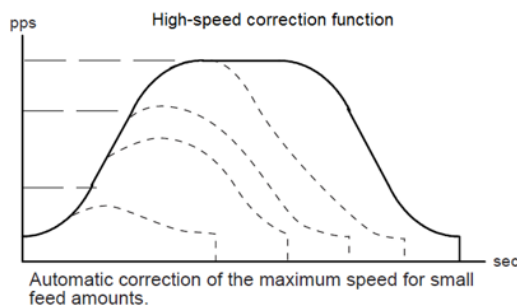
In the example below, the settings will set both the global speed settings as well as the individual speed setting for the X axis.

HSPD=10000
 HSPDX=2000
 LSPD=300
 ACCX=500
 ACC=300
 DEC=300

In this example, once a move command is issued, the global speed settings for the low speed and deceleration, as well as the individual X-axis speed settings for the high speed and acceleration, will be used for motion.

- StepNLoop
 - o When StepNLoop is enabled and not making an interpolated move, high speed units are encoder counts per second.

- o When StepNLoop is disabled or interpolated moves are commanded, the high speed units are pulse per second.
- The minimum and maximum acceleration/deceleration values depend on the high-speed and low-speed settings. Refer to [Speed Settings](#) for details.
- Only in V127 firmware for CMD-4CR and CMD-4EX, the **SSPDM[axis]** Speed Range setting must be set to accommodate the desired speed value, or set to 0 for automatic setting. See [SSPDM – Speed Range](#) for more information.
- Based on speed magnification resolution, actual speed can vary from commanded speed by up to 0.1%.
- Sometimes it is not possible to reach the high speed requested by the high speed command based on the acceleration, deceleration times and distance commanded. In the S-curve mode, the CMD-4CR and CMD-4EX will automatically calculate a new HSPD, which is only used for that move and allows for the quickest and smoothest possible move for the requested acceleration, deceleration times and distance commanded.



See also:

[Motion Settings](#)

[ACC](#), [DEC](#), [JKA](#), [JKD](#), [LSPD](#), [PS](#), [SCV](#), [SSPDM](#)

Commander Manual

[2.1.4 Motion Profile](#)

[Appendix B: Speed Settings](#)

I – Interpolation (X, Y, Z Linear Interpolation)

I – Interpolation (X, Y, Z Linear Interpolation)

This write-only command sets an XYZ interpolated move into the buffer. No speed designation is used with the standalone version, as the global HSPD value is used.

Applies to: [CMD-4CR](#), [CMD-4EX-SA](#)

ASCII	I
Standalone	XYZ

Syntax:

Write:

ASCII

I[posX]:[posY]:[posZ]:[speed]

the ":" character

I[posX]:[posY]

Each target position must be separated by

For the PMX-2EX-SA, PMX-2ED-SA

controllers, only position X and Y are used

Standalone

X[posX]Y[posY]Z[posZ]

There is no separation between each

target position in standalone mode

X[posX]Y[posY]

For the PMX-2EX-SA, PMX-2ED-SA

controllers, only position X and Y are used

Where: [posX] specifies target position for X-axis (28-bit value)
 [posY] specifies target position for Y-axis (28-bit value)
 [posZ] specifies target position for Z-axis (28-bit value)
 [speed] specifies buffered motion high speed (pps) with range of
 1 – 6,553,500.

Reply:

Command is accepted and executed

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Write:

ASCII

I2000:500:0:500

** Write a linear interpolation move to the buffer with X-axis*

target position 2000, Y-axis target position 500, Z-axis target position 0, operation speed 500

Standalone

X2000Y500Z0

** Write a linear interpolation move to the buffer with X-axis*

target position 2000, Y-axis target position 500, Z-axis target position 0, with operation using the global HSPD setting

Notes:

- For the CMD-4CR, CMD-4EX-SA controllers, these commands require that the Buffer mode is enabled first. Use other buffer commands to initiate buffer operation.
- If an axis is not to move by the command, in ABS mode, set the target as the current axis position, in INC mode, set the target for that axis as zero.

See also:

[Buffer Operation](#), [Linear Interpolation](#)

[BO](#), [BF](#), [BSTAT](#), [BSTART](#), [IACC](#)

Commander Manual

[2.3 Interpolation Operation](#)

[2.3.1 Linear interpolation](#)

[2.3.4 Buffered interpolation operation](#)

I – 4 Axis Vector (X, Y, Z, U Linear Interpolation)**I – 4 Axis Vector (X, Y, Z, U Linear Interpolation)**

This write-only command sets an XYZU interpolated move into the buffer.

Applies to: CMD-4CR, CMD-4EX-SA with V202 or newer firmware

ASCII I

Syntax:**Write:**

ASCII

I[posX]:[posY]:[posZ]:[posU]:[speed] *Each target position must be separated by the ":" character*

Where: [posX] specifies target position for X-axis (28-bit value)
 [posY] specifies target position for Y-axis (28-bit value)
 [posZ] specifies target position for Z-axis (28-bit value)
 [posU] specifies target position for U-axis (28-bit value)
 [speed] specifies buffered motion high speed (pps) with range of
 1 – 6,553,500.

Reply:

Command is accepted and executed

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:**Write:**

ASCII

I2000:500:0:100:500 * *Write a linear interpolation move to the buffer with X-axis target position 2000, Y-axis target position 500, Z-axis target position 0, U-axis target position 100, operation speed 500*

Notes:

- For the CMD-4CR, CMD-4EX-SA controllers, these commands require that the Buffer mode is enabled first. Use other buffer commands to initiate buffer operation.
- If an axis is not to move by the command, in ABS mode, set the target as the current axis position, in INC mode, set the target for that axis as zero.

See also:

[Buffer Operation](#), [Linear Interpolation](#)

[BO](#), [BF](#), [BSTAT](#), [BSTART](#), [IACC](#)

Commander Manual[2.3 Interpolation Operation](#)[2.3.1 Linear interpolation](#)[2.3.4 Buffered interpolation operation](#)**IACC – Automatic acceleration for buffered interpolation****IACC – Automatic acceleration for buffered interpolation**

Enables/disables the Interpolation (I) move acceleration (used with buffered I commands). On the PMX-4EX-SA and PMX-4ET-SA controllers, the buffered interpolation feature includes an automatic acceleration option. The IACC command is used to control this feature.

Applies to: PMX-4EX-SA, PMX-4ET-SA

ASCII IACC

Syntax:**Read:**

IACC

Write:

IACC=[0,1]

Where: 0 is to disable, and 1 is to enable

Reply:

Command is accepted and executed

Read:

Returns 0 when disabled, 1 when enabled

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:**Read:**

IACC

** Returns if the automatic I move acceleration is enabled or disabled*

Write:

IACC=1

** Enables the automatic I move acceleration*

Notes:

When IACC=0, acceleration and deceleration must be completed manually. To control the acceleration or deceleration manually, gradually increase or decrease the speed value for each interpolated move.

When IACC=1, acceleration and deceleration are processed automatically. In this case, the speed acceleration profile will be automatically generated between sequential buffered moves. The acceleration/deceleration value used for automatic acceleration control is found in the global acceleration value (accel).

See also:

[Buffer Operation](#)

[BO](#), [BF](#), [I](#), [BSTAT](#), [BSTART](#)

Commander Manual

[2.3.4 Buffered interpolation operation](#)

Commander Manual

[Identification Number](#)

ID – Identification Number

ID – Identification Number

If multiple devices are connected to a single host, then each device should have a unique identification number. This will allow the host to differentiate between multiple controllers. This is applicable to all communication types.

The default value for ID is 01. It is possible to assign any ID number to any device in the range [00-99] by using ID=[01-99]. To save a modified identification number to the flash memory of the CMD-4CR, use the **STORE** command. After a power cycle, the new identification number will be used.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII ID

Syntax:

Read:

ID

Write:

ID=[nn]

Where: [nn] is the new identification number value

** Valid values for n target value is 00 to 99*

Reply:

Command is accepted and executed

Read:

Returns unit model followed by last loaded ID. Example "CMD-4CR-01" where 01 is the ID.

Write:

OK

** Any change to the ID requires a power cycle before it takes*

effect.

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:

ID * *Request the current unit's identification number*

Write:

ID=25 * *Modifies the identification number to 25*

STORE * *Saves the modified identification number to the flash memory*

Notes:

Any changes to ID must be stored to flash memory using the **STORE** command. After a power cycle, the new identification number will be used. Before a power cycle is completed, the settings will not take effect, and any changes will not be returned when requested by the ID command.

See also:

[System Setup](#)

[VER](#), [DB](#), [IP](#)

Commander Manual

[2.1.1 Connect Multiple Commander Cores in a System](#)

IERR – Alarm/Limit Registration

IERR – Alarm/Limit Registration

This command enables or disables the registration of an alarm or limit error. While the command will not prevent an alarm or limit from stopping motion, it will not register as an error, and thus motion can continue without an error clear if the limit or alarm is no longer active.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII IERR

Syntax:

Read:

IERR

Write:

IERR=[enabled]

Where: [enabled] enables or disables the error registration function
0 = register

1 = do not register

Reply*Command is accepted and executed***Read:**

Returns 0 when error is registered, 1 when error is not registered

Write:

OK

*Command cannot be processed*An error message is returned. See [Error Codes](#)**Example:****Read:**IERR ** Returns 0 or 1 (registered or not registered)***Write:**IERR=1 ** Set the system to not register errors for alarms and limits***Notes:**

- Limits

By default IERR=0, which will trigger a limit error MST[axis](bit8~9) when a limit switch is set to active. Once the limit error is set, use the CLR[axis] command to clear the error in ASCII mode or the ECLEAR[axis] command in standalone mode.

When using the end limit during the homing modes, the ERC (deflection counter clear) signal can be set up to automatically clear the limit errors that accrue due to homing.

The limit error states can be ignored by setting IERR=1. In this case, the motor will still stop when the appropriate switch is triggered; however, it will not enter an error state.

- E-Stop

By default when IERR=0, any axis that is operating when the /CEMG signal goes low or the ESTOP command is sent will set an EMG error flag MST[axis](bit16) for that axis. Any axis with a set EMG error flag cannot operate until the EMG error flag has been cleared.

The EMG error flags are ignored by setting IERR=1. In this case, all axes will still stop when /CEMG is triggered or ESTOP command is sent; and all axes can operate again immediately after the /CEMG input is cleared.

See also:

[System Setup](#), [Limits/Safety Setup](#),
[ERC](#), [ERCD](#), [ERCP](#), [INP](#), [POL](#), [SDC](#), [SDE](#), [EXST](#), [STORE](#)

Commander Manual

[2.1.3.1.1.1 Limit Switches](#)

[2.1.3.1.1.2 Emergency stop](#)

IF/ELSEIF/ELSE/ENDIF – Conditional loop**IF/ELSEIF/ELSE/ENDIF – Conditional loop**

The commands described here are only available when writing a standalone program.

Conditionally executes a group of statements, depending on the value of an expression.

Applies to: CMD-4CR, CMD-4EX-SA

Standalone IF
 ELSEIF
 ELSE
 ENDIF

Syntax:**Write:**

```
IF [condition]
  [statement]
ELSEIF [elseifcondition]
  [elseifstatement]
ELSE
  [elsestatement]
ENDIF
```

Where: [condition] Required. Expression. Must evaluate to True or False using a [relational operator](#).

 [statement] One or more statements to be executed if [condition] is TRUE

 [elseifcondition] Required if ELSEIF is present Expression. Must evaluate to True or False using a [relational operator](#).

 [elseifstatement] One or more statements to be executed if [elseifcondition] is TRUE

 [elsestatement] One or more statements that are executed if no previous [condition] or [elseifcondition] expression evaluates to True.

Reply:

Command is accepted and executed

No Reply

Example:**Write:**

```
IF IO4=1      * If configurable IO 4 is on.
  X1000      * Move to 1000 (for X-axis)
  WAITX      * Wait for X-axis move to complete
ELSEIF IO5=1 * If configurable IO 5 is on
  X2000      * Move to 2000 (for X-axis)
  WAITX      * Wait for X-axis move to complete
ENDIF      * End of IF loop
```

Notes:

- None

See also:**Commander Manual**

[2.1.1.1 Identification Number](#)

INC – Incremental Coordinate Mode**INC – Incremental Coordinate Mode**

Write-only command that sets the position coordinate mode to incremental. Relative position will be used for motion commands.

Applies to: CMD-4CR, CMD-4EX-SA, CMD-2ED-SA, CMD-2EX

ASCII/Standalone INC

Syntax:

Write: INC

Reply:

Command is accepted and executed

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

INC ** Positioning coordinate mode is set to incremental*

Notes:

None

See also:

[Motion Settings](#), [Homing](#)
[ABS](#), [MM](#)

Commander Manual

[2.2.4 Positioning operation](#)

[2.2.4.1 Absolute or Incremental](#)

INP – Wait for In Position

This command enables or disables the use of the in-position signal, which delays the operation completion status until a driver inputs the in-position signal or positioning completion.

INP – Wait for In Position

Applies to: CMD-4CR, CMD-4EX-SA

ASCII INP[axis]

Syntax:

Read:

INP[axis]

Where: [axis] is set to X, Y, Z, U

Write:

INP[axis]=[enabled]

Where: [axis] is set to X, Y, Z, U

[enabled] enables or disables the In-Position function

0 = disabled

1 = enabled

Reply:

Command is accepted and executed

Read:

Returns 0 or 1 (disabled or enabled)

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:

INPY ** Returns the enabled status for the Y axis in-position function*

Write:

INPU=1 ** Enables the in-position function for the U axis.*

Notes:

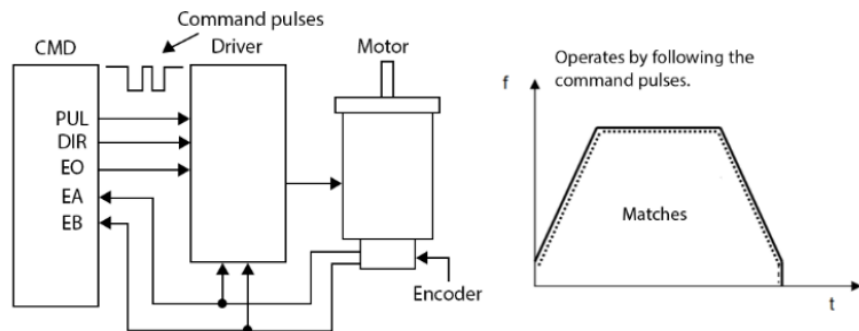
- The key advantage for INP is in servo systems that are operating in a positioning mode.
 - o For a stepper application there is no need to use the INP input.
 - o For servo, if the application is a scanning application where position is not critical,

then there is no need for INP.

- What is the INP input and how does it work?

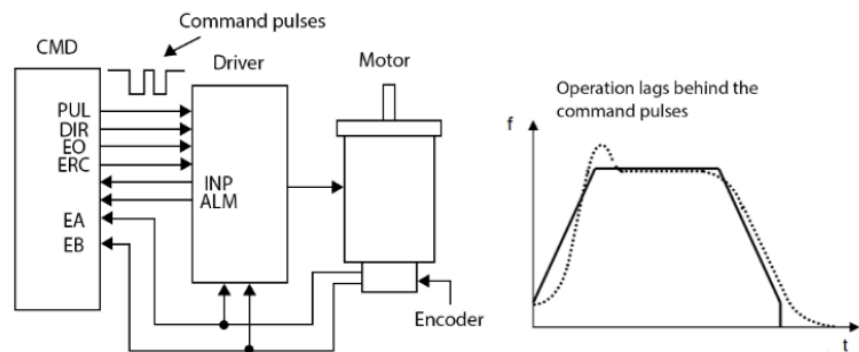
Looking at the picture to the right, notice the difference between the stepper and servo motion profiles. In a servo system there is a lag between where the commanded position is and where the motor actually is.

Stepper Motor – Closed Loop



When you make a motion, without the INP input on (INP[axis]=0) it will show the axis is finished moving as soon as the command pulses are finished being sent, and the next command can be accepted by the controller.

Servomotor – Closed Loop



When you make a

motion, with the INP input on (INP[axis]=1) it will show the axis is finished moving after the command pulses are finished being sent, and then the INP input is triggered active. (MST [axis] bit 11).

See also:

[Motor Setup](#)

[ERC](#), [ERCD](#), [ERCP](#), [POL](#), [SDC](#), [SDE](#), [EXST](#), [IERR](#)

Commander Manual

[2.1.2.2 Digital Servo](#)

IO – General-Purpose I/O Status

INP – Wait for In Position

Read/write command that returns/sets the input/output status of general-purpose I/O ports.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII/Standalone

IO

Read/write all I/O ports at once

IO[portNo]

Read/write individual I/O ports

Syntax:

Read:

IO[portNo]

Where: [portNo] specifies the port number (1-32); if not specified, then the status of all general-purpose I/O ports are returned.

Write:

IO[portNo]=[value]

Where: [portNo] specifies the port number (1-32),
[value] specifies status of output, 0 = off, 1 = on

[portNo] If not specified, then the status of all general-purpose I/O ports are set.

[value] specifies a decimal value representing a 32-bit number, which identifies port and status according to the chart below.

Input values are NOT changed due to this command

Reply:

Command is accepted and executed

Read:

Individual general-purpose I/O port (IO[portNo])

Returns the status of general-purpose I/O port [portNo], With a value of 0 or 1 (0 off, 1 on).

All general-purpose I/O ports (IO)

Returns status of all I/O based on each a 32-bit value based on the following:

Bit	Description	Status
0	General-purpose IO 1	0 Off 1 On
1	General-purpose IO 2	0 Off 1 On
2	General-purpose IO 3	0 Off 1 On
3	General-purpose IO 4	0 Off 1 On
4	General-purpose IO 5	0 Off 1 On
5	General-purpose IO 6	0 Off 1 On
6	General-purpose IO 7	0 Off 1 On
7	General-purpose IO 8	0 Off 1 On

Bit	Description	Status
16	General-purpose IO 17	0 Off 1 On
17	General-purpose IO 18	0 Off 1 On
18	General-purpose IO 19	0 Off 1 On
19	General-purpose IO 20	0 Off 1 On
20	General-purpose IO 21	0 Off 1 On
21	General-purpose IO 22	0 Off 1 On
22	General-purpose IO 23	0 Off 1 On
23	General-purpose IO 24	0 Off 1 On

8	General-purpose IO 9	0 Off On	1
9	General-purpose IO 10	0 Off On	1
10	General-purpose IO 11	0 Off On	1
11	General-purpose IO 12	0 Off On	1
12	General-purpose IO 13	0 Off On	1
13	General-purpose IO 14	0 Off On	1
14	General-purpose IO 15	0 Off On	1
15	General-purpose IO 16	0 Off On	1

2 4	General-purpose IO 25	0 Off On	1
2 5	General-purpose IO 26	0 Off On	1
2 6	General-purpose IO 27	0 Off On	1
2 7	General-purpose IO 28	0 Off On	1
2 8	General-purpose IO 29	0 Off On	1
3 9	General-purpose IO 30	0 Off On	1
3 0	General-purpose IO 31	0 Off On	1
3 1	General-purpose IO 32	0 Off On	1

Write:

Individual general-purpose I/O port (IO[portNo])
OK

All general-purpose I/O ports (IO)

Returns status of all I/O based on each a 32-bit value.

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:

IO ** Returns value for all general-purpose I/O*
Ex. Return a value of 54, or binary 0011 0110, IO2, IO3, IO5, and IO6 are on, all other general-purpose I/O is off.
IO8 ** Returns the status of digital input 8*

Write:

IO18=1 ** turns on general purpose IO18 if it is configured as an output*
IO=147161088 ** 147161088 in binary is 0000 1000 1100 0101 1000 0000 0000 0000; if setup as outputs, general-purpose I/O 16, 17, 19, 23, 24, and 28 are turned on, all other general-purpose I/O are turned off. Any general-purpose I/O setup as inputs are not affected.*

Notes:

- Reading

This command reads the status of I/O regardless of whether it is set up as an input or output.

V300+

- Writing

The IO command only changes general-purpose I/O setup as outputs. Input values are NOT changed due to this command. However the reply will contain the status of I/O regardless of whether it is set up as an input or output.

V1xx and V2xx

- Writing

The IO command only changes general-purpose I/O setup as outputs. Input values are NOT changed due to this command. The reply will echo the value written, a separate read will be needed to get the current status of I/O.

See also:

[I/O Setup](#),

[AI](#), [DI](#), [DIP](#), [DO](#), [DOBOOT](#), [DOP](#), [IOBOOT](#), [IOCFG](#), [IOP](#)

Commander Manual

[2.1.3.3.1 Configurable Digital I/O](#)

IOBOOT – General-Purpose I/O Status on Boot-up

IOBOOT – General-Purpose I/O Status on Boot-up

Read/write command that returns/sets the output status of general-purpose I/O ports on power up (boot-up).

Applies to: CMD-4CR, CMD-4EX-SA

ASCII IOBOOT

Syntax:

Read:

IOBOOT

Write:

IOBOOT=[value]

Where: [value] specifies a decimal value representing a 32-bit number identifying port and status according to the chart below. **Input values are NOT changed due to this command.**

Reply:

Command is accepted and executed

Read:

Returns status of all I/O based on each a 32-bit value, based on the following:

Bit	Description	Status

Bit	Description	Status

0	General-purpose IO 1	0 Off On	1
1	General-purpose IO 2	0 Off On	1
2	General-purpose IO 3	0 Off On	1
3	General-purpose IO 4	0 Off On	1
4	General-purpose IO 5	0 Off On	1
5	General-purpose IO 6	0 Off On	1
6	General-purpose IO 7	0 Off On	1
7	General-purpose IO 8	0 Off On	1
8	General-purpose IO 9	0 Off On	1
9	General-purpose IO 10	0 Off On	1
10	General-purpose IO 11	0 Off On	1
11	General-purpose IO 12	0 Off On	1
12	General-purpose IO 13	0 Off On	1
13	General-purpose IO 14	0 Off On	1
14	General-purpose IO 15	0 Off On	1
15	General-purpose IO 16	0 Off On	1

16	General-purpose IO 17	0 Off On	1
17	General-purpose IO 18	0 Off On	1
18	General-purpose IO 19	0 Off On	1
19	General-purpose IO 20	0 Off On	1
20	General-purpose IO 21	0 Off On	1
21	General-purpose IO 22	0 Off On	1
22	General-purpose IO 23	0 Off On	1
23	General-purpose IO 24	0 Off On	1
24	General-purpose IO 25	0 Off On	1
25	General-purpose IO 26	0 Off On	1
26	General-purpose IO 27	0 Off On	1
27	General-purpose IO 28	0 Off On	1
28	General-purpose IO 29	0 Off On	1
29	General-purpose IO 30	0 Off On	1
30	General-purpose IO 31	0 Off On	1
31	General-purpose IO 32	0 Off On	1

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:

IOBOOT

** Returns value for all general-purpose I/O*

Ex. Return a value of 54, or binary 0011 0110, indicates IO 2, 3, 5, and 6 will turn on at power-up if configured as outputs. All other outputs are off.

Write:

IOBOOT=147161088 ** 147161088 in binary is 0000 1000 1100 0101*

1000 0000 0000 0000, IO 16, 17, 19, 23, 24 and 28 turn on at power up if configured as outputs. all other outputs are off.

Notes:

- Only the General-purpose ports set as Outputs by the [IOCFG](#) command are effected by the IOBOOT command.
- Input values are NOT changed due to this command.
- To write the value to the device's flash memory, use the STORE command.

