Navigator® Motion Processor

Programmer's Reference

Performance Motion Devices, Inc.
55 Old Bedford Road
Lincoln, MA 01773

Revision 1.9, September 2004

NOTICE

Warranty

PMD warrants performance of its products to the specifications applicable at the time of sale in accordance with PMD's standard warranty. Testing and other quality control techniques are utilized to the extent PMD deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Performance Motion Devices, Inc. (PMD) reserves the right to make changes to its products or to discontinue any product or service without notice, and advises customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgement, including those pertaining to warranty, patent infringement, and limitation of liability.

Safety Notice

Certain applications using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage. Products are not designed, authorized, or warranted to be suitable for use in life support devices or systems or other critical applications. Inclusion of PMD products in such applications is understood to be fully at the customer's risk.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards must be provided by the customer to minimize inherent procedural hazards.

Disclaimer

PMD assumes no liability for applications assistance or customer product design. PMD does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of PMD covering or relating to any combination, machine, or process in which such products or services might be or are used. PMD's publication of information regarding any third party's products or services does not constitute PMD's approval, warranty or endorsement thereof.

Related Documents

Navigator Motion Processor User's Guide (MC2000UG)

How to set up and use all members of the Navigator Motion Processor family.

Navigator Motion Processor Programmer's Reference (MC2000PR)

Descriptions of all Navigator Motion Processor commands, with coding syntax and examples, listed alphabetically for quick reference.

Navigator Motion Processor Technical Specifications

Four booklets containing physical and electrical characteristics, timing diagrams, pinouts, and pin descriptions of each series:

MC2100 Series, for brushed servo motion control (MC2100TS);
MC2300 Series, for brushless servo motion control (MC2300TS);
MC2400 Series, for microstepping motion control (MC2400TS);
MC2500 Series, for stepping motion control (MC2500TS);
MC2800 Series, for brushed servo and brushless servo motion control (MC2800TS).

Navigator Motion Processor Developer's Kit Manual (DK2000M)

How to install and configure the DK2000 developer's kit PC board.

Table of Contents

This page intentionally left blank.

1 The Navigator Family

	MC2100 Series	MC2300 Series	MC2400 Series	MC2500 Series	MC2800 Series
Number of axes	4, 2, or 1	4, 2 or 1	4, 2 or 1	4, 2, or 1	4 or 2
Motor type supported	DC Brush	Brushless DC	Microstepping	Pulse and direction	DC Brush + Brushless DC
Output format	Brushed servo (single phase)	Commutated (6-step or sinusoidal)	Microstepping	Pulse and direction	Brushed servo (single phase) + commutated (6-step sinusoidal)
Incremental encoder input	√	√	√	√	√
Parallel word device input	√	√	√	√	√
Parallel communication	√	√	√	√	√
Serial communication	√	√	√	√	√
Diagnostic port	√	√	√	√	√
S-curve profiling	√	√	√	√	√
Electronic gearing	√	√	√	√	√
On-the-fly changes	√	√	√	√	√
Directional limit switches	√	√	√	√	√
Programmable bit output	√	√	√	√	√
Software-invertable signals	√	√	√	√	√
PID servo control	√	√	-	-	√
Feedforward (accel & vel)	√	√	-	-	√
Derivative sampling time	√	√	-	-	√
Data trace/diagnostics	√	√	√	√	√
PWM output	√	√	√	-	√
Motion error detection	√	√	√ (with encoder)	√ (with encoder)	√
Axis settled indicator	√	√	√ (with encoder)	√ (with encoder)	√
DAC-compatible output	√	√	√	-	√
Pulse & direction output	-	-	-	√	-
Index & Home signals	√	√	√	√	√
Position capture	√	√	√	√	√
Analog input	√	√	√	√	√
User-defined I/O	√	√	√	√	√
External RAM support	√	√	√	√	√
Chipset part numbers	MC2140 (4 axes) MC2120 (2 axes) MC2110 (1 axis)	MC2340 (4 axes) MC2320 (2 axes) MC2310 (1 axis)	MC2440 (4 axes) MC2420 (2 axes) MC2410 (1 axis)	MC2540 (4 axes) MC2520 (2 axes) MC2510 (1 axis)	MC2840 (4 axes) MC2820 (2 axes)
Developer's Kit part numbers	DK2100	DK2300	DK2400	DK2500	DK2800

Introduction

This manual describes the formating of the instructions supported by the Navigator family of motion processors from PMD. These devices are members of PMD's second-generation motion processor family, which consists of twelve separate products organized into four series. The series are detailed in the Family Summary at the bottom of this page.

Each of these devices is a complete chip-based motion processor. They provide trajectory generation and related motion control functions. Depending on the type of motor being controlled, they provide servo loop closure, on-board commutation for brushless motors, and high-speed pulse and direction outputs. Together, these products provide a software-compatible family of dedicated motion processors that can handle a large variety of system configurations.

Each of these chips utilize a similar architecture, consisting of a high-speed DSP (Digital Signal Processor) computation unit, along with an ASIC (Application Specific Integrated Circuit). The computation unit contains special on-board hardware which makes it well suited for the task of motion control.

Along with similar hardware architecture, these chips also share most software commands, so that software written for one chipset may be re-used with another; even though the type of motor may be different.

Each chipset consists of two PQFP (Plastic Quad Flat Pack) ICs: a 100-pin Input/Output (I/O) chip, and a 132-pin Command Processor (CP) chip.

The four series in the Navigator family are designed for a particular type of motor or control scheme. The following is a description of each series.

Navigator Family Summary

MC2100 Series (MC2140, MC2120, MC2110) – This series outputs motor commands in either Sign/Magnitude PWM or DAC-compatible format for use with DC brush motors, or with brushless DC motors having external commutation.

MC2300 Series (MC2340, MC2320, MC2310) – This series outputs sinusoidally commutated motor signals appropriate for driving brushless DC motors. Depending on the motor type, the output is a two-phase or three-phase signal in either PWM- or DAC-compatible format.

MC2400 Series (MC2440, MC2420, MC2410) – This series provides microstepping signals for microstepping motors. Two phased signals per axis are generated in either PWM- or DAC-compatible format.

MC2500 Series (MC2540, MC2520, MC2510) – These chipsets provide high-speed pulse and direction signals for pulse and direction motor systems.

MC2800 Series (MC2840, MC2820) – This series outputs sinusoidally or 6-step commutated motor signals appropriate for driving brushless DC motors; as well as PWM- or DAC-compatible outputs for driving DC brush servo motors.

2 Instruction Reference

2.1 How to use this reference

This document is in two parts. First, a detailed description of all host instructions; and second, a set of summary tables listing the instructions by functional group, alphabetically by instruction mnemonic, and numerically by hexadecimal code. In the reference section, instructions are arranged alphabetically; except that all "Set/Get" pairs (for example, SetVelocity and GetVelocity) are described together. Each description begins on a new page; most occupy no more than a page. Each page is organized as follows.

Motor types The motor types and corresponding Navigator part number(s) to which the instructions apply.

Name The instruction mnemonic is shown at the left; its hexadecimal code at the right.

Syntax The instruction mnemonic and its required arguments are shown with all arguments separated by spaces.

Arguments There are two types of arguments: encoded-field and numeric.

Encoded-field arguments are packed into a single 16-bit data word, except for axis, which occupies bits 11-8 of the instruction word. The ***Name*** of the argument is shown in the generic syntax. ***Instance*** is the mnemonic used to represent the data value. ***Encoding*** is the value assigned to the field for that instance.

For numeric arguments, the parameter ***Value***, the ***Type*** (signed or unsigned integer), and ***Range*** of acceptable values are given. Numeric arguments may require one or two data words. For 32-bit arguments, the high-order part is transmitted first.

Buffered Certain parameters and other data written to the chipset are buffered; that is, they are not acted upon until the next Update or MultiUpdate command is executed. These parameters are identified by the word **buffered** in the instruction heading.

Packet structure This is a graphic representation of the 16-bit words transmitted in the packet: the instruction, which is identified by its name, followed by one, two, or three data words. Bit numbers are shown directly below each word. For each field in a word, only the high and low bits are shown. For 32-bit numeric data, the high-order bits are numbered from 16 to 31; the low-order bits from 0 to 15.

The hex code of the instruction is shown in boldface text.

Argument names are shown in their respective words or fields.

For data words, the direction of transfer (read or write) is shown at the left of the word's diagram.

Unused bits are shaded. **In data words and instructions sent (written) to the motion processor, all unused bits must be 0**.

Description Describes what the instruction does and any special information relating to the instruction.

Restrictions Describes the circumstances in which the instruction is not valid; that is, when it should not be issued. For example, velocity, acceleration, deceleration, and jerk parameters may not be issued while an S-curve profile is being executed.

see Refers to related instructions within this manual.

AdjustActualPosition F5h

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

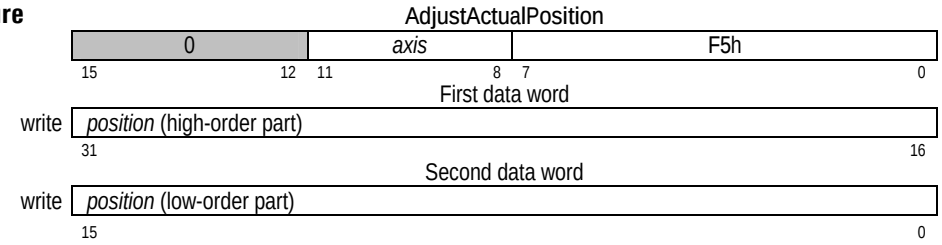
Syntax AdjustActualPosition *axis position*

Arguments

Name	***Instance***	***Encoding***
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3

	Type	***Range***	***Scaling***	***Units***
position	signed 32 bits	-2^{31} *to* $2^{31}-1$	unity	counts steps

Packet structure



Description

The *position* specified as the parameter to AdjustActualPosition is summed with the actual position register (encoder position) for the specified *axis*. This has the effect of adding or subtracting an offset to the actual position. At the same time, the commanded position is replaced by the new actual position value minus the position error. This prevents a servo "bump" when the new axis position is established. The destination position (see SetPosition) is also modified by this amount so that no trajectory motion will occur when the update instruction is issued. In effect, this instruction establishes a new reference position from which subsequent positions can be calculated. It is commonly used to set a known reference position after a homing procedure.

Note: On the MC2400 and MC2500 series, the position error is zeroed.

AdjustActualPosition takes effect immediately, it is not buffered.

Restrictions

see GetPositionError; GetActualVelocity, Set/GetActualPositionUnits, Set/GetActualPosition

ClearInterrupt ACh

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

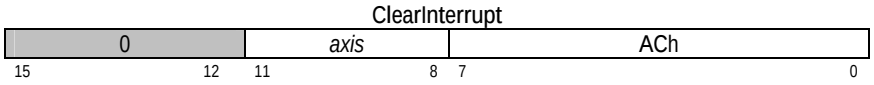
Syntax

ClearInterrupt

Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3

Packet structure



Description

ClearInterrupt resets the HostInterrupt signal to its inactive state. It is used after an interrupt has been recognized and processed by the host. If interrupts are still pending, the HostInterrupt line will return to its active state within one chip cycle. Refer to Set/GetSampleTime for information on chip cycle timing. This command does not effect the event status register. The ResetEventStatus command should be issued prior to the ClearInterrupt command to clear the condition which generated the interrupt. The ClearInterrupt command has no effect if it is executed when no interrupts are pending.

Restrictions

see

GetInterruptAxis, Set/GetInterruptMask, ResetEventStatus

ClearPositionError buffered 47h

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

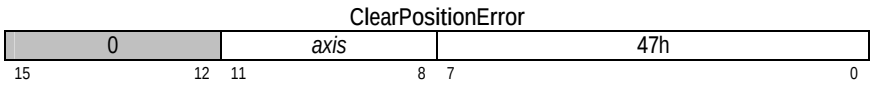
Syntax

ClearPositionError *axis*

Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3

Packet structure



Description

ClearPositionError sets the profile's commanded position equal to the actual position (encoder input), thereby clearing the position error for the specified *axis*. This command can be used when the axis is at rest, or when it is moving.

Restrictions

ClearPositionError is a buffered command. The new value set will not take effect until the next Update or MultiUpdate instruction.

This command should not be sent while the chip is executing a move using the S-curve profile mode.

see

GetPositionError, MultiUpdate, Set/GetPositionErrorLimit, Update

GetActivityStatus A6h

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

Syntax

GetActivityStatus *axis*

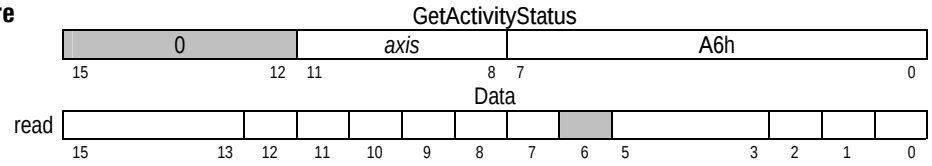
Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3

Returned data

	Type	
status	unsigned 16 bits	*see below*

Packet structure



Description

GetActivityStatus reads the 16-bit activity status register for the specified *axis*. Each of the bits in this register continuously indicate the state of the motion processor without any action on the part of the host. There is no direct way to set or clear the state of these bits, since they are controlled by the motion processor. The following table shows the encoding of the data returned by this command.

Name	Bit Number	Description
Phasing initialized	0	Set to 1 if phasing is initialized (MC2300/MC2800 series only)
At maximum velocity	1	Set to 1 when the trajectory is at maximum velocity. This bit is determined by the trajectory generator, not the actual encoder position.
Tracking	2	Set to 1 when the axis is within the tracking window.
Current profile mode	3-5	Contains trajectory mode encoded as follows: bit 5 bit 4 bit 3 Profile Mode 0 0 0 trapezoidal 0 0 1 velocity contouring 0 1 0 S-curve 0 1 1 electronic gear
reserved	6	Not used; may be 0 or 1.
Axis settled	7	Set to 1 when the axis is settled.
Motor on/off	8	Set to 1 when motor mode is on, 0 when off.
Position capture	9	Set to 1 when a value has been captured by the high speed position capture hardware, but has not yet been read.
In-motion	10	Set to 1 when the trajectory generator is executing a profile.

Name	Bit Number	Description
In positive limit	11	Set to 1 when the positive limit switch is active.
In negative limit	12	Set to 1 when the negative limit switch is active.
Profile segment	13-15	When the profile mode is S-curve, it contains the profile segment number 1-7 while profile is in motion and contains a value of 0 when the profile is at rest. This field is undefined when using the trapezoidal and velocity contouring profile modes.

Restrictions

see GetEventStatus, GetSignalStatus

GetActualVelocity ADh

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

Syntax

GetActualVelocity *axis*

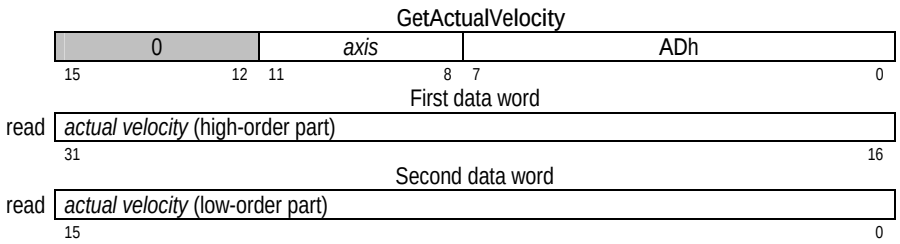
Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3

Returned data

	Type	*Range*	*Scaling*	*Units*
velocity	signed 32 bits	-2^{31} *to* $2^{31}-1$	$1/2^{16}$	counts/cycle

Packet structure



Description

GetActualVelocity reads the value of the velocity for the specified axis. The actual velocity is derived by subtracting the actual position during the previous chip cycle from the actual position for this chip cycle. The result of this subtraction will always be integer because position is always integer. As a result the value returned by GetActualVelocity will always be a multiple of 65,536 since this represents a value of one in the 16.16 number format. The low word is always zero. This value is the result of the last encoder input, so it will be accurate to within one cycle.

Scaling example: If a value of 1,703,936 is retrieved by the GetActualVelocity command (high word: 01Ah, low word: 0h), then this corresponds to a velocity of 1,703,936/65,536, or 26 counts/cycle.

Restrictions

see

GetCommandedVelocity

GetCaptureValue 36h

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

Syntax

GetCaptureValue *axis*

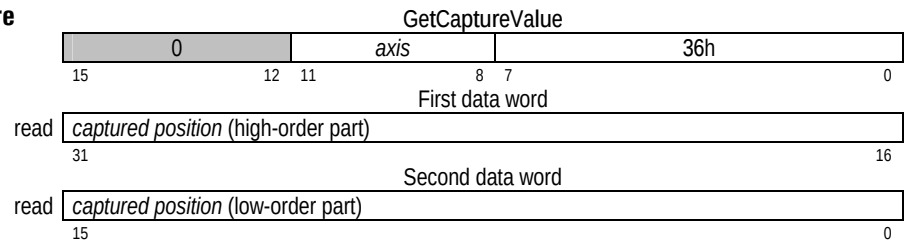
Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3

Returned data

	Type	*Range*	*Scaling*	*Units*
captured position	signed 32 bits	-2^{31} *to* $2^{31}-1$	unity	counts steps

Packet structure



Description

GetCaptureValue returns the contents of the position capture register for the specified axis. This command also resets bit 9 of the activity status register; thus allowing another capture to occur.
If actual position units is set to steps, the returned position will be in units of steps.

Restrictions

see

Set/GetCaptureSource, Set/GetActualPosition, GetActivityStatus

GetChecksum F8h

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

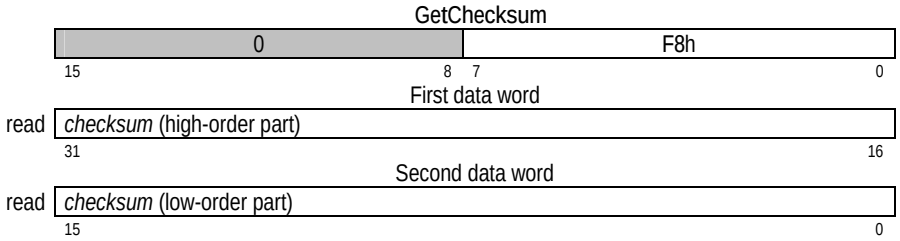
Syntax

GetChecksum

Returned data

	Type
checksum	unsigned 32 bits

Packet structure



Description

GetChecksum reads the chip's internal 32-bit checksum value. The return value is dependent on the silicon revision number of the motion processor. For further information, contact PMD customer support.

Restrictions

see

GetCommandedAcceleration A7h

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

Syntax GetCommandedAcceleration *axis*

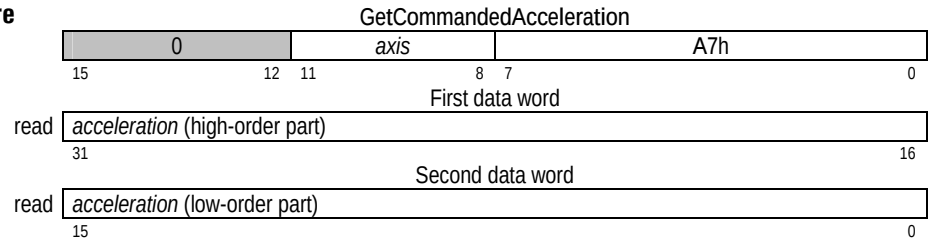
Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3

Returned data

	Type	*Range*	*Scaling*	*Units*
acceleration	signed 32 bits	-2^{31} *to* $2^{31}-1$	$1/2^{16}$	counts/cycle^2 steps/cycle^2

Packet structure



Description GetCommandedAcceleration returns the commanded acceleration value for the specified *axis*. Commanded acceleration is the instantaneous acceleration value output by the trajectory generator.

Scaling example: If a value of 114,688 is retrieved using this command, then this corresponds to 114,688/65,536 = 1.750 counts/cycle^2 acceleration value.

Restrictions

see GetCommandedPosition, GetCommandedVelocity

GetCommandedPosition 1Dh

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

Syntax GetCommandedPosition *axis*

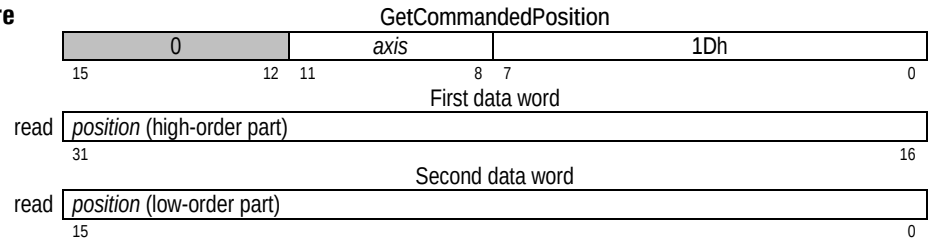
Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3

Returned data

	Type	*Range*	*Scaling*	*Units*
position	signed 32 bits	-2^{31} *to* $2^{31}-1$	unity	counts steps

Packet structure



Description GetCommandedPosition returns the commanded position for the specified *axis*. Commanded position is the instantaneous position value output by the trajectory generator.

Restrictions

see GetCommandedAcceleration, GetCommandedVelocity

GetCommandedVelocity 1Eh

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

Syntax GetCommandedVelocity *axis*

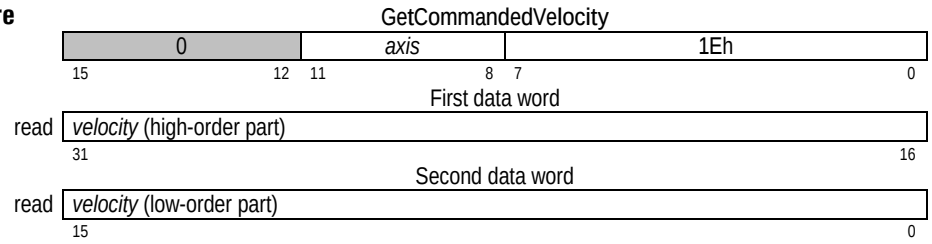
Arguments

Name	***Instance***	***Encoding***
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3

Returned data

	Type	***Range***	***Scaling***	***Units***
velocity	signed 32 bits	-2^{31} *to* $2^{31}-1$	$1/2^{16}$	counts/cycle steps/cycle

Packet structure



Description GetCommandedVelocity returns the commanded velocity value for the specified *axis*. Commanded velocity is the instantaneous velocity value output by the trajectory generator.

Scaling example: If a value of -1,234,567 is retrieved using this command (FFEDh in high word, 2979h in low word), then this corresponds to -1,234,567/65,536 = -18.8380 counts/cycle velocity value.

Restrictions

see GetCommandedAcceleration, GetCommandedPosition

GetCurrentMotorCommand 3Ah

Motor types

DC Brush		Brushless DC		Microstepping		
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	

Syntax GetCurrentMotorCommand *axis*

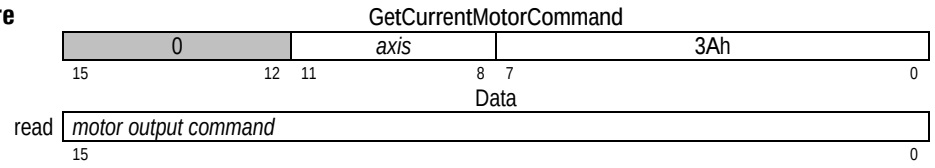
Arguments

Name	***Instance***	***Encoding***
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3

Returned data

	Type	***Range***	***Scaling***	***Units***
motor output command	signed 16 bits	-2^{15} *to* $2^{15}-1$	$100/2^{15}$	% output

Packet structure



Description

GetCurrentMotorCommand returns the value of the motor output command for the specified *axis*. In closed-loop mode, this is the output of the servo filter. In open-loop mode, it is the contents of the motor output command register.

Scaling example: To convert the retrieved value to units of percent of full scale motor output, multiply by 100/32,768. For example, if the value -123 is retrieved by the GetCurrentMotorCommand, this represents -123*100/32,768 or -0.3754% of full scale output.

Restrictions

see Set/GetMotorCommand

GetDerivative 9Bh

Motor types

DC Brush		Brushless DC			
MC2100	MC2800	MC2300	MC2800		

Syntax GetDerivative *axis*

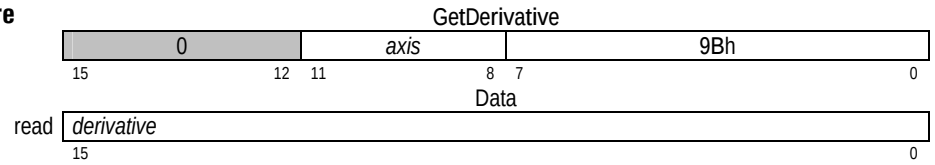
Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3

Returned data

	Type	*Range*	*Scaling*	*Units*
derivative	signed 16 bits	-2^{15} *to* $2^{15}-1$	unity	counts/cycle

Packet structure



Description GetDerivative returns the derivative of the position error as calculated by the servo filter. The derivative value is the previous position error subtracted from the current position error.

See SetDerivativeTime for details on setting the derivative sampling time.

Restrictions The derivative value is only computed when the chipset is in closed-loop mode. See SetMotorMode On.

see GetIntegral, Set/GetDerivativeTime

GetEventStatus 31h

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

Syntax

GetEventStatus *axis*

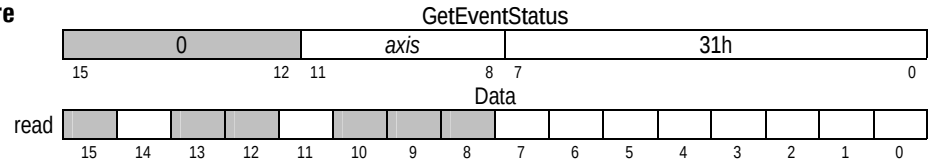
Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3

Returned data

	Type	
status	unsigned 16 bits	see below

Packet structure



Description

GetEventStatus reads the event register for the specified *axis*. All of the bits in this status word are set by the motion processor and cleared by the host. To clear these bits, use the ResetEventStatus command.

The following table shows the encoding of the data returned by this command.

Name	Bit(s)	Description
Motion complete	0	Set to 1 when motion is completed. SetMotionCompleteMode determines if this bit is based on the trajectory generator position or the encoder position.
Wrap-around	1	Set to 1 when the actual (encoder) position has wrapped from maximum allowed position to minimum or vice versa.
Breakpoint 1	2	Set to 1 when breakpoint 1 has been triggered.
Capture received	3	Set to 1 when a position capture has occurred.
Motion error	4	Set to 1 when a motion error has occurred.
In positive limit	5	Set to 1 when the axis has entered a positive limit switch.
In negative limit	6	Set to 1 when the axis has entered a negative limit switch.
Instruction error	7	Set to 1 when an instruction error has occurred.
reserved	8-10	Not used; may be 0 or 1.
Commutation error	11	Set to 1 when a commutation error has occurred.
reserved	12-13	Not used; may be 0 or 1.
Breakpoint 2	14	Set to 1 when breakpoint 2 has been triggered.

Name	Bit(s)	Description
reserved	15	Not used; may be 0 or 1.

