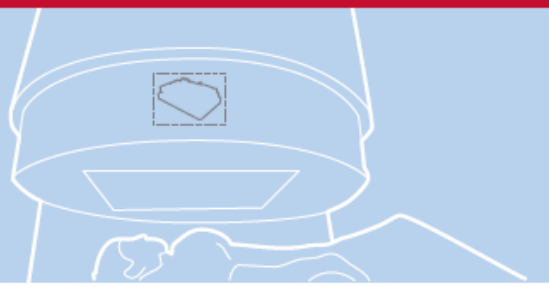
agile
SYSTEMS

Manual

User Manual



microMAX *R* Command Reference

PN# 34-2119 Rev 1
04-25-2007

Table Of Contents

Command Reference

Command Reference Overview

Introduction

The command reference documentation describes all the commands for the MAX3000 and microMAX R series of motion controllers. Developers should consult the documentation for instructions on what commands to use for their applications.

The command reference documentation is provided for users who wish to use the following products:

1. DPWin Command Editor
2. MAXCLib ANSI-C motion control library

Layout Description

Each command is shown using the following table. See below for a description of the fields.

Name

Description			
Transmit	**Receive**	**Execution**	**Command Op-Code**
DPWin Command Editor Syntax			
MAXCLib			
Version Support	**Firmware Minimum**	**Firmware Maximum**	

Name: Each command is described by the command name. The name is the same as the MAXCLib function name and is similiar to the name in the DPWin Command Editor.

Description: This field has the explanation of the command and also describes any pre-conditions. Some commands need specific registers to be set and some commands may need other commands executed before the desired functionality will work properly.

Transmit: This field describes the format of the transmit packet type, or the query packet in the query-response model of communication.

Receive: This field describes the format of the receive packet type, or the response packet in the query-response model of communication.

Execution: Whether the command is executed during interrupt processing (Immediate) or in the background time (Queued).

Command Op-Code: This field shows the op-code for the command. This information is advanced and is not normally necessary if programming using the Command Editor or MAXCLib.

DPWin Command Editor: Syntax for the Command Editor.

MAXCLib: Syntax for Agile's MAXCLib ANSI-C motion library calls. See below for a description of parameters:

> **TxPacket**: A pointer to a buffer where the network packet will be formed. This parameter will contain the packet that is to be communicated to the Agile controller. The buffer size must be 10 bytes.
>
> **NodeID**: Specifies the Node ID. The Node ID identifies one Agile controller from another controller on a network. Each controller that is attached to a multi-node network must contain a

unique Node ID between the ranges of 0 - 255. When only one controller is present, a default value of zero (0) can be used for this parameter.

Axis: Specifies the single axis desired. This paramater can be specified as one of the following: A_AXIS1, A_AXIS2, A_AXIS3, A_AXIS4 (as defined in the 'MAXCLib.h' header file).

ControllerAndOrAxes: Specifies the controller level and/or axis level bitmask to be used. This parameter can be specified with one or a group of the following: A_CONTROLLER, A_AXIS1, A_AXIS2, A_AXIS3, A_AXIS4 (as defined in the 'MAXCLib.h' header file).

Axes: Specifies one or a group of axis to be used. This bitmask parameter can be specified as one or a group of the following: A_AXIS1, A_AXIS2, A_AXIS3, A_AXIS4 (as defined in the 'MAXCLib.h' header file).

Register, **Register1**, **Register2**: Defines the controller or axis level register to be used. This parameter can be any valid unsigned 16-bit value or a register name defined in the 'MainReg.h' header file. Refer to the 'Register Reference' document for more information on Agile registers. Note: axis level registers require an additional value be added (see the difference between registers defined in the 'MainReg.h' and 'Axis1Reg.h' header files for an example). Adding A_A1, A_A2, A_A3, A_A4 to the register value will specify the axis level register correctly.

AuxRegister: Defines the auxiliary register to be used. This parameter can be any valid unsigned 16-bit value or a auxiliary register name defined in the 'MainReg.h' header file. Refer to the 'Register Reference' document for more information on Agile registers.

Value: Specifies the value to be used. Refer to the 'Register Reference' document for a listing of valid ranges and units for each register.

Channel: Specifies the data log channel to be used. The MAX3000 family of controllers contain 4 data log channels that can store up to 8000 data values collectively. This parameter is valid from 0 to 3 indicating data log channel 1 to 4 respectively.

SampleIndex: Specifies the data log sample. The MAX3000 family of controllers allow up to 8000 data samples across 4 channels.

Mask: Specifies a mask value that is, generally, used to compare one value to another and indicate a true/false situation.

State: Specifies a true/false condition. A value of 1 is true and 0 is false.

Condition: Used to perform arithmetic operations. This paramater can be specified as one of the following: A_EQUAL, A_NOT_EQUAL, A_GREATER_THAN, A_LESS_THAN, A_GREATER_THAN_OR_EQUAL, A_LESS_THAN_OR_EQUAL (as defined in the 'MAXCLib.h' header file).

Address: Specifies an address, or line number, of a program stored in the controller. The line numbers for stored programs can be seen through the Command Editor of DPWin. An address of 0 will execute the on-board program from the first line. Specifying an address that is out-of-bounds will cause an overflow fault to occur.

RxPacket: A pointer to a buffer where a network packet is stored. This buffer should contain a response packet that was communicated from the Agile controller. The buffer size must be 10 bytes.

Reply: A pointer to an AS_REPLY structure variable. This variable contains important information about a response packet and is used to obtain values such as data logging samples, controller status information, and register values.

Firmware Minimum: The minimum version of controller software supported.