See also:

[I/O Setup](#)

[AI](#), [DI](#), [DIP](#), [DO](#), [DOBOOT](#), [DOP](#), [IO](#), [IOCFG](#), [IOP](#)

Commander Manual

[Configurable Digital I/O](#)

IOCFG – General-Purpose I/O Configuration

IOCFG – General-Purpose I/O Configuration

Read/Write command that returns/sets the input/output status of general-purpose I/O ports.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII IOCFG

Syntax:

Read:

IOCFG

Write:

IOCFG=[value]

Where: [value] specifies a decimal number representing a 32-bit value that identifies port and status according to the chart below.
The default value for all ports is 4,294,901,760.

Reply:

Command is accepted and executed

Read:

Returns status of all I/O based on each a 32-bit value based on the following

Bit	Port	Status	Default
0	Port 1	0 Input 1 Output	Input
1	Port 2	0 Input 1 Output	Input
2	Port 3	0 Input 1 Output	Input

Bit	Port	Status	Default
16	Port 17	0 Input 1 Output	Output
17	Port 18	0 Input 1 Output	Output
18	Port 19	0 Input 1 Output	Output

3	Port 4	0 Input	1 Output	Input
4	Port 5	0 Input	1 Output	Input
5	Port 6	0 Input	1 Output	Input
6	Port 7	0 Input	1 Output	Input
7	Port 8	0 Input	1 Output	Input
8	Port 9	0 Input	1 Output	Input
9	Port 10	0 Input	1 Output	Input
10	Port 11	0 Input	1 Output	Input
11	Port 12	0 Input	1 Output	Input
12	Port 13	0 Input	1 Output	Input
13	Port 14	0 Input	1 Output	Input
14	Port 15	0 Input	1 Output	Input
15	Port 16	0 Input	1 Output	Input

19	Port 20	0 Input	1 Output	Output
20	Port 21	0 Input	1 Output	Output
21	Port 22	0 Input	1 Output	Output
22	Port 23	0 Input	1 Output	Output
23	Port 24	0 Input	1 Output	Output
24	Port 25	0 Input	1 Output	Output
25	Port 26	0 Input	1 Output	Output
26	Port 27	0 Input	1 Output	Output
27	Port 28	0 Input	1 Output	Output
28	Port 29	0 Input	1 Output	Output
39	Port 30	0 Input	1 Output	Output
30	Port 31	0 Input	1 Output	Output
31	Port 32	0 Input	1 Output	Output

***NOTE: Default status is reference only, as the controller may have been set up in a different configuration**

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:

IOCFG

** Returns decimal value for all general-purpose I/O
Ex. Return a value of 54, or binary 0011 0110,
indicates port 2, 3, 5, and 6 are changed to outputs, all
other ports are at default settings.*

Write:

IO=265,256,960 ** 265,256,960 in binary is 0000 1111 1100 1111
1000 0000 0000 0000, Ports 16, 17, 18, 19, 20, 23, 24, 25, 26, 27, 28 are set as
outputs. All other ports are set as inputs.*

Notes:

- To write the value to the device's flash memory, use the STORE command.

See also:

[I/O Setup](#)

[AI](#), [DI](#), [DIP](#), [DO](#), [DOBOOT](#), [DOP](#), [IO](#), [IOBOOT](#), [IOP](#)

Commander Manual

[Configurable Digital I/O](#)

IOP – General-Purpose I/O Polarity**IOP – General-Purpose I/O Polarity**

Read/write command that toggles the polarity of general-purpose inputs and outputs.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII IOP

Syntax:**Read:**

IOP

Write:

IOP=[pol]

Where: [pol] is a 2-bit value to set polarity of the configurable I/O.
 Bit 0 is set to 0 for negative input logic (active low), 1 for positive input logic (active high)
 Bit 1 is set to 0 for negative output logic (active low), 1 for positive output logic (active high)

Reply:

Command is accepted and executed

Read:

Returns the value set (0-3, 00b-11b) see chart.

Bit	Description	Setting	
0	Logic of input setting	0 - Negative logic Active Low	1 - Positive logic Active High
1	Logic of output setting	0 - Negative logic Active Low	1 - Positive logic Active High

IOP command**Write:**

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:

IOP ** Returns value for general purpose I/O polarity setting*

Write:

IOP=1 ** Set input polarity to 1 positive logic, active high.*

IOP=3 ** Set input and output polarity to 1 positive logic, active high.*

Notes:

- To write the value to the device's flash memory, use the STORE command.

See also:

[I/O Setup](#),

[AI](#), [DI](#), [DIP](#), [DO](#), [DOBOOT](#), [DOP](#), [IO](#), [IOBOOT](#), [IOCFG](#)

Commander Manual

[Configurable Digital I/O](#)

IP – IP Address

IP – IP Address

Sets the deceleration time or acquires the current set value. Deceleration time is available as a global value for all axes and a value settable for each axis

Applies to: CMD-4CR, CMD-4EX-SA, CMD-2ED-SA, CMD-2EX

ASCII/Standalone decel *global acceleration setting*

 decel[axis] *individual axis acceleration setting*

Syntax:

Read:

decel[axis]

Where: [axis] is set to X, Y, Z, U, or not specified to read global value

Write:

decel[axis]=[accel]

Where: [axis] is set to X, Y, Z, U, or not specified to read global value
[accel] is the deceleration time in ms (0 – 65,535)

Reply:

Command is accepted and executed

Read:

Returns the value of the [axis] deviation counter (28-bit value)

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:

DECY ** Returns deceleration time set for the Y axis in ms*

Write:

decel=300 ** Sets deceleration time to 300 ms for all axes.*

Notes:

- This command is not fully working yet Leave as "OUT of DATE" status

See also:

[System Setup](#)

[VER, ID, DB](#)

Commander Manual

[Identification Number](#)

J – JOG

J – JOG

Write-only command that initiates a jog of the specified axis in the indicated direction.

Applies to: CMD-4CR, CMD-4EX-SA, CMD-2ED-SA, CMD-2EX

ASCII	J[axis][dir]
Standalone	JOG[axis][dir]

Syntax:

Write:

J[axis][dir] ** ASCII*
 JOG[axis][dir] ** Standalone*

Where: [axis] specifies the axis to be jogged (X, Y, Z, U) [dir] specifies the direction (+/-) (+) command pulse value counts up (-) command pulse value counts down

Reply:

Command is accepted and executed

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Write:

JX+ * *Jogs the X axis at HSPD in the positive direction until a STOP command is initiated.*

Notes:

- Jog will accelerate based on the SCV command (Linear or S Curve acceleration) from LSPD to HSPD.

See also:

[Axis Motion](#)

[X, Y, Z, U, EO, EOBOOT](#)

Commander Manual

[2.2.2 Jog operation](#)

[2.2.4 Positioning operation](#)

JDEL – Joystick speed change delta

JDEL – Joystick speed change delta

Sets the maximum rate of change of the speed when using joystick operation. This limits the rate of change of speed between cycle updates.

Applies to: CMD-4CR, CMD-4EX-SA, CMD-2ED-SA, CMD-2EX-SA

ASCII

JDEL

Syntax:

Read:

JDEL[axis]

Where: [axis] is set to X or Y

Write:

JDEL[axis]=[delta]

Where: [axis] is set to X or Y

[delta] specifies the maximum speed delta. Range from 1 – 2,000,000 in pps

Reply:

Command is accepted and executed

Read:

Returns delta value in pps

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:

JDELX ** Returns the delta speed value in pps for the X axis*

Write:

JDELX=500 ** Sets the maximum delta value to 500 pps for the X axis.*

Notes:

- None

See also:

[Motion Settings](#)

[JSPD](#), [JMAX](#), [JMIN](#), [JTOL](#), [JENA](#), [JLIM](#), [AI](#)

Commander Manual

[Identification Number](#)

JENA – Joystick enable

JENA – Joystick enable

Enables/disables joystick operation. The joystick enable parameter is a 2-bit value. For example, a joystick enable value of 3 means the joystick feature is enabled on both the X and Y axes.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII	JENA
Standalone	JOYENA

Syntax:

Read:

JENA	<i>* ASCII</i>
JOYENA	<i>* Standalone</i>

Write:

JENA=[enabled]	<i>* ASCII</i>
JOYENA=[enabled]	<i>* Standalone</i>

Where: [enabled] specifies the enable mode based on:

- 0 = Both X-axis and Y-axis are disabled
- 1 = Only X-axis enabled
- 2 = Only Y-axis enabled

3 = Both X-axis and Y-axis are enabled

Reply:*Command is accepted and executed***Read:**

Returns the [enabled] value. Decimal equivalent of two-bit binary number.

Bit	Description	Setting	
0	Joystick control X-axis	0 – Disabled	1 – Enabled
1	Joystick control Y-axis	0 – Disabled	1 – Enabled

Write:

OK

*Command cannot be processed*An error message is returned. See [Error Codes](#)**Example:****Read:**JENA ** Returns the value of joystick enabled status***Write:**JENA=3 ** Enables both X and Y axes.***Notes:**

- Do not enable joystick operation while the analog input port connection is opened. If you activate the joystick motion in the open status, the detected voltage will deviate from the invalid range, and the axis may start unexpected motions.
- When the power is turned on, the command pulse counter is "0", but it may be in a position different from the reference position of the soft limit value. If you operate the joystick in this status, you may hit a mechanical limit before the software limit works. When operating with a joystick, set the reference position "0" in the Homing operation beforehand.

See also:[JOYSTICK](#)[JSPD](#), [JDEL](#), [JMAX](#), [JMIN](#), [JTOL](#), [JLIM](#), [AI](#)**Commander Manual**[Identification Number](#)**JKA – Jerk Control Acceleration****JKA – Jerk Control Acceleration**

The JKA[axis] command for acceleration sets the ratio between S-Curve and linear acceleration. The percentage of S-curve the profile contains can be set, with a range from 0 to 100. A setting of 0 is the same as a complete linear profile, and a setting of 100 is the same as a complete S-curve profile.

Applies to: CMD-4CR, CMD-4EX-SA, CMD-2ED-SA, CMD-2EX-SA

ASCII

JKA

Syntax:

Read:

JKA[axis]

Where: [axis] is set to X, Y, Z, U

Write:

JKA[axis]=[percent]

Where: [axis] is set to X, Y, Z, U

[percent] is the percentage of the S-curve the profile contains
(0 - 100)

Reply:

Command is accepted and executed

Read:

Returns the percentage of the S-curve the profile contains

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:

JKAY

** Returns the percentage of the S-curve the Y axis profile acceleration contains.*

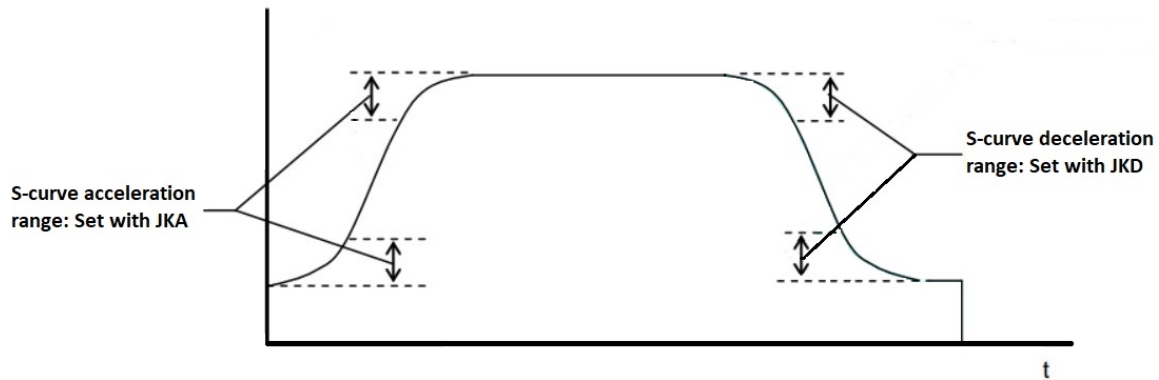
Write:

JKAX=75

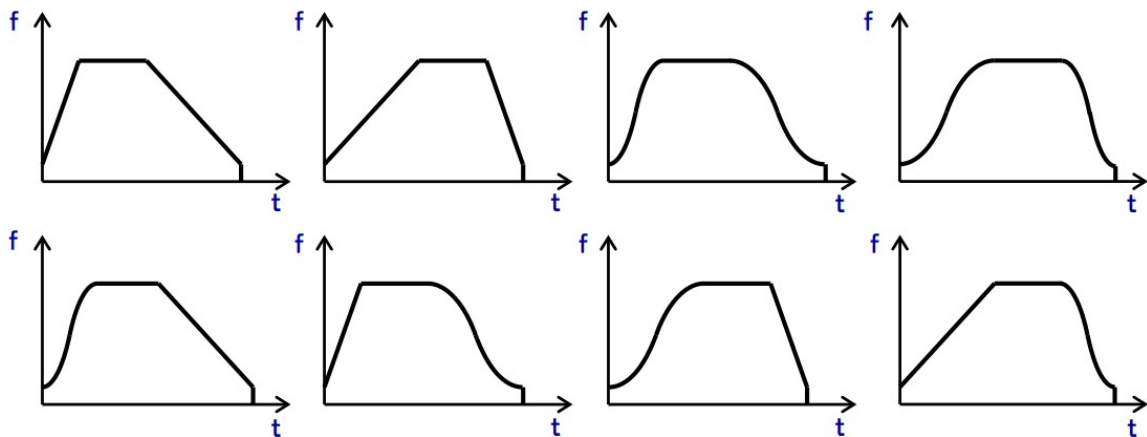
** Sets the X axis profile acceleration to 75% S-curve.*

Notes:

- When S-curve is disabled, linear acceleration/deceleration is used.
- When S-curve is enabled, changing jerk acceleration ([JKA](#)) and jerk deceleration ([JKD](#)) allows for fine-tuning of the S-curve profile.



- It is possible to set mixed acceleration/deceleration types in a single profile. Changing the jerk acceleration (JKA) or jerk deceleration (JKD) to 0 makes the acceleration/deceleration linear.



See also:

[Motion Settings](#)

[ACC](#), [DEC](#), [HSPD](#), [JKD](#), [LSPD](#), [PS](#), [SCV](#), [SSPDM](#)

Commander Manual

[2.1.4 Motion Profile](#)

JKD – Jerk Control Deceleration

JKD – Jerk Control Deceleration

The JKD[axis] command for deceleration sets the ratio between S-Curve and linear deceleration. The percentage of S-curve the profile contains can be set, with a range from 0 to 100. A setting of 0 is the same as a complete linear profile, and a setting of 100 is the same as a complete S-curve profile.

Applies to: CMD-4CR, CMD-4EX-SA, CMD-2ED-SA, CMD-2EX-SA

ASCII

JKD

Syntax:

Read:

JKA[axis]

Where: [axis] is set to X, Y, Z, U

Write:

JKA[axis]=[percent]

Where: [axis] is set to X, Y, Z, U

[percent] is the percentage of the S-curve the profile contains
(0 - 100)

Reply:

Command is accepted and executed

Read:

Returns the percentage of the S-curve the profile contains.

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:

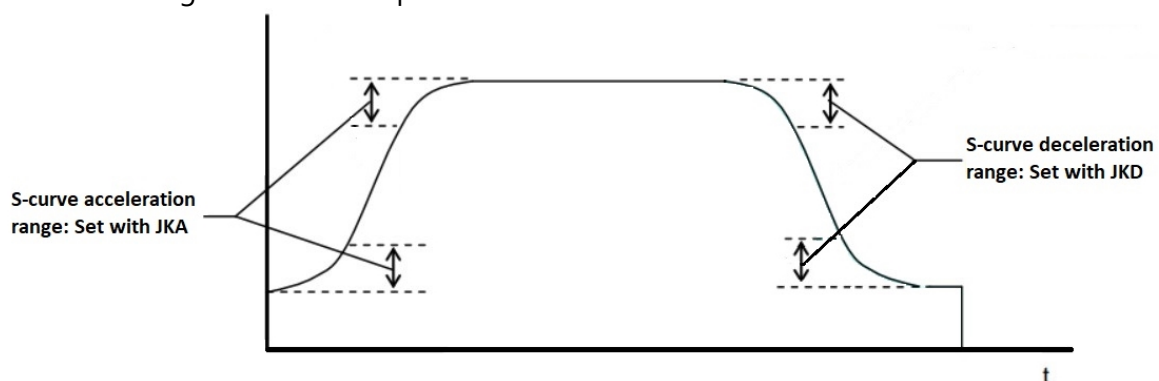
JKDY * Returns the percentage of the S-curve the Y axis profile deceleration contains.

Write:

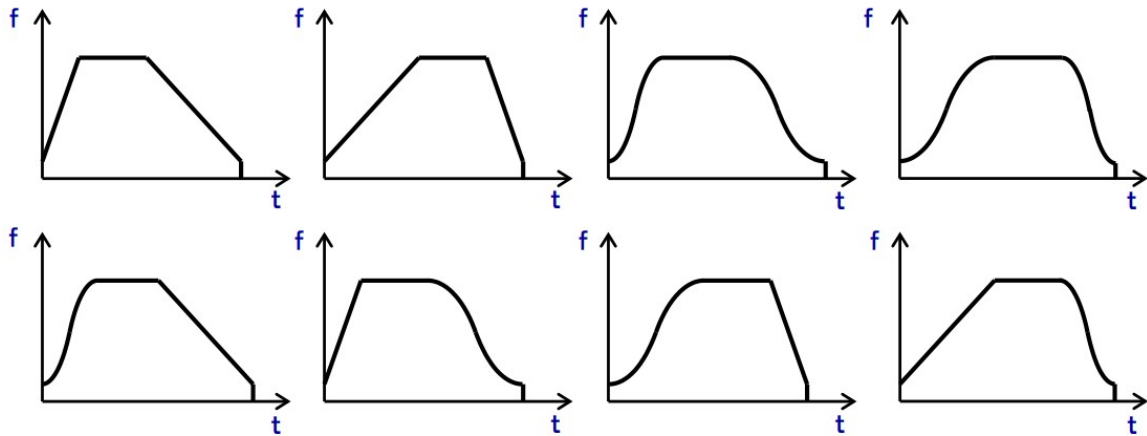
JKDX=0 * Sets the X axis profile deceleration to 0% S-curve (linear)

Notes:

- When S-curve is disabled, linear acceleration/deceleration is used.
- When S-curve is enabled, changing jerk acceleration ([JKA](#)) and jerk deceleration ([JKD](#)) allows for fine-tuning of the S-curve profile.



- It is possible to set mixed acceleration/deceleration types in a single profile. Changing the jerk acceleration (JKA) or jerk deceleration (JKD) to 0 makes the acceleration/deceleration linear.



See also:

[Motion Settings](#)

[ACC](#), [DEC](#), [HSPD](#), [JKA](#), [LSPD](#), [PS](#), [SCV](#), [SSPDM](#)

Commander Manual

[2.1.4 Motion Profile](#)

JLIM – Joystick Limits

JLIM – Joystick Limits

The JLIM command sets the operable range of the Joystick and how it responds. It also returns the current value of the limit setting.

Applies to: CMD-4CR, CMD-4EX-SA, CMD-2ED-SA, CMD-2EX

ASCII

JLIM

Syntax:

Read:

JLIM[number]

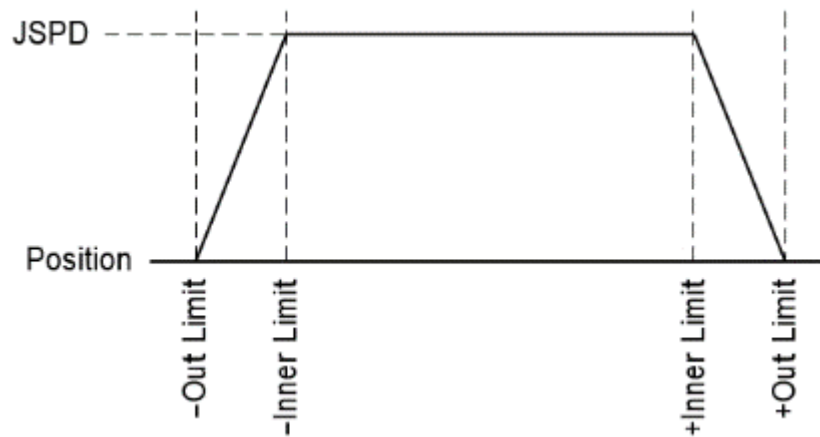
Where: [number] is set based on desired configuration mode from 1 - 8
(see chart below)

Write:

JLIM[number]=[value]

Where: [number] is set based on desired configuration mode from 1 - 8
(see chart below)

[value] specifies the limit value 32-bit range



Command	Description
JLIM1	X-axis negative outer limit
JLIM2	X-axis negative inner limit
JLIM3	X-axis positive inner limit
JLIM4	X-axis positive outer limit
JLIM5	Y-axis negative outer limit
JLIM6	Y-axis negative inner limit
JLIM7	Y-axis positive inner limit
JLIM8	Y-axis positive outer limit

Reply:

Command is accepted and executed

Read:

Returns parameter [value]

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:

JLIM4

** Returns the positive limit value of the X axis*

Write:

JLIM1=-1000000
1000000.

** Sets the negative side limit value of the X axis to -*

Notes:

- None

See also:[Motion Settings](#)[JSPD](#), [JDEL](#), [JMAX](#), [JMIN](#), [JTOL](#), [JENA](#), [AI](#)**Commander Manual**[Identification Number](#)**JMAX – Joystick Maximum Voltage****JMAX – Joystick Maximum Voltage**

The **JMAX[axis]** command can be used to define the maximum analog input value for the specified axis. This parameter has units of millivolts (mV) and a range of 0 to 3300. This value will define the analog input value in which the maximum joystick speed occurs in the positive direction. This command is only available in ASCII mode.

Applies to: CMD-4CR, CMD-4EX-SA, CMD-2ED-SA, CMD-2EX-SA

ASCII

JMAX

Syntax:**Read:**

JMAX[axis]

Where: [axis] is set to X or Y

Write:

JMAX[axis]=[volt]

Where: [axis] is set to X or Y

[volt] specifies the maximum voltage to the analog terminal in mV. Range from 0 – 3300

Reply:*Command is accepted and executed***Read:**

Returns the current set maximum voltage in mV

Write:

OK

*Command cannot be processed*An error message is returned. See [Error Codes](#)**Example:****Read:**

JMAXX

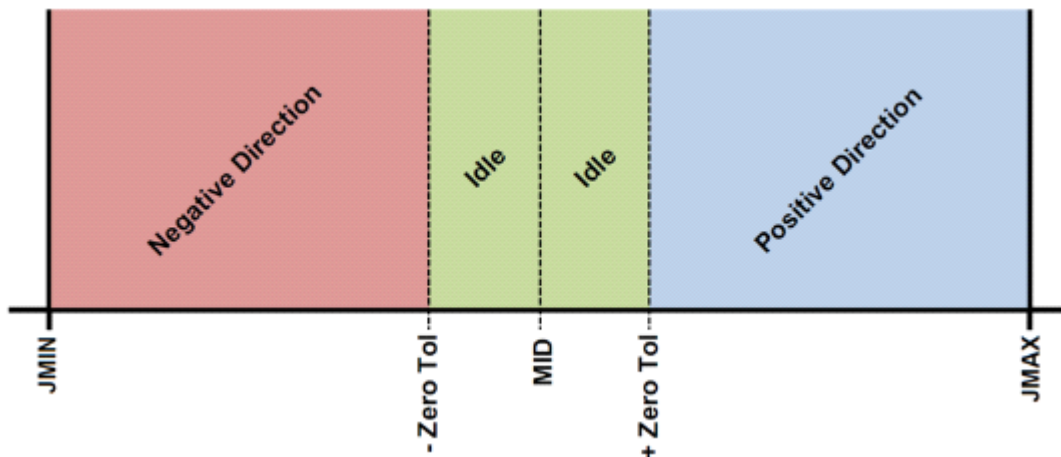
** Returns the value of the maximum voltage on the X-axis analog terminal.*

Write:

JMAXY=3300 * *Sets the maximum voltage value to 3300 mV for the Y-axis analog terminal.*

Notes:

- The ASCII command JTOL[axis] can be used to define the zero-tolerance (or dead) zone around the MID value. No movement will occur while the analog input value is within the zero-tolerance zone. This parameter has units of millivolts (mV) and a range of 0 to 3300. The zero-tolerance parameter will be on the positive and negative side of the MID point to define to the zero-tolerance zone. See the figure below for details.