Restrictions

see GetActivityStatus, GetSignalStatus

GetHostIOError A5h

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

Syntax GetHostIOError

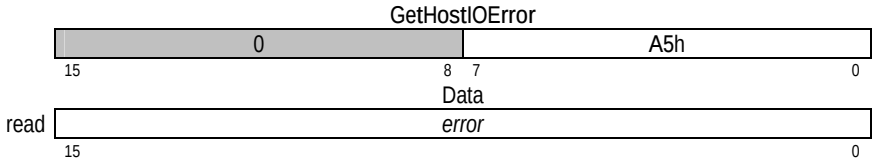
Arguments none

Returned data

Name	***Instance***	***Encoding***
error	No error	0
	Processor Reset	1
	Invalid instruction	2
	Invalid axis	3
	Invalid parameter	4
	Trace running	5
	reserved	6
	Block out of bounds	7
	Trace buffer zero	8
	Bad serial checksum	9
	Not primary port	Ah
	Invalid negative value	Bh
	Invalid parameter change	Ch
	Invalid move after limit condition	Dh
	Invalid move into limit	Eh

Type	***Range***
unsigned 16 bits	0 - Eh

Packet structure



Description

GetHostIOError returns the code for the last host I/O error, and then resets the *error* to zero. Generally, this command is issued only after the instruction error bit in the event status register indicates there was an I/O error. It also resets the HostIOError bit in the I/O status read word to zero.

Restrictions

see GetEventStatus

GetIntegral 9Ah

Motor types

DC Brush		Brushless DC			
MC2100	MC2800	MC2300	MC2800		

Syntax

GetIntegral *axis*

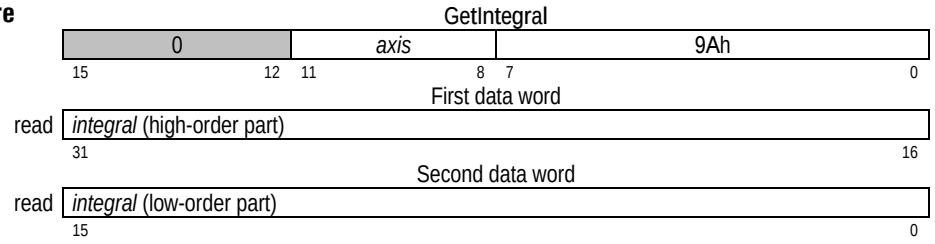
Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3

Returned data

	Type	*Range*	*Scaling*	*Units*
integral	signed 32 bits	-2^{31} *to* $2^{31}-1$	$1/2^8$	counts*cycles

Packet structure



Description

GetIntegral returns the value of the integrated position error of the servo filter for the specified *axis*. GetIntegral can be used to monitor loading on the axis, because changes in the axis loading can be reflected in the value of the integration term.

Scaling example: If a constant position error of 100 counts is present for 256 cycles, then the total accumulated integral value will be 100 (100*256/256). Alternatively, a returned value of 1,000 indicates a total stored value of 256,000 counts*cycles (1,000*256).

Restrictions

The integrated position error is available only when the chipset is in closed-loop mode. See SetMotorMode On.

see

GetDerivative, Set/GetIntegrationLimit

GetInterruptAxis E1h

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

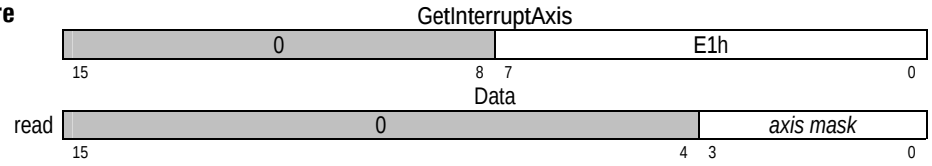
Syntax GetInterruptAxis

Arguments none

Returned data

Name	***Instance***	***Encoding***
axis mask	None	0
	Axis1mask	1
	Axis2mask	2
	Axis3mask	4
	Axis4mask	8

Packet structure



Description GetInterruptAxis returns a field which identifies all axes with pending interrupts. Axis numbers are assigned to the low-order four bits of the returned word; bits corresponding to interrupting axes are set to 1. If there are no pending interrupts, the returned word is 0. If any axis has a pending interrupt, the *HostInterrupt* signal will be in an active state.

Restrictions

see ClearInterrupt, Set/GetInterruptMask

GetPhaseCommand EAh

Motor types

	Brushless DC		Microstepping		
	MC2300	MC2800	MC2400	MC2800	

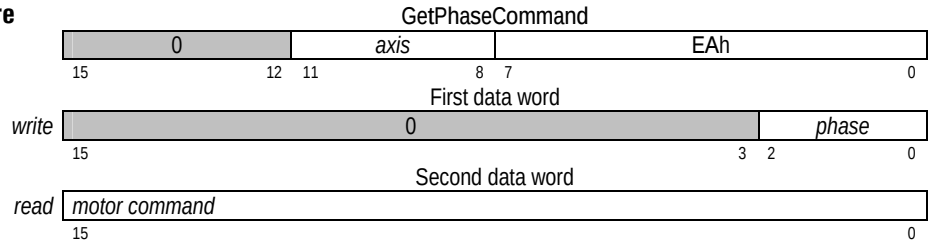
Syntax GetPhaseCommand *axis phase*

Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3
phase	PhaseA	0
	PhaseB	1
	PhaseC	2

	Type	*Range*	*Scaling*	*Units*
Returned data *motor command*	signed 16 bits	-2^{15} *to* $2^{15}-1$	$100/2^{15}$	% output

Packet structure



Description GetPhaseCommand returns the value of the motor output command for phase A, B, or C of the specified *axis*. These are the phase values directly output to the motor after commutation.

Scaling example: If a value of -4,489 is retrieved (EE77h) for a given axis and phase, then this corresponds to -4,489*100/32,767 = -13.7% of full-scale output.

Restrictions PhaseC is only valid when the motor type has been set for a 3-phase commutation.

see InitializePhase, Set/GetNumberPhases

GetPositionError 99h

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

Syntax

GetPositionError *axis*

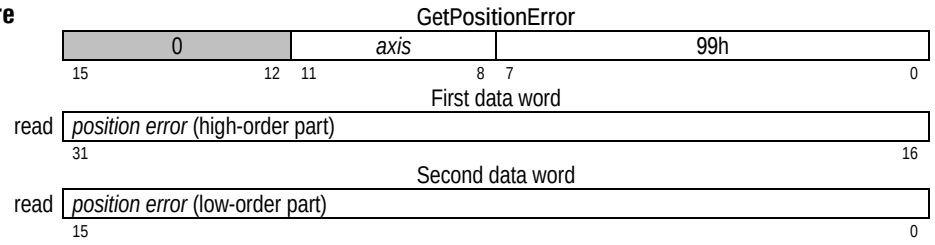
Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3

Returned data

	Type	*Range*	*Scaling*	*Units*
position error	signed 32 bits	-2^{31} *to* $2^{31}-1$	unity	counts steps

Packet structure



Description

GetPositionError returns the position error of the specified *axis*. The error is the difference between the actual position (encoder position), and the commanded position (instantaneous output of the trajectory generator). When used with the microstepping or pulse and direction chipset, the error is defined as the difference between the encoder position (represented in steps), and the commanded position (instantaneous output of the trajectory generator).

Restrictions

see

Set/GetPosition, Set/GetPositionErrorLimit

GetSignalStatus A4h

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

Syntax

GetSignalStatus *axis*

Arguments

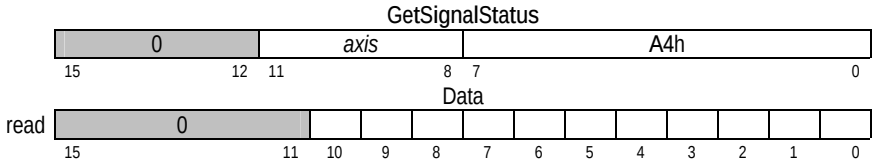
Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3

Returned data

	Description	*Bit Number*
status	EncoderA	0
	EncoderB	1
	EncoderIndex	2
	EncoderHome	3
	PositiveLimit	4
	NegativeLimit	5
	AxisIn	6
	HallA	7
	HallB	8
	HallC	9
	AxisOut	10
	reserved	11-15

Type
unsigned 16 bits

Packet structure



Description

GetSignalStatus returns the contents of the signal status register for the specified *axis*. The signal status register contains the value of the various hardware signals connected to each axis of the motion processor. The value read is combined with the signal sense register (see SetSignalSense), and then returned to the user. For each bit in the signal sense register that is set to 1, the corresponding bit in the GetSignalStatus command will be inverted. Therefore, a low signal will be read as 1, and a high signal will be read as a 0. Conversely, for each bit in the signal sense register that is set to 0, the corresponding bit in the GetSignalStatus command is not inverted. Therefore, a low signal will be read as 0, and a high signal will be read as 1.

All of the bits in the GetSignalStatus command are inputs, except for *AxisOut*. The value read for this bit is equal to the value output by the axis out mechanism. See the SetAxisOutSource command for more details.

Restrictions

see

GetActivityStatus, GetEventStatus, SetGetSignalSense

GetTime 3Eh

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

Syntax

GetTime

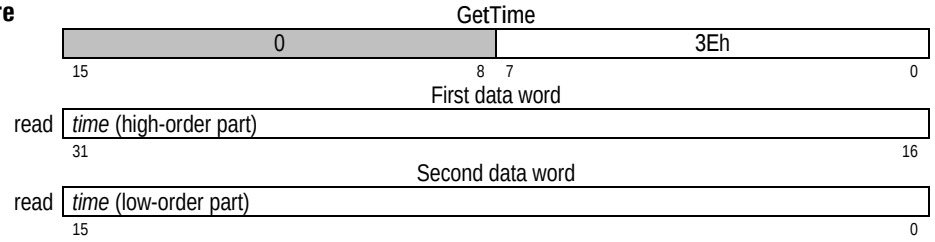
Arguments

none

Returned data

Name	*Type*	*Range*	*Scaling*	*Units*
time	unsigned 32 bits	0 *to* 2^{32}-1	unity	cycles

Packet structure



Description

GetTime returns the number of cycles which have occurred since the motion processor was last reset.

Restrictions

see

GetTraceCount BBh

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

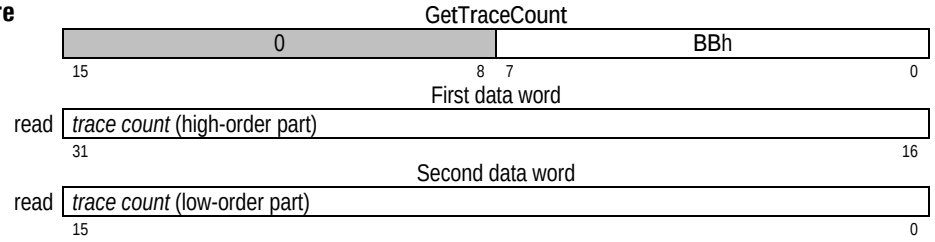
Syntax GetTraceCount

Arguments none

Returned data

Value	*Type*	*Range*	*Scaling*	*Units*
trace count	unsigned 32 bits	0 *to* 2^{32}-1	unity	samples

Packet structure



Description GetTraceCount returns the number of points (variable values) stored in the trace buffer since the beginning of the trace.

Restrictions If the trace mode is set to "rolling" and the buffer wraps, GetTraceCount returns the number of samples in the filled buffer.

see ReadBuffer, Set/GetTraceStart, Set/GetTraceStop, SetGetBufferLength

GetTraceStatus BAh

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

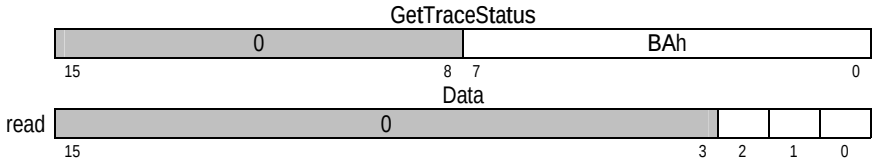
Syntax GetTraceStatus

Arguments none

Returned data

Name	*Type*
see below	unsigned 16 bits

Packet structure



Description

GetTraceStatus returns the trace status. The definitions of the individual status bits are as follows.

Bit	Name	Description
0	Mode	Set to 0 when trace is in one-time mode; 1 when in rolling mode.
1	Activity	Set to 1 when trace is active (currently tracing); 0 if trace not active.
2	Data wrap	Set to 1 when trace has wrapped; 0 if it has not wrapped. If 0, the buffer has not yet been filled, and all recorded data is intact. If 1, the trace has wrapped to the beginning of the buffer; any previous data may have been overwritten if not explicitly retrieved by the host using the ReadBuffer command while the trace is active.

Restrictions

see Set/GetTraceStart, Set/GetTraceMode

GetVersion 8Fh

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

Syntax

GetVersion

Arguments

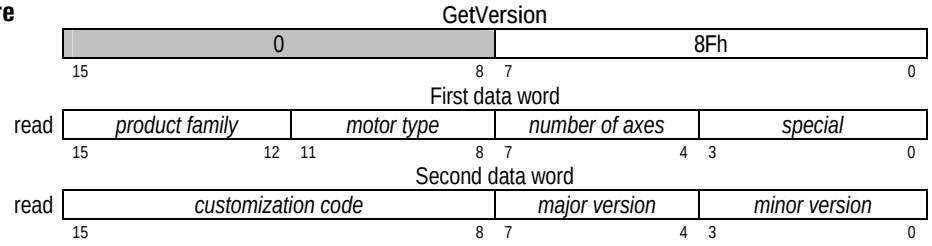
None

Returned data

Name	***Instance***	***Encoding***
product family	Navigator	2
motor type	DC Brush	1
	Brushless DC	3
	Microstepping	4
	Pulse & Direction	5
	Multiple Motor	8
number of axes		1, 2, *or* 4
special	reserved	0 *to* 15
customization code	none	0
	other	1 *to* 255
major version	Major silicon revision number	0 *to* 15
minor version	Minor silicon revision number	0 *to* 15

Type
unsigned 32 bits

Packet structure



Description

GetVersion returns product information encoded as shown in the packet structure diagram.

Restrictions

see

InitializePhase 7Ah

Motor types

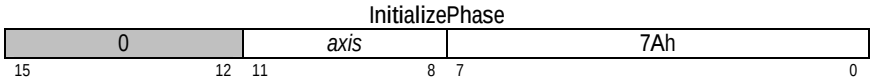
	Brushless DC			
	MC2300	MC2800		

Syntax InitializePhase *axis*

Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3

Packet structure



Description InitializePhase initializes the phase angle for the specified axis using the mode (Hall-based or Algorithmic) specified by the SetPhaseInitializationMode command.

Restrictions **Warning:** If this command is sent after the phase initialization mode has been set to algorithmic, the motor may suddenly move in an uncontrolled manner.

This command is only applicable in the sinusoidal commutation mode (see SetCommutationMode).

see GetPhaseCommand, Set/GetNumberPhases, SetGetCommutationMode

MultiUpdate 5Bh

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

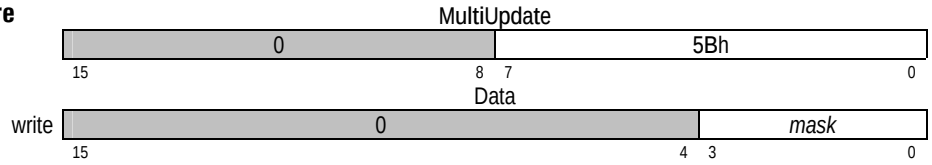
Syntax

MultiUpdate *mask*

Arguments

Name	*Instance*	*Encoding*
mask	None	0
	Axis1mask	1
	Axis2mask	2
	Axis3mask	4
	Axis4mask	8

Packet structure



Description

MultiUpdate causes an Update to occur on all axes whose corresponding bit is set to 1 in the mask argument. After this command is executed, and for those axes which are selected using the mask, all buffered data parameters are copied into the corresponding run-time registers. The following table shows the buffered commands and variables which are made active as a result of the Update command.

Type	Command
General	ClearPositionError
Trajectory	Acceleration
	Deceleration
	GearRatio
	Jerk
	Position
	ProfileMode
	StopMode
	Velocity
Servo	DerivativeTime
	IntegrationLimit
	Kaff
	Kd
	Ki
	Kp
	Kvff
Motor	MotorCommand

Restrictions

see

Update

NoOperation 00h

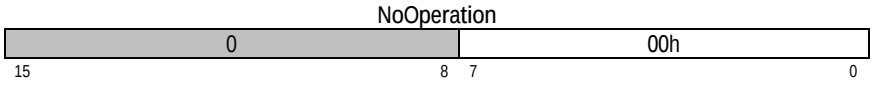
Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

Syntax NoOperation

Arguments none

Packet structure



Description The NoOperation command has no effect on the chipset.

Restrictions

see

ReadAnalog EFh

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

Syntax ReadAnalog *portID*

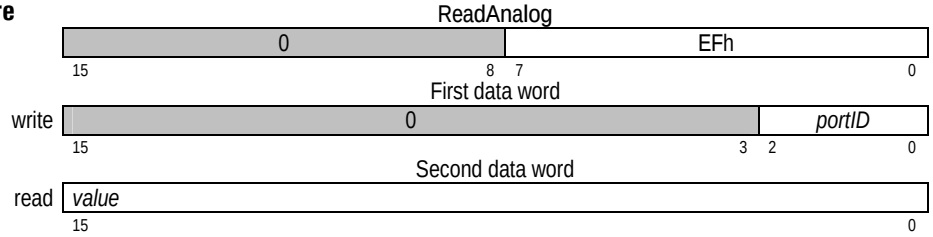
Arguments

Name	*Type*	*Range*	*Scaling*	*Units*
portID	unsigned 16 bits	0 to 7	unity	

Returned data

value	unsigned 16 bits	0 *to* 2^{16}-1	$100/2^{16}$	% input

Packet structure



Description

ReadAnalog returns a 16-bit value representing the voltage (read by an on-chip 10-bit A/D) presented to the specified analog input. The value returned is the result of shifting the 10-bit value 6 bits to the left.

Restrictions

see

ReadBuffer C9h

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

Syntax ReadBuffer *bufferID*

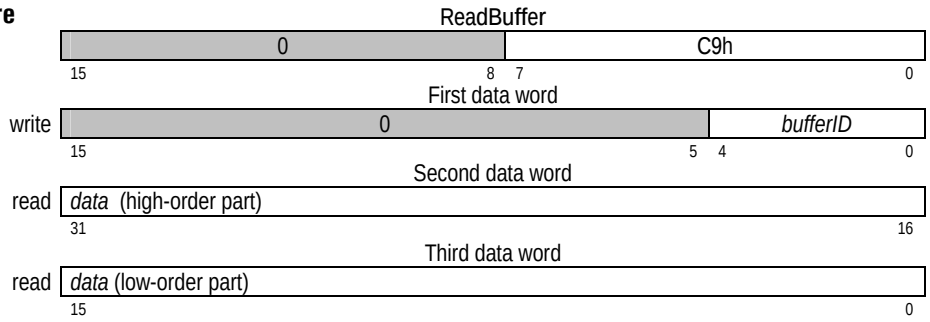
Arguments

Name	*Type*	*Range*
bufferID	unsigned 16 bits	0 to 31

Returned data

data	signed 32 bits	-2^{31} *to* $2^{31}-1$

Packet structure



Description

ReadBuffer returns the 32-bit contents of the location pointed to by the read buffer index in the specified buffer. After the contents have been read, the read index is incremented by 1; if the result is equal to the buffer length (set by SetBufferLength), the index is reset to 0. The read index is automatically changed at the completion of a trace when in rolling trace mode.

Refer to Section 7.6.4 of the Navigator User's Guide for details.

Restrictions

see Set/GetBufferReadIndex, WriteBuffer, SetGetBufferStart, SetGetBufferLength

ReadIO 83h

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

Syntax ReadIO *address*

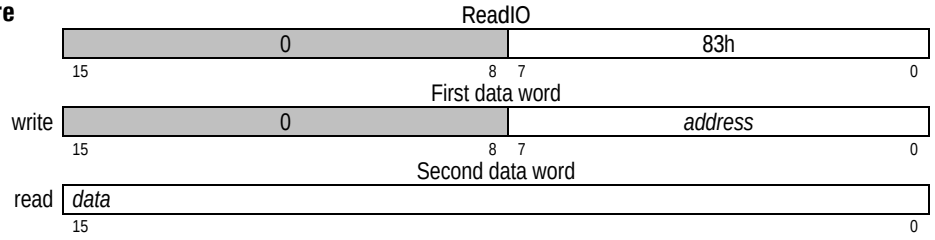
Arguments

Name	*Type*	*Range*
address	unsigned 16 bits	0 to 255

Returned data

data	unsigned 16 bits	0 *to* 2^{16}-1

Packet structure



Description

ReadIO reads one 16-bit word of data from the device at *address*. Address is an offset from location 1000h of the motion processor's peripheral device address space.

The format and interpretation of the 16-bit data word are dependent on the user-defined device being addressed. User-defined I/O can be used to implement a number of features which include additional parallel I/O, flash memory for non-volatile configuration information storage, or display devices such as LED arrays.

Restrictions

see WriteIO

Reset 39h

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

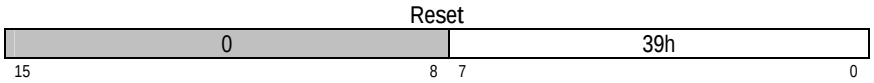
Syntax

Reset

Arguments

none

Packet structure



Description

Reset restores the motion processor to its initial condition, setting all motion processor variables to their default values. Most variables are motor-type independent; however, several default values depend upon the configured motor type of the axis. The motor-type independent values are listed in the following table.

Variable Name	Default Value	Variable Name	Default Value
Acceleration	0	MotionCompleteMode	0
ActualPosition	0	MotorBias	0
ActualPositionUnits	0	MotorCommand	0
AutoStopMode	1	MotorLimit	32767
AuxilliaryEncoderSource	0	MotorMode	1
AxisMode	1	NumberPhases	*see note 1*
AxisOutSource	0	OutputMode	*see note 2*
BiQuadCoefficients	0	PhaseAngle	0
Breakpoint 1	0	PhaseCounts	1
Breakpoint 2	0	PhaseInitializeMode	0
BreakpointValue 1	0	PhaseInitializeTime	0
BreakpointValue 2	0	PhaseOffset	65535
BufferLength	0	PhasePrescaleMode	0
BufferReadIndex	0	Position	0
BufferStart	0	PositionErrorLimit	2^{31}-1
BufferWriteIndex	0	ProfileMode	0
CaptureSource	0	SampleTime	*see note 3*
CommutationMode	0	SettleTime	0
Deceleration	0	SettleWindow	0
DerivativeTime	1	SignalSense	0
EncoderModulus	0	StartVelocity	0
EncoderToStepRatio	001001h	StepRange	*see Restrictions*
EventStatus	1	StopMode	0
GearMaster	0	SynchronizationMode	0
GearRatio	0	TraceMode	0
IntegrationLimit	0	TracePeriod	1
InterruptMask	0	TraceStart	0
Jerk	0	TraceStop	0
Kaff	0	TraceVariable 1	0
Kd	0	TraceVariable 2	0
Ki	0	TraceVariable 3	0
Kout	65535	TraceVariable 4	0
Kp	0	TrackingWindow	0
Kvff	0	Velocity	0
LimitSwitchMode	1		

Notes:

1. The reset value for the number of phases is dependent on the motion processor series, as follows:

   MC2100 1
   MC2300 3
   MC2400 2
   MC2800 3

2. The reset value for the output mode is dependent on the motion processor series, as follows:

   MC2100 1
   MC2300 2
   MC2400 1
   MC2800 2

3. The reset value for SampleTime depends on the number of axes and the motion processor series, as follows:

   MC2100 102 x number of axes
   MC2300 154 x number of axes
   MC2400 154 x number of axes
   MC2500 154 x number of axes
   MC2800 154 x number of axes

All axes supported by the motion processor are enabled at reset.

Restrictions

For the MC2400/MC2500:

AutoPositionUnits Counts

AutoStopMode Off

EncoderSource None

For the MC2500:

StepRange 1

The typical time before the device is ready for communication after a Reset is 1ms.

see

ResetEventStatus 34h

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

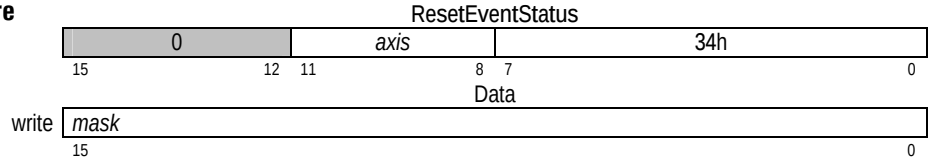
Syntax

ResetEventStatus *axis mask*

Arguments

Name	*Instance*	*Encoding*	*Bit Number*
axis	Axis1	0	
	Axis2	1	
	Axis3	2	
	Axis4	3	
mask	Motion complete	0001h	0
	Wrap-around	0002h	1
	Breakpoint 1	0004h	2
	Capture received	0008h	3
	Motion error	0010h	4
	In positive limit	0020h	5
	In negative limit	0040h	6
	Instruction error	0080h	7
	Commutation error	0800h	11
	Breakpoint 2	4000h	14

Packet structure



Description

ResetEventStatus clears (sets to 0), for the specified *axis*, each bit in the event status register that has a value of 0 in the *mask* sent with this command. All other event status register bits (bits which have a mask value of 1) are unaffected.

Restrictions

see

GetEventStatus

SetAcceleration buffered 90h
GetAcceleration 4Ch

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

Syntax

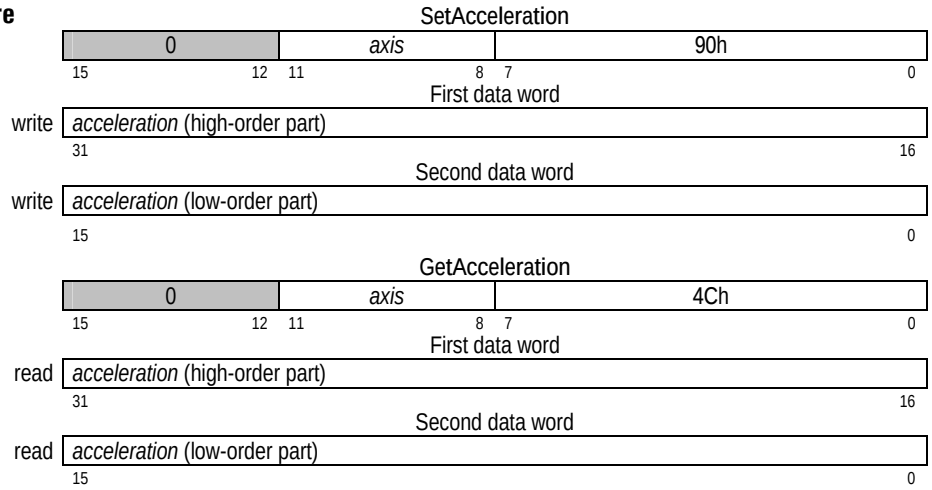
SetAcceleration *axis acceleration*
GetAcceleration *axis*

Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3

	Type	*Range*	*Scaling*	*Units*
acceleration	unsigned 32 bits	0 *to* 2^{31}-1	$1/2^{16}$	counts/cycle^2 steps/cycle^2

Packet structure



Description

SetAcceleration loads the maximum acceleration buffer register for the specified *axis*. This command is used with the trapezoidal, velocity contouring, and S-curve profiling modes.

GetAcceleration reads the maximum acceleration buffer register.

Scaling example: To load a value of 1.750 counts/cycle^2 multiply by 65,536 (giving 114,688), and load the resulting number as a 32-bit number; giving 0001 in the high word and C000h in the low word. Values returned by GetAcceleration must be divided by 65,536 to convert to units of counts/cycle^2 or steps/cycle^2.

Restrictions

SetAcceleration may not be issued while an axis is in motion with the S-curve profile.