Firmware Maximum: The maximum version of controller software supported.

AuxRegisterEqualRegister

Description: Assigns the value stored in a specified Register to a specified aux Register.			
Transmit: 6F	**Receive**: 1T	**Execution**: Queue	**Command Op-Code**: 115
DPWin Command Editor Syntax: [AuxRegister] = [Register]			
MAXCLib Syntax: void AuxRegisterEqualRegister (u8 * TxPacket, u8 NodeID, AS_REGISTER AuxRegister, AS_REGISTER Register)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	All AUX registers		

AuxRegisterEqualValue

Description: Assigns a value to the specified Auxiliary Register.			
Transmit: 5F	**Receive**: 1T	**Execution**: Queue	**Command Op-Code**: 114
DPWin Command Editor Syntax: [AuxRegister] = <Value>			
MAXCLib Syntax: void AuxRegisterEqualValue (u8 * TxPacket, u8 NodeID, AS_REGISTER AuxRegister, f64 Value)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	All AUX registers		

AuxRegisterGet

Description: Reads the value stored in the specified Auxiliary Register.			
Transmit: 2F	**Receive**: 1T	**Execution**: Queue	**Command Op-Code**: 112
DPWin Command Editor Syntax: [AuxRegister] Get			
MAXCLib Syntax: void AuxRegisterGet (u8 * TxPacket, u8 NodeID, AS_REGISTER AuxRegister)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	All AUX registers		

Call

Description: Used in the Command Editor to begin a subroutine. When a stored program is executing this command, the address of the next command is stored on the call stack and the instruction pointer is changed to point to the address of the subroutine label. Each subroutine must end with a Return command.			
Transmit: 3F	**Receive**: 1T	**Execution**: Immediate	**Command Op-Code**: 128
DPWin Command Editor Syntax: Call [Label]			
MAXCLib Syntax: void Call (u8 * TxPacket, u8 NodeID, f64 Value)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	none		

DatalogArm

Description: Gets the datalog ready for a new trigger. When using the data logging tool there are a number of commands and register that must be set in order to log a data properly. Consult the Register Reference for further information.

Typical Set up for time based Datalogging.

1. Select data log mode

   Data Log Sample Mode

2. Set up time between samples.

   Data Log Sample Time

3. Set up number of samples

   Data Log Number of Samples

4. Select triggering method.

   Data Log Trigger Channel

   Data Log Trigger Type

4. Set up what to datalog. There are four data log channal available.

   Data Log Number of Channels

   Data Log Channel A

   Data Log Channel B

   Data Log Channel C

   Data Log Channel D

5. Turn on the Data log tool

   Data Log Enable

6. Send the A_DATALOG_ARM command

7. Send the A_DATALOG_TRIGGER command or issue a move command to autotrigger.

8. Poll for Datalog status

   Data Log Complete

9. Send the A_DATALOG_GET command as many time necessary to retrieve the entire data log.

Transmit: 1F	**Receive**: 1T	**Execution**: Immediate	**Command Op-Code**: 97
DPWin Command Editor Syntax: Datalog Arm			
MAXCLib Syntax: void DatalogArm (u8 * TxPacket, u8 NodeID)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	

DatalogGet

Description: Reads one datalog sample. Note: the datalog buffer size is user configurable, therefore to read the entire datalog buffer multiple A_DATALOG_GET must be called. When using the data logging tool there are a number of commands and register that must be set in order to log a data properly.

Typical Set up for time based Datalogging.

1. Select data log mode

   Data Log Sample Mode

2. Set up time between samples.

   Data Log Sample Time

3. Set up number of samples

   Data Log Number of Samples

4. Select triggering method.

   Data Log Trigger Channel

   Data Log Trigger Type

4. Set up what to datalog. There are four data log channal available.

   Data Log Number of Channels

   Data Log Channel A

   Data Log Channel B

   Data Log Channel C

   Data Log Channel D

5. Turn on the Data log tool

   Data Log Enable

6. Send the A_DATALOG_ARM command

7. Send the A_DATALOG_TRIGGER command or issue a move command to autotrigger.

8. Poll for Datalog status

   Data Log Complete

9. Send the A_DATALOG_GET command as many time necessary to retrieve the entire data log.

Transmit: 17F	**Receive**: 4T	**Execution**: Immediate	**Command Op-Code**: 96
DPWin Command Editor Syntax: Datalog Get [Channel] <SampleIndex>			
MAXCLib Syntax: void DatalogGet (u8 * TxPacket, u8 NodeID, u8 Channel, u16 SampleIndex)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	

DatalogTrigger

Description: Manually Triggers the datalog tool. When using the data logging tool there are a number of commands and register that must be set in order to log a data properly.

Typical Set up for time based Datalogging.

1. Select data log mode

   Data Log Sample Mode

2. Set up time between samples.

   Data Log Sample Time

3. Set up number of samples

   Data Log Number of Samples

4. Select triggering method.

   Data Log Trigger Channel

   Data Log Trigger Type

4. Set up what to datalog. There are four data log channal available.

   Data Log Number of Channels

   Data Log Channel A

   Data Log Channel B

   Data Log Channel C

   Data Log Channel D

5. Turn on the Data log tool

   Data Log Enable

6. Send the A_DATALOG_ARM command

7. Send the A_DATALOG_TRIGGER command or issue a move command to autotrigger.

8. Poll for Datalog status

   Data Log Complete

9. Send the A_DATALOG_GET command as many time necessary to retrieve the entire data log.

Transmit: 1F	**Receive**: 1T	**Execution**: Immediate	**Command Op-Code**: 98
DPWin Command Editor Syntax: Datalog Trigger			
MAXCLib Syntax: void DatalogTrigger (u8 * TxPacket, u8 NodeID)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	