- Example: The zero-tolerance zone at JMAX = 3,250 mV, JMIN = 10 mV, JTOL = 50 mV will be

$$MID = ((JMAX[axis] - JMIN[axis]) / 2) + JMIN[axis]$$

- 1580 mV ~ 1680 mV since $((3,250 - 10) \div 2) + 10 = 1,630$ mV
- 1,630 mV $\pm$ 50 mV

See also:

[Motion Settings](#)

[JSPD](#), [JDEL](#), [JMIN](#), [JTOL](#), [JENA](#), [JLIM](#), [AI](#)

Commander Manual

[Identification Number](#)

JMIN – Joystick Minimum Voltage**JMIN – Joystick Minimum Voltage**

The **JMIN[axis]** command can be used to define the minimum analog input value for the specified axis. This parameter has units of millivolts (mV) and a range of 0 to 3300. This value will define the analog input value in which the maximum joystick speed occurs in the negative direction. This command is only available in ASCII mode.

Applies to: CMD-4CR, CMD-4EX-SA, CMD-2ED-SA, CMD-2EX-SA

ASCII

JMIN

Syntax:**Read:**

JMIN[axis]

Where: [axis] is set to X or Y

Write:

JMIN[axis]=[volt]

Where: [axis] is set to X or Y

[volt] specifies the minimum voltage to the analog terminal in mV. Range from 0 – 3300

Reply:*Command is accepted and executed***Read:**

Returns the current set minimum voltage in mV

Write:

OK

*Command cannot be processed*An error message is returned. See [Error Codes](#)**Example:****Read:**

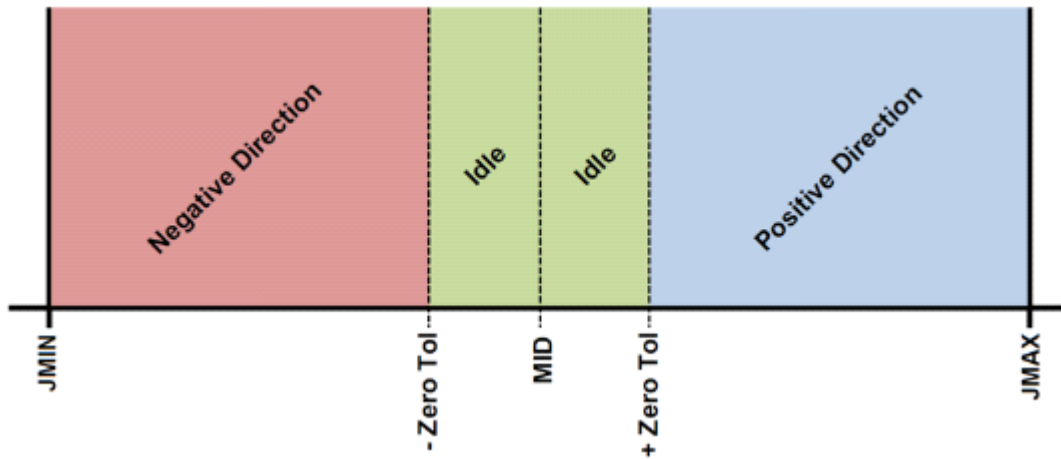
JMINX

*analog terminal.*** Returns the value of the minimum voltage on the X-axis***Write:**

JMINY=5

*analog terminal.*** Sets the minimum voltage value to 5 mV for the Y-axis***Notes:**

- The ASCII command JTOL[axis] can be used to define the zero-tolerance (or dead) zone around the MID value. No movement will occur while the analog input value is within the zero-tolerance zone. This parameter has units of millivolts (mV) and a range of 0 to 3300. The zero-tolerance parameter will be on the positive and negative side of the MID point to define the zero-tolerance zone. See the figure below for details.



- Example: The zero-tolerance zone at JMAX = 3,250 mV, JMIN = 10 mV, JTOL = 50 mV will be

$$MID = ((JMAX[axis] - JMIN[axis]) / 2) + JMIN[axis]$$

- 1580 mV ~ 1680 mV since $((3,250 - 10) \div 2) + 10 = 1,630$ mV
- 1,630 mV $\pm$ 50 mV

See also:

[Motion Settings](#)

[JSPD](#), [JDEL](#), [JMAX](#), [JTOL](#), [JENA](#), [JLIM](#), [AI](#)

Commander Manual

[Identification Number](#)

JSPD – Joystick Max Speed

JSPD – Joystick Max Speed

Sets the maximum operating speed for joystick operations.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII

JSPD

Syntax:

Read:

JSPD[axis]

Where: [axis] is set to X or Y

Write:

JSPD[axis]=[speed]

Where: [axis] is set to X or Y

[speed] specifies the maximum operating speed in pps. Range from 1 – 2,000,000

Reply:

Command is accepted and executed

Read:

Returns the speed value in pps

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:

JSPDX
X-axis.

** Returns the value of the maximum speed setting of*

Write:

JSPDX=50000 ** Sets the maximum speed of X-axis to 50,000 pps.*

Notes:

- None

See also:

[Motion Settings](#)

[JDEL](#), [JMAX](#), [JMIN](#), [JTOL](#), [JENA](#), [JLIM](#), [AI](#)

Commander Manual

[Identification Number](#)

JTOL – Joystick Deadzone

JTOL – Joystick Deadzone

Sets the invalid range voltage value of the analog input voltage terminal (dead zone) for joystick operations. The dead zone is the voltage range where the joystick operation is stopped. The range is determined by adding or subtracting JTOL from the midpoint (Mid) of the joystick voltage range.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII

JTOL

Syntax:

Read:

JTOL[axis]

Where: [axis] is set to X or Y

Write:

JTOL[axis]=[volt]

Where: [axis] is set to X or Y
 [volt] specifies the invalid voltage range above and below midpoint in mV

Reply:

Command is accepted and executed

Read:

Returns the current set midpoint voltage in mV

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:

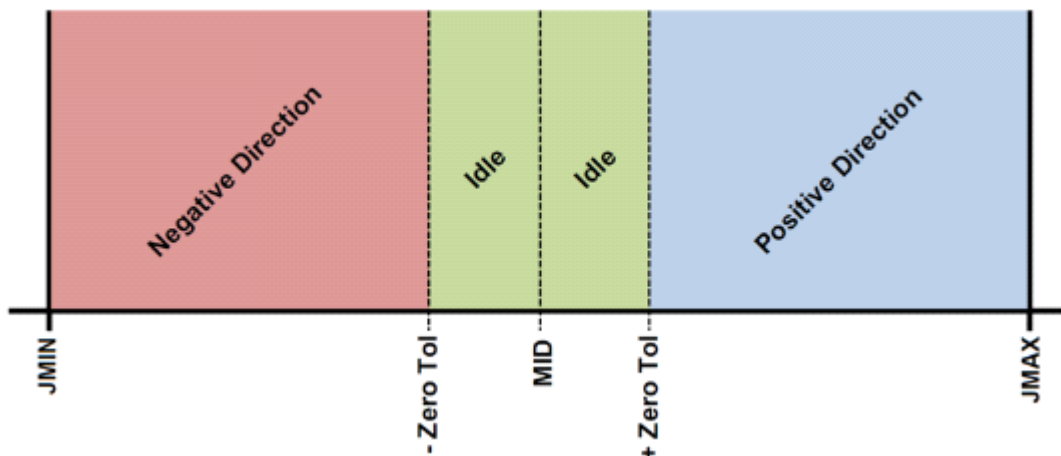
JTOLX ** Returns the set dead zone voltage tolerance for X axis*

Write:

JTOLY=50 ** Sets the dead zone tolerance to 50 mV for the Y axis.*

Notes:

- The ASCII command JTOL[axis] can be used to define the zero-tolerance (or dead) zone around the MID value. No movement will occur while the analog input value is within the zero-tolerance zone. This parameter has units of millivolts (mV) and a range of 0 to 3300. The zero-tolerance parameter will be on the positive and negative side of the MID point to define to the zero-tolerance zone. See the figure below for details.



- Example: The zero-tolerance zone at JMAX = 3,250 mV, JMIN = 10 mV, JTOL = 50 mV will be

$$MID = ((JMAX[axis] - JMIN[axis]) / 2) + JMIN[axis]$$

- 1580 mV ~ 1680 mV since $((3,250 - 10) \div 2) + 10 = 1,630$ mV
- 1,630 mV $\pm$ 50 mV

See also:[Motion Settings](#)[JSPD](#), [JDEL](#), [JMAX](#), [JMIN](#), [JENA](#), [JLIM](#), [AI](#)**Commander Manual**[Identification Number](#)**LSPD – Low Speed Set Point****LSPD – Low Speed Set Point**

Sets the moving start speed (unit = pps) at the time of starting axis motion, or reads the current set value. The moving start speed can be set as a global value, or set for each axis.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII/Standalone	LSPD	<i>global low speed setting</i>
	LSPD[axis]	<i>individual axis low speed setting</i>

For all examples below replace LSPD with LS for ASCII mode when working with a PMX-4EX-SA and PMX-4ET-SA.

Syntax:**Read:**

LSPD[axis]

LS[axis]

** For PMX-4EX, and PMX-4ET ASCII mode only*

Where: [axis] is set to X, Y, Z, U, or not specified to read global value

Write:

LSPD[axis]=[lspd]

LS[axis]=[lspd]

** For PMX-4EX, and PMX-4ET ASCII mode only*

Where: [axis] is set to X, Y, Z, U, or not specified to read global value

[lspd] is motion low speed (pps) with range of 1 – 6,000,000,

<i>* [lspd] must be set lower than the HSPD value.</i>

Reply*Command is accepted and executed***Read:**

Value of LSPD, range of 1 – 6,553,500

Write:

OK

*Command cannot be processed*An error message is returned. See [Error Codes](#)

Example:

Read: LSPD ** returns the global value set for LSPD*

Write: LSPDZ=500 ** sets motion low speed for Z axis to 500 pps*

Notes:

- **LSPD** and **LSPD[axis]** must be a positive value. If a negative value is entered, the absolute value is used. Example entering LSPD=-300 will set LSPD to 300.
- **LSPD** and **LSPD[axis]** must be less than or equal to **HSPD** and **HSPD[axis]**.
- Individual low-speed settings will take priority over the global low-speed settings. A global low-speed setting will only be used when either one of the following conditions is true:
 1. The corresponding individual low-speed setting is defined as zero.
 2. An interpolated motion is being performed.

In the example below, both the global speed settings and the individual speed setting for the X axis will be set.

```
HSPD=10000
HSPDX=2000
LSPD=300
LSPDX=200
ACCX=500
ACC=300
DEC=300
```

In this example, once a move command is issued the global speed settings for the low speed and deceleration, and the individual X-axis speed settings for the high speed, low speed and acceleration, will be used for motion.

- StepNLoop
 - o When StepNLoop is enabled and not making an interpolated move, low speed units are encoder counts per second.
 - o When StepNLoop is disabled or interpolated moves are commanded, the low speed units are pulse per second.
- The minimum and maximum acceleration/deceleration values depend on the high-speed and low-speed settings. Refer to [Speed Settings](#) for details.
- Based on speed magnification resolution, actual speed can vary from commanded speed by up to 0.1%.

See also:

[Motion Settings](#)

[ACC](#), [DEC](#), [HSPD](#), [JKA](#), [JKD](#), [PS](#), [SCV](#), [SSPDM](#)

Commander Manual

[2.1.4 Motion Profile](#)

[Appendix B: Speed Settings](#)

LT – Latch Enable

LT – Latch Enable

Sets the Latch enable/disable for [axis]

Applies to: CMD-4CR, CMD-4EX-SA

ASCII/Standalone LT

Syntax:**Write:**

LT[axis]=[enable]

Where: [axis] is set to X, Y, Z, U

[enable] enables or disables the latching function for [axis]

0: disabled

1: enabled

Reply:

Command is accepted and executed

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:**Write:**

LTX=1 * *Enables the latch function for X axis.*

Notes:

- The [LTS\[axis\]](#) is read to see the current enable status.

See also:

[I/O Setup](#),

[LTE](#), [LTP](#), [LTS](#)

Commander Manual:

[2.1.3.2.1 High-Speed Digital Inputs](#)

LTE – Latched Encoder Position

LTE – Latched Encoder Position

Reads the previously latched encoder position for [axis].

Applies to: CMD-4CR, CMD-4EX-SA

ASCII/Standalone LTE

Syntax:

Read:

LTE[axis]

Where: [axis] is set to X, Y, Z, U

Reply:

Command is accepted and executed

Read:

Returns the previous latched encoder position of [axis].

Example:

Read:

LTEX

** Returns latched position of encoder on X axis.*

Notes:

- None

See also:

[I/O Setup](#),
[LT](#), [LTP](#), [LTS](#)

Commander Manual:

[2.1.3.2.1 High-Speed Digital Inputs](#)

LTP – Latched Step Position

LTP – Latched Step Position

Reads the latched step position of [axis].

Applies to: CMD-4CR, CMD-4EX-SA

ASCII/Standalone

LTP

Syntax:

Read:

LTP[axis]

Where: [axis] is set to X, Y, Z, U

Reply:

Command is accepted and executed

Read:

Returns the latched step position for [axis]

Example:

Read:

LTPY

** Returns latched step position for Y axis.*

Notes:

- None

See also:

[I/O Setup](#),
[LT](#), [LTE](#), [LTS](#)

Commander Manual:

[2.1.3.2.1 High-Speed Digital Inputs](#)

LTS – Latch Function Status**LTS – Latch Function Status**

Reads the latch function enable status.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII/Standalone

LTS

Syntax:**Read:**

LTS[axis]

Where: [axis] is set to X, Y, Z, U

Reply:

Command is accepted and executed

Read:

Returns the latch enable status for [axis] (0 - 2)

Value	Condition
0	Latch off
1	Latch on and waiting for latch trigger
2	Latch triggered

Example:**Read:**

LTSY

** Reads latch enable status for Y axis.*

Notes:

- None

See also:

[I/O Setup](#),
[LT](#), [LTE](#), [LTP](#)

Commander Manual:

[2.1.3.2.1 High-Speed Digital Inputs](#)

MM – Move Mode

MM – Move Mode

Acquire the current move mode setting (Absolute or Incremental).

Applies to: CMD-4CR, CMD-4EX-SA

ASCII MM

Syntax:

Read: MM

Reply:

Read: Returns 0 if set to absolute mode, or 1 if set to incremental mode

Example:

Read:
MM * Positioning coordinate mode is returned
(0 – Absolute, 1 – Incremental)

Notes:

None

See also:

[Motion Settings](#)
[INC](#), [ABS](#)

Commander Manual

[2.2.4.1 Absolute or Incremental](#)

MP – Manual Pulse Generator Counter

MP – Manual Pulse Generator Counter

Read/Write command that returns/sets the counter value for the manual pulse generator in the specified axis.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII MP

Syntax:

Read:

MP[axis]

Where: [axis] is set to X, Y, Z, U

Write:

MP[axis]=[value]

Where: [axis] is set to X, Y, Z, U

[value] Specifies the counter value (28-bit value)

Reply:

Command is accepted and executed

Read:

Returns the manual pulse generator count values for the specified axis.
(28-bit value)

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:

MPX
X axis

** Returns the current manual pulse generator counts for the*

Write:

MPX=2000
2000.

** Set the manual pulse generator counter for the X axis to*

Notes:

- None.

See also:

[Motion Settings](#)
[MPD](#), [MPE](#), [MPM](#)

Commander Manual

[2.5.2 Manual pulse generator \(MPG\) operation](#)

MPD – Manual Pulse Generator Divider

MPD – Manual Pulse Generator Divider

Read/Write command that returns/sets the division ration of an input signal by a manual pulse generator.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII MPD
Standalone MPGD

Syntax:

Read:

MPD[axis]

Where: [axis] is set to X, Y, Z, U

Write:

MPD[axis]=[mpd]

Where: [axis] is set to X, Y, Z, U

[mpd] specifies the numerator in a ratio to determine the manual pulse generator count ratio (values 1 – 2048). Resulting ration is: Ratio = [mpd]/2048

Reply:

Command is accepted and executed

Read:

Returns the value of the [axis] deviation counter (28-bit value)

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

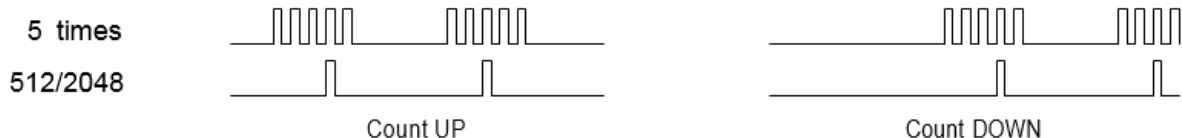
Read:

MPDX * Returns the current manual pulse generator counts for the X axis.

Write:

MPDX=512 * Set the manual pulse generator division ration to 512/2048 = 1/4.

Divide (in the case of MPD = 512) ····· 512/2048 = 1/4



Notes:

- The **MPD[axis]** can be set to a value from [1 to 2048]. The denominator of the divider circuit is fixed as 2048. A value of 2048 will turn the divider off.

See also:

[Motion Settings](#)

[MPE](#), [MPM](#), [MP](#)

Commander Manual

[2.5.2 Manual pulse generator \(MPG\) operation](#)

MPE – Manual Pulse Generator Enable

MPE – Manual Pulse Generator Enable

Read/Write command that enables/disables the manual pulse generator operation.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII

MPE

Syntax:

Read:

MPE[axis]

Where: [axis] is set to X, Y, Z, U

Write:

MPE[axis]=[0 or 1]

Where: [axis] is set to X, Y, Z, U

[0 or 1] enables (1) or disables (0) the operation

Reply:

Command is accepted and executed

Read:

Returns the enable/disable status of the manual pulse generator for the specified axis.

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:

MPEX

** Returns the enable/disable status of the manual pulse generator for the X axis*

Write:

MPEZ=0

** Disables the manual pulse generator for the Z-axis.*

Notes:

- None

See also:

[Motion Settings](#)

[MPD](#), [MPM](#), [MP](#)

Commander Manual

[2.5.2 Manual pulse generator \(MPG\) operation](#)

MPM – Manual Pulse Generator Multiplier**MPM – Manual Pulse Generator Multiplier**

Read/Write command that returns/sets the multiplication factor of the manual pulse generator input signal.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII	MPM[axis]
Standalone	MPGM

Syntax:**Read:**

MPM[axis]

Where: [axis] is set to X, Y, Z, U

Write:

MPM[axis]=[mpm]

Where: [axis] is set to X, Y, Z, U

[mpm] represents the multiplication factor (1 – 32)

Reply:

Command is accepted and executed

Read:

Returns the multiplication factor of the manual pulse generator for the specified axis.

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:**Read:**

MPMY

** Returns the multiplication factor of the manual pulse*

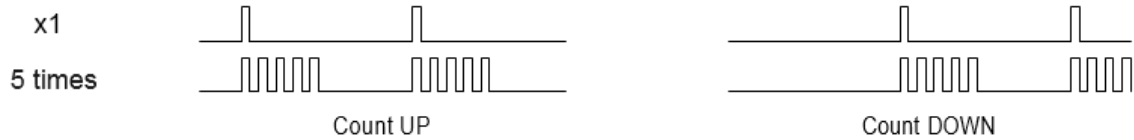
generator for the Y axis

Write:

MPMY=5 * *Set the multiplication factor of the manual pulse*

generator to 5 for the Y axis.

Multiplication (in the case of MPM = 5)



Notes:

- When MPM is used to increase the pulses output by the MPG, these additional pulses are at the speed set by the HSPD.

See also:

[Motion Settings](#)

[MPD](#), [MPE](#), [MP](#)

Commander Manual

[2.5.2 Manual pulse generator \(MPG\) operation](#)

MSR – MicroStep Resolution

MSR – MicroStep Resolution

Read/Write command that sets the MicroStep Resolution for the TBD driver board.

Applies to: CMD-4EX-SA

ASCII

MSR

Syntax:

Read:

MSR

Write:

MSR=[msr]

Where: [msr] is integer (0 to 7)

Reply:

Command is accepted and executed

Read:

Returns an integer corresponding to the current MicroStep Resolution being used. See chart below

MSR	Setting	IO30	IO31	IO32
0	Full Step	0	0	0

1	1/2 step	1	0	0
2	1/4 step	0	1	0
3	1/8 step	1	1	0
4	1/16 step	0	0	1
5	1/32 step	1	0	1
6	1/64 step	0	1	1
7	1/128 step	1	1	1

Write:

OK

*Command cannot be processed*An error message is returned. See [Error Codes](#)**Example:****Read:**

MSR

** Returns the current MicroStep Resolution***Write:**

MSR=2

** Sets the current MicroStep Resolution to 1/4 step for all axes.***Notes:**

- The MSR command is a shortcut to setting IO30 to IO32 for the CMD-4EX-SA when using the TBD driver board. The correlation between the setting of IO30 to IO32 and the MSR command are shown in the chart above.
- All axes on the TBD driver board use the same MicroStep Resolution.

See also:**Commander Manual****MST – Motor Status Acquisition Command****MST – Motor Status Acquisition Command**

A read-only command that returns axis status, including error statuses such as end limit stop, alarm stop, and EMG stop.

Applies to: CMD-4CR, CMD-4EX-SA, CMD-2ED-SA, CMD-2EX-SA

ASCII/Standalone

MST[axis]

Syntax:**Read:**

MST[axis]

Where: [axis] is set to X, Y, Z, U, to specify the axis to acquire the status

of.

Reply

Command is accepted and executed

Applies to: CMD-4CR, CMD-4EX-SA

Read:

The reply returns a 20-bit integer with the following bit structure:

Bit	Command/Function	Condition	
0	Accelerating	0: Not accelerating	1: accelerating
1	Decelerating	0: Not decelerating	1: decelerating
2	Constant Speed	0: Not at constant speed	1: At constant speed
3	Alarm Signal Input Status	0: OFF	1: ON
4	Positive End Limit Status	0: OFF	1: ON
5	Negative End Limit Status	0: OFF	1: ON
6	Home or Origin Status	0: OFF	1: ON
7	Slow Down Input Status	0: OFF	1: ON
8	Positive End Limit Error*	0: No Error	1: Error
9	Negative End Limit Error*	0: No Error	1: Error
10	Alarm Error*	0: No Error	1: Error
11	In-Position Input Status	0: OFF	1: ON
12	Deviation Counter Clear	0: OFF	1: ON
13	Z-index Input Status	0: OFF	1: ON
14	External Start Input	0: OFF	1: ON
15	EMG Signal Status	0: OFF	1: ON
16	EMG Error Status*	0: No Error	1: Error
17	Stop by Slow Down Detection**	0: Not Stopped	1: Stopped
18	Wait for In-Position Input Status	0: Not Waiting	1: Wait
19	Wait for External Start Signal Input	0: Not Waiting	1: Wait
20	External Stop Input***	0: OFF	1: ON

* For Bits 8, 9, 10, and 16, errors must be canceled with the CLR command before next operation.