SetAcceleration is not valid in electronic gearing profile mode. SetAcceleration is a buffered command. The value set using this command will not take effect until the next Update or MultiUpdate instruction.

see

Set/GetDeceleration, Set/GetJerk, Set/GetPosition, Set/GetVelocity, MultiUpdate, Update

SetActualPosition 4Dh
GetActualPosition 37h

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

Syntax

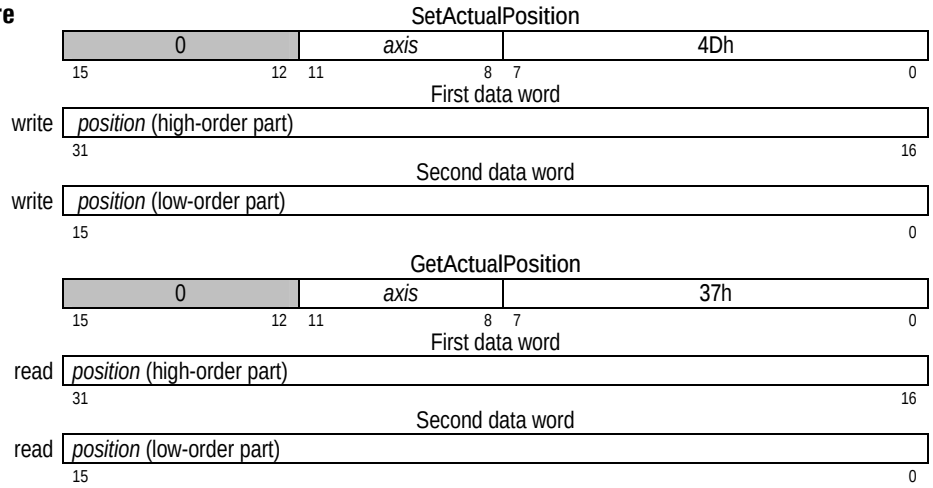
SetActualPosition *axis position*
GetActualPosition *axis*

Arguments

Name	***Instance***	***Encoding***
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3

	Type	***Range***	***Scaling***	***Units***
position	signed 32 bits	-2^{31} *to* $2^{31}-1$	unity	counts steps

Packet structure



Description

SetActualPosition loads the position register (encoder position) for the specified *axis*. At the same time, the commanded position is replaced by the loaded value minus the position error. This prevents a servo "bump" when the new axis position is established. The destination position (see SetPosition) is also modified by this amount so that no trajectory motion will occur when the update instruction is issued. In effect, this instruction establishes a new reference position from which subsequent positions can be calculated. It is commonly used to set a known reference position after a homing procedure. SetActualPosition takes effect immediately, it is not buffered.

Note: On the MC2400 and MC2500 series, the position error is zeroed.

GetActualPosition reads the contents of the encoder's actual position register. This value will be the result of the last encoder input, which will be accurate to within one cycle (as determined by Set/GetSampleTime).

Restrictions

see GetPositionError, Set/GetActualPositionUnits, AdjustActualPosition

SetActualPositionUnits BEh
GetActualPositionUnits BFh

Motor types

		Microstepping		Pulse & Direction
		MC2400		MC2500

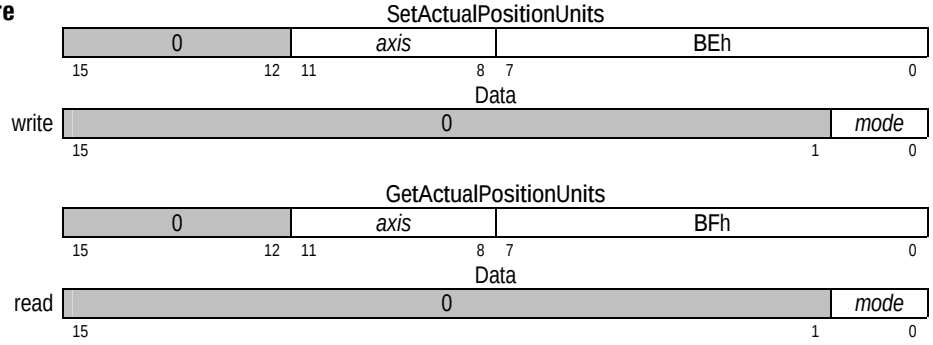
Syntax

SetActualPositionUnits *axis mode*
GetActualPositionUnits *axis*

Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3
mode	Counts	0
	Steps	1

Packet structure



Description

SetActualPositionUnits determines the units used by the Set/GetActualPosition, AdjustActualPosition and GetCaptureValue for the specified *axis*. When set to Counts, position units are in encoder counts. When set to Steps, position units are in steps. The step position is calculated using the ratio as set by the SetEncoderToStepRatio command.

GetActualPositionUnits returns the position units for the specified *axis*.

Restrictions

see

Set/GetActualPosition, Set/GetEncoderToStepRatio, AdjustActualPosition, GetCaptureValue

SetAutoStopMode D2h
GetAutoStopMode D3h

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

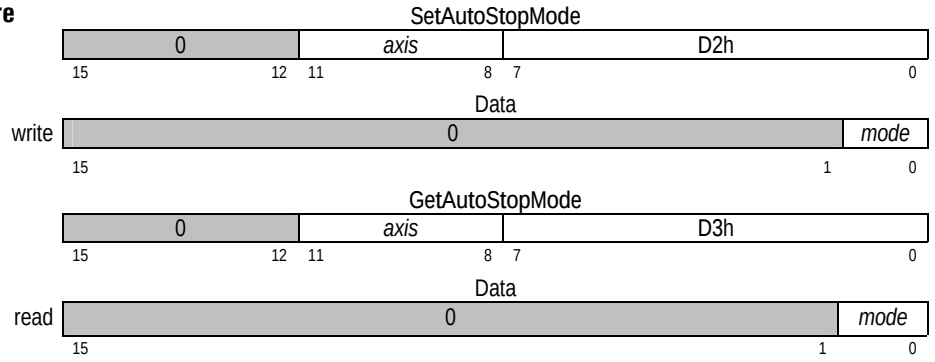
Syntax

SetAutoStopMode *axis mode*
GetAutoStopMode *axis*

Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3
mode	Disable	0
	Enable	1

Packet structure



Description

SetAutoStopMode determines the action to be taken when the motion error bit in the event status register becomes set. If the motion error bit is set, and auto stop mode is enabled, then the *axis* goes into an open-loop mode. This is the equivalent of a SetMotorMode Off command. When auto stop is disabled, the *axis* is not affected by a motion error. Auto stop mode has no effect on the motion error bit of the event status register. The motion error bit always becomes active if the position error limit is exceeded; independent of the setting of the auto stop mode.

GetAutoStopMode returns the state of the auto-stop mode.

Restrictions

When the encoder source is set to none (SetEncoderSource None), setting the auto stop mode to Enable will not stop motion in the event that the position error limit is exceeded.

see

GetEventStatus, SetPositionErrorLimit

SetAxisMode 87h
GetAxisMode 88h

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

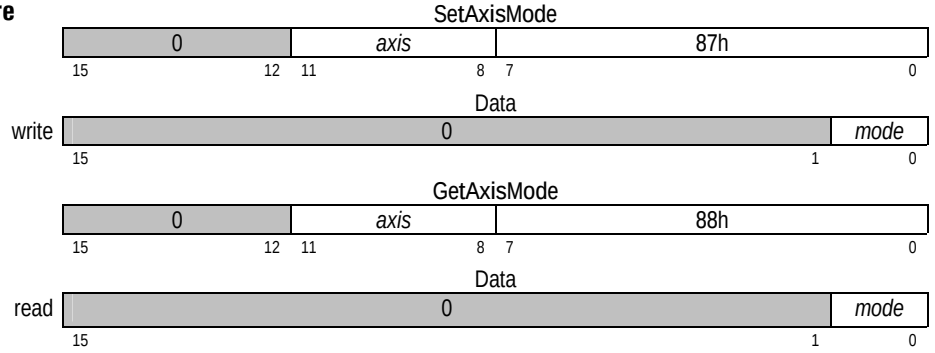
Syntax

SetAxisMode *axis mode*
GetAxisMode *axis*

Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3
mode	Off	0
	On	1

Packet structure



Description

SetAxisMode enables (On) or disables (Off) the specified *axis*. A disabled axis will not respond to profile or other motion commands.

GetAxisMode returns the mode of the specified *axis*.

Restrictions

Disabled axes do not provide encoder feedback. If it is desired that an axis provide encoder feedback even though no profiling or servo control is to be used, that axis must remain enabled.

see

SetAxisOutSource EDh
GetAxisOutSource EEh

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

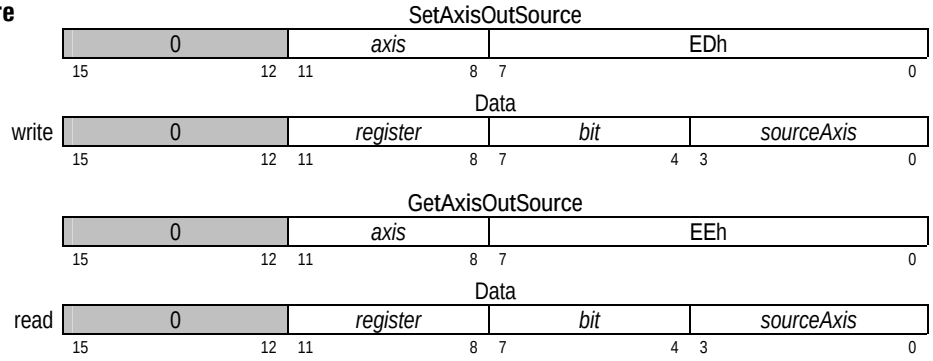
Syntax

SetAxisOutSource *axis sourceAxis bit register*
GetAxisOutSource *axis*

Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3
sourceAxis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3
bit	*see table on next page*	0 to 15
register	Disabled	0
	EventStatus	1
	ActivityStatus	2
	SignalStatus	3

Packet structure



Description

SetAxisOutSource maps the specified ***bit*** of the specified status ***register*** of ***axis*** to the AxisOut pin for the specified ***axis***. The state of the AxisOut pin will thereafter track the state of bit. If ***register*** is absent (encoding of 0), ***bit*** is ignored, and the specified AxisOut pin is, in effect, turned off (inactive). When the AxisOutSource is set to disabled, the AxisOut signal can be set high or low using SetSignalSense bit 10.

GetAxisOutSource returns the mapping of the AxisOut pin of ***axis***.

The following table shows the corresponding value for combinations of ***bit*** and *register*.

bit	event status register	activity status register	signal status register
0	Motion Complete	Phasing Initialized	Encoder A
1	Wrap-around	At maximum velocity	Encoder B
2	Breakpoint 1	Tracking	Encoder index
3	Position capture		Home
4	Motion error		Positive limit
5	In positive limit		Negative limit
6	In negative limit		AxisIn
7	Instruction error	Axis settled	Hall sensor 1
8		Motor on/off	Hall sensor 2
9		Position capture	Hall sensor 3
0Ah		In motion	
0Bh	Commutation error	In positive limit	
0Ch		In negative limit	
0Dh			
0Eh	Breakpoint 2		
0Fh			

Restrictions

see SetSignalSense

SetBreakPoint D4h
GetBreakPoint D5h

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

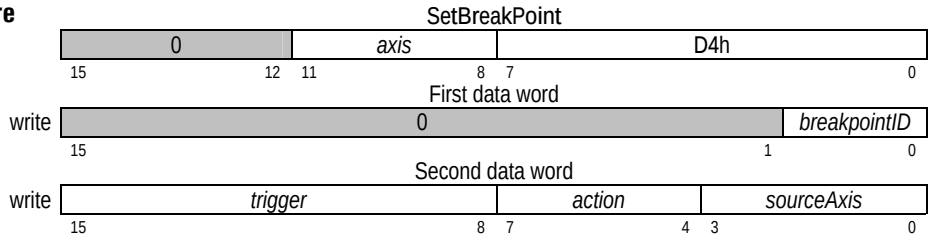
Syntax

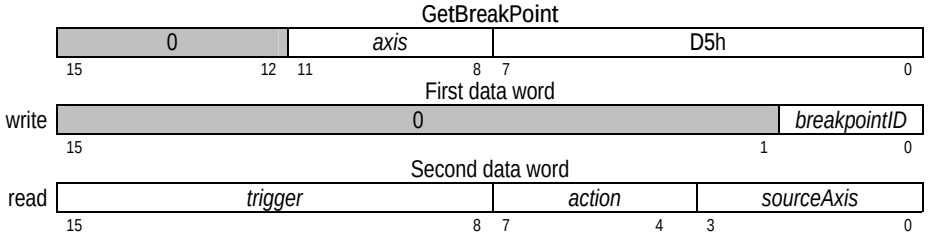
SetBreakPoint *axis breakpointID sourceAxis action trigger*
GetBreakPoint *axis breakpointID*

Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3
breakpointID	Breakpoint1	0
	Breakpoint2	1
sourceAxis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3
action	(none)	0
	Update	1
	AbruptStop	2
	SmoothStop	3
	MotorOff	4
trigger	(none)	0
	GreaterOrEqualCommandedPosition	1
	LesserOrEqualCommandedPosition	2
	GreaterOrEqualActualPosition	3
	LesserOrEqualActualPosition	4
	CommandedPositionCrossed	5
	ActualPositionCrossed	6
	Time	7
	EventStatus	8
	ActivityStatus	9
	SignalStatus	Ah

Packet structure





Description

SetBreakPoint establishes a breakpoint for the specified *axis* to be triggered by a condition or event on *sourceAxis*, which may be the same as or different from *axis*. Two concurrent breakpoints may be set for each axis. Each breakpoint may have its own breakpoint type and comparison value. The *breakpointID* field specifies which breakpoint the SetBreakPoint and GetBreakPoint commands will address.

The six Position breakpoints and the Time breakpoint are *threshold-triggered*; the breakpoint occurs when the indicated value reaches or crosses a threshold. The Status breakpoints are *level-triggered*; the breakpoint occurs when a specific bit or combination of bits in the indicated status register changes state. Thresholds and bit specifications are both set by the SetBreakpointValue instruction.

action determines what the Navigator does when the breakpoint occurs, as shown in the following table.

Action	Resultant command sequence
None	No action
Update	Update *axis*
AbruptStop	The profile executes an abrupt stop
SmoothStop	The profile executes a smooth stop
MotorOff	SetMotorMode *axis*, Off

GetBreakPoint returns the trigger, action, and axis for the specified breakpoint (1 or 2) of the indicated axis. When a breakpoint occurs, the trigger value will be reset to none. The CommandedPositionCrossed and the ActualPositionCrossed triggers are converted to one of the Position trigger types 1-4; depending on the current position when the command is issued.

Restrictions

Always load the breakpoint comparison value (SetBreakPoint command) before setting a new breakpoint condition (SetBreakPointValue command). This is because as soon as the breakpoint condition is set, the chipset will start using the breakpoint value register. Failure to do so will result in unexpected behavior.

see

Set/GetBreakPointValue

SetBreakpointValue D6h
GetBreakpointValue D7h

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

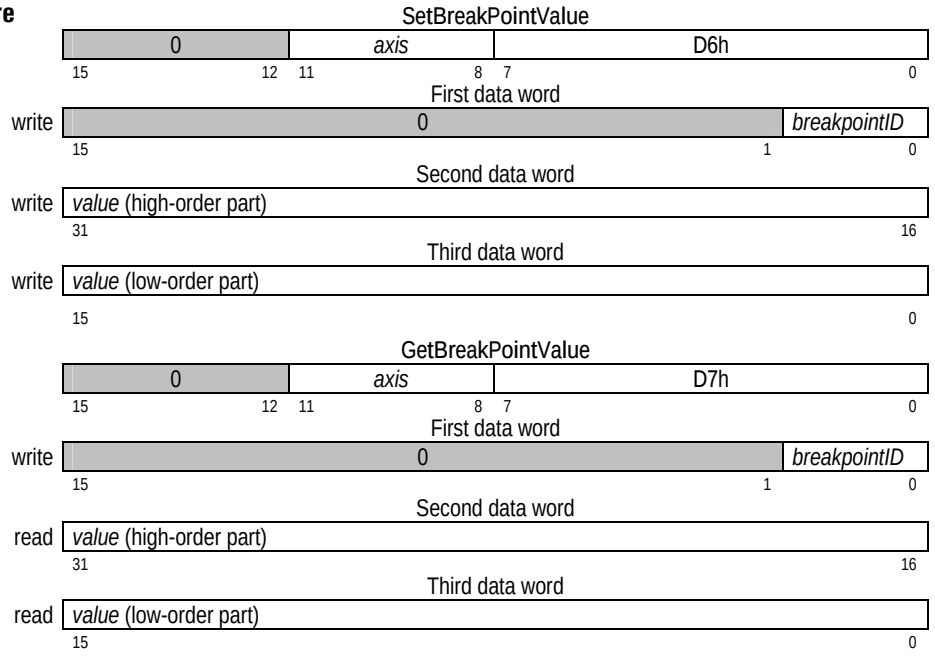
Syntax

SetBreakPointValue *axis breakpointID value*
GetBreakPointValue *axis breakpointID*

Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3
breakpointID	Breakpoint1	0
	Breakpoint2	1
value	*see table on next page*	

Packet structure



Description

SetBreakPointValue sets the breakpoint comparison value for the specified *axis*. For the position and time breakpoints, this is a threshold comparison value. The data format for each trigger condition is shown in the following table.

Breakpoint Trigger	Value Type	Range	Units
GreaterOrEqualCommandedPosition	signed 32 bits	-2^{31} to $2^{31}-1$	counts
LesserOrEqualCommandedPosition	signed 32 bits	-2^{31} to $2^{31}-1$	counts
GreaterOrEqualActualPosition	signed 32 bits	-2^{31} to $2^{31}-1$	counts
LesserOrEqualActualPosition	signed 32 bits	-2^{31} to $2^{31}-1$	counts
CommandedPositionCrossed	signed 32 bits	-2^{31} to $2^{31}-1$	counts
ActualPositionCrossed	signed 32 bits	-2^{31} to $2^{31}-1$	counts
Time	unsigned 32 bits	0 to $2^{32}-1$	cycles
EventStatus	2 word mask	-	Boolean status values
ActivityStatus	2 word mask	-	Boolean status values
SignalStatus	2 word mask	-	Boolean status values

For level-triggered breakpoints, the high-order part of *value* is the selection mask, and the low-order word is the sense mask. For each selection bit that is set to 1, the corresponding bit of the specified status register is conditioned to cause a breakpoint when it changes state. The sense-mask bit determines which state causes the break. If it is 1, the corresponding status-register bit will cause a break when it is set to 1. If it is 0, the status-register bit will cause a break when it is set to 0.

For example, assume that it is desired that the breakpoint type will be set to EventStatus, and that a breakpoint should be recognized whenever the motion complete bit (bit 0 of event status register) is set to 1, or the commutation error bit (bit 11 of event status register) is set to 0. In this situation, the high and low words for *value* would be high word: 0x801 (hex), and low word: 1.

GetBreakPointValue returns the breakpoint value for the specified *breakpointID*.

Two completely separate breakpoints are supported, each of which may have its own breakpoint type and comparison value. The *breakpointID* field specifies which breakpoint the SetBreakPointValue and GetBreakPointValue commands will address.

Restrictions

Always load the breakpoint comparison value (SetBreakPointValue command) before setting a new breakpoint condition (SetBreakPoint command). This is because as soon as the breakpoint condition is set, the chipset will start using the breakpoint value register. Failure to do so will result in unexpected behavior.

see

Set/GetBreakPoint

SetBufferLength C2h
GetBufferLength C3h

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

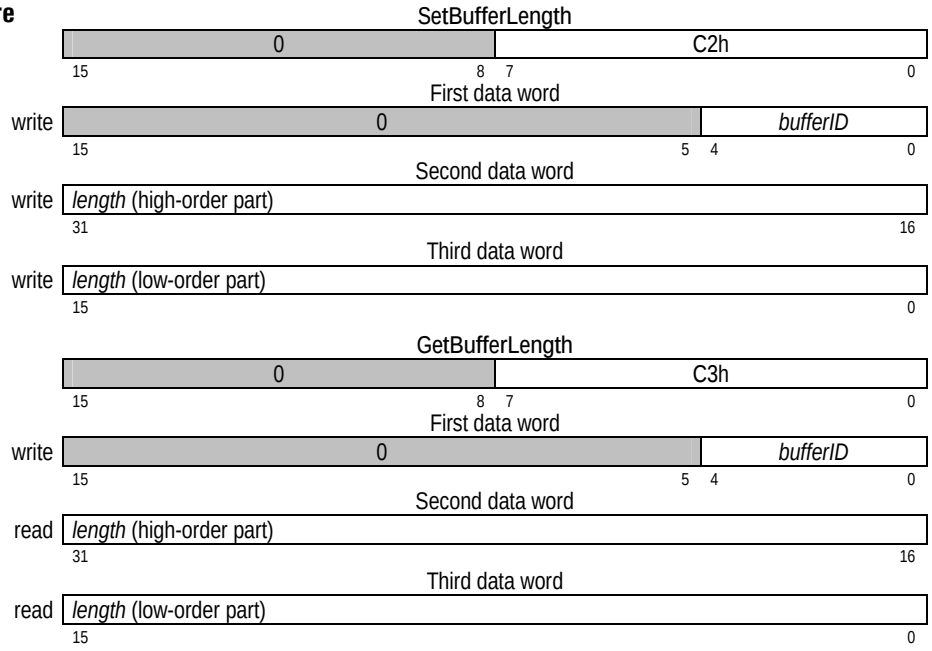
Syntax

SetBufferLength *bufferID length*
GetBufferLength *bufferID*

Arguments

Name	*Type*	*Range*
bufferID	unsigned 16 bits	0 to 31
length	unsigned 32 bits	1 to 2^{30}-1

Packet structure



Description

SetBufferLength sets the length, in number of 32-bit elements, of the buffer in the memory block identified by *bufferID*.

Note: SetBufferLength resets the buffers' read and write indexes to 0.

GetBufferLength returns the length of the specified buffer.

Restrictions

see Set/GetBufferReadIndex; Set/GetBufferStart; Set/GetBufferWriteIndex

SetBufferReadIndex C6h
GetBufferReadIndex C7h

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

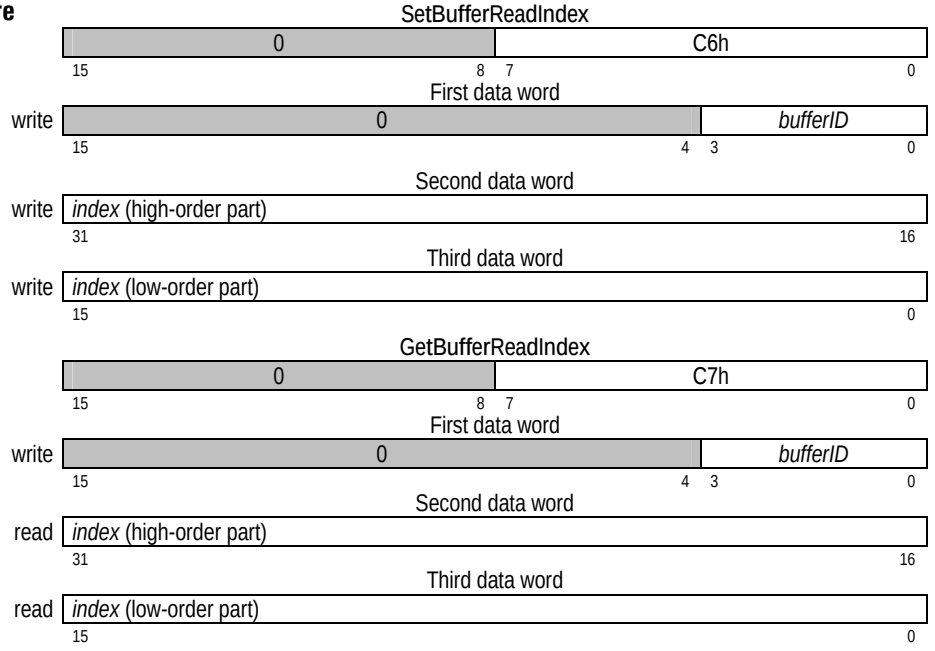
Syntax

SetBufferReadIndex *bufferID index*
GetBufferReadIndex *bufferID*

Arguments

Name	***Type***	***Range***	***Scaling***	***Units***
bufferID	unsigned 16 bits	0 *to* 31	unity	-
index	unsigned 32 bits	0 to buffer length-1	unity	double words (32 bits)

Packet structure



Description

SetBufferReadIndex sets the read index for the specified bufferID.

GetBufferReadIndex returns the read index for the specified bufferID.

Restrictions

If the read index is set to an address beyond the length of the buffer, the command will not be executed and will return host I/O error code 7, buffer bound exceeded.

see

Set/GetBufferLength, Set/GetBufferStart, Set/GetBufferWriteIndex

SetBufferStart C0h
GetBufferStart C1h

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

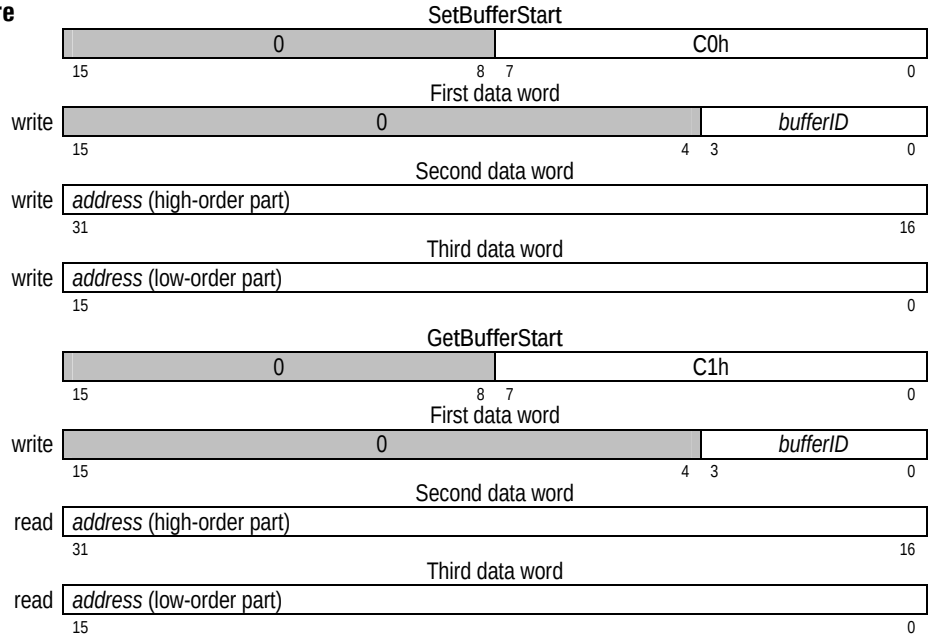
Syntax

SetBufferStart *bufferID address*
GetBufferStart *bufferID*

Arguments

Name	*Type*	*Range*	*Units*
bufferID	unsigned 16 bits	0 *to* 31	-
address	unsigned 32 bits	0 *to* 2^{31}-1	double words (32 bit)

Packet structure



Description

SetBufferStart sets the starting address, in double-words, for the buffer in the memory block identified by bufferID.

Note: SetBufferStart resets the buffers read and write indexes to 0.

GetBufferStart returns the starting address for the specified bufferID.

