

# PACMotion™ PMM345 MULTI-AXIS MOTION CONTROLLER

USER MANUAL

# Contents

<b>Section 1</b>	<b>Introduction .....</b>	<b>1</b>
1.1	Revisions in this Manual .....	1
1.2	PACSystems Documentation .....	2
1.2.1	PACSystems Manuals .....	2
1.2.2	PACMotion Manuals .....	2
1.2.3	RX3i Manuals .....	2
1.3	PACMotion Multi-Axis Motion Controller .....	3
1.3.1	Servo Types Supported .....	3
1.3.2	PMM345 Features .....	3
1.3.3	Performance to Improve Machine Productivity .....	4
1.3.4	Open and Integrated to Improve Engineering Productivity .....	5
1.3.5	Flexibility and Scalability .....	5
1.4	Fiber Terminal Block I/O .....	5
1.4.1	FTB Features .....	6
1.5	PACMotion Servo Drives .....	7
1.5.1	Emerson PACMotion Servo Motors .....	7
<b>Section 2</b>	<b>Getting Started .....</b>	<b>8</b>
2.1	Motion System Overview .....	9
2.1.1	PMM/RX3i Interface .....	10
2.1.2	Fiber Terminal Block I/O .....	11
2.1.3	Servo Drive and Machine Interfaces .....	11
2.1.4	Human-Machine Interfaces .....	12
2.2	Basic Installation .....	13
2.2.1	PMM Installation .....	13
2.2.2	FTB Installation .....	15
2.3	Basic Configuration .....	20
2.3.1	Connecting the Programmer to the RX3i .....	20
2.3.2	Adding a Motion Controller Module to the Hardware Configuration .....	21
2.3.3	Configuring PMM Settings .....	23
2.3.4	Configuring PMM I/O .....	24
2.3.5	Configuring Axis Parameters .....	25
2.3.6	Drive Type .....	26

- 2.3.7 Downloading the Configuration to the RX3i ..... 26
- 2.4 Basic Motion Example – Jogging an Axis ..... 28
- 2.5 Using the Synthetic Motor ..... 31
- 2.6 Axis Power-up and Feedback Device Initialization ..... 31
- 2.7 Basic Motion Example – Axis Initialization ..... 32
  - 2.7.1 Hardware Configuration (HWC)..... 32

**Section 3 I/O Wiring, Connections and LED Operation ..... 36**

- 3.1 PMM Faceplate I/O ..... 37
  - 3.1.1 PMM Faceplate Wiring Diagrams and Pin Assignments ..... 38
  - 3.1.2 PMM345 LED Operation ..... 40
- 3.2 Fiber Terminal Block I/O..... 43
  - 3.2.1 IC695FTB001 Fiber Terminal Block I/O Specifications..... 43
  - 3.2.2 Terminal Header and Cable Options ..... 44
  - 3.2.3 FTB Input Power ..... 45
  - 3.2.4 FTB Wiring Diagrams and Pin Assignment..... 46
  - 3.2.5 Typical External Differential Encoder Connection for FTB ..... 48
  - 3.2.6 Typical Single-Ended Encoder Connection for FTB ..... 49
  - 3.2.7 FTB LED Operation ..... 50
- 3.3 Errors Indicated by LEDs..... 51
  - 3.3.1 PMM LEDs ..... 51
  - 3.3.2 PMM EtherCAT LEDs ..... 52
  - 3.3.3 FTB LEDs..... 53
- 3.4 I/O Circuit Specifications..... 54
  - 3.4.1 PMM Faceplate I/O Circuits..... 54
  - 3.4.2 FTB I/O Circuits..... 57
- 3.5 EtherCAT Command Interface Cable..... 64
- 3.6 FTB to PMM Connection ..... 64
- 3.7 Grounding the PACMotion System ..... 65
  - 3.7.1 Fiber Terminal Block I/O Shield Ground Connection ..... 68
  - 3.7.2 I/O Cable Grounding..... 68

**Section 4 Configuration ..... 72**

- 4.1 Connecting the Programmer to the RX3i ..... 72
  - 4.1.1 PMM345 Programmer Connection ..... 72
- 4.2 Adding a PMM to the Hardware Configuration ..... 73
- 4.3 Configuring PMM Parameters ..... 75

- 4.3.1 Settings..... 77
- 4.3.2 PMM Status Data..... 79
- 4.3.3 I/O Function Assignments ..... 80
- 4.3.4 I/O Interrupts ..... 91
- 4.3.5 Axis Configuration Data..... 93
- 4.3.6 Advanced Parameters ..... 133
- 4.3.7 Power Consumption..... 133
- 4.3.8 Terminals ..... 134

**Section 5      PACMotion Function Block Operation ..... 135**

- 5.1 PACMotion Function and Function Block Types..... 136
- 5.2 Behavior of Motion Instructions ..... 139
  - 5.2.1 Instance Data ..... 139
  - 5.2.2 Immediate Response vs. Deferred Response Function Blocks ..... 140
  - 5.2.3 Administrative vs. Motion-Generating Functions and Function Blocks ... 140
  - 5.2.4 Function Block Triggering (Enabled vs. Executed Instructions)..... 141
- 5.3 Function and Function Block Parameters ..... 149
  - 5.3.1 Permissives, Constants, Variables or Flow used with Motion Function Blocks 149
  - 5.3.2 EN Input and ENO Output..... 149
  - 5.3.3 Input Parameters..... 150
  - Reference ID Variables..... 155
  - 5.3.4 Output Parameters ..... 157
  - 5.3.5 In\_Out Parameters ..... 159
- 5.4 Data Types and Structures ..... 160
  - 5.4.1 HWC Parameter Linked Data Types ..... 160
  - 5.4.2 Enumerated Data Types ..... 161
  - 5.4.3 CAM Profile Linked Data Types ..... 162
- 5.5 Axis States ..... 163
  - 5.5.1 Axis State Diagram ..... 165
- 5.6 Synchronized Motion..... 166

**Section 6      PACMotion Instruction Set Reference..... 167**

- 6.1 MC\_AbortTrigger ..... 167
  - Operands ..... 168
- 6.2 MC\_CamFileRead ..... 169
  - Operands ..... 170



6.3	MC_CamFileWrite .....	171
	Operands .....	172
6.4	MC_CamIn .....	173
	Operands .....	174
	6.4.1 Offset and Scaling .....	176
	6.4.2 Start Mode Mask .....	176
6.5	MC_CamOut .....	181
	Operands .....	182
6.6	MC_CamTableDeselect .....	183
	Operands .....	184
6.7	MC_CamTableSelect .....	185
	Operands .....	186
	6.7.1 Periodic (CAM Cycle Execution Mode) .....	187
6.8	MC_DelayedStart .....	188
	Operands .....	189
6.9	MC_DigitalCamSwitch.....	190
	Operands .....	191
	6.9.1 Requirements for Switch and Track Option Selections .....	192
	6.9.2 Enabling Outputs for DCS Control .....	192
6.10	MC_DL_Activate.....	202
	Operands .....	203
6.11	MC_DL_Configure .....	204
	Operands .....	205
6.12	MC_DL_Delete .....	209
	Operands .....	210
6.13	MC_DL_Get.....	211
	Operands .....	212
	6.13.1 Data Logging Example.....	213
6.14	MC_GearIn .....	223
	Operands .....	224
	6.14.1 MC_GearIn Example .....	226
6.15	MC_GearInPos.....	228
	Operands .....	229
6.16	MC_GearOut .....	233
	Operands .....	234
6.17	MC_Halt.....	235

Operands .....	236
6.18 MC_Home .....	237
6.18.1 Checklist for Find Home Sequence .....	237
6.18.2 Overview of Find Home Sequence .....	238
6.18.3 Homing Modes.....	241
6.19 MC_JogAxis .....	250
Operands .....	251
6.19.1 Example: Jog to Software End of Travel with Warning .....	252
6.20 MC_LibraryStatus .....	257
Operands .....	258
6.21 MC_ModuleReset .....	259
Operands .....	260
6.22 MC_MoveAbsolute .....	261
Operands .....	262
6.22.1 MC_Move Absolute Example .....	264
6.22.2 MC_MoveAbsolute Example with Dwell Operation .....	265
6.23 MC_MoveAdditive .....	266
Operands .....	267
6.23.1 Example .....	268
6.24 MC_MoveRelative .....	270
Operands .....	271
6.25 MC_MoveSuperimposed .....	272
Operands .....	273
6.26 MC_MoveVelocity .....	274
Operands .....	275
6.27 MC_Phasing .....	276
Operands .....	277
6.28 MC_Power.....	278
Operands .....	279
6.28.1 MC_Power Input Combinations.....	280
6.29 MC_ReadActualPosition .....	281
Operands .....	282
6.30 MC_ReadActualVelocity .....	283
Operands .....	284
6.31 MC_ReadAnalogInput .....	285
Operands .....	286

6.31.1 Input Example .....	287
6.32 MC_ReadAnalogOutput .....	288
Operands .....	289
6.33 MC_ReadAxisError .....	290
Operands .....	291
6.34 MC_ReadBoolParameter .....	292
Operands .....	293
6.35 MC_ReadBoolParameters .....	294
Operands .....	295
6.36 MC_ReadDigitalInput .....	296
Operands .....	297
6.36.1 Example .....	297
6.37 MC_ReadDigitalOutput .....	299
Operands .....	300
6.38 MC_ReadDwordParameters .....	301
Operands .....	302
6.39 MC_ReadEventQueue.....	303
6.39.1 Example .....	305
6.39.2 Drive Faults .....	307
6.39.3 Drive Warnings.....	309
6.40 MC_ReadParameter .....	311
Operands .....	312
6.40.1 Example .....	312
6.41 MC_ReadParameters .....	313
Operands .....	314
6.41.1 Example .....	315
6.42 MC_ReadStatus .....	316
Operands .....	317
6.42.1 Axis Status Flags .....	318
6.43 MC_ReadTorqueCommand .....	321
Operands .....	322
6.44 MC_Reset.....	323
Operands .....	324
6.45 MC_SetOverride .....	325
Operands .....	327
6.46 MC_SetPosition .....	328

6.46.1 Synchronous State Operation .....	329
Operands .....	329
6.47 MC_Stop .....	331
Operands .....	332
6.48 MC_SyncStart.....	333
Operands .....	334
6.49 MC_TouchProbe.....	335
Operands .....	336
6.49.1 Pre-trigger Replaces Window Only on PSD Drive Family .....	337
6.49.2 Examples.....	337
6.50 MC_WriteAnalogOutput .....	342
Operands .....	343
6.50.1 Example .....	344
6.51 MC_WriteBoolParameter.....	345
Operands .....	346
6.52 MC_WriteBoolParameters .....	347
Operands .....	348
6.53 MC_WriteDigitalOutput .....	349
Operands .....	350
6.53.1 Example .....	351
6.54 MC_WriteDwordParameters .....	352
Operands .....	353
6.55 MC_WriteParameter.....	354
Operands .....	355
6.56 MC_WriteParameters .....	356
Operands .....	357

## **Section 7 Electronic CAM Programming ..... 358**

7.1 Overview of PACMotion CAM Profile Development .....	359
7.1.1 Point Limits in CAM Profiles .....	360
7.2 CAM Types and Modes for the PMM.....	361
7.2.1 CAM Profile Types .....	362
7.3 CAM Operation Restrictions by Type and Mode .....	364
7.4 Smoothing and Curve Fitting .....	366
7.5 Calculating Slave Axis Velocity and Acceleration .....	367
7.6 Synchronized Motion Function Block Status.....	370
7.6.1 Pending.....	370

7.6.2	Ramping .....	373
7.6.3	InSync .....	374
7.7	CSV CAM File Format .....	378
7.7.1	File Header Format .....	379
7.8	Reference Memory Format for CAM Files .....	380
7.8.1	Parameters of type LREAL 8 bytes .....	380
7.8.2	Boolean Parameters .....	381
<b>Section 8</b>	<b>Parameters for Monitoring and Control .....</b>	<b>382</b>
8.1	Axis Parameter Numbers .....	383
8.1.1	Position Loop Overview .....	383
8.1.2	Axis Parameter Number Index .....	384
8.2	Module Parameter Numbers .....	399
8.3	I/O Data Reference Numbers .....	406
<b>Section 9</b>	<b>Diagnostics .....</b>	<b>409</b>
9.1	PMM Error IDs .....	410
9.1.1	Accessing Error IDs .....	410
9.1.2	Error ID Format .....	410
9.1.3	Clearing PMM Errors .....	411
9.1.4	I/O Fault Table .....	411
9.1.5	Error ID Reference .....	412
9.2	CPU Error Codes .....	459
9.3	Interpreting Drive Faults and Warnings .....	465
9.3.1	Drive Faults .....	465
9.3.2	Drive Warnings .....	467
9.4	PMM Event Queue .....	468
9.4.1	Event Queue Details .....	468
9.5	Accessing the Ten Most Recent Events .....	469
9.5.1	Parameter Errors Caused by Changes in Axis Scaling .....	471
9.6	Diagnostic Logic Blocks .....	474
9.6.1	Using DLBs with PACMotion CAM Profiles .....	474
	<b>Appendix A: Touch Probe and Digital CAM Switch Accuracy Calculations</b>	<b>476</b>
<b>A-1</b>	<b>Touch Probe Accuracy .....</b>	<b>477</b>

<b>A-1.1</b>	<b>External Quadrature Encoder .....</b>	<b>477</b>
<b>A-1.2</b>	<b>Motor Encoder .....</b>	<b>477</b>
<b>A-2</b>	<b>Digital CAM Switch Accuracy .....</b>	<b>478</b>
<b>A-2.1</b>	<b>DCS Accuracy at Constant Velocity .....</b>	<b>478</b>
<b>A-2.2</b>	<b>DCS Accuracy During Acceleration .....</b>	<b>479</b>
	DCS Extrapolation (Look Ahead) Error Calculations .....	479
	DCS Calculated Position Change and Position Error Terms .....	480
	DCS Acceleration Error Examples .....	481
	Using DCS Acceleration Error Formulas .....	482
	<b>Appendix B: Position Feedback Devices .....</b>	<b>483</b>
<b>B-1</b>	<b>Digital Serial Encoders .....</b>	<b>483</b>
<b>B-1.1</b>	<b>Digital Serial Encoder Modes.....</b>	<b>483</b>
	Incremental Encoder Mode Considerations.....	483
	Absolute Encoder Mode Considerations.....	483
	Absolute Encoder Mode - Position Initialization.....	483
	Find Home Cycle - Absolute Encoder Mode .....	484
	Set Position Command - Absolute Encoder Mode .....	484
	Absolute Encoder Mode - PMM Power-Up .....	484
<b>B-2</b>	<b>External Quadrature Encoders .....</b>	<b>485</b>
<b>B-2.1</b>	<b>Example: Connecting an External Encoder to Axis 5485</b>	
	Step 1: Configuration .....	485
	Step 2: Wiring.....	487
	Step 3: Basic Checking .....	487
	Step 4: Programming a Gear .....	488
	<b>Appendix C: Tuning Digital and Analog Servo Systems.....</b>	<b>489</b>
<b>C-1</b>	<b>Validating Axis Direction, Over Travel Switch and Home Switch Inputs .....</b>	<b>490</b>

- C-1.1 Validating Motor Direction ..... 490**
- C-1.2 If an external quadrature encoder feedback device will be used for the motor ..... 490**
- C-1.3 Validating the Over Travel Limit Switch Inputs ... 491**
- C-1.4 Validating the Home Switch Input ..... 491**
  - Follow these steps to verify the Home Switch wiring and operation: ..... 491
- C-1.5 PM EtherCAT Servo System Start-up Diagnostics 492**
- C-2 Forcing Servo Velocity ..... 493**
- C-3 Tuning a PM EtherCAT Servo ..... 493**
- C-3.1 Tuning Requirements ..... 494**
  - Control Loops Block Diagram..... 494
  - Tuning the Velocity Loop ..... 494
  - Tuning the Position Loop ..... 495
  - Preliminary Position Loop Settings for Tuning Session..... 495
  - Setting the Position Loop Gain ..... 495
  - Position Loop Proportional Gain Method 1 ..... 495
  - Terminology..... 495
  - Position Loop Proportional Gain Method - Method 2 ..... 496
  - Optimizing Velocity Feedforward ..... 497
- C-4 Tuning an Analog, Velocity-Controlled Drive..... 498**
- C-4.1 Wiring and Configuration ..... 498**
- C-4.2 Validating Axis Configuration and Servo Drive Settings 500**
- C-5 Tuning an Analog, Torque-Controlled Drive ..... 501**
- C-5.1 Wiring and Configuration ..... 501**
- C-5.2 Verifying Basic Analog Control Functions ..... 503**

<b>C-5.3</b>	<b>Tuning the Torque Mode Velocity Loop .....</b>	<b>504</b>
	Tuning Requirements .....	504
	Analog Mode Torque Interface Control Loops Block Diagram.....	504
	Sending a Velocity Command to the Velocity Loop .....	504
	Method #1 .....	504
	Method #2.....	505
	Tuning the Velocity Regulator .....	505
	Equation 2 .....	506
	Sample Velocity Loop Tuning Session .....	508
<b>C-5.4</b>	<b>Position Loop Tuning .....</b>	<b>523</b>
<b>C-5.5</b>	<b>Advanced Analog Servo (Torque Mode) Tuning..</b>	<b>524</b>
	Digital Servo Parameters .....	524
	Velocity Loop Gain Parameters PK1V, PK2V, PK3V .....	524
	Frequency Ranges for PK1V-PK3V .....	524
	Integral Gain Decay Time Constants .....	525
	Torque Command Filter.....	525
	TCMD Filter Cutoff Frequencies .....	526
	General Contact Information .....	527
	Technical Support.....	527



## Warnings and Caution Notes as Used in this Publication

### **WARNING**

Warning notices are used in this publication to emphasize that hazardous voltages, currents, temperatures, or other conditions that could cause personal injury exist in this equipment or may be associated with its use.

In situations where inattention could cause either personal injury or damage to equipment, a Warning notice is used.

---

### **CAUTION**

Caution notices are used where equipment might be damaged if care is not taken.

---

**Note:** Notes merely call attention to information that is especially significant to understanding and operating the equipment.

These instructions do not purport to cover all details or variations in equipment, nor to provide for every possible contingency to be met during installation, operation, and maintenance. The information is supplied for informational purposes only, and Emerson makes no warranty as to the accuracy of the information included herein. Changes, modifications, and/or improvements to equipment and specifications are made periodically and these changes may or may not be reflected herein. It is understood that Emerson may make changes, modifications, or improvements to the equipment referenced herein or to the document itself at any time. This document is intended for trained personnel familiar with the Emerson products referenced herein.

Emerson may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not provide any license whatsoever to any of these patents.

Emerson provides the following document and the information included therein as-is and without warranty of any kind, expressed or implied, including but not limited to any implied statutory warranty of merchantability or fitness for particular purpose.

# Section 1 Introduction

The PACMotion™ Multi-axis Motion Controller (PMM) is a high performance, easy-to-use servo motion control module that closely integrates with the PACSystems RX3i CPU's logic solve and communications functions. This versatile motion controller combines the benefits of highly integrated motion and machine logic with the performance, flexibility and scalability required for advanced machine automation. The open programming environment simplifies motion and machine logic synchronization without sacrificing real time performance required for high-speed motion applications.

The built-in PMM faceplate I/O supports applications with lower I/O counts, while the optional Fiber Terminal Block I/O (FTB) supports larger applications with extensive distributed user-configurable digital and analog I/O.

This chapter provides an overview of PMM and FTB features.

## 1.1 Revisions in this Manual

Rev	Date	Description
A	Sep 2020	Initial Release

## 1.2 PACSystems Documentation

### 1.2.1 PACSystems Manuals

Description of Manual	GFK Number
PACSystems RX3i and RSTi-EP CPU Reference Manual	GFK-2222
PACSystems RX3i and RSTi-EP CPU Programmer's Reference Manual	GFK-2950
PACSystems RX3i and RSTi-EP TCP/IP Ethernet Communications User Manual	GFK-2224
PACSystems TCP/IP Ethernet Communications Station Manager User Manual	GFK-2225
C Programmer's Toolkit for PACSystems	GFK-2259
PACSystems Memory Xchange Modules User's Manual	GFK-2300
PACSystems Hot Standby CPU Redundancy User Manual	GFK-2308
PACSystems Battery and Energy Pack Manual	GFK-2741
PAC Machine Edition Logic Developer Getting Started	GFK-1918
PAC Process Systems Getting Started Guide	GFK-2487
PACSystems RXi, RX3i, and RSTi-EP Controller Secure Deployment Guide	GFK-2830
PACSystems RX3i & RSTi-EP PROFINET I/O Controller Manual	GFK-2571

### 1.2.2 PACMotion Manuals

Description of Manual	GFK Number
PACMotion PMM335 to PMM345 Migration Guide	GFK-3135
PACMotion Multi-Axis Motion Controller PMM345 User Manual	GFK-3140
PACMotion PSD Installation and User Manual	GFK-3168
PACMotion PSR Installation and User Manual	GFK-3169
PACMotion EtherCAT Communications Manual	GFK-3170
PACMotion Servo Drive IMR	GFK-3171
PACMotion PSD Connectivity Guide	GFK-3172
PACMotion PSD NA Accessories Guide	GFK-3173
PACMotion PSR Motor Transport Instructions Manual	GFK-3174
PACMotion Servo Motor IMR	GFK-3175

### 1.2.3 RX3i Manuals

Description of Manual	GFK Number
PACSystems RX3i System Manual	GFK-2314
DSM324i Motion Controller for PACSystems RX3i and Series 90-30 User's Manual	GFK-2347
PACSystems RX3i PROFIBUS Modules User's Manual	GFK-2301
PACSystems RX3i Max-On Hot Standby Redundancy User's Manual	GFK-2409
PACSystems RX3i Ethernet Network Interface Unit User's Manual	GFK-2439
PACMotion Multi-Axis Motion Controller User's Manual (PMM335)	GFK-2448

In addition to these manuals, datasheets and product update documents describe individual modules and product revisions. The most recent PACSystems documentation is available on the Emerson website <https://www.emerson.com/Industrial-Automation-Controls/support>.

Additional information is available at the PLCOpen web site: <http://www.plcopen.org/>

## 1.3 PACMotion Multi-Axis Motion Controller

The PACMotion Multi-Axis Motion Controller (IC695PMM345) is designed with the performance to deliver improved machine productivity required for today's high-speed machines and lean manufacturing environments. Each module can control up to four servo axes. Up to 40 axes can be controlled from a single RX3i backplane.

The PMM345 supports discrete and synchronous motion control yielding a flexible motion controller that scales to fit your requirements.

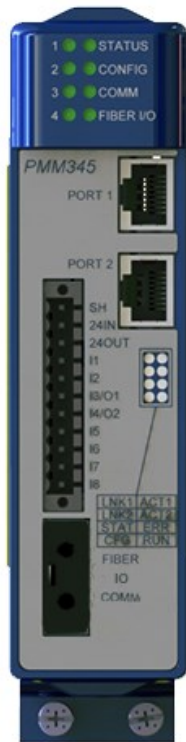
### 1.3.1 Servo Types Supported

- Digital: Supports Emerson EtherCAT based servo drives and motors PACMotion Servo Drives (PSD). For a list of supported Emerson motors, refer to 2.3.6 Drive Type the Emerson motor manual shown in section 1.2.2.
- Analog: Provides analog velocity and analog torque command interfaces to third-party analog servos.

### 1.3.2 PMM345 Features

- Fast motion path (1ms) planning and position update rates (500 $\mu$ s) deliver improved accuracy and faster response to changing control requirements.
- Consistent motion update rate regardless of the number of axes.
- High reliability Emerson servos improve machine uptime.
- High-speed synchronization of up to 40 axes over the PACSystems RX3i backplane.
- Advanced CAM and gearing features for electronic line shaft applications.
- Single software development environment for complete automation control solution simplifies programming.
- Distributed architecture for greater machine flexibility..
- Optional Fiber Terminal Block allows distributed motion-centric I/O to reduce wiring complexity and cost.
- Two high-speed position capture inputs per axis for registration and sequence control.

Figure 1:PMM Faceplate



### 1.3.3 Performance to Improve Machine Productivity

- Real-time synchronization of up to 40 axes.
- Three high-speed time-based or event-driven interrupts enable fast, deterministic event response and synchronization.
- Demand-driven data exchange between the PACSystems RX3i CPU and PACMotion modules reduces scan time impact.
- Digital CAM Switch (PLS) function with multi-track high speed outputs with microsecond-level response.
- All EtherCAT Networks and PMMs synchronized to a common clock yielding highly synchronized motion control across the motion control system.

## 1.3.4 Open and Integrated to Improve Engineering Productivity

- Single software development environment with shared tag database for logic, motion, I/O and operator interface.
- Motion and machine logic in a common program greatly simplifies programming.
- Motion function blocks and state model designed to comply with the PLCopen programming standard to reduce learning curve and training costs.
- Buffer mode allows program logic to queue motion command sequences and specify or change the velocity transition between buffered moves on-the-fly.
- Advanced diagnostic tools included in PACSystems software speed diagnostics and machine time-to-market.

## 1.3.5 Flexibility and Scalability

- Four servo axes per module; Up to 40 axes in a single PACSystems RX3i rack.
- Built-in faceplate I/O and optional fiber I/O terminal block supports extensive user configurable digital and analog I/O.
- Servo Drives and motion I/O can be physically distributed. Virtual (time-based) or real (encoder) master axes over the backplane support advanced CAM and electronic gearing applications for flexible electronic line shaft applications.

## 1.4 Fiber Terminal Block I/O

The DIN rail mounted FTB, IC695FTB001, allows you to connect 5-volt, 24-volt, and analog I/O to motion specific devices, such as limit switches and encoders, over a full-duplex fiber optic link up to 100m from the PMM345. FTB I/O can be configured as overtravel switch, home switch, quadrature encoder and high-speed touch probe position capture inputs, digital CAM switch outputs or general purpose I/O. The FTB's analog outputs can be used as general-purpose analog outputs or configured for closed position loop (velocity interface) or velocity loop (torque interface) control of up to two analog servos.

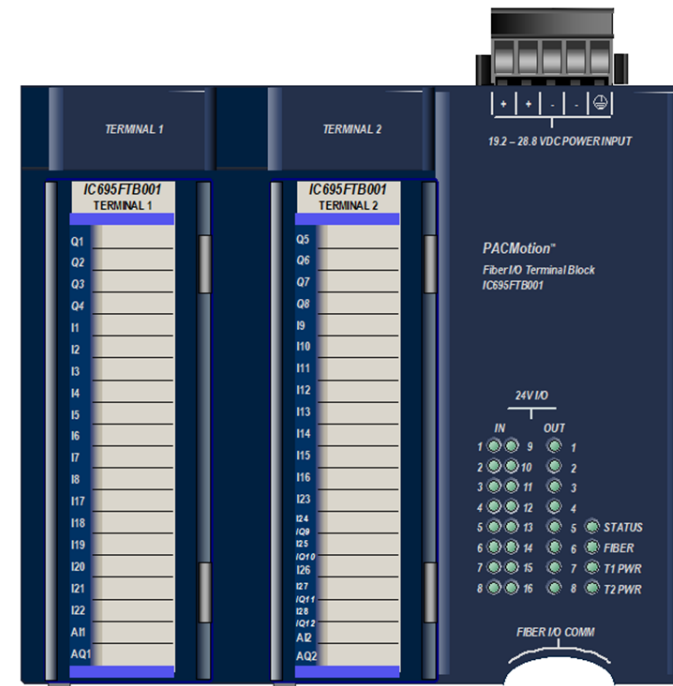
A robust serial protocol encodes and decodes the data as it sent between the PMM and the FTB. In the event of a system malfunction, such as loss of communication with the PMM, the FTB sets its I/O to the user-configured state.

**Note:** For order info, please consult the FTB IPI, GFK-2467.

## 1.4.1 FTB Features

- Fiber optic connection allows choice of cable lengths from 1m to 100m
- Fiber optic connection provides optical isolation between the main rack and the FTB.
- Current capability to 0.5A on all 24Vdc outputs.
- Loss of encoder and open wire fault detection on 5Vdc differential inputs (quadrature encoder lines).
- Visual diagnostics provided via individual LEDs that indicate I/O point state.
- DIN rail mounting allows convenient location of I/O.
- Fiber optic interface reduces remote I/O wiring cost and improves noise immunity.
- Removable RX3i terminal block headers provide ease of use
- A 5Vdc power source for external quadrature encoders.

Figure 2: Fiber Terminal Block I/O (FTB)



## 1.5 PACMotion Servo Drives

The PMM345 supports PACMotion Servo Drives and Motors. For a list of servo and servo drive models, refer to the servo manuals shown in section 1.2.2

The Emerson servos include:

- World leading reliability
- Low maintenance, no component drift, no commutator brushes
- All parameters digitally set; no re-tuning required
- Absolute encoder eliminates re-homing
- 24Vdc motor holding brake (optional)

The servo drives are ideally suited for motion control applications. In the PACMotion servo system, all control loops - current, velocity, and position - are closed in the motion controller or the servo drive. This approach reduces setup time and delivers significant throughput advantages even in the most challenging applications.

The servo drives are less costly to integrate and maintain. Control circuits are unaffected by temperature changes. The servos have a broad application range including a wide load inertia range, flexible acceleration/deceleration and position feedback configurations. PACMotion interfaces with the servo drives through the EtherCAT protocol. The EtherCAT protocol is a high-performance synchronous protocol that is ideal for motion control applications.

### 1.5.1 Emerson PACMotion Servo Motors

The Emerson Series servos offer a broad product offering that is designed to work seamlessly with the Emerson Servo Drives. This includes the ability for the motor to be automatically identified and configured when connected to the Servo Drive. The motors are further characterized by class leading torque to inertia ratios.

The Emerson series Servo Drives are compact, highly efficient servo drives designed to conserve panel space. Easy maintenance, less cabling, and quick power line connections reduce installation cost and downtime during servo drive replacement.

- World class Autotuning functions
- Extensive debug and commissioning tools
- High-resolution serial encoder, 1048576 counts per revolution, for smooth speed control and high position accuracy.
- Optional 24Vdc motor holding brake.
- Absolute feedback is available based on Encoder selection.
- Single Power/Encoder cabling available as an option



## Section 2 Getting Started

This chapter provides a system overview with the basic steps required to install, start up, and configure the PACMotion Multi-Axis Motion Controller (PMM) as well as the optional Fiber Terminal Block I/O (FTB).

Topics covered:

Section 2.1 Motion System Overview

Section 2.2 Basic Installation

Section 2.3 Basic Configuration

Section 2.4 Basic Motion Example – Jogging an Axis

Section 2.5 Using the Synthetic Motor

Section 2.6 Axis Power-up and Feedback Device Initialization

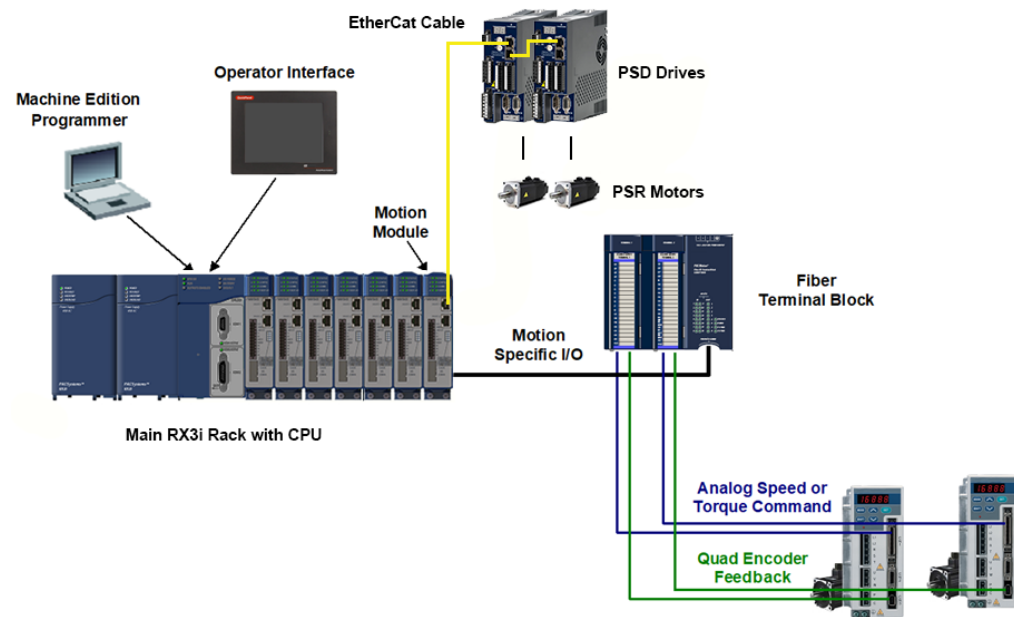
Note: The PMM345 requires version 9.9 or later of the Logic Developer programming and configuration software. Logic Developer is used for the following tasks.

- Configuration. Select module settings and default operational parameters.
- Machine/Application program creation.

## 2.1 Motion System Overview

A typical PACMotion system includes the PMM motion controller, a PACSystems RX3i controller, PAC Machine Edition Programmer, motor(s), servo drive(s), I/O, and a Human Machine Interface (HMI). The following figure shows a system with four servos. This section briefly discusses each element involved in system operation.

**Figure 3: Example PACMotion System**



The PACSystems RX3i PAC controller hosts the PMM and is programmed and configured using PAC Machine Edition -Logic Developer software. Logic Developer provides a powerful and flexible environment to develop control applications, including motion applications, as well as graphic HMI platforms to work alongside the application.

System hardware configuration, machine logic programming, motion programming, communications and operator interface programming are highly integrated to simplify application software development. A common tag database makes it easy to tie machine control and motion variables to your operator interface screens.

The PMM's faceplate and FTB I/O can be configured as touch probes, hardware overtravel limits, home switch inputs, incremental master encoder inputs, digital CAM switches, or general purpose I/O. For information on motion system and I/O connections to the PMM, refer to Chapter 3, I/O Wiring, Connections and LED Operation.

## 2.1.1 PMM/RX3i Interface

The PMM345 and PACSystems RX3i CPU operate together as one integrated motion control package. The PMM345 communicates with the CPU and other PMM345 modules via the RX3i high-speed PCI-based backplane bus. The RX3i CPU and PMM345 communications structures yield a tightly coupled system, providing the user the flexibility necessary for high-speed/high-performance applications. Facilitating this tight integration, the RX3i CPU and PMM345 support multi-scan (interrupt-driven and time-driven) capability and demand-driven communications.

Multi-scan provides a flexible way to partition the application to achieve the highest performance. The base CPU multi-scan capabilities are enhanced by the PMM's ability to generate interrupts and/or periodic signals to the RX3i CPU to cause application logic to be run.

Demand-driven data exchange causes the function block commands to be sent at the instant they are activated and data is returned as quickly as available within the logic scan, which can be less than a single CPU sweep. This allows the logic program to control the rate and amount of data collected from the motion module.

For information about the operation of the RX3i CPU sweep refer to the PACSystems RX3i and RSTi-EP CPU Reference Manual, GFK-2222.

Note that when the CPU transitions from Run mode to Stop mode, any motion that is occurring will stop. In Stop mode, user application logic is no longer executing, which means that the application cannot control motion. In addition, a transition from Run with IO Enabled mode to Run with IO Disabled will stop motion. If your application is in Stop with IO Enabled mode or Run with IO Disabled mode, you can still execute motion logic and control axis motion using a Diagnostic Logic Block (DLB).

---

**Note:** *The PMM motion controller requires a rack-mounted RX3i CPU module, such as CPU310, CPU315, CPU320, CPE330 or higher. The RX3i CPE400, NIU, CMM and CRU modules do not support the PMM, nor does CPE330 when configured in Redundant mode.*

---

## 2.1.2 Fiber Terminal Block I/O

The DIN rail mounted FTB extends the faceplate I/O and allows motion-centric I/O to be distributed up to 100m from the RX3i rack. The FTB contains an external I/O controller that provides a broad range of configurable I/O for the axes controlled by the PMM module.

The FTB provides 5-volt, 24-volt, and analog I/O via two high-density plug-on I/O terminal headers. The FTB I/O points can be configured to communicate the following types of control and status data through a noise immune fiber optic serial link:

- Quadrature Encoder Data: Information from incremental quadrature encoders
- Touch Probe Data: State of any touch probe sensor inputs
- Home Switch: State information from the home switch
- Overtravel Switches: State information for hardware overtravel switches
- Digital CAM Switch function: Precise position/time-driven outputs
- Analog servo control functions: Analog drive status, enable and reset.

Fault detection for the 5Vdc differential inputs includes open wire and loss of encoder power detection. The 24Vdc outputs have open load fault detection. The 5-volt and 24-volt outputs have electronic short circuit protection (ESCP).

Because the 5Vdc differential inputs provide loss of encoder power detection as well as open wire detection, an FTB is recommended for applications that use an external encoder. For more information, please see section 2.1.2, Fiber Terminal Block I/O.

## 2.1.3 Servo Drive and Machine Interfaces

For digital servos, the servo drive interface is made through an Ethernet connector, which provides an EtherCAT interface to the servo drive. This interface carries all control and feedback signals necessary for the PMM to control the axis. Additionally, it provides access to drive based I/O for function including homing and touch probes.

### PMM I/O

A faceplate connector and an optional Fiber Terminal Block I/O (FTB) provide PMM I/O connections. For information about I/O connections to the PMM and FTB, refer to Section 3, I/O Wiring, Connections and LED Operation.

## FTB/PMM Communications

An FTB identifier can be assigned to each PMM/FTB pair. The identifier transfer allows the PMM and FTB to determine when communication is established and whether they are communicating to the expected device. If an identification mismatch occurs, the FIBER LEDs on both devices blink alternately green and yellow. Communications are terminated until the PMM/FTB pair is connected correctly. The PMM/FTB ID pairing allows users to easily setup/debug applications with multiple FTBs where it is important to assure that a particular FTB is connected to a specific module.

FTB identification options are controlled by hardware configuration settings. For details, refer to Section 4, Configuration.

The identifier is stored in the FTB in non-volatile memory. If it becomes necessary to reset the FTB identification so that it will communicate with any PMM, set the identification mode to User Defined and the identifier to 0. You can also use an MC\_WriteDwordParameters function block to write a 0 identifier to the FTB.

If fiber link communications are lost, FTB outputs go to the user-configured state.

### 2.1.4 Human-Machine Interfaces

Human-machine interfaces (HMI) provide a way for the operator to control and monitor the servo system through a control panel or CRT display. These interfaces communicate with the RX3i controller through discrete I/O modules or an intelligent serial communications or network communications module. Additionally, there is the ability to use the PME View application to generate HMI screens that can run on the computer that is being used to program and configure the modules.

HMIs, or other devices that are capable of reading and writing values in the RX3i reference memory, can modify CAM profiles. The MC\_CamFileRead, MC\_CamFileWrite instructions allow you to move the CAM profiles between the RX3i file system and reference memory. CAM files located in reference memory are accessible to the HMI. For details on the use of these instructions, refer to Section 6, PACMotion Instruction Set Reference.

## 2.2 Basic Installation

The PMM must be located in a PACSystems RX3i Universal Backplane (IC695CHS007, IC695CHS012 or IC695CHS016). It cannot be located in an expansion or remote backplane.

The FTB is mounted on a DIN rail and can be located a maximum distance of 100m from the PMM. Fiber optic cables in various standard lengths are available from Emerson.

### 2.2.1 PMM Installation

#### Maximum Number of PACMotion Controllers per RX3i System

The number of motion modules that can be placed in an RX3i rack is determined by the size of the backplane and the available power. The sample configuration below contains 10 PMM modules, which is the maximum number per rack and provides 40 real axes and 10 virtual axes.

The maximum number of axes that can be configured in a system does not include virtual axes.

---

**Note:** *If used, an external encoder typically requires 500 mA (2.5 watts) at +5Vdc, which is supplied by the FTB. For encoders with higher current requirements, a separate user-supplied power supply is required.*

---

#### Sample Configuration

Multifunctional Power Supplies	Qty	+5Vdc Power per Power Supply	+3.3Vdc Power per Power Supply	Total +5Vdc Power Supplied	Total +3.3Vdc Power Supplied	No. of Slots Used	No. of Real Axes
IC695PSA140	2	6A (30W)	9A (29.7W)	12A (30W)	18A (59.4W)	4	

Modules	Qty	+5Vdc Power Required	+3.3Vdc Power Required	Total +5Vdc Power Used	Total +3.3Vdc Power Used	No. of Slots Used	
IC695CPU310	1	1A (5W)	1.25A (4.125W)	1A (5W)	1.25A (4.125W)	2	
IC695PMM345	10	0.45A (2.25W)	1.1A (3.63W)	4.5A (22.5W)	11.00 (36.3W)	10	40
Total				5.5A (27.5W)	12.25 (40.425)	15	40

## Installing the PMM

---

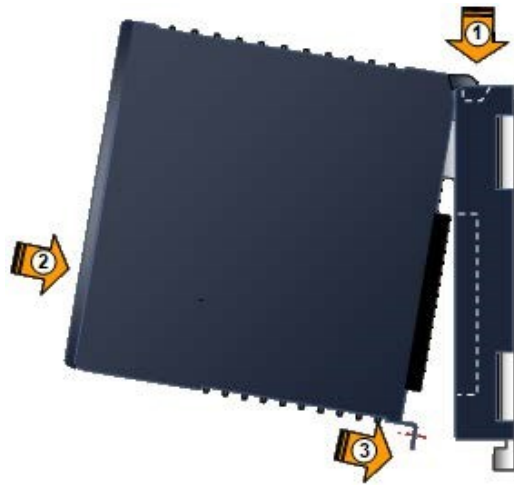
**Note:** *The PMM does not support hot insertion and removal. For details on this and other installation and environmental considerations for PACSystems RX3i components, refer to the PACSystems RX3i System Manual, GFK 2314.*

---

1. Use the configuration software to stop the RX3i. This prevents the local application program, if any, from initiating a command that may affect the module operation on subsequent power-up.
2. Be sure the module catalog number matches the intended slot configuration.
3. Holding the module firmly, align the module with the correct slot and connector.
4. Engage the module's rear pivot hook(s) in the notch on the top of the backplane (Arrow 1, Figure 4).
5. Swing the module down (Arrow 2, Figure 4) until the module's connector engages the backplane connector.
6. Visually inspect the module to be sure it is properly seated.
7. Secure the bottom of the module to the backplane using the machine screws provided with the module (Arrow 3, Figure 4).
8. Connect faceplate I/O wiring. For detailed wiring information, refer to Section 3, I/O Wiring, Connections and LED Operation.

---

**Figure 4:Latch Module into Rack**



- 
9. If an FTB is to be used with the PMM, install it as described in Section 2.2.2, below.
  10. Repeat this procedure for each PMM in your RX3i system.

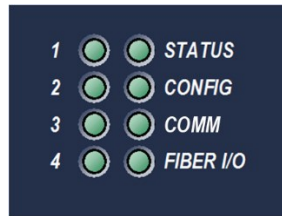
---

**Note:** *When the PMM initially powers up, the module STATUS LED on the module should be on and green (Figure 5).*

---

Configure the PMM(s) and FTB(s) as described in Section 2.3 Basic Configuration.

Figure 5: Status LED following Power-up

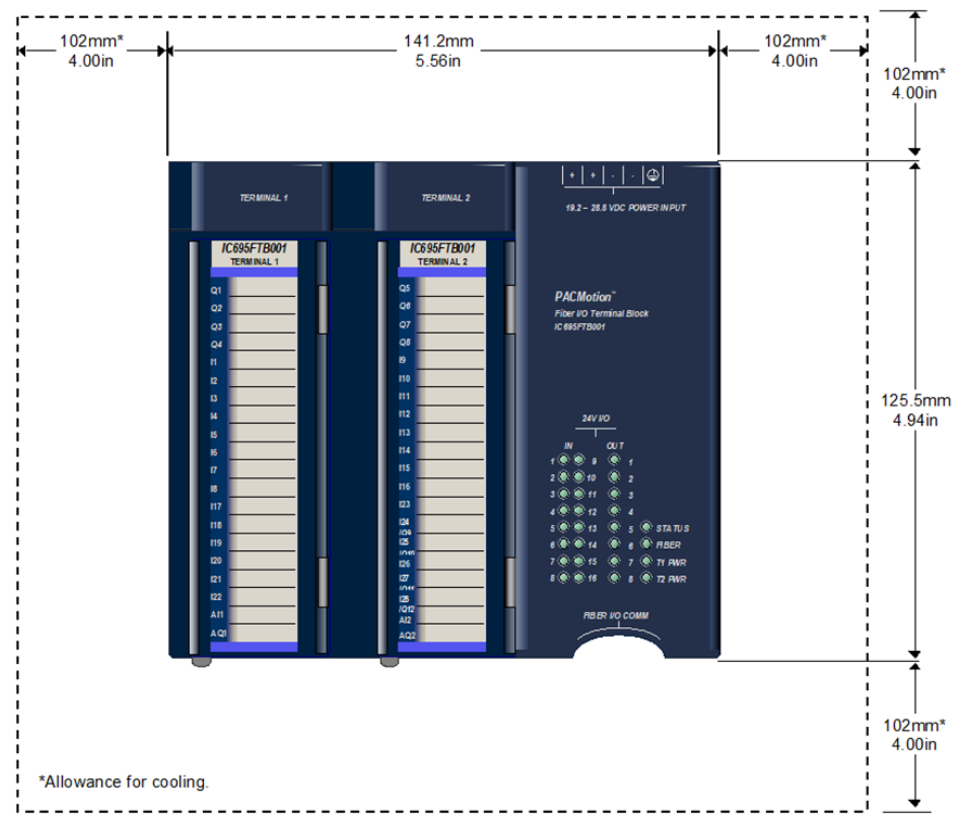


## 2.2.2 FTB Installation

The FTB can be located up to 100m (328.08 ft.) from the PMM and must be mounted on a standard 35 mm DIN rail. For proper cooling, the FTB must be mounted on a vertical surface. Do not mount it on a horizontal surface.

### FTB Mounting Dimensions

Figure 6: FTB Mounting Dimensions & Clearances



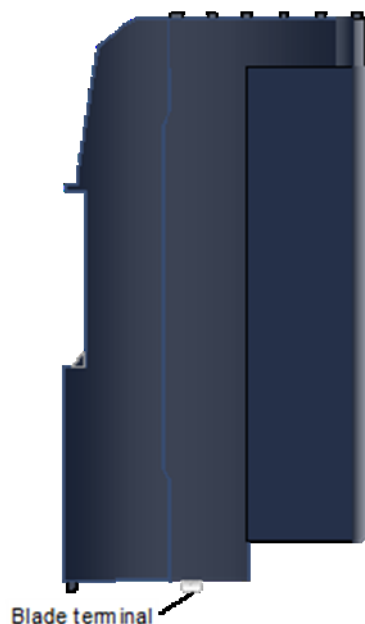
For RX3i environmental requirements, refer to the PACSystems RX3i System Manual, GFK-2314.





---

**Figure 8: Connect FTB Shield Ground to Frame Ground**



---

**⚠ WARNING**

Field power must be turned off when installing or removing a Terminal Block assembly.

- 
4. Install the FTB terminal block assemblies.
    - a. Complete the FTB I/O wiring and secure the wire bundles to the tie-downs on the bottom of the terminal block. (Refer to Section 3, I/O Wiring, Connections and LED Operation for detailed wiring information.)
    - b. Align the top of the terminal block with the bottom of the cover, making sure that the notches in the terminal block match up with the grooves in the cover.
    - c. Slide the terminal block upward until it clicks into place.

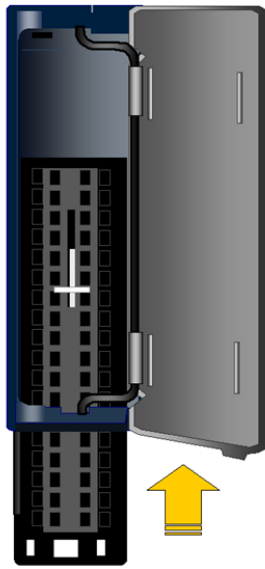
---

**Note:** Terminal blocks must be ordered separately from the FTB.

---

---

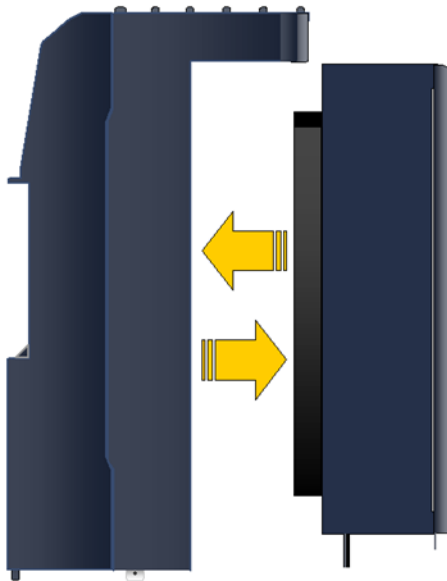
**Figure 9:Slide FTB Terminal Block into Place**



- 
5. Install the terminal blocks on the FTB
    - a. Press the terminal block release lever down.
    - b. Press the terminal block assembly straight toward the FTB until it is partially seated.
    - c. Open the door on the front of the terminal block and push the terminal block release lever up very firmly until it reaches the top of the slot and clicks into place.
    - d. Check to be sure the terminal block is fully seated.

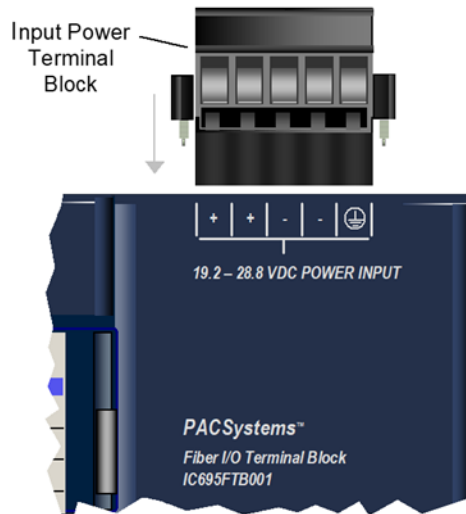
---

**Figure 10:Install Terminal Block onto FTB Module**



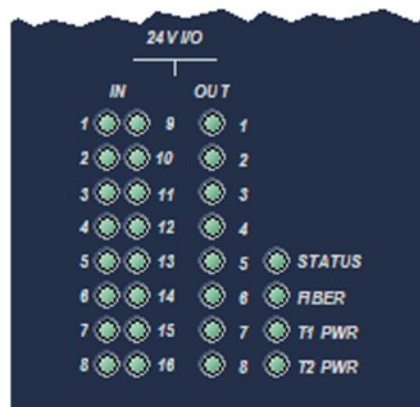
6. Install the input power terminal block onto the FTB power header.
7. Connect 24Vdc input power to one + and one - terminal. Connect the safety ground wire to the terminal marked with a ground symbol.

**Figure 11: Install Input Power Terminal Block onto FTB**



**Note:** When the FTB initially powers up, its STATUS LED briefly flashes amber and then remains on and green. The 24Vdc I/O LEDs will be red if the corresponding I/O connections are not configured.

**Figure 12: FTP LEDs**



Configure the PMM(s) and FTB(s) as described in Section 2.3, Basic Configuration.

## 2.3 Basic Configuration

This section provides a configuration process overview. Refer to Section 4 for details on configuration parameters.

For required programmer and CPU revisions, refer to PACMotion Multi-Axis Controller Important Product Information, GFK-3164.

Topics in this Section:

Section 2.3.1, Connecting the Programmer to the RX3i.

Section 2.3.2, Adding a Motion Controller Module to the Hardware Configuration

Section 2.3.3, Configuring PMM Settings

Section 2.3.4, Configuring PMM I/O

Section 2.3.5, Configuring Axis Parameters

Section 2.3.6, Drive Type

Section 2.3.7, Downloading the Configuration to the RX3i

### 2.3.1 Connecting the Programmer to the RX3i

All PMM programming is done through the configuration/programming software interface, yielding a single point of programming for the module. For more information, please refer to the PACSystems RX3i and RSTi-EP CPU Reference Manual, GFK-2222.

The RX3i programming environment has two communications options. You can connect the programmer directly to one of the COM ports of the CPU, or you can communicate with the CPU through the Ethernet network. If you use a serial port, it must be configured as RTU Slave (default) or SNP Slave.

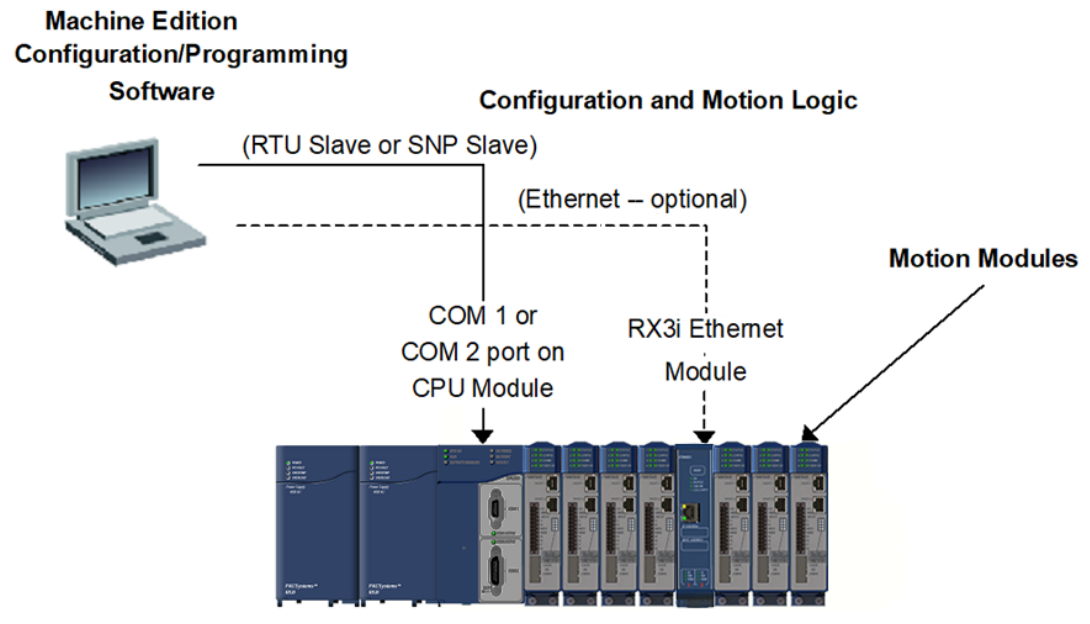
---

**Note:** *An IP address must be set in the RX3i before an Ethernet connection can be established. For details, refer to PACSystems RX3i and RSTi-EP TCP/IP Ethernet Communications User Manual, GFK-2224.*

---

## PMM345 Programmer Connection

Figure 13: Connecting PC hosting PME to RX3i



### 2.3.2 Adding a Motion Controller Module to the Hardware Configuration

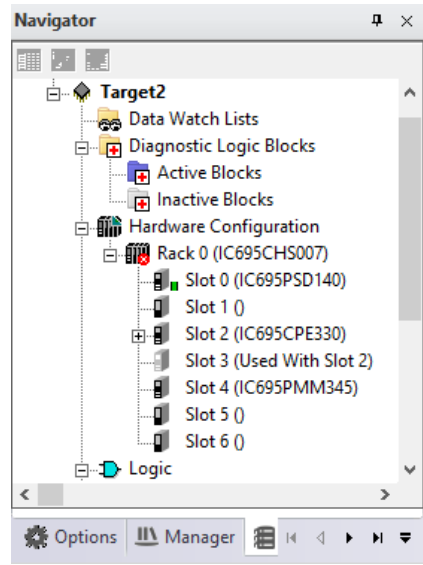
The hardware configuration defines the type and location of each module present in the RX3i racks. This is done by completing setup screens that represent the modules in a baseplate and then saving the information to a configuration file, which is downloaded to the RX3i CPU.

To configure a PMM using the PME software:

1. Create or open a project containing an RX3i target.
2. In the Navigator window, expand the Hardware Configuration folder.
3. If necessary, replace the power supply and/or CPU with the models that will be used in your application. To replace a module, right click and choose Replace Module.
4. Right click the slot where the PMM is to be configured and choose Add Module (choose Replace Module if a module is already configured in the slot).
5. In the Module Catalog, select the Motion tab, choose the PMM345 and click OK.

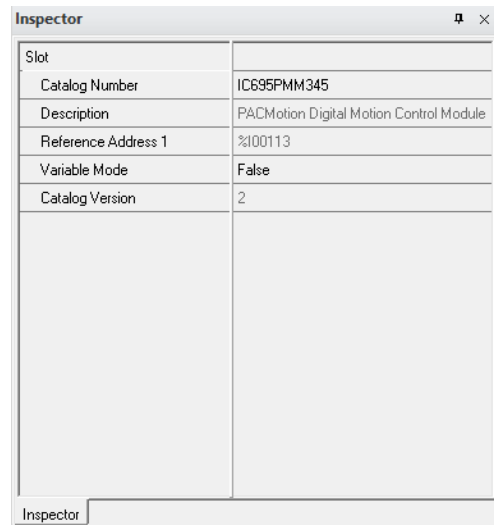
This operation adds the PMM345 to the RX3i rack and displays the PMM345 configuration screens that allow you to customize the PMM345 to your particular application.

Figure 14: Adding PMM to RX3i HWC in PME



To map I/O variables to the status data that the PMM sends to the RX3i controller, set the Variable Mode property for the module to True.

Figure 15: Setting Variable Mode Property



## 2.3.3 Configuring PMM Settings

**Note:** For detailed descriptions of these parameters, refer to Section 4, Configuration.

1. Select the starting reference address for I/O status data.
2. Assign a name to the module.
3. Select the desired CAM library management mode. The Automatic mode is recommended for most applications.
4. Select the fault-logging mode for errors and warnings.
5. Select the number of axes that you will be using.
6. Select whether you will be using a virtual axis.
7. Assign names to the axes that you will be using.
8. Select enabled or disabled mode for each axis.
9. If you want the CPU to poll the module's I/O data less than once per I/O scan, select an I/O scan set that has been defined in the CPU configuration. The default selection, which is 1, causes the status to be read every I/O scan.

### CAUTION

You should assign names to a PMM and its axes before writing application logic to address the PMM. Assigning a PMM name in hardware configuration creates a globally scoped variable of type MODULE\_REF and associated hardware variables that can be accessed by any logic block in the target.

If you attempt to reference a hardware variable in logic without first creating it in hardware configuration, the logic developer software creates a new variable that is locally scoped relative to the logic block. Subsequently assigning the same name in hardware configuration results in having two variables with the same name, but different scope.

**Figure 16: Parameters Configured in Settings Tab**

Settings   FP I/O   FTB Inputs   FTB Outputs   I/O Interrupts   Axis 1   Axis 2   Axis 3   Axis 4   Advanced   Power Consumption	
Parameters	Values
I/O Status Data Reference	%I00113
I/O Status Data Length	32
Module	M4_1
Cam Library Management	Automatic Mode
Log Messages in I/O Fault Table	Errors Only
Number of Axes	4
Number of Virtual Axes	0
Axis 1	M4_Axis1_1
Axis 1 Mode	PM EtherCAT Servo
Axis 2	M4_Axis2_1
Axis 2 Mode	PM EtherCAT Servo
Axis 3	M4_Axis3_1
Axis 3 Mode	PM EtherCAT Servo
Axis 4	M4_Axis4_1
Axis 4 Mode	PM EtherCAT Servo
I/O Scan Set	1



## 2.3.4 Configuring PMM I/O

The PMM faceplate includes eight built-in, configurable single-ended 24Vdc digital I/O points (six inputs and two input/outputs) on a plug-on screw terminal connector. The faceplate I/O is suitable for applications that require only a few motion specific I/O points

The faceplate I/O can be configured for a wide range of motion functions, including general-purpose and high-speed inputs, touch probes, hardware over-travel limits, and home switch inputs.

The faceplate I/O can be extended when necessary using the FTB, which supports a wider range of motion functions, including external encoder inputs for real axes and differential 5 Vdc I/O as well as single-ended I/O.