** For Bit 17, additional operations can be performed with flag ON, but flag will remain ON until canceled by CLR command.

*** bit 20 is valid for firmware version V301 and later version. Older versions will always return a 0 at bit 20.

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read: MSTX ** returns the status of X-axis*

Reply decoding examples

** reply examples based on the CMD-4CR*

Reply: 64 * Returns 64 after motor at home limit, or 0000 0000 0000 0100 0000, bit 6 indicates unit is in home position limit is on.

3080 * Returns 3080 for X-axis with alarm on and alarm error message. Equivalent in binary is 0000 0000 1100 0000 1000.
Bits 3, 10, and 11 are on. This indicates alarm input is on, alarm error is on, and motor in position limit is on.

Notes:

- An online Excel calculator to decode the MST reply for the Commander series is available [here](#).

See also:

[Limits/Safety Setup](#),
[CLR](#), [PS](#)

Commander Manual

[2.1.3.1.1.1 Limit Switches](#)
[2.1.3.1.1.2 Slowdown switches](#)
[2.1.3.1.2 Emergency stop](#)
[2.6.2.1 Axis Status Acquisition Command](#)

Operators – A-script operators

Operators – A-script operators

The commands described here are only available when writing a standalone program.

A-script provides a wide variety of operators for performing various operations. An operator is a character that acts on a value or variable. For example, + is an operator for addition.

Arithmetic Operators in A-script

Arithmetic operators perform mathematical operations such as addition, subtraction, multiplication, and division on numbers (constants and variables).

[Examples using Arithmetic operators](#)

Operator	Description	Example
+	Integer Addition	V1=V2+V3
-	Integer Subtraction	V1=V2-V3
*	Integer Multiplication	V1=V2*V3
/	Integer Division (round down)	V1=V2/V3
%	Modulus	V1=V2%5
=	Assign a variable to be the same as a value or variable	V1=9

Arithmetic Operations

Bit-wise operators in A-Script

During calculations, mathematical operations such as addition, subtraction, multiplication, and division are transferred in bit units to speed up processing. A-script uses bit-level operators to perform bit-level operations.

[Examples using Bitwise operators](#)

Operat or	Description	Example
>>	Bit Shift Right	V1=V2>> 2
<<	Bit Shift Left	V1=V2<< 2
&	Bit-wise AND	V1=V2&7
	Bit-wise OR	V1=V2 8
~	Bitwise NOT	V1=~V2

Bit-wise Operations

Relational Operators in A-Script

A relational operator checks the relationship between two operands. If the relation is true, it returns 1; if the relation is false, it returns value 0. Relational operators are used in decision-making and loops.

Operat or	Description	Example
=	Equal to	7=2 is false (0)
!=	Not equal to	7!=2 is true (1)
>	Greater than	7>2 is true (1)
<	Less than	7<2 is false (0)
>=	Greater than or equal to	7>=2 is true (1)
<=	Less than or equal to	7<=2 is false (0)

Relational Operations

See also:

[Standalone Programming Commands](#)

Commander Manual

[2.7 Standalone program Specification](#)

[2.7.2 Conditional loop control](#)[2.7.6 Standalone Variables](#)[Arithmetic Operators examples](#)

Arithmetic Operators examples

Download this program:

[StandAloneArithmeticOperatorsExamples.PRG](#)

Example Program

```

1  ; Standalone Example Arithmetic operators examples
2  ; Example of how the arithmetic operators work in A-script
3  PRG 0                                ;Program start line, program 0
4      V1=9                            ;Set variable 1 to 9
5      V2=4                            ;Set variable 2 to 4
6      V3=V1+V2                        ;Set variable 3 to 9 + 4
7      V4=V1-V2                        ;Set variable 4 to 9 - 4
8      V5=V1*V2                        ;Set variable 5 to 9 * 4
9      V6=V1/V2                        ;Set variable 6 to 9 / 4
10     V7=V1%V2                        ;Set variable 7 to 9 % 4
11 END                                ;End of program 0

```

Results

V1	9
V2	4
V3	13
V4	5
V5	36
V6	2
V7	1

As expected, the operators +, -, and * perform addition, subtraction, and multiplication. In a normal calculation, $9/4 = 2.25$. However, the program outputs 2. This is because the only variables available in the A script are integer values. Therefore, the output will be an integer. The compiler ignores the decimal term and produces an answer of 2 instead of 2.25.

The modulo % operator computes the remainder: $a=9$ divided by $b=4$ yields a remainder of 1.

[Bitwise Operators examples](#)

Bitwise Operators examples

Download this program:

[StandAloneBitwiseOperatorsExamples.PRG](#)

Example Program


```

1 ; Standalone Example Bitwise operators examples
2 ; Example of how the arithmetic operators work in A-script
3 PRG 0 ;Program start line, program 0
4 V1=12 ;Set variable 1 to 12 = 00001100 (In Binary)
5 V2=25 ;Set variable 2 to 25 = 00011001 (In Binary)
6 V3=V1&V2 ;Set variable 3 to 00001100 and 00011001 = 00001000 = 8 (
7 V4=V1|V2 ;Set variable 4 to 00001100 or 00011001 = 00011101 = 29 (I
8 V5=212 ;Set variable 5 to 212 = 11010100 (In binary)
9 V6=V5>>2 ;Set variable 6 to 11010100 Right shift by two bits = 0011010
10 V7=V5>>7 ;Set variable 7 to 11010100 Right shift by seven bits = 0000001
11 V8=V5>>8 ;Set variable 8 to 11010100 Right shift by eight bits = 00000001
12 V9=V5>>0 ;Set variable 9 to 11010100 Right shift by zero bits = 1101010
13 V10=V5<<1 ;Set variable 10 to 11010100 Left shift by one bits = 11010101
14 V11=V5<<0 ;Set variable 11 to 11010100 Left shift by zero bits = 1101010
15 V12=V5<<4 ;Set variable 12 to 11010100 Left shift by four bits = 11010101
16 END ;End of program 0

```

Results

V0	0
V1	12
V2	25
V3	8
V4	29
V5	212
V6	53
V7	1
V8	0
V9	212
V10	424
V11	212
V12	3392

Bitwise AND operator (&)

The result of a bitwise AND is 1 if the corresponding bits of the two operands are 1. If any of the operand bits is 0, the result of the corresponding bit evaluates to 0.

Suppose we perform a bitwise AND operation on two integers, 12 = 00001100 (in binary) and 25 = 00011001 (in binary).

In the example above V3. Variable 3 is set to 00001100 and 00011001 = 00001000 = 8 (decimal notation).

Bitwise OR operator (|)

The result of a bitwise OR operation is 1 if at least one corresponding bit of the two operands is 1. In Ascript, bitwise OR operators are denoted as |.

In the example above V4. Variable 4 is set to 00001100 or 00011001 = 00011101 = 29 (decimal notation).

Ascript bit shift operator

The Ascript language has two shift operators.

- The bit right-shift operator >>.
- The bit left-shift operator <<.

Notice how 212 = 11010100 (binary) is converted by the bit shift operator in the

examples above.

Right-shift operator (>>)

The right-shift operator shifts all bits by the specified number of bits to the right. Bit positions freed by the right-shift operator are filled with zeros. It is indicated by the symbol >>.

Variable 6 is set to 11010100 shift 2 bits right = 00110101 (binary) = 53 (decimal)

Variable 7 is set to 11010100 shift right by 7 bits = 00000001 (binary) = 1 (decimal)

Variable 8 is set to 11010100 shift right by 8 bits = 0000000000 (binary) = 0 (decimal)

Variable 9 is set to 11010100 shift right, 0 bit = 11010100 (binary) No shift

Left-shift operator (<<)

The left-shift operator shifts all bits by the specified number of bits to the left. Bit positions freed by the left shift operator are filled with zeros. The symbol for the left-shift operator is <<.

Variable 10 is set to 11010100 Shift 1 bit left = 110101000 (binary) = 424 (decimal)

Variable 11 is set to 11010100 Left shift by zero bit = 11010100 (binary) No shift

Variable 12 is set to 11010100 4-bit left shift = 1101010000 (binary) = 3392 (decimal)

P – Pulse counter value (axis)

P – Pulse counter value (axis)

Read/write command that returns/sets the command pulse counter value for the designated axis.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII/Standalone

P[axis]

Syntax:

Read:

P[axis]

Where: [axis] is set to X, Y, Z, U

Write:

P[axis]=[value]

Where: [axis] is set to X, Y, Z, U

[value] specifies the value to command pulse counter value (32-bit number)

Reply:

Command is accepted and executed

Read:

Returns the value of the command pulse counter value (32-bit number)

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:

PY

** Returns the command pulse counter value for the Y axis.*

Write:

PX=2400

** Sets the command pulse counter to 2400 for the X axis.*

Notes:

None

See also:

[Motion Settings](#)

[D](#), [EP](#), [E](#), [PP](#)

Commander Manual

[2.6.1.1 Output Pulse Counter](#)

POL – Polarity

Read/write command that returns/sets the input modes or input logic of various signals.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII

POL[axis]

Syntax:

Read:

POL[axis]

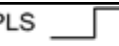
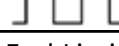
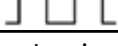
Where: [axis] specifies the axis to be addressed (X, Y, Z, U)

Write:

POL[axis]=[value]

Where: [axis] specifies the axis to be addressed (X, Y, Z, U)
[value] 17-bit value according to the chart below:

Bit	Description			
0 – 2	Output modes of command pulse signals			
	Operation in the (+) direction		Operation in the (-) direction	
	PLS output	DIR output	PLS output	DIR output
000		High		Low
001		High		Low
010		Low		High
011		Low		High

100		High	High	
101	PLS  DIR 			PLS  DIR 
110	PLS  DIR 			PLS  DIR 
111		Low	Low	
3	End Limit Signal (+/-L)	0 Positive Logic	1 Negative Logic	
4	Home Logic Signal (H)	0 Negative Logic	1 Positive Logic	
5	Alarm Signal (ALM)	0 Negative Logic	1 Positive Logic	
6	Deceleration Signal (SD)	0 Negative Logic	1 Positive Logic	
7	In-Position Signal (INP)	0 Negative Logic	1 Positive Logic	
8	Deviation Counter Clear Signal (ERC)	0 Negative Logic	1 Positive Logic	
9	Enable Axis Signal (EO)	0 Negative Logic	1 Positive Logic	
10	Direction to Count Feedback	0 Do Not Reverse	1 Reverse	
11	Specification of Feedback Pulse Signal	00 x1	01 x2	
12		10 x4	11 CW/CCW	
13	Z-Axis Signal	0 Falling Edge	1 Rising Edge	
14	Direction to Count Pulse Generator Signal	0 Do Not Reverse	1 Reverse	
15	Specification of Manual Pulse Generator	00 x1	01 x2	
16		10 x4	11 CW/CCW	

Reply:

Command is accepted and executed

Read:

Returns the input modes and input logic of various signals for the specified axis (17-bit value)

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:

POLY * Returns the input modes and logic for various signal for the Y axis

Write:

POLX=4128 * Sets the input modes and logic based on the 17-bit value.
Ex. 4128 converts to 1 0000 0010 0000. Using the chart above, the result is setting "PLS/DIR input," "Alarm with Positive Logic," and "Feedback set at x4" for the X axis.

Notes:

- Below is a link to an Excel spreadsheet Polarity of the Input/Output Logic Calculator. This calculator is to help you encode the value sent to and decode the value returned by the POL function.

[Commander Calculators.xlsx](#)

See also:

[System Setup](#), [Motor Setup](#), [Encoder Setup](#), [Limits/Safety Setup](#),
[ERC](#), [ERCD](#), [ERCP](#), [INP](#), [SDC](#), [SDE](#), [EXST](#), [IERR](#), [STORE](#)

Commander Manual

[2.1.2.1 Stepper](#)

[2.1.2.2 Digital Servo](#)

[Encoder Inputs](#)

[2.1.3.1.1.2 Slowdown switches](#)

[2.1.6.3 Input modes and the polarity of the input/output logic](#)

[2.2.1 Motor Power](#)

[2.5.2 Manual pulse generator \(MPG\) operation](#)

PRG/END – Standalone PRG/END Command**PRG/END – Standalone PRG/END Command**

The commands described here are only available when writing a standalone program.

In standalone programs, this command is used to define the beginning and end of a main program. All code is written between the PRG and END lines.

For 4 axis controllers, four (4) standalone programs are available.

For 2 axis controllers, two (2) standalone programs are available.

Applies to: CMD-4CR, CMD-4EX-SA, PMX-4EX-SA, PMX-4ET-SA, PMX-2ED-SA, PMX-2EX-SA

```
Standalone      PRG [prg#]
                  ;main program
                  END
```

Syntax:**Write:**

```
PRG [prg#]
```

Where: [prg#] [0-3] for 4 axis controllers [CMD-4CR, CMD-4EX-SA, PMX-4EX-SA, PMX-4ET-SA] [0-1] for 2 axis controllers [PMX-2ED-SA, PMX-2EX-SA]

```
END
```

Reply:

Command is accepted and executed

No Reply

Example:

Write:

PRG 0	<i>* Program start line, program 0</i>
HSPD=20000	<i>* Set the high speed to 20000 pulse/sec</i>
LSPD=500	<i>* Set the low speed to 500 pulse/sec</i>
;main program	<i>* main program code.</i>
END	<i>* End of program 0</i>

Notes:

- If PRG number is not included then program is assigned to program 0.
- If only one main program is used in the standalone, then PRG is not required.

See also:

[Standalone Programming Commands](#)

Commander Manual

[Identification Number](#)

PP – Pulse counter value (all)

PP – Pulse counter value (all)

A read-only command, returns the current pulse counter values for all axes at once.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII/Standalone

PP

Syntax:

Read:

PP

Reply:

Command is accepted and executed

Read:

Four integers in the format "%d:%d:%d:%d" [axisx]:[axisy]:[axisz]:[axisu]

Where: [axisx] – indicates the pulse portion for axis X [axisy] – indicates the pulse portion for axis Y [axisz] – indicates the pulse portion for axis Z [axisu] – indicates the pulse portion for axis U

Example:

Read:

PP ** Returns the current pulse counter values of all axes*
Ex. Return value of 1000:0:542:-2000, indicates
Axis X = 1000
Axis Y = 0
Axis Z = 542
Axis U = -2000

Notes:

None

See also:

[Motion Settings](#)

[D](#), [EP](#), [E](#), [P](#)

Commander Manual

[2.6.1.1 Output Pulse Counter](#)

PS – Pulse Speed

PS – Pulse Speed

Read-only command that returns the current axis operation speed. When axis is stopped, "0" is achieved.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII/Standalone	PS	<i>speed of each axis returned separated by :</i>
	PS[axis]	<i>speed of specified axis returned</i>

Syntax:

Read:

PS[axis]

Where: [axis] is set to X, Y, Z, U, or not specified to read global value

Reply:

Command is accepted and executed

Read:

global speed reply

Four integers in the format "%d:%d:%d:%d" [axisx]:[axisy]:[axisz]:[axisu]

Where: [axisx] – indicates the speed for axis X

[axisy] – indicates the speed for axis Y

[axisz] – indicates the speed for axis Z

[axisu] – indicates the speed for axis U

individual axis speed reply

Returns the value of the [axis] speed

Example:

Read:

Reply: PSX ** Request the current speed of the X axis*

10000 ** Returns X axis is moving at 10,000pps*

Read: PS ** Request the current speed of all axes*

Reply: 10000:0:200:0 ** Returns that X axis is moving at 10,000pps, Y and U axes are stopped (0pps), Z axis is moving at 200pps*

Notes:

- If StepNLoop mode is active then the returned value units are encoder counts / sec
- For all other modes, the units for returned values are pulse / sec
- Based on speed magnification resolution, actual speed can vary from commanded speed by up to 0.1%.

See also:

[Motion Settings](#)

[ACC](#), [DEC](#), [HSPD](#), [JKA](#), [JKD](#), [LSPD](#), [SCV](#), [SSPDM](#)

Commander Manual

[2.6.2.2 Pulse Speed register](#)

PWM – Pulse Width Modulation Duty Cycle

PWM – Pulse Width Modulation Duty Cycle

Sets the PWM duty cycle in %. The PWM outputs run at 500Hz and can range from 1% - 99% on-time duty cycle

Applies to: CMD-4CR, CMD-4EX-SA, CMD-2ED-SA, CMD-2EX-SA

ASCII/Standalone PWM

Syntax:

Read:

PWM[channel]

Where: [channel] specifies the PWM output (1 or 2)

Write:

PWM[channel]=[percentage]

Where: [channel] specifies the PWM output (1 or 2)

[percentage] specifies the on-time duty cycle in % (1-99)

Reply:

Command is accepted and executed

Read:

Returns the % on-time duty cycle for the specified PWM output (1 or 2)

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:

PWM2 ** Returns the on-time duty cycle value the signal is high*

Write:

PWM1=50 ** Sets the time the signal is low for PWM output 1 to 50%.*

Notes:

- None

See also:

[I/O Setup](#)

Commander Manual

[Pulse-width modulated Outputs](#)

SASTAT – Standalone Program Status

SASTAT – Standalone Program Status

Read-only command that returns the operation status of a standalone program.

Applies to: CMD-4CR, CMD-4EX-SA, PMX-4EX-SA, PMX-4ET-SA, PMX-2ED-SA, PMX-2EX

ASCII SASTAT[prgNo]

Syntax:

Read:

SASTAT[prgNo]

Where: [prgNo] specifies the program number (0-3)

Reply:

Command is accepted and executed

Read:

Returns a value 0 – 3 indicating current program operation status

Value	Program Status
0	Program [prgNo] is Idle
1	Program [prgNo] is Running
2	Program [prgNo] is Paused
3	Program [prgNo] has an Error

Example:

Read:

SASTAT0

** Returns operation status of Program 0*

Notes:

- Note

See also:

[Motion Settings](#)

[SACTRL](#), [SAP](#), [SLOAD](#), [V](#), [GS](#), [SPC](#)

Commander Manual

[Identification Number](#)

SACTRL – Standalone Control

SACTRL – Standalone Control

Write-only command that controls the start and stop of a standalone program.

Applies to: CMD-4CR, CMD-4EX-SA, CMD-2ED-SA, CMD-2EX-SA

ASCII SACTRL
Standalone SR

Syntax:

Write:

SACTRL[prgNo]=[control]

Where: [prgNo] specifies the program number (0-3)
[control] specifies control action (0-3)

Value	Control Action
0	Stop program [prgNo]
1	Start program [prgNo]
2	Pause program [prgNo]
3	Continue program [prgNo]

Reply:

Command is accepted and executed

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Write:

SACTRL0=1 ** Start standalone program 0.*

Notes:

- None

See also:

[Motion Settings](#)

[SAP](#), [SASTAT](#), [SLOAD](#), [V](#), [GS](#), [SPC](#)

Commander Manual

[Identification Number](#)

SAMAX – Standalone Maximum Lines

SAMAX – Standalone Maximum Lines

Read-only command that returns the maximum number of standalone program lines. The line number here is the number of the Assembly code.

Applies to: CMD-4CR, CMD-4EX-SA, CMD-2ED-SA, CMD-2EX-SA

ASCII SAMAX

Syntax:

Read:

SAMAX

Reply:

Command is accepted and executed

Read:

Returns the maximum number of standalone program lines.

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:

SAMAX ** Returns the maximum number of standalone program lines.*

Notes:

- This value will not change on your Commander, but may be different from one version of firmware to the next.

See also:

[Motion Settings](#)

Commander Manual

[Identification Number](#)

SAP – Standalone Error Handling**SAP – Standalone Error Handling**

This command determines at what line a program will restart after an error is cleared. When an error occurs in a standalone program, it will automatically jump to subroutine 31. Typically, the user will establish an error-handling routine to cancel the error. The SAP command indicates where the program should return – either jump back to the first line in the program or return to the line following where the error occurred.

Applies to: CMD-4CR, CMD-4EX-SA, CMD-2ED-SA, CMD-2EX-SA

ASCII

SAP

Syntax:**Read:**

SAP

Write:

SAP=[mode]

Where: [mode] specifies which line to return to in the program after the error is cleared:

0: Continue from the next line where the error occurred

1: Continue from the first line of the program

Reply:

Command is accepted and executed

Read:

Returns the setting of the command (0-1)

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:**Read:**

SAP

* Returns the setting of return line on error clear; 0, or 1

Write:

SAP=1

* After the error is cleared, return to the first line of the program.

Notes:

- None

See also:[Motion Settings](#)[SACTRL](#), [SASTAT](#), [SLOAD](#), [V](#), [GS](#), [SPC](#)**Commander Manual**[Identification Number](#)**SCV – S-Curve Acceleration****SCV – S-Curve Acceleration**

Enables/disables the S-curve acceleration/deceleration mode of an axis, or returns the current set value. S-curve mode provides smoother starts and stops during acceleration/deceleration, minimizing or eliminating overshoot and shake. When S-curve acceleration/deceleration is disabled, linear acceleration/deceleration is used.

Applies to: CMD-4CR, CMD-4EX-SA, CMD-2ED-SA, CMD-2EX-SA

ASCII/Standalone

SCV[axis]

Syntax:**Read:**

SCV[axis]

Where: [axis] is the axis (X,Y,Z,U, not specified) being queried.

Write:

SCV[axis]=[0 or 1]

Where: [axis] is set to X, Y, Z, U to specify which axis is enabling/disabling S-curve mode

Valid values for SCV: 0 (disabled), 1 (enabled)

Reply*Command is accepted and executed***Read:**

Returns 0 when disabled, 1 when enabled

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:

SCVY

** Returns the status of the Y axis S-Curve mode setting*

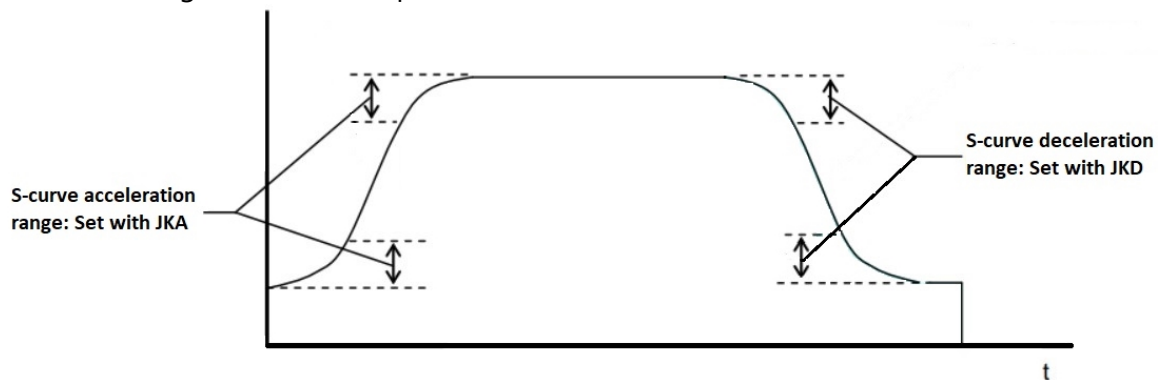
Write:

SCVX=1

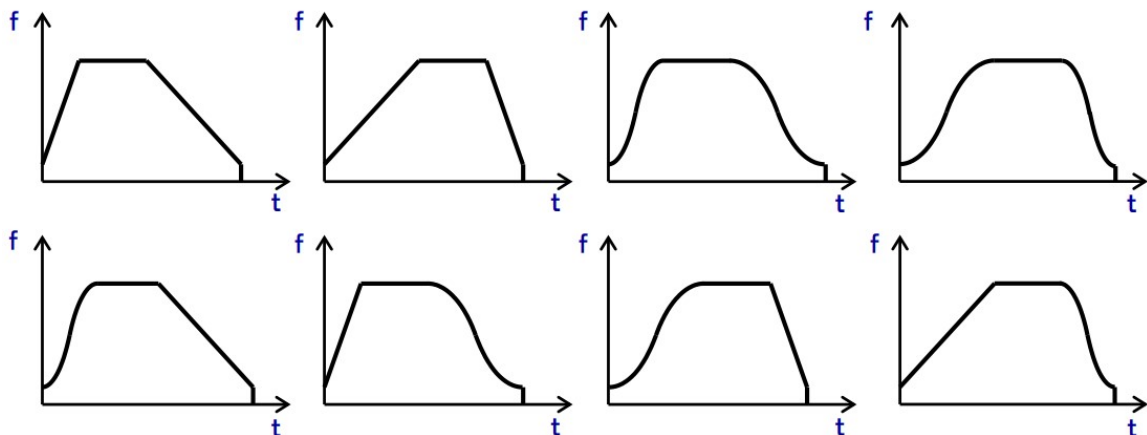
** Enables S-curve acceleration/deceleration mode for X axis*

Notes:

- When S-curve is disabled, linear acceleration/deceleration is used.
- When S-curve is enabled, changing jerk acceleration ([JKA](#)) and jerk deceleration ([JKD](#)) allows for fine-tuning of the S-curve profile.