GetGroupAStatus

Description: Returns the low 8 bits of all axes status and controller status. This can be used to poll all information simultaneously and get updates of motion done, faults etc.			
Transmit: 1F	**Receive**: 1T	**Execution**: Immediate	**Command Op-Code**: 144
DPWin Command Editor Syntax: GetGroupAStatus			
MAXCLib Syntax: void GetGroupAStatus (u8 * TxPacket, u8 NodeID)			
Version Support	**Minimum**: 9.00	**Maximum**: 10.00	
Related Registers	status registers		

Halt

Description: Halts the execution of a stored program at the current instruction pointer.			
Transmit: 1F	**Receive**: 1T	**Execution**: Immediate	**Command Op-Code**: 255
DPWin Command Editor Syntax: Halt			
MAXCLib Syntax: void Halt (u8 * TxPacket, u8 NodeID)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	none		

IfConditionOnMask

Description: Allows users to have logical decision based on register and value masking. Note: The register is masked with the value and if the result is zero (false) or non-zero (true) the ThenJump command is executed or not. If conditions must be followed with a ThenJump command.			
Transmit: 21F	**Receive**: 1T	**Execution**: Immediate	**Command Op-Code**: 134
DPWin Command Editor Syntax: If [Register] & <MaskValue> [State]			
MAXCLib Syntax: void IfConditionOnMask (u8 * TxPacket, u8 NodeID, AS_REGISTER Register, f64 Mask, boolean State)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	none		

IfConditionOnRegister

Description: Allows users to have logical decision based on a register to register comparison. Note: Includes (==, !=,<, >, <=, >=). If conditions must be followed with a ThenJump command.			
Transmit: 22F	**Receive**: 1T	**Execution**: Immediate	**Command Op-Code**: 133
DPWin Command Editor Syntax: If [Register] [Condition] [Register]			
MAXCLib Syntax: void IfConditionOnRegister (u8 * TxPacket, u8 NodeID, AS_REGISTER Register1, AS_CONDITION Condition, AS_REGISTER Register2)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	none		

IfConditionOnValue

Description: Allows users to have logical decision based on a Register to value comparison. If conditions must be followed with a ThenJump command.			
Transmit: 21F	**Receive**: 1T	**Execution**: Immediate	**Command Op-Code**: 130
DPWin Command Editor Syntax: If [Register] [Condition] <Value>			
MAXCLib Syntax: void IfConditionOnValue (u8 * TxPacket, u8 NodeID, AS_REGISTER Register, AS_CONDITION Condition, f64 Value)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	none		

Jump

Description: Allows users to jump to a specified address within the stored program. When using MAXCLib the address is equivalent to the line number.			
Transmit: 3F	**Receive**: 1T	**Execution**: Immediate	**Command Op-Code**: 139
DPWin Command Editor Syntax: Jump [Label]			
MAXCLib Syntax: void Jump (u8 * TxPacket, u8 NodeID, f64 Address)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	none		

MoveAbsoluteRegister

Description: Used to set up an absolute move on a specified axis where the position is provided through a Register. A MoveGo command will begin the motion.			
Transmit: 9F	**Receive**: 1T	**Execution**: Queue	**Command Op-Code**: 53
DPWin Command Editor Syntax: Move Absolute [Axis] [Register]			
MAXCLib Syntax: void MoveAbsoluteRegister (u8 * TxPacket, u8 NodeID, u8 Axis, AS_REGISTER Register)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	Acceleration Negative Maximum Acceleration Positive Maximum Jerk Acceleration Start Maximum Jerk Acceleration Stop Maximum Jerk Deceleration Start Maximum Jerk Deceleration Stop Maximum Velocity Maximum		

MoveAbsoluteValue

Description: Used to set up an absolute move on a specified axis where the position is provided through by a value. A MoveGo command will begin the motion.			
Transmit: 8F	**Receive**: 1T	**Execution**: Queue	**Command Op-Code**: 52
DPWin Command Editor Syntax: Move Absolute [Axis] <Value>			
MAXCLib Syntax: void MoveAbsoluteValue (u8 * TxPacket, u8 NodeID, u8 Axis, f64 Value)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	Acceleration Negative Maximum Acceleration Positive Maximum Jerk Acceleration Start Maximum Jerk Acceleration Stop Maximum Jerk Deceleration Start Maximum Jerk Deceleration Stop Maximum Velocity Maximum		

MoveGo

Description: Used to start a motion. Note a MoveRelative or a MoveAbsolute command must precede the Go command. One or multiple axes can be specified for a start of motion.			
Transmit: 18F	**Receive**: 1T	**Execution**: Both	**Command Op-Code**: 56
DPWin Command Editor Syntax: Move Go [Axes]			
MAXCLib Syntax: void MoveGo (u8 * TxPacket, u8 NodeID, u8 Axes)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	None		

MoveJogRegister

Description: Used to set a jog velocity from a value stored in a Register and starts the motion on the specified axis. A velocity profile must be configured.			
Transmit: 9F	**Receive**: 1T	**Execution**: Both	**Command Op-Code**: 55
DPWin Command Editor Syntax: Move Jog [Axis] [Register]			
MAXCLib Syntax: void MoveJogRegister (u8 * TxPacket, u8 NodeID, u8 Axis, AS_REGISTER Register)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	Acceleration Positive Maximum Jerk Acceleration Start Maximum Jerk Acceleration Stop Maximum		