For detailed I/O connection information, refer to Section 3 I/O Wiring, Connections and LED Operation.

**Figure 17:Parameters Configured in FP I/O, FTB Inputs and FTB Outputs Tabs**

Settings		FP I/O	FTB Inputs	FTB Outputs	I/O Interrupts	Axis 1	Axis 2	Axis 3	Axis 4	Advanced	Power Consumption
Parameters		Values									
<i>FP IN1</i>		Axis 1 Touch Probe 1									
FP IN1 Input Ref		M4_FP_IN1_1									
<i>Touch Probe Detection</i>		Positive Edge Trigger									
<i>FP IN1 Open Wire Detect</i>		Disabled									
<i>FP IN2</i>		Axis 2 Touch Probe 1									
FP IN2 Input Ref		M4_FP_IN2_1									
<i>Touch Probe Detection</i>		Positive Edge Trigger									
<i>FP IN2 Open Wire Detect</i>		Disabled									
<i>FP IN3/OUT1</i>		Axis 1 Home Switch * For Analog or External									
FP IN3 Input/Output Ref		M4_FP_IN3_1									
<i>FP IN4/OUT2</i>		Axis 2 Home Switch * For Analog or External									
FP IN4 Input/Output Ref		M4_FP_IN4_1									
<i>FP IN5</i>		Axis 1 Overtravel +									
FP IN5 Input Ref		M4_FP_IN5_1									
<i>FP IN6</i>		Axis 1 Overtravel -									
FP IN6 Input Ref		M4_FP_IN6_1									
<i>FP IN7</i>		Axis 2 Overtravel +									
FP IN7 Input Ref		M4_FP_IN7_1									
<i>FP IN8</i>		Axis 2 Overtravel -									
FP IN8 Input Ref		M4_FP_IN8_1									
<i>FP Inputs Mode</i>		Source									
<i>FP Outputs Default</i>		Force Off									

## 2.3.5 Configuring Axis Parameters

Axis operational characteristics are configured on the Axis tabs. For detailed axis configuration information, refer to section 4.3.5.

For initial validation of configuration and system setup of an PM EtherCAT Servo axis, **Position Feedback Source** should be set to **Motor Encoder**. This is the default value.

Set the **Drive Type** to **PACMotion PSD411**. Ensure that **Motor Encoder Counts per Revolution** is set to the default value of **1048576**, which matches the value set for the Servo Drive. The default Drive Type “Synthetic” specifies a Synthetic Motor. This can be used to test applications without the need for real motors/servo drives.

**Figure 18: Parameters Configured in Axis Tab**

Settings   FP I/O   FTB Inputs   FTB Outputs   I/O Interrupts   Axis 1   Axis 2   Axis 3   Axis 4   Advanced   Power Consumption	
Parameters	Values
Stop Axis on FTB Error	Enabled
<i>Position Feedback Source</i>	Motor Encoder
<i>Axis Positioning Mode</i>	Linear
<i>Motor Encoder Mode</i>	Absolute
Motor Encoder User Units	1.0
Motor Encoder Counts	1
Motor Encoder Counts Per Motor Revolut...	1048576
Motor Encoder Maximum Positive RPM Li...	8191
Motor Encoder Maximum Negative RPM ...	8191
<i>External Device</i>	None
Over Travel Limit Switch	Enabled
Axis Direction	Normal
<i>Software End of Travel</i>	Enabled
High Software EDT Limit (uu)	8388607.0
Low Software EDT Limit (uu)	-8388608.0
Max Velocity System (RPM)	4000.0
Equivalent Velocity (uu/sec)	4369066.66666667
Max Acceleration System (RPM/sec)	40000.0
Equivalent Acceleration (uu/sec**2)	43690666.6666667
Max Deceleration System (RPM/sec)	40000.0
Equivalent Deceleration (uu/sec**2)	43690666.6666667
Max Jerk (uu/sec **3)	10000000000000.0
Drive Disable Delay (ms)	100
<i>Drive Type</i>	PACMotion PSD411
Motor Velocity Limit (RPM)	4000
Torque Limit (%)	100.0
<i>Position Lag Monitoring</i>	Disabled
Max Position Error (Motor Revs)	10.0
Equivalent Position Error (uu)	655360.0
In Position Zone (uu)	10.0
Position Loop Time Constant (ms)	100.0
Velocity Feedforward (%)	0.0
Error Stop Deceleration (uu/sec **2)	1000000.0
Error Stop Jerk (uu/sec **3)	10000000.0
Master Axis Velocity Filter (ms)	8
Feedback Moving Deadband (uu/sec)	100.0
Command Moving Deadband (uu/sec)	0.0
Sync Master Position Deadband (Master ...	10.0
Disabled Direction Deadband (uu)	0.0
Touchprobe 1 Drive Input	Disabled
Touchprobe 1 Detection	Positive Edge Trigger
Touchprobe 2 Drive Input	Disabled
Touchprobe 2 Detection	Positive Edge Trigger
Drive Homing Mode	Disabled
Drive Homing Direction	Positive

## 2.3.6 Drive Type

The Drive Type selects the data to be exchanged between the PMM345 and the Servo Drive. The default selection is **PACMotion PSD411**. This is the selection to utilize with PSD drive. The user can also select **Synthetic** to utilize simulated motor. The **Synthetic** drive type is useful for programming an application prior to having real motors and drives.

## 2.3.7 Downloading the Configuration to the RX3i

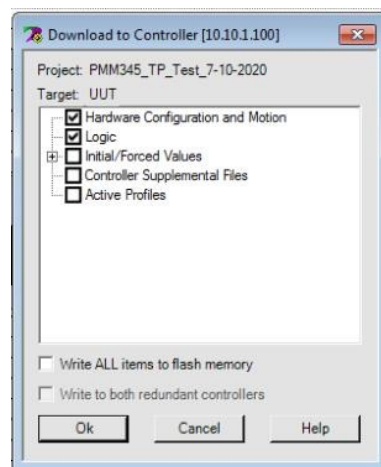
A PACSystems control system is configured by creating a configuration file in the programming software, then transferring (downloading) the file from the programmer to the CPU via serial port1, serial port 2, or an Ethernet Interface.

The CPU stores the configuration file in its non-volatile RAM memory. After the configuration is stored, I/O scanning is enabled or disabled according to the newly stored configuration parameters.

With the programmer connected and communicating with the RX3i (Section 2.3.1), perform the following steps.

1. In the Logic Developer software, go to the Project tab of the Navigator, right click the Target node, and choose Online Commands, Set Programmer Mode.
2. Make sure the CPU is in Stop mode.
3. In the Logic Developer software, go to the Project tab of the Navigator, right click the Target node, and choose Download to Controller.
4. In the Download to Controller dialog box, select the items to download and click OK.

**Figure 19: Downloading Configuration to the Target RX3i**



**⚠ CAUTION**

Initial/Forced Values must be selected for download.

---

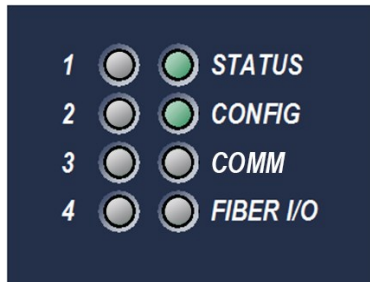
*Note: If I/O variables are configured, hardware configuration and logic cannot be stored independently. They must be stored at the same time.*

---

When the configuration download is complete, the CONFIG LED of the PMM is on and green.

---

**Figure 20: CONFIG LED State following Configuration Download**



## 2.4 Basic Motion Example – Jogging an Axis

This example provides sample logic for moving an axis using the jogging function.

PACMotion function blocks are implemented by either an edge-triggered (Execute) or Enable input. Edge triggered function blocks, such as MC\_Stop, operate only on the rising edge of the Execute input. To execute the function, the Execute input must first transition low for one function block invocation. This example does not include the logic required for setting Execute permissives low.

For a comparison of Executed versus Enabled function block operation, refer to Section 5.2.4, Function Block Triggering (Enabled vs. Executed Instructions).

### Hardware Configuration (HWC)

This example uses default HWC settings for the PMM except where noted.

#### Settings Tab

<b>Module:</b>	M5
<b>Axis 1:</b>	M5_Axis1

#### Terminals Tab

Axis1\_OK is an I/O variable mapped to the Axis 1 flag (bit 18) in the PMM Status Data (Section 4.3.2). When ON, this flag indicates that the axis is enabled in HWC and is not in the ErrorStop state.

## Ladder Diagram Example

**Note:** The MC\_JogAxis function block requires the axis to be powered on, but unlike other motion-generation function blocks, does not require the axis to have a valid initial position.

Comments	Sample Logic
<p>To perform motion, an axis must be enabled by an MC_Power function block. If Enable_Positive and Enable_Negative have the same state, motion is enabled in both directions.</p>	<p><b>Figure 21: Ladder Diagram Example (Rung 1)</b></p>
<p>Axis1_Pwr is controlled by the Status output of the MC_Power function block. When Axis1_Pwr is on, the MC_JogAxis function block has input power. To move the axis, apply power to the MC_JogAxis function block and to either the Enable_Positive or the Enable_Negative input.</p>	<p><b>Figure 22: Ladder Diagram Example (Rung 2)</b></p>

Comments	Sample Logic																					
<p>As long as MC_Power is active, jogging will continue until the Enable_Positive or Enable_Negative input is lowered or the MC_JogAxis is aborted by another command. An MC_Stop function block can be used to stop axis motion.</p>	<p><b>Figure 23: Ladder Diagram Example (Rung 3)</b></p>																					
<p>Axis1_Pwr is controlled by the Status output of the MC_Power function block. When Axis1_Pwr is on, the actual position of the axis is output to Axis_Position.</p>	<p><b>Figure 24: Ladder Diagram Example (Rung 4)</b></p> <table border="1" data-bbox="446 1197 933 1396"> <thead> <tr> <th>Variable Name</th> <th>Address</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>BasicExample.Axis1_OK</td> <td>%(IX0.5.0.19)</td> <td>1</td> </tr> <tr> <td>BasicExample.XAxis1Pwr</td> <td></td> <td>1</td> </tr> <tr> <td>BasicExample.Jog_Pos</td> <td></td> <td>1</td> </tr> <tr> <td>BasicExample.Jog_Neg</td> <td></td> <td>0</td> </tr> <tr> <td>BasicExample.XStop_Axis1</td> <td></td> <td>0</td> </tr> <tr> <td>BasicExample.Axis_Position</td> <td></td> <td>955881.0</td> </tr> </tbody> </table>	Variable Name	Address	Value	BasicExample.Axis1_OK	%(IX0.5.0.19)	1	BasicExample.XAxis1Pwr		1	BasicExample.Jog_Pos		1	BasicExample.Jog_Neg		0	BasicExample.XStop_Axis1		0	BasicExample.Axis_Position		955881.0
Variable Name	Address	Value																				
BasicExample.Axis1_OK	%(IX0.5.0.19)	1																				
BasicExample.XAxis1Pwr		1																				
BasicExample.Jog_Pos		1																				
BasicExample.Jog_Neg		0																				
BasicExample.XStop_Axis1		0																				
BasicExample.Axis_Position		955881.0																				

## 2.5 Using the Synthetic Motor

A synthetic motor provides the ability to send motion commands to the module without requiring an actual servo to be connected. This can be very beneficial for testing motion commands prior to implementation or in scenarios where an extra virtual path planner is needed. The synthetic axis performs a simplistic motor emulation by implementing an ideal velocity loop. The position loop is still active and requires tuning to achieve a low Position Error.

To use the synthetic motor with an axis, set the axis hardware configuration parameter, Drive Type to Synthetic . Note that a module can have both real and synthetic motors configured, but the synthetic motors must be on higher numbered axes. For example, if Axis 4 is configured to control an actual motor, Axis 1 cannot use the synthetic motor. The maximum encoder resolution, 1,048,576 counts per motor revolution, can be selected, which allows the maximum ranges for axis tuning parameters that are dependent on encoder resolution.

Certain hardware parameters are invalid for the synthetic motor because they relate to feedback from an actual motor. All axis-related parameters with numbers in the 10,000 range are ignored by axes using a synthetic motor. The parameters *Load Inertia Ratio*, *Over Travel Limit Switch*, and *Position Lag Monitoring*, which are configured by the programming software are also invalid. All other user-accessible axis configuration parameters apply when the synthetic motor is used.

Two tuning parameters to be particularly aware of are Position Loop Time Constant and Velocity Feedforward. The Position Loop Time Constant specifies how aggressively to remove position error from the position loop. A value that is too low will cause instability in the system. Velocity Feedforward specifies the percentage of commanded velocity that should be directly applied to the motor. For a synthetic motor, you can set Velocity Feedforward to 100% to reduce position error to zero.

For details on the effects of these and other configuration parameter settings, refer to Section 4, Configuration.

## 2.6 Axis Power-up and Feedback Device Initialization

To successfully execute motion-generating function blocks on an axis, the axis must be powered on by an MC\_Power function block and set to a valid initial position by an MC\_SetPosition or MC\_Home function block. The PositionValid axis status bit (parameter no. 1201) indicates successful initialization of the axis and must be ON for an axis to accept motion function blocks that generate motion other than for setting up the axis and/or establishing a valid home position.

If an axis has previously been set valid, when power is removed and then applied, the axis will retain its commanded position and PositionValid status.



## 2.7 Basic Motion Example – Axis Initialization

This example provides sample logic for moving an axis using the MC\_MoveAbsolute function.

PACMotion function blocks are implemented by either an edge-triggered (Execute) or Enable input. Edge-triggered function blocks, such as MC\_Stop, operate only on the rising edge of the Execute input. To execute the function, the Execute input must first transition low for one function block invocation. This example does not include the logic required for setting Execute permissives low.

For a comparison of Executed versus Enabled function block operation, refer to Section 5.2.4, Function Block Triggering (Enabled vs. Executed Instructions).

### 2.7.1 Hardware Configuration (HWC)

This example uses default HWC settings for the PMM except where noted. A synthetic axis (drive type Synthetic) is used, which requires Position Feedback Source to be set to Motor Encoder and External Device to be set to None.

#### Settings Tab

<b>Module:</b>	M5
<b>Axis 1:</b>	M5_Axis1

#### Terminals Tab

Axis1\_OK is an I/O variable mapped to the Axis 1 flag (bit 18) in the PMM Status Data (Section 4.3.2). When ON, this flag indicates that the axis is enabled in HWC and is not in the ErrorStop state.

## Ladder Diagram Example

Comments	Sample Logic
<p>To perform motion, an axis must be enabled by an MC_Power function block. If Enable_Positive and Enable_Negative have the same state, motion is enabled in both directions.</p>	<p><b>Figure 25: Ladder Logic to Enable Axis Motion</b></p>
<p>The axis is set to a valid initial position by an MC_SetPosition function block.</p>	<p><b>Figure 26: Ladder Logic for Setting Initial Position</b></p>

Comments	Sample Logic															
<p>The PositionValid axis status flag is on, indicating the axis is initialized and ready to perform motion.</p>	<p><b>Figure 27: Monitoring PositionValid Axis Status Flag</b></p> <thead> <tr> <th>Variable Name</th> <th>Address</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>BasicExample.M5_Axis1_OK</td> <td>%IX0.5...</td> <td>On</td> </tr> <tr> <td>BasicExample.XAxis_Pwr</td> <td></td> <td>On</td> </tr> <tr> <td>BasicExample.XAxisSet</td> <td></td> <td>On</td> </tr> <tr style="background-color: #008080; color: white;"> <td>BasicExample.Axis1_PosValid</td> <td></td> <td>On</td> </tr> </tbody>	Variable Name	Address	Value	BasicExample.M5_Axis1_OK	%IX0.5...	On	BasicExample.XAxis_Pwr		On	BasicExample.XAxisSet		On	BasicExample.Axis1_PosValid		On
Variable Name	Address	Value														
BasicExample.M5_Axis1_OK	%IX0.5...	On														
BasicExample.XAxis_Pwr		On														
BasicExample.XAxisSet		On														
BasicExample.Axis1_PosValid		On														

<p>xAxis1_MoveAbs transitions ON, executing the MC_MoveAbsolute function block.</p>	<p><b>Figure 28: Execute the MC_MoveAbsolute Function Block</b></p>
---	---

Comments	Sample Logic																														
<p>MC_ReadActualPosition and MC_ReadActualVelocity are used to monitor axis actual position and velocity. These values can also be monitored using MC_ReadParameter(s) to read parameter numbers 10 and 1300 for the axis.</p>	<p><b>Figure 29: Monitor Axis Actual Position and Velocity</b></p> <p>The diagram illustrates two PLC rungs. The top rung is for monitoring axis actual position. It features a normally open contact labeled #ALW_ON and a coil labeled M5_Axis1. The output is connected to the 'Enable' input of the MC_READACTUALPOSITION block. The block's 'Axis' input is also labeled M5_Axis1. The outputs of this block are: Axis (M5_Axis1), Valid, Busy, Warning, Error (0), ErrorID (0), and Position (1200.0). The ErrorID output is connected to the variable ErrID_ReadAct1_Position, and the Position output is connected to Axis1_ActualPosition.</p> <p>The bottom rung is for monitoring axis actual velocity. It features a normally open contact labeled #ALW_ON and a coil labeled M5_Axis1. The output is connected to the 'Enable' input of the MC_READACTUALVELOCITY block. The block's 'Axis' input is also labeled M5_Axis1. The outputs of this block are: Axis (M5_Axis1), Valid, Busy, Warning, Error (0), ErrorID (0.0), and ActualVelocity (0.0). The ErrorID output is connected to the variable ErrID_READAct1_Velocity, and the ActualVelocity output is connected to Axis1_ActualVelocity.</p> <p>A Data Watch window is open, displaying the following table:</p> <table border="1"> <thead> <tr> <th>Variable Name</th> <th>Address</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>BasicExample.M5_Axis1_OK</td> <td>%I0.5...</td> <td>On</td> </tr> <tr> <td>BasicExample.XAxis_Pwr</td> <td></td> <td>On</td> </tr> <tr> <td>BasicExample.XAxisSet</td> <td></td> <td>On</td> </tr> <tr> <td>BasicExample.Axis1_PosValid</td> <td></td> <td>On</td> </tr> <tr> <td>BasicExample.xAxis1_MoveAbs</td> <td></td> <td>On</td> </tr> <tr> <td>BasicExample.Axis1_InVelocity</td> <td></td> <td>On</td> </tr> <tr> <td>BasicExample.Axis1_ActualPosition</td> <td></td> <td>1200.0</td> </tr> <tr> <td>BasicExample.Axis1_ActualVelocity</td> <td></td> <td>0.0</td> </tr> <tr> <td>BasicExample.xStopAllAxes</td> <td></td> <td>Off</td> </tr> </tbody> </table>	Variable Name	Address	Value	BasicExample.M5_Axis1_OK	%I0.5...	On	BasicExample.XAxis_Pwr		On	BasicExample.XAxisSet		On	BasicExample.Axis1_PosValid		On	BasicExample.xAxis1_MoveAbs		On	BasicExample.Axis1_InVelocity		On	BasicExample.Axis1_ActualPosition		1200.0	BasicExample.Axis1_ActualVelocity		0.0	BasicExample.xStopAllAxes		Off
Variable Name	Address	Value																													
BasicExample.M5_Axis1_OK	%I0.5...	On																													
BasicExample.XAxis_Pwr		On																													
BasicExample.XAxisSet		On																													
BasicExample.Axis1_PosValid		On																													
BasicExample.xAxis1_MoveAbs		On																													
BasicExample.Axis1_InVelocity		On																													
BasicExample.Axis1_ActualPosition		1200.0																													
BasicExample.Axis1_ActualVelocity		0.0																													
BasicExample.xStopAllAxes		Off																													

# Section 3 I/O Wiring, Connections and LED Operation

This chapter provides details on cable and wiring connections to the PACMotion Multi-axis Motion Controller (PMM) and the Fiber Terminal Block I/O (FTB)

Topics Covered:

Section 3.1 PMM Faceplate I/O

Section 3.2 Fiber Terminal Block I/O

Section 3.3 Errors Indicated by LEDs

Section 3.4 I/O Circuit Specifications

Section 3.5 EtherCAT Command Interface Cable

Section 3.6 FTB to PMM Connection

Section 3.7 Grounding the PACMotion System

## 3.1 PMM Faceplate I/O

The PMM faceplate provides access to eight digital I/O circuits, consisting of

- Six general-purpose 24Vdc inputs, and
- Two high-speed 24Vdc source/sink inputs.

Two of the I/O points can be configured as general-purpose 24Vdc inputs or 24Vdc source/sink output drivers.

The inputs and outputs are in a single group with power and common terminals. The PME hardware configuration tool allows you to assign I/O functions to specific faceplate I/O points based upon how connections to the module are wired.

The two high-speed inputs, IN1 and IN2, accept a 24Vdc master axis quadrature encoder at count rates up to 500 kHz. These two inputs provide open wire detection, which is enabled or disabled in hardware configuration.

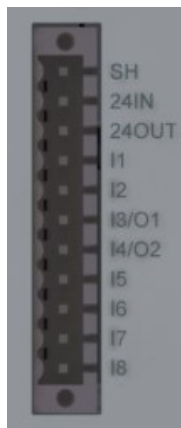
The output drivers are connected to the same terminals as two low-speed inputs, IN3 and IN4, allowing the terminals to be used as inputs or outputs. The two output points can provide a total current of up to 250mA (125mA on each point, or 200mA on one and 50mA on the other, etc.). A shield terminal is provided for termination of optional cable shields.

The PMM faceplate I/O requires 24Vdc user-supplied power.

The mating plug supplied with the PMM can use AWG #14-22 wire for each terminal. The plug should be tightened with the retaining screws with a max torque of 2.25 lb-in(0.25 N-m).

---

**Figure 30: I/O Terminals on PMM Faceplate**



---

**Note:** *If Open Wire Detection is enabled on faceplate inputs IN1 and IN2, a broken or disconnected signal wire will generate faults. Open wire detection is recommended if those signals are encoder inputs used as a master position source, or any other application where sensing loss of encoder feedback is critical. The faceplate encoder inputs, because they are single-ended, cannot generate a fault on loss of encoder power in the way that differential FTB encoder inputs can. However, if the faceplate encoder is powered from the same source as the faceplate, loss of power will trigger a fault.*

---

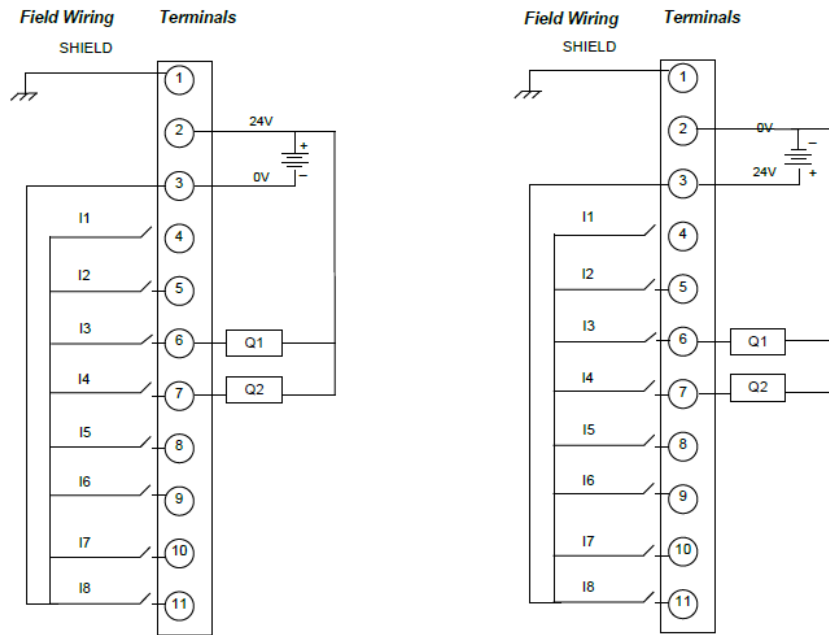
For specifications of I/O circuits, refer to Section 3.4.1, PMM Faceplate I/O Circuits.

### 3.1.1 PMM Faceplate Wiring Diagrams and Pin Assignments

#### Faceplate I/O Connector Pin Assignments

Pins	Faceplate Identifier	Circuit Identifier	Circuit Type	Default Circuit Function
1	SHIELD	SHIELD	Frame Ground	Termination of optional cable shields
2	24Vdc INCOM	—	24Vdc Common	24Vdc Inputs Common
3	24Vdc OUTCOM	—	24Vdc Common	24Vdc Outputs Common
4	IN1	I1	High-speed 24Vdc Input	Axis 1 Touch Probe
5	IN2	I2	High-speed 24Vdc Input	Axis 2 Touch Probe
6	IN3/OUT1	I3	General-purpose 24Vdc Input or 24Vdc Output	Axis 1 Home Switch 1
7	IN4/OUT2	I4	General-purpose 24Vdc Input or 24Vdc Output	Axis 2 Home Switch 2
8	IN5	I5	General-purpose 24Vdc Input	OT1+
9	IN6	I6	General-purpose 24Vdc Input	OT1-
10	IN7	I7	General-purpose 24Vdc Input	OT2+
11	IN8	I8	General-purpose 24Vdc Input	OT2-

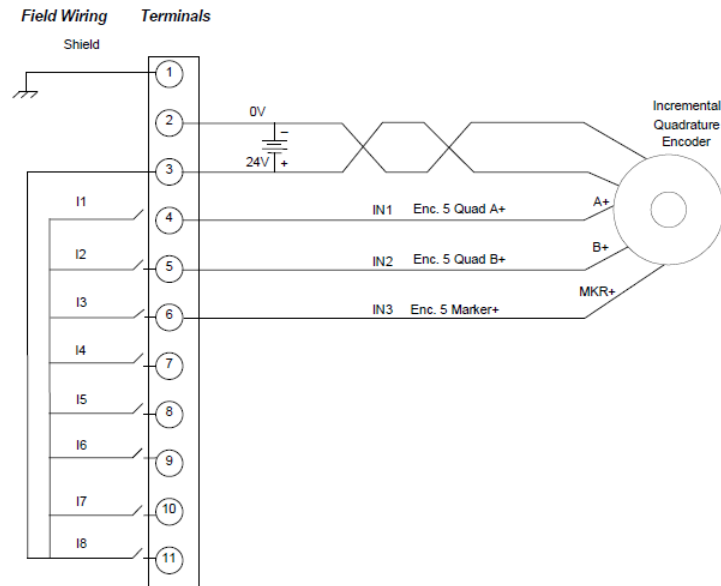
Figure 31: Sink Mode Wiring Figure 32: Source Mode Wiring



## Typical External Encoder Connection

When a quadrature encoder is used with the faceplate, the quadrature totem pole drivers are powered from the 24Vdc supply across pins 2 and 3. The encoder Quadrature signals for Axis 5 are connected to pins 4 and 5. The Marker signal, which can be connected to any of pins 6–8, is connected to pin 6 in the following example.

**Figure 33: Typical External Encoder Connection**

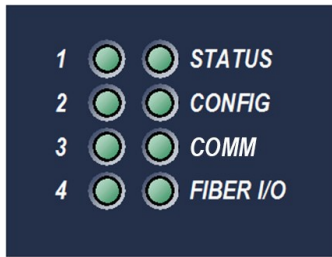




### 3.1.2 PMM345 LED Operation

The PMM has eight LEDs that provide status indications. Four of the LEDs indicate individual axis status. Two LEDs indicate the operating status of the module. The remaining two LEDs indicate the status of the COM and FTB communications links.

**Figure 34: PMM Axis LEDs & Status LEDs**



#### Axis LEDs

LED	LED State	PMM345 Operating State
1 (Axis 1)	Green ON	The Axis drive corresponding to this LED is enabled.
2 (Axis 2)	Red ON	A normal stop error has occurred on this axis.
3 (Axis 3)	Red, blinking 500ms interval	A fast stop error has occurred on this axis.
4 (Axis 4)		

#### PMM Status LEDs

LED	LED State	PMM345 Operating State
STATUS	OFF	The PMM does not have power.
	Green ON	The PMM345 is functioning properly and there are no errors on the module.
	Green, blinking 1 second interval	Indicates that a Warning or an Error not requiring a stop has occurred.
	Green, blinking 500ms interval	Indicates that an Error requiring a fast or normal stop has occurred.
	Green, blinking blink code; CONFIG and axis LEDs OFF	The blink pattern indicates that a fatal error has occurred on the module. Record the number of blinks in the sequence and contact Technical Support for additional information.
	Green, blinking simultaneously with the CONFIG and axis LEDs	The module is currently loading a firmware upgrade
	Amber ON	A severe module hardware error or watchdog timeout has occurred.
CONFIG	Green ON	The PMM has received a valid configuration from the RX3i CPU.
	Green, blinking	The PMM has not yet received a configuration from the programmer.
	Amber ON	The PMM is in boot mode.
	Amber, blinking	The PMM received an invalid configuration from the programmer.

## Communications Status LEDs

LED	LED State	PMM345 Operating State
COMM	OFF	EtherCAT communications is inactive.
	Green ON	EtherCAT communications is active.
	Green, blinking slow	One or more servo drives are not present on the EtherCAT network.
	Green, blinking	EtherCAT setup is in progress, EtherCAT network is disconnected, or no servo drives are present on the network. Axes can be used with synthetic motor.
Fiber I/O	OFF	Fiber I/O communications is inactive.
	Green ON	Fiber I/O communications is active.
	Green blinking	Fiber I/O configuration is in progress.
	Red ON	Fiber I/O communication link has failed.
	Alternately blinking green and red	Indicates an FTB ID error. The PMM is attempting to communicate with an FTB that does not have the correct PMM/FTB communications link ID.

Figure 35: EtherCAT/Network Status LEDs



## EtherCAT/Network Status LEDs

LED (Name on Faceplate)	LED State	PMM345 Operating State
LED1 (LNK1)	Green ON	Port 1 is linked to the Ethernet.
	OFF	Port 1 has no link to the Ethernet.
LED2 (ACT1)	Yellow, flickering (load dependent)	Port 1 sends/receives EtherCAT frames.
	OFF	Port 1 does not send/receive EtherCAT frames.
LED3 (LNK2)	Green ON	Port 2 is linked to the Ethernet.
	OFF	Port 2 has no link to the Ethernet.
LED4 (ACT2)	Yellow, flickering (load dependent)	Port 2 sends/receives EtherCAT frames.
	OFF	Port 2 does not send/receive EtherCAT frames.
LED5 (STAT)	OFF	EtherCAT master is in the INIT state.
	Green, flickering (10 Hz)	EtherCAT master is in Boot mode.
	Green, blinking (2.5 Hz)	EtherCAT master is in the PRE-OPERATIONAL state.
	Green, Single flash	EtherCAT master is in the SAFE-OPERATIONAL state.
	Green, On	EtherCAT master is in the OPERATIONAL state.
LED6 (ERR)	OFF	EtherCAT master has no errors.
	Red, single flash	EtherCAT Bus Sync error.
	Red, double flash	Internal stop of the EtherCAT bus cycle.
	Red, triple flash	Watchdog has expired.
	Red, quadruple flash	Hardware failure.
	Red, blinking (2.5 Hz)	EtherCAT network configuration error.
	Red, single flickering	EtherCAT channel initialization, transient may or may not be visible.
	Red, double flickering	Configured EtherCAT drive is missing, misconfigured, or bus is not connected.
LED7 (CFG)	Red, flickering (10 Hz)	EtherCAT master boot-up was stopped due to an error.
	Yellow, flashing	EtherCAT master is booting.
LED8 (RUN)	Yellow, solid	EtherCAT master error during boot.
	Green, flashing	EtherCAT master second stage booting.
	Green, solid	EtherCAT master is running.
	OFF	No power or hardware failure.

## 3.2 Fiber Terminal Block I/O

The FTB provides the ability to locate I/O connections near motors and controlled devices. The FTB contains an external I/O controller that communicates axis control and status data, including signals from incremental quadrature encoders, touch probes, home switches and overtravel switches, through a noise immune fiber optic serial link.

Fault detection for the 5Vdc differential inputs includes open wire and loss of encoder power. Fault detection is enabled or disabled in hardware configuration.

The PME hardware configuration tool allows you to assign I/O functions to specific terminal block I/O points based upon how connections to the module are wired. Sample connection diagrams that assign the resources for a typical application are provided in Section 3.2.4, FTB Wiring Diagrams and Pin Assignment.

The FTB uses two removable 36-pin spring-style or screw-style terminal blocks. These must be ordered separately. Each terminal type is available with an extended wiring shroud. Part numbers for each terminal block type are shown below.

Shielded wiring is required for 5Vdc and analog I/O lines to provide noise immunity.

### 3.2.1 IC695FTB001 Fiber Terminal Block I/O Specifications

Power requirements	19.2Vdc –28.8Vdc main power, 0.45A at 24Vdc
Physical dimensions	141.2 mm wide x 125.5 mm tall x 62.5 mm deep (5.56" x 4.94" x 2.46")
24Vdc Outputs (differential)	8, 2 groups of 4 optically isolated @ 4A max per group
24Vdc Inputs (general purpose)	16, 4 groups of 4 optically isolated
5Vdc Input/Outputs	4, differential inputs or 4, differential outputs
5Vdc Inputs	8, differential (6 can be used as Single-ended)
Analog Inputs	2, ±10Vdc differential
Analog Outputs	2, ±10Vdc single-ended
5Vdc Power Output	0.5A max.
Quad Encoder Open Circuit Detection	Yes
Encoder Power output	5Vdc at 0.5A max

For RX3i environmental specifications, refer to the PACSystems RX3i System Manual, GFK-2314.

## 3.2.2 Terminal Header and Cable Options

The following types of terminal blocks are available for use with the FTB. For information on the use of these terminal blocks, refer to the PACSystems RX3i System Manual, GFK 2314.

High Density Terminal Header Options	
36-point Spring Clip Terminals	IC694TBS032
36-point Captive Screw Terminals	IC694TBB032
Spring Clip Terminals, Extended Shroud	IC694TBS132
Captive Screw Terminals, Extended Shroud	IC694TBB132
[optional] Sheathed Fiber Optic Cable <sup>1</sup>	
1 meter	ZA66L-6001-0026#L1R003
5 meter	ZA66L-6001-0026#L5R003
10 meter	ZA66L-6001-0026#L10R03
20 meter	ZA66L-6001-0026#L20R03
50 meter	ZA66L-6001-0026#L50R03
100 meter	ZA66L-6001-0026#L100R3
[optional] Unsheathed Fiber Optic Cable <sup>2</sup>	
0.30 meter	ZA66L-6001-0023#L300R003
1 meter	ZA66L-6001-0023#L1R003
2 meter	ZA66L-6001-0023#L2R003
3 meter	ZA66L-6001-0023#L3R003

### WARNING

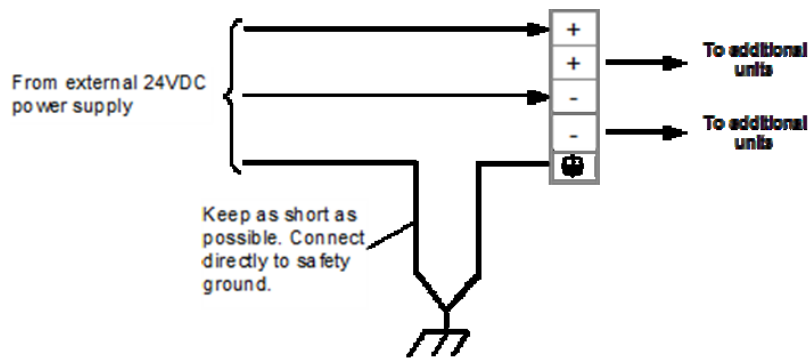
Power to I/O devices must be turned off when installing or removing a terminal block assembly.

<sup>1</sup> For fiber optic lengths less than 10 meters, unsheathed cable is recommended.  
 For fiber optic lengths from 10 to 100 meters, sheathed cable is required.

### 3.2.3 FTB Input Power

Each terminal accepts one AWG #14 (2.1 mm<sup>2</sup>) or two AWG #16 (1.3mm<sup>2</sup>) copper 75°C (167°F) wires. The suggested torque for the input power terminals is 12 in-lbs (1.36 Newton-meters). Each terminal can accept solid or stranded wires. Both the wires in a terminal should be the same type.

**Figure 36: FTB Input Power Wiring**

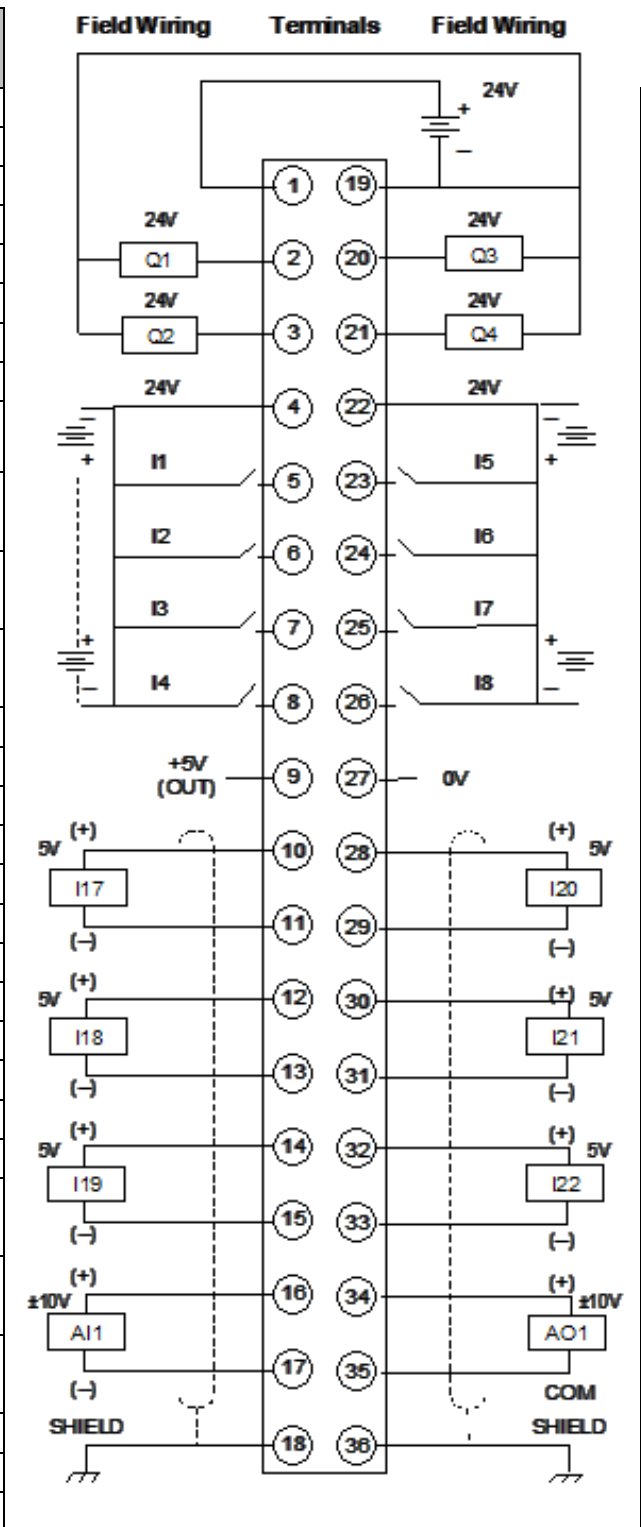


### 3.2.4 FTB Wiring Diagrams and Pin Assignment

FTB Terminal 1 Pin Assignments

Pin	Circuit Identifier	Circuit Type	Default Circuit Function
1	24Vdc+	24Vdc Output	Q1–Q4 Power
2	Q1	24Vdc (ESCP) Output	24Vdc Output
3	Q2	Output	24Vdc Output
4	24-	24Vdc-	I1–I4 Common
5	I1	24Vdc Input	Digital Input
6	I2		Digital Input
7	I3		Digital Input
8	I4		Digital Input
9	+5Vdc (OUT)	+5Vdc Output	External Power
10	I17+	5Vdc Diff Input+	Fast Digital Input
11	I17-	5Vdc Diff Input-	
12	I18+	5Vdc Diff Input+	Fast Digital Input
13	I18-	5Vdc Diff Input-	
14	I19+	5Vdc Diff Input+	Fast Digital Input
15	I19-	5Vdc Diff Input-	
16	AI1+	± 10Vdc Analog Input	Analog In 1 (+)
17	AI1-		Analog In 1 (-)
18	Shield	Shield	Frame Ground
19	24Vdc-	24Vdc-	Q1–Q4 Common
20	Q3	24Vdc (ESCP) Output	24Vdc Output
21	Q4	Output	24Vdc Output
22	24Vdc-	24Vdc-	I5–I8 Common
23	I5	24Vdc Input	Digital Input
24	I6		Digital Input
25	I7		Digital Input
26	I8		Digital Input
27	0V	0V	External Power
28	I20+	5Vdc Diff Input+	Fast Digital Input
29	I20-	5Vdc Diff Input-	
30	I21+	5Vdc Diff Input+	Fast Digital Input
31	I21-	5Vdc Diff Input-	
32	I22+	5Vdc Diff Input+	Fast Digital Input
33	I22-	5Vdc Diff Input-	
34	AO1+	±10Vdc Analog Output	Analog Out 1
35	COM		AO1 Common
36	Shield	Shield	Frame Ground

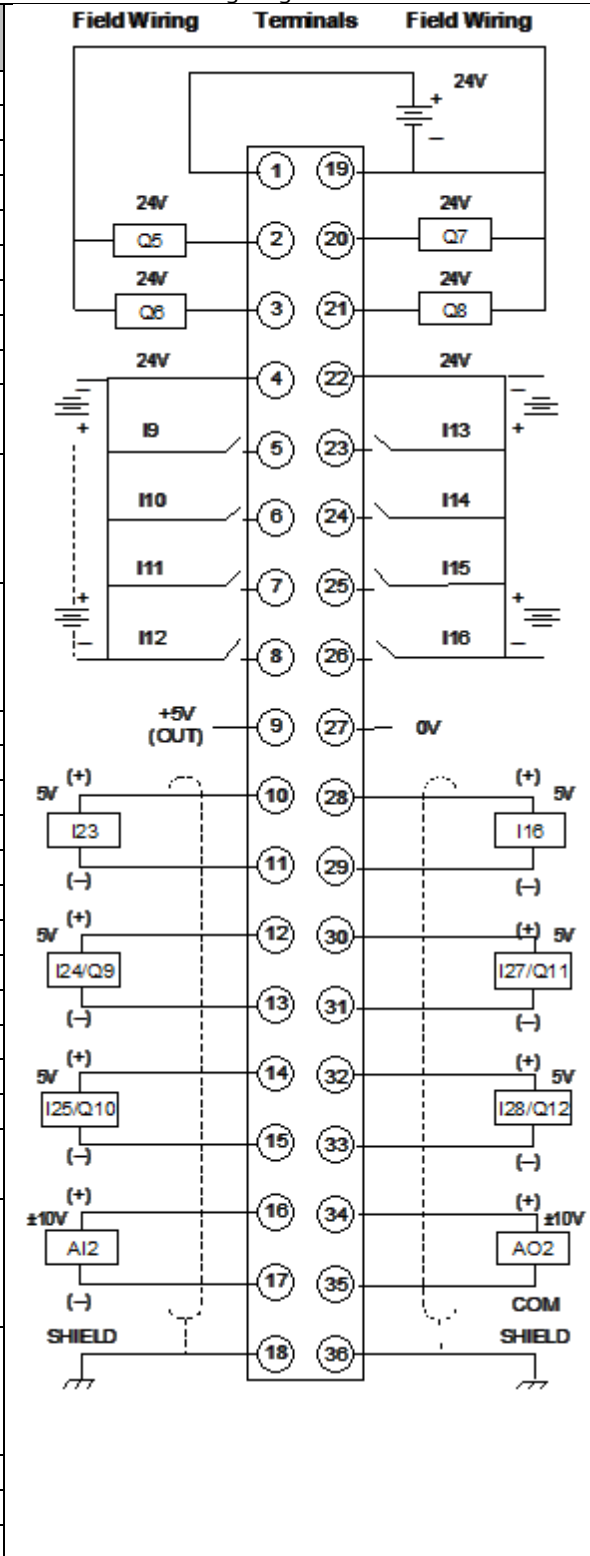
FTB Terminal 1 Wiring Diagram



FTB Terminal 2 Pin Assignments

Pin	Circuit Identifier	Circuit Type	Default Circuit Function
1	24Vdc+	24Vdc Output	Q5—Q8 Power
2	Q5	24Vdc (ESCP) Output	24Vdc Output
3	Q6	Output	24Vdc Output
4	24-	24Vdc-	I9—I12 Common
5	I9	24Vdc Input	Digital Input
6	I10		Digital Input
7	I11		Digital Input
8	I12		Digital Input
9	+5Vdc (OUT)	+5Vdc OUT	External Power
10	I23+	5Vdc Diff Input+	Fast Digital Input
11	I23-	5Vdc Diff Input-	
12	I24+/Q9+	5Vdc Diff Input+/ 5Vdc Diff Output+	Fast Digital Input
13	I24-/Q9-	5Vdc Diff Input-/ 5Vdc Diff Output-	
14	I25+/Q10+	5Vdc Diff Input+/ 5Vdc Diff Output+	Fast Digital Input
15	I25-/Q10-	5Vdc Diff Input-/ 5Vdc Diff Output-	
16	AI2+	±10Vdc Analog Input	Analog In 2 (+)
17	AI2-		Analog In 2 (-)
18	SHIELD	Frame Ground	Shield
19	24Vdc-	24Vdc-	Q5—Q8 Common
20	Q7	24Vdc (ESCP) Output	24Vdc Output
21	Q8	Output	24Vdc Output
22	24Vdc+	24Vdc-	I13—I16 Common
23	I13	24Vdc (ESCP) Input	Digital Input
24	I14		Digital Input
25	I15		Digital Input
26	I16		Digital Input
27	0V	0V	External Power
28	I26+	5Vdc Diff Input	Fast Digital Input
29	I26-	5Vdc Diff Input	
30	I27/Q11+	5Vdc Diff Input+/ 5Vdc Diff Output+	Fast Digital Input
31	I27/Q11-	5Vdc Diff Input-/ 5Vdc Diff Output-	
32	I28/Q12+	5Vdc Diff Input+/ 5Vdc Diff Output+	Fast Digital Input
33	I28/Q12-	5Vdc Diff Input-/ 5Vdc Diff Output-	
34	AO2+	± 10Vdc Analog Output	Analog Output 2
35	COM	Output	AO2 Common
36	SHIELD	Frame Ground	Shield

FTB Terminal 2 Wiring Diagram



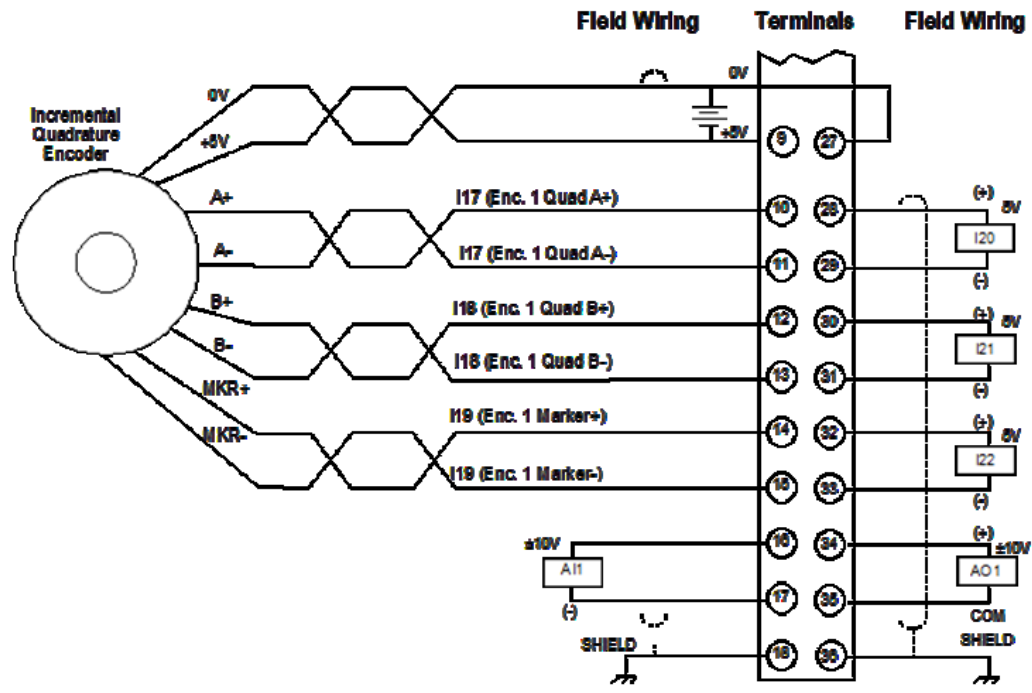


### 3.2.5 Typical External Differential Encoder Connection for FTB

When a differential quadrature encoder is used with the FTB, the quadrature drivers are powered from the 5Vdc supply across pins 9 and 27. In the following example, the encoder Quadrature signals are connected to pins 10–13. The Marker signal is connected to pins 14 and 15.

To avoid noise issues, ensure the motor frame is well grounded.

**Figure 37: External Differential Encoder Connection for FTB**

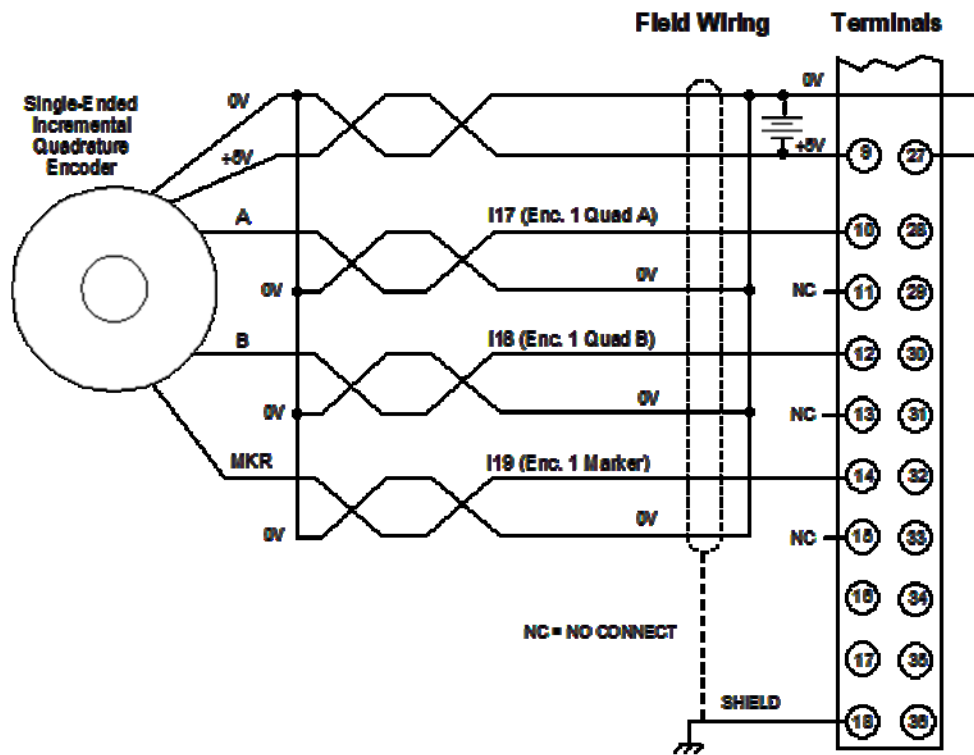


### 3.2.6 Typical Single-Ended Encoder Connection for FTB

Section 3.2.4, FTB Wiring Diagrams and Pin Assignment maps the pins to the I/O signals. (Refer to Section 3.2.5 for a wiring example of a differential encoder.)

Since this configuration is single-ended, only the + side should be connected. The +5Vdc output can be used to power the encoder and, as shown below, the A channel should be connected to pin 10, the B channel connected to pin 12, and the Z (marker) channel connected to pin 14. The “NC” pins (11, 13, and 15) must be left floating and cannot be used for another purpose in this mode. To improve noise immunity and reduce channel coupling, a twisted pair can be used for each encoder channel with one wire of each pair connected to 0V at each end.

**Figure 38: Single -Ended Encoder Connection for FTB**



For an example that illustrates how to configure and program an axis with a single-ended external encoder to act as the master to a velocity-following (gear) axis, refer to Appendix Section B-2.

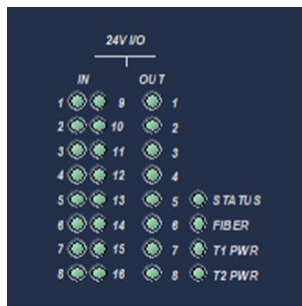
## 3.2.7 FTB LED Operation

The FTB has one LED for each of the 24Vdc I/O points, which indicates the state of the I/O point to allow visual debugging. Four additional LEDs indicate the operating status of the FTB.

### 24Vdc I/O LEDs

LED	LED State	FTB Operating State
In 1-16	Green	Input is ON
	Off	Input is OFF
Out 1-8	Green	Output is ON
	Off	Output is OFF
	Red	Output is in fault state

Figure 39: FTP I/O & Status LEDs



### Status LEDs

LED	LED State	FTB Operating State
STATUS	Off	No power applied to the FTB.
	Green	Proper operation.
	Amber	Configuration not yet received.
FIBER	Off	No power applied to the FTB.
	Green	I/O link established.
	Green, blinking	I/O link configuration data being transferred.
	Alternately blinking Green and Red	Indicates an FTB ID error. The PMM is attempting to communicate with an FTB that does not have the correct PMM/FTB communications link ID.
	Red	I/O link not established.
T1 PWR	Off	The FTB OUT 1-4 Power Detect setting in hardware configuration is Disabled.
	Green	The FTB OUT 1-4 Power Detect setting in hardware configuration is Enabled, and 24Vdc power is applied to terminal 1 (OUT POWER) and terminal 19 (OUT COMMON) of terminal block 1.
	Red	The FTB OUT 1-4 Power Detect setting in hardware configuration is Enabled and no power is applied to terminals 1 and 19 of terminal block 1.
T2 PWR	Off	The FTB OUT 5-8 Power Detect setting in hardware configuration is Disabled.
	Green	The FTB OUT 5-8 Power Detect setting in hardware configuration is Enabled and 24Vdc power is applied to terminals 1 and 19 of terminal block 2.
	Red	The FTB OUT 5-8 Power Detect setting in hardware configuration is Enabled and no power is applied to terminals 1 and 19 of terminal block 2.

## 3.3 Errors Indicated by LEDs

Use the MC\_ReadAxisError or MC\_ReadEventQueue function block to obtain ErrorIDs that are reported in Axis Error Codes.

### 3.3.1 PMM LEDs

Symptom	Possible Causes	Suggested Correction
Axis LED red, ON	A normal stop error has occurred on this axis	Check the Axis Error Code for additional information.
Axis LED red, blinking at 500ms interval	A fast stop error has occurred on this axis. The STATUS LED will blink at the same rate for these types of errors.	Correct the source of the error. Then use MC_Reset to transition the axis out of the ErrorStop state.
STATUS LED green, blinking at 1-second interval.	A Warning or an Error not requiring a stop has occurred.	Correct the source of the error(s). Then use a MC_ModuleReset to clear all errors on the PACMotion module and return any axes in the ErrorStop state to the Standstill state.
STATUS LED green, blinking 500ms interval	An Error requiring a fast or normal stop has occurred.	
STATUS LED green, blinking blink code; CONFIG and axis LEDs OFF	A fatal error has occurred on the module. The STATUS LED flashes the number of times corresponding to the error code, pauses, and then repeats the pattern.	Record the number of blinks in the sequence and contact Technical Support for additional information.
STATUS LED amber	A severe module hardware error or watchdog timeout has occurred.	Contact Technical Support with the fault table content and event queue, captured as soon after the error occurred as possible, from the PMM generating the error. Technical Support personnel may also ask for copies of the logic and/or HWC for the system experiencing the error.
CONFIG LED green, blinking	The PMM has not yet received a configuration from the programmer.	Store a configuration to the RX3i.
CONFIG LED amber, blinking	The PMM received an invalid configuration from the programmer.	Verify the module has the correct firmware version to support the features being configured. The latest firmware version can be downloaded from the Technical Support website. Store a valid configuration to the RX3i.
COMM Green, blinking slow	EtherCAT Cable Disconnected between Servo Drives	Check to make sure all Required EtherCAT Drives in your HW Cfg are connected and have power
COMM Green, blinking	EtherCAT setup is in progress, EtherCAT network is disconnected, or no servo drives are present on the network. Axes can be used with synthetic motor.	Check EtherCAT Cable connections to the Servo Drives and Servo Drives have power.
FTB LED alternately blinking green and red	The PMM is attempting to communicate with an FTB that does not have the correct PMM/FTB communications link ID.	Make sure that the correct FTB is connected to the PMM. It may be necessary to reset the Fiber Terminal Block Identifier to 0 in the hardware configuration.

### 3.3.2 PMM EtherCAT LEDs

Symptom	Possible Causes	Suggested Correction
LNK1 LED is off	First configured EtherCAT drive is disconnected or powered off.	Verify drive is powered on and Ethernet cable is securely connected.
ACT1 LED is off	Hardware configuration has not been downloaded or does not match the current network topology.	
LNK2 LED is off	Port 2 is not currently supported; this LED will remain off.	Do not connect Port 2 to the EtherCAT network.
STAT LED	OFF	EtherCAT master is in the INIT state.
	Green, flickering (10 Hz)	EtherCAT master is in Boot mode.
	Green, blinking (2.5 Hz)	EtherCAT master is in the PRE-OPERATIONAL state.
	Green, Single flash	EtherCAT master is in the SAFE-OPERATIONAL state.
	Green, On	EtherCAT master is in the OPERATIONAL state.
ERR is flashing red	OFF	EtherCAT master has no errors.
	Red, single flash	EtherCAT Bus Sync error.
	Red, double flash	Internal stop of the EtherCAT bus cycle.
	Red, triple flash	Watchdog has expired.
	Red, quadruple flash	Hardware failure.
	Red, blinking (2.5 Hz)	EtherCAT network configuration error.
	Red, single flickering	EtherCAT channel initialization, transient may or may not be visible.
	Red, double flickering	Configured EtherCAT drive is missing, misconfigured, or bus is not connected.
	Red, flickering (10 Hz)	EtherCAT master boot-up was stopped due to an error
CFG LED is flashing or solid yellow	Yellow, flashing	EtherCAT master is booting.
	Yellow, solid	EtherCAT master error during boot.
RUN LED is off	Green, flashing	EtherCAT master second stage booting.
	Green, solid	EtherCAT master is running.
	OFF	No power or hardware failure.

### 3.3.3 FTB LEDs

Symptom	Possible Causes	Suggested Correction
24Vdc IO Out1 – Out 8 LED is red, ON	The specified 24-volt output has open load error.	Check wiring, or disable open load fault detection in HWC.
STATUS LED is amber, ON	Configuration not yet received.	Download a valid configuration to the RX3i.
FIBER LED alternately blinking Green and Red	The PMM is attempting to communicate with an FTB that does not have the correct PMM/FTB communications link ID.	Make sure that the correct FTB is connected to the PMM. It may be necessary to reset the Fiber Terminal Block Identifier in the hardware configuration.
FIBER LED is red, ON.	I/O link not established. An FTB has been configured, but is not communicating with the PMM. FTB hardware is incompatible with PMM.	1) Check power to FTB. 2) Check fiber cable connecting FTB and PMM. Contact Technical Support with the fault table content and event queue, captured as soon after the error occurred as possible, from the PMM generating the error. Technical Support personnel may also ask for copies of the logic and/or HWC for the system experiencing the error.
T1 PWR LED is red, ON	The FTB OUT 1-4 Power Detect setting in hardware configuration is Enabled and no power is applied to terminals 1 and 19 of terminal block 1	Check wiring or disable Power External Connection detection in HWC.
T2 PWR LED is red, ON	The FTB OUT 5-8 Power Detect setting in hardware configuration is Enabled and no power is applied to terminals 1 and 19 of terminal block 2.	Check wiring or disable Power External Connection detection in HWC.