Restrictions

see Set/GetBufferLength, Set/GetReadIndex, Set/GetBufferWriteIndex

SetBufferWriteIndex C4h
GetBufferWriteIndex C5h

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

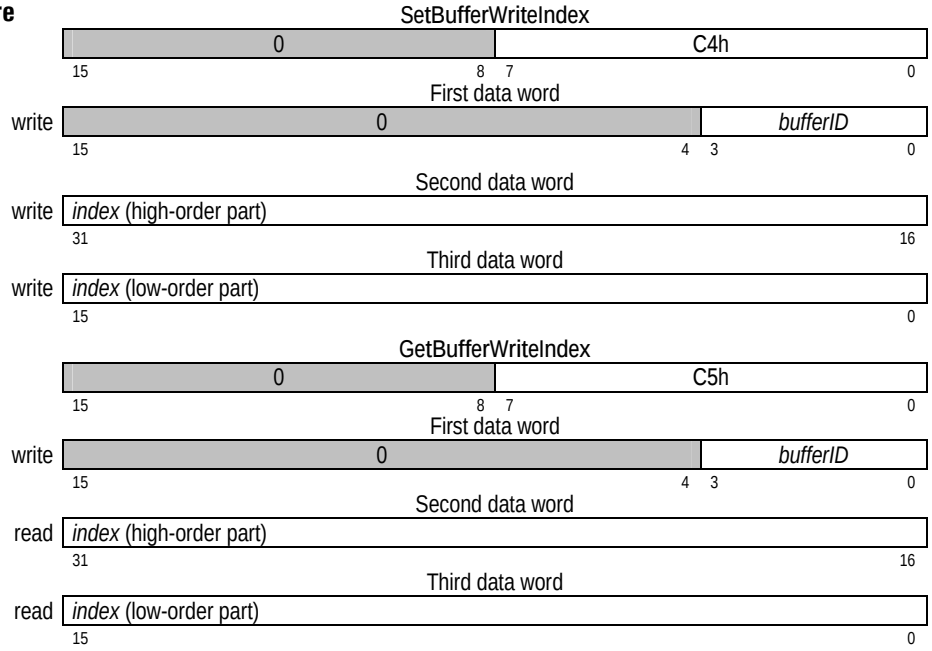
Syntax

SetBufferWriteIndex *bufferID index*
GetBufferWriteIndex *bufferID*

Arguments

Name	*Type*	*Range*	*Scaling*	*Units*
bufferID	unsigned 16 bits	0 *to* 31	unity	-
index	unsigned 32 bits	0 *to* buffer length-1	unity	double words (32 bits)

Packet structure



Description

SetBufferWriteIndex sets the write index for the specified bufferID. If the write index is set to an address beyond the length of the buffer, the command will not be executed and will return an error.

GetBufferWriteIndex returns the current write index for the specified bufferID.

Restrictions

see

Set/GetBufferLength, Set/GetBufferReadIndex, Set/GetBufferStart

SetCaptureSource D8h
GetCaptureSource D9h

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

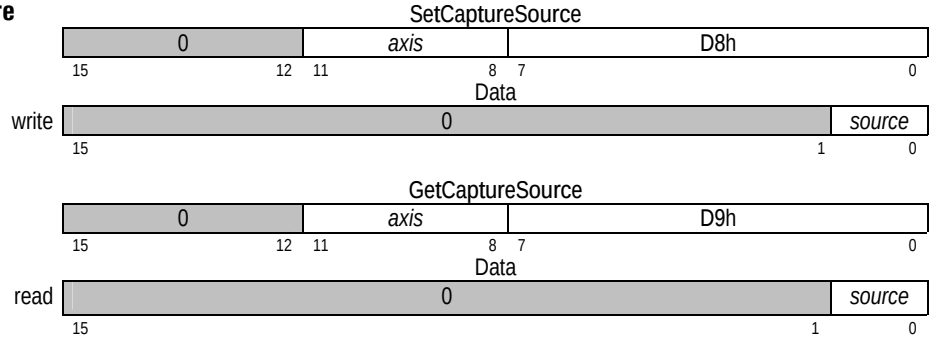
Syntax

SetCaptureSource *axis source*
GetCaptureSource *axis*

Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3
source	Index	0
	Home	1

Packet structure



Description

SetCaptureSource determines which of two encoder signals, Index or Home, is used to trigger the high-speed capture of the axis position for the specified *axis*.

GetCaptureSource returns the capture signal *source* for the selected *axis*.

Restrictions

see

GetCaptureValue

SetCommutationMode E2h
GetCommutationMode E3h

Motor types

	Brushless DC			
	MC2300	MC2800		

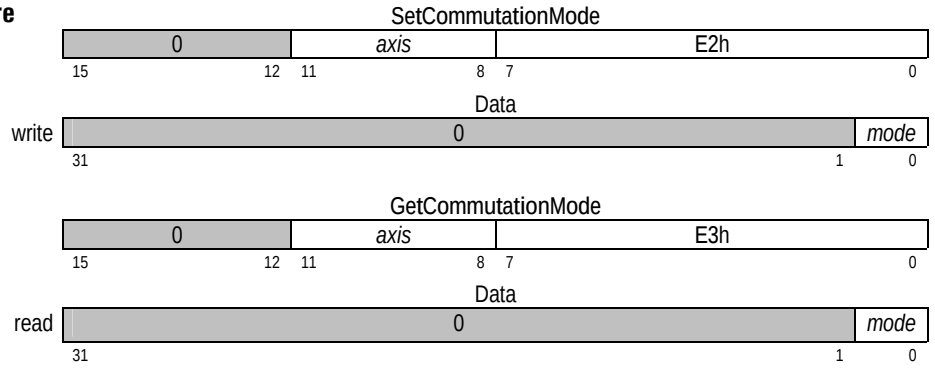
Syntax

SetCommutationMode *axis mode*
GetCommutationMode *axis*

Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3
mode	Sinusoidal	0
	Hall-Based	1

Packet structure



Description

SetCommutationMode sets the phase commutation mode for the specified *axis*. When set to sinusoidal, as the motor turns, the encoder input signals are used to calculate the phase angle. This angle is in turn used to generate sinusoidally varying outputs to each motor winding.

When set to Hall-based, the hall effect sensor inputs are used to commutate the motor windings using a "six-step" or "trapezoidal" waveform method.

GetCommutationMode returns the value of the commutation mode.

Restrictions

see

Set/GetPhasePrescale, Set/GetPhaseCounts

SetDeceleration buffered 91h
GetDeceleration 92h

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

Syntax

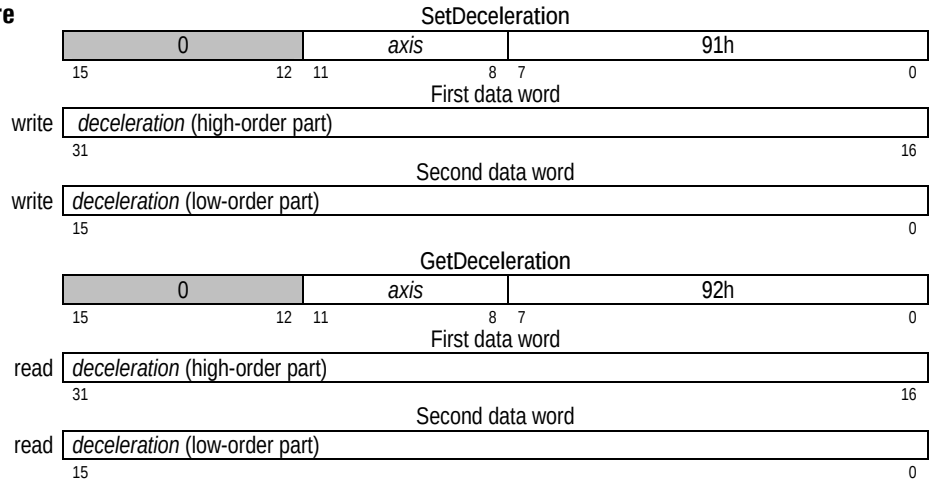
SetDeceleration *axis deceleration*
GetDeceleration *axis*

Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3

	Type	*Range*	*Scaling*	*Units*
deceleration	unsigned 32 bits	0 *to* 2^{31}-1	$1/2^{16}$	counts/cycle^2 steps/cycle^2

Packet structure



Description

SetDeceleration loads the maximum deceleration buffer register for the specified *axis*.

GetDeceleration returns the value of the maximum deceleration buffer.

Scaling example: To load a value of 1.750 counts/cycle^2, multiply by 65,536 (giving 114,688), and load the resulting number as a 32-bit number, giving 0001 in the high word, and C000h in the low word. Retrieved numbers (GetDeceleration) must be divided by 65,536 to convert to units of counts/cycle^2 or steps/cycle^2.

Restrictions

This is a buffered command. The new value set will not take effect until the next Update or MultiUpdate instruction is entered. These commands are used with the trapezoidal and velocity contouring profile modes. They are not used with the S-curve or the electronic gearing profile mode.

Note: If deceleration is set to zero, then the value specified for acceleration (SetAcceleration) will automatically be used to set the magnitude of deceleration.

see

Set/GetAcceleration, Set/GetJerk, Set/GetPosition, Set/GetVelocity, MultiUpdate, Update

SetDerivativeTime buffered 9Ch
GetDerivativeTime 9Dh

Motor types

DC Brush		Brushless DC			
MC2100	MC2800	MC2300	MC2800		

Syntax

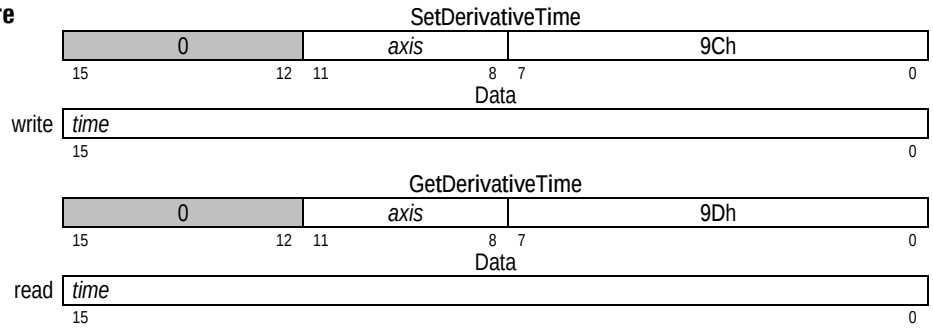
SetDerivativeTime *axis time*
GetDerivativeTime *axis*

Arguments

Name	***Instance***	***Encoding***
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3

	Type	***Range***	***Scaling***	***Units***
time	unsigned 16 bits	1 *to* 2^{15}-1	unity	cycles

Packet structure



Description

SetDerivativeTime sets the sampling time, in number of servo cycles, for the servo filter to use in calculating the derivative term for the specified *axis*.

GetDerivativeTime returns the derivative sampling time.

Restrictions

SetDerivativeTime is a buffered command. The value set using this command will not take effect until the next Update or MultiUpdate instruction.

This command does not effect the overall cycle time of the chipset; only the derivative sampling time. The overall cycle time of the motion processor is set using the command SetSampleTime.

see

GetDerivative, GetIntegral, MultiUpdate, Update

SetDiagnosticPortMode 89h
GetDiagnosticPortMode 8Ah

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

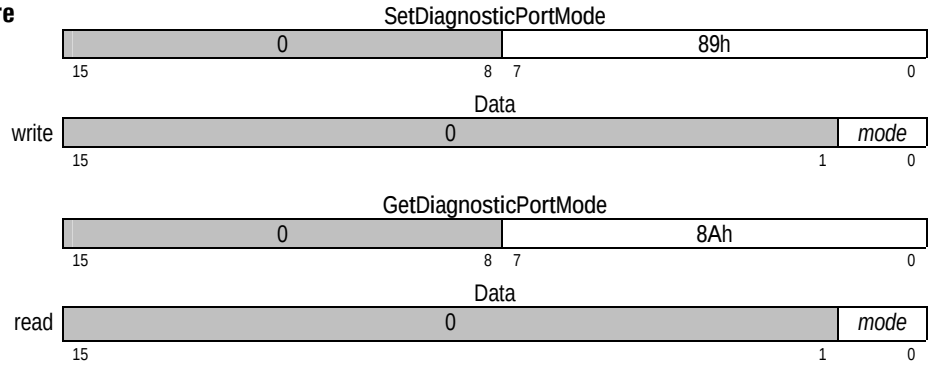
Syntax

SetDiagnosticPortMode *mode*
GetDiagnosticPortMode

Arguments

Name	*Instance*	*Encoding*
mode	Limited	0
	Full	1

Packet structure



Description

SetDiagnosticPortMode determines the instruction set that can be executed through the diagnostic (serial) port. When set to Limited, only the following instructions may be executed:

all Get instructions

The SetBufferReadIndex instruction

When set to Full, all instructions may be executed.

GetDiagnosticPortMode returns the current mode of the diagnostic port.

Restrictions

See

Set/GetSerialPortMode

SetEncoderModulus 8Dh
GetEncoderModulus 8Eh

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

Syntax

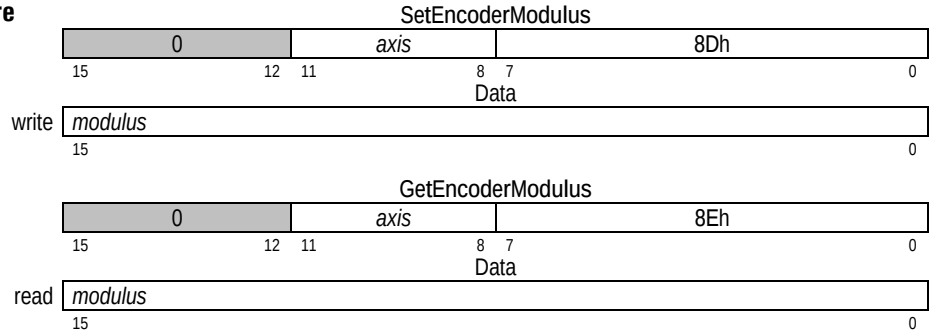
SetEncoderModulus *axis modulus*
GetEncoderModulus *axis*

Arguments

Name	***Instance***	***Encoding***
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3

	Type	***Range***	***Scaling***	***Units***
modulus	unsigned 16 bits	0 *to* 2^{16}-1	unity	counts

Packet structure



Description

SetEncoderModulus sets the parallel word range for the specified *axis* when parallel-word feedback is used. *Modulus* determines the range of the connected device. For multi-turn systems, this value is used to determine when a position wrap condition has occurred. The value provided should be one-half of the actual range of the axis. For example, if the parallel-word input is used with a linear potentiometer connected to an external A/D (analog-to-digital converter) which has 12 bits of resolution, then the total range is 4,096, and a value of 2,048 should be loaded with this command.

GetEncoderModulus returns the encoder modulus.

Restrictions

A value for encoder modulus is only required when the encoder source is set to parallel.

see

Set/GetEncoderSource

SetEncoderSource DAh
GetEncoderSource DBh

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

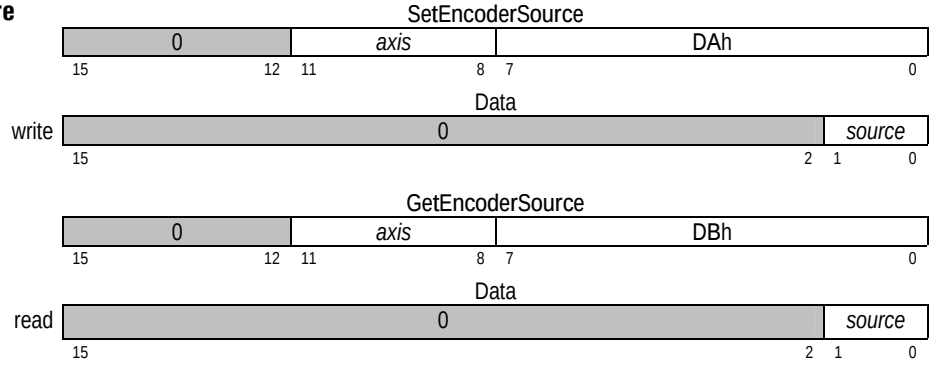
Syntax

SetEncoderSource *axis source*
GetEncoderSource *axis*

Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3
source	Incremental	0
	Parallel	1
	None	2

Packet structure



Description

SetEncoderSource sets the type of feedback (incremental quadrature encoder or parallel-word) for the specified *axis*. When incremental quadrature is selected, the motion processor expects A and B quadrature signals to be input at the quad A/B axis inputs. When parallel-word is selected, the motion processor expects user-defined external circuitry connected to the chipset's external bus to load a 16-bit word containing the current position value for the selected axis. External feedback devices with less than 16 bits may be used, but the unused bits must be sign extended or zeroed.

GetEncoderSource returns the code for the current type of feedback.

Restrictions

A source value of None is only valid for microstepping and pulse & direction motors.

see

Set/GetEncoderModulus

SetEncoderToStepRatio DEh
GetEncoderToStepRatio DFh

Motor types

		Microstepping		Pulse & Direction
		MC2400		MC2500

Syntax

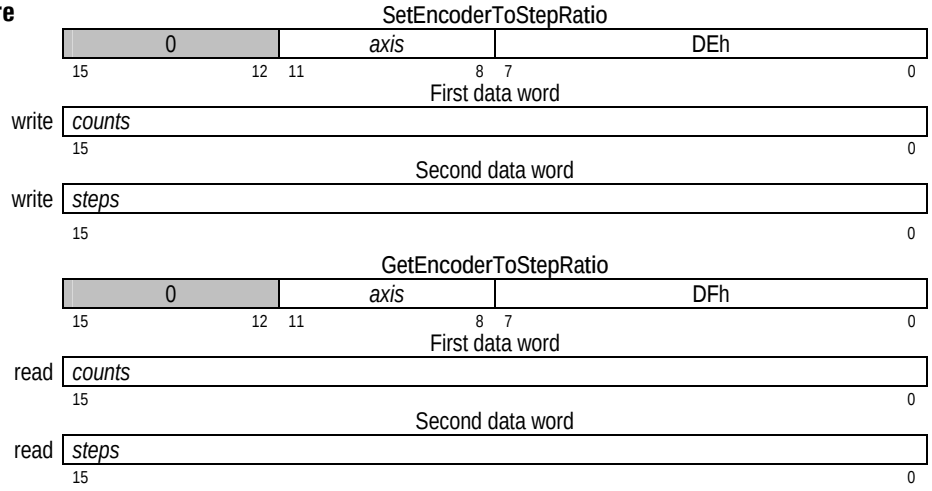
SetEncoderToStepRatio *axis counts steps*
GetEncoderToStepRatio *axis*

Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3

	Type	*Range*	*Scaling*	*Units*
counts	unsigned 16 bits	0 *to* 2^{15}-1	unity	counts
steps	unsigned 16 bits	0 *to* 2^{15}-1	unity	steps

Packet structure



Description

SetEncoderToStepRatio sets the ratio of the number of encoder counts to the number of output steps per motor rotation used by the motion processor to convert encoder counts into steps. *Counts* is the number of encoder counts per full rotation of the motor. *Steps* is the number of steps output by the motion processor per full rotation of the motor. Since this command sets a ratio, the parameters do not have to be for a full rotation as long as they correctly represent the encoder count to step ratio.

GetEncoderToStepRatio returns the ratio of the number of encoder counts to the number of output steps per motor rotation.

Restrictions

see Set/GetActualPositionUnits

SetGearMaster AEh
GetGearMaster AFh

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

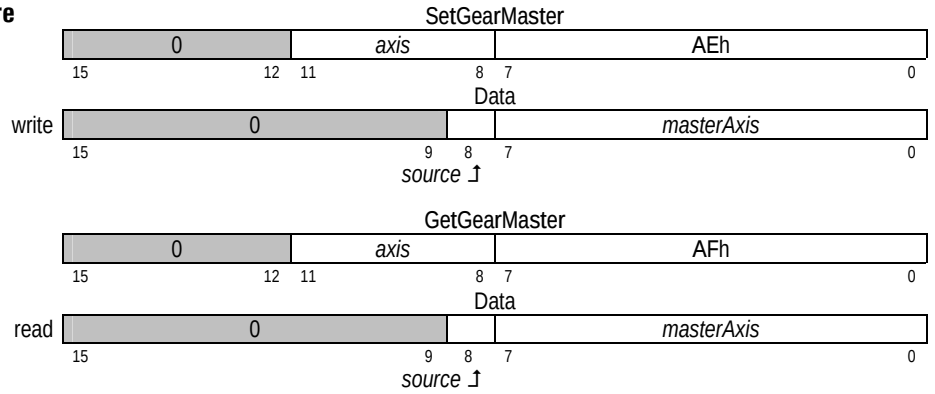
Syntax

SetGearMaster *axis masterAxis source*
GetGearMaster *axis*

Arguments

Name	*Instance*	*Encoding*	*Type*
axis	Axis1	0	unsigned 16 bits
	Axis2	1	
	Axis3	2	
	Axis4	3	
masterAxis	Axis1	0	
	Axis2	1	
	Axis3	2	
	Axis4	3	
source	Actual	0	
	Commanded	1	

Packet structure



Description

SetGearMaster establishes the slave (*axis*) and master (*masterAxis*) axes for the electronic-gearing profile; and sets the source, Actual or Commanded, of the master axis position data to be used.

The *masterAxis* determines the axis which will drive the slave axis. Both the slave and the master axes must be enabled (SetAxisMode command). The source determines whether the master axis' commanded position as determined by the trajectory generator will be used to drive the slave axis, or whether the master axis' encoder position will be used to drive the slave.

GetGearMaster returns the value for the geared axes and position source.

Restrictions

For electronic gear mode to operate properly, the master axis must be enabled.

see

Set/GetGearRatio

SetGearRatio buffered 14h
GetGearRatio 59h

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

Syntax

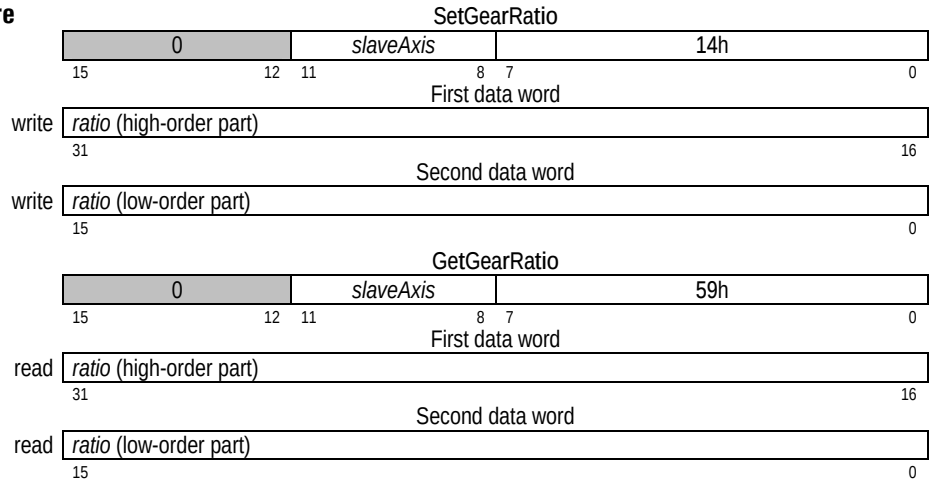
SetGearRatio *slaveAxis ratio*
GetGearRatio *slaveAxis*

Arguments

Name	***Instance***	***Encoding***
slaveAxis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3

	Type	***Range***	***Scaling***	***Units***
ratio	signed 32 bits	-2^{31} *to* $2^{31}-1$	$1/2^{16}$	SlaveCounts/ MasterCounts

Packet structure



Description

SetGearRatio sets the ratio between the master and slave axes for the electronic gearing profile for the current *axis*. Positive ratios cause the slave to move in the same direction as the master; negative ratios in the opposite direction. The specified ratio has a unity scaling of 65,536.

GetGearRatio returns the gear ratio set for the specified slave axis.

Scaling examples:

Ratio value	Resultant ratio
-32,768	.5 negative slave counts for each positive master count
1,000,000	15.259 positive slave counts for each positive master count
123	.0018 positive slave counts for each positive master count

Restrictions

This is a buffered command. The new value set will not take effect until the next Update or MultiUpdate instruction is entered.

See

Set/GetGearMaster, MultiUpdate, Update

SetIntegrationLimit buffered 95h
GetIntegrationLimit 96h

Motor types

DC Brush		Brushless DC			
MC2100	MC2800	MC2300	MC2800		

Syntax

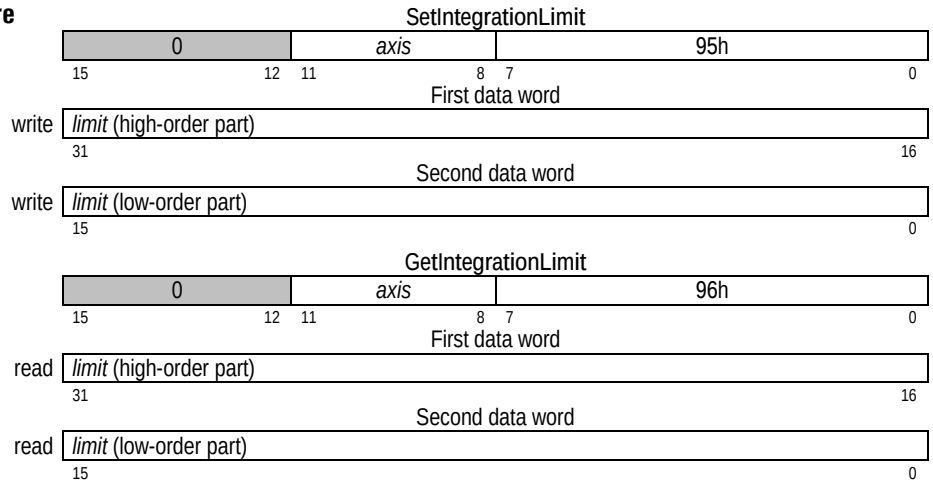
SetIntegrationLimit *axis limit*
GetIntegrationLimit *axis*

Arguments

Name	***Instance***	***Encoding***
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3

	Type	***Range***	***Scaling***	***Units***
limit	unsigned 32 bits	0 *to* 2^{31}-1	$1/2^8$	count*cycles

Packet structure



Description

SetIntegrationLimit loads the integration-limit register of the servo filter for the specified *axis*.

GetIntegrationLimit returns the value of the current integration limit.

Scaling example: The scaling is the same as the GetIntegral command. For example, a constant position error of 100 counts which is present for 256 cycles will result in an integral value of 100 (100*256/256) , and therefore an IntegrationLimit value of 100 will limit the total accumulated integration error to 25,600 count*cycles.

Restrictions

This is a buffered command. The value set using this command will not take effect until the next Update or MultiUpdate instruction.

see

GetIntegral, MultiUpdate, Update

SetInterruptMask 2Fh
GetInterruptMask 56h

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

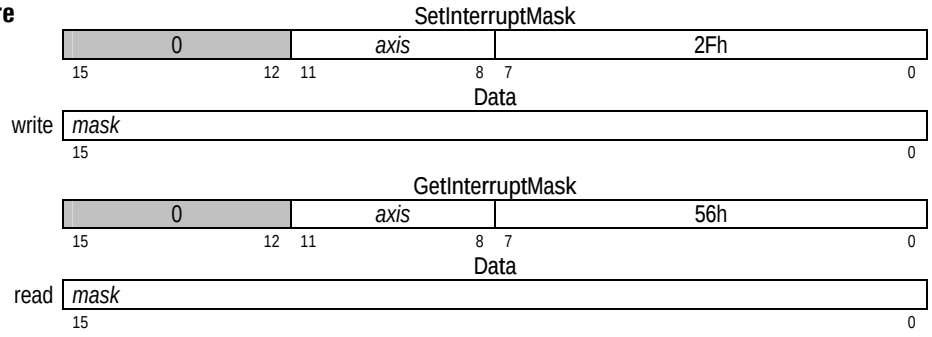
Syntax

SetInterruptMask *axis interruptMask*
GetInterruptMask *axis*

Arguments

Name	*Instance*	*Encoding*	*Bit Number*
axis	Axis1	0	
	Axis2	1	
	Axis3	2	
	Axis4	3	
mask	Motion complete	0001h	0
	Wrap-around	0002h	1
	Breakpoint 1	0004h	2
	Capture received	0008h	3
	Motion error	0010h	4
	In positive limit	0020h	5
	In negative limit	0040h	6
	Instruction error	0080h	7
	Commutation error	0800h	11
	Breakpoint 2	4000h	14

Packet structure



Description

SetInterruptMask determines which bits in the event status register of the specified axis will cause a host interrupt. For each interrupt mask bit that is set to 1, the corresponding event status register bit will cause an interrupt when that status register bit goes active (is set to 1). Interrupt mask bits set to 0 will not generate interrupts.