- It is possible to set mixed acceleration/deceleration types in a single profile. Changing the jerk acceleration (JKA) or jerk deceleration (JKD) to 0 makes the acceleration/deceleration linear.



See also:

[Motion Settings](#)

[ACC](#), [DEC](#), [HSPD](#), [JKA](#), [JKD](#), [LSPD](#), [PS](#), [SSPDM](#)

Commander Manual

[2.1.4 Motion Profile](#)

SDC – Slow Down Configuration

SDC – Slow Down Configuration

This is the command to set the operation when the slowdown input signal is detected or obtain the current set value.

If "**decelerate only**" is selected, the speed will return to the high speed when the slowdown input signal turns off.

If "**decelerate and stop**" is selected, the operation will not restart even if the slowdown input signal turns off after the axis motion stops.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII SDC[axis]

Syntax:

Read:

SDC[axis]

Where: [axis] is set to X, Y, Z, U

Write:

SDC[axis]=[sdc]

Where: [axis] is set to X, Y, Z, U

[sdc] specify a value to set the operation when detecting a signal (0, 1)

0 : Decelerate only

1 : Decelerate and stop

Reply:

Command is accepted and executed

Read:

Returns the value of the [axis] operation when the slowdown input is active. (0, 1)

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:

SDCY

** Returns the operation mode for the Y axis when the slowdown input is triggered.*

Write:

SDCZ=1

** Sets axis Z to decelerate and stop when the slowdown*

input is triggered.

Notes:

The **SDC[axis]** command determines the specific operation of the SD input. By default, **SDC[axis]=0** sets the operation, when triggered, to decelerate the motor to LSPD without stopping. When **SDC[axis]=1**, the operation is set to decelerate and stop when the SD input is triggered.

Value	Description
0	Decelerate only
1	Decelerate and stop

See also:

[Limits/Safety Setup](#)

[ERC](#), [ERCD](#), [ERCP](#), [INP](#), [POL](#), [SDE](#), [EXST](#), [IERR](#), [STORE](#)

Commander Manual

[2.1.3.1.1.2 Slowdown switches](#)

SDE – Slow Down Enable/Disable

SDE – Slow Down Enable/Disable

This is the command to “enable/disable” the slowdown input signal and sets the latching function to obtain the current set value.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII SDE[axis] *individual axis acceleration setting*

Syntax:

Read:

SDE[axis]

Where: [axis] is set to X, Y, Z, U

Write:

SDE[axis]=[sde]

Where: [axis] is set to X, Y, Z, U, or not specified to read global value

[sde] 2-bit value

bit 0 SD input 0 Disabled 1 Enabled

bit 1 SD latching 0 Disabled 1 Enabled

Reply:

Command is accepted and executed

Read:

Returns the 2-bit value of the [axis] enable/latching status

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:

SDEY ** Returns the enable/latching status for the Y axis*

Write:

SDEX=1 ** Enables the slowdown input for the X axis.*

Notes:

- The 2-bit **SDE[axis]** command enables the SD inputs and sets the latching function. By default, **SDEX=0** disables the X-axis SD inputs. The inputs can be enabled without a latch function using **SDEX=1**, or enabled with a latch function **SDEX=3**.

Bit	Description	Setting	
0	SD input	0 - Disabled	1 - Enabled
1	SD latching	0 - Disabled	1 - Enabled

See also:

[Limits/Safety Setup](#),
[ERC](#), [ERCD](#), [ERCP](#), [INP](#), [POL](#), [SDC](#), [EXST](#), [IERR](#), [STORE](#)

Commander Manual

[2.1.3.1.1.2 Slowdown switches](#)

[2.1.6 System settings](#)

[2.1.7 Flash Memory](#)

[4.1 ASCII Command Set](#)

SFL – Soft-limit Enable/Disable

SFL – Soft-limit Enable/Disable

Enables/disables the soft-limit function of an axis, or returns the current set value. The soft-limits can be used in a system where physical limits are impossible or for setting operation limits within the allowable mechanical stroke limits.

Applies to: CMD-4CR, CMD-4EX-SA, Firmware V 2.01+

ASCII

SFL[axis]

Syntax:

Read:

SFL[axis]

Where: [axis] is set to X, Y, Z, U

Write:

SFL[axis]=[0 or 1]

Where: [axis] is set to X, Y, Z, U

[0 or 1] Valid values for SFL: 0 (disabled), 1 (enabled)

Reply:

Command is accepted and executed

Read:

Returns 0 when disabled, 1 when enabled

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

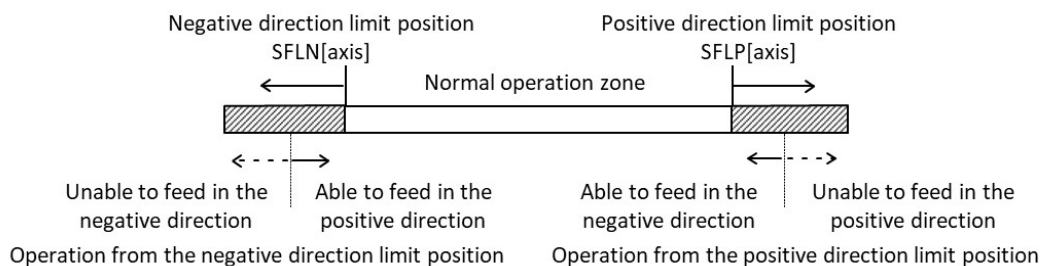
Read:

SFLY ** Returns the status of the Y axis soft-limit function setting*

Write:

SFLX=1 ** Enables soft-limit function for X axis*

Notes:



- Before enabling soft-limits, check that softLimitPositive SFLP[axis] and softLimitNegative SFLN[axis] are different and that the positive value is greater than the negative value. If not, an error is returned.
- Soft-limits act similarly to limit switches. While the controller will command a stop when a soft-limit is encountered, it is possible to overrun the soft limit. The amount the system can overrun the limit depends on the speed the motor is moving at when the soft-limit is encountered, and the mode selected for stopping the motor ([SFLSD\[axis\]](#)). As such, a safety buffer should be included in any any soft-limit settings.

See also:

[Limits/Safety Setup](#),
[SFLP](#), [SFLN](#), [SFLSD](#), [SFLSO](#)

Commander Manual

[2.1.3.1.1.4 Soft-Limits](#)

SFLN – Negative Soft-limit setting**SFLN – Negative Soft-limit setting**

Sets/returns the comparator value used for a negative direction soft-limit.

Applies to: CMD-4CR, CMD-4EX-SA, Firmware V 2.01+

ASCII SFLN[axis]

Syntax:**Read:**

SFLN[axis]

Where: [axis] is set to X, Y, Z, U

Write:

SFLN[axis]=[pos]

Where: [axis] is set to X, Y, Z, U

[pos] specifies target of negative soft-limit (28-bit value)

Reply:

Command is accepted and executed

Read:

Returns the comparator value of the [axis] negative direction soft-limit
 (28-bit value)

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

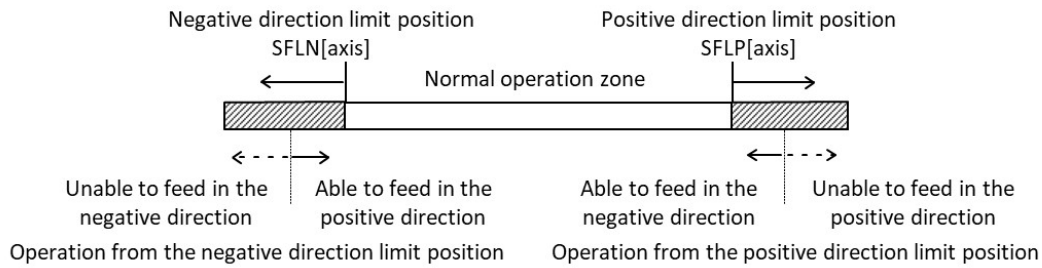
Example:**Read:**

SFLNY ** Returns the comparator value of the Y axis negative
 direction soft-limit*

Write:

SFLNX=-6000 ** Sets X axis negative direction soft-limit to -6000*

Notes:



- Before enabling soft-limits, check that `softLimitPositive SFLP[axis]` and `softLimitNegative SFLN[axis]` are different and that the positive value is greater than the negative value. If not, an error is returned.
- Soft-limits act similarly to limit switches. While the controller will command a stop when a soft-limit is encountered, it is possible to overrun the soft limit. The amount the system can overrun the limit depends on the speed the motor is moving at when the soft-limit is encountered, and the mode selected for stopping the motor ([SFLSD\[axis\]](#)). As such, a safety buffer should be included in any soft-limit settings.

See also:

[Limits/Safety Setup](#),
[SFL](#), [SFLP](#), [SFLSD](#), [SFLSO](#)

Commander Manual

[2.1.3.1.1.4 Soft-Limits](#)

SFLP – Positive Soft-limit setting

SFLP – Positive Soft-limit setting

Sets/returns the comparator value used for a positive direction soft-limit.

Applies to: CMD-4CR, CMD-4EX-SA, Firmware V 2.01+

ASCII SFLP[axis]

Syntax:

Read:

SFLP[axis]

Where: [axis] is set to X, Y, Z, U

Write:

SFLP[axis]=[pos]

Where: [axis] is set to X, Y, Z, U

[pos] specifies target of positive soft-limit (28-bit value)

Reply:

Command is accepted and executed

Read:

Returns the comparator value of the [axis] positive direction soft-limit (28-bit value)

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

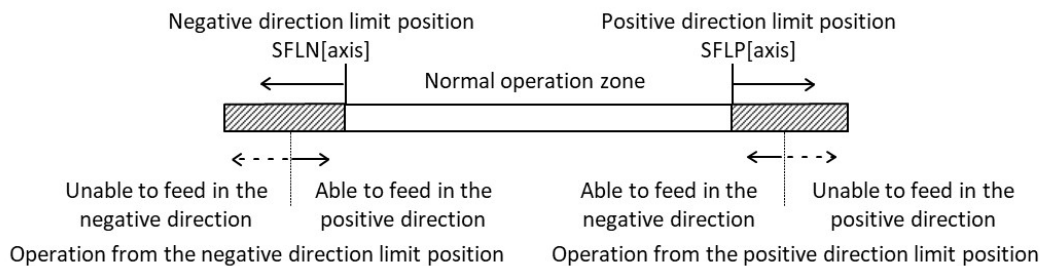
Read:

SFLPY ** Returns the comparator value of the Y axis positive direction soft-limit*

Write:

SFLPX=8000 ** Sets X axis positive direction soft-limit to 8000*

Notes:



- Before enabling soft-limits, check that softLimitPositive SFLP[axis] and softLimitNegative SFLN[axis] are different and that the positive value is greater than the negative value. If not, an error is returned.
- Soft-limits act similarly to limit switches. While the controller will command a stop when a soft-limit is encountered, it is possible to overrun the soft limit. The amount the system can overrun the limit depends on the speed the motor is moving at when the soft-limit is encountered, and the mode selected for stopping the motor ([SFLSD\[axis\]](#)). As such, a safety buffer should be included in any any soft-limit settings.

See also:

[Limits/Safety Setup](#),
[SFL](#), [SFLN](#), [SFLSD](#), [SFLSO](#)

Commander Manual

[2.1.3.1.1.4 Soft-Limits](#)

SFLSD – Soft-limit Stop Mode

SFLSD – Soft-limit Stop Mode

Sets/returns the stopping method used when reaching a soft-limit.

Applies to: CMD-4CR, CMD-4EX-SA, Firmware V 2.01+

ASCII SFLSD[axis]

Syntax:

Read:

SFLSD[axis]

Where: [axis] is set to X, Y, Z, U

Write:

SFLSD[axis]=[0 or 1]

Where: [axis] is set to X, Y, Z, U

Valid values for SFLSD: 0 (Immediate stop), 1
(Decelerate and stop)

Reply:

Command is accepted and executed

Read:

Returns 0 in Immediate stop mode, 1 when Decelerate and stop mode

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:

SFLSDY * Returns stopping method used for the Y axis

Write:

SFLSDX=1 * Sets stopping method used for X axis to decelerate and stop.

Notes:

- When set to decelerate and stop the motor stops at the deceleration rate set with the DEC once the soft-limit is reached.
- Soft-limits act similarly to limit switches. While the controller will command a stop when a soft-limit is encountered, it is possible to overrun the soft limit. The amount the system can overrun the limit depends on the speed the motor is moving at when the soft-limit is encountered, and the mode selected for stopping the motor ([SFLSD\[axis\]](#)). As such, a safety buffer should be included in any any soft-limit settings.

See also:

[Limits/Safety Setup](#),
[SFL](#), [SFLP](#), [SFLN](#), [SFLSO](#)

Commander Manual

[2.1.3.1.1.4 Soft-Limits](#)

SFLSO – Soft-limit Source

SFLSO – Soft-limit Source

The **SFLSO[axis]** command selects the encoder counter (**1**) or pulse generator counter (**0**) for the soft-limit axis position.

Applies to: CMD-4CR, CMD-4EX-SA, Firmware V 2.01+

ASCII SFLSO[axis]

Syntax:

Read:

SFLSO[axis]

Where: [axis] is set to X, Y, Z, U

Write:

SFLSO[axis]=[source]

Where: [axis] is set to X, Y, Z, U

[source] encoder counter (**1**) or pulse generator counter (**0**)

Reply:

Command is accepted and executed

Read:

Returns 1 for the encoder counter or 0 if pulse generator counter is the source for the the soft-limit axis position.

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:

SFLSOY
for the Y axis

** Returns source for the the soft-limit axis position*

Write:

SFLSOX=1
limit axis position for the X axis.

** Sets the encoder counter as the source for the the soft-*

Notes:

- It is safest to use the encoder counter if an encoder is available, as it will allow the tracking of the motor even if an outside force moves the motor.

See also:

[Limits/Safety Setup](#),
[SFL](#), [SFLP](#), [SFLN](#), [SFLSD](#)

Commander Manual

[2.1.3.1.1.4 Soft-Limits](#)

SL – StepNLoop Enable**SL – StepNLoop Enable**

Enables/disables the StepNLoop function for the specified axis. At the end of any targeted move, StepNLoop will perform a correction if the current error is greater than the tolerance value

Applies to: CMD-4CR, CMD-4EX-SA, CMD-2ED-SA, CMD-2EX

ASCII/Standalone

SL

Syntax:**Read:**

SL[axis]

Where: [axis] specifies the axis to be enabled/disabled (X, Y, Z, U)

Write:

SL[axis]=[value]

Where: [axis] specifies the axis to be enabled/disabled (X, Y, Z, U)

Where: [value] specify 0 to disable or 1 to enable

Reply:

Command is accepted and executed

Read:

Returns the enable/disable status (0 – disabled, 1 – enabled)

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:**Read:**

SLY

** Returns the value (0, 1) of the Y-axis*

Write:

SLX=1 ** Enables StepNLoop function for the X-axis*

Notes:

- Joystick and buffer operation are invalid during StepNLoop operation

See also:

[StepNLoop](#)

[SLA](#), [SLE](#), [SLR](#), [SLRA](#), [SLS](#), [SLT](#),

Commander Manual

[2.5.3 StepNLoop Closed Loop Control operation](#)

SLA – StepNLoop Attempts**SLA – StepNLoop Attempts**

The SLA command is used to set the number of attempts to complete the StepNLoop operation. The operation is completed when the position of the axis is within the tolerance range of the target position (SLT). If after the number of attempts is reached, and the axis position is not within the tolerance range, the StepNLoop operation is canceled, and an error status is output

Applies to: CMD-4CR, CMD-4EX-SA

ASCII

SLA

Syntax:**Read:**

SLA[axis]

Where: [axis] specifies the axis to be set (X, Y, Z, U)

Write:

SLA[axis]=[value]

Where: [axis] specifies the axis to be set (X, Y, Z, U)

[value] specifies the maximum number of attempts (0 – 2,147,483,647) 31-bit value (0 is unlimited attempts)

Reply:

Command is accepted and executed

Read:

Number of attempts for the specified axis

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:**Read:**

SLAX ** Returns the value set indicating the number of attempts allowed for the X-axis*

Write:

SLAY=10 ** Sets the number of retry limit to 10 for the Y-axis*

Notes:

- Joystick and buffer operation are invalid during StepNLoop operation
- When SLA is set to 0, there is no limit and the commander will continue to try and reach the tolerance range of the target position until successful.

See also:

[StepNLoop](#)

[SL](#), [SLE](#), [SLR](#), [SLRA](#), [SLS](#), [SLT](#),

Commander Manual

[2.5.3 StepNLoop Closed Loop Control operation](#)

SLE – StepNLoop Error Range**SLE – StepNLoop Error Range**

The SLE command sets the error tolerance range for the StepNLoop operation. If the position is outside this error range compared to the target position after a move is completed, the operation will stop and an error indicated.

Applies to: CMD-4CR, CMD-4EX-SA, CMD-2ED-SA, CMD-2EX

ASCII

SLE

Syntax:**Read:**

SLE[axis]

Write:

SLE[axis]=[value]

Where: [axis] specifies the axis to be set (X, Y, Z, U)
 [value] specifies the maximum number of pulses from target position (32-bit value)

Reply:

Command is accepted and executed

Read:

Returns the value set in pulses

Write:

OK

*Command cannot be processed*An error message is returned. See [Error Codes](#)**Example:****Read:**

SLEX

** Returns allowable error in pulses for the X-axis***Write:**

SLEZ=1000

** Sets allowable error to 1000 pulses from target position
for the Z-axis***Notes:**

- Joystick and buffer operation are invalid during StepNLoop operation

See also:[StepNLoop](#)[SL](#), [SLA](#), [SLR](#), [SLRA](#), [SLS](#), [SLT](#),**Commander Manual**[2.5.3 StepNLoop Closed Loop Control operation](#)**SLOAD – Standalone Program Autostart****SLOAD – Standalone Program Autostart**

This command controls whether a program will start on power up.

Applies to: CMD-4CR, CMD-4EX-SA, CMD-2ED-SA, CMD-2EX

ASCII

SLOAD[prgNo]

Syntax:**Read:**

SLOAD[prgNo]

Write:

SLOAD[prgNo]=[value]

Where: [prgNo] specifies the program number (0-3)
 [value] specifies if the program will start on power up, or remain idle
 as follows:

0: Stay in idle
 1: Start program on power up

Reply:*Command is accepted and executed***Read:**

Returns the value set for standalone program start (0-1)

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:

SLOAD1 ** Returns value set on power up for Program 1*

Write:

SLOAD0=0 ** Sets standalone Program 0 to be idle on power up*

Notes:

- Joystick and buffer operation are invalid during StepNLoop operation

See also:

[Standalone Programming Commands](#)

[SACTRL](#), [SAP](#), [SASTAT](#), [SPC](#), [V](#), [GS](#)

Commander Manual

[Identification Number](#)

SLR – StepNLoop Pulse Conversion

SLR – StepNLoop Pulse Conversion

The SLR command sets the pulse conversion value for the StepNLoop operation. It is the ratio of motor pulses to encoder counts.

Applies to: CMD-4CR, CMD-4EX-SA, PMX-4EX-SA, PMX-4ET-SA, PMX-2ED-SA, PMX-2EX-SA

ASCII SLR SLRAX:1000000:5

Syntax:

Read:

SLR[axis]

Write:

SLR[axis]=[value]

Where: [axis] specifies the axis to be set (X, Y, Z, U)

[value] specifies the pulse conversion value (0.001 – 999.999)

Reply:

Command is accepted and executed

Read:

Returns the set conversion value (0.001 – 999.999)

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:

SLRX

** Returns pulse conversion value for the X axis*

Write:

SLRY=0.5

** Sets the pulse conversion value to 0.5 for the Y axis*

Notes:

- The easy way to set SLR[axis] is by using the [SLRA\[axis\]](#) the automatic setting command.
- A convenient way to find the StepNLoop ratio is to set EX=0, PX=0 and move the motor +1000 pulses. The ratio can be calculated by dividing 1000 by the resulting EX value. Note that the value must be positive. If it is not, then the direction polarity must be adjusted. This test should be performed while StepNLoop is disabled.
- The ratio will depend on the motor type, micro-stepping, encoder resolution, and decoding multiplier. Joystick and buffer operation are invalid during StepNLoop operation.

See also:

[StepNLoop](#)

[SL](#), [SLA](#), [SLE](#), [SLRA](#), [SLS](#), [SLT](#),

Commander Manual

[2.5.3 StepNLoop Closed Loop Control operation](#)

SLRA – StepNLoop Automatic Pulse Conversion

SLRA – StepNLoop Automatic Pulse Conversion

The SLRA command automatically sets the pulse conversion ratio for the StepNLoop operation. It is the ratio of motor pulses to encoder counts. There is a fully automated and semi-automated version of the command.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII

SLRA[axis]

Syntax:

Write: (fully automated)

SLRA[axis]

Where: [axis] specifies the axis to be set (X, Y, Z, U)

Automatic test 1000 pulses, back and forth 2 times

Write: (semi-automated)

SLRA[axis]:[distance]:[times]

Where: [axis] specifies the axis to be set (X, Y, Z, U)
 [distance] specifies the number of pulse to move the motor (1 - 10,000)
 [times] specifies the number of cycles the motor will move (1 - 10)

Reply:

Command is accepted and executed

Write:

Returns the calculated ratio value (0.001 – 999.999)

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Write:

SLRAX:2000:5 * *Calculates and sets the pulse conversion value for the X axis. moves 2,000 steps forward and back 5 times*

SLRAY * *Calculates and sets the pulse conversion value for the X axis. moves 1,000 steps forward and back 2 times*

Notes:

- This test should be performed while StepNLoop is disabled.
- The motor must be enabled before sending the SLRA[axis] command.
- The fully automated test, will move the motor +1000 pulses in the forward and reverse directions 2 times for a total of 4 movements. If your system needs a different distance or number of times use the semi-automated version of the command.
- The ratio will depend on the motor type, micro-stepping, encoder resolution, and decoding multiplier. Joystick and buffer operation are invalid during StepNLoop operation.