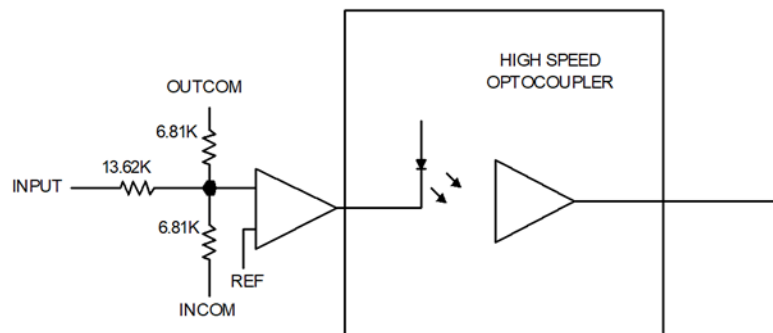
## 3.4 I/O Circuit Specifications

### 3.4.1 PMM Faceplate I/O Circuits

#### PMM Faceplate 24Vdc High-Speed Inputs

Circuit Identifiers	IN1, IN2
IO Type	Optically isolated 24Vdc Source/Sink Inputs
Circuit Type	Source/Sink Source/Sink mode depends on relative polarity of power applied to INCOM and OUTCOM. FP Input Mode in Hardware Configuration must also be set to the correct mode.
Input Impedance	17 kΩ referenced to mid-point of power supply
Maximum Input Voltage	±30.0V (referenced to Input COM)
Logic 0 Threshold	64% Typical, 55% Min percentage of 24Vdc supply
Logic 1 Threshold	65% Typical, 75% Max percentage of 24Vdc supply
Input Filtering	1 μs maximum (configured for Fast Digital Input) 6.5 ms maximum (configured for Digital Input)
Open (Floating) Wire Detect (Optional – enabled in Hardware Configuration)	1.0 ms filtering
Maximum quadrature input frequency (count rate is 4x input frequency)	125 kHz Input Frequency 500 kHz Count Rate Frequency
Quadrature tolerance at maximum input frequency	90 degrees ±45 degrees

Figure 40: PMM 24Vdc High-Speed Input Circuit Diagram

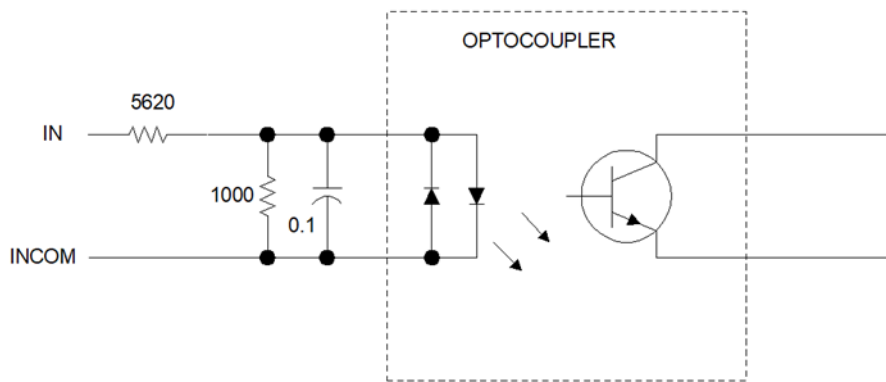


## PMM Faceplate 24Vdc General Purpose Inputs

Circuit Identifiers	IN3, IN4, IN5, IN6, IN7, IN8
IO Type	Optically isolated 24Vdc Source/Sink Inputs
Circuit Type	Source/Sink
Input Impedance	5.62 k $\Omega$ to INCOM
Maximum Input Voltage	$\pm 30.0V$ (referenced to INCOM)
Logic 0 Threshold	$\pm 6.0V$ maximum (referenced to INCOM)
Logic 1 Threshold	$\pm 18.0V$ minimum (referenced to INCOM)
Input Filtering	500 $\mu s$ maximum (configured for Fast Digital Input) 6.5 ms maximum (configured for Digital Input)

**Note:** These inputs use bi-directional optocouplers. For IN3 and IN4 (shared with OUT1 and OUT2 terminals), Source/Sink mode is determined by the relative polarity of power applied to INCOM and OUTCOM. For IN5 – IN8 Source/Sink mode is determined by the relative polarity of the input and INCOM.

**Figure 41: PMM 24Vdc General Purpose Input Circuit Diagram**



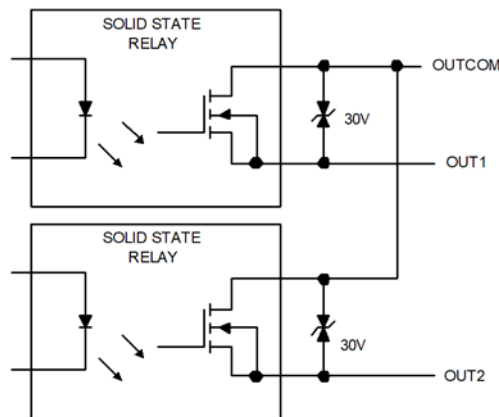


## PMM Faceplate 24Vdc General Purpose Outputs

Circuit Identifier	OUT1, OUT2
IO Type	Dual Optically isolated 24Vdc Source/Sink Outputs both connected to OUTCOM. Source/Sink operation depends on relative polarity of power applied to INCOM and OUTCOM.
Circuit Type	Solid State Relay (SSR) with electronic short circuit protection.
Power Supply Voltage	30.0V max, 18.0V min
Output Current	250 mA continuous (total for both outputs)
Output Voltage Drop	1.75V max at 250 mA output current
Output Leakage Current	1.0 $\mu$ A maximum
Output Turn On Delay	0.9 ms typical, 3.0 ms maximum driver delay
Output Turn Off Delay	0.5 ms typical, 2.0 ms maximum driver delay

**Note:** If a short circuit occurs, the output automatically switches off. The MC\_ModuleReset function block must be used to reset the output short circuit protection.

**Figure 42: PMM 24Vdc General Purpose Output Circuit Diagram**



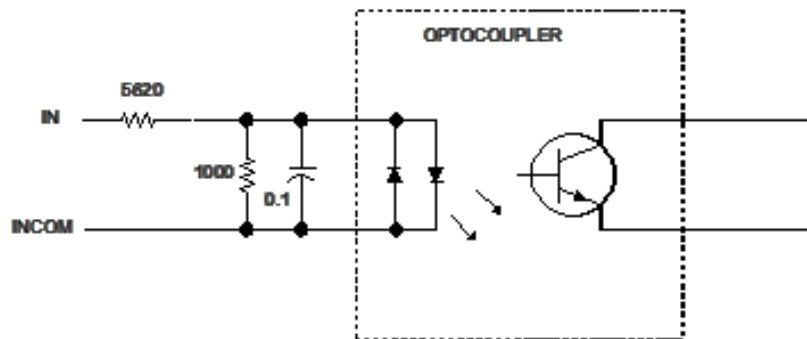
### 3.4.2 FTB I/O Circuits

#### FTB 24Vdc General Purpose Inputs

Circuit Identifiers	I1 – I16
IO Type	Optically isolated 24Vdc Source/Sink Inputs
Circuit Type	Source/Sink
Input Impedance	5.62 kΩ to Input COM @24Vdc
Maximum Input Voltage	±30.0V (referenced to Input COM)
Logic 0 Threshold	±6.0V maximum (referenced to Input COM)
Logic 1 Threshold	±18.0V minimum (referenced to Input COM)
Input Filtering	500 μs maximum (configured for Fast Digital Input) 6.5 ms maximum (configured for Digital Input)

**Note:** These inputs use bi-directional optocouplers and can be turned on with either a positive or negative input with respect to INCOM.

**Figure 43: FTB 24Vdc General Purpose Input Circuit Diagram**



## FTB Differential/Single-Ended 5Vdc Inputs

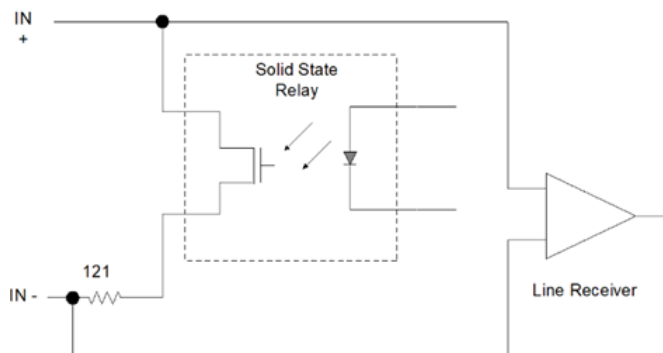
These inputs use a solid-state relay in series with the differential termination resistor. Configuring the input for single-ended operation opens the 121Ω termination resistor. With Single-ended operation, only the + input is connected.

Circuit Identifiers	I17 – I22
IO Type	Differential or single-ended 5Vdc input (configurable)
Circuit Type	RS422 / RS485 Line Receiver with fault detection
Input Impedance (Differential)	121 Ω
Input Impedance (Single-ended)	90kΩ minimum (+ input)
Maximum Input Voltage	±20V differential ±10Vdc common mode
Logic 0 Threshold	-0.475V max differential 1.2V Single-ended (typical)
Logic 1 Threshold	+0.475V max differential 1.5Vdc Single-ended (typical)
Input Filtering	100 ns typical
Fault Detect Filtering	100 μs typical
Maximum quadrature input frequency (count rate is 4x input frequency)	2.50 MHz per channel (differential) 125 kHz per channel (Single-ended)
Quadrature tolerance at maximum input frequency	90 degrees ±45 degrees
Touch Probe Response (at constant velocity)	Minimum Pulse Width: 200 ns (differential) 10 μs (Single-ended)  Position Capture Accuracy: Refer to Appendix A-1.

**Note:** For single-ended mode, use the + input and leave the - input floating.

Use terminal block 0V pins for common mode reference or Single-ended signal return. Inputs can be driven by 5Vdc TTL or CMOS logic.

**Figure 44: FTB Differential/Single-Ended 5Vdc Input Circuit Diagram**

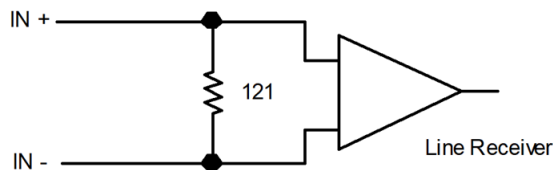


## FTB Differential 5Vdc Inputs

Circuit Identifiers	I23 – I28
IO Type	Differential 5Vdc input
Circuit Type	RS422 / RS485 Line Receiver with fault detection
Input Impedance (Differential)	121 $\Omega$
Input Impedance (Single-ended)	90k $\Omega$ minimum (+ input)
Maximum Input Voltage	$\pm 20V$ differential $\pm 10Vdc$ common mode
Logic 0 Threshold	-0.475V max differential
Logic 1 Threshold	+0.475V max differential
Input Filtering	100ns typical
Fault Detect Filtering	100 $\mu s$ typical
Maximum quadrature input frequency (count rate is 4x input frequency)	2.50MHz (differential)
Quadrature tolerance at maximum input frequency	90 degrees $\pm 45$ degrees
Touch Probe Response (at constant velocity)	Minimum Pulse Width: 200ns (differential) Position Capture Accuracy: Refer to Appendix A-1.

**Note:** Use terminal block 0V pins for common mode reference. Inputs can be driven by 5Vdc TTL or CMOS logic.

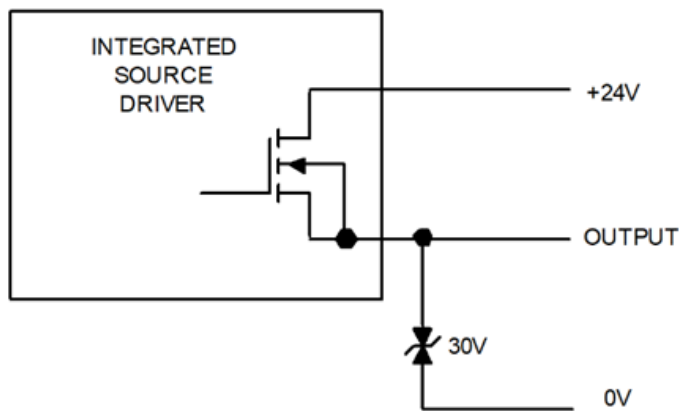
**Figure 45: FTB Differential 5Vdc Input Circuit Diagram**



## FTB 24Vdc General Purpose Outputs

Circuit Identifiers	Q1 – Q8
IO Type	Optically isolated 24Vdc source outputs, four per group
Circuit Type	Source (Open Drain pullup to +24Vdc output power supply)
Power Supply Voltage	30.0V maximum, 18.0V minimum
Output Current	1.5 A continuous per point, 4 amps continuous total per group
Output Voltage Drop	0.375V maximum at 1.5 amp output current
Short Circuit Detection Voltage	2.0V minimum, 4.0V maximum referenced to +24Vdc of output power supply
Open Load Detection Voltage	2.0V minimum, 4.0V maximum referenced to 0V of output power supply
Output Leakage Current	1.4 mA maximum with 24Vdc across output
Turn-Off / Turn-On Delay	100 $\mu$ s max driver delay

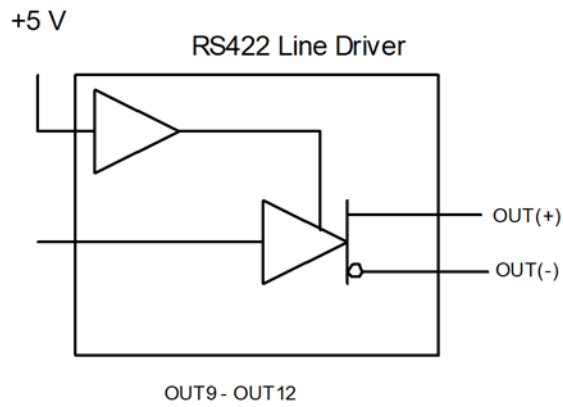
**Figure 46: FTB 24Vdc General Purpose Output Circuit Diagram**



## FTB 5Vdc Differential Outputs

Circuit Identifiers	Q9 – Q12
IO Type	Differential 5Vdc output
Circuit Type	RS422 Line Driver with short circuit protection
Power Supply Voltage	Internal +5Vdc
Output Current	+48mA / -20mA maximum sink/source
Differential Output Voltage	2.0V minimum with 100Ω differential load
Turn On/Turn Off Delay	15ns maximum driver delay

**Figure 47: FTB 5Vdc Differential Output Circuit Diagram**

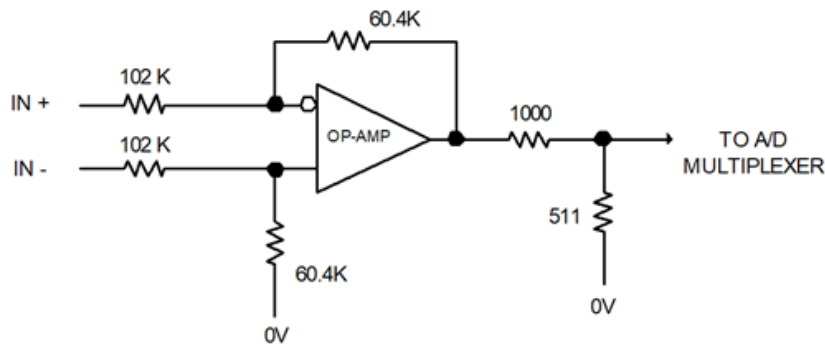


## FTB Differential $\pm 10\text{Vdc}$ Analog Inputs

Circuit Identifiers	AI1, AI2
IO Type	Differential $\pm 10.0\text{V}$ analog inputs
Circuit Type	Differential Analog input
Input Impedance (Differential)	204 k $\Omega$
Input Impedance (Common mode)	102 k $\Omega$ common mode with respect to FTB connector 0V
Maximum Input Voltage	$\pm 15\text{V}$ common mode $\pm 20\text{V}$ differential
Resolution	14 bits
Linearity	12 bits
Input Offset	$\pm 5.0\text{ mV}$ max
Gain Factor	10.0V = 10.000 (reported to the RX3i as floating-point data)
Gain Accuracy <sup>2</sup>	$\pm 1.0\%$
Update Rate	1 kHz

**Note:** Use terminal block 0V pin for common mode reference.

**Figure 48: FTB Differential  $\pm 10\text{Vdc}$  Analog Input Circuit Diagram**



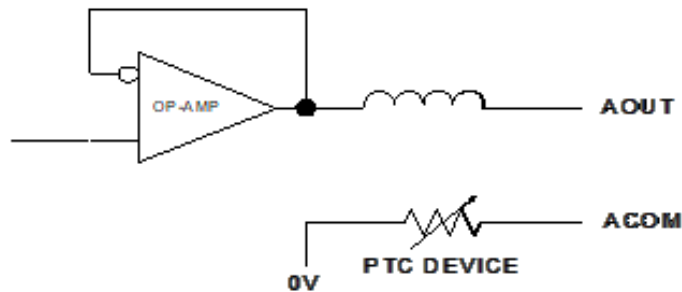
<sup>2</sup> In the presence of severe RF interference, the accuracy may be degraded to  $\pm 5.0\%$ .

## FTB Single-Ended $\pm 10\text{Vdc}$ Analog Outputs

Circuit Identifiers	AO1, AO2
IO Type	Single-ended Analog Output
Circuit Type	Op Amp output
Load Impedance	2k $\Omega$ minimum
Output Current	5 mA maximum
Resolution	12 bits
Linearity	12 bits
Output Offset Voltage	$\pm 5.0$ mV max
Gain Factor	10.000 floating-point data from the RX3i = 10.0V output
Gain Accuracy <sup>3</sup>	$\pm 2.0$ %
Update Rate	1 kHz

**Note:** Since this is a Single-ended output, it should normally drive a user device with a differential input to prevent common mode noise problems. The positive differential input should be connected to AO +10Vdc and the negative differential input to AO COM.

**Figure 49: FTB Single -Ended  $\pm 10\text{Vdc}$  Analog Output Circuit Diagram**



## FTB +5Vdc Power

This output is intended to power external devices such as Incremental Quadrature Encoders.

Circuit Identifier	+5Vdc (OUT)
I/O Type	+5Vdc Encoder Power
Circuit Type	+5Vdc Power with Electronic Short Circuit Protection
Output Voltage	4.70Vdc to 5.20Vdc at 0.5 A
Output Current	0.5A maximum (Per terminal block and total for both FTB terminal blocks.)



### 3.5 EtherCAT Command Interface Cable

Shielded Ethernet cables are used to interface up to four servo drives to the PMM motion controller. Shielded Ethernet cables are available in various lengths. The line Network topology is supported by the PMM345. The line network topology supports the critical EtherCAT distributed clock functionality required for high performance motion control. At this time, the ring topology is not supported because it does not support distributed clocks. Reference the EtherCAT network specifications for additional details regarding the distributed clock functionality.

### 3.6 FTB to PMM Connection

Any FSSB cable listed in Section 3.2.2 Terminal Header and Cable Options can be used for the fiber optic connection between the PMM and the FTB.

Figure 50: Interconnecting PMM and FTB



## 3.7 Grounding the PACMotion System

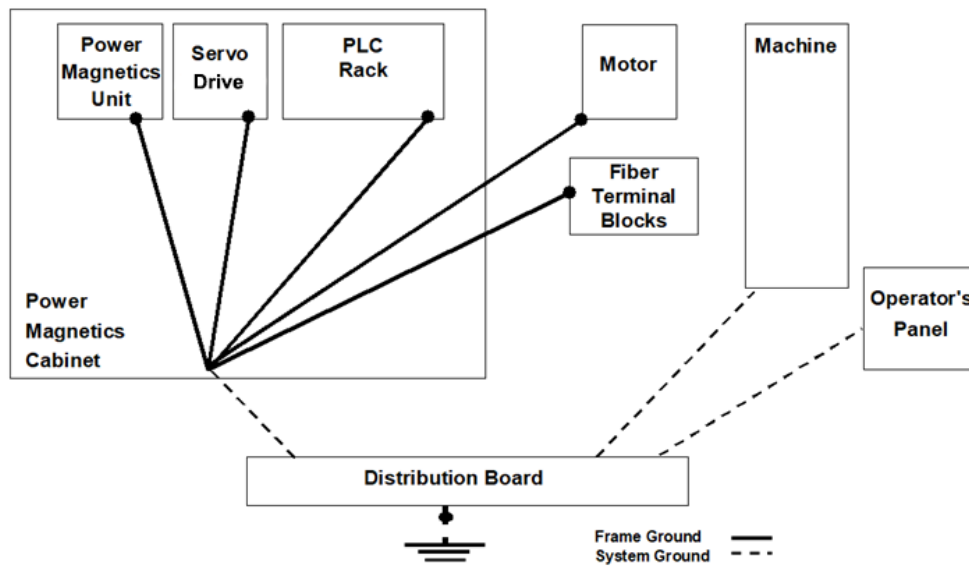
The motion system must be properly grounded. Problems such as erratic system operation may occur if this practice is not followed.

A sample grounding system is shown below. Guidelines for grounding and noise reduction are provided below and on the following pages.

**Note:** The metal collar around the RJ-45 port has frame ground potential. Long cable runs or cabling from an outside panel could be cause for ground loops unless a LAN shield is isolated at one end.

### System Grounding Example

Figure 51: Example of System Grounding



### Grounding Systems

Frame Ground	Used for safety and to suppress external and internal noises. In a frame ground system, the frames, unit cases, panels, and shields for the interface cables between the units are connected.
System Ground	Used to connect the frame ground systems connected between devices or units with the ground.

## Grounding and Noise Reduction Guidelines

### General Guidelines

- The servo drive ground connections, power earth (PE) connections, and motor frame ground connections should be wired to conform to local electrical wiring regulations. For installations that conform to CE Mark directives, refer to Section 3.7.2, I/O Cable Grounding.
- AC Main PE Ground is supplied in accordance to local code practices and may vary, depending on AC power distribution in the facility. In general, the PE ground should be referenced to an earth ground and not indicate common mode voltage to the instrumentation earth ground.
- If an FTB is used, the ¼ inch blade terminal at the bottom of the block must be connected to the panel frame ground (Figure 53).
- For installations that must meet IEC electrical noise immunity standards, the RX3i system that contains the PMM must be mounted in a metal enclosure or the equivalent. All surfaces of the enclosure must be adequately grounded to adjacent surfaces to provide electrical conductivity. Wiring external to the enclosure must be routed in metal conduit or the equivalent. Using shielded cables and power line filtering is equivalent to using metal conduit.
- For additional information, refer to Product Certifications and Installation Guidelines for Conformance in the PACSystems RX3i System Manual, GFK-2314.

### Guidelines for Servo Drive Connections

- The Motor Control Center (MCC) relay used to switch the three-phase AC main power to the servo drive should have an appropriate noise (spark) arrester on its drive coil.
- The 24Vdc power supply used to supply the logic power to the servo drives should be a regulated supply free of excessive noise. If possible examine the DC voltage with an oscilloscope for noise. If a 24Vdc motor-mounted holding brake is used, it should not use the same power supply as the control logic power.
- Servo drive Chassis Ground must be referenced to earth ground with a class 3 (100Ω or less) system ground. Use an ohmmeter to measure the resistance from the servo drive frame to a known earth ground rod or grid. A tapped and threaded hole is provided on the servo drive frame for this purpose.
- An AC line filter is recommended to suppress high frequency line noise on the servo drive main power lines. When an isolation transformer is used to convert AC main power to servo drive input power levels, the AC line filter is not required. Emerson supplies a 3-phase line filter sized for 5.4kW or 10.5kW especially for this purpose. This filtered AC main power should not be shared with other equipment in the panel, especially with devices such as inverter drives or motor starters that have high power consumption.

The Motor Feedback cable should have a Z44B295864-001 Grounding Bar and one ZA99L-0035-001 Grounding Clamp per axis installed near the servo drive. For installation details, refer to in Section 3.7.2.

**Note:** For more in-depth information, please consult *PACMotion PSD Installation and User Manual*, GFK-3168.

## Guidelines for Motor Connections

- The motor power connector servo motor frame ground connection should always be installed.
- The motor frame must be referenced to earth ground with a class 3 (100Ω or less) system ground. Use an ohmmeter to measure the resistance from the servomotor frame to a known earth ground rod or grid. The frame-to-ground resistance should be within 1 to 2 Ω.
- In a high noise environment, installing a ground wire on the motor frame and routing it directly to the nearest available earth ground can improve noise immunity. Some servo motors have a tapped hole on the frame or a blind hole that can be tapped. For smaller motors, connect to the motor mounting bolts.
- The Motor Power cable should not be a shielded cable. If a custom-built cable with shield has been used for motor power, clip off the shield connection at both ends of the cable. If a shield is attached, especially at the motor end, it acts as an antenna, which couples noise into the encoder.
- In a high noise environment, installing a ferrous bead on the feedback cable within a short distance of the servo drive connector can also improve noise immunity.

Input power and signal lines must be separated. Group A signals (Servo drive main AC power, Motor Power Cable and MCC drive coil) signals must be separated from Group B signals (Motor Feedback cable) by at least a 10cm distance. Refer to in Section 3.7.2 for recommendations.

**Note:** For more in-depth information, please consult *PACMotion PSR Installation and User Guide, GFK-3169*.

## Guidelines for System Connections

- The system ground connection cable must be integrated with the AC power line such that power cannot be supplied if the ground wire is disconnected.
- The grounding resistance of the system ground should be 100Ω or less (class 3 grounding).
- The system ground cable must have sufficient cross-sectional area to safely carry the accidental current flow into the system ground when an accident such as a short circuit occurs. Typically, it must have at minimum the cross-sectional area of the AC power cable.

## Additional Recommendations to Avoid Noise Issues on External Quadrature Encoder Input Channels

If noise issues persist, consider the following solutions:

Use an encoder with differential outputs. The encoder should be connected using shielded twisted pair cable. Use a twisted pair for each encoder channel and an additional twisted pair for encoder power and 0V. Refer to the connection diagram in Section 3.2.6, Typical Single-Ended Encoder Connection for FTB. Ground the cable shield, first one end, or if necessary, both ends.

- If an encoder with single-ended outputs must be used, it should be connected using shielded twisted pair cable. Use a twisted pair for each encoder channel with one wire in each pair connected to 0v at each end. Use an additional twisted pair for encoder power and 0v. Refer to the connection diagram in Section 3.2.6, Typical Single-Ended Encoder Connection for FTB. Ground the cable shield, first one end, or if necessary, both ends.

---

**Note:** Single-ended 5Vdc encoders can only be connected to FTB inputs I17–I22. The input mode must be configured as Single-Ended.

---

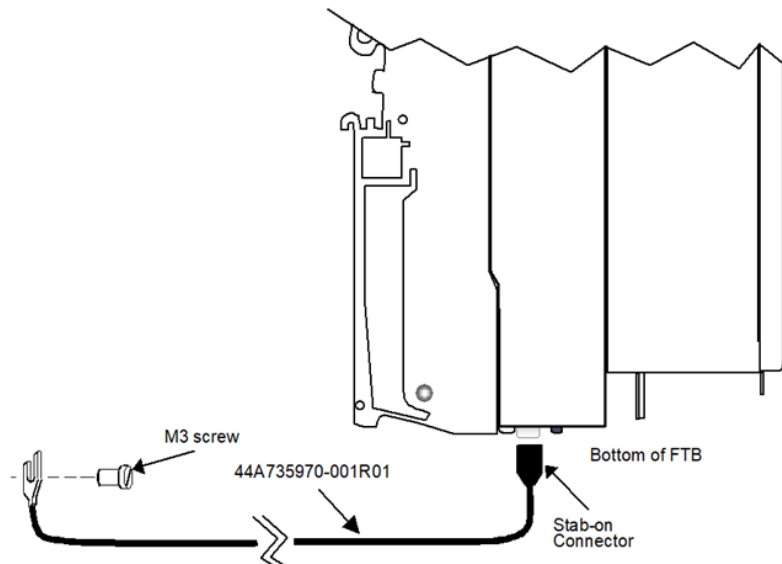
- Use additional grounding or isolation of signals and power sources (provide an opto-isolator).

### 3.7.1 Fiber Terminal Block I/O Shield Ground Connection

The faceplate shield on the FTB must be connected to frame ground. The connection from the FTB to frame ground can be made using the green ground wire (part number 44A735970-001R01) provided with the FTB. The wire has a stab-on connector on one end for connection to a ¼ inch terminal located at the bottom of the FTB between the two I/O terminals.

---

**Figure 52: FTB Shield Ground Connection**



### 3.7.2 I/O Cable Grounding

Properly routing signal cables, servo drive power cables and motor power cables along with installation of proper Class 3 grounding will insure reliable operation. Typically, Class 3 grounding specifies a ground conductor of a minimum wire diameter larger than the power input wire diameter, connected via a maximum 100Ω resistance to an earth ground. Consult local electrical codes and install in conformance to local regulations.

The specifications for completing digital drive installation and wiring, including driving grounding are described in GFK-3168, *PACMotion PSD Installation and User Manual*.

When routing signal lines, amplifier input power line and motor power line, the signal lines must be separated from the power lines. The following table indicates how to separate the cables.

## Separation of Signal Lines

Do not tie Group A and B signals together with cable ties or wraps at any point. An alternative is to separate these two groups by means of a grounded metal (steel) plate.

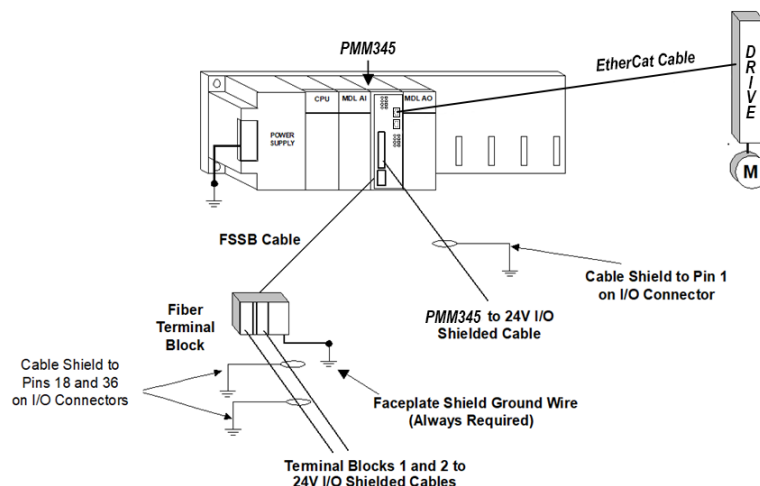
Group	Signal	Action
A	Amplifier input power Motor Power	Separate a minimum 10cm from group “B” signals by bundling separately or use electromagnetic shielding (grounded steel plate). Use noise protector for MCC.
B	FTB to 5Vdc I/O Terminal cable PMM or FTB to 24Vdc I/O Terminal cable Encoder feedback cable	Separate a minimum 10cm from group “A” signals by bundling separately or use electromagnetic shielding (grounded steel plate). Use all required individual cable shield grounds and grounding bar connections.

## Digital Servo Drive Signal Cable Grounding

The signal cables used with the PMM and FTB contain shields that must be properly grounded to ensure reliable operation. The illustration below shows cable grounding recommendations for typical installations. The following points should be considered:

- The PMM faceplate ground wire must be connected to a reliable panel frame ground.
- The PMM I/O connector provides ground via pin 1. A short ground wire must be connected from each of these terminals to a reliable panel ground.
- If an FTB is used, the FTB faceplate ground wire must be connected to a reliable panel frame ground.
- Each terminal block in the FTB provides ground via pins 18 and 36. A short ground wire must be connected from each of these terminals to a reliable panel ground.
- Provide shield grounding for the Digital Servo amplifier encoder feedback cable as described on the next page.

**Figure 53: I/O Cable Grounding Diagram**

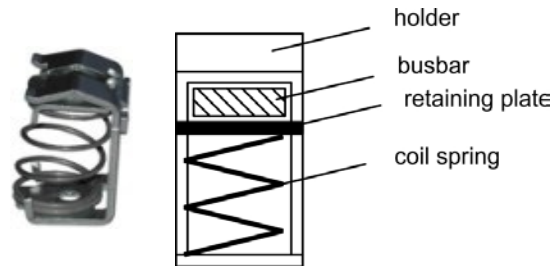


## Grounding Bar and Cable Installation

**Note:** External Shielding Busbar sold separately.

In special cases, the cable shields can be routed to an additional busbar via shield clamps. The following shield clamp is recommended:

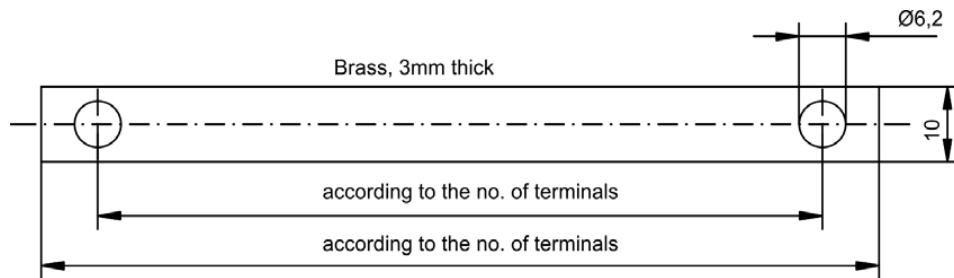
**Figure 54: External Shield Busbar**



A possible scenario for setting up a busbar for the above shield clamps is described below.

1. Cut a busbar of the required length from a brass rail (cross-section 10x3 mm) and drill holes in it as indicated. All shield clamps required must fit between the drill holes.

**Figure 55: Cutting the Busbar**



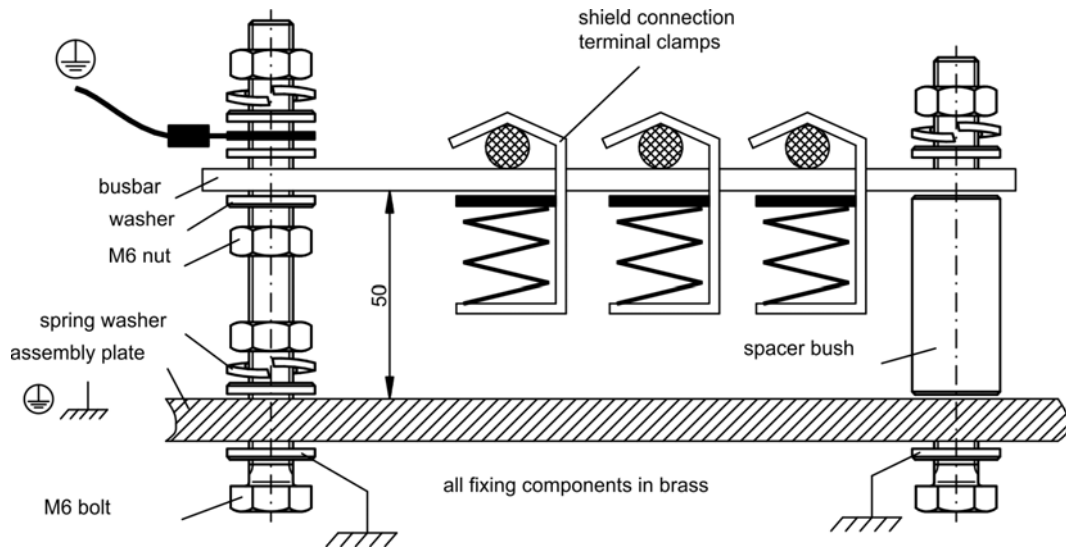
2. Squeeze together the coil spring and the supporting plate and push the busbar through the opening in the holder.

### CAUTION

Risk of injury is present due to the spring force of the coil spring. Use pliers.

3. Mount the busbar with the shield clamps fitted on the assembly plate. Use either metal spacer bushes or screws with nuts and accessories to maintain a spacing of 50 mm. Earth the busbar using a single conductor with a cross-section of at least 2.5 mm<sup>2</sup>.

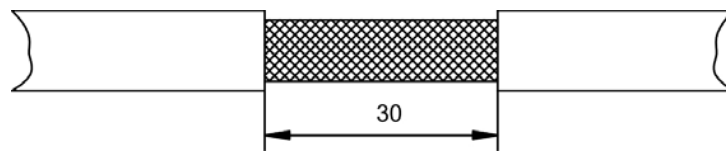
Figure 56: Mounting the Busbar



- Strip the external cable sheath to a length of approximately 30 mm, taking care not to damage the braided shield. Push the shield clamp up and route the cable to it via the busbar.

**Note:** Make sure good contact exists between the shield clamp and the braided shield.

Figure 57: Cable Sheath





# Section 4 Configuration

This chapter describes configuration details necessary to set up the PACMotion Multi-Axis Motion Controller (PMM) for a specific application. The PMM is configured using PME Logic Developer software version 9.9 or later.

Topics covered:

- Section 4.1 Connecting the Programmer to the RX3i
- Section 4.2 Adding a PMM to the Hardware Configuration
- Section 4.3 Configuring PMM Parameters

## 4.1 Connecting the Programmer to the RX3i

All PMM programming is done through the configuration/programming software interface, yielding a single point of programming for the module. For more information, please refer to the PACSystems RX3i and RSTi-EP CPU Reference Manual, GFK-2222.

The RX3i programming environment has two communications options. You can connect the programmer directly to one of the CPU's COM ports, or you can communicate with the CPU through the Ethernet network.

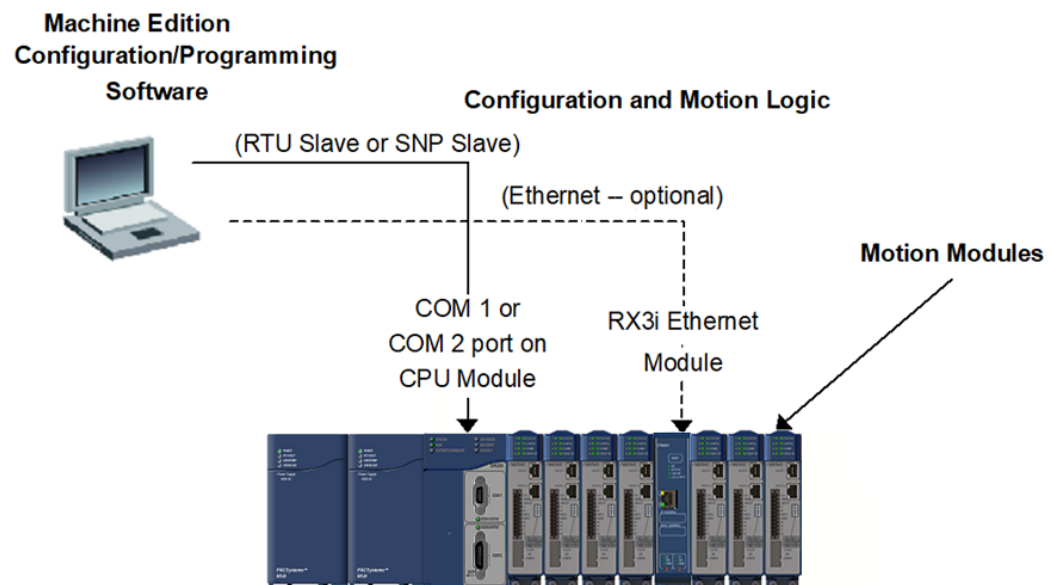
---

**Note:** An IP address must be set in the RX3i before an Ethernet connection can be established. For details, refer to PACSystems RX3i and RSTi-EP TCP/IP Ethernet Communications User Manual, GFK-2224.

---

### 4.1.1 PMM345 Programmer Connection

**Figure 58: Methods of Connecting PC Hosting PME to RX3i**



## 4.2 Adding a PMM to the Hardware Configuration

The hardware configuration defines the type and location of each module present in the RX3i racks. This is done by completing setup screens that represent the modules in a backplane, and then saving the information to a configuration file, which is downloaded to the RX3i CPU.

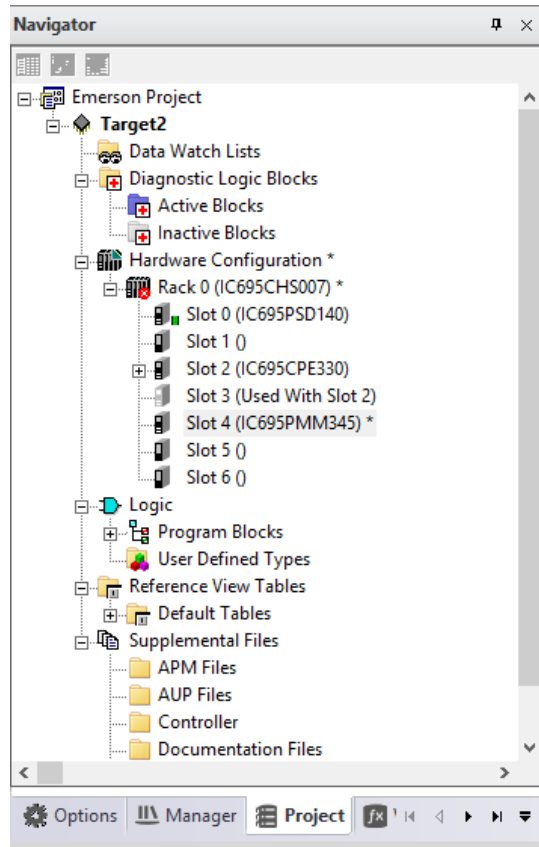
To configure a PMM using PME software:

1. Create or open a project containing an RX3i target.
2. In the Navigator window, expand the Hardware Configuration.
3. If necessary, replace the power supply and/or CPU with the models that will be used in your application. To replace a module, right click and choose Replace Module.
4. Right click the slot where the PMM is to be configured and choose Add Module (choose Replace Module if a module is already configured in the slot).
5. In the Module Catalog, select the Motion tab, choose the PMM345 and click OK.

This operation adds the PMM345 to the RX3i rack and displays the PMM345 configuration screens that allow you to customize the PMM345 to your particular application.

---

**Figure 59: Adding PMM to RX3i HWC**



## 4.3 Configuring PMM Parameters

The PAC Machine Edition hardware configuration tool presents the PMM hardware configuration parameters in a tabular format. For details concerning the operation of the configuration software, please consult the online help or PAC Machine Edition Logic Developer Getting Started, GFK-1918.

**Figure 60: PME Configuration Screen Presenting Multiple Tabs**

Settings		FP I/O	FTB Inputs	FTB Outputs	I/O Interrupts	Axis 1	Axis 2	Axis 3	Axis 4	Advanced	Power Consumption
Parameters		Values									
I/O Status Data Reference	%I00113										
I/O Status Data Length	32										
Module	M4_1										
Cam Library Management	Automatic Mode										
Log Messages in I/O Fault Table	Errors Only										
<i>Number of Axes</i>	4										
<i>Number of Virtual Axes</i>	0										
Axis 1	M4_Axis1_1										
<i>Axis 1 Mode</i>	PM EtherCAT Servo										
Axis 2	M4_Axis2_1										
<i>Axis 2 Mode</i>	PM EtherCAT Servo										
Axis 3	M4_Axis3_1										
<i>Axis 3 Mode</i>	PM EtherCAT Servo										
Axis 4	M4_Axis4_1										
<i>Axis 4 Mode</i>	PM EtherCAT Servo										
I/O Scan Set	1										

Parameters	Function	Page
Settings	Contains the status data reference assignment, axis selection and other module-level data.	77
I/O Function Assignments		80
FP Inputs	Used for configuring the PMM's faceplate inputs and outputs.	80
FTB Inputs	Used for configuring the Fiber Terminal Block I/O (FTB) inputs.	82
FTB Outputs	Used for configuring the FTB outputs.	85
I/O Interrupts	Used to specify interrupts that will trigger execution of a logic block in the CPU.	91
Axis Configuration	Used for configuring axis operational characteristics. The Number of Axes parameter on the Settings tab determines how many axis tabs are displayed.	93
Axis 1—Axis 4 Digital Servo Modes	PM EtherCAT Servo (default) Axis Mode selects digital servo mode.	98
Axis 1—Axis 4, Analog Servo Modes	Analog Servo Velocity and Analog Servo Torque modes are available.	105
Axis 5	Displayed when Number of Virtual Axes on the Settings tab is 1.	112
Advanced	Contains advanced tuning data for axes 1 — 4.	133
Power Consumption	Lists PMM power required from backplane supply.	133
Terminals	Appears when Variable Mode in the module properties is set to True. Allows you to configure the reference addresses used by the module for I/O status.	134

---

**Note:** Application logic can read and modify many configuration parameters using the *MC\_ReadParameter* and *MC\_WriteParameter* function blocks. For a list of parameter numbers refer to *Axis Parameter Number Index* in Section 8.1.1.

---

## 4.3.1 Settings

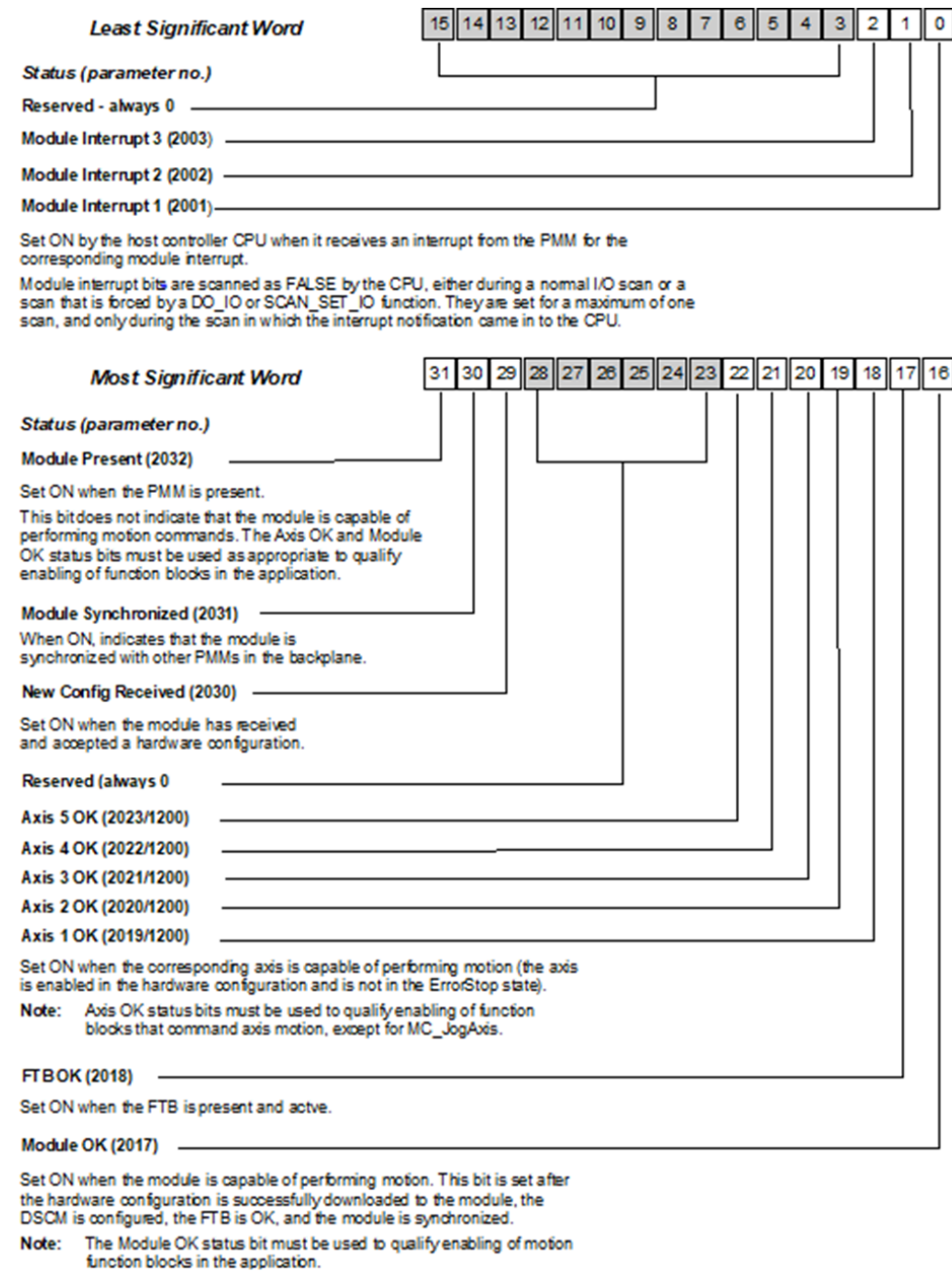
The Settings tab contains configuration information that allows you to define basic module operation. The configuration parameters in the Settings tab are described in the table below.

Configuration Parameter	Description
I/O Status Data Reference	The starting reference address used by the CPU to store the I/O status data received from the PMM. For bit assignments, refer to Section 4.3.2, PMM Status Data. If the module's Variable Mode property is set to True, this parameter is removed from the Settings tab. Instead, references are defined as I/O variables on the Terminals tab. Valid memory areas: %I, %T, %M Default memory area: %I
I/O Status Data Length	(Read-only) The number of bits used to store the I/O status data received from the PMM. If the module's Variable Mode property is set to True, this parameter is removed from the Settings tab. Instead, references are defined as I/O variables on the Terminals tab.
Module	Assigns a Module Reference ID, a symbolic variable of type MODULE_REF, used by some Motion function blocks to indicate the PMM that is to perform the action. Names must consist of A-Z, a-z, 0-9, or _ and must start with a letter or \$. No consecutive underlines are allowed. 32 characters max. This parameter is required. Default: Mx, where x is the module number
CAM Library Management	The CAM library occupies an area of fixed size in PMM memory. In automatic mode, the library is managed without user interaction. In this mode, the oldest profiles are deleted as needed to allow additional profiles to be stored into the library. In manual mode, you are responsible for maintaining the library. For details on the Motion Function blocks used to manage the CAM library, refer to the Programming Reference (volume II of this manual). Choices: Automatic mode, Manual Mode Default: Automatic Mode
Log Messages in I/O Fault Table	Determines which events on the PMM result in I/O faults in the fault table. If Errors Only is selected, error events in the event queue result in I/O faults to the fault table. If Errors & Warnings is selected, both error and warning events are logged in the I/O fault table. <hr/> <b>Note:</b> <i>Error and Warning messages are recorded in the PMM's event queue, regardless of whether they are logged in the fault table.</i> <hr/> Choices: None, Errors Only, Errors & Warnings Default: Errors Only
Number of Axes	Selects the number of axes, excluding virtual axes, to be configured on the PMM. Choices: 1, 2, 3, or 4 Default: 4
Number of Virtual Axes	Specifies whether the virtual axis (Axis 5) is configured on the PMM. Choices: 1, 0 Default: 0

Configuration Parameter	Description
Axis 1, Axis 2, Axis 3, Axis 4, Axis 5	<p>Assigns an axis name, which is a symbolic variable of type AXIS_REF, to the axis. Some Motion function blocks use the AXIS_REF variable to indicate the axis that is to perform an action. Each axis name must be unique to the Target, consist of A-Z, a-z, 0-9, or _ and start with a letter or \$ and have a maximum of 32 characters. No consecutive underlines are allowed. Every axis configured must have a name.</p> <p>Default: Mx_Axisy, where x is the module number and y is the axis number.</p>
Axis 1 Mode, Axis 2 Mode, Axis 3 Mode, Axis 4 Mode	<p>Selects the axis mode.</p> <p>PM EtherCAT Servo selects the Ethernet command interface to a Emerson EtherCAT servo drive.</p> <p>Analog Servo Velocity Mode selects a <math>\pm 10</math> Vdc analog velocity command interface to an analog servo drive via an FTB analog output.</p> <p>Analog Servo Torque Mode selects a <math>\pm 10</math> Vdc analog torque command interface to an analog servo drive via an FTB analog output.</p> <p>Disabled temporarily removes the axis from the configuration.</p> <p>Default: PM EtherCAT Servo</p> <hr/> <p><b>Note:</b> <i>Axes 1 through 4 must be enabled in sequential order, starting with Axis 1.</i></p> <p><i>A maximum of two analog axes can be configured on a PMM.</i></p> <p><i>The analog axis modes require PMM version 1.50 or higher.</i></p> <p><i>The allowed order for configuring axes is:</i></p> <p><i>Any real PM EtherCAT Servo (Drive Type not equal to Synthetic)</i></p> <p><i>Any Analog Servos (Velocity Mode or Torque Mode)</i></p> <p><i>Any Synthetic Axes (PM EtherCAT Servo with Drive Type equal to Synthetic).</i></p> <hr/>
Axis 5 Mode	<p>In Virtual mode, Axis 5 can act as master position source for other axes by executing a subset of motion function blocks, by reading an external quadrature encoder, or by doing both at the same time.</p> <p>Choices: Disabled, Virtual</p> <p>Default: Virtual</p>
I/O Scan Set	<p>Assigns the module I/O status data to a scan set defined in the CPU configuration. The scan set determines how often the RX3i polls the data. For more information, refer to the PACSystems RX3i and RSTi-EP CPU Reference Manual, GFK-2222.</p> <p>Range: 1 to 32</p> <p>Default: 1 (Read status every I/O sweep.)</p>

## 4.3.2 PMM Status Data

Figure 61: Bit Definitions PMM Status Data





### 4.3.3 I/O Function Assignments

Basic I/O functions are available on the PMM faceplate (FP), and basic and advanced I/O functions are available on the FTB.

Explanations of these functional choices are provided in PMM Faceplate and FTB I/O Function Descriptions in Section 4.3.3.

The PMM Faceplate I/O Functions Summary and FTB I/O Functions Summary sections below identify the functions that can be assigned to the I/O points on the PMM faceplate and on the FTB.

---

**Note:** *With the exception of Fast Digital Input, Digital Input and Digital Output, each function can be applied to only one faceplate or FTB point at a time.*

*If a faceplate input is selected as an I/O Interrupt Source, neither the Digital Output function nor an Axis Encoder input can be assigned to that input.*

---

#### FP Inputs

The FP Inputs configuration tab allows you to select a function for each digital I/O connector on the PMM faceplate (FP).

Configuration Parameter	Description	Parameter Number
FP IN1	Assigns the function of the FP input, IN1.  <b>Note:</b> <i>If IN2 is configured as Axis 5 Encoder Channel B, IN1 must be configured as Axis 5 Channel A.</i>	3000 (Read)
FP IN1 Input Ref	Symbolic variable of type INPUT_REF, associated with this FP input. Default: Mx_FP_INx	NA
FP IN1 Open Wire Detect	Enables open wire fault detection for IN1. Choices: Disabled, Enabled Default: Disabled	NA
FP IN2	Assigns the function of the FP input, IN2.  <b>Note:</b> <i>If FP IN1 is configured as Axis 5 Encoder Channel A, FP IN2 must be configured as Axis 5 Channel B.</i>	3001 (Read)
FP IN2 Input Ref	Symbolic variable of type INPUT_REF, associated with this FP input. Default: Mx_FP_INx	NA
FP IN2 Open Wire Detect	Enables open wire fault detection for IN2. Choices: Disabled, Enabled Default: Disabled	NA
FP IN3/OUT1	Assigns the function of the FP IN3/OUT1 connector. Default: Axis 1 Home Switch	FP IN3: 3002 (Read) FP OUT1: 3128 (Read/Write)

Configuration Parameter	Description	Parameter Number
FP IN3 Input/Output Ref	Symbolic variable of type INPUT_REF, associated with this FP input. Default: Mx_FP_INx or Mx_FP_OUTx	NA
FP IN4/OUT2	Assigns the function of the FP IN4/OUT2 connector. Default: Axis 2 Home Switch	FP IN4: 3003 (Read) FP OUT2: 3129 (Read/Write)
FP IN4 Input/Output Ref	Symbolic variable of type INPUT_REF, associated with this FP input. Default: Mx_FP_INx or Mx_FP_OUTx	NA
FP IN5	Assigns the function of the FP input, IN5. Default: Axis 1 Overtravel +	3004 (Read)
FP IN5 Input Ref	Symbolic variable of type INPUT_REF, associated with this FP input. Default: Mx_FP_INx	NA
FP IN6	Assigns the function of the FP input, IN6. Default: Axis 1 Overtravel -	3005 (Read)
FP IN6 Input Ref	Symbolic variable of type INPUT_REF, associated with this FP input. Default: Mx_FP_INx	NA
FP IN7	Assigns the function of the FP input IN7. Default: Axis 2 Overtravel +	3006 (Read)
FP IN7 Input Ref	Symbolic variable of type INPUT_REF, associated with this FP input. Default: Mx_FP_INx	NA
FP IN8	Assigns the function of the FP input IN8. Default: Axis 2 Overtravel -	3007 (Read)
FP IN8 Input Ref	Symbolic variable of type INPUT_REF, associated with this FP input. Default: Mx_FP_INx	NA
Touch Probe Detection	Appears only if an axis Touch Probe input is selected. Determines how the Touch Probe input captures the axis Actual Position. Default: Positive Edge Trigger	NA
FP Inputs Mode	Selects whether the inputs FP IN1 and FP IN2 will operate with positive logic (Source) or negative logic (Sink). Default: Source	NA
FP Outputs Default	Selects the default output mode for all FP outputs. If Force Off is selected, outputs will go to 0 when the CPU is in Stop – IO Enabled mode. If Hold Last State is selected, outputs retain their last programmed value when the CPU is in Stop – IO Enabled or Stop – IO Disabled mode.  <b>Note:</b> <i>When communication with the CPU is lost, FTB outputs are set to 0.</i>  Default: Force Off	NA

## FTB Inputs

The FTB Inputs configuration tab allows you to select an input function for each digital I/O connector on the Fiber Terminal Block I/O. Four differential 5Vdc outputs can also be selected on this tab.

For a summary of functions that can be assigned to the FTB I/O points, refer to the FTB I/O Functions Summary Section for explanations of these functions, refer to PMM Faceplate and FTB I/O Function Descriptions4.3.3 I/O Function Assignments.