MoveJogValue

Description: Used to set a jog velocity to the specified value in count/ms, and starts the motion on the specified axis.A velocity profile must be configured.			
Transmit: 8F	**Receive**: 1T	**Execution**: Both	**Command Op-Code**: 54
DPWin Command Editor Syntax: Move Jog [Axis] <Value>			
MAXCLib Syntax: void MoveJogValue (u8 * TxPacket, u8 NodeID, u8 Axis, f64 Value)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	Acceleration Positive Maximum Jerk Acceleration Start Maximum Jerk Acceleration Stop Maximum		

MoveRelativeRegister

Description: Used to set up a relative move on a specified axis. The distance is provided through a Register and the end position is calculated relative to the present position. A MoveGo command will begin the motion.			
Transmit: 9F	**Receive**: 1T	**Execution**: Queue	**Command Op-Code**: 51
DPWin Command Editor Syntax: Move Relative [Axis] [Register]			
MAXCLib Syntax: void MoveRelativeRegister (u8 * TxPacket, u8 NodeID, u8 Axis, AS_REGISTER Register)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	Acceleration Negative Maximum Acceleration Positive Maximum Jerk Acceleration Start Maximum Jerk Acceleration Stop Maximum Jerk Deceleration Start Maximum Jerk Deceleration Stop Maximum Velocity Maximum		

MoveRelativeValue

Description: Used to set up a relative move on a specified axis. The distance is provided by a value and the end position is calculated relative to the current position. A MoveGo command will begin the motion.			
Transmit: 8F	**Receive**: 1T	**Execution**: Queue	**Command Op-Code**: 50
DPWin Command Editor Syntax: Move Relative [Axis] <Value>			
MAXCLib Syntax: void MoveRelativeValue (u8 * TxPacket, u8 NodeID, u8 Axis, f64 Value)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	Acceleration Negative Maximum Acceleration Positive Maximum Jerk Acceleration Start Maximum Jerk Acceleration Stop Maximum Jerk Deceleration Start Maximum Jerk Deceleration Stop Maximum Velocity Maximum		

ProcessReply

Description: Takes the RxPacket buffer and fills the Reply structure with appropraite data based on the reply packet type. Also checks for valid CRC8.			
Transmit:	**Receive**:	**Execution**:	**Command Op-Code**:
DPWin Command Editor Syntax: ProcessReply			
MAXCLib Syntax: boolean ProcessReply (u8 * RxPacket, AS_REPLY * Reply)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	none		

ProfileDeltaTimeRegister

Description: Used to select the delta time for profile motion segments. The delta time is provided through a Register. See ProfileSplinePositionValue for more information.			
Transmit: 9F	**Receive**: 1T	**Execution**: Immediate	**Command Op-Code**: 69
DPWin Command Editor Syntax: Profile DeltaTime [Axis] [Register]			
MAXCLib Syntax: void ProfileDeltaTimeRegister (u8 * TxPacket, u8 NodeID, u8 Axis, AS_REGISTER Register)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	Profile Arm Profile Lower Threshold Profile Upper Threshold		

ProfileDeltaTimeValue

Description: Used to select the delta time for profile motion segments. The delta time is provided by a value. See ProfileSplinePositionValue for more information.			
Transmit: 8F	**Receive**: 1T	**Execution**: Immediate	**Command Op-Code**: 68
DPWin Command Editor Syntax: Profile DeltaTime [Axis] <Value>			
MAXCLib Syntax: void ProfileDeltaTimeValue (u8 * TxPacket, u8 NodeID, u8 Axis, f64 Value)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	Profile Arm Profile Lower Threshold Profile Upper Threshold		

ProfileEnd

Description: Stops the profile segment algorithm from processing any more profile points. In PVT mode there are number of commands and register that must be set in order to successfully execute PVT mode. The commands and registers also need to be called in a specific sequence. See ProfileSplinePositionValue for more information.			
Transmit: 7F	**Receive**: 1T	**Execution**: Immediate	**Command Op-Code**: 72
DPWin Command Editor Syntax: Profile End [Axis]			
MAXCLib Syntax: void ProfileEnd (u8 * TxPacket, u8 NodeID, u8 Axis)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	Profile Arm Profile Lower Threshold Profile Upper Threshold		

ProfileFlush

Description: Used to clear the profile queue and reset the profile pointer. In PVT mode there are number of commands and register that must be set in order to successfully execute PVT mode. The commands and registers also need to be called in a specific sequence. See ProfileSplinePositionValue for more information.			
Transmit: 8F	**Receive**: 1T	**Execution**: Immediate	**Command Op-Code**: 74
DPWin Command Editor Syntax: Profile Flush [Axis] <Value>			
MAXCLib Syntax: void ProfileFlush (u8 * TxPacket, u8 NodeID, u8 Axis, f64 Value)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	Profile Arm Profile Lower Threshold Profile Upper Threshold		

ProfilePositionRegister

Description: Used to send a position value stored in a Register to the specified profile queue. Each profile segment has to end with a profile position command. In PVT mode there are number of commands and registers that must be set in order to successfully execute PVT mode. The commands and registers also need to be called in a specific sequence. See ProfilePositionValue for further information.			
Transmit: 9F	**Receive**: 1T	**Execution**: Immediate	**Command Op-Code**: 65
DPWin Command Editor Syntax: Profile Position [Axis] [Register]			
MAXCLib Syntax: void ProfilePositionRegister (u8 * TxPacket, u8 NodeID, u8 Axis, AS_REGISTER Register)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	Profile Arm Profile Lower Threshold Profile Upper Threshold		

ProfilePositionValue

Description: Used to send a position value to the specified profile queue. Each profile segment has to end with a profile position command. In PVT mode there are number of commands and registers that must be set in order to successfully execute PVT mode. The commands and registers also need to be called in a specific sequence.