See also:

[StepNLoop](#)

[SL](#), [SLA](#), [SLE](#), [SLR](#), [SLS](#), [SLT](#),

Commander Manual

[2.5.3 StepNLoop Closed Loop Control operation](#)

SLS – StepNLoop status

SLS – StepNLoop status

This read-only command indicates the execution status of the StepNLoop closed-loop control operation.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII/Standalone

SLS

Syntax:

Read:

SLS[axis]

Where: [axis] is set to X, Y, Z, U

Reply:

Command is accepted and executed

Read:

Returns a numeral value of 0 – 12 indicating the current StepNLoop status (see chart below)

Number	Description
0	Idle
1	Moving
2	Correcting
3	Stopping
4	Aborting
5	Jogging
6	Homing
7	Z-Homing
8	Correction range error*
9	Correction attempt error*
10	Stall error – Deviation counter value has exceeded correction range value*
11	Limit error
12	N/A

* Cancel the error status with the CLR command before executing the next operation

Example:

Read:

SLSX

* Returns the status value of the StepNLoop operation on the X axis

Notes:

- Joystick and buffer operation are invalid during StepNLoop operation.

See also:

[StepNLoop](#)

[SL](#), [SLA](#), [SLE](#), [SLR](#), [SLRA](#), [SLT](#),

Commander Manual

[2.5.3 StepNLoop Closed Loop Control operation](#)

SLT – StepNLoop In Position Range

SLT – StepNLoop In Position Range

Sets or reads the in-position tolerance range for the StepNLoop operation.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII/Standalone

SLT[axis]

Syntax:

Read:

SLT[axis]

Where: [axis] is set to X, Y, Z, U

Write:

SLT[axis]=[value]

Where: [axis] is set to X, Y, Z, U

[value] is the positioning completion range value (32-bit value)

Reply:

Command is accepted and executed

Read:

Returns the value of the positioning completion range value (32-bit value)

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:

SLTX

** Returns in-position range in pulses for the X axis*

Write:

SLTZ=100

** Sets in-position range to 100 pulses from target position for the Z axis*

Notes:

- The value for SLT should always be less than the value for **SLE[axis]**.
- Joystick and buffer operation are invalid during StepNLoop operation.

See also:

[StepNLoop](#)

[SL](#), [SLA](#), [SLE](#), [SLR](#), [SLRA](#), [SLS](#),

Commander Manual

[2.5.3 StepNLoop Closed Loop Control operation](#)

SN – Commander Serial Number

SN – Commander Serial Number

A read only command that returns the serial number of the Commander.

Applies to: CMD-4CR, CMD-4EX-SA

Firmware: V301+

ASCII SN

Syntax:

Read:

SN

Reply:

Command is accepted and executed

Read:

Returns the unique Commander serial number.

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:

SN

** Returns the on-time duty cycle value the signal is high*

Notes:

- The SN is formatted in one of two ways
 - o NPA-XX-MYY-mmNNN MY is the Date Code where M is Later for Month (A=January, B=February, etc) and YY is last two digits for year Example 2025 = 25
 - o Returns the 32 character unique hardware serial number. (Some Gen 5 and all Gen 4 Commanders use this format)

See also:

[I/O Setup](#)

Commander Manual

[Pulse-width modulated Outputs](#)

TODO: See also and Commander Manual links

SPC – Standalone index line

SPC – Standalone index line

This command obtains a line number in a standalone program that is being executed. The line number here is the number of the Assembly code. It is different from the line number that is actually written. This command is read-only.

Applies to: CMD-4CR, CMD-4EX-SA, CMD-2ED-SA, CMD-2EX

ASCII

SPC

Syntax:

Read:

SPC[prgNo]

Where: [prgNo] specifies the program number (0-3)

Reply:

Command is accepted and executed

Read:

Returns the line number which is being executed. (0~1799)

Example:

Read:

SPC0

** Returns the line number which is being executed in the standalone program 0.*

Notes:

- None

See also:

[Motion Settings](#)

[SACTRL](#), [SAP](#), [SASTAT](#), [SLOAD](#), [V](#), [GS](#), [SPC](#)

Commander Manual

[Identification Number](#)

SPL – Standalone line

SPL – Standalone line

Sets the deceleration time or acquires the current set value. Deceleration time is available as a global value for all axes and a value settable for each axis

Applies to: CMD-4CR, CMD-4EX-SA, PMX-4EX-SA, PMX-4ET-SA, PMX-2ED-SA, PMX-2EX

ASCII/Standalone

decel

global acceleration setting

decel[axis] *individual axis acceleration setting*

Syntax:

Read:

decel[axis]

Where: [axis] is set to X, Y, Z, U, or not specified to read global value

Write:

decel[axis]=[accel]

Where: [axis] is set to X, Y, Z, U, or not specified to read global value
[accel] is the deceleration time in ms (0 – 65,535)

Reply:

Command is accepted and executed

Read:

Returns the value of the [axis] deviation counter (28-bit value)

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:

DECY ** Returns deceleration time set for the Y axis in ms*

Notes:

- This command is not fully working yet Leave as "OUT of DATE" status

See also:

[Motion Settings](#)

[accel](#), [SCV](#), [HSPD](#), [LSPD](#)

Commander Manual

[Identification Number](#)

SSPD – On-the-Fly Speed Change

SSPD – On-the-Fly Speed Change

Write-only command that changes the speed of a current operation on the fly.

Applies to: CMD-4CR, CMD-4EX-SA, CMD-2ED-SA, CMD-2EX

ASCII/Standalone

SSPD

Syntax:

Write:

SSPD[axis]=[newspeed]

Where: [axis]	specifies the axis to change speed (X, Y, Z, U)
[newspeed]	specifies the new speed in pps (1 – 6,550,000)

Reply:*Command is accepted and executed***Write:**

OK

*Command cannot be processed*An error message is returned. See [Error Codes](#)**Example:****Write:**SSPDX=100000 * *Changes the speed of the X-axis to 100,000 pps***Notes:**

- SSPDM must be set correctly for new speed value, otherwise an ?Error SSPD with be indicated.
- Buffer(BO) and S-Curve(SCV) must be disabled.

See also:

[Axis Motion](#)
[SSPDM, T](#)

Commander Manual[2.2.4 Positioning operation](#)**SSPDM – Speed Range****SSPDM – Speed Range**

Sets/returns the speed range and relationship between the reference clock speed and the output speed for a given axis. As the magnification rate is increased, the speed setting units will tend to be approximations. It is recommended to set the magnification rate as low as possible.

The relationship between the value entered and the magnification rate is shown in the table below.

| [Applies to: CMD-4CR, CMD-4EX-SA](#)

ASCII/Standalone

SSPDM[axis]

Syntax:

Read:

SSPDM[axis]

Write:

SSPDM[axis]=[rmg]

Where: [axis] specifies the axis the speed range will be set for (X, Y, Z, U)
 [rmg] specifies the speed range according to the chart below:

[rmg]]	HSPD Value	Minimum LSPD Value	Magnification
0	Automatic switching of speed range based on the setting of HSPD	-	-
1	1 – 65,535	1	1
2	2 – 131,070	2	2
3	5 – 327,675	5	5
4	10 – 655,350	10	10
5	20 – 1,310,700	20	20
6	50 – 3,276,750	50	50
7	100 – 6,553,500	100	100

Reply:*Command is accepted and executed***Write:**

OK

*Command cannot be processed*An error message is returned. See [Error Codes](#)**Example:****Read:**SSPDMX ** returns the speed range value code (rmg – see Chart above)***Write:**SSPDM=5 ** Sets the speed range value to 20x magnification (rmg=5)***Notes:**

- The **SSPDM[axis]** setting will limit the maximum speed available for the **HSPD[axis]** according to the speed range table listed above. If the SSPDM is set for a lower speed range than the value for the HSPD command, the actual high-speed operation will be limited to the top value of the set range.
- The speed range must be set prior to issuing the SSPD command if the new speed value is in a different range than what is already set. Otherwise, an ?Error SSPD will be indicated.
- For CMD-4CR and CMD-4EX, V127 firmware, the **SSPDM[axis]** Speed Range is set to 1 by default, the **SSPDM[axis]** setting must be set to accommodate the desired speed value, or set to 0 for automatic setting.
- For all other products and firmware versions, the **SSPDM[axis]** Speed Range is set to 0 by default.
- Based on speed magnification resolution, actual speed can vary from commanded speed by

up to 0.1%.

See also:

[Motion Settings](#)

[ACC](#), [DEC](#), [HSPD](#), [JKA](#), [JKD](#), [LSPD](#), [PS](#), [SCV](#)

Commander Manual

[2.2.4 Positioning operation](#)

STOP – Controlled Stop

STOP – Controlled Stop

Decelerate and stop axis motion. This command is write-only. It is possible to stop each axis separately, or all axes at once.

Applies to: CMD-4CR, CMD-4EX-SA, CMD-2ED-SA, CMD-2EX

ASCII/Standalone	STOP	<i>global abort</i>
	STOP[axis]	<i>individual axis abort</i>

Syntax:

Write:

STOP[axis]

Where: [axis] is set to X, Y, Z, U, or not specified to affect all axes

Reply:

Command is accepted and executed

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

STOPX	<i>* Decelerate and stop X axis only</i>
STOP	<i>* Decelerate and stop all axes</i>

Notes:

- If an interpolation operation is in progress when a STOP[axis] or ABORT[axis] command is entered, all axes involved in the interpolation operation will stop.
- Stop of X, Y or Z axis disables buffer operation. Use BO command to turn buffer mode back on.
- Stop disables MPG (pulser) operation.
- Stop of X or Y axis disables Joystick operation, and resets the Joystick settings.

See also:

[ABORT](#), [ESTOP](#)

Commander Manual[2.1.1.1 Identification Number](#)**STORE – Save System Settings****STORE – Save System Settings**

Write-only command to save the setting value of various system settings to flash memory.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII/Standalone STORE *global acceleration setting*

Syntax:**Write:**

STORE

There is no parameter to specify.

The items shown in the table below are stored to flash by issuing the STORE command.

ASCII Commands	Description
	AntiCollision
DB	Serial communication baud rate
DIP	Digital input polarity
DOBOOT	DO configuration at boot-up
DOP	Digital output polarity
EINT	Linear interpolation feature
EOBOOT	EO configuration at boot-up
ERC[axis], ERCP[axis], ERCD[axis]	Deviation counter parameters
EXST	External start feature
ID	Identification number
IERR	Ignore Error State
INP[axis]	In position feature enable
IOBOOT	Configurable IO status at boot-up
IOCFG	IO configuration
IOP	Configurable IO polarity
JENA , JMAX[axis], JMIN[axis], JDEL[axis], JTOL[axis], JSPD[axis], JLIM[ch]	Joystick parameters
JKA[axis], JKD[axis]	Jerk Control / S-curve percentage
MPE[axis], MPM[axis], MPD[axis]	Manual pulsar parameters
POL[axis]	Polarity settings
SAP	Jump return line select (Standalone error handling)
SCV[axis]	S-curve enable
SDE[axis], SDC[axis]	Deceleration Signal parameters
SFL[axis], SFLSO[axis], SFLP[axis], SFLN[axis],	Soft-Limit parameters

SFLSD[axis]	
SL[axis], SLR[axis], SLE[axis], SLT[axis], SLA[axis]	StepNLoop parameters
SLOAD	Standalone program run on boot-up parameter
V0-V99	Standalone variables
ZCNT[axis]	

Reply:

Command is accepted and executed

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Write:

STORE * *Stores select system settings to flash memory.*

Notes:

- None

See also:

[Limits/Safety Setup](#), [System Setup](#), [I/O Setup](#),
[ERC](#), [ERCD](#), [ERCP](#), [INP](#), [POL](#), [SDC](#), [SDE](#), [EXST](#), [IERR](#)

Commander Manual

[2.1.1 Connect Multiple Commander Cores in a System Identification Number](#)
[2.1.3.2.2 High-Speed Digital Output Configurable Digital I/O](#)
[2.2.1 Motor Power](#)

SSTART – Simultaneous start

SSTART – Simultaneous start

Output one shot of the start pulse from the /CSTA terminal.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII

SSTART

Syntax:

Write:

SSTART

Reply:

Command is accepted and executed

Read:

Returns the value of the [axis] deviation counter (28-bit value)

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:

DECY * *Returns deceleration time set for the Y axis in ms*

Write:

DEC=300 * *Sets deceleration time to 300 ms for all axes.*

Notes:

- This command is not fully working yet Leave as "OUT of DATE" status

See also:

Commander Manual

[Identification Number](#)

SUB/ENDSUB – Subroutines

SUB/ENDSUB – Subroutines

The commands described here are only available when writing a standalone program.

Specifies the subroutine program number in a standalone program. Subroutines are independent program snippets that can be accessed based on the needs of the programmer. Typically, subroutines are used to perform functions that are repeated throughout the operation and can be used in conjunction with logic statements related to motor actions or the status of inputs and outputs.

Applies to: CMD-4CR, CMD-4EX-SA, CMD-2ED-SA, CMD-2EX-SA

Standalone	SUB #
	;subroutine program
	ENDSUB

Syntax:**Write:**

SUB [#]

Where: [#] is the subroutine number (0 - 31)

ENDSUB

Reply:*Command is accepted and executed***No Reply****Example:****Write:**

```

SUB 1  * Start subroutine program 1
X1000  * Move to 1000 target position (for X axis)
WAITX  * Wait for X axis move to complete
ENDSUB * End subroutine program 1.

```

Notes:

- Up to 32 (0-31) subroutines can be called upon.
- Subroutine 31 is used specifically for error handling and is executed whenever an error occurs.
- Subroutines can be shared by both standalone programs and the ASCII command set. Once a subroutine is written into the flash memory, it can be recalled via ASCII communication using the **GS** command. Standalone programs can also jump to a subroutine using the **GOSUB** command.
- If a subroutine number is not defined, the controller will return with an error.

See also:[Standalone Programming Commands](#)**Commander Manual**[Identification Number](#)**SYNC – Synchronization Mode****SYNC – Synchronization Mode**

Returns or sets the value of the synchronization mode (condition and source). Bit 0-3 indicates the synchronization type and condition; Bit 4 indicates the source used for the counter encoder/pulse position that is to be used to trigger the synchronization output. The [SYNO\[axis\]](#), [SYNON\[axis\]](#), or [SYNWO\[axis\]](#) command must be executed for the command to be sent from local memory to the ASIC, in both Standalone and ASCII operations.

[Applies to: CMD-4CR, CMD-4EX-SA](#)

ASCII

SYNC[axis]

Standalone

SYNCFG[axis]

Syntax:

Read:

SYNC[axis]

SYNCFG[axis]

Where: [axis] is set to X, Y, Z, U

Write:

SYNC[axis]=[n]

SYNCFG[axis]=[n]

Where: [axis] is set to X, Y, Z, U

[n] is the decimal equivalent of the 5-bit number for the synchronization mode. See the chart below.

Bit	Description		Decimal	
			Pulse Position	Encoder Position
0-3	Synchronization Condition:		From sync source:	
	0001	Trigger when sync position = actual position. Any direction	1	17
	0010	Trigger when sync position = actual position. Positive direction	2	18
	0011	Trigger when sync position = actual position. Negative direction	3	19
	0100	Trigger when sync position > actual position	4	20
	0101	Trigger when sync position < actual position	5	21
	0111	Reserved/not available		
	1000	Trigger every sync position. Any direction	8	24
	1001	Trigger every sync position. Positive motion	9	25
	1010	Trigger every sync position. Negative motion	10	26
4	Synchronization Source:			
	0	Sync position is compared to pulse position		
	1	Sync position is compared to encoder position		

Reply:

Command is accepted and executed

Read:

Returns the decimal equivalent of the 4-bit number for the synchronization mode; see chart above.

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:

SYNCY * Returns current synchronization mode for the Y axis in decimal

Write:

ASCII

Example: Turning on synchronization output

SYNPX=4 * Sets target synchronization position to 4 on X-axis

SYNCX=1 * Sets synchronization mode to at-position, in any direction, on X-axis, based on pulse position

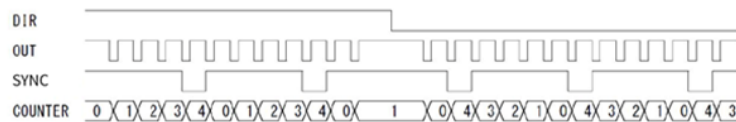
SYNOX * Writes data to ASIC then turns on synchronization for the X-axis

Example: Making changes to synchronization mode with an output timing chart

SYNCX=8 * Sets sync mode to continuous, in any direction, on X-axis, based on pulse position

SYNOX * Turns on synchronization for the X-axis

Output based on the above two examples (SYNPX=4 and SYNCX=8):



Example: Making changes to synchronization mode

SYNCX=3 * Sets synchronization mode to at-position, in the negative direction, on X-axis, based on pulse position

SYNOX * Writes new values to the ASIC for the X-axis

Standalone

Example: Turning on synchronization output

SYNPOSZ=700 * Sets target synchronization position to 700 on Z-axis

SYNCFGZ=8 * Sets synchronization mode to continuous, in any direction, on Z-axis, based on pulse position

SYNONZ * Writes data to ASIC then turns on synchronization for the Z-axis

Example: Making changes to synchronization mode

SYNCFGZ=24 * Sets synchronization mode to continuous, in any direction, on Z-axis, based on encoder position

SYNONZ * *Writes new values to the ASIC for the Z-axis*

Notes:

- The synchronization mode for each axis is read and set independently.
- The synchronization digital output will remain ON as long as the comparison condition is true.

See also:

[I/O Setup](#),

[SYNF](#), [SYNO](#), [SYNP](#), [SYNS](#), [SYNWO](#), [SYNWF](#), [SYNMAX](#), [SYNMIN](#)

Commander Manual

[2.1.3.2.2 High-Speed Digital Output](#)

SYNF – Synchronization mode off

SYNF – Synchronization mode off

Disables the synchronization output for an axis. If the synchronization window function was active, it is also disabled.

Applies to: CMD-4CR, CMD-4EX-SA, PMX-4EX-SA, PMX-4ET-SA, PMX-2ED-SA, PMX-2EX-SA

ASCII	SYNF[axis]
Standalone	SYNOFF[axis]

Syntax:

Write:

SYNF[axis]

Where: [axis] is set to X, Y, Z, U

Reply:

Command is accepted and executed

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Write:

SYNFX * *Disables synchronization window and synchronization output on the X-axis.*

Notes:

- The synchronization function for each axis is independent.

See also:

[I/O Setup](#),

[SYNC](#), [SYNO](#), [SYNP](#), [SYNS](#), [SYNWO](#), [SYNWF](#), [SYNMAX](#), [SYNMIN](#)

Commander Manual

[2.1.3.2.2 High-Speed Digital Output](#)

SYNBO – Sync Position Buffer Enable

SYNBO – Sync Position Buffer Enable

Write-only command that enables the sync target position buffer operation.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII SYNBO

Syntax:

Write:

SYNBO

Reply:

Command is accepted and executed

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Write: SYNBO * *Enables the Sync position buffer operation*

Notes:

When this command is sent, the Sync position buffer is reset and cleared.

See also:

[Buffer Operation](#)

[SYNBF](#), [SYNP](#), [SYNBSTAT](#)

Commander Manual

[Identification Number](#)

SYNBF – Sync Position Buffer Disable

SYNBF – Sync Position Buffer Disable

Write-only command that disables the sync target position buffer operation.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII SNYBF

Syntax:

Write:

SYNBF

Reply:

Command is accepted and executed

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Write: SYNBF * *Disables the Sync position buffer operation*

Notes:

When this command is sent, the Sync position buffer is reset and cleared.

See also:

[Buffer Operation](#)

[SYNBO](#), [SYNP](#), [SYNBSTAT](#)

Commander Manual

[2.1.1.1 Identification Number](#)

SYNBSTAT – Buffer Status

SYNBSTAT – Buffer Status

A read-only command that returns the current status of the sync position buffer operation.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII SYNBSTAT

Syntax:

Read:

SYNBSTAT

Reply

Command is accepted and executed

Read:

Three integers in the format "%d:%d:%d" [buffer enabled]:[buffer positions used]:[available sync buffer positions]

Where: [buffer enabled] – indicates the enabled or disabled state (0 = disabled, 1 = enabled),
[buffer positions used] – indicates the current number of

positions used of the buffer (0 – 1024)
 [available sync buffer positions] – indicates the total number of
 Sync buffer positions (1024)

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read: SYNSTAT ** returns the sync position buffer status*
 Reply: 1:34:1024 ** signifies buffer is enabled, there are 34 sync positions stored in the buffer, there are a total of 1024 positions in the buffer*

Notes:

None

See also:

[Buffer Operation](#)
[SYNBO](#), [SYNBF](#), [SYNP](#)

Commander Manual

[Identification Number](#)

SYNMAX – Synchronization Window Upper Limit

SYNMAX – Synchronization Window Upper Limit

Reads the value of the synchronization window upper limit. This will indicate the maximum encoder/pulse position the synchronization digital output will activate during continuous synchronization mode. Set the synchronization window upper limit by using SYNMAX[axis]=[n]. Any position larger than the set SYNMAX[axis] will not output synchronization pulses. The default value for SYNMAX[axis] is 0.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII SYNMAX[axis]

Syntax:

Read:

SYNMAX[axis]

Where: [axis] is set to X, Y, Z, U

Write:

SYNMAX[axis]=[n]

Where: [axis] is set to X, Y, Z, U

[n] is the target value

** Valid values for n target value*

Any value inside of the positioning control range (-

134,217,728 to +134,217,727) (28bit)
 SYNMAX[axis] must be larger than SYNMIN[axis]

Reply:

Command is accepted and executed

Read:

Returns the current set value for the sync window upper limit (28-bit value)

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Enable the synchronization window and set min/max values.

SYNPX=1000 * 1000 counts between synchronization output pulses on the X-axis.

SYNCX=8 * Sets sync mode to continuous for X-axis, based on pulse position.

SYNMAXX=10000 * Sets the maximum value for the synchronization window to 100,000 on the X-axis.
 0

SYNMINX=2000 * Sets the minimum value for the synchronization window to 2000 on the X-axis.

SYNWOX * Enables sync output and synchronization window on the X-axis.

Notes:

- The synchronization window lower limit function is independent for each axis.
- The synchronization window function overlays the continuous synchronization mode. An error message will appear if the synchronization mode ([SYNC\[axis\]](#)) is not set to one of the continuous modes.
- The value set to SYNMAX[axis] must be within the positioning control range (-134,217,728 to +134,217,727) 28-bit
- If SYNMAX[axis] will be set to a negative value, [SYNMIN\[axis\]](#) should be set first.