**Note:** *With the exception of Fast Digital Input, Digital Input and Digital Output, each function can be assigned to only one faceplate or FTB connector at a time.*

*If an FTB input is selected as an I/O Interrupt Source, an Axis Encoder input cannot be assigned to that input.*

Configuration Parameter	Description	Parameter Number
Fiber Terminal Block Setting	Specifies whether an FTB is to be used with the PMM being configured. Should be set to Enabled when an FTB will be connected to the PMM. If one or more axes are configured as Analog Servo Velocity Mode or Analog Servo Torque Mode, this parameter is automatically set to Enabled. Choices: Disabled, Enabled Default: Disabled	NA
Fiber Terminal Block Identification Mode	Selects the mode used to identify the FTB/PMM communications link between the FTB and the PMM. Choices: Auto: A unique identifier based on the PMM rack and slot position is used to identify the PMM/FTB link pair to prevent connection of an FTB to the wrong PMM. User Defined: Allows you to specify a four-character Fiber Terminal Block Identifier for the PMM/FTB pair. Disabled: No validation of link partner identification is performed on link startup. Default: Auto	NA
Fiber Terminal Block Identifier	When the Fiber Terminal Block Identification Mode is set to User Defined, you can define a four-character name for the communications link between the PMM and the FTB. You can alias two or more FTBs by assigning the same identifier to multiple PMM/FTB pairs. The name is stored in the FTB in non-volatile memory. To reset the FTB identifier so that it will communicate with any PMM, set the identification mode to User Defined and the identifier to NONE. Default: NONE	NA
FTB IN1 — FTB IN16	Selects the input type for the 24Vdc inputs I1 through I16. Default: Digital Input	3032 —3047 (Read)
FTB IN1 — FTB IN16	Selects the input type for the 24Vdc inputs I1 through I16. Default: Digital Input	3032 —3047 (Read)
FTB IN1 Input Ref — FTB IN16 Input Ref	Symbolic variable of type INPUT_REF, associated with this FTB input. Default: Mx_FTb_INx	NA

Configuration Parameter	Description	Parameter Number
FTB IN17	Selects the input type for the 5Vdc differential input. Default: Fast Digital Input	3048 (Read)
FTB IN17 Input Ref – FTB IN26 Input Ref	Symbolic variable of type INPUT_REF, associated with this FTB input. Default: Mx_FTb_INx	NA
FTB IN17 Mode – FTB IN22 Mode	Specifies the communications connection method for the 5Vdc inputs FTB IN17 through FTB IN22. Choices: Differential, Single-ended Default: Differential	NA
FTB IN17 – FTB IN23 Fault Detect	Enables or disables detection of an open wire fault or loss of encoder power for the 5Vdc inputs FTB IN17 through FTB IN23. Applies only if Mode is Differential. Choices: Enabled, Disabled Default: Enabled	NA
FTB IN18	Selects the input type for the 5Vdc differential input. Default: Fast Digital Input	3049 (Read)
FTB IN19	Selects the input type for the 5Vdc differential input. Default: Fast Digital Input	3050 (Read)
FTB IN20	Selects the input type for the 5Vdc differential input. Default: Fast Digital Input	3051 (Read)
FTB IN21	Selects the input type for the 5Vdc differential input. Default: Fast Digital Input	3052 (Read)
FTB IN22	Selects the input type for the 5Vdc differential input. Default: Fast Digital Input	3053 (Read)
FTB IN23	Selects the input type for the 5Vdc differential input. Default: Fast Digital Input	3054 (Read)
FTB IN24/Out9	Selects Fast Digital Output or an input type for the 5Vdc differential input. Default: Fast Digital Input	IN24: 3055 (Read) OUT9: 3168 (Read/Write)
FTB IN24/Out9 – FTB IN28/Out12 Fault Detect	Enables or disables detection of an open wire fault or loss of encoder power for the 5Vdc inputs FTB IN24 through FTB IN28. Choices: Enabled, Disabled Default: Enabled	NA
FTB IN24 Input/Output Ref – FTB IN28 Input/Output Ref	Symbolic variable of type INPUT_REF, associated with this FTB input. Default: Mx_FTb_INx	NA
FTB IN25/Out10	Selects Fast Digital Output or an input type for the 5Vdc differential input. Default: Fast Digital Input	IN25: 3056 (Read) OUT10: 3169 (Read/Write)
FTB IN26	Selects an input type for the 5Vdc differential input. Default: Fast Digital Input	IN26: 3057 (Read)

Configuration Parameter	Description	Parameter Number
FTB IN27/Out11	Selects Fast Digital Output or an input type for the 5Vdc differential input. If FTB IN27 is selected as an I/O Interrupt Source, neither the Fast Digital Output function nor an Axis Encoder input can be assigned to this connector. Default: Fast Digital Input	IN27: 3058 (Read) OUT12: 3170 (Read/Write)
FTB IN28/Out12	Selects Fast Digital Output or an input type for the 5Vdc differential input. If FTB IN28 is selected as an I/O Interrupt Source, neither the Fast Digital Output function nor an Axis Encoder input can be assigned to this connector. Default: Fast Digital Input	IN28: 3059 (Read) OUT13: 3171 (Read/Write)
FTB ALGIN1 Ref	Selects the signal that is applied to analog input 1.	3256 (Read)
FTB ALGIN2 Ref	Selects the signal that is applied to analog input 1.	3257 (Read)

## FTB Outputs

The FTB Outputs tab allows you to change the default variable name and enable or disable open load detection for the 24Vdc outputs on the Fiber Terminal Block I/O.

Configuration Parameter	Description	Parameter Number
FTB OUT1—FTB OUT8	24Vdc Output.	3160—3167 (Read/Write for digital servos. Read-only if used as Analog Servo Drive Enable or Reset.)
FTB OUT1 Ref — FTB OUT8 Ref	Variable name associated with this FTB output. Default: Mx_FTb_OUTy. Where x is the module number and y is the output point number.	NA
FTB OUT1 Open Load Detect — FTB OUT8 Open Load Detect	Enables or disables open load detection for the output. Choices: Enabled, Disabled Default: Enabled	NA
FTB Outputs Default	Selects the default output mode for the FTB outputs. If Force Off is selected, outputs will go to 0 when the CPU is in Stop – IO Enabled mode. If Hold Last State is selected, outputs retain their last programmed value when the CPU is in Stop – IO Enabled or Stop – IO Disabled mode.  <i>Note:</i> When communication with the CPU is lost, FTB outputs are set to 0.  Choices: Force Off, Hold Last State Default: Force Off	NA
FTB OUT1—4 Power External Detection	When enabled, detects whether power is applied to terminals 1 and 19 of terminal block 1. Choices: Enabled, Disabled Default: Enabled	NA
FTB OUT5—8 Power External Detection	When enabled, detects whether power is applied to terminals 1 and 19 of terminal block 2. Choices: Enabled, Disabled Default: Enabled	NA
FTB ALGOUT1 Ref	Variable name associated with this analog output. Default: Mx_FTb_ALGOUTy. Where x is the module number and y is the output point number.	NA
FTB ALGOUT2 Ref		NA
FTB ALGOUT 1	Selects the signal that is output to analog output 1. Choices: Analog Output, Axis x Analog Servo Control. Default: Analog Output.	2104 (Read/Write Read-only if used as Analog Servo control)

Configuration Parameter	Description	Parameter Number
FTB ALGOUT 2	Selects the signal that is output to analog output 2. Choices: Analog Output, Axis x Analog Servo Control. Default: Analog Output.	2105 (Read/Write Read-only if used as Analog Servo control)

## PMM Faceplate and FTB I/O Function Descriptions

For details on the wiring of I/O connections, refer to Section 3, I/O Wiring, Connections and LED Operation.

For a summary of I/O functions that can be assigned to the PMM faceplate and FTB I/O points, refer to the PMM Faceplate I/O Functions Summary Section.

### Fast Digital Input

A 24Vdc input with high-speed input filtering.

Input Type	Input Filtering
FTB 24Vdc	500 $\mu$ s maximum
Faceplate General Purpose 24Vdc	500 $\mu$ s maximum
Faceplate High-speed 24Vdc	1 $\mu$ s maximum

### Digital Input

A 24Vdc input configured for low speed input filtering. Filter time for FTB and Faceplate 24Vdc is 5.5ms maximum.

### Axis 1– Axis 5 Touch Probes 1 and 2

A strobe input that captures the axis position on the input’s rising or falling edge.

If the Position Feedback Source is External Quadrature Encoder, the Touch Probe inputs must be assigned to the same device as the Encoder A and Encoder B inputs for that axis. They cannot be mixed between faceplate and FTB inputs.

The Touch Probe Detection parameter, which appears when a touch probe input is selected, determines whether the input is triggered on the rising or falling edge.

Touch Probe inputs for an analog axis can be configured only on the FTB Inputs tab.

### Axis 1 – Axis 5 Encoder Channel A/Axis 1 – Axis 5 Encoder Channel B

---

**Note:** *It is strongly recommended that Fault Detect be Enabled for Encoder inputs.*

---

The Channel A and Channel B inputs from an external quadrature encoder, which are used to determine axis direction and speed. Channel A and Channel B signals must be assigned to adjacent inputs.

If the Position Feedback Source is External Quadrature Encoder, the Touch Probe inputs must be assigned to the same device as the Encoder A and Encoder B inputs for that axis. They cannot be mixed between faceplate and FTB inputs.

The following parameters can be accessed using a MC\_ReadParameter(s) function block for information about the encoder:

- 1004, ExternalDeviceUserUnits
- 1005, ExternalDeviceCounts
- 1006, ExternalDevicePositionRange
- 1007, ExternalDeviceLowPositionLimit
- 1308, ExternalDevicePosition
- 1309, External DeviceVelocity

### Axis 1 Encoder Marker – Axis 5 Encoder Marker

A pulse that is produced once per axis revolution. Used to precisely establish the home reference position in a Find Home cycle.

### Axis 1 Overtravel + – Axis 4 Overtravel +

Hardware overtravel limits in the positive direction for Axes 1 through 4. To use this input, Overtravel Limit Switch must be enabled on the Axis tab.

If Over Travel Limit Switch is enabled, you must configure inputs for both positive and negative directions. For details, refer to Over Travel Limit Switch in Section 4.3.5.

### Axis 1 Overtravel – – Axis 4 Overtravel –

Hardware overtravel limits in the negative direction for Axes 1 through 4. To use this input, Overtravel Limit Switch must be enabled on the Axis tab.

### Axis 1 Home Switch 1 – Axis 4 Home Switch

Used to approximately establish the home reference position in a Find Home cycle. An open (logic 0) Home Switch input indicates the axis is on the positive side of the home switch and a closed (logic 1) Home Switch input indicates the axis is on the negative side of the home switch.

### Axis 1 Drive Status – Axis 4 Drive Status

These FTB inputs are used only with analog servo axes and are valid only if the Drive Status Input on the Axis tab is set to Drive Ready or Drive Available.

The feedback status signal from the analog servo drive. If the Drive Status Input on the Axis Tab is set to Drive Ready or Drive Available, an FTB input must be configured. Inputs numbered FTB IN 1 – to FTB IN 16 can be used for the Drive Status input.

### Digital Output

General-purpose 24Vdc output. (O1 and O2 on the PMM faceplate.)

### Fast Digital Output

5-Volt differential output used for general purpose signaling.



## 24Vdc Output

General-purpose 24Vdc output. In the event of logic power loss, these outputs default to OFF.

## Analog Output

General-purpose  $\pm 10$ Vdc single-ended analog output. Each Analog mode axis requires an Analog Output dedicated to Analog Servo Control.

## Analog Axis 1 Drive Enable – Analog Axis 4 Drive Enable

---

**Note:** *It is strongly recommended that Open Load Detection be Enabled for outputs and FTB Outputs Default be set to Force Off.*

---

Analog Axis 1–4 Drive Enable is used to control power on the servo. This signal should be connected to the servo amplifier enable input. Outputs numbered FTB OUT1 to FTB OUT8 can be used for the Drive Enable output.

## Analog Axis 1 Reset – Analog Axis 4 Reset

Analog Axis 1–4 Reset is an optional output used to clear errors on the analog servo when a compatible reset input is available on the servo drive. When the MC\_Reset or MC\_ModuleReset function block is executed to clear a Normal Stop or Fast Stop error, the Analog Axis Reset output will turn on for 250ms to attempt to clear any error on the analog servo. This signal should be connected to the servo drive reset input. Outputs numbered FTB OUT1 – FTB OUT8 can be used for the Drive Reset output.

## Axis 1 Analog Servo Control – Axis 4 Analog Servo Control

This signal should be connected to the analog velocity command or torque command input of the servo drive. FTB outputs numbered ALGOUT1 and FTB ALGOUT2 can be used for the Analog Servo Control output.

## PMM Faceplate I/O Functions Summary

Legend	Function	IN1	IN2	IN3/ OUT1	IN4/ OUT2	IN5	IN6	IN7	IN8
	Fast Digital Input	A	A	A	A	A	A	A	A
	Digital Input	Y	Y	Y	Y	Y	Y	Y	Y
	Axis 1–5 Touch Probe 1 <sup>3</sup>	A	A	A	A	A	A	A	A
	Axis 1–5 Touch Probe 2 <sup>4</sup>	A	A	A	A	A	A	A	A
	Axis 1 Encoder Ch. A								
	Axis 1 Encoder Ch. B								
	Axis 2 Encoder Ch. A								
	Axis 2 Encoder Ch. B								
	Axis 3 Encoder Ch. A								
	Axis 3 Encoder Ch. B								
	Axis 4 Encoder Ch. A								
	Axis 4 Encoder Ch. B								
	Axis 5 Encoder Ch. A	Y							
	Axis 5 Encoder Ch. B		Y						
	Axis 1 Encoder Marker								
	Axis 2 Encoder Marker								
	Axis 3 Encoder Marker								
	Axis 4 Encoder Marker								
	Axis 5 Encoder Marker			Y	Y	Y	Y	Y	Y
	Axis 1–4 Overtravel +	Y		Y		Y		Y	
	Axis 1–4 Overtravel –		Y		Y		Y		Y
	Axis 1–4 Home Switch	Y	Y	Y	Y	Y	Y	Y	Y
	Analog Axis 1–4 Drive Status								
	Digital Output			Y	Y				
	Fast Digital Output								
	24Vdc Output								
	Analog Axis 1–4 Drive Enable								
	Analog Axis 1–4 Drive Reset								

<sup>3</sup> Touch Probe inputs for an Analog Servo axis cannot be configured on the Faceplate I/O tab.

## FTB I/O Functions Summary

Function	IN1 – 16	IN17	IN18	IN19	IN20	IN21	IN22	IN23	IN24/ Out9	IN25/ Out10	IN26	IN27/ Out11	IN28/ Out128	Out1–
Fast Digital Input	A	A	A	A	A	A	A	A	A	A	A	A	A	
Digital Input	Y													
Axis 1–5 Touch Probe 1	A	A	A	A	A	A	A	A	A	A	A	A	A	
Axis 1–5 Touch Probe 2	A	A	A	A	A	A	A	A	A	A	A	A	A	
Axis 1 Encoder Ch. A		Y												
Axis 1 Encoder Ch. B			Y											
Axis 2 Encoder Ch. A						Y								
Axis 2 Encoder Ch. B							Y							
Axis 3 Encoder Ch. A				Y				Y						
Axis 3 Encoder Ch. B					Y				Y					
Axis 4 Encoder Ch. A										Y		Y		
Axis 4 Encoder Ch. B											Y		Y	
Axis 5 Encoder Ch. A		Y		Y				Y						
Axis 5 Encoder Ch. B			Y		Y				Y					
Axis 1 Encoder Marker				Y										
Axis 2 Encoder Marker					Y									
Axis 3 Encoder Marker										Y				
Axis 4 Encoder Marker											Y			
Axis 5 Encoder Marker				Y						Y				
Axis 1–4 Overtravel +	Y													
Axis 1–4 Overtravel –	Y													
Axis 1–4 Home Switch	Y													
Analog Axis 1–4 Drive Status	Y													
Digital Output														
Fast Digital Output									Y	Y		Y	Y	
24Vdc Output														Y
Analog Axis 1–4 Drive Enable														Y
Analog Axis 1–4 Drive Reset														Y

## 4.3.4 I/O Interrupts

The PMM supports I/O interrupts from the faceplate and FTB, and timed interrupts generated by the PMM. Any of these interrupts may be configured to trigger an interrupt block in the application logic.

Configuration Parameter	Description	Parameter Number
I/O Interrupt 1 Config, I/O Interrupt 2 Config, I/O Interrupt 3 Config	<p><b>Disabled</b> - Interrupts are not used.</p> <p><b>FP Input</b> - Any of the PMM's external discrete faceplate or FTB inputs can be configured as an I/O Interrupt to the CPU. I/O interrupts can be positive or negative edge detected.</p> <p><b>FTB Input</b> - Any of the PMM's external discrete faceplate or FTB inputs can be configured as an I/O Interrupt to the CPU. I/O interrupts can be positive or negative edge detected.</p> <p><b>Timed</b> - For a timed interrupt, an interval at which the PMM can generate the interrupt if configured. Only one of the three interrupts can be configured as Timed. Range is 2.0ms through 40.0ms in 1 ms increments. Only one timed interrupt can be configured per module. The timed interrupt is synchronized with command processing within the PMM. Default: 10.0 ms</p>	NA
I/O Interrupt n Source ID	<p>Available when I/O Interrupt n Config is set to FP Input or FTB Input.</p> <p>If I/O Interrupt n Config is set to FP Input: FP IN1 — FP IN8 If I/O Interrupt n Config is set to FTB Input: FTB IN1 — FTB IN28</p> <p>Default: If I/O Interrupt n Config is set to FP Input: FP IN1 If I/O Interrupt n Config is set to FTB Input: FTB IN1</p>	NA
I/O Interrupt n Detection	<p>Available when I/O Interrupt n Config is set to FP Input or FTB Input</p> <p>Negative Edge Trigger Positive Edge Trigger Default: Positive Edge Trigger</p> <p>Note that the maximum frequency for each configured I/O interrupt is once per 2mS. Interrupts occurring at a faster rate will be ignored and a corresponding fault will be generated.</p>	NA
I/O Interrupt n Interval (ms)	<p>Available when I/O Interrupt n Config is set to Timed.</p> <p>Specifies the interval at which the interrupt will be generated. For example, a value of 2 results in the interrupt occurring every 2ms.</p> <p>Range: 2.0—40.0 Default: 10.0 ms</p>	NA

## Using an I/O Interrupt with the PMM

The following steps summarize the procedure for configuring an I/O interrupt that will initiate execution of an interrupt logic block.

1. Decide which faceplate (FP) or fiber terminal block (FTB) input you will use for the interrupt. Considerations are voltage levels, single-ended or differential wiring and latency. All FP inputs are 24Vdc. The FTB has mixed 5Vdc and 24Vdc inputs. Wiring and terminal connection will depend on which input is selected as an interrupt input.

FP inputs IN1 through IN8 can be individually configured in the PMM FP I/O tab as Fast Digital Inputs, which provide a single-ended, 24Vdc interrupt input. If you are using FP inputs for home and overtravel inputs, make sure you have enough FP inputs to spare.

On the FTB, IN1 to IN16 (24Vdc Single-ended inputs) and IN17 to IN23 (5Vdc differential inputs) can be configured as Fast Digital Inputs.

For input specifications and wiring information, refer to Section 3, I/O Wiring, Connections and LED Operation.

2. In hardware configuration:
  - Go to the FP I/O or FTB Inputs tab and configure the input that will be used for the interrupt as a Fast-Digital Input.
  - On the I/O Interrupts tab configure I/O Interrupt 1, 2 or 3. Select the device, the input point and whether detection will be by positive or negative edge triggering.
  - Determine the status data reference for the interrupt you configured in step 3. Look at the %I start address on the Settings tab or, if Variable Mode is used, the terminal ID on the Terminals tab. The status data reference bits 0, 1 and 2 correspond to PMM interrupts 1, 2 and 3. For a summary of I/O status bits, refer to Section 4.3.2, PMM Status Data.
3. Create a new program block. In the block properties, open the Scheduling property. Create an interrupt of type I/O Interrupt and set the trigger as the %I bit (or Terminal ID) corresponding to the PMM interrupt status data bit.

Now when the configured Fast Digital Input is appropriately edge triggered, the PMM will generate the configured interrupt and the corresponding Status Data bit will call the configured logic block in the PACSystems controller.

### Recommendations:

- Keep the code in the interrupt logic block as short as possible.
- Many motion function blocks use an edge triggered execute permissive. For an example of the programming technique required for these instructions, refer to Calling an Executed Motion Function Block from an Interrupt Block in Section 5.2.4.

## 4.3.5 Axis Configuration Data

The Axis tabs contain parameters for configuring the operational characteristics for each axis. The number of axes that can be configured is selected on the Settings tab.

- For Digital Axis 1–4 parameters, refer to *Virtual Axis Parameters*.
- For Analog Axis parameters refer to *Axis 1 through Axis 4 – Analog Servo Modes*.

For Axis 5 parameters refer Virtual Axis Parameters section below.

Application logic can read and modify axis configuration parameters using the MC\_ReadParameter and MC\_WriteParameter function blocks. For a list of parameter numbers refer to Axis Parameter Number Index in Section 8.1.1.

### Preliminary Calculations

Before configuring an axis, you should first determine the User Units to Counts (UU/Cts) ratio for each feedback source that you will be using. It is important to set this relationship at the beginning of the configuration session because many configuration parameters are specified in user units.

These calculations are described in Computing Data Limits in User Units in Section 4.3.5.

### User Units to Counts Ratio

The UU/Cts ratio sets the number of position programming units for each feedback count. This allows you to program the PMM in application-specific units, such as degrees, millimeters or inches.

---

**Note:** PACMotion requires Uu:Cts ratios in which a User Unit represents an integer Count value. Ratios that would yield false resolution, where the User Unit would represent a fraction of a count are not supported. The Uu:Cts ratio must be  $\leq 1$ .

---

The basic equation to satisfy is:

$$\frac{\text{User Units}}{\text{Counts}} = \frac{(\text{Load Movement per Motor Rotation}) \div (\text{Desired User Units Resolution})}{\text{Encoder Counts per Motor Rotation}}$$

The numerator and denominator must fit within the following range limits, with the specified User Units less than or equal to Counts.

Range	Motor Encoder	External Quadrature Encoder
User Units range	0.000001 to 131,072.000000	0.000001 to 65,536.0
Counts range	1 to 1,048,576	1 to 65,536
Uu/Cts ratio range	0.000001:1,048,576 to 1	0.000001:65536 to 1

For example, if there is 1.000 inch of travel for 8192 feedback counts, a 1.000:8192 Uu:Cts ratio yields a User Units resolution of 1 User Unit per 0.001 inch.

The Uu:Cts ratio must be set correctly for the mechanical systems coupled to the axis, otherwise movement to unsafe and inaccurate positions may occur.

## Scaling Example

The Uu/Cts ratio is a powerful scaling feature because it can be configured to allow programming in other than default counts.

In a simplified example, an external encoder feedback application has an encoder that produces 1,000 quadrature counts per revolution (250 lines) and is geared to a machine that produces one inch of movement per revolution of the motor. The default unit would be one thousandth of an inch per count. However, you may want to write programs and use the PMM with metric units. A ratio of 254 User Units to 1000 Counts can be configured to allow this. With this ratio, one user unit would represent 0.1 mm and 254 user units would represent 25.4 mm (one inch) of travel.

## Sample Application

Use the Uu/Cts ratio to configure the PMM so you can program in engineering units instead of encoder counts. Assume a machine has a motor with a motor-mounted quadrature encoder connected through a gear reducer to a spur gear. The spur gear is mounted to the end of a pinch-roller shaft. The pinch roller feeds sheet material for a cut-to-length application. The motion program will specify the length of cut sheets. The programmer wishes to program in 0.01-inch resolution.

The following data is given:

- 2000-line encoder (x4 = 8000 quadrature counts per encoder revolution)
- 20:1 gear reduction (20 motor revolutions per spur gear revolution)
- 14.336-inch pitch diameter spur gear
- Inch desired programming unit (.01 inch)

Although several approaches are possible, the most straightforward is to base the calculations on a single spur gear revolution.

- Determine the number of User Units per spur gear revolution:

$$14.336\text{-inch pitch diameter} \times \pi = 45.0378 \text{ inches circumference}$$

$$45.0378 \text{ inches} / 0.001 \text{ inch desired programming units} = 45037.8 \text{ User Units per revolution of spur gear}$$

- Determine the number of encoder counts per spur gear revolution:

$$2000 \text{ lines} \times \frac{4 \text{ counts}}{\text{line}} \times \frac{20 \text{ motor revs.}}{1 \text{ gear rev.}} = 160,000 \text{ encoder counts per spur gear revolution}$$

- Check the value of the User Units to Counts ratio. The two numbers and their ratio must be within their allowed ranges, listed in User Units to Counts Ratio in Section 4.3.5.

$$45037.8 \text{ User Units} / 160,000 \text{ encoder counts} = 0.28148625$$

Consequently a 45,037.8 / 160,000 ratio would be used.

If the User Units or Counts value is too large, both numbers can be divided by a power of 10 to bring them within the allowed range. You must determine if any rounding error, if present, is of significance.

## Computing Data Limits in User Units

Four fixed limit values apply to axis hardware configuration: **MaxPosnLim**, **MaxVelLim**, **MaxAccLim** and **MaxJerkLim**. These limits are the maximum allowed values for position, velocity, acceleration and jerk, and are expressed in feedback device counts.

The hardware configuration tool converts these values to user units using the *Counts* and *User Units* parameter values for the configured *Position Feedback Source*, yielding the variables: **MaxPosnUu**, **MaxVelUu**, **MaxAccUu** and **MaxJerkUu**.

The *Counts* and *User Units* parameter values are also used to calculate the minimum data limits, **MinPosnUu**, **MinVelUu**, **MinAccUu** and **MinJerkUu**, as shown in *Formulas for Computing Minimum Data Limits in User Units*, below. When used in the Configuration Parameter Descriptions, these values refer to the corresponding value for the configured Position Feedback Source. For example, if *External Device* is selected, MaxPosnUu refers to MaxPosnUuExt, MinVelUu refers to MinVelUuExt, etc.

## Constants Used in Data Limit Calculations

Limit in Counts	Constant	Value	Units
Maximum Position	MaxPosnLim	$4.0 \times 10^{10}$	Counts
Maximum Velocity	MaxVelLim	$1.1 \times 10^8$	Counts/Sec
Maximum Acceleration	MaxAccLim	$1.0 \times 10^{12}$	Counts/Sec <sup>2</sup>
Maximum Jerk	MaxJerkLim	$1.0 \times 10^{15}$	Counts/Sec <sup>3</sup>

## Formulas for Computing Maximum Data Limits in User Units

Feedback Source	Limit	Maximum Data Limit	Formula
Motor Encoder	Position	MaxPosnUuMtr =	$MaxPosnLim * \frac{MotorEncodedUserUnits}{MotorEncodedCounts}$
	Velocity	MaxVelUuMtr =	$MaxVelLim * \frac{MotorEncodedUserUnits}{MotorEncodedCounts}$
	Acceleration	MaxAccUuMtr =	$MaxAccLim * \frac{MotorEncodedUserUnits}{MotorEncodedCounts}$
	Jerk	MaxJerkUuMtr =	$MaxJerkLim * \frac{MotorEncodedUserUnits}{MotorEncodedCounts}$
External Quadrature Encoder	Position	MaxPosnUuExt =	$MaxPosnLim * \frac{ExternalDeviceUserUnits}{ExternalDeviceCounts}$
	Velocity	MaxVelUuExt =	$MaxVelLim * \frac{ExternalDeviceUserUnits}{ExternalDeviceCounts}$
	Acceleration	MaxAccUuExt =	$MaxAccLim * \frac{ExternalDeviceUserUnits}{ExternalDeviceCounts}$
	Jerk	MaxJerkUuExt =	$MaxJerkLim * \frac{ExternalDeviceUserUnits}{ExternalDeviceCounts}$
Axis 5 Path Generator	Position	MaxPosnCmd =	MaxPosnLim × Command Position Resolution
	Velocity	MaxVelCmd =	MaxVelLim × Command Position Resolution
	Acceleration	MaxAccCmd =	MaxAccLim × Command Position Resolution
	Jerk	MaxJerkCmd =	MaxJerkLim × Command Position Resolution



### Formulas for Computing Minimum Data Limits in User Units

Feedback Source	Limit	Maximum Data Limit Variable	Formula
Motor Encoder	Position	MinPosnUuMtr =	$0.1 * \frac{MotorEncoderCountsPerRev * MotorEncoderUserUnits}{MotorEncoderCounts}$
	Velocity	MinVelUuMtr =	$\frac{1}{600} * \frac{MotorEncoderCountsPerRev * MotorEncoderUserUnits}{MotorEncoderCounts}$
	Acceleration	MinAccUuMtr =	$0.1 * MinVelUnits$
	Jerk	MinJerkUuMtr =	$0.1 * MinAccUnits$
External Quadrature Encoder <sup>4</sup>	Position	MinPosnUuExt =	$0.1 * \frac{ExternalDeviceCountsPerRev * ExternalDeviceUserUnits}{ExternalDeviceCounts}$
	Velocity	MinVelUuExt =	$\frac{1}{600} * \frac{ExternalDeviceCountsPerRev * ExternalDeviceUserUnits}{ExternalDeviceCounts}$
	Acceleration	MinAccUuExt =	$0.1 * MinVelUserUnits$
	Jerk	MinJerkUuExt =	$0.1 * MinAcceUserUnits$
Axis 5 Path Generator	Position	MinPosnCmd =	$0.1 * Command Counts per Motor Revolution * Command Position Resolution$
	Velocity	MinVelCmd =	$0.1 * Command Counts per Motor Revolution * Command Position Resolution$
	Acceleration	MinAccCmd =	$0.01 * Command Counts per Motor Revolution * Command Position Resolution$
	Jerk	MinJerkCmd =	$\frac{1}{60000} * Command Counts Per Motor Revoluion * Command Position Resolution$

<sup>4</sup> On axis 5, and axes 1-4, if Position Feedback Source is Motor Encoder, 1024 is used for External Device Counts per Rev.

### Formulas for Converting Revolutions and RPM to User Units

Some limits are specified in revolutions (Revs) or RPMs. The following formulas convert Revs and RPMs to the equivalent User Units. In the following equations, Revs, UU, RPM, and UU/sec refer to the parameters to be converted.

Feedback Source	Conversion	Formula
Motor Encoder	Revs to UU	$UU = Revs * Motor\ Encoder\ Counts\ per\ Rev * \frac{Motor\ Encoder\ User\ Units}{Motor\ Encoder\ Counts}$
	UU to Revs	$Revs = U * \frac{1}{Motor\ Encoder\ Counts\ per\ Rev} * \frac{Motor\ Encoder\ Counts}{Motor\ Encoder\ User\ Units}$
	RPM to UU/sec	$UU/sec = RPM * Motor\ Encoder\ Counts\ per\ Rev * \frac{Motor\ Encoder\ User\ Units}{Motor\ Encoder\ Counts} * \frac{1}{60}$
	UU/sec to RPM	$RPM = UU/sec * \frac{1}{Motor\ Encoder\ Counts\ per\ Rev} * \frac{Motor\ Encoder\ Counts}{Motor\ Encoder\ User\ Units} * 60$
External Quadrature Encoder	Revs to UU	$UU = Revs * External\ Device\ Counts\ per\ Rev * \frac{ExternalDeviceUserUnits}{ExternalDeviceCounts}$
	UU to Revs	$Revs = UU * \frac{1}{External\ Device\ Counts\ per\ Rev} * \frac{ExternalDeviceUserUnits}{ExternalDeviceCounts}$
	RPM to UU/sec	$UU/sec = RPM * External\ Device\ Counts\ per\ Rev * \frac{External\ Device\ User\ Units}{External\ Device\ Counts} * \frac{1}{60}$
	UU/sec to RPM	$RPM = UU * \frac{1}{Command\ Counts\ per\ Rev} * \frac{External\ Device\ Counts}{External\ Device\ User\ Units} * 60$
Axis 5 Path Generator	Revs to UU	$UU = Revs * Command\ Counts\ per\ Motor\ Rev * Command\ Position\ Resolution$
	UU to Revs	$RPM = UU * \frac{1}{Command\ Counts\ per\ Rev} * \frac{1}{Command\ Position\ Resolution}$
	RPM to UU/sec	$\frac{UU}{sec} = RPM * CommandCountsPerMotorRev * CommandPositionResolution * \frac{1}{6}$
	UU/sec to RPM	$RPM = UU/sec * \frac{1}{Command\ Counts\ per\ Motor\ Rev} * \frac{1}{Command\ Position\ Resolution} * 60$

## Virtual Axis Parameters

Axis 5 functions as a virtual axis that can act as the master position source for other axes. The total number of actual motors controlled by the module does not include the virtual axis. Axis 5 consists of a Path Generator and optional External Device. These typically operate independently of each other

Configuration Parameter	Description	Units	Ref.	Parameter Number
Stop Axis on FTB Error	When enabled, an FTB communications fault or a failure to configure the FTB will cause a normal stop on the axis. Motion on the axis stops and the axis goes to the ErrorStop state. Choices: Enabled, Disabled Default: Enabled Must be Enabled if Position Feedback Source is External Device.	NA	Page 116	NA
Position Feedback Source	Selects the position feedback source for the axis. Choices: Motor Encoder, External Device. Default: Motor Encoder	NA	Page 117	NA
Axis Positioning Mode	Selects the positioning mode for the axis. Choices: Linear, Rotary Default: Linear	N/A	Page 117	1225 (Read)
Motor Encoder Mode	Selects the Incremental or Absolute mode for the serial encoder that is mounted on the motor. Choices: Incremental, Absolute Default: Absolute	NA	Page 118	NA
Motor Encoder User Units	Used to calculate the Uu/Cts ratio, a scaling factor that relates user programming units to encoder counts for Motor Encoder parameters and general user unit axis parameters when Motor Encoder is specified as the Position Feedback Source.  <u><b>Note:</b> <i>User Units must be ≤ Counts.</i></u>  Range: 0.000001 – 131,072 Default: 1.0	N/A	Page 118	1000 (Read/Write)
Motor Encoder Counts	Used to calculate the Uu/Cts ratio, a scaling factor that relates user programming units to encoder counts for Motor Encoder parameters and general user unit axis parameters when Motor Encoder is specified as the Position Feedback Source. Range: 1 – 1,048,576 Default: 1	N/A	Page 118	1001 (Read/Write)

Configuration Parameter	Description	Units	Ref.	Parameter Number
Motor Encoder Position Range	Available if Position Feedback Source is set to Motor Encoder and Axis Positioning Mode is set to Rotary. Has no effect if Position Feedback Source is set to External. Specifies the position range of the Motor Encoder. High Position Limit = Low Position Limit + Position Range. Low Limit: $\text{MinPosnUuMtr}^5$ High Limit: $\text{MaxPosnUuMtr}^5$ Motor Encoder Low Position Limit + Motor Encoder Position Range $\leq$ MaxPosnUuMtr Default: 16777216.0	Uu	Page 118	1002 (Read/Write)
Motor Encoder Low Position Limit	Available if Position Feedback Source is set to Motor Encoder and Axis Positioning Mode is set to Rotary. Specifies the low limit of the Motor Encoder Position Range. High Limit: $-\text{MaxPosnUuMtr}^6$ Low Limit: $\text{MaxPosnUuMtr}^6 - \text{MinPosnUuMtr}^6$ $-\text{MaxPosnUuMtr} \leq$ Motor Encoder Low Position Limit $\leq$ MaxPosnUuMtr Default: -8388608	Uu	Page 119	1003 (Read/Write)
Motor Encoder Counts per Motor Revolution	Selects encoder resolution. Must be $\leq$ the maximum encoder resolution supported by the Drive Type. Choices: 65536, 131072, 1048576 Default: 1048576	Counts	Page 118	NA
Motor Encoder Maximum Positive RPM Limit	Motor encoder maximum positive RPM limit. Range: 10 – 10,000 Default: 8191	RPM	Page 119	NA
Motor Encoder Maximum Negative RPM Limit	Motor encoder maximum negative RPM limit. Range: 10 – 10,000 Default: 8191	RPM	Page 119	NA
External Device	Selects the external feedback source. Choices: External Quadrature Encoder, None Must be External Quadrature Encoder if Position Feedback Source is set to External Device. Default: None	NA	Page 119	NA

<sup>6</sup> For definitions of MaxAccUu, MaxPosnUu, MaxVelUu and MaxJerkUu, refer to Computing Data Limits in User Units in Section 4.3.5

Configuration Parameter	Description	Units	Ref.	Parameter Number
External Device User Units	<p>Available only if External Device is set to External Quadrature Encoder.</p> <p>Used to calculate the Uu/Cts ratio, a scaling factor that relates user programming units to encoder counts for External Device parameters and general user unit axis parameters when External Device is specified as the Position Feedback Source.</p> <hr/> <p><b>Note:</b> <i>User Units must be <math>\leq</math> Counts.</i></p> <p>Range: 0.000001 – 65,536.0 Default: 1.0</p>	N/A	Page 122	1004 (Read/Write)
External Device Counts	<p>Available only if External Device is set to External Quadrature Encoder.</p> <p>Used to calculate the Uu/Cts ratio, a scaling factor that relates user programming units to encoder counts for External Device parameters and general user unit axis parameters when External Device is specified as the Position Feedback Source.</p> <p>Range: 1 – 65,536 Default: 1</p>	N/A	Page 122	1005 (Read/Write)
External Device Position Range	<p>Available if External Device is set to External Quadrature Encoder and Axis Positioning Mode is set to Rotary.</p> <p>Specifies the position range of the External Quadrature Encoder.</p> <p>High Position Limit = Low Position Limit + Position Range.</p> <p>Low Limit:           MinPosnUuExt<sup>6</sup> High limit:           MaxPosnUuExt<sup>6</sup> External Device Low Position Limit + External Device Position Range <math>\leq</math> MaxPosnUuExt Default: 16,777,216.0</p>	Uu	Page 122	1006 (Read/Write)
External Device Low Position Limit	<p>Available if External Device is set to External Quadrature Encoder and Axis Positioning Mode is set to Rotary.</p> <p>Specifies the low limit of the External Device Position Range.</p> <p>Range: -MaxPosnUuExt – MaxPosnUuExt<sup>6</sup> -MaxPosnExt <math>\leq</math> Command Low Position Limit <math>\leq</math> MaxPosnExt Default: -8,388,608</p>	Uu	Page 122	1007 (Read/Write)

Configuration Parameter	Description	Units	Ref.	Parameter Number
External Device Counts per Motor Revolution	Available only if External Device is set to External Quadrature Encoder and Position Feedback Source is set to External Device. Selects the resolution of the External Quadrature Encoder. Range: 100 – 1,048,576.0 Default: 8,192	Counts	Page 122	NA
Over Travel Limit Switch	Over Travel Limit Switch Enable / Disable Ignored by axes using Drive Type Synthetic. Choices: Enabled, Disabled Default: Enabled	N/A	Page 123	1400 (Read/Write)
Axis Direction	Defines the positive axis direction as counter-clockwise motor direction (Normal) or clockwise motor direction (Reverse). Choices: Normal, Reverse Default: Normal	N/A	Page 123	NA
Software End of Travel	Enables or disables the High Software EOT and Low Software EOT limits. Choices: Disabled, Enabled Must be set to Disabled if Axis Positioning Mode is Rotary. Default: Disabled	N/A	Page 124	4 (Read/Write)
High Software EOT Limit	Available only if Software End of Travel is set to Enabled. Software end of travel limit in the positive direction. Range: -MaxPosnUu – MaxPosnUu <sup>6</sup> High Software EOT Limit must be > Low Software EOT Limit Range is determined by the Uu and counts for the configured Position Feedback Source. Default: +8,388,607.0	Uu	Page 124	2 (Read/Write)
Low Software EOT Limit	Available only if Software End of Travel is set to Enabled. Software end of travel limit in the negative direction. Range: -MaxPosnUu – MaxPosnUu <sup>6</sup> Low Software EOT Limit must be < High Software EOT Limit Range is determined by the Uu and counts for the configured Position Feedback Source. Default: -8,388,608.0	Uu	Page 124	3 (Read/Write)
Max Velocity System	Specifies the maximum axis velocity. Low limit: 0.1 High limit: Maximum velocity of Drive Type Default: 4,000.0	RPM	Page 125	8 (Read)

Configuration Parameter	Description	Units	Ref.	Parameter Number
Max Acceleration System	Specifies the maximum rate of velocity increase. Range: 0.01 – 60E12 / Feedback Device Counts per Rev Default: 40,000	RPM/sec	Page 125	12 (Read)
Max Deceleration System	Specifies the maximum rate of velocity decrease. Range: 0.01 – 60E12 / Feedback Device Counts per Rev Default: 40,000	RPM/sec	Page 125	14 (Read)
Max Jerk	Specifies the maximum rate of change in acceleration or deceleration. Low limit: MinJerkUu <sup>6</sup> High limit: MaxJerkUu <sup>6</sup> Default: 100,000,000,000,000.0 (1E14)	Uu/sec <sup>3</sup>	Page 125	16 (Read/Write)
Drive Disable Delay (ms)	Time after an error that the power is forced off on the servo. Range: 0–60,000 Default: 100	ms	Page 125	NA
Drive Type	Drive type code identifying the motor attached to the axis.  Default: Synthetic PACMotion PSD411	N/A	Section 2.3.6, Drive Type	NA
Motor Velocity Limit	Specifies the maximum speed of the motor. Range: 60 – 1.1 x maximum velocity of configured Drive Type Default: 4,000	RPM	Section 2.3.6, Drive Type	NA
Torque Limit	Specifies the maximum allowed torque, in percent of available torque, to be produced by the servomotor at commanded velocity. Range: 0.0 – 100.0 Default: 100.0	%	Section 2.3.6, Drive Type	1015 (Read/Write)
Position Lag Monitoring	When enabled, allows detection of position lag. The maximum position lag allowed is specified by the Max Position Lag parameter. Choices: Disabled, Enabled Default: Disabled	NA	Section 2.3.6, Drive Type	6 (Read/Write)

Configuration Parameter	Description	Units	Ref.	Parameter Number
Max Position Lag	<p>Maximum position lag allowed when controlling the servo.</p> <p>Position Lag = Commanded Position - Actual Position</p> <p>Low Limit = <math>0.001 \times \text{MaxVelocitySystem (RPM)} \times 1/60 \text{ min/s} \times \text{Counts per Rev} \times \text{Uu/Counts}</math></p> <p>High Limit = <math>100 \times (\text{Counts per /Rev}) \times (\text{Uu/Counts})</math></p> <p>Must be <math>\leq \text{Max Position Error} \times \text{Counts per Rev} \times \text{Uu/Count}</math>.</p> <p>Default: 500,000.0</p> <p>Range is determined by the Counts per Rev, Uu and counts for the configured Position Feedback Source.</p>	Uu	Section 2.3.6, Drive Type	7 (Read/Write)
Max Position Error	<p>An absolute value used to determine when the servo is out of sync and should be stopped.</p> <p>Low limit = <math>0.002 \times \text{MaxVelocitySystem (RPM)} \times 1/60</math></p> <p>High limit = 100</p> <p>Default: 10</p>	Motor Revs	Page 127	1016 (Read)
In Position Zone	<p>Maximum allowed position error. If the position error is less than this value, the axis is considered to be in the position zone.</p> <p>Range: <math>0.0\text{--}60000.0 \times (\text{Uu/Counts})</math></p> <p>Range is determined by the Uu and counts for the configured Position Feedback Source.</p> <p>Default: 10.0</p>	Uu	Page 127	1008 (Read/Write)
Position Loop Time Constant	<p>Determines the response speed of the closed position loop.</p> <p>Range: 0.0–1,0000.0</p> <p>Default: 100.0</p>	ms	Page 128	1009 (Read/Write)
Velocity Feedforward	<p>Specifies the percentage of commanded velocity that is added to the PMM's position loop velocity command output.</p> <p>Range: 0.0 – 120.0</p> <p>Default: 0.0</p>	%	Page 129	1010 (Read/Write)
Error Stop Deceleration	<p>Maximum deceleration allowed for a normal stop.</p> <p>Low limit: <math>\text{MinAccUu}^6</math></p> <p>High limit: The lesser of <math>\text{MaxAccUu}^6</math> or Max Deceleration System in <math>\text{Uu/sec}^2</math></p> <p>Default: 1,000,000.0</p>	$\text{Uu/sec}^2$	Page 130	1013 (Read/Write)



Configuration Parameter	Description	Units	Ref.	Parameter Number
Error Stop Jerk	Maximum jerk allowed for a normal stop. Low limit: MinJerkUu <sup>6</sup> High limit: The lesser of MaxJerkLim or MaxJerkUu <sup>6</sup> Default: 10,000,000.0	Uu/sec <sup>3</sup>	Page 130	1014 (Read/Write)
Master Axis Velocity Filter	Specifies filter width of the master axis velocity signal. Valid values are 1, 2, 4, 8, 16, 32, 64, 128 and 256. Default: 8	ms	Page 130	1321 (Read/Write)
Feedback Moving Deadband	Specifies a deadband range for Actual Velocity to allow proper operation of the Feedback Moving status bit. Range: 0—200,000 × (Uu/Counts) Range is determined by the Uu and counts for the configured Position Feedback Source. Default: 100.0	Uu/sec	Page 130	1024 (Read/Write)
Command Moving Deadband	Specifies a deadband range for Commanded Velocity to allow proper operation of the CommandMoving status bit. Range: 0—200,000 × (Uu/Counts) Default: 0.0	Uu/sec	Page 131	1025 (Read/Write)
Sync Master Position Deadband	Specifies a positional deadband to be applied to the master's observed position by this axis when it is a slave. Range: 0.0—60,000.0 Default: 10.0	Master Uu	Page 131	1312 (Read/Write)
Disabled Direction Deadband	Specifies a deadband range in the direction not enabled by MC_POWER. Low limit: 0 High limit: MinPosnUu <sup>6</sup>	Uu	Page 132	1313 (Read/Write)

## Axis 1 through Axis 4 – Analog Servo Modes

Configuration Parameter	Description	Units	Ref.	Parameter Number
Stop Axis on FTB Error	When enabled, an FTB communications fault or a failure to configure the FTB will cause a normal stop on the axis. Motion on the axis stops and the axis goes to the ErrorStop state. Must be Enabled for analog servos.	NA	Page 116	NA
Position Feedback Source	Selects the position feedback source for the axis. For analog servos, this parameter is automatically set to External Device.	NA	Page 117	NA
Axis Positioning Mode	Selects the positioning mode for the axis. Choices: Linear, Rotary Default: Linear	N/A	Page 117	1225 (Read)
Drive Status Input	Selects whether a status output from the analog servo drive is connected to the Drive Status input on the FTB and, if so, selects the operating mode of the input. If set to Drive Ready or Drive Available, a Drive Status input must be configured on the FTB. Choices: Drive Ready, Drive Available, Disabled Default: Drive Ready	NA	Page 117	NA
External Device	Selects the external feedback source. Must be External Quadrature Encoder for analog servos. Default: None	NA	Page 119	NA
External Device User Units	Used to calculate the Uu/Cts ratio, a scaling factor that relates user programming units to encoder counts for External Device parameters and general user unit axis parameters when External Device is specified as the Position Feedback Source.  <b>Note:</b> <i>User Units must be ≤ Counts.</i>  Range: 0.000001 – 65,536.0 Default: 1.0	N/A	Page 122	1004 (Read/Write)

Configuration Parameter	Description	Units	Ref.	Parameter Number
External Device Counts	Used to calculate the Uu/Cts ratio, a scaling factor that relates user programming units to encoder counts for External Device parameters and general user unit axis parameters when External Device is specified as the Position Feedback Source. Range: 1 – 65,536 Default: 1	N/A	Page 122	1005 (Read/Write)
External Device Position Range	Available if Axis Positioning Mode is set to Rotary. Specifies the position range of the External Quadrature Encoder. High Position Limit = Low Position Limit + Position Range. Low Limit: $\text{MinPosnUuExt}^6$ High limit: $\text{MaxPosnUuExt}^6$ External Device Low Position Limit + External Device Position Range $\leq$ $\text{MaxPosnUuExt}$ Default: 16,777,216.0	Uu	Page 122	1006 (Read/Write)
External Device Low Position Limit	Available if External Device is set to External Quadrature Encoder and Axis Positioning Mode is set to Rotary. Specifies the low limit of the External Device Position Range. Range: $-\text{MaxPosnUuExt} - \text{MaxPosnUuExt}^6$ $-\text{MaxPosnExt} \leq$ Command Low Position Limit $\leq$ $\text{MaxPosnExt}$ Default: -8,388,608	Uu	Page 122	1007 (Read/Write)
External Device Counts per Motor Revolution	Selects the resolution of the External Quadrature Encoder. Range: 100 – 1,048,576.0 Default: 8,192	Counts	Page 122	NA
External Encoder Maximum Positive RPM Limit	External encoder maximum positive RPM limit. Range: 10 – 20000 Default: 8191	RPM	Page 122	NA
External Encoder Maximum Negative RPM Limit	External encoder maximum negative RPM limit. Range: 10 – 20000 Default: 8191	RPM	Page 122	NA

Configuration Parameter	Description	Units	Ref.	Parameter Number
Motor Velocity at 10 Volts	Analog Servo Velocity mode only. The actual servo velocity commanded when 10Vdc is commanded to the analog output. Range: 0.1 – 65635.0 Default: 4000	RPM	Page 122	NA
Minimum Velocity Output	Analog Servo Velocity mode only. The minimum velocity output that will be commanded when the velocity command is non-zero. Range: 0 – 1000 Default: 0	mV	Page 123	1323 (Read/Write)
Over Travel Limit Switch	Over travel limit switch enable / disable Choices: Enabled, Disabled Default: Enabled	N/A	Page 123	1400 (Read/Write)
Software End of Travel	Enables or disables the High Software EOT and Low Software EOT limits. Choices: Disabled, Enabled Must be set to Disabled if Axis Positioning Mode is Rotary. Default: Disabled	N/A	Page 124	4 (Read/Write)
High Software EOT Limit	Available only if Software End of Travel is set to Enabled. Software end of travel limit in the positive direction. Range: -MaxPosnUu – MaxPosnUu <sup>6</sup> High Software EOT Limit must be > Low Software EOT Limit Range is determined by the Uu and counts for the configured Position Feedback Source. Default: +8,388,607.0	Uu	Page 124	2 (Read/Write)
Low Software EOT Limit	Available only if Software End of Travel is set to Enabled. Software end of travel limit in the negative direction. Range: -MaxPosnUu – MaxPosnUu <sup>6</sup> Low Software EOT Limit must be < High Software EOT Limit Range is determined by the Uu and counts for the configured Position Feedback Source. Default: -8,388,608.0	Uu	Page 124	3 (Read/Write)
Max Velocity System	Specifies the maximum axis velocity. Low limit: 0.1 High limit: 6000 Default: 4,000.0	RPM	Page 125	8 (Read)

Configuration Parameter	Description	Units	Ref.	Parameter Number
Max Acceleration System	Specifies the maximum rate of velocity increase. Range: 0.01 – 60E12 / Feedback Device Counts per Rev Default: 40,000	RPM/sec	Page 125	12 (Read)
Max Deceleration System	Specifies the maximum rate of velocity decrease. Range: 0.01 – 60E12 / Feedback Device Counts per Rev Default: 40,000	RPM/sec	Page 125	14 (Read)
Max Jerk	Specifies the maximum rate of change in acceleration or deceleration. Low limit: MinJerkUu <sup>6</sup> High limit: MaxJerkUu <sup>6</sup> Default: 100,000,000,000,000.0 (1E14)	Uu/sec <sup>3</sup>	Page 125	16 (Read/Write)
Drive Disable Delay (ms)	Time after an error that the power is forced off on the servo. Range: 0–60,000 Default: 100	ms	Page 125	NA
Motor Velocity Limit	Specifies the maximum speed of the motor. Range: 60–10000 Default: 4,000	RPM	Section 2.3.6, Drive Type	NA
Current Limit	Specifies the maximum allowed drive current, in percent of available continuous current, to be produced by the servomotor at commanded velocity. Range: 0.0 – 300.0 Default: 300.0	%	Section 2.3.6, Drive Type	1015 (Read/Write)
Current Limit Active	Specifies that the commanded torque has exceeded the torque limit setting	Boolean	Section 2.3.6, Drive Type	1207 (Read/Write)
Position Lag Monitoring	When enabled, allows detection of position lag. The maximum position lag allowed is specified by the Max Position Lag parameter. Choices: Disabled, Enabled Default: Disabled	NA	Section 2.3.6, Drive Type	6 (Read/Write)

Configuration Parameter	Description	Units	Ref.	Parameter Number
Max Position Lag	<p>Maximum position lag allowed when controlling the servo.</p> <p>Position Lag = Commanded Position - Actual Position</p> <p>Low Limit = <math>0.001 \times \text{MaxVelocitySystem (RPM)} \times 1/60 \text{ min/s} \times \text{Counts per Rev} \times \text{UU/Counts}</math></p> <p>High Limit = <math>100 \times (\text{Counts per /Rev}) \times (\text{Uu/Counts})</math></p> <p>Must be <math>\leq</math> Max Position Error <math>\times</math> Counts per Rev <math>\times</math> Uu/Count.</p> <p>Default: 500,000.0</p> <p>Range is determined by the Counts per Rev, Uu and counts for the configured Position Feedback Source.</p>	Uu	Section 2.3.6, Drive Type	7 (Read/Write)
Max Position Error	<p>An absolute value used to determine when the servo is out of sync and should be stopped.</p> <p>Low limit = <math>0.002 \times \text{MaxVelocitySystem (RPM)} \times 1/60</math></p> <p>High limit = 100</p> <p>Default: 10</p>	Motor Revs	Page 127	1016 (Read)
In Position Zone	<p>Maximum allowed position error. If the position error is less than this value, the axis is considered to be in the position zone.</p> <p>Range: <math>0.0\text{--}60000.0 \times (\text{Uu/Counts})</math></p> <p>Range is determined by the Uu and counts for the configured Position Feedback Source.</p> <p>Default: 10.0</p>	Uu	Page 127	1008 (Read/Write)
Position Loop Time Constant	<p>Determines the response speed of the closed position loop.</p> <p>Range: <math>0.0\text{--}1,0000.0</math></p> <p>Default: 100.0</p>	ms	Page 128	1009 (Read/Write)
Velocity Feedforward	<p>Specifies the percentage of commanded velocity that is added to the PMM's position loop velocity command output.</p> <p>Range: <math>0.0\text{--}120.0</math></p> <p>Default: 0.0</p>	%	Page 129	1010 (Read/Write)
Load Inertia Ratio (256 = 1.1)	<p>Analog Servo <b>Torque</b> Mode only.</p> <p>Specifies the gain applied to the velocity control loop to match load inertia to motor inertia.</p> <p>Range: <math>0\text{--}4096</math></p> <p>Default: 128</p>	NA	Page 129	10032 (Read/Write)

Configuration Parameter	Description	Units	Ref.	Parameter Number
Velocity Loop Proportional Gain	Analog Servo <b>Torque</b> Mode only. Sets the velocity regulator proportional gain. Range: 0 – 32767 Default: 1500	NA	Page 129	10007 (Read/Write)
Velocity Loop Integral Gain	Analog Servo <b>Torque</b> Mode only. Sets the velocity regulator integral gain. Range: 0 – 32767 Default: 0	NA	Page 129	10006 (Read/Write)
Torque Command Filter	Analog Servo <b>Torque</b> Mode only. Sets the torque command filter cutoff frequency 3db point in Hz. A value of 0 disables the filter. When set to 0, the Torque Filter is disabled. Range: 0, or 60 – 400 Default: 0	Hz	Page 129	1322 (Read/Write)
Error Stop Deceleration	Maximum deceleration allowed for a normal stop. Low limit: MinAccUu <sup>6</sup> High limit: The lesser of MaxAccUu or Max Deceleration System in Uu/sec <sup>2</sup> Default: 1,000,000.0	Uu/sec <sup>2</sup>	Page 130	1013 (Read/Write)
Error Stop Jerk	Maximum jerk allowed for a normal stop. Low limit: MinJerkUu <sup>6</sup> High limit: The lesser of MaxJerkLim or MaxJerkUu <sup>6</sup> Default: 10,000,000.0	Uu/sec <sup>3</sup>	Page 130	1014 (Read/Write)
Master Axis Velocity Filter	Specifies filter width of the master axis velocity signal. Valid values are 1, 2, 4, 8, 16, 32, 64, 128 and 256. Default: 8	ms	Page 130	1321 (Read/Write)
Feedback Moving Deadband	Specifies a deadband range for Actual Velocity to allow proper operation of the Feedback Moving status bit. Range: 0–200,000 × (Uu/Counts) Range is determined by the Uu and counts for the Quadrature Encoder. Default: 100.0	Uu/sec	Page 130	1024 (Read/Write)
Command Moving Deadband	Specifies a deadband range for Commanded Velocity to allow proper operation of the CommandMoving status bit. Range: 0–200,000 × (Uu/Counts) Default: 0.0	Uu/sec	Page 131	1025 (Read/Write)

Configuration Parameter	Description	Units	Ref.	Parameter Number
Sync Master Position Deadband	Specifies a positional deadband to be applied to the master's observed position by this axis when it is a slave. Range: 0.0–60,000.0 Default: 10.0	Master Uu	Page 131	1312 (Read/Write)
Disabled Direction Deadband	Specifies a deadband range in the direction not enabled by MC_POWER. Low limit: 0 High limit: MinPosnUu <sup>6</sup>	Uu	Page 132	1313 (Read/Write)



## Virtual Axis Parameters

Axis 5 functions as a virtual axis that can act as the master position source for other axes. The total number of actual motors controlled by the module does not include the virtual axis. Axis 5 consists of a Path Generator and optional External Device. These typically operate independently of each other.

Configuration Parameter	Description	Units	Ref	Parameter Number
Stop Axis on FTB Error	When enabled, an FTB communications fault or a failure to configure the FTB will cause a normal stop on the axis. Motion on the axis stops and the axis goes to the ErrorStop state. Choices: Enabled, Disabled. Must be Enabled if Position Feedback Source is External Device. Default: Enabled	NA	Page 116	NA
Axis Positioning Mode	Selects the linear or rotary positioning mode for the axis 5 path generator. Choices: Linear, Rotary Default: Linear	NA	Page 117	1225 (Read)
Command Position Resolution	Specifies the resolution of the Axis 5 path generator. Range: 0.0000001 – 1.0 Default: 1.0	Uu	Page 131	1000 (Read/Write)
Command Position Range	Available only if Axis Positioning Mode is Rotary. Specifies the position range of Axis 5 path generator. Low limit: MinPosnCmd <sup>6</sup> High limit: MaxPosnCmd <sup>6</sup> Command Low Position Limit + Command Position Range must be ≤ MaxPosnCmd Default: 1,677,216	Uu	Page 133	1022
Command Low Position Limit	Available only if Axis Positioning Mode is Rotary. Specifies the low limit of the Command Position Range. Low Limit: -MaxPosnCmd <sup>6</sup> High Limit: MaxPosnCmd <sup>6</sup> - MinPosnCommand <sup>6</sup> -MaxPosnCmd ≤ Command Low Position Limit ≤ MaxPosnCmd Default: -8,388,608	Uu	Page 133	1023
Command Counts Per Motor Revolution	Axis 5 Path Generator resolution Choices: 65,536, 131,072, 1,048,576 Default: 65,536	Cts	Page 133	NA

Configuration Parameter	Description	Units	Ref	Parameter Number
External Device	Selects an optional external feedback source. Choices: None, External Quadrature Encoder Default: None.	NA	Page 119	NA
External Device User Units	Used to calculate the Uu/Cts ratio, a scaling factor that relates user programming units to encoder counts.  <u>Note:</u> <i>User Units must be <math>\leq</math> Counts.</i>  Range: 0.000001 – 65,536.0 Default: 1.0	N/A	Page 122	1004 (Read/Write)
External Device Counts	Used to calculate the Uu/Cts ratio, a scaling factor that relates user programming units to encoder counts. Range: 1 – 65,536 Default: 1	N/A	Page 122	1005 (Read/Write)
External Device Position Range	Specifies the range of values allowed for the axis actual position. High Position Limit = Low Position Limit + Position Range. Low Limit: $102.4 \times (\text{External encoder Uu/Counts})$ High limit: $\text{MaxPosnUuExt}^6$ Default: 16777216.0	Uu	Page 122	1006 (Read/Write)
External Device Low Position Limit	Specifies the lower limit of the axis actual position. Low Limit: $-\text{MaxPosnUuExt}^6$ High Limit: $\text{MaxPosnUuExt}^6 - \text{MinPosnUuExt}^6$ Default: -8388608	Uu	Page 122	1007 (Read/Write)
External Device Counts per Motor Revolution	Selects the resolution of the External Quadrature Encoder. Range: 100 to 10,000,000 Default: 8,192	Counts	Page 122	NA
Axis Direction	Defines the positive axis direction as counterclockwise motor direction (Normal) or clockwise motor direction (Reverse). Choices: Normal, Reverse Default: Normal	NA	Page 123	NA

Configuration Parameter	Description	Units	Ref	Parameter Number
Software End of Travel	Enables or disables the Path Generator High Software EOT and Low Software EOT limits for the Linear axis positioning mode. Choices: Disabled, Enabled Must be set to Disabled if Axis Positioning Mode is Rotary. Default: Disabled	N/A	Page 124	4 (Read/Write)
High Software EOT Limit	Software end of travel limit in the positive direction. Low limit: -MaxPosnCmd High limit: +MaxPosnCmd High Software EOT Limit must be $\geq$ Low Software EOT Limit Default: +8388607.0	Uu	Page 124	2 (Read/Write)
Low Software EOT Limit	Software end of travel limit in the negative direction. Low limit: -MaxPosnCmd High limit: +MaxPosnCmd Low Software EOT Limit must be $\leq$ High Software EOT Limit Default: -8388608.0	Uu	Page 124	3 (Read/Write)
Max Velocity System	Specifies the maximum Path Generator axis velocity. Range: 0.1 – 6,000 Default: 4,000.0	RPM	Page 125	8 (Read)
Max Acceleration System	Specifies the maximum Path Generator rate of velocity increase. Range: 0.01 – 6E13 / Command Counts per Motor Rev Default: 40,000	Uu/sec <sup>2</sup>	Page 125	12 (Read)
Max Deceleration System	Specifies the maximum Path Generator rate of velocity decrease. Range: 0.01 – 6E13 / Command Counts per Motor Rev Default: 40,000	RPM/sec	Page 125	14 (Read)
Max Jerk	Specifies the maximum Path Generator rate of change in acceleration or deceleration. Range: MinJerkCmd <sup>6</sup> – MaxJerkCmd <sup>6</sup> Default: 100,000,000,000,000.0 (1E15)	Uu/sec <sup>3</sup>	Page 125	16 (Read/Write)

Configuration Parameter	Description	Units	Ref	Parameter Number
Error Stop Deceleration	Maximum Path Generator deceleration allowed for a normal stop. Low limit: MinAccUu High limit: The lesser of MaxAccCmd <sup>6</sup> or Max Deceleration System in Uu/sec <sup>2</sup> Default: 1,000,000.0	Uu/sec <sup>2</sup>	Page 130	1013 (Read/Write)
Error Stop Jerk	Maximum Path Generator jerk allowed for a normal stop. Low limit: MinJerkUu <sup>6</sup> High limit: The lesser of MaxJerkLim or MaxJerkCmd <sup>6</sup> Default: 10,000,000.0	Uu/sec <sup>3</sup>	Page 130	1014 (Read)
Feedback Moving Deadband	Specifies a deadband range for External Device Actual Velocity to allow proper operation of the Feedback Moving status bit. Range: 0–200,000 × Command Position Resolution Default: 100.0	Uu/sec	Page 130	1024 (Read/Write)

## Axis Parameter Descriptions

### Stop Axis on FTB Error

When enabled, an FTB communications fault or a failure to configure the FTB will cause a normal stop on the axis. Motion on the axis stops and the axis goes to the ErrorStop state.