Typical usage.

1.Setup PVT registers

Profile Lower Threshold

Profile Upper Threshold

Profile Arm

2. Send PVT Flush Command

ProfileFlush

3. Send PVT points

ProfilePositionValue

ProfileVelocityValue

4. Sent PVT Delta time.

ProfileDeltaTimeValue

5. Send PVT start command

ProfileStartArmed

6. Send PVT points. Note: These two commands have to be sent for each point while the queue is within the thresholds.

ProfilePositionValue

ProfileVelocityValue

7. Send PVT stop

ProfileEnd

Transmit: 8F	**Receive**: 1T	**Execution**: Immediate	**Command Op-Code**: 64
DPWin Command Editor Syntax: Profile Position [Axis] <Value>			
MAXCLib Syntax: void ProfilePositionValue (u8 * TxPacket, u8 NodeID, u8 Axis, f64 Value)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	Profile Arm Profile Lower Threshold Profile Upper Threshold		

ProfileSetRegisterRegister

Description: Used to set Registers from within the profile queue. Similar to the RegisterEqualRegister command but sent through the profile queue to synchronize with the profile motion. Ex. During the execution of a profile the user may want to modify the KP term in the position loop. This command can be stored in the profile queue and when the profile algorithm gets to the specified queue location with this command, it will set the Kp Register with the specified value stored in another Register . In PVT mode there are number of commands and register that must be set in order to successfully execute in PVT mode. The commands and registers also need to be called in a specific sequence.			
Transmit: 11F	**Receive**: 1T	**Execution**: Immediate	**Command Op-Code**: 71
DPWin Command Editor Syntax: Profile SetRegister [Axis] [Register] [Register]			
MAXCLib Syntax: void ProfileSetRegisterRegister (u8 * TxPacket, u8 NodeID, u8 Axis, AS_REGISTER Register1, AS_REGISTER Register2)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	Profile Arm Profile Lower Threshold Profile Upper Threshold		

ProfileSetRegisterValue

Description: Used to set Registers from within the profile queue. Similar to the RegisterEqualValue command but sent through the profile queue to synchronize with the profile motion. Ex. During the execution of a profile the user may want to want to change outputs when reaching certain positions. This command can be stored in the profile queue and when the profile algorithm gets to the specified queue location with this command, it will set the output register with the specified value.			
Transmit: 10F	**Receive**: 1T	**Execution**: Immediate	**Command Op-Code**: 70
DPWin Command Editor Syntax: Profile SetRegister [Axis] [Register] <Value>			
MAXCLib Syntax: void ProfileSetRegisterValue (u8 * TxPacket, u8 NodeID, u8 Axis, AS_REGISTER Register, f64 Value)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	Profile Arm Profile Lower Threshold Profile Upper Threshold		

ProfileSplinePositionRegister

Description: Used to send a spline position value stored in a Register to the specified profile queue. Each profile segment has to end with a profile spline position command. In PVT mode there are number of commands and registers that must be set in order to successfully execute in PVT mode. The commands and registers also need to be called in a specific sequence. See ProfileSplinePositionValue for further information.			
Transmit: 9F	**Receive**: 1T	**Execution**: Immediate	**Command Op-Code**: 79
DPWin Command Editor Syntax: Profile SplinePosition [Axis] [Register]			
MAXCLib Syntax: void ProfileSplinePositionRegister (u8 * TxPacket, u8 NodeID, u8 Axis, AS_REGISTER Register)			
Version Support	**Minimum**: 9.00	**Maximum**: 10.00	
Related Registers	Profile Arm Profile Lower Threshold Profile Upper Threshold		

ProfileSplinePositionValue

Description: Used to send a spline position value to the specified profile queue. Each profile segment has to end with a profile spline position command. In PVT mode there are number of commands and registers that must be set in order to successfully execute in PVT mode. The commands and registers also need to be called in a specific sequence.

Typical usage.

1.Setup PVT registers

Profile Lower Threshold

Profile Upper Threshold

Profile Arm

2. Send PVT Flush Command

ProfileFlush

3. Send PVT points

ProfileSplinePositionValue

4. Sent PVT Delta time.

ProfileDeltaTimeValue

5. Send PVT start command

ProfileStartArmed

6. Send PVT points. Note: This command has to be sent for each point while the queue is within the thresholds.

ProfileSplinePositionValue

7. Send PVT stop

ProfileEnd

Transmit: 8F	**Receive**: 1T	**Execution**: Immediate	**Command Op-Code**: 78
DPWin Command Editor Syntax: Profile SplinePosition [Axis] <Value>			
MAXCLib Syntax: void ProfileSplinePositionValue (u8 * TxPacket, u8 NodeID, u8 Axis, f64 Value)			
Version Support	**Minimum**: 9.00	**Maximum**: 10.00	
Related Registers	Profile Arm Profile Lower Threshold Profile Upper Threshold		

ProfileStartArmed

Description: Used to start the profile algorithm. Note: Once this command is executed only the axes specified to run using the Profile arm register will start processing profile points. In PVT mode there are number of commands and register that must be set in order to successfully execute in PVT mode. The commands and registers also need to be called in a specific sequence. See ProfileSplinePositionValue for more information.			
Transmit: 1F	**Receive**: 1T	**Execution**: Both	**Command Op-Code**: 75
DPWin Command Editor Syntax: Profile StartArmed			
MAXCLib Syntax: void ProfileStartArmed (u8 * TxPacket, u8 NodeID)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	Profile Arm Profile Lower Threshold Profile Upper Threshold		