GetInterruptMask returns the mask for the specified axis.

Example: The interrupt mask value 28h will generate an interrupt when either the "in positive limit" bit or the "capture received" bit of the event status register goes active (set to 1).

Restrictions

see ClearInterrupt, GetInterruptAxis

SetJerk buffered 13h
GetJerk 58h

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

Syntax

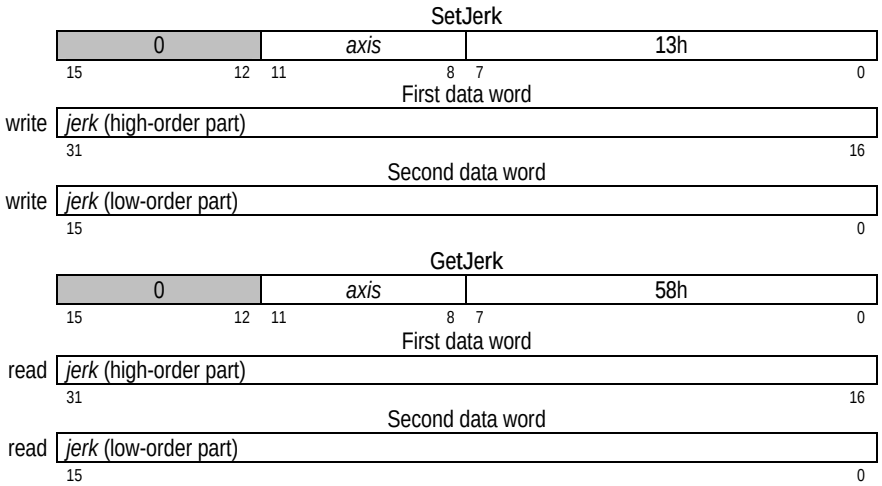
SetJerk *axis jerk*
GetJerk *axis*

Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3

	Type	*Range*	*Scaling*	*Units*
jerk	unsigned 32 bits	0 *to* 2^{31}-1	$1/2^{32}$	counts/cycle^3 steps/cycle^3

Packet structure



Description

SetJerk loads the jerk register in the parameter buffer for the specified *axis*.

GetJerk reads the contents of the jerk register.

Scaling example: To load a jerk value (rate of change of acceleration) of 0.012345 counts/cycle^3 (or steps/cycle^3), multiply by 2^{32} or 4,294,967,296. This gives a value to load of 53,021,371 (decimal), which corresponds to a high word of 0329h and a low word of 0ABBh when loading each word in hexadecimal.

Restrictions

SetJerk is a buffered command. The value set using this command will not take effect until the next Update or MultiUpdate instruction.

This command is used only with the S-curve profile mode. It is not used with the trapezoidal, velocity contouring, or electronic gear profile modes.

see

Set/GetAcceleration, Set/GetDeceleration, Set/GetPosition, Set/GetVelocity, MultiUpdate, Update

SetKaff buffered 93h
GetKaff 94h

Motor types

DC Brush		Brushless DC			
MC2100	MC2800	MC2300	MC2800		

Syntax

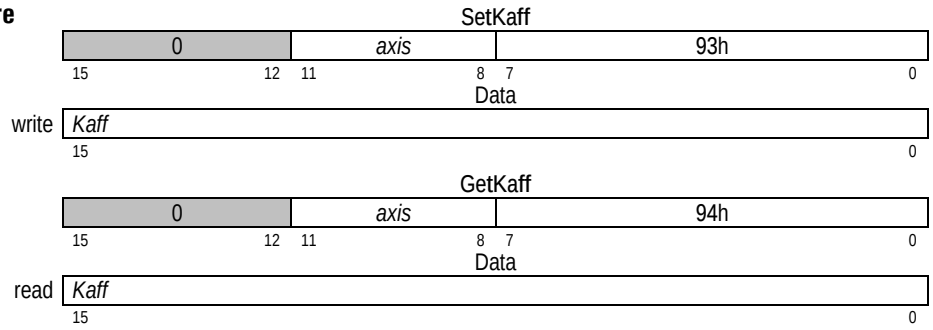
SetKaff *axis Kaff*
GetKaff *axis*

Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3

	Type	*Range*	*Scaling*
Kaff	unsigned 16 bits	0 *to* 2^{15}-1	unity

Packet structure



Description

SetKaff sets the acceleration feed-forward gain of the digital servo filter for the specified *axis*.

GetKaff reads the value of the acceleration feed-forward gain.

Restrictions

SetKaff is a buffered command. . The value set using this command will not take effect until the next Update or MultiUpdate instruction.

see

Set/GetKd, Set/GetKi, Set/GetKout, Set/GetKp, Set/GetKvff, MultiUpdate, Update

SetKd buffered 27h
GetKd 52h

Motor types

DC Brush		Brushless DC			
MC2100	MC2800	MC2300	MC2800		

Syntax

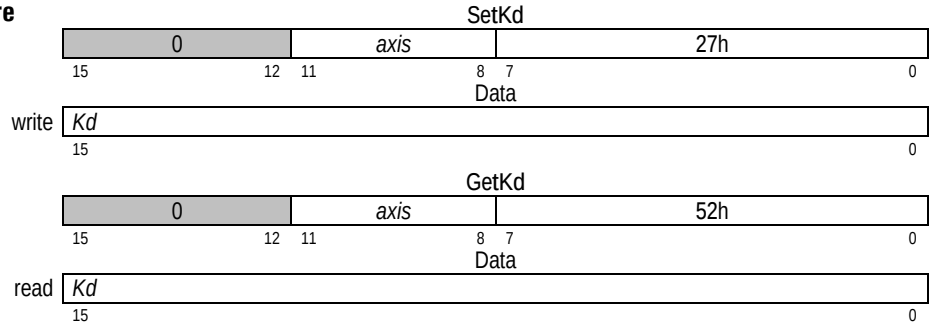
SetKd *axis Kd*
GetKd *axis*

Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3

	Type	*Range*	*Scaling*
Kd	unsigned 16 bits	0 *to* 2^{15}-1	unity

Packet structure



Description

SetKd sets the derivative gain of the digital servo filter for the specified axis.

GetKd reads the value of the derivative gain.

Restrictions

SetKd is a buffered command. The value set using this command will not take effect until the next Update or MultiUpdate instruction.

see

Set/GetKaff, Set/GetKi, Set/GetKout, Set/GetKp, Set/GetKvff, MultiUpdate, Update

SetKi buffered 26h
GetKi 51h

Motor types

DC Brush		Brushless DC			
MC2100	MC2800	MC2300	MC2800		

Syntax

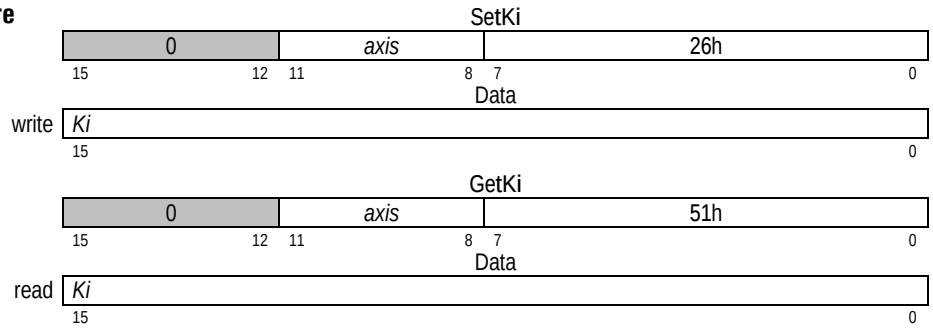
SetKi *axis Ki*
GetKi *axis*

Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3

	Type	*Range*	*Scaling*
Ki	unsigned 16 bits	0 *to* 2^{15}-1	unity

Packet structure



Description

SetKi sets the integral gain of the digital servo filter for the specified *axis*.

GetKi reads the value of the integral gain.

Restrictions

SetKi is a buffered command. The value set using this command will not take effect until the next Update or MultiUpdate instruction.

see

Set/GetKaff, Set/GetKd, Set/GetKout, Set/GetKp, Set/GetKvff, MultiUpdate, Update

SetKout 9Eh
GetKout 9Fh

Motor types

DC Brush		Brushless DC			
MC2100	MC2800	MC2300	MC2800		

Syntax

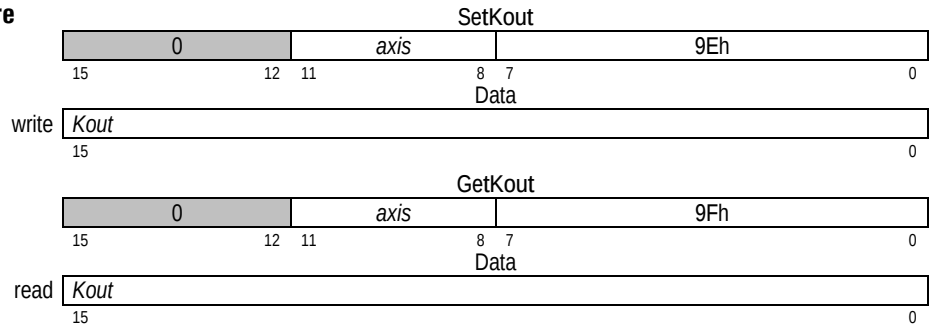
SetKout *axis Kout*
GetKout *axis*

Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3

	Type	*Range*	*Scaling*	*Units*
Kout	unsigned 16 bits	0 *to* 2^{16}-1	$100/2^{16}$	% output

Packet structure



Description

SetKout sets the output scale factor of the digital servo filter for the specified axis. The default value of Kout is 65535; approximately 100% output.

GetKout reads the value of the output scale factor.

Example:

To set the output scaling of the servo filter to 50%, set the Kout register to 32767.

Restrictions

This command is not buffered. It will take effect immediately after it is sent.

see

Set/GetKaff, Set/GetKd, Set/GetKi, Set/GetKp, Set/GetKvff

SetKp buffered 25h
GetKp 50h

Motor types

DC Brush		Brushless DC			
MC2100	MC2800	MC2300	MC2800		

Syntax

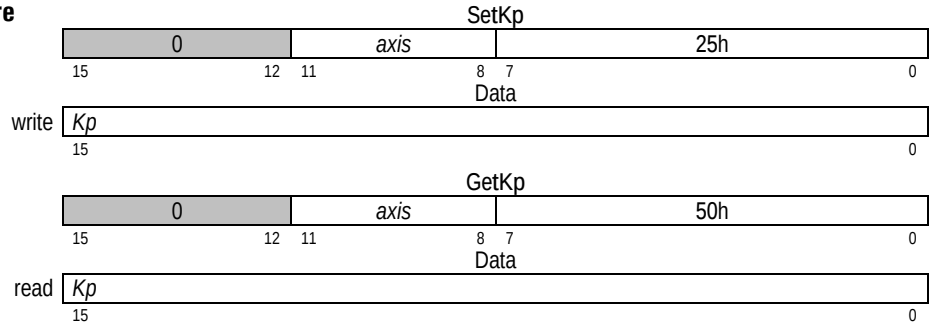
SetKp *axis Kp*
GetKp *axis*

Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3

	Type	*Range*	*Scaling*
Kp	unsigned 16 bits	0 *to* 2^{15}-1	unity

Packet structure



Description

SetKp sets the proportional gain of the digital servo filter for the specified *axis*.

GetKp reads the value of the proportional gain.

Restrictions

SetKp is a buffered command. The value set using this command will not take effect until the next Update or MultiUpdate instruction.

see

Set/GetKaff, Set/GetKd, Set/GetKi, Set/GetKout, Set/GetKvff, MultiUpdate, Update

SetKvff buffered 2Bh
GetKvff 54h

Motor types

DC Brush		Brushless DC			
MC2100	MC2800	MC2300	MC2800		

Syntax

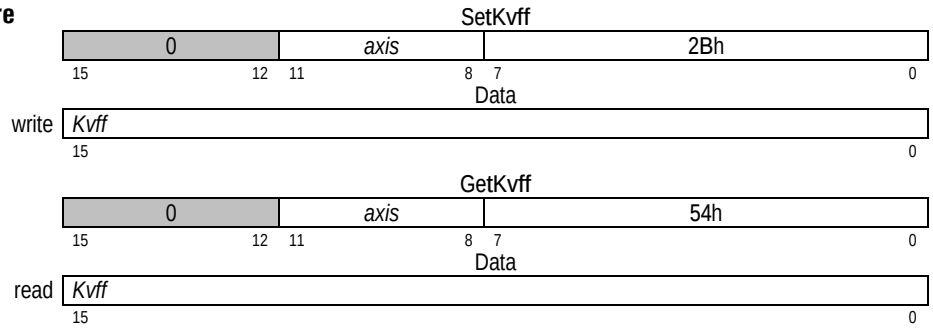
SetKvff *axis Kvff*
GetKvff *axis*

Arguments

Name	***Instance***	***Encoding***
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3

	Type	***Range***	***Scaling***
Kvff	unsigned 16 bits	0 *to* 2^{15}-1	unity

Packet structure



Description

SetKvff sets the velocity feed-forward gain of the digital servo filter for the specified *axis*.

GetKvff reads the value of the velocity feed-forward gain.

Restrictions

SetKvff is a buffered command. The value set using this command will not take effect until the next Update or MultiUpdate instruction.

see

Set/GetKaff, Set/GetKd, Set/GetKi, Set/GetKout, Set/GetKp, MultiUpdate, Update

SetLimitSwitchMode 80h
GetLimitSwitchMode 81h

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

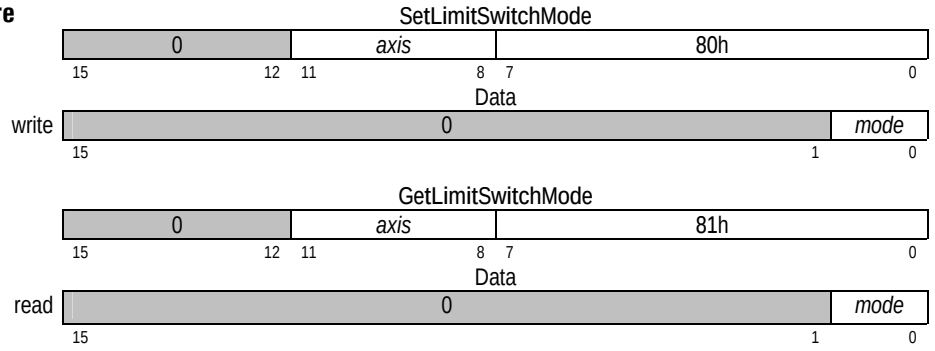
Syntax

SetLimitSwitchMode *axis mode*
GetLimitSwitchMode *axis*

Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3
mode	Off	0
	On	1

Packet structure



Description

SetLimitSwitchMode enables (On) or disables (Off) limit-switch sensing for the specified *axis*. When the mode is enabled, the axis will cause the corresponding limit-switch bits in the event status register and activity status register to be set when it enters either the positive or negative limit switches, and the axis will be brought to an abrupt stop. When it is disabled, the status bits are not set and the axis is not stopped; regardless of whether or not the axis is in a limit switch.

GetLimitSwitchMode returns the value for the state of the limit-sensing mode.

Restrictions

see

GetActivityStatus, GetEventStatus

SetMotionCompleteMode EBh
GetMotionCompleteMode ECh

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

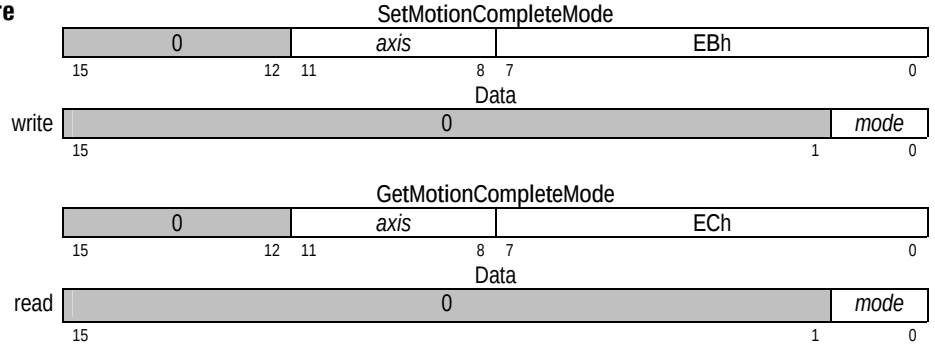
Syntax

SetMotionCompleteMode *axis mode*
GetMotionCompleteMode *axis*

Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3
mode	Commanded	0
	Actual	1

Packet structure



Description

SetMotionCompleteMode establishes the source for the comparison which determines the motion-complete status for the specified *axis*. When set to commanded mode, the motion is considered complete when the profile velocity reaches zero and no further motion will occur without an additional host command. This mode is unaffected by the actual encoder location.

When set to actual mode, the motion complete bit will be set when the above condition is true, and when the actual encoder position has been within the settle window (SetSettleWindow command) for the number of cycles specified by the SetSettleTime command. The settle timer is started at zero at the end of the trajectory profile motion, so at minimum a delay of SettleTime cycles will occur after the trajectory profile motion is complete.

GetMotionCompleteMode returns the value for the motion-complete mode.

Restrictions

see

Set/GetSettleTime, Set/GetSettleWindow

SetMotorBias 0Fh
GetMotorBias 2Dh

Motor types

DC Brush		Brushless DC			
MC2100	MC2800	MC2300	MC2800		

Syntax

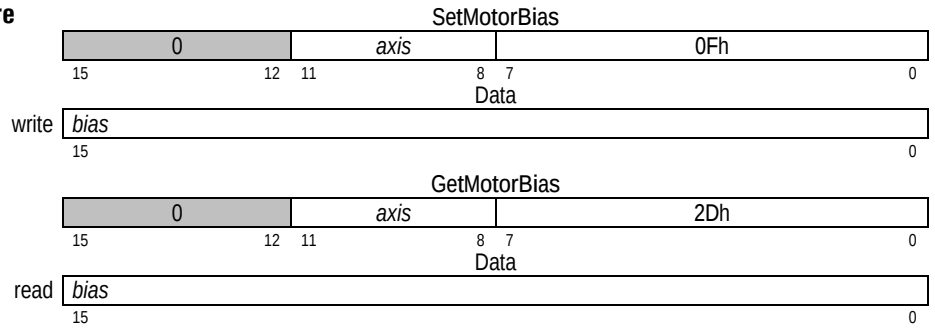
SetMotorBias *axis bias*
GetMotorBias *axis*

Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3

	Type	*Range*	*Scaling*	*Units*
bias	signed 16 bits	-2^{15} *to* $2^{15}-1$	$100/2^{15}$	% output

Packet structure



Description

SetMotorBias sets the output bias of the digital servo filter for the specified axis.

GetMotorBias reads the value of the bias of the digital servo filter.

Scaling example:

If it is desired that a motor bias value of -2.5 % of full scale be placed on the servo filter output, then this register should be loaded with a value of -2.5*32,768/100 = -819 (decimal). This corresponds to a loaded hexadecimal value of 0FCCDh.

Restrictions

see Set/GetMotorCommand, Set/GetMotorLimit

SetMotorCommand buffered 77h
GetMotorCommand 69h

Motor types

DC Brush		Brushless DC		Microstepping		
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	

Syntax

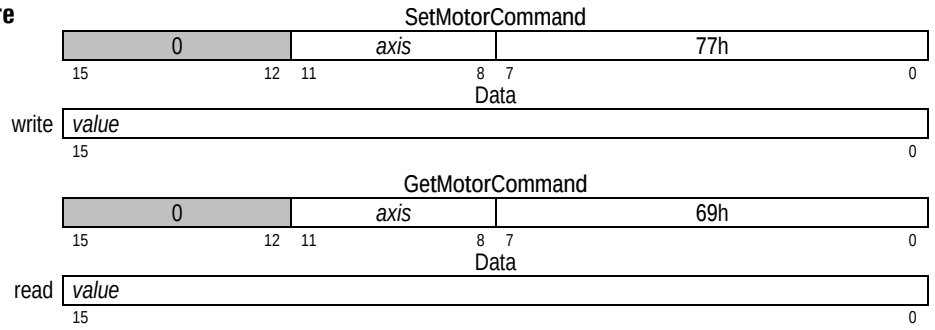
SetMotorCommand *axis value*
GetMotorCommand *axis*

Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3

	Type	*Range*	*Scaling*	*Units*
value	signed 16 bits	-2^{15} *to* $2^{15}-1$	$100/2^{15}$	% output

Packet structure



Description

SetMotorCommand loads the motor-command buffer register of the specified *axis*. For axes configured for microstepping motors, this command is used to control the magnitude of the output waveform. For DC brush and brushless DC motors, this command directly sets the servo output register when the motor mode has been set to Off (SetMotorMode). For microstepping motors, this command sets the amplitude of the sinusoidal output waveform.

GetMotorCommand reads the contents of the motor-command buffer register.

Scaling example: If it is desired that a motor command value of 13.7% of full scale be output to the motor, then this register should be loaded with a value of 13.7 * 32,768/100 = 4,489 (decimal). This corresponds to a hexadecimal value of 1189h.

Restrictions

SetMotorCommand is a buffered command. The value set using this command will not take effect until the next Update or MultiUpdate instruction.

see

Set/GetMotorBias, Set/GetMotorLimit, Set/GetMotorMode, MultiUpdate, Update

SetMotorLimit 06h
GetMotorLimit 07h

Motor types

DC Brush		Brushless DC			
MC2100	MC2800	MC2300	MC2800		

Syntax

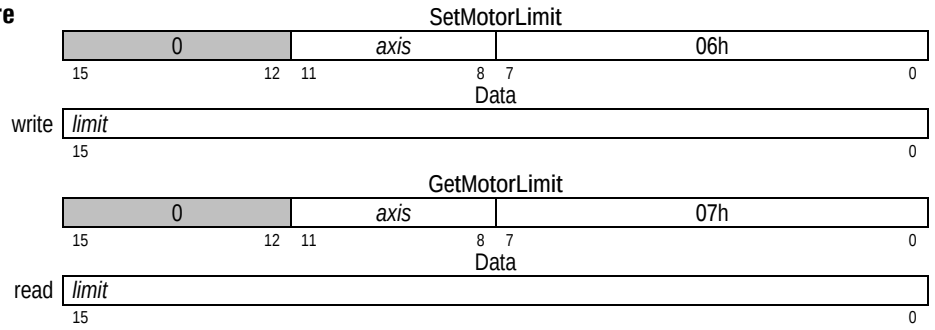
SetMotorLimit *axis limit*
GetMotorLimit *axis limit*

Arguments

Name	***Instance***	***Encoding***
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3

	Type	***Range***	***Scaling***	***Units***
limit	unsigned 16 bits	0 *to* 2^{15}-1	$100/2^{15}$	% output

Packet structure



Description

SetMotorLimit sets the maximum value for the motor output command allowed by the digital servo filter of the specified *axis*. Motor command values beyond this value will be clipped to the specified motor command limit. For example, if the motor limit was set to 1,000, and the servo filter determined that the current motor output value should be 1,100, then the actual output value would be 1,000. Conversely, if the output value were -1,100, then it would be clipped to -1,000. This command is useful for protecting amplifiers, motors, or system mechanisms when it is known that a motor command which exceeds a certain value will cause damage.

GetMotorLimit reads the current motor limit value.

Scaling example:

If it is desired that a motor limit of 75 % of full scale be established, then this register should be loaded with a value of 75.0 *32,767/100 = 24,576 (decimal). This corresponds to a hexadecimal value of 06000h.

Restrictions

This command only effects the motor output when the axis motor mode is on (SetMotorMode). When the chipset is in open loop mode, this command has no effect.

see

Set/GetMotorBias, Set/GetMotorCommand, Set/GetMotorMode

SetMotorMode DCh
GetMotorMode DDh

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

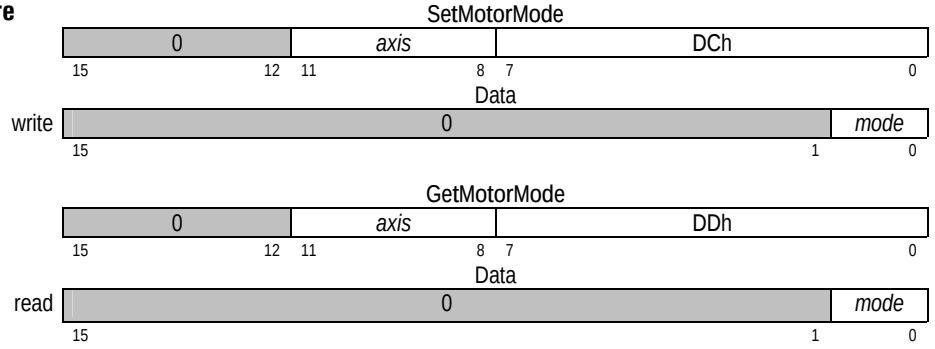
Syntax

SetMotorMode *axis mode*
GetMotorMode *axis*

Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3
mode	Off	0
	On	1

Packet structure



Description

SetMotorMode determines the mode of motor operation. When set to On, several events take place. For DC brush and brushless DC axes, the axis is placed in *closed-loop* mode, and is controlled by the output of the servo filter. For microstepping and pulse and direction motors, the trajectory generator controls the motor output. For all motor types, when the encoder source (Set/GetEncoderSource) is set to incremental or parallel, the position error is cleared; equivalent to a ClearPositionError command. When the motor mode is set to Off, the axis is in *open-loop* mode, and is controlled by commands placed directly into the motor output register by the host. Setting the motor mode to Off also resets the trajectory generator; bringing any active motion to an abrupt stop. In addition, the maximum velocity (Set/GetVelocity) is set to zero. For microstepping and pulse and direction motors, the step generator is switched off when the motor mode is set to Off.

GetMotorMode returns the value of the motor mode.

The following table shows the motor output source for each motor type and mode.

Motor type	Motor mode	Motor output source
DC brush; Brushless DC	Off	motor command register
	On	servo filter
Microstepping Pulse & direction	Off	N/A
	On	trajectory generator

Restrictions

see GetActivityStatus, Set/GetMotorCommand

SetNumberPhases 85h
GetNumberPhases 86h

Motor types

	Brushless DC		Microstepping		
	MC2300	MC2800	MC2400	MC2800	

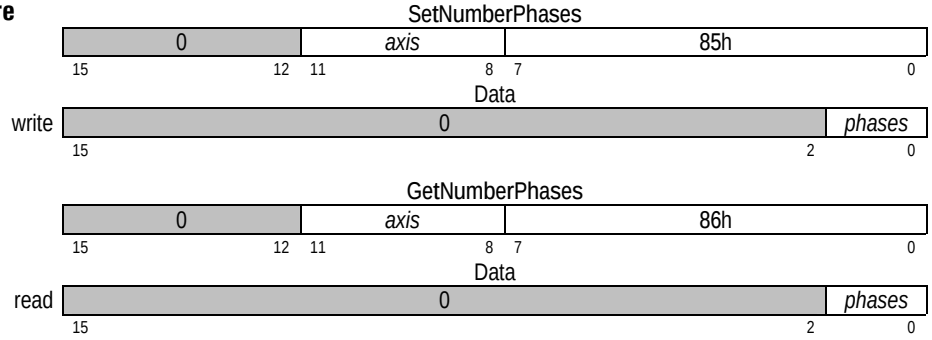
Syntax

SetNumberPhases *axis phases*
GetNumberPhases *axis*

Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3
phases	1Phase	1
	2Phases	2
	3Phases	3

Packet structure



Description

SetNumberPhases establishes the number of phases (one, two, or three) for commutation of the specified *axis*.