If the axis will be configured to use inputs that are connected through the FTB, this parameter should be set to Enabled.

Inputs that can be connected through the FTB include Channel A and Channel B signals from an external quadrature encoder, Overtravel switches, and a Home switch. These inputs are configured on the FTB Inputs tab.

Type of Error	Fault Action
FTB communications fault when Stop on FTB Error is enabled.	Generates a normal stop error for the associated axis.
Failure to configure FTB when Stop on FTB Error is enabled.	
FTB communications fault when Stop on FTB Error is not enabled	Generates a warning for the associated axis.
Failure to configure FTB when Stop on FTB Error is not enabled.	

The following errors are independent of the Stop on FTB Error setting:	
Type of Error	Fault Action
Open wire on any 5Vdc input (unless input is for quadrature encoder)	Generates a warning for the associated axis.
Open load on any 24Vdc output	
Short circuit on any 24Vdc output	
Quadrature error for quadrature encoder input used as axis 1–4 feedback.	Generates a fast stop error for the associated axis.
Quadrature error for quadrature encoder inputs on axis 1–5 (non-feedback on axis 1–4).	Generates a warning for the associated axis.
Open wire on a 5Vdc input used as axis 1–4 quadrature signal (encoder used for feedback)	Generates a fast stop error for the associated axis
Open wire on a 5Vdc input used as axis 1–5 quadrature signal (encoder not used for feedback on axis 1–4)	Generates a warning for the associated axis.
An FTB communications fault if 5Vdc inputs are used for slave axis 1–4 quadrature feedback.	Generates a fast stop error for the associated axis.
An FTB communications fault if 5Vdc inputs are used for axis 1–5 quadrature encoder (non-feedback on axis 1–4).	Generates a warning error for the associated axis.
Quadrature error for an axis 5 quadrature encoder	Freezes the axis 5 encoder data, generates an error code and initiates an internal linear rampdown from the last valid encoder velocity and position. The linear rampdown method prevents instantaneous stop of CAM or follower slave axes using axis 5 encoder data as a master axis.
Open wire error on input for an axis 5 quadrature encoder	
FTB communications fault if axis 5 encoder data is from FTB	

## Position Feedback Source

To use an internal serial encoder, select Motor Encoder.

To use an external quadrature encoder, select External Device.

If External Device is selected, Axis x Encoder A Channel and Axis x Encoder B must be assigned to a valid faceplate or FTB input, and the External Device parameter must be set to External Quadrature Encoder.

## Axis Positioning Mode

Selects Linear or Rotary axis positioning.

In Linear mode, the Software End of Travel limits can be used to restrict axis motion. When Axis Positioning Mode is Linear and Software End of Travel is Enabled, the following conditions must be met:

- Low Software EOT Limit must be greater than or equal to MinPosnUu
- High Software EOT Limit must be less than or equal to MaxPosnUu

MaxPosnUu and MinPosnUu are calculated in Formulas for Computing Maximum Data Limits in User Units and Formulas for Computing Minimum Data Limits in User Units, subsections of 4.3.5.

Rotary mode allows the controller to command continuous motion in either direction, where the position will roll over every time it traverses the position range.

## Drive Status Input

Selects whether a status output from the analog servo drive is connected to the Drive Status input on the FTB and, if so, selects the operating mode of the input. Refer to the servo drive documentation to determine the most appropriate choice.

**Drive Ready** – The Drive Status input is enabled when the servo drive is supplying power to control the servo. Drive Status must be turned on within 500ms after Drive Enable is turned on or an error will occur. Drive Status must be turned off within 500ms after Drive Enable is turned off or an error will occur.

**Drive Available** – The Drive Status input is enabled when the servo drive has power available to control the servo. Drive Status may be turned on prior to Drive Enable being turned on and must remain on while Drive Enable is on or an error will occur.

**Disabled** – There is no feedback indicating the drive status. Should be used if the Analog Servo drive does not provide a status signal compatible with either Drive Ready or Drive Available.

## Motor Encoder Mode

Selects the *Incremental* or *Absolute* mode encoder that is mounted on the motor.

In *Incremental* mode, encoder counts are not retained through a power cycle and encoder battery alarms are not reported.

In *Absolute* mode, the serial encoder is used as an absolute type encoder by adding a battery pack to retain servo position while system power is off. In this mode, the serial encoder maintains position if system power is cycled. In *Absolute* mode, encoder battery alarms are reported.

For details about digital encoder operation, refer to Appendix B

## Motor Encoder User Units, Motor Encoder Counts

Used to calculate the Uu/Cts ratio, a scaling factor that relates user programming units to encoder counts. This ratio should be determined before configuring the axis. For calculation examples, refer to Preliminary Calculations in Section 4.3.5.

---

**Note:** *When External Quadrature Encoder is selected for feedback, the Motor Encoder User Units and Motor Encoder Counts values have no effect on motor control. However, for PM EtherCAT servos, they do affect the scaling of the built-in motor encoder velocity and position parameters, which can be read using the MC\_ReadParameter(s) function block. These parameters are updated regardless of the feedback selection. One possible use of this feature would be to set up the same scaling for the motor encoder and the external encoder to monitor both sets of parameters to detect slippage. Because the motor encoder velocity and position should always be correct, if they do not match the reported external velocity and position, then there is mechanical slippage.*

---

## Motor Encoder Counts per Motor Revolution

Selects the resolution of the motor encoder. This value must be compatible with the motor specified in Drive Type.

## Motor Encoder Position Range

Specifies the range, in user units, of values allowed for the axis actual position.

When moving in the positive direction, the Actual Position will roll over to the Low Position Limit when the High Position Limit is reached. High Position Limit = Low Position Limit + Position Range.

When the Software End of Travel parameter is set to *Disabled*, the Position Limits can be used for continuous rotary applications. The Position Range should always be set one User Unit smaller than the desired cycle. For example, a 360° machine would have a High Position Limit setting of 359. At the next count past 359, the count would roll over to the value set in the Low Position Limit parameter (0 in this example).

For proper operation, the range must always be greater than the distance traveled by the axis in one position loop sample time (normally 2ms).

When *Motor Encoder* is not the Position Feedback Source, the Motor Encoder always operates in rotary mode.

### Motor Encoder Low Position Limit

When moving in the negative direction, the Actual Position will roll over to the High Position Limit when this value is reached. High Position Limit = Low Position Limit + Position Range.

When the Software End of Travel configuration is set to *Disabled*, the Position Range can be used for continuous rotary applications. For proper operation, the range must always be greater than the distance traveled by the axis in one position loop sample time (normally 2 ms).

### Motor Encoder Maximum Positive RPM Limit

Motor encoder maximum positive RPM limit with a range of 10–10000 rpm. If the motor encoder speed exceeds the limit, axis fast stop error E5h will be reported. Response time of the limiter is 1 ms or less.

### Motor Encoder Maximum Negative RPM Limit

Motor encoder maximum negative RPM limit with a range of 10–10000 rpm. If the motor encoder speed exceeds the limit, axis fast stop error E6h will be reported. Response time of the limiter is 1 ms or less.

### External Device

Allows you to enable position feedback data from an external quadrature encoder.

The PMM supports up to five external quadrature encoders with a Marker input (also called the index or Z channel) for each.

If Position Feedback Source is set to External Device, this parameter must be set to External Quadrature Encoder, and the encoder inputs must be assigned to valid faceplate or FTB inputs. For valid input configurations, refer to the sub-section Input Configuration for External Position Feedback, below.

The external quadrature encoder must be wired in phase with the encoder that is in use on the motor. A procedure for verifying that the encoders are wired correctly is provided in the sub-section Verifying Correct Quadrature Encoder/Digital Encoder Phase, below.

### External Quadrature Encoder not used for Motor Control Feedback (Position Feedback Source set to *Motor Encoder*)

In this configuration, the external device can be used to report feedback data for devices other than the servo motor. The device can report position data to the RX3i and can be used by the application as a position source. You can configure and scale the device to have all the attributes (user units to counts, low position limit and position range) of the device when used for motor position feedback.

If not selected as the Position Feedback Source, an external quadrature encoder operates in *Rotary* mode and may be used to provide auxiliary position information.



## External Quadrature Encoder used for Motor Control Feedback (Position Feedback Source set to *External Device*)

In this configuration, the servo position loop of the axis is controlled from a position feedback device (external quadrature encoder) instead of the position encoder mounted on the motor.

### **WARNING**

If the external quadrature encoder is used for motor feedback, it must be mechanically linked to the axis movement at all times when the drive is enabled. This means that any movement of the servo axis will cause in-phase movement of the external quadrature encoder. Any attempt to enable an axis configured for Feedback Source: External Quadrature Encoder when the external quadrature encoder is not mechanically connected is a potentially hazardous situation.

---

## Input Configuration for External Position Feedback

Axes can be configured to use an external encoder connected to the quadrature encoder input. When an axis is configured so that the Feedback Source is *External Quadrature Encoder*, that axis will use the associated encoder, which must be configured as an input on the *FTB Inputs* tab (Axis 5 external encoder inputs can also be configured on the *FP Inputs* tab). For example, Axis 1 will use Encoder 1 Channel A and Encoder 1 Channel B, Axis 2 will use Encoder 2 Channel A and Encoder 2 Channel B, etc.

If a Find Home cycle will be used to initialize axis position, a Marker input, also called the index or Z channel, is required. (This input is not available on all quadrature encoders.) Only one marker can be configured per channel.

---

**Note:** *Inputs that can be configured for encoder A or B signals must be assigned in pairs. For example, when FTB\_I19 is configured as encoder 3A, FTB\_I20 is automatically configured as 3B.*

---

### PMM345 Inputs Configurable for Encoder

PMM input	Level	1A	1B	1Z	2A	2B	2Z	3A	3B	3Z	4A	4B	4Z	5A	5B	5Z
FP_I1	24 V													x		
FP_I2	24 V														x	
FP_I3/Q1	24 V															x
FP_I4/Q2	24 V															x
FP_I5	24 V															x
FP_I6	24 V															x
FP_I7	24 V															x
FP_I8	24 V															x
FTB_I17	5 V	x												x		
FTB_I18	5 V		x												x	
FTB_I19	5 V			x				x						x		x
FTB_I20	5 V						x		x						x	
FTB_I21	5 V				x											
FTB_I22	5 V					x										
FTB_I23	5 V							x						x		
FTB_I24/Q9	5 V								x						x	
FTB_I25/Q10	5 V									x	x					x
FTB_I26	5 V											x	x			
FTB_I27/Q11	5 V										x					
FTB_I28/Q12	5 V											x				

When an axis is configured for external feedback, the Find Home cycle uses the 24Vdc Home Switch input configured on the module faceplate or the FTB inputs. Similarly, the Axis Overtravel Limit Switch inputs are configured as faceplate or FTB inputs. Axis strobes will use the configured Touch Probe inputs and are subject to analog strobe timing specifications.

A table showing all available input selections is provided in PMM Faceplate I/O Functions Summary above.

### Verifying Correct Quadrature Encoder/Digital Encoder Phase

The quadrature encoder must be wired in phase with the encoder that is in use on the motor. The procedure for checking the correct operation is as follows:

- Wire the external quadrature encoder to Axis 1.
- Set the Position Feedback Source parameter to External Device.
- Set the External Device parameter to External Quadrature Encoder.
- Turn off the position loop by setting the Position Loop Time Constant to zero.
- Set velocity feed-forward to 100%.
- Use the MC\_JogAxis function block to jog the axis.
- Confirm that the actual motor velocity reported matches the commanded velocity.

For example: If the reported Axis 1 actual commanded velocity is positive and increasing, the reported actual velocity should also be positive and increasing.

If the velocities are out of phase (i.e. positive/negative) reverse the encoder channel A and channel B inputs for the external encoder and repeat the above test. If this does not fix the problem, consult the documentation that came with your encoder.

- Once the actual motor velocity and commanded velocity match, the verification process is complete.

### External Device User Units, External Device Counts

Used to calculate the Uu/Cts ratio for the External Quadrature Encoder, a scaling factor that relates user programming units to encoder counts. This ratio should be determined before configuring the axis. For calculation examples, refer to Preliminary Calculations in Section 4.3.5.

### External Device Position Range

Specifies the range of the actual position of the axis.

### External Device Low Position Limit

Specifies the low limit of the range of the actual position of the axis.

When moving in the negative direction, the Actual Position will roll over to the High Position Limit when this value is reached. High Position Limit = Low Position Limit + Position Range.

When the Software End of Travel configuration is set to *Disabled*, the Position Range can be used for continuous rotary applications. For proper operation, the range must always be greater than the distance traveled by the axis in one position loop sample time (normally 2 ms).

### External Device Counts Per Motor Revolution

Selects the resolution of the External Quadrature Encoder.

### External Encoder Maximum Positive RPM Limit

For an analog servo, the maximum positive external encoder RPM limit with a range of 10–20000 RPM. If the motor encoder speed exceeds the limit, axis fast stop error E5h will be reported. Response time of the limiter is 1 ms or less.

### External Encoder Maximum Negative RPM Limit

For an analog servo, the maximum negative external encoder RPM limit with a range of 10–20000 RPM. If the motor encoder speed exceeds the limit, axis fast stop error E6h will be reported. Response time of the limiter is 1 ms or less.

### Motor Velocity at 10 Volts

For an analog servo, the actual servo velocity commanded when 10 Vdc is commanded to the analog output. Value must match the configuration of the Analog Servo. Force Servo Velocity parameters 1311 and 1320 can be used to command a velocity to determine the correct value empirically if necessary. It is recommended that this value be at least 10% greater than Max Velocity System to allow proper operation at max velocity.

## Minimum Velocity Output

Sets the minimum velocity output (millivolts) for analog servos. When Minimum Velocity Output is set to zero the axis may finish a move with non-zero position error. By increasing the minimum velocity output any non-zero velocity command will be at least the minimum velocity. The Minimum Velocity Output should be increased until the servo can pull in to  $\pm 1$  count of position error. The recommended starting value is 5 – 10mv

## Over Travel Limit Switch

Selects whether the axis uses the hardware over travel limit switch inputs.

**DISABLED:** The over travel inputs (default location FP IN5 – FP IN8) may be used as general-purpose motion program flow control and program branching inputs.

**ENABLED:** The PMM checks the axis over travel inputs continuously, every 10 ms whenever the Drive Enabled status is true.

The overtravel inputs are operated in the fail-safe mode. If either limit switch opens (the input goes to logic zero, Off) all motion is immediately commanded to stop. No deceleration control is active; the servo velocity command is set to zero. The solid-state axis enable relay will not open until after the Enable Drive command is set to zero. An error code indicating which limit is tripped is reported to the Axis Error Code and the axis transitions to the ErrorStop state. At this point, the only motion command allowed is MC\_JogAxis, which you can use to back away from the Limit Switch. You may also manually move the disabled axis off the limit switch. After the error condition is cleared, normal operation may resume.

This parameter does not apply to Synthetic Motors (drive type Synthetic) because there is no actual motor overtravel limit switch to activate.

---

**Note:** *If Over Travel Limit Switch is enabled, you must configure inputs for both positive and negative directions. If you enable over travel switches and only assign an input to an over travel switch in one direction, the other switch will default to being tripped in the PMM. As soon as the PMM starts up it will detect an over travel on the unassigned switch and stop motion. In this condition, the over travel condition cannot be cleared because the switch is not assigned to an input.*

---

## Axis Direction

For all PM EtherCAT servos, a configured axis direction of Normal defines the positive axis direction as counter clockwise (CCW) motor shaft rotation when viewed looking into the motor shaft. A configured axis direction of Reverse defines the positive axis direction as clockwise (CW) shaft rotation.

## Software End of Travel

Enables or disables the High Software EOT and Low Software EOT limits.

---

**Note:** *If Software EOT is Disabled, the High/Low Software EOT Limit parameters are hidden. The firmware internally sets the High/Low Software EOT Limits to MaxPosnUu and -MaxPosnUu as follows:*

*MaxPosnLim = 4E10 counts*

*MaxPosnUu = [MaxPosnLim × (Uu/cts ratio from HWC)]*

---

For information on how MaxPosnUu is calculated, refer to Computing Data Limits in User Units in Section 4.3.5.

---

**Note:** *When a linear axis is being used as a master, it is recommended that software EOTs be set on the slave axes instead of the master.*

---

## High Software EOT Limit

The High Software EOT limit is used when Software End of Travel is set to Enabled. If the limit is enabled and the axis is commanded to go to a position greater than the High Software EOT value, an error will result and the PMM will not allow axis motion.

The High Software EOT Limit is ignored for slave axis motion resulting from master axis commands. The limit applies only to slave axis motion resulting from internally generated jog and motion program commands. The limit is always ignored for Move at Velocity commands.

If the High Software EOT Limit is greater than the High Position Limit (Low Position Limit + Position Range), the High Software EOT Limit will internally be set equal to the High Position Limit. Axis error code 0x17 will also be reported, indicating that the limit has been adjusted.

The High Software EOT Limit is ignored for Jog commands if the axis is in the ErrorStop state.

## Low Software EOT Limit

The Low Software EOT limit is used only when the Software End of Travel parameter is set to Enabled. If the limit is enabled and the PMM is programmed to go to a position less than the Low Software EOT, an error will result and the PMM will not allow axis motion. If the Follower control loop is enabled, the High Software EOT Limit is ignored for slave axis motion resulting from master axis commands. The limit only applies to slave axis motion resulting from internally generated jog and motion program commands. The limit is always ignored for Move at Velocity commands.

If the Low Software EOT Limit is enabled and its value is more negative than the Low Position Limit, the Low Software EOT Limit will internally be set equal to the Low Position Limit. Axis error code 0x17 will also be reported, indicating that the limit has been adjusted.

The Low Software EOT limit is ignored for Jog commands if the axis is in the ErrorStop state.

## Max Velocity System

Specifies the maximum commanded axis velocity in RPMs. The Velocity Limit applies to the sum of all velocity command sources for an axis, including the internal path generator and external follower master axis commands. If a servo velocity command exceeds the limit, an error code will be reported and the servo command will internally be set to the limit value.

## Max Acceleration System

Specifies the maximum commanded rate of velocity increase in RPM/sec. This limit applies to the sum of all velocity command sources for an axis, including the internal path generator and external follower master axis commands. If a servo velocity command exceeds the limit, an error code will be reported and the servo command will internally be set to satisfy the limit value.

## Max Deceleration System

Specifies the maximum commanded rate of velocity decrease in RPM/sec. This limit applies to the sum of all velocity command sources for an axis, including the internal path generator and external follower master axis commands. If a servo velocity command exceeds the limit, an error code will be reported, and the servo command will internally be set to satisfy the limit value.

## Max Jerk

Specifies the maximum allowed rate of acceleration or deceleration change in  $Uu/sec^3$ . This limit applies to the sum of all velocity command sources for an axis, including the internal path generator and external follower master axis commands. If a servo velocity command exceeds the limit, an error code will be reported, and the servo command will internally be set to satisfy the limit value.

## Drive Disable Delay (ms)

The time delay from the time the zero-velocity command is received until the axis enters the Disabled state. Disable Delay is effective when the Drive Enabled bit is turned off or if a fault occurs that causes the axis to go to the ErrorStop state.

Because the PMM cannot command the servo when the axis is in the Disabled or ErrorStop states, there are times when axis should remain enabled. For example, if the servo runs into an End of Travel Limit and the axis was immediately disabled due to the error, the servo could continue moving until it coasted to a stop. Thus, to allow the PMM to command and control a fast stop, the Drive Disable Delay should be longer than the worst-case deceleration time of the servo from maximum speed.

The Disable Delay may be used to control when torque is removed from the motor shaft. Applications using an electro-mechanical brake generally need time for the brake to engage prior to releasing servo torque. The delay should be set to a value longer than the engagement time for the brake.

Drive Type selects the PM EtherCAT Servo drive interface to be used with the PMM. This is important as it determines the data exchanged between the PMM and the EtherCAT based servo drive.

The default value of Synthetic specifies a Synthetic Motor, which can be used to test applications as they are being developed.

---

**Note:** *A module can have both real and synthetic axes configured, but the synthetic motors must be on higher numbered axes than the real motors.*

---

## Drive Type

The Drive Type selects the data to be exchanged between the PMM345 and the Servo Drive. The default selection is **PACMotion PSD411**. This is the selection to utilize with Emerson Servo Drive PSD411. The user can also select **Synthetic** to utilize simulated motor. The **Synthetic** drive type is useful for programming an application prior to having real motors and drives.

## Motor Velocity Limit

The motor velocity limit maximum is equal to 1.1 times the maximum velocity in RPM of the configured motor. For maximum motor speed, refer to the corresponding column in the table in Section 2.3.6, Drive Type.

## Current Limit

Specifies the maximum allowed drive current, in percent of available continuous drive current, to be produced by the servo motor at commanded velocity. The current limit range covers the drive continuous current rating (current limit = 100%) to the peak drive current rating (current limit = 300%). Drive current output is related to motor torque. Thus, setting the current limit yields the ability to approximately set the maximum motor torque. To determine the actual value of torque output available at a given velocity, refer to the motor torque curve and Motor Torque Constant in the appropriate servo motor manual.

## Position Lag Monitoring

When enabled, allows detection of position lag (the difference between commanded position and actual position). The maximum lag allowed is specified by the Max Position Lag parameter.

## Max Position Lag

Displayed when Position Lag Monitoring is set to Enabled. Specifies the maximum Position Lag (Commanded Position - Actual Position) allowed when the PMM is controlling a servo. Max Position Lag should normally be set to a value 10% to 20% higher than the highest position lag encountered under normal servo operation. Must be less than or equal to Max Position Error.

The Max Position Lag range is:

$$\text{Low limit} = 0.001 \times \text{MaxVelocitySystem (RPM)} \times (1/60 \text{ min per sec}) \\ \times \text{feedback encoder counts per rev} \times \text{Uu/cts}$$

$$\text{High limit} = 100 \times \text{feedback encoder counts per rev} \times \text{Uu/cts}$$

If Velocity Feedforward is not used, Max Position Lag can be set to a value approximately 20% higher than the position lag required to produce a 4000-rpm command. The position lag required to produce a 4000-rpm command with 0% Velocity Feedforward is:

$$\text{Position Error Lag (user units)} = \frac{\text{Position Loop Time Constant (ms)} \times \text{Servo Velocity @ 4000 rpm (user units/sec)}}{1000}$$

### Max Position Lag Example

The User Units:Counts ratio is 1:2 and the Position Loop Time Constant is 50 ms.

#### Step 1:

$$\text{Calculate servo velocity at 4000 rpm} = \frac{(0.5 \text{ user units/count}) \times (8192 \text{ counts/rev}) \times (4000 \text{ revs/minute})}{(60 \text{ seconds/minute})}$$

$$= 273067 \text{ user units/second}$$

#### Step 2:

$$\text{Calculate Position Error at 4000 rpm} = \frac{(50 \text{ ms}) \times (273067 \text{ user units/second})}{1000 \text{ ms/second}}$$

$$= 13653 \text{ user units}$$

If Velocity Feedforward is used to reduce the following error, a smaller error limit value can be used, but in general, the error limit value should be 10% to 20% higher than the largest expected following error.

### Max Position Error

An absolute value used to determine when the servo is out of sync and should be stopped. (Position Error = Commanded Position – Actual Position.)

### In Position Zone

When the Position Error is less than or equal to the In Position Zone value, the In Zone axis status bit will be ON. In cases where the servo stops and is required to reach the target actual position before proceeding, the In Zone status bit (PN1205) can be tested for an ON state before activating the next motion function block.



## Position Loop Time Constant

The Position Loop Time Constant (in ms) sets the position loop gain and determines the response speed of the closed position loop.

This value specifies the amount of time required for the servo velocity output to reach 63% of its final value when a step change occurs in the Velocity command. The lower the value, the faster the system response is. Values that are too low will cause system instability and oscillation.

---

**Note:** For accurate commanded velocity profile tracking, Position Loop Time Constant should be 1/4 to 1/2 of the minimum system acceleration or deceleration time. For example, if the fastest acceleration that must occur occupies 100ms, the Position Loop Time Constant should be between 25 to 50ms. To maintain system stability, use the largest value possible.

---

For users familiar with servo bandwidth expressed in radians/sec:

$$\text{Bandwidth (radians/sec)} = 1000 / \text{Position Loop Time Constant (ms)}$$

For users familiar with servo gain expressed in ipm/mil:

$$\text{Gain (ipm/mil)} = 60 / \text{Position Loop Time Constant (ms)}$$

## Gain / Bandwidth / Position Loop Time Constant

Gain (ipm/mil)	Bandwidth (radians/sec)	Position Loop Time Constant (ms)
0.5	8.3	120
0.75	12.5	80
1.0	16.6	60
1.5	25.0	40
2.0	33.3	30
2.5	41.7	24
3.0	50.0	20

For applications that do not require feedback control or employ very crude positioning systems, an **Open Loop Mode** exists. To select Open Loop Mode, set the Position Loop Time Constant to 0, which disables the positioning loop. Note that in Open Loop Mode, the only way to generate motion is to program a non-zero Velocity Feedforward. The Position Error is no longer used to generate motion because Position Error is based on position feedback and Open Loop Mode ignores all feedback.

## Velocity Feedforward

Velocity Feedforward specifies the percentage of commanded velocity that is added to the PMM's position loop velocity command output. This term is used to compensate for position error when the motor is moving.

Increasing Velocity Feedforward causes the servo to operate with faster response and reduced position error. The optimum value for each system must be determined individually. For digital servos, a Velocity Feedforward value of 100% is a good starting point. However, it is recommended that the Load Inertia Ratio and Position Loop Time Constant be properly adjusted before setting Velocity Feedforward to a non-zero value. The servo system capabilities will determine the optimum value. Refer to Appendix for more information. If Velocity Feedforward is changed, Max Position Lag may require adjustment.

## Velocity Loop Proportional Gain

Analog Servo Torque Mode only. Sets the velocity regulator proportional gain. The proportional gain is multiplied by velocity error (velocity command - velocity feedback) to generate the portion of the torque command due to the proportional term. Correctly setting this value determines how well the velocity regulator performs in the control system. Refer to the section on tuning in the Appendix for a method to correctly set this value.

## Velocity Loop Integral Gain

Analog Servo Torque Mode only. Sets the velocity regulator integral gain. The integral gain is the term multiplied by the velocity error (velocity command - velocity feedback) to generate the portion of the torque command due to the integral term. Correctly setting this value determines how well the velocity regulator performs in the control system. Refer to the section on tuning in the Appendix for a method to correctly set this value.

## Torque Command Filter

Analog Servo Torque Mode only. Sets the torque command filter cutoff frequency 3db point in Hz. A value of 0 disables the filter. The torque command filter allows the user to activate a low pass filter for the velocity regulator output. The filter is used to keep the controller from exciting a machine resonance. Refer to Appendix Section C-3, Tuning a PM EtherCAT Servo for details on setting this value.

## Load Inertia Ratio

Analog Servo Torque Mode only.

Applies a gain to the velocity control loop to match load inertia ( $J_L$ ) to motor inertia ( $J_M$ ). A suggested initial value for Load Inertia Ratio is:

$$\text{Load Inertia Ratio} = \frac{\text{Load Inertia } (J_L)}{\text{Motor Inertia } (J_M)} \times 256$$

If the motor shaft is not attached to a load, a value of 0 should be used.

This parameter does not apply to axes using Synthetic Motors (drive type Synthetic).

### Error Stop Deceleration

Specifies the maximum deceleration allowed when the axis undergoes an error stop. Normal Stop Deceleration must be  $\leq$  Deceleration Limit.

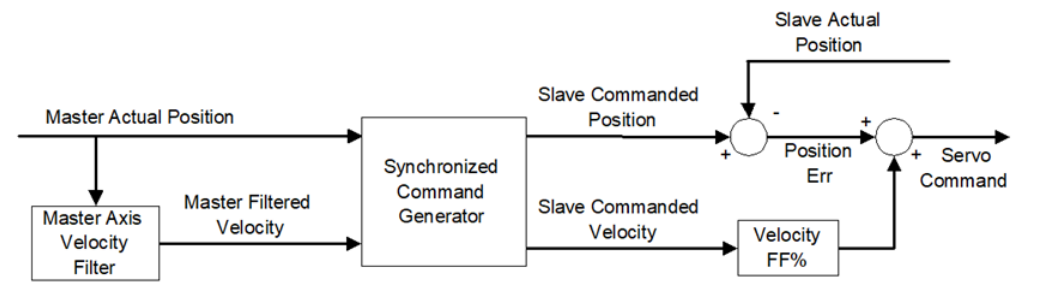
### Error Stop Jerk

Specifies the maximum jerk allowed during an error stop of the axis. Error Stop Jerk must be  $\leq$  Max Jerk.

### Master Axis Velocity Filter

Specifies filter width of the master axis velocity signal. It is applied when this axis is in the Synchronous state and is following an Actual Position Source. The filter is a moving average of the derived velocity.

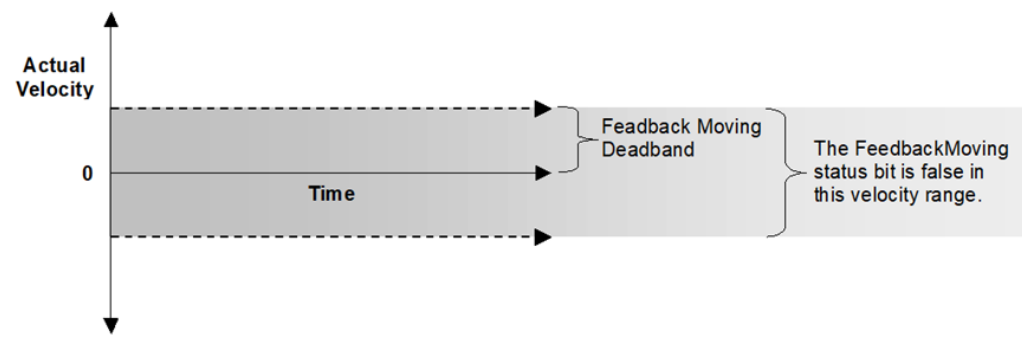
**Figure 62: Servo Command Loop showing Master Axis Velocity Filter**



### Feedback Moving Deadband

Specifies a deadband range in UU/sec for setting the actual motor status bit, FeedbackMoving. When the actual motor velocity is within this range, the axis status bit, FeedbackMoving is OFF.

**Figure 63: Behavior of Feedback Moving Deadband**

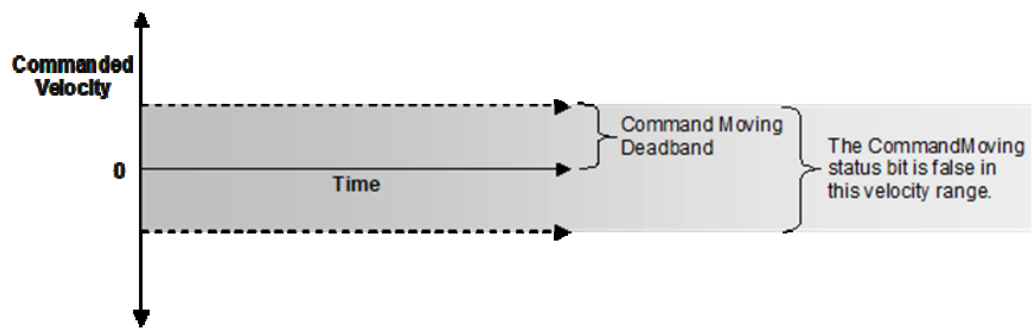


## Command Moving Deadband

Specifies a deadband range in UU/sec for setting the commanded axis velocity status bit, CommandMoving. When the commanded motor velocity is within this range, the axis status bit, CommandMoving is ON.

Because the states of the CommandMoving, Accelerating, Decelerating and ConstantVelocity status bits are determined based on changes in the commanded velocity of the axis, this deadband introduces lag in setting these bits. Unless the axis is using synchronous motion (the axis is following a master), Command Moving Deadband should be set to zero.

**Figure 64: Behavior of Command Moving Deadband**



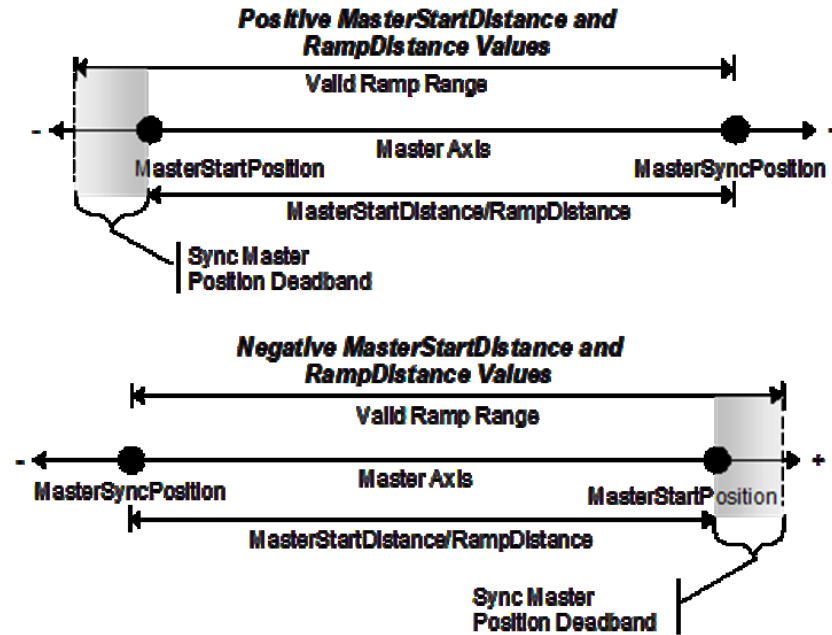
## Sync Master Position Deadband

Specifies a positional deadband, in Master Axis Uu, to be applied to the Master's observed position by this axis when it is a Slave. The deadband may be applied to master positions such as the Master CAM Rollover position and the start position of Ramp onto a CamIn or GearInPos operation.

For MC\_GearInPos:  $\text{MasterStartPosition} = \text{MasterSyncPosition} - \text{MasterStartDistance}$

For MC\_CamIn:  $\text{MasterSyncPosition} = \text{Current Master Position} + \text{RampDistance}$

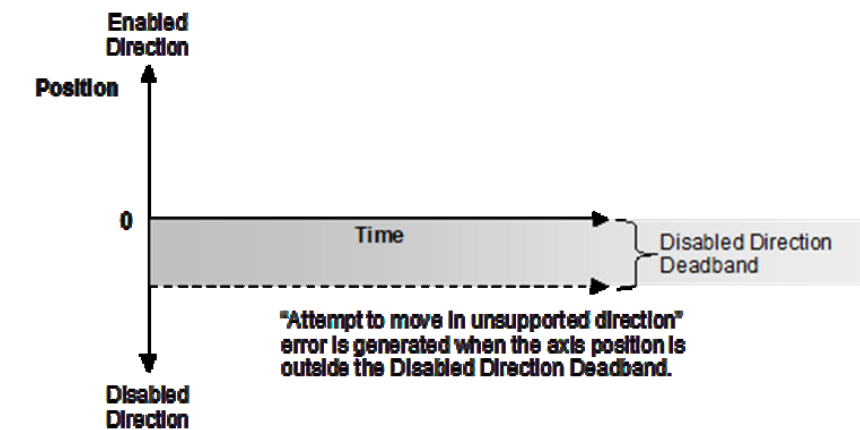
Figure 65: Behavior of Sync Master Position Deadband



### Disabled Direction Deadband

Specifies a deadband range, in Uu, in the direction not enabled by MC\_POWER. If the axis position exceeds this range, the 0308H error, *Attempt to move in unsupported direction* is generated.

Figure 66: Behavior of Disabled Direction Deadband



### Command Position Range

(Axis 5 only) Specifies the range of values allowed for the axis commanded position. High Position Limit = Low Position Limit + Position Range. Applies only if Axis Positioning Mode is set to Rotary.

### Command Low Position Limit

(Axis 5 only) Specifies the lower limit of the axis commanded position. Applies only if Axis Positioning Mode is set to Rotary.

### Command Position Resolution

(Axis 5 only) Specifies the resolution of the Axis 5 path generator. Because the Axis 5 path generator can be used with various Uu/Cts ratios, this parameter provides a way to adjust the resolution on the Axis 5 path generator.

### Command Counts Per Motor Revolution

(Axis 5 only) Specifies the counts per revolution of the Axis 5 path generator.

## 4.3.6 Advanced Parameters

The Advanced tab allows up to 32 parameters and associated data to be entered. Parameters configured on this tab do not apply to Analog mode axes.

Do not use this tab to configure parameters that are already configured on the Axis tabs, for example parameters 2 (SWLimitPos), 3 (SWLimitNeg), and 1008 (InPositionZone). Attempting to set an unsupported parameter will cause the configuration to be rejected.

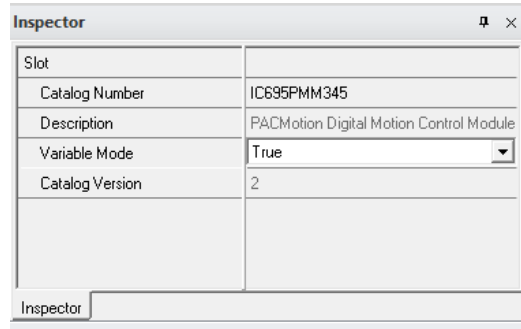
## 4.3.7 Power Consumption

This is a display-only tab that displays the power required by the PMM module.

## 4.3.8 Terminals

This tab is displayed when the Variable Mode property of the module is set to True.

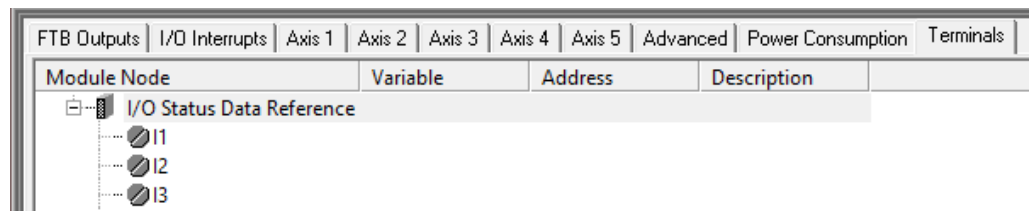
**Figure 67: Variable Module in the Inspector Tab**



When Variable Mode is selected, the I/O status bits are referenced as I/O variables that are mapped to the bits on this configuration tab. The use of I/O variables allows you to configure the module without having to specify the reference addresses to use for the module data. Instead, you can assign variable names directly to the status bits.

For a summary of I/O status data, refer to Section 4.3.2, PMM Status Data.

**Figure 68: PME Terminals Tab**



# Section 5 PACMotion Function Block Operation

This chapter provides an overview of PACMotion functions and function blocks and their common operational features.

Topics Covered:

Section 5.1 PACMotion Function and Function Block Types

Section 5.2 Behavior of Motion Instructions

5.2.1 Instance Data

5.2.2 Immediate Response vs. Deferred Response Function Blocks

5.2.3 Administrative vs. Motion-Generating Functions and Function Blocks

5.2.4 Function Block Triggering (Enabled vs. Executed Instructions))

Section 5.3 Function and Function Block Parameters

5.3.1 Permissives, Constants, Variables or Flow used with Motion Function Blocks

5.3.2 EN Input and ENO Output

5.3.3 Input Parameters

5.3.4 Output Parameters

5.3.5 In\_Out Parameters

5.4 Data Types and Structures

5.4.1 HWC Parameter Linked Data Types

5.4.2 Enumerated Data Types

5.4.3 CAM Profile Linked Data Types

5.5 Axis States

5.5.1 Axis State Diagram

Section 5.6 Synchronized Motion

For details about the use and operation of specific instructions, refer to Section 6, PACMotion Instruction Set Reference.



## 5.1 PACMotion Function and Function Block Types

The following table contains a reference to all motion functions and function blocks supported by the PMM module. For details on usage, refer to Section 6, PACMotion Instruction Set Reference.

Context	Functional Description	MC_Function Block
Reset	Clear axis errors	MC_Reset
	Clear all errors on a module	Module Reset
Axis Power-up and Initialization	Apply power to the axis	MC_Power
	Establish or recalibrate axis position	MC_SetPosition
	Execute a Find Home sequence to establish a valid actual position	MC_Home
Touch Probe	Capture actual axis position of an axis in response to a touch probe event	MC_TouchProbe
	Terminate a touch probe function	MC_AbortTrigger
Discrete Motion	Execute a Find Home sequence to establish a valid actual position	MC_Home
	Move to a specified absolute position	MC_MoveAbsolute
	Move a specified distance in addition to the prior commanded distance.	MC_MoveAdditive
	Move a specified distance relative to the actual position at the time of the execution	MC_MoveRelative
	Move a specified relative distance in addition to an existing motion	MC_MoveSuperimposed
Continuous Motion	Jog axis forward or backward	MC_JogAxis
	Move axis at a specified velocity	MC_MoveVelocity
	Move a specified distance in addition to the actual position at the time of the execution	MC_MoveAdditive
Setting Override Parameters	Set velocity, acceleration, deceleration and jerk override factors for all functions that command motion on an axis	MC_SetOverride
Controlled Stops	Transition an axis to the Stopping state	MC_Stop
	Transition an axis to the Standstill state	MC_Halt
Axis Synchronization	Start up to eight axes at the same time	MC_SyncStart
	Start up to eight axes with a delay relative to each other	MC_DelayedStart
Axis Gearing	Synchronize a slave axis to a master axis at a specified velocity ratio	MC_GearIn
	Synchronize velocity and position of a slave axis to a master axis.	MC_GearInPos
	Terminate a MC_GearIn or MC_GearInPos function block	MC_GearOut
Superimposed Motion	Move a specified distance in addition to an existing motion	MC_MoveSuperimposed

Context	Functional Description	MC_Function Block
CAM Programming	Load a CAM profile from the CPU onto a PMM	MC_CamTableSelect
	Engage a CAM profile on a master and a slave axis	MC_CamIn
	Disengage a slave axis from a CAM profile	MC_CamOut
	Delete a CAM profile from a PMM	MC_CamTableDeselect
Phasing	Command a phase shift on a CAM slave axis	MC_Phasing
CAM File Management	Read a CAM profile from the RX3i file system into CPU reference memory	MC_CamFileRead
	Copy a CAM profile from CPU reference memory into the RX3i file system	MC_CamFileWrite
	Retrieve information on CAM profile memory usage in a PMM	MC_LibraryStatus
Monitoring Axis Operation	Read the current state of an axis. For definitions of axis states, refer to Section 5.5, Axis States.	MC_ReadStatus
	Read the ErrorID of the current axis error or warning.	MC_ReadAxisError
	Read the actual position of an axis.	MC_ReadActualPosition
	Read the actual velocity of an axis	MC_ReadActualVelocity
	Read the commanded torque of an axis.	MC_ReadTorqueCommand
Reading/Writing I/O Reference IDs	Read the value of a discrete faceplate or FTB input	MC_ReadDigitalInput
	Read the value of a discrete faceplate or FTB output	MC_ReadDigitalOutput
	Write a value to a discrete faceplate or FTB output	MC_WriteDigitalOutput
	Read the value of an FTB analog input	MC_ReadAnalogInput
	Read the value of an FTB analog output	MC_ReadAnalogOutput
	Write a value to an FTB analog output	MC_WriteAnalogOutput
Reading/Writing Axis and Module Parameters	Read the value of a Boolean hardware configuration parameter or axis status flag	MC_ReadBoolParameter
	Read the value of an array of Boolean hardware configuration parameters or axis status flags	MC_ReadBoolParameters
	Read the value of an array of DWORD parameters	MC_ReadDwordParameters
	Read the value of an LREAL parameter.	MC_ReadParameter
	Read the value of an array of LREAL parameters.	MC_ReadParameters
	Change the value of a Boolean hardware configuration parameter	MC_WriteBoolParameter
	Change the values of an array of Boolean hardware configuration parameters	MC_WriteBoolParameters

Context	Functional Description	MC_ Function Block
	Change the values of an array of DWORD hardware configuration parameters	MC_WriteDwordParameters
	Change the value of an LREAL hardware configuration parameter	MC_WriteParameter
	Change the value of an array of LREAL hardware configuration parameters	MC_WriteParameters
Module-Level Operations	Control an output point based on axis position and position/time	MC_DigitalCamSwitch
	Specify parameters for data to be monitored in the Data Logging Window	MC_DL_Configure
	Start logging the data specified by an MC_DL_Configure function block instance	MC_DL_Activate
	Write data log file to CPU reference memory	MC_DL_Get
	Delete a data logging configuration	MC_DL_Delete
	Clear all errors on a module	MC_ModuleReset
	Read the most recent 100 PMM events	MC_ReadEventQueue

## 5.2 Behavior of Motion Instructions

Motion instructions consist of standard function blocks, which have instance data, and functions, which do not have instance data.

Motion functions and function blocks can be invoked from any block in a PACSystems application except an external block (for example, a C block).

### 5.2.1 Instance Data

Standard function blocks have instance data consisting of a structure variable. Instance data is located in RX3i memory and is automatically allocated when a function block is stored to the RX3i for the first time. Each function block used in the application program has its own instance data. The values of the instance data persist from one execution of the function block to the next so that the status of the function block execution is retained.

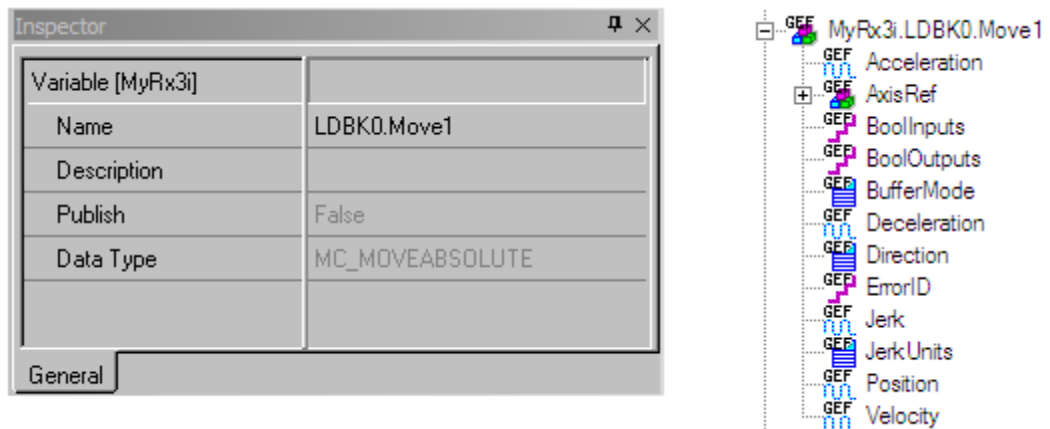
Each MFB has a unique data type that contains the instance data for that particular function block. For example, an MC\_MoveAbsolute function block has a corresponding data type MC\_MoveAbsolute that contains elements for input and output parameters as well as internal data.

For additional information on the use of instance data by function blocks, refer to Chapter 2 Program Organization in the PACSystems RX3i and RSTi-EP CPU Programmer's Reference Manual, GFK-2950.

Default initial input and output values are established for some input and output parameters. You can change the default initial values for any instance of a function block by editing the variable properties in Logic Developer. The initial values can be stored in Stop mode and remain in effect until the function block instance is executed.

For a discussion of how PACMotion manages instance data and MFB output parameters, refer to Section 5.3.4 sOutput Parameters.

**Figure 69: Definition of Structured Variable**      **Figure 70: Edit Instance Data in PME**



## 5.2.2 Immediate Response vs. Deferred Response Function Blocks

MFBs are divided into two execution types: immediate execute/*immediate response* and immediate execute/*deferred response*

### Immediate Execution/Immediate Response

Immediate Execute/Immediate response function blocks complete any required processing and return final results at the activation point (the point where the permissive logic leading to the instruction calls for the instruction to execute). The specifications for the execution time for the immediate response instructions can be found in the PACSystems RX3i and RSTi-EP CPU Reference Manual, GFK-2222.

When an immediate response function block is enabled, the CPU reads the immediate data from the PMM's shared memory. The CPU then writes that data to the appropriate reference memory as determined by the output parameters of the MFB. This all occurs during the same invocation of the MFB.

### Immediate Execution/Deferred Response

Certain function blocks, for example, those that command motion, may take a significant time to complete. The CPU may continue processing other data while the PMM executes the command.

When a deferred response MFB is executed, the CPU immediately sends the corresponding motion command to the specified PMM and axis. The CPU does not wait until the end of the current sweep (I/O scan) to initiate the command.

The function block has a default output data state that indicates its status when first received by the PMM's command queue. When the PMM notifies the CPU that an output has changed, the CPU updates the function block instance data with the latest data. The instance data then updates the function block outputs when the function block is encountered in the logic scan.

## 5.2.3 Administrative vs. Motion-Generating Functions and Function Blocks

Motion functions and function blocks are divided into two action types: Administrative and Motion.

Administrative functions and function blocks do not cause axis motion, while motion functions and function blocks control axis motion. For details on how these instructions affect the axis state, refer to Figure 81.

## 5.2.4 Function Block Triggering (Enabled vs. Executed Instructions)

Functions and function blocks are activated either by an Enable input (level-triggered) or by an Execute input (edge triggered).

For functions or function blocks that have an Enable input, the parameters are applied when Enable is true. In function blocks that use an Execute input, the input parameters are applied with the rising edge of the Execute input.

### Enabled Motion Function Blocks

The enabled MFBs are active only while Enable is ON. The Enable parameter is used with instructions that perform cyclical actions, for example MC\_DL\_Activate, MC\_ReadActualPosition, MC\_Power, MC\_DigitalCamSwitch, MC\_JogAxis and MC\_SetOverride.

### Multiple Instances

The programming environment supports multiple instances of enabled function blocks. However, in most applications using multiple instances of these function blocks is unnecessary and may lead to confusing behavior. Unless necessary for the application, it is advised that only one instance of these type blocks be used in a given program. The resulting program will be simpler to maintain.

When one instance of an enabled MFB supersedes another, the instance that is being superseded has its Busy output set false, or in the case of MC\_SetOverride the Enabled output is set false. The instance that is being superseded sets its Warning output true and generates an ErrorID to indicate that it has been superseded.

To provide additional notification that instance transitions are occurring, when one instance supersedes another, a warning is placed in the axis error code and an associated I/O fault is generated. The fault and axis error code are generated on the first detection of this event but are not logged on successive detections until an MC\_Reset is applied to the axis. Separate ErrorIDs and faults are generated for each enabled function block type so that the feedback in the axis error code and the fault table is specific to the function block type that is being superseded.

---

**Note:** *Superseding function blocks can cause confusing and erroneous behavior. It is strongly recommended that application logic monitor the Warning output of enabled function blocks or that warnings be reported in the I/O Fault Table. To log PMM warnings in the fault table, change the Log Messages in I/O Fault Table hardware configuration parameter from the default Errors Only setting to Errors\_Warnings.*

---

### Outputs of Enabled Function Blocks

Each enabled MFB has one or more enabling inputs, as well as a power flow input, EN. An enabled MFB may have Busy, Active and Done outputs, as well as other outputs. Behavior of the outputs of enabled MFBs is determined by the enabling input and the state of the processing of the MFB.

When the Enable input is set false, the outputs are set false. Note that it may take at least one host controller scan for the Busy, Done, Active, Valid or CommandAborted outputs to transition off when Enable is set false.

For general information about the effects of the MFB input power and Enable states on output behavior, refer to Section 5.3.4, Output Parameters.

### Changing the Axis on an Active Instance

A given instance of an enabled MFB is allowed to operate on only one axis at a time. If the instance is already operating on one axis and the input changes to different axis, the MFB instance will first be disabled on the original axis and will then be enabled on the new axis. This behavior is the same as if you explicitly set Enable to OFF on the original axis, changed the Axis parameter and then set Enable to ON.

### Multiple Instances Per Axis

Multiple instances of an enabled MFB can refer to the same axis in a single application. If two function blocks of the same type attempt to operate on the same axis, they are processed in the order they are encountered during the execution of the application logic. When a new instance invocation is encountered, the currently active instance for the function block type is terminated and the new instance invocation assumes control.

### Executed (Edge-Triggered) Motion Function Blocks

The *Execute* command input triggers the function block run at the rising edge (low to high transition) of the input. To repeat the function action, the *Execute* input must transition low for one MFB invocation prior to the new rising edge action. This type of triggering allows precise management of the instant a motion command is performed.

Input parameters are used with the rising edge of the *Execute* input. To modify input parameter(s), you must change the parameter(s) and trigger the function block again.

The falling edge of *Execute* does not stop or influence the execution of the function block. The action that was initiated by the function block will continue until it is completed or until another function block interrupts or stops the action.

It is possible for a function block to stop its previously-started action on an axis (or module). If the same instance of a function block is retriggered to execute *on the same axis*, the current execution of that instance on that axis will be aborted, and a new execution of that instance on that axis will be started with new input parameter values.

It is also possible for a function block instance to execute on more than one axis (or module) at the same time. If the same instance of a function block is retriggered to execute *on a different axis*, the current execution of that instance on the original axis will continue but will be unable to update its status to the instance data outputs. For the aforementioned reason, this is not a suggested programming mechanism unless required by the application. A new execution of the instance will be started on the new axis, and it will update its status to the instance data outputs. If the instance is started on still other axes, it is always the last started execution of the instance that has the ability to update its status to the instance data outputs.

You should carefully consider the logical state of an *Execute* input permissive during power cycle or RX3i restart when deciding whether to use a retentive or non-retentive data type as the *Execute* input permissive. Additionally, when using function blocks with *Execute* inputs in a subroutine, be careful not to call the subroutine with a one-shot action.

### Permissive Logic Examples for Executed Function Blocks

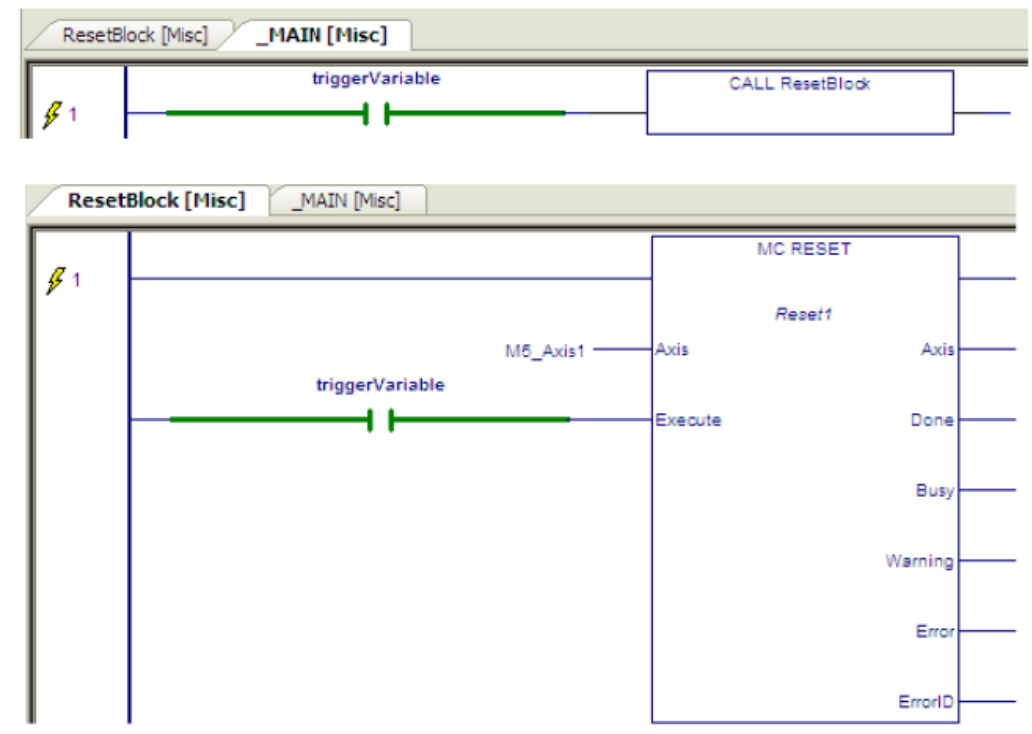
To start the execution of an edge-triggered MFB instance, you must ensure that the last time the MFB instance was called, its *Execute* input was FALSE and then call the MFB with a TRUE value for the *Execute* input.

#### Example: Resetting an Axis

In the following example, the same variable, *triggerVariable*, is used both to call *ResetBlock* and to execute the MC\_Reset function block. This type of structure is *not* recommended because MC\_Reset can be executed only once. After the initial call to *ResetBlock*, there is no way to enable MC\_Reset with a FALSE value for the *Execute* input.

#### Not Recommended

Figure 71: Resetting an Axis - Method Not Recommended



#### Recommended

Calling *ResetBlock* with a different variable allows the MC\_Reset to be called multiple times.