ProfileVelocityRegister

Description: Used to select a velocity though a Register. In PVT mode there are number of commands and registers that must be set in order to successfully execute in PVT mode. The commands and registers also need to be called in a specific sequence. See ProfilePositionValue for further information.			
Transmit: 9F	**Receive**: 1T	**Execution**: Immediate	**Command Op-Code**: 67
DPWin Command Editor Syntax: Profile Velocity [Axis] [Register]			
MAXCLib Syntax: void ProfileVelocityRegister (u8 * TxPacket, u8 NodeID, u8 Axis, AS_REGISTER Register)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	Profile Arm Profile Lower Threshold Profile Upper Threshold		

ProfileVelocityValue

Description: Used to select a velocity by value. In PVT mode there are number of commands and registers that must be set in order to successfully execute in PVT mode. The commands and registers also need to be called in a specific sequence. See ProfilePositionValue for further information.			
Transmit: 8F	**Receive**: 1T	**Execution**: Immediate	**Command Op-Code**: 66
DPWin Command Editor Syntax: Profile Velocity [Axis] <Value>			
MAXCLib Syntax: void ProfileVelocityValue (u8 * TxPacket, u8 NodeID, u8 Axis, f64 Value)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	Profile Arm Profile Lower Threshold Profile Upper Threshold		

RegisterAndEqualRegister

Description: Used to AND a value with register2 from a value in register1 and the result stored in register 1. Ex. Register1 = Register1 AND Register2			
Transmit: 6F	**Receive**: 1T	**Execution**: Immediate	**Command Op-Code**: 29
DPWin Command Editor Syntax: [Register] &= [Register]			
MAXCLib Syntax: void RegisterAndEqualRegister (u8 * TxPacket, u8 NodeID, AS_REGISTER Register1, AS_REGISTER Register2)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	All registers that have write access. Each register in the register reference has a "Writeable" field.		

RegisterAndEqualValue

Description: Used to AND a value with a value in Register1, and the result stored in Register1. Ex. Register1 = Register1 AND 0x0001			
Transmit: 5F	**Receive**: 1T	**Execution**: Immediate	**Command Op-Code**: 28
DPWin Command Editor Syntax: [Register] &= <Value>			
MAXCLib Syntax: void RegisterAndEqualValue (u8 * TxPacket, u8 NodeID, AS_REGISTER Register, f64 Value)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	All registers that have write access. Each register in the register reference has a "Writeable" field.		

RegisterDivideEqualRegister

Description: Used to divide a value from register2 from a value in register1 and the result stored in register 1. Ex. Register1 = Register1 / Register2			
Transmit: 6F	**Receive**: 1T	**Execution**: Immediate	**Command Op-Code**: 25
DPWin Command Editor Syntax: [Register] /= [Register]			
MAXCLib Syntax: void RegisterDivideEqualRegister (u8 * TxPacket, u8 NodeID, AS_REGISTER Register1, AS_REGISTER Register2)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	All registers that have write access. Each register in the register reference has a "Writeable" field.		

RegisterDivideEqualValue

Description: Used to divide a value from a value in Register1, and the result stored in Register1. Ex. Register1 = Register1 / 1			
Transmit: 5F	**Receive**: 1T	**Execution**: Immediate	**Command Op-Code**: 24
DPWin Command Editor Syntax: [Register] /= <Value>			
MAXCLib Syntax: void RegisterDivideEqualValue (u8 * TxPacket, u8 NodeID, AS_REGISTER Register, f64 Value)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	All registers that have write access. Each register in the register reference has a "Writeable" field.		

RegisterEqualRegister

Description: Used to assign Register1 to Register 2. Ex Register1 = Register2.			
Transmit: 6F	**Receive**: 1T	**Execution**: Immediate	**Command Op-Code**: 17
DPWin Command Editor Syntax: [Register] = [Register]			
MAXCLib Syntax: void RegisterEqualRegister (u8 * TxPacket, u8 NodeID, AS_REGISTER Register1, AS_REGISTER Register2)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	All registers that have write access. Each register in the register reference has a "Writeable" field.		

RegisterEqualValue

Description: Used to assign a value to a specified Register. Ex Register = 1.			
Transmit: 5F	**Receive**: 1T	**Execution**: Immediate	**Command Op-Code**: 16
DPWin Command Editor Syntax: [Register] = <Value>			
MAXCLib Syntax: void RegisterEqualValue (u8 * TxPacket, u8 NodeID, AS_REGISTER Register, f64 Value)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	All registers that have write access. Each register in the register reference has a "Writeable" field.		

RegisterGet

Description: Used to read a specified register on the MAX Controller. Note: the data returned is in the units specified by the register.			
Transmit: 2F	**Receive**: 2T	**Execution**: Immediate	**Command Op-Code**: 0
DPWin Command Editor Syntax: [Register] Get			
MAXCLib Syntax: void RegisterGet (u8 * TxPacket, u8 NodeID, AS_REGISTER Register)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	All registers		

RegisterMinusEqualRegister

Description: Used to subtract a value from register2 from a value in register1 and the result stored in register 1. Ex. Register1 = Register1 - Register2			
Transmit: 6F	**Receive**: 1T	**Execution**: Immediate	**Command Op-Code**: 21
DPWin Command Editor Syntax: [Register] -= [Register]			
MAXCLib Syntax: void RegisterMinusEqualRegister (u8 * TxPacket, u8 NodeID, AS_REGISTER Register1, AS_REGISTER Register2)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	All registers that have write access. Each register in the register reference has a "Writeable" field.		