See also:

[I/O Setup](#),

[SYNC](#), [SYNE](#), [SYNO](#), [SYNP](#), [SYNS](#), [SYNWO](#), [SYNWE](#), [SYNMIN](#)

Commander Manual

[2.1.3.2.2 High-Speed Digital Output](#)

SYNMIN – Synchronization Window Lower Limit

SYNMIN – Synchronization Window Lower Limit

Reads the value of the synchronization window lower limit. This will indicate the minimum encoder/pulse position to which the synchronization pulse can be output during continuous synchronization mode. Synchronization pulse will not be output at encoder/pulse position with values lower than the set minimum (SYNMIN[axis]). The default value for SYNMIN[axis] is 0. Set

the synchronization window lower limit by using SYNMIN[axis]=n.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII SYNMIN[axis]

Syntax:

Read:

SYNMIN[axis]

Where: [axis] is set to X, Y, Z, U

Write:

SYNMIN[axis]=[n]

Where: [axis] is set to X, Y, Z, U

[n] is the target value

** Valid values for n target value*

Any value inside of the positioning control range (-134,217,728 to +134,217,727) (28bit)

SYNMIN[axis] must be smaller than SYNMAX[axis]

Reply:

Command is accepted and executed

Read:

Returns the current set value for the sync window lower limit (28-bit value)

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Enable the synchronization window and set min/max values.

SYNPX=1000 ** 1000 counts between synchronization output pulses on the X-axis.*

SYNCX=8 ** Sets sync mode to continuous for X-axis, based on pulse position.*

SYNMAXX=10000 ** Sets the maximum value for the synchronization window to 100,000 on the X-axis.*

SYNMINX=2000 ** Sets the minimum value for the synchronization window to 2000 on the X-axis.*

SYNWOX ** Enables sync output and synchronization window on the X-axis.*

Notes:

- The synchronization window lower limit function is independent for each axis.
- The synchronization window function overlays the continuous synchronization mode. An error message will appear if the synchronization mode ([SYNC\[axis\]](#)) is not set to one of the

continuous modes.

- The value set to SYNMIN[axis] must be within the positioning control range (-134,217,728 to +134,217,727) 28-bit
- If SYNMIN[axis] is set to a positive value, [SYNMAX\[axis\]](#) should be set first.

See also:

[I/O Setup](#),

[SYNC](#), [SYNF](#), [SYNO](#), [SYNP](#), [SYNS](#), [SYNWO](#), [SYNWF](#), [SYNMAX](#)

Commander Manual

[2.1.3.2.2 High-Speed Digital Output](#)

SYNO – Synchronization mode on

SYNO – Synchronization mode on

When executed, the data for synchronization mode is written from local memory to the ASIC to activate the synchronization mode. These commands are also used when changes are made to synchronization mode to move the values from local memory to the ASIC in both standalone and ASCII operations.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII	SYNO[axis]
Standalone	SYNON[axis]

Syntax:

Write:

SYNO[axis]

Where: [axis] is set to X, Y, Z, U

Reply:

Command is accepted and executed

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

ASCII

Turn on synchronization output.

SYNPX=1000

** Sets target synchronization position to 1000 on X-axis.*

SYNCX=1

** Sets synchronization mode to at-position, in any direction, on X-axis, base position.*

Make changes to synchronization mode.

SYNCX=3

** Sets synchronization mode to at-position, in negative direction, on X-axis,*

pulse position

SYNOX ** Writes new values to the ASIC for the X-axis.*

Standalone

Turn on synchronization output.

SYNPOSZ=700 ** Sets target synchronization position to 700 on Z-axis.*

SYNCFGZ=8 ** Sets synchronization mode to continuous, in any direction, on Z-axis, based on encoder position.*

SYNONZ ** Writes data to ASIC then turns on synchronization for the Z-axis.*

Make changes to synchronization mode.

SYNCFGZ=24 ** Sets synchronization mode to continuous, in any direction, on Z-axis, based on encoder position.*

SYNONZ ** Writes new values to the ASIC for the Z-axis.*

Notes:

- The synchronization function for each axis is independent.
- Manual pulsar function must be disabled to use the synchronization mode. ([MPE\[axis\]](#))

See also:

[I/O Setup](#),

[SYNC](#), [SYNE](#), [SYNP](#), [SYNS](#), [SYNWO](#), [SYNWE](#), [SYNMAX](#), [SYNMIN](#)

Commander Manual

[2.1.3.2.2 High-Speed Digital Output](#)

SYNP – Synchronization Position

SYNP – Synchronization Position

Returns the value of the comparator, indicating the encoder/pulse position used to trigger the synchronization output. The default value of SYNP[axis] / SYNPOS[axis] is 0. The synchronization positioning commands can be used to set the comparator value by using SYNP[axis]=[n] or SYNPOS[axis]=[n]. Up to 1024 synchronization positions can be buffered using the Sync Position Buffer.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII SYNP[axis]

Standalone SYNPOS[axis]

Syntax:

Read:

SYNP[axis]

Where: [axis] is the axis (X, Y, Z, U) being addressed

Write:

SYNP[axis]=[pos]

Where: [axis] is the axis (X, Y, Z, U) being addressed
[pos] is the target value for the sync

Valid values for [pos] target value

When synchronization mode is at-position or >, < position:

Any value inside of the positioning control range (-134,217,728 to +134,217,727) (28bit)

Target [pos] value is the absolute value relative to the position counter (P[axis]) or encoder counter (E[axis]).

When synchronization mode is continuous:

A positive value inside of the positioning control range (+1 to +134,217,727)

Target [pos] is a relative value equal to the counts between synchronization output pulses. The counter is set to zero (0) based on the zero (0) position of the position counter (P[axis]) or encoder counter (E[axis]). It will count from zero (0) to the value set by [pos] at which point the synchronization output pulse is triggered. The next step will set the counter to zero (0) and continue to count up.

Reply

Command is accepted and executed

Read:

Returns current target value set for synchronization mode (28-bit value)

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

ASCII

Request the current synchronization target value and set it to a new value.

SYNPY * *Request the current synchronization target value for the Y-axis.*

SYNPZ=25000 * *Sets the synchronization target value for the Z-axis to 25,000.*

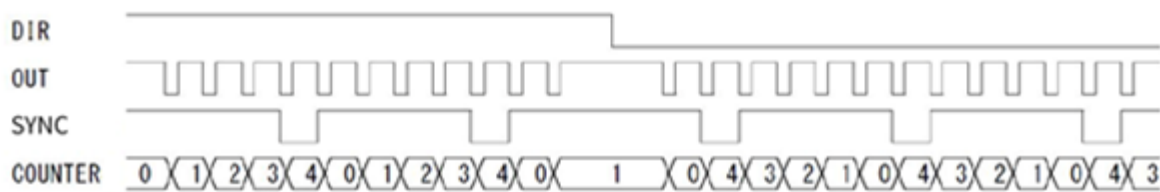
Set synchronization target value with the output timing chart.

SYNPX=4 * *Sets the synchronization target value for the X-axis to 4.*

SYNCX=8 * *Sets sync mode to continuous, in any direction, on X-axis, based on pulse*

SYNOX * *Turns on synchronization for the X-axis.*

Output based on this example (SYNPX=4 and SYNCX=8):



Standalone

Request the current synchronization target value and set to a new value.

V1=SYNPOSX * Variable V1 is set to the current synchronization target value for X-axis.
SYNPOSU=800 * Sets the synchronization target value for the U-axis to 800.

Notes:

- The synchronization comparator position value for each axis is read and set independently.
- Changes to the target value set by SYNO[axis] (SYNON[axis]) take effect in real-time. They are referenced to the zero (0) position of the position counter (P[axis]) or encoder counter (E[axis]).
- The target value set to SYNO[axis] (SYNON[axis]) must be within the positioning control range 28bit (-134,217,728 to +134,217,727) except when synchronization is set to continuous mode, in this case SYNO[axis]/SYNON[axis] must be a positive value (+1 to +134,217,727)

See also:

[I/O Setup](#),
[SYNC](#), [SYNE](#), [SYNO](#), [SYNS](#), [SYNWO](#), [SYNWF](#), [SYNMAX](#), [SYNMIN](#)

Commander Manual

[2.1.3.2.2 High-Speed Digital Output](#)

SYNS – Synchronization Status

SYNS – Synchronization Status

A read-only command that returns the current status of the synchronization output operation.

[Applies to: CMD-4CR, CMD-4EX-SA](#)

ASCII SYNS[axis]
Standalone SYNSTAT[axis]

Syntax:

Read:

SYNS[axis]
Where: [axis] is the axis (X, Y, Z, U) being addressed

Reply

Command is accepted and executed

Read: Returns 2-bit synchronization status. Valid reply are 0, 1, or 3.

Bit	Description	Setting	
0	Sync Operation	0 – Disabled	1 – Enabled
1	Sync Window	0 – Disabled	1 – Enabled

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read: SYNSX ** returns the status of the synchronization output operation for the X axis*

Reply: 3 ** signifies the sync operation and the sync window are enabled for the X axis*

Read: SYNSZ ** returns the status of the synchronization output operation for the Z axis*

Reply: 1 ** signifies the sync operation is enabled for the Z axis*

Notes:

- Synchronization status will return 1 as long as synchronization mode is active for that axis.
- The synchronization status for each axis is independent.

See also:

[I/O Setup](#),

[SYNC](#), [SYNF](#), [SYNO](#), [SYNP](#), [SYNWO](#), [SYNWF](#), [SYNMAX](#), [SYNMIN](#)

Commander Manual

[2.1.3.2.2 High-Speed Digital Output](#)

SYNWF – Synchronization Window Off**SYNWF – Synchronization Window Off**

Disables the synchronization window function. The synchronization function is not disabled, and the synchronization pulse will continue to be output whenever the synchronization conditions are met, without limitation to position. There is no loss of synchronization from one pass to the next, and the synchronization function will continue until [SYNF\[axis\]](#) is executed.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII SYNWF[axis]

Syntax:**Write:**

SYNWF[axis]

Where: [axis] is the axis (X, Y, Z, U) being addressed

Reply:*Command is accepted and executed***Write:**

OK

*Command cannot be processed*An error message is returned. See [Error Codes](#)**Example:****Write:**SYNWFX ** Disables only sync window on the X-axis.***Notes:**

- The synchronization window function for each axis is independent.

See also:[I/O Setup](#),SYNC, [SYNF](#), [SYNO](#), [SYNP](#), [SYNS](#), [SYNWO](#), [SYNMAX](#), [SYNMIN](#)**Commander Manual**[2.1.3.2.2 High-Speed Digital Output](#)**SYNWO – Synchronization Window On****SYNWO – Synchronization Window On**

In continuous mode, the synchronization window function allows the synchronization function to run continuously, but only output synchronization pulses when the counter is between the synchronization maximum and minimum points. When the synchronization function is enabled, there is no loss of synchronization from one pass to the next.

When SYNWO[axis] is executed, it first confirms that synchronization mode is set to continuous in local memory. The data for synchronization mode is written from local memory to the ASIC, synchronization mode is activated, and the synchronization window function is enabled.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII SYNWO

Syntax:**Write:**

SYNWO[axis]

Where: [axis] is the axis (X, Y, Z, U) being addressed

Reply:

Command is accepted and executed

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:**Write:**

SYNPX=1000 * 1000 counts between synchronization output pulses on the X-axis.

SYNCX=8 * Sets sync mode to continuous for X-axis, based on pulse position.

SYNMAXX=10000 * Sets the maximum value for the synchronization window to 100,000 on the X-axis.
0

SYNMINX=2000 * Sets the minimum value for the synchronization window to 2000 on the X-axis.

SYNWOX * Enables sync output and synchronization window on the X-axis

Notes:

- The synchronization window function for each axis is independent.
- The synchronization window function overlays the continuous synchronization mode. An error message will appear if the synchronization mode (SYNC[axis]) is not set to one of the continuous modes.
- When the synchronization window is enabled, it will watch the referenced position (pulse position or encoder position, as set in Bit 4 of SYNC[axis]).
- The manual pulsar function must be disabled to use the synchronization mode. (MPE[axis]).

See also:

[I/O Setup](#),

[SYNC](#), [SYNF](#), [SYNO](#), [SYNP](#), [SYNS](#), [SYNWF](#), [SYNMAX](#), [SYNMIN](#)

Commander Manual

[2.1.3.2.2 High-Speed Digital Output](#)

T – On-the-Fly Target Position Change**T – On-the-Fly Target Position Change**

Write-only command to change the target position while the axis is moving.

Applies to: CMD-4CR, CMD-4EX-SA, PMX-4EX-SA, PMX-4ET-SA, PMX-2ED-SA, PMX-2EX

ASCII

T

Syntax:**Write:**

T[axis][newtarget]

Where: [axis] specifies the axis to send a new target position
(X, Y, Z, U)
[newtarget] specifies the target position, or increment to move

Reply:

Command is accepted and executed

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Write:

TX-1000 ** changes the target position of the X axis to -1000 in absolute mode, or changes the incremental move to be -1000 pulses from the original start position.*

Notes:

- The target position change command will only be valid if the axis is currently performing an individual position move. If the new target position has been passed, the axis will reverse direction once the target position command is entered. If the absolute position has passed in absolute mode, the motor will reverse and go to new target position. If the increment of motion has been exceeded in the incremental mode, the motor will reverse and move to the correct increment from the start position.
- A jogging, interpolated, or homing routine will not accept a target position change command.

See also:

[Axis Motion](#)
[SSPD](#), [SSPDM](#)

Commander Manual

[2.2.4 Positioning operation](#)
[2.4.2 On-the-fly target position change \(Target position change\)](#)

V – Variables

V – Variables

Command used to set or return the value of a variable. When used in standalone programming, variables can also be used in mathematical operations.

Applies to: CMD-4CR, CMD-4EX-SA, CMD-2ED-SA, CMD-2EX

ASCII/Standalone

V[var]

Syntax:**Read:**

V[var]

Where: [var] specifies variable number (0-99 for 4-axis controllers, 0-64 for 2-axis controllers)

Write:

V[var]=[value]

Where: [var] specifies variable number (0-99 for 4-axis controllers, 0-64 for 2-axis controllers)
[value] specifies the set value for the variable, 32-bit value range**Reply:***Command is accepted and executed***Read:**

Return the standalone variable value

Write:

OK

*Command cannot be processed*An error message is returned. See [Error Codes](#)**Example:****Read:**V12 ** Returns the value of variable V12***Write:**V88=1000 ** Sets the variable V88 to equal 1000**Applies to: Standalone mode only*V3=V3+7 ** Sets the value of V3 to the current value of V3 + 7***Notes:**

- Variables can be set to specific values between -2,147,483,648 and +2,147,483,647
- Variables can be used with the following mathematical expressions. The syntax for all available operations can be found below.

Operator	Description	Example
+	Integer Addition	V1=V2+V3
-	Integer Subtraction	V1=V2-V3
*	Integer Multiplication	V1=V2*V3

/	Integer Division (round down)	$V1 = V2 / V3$
%	Modulus	$V1 = V2 \% 5$
>>	Bit Shift Right	$V1 = V2 >> 2$
<<	Bit Shift Left	$V1 = V2 << 2$
&	Bitwise AND	$V1 = V2 \& 7$
	Bitwise OR	$V1 = V2 8$
~	Bitwise NOT	$V1 = \sim V2$

Operations

- Note that mathematical operations can only be performed in standalone programming.

Applies to: CMD-4CR, CMD-4EX-SA V303BL+

- Variables V50 through V99 can be stored to flash memory using the STORE command
- Variables V0-V49 will be initialized to zero on power up

Applies to: CMD-4CR, CMD-4EX-SA V1xx, V2xx

- Variables V0 through V99 can be stored to flash memory using the STORE command

See also:

[Standalone Programming](#)

[SACTRL](#), [SAP](#), [SASTAT](#), [SLOAD](#), [GS](#), [SPC](#)

Commander Manual

[Identification Number](#)

VEC– Vector Motion Enable

VEC– Vector Motion Enable

Write-only command that enables the vector motion operation.

- 3-axis buffer mode is available in all 4axis and 8 axis versions of the Commander.
- 4-axis vector (linear) buffer mode is available in all 4axis and 8 axis versions of the commander with V202 firmware or newer.
- 2-axis vector (linear) buffer mode is available in all Gen 5 versions of the commander with V301 firmware or newer.
- 6-axis and 8-axis vector

Applies to: CMD-4CR, CMD-4EX-SA

ASCII BO

Standalone BUFON

Syntax:

Applies to: CMD-4CR, CMD-4EX-SA

Write:

BO

Applies to: CMD-4CR, CMD-4EX-SA with V202 or newer firmware

Write:

BO=[buffer mode]

Where: [buffer mode] specifies the buffer mode 1 = 3 axis buffer mode, and 2 = 4 axis in vector (linear) buffer mode, for backwards compatibility, when only BO is sent, 3 axis buffer mode is enabled.

Reply:

Command is accepted and executed

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Write: BO ** Enables the 3 axis buffer operation*

Applies to: CMD-4CR, CMD-4EX-SA with V202 or newer firmware

Write:

BO=2 ** Enables the 4 axis buffer operation*

Notes:

- In 3-axis buffer mode - axis X,Y, and Z are used and supports, [ARC](#), [ARCT](#), [CIR](#), [CIRT](#), and the 3 axis version of the [I](#) command. In this mode axis U is not available for any motion.
- In 4 axis vector (linear) buffer mode only the 4 axis version of the [I](#) command can be buffered.
- With V202 firmware for the CMD-4CR and CMD-4EX-SA a 4 axis in vector (linear) buffer mode, was added. To access the new functionality add BO=2 to enable 4 axis in vector (linear) buffer mode.
- BO still works as originally designed as well as BO=1 for 3 axis buffer mode.
- BF must be called between moving from 3 to 4 axis or vice versa.
- In 4 axis vector (linear) buffer mode only the 4 axis version of the [I](#) command can be buffered.
-
- BF, BSTAT, and BSTART all operate the same for the 3 axis and 4 axis version of the buffer command.

See also:

[Buffer Operation](#)

[BF](#), [I](#), [BSTAT](#), [BSTART](#), [IACC](#)

Commander Manual

[2.3.4 Buffered interpolation operation](#)

VER – Firmware Version

VER – Firmware Version

Read-only command to return the current firmware version information.

Applies to: CMD-4CR, CMD-4EX-SA, PMX-4EX-SA, PMX-4ET-SA, PMX-2ED-SA, PMX-2EX

ASCII VER

Syntax:

Read:

VER

Reply:

Command is accepted and executed

Read:

Returns the current firmware version in the following formats

Applies to: CMD-4CR, CMD-4EX-SA

Three-digit version number is returned

Applies to: PMX-4EX-SA, PMX-4ET-SA, PMX-2ED-SA, PMX-2EX

V followed by three-digit version number

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:

VER	<i>* Requested the firmware version number</i>
201	<i>* Returned current firmware version 2.01</i>

Notes:

- Three-digit number is returned where 201 means version 2.01

See also:

[System Setup](#)

[ID, DB, IP](#)

Commander Manual

[2.1.1 Connect Multiple Commander Cores in a System](#)

WAIT – Standalone wait command

WAIT – Standalone wait command

The commands described here are only available when writing a standalone program.

In standalone programs, this command will have the program wait for the current motion to complete before processing the next line.

Applies to: CMD-4CR, CMD-4EX-SA, CMD-2ED-SA, CMD-2EX

Standalone WAIT[axis]

Syntax:

Write:

WAIT[axis]

Where: [axis] is set to X, Y, Z, U

Reply:

Command is accepted and executed

No Reply

Example:

Write:

X1000 ** move X axis to position 1000.*

WAITX ** wait for X axis to complete move before going to next command.*

Notes:

- None

See also:

[Standalone Programming Commands](#)

Commander Manual

[Identification Number](#)

WHILE/ENDWHILE – Conditional loop

WHILE/ENDWHILE – Conditional loop

The commands described here are only available when writing a standalone program.

Runs a series of statements as long as a given condition is True.

Applies to: CMD-4CR, CMD-4EX-SA, CMD-2ED-SA, CMD-2EX-SA

Standalone WHILE
ENDWHILE

Syntax:

Write:

WHILE [condition]
[statement]
ENDWHILE

Where: [condition] Required. Expression. Must evaluate to True or False using a [relational operator](#).
[statement] One or more statements to be executed while
[condition] is TRUE

Reply:

Command is accepted and executed

No Reply

Example:

Write:

```
WHILE V1<10 * Loop while variable 1 is less than 10
X1000      * Move to 1000 (for X-axis)
WAITX      * Wait for X-axis move to complete
X0         * Move to zero (for X-axis)
WAITX      * Wait for X-axis move to complete
V1=V1+1    * Incremental variable 1
ENDWHILE   * Go back to WHILE statement
```

Notes:

- None

See also:

[Standalone Programming Commands](#)

Commander Manual

[Identification Number](#)

TODO: See also and Commander Manual links

X, Y, Z, U – Positioning Function

X, Y, Z, U – Positioning Function

This write-only command initiates a move of the specified axis, either as an absolute move to a target position, or an incremental move, depending on the coordinate designation (set by MM for incremental or absolute mode).

Applies to: CMD-4CR, CMD-4EX-SA, CMD-2ED-SA, CMD-2EX

ASCII/Standalone

[axis][pos]

Syntax:

Write:

[axis][pos]

Where: [axis] specifies the axis to initiate a move (X, Y, Z, U)
 [pos] for MM=0, specifies target position (counter value, 28-bit value)
 [pos] for MM=1, specifies incremental movement (pulse counts, 28-bit value)

When EINT is set to 1, an interpolated move can be written as: **X[x target]Y[y target]Z[z**

target]U[u target] using any two (or more) axes.

2 - axis linear interpolation
[axis][pos][axis][pos]

3 - axis linear interpolation
[axis][pos][axis][pos][axis][pos]

4 - axis linear interpolation
[axis][pos][axis][pos][axis][pos][axis][pos]

Reply:

Command is accepted and executed

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Write:

X20000 ** Start single axis move for X axis in the positive direction for 20000 pulses when MM=1*

X200Y100Z500 ** Start 3 - axis linear interpolation move for X, Y, and Z axis to target position counter X200 , Y100, Z500 when MM=0*

Notes:

- MM is set to 0 or 1 by issuing an ABS or INC command (0 = absolute, 1 = incremental)

See also:

[Axis Motion](#)

[J](#), [EO](#), [EOBOOT](#), [EINT](#)

Commander Manual

[2.2.4 Positioning operation](#)

[2.3 Interpolation Operation](#)

[2.3.1 Linear interpolation](#)

[2.3.4 Buffered interpolation operation](#)

ZCNT – Z-index pulse counter

ZCNT – Z-index pulse counter

Identifies the number of Z-index pulses to count when using the the ZMOVE, or Homing commands.

The ZCNT[axis] command also works with any homing operation that uses the Z-index pulses as part of the homing operation to set the number of Z-index pulses that need to be detected.

The range for ZCNT[axis] is 1-16. The number of Z-index pulses to be detected will be the number set.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII/Standalone ZCNT[axis]

Syntax:

Read:

ZCNT[axis]

Where: [axis] is set to X, Y, Z, U

Write:

ZCNT[axis]=[count]

Where: [axis] is set to X, Y, Z, U

[count] specifies the number of Z-pulses to count. Setting range (1 – 16)

Reply:

Command is accepted and executed

Read:

Returns the value of the [axis] counts.

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:

ZCNTX ** Returns the value of Z-index count for the X-axis*

Write:

ZCNTX=1 ** Sets the Z-index count to stop on the 1st Z-index pulse detection for the X axis*

Notes:

- Any value entered for [count] less than 1 will return a 1, and any value entered greater than 16 will return a 16.

Operating procedure

If there is no change in the previous set data, the operations 1 and 2 below are not necessary.