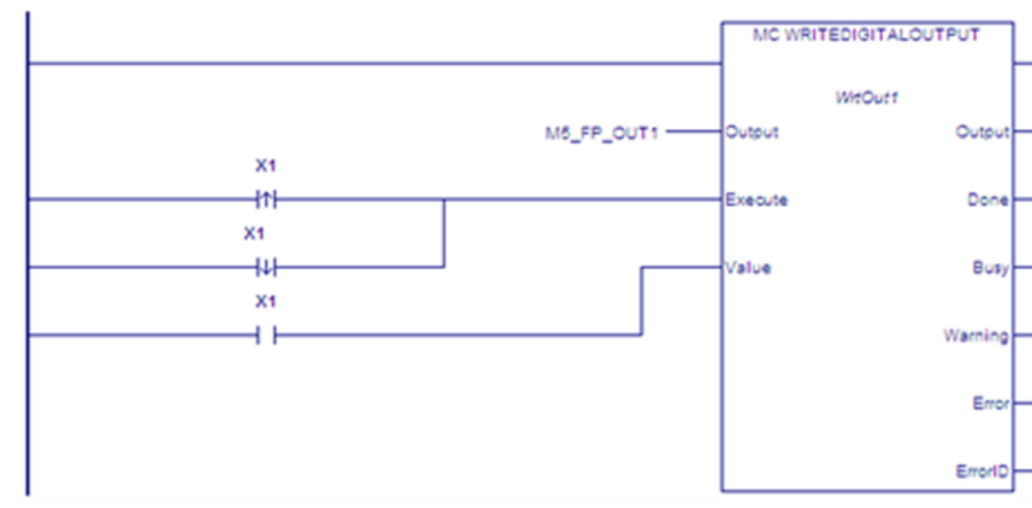
Figure 72: Resetting an Axis - Method Recommended



### Example: Writing to a Digital Output

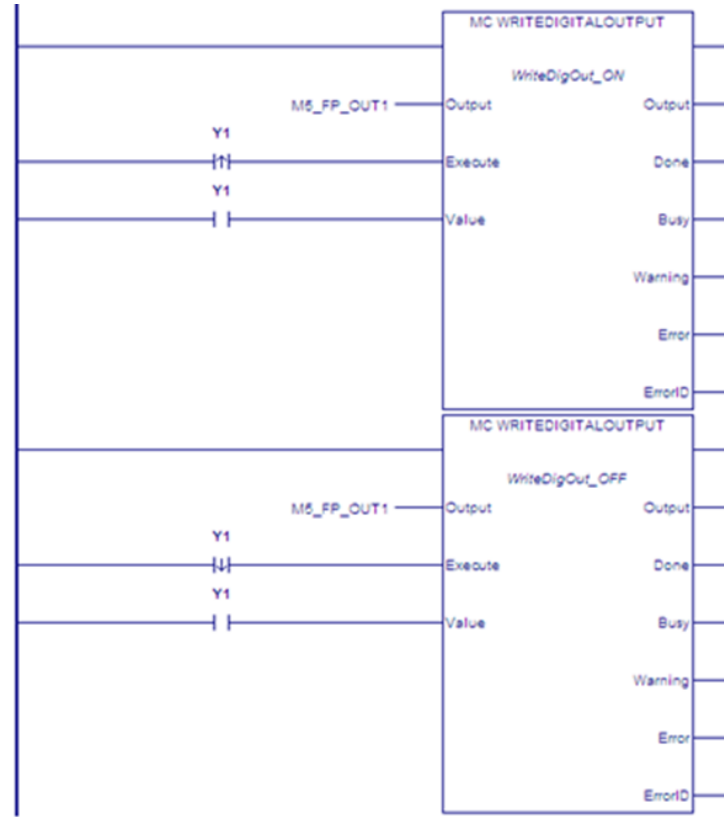
In the following example, you might expect the MC\_WriteDigitalOutput function block to send the value of X1 to the digital output, M5\_FP\_OUT1, when the value of X1 changes. However, the permissive logic, consisting of negative and positive transition contacts in parallel, causes power to be passed to the *Execute* input whenever the value of X1 changes. Since *Execute* never transitions FALSE, the function block cannot be triggered more than once.

Figure 73: One-Time Execution of Function Block



The following sample logic accomplishes the result of writing the value of Y1 to M5\_FP\_OUT1 whenever the value of Y1 changes.

Figure 74: Replicate Y1 to M5\_FP\_OUT1



### Inputs of Executed (Edge-Triggered) Function Blocks

Instance data inputs of executed function blocks are updated only when the *Execute* input transitions high. Instance data inputs cannot be used in application logic.

### Outputs of Executed (Edge-Triggered) Function Blocks

The instance data outputs and corresponding output arguments for Done, InGear, InSync, InVelocity, Error, ErrorID and CommandAborted are reset with the falling edge of Execute. The values of the corresponding output arguments are set on for at least one cycle (that is, one invocation of the function block instance), even if Execute transitions to false before the function block completed its action. The output arguments are updated only when the power flow input, EN, is true.

The falling edge of Execute resets the following outputs:

- Done
- Error
- ErrorID, if associated with an error
- CommandAborted

All other outputs, including InGear, InSync and InVelocity, are maintained at their current state when Execute transitions low. If the Warning output is set, the ErrorID associated with the warning is not cleared.

Instance data output parameters can be used as inputs but not as outputs in application logic.

### Calling an Executed Motion Function Block from an Interrupt Block

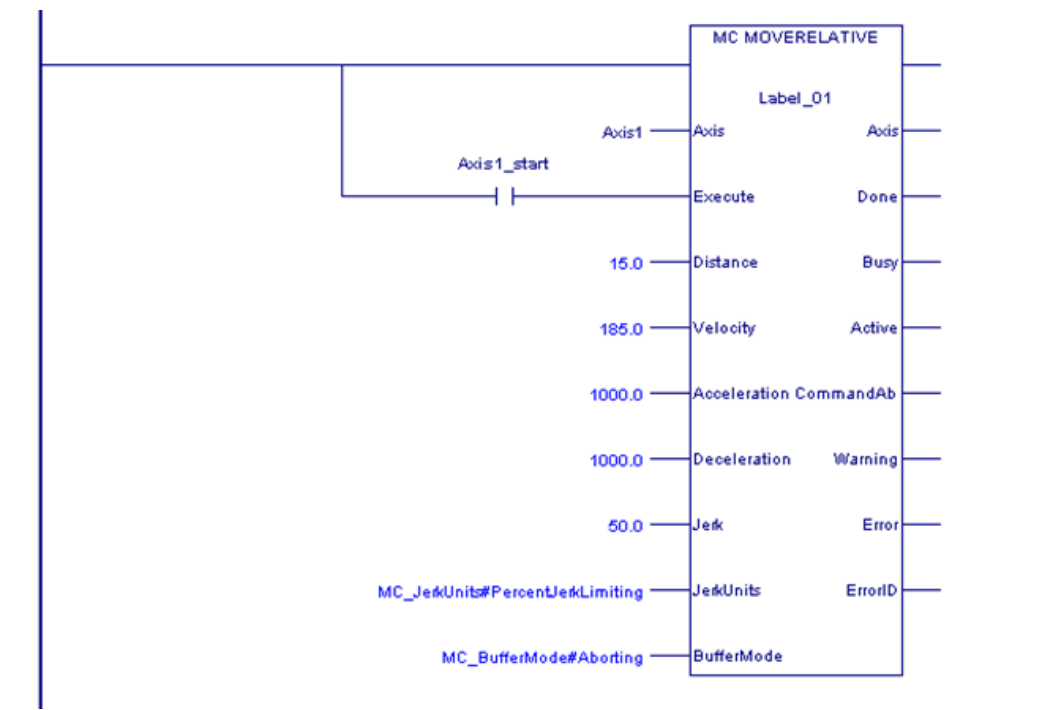
Some applications may utilize an interrupt event to trigger a rapid motion response (such as a photo-detector that triggers a labeling sequence). This type of application could be implemented using an input that initiates an interrupt block containing the associated move commands.

Unexpected operation may result from calling an edge-triggered MFB within an interrupt block unless you guarantee that a positive transition of the *Execute* input occurs with each interrupt.

### Incorrect Implementation of Interrupt MFB Call

In the following sample logic, the MC\_MoveRelative will execute only on the first execution of the interrupt block. After the first execution, *Label\_01.Execute* will remain ON independently of the state of *Axis1\_start*. Subsequent executions of

**Figure 75: Incorrect Implementation of Interrupt MFB Call**

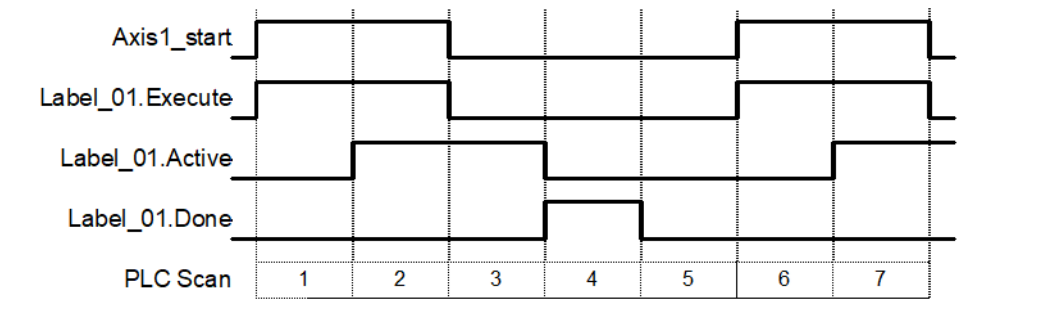


the interrupt block will not execute the function block. No transition of *Label\_01.Execute* occurs because the input was ON when the block was last called.

### Timing Diagram, Conventional Logic Call

The response of the Move command shown above, when called from conventional logic is detailed in the timing diagram below. The state of *Axis1\_start* is reflected in the state of *Label\_01.Execute*. When *Axis1\_start* re-triggered in state 6 a new move command is initiated as expected.

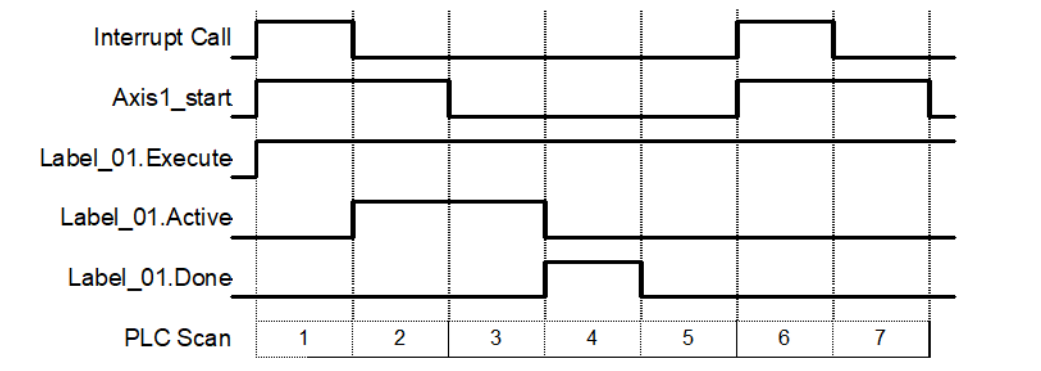
**Figure 76: Timing Diagram Resulting from Incorrect Implementation**



### Timing Diagram, Interrupt Logic Call

The response of the same Move command when called from an interrupt block is detailed in the following timing diagram. In this example, the interrupt block is only processed during steps 1 and 6. A result of the interrupt call is that the instance input data is not updated when *Axis1\_start* transitions OFF. Thus, no positive transition of *Label\_01.Execute* occurs in step 6 (and no new Move is initiated).

**Figure 77: Timing Diagram Resulting from Correct Implementation**

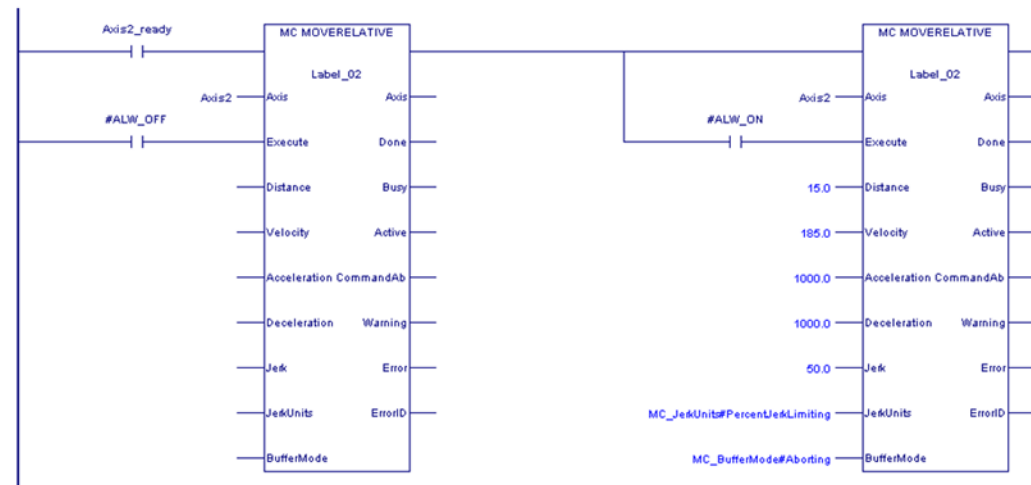


You can ensure that a positive transition of the *Execute* input occurs with each interrupt event by *double calling* the Move instruction: once with the *Execute* input OFF and once with the *Execute* input ON. This method is shown in the recommended call example below (Figure 78).

## Recommended Implementation of Interrupt MFB Call

In this example, the first Move call establishes *Label\_02.Execute* as OFF. The second Move call establishes *Label\_02.Execute* as ON. Thus, a positive transition of the execute input always occurs with each interrupt event.

**Figure 78: Recommended Implementation of Interrupt MFB Call**



It should be noted that edge-triggered PACMotion move commands have an immediate execute, deferred response behavior. When double calling the move in this manner, both occurrences are immediately processed in sequence. However, comparatively little execution time is associated with calling a MFB with the Execute input OFF. For example, MC\_MoveRelative execution time on an IC695CPU310 is 16.32μsec with execute OFF and 168.23μsec with execute ON.

It is recommended that both calls of the move instruction be contained on the same logic rung. Having multiple occurrences of a MFB instance (Label\_02 in the example above) is not generally recommended. Keeping these occurrences on the same logic rung will help avoid unintended conflicts.

## 5.3 Function and Function Block Parameters

### 5.3.1 Permissives, Constants, Variables or Flow used with Motion Function Blocks

A permissive is the state of a Boolean memory type or the result of previous program logic that indicates logic true or false to the Enable or Execute input parameter of the MFB. A permissive can be the state of an input, an LD coil instruction or a group of LD contacts.

Constants may be Boolean or numeric and are input parameter values to MFBs initialized during RX3i power on. These data values do not change unless the RX3i program has changed. Data type is dependent on the particular input parameter and function block used.

Variables may be Boolean or numeric and are used for MFB input and output parameter values. Variables must be created in the program variable list and have associated properties based on the variable type. These data values may change during program execution. Variables may be set to an initial value, default value or hold last value during RX3i power-up or new configuration download. Data type is dependent on the particular parameter and function block used.

Flow is a special construct that is used to pass variable data between program elements without the overhead of creating a variable list entry. In LD programming, flow is indicated by a line connecting the output of one function block to a compatible input of the next function block. Data types must agree between output and input parameters.

For more information on function block operation, refer to the PACSystems RX3i and RSTi-EP CPU Programmer's Reference Manual , GFK-2950.

### 5.3.2 EN Input and ENO Output

These parameters should not be confused with the *Enable* parameter used by some functions and function blocks. The EN input causes the function block to be run by the CPU logic engine. If EN is true, the function block is a part of the CPU's logic scan and as such the block is active. That does not mean that the function block is currently running; it means it is being processed within the CPU logic scan. These parameters should not be confused with the *Enable* parameter used by some functions and function blocks. The *Execute* or *Enable* inputs determine if the MFB is running the command specified by the particular MFB.

- In LD language, the EN input indicates LD power flow from the power rail. If EN is false, ENO will be set false and the function or function block will not be executed. If EN is true, the function or function block will be executed and ENO will be initialized to true, although it can later be set false due to certain problems during execution.

- In FBD language, the EN and ENO parameters are visible to the user and are optional. If EN is not specified, all motion function or function block calls are executed as though EN were true.
- In ST language, the EN parameter is not visible to the user. All motion function or function block calls are executed as though EN were true. The value of ENO is not available in informal-style motion function or function block calls, but is available to the logic in formal-style motion function or function block calls.

### 5.3.3 Input Parameters

In the first invocation of a function block instance, initial values are applied. If the first invocation does not have a valid input, an error will occur. Certain parameters, such as Axis, do not use an initial value. Axis variables are assigned in hardware configuration as reference ID variables of type AXIS\_REF.

If an input parameter of a function block is missing, the value from the previous invocation of the function block instance is used. To modify an input parameter, it is necessary to write the correct value and invoke the function block instance again.

The CPU also initializes input parameters based on the input arguments passed into them. If an input argument is not provided in the logic for a given invocation of a function block instance, the existing value in instance data is used. This value can come from the input argument provided by the previous execution of the function block instance, or if no input argument was provided, from the initial stored value. If no initial value was stored, the function block uses the value that was last stored to the instance data memory location, for example, as a result of a memory clear or an unrelated write to that memory.

#### Common Input Parameters

This section describes the use of inputs common to many instructions. Inputs that are unique to a single or few instructions are described in the description of that instruction.

##### Enable

The *Enable* input determines whether the action specified by the function or function block should be performed. Function blocks that use this input are also called *level-triggered function blocks*. For details on this type of function block, refer to Section 5.2.4, Function Block Triggering (Enabled vs. Executed Instructions).

In LD, this input must be flow. In other languages, *Enable* can be any Boolean variable except those in %S memory.

##### Execute

The *Execute* input triggers the function block to be run at the rising edge (low to high transition) of the input. In LD, this input must be flow. In other languages, all operands are allowed. For details on the operation of the *Execute* input, refer to Executed (Edge-Triggered) Motion Function Blocks in Section 5.2.4.

## Jerk and JerkUnits

Jerk is the rate of change in acceleration/derivative of acceleration with respect to time. Typically, the rate of change is specified in  $UU/sec^3$ . Jerk can be specified as a percentage if required. The JerkUnits parameter, which has data type MC\_JerkUnits, selects which method to use.

---

**Note:** If the Jerk input is set to 0, Jerk will be unlimited and the move is acceleration-limited.

---

## MC\_JERKUNITS Data Type

This data type is used with single-axis MFBs to specify the jerk limiting mode.

It is an enumerated text (ENUM TEXT) type with two possible values:

Value	Definition
UserUnitsperSecondCubed	$UU/sec^3$ . Specifies jerk limiting as a maximum allowed rate of change in acceleration. (Default)
PercentJerkLimiting	Percent jerk limiting (0-100). An LREAL value that specifies the percentage of the acceleration that is jerk limited, given that the move reaches maximum velocity, maximum acceleration and maximum deceleration with acceleration and deceleration starting at 0.

Using percentage units, 0% is constant or linear acceleration with no jerk limiting and 100% represents full S-Curve acceleration.

You are not limited to linear or full S-Curve acceleration. You may specify intermediate jerk values to optimize the peak torque availability versus the shock introduced into the mechanical system during acceleration.

The jerk percentage is calculated with the assumption that the move reaches maximum velocity, maximum acceleration and maximum deceleration. This assures that the move uses the same jerk value in engineering units regardless of the distance the move travels. For example, a short move may not reach maximum velocity due to a position constraint but it will use the same jerk value in engineering units as a longer move with the same maximum velocity, acceleration, deceleration and jerk percent inputs. This assures the application that short moves do not use higher jerk values than long moves. Additionally, the jerk values are calculated independently for acceleration and deceleration since acceleration and deceleration can be different for a given move.

**Example:** If JerkUnits is set to PercentJerkLimiting, a Jerk value of 50% is specified as 50.



## Buffer Mode

This input is used to determine whether a MFB works in non-buffered mode (Aborting) or in one of the buffered modes. The modes determine when the motion action is started.

Every function block that can be buffered has an *Active* output that is set when the function block takes control of the axis. The BufferMode input on a function block is used to control its processing before the instance becomes active (before the *Active* output on the function block goes true). Once the *Active* output on the function block becomes true, changes to the buffer mode are ignored until *Active* goes false again.

A command in the non-buffered, Aborting mode acts immediately, even if this interrupts another motion. A single non-buffered mode function block may be active for each axis.

A buffered MFB is pre-loaded to a PMM for a given axis so that, as the currently executing command completes, the buffered command automatically executes. A command in a buffered mode waits until the current function block sets its *Done*, *InPosition*, or *InVelocity* output.

Only one function block per axis can be buffered at a time. If a buffered function block has been executed but is not active and another buffered function block is executed, the new function block replaces the original function block, which will be aborted. The BufferMode input variable has the enumerated text data type MC\_BufferMode.

A *blended* move (any Blending mode) that changes axis direction is not allowed. A blended move cannot change directions because doing so would require the axis to overshoot the position of the first command. If a blended move tries to go from the active direction to the opposite direction the axis will be *Normal Stopped*.

## MC\_BufferMode Values

Aborting	Default mode (no buffering). The command aborts any ongoing motion and acts immediately.
Buffered	Motion starts as soon as the previous function block sets its Done, InPosition or InVelocity output. There is no blending.
BlendingLow	Motion starts after the previous function block (FB 1) has finished. The velocity transition uses the lowest velocity of both function blocks (FB 1 and FB 2) starting at the end-position of FB 1.
BlendingPrevious	Motion starts after the previous function block (FB 1) has finished. The velocity transition uses velocity of FB 1 at the end-position of FB 1.
BlendingNext	Motion starts after the previous function block (FB 1) has finished. The velocity transition uses velocity of FB 2 starting at the end-position of FB 1.
BlendingHigh	Motion starts after the previous function block (FB 1) has finished. The velocity transition uses the highest velocity of FB 1 and FB 2 starting at the end-position of FB 1.

## Conditions for Activating a Buffered Command

The following table lists conditions used to activate the buffered function block for each active function block type.

Active Function Block	Condition(s) that activate the buffered function block
MC_CamIn	EndOfProfile
MC_GearIn	—6
MC_GearInPos	InSync
MC_Halt	Done
MC_Home	Done
MC_MoveAbsolute	Done
MC_MoveAdditive	Done
MC_MoveRelative	Done
MC_MoveVelocity	InVelocity
MC_Power	Status
MC_Stop	Done AND NOT Execute
MC_JogAxis	—7
MC_SyncStart	—7
MC_DelayedStart	—7
MC_MoveSuperimposed	—7

<sup>6</sup> Buffering not allowed after this command.

## Buffer Mode Handling

The table below lists the commands that support BufferMode and how they are handled with respect to other commands that might be active.

<i>Active Command</i>	Command to be Buffered										
	MC_CamIn	MC_GearIn	MC_GearInPos	MC_Halt	MC_Home	MC_MoveAbsolute	MC_MoveAdditive	MC_MoveRelative	MC_MoveVelocity	MC_Power	MC_Stop
MC_CamIn	1	1	1	1	1	1	1	1	1	1	1
MC_GearIn	—	—	—	—	—	—	—	—	—	—	—
MC_GearInPos	2	2	2	2	2	2	2	2	2	2	2
MC_Halt	2, 5	2	2, 5	2, E	2	2	2	2	2	2	2
MC_Home	2	2	2	2, E	2	2	2	2	2	2	2
MC_MoveAbsolute	2, 5	2	2, 5	4	2	All	All	All	All	2	4
MC_MoveAdditive	2, 5	2	2, 5	4	2	All	All	All	All	2	4
MC_MoveRelative	2, 5	2	2, 5	4	2	All	All	All	All	2	4
MC_MoveVelocity	2	2	2	4	2	All	All	All	All	2	4
MC_Power	2, 3	2, 3	2, 3	2, 3	2, 3	2, 3	2, 3	2, 3	2, 3	2, 3	2, 3
MC_Stop	2	2	2	2, E	2	2	2	2	2	2	2
MC_JogAxis	—	—	—	—	—	—	—	—	—	—	—
MC_SyncStart	N	N	N	N	N	N	N	N	N	N	N
MC_DelayedStart	N	N	N	N	N	N	N	N	N	N	N
MC_MoveSuperimposed	S	S	S	S	S	S	S	S	S	S	S

## Legend for Buffer Mode Handling

1	Beginning with release 1.50, a command can be buffered after a periodic CAM if CAM Exit Distance is non-zero. For detailsCAM, refer to CAM Exit Distancein Section 7.6.3. In earlier releases, the active CAM must be non-periodic. Any blending buffer mode is treated as Buffered and a warning will be issued on the command to be buffered.
2	Any blending buffer mode is treated as Buffered and a warning is issued on the command to be buffered.
3	Once active, nothing buffers to an MC_Power function block. Note that the BufferMode input on MC_Power is used only when the function block instance is being dispatched (before the Active output becomes true). Once the Active output becomes true, any changes to BufferMode are ignored since the function block is active.
4	BlendLow and BlendNext are treated as Buffered and a warning is issued on the command to be buffered. BlendHigh and BlendPrevious are supported with the expected semantics.
5	Axis goes to Standstill state for 1ms.
—	Buffering after this command not allowed. Note that MC_JogAxis does not have a buffer mode. Buffering of any command after MC_JogAxis is not supported.
All	All buffer modes are supported with the expected semantics.
E	The nature of the commands and buffering would result in an error in this combination.
N	MC_SyncStart and MC_DelayedStart are not affected by BufferMode. The motion commands that are started with Sync/DelayedStart will still have their BufferMode applied when they are started.
S	Buffering an MC_MoveSuperimposed is supported only if the active command is an MC_GearIn. If an MC_MoveSuperimposed is active and another MC_MoveSuperimposed is issued, the second superimposed move aborts the first (as if it had specified an Aborting buffer mode) but it does not affect the MC_GearIn active with the MC_MoveSuperimposed.

## Reference ID Variables

Reference ID Variables (RIVs) are associated with HWC parameters, CAM profiles, or files created by the MC\_ReadEventQueue and MC\_DL\_Get function blocks.

RIVs will be *linked* or *unlinked*, depending on how they are created.

*System RIVs* are automatically created by Logic Developer when you create an associated element, such as a PMM module configuration in HWC. These RIVs are linked to their associated elements. You cannot change their values in logic or via online tools in Logic Developer, such as Data Watch or Reference View windows. Because changes to HWC or other system components can change System RIV values, these variables are updated when the Controller transitions from Stop to Run mode.

System RIVs are retentive over a power cycle.

*User-created RIVs* are not linked to their associated elements and can be changed by the application logic. When the Controller transitions from Stop to Run mode, these unlinked RIVs are set to 0 and must be re-initialized by the application logic.

User-created RIVs are retentive over a power cycle.



## RIV Types

System RIVs and User-created RIVs can have the data types listed in the following table.

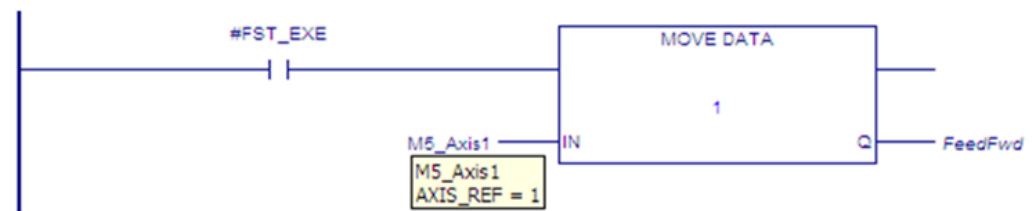
Category	Data Type	Associated With
HWC Parameters	AXIS_REF	Axis
	MODULE_REF	Module
	INPUT_REF	Faceplate or FTB input
	OUTPUT_REF	Faceplate or FTB output
CAM Profile	MC_CAM_REF	Active CAM profile.
File Name Variables	EVENTQUEUE_FILE_REF	Event queue file.
	DATALOG_FILE_REF	Data log file.

### Example: Initializing and Changing a User-Created RIV

In the following sample logic, the System RIV, M5\_Axis1, has been automatically assigned a value of 1. Whenever the Controller transitions from Stop to Run mode, the MOVE\_DATA function initializes the User-created RIV, FeedFwd, to the value of M5\_Axis1.

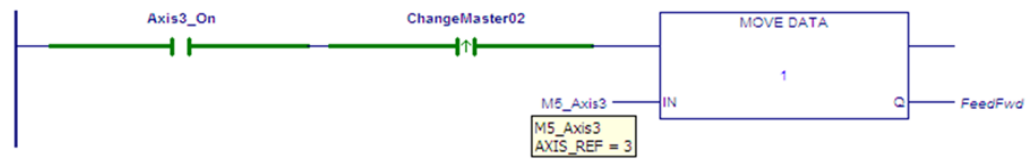
On a power cycle, the value of M5\_Axis1 is retained. The value of FeedFwd is not retained, and therefore must be re-initialized.

**Figure 79:Initializing a User-Created RIV, FeedFwd**



The following rung changes FeedFwd to the value of a different axis, based on permissive logic, which is not shown, that transitions ChangeMaster02 to ON.

**Figure 80:Changing a User-Created RIV, FeedFwd**



## 5.3.4 Output Parameters

When a function block instance is executed, (its Enable or Execute parameter transitions to 1), the CPU initializes output parameters to their default initial values. This includes setting the MFB's Busy output to 1 and the rest of the Boolean outputs to 0.

Execution of a deferred response MFB on the PMM can take many CPU sweeps to complete. As execution proceeds on the PMM, the output values of the MFB can change. When they do, the PMM writes the changed output values back to the MFB's instance data in the RX3i CPU.

As long as the MFB's power flow input, EN, is true, the CPU writes the output values in its instance data to the MFB's output arguments. However, if the MFB instance does not have power flow, the CPU cannot update the output arguments. In this case, the application can read the current output values from the instance data, but the MFB output arguments would not have valid values since the instance did not have power flow and was not being called by the CPU logic program.

---

**Note:** *If your application needs to respond as quickly as possible to changes in the status of executing MFBs, it can directly reference the MFB's output instance data instead of the MFB's output arguments. This is because the PMM updates the instance data as soon as the values change, while the MFB's outputs are only updated when the CPU runs that MFB as part of the logic scan. In cases where the MFBs are chained together in the program in the order they wish to be called there is no advantage to either method.*

---

For details on output behavior specific to enabled function blocks, refer to Outputs of Enabled Function Blocks in Section 5.2.4.

For details on output parameter behavior specific to edge-triggered function blocks, refer to Outputs of Executed (Edge-Triggered) Function Blocks in Section 5.2.4.

### Common Output Parameters

This section defines the usage of outputs common to many function blocks. Outputs that are unique to single or few function blocks are described in the descriptions of those function blocks.

Normally, the Busy, Done, Error, and CommandAborted outputs are mutually exclusive: only one of them can be true at any time on a function block. If Execute or Enable is true, one of these outputs is true. An exception to this behavior occurs when a function block is executed with incorrect input values. In this case, the PMM immediately sends a response to the CPU with the appropriate ErrorID and the Error output set On. The Busy output remains on, but the Error and ErrorID do not show up on the outputs of the MFB until the sweep after the Enable or Execute input transitioned on. The Busy output then transitions off in the next sweep.

Active and Busy can be true at the same time. However, Active is mutually exclusive with Done, Error, and CommandAborted.

## Done

The Done output is set when the commanded action has been executed successfully. This provides proper sequencing of multiple function blocks that work on the same axis in series.

Some instructions use an InGear, InSync or InVelocity output for this purpose. These outputs are reset by the falling edge of the Execute input. If Execute is reset before the function block has set the Done output, the Done output will be set for one sweep.

When a movement on an axis is interrupted by another movement without having reached the final goal, the Done output of the first function block is not set.

In LD, this output must be flow. In other languages, any operand is allowed except constants.

## Busy

Busy indicates that the function block has been executed on an axis and has not yet completed its action. It is used for function blocks that could have a long-time span between the start of execution and completion. The function block becomes Busy immediately after it is executed, and remains Busy until completion and either Done, Error, or CommandAborted is set.

In LD, this output must be flow. In other languages, any operand is allowed except constants.

## Valid

The Valid output is set to 1 if valid output data is available. This output is reset when Enable transitions low.

In LD, this output must be flow. In other languages, any operand is allowed except constants.

## Active

Indicates that the MFB has control of the axis.

Every MFB that can be buffered has an Active output that is set when the MFB takes control of the axis. For unbuffered MFBs and buffered MFBs in Aborting mode, the Active and Busy outputs have the same value. Several MFBs might be Busy, but only one MFB can be Active on an axis at a time. Exceptions are MC\_SuperImposed and MC\_Phasing, where more than one MFB related to an axis can be Active.

In LD, this output must be flow. In other languages, any operand is allowed except constants.



## CommandAborted

CommandAborted is set when another motion command or the MC\_Stop function block interrupts a commanded motion. When CommandAborted is set, the output signals Done, InVelocity, InSync and InGear are reset.

CommandAborted is reset with the falling edge of Execute. If Execute is reset before the function block is aborted, CommandAborted will be set for one sweep.

In LD, this output must be flow. In other languages, any operand is allowed except constants.

## Error, Warning, and ErrorID

All functions and function blocks have three outputs associated with errors and warnings that can occur while executing.

Error indicates that an error occurred during the execution of the function block instance. Error and warning are mutually exclusive: only one of these outputs can be true at a time. If an error occurs, it takes precedence over a warning and results in the InOperation, Busy and Warning outputs being set false.

Warning indicates that a warning occurred during the execution of the function block. The function block can successfully complete with a warning.

In LD, the Error and Warning outputs must be flow. In other languages, they can be any Boolean variable except those in %S memory.

ErrorID identifies the specific error or warning that occurred. This variable can be of any type except constants and variables located in %S memory. The falling edge of the Execute input resets ErrorID, if the ErrorID is associated with an error. ErrorIDs that are associated with a warning are maintained on the falling edge of Execute.

Note that if an error occurs after a warning has been set, the outputs will reflect the error and overwrite the warning.

Error, Warning and ErrorID are reset with the falling edge of Execute or when Enable transitions low.

For a list of ErrorID values and their meanings, refer to Section 9.1.5, Error ID Reference.

## 5.3.5 In\_Out Parameters

In\_Out parameters, for example Axis, Master and Slave, pass the value from the left side (input) of the function block to the right side (output) of the function block. The value is passed as long as Enable or Execute is set to 1 and the input arguments are valid. The outputs of IN\_OUT parameters can only be used as a flow connection to another function or function block. The output flow connection is optional.

In\_Out parameters can be symbolic variables only.

## 5.4 Data Types and Structures

This section describes the data types common to many function blocks. Data types that are unique to a single or few function blocks are included in the description of that function block. These data types can be used only with MFBs and are supported in LD and ST.

### 5.4.1 HWC Parameter Linked Data Types

#### AXIS\_REF

Each axis in use must have an assigned name, which is a variable of data type AXIS\_REF. These symbolic variables are part of the hardware configuration (HWC) and can be added or deleted only during a Stop Mode Store.

An AXIS\_REF variable is a structure with one element:

Element	Type	Description
REFERENCE_ID	UINT	

#### MODULE\_REF

Each PMM configured in a project has an assigned name, which is a variable of data type MODULE\_REF. These symbolic variables are part of the HWC and can be added or deleted only during a Stop Mode Store.

A MODULE\_REF variable is a structure with one element:

Element	Type	Description
REFERENCE_ID	UINT	

## INPUT\_REF

An input reference variable that identifies a specific input on a particular module in a configuration. This data type is used with functions that read the value of an input signal source.

### An INPUT\_REF variable has two elements:

Element	Type	Description
Module	MODULE_REF	Identifies the PMM on which the input value will be read.
Input	UINT	Parameter number that identifies the input. For details refer to Axis Parameter Number Index in Section 8.1.1.

## OUTPUT\_REF

An output reference variable that identifies a specific output on a particular module in a configuration. This data type is used with functions that read or write the value of an output.

An OUTPUT\_REF variable has two elements:

Element	Type	Description
Module	MODULE_REF	Identifies the PMM on which the output value will be written.
Output	UINT	Parameter number that identifies an output. For details refer to Axis Parameter Number Index in Section 8.1.1.

## 5.4.2 Enumerated Data Types

Enumerated data types are:

DL_OperatingMode DL_PostSample DL_SamplingRate DL_TriggerMode	Used by MC_DL_Configure function block. Refer to Section 6.11.
MC_BufferMode	Used by many function blocks. Refer to Buffer Mode in Section 5.3.3.
MC_Direction	Used by MC_MoveVelocity and MC_MoveAbsolute function blocks. Refer to MC_DIRECTION Data Type in Section 6.22.
MC_Encoder	Used by MC_SetPosition function block.
MC_HomingMode	Used by MC_Home function block. Refer to Section 6.18.3, Homing Modes.
MC_JerkUnits	Used by function blocks that command motion. Refer to Jerk and JerkUnits in Section 5.3.3.
MC_PositionSource	Used by MC_CamIn, MC_DigitalCamSwitch, MC_GearIn and MC_GearInPos function blocks. Refer to corresponding discussions in Section 6.4, Section 6.14.1, and Section 6.15.
MC_SyncMode	Used by MC_GearInPos function block. Refer to Section 6.15 MC_GearInPos.

## 5.4.3 CAM Profile Linked Data Types

### MC\_CAM\_REF

This data type is used with the administrative function blocks used to access CAM profile data. The CamTable input parameter is a three-element structure variable that identifies the CAM profile to be accessed.

An MC\_CAM\_REF variable has three elements:

Element	Type	Description
FileName	STRING	The value for this will be set to the name of the CAM profile to which the MC_CAM_REF variable is associated.
FileNameCksm	DWORD	
ProfileID	UINT	

### MC\_CAM\_ID

An MC\_CAM\_ID variable has one element:

Element	Type	Description
CHECKSUM	DWORD	
ID	DWORD	Identifies CAM table to be used in the MC_CamIn and MC_CamSelect function blocks.

## 5.5 Axis States

When an axis is enabled (the servo drive has power) it is always in one of the states defined by the state machine shown in Figure 81. Any motion command is a transition that changes the state of the axis and, as a consequence, modifies the way the current motion is computed.

The Axis State diagram shows the behavior of a single axis when multiple motion control function blocks are simultaneously activated. Motion commands are always executed sequentially.

The multiple-axis function blocks, MC\_CamIn, MC\_GearIn and MC\_Phasing, can be looked at as affecting multiple single-axes, all in specific states. For instance, the CAM-master can be in the Continuous Motion state. The corresponding slave is in the Synchronized Motion state. Connecting a slave axis to a master axis does not affect the master axis state.

The following administrative function blocks do not affect the axis state: MC\_ReadStatus, MC\_ReadAxisError, MC\_ReadParameter, MC\_ReadBoolParameter, MC\_WriteParameter, MC\_WriteBoolParameter, MC\_ReadActualPosition, MC\_CamTableSelect, MC\_CamTableDeselect, MC\_CamFileRead, MC\_CamFileWrite and MC\_LibraryStatus.

### Standstill

In this state, the axis is enabled and is not moving. The axis is ready to perform a motion and there are no errors. After the system is started, MC\_Power is used to enter the Standstill state. The axis also enters this state every time a move successfully comes to a stop and when an error is reset.

### Errorstop

An error can occur in any state. When an error occurs, the axis goes to the ErrorStop state and, if it was in motion, will stop. The only way to exit the ErrorStop state is to clear the source of the error and use MC\_Reset to return to the Standstill state. In some situations, MC\_JogAxis can be used while in ErrorStop to clear the error by moving the axis back within the end of travel limits.

### Homing

This state indicates that the axis is performing a home cycle. The only way to enter this state is using the MC\_Home function block while at Standstill. Only the MC\_Stop command can interrupt the Homing state.

## Discrete Motion

Discrete MFBs: MC\_MoveAbsolute, MC\_MoveRelative, MC\_MoveSuperimposed and MC\_MoveAdditive.

In this state, a motion that will end at a defined point is being performed. When this motion is complete, the axis returns to the Standstill state.

## Continuous Motion

Continuous MFBs: MC\_MoveVelocity and MC\_JogAxis. In this state, a move that does not have a definitive end point is being performed. This motion continues until MC\_Stop is used to bring the axis to a halt, it is interrupted by another move, or an error occurs.

## Synchronized Motion

Synchronized MFBs: MC\_GearIn, MC\_GearInPos, MC\_CamIn and MC\_Phasing. In this state, the axis is a slave axis that is being controlled by a master axis. The axis remains in this state until a command for a non-synchronized move or MC\_Stop is issued, or an error occurs.

## Stopping

MC\_Stop is used to stop a Discrete Motion before it has completed or to stop Continuous and Synchronized Motion. MC\_Stop causes the current motion to abort and the state to change to the Stopping state. While the axis is in this state, other MFBs can be called but they will not be executed. When the axis has stopped, MC\_Stop will set Done true. As long as Execute is true, the axis remains in the Stopping state. Only after Done is true and Execute is false does the state change to Standstill.

## Setup

In this state, the axis is in a diagnostic or tuning condition. While in this state, MFBs that do not control axis setup are not executed.

## Jogging

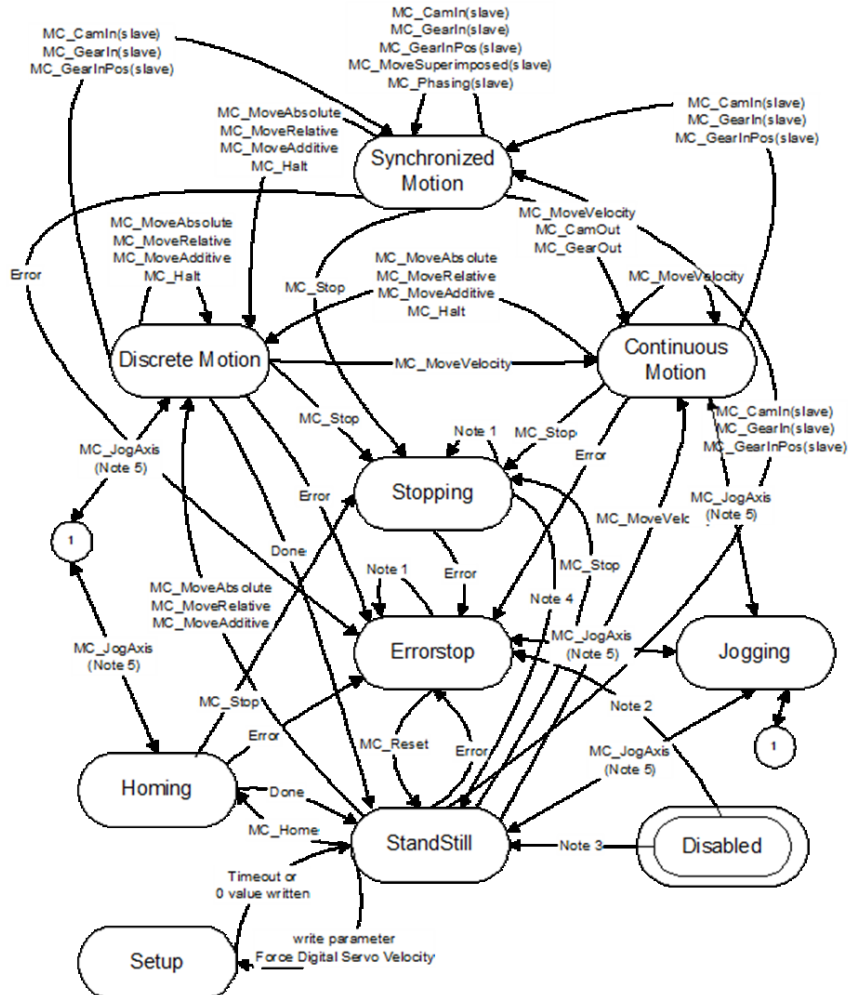
In this state, the axis is under the control of an MC\_JogAxis function block. While in this state, motion-generating function blocks other than MC\_JogAxis are not executed. After both Enable inputs on the MC\_JogAxis are disabled, the axis will return to the state that was active when the MC\_JogAxis was enabled. Even if the jog clears an error (by moving the axis off a hardware end-of-travel), the axis will return to ErrorStop if it was in ErrorStop before the jog. An MC\_Reset or MC\_ModuleReset must subsequently be performed to transition the axis out of ErrorStop.

## Disabled

The servo drive does not have power.

## 5.5.1 Axis State Diagram

Figure 81: Axis State Diagram



### Notes for Axis State Diagram

**Note:** In the ErrorStop or Stopping states all Function Blocks can be called, although they will not be executed, except MC\_Reset and Error. MC\_Reset will generate the transition from ErrorStop to Standstill and an Error will immediately transition from Stopping to ErrorStop.

MC\_Power.Enable is TRUE and there is an error in the Axis

MC\_Power.Enable is TRUE and there is no error in the Axis

MC\_Stop.Done is TRUE and MC\_Stop.Execute is FALSE

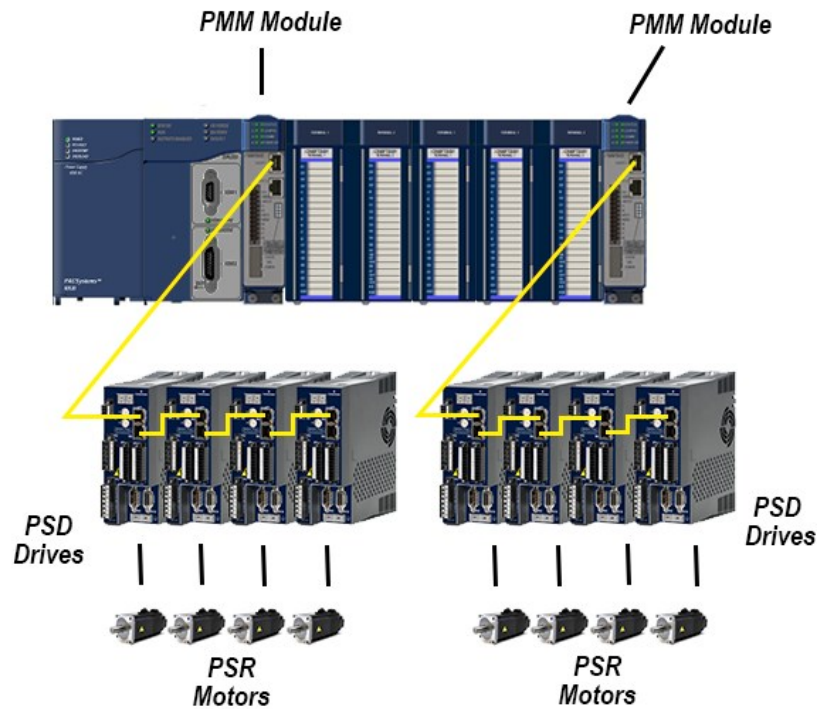
Unless the axis is in Standstill or ErrorStop state, MC\_JogAxis may be issued only if the axis is stopped due to a feed hold. (The commanded velocity is zero as a result of an MC\_SetOverride specifying a zero percent VelFactor input).

## 5.6 Synchronized Motion

High-speed applications require precise coordination between axes. PACMotion provides an extensive set of features to support this coordination. All of the planning and control functions are synchronized across modules, yielding very tight control for starting and stopping functions. Additionally, the modules share axis data across the backplane such that any module can utilize critical data. These abilities allow a PACMotion system with multiple PMMs to function as if it was a single motion controller with a large number of axes.

Synchronized motion function blocks, such as *MC\_CamIn*, *MC\_GearIn*, *MC\_Phasing*, and *MC\_SyncStart* seamlessly access the shared data using the symbolic variables names of type *AXIS\_REF*. These names are assigned to each axis via hardware configuration.

**Figure 82: System Using Synchronized Motion**



The Synchronized Motion Function blocks *MC\_CamIn* and *MC\_GearInPos* can be Pending, Ramping, or InSync. The *MC\_GearIn* function block status is never Pending but can be Ramping or In Gear. For a discussion of these states, refer to Section 5.6, Synchronized Motion.

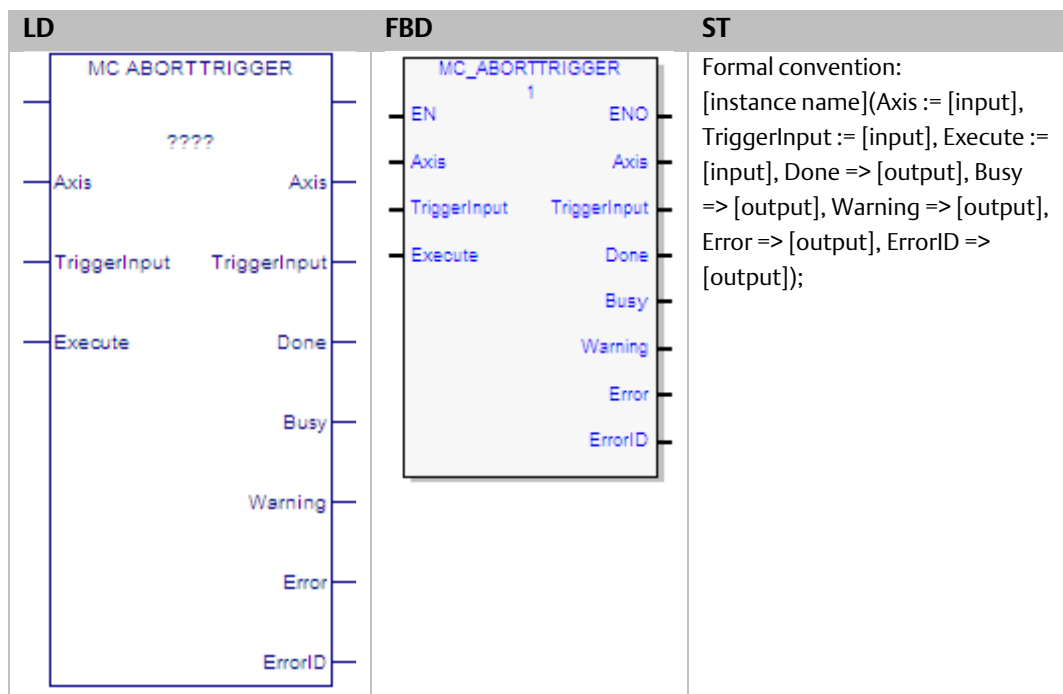


# Section 6 PACMotion Instruction Set Reference

This chapter contains details of the functions and function blocks that make up the PACMotion instruction set. Instructions are presented in alphabetical order, with an overview of each instruction’s operation and definitions of its operands. Refer to Table of Contents for a complete listing.

For information on constructs and operational requirements that are common to the PACMotion instruction set, refer to Section 5, PACMotion Function Block Operation

## 6.1 MC\_AbortTrigger



This function block is used to terminate an MC\_TouchProbe function block operating on the specified Axis. The MC\_TouchProbe function block is described in Section 6.49, MC\_TouchProbe.

Execution type: Immediate execution/deferred response.

## Operands

Parameter	Description	Allowed Data Types	Initial Value
????	Instance variable name.	MC_ABORTTRIGGER	N/A
<b>Input_Output Parameters</b>			
Axis	The axis on which the function block is to be executed.	AXIS_REF	N/A
<b>Inputs</b>			
TriggerInput	Reference to the strobe trigger signal source. A value of 1 indicates Touch Probe 1; a value of 2 indicates Touch Probe 2.	INPUT_REF	N/A
Execute	Terminates the MC_TouchProbe function block operating on the specified Axis.	LD: flow Other languages: all except constants	0
<b>Outputs</b>			
Done	Trigger functionality has been aborted.	LD: flow	0
Busy	Indicates the function block has been executed on an axis and has not yet completed its action.	Other languages: all except constants	1
Warning	Indicates that a warning has occurred within the function block.		0
Error	Indicates that an error has occurred within the function block.		0
ErrorID	Error or warning identification.	WORD	0

## 6.2 MC\_CamFileRead

LD	FBD	ST
		<p>Formal convention:  [instance name](CamTable := [input], Execute := [input], Description := [input], Length := [input], Done =&gt; [output], Busy =&gt; [output], Warning =&gt; [output], Error =&gt; [output], ErrorID =&gt; [output], Data =&gt; [output], DataSize =&gt; [output]);</p>

The MC\_CamFileRead and MC\_CamFileWrite function blocks are used to read CAM profiles into reference memory and write them out of reference memory, allowing online, programmable updates of CAM profiles. The MC\_CamFileRead function block copies the contents of a CAM file from the RX3i file system into reference memory. This function block may take several CPU sweeps to complete, based on the amount of data to be exported from the CAM file. Its impact on any single CPU sweep is limited to 2 ms or less.

Only one instance of MC\_CamFileRead can be executed at a time. If another operation is writing to the CAM file when an MC\_CamFileRead is invoked, the function block will fail. Also, if the same instance of the MC\_CamFileRead or MC\_CamFileWrite is retriggered before the previous read/write operation has completed, the operation will fail.

On the falling edge of Execute, the Done, Error, and ErrorID outputs are set to off. If the function block instance hasn't yet completed, when it does, or if it aborts due to an error, the Done or the Error/ErrorID outputs will be set on for at least one execution of the instance. ErrorID is the only non Boolean output that is cleared. The non Boolean outputs, CamTable, Data and DataSize are not cleared.

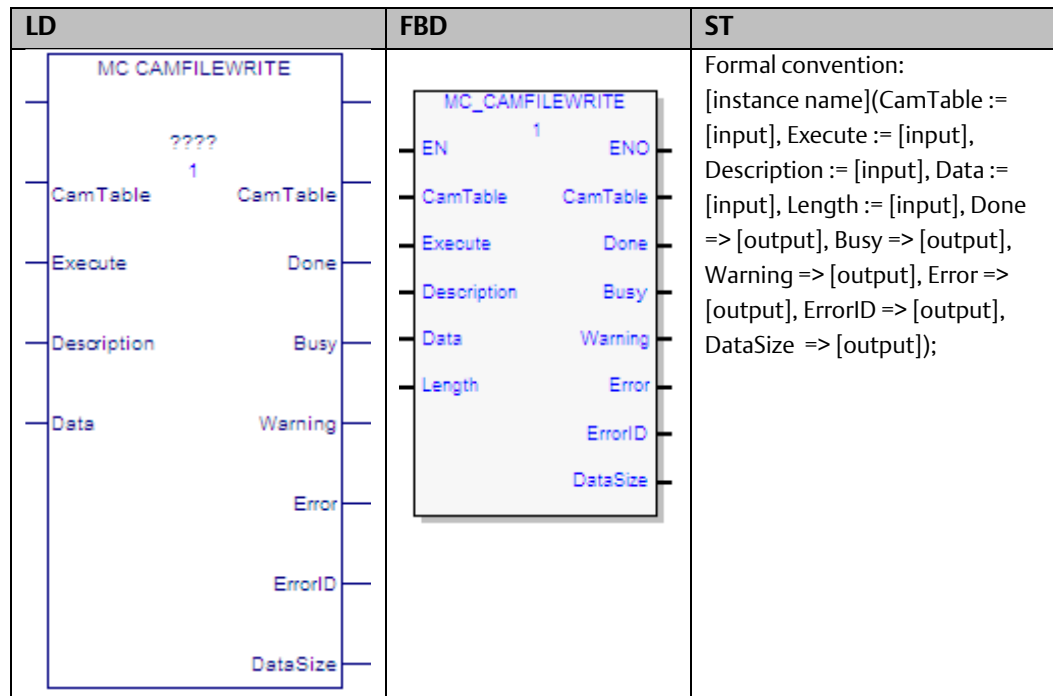
If the host controller transitions to stop mode while an MC\_CamFileRead or MC\_CamFileWrite is in progress, the operation will be completed and the instance data will be updated. On the next transition to run mode, the instance data values will be updated to the respective output parameters.

Execution type: Immediate execution/deferred response

## Operands

Instance Variable	Description	Allowed Data Types	Initial Value
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as ??? in LD.)	MC_CAMFILEREAD	NA
Parameter	Description	Allowed Data Types	Initial Value
??	Length. The number of DWORDs to be read. If Data variable is symbolic, Range is 1 to 9999; if mapped, Range is 1 to 32767.	INT	1
<b>Input_Output Parameters</b>			
CamTable	The file to be read.	MC_CAM_REF	N/A
Inputs			
Execute	When Execute transitions from OFF to ON, new input values are read and the transfer begins.	LD: flow Other languages: all except constants	0
Description	Specifies the number of CAM file elements to be copied and identifies each element to be copied. Only one element, consisting of the entire file, can be copied. Therefore, the first two words of data must be 00000001, 00000001. The remaining words must be 00000000.	WORD array, consisting of 16 words	0
<b>Outputs</b>			
Done	The CamFileRead has completed successfully.	LD: flow Other languages: all except constants	0
Busy	CamFileRead operation has successfully started, but has not completed.		0
Warning	Indicates that a warning has occurred within the function block.		0
Error	Indicates that an error has occurred within the function block.		0
ErrorID	Indicates the type of error or warning.	WORD	0
Data	The variable to receive the number, specified by Length, of DWORDs read from the file. Symbolic arrays are limited to 9999 elements. If a larger amount of data must be read, a mapped variable must be used, which will allow an import size as large as the Data variable's memory segment will support.	DWORD array.	0
DataSize	Indicates the actual number of bytes read from the CAM profile file into the Data argument.	DINT	0

## 6.3 MC\_CamFileWrite



The MC\_CamFileRead and MC\_CamFileWrite function blocks are used to read CAM profiles into reference memory and write them out of reference memory allowing online, programmable updates of CAM profiles.

The MC\_CamFileWrite function block copies CAM data from reference memory to an existing CAM file in the RX3i file system, overwriting the original data in the CAM file. This function block may take several sweeps to complete, based on the amount of data to be written into the CAM file. Its impact on any single CPU sweep is limited to 2 ms or less.

Only one instance of MC\_CamFileWrite or MC\_CamFileRead can be executed at a time. If some other operation is reading or writing the selected CAM file when an MC\_CamFileWrite is invoked, the function block will fail. Also, if the same instance of the MC\_CamFileRead or MC\_CamFileWrite is retriggered before the previous read/write operation has completed, the operation will fail.

On the falling edge of Execute, the Done, Error, and ErrorID outputs are set off. If the function block instance hasn't yet completed, when it does, or if it aborts due to an error, the Done or the Error/ErrorID outputs will be set on for at least one execution of the instance. ErrorID is the only non Boolean output that is cleared. The non Boolean outputs, CamTable, and DataSize are not cleared.

If the host controller transitions to stop mode while an MC\_CamFileWrite is in progress, the operation will be completed and the instance data will be updated. On the next transition to run mode, the instance data values will be updated to the respective output parameters.

Execution type: Immediate execution/deferred response

## Operands

Instance Variable	Description	Allowed Data Types	Initial Value
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as ??? in LD.)	MC_CAMFILEWRITE	NA
Parameter	Description	Allowed Data Types	Initial Value
??	Length. Sets the required length in DWORDs of the Data parameter. If Data variable is symbolic, Range is 1 to 9999; if mapped, range is 1 to 32767.	INT	1
<b>Input_Output Parameters</b>			
CamTable	The name of the file to be written.	MC_CAM_REF	N/A
<b>Inputs</b>			
Execute	When Execute transitions from OFF to ON, new input values are read and the transfer begins	LD: flow Other languages: all except constants.	0
Description	Specifies the number of CAM file elements and the identification of each element to be written. Only one element, consisting of the entire file, can be written. Therefore, the first two words (Words 00 and 01) must be 00000001, 00000001. The remaining words must be 00000000.	WORD array, consisting of 16 words	0
Data	The variable that contains the data to be written into a file. Symbolic arrays are limited to 9999 elements. If a larger amount of data must be written, a mapped variable must be used, which will allow an export size as large as the Data variable's memory segment will support.	DWORD array	0
<b>Outputs</b>			
Done	The CamFileWrite has completed successfully	LD: flow Other languages: all except constants	0
Busy	CamFileWrite operation has successfully started, but has not completed		0
Warning	Indicates that a warning has occurred within the function block.		0
Error	Indicates that an error has occurred within the function block.		0
ErrorID	Indicates the type of error or warning		0
DataSize	Indicates the actual number of bytes written into the CAM profile file from the Data argument.	DINT	0

## 6.4 MC\_CamIn

LD	FBD	ST
		<p>Formal convention:  [instance name](Master := [input], Slave := [input], Execute := [input], MasterOffset := [input], SlaveOffset := [input], MasterScaling := 1.0, SlaveScaling := 1.0, StartMode := [input], CamTableID := [input], BufferMode := [input], RampDistance := [input], PositionSource := [input], InSync =&gt; [output], Busy =&gt; [output], Active =&gt; [output], CommandAborted =&gt; [output], Warning =&gt; [output], Error =&gt; [output], ErrorID =&gt; [output], EndOfProfile =&gt; [output], CycleCount =&gt; [output]);</p>

The MC\_CamIn function block is used to engage a CAM profile that has been loaded using the MC\_CamTableSelect function. The CamTableID must be valid on the PMM that the slave AXIS\_REF resides on. To execute this function block, both axes must have their *PositionValid* status flags set.

When the slave axis is synchronized on the CAM profile, the *InSync* output is ON. *InSync* will stay on as long as the slave remains synchronized, even if the *Execute* input goes low.

The Virtual Axis (Axis 5) can be used as a Master input but not as a Slave input to this function block.