RegisterMinusEqualValue

Description: Used to subtract a value from the value in Register1, and the result stored in Register1. Ex. Register1 = Register1 - 1			
Transmit: 5F	**Receive**: 1T	**Execution**: Immediate	**Command Op-Code**: 20
DPWin Command Editor Syntax: [Register] -= <Value>			
MAXCLib Syntax: void RegisterMinusEqualValue (u8 * TxPacket, u8 NodeID, AS_REGISTER Register, f64 Value)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	All registers that have write access. Each register in the register reference has a "Writeable" field.		

RegisterMultiplyEqualRegister

Description: Used to multiply a value from register2 from a value in register1 and the result stored in register 1. Ex. Register1 = Register1 * Register2			
Transmit: 6F	**Receive**: 1T	**Execution**: Immediate	**Command Op-Code**: 23
DPWin Command Editor Syntax: [Register] *= [Register]			
MAXCLib Syntax: void RegisterMultiplyEqualRegister (u8 * TxPacket, u8 NodeID, AS_REGISTER Register1, AS_REGISTER Register2)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	All registers that have write access. Each register in the register reference has a "Writeable" field.		

RegisterMultiplyEqualValue

Description: Used to multiply a value from a value in Register1, and the result stored in Register1. Ex. Register1 = Register1 * 1			
Transmit: 5F	**Receive**: 1T	**Execution**: Immediate	**Command Op-Code**: 22
DPWin Command Editor Syntax: [Register] *= <Value>			
MAXCLib Syntax: void RegisterMultiplyEqualValue (u8 * TxPacket, u8 NodeID, AS_REGISTER Register, f64 Value)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	All registers that have write access. Each register in the register reference has a "Writeable" field.		

RegisterOrEqualRegister

Description: Used to bitwise OR a value with register2 from a value in register1 and the result stored in register 1. Ex. Register1 = Register1 OR Register2			
Transmit: 6F	**Receive**: 1T	**Execution**: Immediate	**Command Op-Code**: 27
DPWin Command Editor Syntax: [Register] |= [Register]			
MAXCLib Syntax: void RegisterOrEqualRegister (u8 * TxPacket, u8 NodeID, AS_REGISTER Register1, AS_REGISTER Register2)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	All registers that have write access. Each register in the register reference has a "Writeable" field.		

RegisterOrEqualValue

Description: Used to bitwise OR a value with a value in Register1, and the result stored in Register1. Ex. Register1 = Register1 OR 0x0001			
Transmit: 5F	**Receive**: 1T	**Execution**: Immediate	**Command Op-Code**: 26
DPWin Command Editor Syntax: [Register] |= <Value>			
MAXCLib Syntax: void RegisterOrEqualValue (u8 * TxPacket, u8 NodeID, AS_REGISTER Register, f64 Value)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	All registers that have write access. Each register in the register reference has a "Writeable" field.		

RegisterPlusEqualRegister

Description: Used to add a value in Register 2 to a value in Register1 and the result stored in Register1. Ex. Register1 = Register1 +Register2			
Transmit: 6F	**Receive**: 1T	**Execution**: Immediate	**Command Op-Code**: 19
DPWin Command Editor Syntax: [Register] += [Register]			
MAXCLib Syntax: void RegisterPlusEqualRegister (u8 * TxPacket, u8 NodeID, AS_REGISTER Register1, AS_REGISTER Register2)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	All registers that have write access. Each register in the register reference has a "Writeable" field.		

RegisterPlusEqualValue

Description: Used to add a value to a value in Register1, and the result stored in Register1. Ex. Register1 = Register1 +1			
Transmit: 5F	**Receive**: 1T	**Execution**: Immediate	**Command Op-Code**: 18
DPWin Command Editor Syntax: [Register] += <Value>			
MAXCLib Syntax: void RegisterPlusEqualValue (u8 * TxPacket, u8 NodeID, AS_REGISTER Register, f64 Value)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	All registers that have write access. Each register in the register reference has a "Writeable" field.		

RegisterSave

Description: Used to save all the register values to non-volatile memory.			
Transmit: 1F	**Receive**: 1T	**Execution**: Queue	**Command Op-Code**: 1
DPWin Command Editor Syntax: RegisterSave			
MAXCLib Syntax: void RegisterSave (u8 * TxPacket, u8 NodeID)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	All registers that have write access. Each register in the register reference has a "Writeable" field.		

RegisterSend

Description: Used to send unsolicited packets to the host. Note: This command was designed for stored programs only. This command is useful for notifying the host that an event has occurred on the controller.			
Transmit:	**Receive**: 5T	**Execution**: Immediate	**Command Op-Code**: 2
DPWin Command Editor Syntax: [Register] Send			
MAXCLib Syntax: void RegisterSend (u8 * TxPacket, u8 NodeID, AS_REGISTER Register)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	All registers		

RegisterXorEqualRegister

Description: Used to XOR a value with register2 from a value in register1 and the result stored in register 1. Ex. Register1 = Register1 XOR Register2			
Transmit: 6F	**Receive**: 1T	**Execution**: Immediate	**Command Op-Code**: 31
DPWin Command Editor Syntax: [Register] ^= [Register]			
MAXCLib Syntax: void RegisterXorEqualRegister (u8 * TxPacket, u8 NodeID, AS_REGISTER Register1, AS_REGISTER Register2)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	All registers that have write access. Each register in the register reference has a "Writeable" field.		