- Set the operating speeds
- Set the Z-index count numbers and the Z-index input logic
- Issue a homing command

Operating conditions

- Joystick motion is disabled
- MPG operation is disabled
- Operation is stopped
- The error status is clear

See also:

[Homing Settings](#)

[H, ZMOVE](#)

Commander Manual

[2.2.5.1 Z-move Operation](#)

ZMOVE – Z-move Operation**ZMOVE – Z-move Operation**

Write-only command, where when the ZMOVE command is sent, the selected axis will move in the direction commanded at [HSPD](#) until such time as the number of Z-index pulses are detected as set by the [ZCNT](#) command.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII/Standalone

ZMOVE[axis][dir]

Syntax:**Write:**

ZMOVE[axis][dir]

Where: [axis] specifies the axis to move X, Y, Z, U

[dir] specifies the move direction (+, -)

+ moves in the direction where the command

pulse counter counts up

- moves in the direction where the command

pulse counter counts down

Reply:

Command is accepted and executed

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:**Write:**

HSPD=1000 * *sets motion target speed to 1000 pps*

LSPD=100 * *sets motion start speed to 100 pps*

ZCNTZ=3 * *Sets the Z-index count to stop on the 3rd Z-index pulse detection for the Z-axis*

POLZ=8192 * Sets the input modes and logic based on the 17-bit value.
 bit 13 is 1 setting logic of Z-index to Rising Edge
 ZMOVEZ- * move the Z axis in the negative direction until the number
 of encoder index pulses set in ZCNT (3) are detected.

Notes:

Operating procedure

If there is no change in the previous set data, the operations 1 and 2 below are not necessary.

1. Set the operating speeds (HSPD, LSPD)
2. Set the Z-index count numbers (ZCNT) and the Z-index input logic (POL bit 13)
3. Issue ZMOVE command

Operating conditions

- o Joystick motion must be disabled
- o MPG operation must be disabled
- o Buffer operation must be disabled
- o Operation is stopped
- o The error status is clear

See also:

[Homing Settings](#)
[H, ZCNT](#)

Commander Manual

[2.2.5.1 Z-move Operation](#)

%RST – Soft reset

%RST – Soft reset

Performs a soft reset of the Commander motion controller.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII %RST

Syntax:

Write:

%RST

Reply:

Command is accepted and executed

Write:

None

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:**Write:**

%RST * *Request soft reset of Commander.*

Notes:

- None

See also:

[System Setup](#)

Commander Manual

[Identification Number](#)

TODO: See also and Commander Manual links

Unpublished

The following commands are unpublished and intended for use only in Nippon Pulse software, or by Nippon Pulse personnel. Service commands are only supported via ASCII commands.

These commands should be marked "SERVICE" green after reviewed.

%BLD – Enter boot-load mode

Sets the deceleration time or acquires the current set value. Deceleration time is available as a global value for all axes and a value settable for each axis

Applies to: CMD-4CR, CMD-4EX-SA, PMX-4EX-SA, PMX-4ET-SA, PMX-2ED-SA, PMX-2EX

ASCII/Standalone	decel	<i>global acceleration setting</i>
	decel[axis]	<i>individual axis acceleration setting</i>

Syntax:**Read:**

decel[axis]

Where: [axis] is set to X, Y, Z, U, or not specified to read global value

Write:

decel[axis]=[accel]

Where: [axis] is set to X, Y, Z, U, or not specified to read global value
[accel] is the deceleration time in ms (0 – 65,535)

Reply:

Command is accepted and executed

Read:

Returns the value of the [axis] deviation counter (28-bit value)

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:
 DECY ** Returns deceleration time set for the Y axis in ms*

Write:
 decel=300 ** Sets deceleration time to 300 ms for all axes.*

Notes:

- Deceleration is time entered in milliseconds. Deceleration rate in pps² (pulses/second²) can be calculated with the following formula.

$$\text{Deceleration rate} = \frac{\text{HSPD} - \text{LSPD}}{\text{ACC}/1000}$$

- The minimum and maximum deceleration values depend on the high-speed and low-speed settings. Refer to [Speed Settings](#) for details.
- Individual deceleration settings will take priority over the global deceleration settings. A global deceleration setting will only be used when, either one of the following conditions is true:
 - The corresponding individual deceleration setting is defined as zero.
 - A coordinated motion is being performed.

In the example below, the settings will set both the global speed settings as well as the individual speed setting for the X axis.

```
HSPD=10000
HSPDX=2000
LSPD=300
accelX=500
accel=300
decel=300
```

Once a move command is issued in this example, the global speed settings for the low speed and deceleration, and the individual X-axis speed settings for the high speed and acceleration, will be used for motion.

See also:

[Motion Settings](#)
[accel](#), [SCV](#), [HSPD](#), [LSPD](#)

Commander Manual

[2.1.1.1 Identification Number](#)

%DID – SOC TYPE ID

Sets the deceleration time or acquires the current set value. Deceleration time is available as a global value for all axes and a value settable for each axis

Applies to: CMD-4CR, CMD-4EX-SA, PMX-4EX-SA, PMX-4ET-SA, PMX-2ED-SA, PMX-2EX

ASCII/Standalone decel *global acceleration setting*
 decel[axis] *individual axis acceleration setting*

Syntax:

Read:

`decel[axis]`

Where: `[axis]` is set to X, Y, Z, U, or not specified to read global value

Write:

`decel[axis]=[accel]`

Where: `[axis]` is set to X, Y, Z, U, or not specified to read global value
`[accel]` is the deceleration time in ms (0 – 65,535)

Reply:

Command is accepted and executed

Read:

Returns the value of the `[axis]` deviation counter (28-bit value)

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:

DECY ** Returns deceleration time set for the Y axis in ms*

Write:

`decel=300` ** Sets deceleration time to 300 ms for all axes.*

Notes:

- Deceleration is time entered in milliseconds. Deceleration rate in pps² (pulses/second²) can be calculated with the following formula.

$$\text{Deceleration rate} = \frac{\text{HSPD} - \text{LSPD}}{\text{ACC}/1000}$$

- The minimum and maximum deceleration values depend on the high-speed and low-speed settings. Refer to [Speed Settings](#) for details.
- Individual deceleration settings will take priority over the global deceleration settings. A global deceleration setting will only be used when, either one of the following conditions is true:
 - The corresponding individual deceleration setting is defined as zero.
 - A coordinated motion is being performed.

In the example below, the settings will set both the global speed settings as well as the individual speed setting for the X axis.

```
HSPD=10000
HSPDX=2000
LSPD=300
accelX=500
accel=300
decel=300
```

Once a move command is issued in this example, the global speed settings for the low speed and deceleration, and the individual X-axis speed settings for the high speed and acceleration, will be used for motion.

See also:

[Motion Settings](#)
[accel](#), [SCV](#), [HSPD](#), [LSPD](#)

Commander Manual[2.1.1.1 Identification Number](#)**%RDATA – Reset the data section of flash**

Sets the deceleration time or acquires the current set value. Deceleration time is available as a global value for all axes and a value settable for each axis

Applies to: CMD-4CR, CMD-4EX-SA, PMX-4EX-SA, PMX-4ET-SA, PMX-2ED-SA, PMX-2EX

ASCII/Standalone `decel` *global acceleration setting*
 `decel[axis]` *individual axis acceleration setting*

Syntax:**Read:**

`decel[axis]`

Where: `[axis]` is set to X, Y, Z, U, or not specified to read global value

Write:

`decel[axis]=[accel]`

Where: `[axis]` is set to X, Y, Z, U, or not specified to read global value
 `[accel]` is the deceleration time in ms (0 – 65,535)

Reply:

Command is accepted and executed

Read:

Returns the value of the `[axis]` deviation counter (28-bit value)

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:**Read:**

DECY ** Returns deceleration time set for the Y axis in ms*

Write:

`decel=300` ** Sets deceleration time to 300 ms for all axes.*

Notes:

- Deceleration is time entered in milliseconds. Deceleration rate in pps² (pulses/second²) can be calculated with the following formula.

$$\text{Deceleration rate} = \frac{\text{HSPD} - \text{LSPD}}{\text{ACC}/1000}$$

- The minimum and maximum deceleration values depend on the high-speed and low-speed settings. Refer to [Speed Settings](#) for details.

- Individual deceleration settings will take priority over the global deceleration settings. A global deceleration setting will only be used when, either one of the following conditions is true:
 - The corresponding individual deceleration setting is defined as zero.
 - A coordinated motion is being performed.

In the example below, the settings will set both the global speed settings as well as the individual speed setting for the X axis.

```
HSPD=10000
HSPDX=2000
LSPD=300
accelX=500
accel=300
decel=300
```

Once a move command is issued in this example, the global speed settings for the low speed and deceleration, and the individual X-axis speed settings for the high speed and acceleration, will be used for motion.

See also:

[Motion Settings](#)
[accel](#), [SCV](#), [HSPD](#), [LSPD](#)

Commander Manual

2.1.1.1 Identification Number

REG – p645 Register read/write

Reads the value contained in, or writes a new value to the selected p645 register.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII REG[RD/WR]0X[axis][reg]

Syntax:

Read:

REGRD[axis]0X[reg]

Where: [axis] is set to X, Y, Z, U
 [reg] is the HEX address of the register to read (see table below)

Write:

REGWR[axis]0X[reg]=[value]

Where: [axis] is set to X, Y, Z, U
 [reg] the HEX address of the register to write (see table below)
 [value] value to write to Register (must be HEX and begin with 0x)

Table of register addresses. All address are in HEX.

Detail	Register					2nd pre-register				
	Name	Read command		Write command		Name	Read command		Write command	
		Address	Symbol	Address	Symbol		Address	Symbol	Address	Symbol
Feed amount, target position	RMV	D0h	RRMV	90h	WRMV	PRMV	C0h	RPRMV	80h	WPRMV
Initial speed	RFL	D1h	RRFL	91h	WRFL	PRFL	C1h	RPRFL	81h	WPRFL
Operation speed	RFH	D2h	RRFH	92h	WRFH	PRFH	C2h	RPRFH	82h	WPRFH
Acceleration rate	RUR	D3h	RRUR	93h	WRUR	PRUR	C3h	RPRUR	83h	WPRUR
Deceleration rate	RDR	D4h	RRDR	94h	WRDR	PRDR	C4h	RPRDR	84h	WPRDR

Speed magnification rate	RMG	D5h	RRMG	95h	WRMG	PRMG	C5h	RPRMG	85h	WPRMG
Ramping-down point	RDP	D6h	RRDP	96h	WRDP	PRDP	C6h	RPRDP	86h	WPRDP
Operation mode	RMD	D7h	RRMD	97h	WRMD	PRMD	C7h	RPRMD	87h	WPRMD
Circular interpolation center	RIP	D8h	RRIP	98h	WRIP	PRIP	C8h	RPRIP	88h	WPRIP
Acceleration S-curve range	RUS	D9h	RRUS	99h	WRUS	PRUS	C9h	RPRUS	89h	WPRUS
Deceleration S-curve range	RDS	DAh	RRDS	9Ah	WRDS	PRDS	CAh	RPRDS	8Ah	WPRDS
Feed amount correction speed	RFA	DBh	RRFA	9Bh	WRFA					
Environment setting 1	RENV1	DCh	RRENV1	9Ch	WRENV1					
Environment setting 2	RENV2	DDh	RRENV2	9Dh	WRENV2					
Environment setting 3	RENV3	DEh	RRENV3	9Eh	WRENV3					
Environment setting 4	RENV4	DFh	RRENV4	9Fh	WRENV4					
Environment setting 5	RENV5	E0h	RRENV5	A0h	WRENV5					
Environment setting 6	RENV6	E1h	RRENV6	A1h	WRENV6					
Environment setting 7	RENV7	E2h	RRENV7	A2h	WRENV7					
COUNTER1 (command position)	RCUN1	E3h	RRCUN1	A3h	WRCUN1					
COUNTER2 (mechanical position)	RCUN2	E4h	RRCUN2	A4h	WRCUN2					
COUNTER3 (deviation counter)	RCUN3	E5h	RRCUN3	A5h	WRCUN3					
COUNTER4 (general purpose)	RCUN4	E6h	RRCUN4	A6h	WRCUN4					
Data for comparator 1	RCMP1	E7h	RRCMP1	A7h	WRCMP1					
Data for comparator 2	RCMP2	E8h	RRCMP2	A8h	WRCMP2					
Data for comparator 3	RCMP3	E9h	RRCMP3	A9h	WRCMP3					
Data for comparator 4	RCMP4	EAh	RRCMP4	AAh	WRCMP4					
Data for comparator 5	RCMP5	EBh	RRCMP5	ABh	WRCMP5	PRCP5	CBh	RPRCP5	8Bh	WPRCP5
Event INT setting	RIRQ	ECh	RRIRQ	ACH	WRIRQ					
COUNTER1 latched data	RLTC1	EDh	RRLTC1							
COUNTER2 latched data	RLTC2	EEh	RRLTC2							

COUNTER3 latched data	RLTC3	EFh	RRLTC3							
COUNTER4 latched data	RLTC4	F0h	RRLTC4							
Extension status	RSTS	F1h	RRSTS							
Error INT status	REST	F2h	RREST	B2h	WREST					
Event INT status	RIST	F3h	RRIST	B3h	WRIST					
Positioning counter	RPLS	F4h	RRPLS							
EZ counter, speed monitor	RSPD	F5h	RRSPD							
Ramping-down point	RSDC	F6h	RRSD C							
Circular interpolation stepping number	RCI	FCh	RRCI	BCh	WRCI	PRCI	CCh	RPRCI	8Ch	WPRCI
Circular interpolation step counter	RCIC	FDh	RRCIC							
Interpolation status	RIPS	FFh	RRIPS							

Reply:*Command is accepted and executed***Read:**

Returns the value of the [axis] [reg] value (28-bit value)

Write:

OK

*Command cannot be processed*An error message is returned. See [Error Codes](#)**Example:****Read:**REGRD $\overline{X}$ 0XDD
2 (RENV2) register** Returns value current in axis X, Environment setting***Write:**REGWR $\overline{X}$ 0X9D=0x3bc5555
to HEX value 3bc5555.** Sets axis X Environment setting 2 (RENV2) register***Notes:**

- HEX address of the registers can be found in Appendix 1 of the PCL-6045BL manual. The table above is a copy of this table.
- Some registers are read only so the write address is not given.
- Some registers have preregisters, these registers have the preregister address given.

See also:

[Motion Settings](#)
[accel](#), [SCV](#), [HSPD](#), [LSPD](#)

Commander Manual[2.1.1.1 Identification Number](#)**SA – Standalone Program Line**

Sets the deceleration time or acquires the current set value. Deceleration time is available as a global value for all axes and a value that can be set for each axis.

Applies to: CMD-4CR, CMD-4EX-SA, PMX-4EX-SA, PMX-4ET-SA, PMX-2ED-SA, PMX-2EX-SA

ASCII SA[prgNo]

Syntax:

Read:

SA[prgNo]

Where: [prgNo] Specifies the specific program number to retrieve the code line number

Reply:

Command is accepted and executed

Read:

Returns the value of the line number referencing the assembled code being executed (0-1799)

Example:

Read:

SA0 * Returns the line number being executed in Program 0.

Notes:

-

See also:

[Motion Settings](#)
 SA, SACTL, SAP, SASTAT, SLOAD, V, GS

Commander Manual[2.1.1.1 Identification Number](#)

TODO: See also and Commander Manual links

%SBLK – LED Blink bit

%SBLK <0 or 1> - 1 Sets blink for power LED, 0 LED is on all the time except in reset to bootloader mode.

Sets or clears the blink bit for LED output. This sets the behavior of the LED pin (2-11) when the commander is active.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII %SBLK

Syntax:

Read:

%SBLK

Write:

%SBLK=[blinkbit]

Where: [blinkbit] 1 – This causes the LED pin (2-11) to cycle on and off.
0 – This causes the LED pin (2-11) to stay on.

Reply:

Command is accepted and executed

Read:

Returns the value of the blink bit.

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:

%SBLK

** Returns the status of the blink bit (0, 1)*

Write:

%SBLK=1

** Sets the blink bit to 1, will cause the LED pin (2-11) to blink when the commander is active.*

Notes:

- Firmware version 201 and later
- The blink bit is reset to 0 when there is a power cycle or the %RST command is sent.

See also:

[Service Commands](#)

Commander Manual

[2.1.1.1 Identification Number](#)

%SCLK – Set System Clock Frequency

%SCLK – This command is used to change the commander system clock value when an external buss clock is used with a commander. The value is in Hz.

*Applies to: CMD-2CR, CMD-4CR, CMD-4EX-SA, CMD-4ED-SA, CMD-8CR
Firmware: V301+*

ASCII %SCLK

Syntax:

Read:

%SCLK

Write:

%SCLK=[frequency]

Where: [frequency] is frequency of the clock in Hz

Reply:

Command is accepted and executed

Read:

Returns the frequency in Hz used for the system clock

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:

%SCLK

** Returns system clock frequency in Hz*

Write:

%SCLK=19660800

** Sets the system clock value to 19.6608MHz.*

Notes:

- This value should match the clock frequency used for the commander. This value does not change the clock frequency, it is used to return accurate speed data from the Commander.
- The default value when using the internal system clock from the Gen5 Commander is 21.6MHz (21600000Hz) .

See also:

[Motion Settings](#)
[accel](#), [SCV](#), [HSPD](#), [LSPD](#)

Commander Manual

[2.1.1.1 Identification Number](#)

TODO: See also and Commander Manual links

%SMAC - MAC address

Sets the deceleration time or acquires the current set value. Deceleration time is available as a global value for all axes and a value settable for each axis

Applies to: CMD-4CR, CMD-4EX-SA, PMX-4EX-SA, PMX-4ET-SA, PMX-2ED-SA, PMX-2EX

ASCII/Standalone

decel

global acceleration setting

decel[axis]

individual axis acceleration setting

Syntax:

Read:

decel[axis]

Where: [axis] is set to X, Y, Z, U, or not specified to read global value

Write:

decel[axis]=[accel]

Where: [axis] is set to X, Y, Z, U, or not specified to read global value
[accel] is the deceleration time in ms (0 – 65,535)

Reply:

Command is accepted and executed

Read:

Returns the value of the [axis] deviation counter (28-bit value)

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:

DECY ** Returns deceleration time set for the Y axis in ms*

Write:

decel=300 ** Sets deceleration time to 300 ms for all axes.*

Notes:

- Deceleration is time entered in milliseconds. Deceleration rate in pps² (pulses/second²) can be calculated with the following formula.

$$\text{Deceleration rate} = \frac{\text{HSPD} - \text{LSPD}}{\text{ACC}/1000}$$

- The minimum and maximum deceleration values depend on the high-speed and low-speed settings. Refer to [Speed Settings](#) for details.
- Individual deceleration settings will take priority over the global deceleration settings. A global deceleration setting will only be used when, either one of the following conditions is true:
 - The corresponding individual deceleration setting is defined as zero.
 - A coordinated motion is being performed.

In the example below, the settings will set both the global speed settings as well as the individual speed setting for the X axis.

```
HSPD=10000
HSPDX=2000
LSPD=300
accelX=500
accel=300
decel=300
```

Once a move command is issued in this example, the global speed settings for the low speed and deceleration, and the individual X-axis speed settings for the high speed and acceleration, will be used for motion.

See also:

[Motion Settings](#)
[accel](#), [SCV](#), [HSPD](#), [LSPD](#)

Commander Manual

[2.1.1.1 Identification Number](#)

%SRID – Auto ID set

%SRID <number from 0 to 99 > (Can add other limits) – Sets device ID to number specified or '01' if no number specified, updates FLASH device ID, and does a soft reset. Reset needed to keep USB in sync with ID change.

Sets the deceleration time or acquires the current set value. Deceleration time is available as a global value for all axes and a value settable for each axis

Applies to: CMD-4CR, CMD-4EX-SA, PMX-4EX-SA, PMX-4ET-SA, PMX-2ED-SA, PMX-2EX

ASCII/Standalone `decel` *global acceleration setting*
 `decel[axis]` *individual axis acceleration setting*

Syntax:

Read:

`decel[axis]`

Where: `[axis]` is set to X, Y, Z, U, or not specified to read global value

Write:

`decel[axis]=[accel]`

Where: `[axis]` is set to X, Y, Z, U, or not specified to read global value
`[accel]` is the deceleration time in ms (0 – 65,535)

Reply:

Command is accepted and executed

Read:

Returns the value of the `[axis]` deviation counter (28-bit value)

Write:

OK

Command cannot be processed

An error message is returned. See [Error Codes](#)

Example:

Read:

DECY ** Returns deceleration time set for the Y axis in ms*

Write:

`decel=300` ** Sets deceleration time to 300 ms for all axes.*

Notes:

- Deceleration is time entered in milliseconds. Deceleration rate in pps² (pulses/second²) can be calculated with the following formula.

$$\text{Deceleration rate} = \frac{\text{HSPD} - \text{LSPD}}{\text{ACC}/1000}$$

- The minimum and maximum deceleration values depend on the high-speed and low-speed settings. Refer to [Speed Settings](#) for details.
- Individual deceleration settings will take priority over the global deceleration settings. A global deceleration setting will only be used when, either one of the following conditions is true:

1. The corresponding individual deceleration setting is defined as zero.
2. A coordinated motion is being performed.

In the example below, the settings will set both the global speed settings as well as the individual speed setting for the X axis.

```
HSPD=10000
HSPDX=2000
LSPD=300
accelX=500
accel=300
decel=300
```

Once a move command is issued in this example, the global speed settings for the low speed and deceleration, and the individual X-axis speed settings for the high speed and acceleration, will be used for motion.

See also:

[Motion Settings](#)
[accel](#), [SCV](#), [HSPD](#), [LSPD](#)

Commander Manual

[2.1.1.1 Identification Number](#)

%WSN – Write Commander Serial Number

Returns the 32-character unique hardware serial number.

Applies to: CMD-4CR, CMD-4EX-SA

ASCII %SN

Syntax:

Read: %SN

Reply:

Command is accepted and executed

Read: Returns the 32 character unique hardware serial number.

Example:

Read: %SN ** Returns the hardware serial number*

Notes:

- The hardware serial is tied to the hardware on the Commander PCB and is unique to the product.

See also:

[System Setup](#)

Commander Manual

[2.1.1.1 Identification Number](#)

TODO: See also and Commander Manual links

Revision History

Revision History

This is the revision history of this document

Date and Revision	Firmware Compatibility	Changes Made
October 2019 Version 1.0	V126BL	New Document
November 2019 Version 2.0	V127BL	Updated Table 6-44 Pins by function to agree with Note 2 on page 106. ERCP add mode 7, level output. PWM polarity corrected. MPGM, MPGD, and SCV support added to standalone. BF and BSTART command replay corrected. High-speed correction function added. An added calculation to get accel/decel rate. Expanded information on-the-fly speed change accel/decel rate.
December 2019 Version 2.1	V127BL	Updating SSPDM command information and HSPD operation related to SSPDM. SYNS command operation update.
February 2020 Version 2.2	V127BL	Table 2-12 MOSI0 changes to pin 2-13, Added notes regarding Linear Interpolation.
October 2021 Version 3.0	V201BL	Updated Document to reflect move to V200 of CMD firmware and RTOS system.
November 2023 Version 3.2	V205BL	Updated Service functions, MST
March 2024 Version 3.3	V205.3BL	Updated Links
July 2025 Version 4.0	V301BL	Updated following pages with changes to V3XX firmware: - IO – General-Purpose I/O Status - SLA – StepNLoop Attempts - EXSTA – External Start - EXSTP – External Stop

Last Build: 7/9/2026 6:29 PM

Contact information



Nippon Pulse America, Inc.

4 Corporate Drive Street, Radford, VA 24141 USA

Phone: 1-540-633-1677

E-mail: info@nipponpulse.com

Web: <https://www.nipponpulse.com>