**Note:** If Axis 5 is the master, *PositionSource* must be *Commanded Position*, unless an external encoder is provided. For sample configuration and logic using Axis 5 with an External Encoder, refer to Appendix B-2 Example: Connecting an External Encoder to Axis 5.

Changing the PositionSource (Commanded to Actual or Actual to Commanded) between calls to an MC\_CamIn function block causes the slave to follow the new source, which introduces a change equal to the master's Position Error in the observed master position. A ramp (non-zero RampDistance) may be necessary to succeed. If the second MC\_CamIn is Buffered, an insufficient RampDistance (smaller than the master's Position Error) can cause the observed master position to be off the CAM profile, resulting in the axis generating an error (0x533E) and going to the ErrorStop state.

Changes to the Low Limit, Range or End of Travel configured for the master axis (via an MC\_WriteParameter) are not allowed while an MC\_CamIn is engaged or pending on that axis.

---

**Note:** *If the master experiences an error and goes to the ErrorStop state, the slave will continue to follow the master to zero velocity. If the master becomes inaccessible, the slave will go to the ErrorStop state and stop the slave axis motion.*

---

Execution type: Immediate execution/deferred response

## Operands

Instance Variable	Description	Allowed Data Types	Initial Value
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as ??? in LD.)	MC_CAMIN	NA
Parameter	Description	Allowed Data Types	Initial Value
<b>Input_Output Parameters</b>			
Master	Reference to master axis.	AXIS_REF	N/A
Slave	Reference to slave axis. Virtual Axis (Axis 5) not supported.	AXIS_REF	N/A
<b>Inputs</b>			
Execute	Start at rising edge.	LD: flow Other languages: all except constants	0
MasterOffset	Offset of master table. Refer 6.4.1, Offset and Scaling.	LREAL	0
SlaveOffset	Offset of slave table.	LREAL	0
MasterScaling	Factor for the master profile.	LREAL	1.0
SlaveScaling	Factor for the master profile.	LREAL	1.0
StartMode	Specifies the Ramp, Slave and Master modes. Refer to Section Start Mode Mask.	DWORD.	0
CamTableID	Identifier of the CAM table to be used. Generated by MC_CamTableSelect.	MC_CAM_ID	
BufferMode	Defines the behavior of the axis: modes are Aborting and Buffered. (Blending not allowed.)	MC_BufferMode	0



Instance Variable	Description	Allowed Data Types	Initial Value
RampDistance	The distance in Uu the Master axis travels, within which the slave and master will be synchronized on the profile. This is a fixed distance the master will always cover during the ramp. If the value is 0, it is assumed the master and slave are already in sync and the axes will synchronize to the profile within the axis velocity, acceleration and deceleration limits specified in HWC.	LREAL	0
PositionSource	Defines the source on the master to follow. Actual Position: Source is the configured feedback device. Commanded Position: Source is the instantaneous position generated by the PMM's internal path generator. If Axis 5 is the master, PositionSource must be Commanded Position, unless an external encoder is provided.	MC_PositionSource	0
<b>Outputs</b>			
InSync	CAM is engaged for the first time. InSync will stay on as long as the slave remains synchronized, even if the Execute input goes low.	BOOL	0
Busy	Indicates the function block has been executed and has not yet completed its action.	LD: flow Other languages: all except constants	1
Active	Indicates that the function block has control of the axis.		0
CommandAborted	Command is aborted by another command.		0
Warning	Signals that a warning has occurred within the function block.		0
Error	Signals that an error has occurred within the function block.		0
ErrorID	Indicates the type of error or warning.	WORD	0
EndOfProfile	Signals the cyclic end of the CAM profile. When activated, this output is held high for one invocation of the function block then taken low on the next invocation. Note that this transition can happen multiple times while the MC_CamIn is active.	BOOL	0
CycleCount	The number of times that the CAM crosses the master rollover position. This value is retained as long as the CAM is engaged, even if Execute transitions low.	INT	0

## 6.4.1 Offset and Scaling

The Offset and Scaling parameters are applied to the CAM Function. The slave command position is generated according to the following formula.

$$f = \mathbf{CamFunction}, x = \mathbf{MasterPsn}, y = \mathbf{SlavePsn}$$

$$y = \mathbf{SlaveScaling} * f\left(\frac{x - \mathbf{MasterOffset}}{\mathbf{MasterScaling}}\right) + \mathbf{SlaveOffset}$$

Using the Scaling and Offset to calculate the Lower and the Upper limits of the profile:

$$y = f(x), \quad x_{\min} \leq x \leq x_{\max}, \quad y_{\min} \leq y \leq y_{\max}$$

O = Offset; S=Scaling

$$S_x x_{\min} + O_x \leq x \leq S_x x_{\max} + O_x, \quad S_x > 0$$

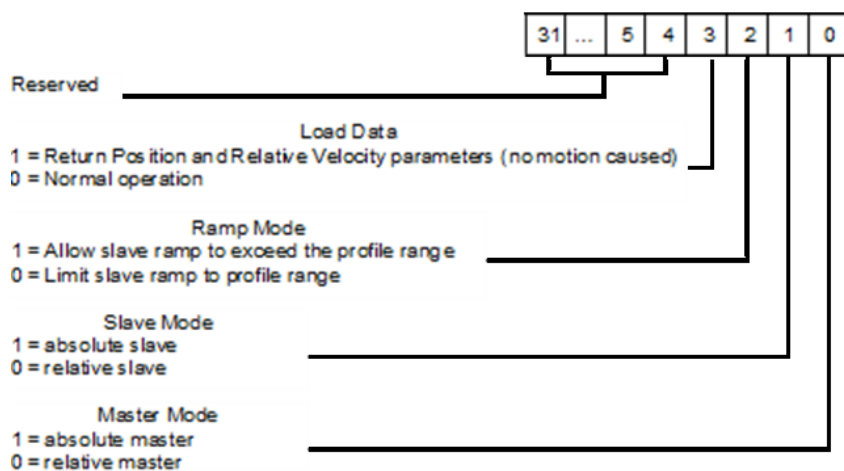
$$S_x x_{\max} + O_x \leq x \leq S_x x_{\min} + O_x, \quad S_x < 0$$

$$S_y y_{\min} + O_y \leq y \leq S_y y_{\max} + O_y, \quad S_y > 0$$

$$S_y y_{\max} + O_y \leq y \leq S_y y_{\min} + O_y, \quad S_y < 0$$

## 6.4.2 Start Mode Mask

Figure 83: Bit Definition Start Mode Mask



## Load Data

The slave position of a CAM profile can be determined from the master position without requiring motion of the slave. The *Load Data* bit of the StartMode input returns the position and relative velocity data in the slave axis parameters, #1331 and #1332. When this bit is set, the function block may only be called with the Slave in the *Standstill* state. Parameter #1331 is of type LREAL and will hold the returned slave position. Parameter #1332 is of type LREAL and will hold the returned slave relative velocity. Refer to CAM Load Data in Section 7.6.3 for additional information regarding usage.

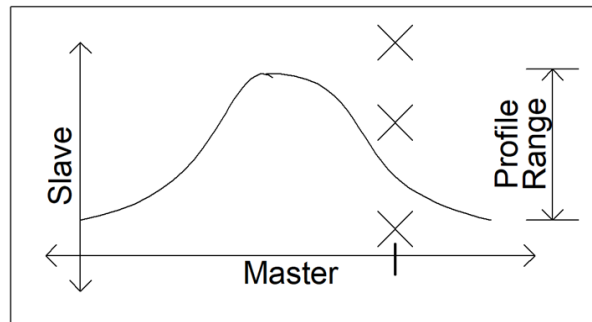
## Ramp Mode

During ramping, the Slave attempts to minimize the acceleration it must undergo. If backup is allowed (RampMode is set to 1) a consequence of this is that the ramp may move the slave backward, away from the synchronization position or past the synchronization position so that it can be moving in the opposite direction when it does synchronize.

In some situations, for example when the master reverses direction while ramping with Backup Allowed, the slave may exhibit unexpected behavior. To prevent this behavior RampMode should be set to 0 (No Backup Allowed).

For some applications, it is desirable that the Slave never exceed the boundaries of the profile. This is the default behavior when Ramp Mode is set to 0.

**Figure 84: Restricting Slave to Stay within Profile Range**



## CAM Axis Mode

In the PMM the master axis or the slave axis may be set in either absolute or relative mode when the MC\_CamTableSelect function is called.

**Absolute mode** means that the position of the master axis or slave axis is treated as absolute position. Both axes must have configured High/Low position limits and be referenced to a value within the position limits. Absolute position values cannot go past the limits. For absolute master, when the MC\_CamIn function is called, the absolute value of the master axis at the time of execution will determine the slave axis location. For an absolute slave axis, the master position commands the slave axis to an absolute position.

**Relative mode** indicates that position values will increment from their relative location at the time the MC\_CamIn is executed. The relative master axis starts at the beginning of the CAM profile and advances through the CAM profile incrementally. The slave axis relative mode means that the slave axis moves incrementally without reference to the slave axis absolute position. The relative mode can exceed the Hi/Low position limits and will roll over to the next axis cycle modulus.

### Absolute Master

If the Slave or Master is Executed as *Absolute*, then the positions in the CAM profile will be used after the Scaling and Offsets are applied.

For example, if the original profile goes through the following points:

Master	Slave
0	0
500	500
1000	1000

Then, using the example scaling and offset parameters, the scaled and offset profile will go through the following modified points:

MasterScaling: 2; SlaveScaling: 0.5; MasterOffset: -100; SlaveOffset: 50

Master	Slave
-100	50
900	300
1900	550

If the master is at 900 when the MC\_CamIn is executed, the slave will be commanded to 300.

## Relative Master

If the Master is Executed as *Relative*, the lower-limit of the profile, *after* the MasterScaling is applied but *before* the MasterOffset is applied, is assumed to be the current master position. The MasterOffset is then applied.

If the profile has been selected as non-periodic (see MC\_CamTableSelect), then this means that any positive MasterOffset value will "push" the profile away from the current master position. This will result in an error unless the RampDistance is sized to bridge the gap by being greater-than or equal to the MasterOffset. If the profile is periodic, the profile exists "everywhere" on the master so it is not possible to be off of the profile.

If the profile has been selected as non-periodic and the master is moving in the negative direction, the MasterOffset can be set to a negative value to "push" the profile to the left.

For example (assume the Slave is *Absolute*) and using the Table of positions and Scaling and Offset parameters above, if the Master is at 900 when the MC\_CamIn is Executed, the master profile lower-limit is set equal to the current master position, so the modified points look like this:

Master	Slave
900	50
1900	300
2900	550

Next, the MasterOffset is applied and the points look like this:

Master	Slave
800	50
1800	300
2800	550

Assuming a Linear curve fit type (calculation below), since the master is at 900, the slave is commanded to be at 75.

$$y = \left( \frac{300 - 50}{1800 - 800} \right) * (900 - 800) + 50 = 75$$

---

**Note:** *The one exception to this rule is when a Relative Master, buffered non-periodic MC\_CamIn is executed after another MC\_CamIn. In this case, the profile edge that is left is stored and used to set the opposite edge of the buffered profile (e.g. if the master crosses the left edge of the first profile, that edge value will be used to set the right edge value of the next profile). This allows for a series of Buffered Relative MC\_CamIn function blocks that can be executed in loop without the profiles "drifting".*

---

## Relative Slave

If the Slave is Executed as Relative, the current slave position is assumed to be on the profile after SlaveScaling is applied but before the SlaveOffset is applied.

For example, if the original profile goes through the following points:

Master	Slave
0	0
500	500
1000	1000

Then, using the example scaling and offset parameters, the scaled profile will go through the following modified points:

MasterScaling: 2; SlaveScaling: 0.5

Master	Slave
0	0
1000	250
2000	500

If the Master is at 1000 and the Slave is at 100 when the MC\_CamIn is Executed, then the Slave is currently 150 less than in the profile: the profile is modified so that the points look like this:

Master	Slave
0	-150
1000	100
2000	350

With SlaveOffset equal to zero, the Slave will always be on the profile. If we add a SlaveOffset, the Slave is never on the profile exactly.

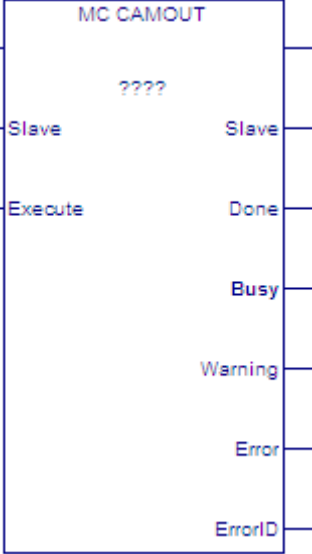
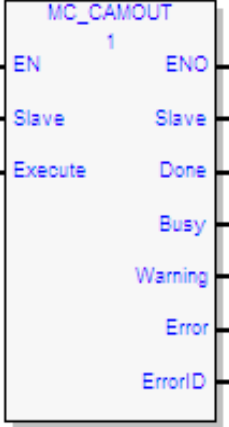
For example, with the following offsets, we get the following points:

MasterOffset: 0; SlaveOffset: 50

Master	Slave
0	-100
1000	150
2000	400

The Slave is currently at 100, but if the profile requires it to be at 150. In this case, Ramping is probably required. Refer to Section 5.6 Synchronized Motion for more on Ramping.

## 6.5 MC\_CamOut

LD	FBD	ST
 <p>The LD diagram shows a rectangular block titled "MC CAMOUT". On the left side, there are three input lines labeled "Slave" and "Execute". On the right side, there are six output lines labeled "Slave", "Done", "Busy", "Warning", "Error", and "ErrorID". The interior of the block contains the text "????".</p>	 <p>The FBD diagram shows a rectangular block titled "MC_CAMOUT" with a small "1" above it. On the left side, there are three input lines labeled "EN", "Slave", and "Execute". On the right side, there are six output lines labeled "ENO", "Slave", "Done", "Busy", "Warning", and "ErrorID".</p>	<p>Formal convention: [instance name](Slave := [input], Execute := [input], Done =&gt; [output], Busy =&gt; [output], Warning =&gt; [output], Error =&gt; [output], ErrorID =&gt; [output]);</p>

The MC\_CamOut function block disengages a slave axis from the master. The slave axis continues to move at the final velocity reached during the CAM execution.

If the slave axis had a commanded acceleration, the axis will use its acceleration and deceleration application limits to achieve the last command velocity. This command is usually followed by another command.

When MC\_CamOut is executed, the Busy output of the MC\_CamIn function block that engaged this slave is cleared.

Execution type: Immediate execution/deferred response.

## Operands

Instance Variable	Description	Allowed Data Types	Initial Value
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as ????) in LD.)	MC_CAMOUT	NA
Parameter	Description	Allowed Data Types	Initial Value
<b>Input_Output Parameters</b>			
Slave	Reference to slave axis. Virtual Axis (Axis 5) not supported.	AXIS_REF	N/A
<b>Inputs</b>			
Execute	Start to disengage the slave from the master.	LD: flow Other languages: all except constants	0
<b>Outputs</b>			
Done	Disengaging completed.	LD: flow Other languages: all except constants	0
Busy	Indicates the function block has been executed and has not yet completed its action.		1
Warning	Signals that warning has occurred within Function block.		0
Error	Signals that error has occurred within Function block.		0
ErrorID	Indicates the type of error or warning.	WORD	0



## 6.6 MC\_CamTableDeselect

LD	FBD	ST
		<p>Formal convention:  [instance name](Module := [input], CamTable := [input], Execute := [input], RemoveAll := [input], Done =&gt; [output], Busy =&gt; [output], Warning =&gt; [output], Error =&gt; [output], ErrorID =&gt; [output]);</p>

The MC\_CamTableDeselect function block deletes a CAM profile from the specified PMM to free memory. For information about checking memory, see the MC\_LibraryStatus function block.

Multiple CamTableIDs may point to a single CAM table. CamTableIDs that correspond to deselected CAM tables are invalid as inputs to MC\_CamIn.

If RemoveAll is true (1), it will attempt to remove all CAM profiles on the Module. Cams that are engaged (i.e. a CamTable input to an Active MC\_CamIn) will not be removed. If some, but not all, CAM Tables are removed, the Done and Warning outputs will be set. If none of the CAM Tables are removed, the Error output will be set.

Execution type: Immediate execution/deferred response

## Operands

Instance Variable	Description	Allowed Data Types	Initial Value
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as ????) in LD.)	MC_CAMTABLEDESELECT	NA
Parameter	Description	Allowed Data Types	Initial Value
<b>Input_Output Parameters</b>			
Module	Reference to module.	MODULE_REF	N/A
CamTable	Reference to CAM profile.	MC_CAM_REF	N/A
	Inputs		
Execute	Delete CamTable at rising edge.	LD: flow Other languages: all except constants	0
RemoveAll	When RemoveAll is set to 1, deletes all CAM tables at the rising edge of Execute.	Bool	
<b>Outputs</b>			
Done	CamTable unload done	LD: flow Other languages: all except constants	0
Busy	Indicates the function block has been executed and has not yet completed its action.		1
Warning	Signals that a warning has occurred within the function block.		0
Error	Signals that an error has occurred within the function block.		0
ErrorID	Indicates the type of error or warning.		WORD

## 6.7 MC\_CamTableSelect

LD	FBD	ST
		<p>Formal convention:  [instance name](Master :=  [input], Slave := [input],  CamTable := [input], Execute :=  [input], Periodic := [input],  Done =&gt; [output], Busy =&gt;  [output], Warning =&gt; [output],  Error =&gt; [output], ErrorID =&gt;  [output], CamTableID =&gt;  [output]);</p>

This function block loads the CAM profile from the CPU onto the PMM that the slave axis resides on. The CAM profile is available for use by any other axis on that module. The specific master and slave are not assigned until MC\_CamIn is executed.

The CamTableID is used as an input to the MC\_CamIn function block.

The master, slave, and CAM table inputs are independent. Any axis may be chosen as a master or slave for any CAM table, as long as the following conditions are met.

Loading the CAM table may take multiple sweeps. The number of sweeps depends on the size of the CAM table and the sweep frequency. When the CAM table has completed loading, the *Done* output is set and a CamTableID is available.

Execution type: Immediate execution/deferred response

## Operands

Instance Variable	Description	Allowed Data Types	Initial Value
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as ??? in LD.)	MC_CAMTABLESELECT	NA
Parameter	Description	Allowed Data Types	Initial Value
Input_Output Parameters			
Master	Reference to master axis.	AXIS_REF	N/A
Slave	Reference to slave axis.	AXIS_REF	N/A
CamTable	Reference to CAM description.	MC_CAM_REF	N/A
Inputs			
Execute	Selection at rising edge.	LD: flow Other languages: all except constants	0
Periodic	Selects one of the following CAM cycle execution modes: 1 = Periodic 0 = Non Periodic	BOOL	0
Outputs			
Done	Pre-selection done.	LD: flow	0
Busy	Indicates the function block has been executed and has not yet completed its action.	Other languages: all except constants	1
Warning	Signals that warning has occurred within Function block.		0
Error	Signals that error has occurred within the function block.		0
ErrorID	Indicates the type of error or warning.	WORD	0
CamTableID	Identifier of CAM Table to be used in the MC_CamIn function block. CamTableID is not reset on the falling edge of Execute.	MC_CAM_ID	0

## 6.7.1 Periodic (CAM Cycle Execution Mode)

### Periodic



In *Periodic* mode, when the Master axis position reaches the maximum value in the Master Position column of the CAM table it rolls over to the minimum position and vice-versa. The MC\_CamIn function block's EndofProfile output is set each time the Master axis position rolls over.

### Non-Periodic

In this mode, when the Master axis position reaches the maximum or minimum value in the Master Position column of the CAM table, the MC\_CamIn function block sets its EndofProfile output and the CAM disengages. The slave axis transitions from the *Synchronized Motion* to the *Continuous Motion* state and continues moving at the last commanded velocity.

A CAM selected as non-periodic allows buffering of function blocks after it.

## 6.8 MC\_DelayedStart

LD	FBD	ST
 <p>LD symbol for MC_DELAYEDSTART. The symbol is a rectangle with the text 'MC_DELAYEDSTART' at the top. Inside, there is a '1' and '????' above it. On the left side, there are six input lines labeled 'AxisArray', 'Execute', 'DelayTimes', 'StartTime', and 'AbsoluteStart'. On the right side, there are six output lines labeled 'AxisArray', 'Done', 'Busy', 'Warning', and 'Error'. At the bottom right, there is an output line labeled 'ErrorID'.</p>	 <p>FBD symbol for MC_DELAYEDSTART. The symbol is a rectangle with the text 'MC_DELAYEDSTART' at the top and a '1' below it. On the left side, there are six input lines labeled 'EN', 'AxisArray', 'Execute', 'DelayTimes', 'StartTime', and 'AbsoluteStart'. On the right side, there are six output lines labeled 'ENO', 'AxisArray', 'Done', 'Busy', 'Warning', and 'Error'. At the bottom right, there is an output line labeled 'ErrorID'.</p>	<p>Formal convention:          [instance name](AxisArray          := [input], Execute :=          [input], DelayTimes :=          [input], StartTime :=          [input], AbsoluteStart :=          [input], Length := [input],          Done =&gt; [output], Busy =&gt;          [output], Warning =&gt;          [output], Error =&gt;          [output], ErrorID =&gt;          [output]);</p>

The MC\_DelayedStart function block operates very similarly to the MC\_SyncStart function block, except that the axes can start with a delay relative to each other.

The DelayTimes array is the same length as the AxisArray. Each element specifies the delay for the corresponding axis. The delay times indicate the delay in seconds between when all axes are ready to start and when the axis will actually start. There is also a setup time required that is specified by StartTime, which is accurate to 1ms.

MC\_DelayedStart will be aborted if an MC\_Stop, MC\_Reset, MC\_Power, or MC\_ModuleReset function block is executed on an axis or module associated with the MC\_DelayedStart instruction.

Execution type: Immediate execution/deferred response

## Operands

Instance Variable	Description	Allowed Data Types	Initial Value
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as ??? in LD.)	MC_DELAYEDSTART	NA
Parameter	Description	Allowed Data Types	Initial Value
??	Length. The number of axes to be synchronized (valid range is 1–8), as specified in AxisArray and DelayTimes.	Constant	1
Input_Output Parameters			
AxisArray	Array of axes to be synchronized (maximum of 8).	AXIS_REF[ ]	N/A
Inputs			
Execute	Execute MC_DelayedStart function block	LD: flow Other languages: all except constants	0
DelayTimes	Array of delay times in seconds between when each axis will actually start and when all axes are ready to start (maximum of 8).	LREAL[ ]	0
StartTime	The maximum time (in ms) that can elapse between the execution of the MC_DelayedStart function block and when the axes are ready to start. If motion is not ready to start on all axes in this amount of time, an error occurs. A time of 0 indicates a time limit of five minutes. The time when the axes will actually start is determined by the DelayTimes array.	UINT	0
AbsoluteStart	If set to 1, the axes must be ready at exactly the time specified by StartTime. If set to 0, the motion will start as soon as can be coordinated.	BOOL	0
Outputs			
Done	Axes have started in sync	LD: flow	0
Busy	Indicates the function block has been executed and has not yet completed its action.	Other languages: all except constants	0
Warning	Signals that warning has occurred within Function block		0
Error	Signals that error has occurred within Function block		0
ErrorID	Indicates the type of error or warning	WORD	0

## 6.9 MC\_DigitalCamSwitch

LD	FBD	ST
<p>LD diagram showing the MC_DIGITALCAMSWITCH function block with inputs: Axis, Switches, Outputs, TrackOptions, Enable, EnableMask, PositionSource and outputs: Axis, Switches, Outputs, TrackOptions, InOperation, Busy, Warning, Error, ErrorID.</p>	<p>FBD diagram showing the MC_DIGITALCAMSWITCH function block with inputs: EN, Axis, Switches, Outputs, TrackOptions, Enable, EnableMask, PositionSource and outputs: ENO, Axis, Switches, Outputs, TrackOptions, InOperation, Busy, Warning, Error, ErrorID.</p>	<p>Formal convention:  [instance name](Axis := [input], Switches := [input], Outputs := [input], TrackOptions := [input], Enable := [input], EnableMask := [input], PositionSource := [input], InOperation =&gt; [output], Busy =&gt; [output], Warning =&gt; [output], ErrorID =&gt; [output]);</p>

The ability to control an output point based on position and position/time is a feature utilized in many motion applications. This feature has many names including Digital CAM Switch (DCS) and Programmable Limit Switch. The function is implemented in the PACMotion controller via the function block MC\_DigitalCamSwitch.

This function allows the user to control an output point to emulate and extend the function of a mechanically controlled CAM switch. To illustrate the usage of this function each input is described, and an example given.

Each DCS command specifies an axis as its position source. Axes 1–4 support DCS commands; Axes 5 (Virtual) does not support DCS commands. Only one DCS command may be enabled at a time per axis, and no more than four output points may be under DCS control at a time. Each DCS command may control up to four outputs.

An output point can be controlled by one DCS command at a time. So, up to four DCS commands may be enabled on a PMM at a time.

When an MC\_DigitalCamSwitch function block is first enabled, the output points it controls are not affected until the axis is moving. Once the axis moves in either direction, the MC\_DigitalCamSwitch function block controls the output points based on the axis position and the defined switch points.



When an MC\_DigitalCamSwitch function blocks transitions from enabled to disabled, or an error occurs that aborts the MC\_DigitalCamSwitch function block, the outputs will either Hold Last State or default Off based on the Outputs Default setting in hardware configuration.

If an operational error occurs that affects an MC\_DigitalCamSwitch function block that is controlling outputs, the axis the MC\_DigitalCamSwitch is running on will be Normal Stopped. For example, if an input parameter of an active MC\_DigitalCamSwitch function block is changed to an invalid value, the MC\_DigitalCamSwitch function block will be aborted with an appropriate Error ID and the axis will perform a Normal Stop.

Execution type: Immediate execution/deferred response.

## Operands

Instance Variable	Description	Allowed Data Types	Initial Value
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as ??? in LD.)	MC_DigitalCamSwitch	NA
Parameter	Description	Allowed Data Types	Initial Value
Input_Output Parameters			
Axis	Axis that receives function block command.	AXIS_REF	N/A
Switches	Reference to the switching actions. For details, refer to Switches in Section 6.9.2.	CAMSWITCH_REF	N/A
Outputs	Reference to the signal outputs directly related to the referenced tracks. Maximum of four. The first output corresponds to the first TrackNumber. For details, refer to Outputs in Section 6.9.2.	DCS_OUTPUTs	NA
TrackOptions	Reference to structure containing Hysteresis and ON and OFF compensations for each output/track. For details, refer to TrackOptions in Section 6.9.2.	TRACK_REF	N/A
Inputs			
Enable	Enables the Switches outputs.	BOOL	0
EnableMask	Four bits. Enables the different tracks. Least significant data is related to the lowest TrackNumber. To enable a TrackNumber, set the corresponding mask bit to 1.	DWORD	0
PositionSource	Identifies the source of the Position. Actual position: Source is the configured feedback device. Commanded position: Source is the instantaneous position generated by the PMM's internal path generator. If Actual Position is selected and Axis 5 is the master, Axis 5 must use an external feedback device.	MC_PositionSource	0
Outputs			
InOperation	Indicates when the commanded tracks are enabled.	LD: flow	0

Instance Variable	Description	Allowed Data Types	Initial Value
Busy	Indicates the function block is enabled and has not completed its action.	Other languages: all except constants	1
Warning	Indicates that a warning has occurred within the function block.		0
Error	Indicates that an error has occurred within the function block.		0
ErrorID	Error or warning identification.	WORD	0

## 6.9.1 Requirements for Switch and Track Option Selections

The following requirements should be observed when configuring a digital CAM switch.

- Either Hysteresis or On/Off Compensation TrackOptions may be used, but not both.
- The TrackOptions of Hysteresis or On/Off Compensation can only be used with a switch that has a CamSwitchMode set to Position.
- If Hysteresis is to be used, the switch's AxisDirection must be set to Both.
- Switches should be designed so that they do not overlap. Overlap of position-based switch points, including any hysteresis window, results in an error when the DCS is activated. Overlap of time-based switch points, including position-based switch points with on/off compensation, results in an error during runtime when the overlap occurs. If run-time overlap occurs, the active switch point remains active and the other switch point is ignored.

## 6.9.2 Enabling Outputs for DCS Control

Before you can use DCS, outputs must be enabled for DCS control by writing the appropriate mask values to parameter 2114 for faceplate digital outputs and to parameter 2115 for FTB outputs. Any of the Digital Output Sources may be *enabled* for DCS control, but only four may be controlled at a time.

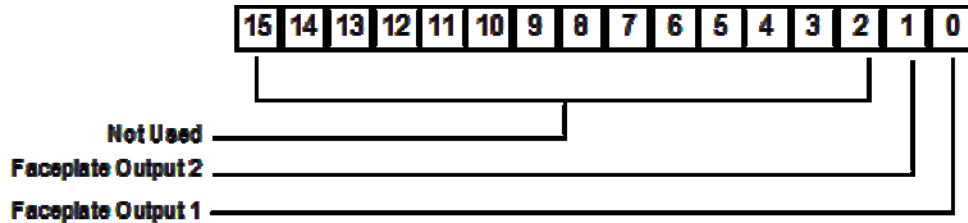
### DCS Mask Values

To enable a digital output for use with the DCS, its corresponding mask bit must be set to 1.

## Parameter 2114 – Faceplate Digital Output Source

If a Faceplate digital output is used, it must be configured as Digital Output in HWC.

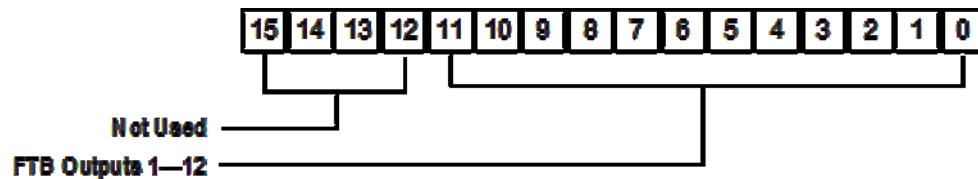
Figure 85: Faceplate Digital Output Source



## Parameter 2115 – FTB Digital Output Source

If an FTB output 9–12 is used, it must be configured as Fast Digital Output in HWC.

Figure 86: FTB Digital Output Source



Example: To enable FTB outputs 1 through 4, write a value of 0x000 000F to parameter number 2115.

### Axis

The Axis input is used to associate the switches to a particular axis position source. The Position Source input allows you to define whether the position source is actual or commanded position. Only one Digital CAM Switch command can be active on an axis at a time.

### Switches

The Switches input is used to define the relationship between desired switching points and a track. A track is a virtual representation of a particular physical output point. The Outputs input parameter (refer to Outputs in Section 6.9.2) allows you to associate the track to the physical output point. The Switches input is a variable of type CAMSWITCH\_REF. This data type is a structure that allows you to configure eight switches and their track associations.

## CAMSWITCH\_REF Data Type Structure

A CAMSWITCH\_REF variable has the following elements:

Element	Type	Values
Switch1 through Switch 8	SWITCH	
AxisDirection	INT	0 = both 1 = positive 2 = negative
CamSwitchMode	INT	0 = position-based 1 = time-based
Duration	LREAL	Time value. Applies only to the time-based CamSwitchMode.
FirstOnPosition	LREAL	Lower boundary where the switch is ON.
LastOnPosition	LREAL	Upper boundary where the switch is ON.
TrackNumber	INT	Reference to the track number (1-4).
Switches	INT	Number of the above Switch structures to activate, starting with the first (1-8).

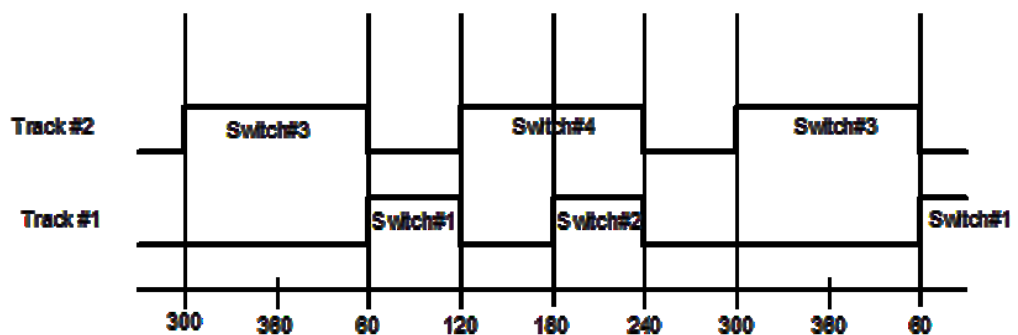
### Example: Setting up the DCS Switches Parameter

This example illustrates how to set up two tracks with two switch points each. In this example, the *Axis Position Mode* is *Rotary* (set up in the axis hardware configuration) and the positions are programmed in degrees. The application requires the following switch patterns.

	Axis Direction	Cam Switch Mode	Duration	First On Position	Last On Position	Track Number
Switch 1	0=Both	0=Position	-	60	120	1
Switch 2	0=Both	0=Position	-	180	240	1
Switch 3	0=Both	0=Position	-	300	60	2
Switch 4	0=Both	0=Position	-	120	240	2

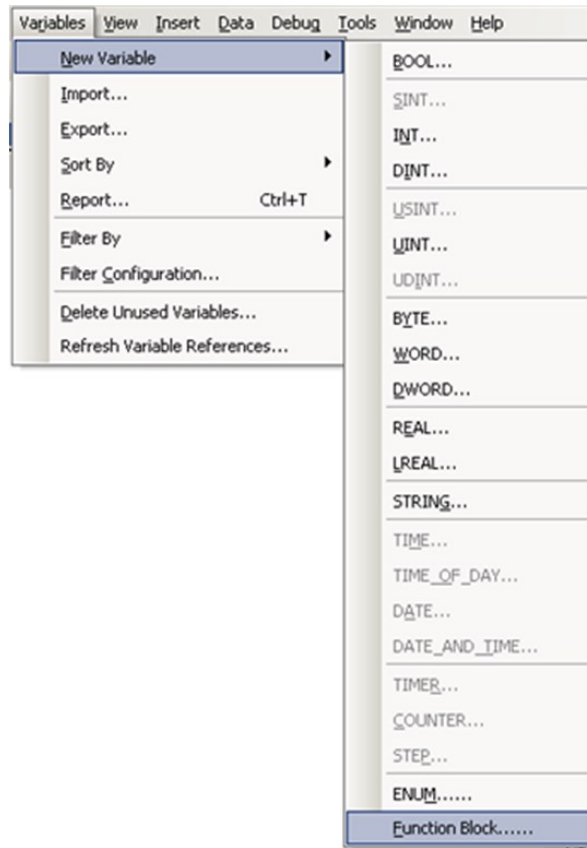
These switch patterns correspond to the following switching sequences.

Figure 87: Switching Sequence Example



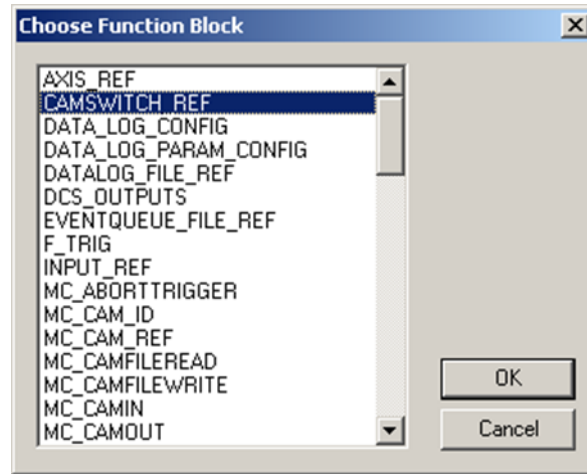
To achieve this switch pattern, first create a new variable called Rotary\_Switch with the type CAMSWITCH\_REF. To perform this step, select Variables from the main menu and the sub menu NewVariable followed by the selection Function Block.

Figure 88: Create New Variable



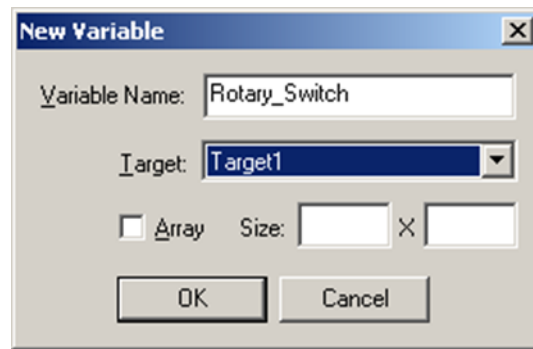
The Choose Function Block dialog box appears, which allows you to select the type for the variable.

Figure 89: Choose Function Block



Click OK and then give the variable the name Rotary\_Switch as shown below

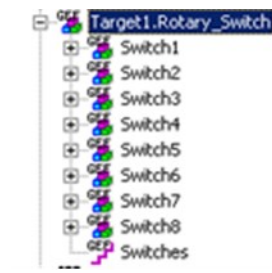
Figure 90: Assign Variable Name



The Switches input on the digital CAM switch contains a structure that defines the switch points and the tracks association. (The Outputs input will be used to pair a CAM switch track to a physical output.)

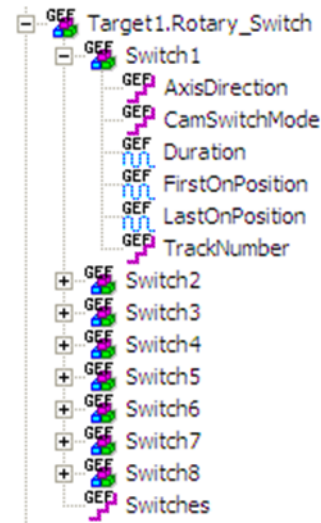
This places a new variable called Rotary\_Switch in the variable list. Navigate to this variable in the Variables tab in the Navigator pane.

Figure 91: View Rotary\_Switch in PME Variables Tab



Edit the Properties of the Rotary\_Switch variable elements to define the desired switching pattern. The variable elements in the Switch1–Switch8 structures are defined as follows:

**Figure 92: Edit the Properties of the *Rotary\_Switch* Variable Elements**



Axis Direction – Specifies the motion direction that will cause the switch to function. The valid directions are 0 = Both, 1 = Positive Direction, 2 = Negative Direction.

CAM Switch Mode – Specifies whether the last switch point is controlled by the LastOnPosition or the Duration parameter. 0 = LastOnPosition (position-based), 1 = Duration (time-based).

Duration – Specifies the time in ms that a time-based switch remains on after the axis passes the FirstOnPosition.

FirstOnPosition – Lower boundary where the digital CAM switch is to activate. If FirstOnPosition > LastOnPosition, the switch is inverted. This means the switch is off during the interval instead of on.

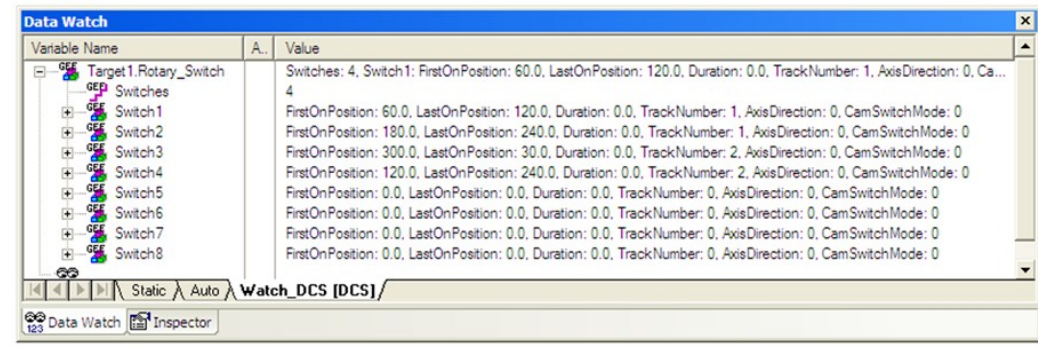
LastOnPosition- Upper boundary where CAM switch is on.

TrackNumber- Track number to associate with the switch (1–4).

Set the Switches element to the number of switches that will be activated in the DCS.

Once the switching pattern definition is complete, you can view the variables in the Data Watch window to check that the definition is correct.

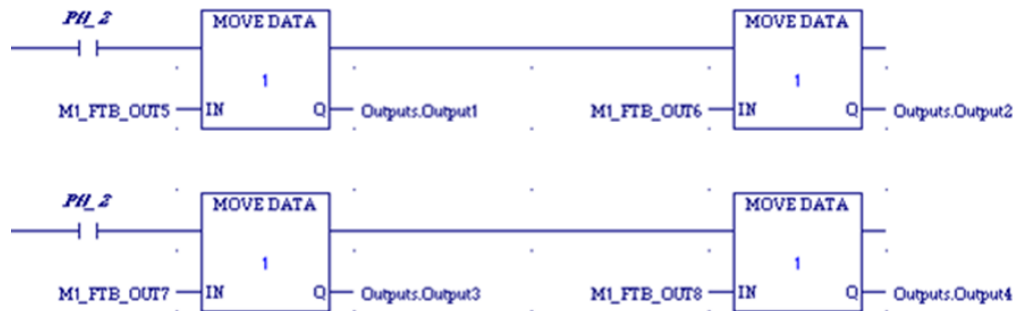
Figure 93: View the Variables in the Data Watch Window



## Outputs

The outputs structure links the physical output to the tracks (First Track Number = First Output). The digital CAM switch is supported on four outputs per module at a time. The outputs must be on the same module as the position source axis. To populate the DCS\_OUTPUTS array, copy the desired output variables into the array. A ladder example to copy four output references (M1\_FTБ\_OUT5, M1\_FTБ\_OUT6, M1\_FTБ\_OUT7, and M1\_FTБ\_OUT8) is shown below.

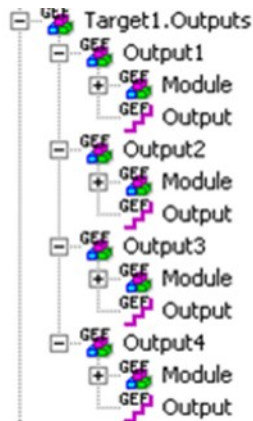
Figure 94: Ladder Logic to Copy Four Outputs to DCS\_OUTPUTS array



This moves the references into the Outputs array shown below and associates the four outputs with the tracks. In this case track 1 is linked to M1\_FTБ\_OUT5, track 2 is linked to M1\_FTБ\_OUT6, track 3 is linked to M1\_FTБ\_OUT7, and track 4 is linked to M1\_FTБ\_OUT8 via the DCS function block. An output can be controlled by only one Digital CAM Switch command at a time.



Figure 95: Linkage of Four Outputs with Tracks



## TrackOptions

The TrackOptions structure allows you to apply Hysteresis or On/Off compensation to each track.

---

**Note:** *Hysteresis and On/Off compensation cannot be used on the same track. Only one type of compensation is allowed per track.*

---

**Hysteresis** - The hysteresis parameter is used to avoid switching chatter caused by position oscillating around a switch point. The parameter specifies the distance from the switch point in both the positive and negative direction in which the switch is not executed until the position has left this area.

Hysteresis can only be used with position-based switches. It cannot be used with time-based (Duration) switches. If hysteresis is used, AxisDirection must be set to Both.

In the following example (Figure 98), a switch on position is set to 4000 with both directions of travel enabled. The CAM switch is executed, and the axis stops exactly at position 4000. For this example, the Position Source parameter is set to Actual and the position oscillates around 4000 by 1 user unit. The Hysteresis is set to 2 user units so the switch will remain on when the position oscillates between 3999 and 4001. When motion in the negative direction is triggered, the switch will turn on when the axis reaches 4002 and off when the position moves past 3998.

Figure 96: Track Options – Hysteresis

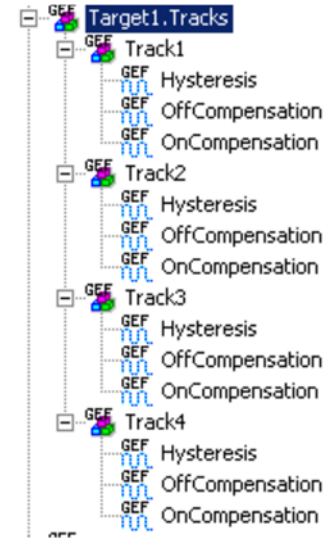
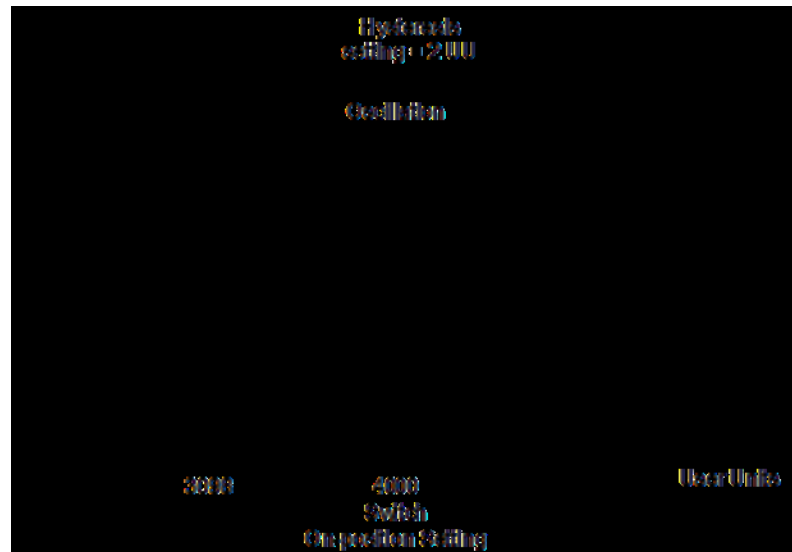


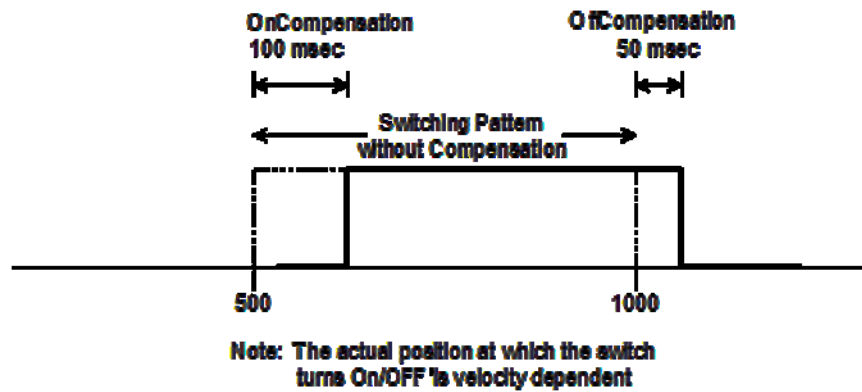
Figure 97: Hysteresis Setting Example



**OffCompensation** – This parameter allows the switch-off time to be delayed by the compensation amount. This applies for the entire track. This is illustrated in the figure below.

**OnCompensation**- This parameter allows the switch-on time to be advanced in time by the compensation amount. This applies for the entire track. This is illustrated in the figure below.

Figure 98: On/Off Compensation Example



Acceleration or deceleration may affect the accuracy of the compensation. For details, refer to Appendix A-2 Digital CAM Switch Accuracy.

## Enable

The enable input activates the switches that are enabled in the EnableMask.

## EnableMask

The enable mask allows you to selectively enable specific (1-4) tracks. The least significant bit in the mask corresponds to track number 1. For example, to enable tracks 1, 3 and 4 the value for this variable would be 000D hex (1101 in binary).

## 6.10 MC\_DL\_Activate

LD	FBD	ST
		<p>Formal convention: [instance name](Module := [input], Enable := [input], DataCaptureID := [input], InputTrigger := [input], DataAvailable =&gt; [output], LogNumSamples =&gt; [output], LogDataCaptureID =&gt; [output], Logging =&gt; [output], Warning =&gt; [output], Error =&gt; [output], ErrorID =&gt; [output]);</p>

This function block is used to start data logging on the module. Its operation is controlled by the MC\_DL\_Configure function block, described in Section 6.11.

The Enable input to MC\_DL\_Activate enables logging. Setting this input false disables data logging.

In *triggered* start mode, the Trigger Mode, which is specified by the MC\_DL\_Configure function block, determines when data logging starts and stops.

The Logging output of the instance is set true as long as the function block is collecting data.

Even if the *Enable* input is off, any error in the inputs of the MFB will result in the Error and ErrorID outputs being set. Setting the inputs to valid values clears the Error and ErrorID outputs. When the *Enable* input transitions off, if data has been logged, the DataAvailable, LogNumberSamples and LogDataCaptureID are set to the values from the data logging session.

If a warning is present, the Warning and ErrorID outputs are set and other outputs are off. If no data is logged or if an error has occurred that prevented data logging, all outputs of the function block are cleared when the *Enable* input transitions off.

If the *Enable* input is on and the inputs of the MC\_DL\_Activate are changed from valid values to invalid values, the Error and ErrorID outputs of the MC\_DL\_Activate will be set and data capture will be terminated.

For an example of data logging operation, refer to 6.13.1 Data Logging Example.

**Note:** There can only be one data logging session active with MC\_DL\_Activate. Multiple instances of MC\_DL\_Activate on the same module are allowed, but not recommended. The last executed instance of MC\_DL\_Activate takes precedence over the previous instance.

Execution type: Immediate execution/deferred response.

## Operands

Instance Variable	Description	Allowed Data Types	Initial Value
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as ????) in LD.)	MC_DL_ACTIVATE	NA
Parameter	Description	Allowed Data Types	Initial Value
<b>Input_Output Parameters</b>			
Module	Identifies the module to work upon.	MODULE_REF	N/A
Inputs			
Enable	Start logging data	LD: flow Other languages: all except constants	0
DataCapture ID	Identifies the Data Logging configuration output of the MC_DL_CONFIGURE function block (Section 6.11 MC_DL_Configure).	WORD	0
InputTrigger	(Optional) The operation of this input is controlled by the trigger mode selected by the MC_DL_Configure function block instance that is specified by the DataCaptureID parameter.	BOOL	0
<b>Outputs</b>			
DataAvailable	Indicates either a full set of data samples has been logged or some data has been logged and the data logger was deactivated (the Enable input transitioned low).	BOOL	0
LogNumSamples	The number of samples of data logged, when DataAvailable transitioned high.	DWORD	
LogCaptureID	The DataCapture for the logged data, when DataAvailable transitioned high.	WORD	
Logging	Set while the function is actively logging data.	BOOL	0
Warning	Indicates that a warning has occurred within the function block.	LD: flow Other languages: all except constants	0
Error	Indicates that an error has occurred within the function block.		0
ErrorID	Error or warning identification.	WORD	0

## 6.11 MC\_DL\_Configure

LD	FBD	ST
		<p>Formal convention: [instance name](Module := [input], ParameterConfig := [input], DataLogConfig := [input], Execute := [input], Done =&gt; [output], Busy =&gt; [output], Warning =&gt; [output], Error =&gt; [output], ErrorID =&gt; [output], DataCaptureID =&gt; [output]);</p>

This function block specifies the parameters for data monitored on the PMM using the Data Logging Window.

The ParameterConfig input parameter specifies the list of Module and Axis parameters to capture by specifying the parameter ID in the module or axis parameters arrays. For details, refer to DATA\_LOG\_PARAM\_CONFIG Data Structure in Section DATA\_LOG\_PARAM\_CONFIG Data Structure.

The DataLogConfig specifies the Number of Samples, Sampling Rate, Operating Mode, Trigger Mode and Post Sample Percentage. For details, refer to DATA\_LOG\_PARAM\_CONFIG Data Structure in Section.

Data logging is initiated by the MC\_DL\_Activate function block, described in Section 6.10, MC\_DL\_Activate.

For an example of data logging operation, refer to Section 6.13.1.

Execution type: Immediate execution/deferred response.

## Operands

Instance Variable	Description	Allowed Data Types	Initial Value
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as ????) in LD.)	MC_DL_CONFIGURE	NA
Parameter	Description	Allowed Data Types	Initial Value
????	Instance variable name.	MC_DL_CONFIGURE	N/A
Input_Output Parameters			
Module	The module on which the function block is to be executed.	MODULE_RE	
ParameterConfig	Identifies the module to work upon, its parameters, the axes on the module and their parameters. Refer to DATA_LOG_PARAM_CONFIG Data Structure in Section DATA_LOG_PARAM_CONFIG Data Structure.	DATA_LOG_PARAM_CONFIG	N/A
DataLogConfig	Data Logger configuration. Refer to DATA_LOG_CONFIG Data Structure in Section DATA_LOG_PARAM_CONFIG Data Structure.	DATA_LOG_CONFIG	N/A
Inputs			
Execute	The rising edge configures the data logger.	LD: flow Other languages: all except constants	0
Outputs			
Done	Data logger configuration has completed and the DataCaptureID is available	LD: flow Other languages: all except constants	0
Busy	Indicates the function block been executed and has not yet completed its action.		1
Warning	Indicates that a warning has occurred within the function block.		0
Error	Indicates that an error has occurred within the function block.		0
ErrorID	Error or warning identification.	WORD	0
DataCaptureID	Identifier of this configuration, used as an input to the MC_DL_Activate, MC_DL_Get and MC_DL_Delete function blocks.	WORD	0

## DATA\_LOG\_PARAM\_CONFIG Data Structure

This data type is used to define the parameters associated with the module and axis to be monitored by the data logger. A DATA\_LOG\_PARAM\_CONFIG variable contains the following elements. A value of zero for a parameter indicates that it is not being used.

For a list of parameter numbers, refer to Axis Parameter Number Index in Section 8.1.1.

Element	Type	Description
Axis1Parameters— Axis5Parameters	Array of eight INT variables	Specifies an array of axis parameter numbers to be logged on the PMM.  If a sampling rate of 500us is configured, only six parameters can be specified. If a sampling rate of 250us is configured, only four parameters can be specified, and the four parameters must be axis parameters, not module parameters.  Each parameter must be listed only once. Duplicate parameters will cause the data logging session to fail.
ModuleParameters	Array of eight INT variables	Specifies an array of module parameter numbers to be logged on the PMM.  If axis parameters are specified in the ModuleParameters array, Axis 1 parameters values will be returned.

## DATA\_LOG\_CONFIG Data Structure

This data type is used to define the parameters associated with the Data Logger function. A DATA\_LOG\_CONFIG variable contains the following elements:

Element	Type	Description
NumberSamples	DINT	Specifies the number of samples to be logged. The PMM provides 2MB of memory for logging data. If specifying a combination of parameters and sampling rate that will yield a large number of samples, you may want to calculate the maximum number of samples that can be logged as documented in Calculating the Maximum Number of Samples, below.
OperatingMode	DL_OPERATING_MODE	Single (default) or Circular
PostSamplePercent	DL_POST_SAMPLE	Specifies the percentage of the buffer that will be filled with data after the trigger condition is true. Possible values: 25%, 50%, 75%  This parameter applies only to Combined Trigger and Pre-Trigger modes.
SamplingRate	DL_SAMPLING_RATE	Specifies the sample rate. Possible values: 250μs, 500μs, 1000μs, 2000μs
TriggerMode	DL_TRIGGER_MODE	Specifies the data logging trigger mode. Possible values: None, Pre-Trigger, Post-Trigger, Combined Trigger



## Operating Modes

Single	The number of samples specified by NumberSamples is logged in the Data Logger buffer in PMM memory. Once this number of samples has been logged, data logging stops. The data logger stops when it is de-activated, even if the number of samples specified has not been logged. Supports None and PostTrigger trigger modes.
Circular	Data is continuously logged in the Data Logger buffer space. Once the number of samples specified has been logged, the data logger wraps around to the beginning of the buffer to start logging data again. Supports all trigger modes.

## Trigger Modes

Trigger Mode	Logging Start/Stop	Single Operating Mode	Circular Operating Mode
None	Starts when	MC_DL_Activate Enable input transitions high.	MC_DL_Activate Enable input transitions high. (The DataAvailable output will transition high the first time the number of samples fills the buffer size.)
	Stops when	NumberSamples has been logged. (The DataAvailable output will be high.) or The MC_DL_Activate Enable input transitions low. (The DataAvailable output will be high if data is available.)	The MC_DL_Activate Enable input transitions low. (The DataAvailable output will be high if data is available.)
PreTrigger	Starts when	Not applicable	MC_DL_Activate Enable transitions high.
	Stops when	Not applicable	The InputTrigger transitions high and the logging of post samples is complete. (The DataAvailable output will be high if data is available.) or The Enable input transitions low. (The DataAvailable output will be high if data is available.)
PostTrigger	Starts when	MC_DL_Activate Enable input transitions high.	The MC_DL_Activate is active (Enable input is high) and InputTrigger transitions high.
	Stops when	Data logging stops when the MC_DL_Activate InputTrigger input or Enable input transitions low. (The DataAvailable output will be high.)	MC_DL_Activate Enable or InputTrigger transitions low. (The DataAvailable output will be high if data is available.)

Trigger Mode	Logging Start/Stop	Single Operating Mode	Circular Operating Mode
Combined Trigger	Starts when	Not applicable	MC_DL_Activate Enable input transitions high. When InputTrigger goes high, logging of post samples starts.
	Stops when	Not applicable	InputTrigger is high and post sampling is complete. (The DataAvailable output will be high.) or The MC_DL_Activate Enable input transitions low. (The DataAvailable output will be high if data is available.)

## Calculating the Maximum Number of Samples

**Note:** This calculation is required only when configuring larger data logs. For small data logs the desired number of samples can be specified without going through this computation.

The PMM provides 2MB of memory for logging data. The maximum number of samples is based on the parameter set of the data to be logged. To calculate the maximum number of samples:

$$\text{MaxNumberSamples} = 2\text{Mbyte} / (\text{DataSetSize} + \text{TimeStampSize})$$

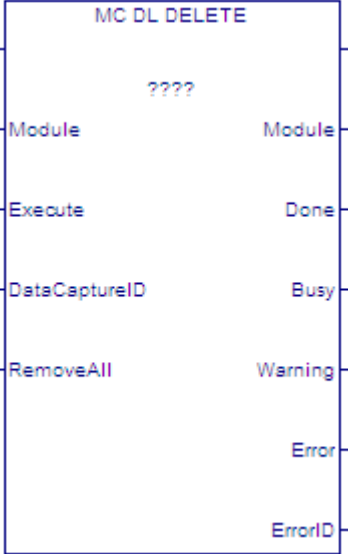

- DataSetSize is based on the number of parameters to log, the parameter type and the number of axes.
- TimeStampSize is 4, 32bit number for indicating the timestamp of every position loop interval.
- The maximum space available for data logging is 2MB (2097152 bytes)

### Example for calculating the maximum number of samples:

- Two real parameters (8 bytes)
- Two axes

$$\text{MaxNumberSamples} = \frac{[2097152 \text{ bytes}]}{[8\text{bytes} * 2 * 2] + 4\text{bytes}} \rightarrow 58254 \text{ samples}$$

## 6.12 MC\_DL\_Delete

LD	FBD	ST
 <p>The LD diagram shows a rectangular block titled "MC DL DELETE" with a "?????" label inside. On the left side, there are four input terminals labeled "Module", "Execute", "DataCaptureID", and "RemoveAll". On the right side, there are six output terminals labeled "Module", "Done", "Busy", "Warning", "Error", and "ErrorID".</p>	 <p>The FBD diagram shows a rectangular block titled "MC_DL_DELETE" with a "1" label inside. On the left side, there are four input terminals labeled "EN", "Module", "Execute", and "RemoveAll". On the right side, there are six output terminals labeled "ENO", "Module", "Done", "Busy", "Warning", and "ErrorID".</p>	<p>Formal convention:  [instance name](Module := [input], Execute := [input], DataCaptureID := [input], RemoveAll := [input], Done =&gt; [output], Busy =&gt; [output], Warning =&gt; [output], Error =&gt; [output], ErrorID =&gt; [output]);</p>

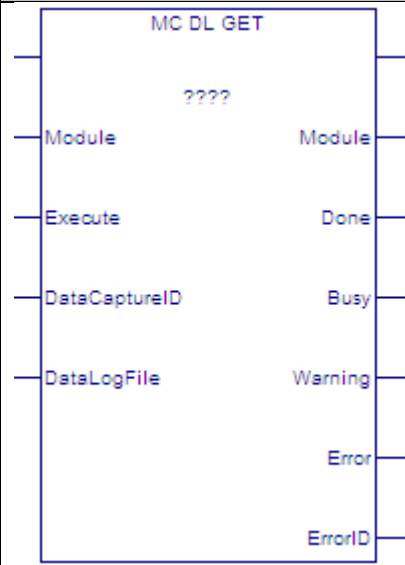
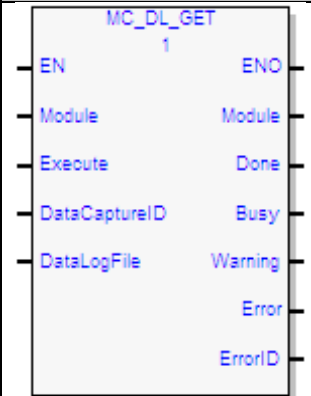
This function block deletes a data logger configuration from the PMM memory. Once deleted, the space allocated to logging the data defined by the configuration is freed.