GetNumberPhases returns the number of phases set for the *axis*.

Restrictions

In PWM Sign/Magnitude output mode, the number of phases can be set to one or two.

In PWM 50/50 output mode, the number of phases can be set to one, two, or three.

For MC2300 & MC2400, the number of phases cannot be set to one (an "invalid parameter" error occurs).

see

GetPhaseCommand

SetOutputMode E0h
GetOutputMode 6Eh

Motor types

DC Brush		Brushless DC		Microstepping		
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	

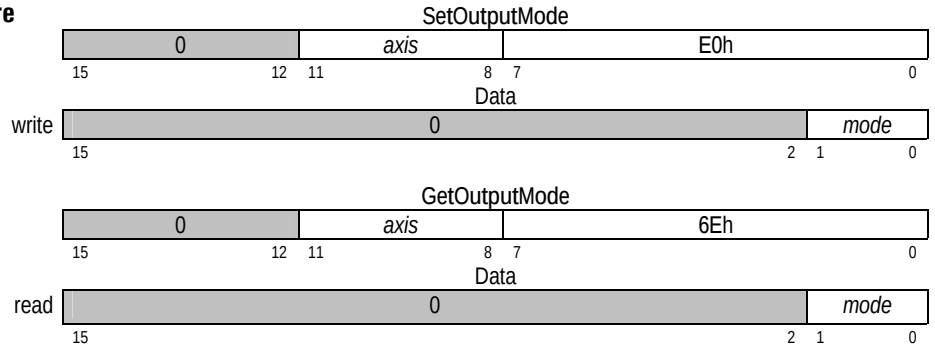
Syntax

SetOutputMode *axis mode*
GetOutputMode *axis*

Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3
mode	BipolarDAC	0
	PWMSignMagnitude	1
	PWM5050Magnitude	2

Packet structure



Description

SetOutputMode sets the form of the motor output signal of the specified *axis*.

GetOutputMode returns the value for the motor output mode.

Restrictions

If the number of phases (SetNumberPhases) is set to three, PWM Sign/Magnitude output mode is not available.

see

SetPhaseAngle 84h
GetPhaseAngle 2Ch

Motor types

	Brushless DC		Microstepping		
	MC2300	MC2800	MC2400	MC2400	

Syntax

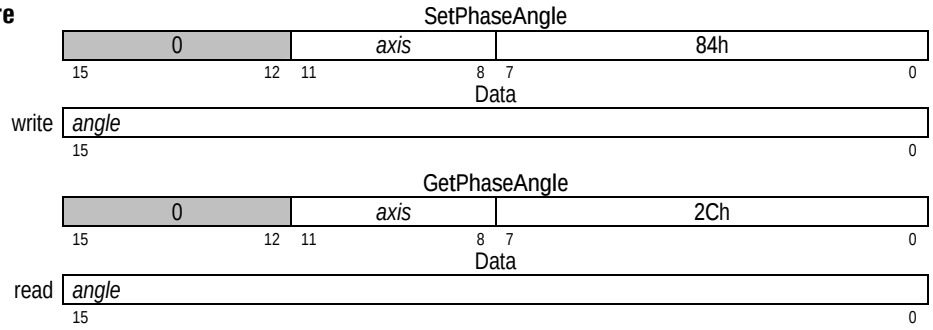
SetPhaseAngle *axis angle*
GetPhaseAngle *axis*

Arguments

Name	***Instance***	***encoding***
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3

	Type	***Range***	***Scaling***	***Units***
angle	unsigned 16 bits	0 *to* 2^{15}-1	unity	counts microsteps

Packet structure



Description

SetPhaseAngle sets the instantaneous commutation angle for the specified *axis*. For brushless DC motors, the phase angle is specified in units of encoder counts. For microstepping motors, it is specified in units of microsteps.

GetPhaseAngle returns the value of the phase angle. To convert counts to an actual phase angle, divide by the number of encoder counts per electrical cycle and multiply by 360. For example, if a value of 500 is retrieved using GetPhaseAngle, and the counts per electrical cycle value has been set to 2,000 (SetPhaseCounts command), then this corresponds to an angle of (500/2,000)*360 = 90 degrees current phase angle position. SetPhaseAngle resets the phase offset previously set by SetPhaseOffset. SetPhaseAngle resets the phase offset previously set by SetPhaseOffset.

Restrictions

The specified angle must not exceed the number set by the SetPhaseCounts command.

see

GetPhaseCommand, Set/GetPhaseCounts

SetPhaseCorrectionMode E8h
GetPhaseCorrectionMode E9h

Motor types

	Brushless DC			
	MC2300	MC2800		

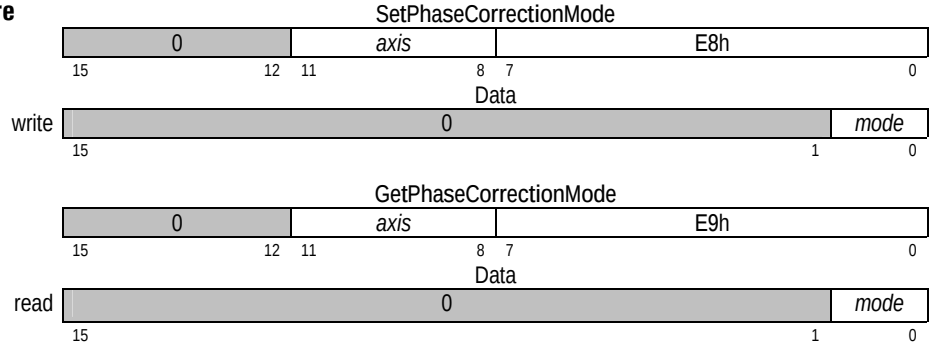
Syntax

SetPhaseCorrectionMode *axis mode*
GetPhaseCorrectionMode *axis*

Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3
mode	Disabled	0
	Enabled	1

Packet structure



Description

SetPhaseCorrectionMode sets the phase correction mode for the specified *axis* to either 0 (disabled) or 1(enabled). When phase correction is enabled, the encoder index signal is used to update the commutation phase angle once per motor revolution. This ensures that the commutation angle will remain correct, even if some encoder counts are lost due to electrical noise, or due to the number of encoder counts per electrical phase not being an integer.

GetPhaseCorrectionMode returns the phase correction mode.

Restrictions

see

GetPhaseCommand, Set/GetPhaseCounts

SetPhaseCounts 75h
GetPhaseCounts 7Dh

Motor types

	Brushless DC		Microstepping		
	MC2300	MC2800	MC2400	MC2800	

Syntax

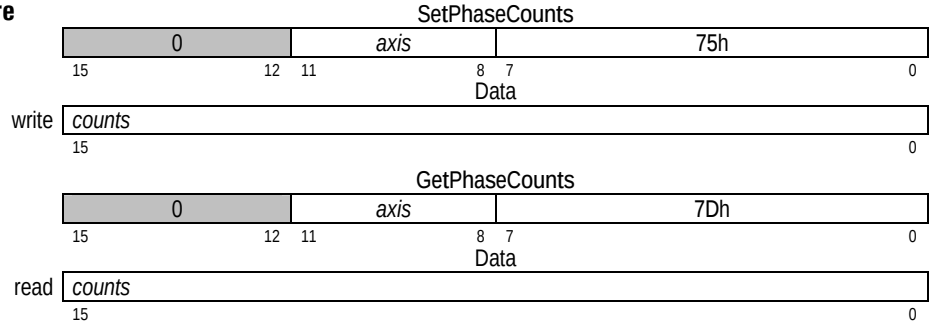
SetPhaseCounts *axis counts*
GetPhaseCounts *axis*

Arguments

Name	***Instance***	***encoding***
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3

	Type	***Range***	***Scaling***	***Units***
counts	unsigned 16 bits	1 *to* 2^{15}-1	unity	counts microsteps

Packet structure



Description

For axes configured for brushless DC motor types, SetPhaseCounts sets the number of encoder counts per electrical cycle of the motor. The number of electrical cycles is equal to 1/2 the number of motor poles. If this value is not an integer, then the closest integer value should be used; and phase correction mode should be enabled. See SetPhaseCorrectionMode.

For axes configured for microstepping motor types, the number of microsteps per full step is set using the command SetPhaseCounts. The parameter used for this command represents the number of microsteps per electrical cycle (4 times the desired number of microsteps). For example, to set 64 microsteps per full step, the command SetPhaseCounts 256 should be used. The maximum number of microsteps that can be generated per full step is 256, giving a maximum parameter value of 1024.

GetPhaseCounts returns the number of counts or microsteps per electrical cycle.

Restrictions

see Set/GetPhaseAngle

SetPhaseInitializeMode E4h
GetPhaseInitializeMode E5h

Motor types

	Brushless DC		Microstepping		
	MC2300	MC2800		MC2800	

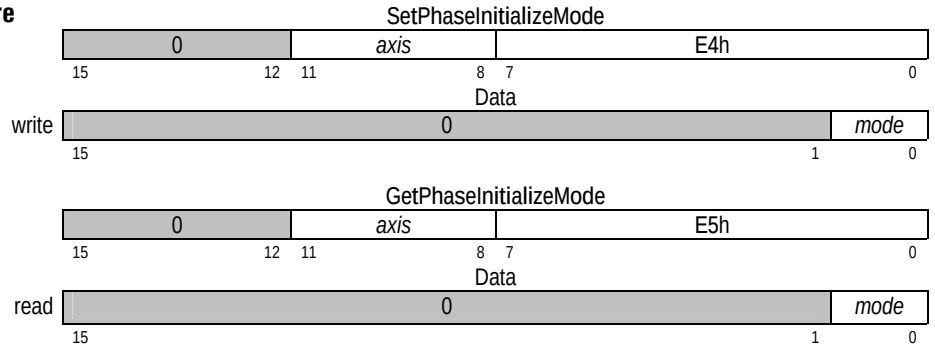
Syntax

SetPhaseInitializeMode *axis position*
GetPhaseInitializeMode *axis*

Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3
mode	Algorithmic	0
	Hall-based	1

Packet structure



Description

SetPhaseInitializeMode establishes the mode in which the specified axis is to be initialized for commutation. The options are Algorithmic and Hall-based. In algorithmic mode, the chipset briefly stimulates the motor windings and sets the initial phasing based on the observed motor response. In Hall-based initialization mode, the three Hall sensor signals are used to determine the motor phasing.

GetPhaseInitializeMode returns the value of the initialization mode.

Restrictions

Algorithmic mode should only be selected if it is known that the axis is free to move in both directions, and that a brief uncontrolled move can be tolerated by the motor, mechanism, and load.

see

Set/GetPhaseInitializeTime, InitializePhase, Set/GetNumberPhases

SetPhaseInitializeTime 72h
GetPhaseInitializeTime 7Ch

Motor types

	Brushless DC		Microstepping		
	MC2300	MC2800		MC2800	

Syntax

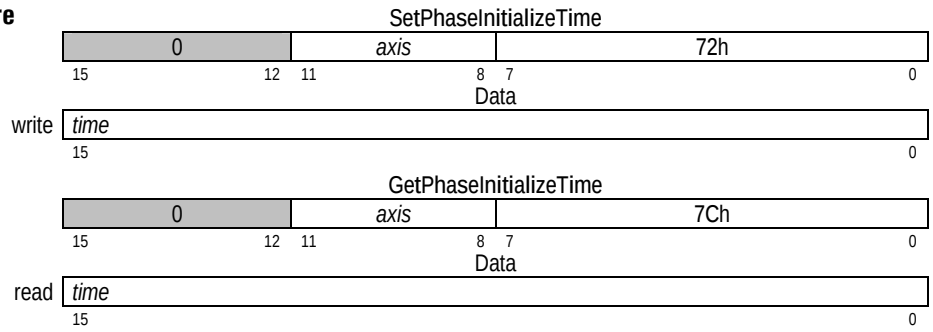
SetPhaseInitializeTime *axis time*
GetPhaseInitializeTime *axis*

Arguments

Name	***Instance***	***encoding***
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3

	Type	***Range***	***Scaling***	***Units***
time	unsigned 16 bits	0 *to* 2^{15}-1	unity	cycles

Packet structure



Description

SetPhaseInitializeTime sets the time value (in chip cycles) to be used during the algorithmic phase initialization procedure. This value determines the duration of each of the four segments in the phase initialization algorithm.

GetPhaseInitializeTime returns the value of the phase initialization time.

Restrictions

see Set/GetPhaseInitializeMode, InitializePhase, Set/GetNumberPhases

SetPhaseOffset 76h
GetPhaseOffset 7Bh

Motor types

	Brushless DC		Microstepping		
	MC2300	MC2800		MC2800	

Syntax

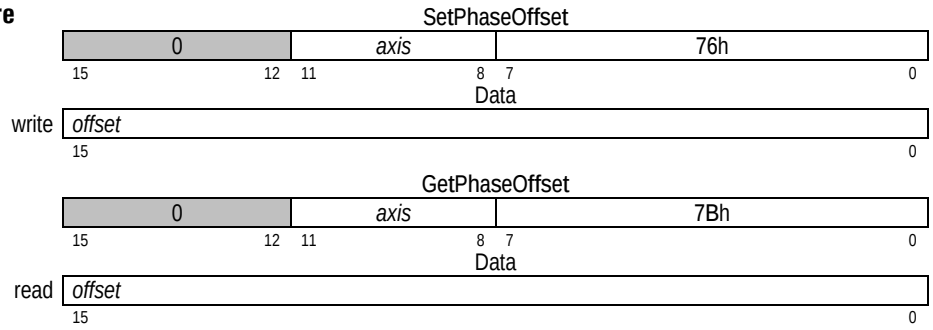
SetPhaseOffset *axis offset*
GetPhaseOffset *axis*

Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3

	Type	*Range*	*Scaling*	*Units*
offset	unsigned 16 bits	-1 *to* 2^{15}-1	unity	counts

Packet structure



Description

SetPhaseOffset sets the offset from the index mark of the specified axis to the maximum output value of phase A. This command will have no immediate effect on the commutation angle, but will have an effect once the index pulse is encountered. The settable range of phase offset is 0 to 32,767.

GetPhaseOffset returns the value of the phase offset.

To convert counts to a phase angle in degrees, divide by the number of encoder counts per electrical cycles, and multiply by 360. For example, if a value of 500 is specified using SetPhaseOffset and the counts per electrical cycle value has been set to 2,000 (SetPhaseCounts command), then this corresponds to a phase angle of (500/2,000)*360 = 90 degrees at the index mark.

Restrictions

Before the first index capture has occurred, GetPhaseOffset will return -1.

see

SetPhasePrescale E6h
GetPhasePrescale E7h

Motor types

	Brushless DC		Microstepping		
	MC2300	MC2800		MC2800	

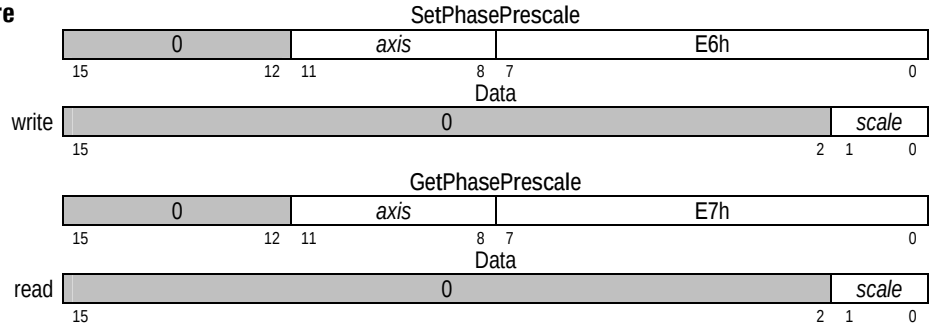
Syntax

SetPhasePrescale *axis scale*
GetPhasePrescale *axis*

Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3
mode	Off	0
	1/64	1
	1/128	2
	1/256	3

Packet structure



Description

SetPhasePrescale controls scaling of the encoder counts before they are used to calculate a commutation angle for the specified *axis*. When operated in the pre-scale mode, the motion processor can commutate motors with a high number of counts per electrical cycle; such as motors with very high accuracy encoders.

SetPhasePrescale Off removes the scale factor.

GetPhasePrescale returns the value of the scaling mode.

Restrictions

see

SetPosition buffered 10h
GetPosition 4Ah

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

Syntax

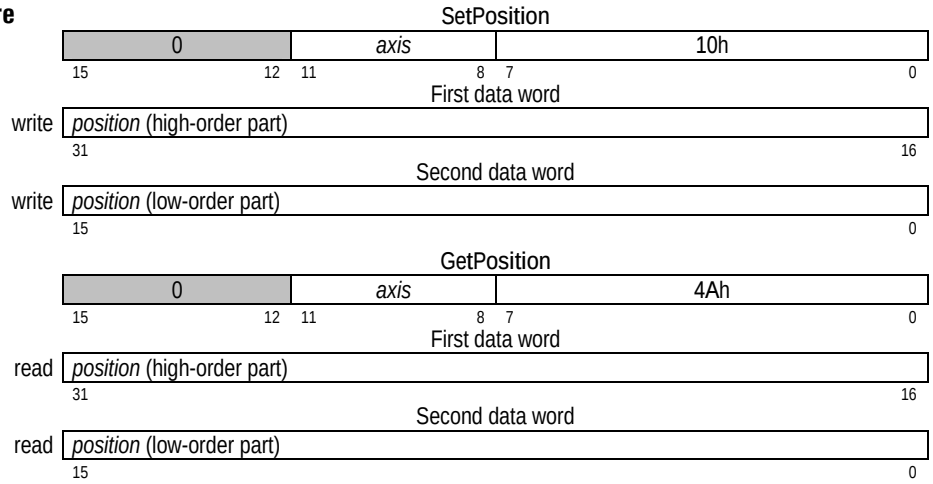
SetPosition *axis position*
GetPosition *axis*

Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3

	Type	*Range*	*Scaling*	*Units*
position	signed 32 bits	-2^{31} *to* $2^{31}-1$	unity	counts steps

Packet structure



Description

SetPosition specifies the trajectory destination of the specified *axis*. It is used in the trapezoidal and S-curve profile modes.

GetPosition reads the contents of the buffered position register.

Restrictions

SetPosition is a buffered command. The value set using this command will not take effect until the next Update or MultiUpdate instruction.

see

Set/GetAcceleration, Set/GetDeceleration, Set/GetJerk, Set/GetVelocity, MultiUpdate, Update

SetPositionErrorLimit 97h
GetPositionErrorLimit 98h

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

Syntax

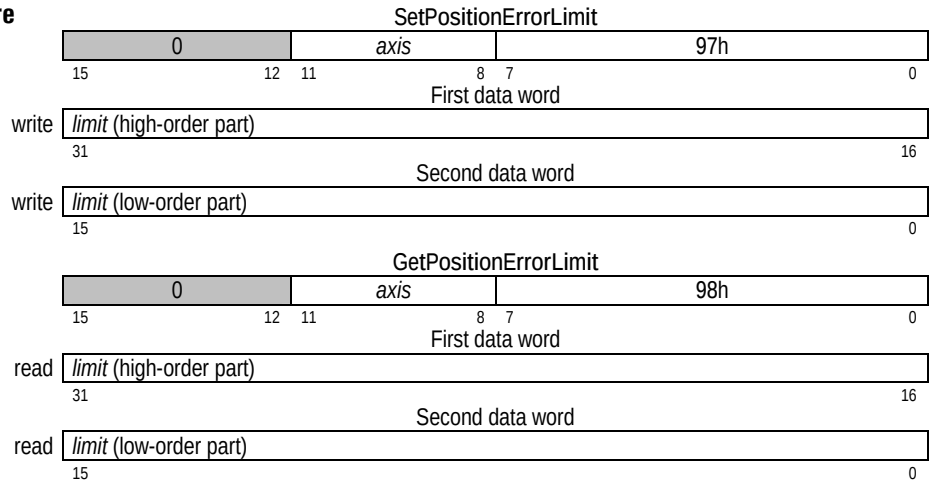
SetPositionErrorLimit *axis limit*
GetPositionErrorLimit *axis*

Arguments

Name	***Instance***	***Encoding***
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3

	Type	***Range***	***Scaling***	***Units***
limit	unsigned 32 bits	0 *to* 2^{31}-1	unity	counts

Packet structure



Description

SetPositionErrorLimit sets the absolute value of the maximum position error allowable by the motion processor for the specified *axis*. If the position error exceeds this limit, the motion error bit in the event status register is set. This occurrence may or may not cause the axis to stop moving, depending on the value set using the SetAutoStopMode command.

GetPositionErrorLimit returns the value of the position error limit.

Restrictions

see

GetPositionError, GetActualPosition

SetProfileMode buffered A0h
GetProfileMode A1h

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

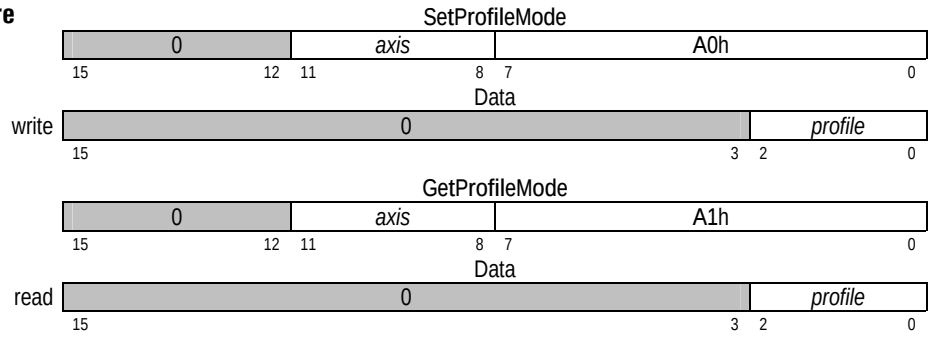
Syntax

SetProfileMode *axis profile*
GetProfileMode *axis*

Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3
profile	Trapezoidal	0
	Velocity contouring	1
	S-curve	2
	Electronic gear	3

Packet structure



Description

SetProfileMode sets the profile mode for the specified *axis*.

GetProfileMode returns the contents of the buffered profile-mode register for the specified *axis*.

Restrictions

SetProfileMode is a buffered command. The value set using this command will not take effect until the next Update or MultiUpdate instruction.

see

MultiUpdate, Update

SetSampleTime 38h
GetSampleTime 61h

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

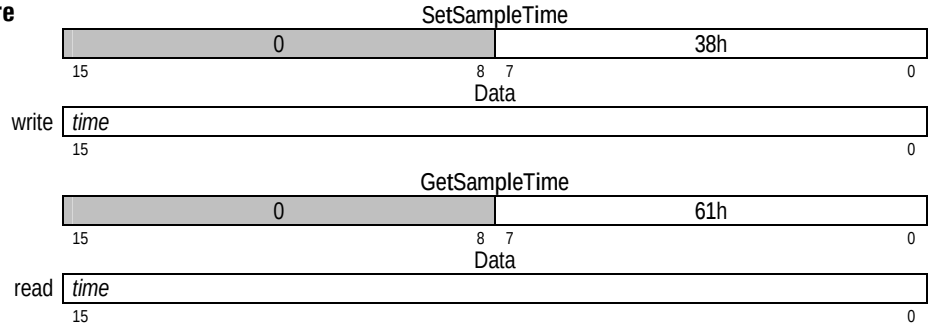
Syntax

SetSampleTime *time*
GetSampleTime

Arguments

Name	*Type*	*Range*	*Units*
time	unsigned 16 bits	1 *to* 2^{15}-1	µsec/cycle

Packet structure



Description

SetSampleTime sets the cycle time for the chipset. This is the time between servo loop updates and trajectory calculations. The value is expressed in microseconds. Only certain values are allowed, as shown in the following table.

Product	Allowed values
MC2100 series	multiples of 51.2 and at least 102 µsec per enabled axis
MC2300 series	multiples of 51.2 and at least 154 µsec per enabled axis
MC2400 series	multiples of 51.2 and at least 154 µsec per enabled axis
MC2500 series	multiples of 51.2 and at least 154 µsec per enabled axis
MC2800 series	multiples of 51.2 and at least 154 µsec per enabled axis

GetSampleTime returns the current sample time value.

Result of invalid sample time arguments:

The PMD device does not return an error when an invalid sample time is attempted. If the value is less than the required minimum (based on the number of enabled axes), the sample time will be set to the minimum value. If the value is not an increment of 51.2 µsec, then the sample time will be set to the closest valid increment to that value.

Restrictions

This command effects the cycle time for all axes. This command cannot be used to set a sample time lower than the required minimum cycle time for the current configuration. Attempting to do so will set the sample time to the required minimum cycle time, as specified in the preceeding table.

see

SetSerialPortMode 8Bh
GetSerialPortMode 8Ch

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

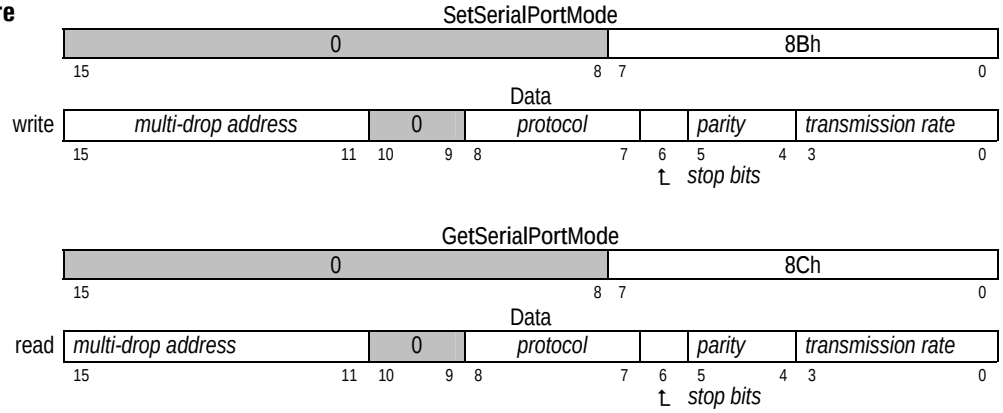
Syntax

SetSerialPortMode *mask*
GetSerialPortMode

Arguments

Name	*Type*	*Encoding*
mask	unsigned 16 bits	*see table on following page*

Packet structure



Description

SetSerialPortMode sets the configuration for the asynchronous serial port.

Note: It is recommended that two stop bits be used for baud rates greater than 19200bps.

GetSerialPortMode returns the configuration for the asynchronous serial port.

The encoding of the data used by this command is shown in the following table.