RegisterXorEqualValue

Description: Used to XOR a value with a value in Register1, and the result stored in Register1. Ex. Register1 = Register1 XOR 0x0001			
Transmit: 5F	**Receive**: 1T	**Execution**: Immediate	**Command Op-Code**: 30
DPWin Command Editor Syntax: [Register] ^= <Value>			
MAXCLib Syntax: void RegisterXorEqualValue (u8 * TxPacket, u8 NodeID, AS_REGISTER Register, f64 Value)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	All registers that have write access. Each register in the register reference has a "Writeable" field.		

ResetCurrentLoop

Description: Resets the current loop feedback variables for one or more axes. This command should be issued when the current loop gains have been changed.			
Transmit: 18F	**Receive**: 1T	**Execution**: Queue	**Command Op-Code**: 85
DPWin Command Editor Syntax: Reset CurrentLoop [Axes]			
MAXCLib Syntax: void ResetCurrentLoop (u8 * TxPacket, u8 NodeID, u8 Axes)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	None		

ResetFaults

Description: Clears the faults Registers for the controller and/or axes.			
Transmit: 18F	**Receive**: 1T	**Execution**: Queue	**Command Op-Code**: 86
DPWin Command Editor Syntax: Reset Faults [ControllerAxes]			
MAXCLib Syntax: void ResetFaults (u8 * TxPacket, u8 NodeID, u8 ControllerAndOrAxes)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	None		

ResetOriginRegister

Description: Sets the axis or axes position (actual and setpoint) to the value stored at a specified Register.			
Transmit: 20F	**Receive**: 1T	**Execution**: Queue	**Command Op-Code**: 83
DPWin Command Editor Syntax: Reset Origin [Axis] [Register]			
MAXCLib Syntax: void ResetOriginRegister (u8 * TxPacket, u8 NodeID, u8 Axes, AS_REGISTER Register)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	None		

ResetOriginValue

Description: Sets the axis or axes position (actual and setpoint) to a specified value.			
Transmit: 19F	**Receive**: 1T	**Execution**: Queue	**Command Op-Code**: 82
DPWin Command Editor Syntax: Reset Origin [Axis] <Value>			
MAXCLib Syntax: void ResetOriginValue (u8 * TxPacket, u8 NodeID, u8 Axes, f64 Value)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	None		

ResetPositionLoop

Description: Resets the position loop filter state variables for one or more axes. This command should issued when any of the position loop filter coefficients have been changed.			
Transmit: 18F	**Receive**: 1T	**Execution**: Queue	**Command Op-Code**: 84
DPWin Command Editor Syntax: Reset PositionLoop [Axes]			
MAXCLib Syntax: void ResetPositionLoop (u8 * TxPacket, u8 NodeID, u8 Axes)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	None		

ResetSystem

Description: Forces the controller to reset.			
Transmit: 1F	**Receive**: 1T	**Execution**: Queue	**Command Op-Code**: 80
DPWin Command Editor Syntax: Reset System			
MAXCLib Syntax: void ResetSystem (u8 * TxPacket, u8 NodeID)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	None		

Return

Description: Used in the Command Editor to end a subroutine. When a stored program is executing this command, the instruction pointer is set to the address stored at the top of the call stack. Note: This command should be used with the Call command.			
Transmit: 1F	**Receive**: 1T	**Execution**: Immediate	**Command Op-Code**: 140
DPWin Command Editor Syntax: Return			
MAXCLib Syntax: void Return (u8 * TxPacket, u8 NodeID)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	none		

ServoDisable

Description: Used to turn off the servo loop to disable the motor. One or several axes can be disabled at once.			
Transmit: 18F	**Receive**: 1T	**Execution**: Queue	**Command Op-Code**: 49
DPWin Command Editor Syntax: Servo Disable [Axes]			
MAXCLib Syntax: void ServoDisable (u8 * TxPacket, u8 NodeID, u8 Axes)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	None		

ServoEnable

Description: Used to turn on the servo loop to enable the motor. The alignment method has to be specified before issuing this command. One or several axes can be enabled at once.			
Transmit: 18F	**Receive**: 1T	**Execution**: Queue	**Command Op-Code**: 48
DPWin Command Editor Syntax: Servo Enable [Axes]			
MAXCLib Syntax: void ServoEnable (u8 * TxPacket, u8 NodeID, u8 Axes)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	Alignment Type		

StopHard

Description: Used to stop a motion that is executing. Note. The hard stop aborts the running trajectory as fast as possible, therefore there is no deceleration parameter. The trajectory stops wherever it presently resides. One or more axes can be specified in this command.			
Transmit: 18F	**Receive**: 1T	**Execution**: Immediate	**Command Op-Code**: 57
DPWin Command Editor Syntax: Stop Hard [Axes]			
MAXCLib Syntax: void StopHard (u8 * TxPacket, u8 NodeID, u8 Axes)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	None		

StopSoft

Description: Used to stop a motion that is executing. Note. The soft stop recalculate a destination based on the deceleration value stored in soft stop register. One or more axes can be specified in the command.			
Transmit: 18F	**Receive**: 1T	**Execution**: Immediate	**Command Op-Code**: 58
DPWin Command Editor Syntax: Stop Soft [Axes]			
MAXCLib Syntax: void StopSoft (u8 * TxPacket, u8 NodeID, u8 Axes)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	Acceleration Negative Soft Stop		

ThenJump

Description: Used after a IF command to specify which address to jump to.			
Transmit: 3F	**Receive**: 1T	**Execution**: Immediate	**Command Op-Code**: 138
DPWin Command Editor Syntax: ThenJump [Label]			
MAXCLib Syntax: void ThenJump (u8 * TxPacket, u8 NodeID, f64 Address)			
Version Support	**Minimum**: 8.50	**Maximum**: 10.00	
Related Registers	none		

Index