Bit Number	Name	Instance	Encoding
0-3	transmission rate	1200 baud 2400 9600 19200 57600 115200 250000 416667	0 1 2 3 4 5 6 7
4-5	parity	none odd even	0 1 2
6	stop bits	1 2	0 1
7-8	protocol	Point-to-point Multi-drop using address bit Multi-drop using idle-line detection	0 2 3
11-15	multi-drop address	Address 0 Address 1 … Address 31	0 1 … 31

Restrictions

see Set/GetDiagnosticPortMode

SetSettleTime AAh
GetSettleTime ABh

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

Syntax

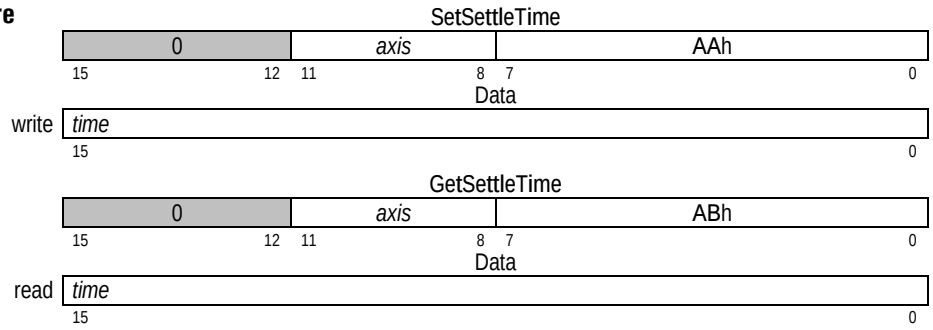
SetSettleTime *axis time*
GetSettleTime *axis*

Arguments

Name	***Instance***	***Encoding***
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3

	Type	***Range***	***Scaling***	***Units***
time	unsigned 16 bits	0 *to* 2^{15}-1	unity	cycles

Packet structure



Description

SetSettleTime sets the time, in number of cycles, that the specified *axis* must remain within the settle window before the axis-settled indicator in the activity status register is set.

GetSettleTime returns the value of the settle time for the specified *axis*.

Restrictions

see Set/GetMotionCompleteMode, Set/GetSettleWindow, GetActivityStatus

SetSettleWindow BCh
GetSettleWindow BDh

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

Syntax

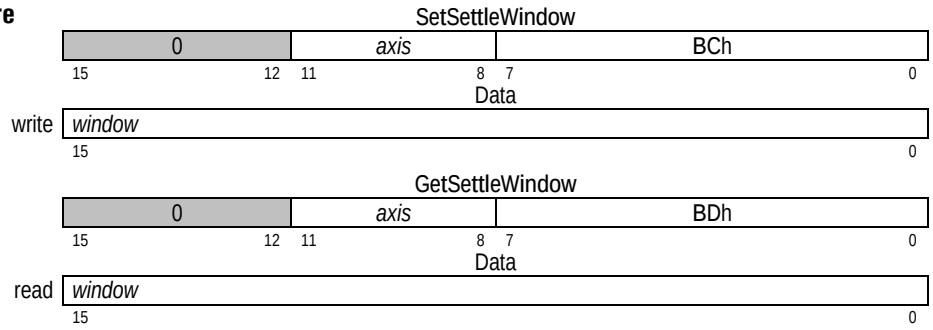
SetSettleWindow *axis window*
GetSettleWindow *axis*

Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3

	Type	*Range*	*Scaling*	*Units*
window	unsigned 16 bits	0 *to* 2^{16}-1	unity	counts

Packet structure



Description

SetSettleWindow sets the position range within which the specified *axis* must remain for the duration specified by SetSettleTime before the axis-settled indicator in the activity status register is set.

GetSettleWindow returns the value of the settle window.

Restrictions

see

Set/GetMotionCompleteMode, Set/GetSettleTime, GetActivityStatus

SetSignalSense A2h
GetSignalSense A3h

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

Syntax

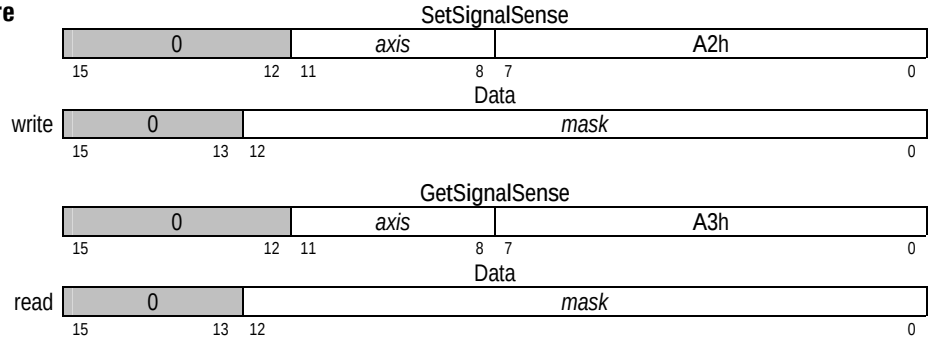
SetSignalSense *axis mask*
GetSignalSense *axis*

Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3

	Indicator		*Bit Number*
mask	EncoderA	0001h	0
	EncoderB	0002h	1
	EncoderIndex	0004h	2
	EncoderHome	0008h	3
	PositiveLimit	0010h	4
	NegativeLimit	0020h	5
	AxisIn	0040h	6
	HallA	0080h	7
	HallB	0100h	8
	HallC	0200h	9
	AxisOut	0400h	10
	StepOutput	0800h	11
	MotorDirection	1000h	12
	reserved		13 - 15

Packet structure



Description

SetSignalSense establishes the sense of the corresponding bits of the signal status register, with the addition of StepOutput and MotorDirection, for the specified *axis*. For all input signals, the input is inverted if the corresponding sense bit is one; otherwise it is not inverted.

For encoder index/home: if the sense bit is 1, a capture will occur on a low-to-high signal transition. Otherwise, a capture will occur on a high-to-low transition.

For positive and negative limit: if the sense bit is 1, an overtravel condition will occur if the signal is high. Otherwise, an overtravel condition will occur when the signal is low.

The AxisOut signal is inverted if the sense bit is set to 1; otherwise it is not inverted.

When the StepOutput bit is set to 1, a step will be generated by the motion processor with a low-to-high transition on the Pulse signal. Otherwise, a step will be generated by the motion processor with a high-to-low transition on the Pulse signal.

Setting the MotorDirection bit has the effect of swapping the sense of positive and negative motor movement.

GetSignalSense returns the value of the signal sense mask.

Restrictions

Inverting the encoder A or B signal may prevent the index capture mechanism from operating correctly.

see

GetSignalStatus

SetStartVelocity 6Ah
GetStartVelocity 6Bh

Motor types

		Microstepping		Pulse & Direction
		MC2400		MC2500

Syntax

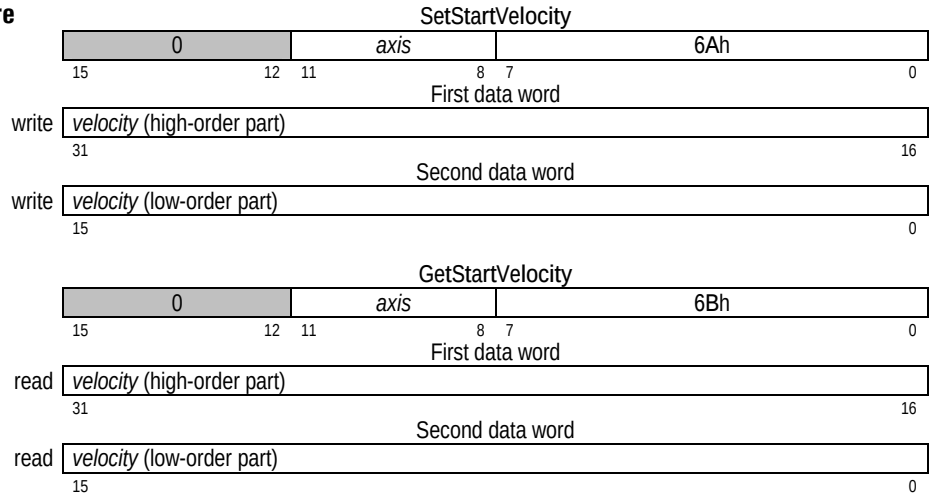
SetStartVelocity *axis velocity*
GetStartVelocity *axis*

Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3

	Type	*Range*	*Scaling*	*Units*
velocity	unsigned 32 bits	0 *to* 2^{31}-1	$1/2^{16}$	counts/cycle steps/cycle

Packet structure



Description

SetStartVelocity loads the starting velocity register for the specified *axis*. The start velocity is the instantaneous velocity at the start and at the end of the profile.

GetStartVelocity reads the starting velocity buffer register.

Scaling example: To load a starting velocity value of 1.750 counts/cycle, multiply by 65,536 (giving 114,688), and load the resulting number as a 32-bit number; giving 0001 in the high word, and C000h in the low word. Values returned by GetStartVelocity must be divided by 65,536 to convert them to units of counts/cycle.

Restrictions

StartVelocity is only used in the velocity contouring and trapezoidal profile modes.

SetVelocity is a buffered command. The value set using this command will not take effect until the next Update or MultiUpdate instruction.

see

Set/GetAcceleration, Set/GetDeceleration, Set/GetPosition

SetStepRange CFh
GetStepRange CEh

Motor types

			Pulse & Direction
			MC2500

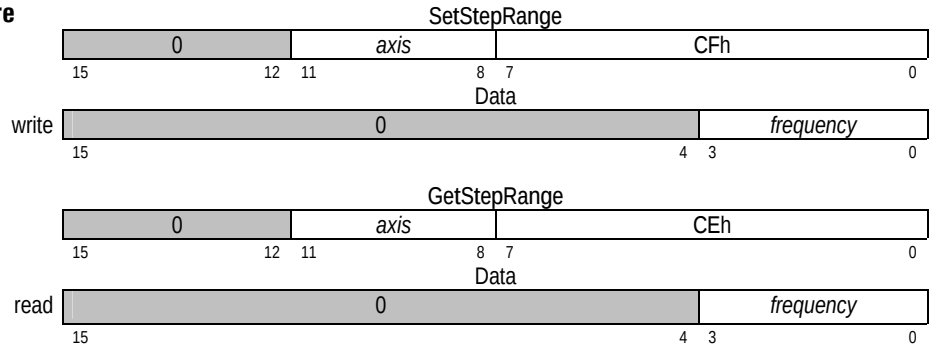
Syntax

SetStepRange *axis frequency*
GetStepRange *axis*

Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3
frequency	5 MHz	1
	625 kHz	4
	156.25 kHz	6
	39.062 kHz	8

Packet structure



Description

SetStepRange sets the maximum pulse rate frequency for the specified *axis*. For example, if the desired maximum is 200,000 pulses/second, the command SetStepRange 4 should be issued.

GetMaxStepRange returns the maximum pulse rate frequency for the specified *axis*.

Restrictions

The SetStepRange command should only be used once per axis after a chip reset.

see

SetStopMode buffered D0h
GetStopMode D1h

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

Syntax

SetStopMode *axis mode*
GetStopMode *axis*

Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3
mode	Disabled	0
	AbruptStop	1
	SmoothStop	2

Packet structure

Description

SetStopMode stops the specified *axis*. The available stop modes are AbruptStop, which instantly (without any deceleration phase) stops the *axis*, and SmoothStop which uses the programmed deceleration value and profile shape for the current profile mode to stop the axis. Disabled is generally used to turn off a previously issued set stop command.

Note: After an Update, a buffered stop command (SetStopMode command) will reset to the Disabled condition. In other words, if the command SetStopMode is followed by an Update command and then by a GetStopMode command, the retrieved stop mode will be Disabled.

GetStopMode returns the value of the stop mode.

Restrictions

SmoothStop mode is not available in the electronic-gearing profile.

SetStopMode is a buffered command. The value set using this command will not take effect until the next Update or MultiUpdate instruction.

see

MultiUpdate, Update

SetSynchronizationMode F2h
GetSynchronizationMode F3h

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2103	MC2803	MC2303	MC2500	MC2403	MC2803	MC2500

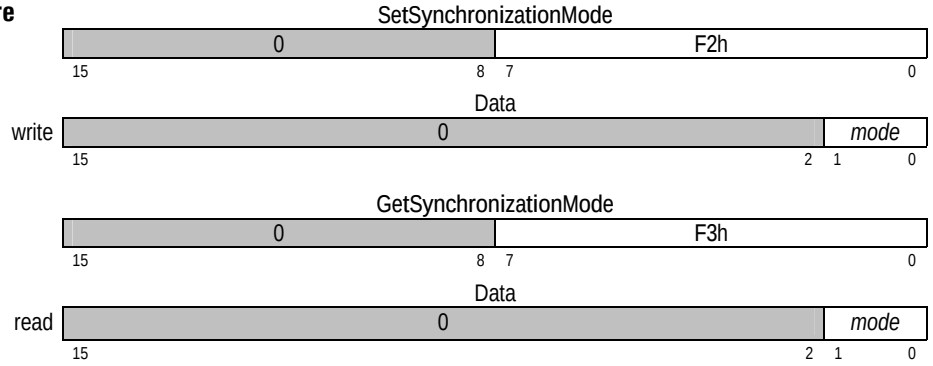
Syntax

SetSynchronizationMode *mode*
GetSynchronizationMode

Arguments

Name	*Instance*	*Encoding*
Mode	Disabled	0
	Master	1
	Slave	2

Packet structure



Description

SetSynchronizationMode sets the mode of the pin used for the synchronization of the internal timer across multiple motion processors. In the disabled mode, the pin is configured as an input and is not used. In the master mode, the pin outputs a synchronization pulse that can be used by slave nodes or other devices to synchronize with the internal chip cycle of the master node. In the slave mode, the pin is configured as an input and a pulse on the pin synchronizes the internal chip cycle.

GetSynchronizationMode returns the value of the synchronization mode.

Restrictions

If the motion processor is configured as a slave, and any axis is configured for pulse & direction output, multi-chip synchronization cannot be used.

see

SetTraceMode B0h
GetTraceMode B1h

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

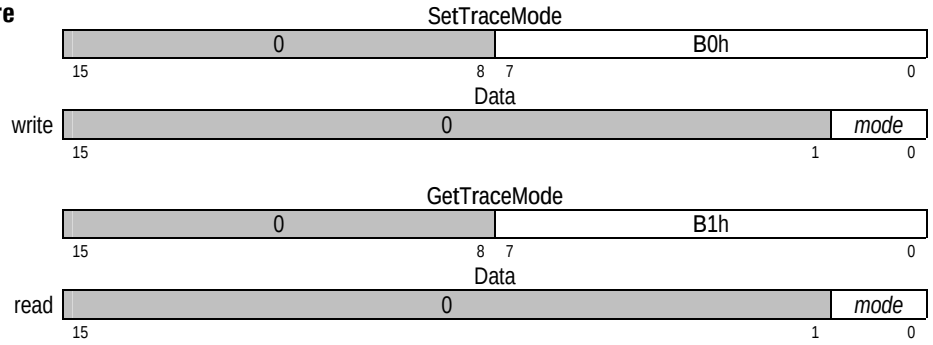
Syntax

SetTraceMode *mode*
GetTraceMode

Arguments

Name	*Instance*	*Encoding*
mode	OneTime	0
	RollingBuffer	1

Packet structure



Description

SetTraceMode sets the buffer usage for the next trace. In OneTime mode, the trace continues until the trace buffer is filled, then stops. In RollingBuffer mode, the trace continues from the beginning of the trace buffer after the end is reached. When in the rolling mode, values stored at the beginning of the trace buffer are lost if they are not read before being overwritten by the wrapped data.

GetTraceMode returns the value for the trace mode.

Restrictions

see GetTraceStatus

SetTracePeriod B8h
GetTracePeriod B9h

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

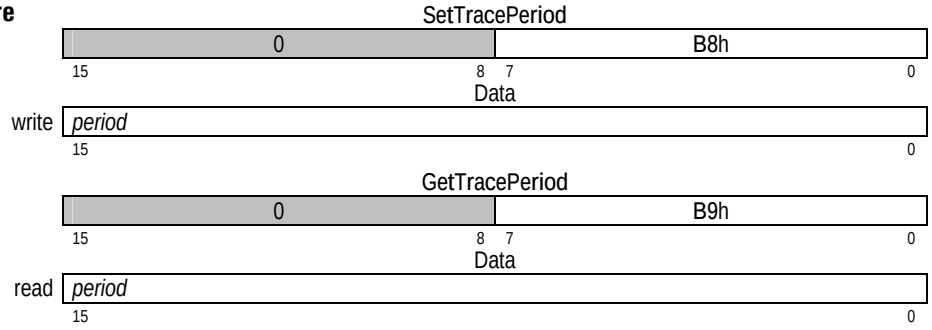
Syntax

SetTracePeriod *period*
GetTracePeriod

Arguments

Name	*Type*	*Range*	*Scaling*	*Units*
period	unsigned 16 bits	1 *to* 2^{16}-1	unity	cycles

Packet structure



Description

SetTracePeriod sets the interval between contiguous trace captures. For example, if the trace period is set to one, trace data will be captured at the end of every chip cycle. If the trace period is set to two, trace data will be captured at the end of every second chip cycle, and so on.

GetTracePeriod returns the value for the trace period.

Restrictions

see

Set/GetTraceStart, Set/GetTraceStop

SetTraceStart B2h
GetTraceStart B3h

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

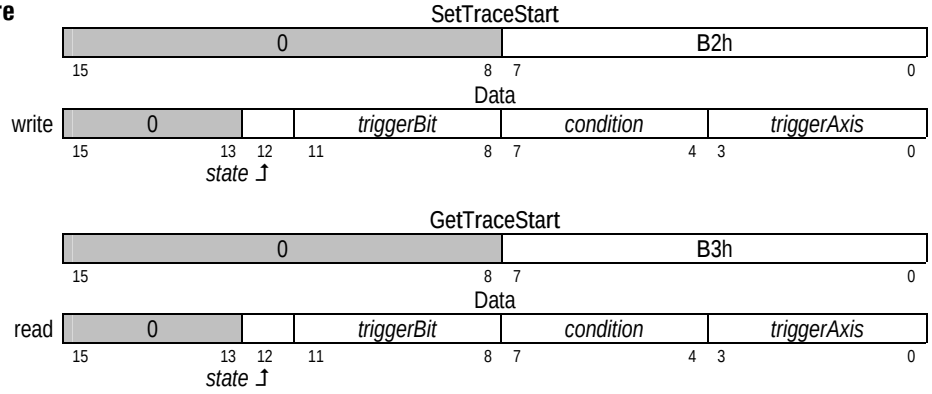
Syntax

SetTraceStart *triggerAxis condition triggerBit triggerState*
GetTraceStart

Arguments

Name	*Instance*	*Encoding*
triggerAxis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3
	Description	
condition	Immediate	0
	Next update	1
	Event Status	2
	Activity Status	3
	Signal Status	4
triggerBit	Status register bit	0 *to* 15
triggerState	Triggering state of the bit	0 (value = 0) 1 (value = 1)

Packet structure



Description

SetTraceStart sets the condition for starting the trace. The Immediate condition requires no axis to be specified and the trace will begin upon execution of this instruction. The other four conditions require an axis to be specified; and when the condition for that axis is attained, the trace will begin.

When a status register bit is the trigger, the bit number and state must be included in the argument. The trace is started when the indicated bit reaches the specified state (0 or 1).

GetTraceStart returns the value of the trace-start trigger.

Once a trace has started, the trace-start trigger is reset to zero.

The following table shows the corresponding value for combinations of *triggerBit* and *register*.

triggerBit	event status register	activity status register	signal status register
0	Motion Complete	Phasing Initialized	Encoder A
1	Wrap-around	At maximum velocity	Encoder B
2	Breakpoint 1	Tracking	Encoder index
3	Position capture		Home
4	Motion error		Positive limit
5	In positive limit		Negative limit
6	In negative limit		AxisIn
7	Instruction error	Axis settled	Hall sensor 1
8		Motor on/off	Hall sensor 2
9		Position capture	Hall sensor 3
0Ah		In motion	
0Bh	Commutation error	In positive limit	
0Ch		In negative limit	
0Dh			
0Eh	Breakpoint 2		
0Fh			

Description

Examples:

If it is desired that the trace begin on the next Update for axis 3, then a 2 is set for the axis number, a 1 is set for the condition, and bit number and state can be loaded with zeroes since they are not used. The actual data word sent to the motor processor in this case is 0012h.

If it is desired that the trace begin when bit 7 of the activity status register for axis 2 goes to 0, then the trace start is loaded as follows: a 1 is loaded for axis number, a 3 is loaded for condition, a 7 is loaded for bit number, and a 0 is loaded for state. The actual data word sent to the motor processor in this case is 0731h.

Restrictions

see GetTraceCount, Set/GetTraceMode, Set/GetTracePeriod, Set/GetTraceStop

SetTraceStop B4h
GetTraceStop B5h

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

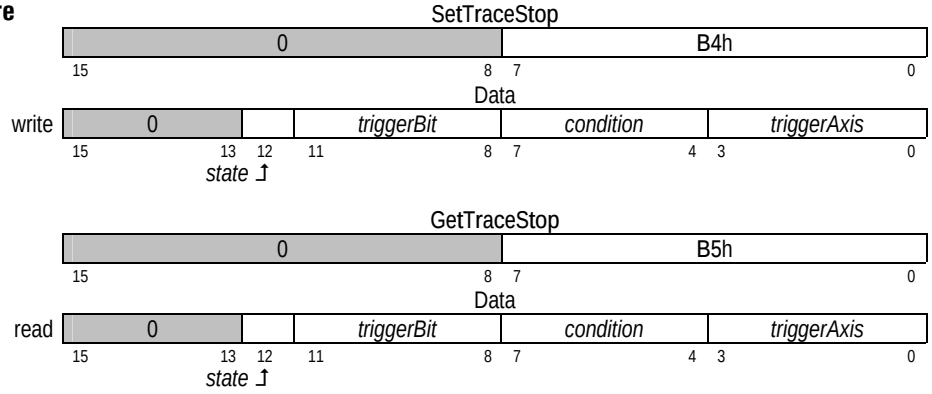
Syntax

SetTraceStop *triggerAxis condition triggerBit triggerState*
GetTraceStop

Arguments

Name	***Instance***	***Encoding***
triggerAxis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3
	Description	
condition	Immediate	0
	Next update	1
	Event Status	2
	Activity Status	3
	Signal Status	4
triggerBit	Status register bit	0 *to* 15
triggerState	Triggering state of the bit	0 (value = 0) 1 (value = 1)

Packet structure



Description

SetTraceStop sets the condition for stopping the trace. The Immediate condition requires no axis to be specified and the trace will stop upon execution of this instruction. The other four conditions require an axis to be specified; and when the condition for that axis is attained, the trace will stop.
When a status register bit is the trigger, the bit number and state must be included in the argument. The trace is stopped when the indicated bit reaches the specified state (0 or 1).

GetTraceStop returns the value of the trace-stop trigger.

Once a trace has stopped, the trace-stop trigger is reset to zero.

The following table shows the corresponding value for combinations of *triggerBit* and *register*.

triggerBit	event status register	activity status register	signal status register
0	Motion Complete	Phasing Initialized	Encoder A
1	Wrap-around	At maximum velocity	Encoder B
2	Breakpoint 1	Tracking	Encoder index
3	Position capture		Home
4	Motion error		Positive limit
5	In positive limit		Negative limit
6	In negative limit		AxisIn
7	Instruction error	Axis settled	Hall sensor 1
8		Motor on/off	Hall sensor 2
9		Position capture	Hall sensor 3
0Ah		In motion	
0Bh	Commutation error	In positive limit	
0Ch		In negative limit	
0Dh			
0Eh	Breakpoint 2		
0Fh			

Examples:

If it is desired that the trace ends on the next Update for axis 3, then a 2 is set for the axis number, a 1 is set for the condition, and bit number and state can be loaded with zeroes since they are not used. The actual data word sent to the motor processor in this case is 0012h.

If it is desired that the trace ends when bit 7 of the activity status register for axis 2 goes to 0, then the trace stop is loaded as follows: a 1 is loaded for axis number, a 3 is loaded for condition, a 7 is loaded for bit number, and a 0 is loaded for state. The actual data word sent to the motor processor in this case is 0731h.

Restrictions

see Set/GetTraceCount, SetGetTraceMode, Set/GetTraceStart, Set/GetTraceStatus

SetTraceVariable B6h
GetTraceVariable B7h

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

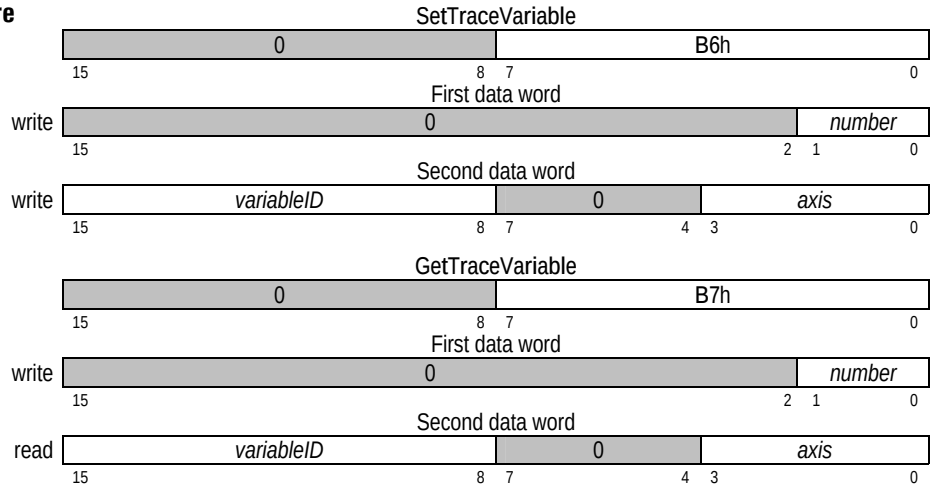
Syntax

SetTraceVariable *variableNumber traceAxis variableID*
GetTraceVariable *variableNumber*

Arguments

Name	***Instance***	***Encoding***
variableNumber	Variable1	0
	Variable2	1
	Variable3	2
	Variable4	3
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3
variableID	None (disable the variable)	0
	PositionError	1
	CommandedPosition	2
	CommandedVelocity	3
	CommandedAcceleration	4
	ActualPosition	5
	ActualVelocity	6
	MotorCommand	7
	ChipTime	8
	CaptureRegister	9
	Integral	Ah
	Derivative	Bh
	EventStatusRegister	Ch
	ActivityStatusRegister	Dh
	SignalStatusRegister	Eh
	PhaseAngle	Fh
	PhaseOffset	10h
	PhaseA	11h
	PhaseB	12h
	PhaseC	13h
	AnalogInput1	14h
	AnalogInput2	15h
	AnalogInput3	16h
	AnalogInput4	17h
	AnalogInput5	18h
	AnalogInput6	19h
	AnalogInput7	1Ah
	AnalogInput8	1Bh

Packet structure



Description

SetTraceVariable assigns the given variable to the specified *variableNumber* location in the trace buffer. Up to four variables may be traced at one time. All combinations of axis numbers and trace variables are supported.

All variable assignments must be contiguous starting with *variableNumber* = 0.

GetTraceVariable returns the variable and axis of the specified *variableNumber*.

Example: to set up a three variable trace capturing the commanded acceleration for axis 1, the actual position for axis 1, and the event status word for axis 3, the following sequence of commands would be used. First, a SetTraceVariable command with *variableNumber* of 0, *axis* of 0, and *variableID* of 4 would be sent. Then, a SetTraceVariable command with *variableNumber* of 1, *axis* of 0, and *variableID* of 5 would be sent. Finally, a SetTraceVariable command with a *variableNumber* of 3, *axis* of 2 and *variableID* of 0h would be sent.

Restrictions

see

SetTrackingWindow A8h
GetTrackingWindow A9h

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

Syntax

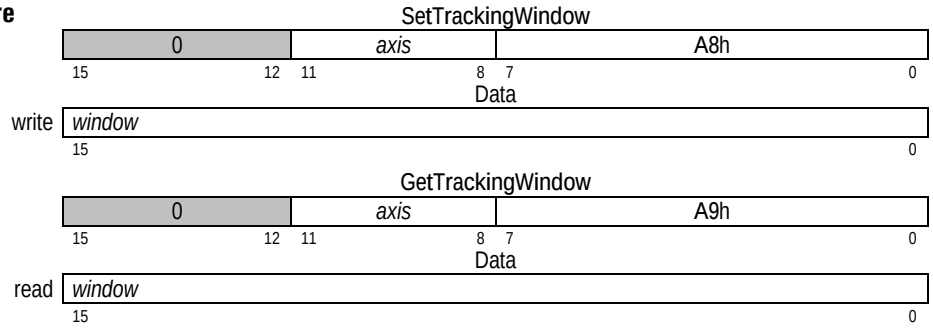
SetTrackingWindow *axis window*
GetTrackingWindow *axis*

Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3

	Type	*Range*	*Scaling*	*Units*
window	unsigned 16 bits	0 *to* 2^{16}-1	unity	counts

Packet structure



Description

SetTrackingWindow sets boundaries for the position error of the specified axis. If the absolute value of the position error exceeds the tracking window, the tracking indicator (bit 2 of the activity status register) is set to 0. When the position error returns to within the window, the tracking indicator is set to 1.

GetTrackingWindow returns the value of the tracking window.

Restrictions

see

GetActivityStatus, GetActualPosition

SetVelocity buffered 11h
GetVelocity 4Bh

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

Syntax

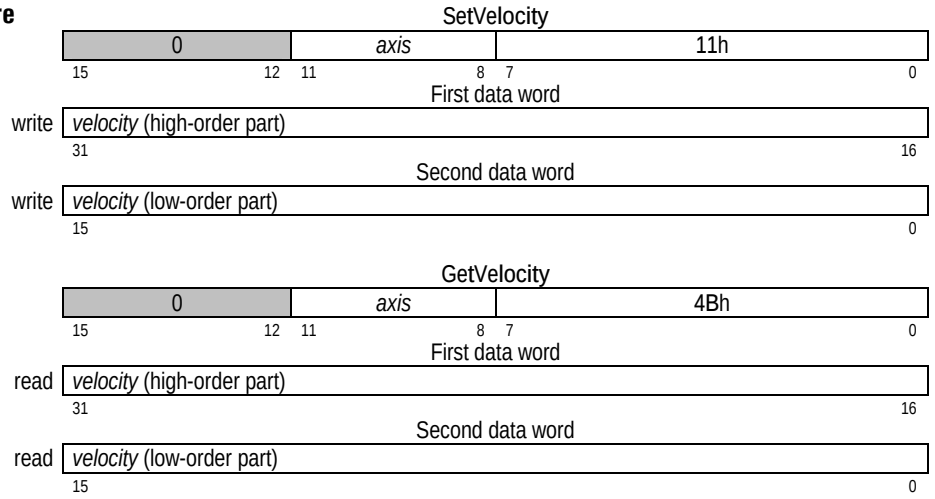
SetVelocity *axis velocity*
GetVelocity *axis*

Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3

	Type	*Range*	*Scaling*	*Units*
velocity	signed 32 bits	-2^{31} *to* $2^{31}-1$	$1/2^{16}$	counts/cycle steps/cycle

Packet structure



Description

SetVelocity loads the maximum velocity buffer register for the specified *axis*.

GetVelocity returns the contents of the maximum velocity buffer register.

Scaling example: To load a velocity value of 1.750 counts/cycle, multiply by 65,536 (giving 114,688), and load the resulting number as a 32-bit number; giving 0001 in the high word and C000h in the low word. Numbers returned by GetVelocity must be divided by 65,536 to convert to units of counts/cycle.

Restrictions

SetVelocity may not be issued while an axis is in motion with the S-curve profile.

SetVelocity is not valid in electronic gearing profile mode.

The velocity cannot be negative, except in the velocity contouring profile mode.

SetVelocity is a buffered command. The value set using this command will not take effect until the next Update or MultiUpdate instruction.

see

Set/GetAcceleration, Set/GetDeceleration, Set/GetJerk, Set/GetPosition, MultiUpdate, Update

Update 1Ah

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

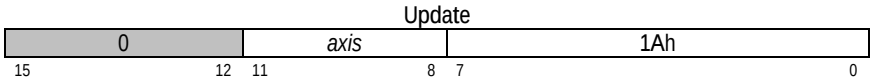
Syntax

Update *axis*

Arguments

Name	*Instance*	*Encoding*
axis	Axis1	0
	Axis2	1
	Axis3	2
	Axis4	3

Packet structure



Description

Update causes all buffered data parameters to be copied into the corresponding run-time registers on the specified *axis*.

The following table shows the buffered commands and variables which are made active as a result of the Update command.

Type	Command
General	ClearPositionError
Trajectory	Acceleration Deceleration GearRatio Jerk Position ProfileMode StopMode Velocity
Servo	DerivativeTime IntegrationLimit Kaff Kd Ki Kp Kvff
Motor	MotorCommand

Restrictions

see

MultiUpdate

WriteBuffer C8h

Motor types

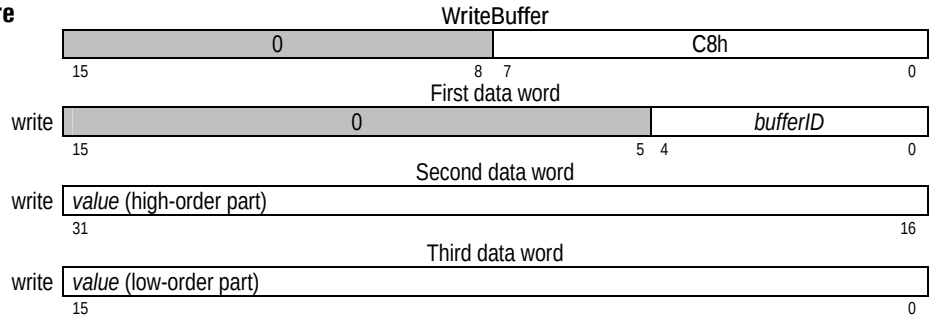
DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

Syntax WriteBuffer *bufferID value*

Arguments

Name	*Type*	*Range*
bufferID	unsigned 16 bits	0 *to* 31
value	signed 32 bits	-2^{31} *to* $2^{31}-1$

Packet structure



Description WriteBuffer writes the 32-bit value into the location pointed to by the write buffer index in the specified buffer. After the contents have been written, the write index is incremented by 1; if the result is equal to the buffer length (set by SetBufferLength), the index is reset to 0.

Restrictions

see ReadBuffer, Set/GetBufferWriteIndex

WriteIO 82h

Motor types

DC Brush		Brushless DC		Microstepping		Pulse & Direction
MC2100	MC2800	MC2300	MC2800	MC2400	MC2800	MC2500

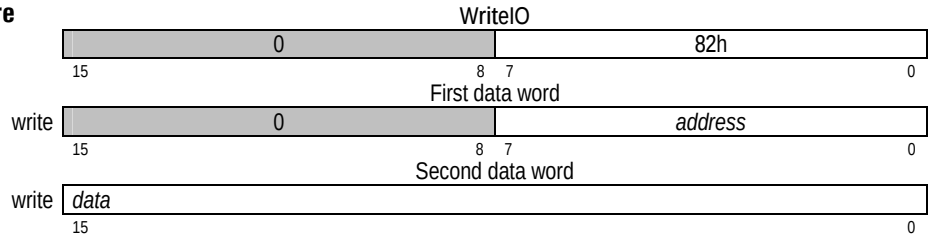
Syntax

WriteIO *address data*

Arguments

Name	*Type*	*Range*
address	unsigned 16 bits	0 *to* 255
data	unsigned 16 bits	0 *to* 2^{16}-1

Packet structure



Description

WriteIO writes one 16-bit word of data at *address*. Address is an offset from location 1000h of the motion processor's peripheral device address space.

The format and interpretation of the 16-bit data word are dependent on the user-defined device being addressed. User-defined I/O can be used to implement a variety of features such as additional parallel I/O, flash memory for non-volatile configuration information storage, or display devices such as LED arrays.

Restrictions

see

ReadIO

3 Instruction Summary Tables

3.1 Descriptions by Functional Category

Breakpoints and Interrupts		Page
ClearInterrupt	Reset interrupt line.	11
Set/GetBreakPoint	Set/Get breakpoint type.	51
GetBreakPointValue	Set interrupt mask; Get breakpoint comparison value.	53
GetInterruptAxis	Get the axes with pending interrupts.	27
GetInterruptMask	Get interrupt mask.	70
SetBreakPoint	Set breakpoint type.	51
SetBreakPointValue	Set breakpoint comparison value.	53
Set/GetInterruptMask	Set/Get interrupt mask.	70
Commutation		
Set/GetCommutationMode	Set/Get the commutation mode (Hall-based, sinusoidal, or microstepping).	60
GetPhaseAngle	Set/Get current commutation phase angle.	86
GetPhaseCommand	Get the motor output command for a given phase A, B, or C.	28
Set/GetPhaseCorrectionMode	Set/Get phase correction mode (on or off).	87
Set/GetPhaseCounts	Set/Get number of encoder counts per commutation cycle.	88
Set/GetPhaseInitializeMode	Set/Get phase initialization method (hall-based or algorithmic).	89
Set/GetPhaseInitializeTime	Get the time parameters for algorithmic phase initialization.	90
Set/GetPhaseOffset	Set/Get phase offset value.	91
Set/GetPhasePrescale	Set/Get commutation prescaler mode.	92
InitializePhase	Perform phase initialization procedure.	35
SetCommutationMode	Set the commutation mode (Hall-based, sinusoidal, or microstepping).	60
SetNumberPhases	Set the number of phases (1, 2, or 3).	84
SetPhaseAngle	Set current commutation phase angle.	86
SetPhaseCorrectionMode	Set phase correction mode (on or off).	87
SetPhaseCounts	Set number of encoder counts per commutation cycle.	88
SetPhaseInitializeMode	Set phase initialization method (hall-based or algorithmic).	89
SetPhaseInitializeTime	Set the time parameters for algorithmic phase initialization.	90
SetPhaseOffset	Set phase offset value.	91
SetPhasePrescale	Set commutation prescaler mode (enable or disable).	92
Digital Servo Filter		
ClearPositionError	Set position error to 0.	12
Set/GetAutoStopMode	Set/Get auto stop on position error (on or off).	47
Set/GetDerivative	Set/Get the derivative of the error signal.	22
Set/GetDerivativeTime	Set/Get derivative sampling time.	62
GetIntegral	Get integrated position error value.	26
Set/GetIntegrationLimit	Set/Get integration limit.	69
Set/GetKaff	Set/Get acceleration feedforward gain.	72
Set/GetKd	Set/Get derivative gain.	73
Set/GetKi	Set/Get integral gain.	74
Set/GetKout	Set/Get servo filter output scaler.	75
Set/GetKp	Set/Get proportional gain.	76
Set/GetKvff	Set/Get velocity feedforward gain.	77
Set/GetMotorBias	Set/Get motor output bias.	80
Set/GetMotorLimit	Set/Get motor output limit.	82
GetPositionError	Get actual position error.	29

Set/GetPositionErrorLimit	Set/Get the maximum position error limit.	94
Encoder		
AdjustActualPosition	Sums the specified offset with the actual encoder position.	10
Set/GetActualPosition	Set/Get the actual encoder position.	45
Set/GetActualPositionUnits	Set/Get the unit type returned for the actual encoder position.	46
GetActualVelocity	Get the actual encoder velocity.	15
Set/GetCaptureSource	Set/Get the capture source (home or index).	59
GetCaptureValue	Get current axis position capture value, and reset the capture.	16
Set/GetEncoderModulus	Get the full scale range of the parallel-word encoder.	64
Set/GetEncoderSource	Set/Get the encoder type.	65
Set/GetEncoderToStepRatio	Set/Get encoder count to step ratio.	66
SetActualPosition	Set the actual encoder position.	45
SetActualPositionUnits	Set the unit type returned for the actual encoder position.	46
SetCaptureSource	Set capture source (home or index).	59
SetEncoderModulus	Set the full scale range of the parallel-word encoder.	64
SetEncoderSource	Set encoder type (incremental or 16-bit parallel word).	65
SetEncoderToStepRatio	Set encoder count to step ratio.	66
External RAM		
Set/GetBufferLength	Set/Get the length of a memory buffer.	55
Set/GetBufferReadIndex	Set/Get the buffer read pointer for a particular buffer.	56
Set/GetBufferStart	Set/Get the start location of a memory buffer.	57
Set/GetBufferWriteIndex	Set/Get the buffer write pointer for a particular buffer.	58
ReadBuffer	Read a long word value from a buffer memory location.	39
WriteBuffer	Write a long word value to a buffer memory location.	120
Motor Output		
GetCurrentMotorCommand	Read the current motor command value.	21
Set/GetMotorCommand	Set direct value to motor output register, Read buffered motor output command.	81
Set/GetMotorMode	Set/Get motor loop mode.	83
Set/GetOutputMode	Set/Get the motor output mode (PWM sign-magnitude, PWM 50%, or DAC).	85
SetStepRange	Sets the allowable range (in KHz) for step output generation.	106
Profile generation		
Set/GetAcceleration	Set/Get acceleration limit.	44
GetCommandedAcceleration	Get commanded (instantaneous desired) acceleration.	18
GetCommandedPosition	Get commanded (instantaneous desired) position.	19
GetCommandedVelocity	Get commanded (instantaneous desired) velocity.	20
Set/GetDeceleration	Set/Get deceleration limit.	61
Set/GetGearMaster	Set/Get the electronic gear mode master axis and source.	67
Set/GetGearRatio	Set/Get commanded electronic gear ratio.	68
Set/GetJerk	Set/Get jerk limit.	71
Set/GetPosition	Set/Get destination position.	93
Set/GetProfileMode	Set/Get current profile mode.	95
Set/GetStartVelocity	Set/Get start velocity.	105
Set/GetStopMode	Set/Get stop command; abrupt, smooth, or none.	107
Set/GetVelocity	Set/Get velocity limit.	118
MultiUpdate	Forces buffered command values to become active for multiple axes.	36
Update	Forces buffered command values to become active.	119
Servo loop control		
Set/GetAxisMode	Set/Get the axis operation mode (enabled or disabled).	48
Set/GetLimitSwitchMode	Set/Get limit switch mode (on or off).	78
Set/GetMotionCompleteMode	Get the motion complete mode (target or actual).	79
Set/GetSampleTime	Set/Get servo loop sample time.	96
Set/GetSettleTime	Set/Get the axis-settled time.	100

Set/GetSettleWindow	Set/Get the settle-window boundary value.	101
GetTime	Get current chipset time (number of servo loops).	31
Set/GetTrackingWindow	Set/Get the tracking window boundary value.	117
Status Registers and AxisOut Indicator		
GetActivityStatus	Get Activity Status.	13
Set/GetAxisOutSource	Set/Get axis out signal monitor source.	49
GetEventStatus	Get event status word.	23
GetSignalStatus	Get the current axis signal status register.	30
Set/GetSignalSense	Set/Get the interpretation of the signal status bits.	102
ResetEventStatus	Reset bits in event status word.	43
Traces		
GetTraceCount	Get the number of traced data points.	32
Set/GetTraceMode	Set/Get the trace mode (rolling or one-time).	109
Set/GetTracePeriod	Set/Get the trace period.	110
GetTraceStart	Get the trace start condition.	111
GetTraceStatus	Get the trace status word.	33
Set/GetTraceStop	Set/Get the trace stop condition.	113
Set/GetTraceVariable	Set/Get a trace variable setting.	115
Miscellaneous		
GetChecksum	Reads the internal chip checksum.	17
GetDiagnosticPortMode	Get the diagnostic port valid instruction mode.	63
GetHostIOError	Get the most recent I/O error code.	25
Set/GetSerialPortMode	Set/Get the serial-port configuration data.	98
Set/GetSynchronizationMode	Set/Get the synchronization mode.	108
GetVersion	Get chipset software version information.	34
NoOperation	Perform no operation, used to verify communications.	37
ReadIO	Read user defined I/O value.	40
Reset	Reset chipset.	41
SetDiagnosticPortMode	Set the diagnostic port valid instruction mode (limited or full).	63
WriteIO	Write user-defined I/O value.	121

3.2 Alphabetical Listing

Note: Get/Set instruction pairs are shown together on the same line of the table

Instruction	Code	Page	Instruction	Code	Page
AdjustActualPosition	F5	10			
ClearInterrupt	AC	11			
ClearPositionError	47	12			
GetAcceleration	4C	44	SetAcceleration	90	44
GetActivityStatus	A6	13			
GetActualPosition	37	45	SetActualPosition	4D	45
GetActualPositionUnits	BF	46	SetActualPositionUnits	BE	46
GetActualVelocity	AD	15			
GetAutoStopMode	D3	47	SetAutoStopMode	D2	47
GetAxisMode	88	48	SetAxisMode	87	48
GetAxisOutSource	EE	49	SetAxisOutSource	ED	49
GetBreakpoint	D5	51	SetBreakpoint	D4	51
GetBreakpointValue	D7	53	SetBreakpointValue	D6	53
GetBufferLength	C3	55	SetBufferLength	C2	25
GetBufferReadIndex	C7	56	SetBufferReadIndex	C6	56
GetBufferStart	C1	57	SetBufferStart	C0	57
GetBufferWriteIndex	C5	58	SetBufferWriteIndex	C4	58
GetCaptureSource	D9	59	SetCaptureSource	D8	59
GetChecksum	F8	17			
GetCaptureValue	36	16			
GetCommandedAcceleration	A7	18			
GetCommandedPosition	1D	19			
GetCommandedVelocity	1E	20			
GetCommutationMode	E3	60	SetCommutationMode	E2	60
GetCurrentMotorCommand	3A	21			
GetDeceleration	92	61	SetDeceleration	91	61
GetDerivative	9B	22			
GetDerivativeTime	9D	62	SetDerivativeTime	9C	62
GetDiagnosticPortMode	8A	63	SetDiagnosticPortMode	89	63
GetEncoderModulus	8E	64	SetEncoderModulus	8D	64
GetEncoderSource	DB	65	SetEncoderSource	DA	65
GetEncoderToStepRatio	DF	66	SetEncoderToStepRatio	DE	66
GetEventStatus	31	23			
GetGearMaster	AF	57	SetGearMaster	AE	67
GetGearRatio	59	68	SetGearRatio	14	68
GetHostIOError	A5	25			
GetIntegral	9A	26			
GetIntegrationLimit	96	69	SetIntegrationLimit	95	69
GetInterruptAxis	E1	27			
GetInterruptMask	56	70	SetInterruptMask	2F	70
GetJerk	58	71	SetJerk	13	71
GetKaff	94	72	SetKaff	93	72
GetKd	52	73	SetKd	27	73
GetKi	51	74	SetKi	26	74
GetKout	9F	75	SetKout	9E	75
GetKp	50	76	SetKp	25	76
GetKvff	54	77	SetKvff	2B	77
GetLimitSwitchMode	81	78	SetLimitSwitchMode	80	78
GetMotionCompleteMode	EC	79	SetMotionCompleteMode	EB	79
GetMotorBias	2D	80	SetMotorBias	0F	80
GetMotorCommand	69	81	SetMotorCommand	77	81
GetMotorLimit	07	82	SetMotorLimit	06	82
GetMotorMode	DD	83	SetMotorMode	DC	83

Instruction	Code	Page	Instruction	Code	Page
GetNumberPhases	86	84	SetNumberPhases	85	84
GetOutputMode	6E	85	SetOutputMode	E0	85
GetPhaseAngle	2C	86	SetPhaseAngle	84	86
GetPhaseCommand	EA	28			
GetPhaseCorrectionMode	E9	87	SetPhaseCorrectionMode	E8	87
GetPhaseCounts	7D	88	SetPhaseCounts	75	88
GetPhaseInitializeMode	E5	89	SetPhaseInitializeMode	E4	89
GetPhaseInitializeTime	7C	90	SetPhaseInitializeTime	72	90
GetPhaseOffset	7B	91	SetPhaseOffset	76	91
GetPhasePrescale	E7	92	SetPhasePrescale	E6	92
GetPosition	4A	93	SetPosition	10	93
GetPositionError	99	29			
GetPositionErrorLimit	98	94	SetPositionErrorLimit	97	94
GetProfileMode	A1	95	SetProfileMode	A0	95
GetSampleTime	61	96	SetSampleTime	38	96
GetSerialPortMode	8C	98	SetSerialPortMode	8B	98
GetSettleTime	AB	100	SetSettleTime	AA	100
GetSettleWindow	BD	101	SetSettleWindow	BC	101
GetSignalStatus	A4	30			
GetSignalSense	A3	102	SetSignalSense	A2	102
GetStartVelocity	6B	105	SetStartVelocity	6A	105
GetStepRange	CE	106	SetStepRange	CF	106
GetStopMode	D1	107	SetStopMode	D0	107
GetSynchronizationMode	F3	108	SetSynchronizationMode	F2	108
GetTime	3E	31			
GetTraceCount	BB	32			
GetTraceMode	B1	109	SetTraceMode	B0	109
GetTracePeriod	B9	110	SetTracePeriod	B8	110
GetTraceStart	B3	111	SetTraceStart	B2	111
GetTraceStatus	BA	33			
GetTraceStop	B5	113	SetTraceStop	B4	113
GetTraceVariable	B7	115	SetTraceVariable	B6	115
GetTrackingWindow	A9	117	SetTrackingWindow	A8	117
GetVelocity	4B	118	SetVelocity	11	118
GetVersion	8F	34			
InitializePhase	7A	35			
MultiUpdate	5B	36			
NoOperation	00	37			
ReadAnalog	EF	38			
ReadBuffer	C9	39			
ReadIO	83	40			
Reset	39	41			
ResetEventStatus	34	43			
Update	1A	119			
WriteBuffer	C8	120			
WriteIO	82	121			

3.3 Numeric Listing

Code	Instruction	Page	Code	Instruction	Page	Code	Instruction	Page
00	NoOperation	37	85	SetNumberPhases	84	BE	SetActualPositionUnits	46
06	SetMotorLimit	82	86	GetNumberPhases	84	BF	GetActualPositionUnits	46
07	GetMotorLimit	82	87	SetAxisMode	48	C0	SetBufferStart	57
0F	SetMotorBias	80	88	GetAxisMode	48	C1	GetBufferStart	57
10	SetPosition	93	89	SetDiagnosticPortMode	63	C2	SetBufferLength	55
11	SetVelocity	118	8A	GetDiagnosticPortMode	63	C3	GetBufferLength	55
13	SetJerk	71	8B	SetSerialPortMode	98	C4	SetBufferWriteIndex	58
14	SetGearRatio	68	8C	GetSerialPortMode	98	C5	GetBufferWriteIndex	58
1A	Update	119	8D	SetEncoderModulus	64	C6	SetBufferReadIndex	56
1D	GetCommandedPosition	19	8E	GetEncoderModulus	64	C7	GetBufferReadIndex	56
1E	GetCommandedVelocity	20	8F	GetVersion	34	C8	WriteBuffer	120
25	SetKp	76	90	SetAcceleration	44	C9	ReadBuffer	39
26	SetKi	74	91	SetDeceleration	61	CE	GetStepRange	106
27	SetKd	73	92	GetDeceleration	61	CF	SetStepRange	106
2B	SetKvff	77	93	SetKaff	72	D0	SetStopMode	107
2C	GetPhaseAngle	86	94	GetKaff	72	D1	GetStopMode	107
2D	GetMotorBias	80	95	SetIntegrationLimit	69	D2	SetAutoStopMode	47
2F	SetInterruptMask	70	96	GetIntegrationLimit	69	D3	GetAutoStopMode	47
31	GetEventStatus	23	97	SetPositionErrorLimit	94	D4	SetBreakpoint	51
34	ResetEventStatus	43	98	GetPositionErrorLimit	94	D5	GetBreakpoint	51
36	GetCaptureValue	16	99	GetPositionError	29	D6	SetBreakpointValue	53
37	GetActualPosition	45	9A	GetIntegral	26	D7	GetBreakpointValue	53
38	SetSampleTime	96	9B	GetDerivative	22	D8	SetCaptureSource	59
39	Reset	41	9C	SetDerivativeTime	62	D9	GetCaptureSource	59
3A	GetCurrentMotorCommand	21	9D	GetDerivativeTime	62	DA	SetEncoderSource	65
3E	GetTime	31	9E	SetKout	75	DB	GetEncoderSource	65
47	ClearPositionError	12	9F	GetKout	75	DC	SetMotorMode	83
4A	GetPosition	93	A0	SetProfileMode	95	DD	GetMotorMode	83
4B	GetVelocity	118	A1	GetProfileMode	95	DE	SetEncoderToStepRatio	66
4C	GetAcceleration	44	A2	SetSignalSense	102	DF	GetEncoderToStepRatio	66
4D	SetActualPosition	45	A3	GetSignalSense	102	E0	SetOutputMode	85
50	GetKp	76	A4	GetSignalStatus	30	E1	GetInterruptAxis	27
51	GetKi	74	A5	GetHostIOError	25	E2	SetCommutationMode	60
52	GetKd	73	A6	GetActivityStatus	13	E3	GetCommutationMode	60
54	GetKvff	77	A7	GetCommandedAcceleration	18	E4	SetPhaseInitializeMode	89
56	GetInterruptMask	70	A8	SetTrackingWindow	117	E5	GetPhaseInitializeMode	89
58	GetJerk	71	A9	GetTrackingWindow	117	E6	SetPhasePrescale	92
59	GetGearRatio	68	AA	SetSettleTime	100	E7	GetPhasePrescale	92
5B	MultiUpdate	36	AB	GetSettleTime	100	E8	SetPhaseCorrectionMode	87
61	GetSampleTime	96	AC	ClearInterrupt	11	E9	GetPhaseCorrectionMode	87
69	GetMotorCommand	81	AD	GetActualVelocity	15	EA	GetPhaseCommand	28
6A	SetStartVelocity	105	AE	SetGearMaster	67	EB	SetMotionCompleteMode	79
6B	GetStartVelocity	105	AF	GetGearMaster	67	EC	GetMotionCompleteMode	79
6E	GetOutputMode	85	B0	SetTraceMode	109	ED	SetAxisOutSource	49
72	SetPhaseInitializeTime	90	B1	GetTraceMode	109	EE	GetAxisOutSource	49
75	SetPhaseCounts	88	B2	SetTraceStart	111	EF	ReadAnalog	38
76	SetPhaseOffset	91	B3	GetTraceStart	111	F2	SetSynchronizationMode	108
77	SetMotorCommand	81	B4	SetTraceStop	113	F3	GetSynchronizationMode	108
7A	InitializePhase	35	B5	GetTraceStop	113	F5	AdjustActualPosition	10
7B	GetPhaseOffset	91	B6	SetTraceVariable	115	F8	GetChecksum	17
7C	GetPhaseInitializeTime	90	B7	GetTraceVariable	115			
7D	GetPhaseCounts	88	B8	SetTracePeriod	110			
80	SetLimitSwitchMode	80	B9	GetTracePeriod	110			
81	GetLimitSwitchMode	80	BA	GetTraceStatus	33			
82	WriteIO	121	BB	GetTraceCount	32			
83	ReadIO	40	BC	SetSettleWindow	101			
84	SetPhaseAngle	86	BD	GetSettleWindow	101			

Contact Information

For additional information, or for technical assistance, please contact PMD at (781) 674-9860.

You may also e-mail your request to: support@pmdcorp.com

Visit our website at: http://www.pmdcorp.com

Performance Motion Devices
55 Old Bedford Rd.
Lincoln, MA 01773