*Execution type:* Immediate execution/deferred response.

## Operands

Instance Variable	Description	Allowed Data Types	Initial Value
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as ???? in LD.)	MC_DL_CONFIGURE	NA
Parameter	Description	Allowed Data Types	Initial Value
<b>Input_Output Parameters</b>			
Module	Identifies the module to work upon.	MODULE_REF	N/A
<b>Inputs</b>			
Execute	The rising edge deletes the data logging configuration.	LD: flow Other languages: all except constants	0
DataCaptureID	Identifies the Data Logging configuration to be deleted. This is an output of MC_DL_CONFIGURE (refer to Section 6.11).	WORD	0
RemoveAll	If this input is ON, on the rising edge of Execute, all data log configurations will be deleted from the specified motion module.	BOOL	
<b>Outputs</b>			
Done	Indicates the data logger configuration specified by DataCaptureID has been deleted or all the data logger configurations on the module have been deleted.	LD: flow Other languages: all except constants	0
Busy	Indicates the function block has been executed on an axis and has not yet completed its action.		1
Warning	Indicates that a warning has occurred within the function block.		0
Error	Indicates that an error has occurred within the function block.		0
ErrorID	Error or warning identification.	WORD	0

## 6.13 MC\_DL\_Get

LD	FBD	ST
 <p>MC DL GET</p> <p>????</p> <p>Module</p> <p>Execute</p> <p>DataCaptureID</p> <p>DataLogFile</p> <p>Done</p> <p>Busy</p> <p>Warning</p> <p>Error</p> <p>ErrorID</p>	 <p>MC_DL_GET 1</p> <p>EN</p> <p>Module</p> <p>Execute</p> <p>DataCaptureID</p> <p>DataLogFile</p> <p>ENO</p> <p>Module</p> <p>Done</p> <p>Busy</p> <p>Warning</p> <p>Error</p> <p>ErrorID</p>	<p>Formal convention: [instance name](Module := [input], Execute := [input], DataCaptureID := [input], DataLogFile := [input], Done =&gt; [output], Busy =&gt; [output], Warning =&gt; [output], Error =&gt; [output], ErrorID =&gt; [output]);</p>

This function block gets the data log file specified by the DataCaptureID output of the MC\_DL\_Configure function block and writes the data into the memory location specified by the DataLogFile input.

The MC\_DL\_GET operation adds approximately 40 ms to the host controller sweep time while the data file is being transferred to the CPU from the PMM module. The Controller Communication Window Mode should be set to Limited to minimize the impact of data logging file transfer impact to the host controller time.

Execution type: Immediate execution/deferred response.

## Operands

Instance Variable	Description	Allowed Data Types	Initial Value
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as ????) in LD.)	MC_DL_GET	NA
Parameter	Description	Allowed Data Types	Initial Value
<b>Input_Output Parameters</b>			
Module	Identifies the module to work upon.	MODULE_REF.	N/A
<b>Inputs</b>			
Execute	The rising edge stores the data logged into a file.	LD: flow Other languages: all except constants	0
DataCaptureID	Identifies the Data Logging configuration output of the MC_DL_CONFIGURE function block (refer to Section 6.11).	WORD	0
DataLogFile	Specifies a file reference to the stored data log.	DATALOG_FILE_REF	0
<b>Outputs</b>			
Done	Data has been read into the file	LD: flow Other languages: all except constants	0
Busy	Indicates the function block has been executed on an axis and has not yet completed its action.		1
Warning	Indicates that a warning has occurred within the function block.		0
Error	Indicates that an error has occurred within the function block.		0
ErrorID	Error or warning identification.	WORD	0

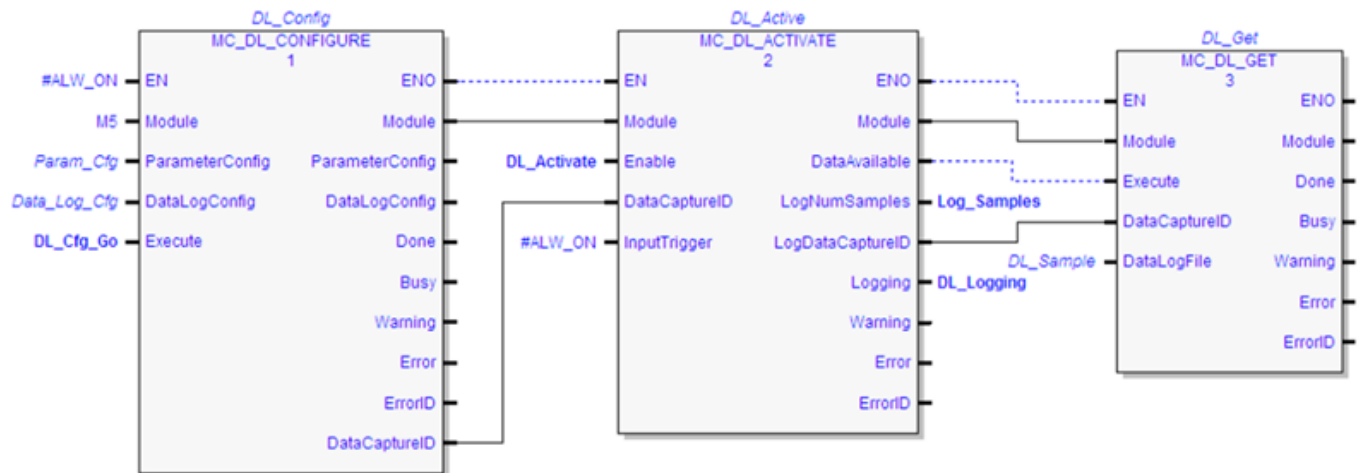
## 6.13.1 Data Logging Example

The data logging function provides the ability to capture data locally on a PMM at a configurable rate up to the module sample rate. This function provides a diagnostic and tuning tool to aid application development.

The basic steps for data logging are:

1. Configure the information to be logged using the MC\_DL\_Configure function block.
2. Activate the data logging session using the MC\_DL\_Activate function block.
3. Wait for the data logging to trigger and complete capture in the PMM module. Completion of the data logging can be determined by monitoring the DataAvailable output. The DataAvailable output is set to complete once the requested data has been logged.
4. Transfer the logged data information from the PMM to the controller using the MC\_DL\_Get function block, specifying the name of the DLOG file to be created.
5. Upload the data log (DLOG) files from the PMM module to the PC using the Controller File Explorer utility.
6. Examine the resulting data using the Data Logging View utility.

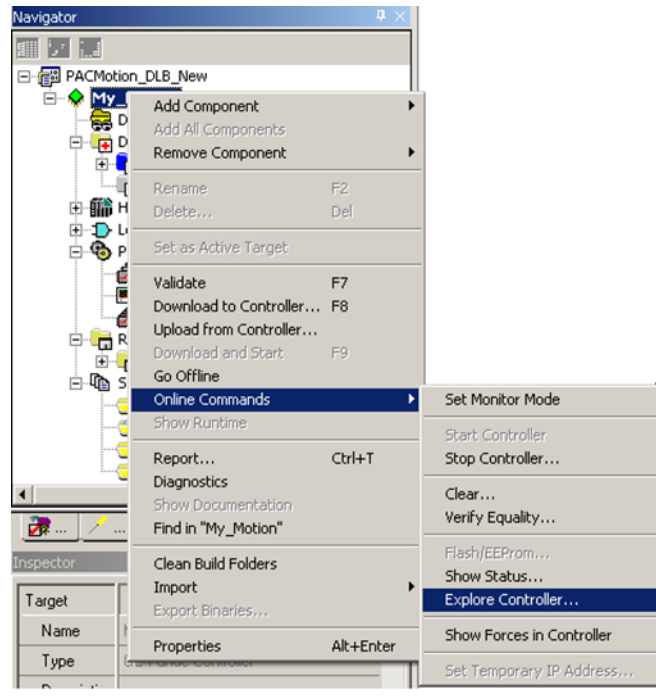
**Figure 99: Sample Logic to Configure, Activate, and Get a Data L**



## Uploading Data Logging (DLOG) Files from PACSystems Controller

The uploading of DLOG files is performed using the Controller File Explorer. This utility is accessed from the Online Commands menu on the Active Target or from the Target menu on the Main Menu.

**Figure 100: Uploading Data Logging (DLOG) Files from PACSystems Controller**

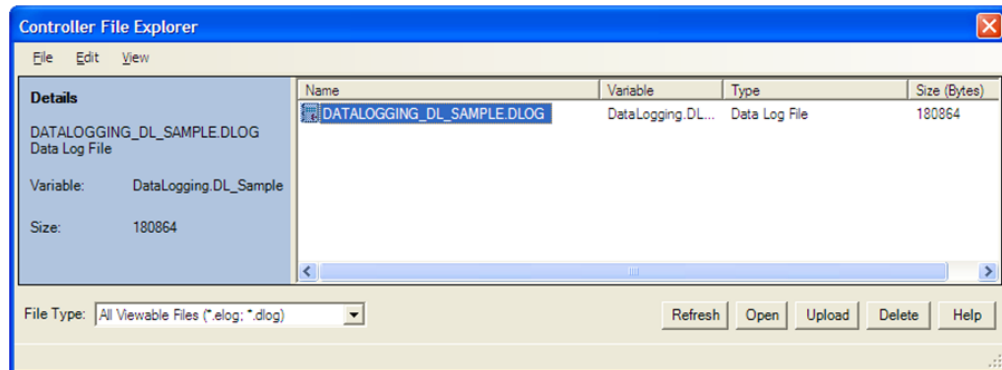


If you select the Online Commands > Explore Controller... menu item, the Controller File Explorer dialog is presented.



## Controller File Explorer

Figure 101: Manipulate Files via Controller File Explorer



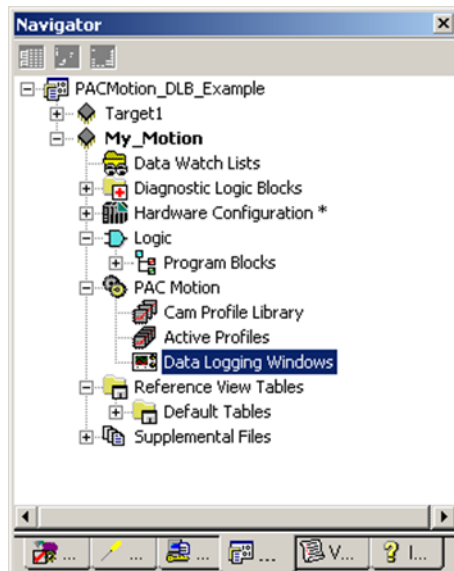
You can open, upload, or delete files from the list of dlog files that exist on the controller. The list shows all the dlog files that are available. To update the list of files, click the Refresh button. To create a Data Logging Session and upload the DLOG file to the project, click the Open button. To upload files to a PC directory in CSV format, click the Upload button. In this example, the resulting file in this example would be named DL\_SAMPLE.CSV.

## Displaying Data Logging (DLOG) Files

Once files have been uploaded from the controller, the logged data may be displayed. The DLOG information for the PMM is managed for the controller target in the Project Navigator using the Data Logger Window node highlighted below.

## Data Logging Windows Node

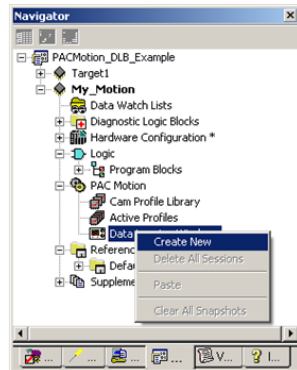
Figure 102: Displaying Data Logging Windows Node



## Managing and Viewing the Data Logging Information

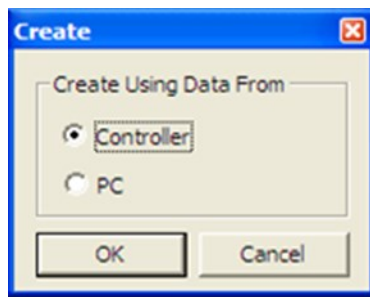
1. Create a new Data Logging Window Session for a DLOG File using the Create New session menu item –or- Select the Open button from the Explore Controller dialog:

Figure 103: Create New Session (Optional)



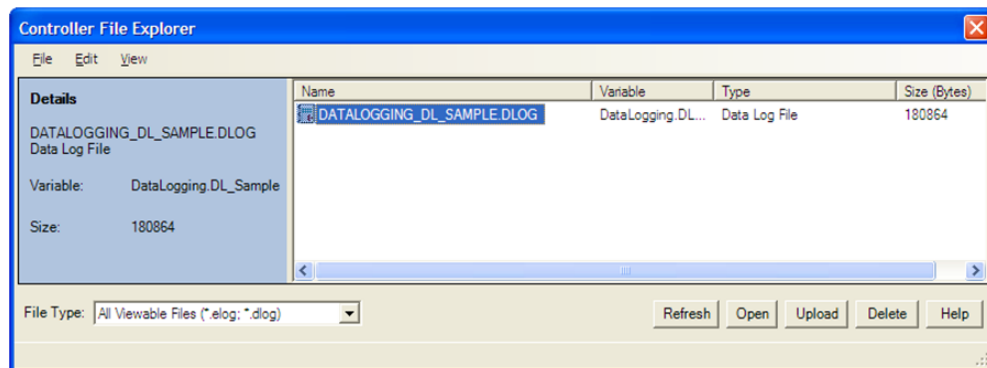
2. If using the Create New menu, specify the Data Logging data source.

Figure 104: Specify the Data Logging Data Source for Create New



3. Select the DLOG file that you want to view and select Open.

Figure 105: Select the DLOG file



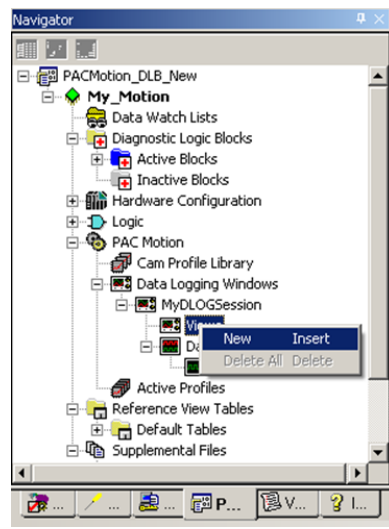
4. The session node now has two child nodes, one to manage the Views of the Data and a second to manage the Data Sources.

**Figure 106: Two Child Nodes of Session Node**



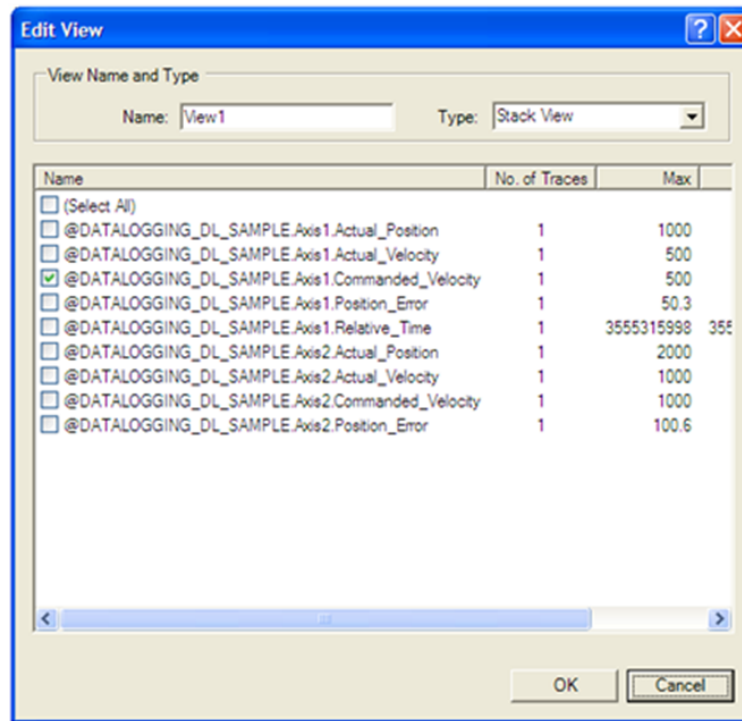
5. To create a View to display the data logging information, select the Views node and right-click the New menu item.

**Figure 107: Create a View to Display the Data Logging Information**



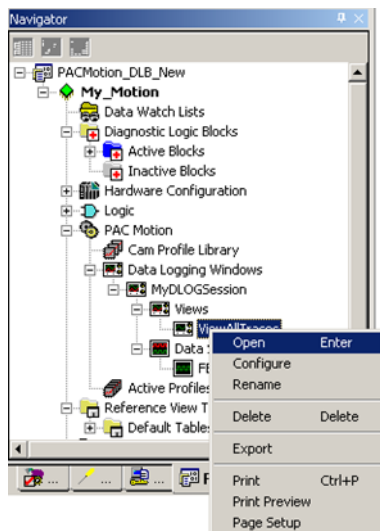
6. The Edit View dialog lets you define the Name of the View (View1 is the default name), select which signals to trace, and select the general display Type (Stack View or Tab View).

Figure 108: Edit View Dialog Box



- Once the View has been defined, you can display the view using the Open menu, or by pressing the Enter key:

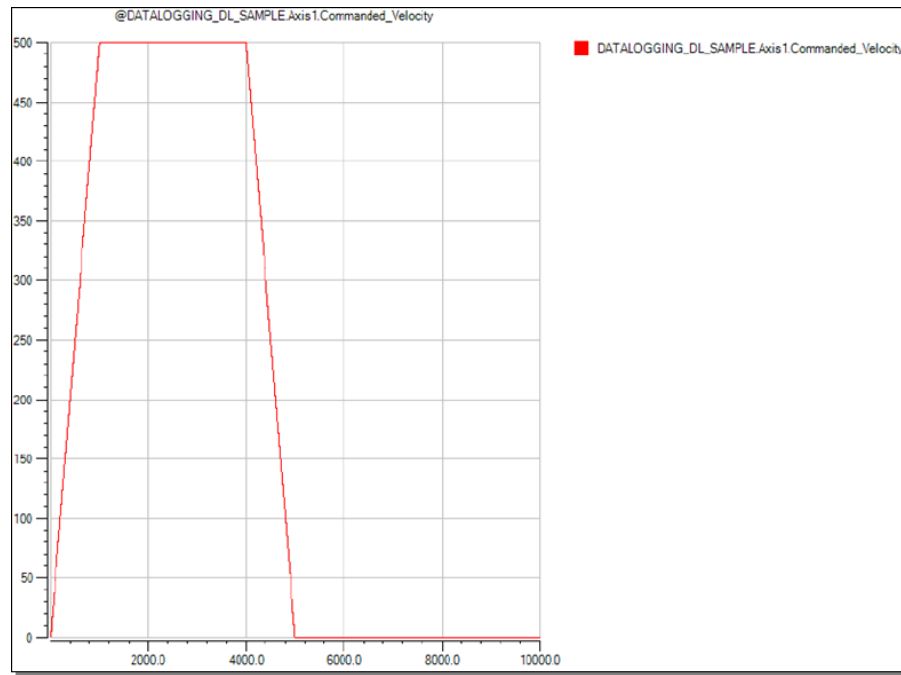
Figure 109: Display the View



- An example of a data logging session view is shown below. Once you have created these data logging sessions, they are saved in your project.

## Sample Data Logging Window View with One Trace

Figure 110: Sample Data Logging Window View with One Trace

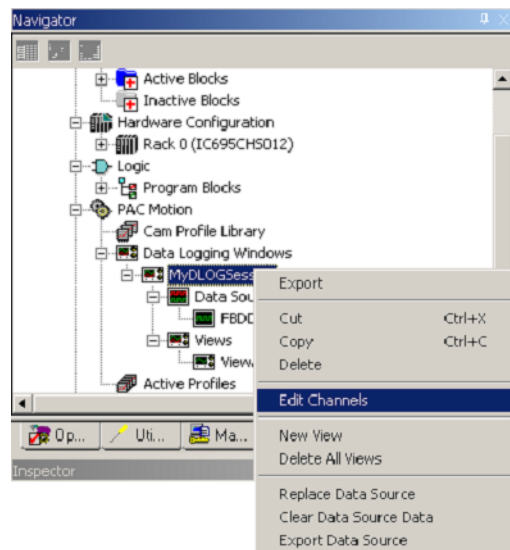


## Defining Data Logging Channels

A Channel is a grouping of traces that you want to display on the same time and value scale. This provides the ability to compare trace values in the same display.

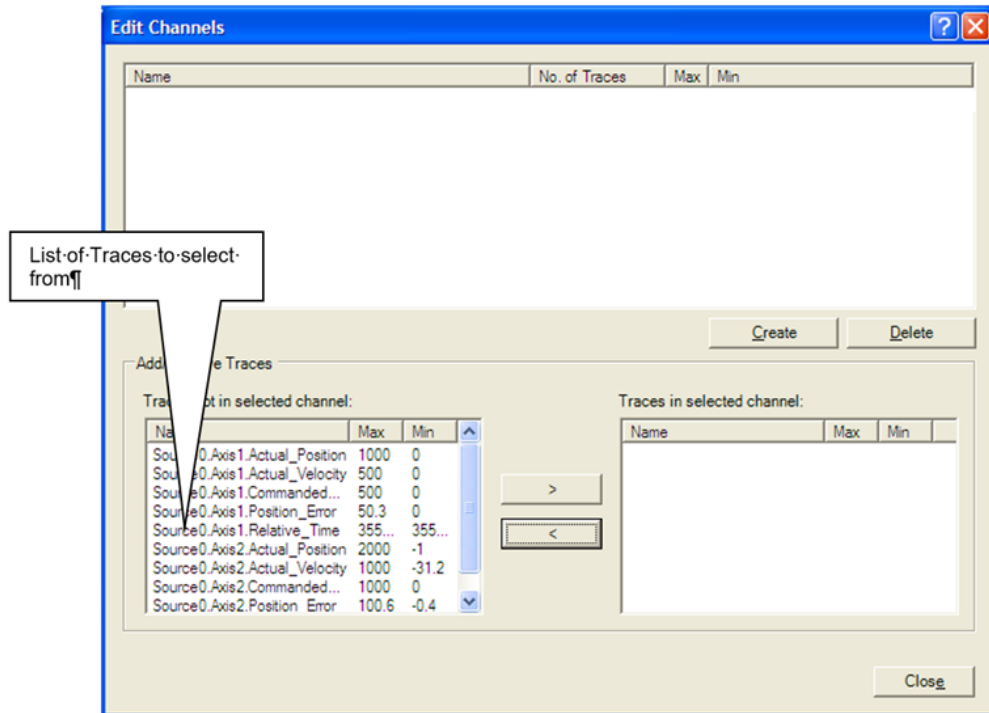
To create a Channel, select a Session node in the Navigator, then select the Edit Channels menu item.

Figure 111: Edit Channels while Session Node is Selected



The **Edit Channels** dialog is displayed:

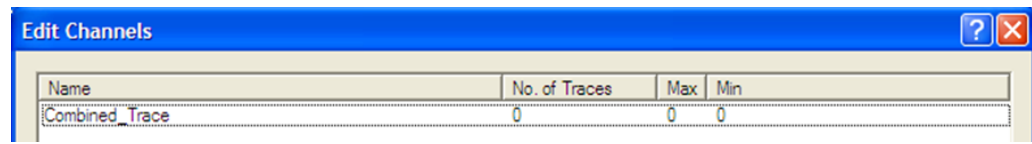
**Figure 112: Edit Channels Dialog Box**



To define a Channel, click the **Create** button (Figure 112).

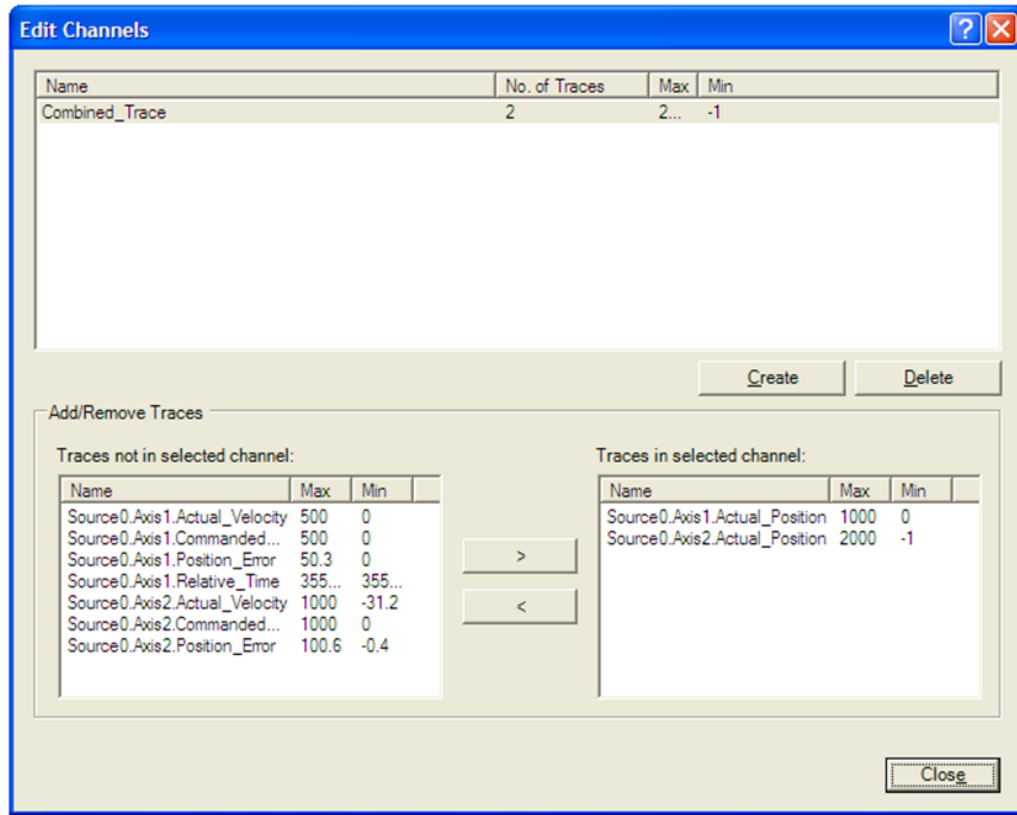
Give the new Channel a descriptive name, such as `Combined_Trace`.

**Figure 113: Assign Descriptive Name to Channel**



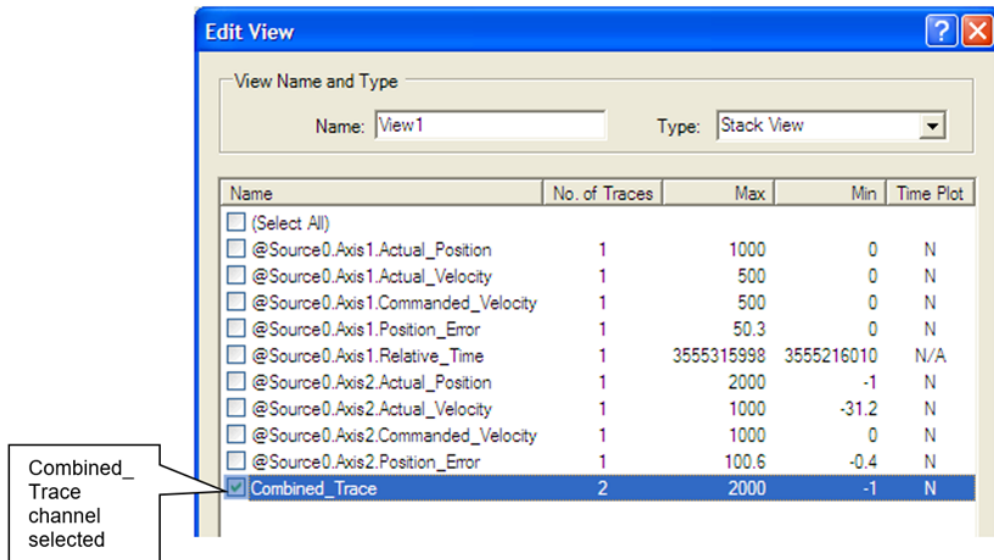
Next, select the Traces that define this Channel using the **Add/Remove Traces** options at the bottom of the dialog. Select the Traces on the left and move them to the right.

Figure 114: Add/Remove Traces to Define Channel



Now, when you create a View you can select the Combined\_Trace Channel:

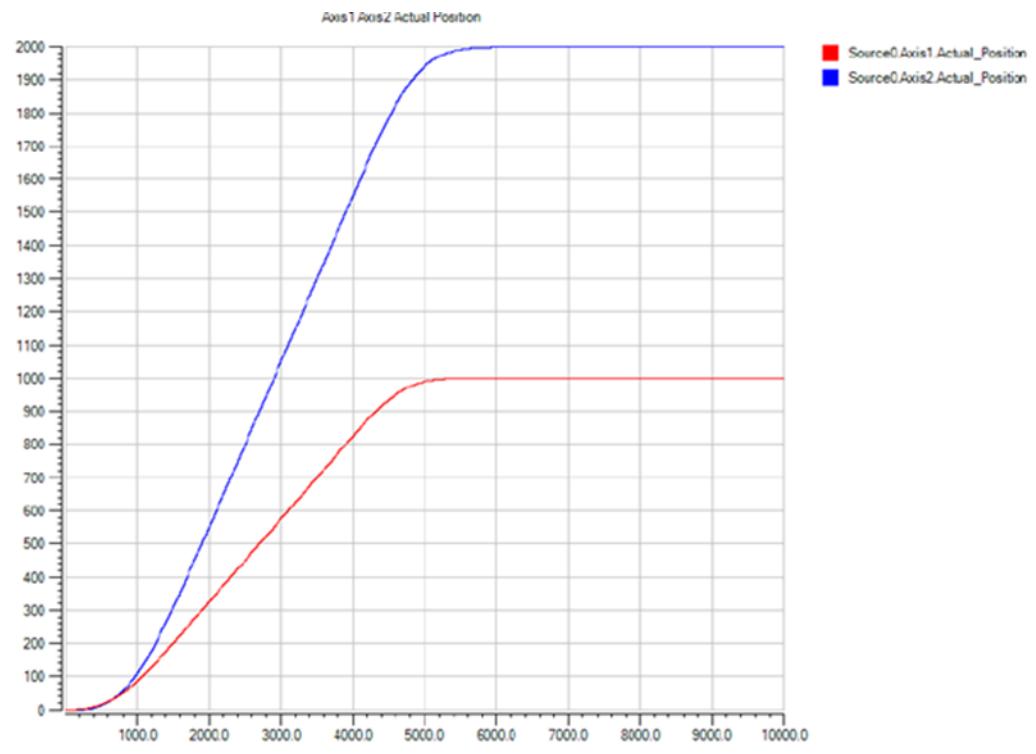
Figure 115: Select the Combined\_Trace Channel



This will result in a display with combined traces, similar to the following sample trace.

## Channel View with Two Traces

Figure 116: Channel View with Combined Traces



The data logging view offers the following display options including the following, which can be selected by right clicking in the data logger view window:

- Mouse Modes
  - Select – for selecting and scrolling the trace display
  - Zoom-box - for selecting and zooming into a portion of the trace display
  - Data cursor – for viewing data points along the trace display
- Display Axis Modes
  - Scroll – for scrolling in the X and Y axes
  - Zoom– for zooming in the X and Y axes
- Turn Legend On/Off
- Zoom In/Out/To Fit

The Data Logging session nodes also support the ability to:

- Create and manage Channels, where a channel is a collection of signal traces
- Import and Export Data Sources
- Print Preview, Print, and Page Setup



## 6.14 MC\_GearIn

LD	FBD	ST
		<p>Formal convention: [instance name](Master := [input], Slave := [input], Execute := [input], RatioNumerator := 1, RatioDenominator := 1, Acceleration := [input], Deceleration := [input], BufferMode := [input], PositionSource := [input], InGear =&gt; [output], Busy =&gt; [output], Active =&gt; [output], CommandAborted =&gt; [output], Warning =&gt; [output], Error =&gt; [output], ErrorID =&gt; [output]);</p>

The MC\_GearIn function block is used to command the slave axis velocity at a ratio of the master axis velocity. When the slave axis is in gear, position locking is used in addition to velocity locking. PACMotion uses a ramping function to synchronize the slave to a moving master. When the function block is executed, the Slave axis begins accelerating or decelerating to the gear velocity.

$$\text{Gear velocity} = \text{RatioNumerator} / \text{RatioDenominator} \times \text{Master velocity}$$

The Acceleration and Deceleration specified on the function block are used to command the slave axis only during ramping. After the slave axis reaches the gear velocity, the *InGear* output is set true. If the master velocity changes, the Slave axis velocity follows it, within the constraints of the acceleration, deceleration, and velocity application limits specified for the slave axis. The slave axis continues to follow the gear velocity until the MC\_GearOut function block disengages the slave axis, an error occurs, or the command is aborted.

MC\_GearIn does not support the buffering of a command after it. Function blocks executed on the Slave axis with buffer modes other than Aborting will be rejected.

The Virtual Axis (Axis 5) can be used as a Master input, but not as a Slave input to this function block.

Execution type: Immediate execution/deferred response

**Note:** If the master experiences an error and goes to the ErrorStop state, the slave will continue to follow the master to zero velocity. If the master becomes inaccessible (for example, the master module is swapped out), the slave will go to the ErrorStop state after stopping the slave axis.

## Operands

Instance Variable	Description	Allowed Data Types	Initial Value
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as ???? in LD.)	MC_GEARIN	NA
Parameter	Description	Allowed Data Types	Initial Value
Input_Output Parameters			
Master	Reference to master axis.	AXIS_REF	N/A
Slave	Reference to slave axis. Virtual Axis (Axis 5) not supported.	AXIS_REF	N/A
Inputs			
Execute	Start the gearing process at the rising edge.	LD: flow Other languages: all except constants	0
RatioNumerator	Gear ratio numerator for calculating gear velocity.	INT	1
RatioDenominator	Gear ratio denominator for calculating gear velocity.	INT	1
Acceleration	(UU/second <sup>2</sup> ). The acceleration rate for gearing in when the energy of the motor is increasing. Depending on the configured maximum Jerk and the master axis velocity, Acceleration is not necessarily reached. (This value is always positive.)	LREAL	0
Deceleration	(UU/second <sup>2</sup> ). The deceleration rate for gearing in when the energy of the motor is decreasing. Depending on the configured maximum Jerk and the master axis velocity, Deceleration is not necessarily reached. (This value is always positive.)	LREAL	0
BufferMode	Defines the behavior of the axis: modes are Aborting, Buffered. (Blending not allowed.)	MC_BufferMode	0
PositionSource	Defines the source on the master to follow. Actual position: Source is the configured feedback device. Commanded position: Source is the instantaneous position generated by the PMM's internal path generator. If Actual Position is selected and Axis 5 is the master, Axis 5 must use an External Encoder.	MC_PositionSource	0

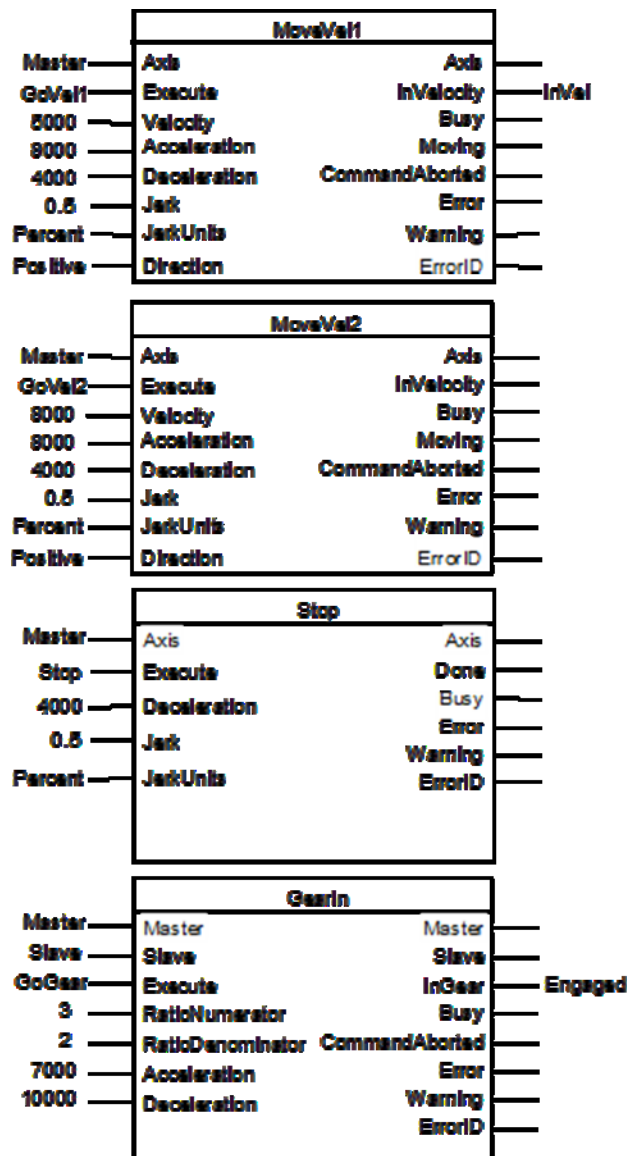
Instance Variable	Description	Allowed Data Types	Initial Value
<b>Outputs</b>			
InGear	Slave axis has reached the gear velocity.	BOOL	0
Busy	Indicates the function block has been executed and has not yet completed its action.	LD: flow Other languages: all except constants	0
Active	Indicates that the function block has control of the axis.		0
CommandAborted	Command is aborted by another command.		0
Warning	Signals that a warning has occurred within the function block.		0
Error	Signals that an error has occurred within the function block.		0
ErrorID	Indicates the type of error or warning.	WORD	0

## 6.14.1 MC\_GearIn Example

In this example, two MC\_MoveVelocity function blocks are executed on a Master axis. MC\_GearIn is executed and the Slave axis ramps up to the velocity of the Master axis and follows it with a 3/2 gear ratio. The GoGear input starts the execution of the MC\_GearIn and the Engaged output indicates when the Slave is in gear.

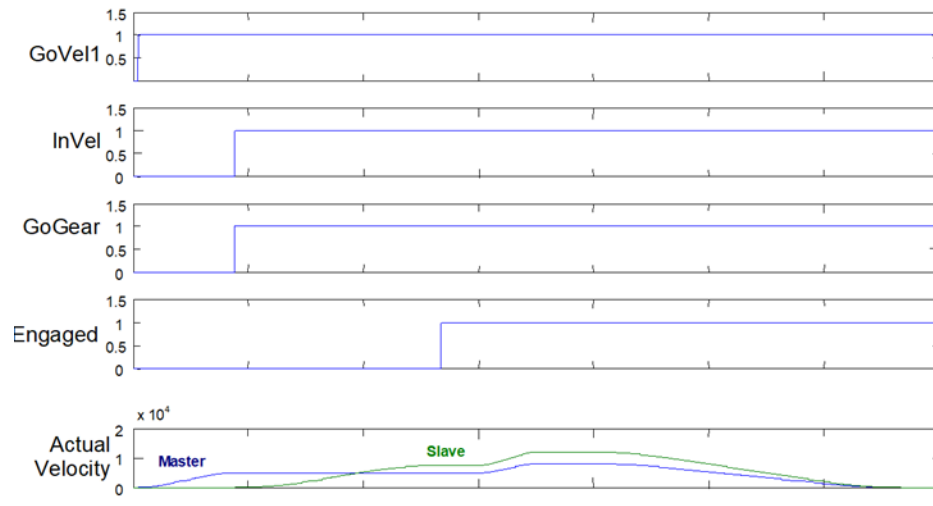
### MC\_GearIn Example Block Diagram

Figure 117: MC\_GearIn Example Block Diagram



## MC\_GearIn Example Timing Diagram

Figure 118: MC\_GearIn Example Timing Diagram



## 6.15 MC\_GearInPos

LD	FBD	ST
		<p>Formal convention:  [instance name](Master := [input], Slave := [input], Execute := [input], RatioNumerator := 1, RatioDenominator := 1, MasterSyncPosition := [input], SlaveSyncPosition := [input], SyncMode := [input], MasterStartDistance := [input], Velocity := [input], Acceleration := [input], Deceleration := [input], BufferMode := [input], PositionSource := [input], StartSync =&gt; [output], InSync =&gt; [output], Busy =&gt; [output], Active =&gt; [output], CommandAborted =&gt; [output], Warning =&gt; [output], Error =&gt; [output], ErrorID =&gt; [output]);</p>

The MC\_GearInPos function block synchronizes velocity and position of a slave axis to a master axis.

To synchronize the master and slave axes, the slave axis begins moving so that it arrives at the SlaveSyncPosition when the master axis arrives at the MasterSyncPosition. This ramping process begins when the master axis crosses the Start Position in the *correct* direction. The correct direction is defined as the sign of the MasterStartDistance.

$$\text{Start Position} = \text{MasterSyncPosition} - \text{MasterStartDistance}.$$

If MasterStartDistance is 0, the master and slave axes will begin ramping immediately when the *Execute* input transitions ON.

When the slave axis arrives at the SlaveSyncPosition it will be traveling at a velocity and direction that is synchronized with the master axis and the InSync output will be on. Once the master and slave are synchronized, the slave is position and velocity locked to the master.

The slave axis continues to follow the master's gear position and velocity until the MC\_GearOut function block disengages the slave axis, an error occurs, or the command is aborted.

For additional information about Ramping refer to Section 5.6 Synchronized Motion.

Changes to the Low Limit, Range or End of Travel configured for the master axis (via an MC\_WriteParameter) are not allowed while an MC\_GearInPos is engaged or pending on that axis.

When the master axis goes to zero velocity, the slave also stops. If the slave is tracking a slow-moving master that reaches zero velocity while its status indicates it is still moving, the slave may cease tracking the master and go to zero velocity immediately. To minimize this behavior, the Master Axis Velocity Filter (PN 1321) should be increased.

The Virtual Axis (Axis 5) can be used as a Master input, but not as a Slave input to this function block.

Execution type: Immediate execution/deferred response

---

**Note:** *If the master experiences an error and goes to the ErrorStop state, the slave will continue to follow the master to zero velocity. If the master becomes inaccessible (for example, the master module is swapped out), the slave will stop the slave axis motion and go to the ErrorStop state.*

---

## Operands

Instance Variable	Description	Allowed Data Types	Initial Value
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as ??? in LD.)	MC_GEARINPOS	NA
Parameter	Description	Allowed Data Types	Initial Value
Input_Output Parameters			
Master	Reference to master axis.	AXIS_REF	N/A
Slave	Reference to slave axis. Virtual Axis (Axis 5) not supported.		N/A
Inputs			
Execute	Starts the gearing process at the rising edge.	LD: flow Other languages: all except constants	0

Instance Variable	Description	Allowed Data Types	Initial Value
RatioNumerator	Gear ratio numerator.	INT	1
RatioDenominator	Gear ratio denominator.	INT	1
MasterSyncPosition	Master position at which the axes are running in sync.	LREAL	0
SlaveSyncPosition	Slave position at which the axes are running in sync.	LREAL	0
SyncMode	Selects the type of synchronization: NoBackupAllowed BackupAllowed Refer to *Sync Mode Recommendation below.	MC_SyncMode	N/A
MasterStartDistance	Master distance for gear in procedure (when the Slave axis motion is started to get into synchronization). This is a fixed distance the master will always cover during the ramp. If MasterStartDistance is 0, the master and slave axes will begin ramping immediately when the Execute input transitions ON.	LREAL	0
Velocity	Maximum velocity during the time difference between the start of synchronization (StartSync) and when the slave axis is synchronized (InSync).	LREAL	0
Acceleration	Maximum Acceleration during the time difference between StartSync and InSync.  <hr/> <b>Note:</b> <i>The Acceleration and Deceleration inputs are used only for getting the slave axis synchronized. Once the slave is synchronized, it is limited only by the application velocity limit of the slave axis</i> <hr/>	LREAL	0
Deceleration	Maximum Deceleration during the time difference StartSync and InSync.	LREAL	0



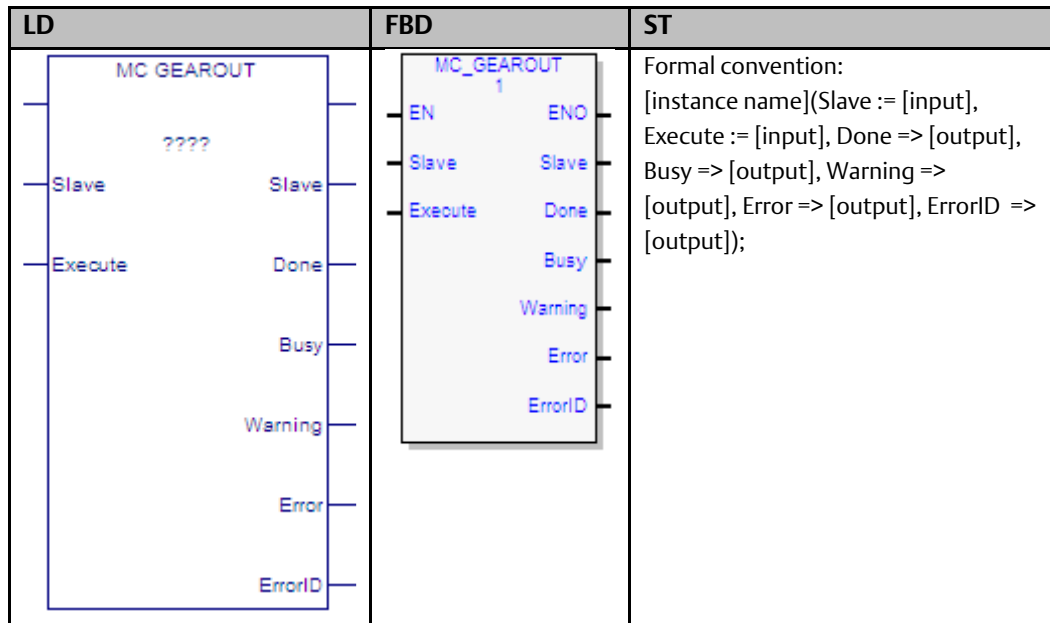
Instance Variable	Description	Allowed Data Types	Initial Value
BufferMode	Specifies the behavior of the axis. Modes are Aborting and Buffered. Blending is not allowed. If the BufferMode is Aborting, any active command remains active until the MC_GearInPos is able to start execution (the master is within the specified MasterStartDistance and the StartSync output of the MC_GearInPos is ON). Other buffer modes are processed normally: the MC_GearInPos will not begin processing until the active command is Done.	MC_BufferMode	0
PositionSource	Specifies the source on the master to follow. Actual position: Source is the configured feedback device. Commanded position: Source is the instantaneous position generated by the PMM's internal path generator. If Actual Position is selected and Axis 5 is the master, Axis 5 must use an External Encoder.	MC_PositionSource	0
<b>Outputs</b>			
StartSync	Commanded gearing started. Slave axis is Ramping.	BOOL	0
InSync	Commanded gearing reached. Slave axis is Synchronized.	BOOL	0
Busy	Indicates the function block has been executed and has not yet completed its action.	LD: flow Other languages: all except constants	0
Active	The function block has control of the axis.		0
CommandAborted	Command is aborted by another command.		0
Warning	A warning has occurred within the function block.		0
Error	An error has occurred within the function block.		0
ErrorID	Indicates the type of error or warning.	WORD	0

## Sync Mode Recommendation

During ramping, the Slave attempts to minimize the acceleration it must undergo. If backup is allowed (SyncMode is set to BackupAllowed) a consequence of this is that the ramp may move the slave backward, away from the synchronization position, or past the synchronization position so that it can be moving in the opposite direction when it does synchronize.

In some situations, for example when the master reverses direction while ramping with Backup Allowed, the slave may exhibit unexpected behavior. To prevent this behavior, SyncMode should be set to NoBackupAllowed.

## 6.16 MC\_GearOut



The MC\_GearOut function block is used to disengage from an MC\_GearIn or MC\_GearInPos function block. When executed the slave axis disengages from the MC\_GearIn or MC\_GearInPos and continues to be commanded at the last commanded velocity.

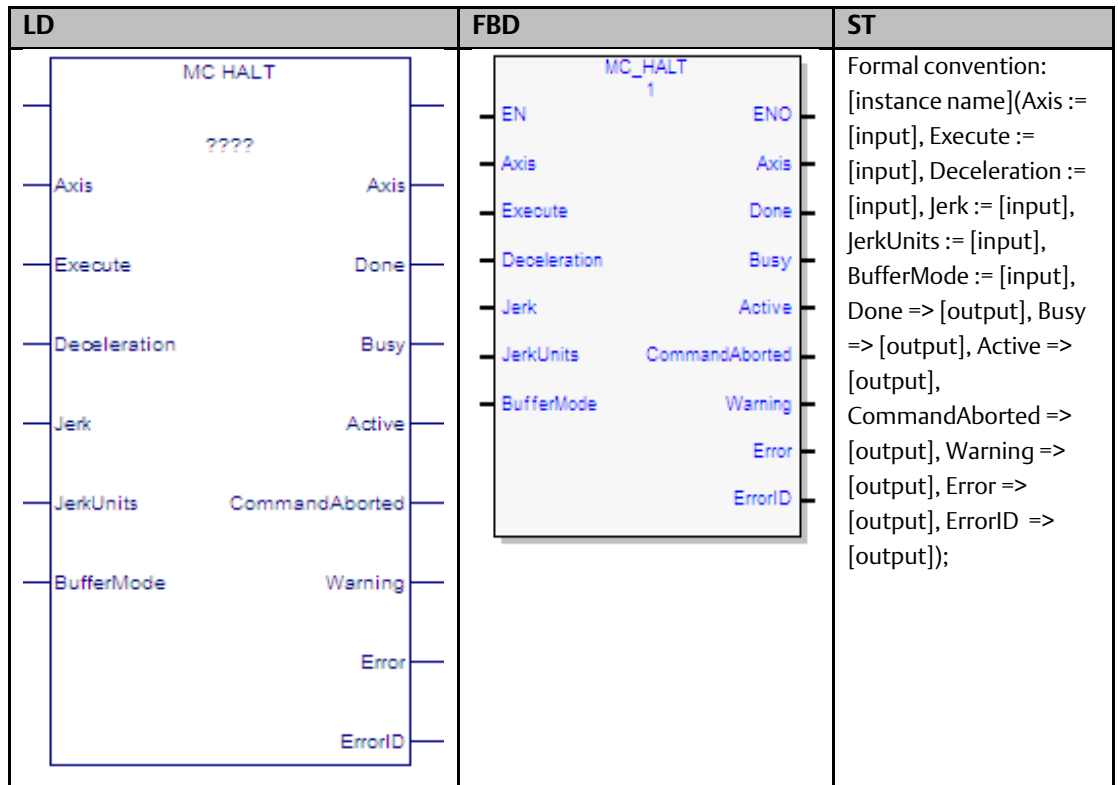
If the slave axis had a commanded acceleration, the axis will use its acceleration and deceleration application limits to achieve the last command velocity. This command is usually followed by another command.

Execution type: Immediate execution/deferred response

## Operands

Instance Variable	Description	Allowed Data Types	Initial Value
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as ???? in LD.)	MC_GEAROUT	NA
Parameter	Description	Allowed Data Types	Initial Value
<b>Input_Output Parameters</b>			
Slave	Reference to slave axis. Virtual Axis (Axis 5) not supported.	AXIS_REF	N/A
<b>Inputs</b>			
Execute	Start to disengage the slave from the master.	LD: flow Other languages: all except constants	0
<b>Outputs</b>			
Done	Disengaging completed.	LD: flow	0
Busy	Indicates the function block has been executed and has not yet completed its action.	Other languages: all except constants	0
Warning	Signals that warning has occurred within Function block.		0
Error	Signals that error has occurred within Function block.		0
ErrorID	Indicates the type of error or warning.	WORD	0

## 6.17 MC\_Halt



This function block commands a controlled motion stop using programmed Deceleration and Jerk values and aborts any ongoing function block execution. The axis transitions to the state DiscreteMotion, until the velocity is zero. With the Done output set, the state transitions to Standstill.

It is important to specify a deceleration rate that results in a satisfactory stopping distance.

MC\_Halt differs from MC\_Stop in that it does not transition the axis to the Stopping state. When MC\_Halt is executed, the axis transitions to the DiscreteMotion state while the axis decelerates and when the Done output is set, transitions to the Standstill state.

Execution type: Immediate execution/deferred response.

## Operands

Instance Variable	Description	Allowed Data Types	Initial Value
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as ??? in LD.)	MC_HALT	NA
Parameter	Description	Allowed Data Types	Initial Value
Input_Output Parameters			
Axis	Axis that receives function block command.	AXIS_REF	N/A
Inputs			
Execute	Start the halt at rising edge	LD: flow Other languages: all except constants	0
Deceleration	The maximum move deceleration rate (energy is decreasing). Maximum deceleration is not necessarily reached. (Always positive.) [Units = UU/second <sup>2</sup> ]	LREAL	0
Jerk	The Jerk (rate of change in acceleration) used for the move. (Always positive.) If set to 0, jerk will be unlimited. [Units = UU/sec <sup>3</sup> or %]	LREAL <a href="#">For details, refer to Jerk and JerkUnits in Section 5.3.3.</a>	0
JerkUnits	Selects units for Jerk input: UU/sec <sup>3</sup> or %	MC_JERKUNITS	0
BufferMode	Defines the axis buffering behavior. Valid modes are Aborting, Buffered, and Blending.	MC_BufferMode	0
Outputs			
Done	Zero velocity reached.	LD: flow	0
Busy	Indicates the function block has been executed on an axis and has not yet completed its action.	Other languages: all except constants	1
Active	Indicates that the block has control of the axis.		0
CommandAborted	Command is aborted by another command.		0
Warning	Indicates that a warning has occurred within the function block.		0
Error	Indicates that an error has occurred within the function block.		0
ErrorID	Error or warning identification.	WORD	0

## 6.18 MC\_Home

LD	FBD	ST
<p>MC HOME</p> <p>????</p> <p>Axis</p> <p>Execute</p> <p>Position</p> <p>HomingMode</p> <p>BufferMode</p> <p>HomeOffset</p> <p>FinalHomeVelocit</p> <p>FindHomeVelocity</p> <p>Acceleration</p> <p>Deceleration</p> <p>Jerk</p> <p>JerkUnits</p>	<p>MC_HOME 1</p> <p>EN</p> <p>Axis</p> <p>Execute</p> <p>Position</p> <p>HomingMode</p> <p>BufferMode</p> <p>HomeOffset</p> <p>FinalHomeVelocity</p> <p>FindHomeVelocity</p> <p>Acceleration</p> <p>Deceleration</p> <p>Jerk</p> <p>JerkUnits</p> <p>ENO</p> <p>Axis</p> <p>Done</p> <p>Busy</p> <p>Active</p> <p>CommandAborted</p> <p>Warning</p> <p>Error</p> <p>ErrorID</p>	<p>Formal convention:</p> <p>[instance name](Axis := [input], Execute := [input], Position := [input], HomingMode := [input], BufferMode := [input], HomeOffset := [input], FinalHomeVelocity := [input], FindHomeVelocity := [input], Acceleration := [input], Deceleration := [input], Jerk := [input], JerkUnits := [input], Done =&gt; [output], Busy =&gt; [output], Active =&gt; [output], CommandAborted =&gt; [output], Warning =&gt; [output], Error =&gt; [output], ErrorID =&gt; [output]);</p>

The Find Home sequence is used to establish a valid actual position relative to a known reference point. The configured Home Offset defines the location of Home Position as the offset distance from the Home Marker.

### 6.18.1 Checklist for Find Home Sequence

- The axis must be powered on during an entire home cycle.
- When the Execute input transitions ON, the PositionValid axis status bit is turned OFF until the end of the home cycle.
- If an MC\_Stop function block halts a home cycle, the PositionValid bit does not turn back ON.

**Note:** No motion function blocks that cause motion can be executed unless the PositionValid bit is ON with the exception of MC\_JogAxis. The Position input sets

the absolute position when a reference signal is detected. The configured Home Offset defines the location of Home Position as the offset distance from the Home Marker.

## 6.18.2 Overview of Find Home Sequence

The function block completes in the Standstill state. Issuing the MC\_Home command in any state other than Standstill will result in an ErrorStop.

The MC\_Home function can operate in one of four modes, selected by the HomingMode input. For details and examples of the four homing modes, refer to Section 6.18.3 Homing Modes.

**Drive Controlled: EtherCAT axes with motor feedback devices.** In this mode, the drive will perform the homing using the Drive Homing Mode set in hardware configuration. The MC\_Home inputs will be written to corresponding drive parameters. MC\_SetOverride is not available in this mode. Reference PACMotion Workbench online help and Servo Drive Manuals for additional information on supported homing modes.

**External Encoder: Limit Switch Reference Pulse (Absolute Home Switch): Axes 1 – 4.** In this mode, the configured Home Switch input is used to trigger the home cycle to look for the next Encoder Marker pulse. The next Encoder Marker pulse encountered when traveling in the negative direction sets the home position location.

**Note:** Axis 5 does not support this mode.

**External Encoder: Move+ (RefPulse).** In this mode, the first Encoder Marker pulse encountered when moving in the positive direction after the Find Home command is given is used to establish the exact location.

**External Encoder: Move – (RefPulseNeg).** In this mode, the first Encoder Marker pulse encountered when moving in the negative direction after the Find Home command is given is used to establish the exact location.

Axis 5 provides a Virtual Path Planner and interface to an External Device. Move+ and Move- are supported on Axis 5 only when configured with an External Device. In this case the application must provide the means to rotate the External Device. The home position will be set when the first Encoder Marker pulse is encountered; regardless of the direction in which the axis is moving. If the Axis 5 Virtual Path Planner is used in the application, its position must be set independently, using MC\_SetPosition.

Execution type: Immediate execution/deferred response.



## Operands

Instance Variable	Description	Allowed Data Types	Initial Value
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as ??? in LD.)	MC_HOME	NA
Parameter	Description	Allowed Data Types	Initial Value
<b>Input_Output Parameters</b>			
Axis	Axis that receives function block command.	AXIS_REF	N/A
<b>Inputs</b>			
Execute	Starts motion at rising edge.	LD: flow Other languages: all except constants	0
Position	The value assigned to Commanded Position when a Find Home cycle completes. [Units = UU]	LREAL	0
HomingMode	The method used to find home during a Find Home cycle. Choices are: DriveControl – Axes 1-4 only LimitSwitchRefPulse – Axes 1–4 only RefPulse and RefPulseNeg – All axes	MC_HomingMode For details, refer to Section 6.18.3 Homing Modes	0
BufferMode	Defines the behavior of the axis. Allowed modes are Aborting, Buffered, and Blending.	MC_BUFFERMODE	0
HomeOffset	Value added to or subtracted from the servo's final stopping point when a Find Home cycle completes. Home Offset adjusts the final servo stopping point relative to the encoder marker. [Units = UU]	LREAL	0
FinalHomeVelocity	The maximum velocity move seeks the final Home Switch transition and Encoder Marker pulse at the end of a Find Home cycle. This value must be slow enough to allow a delay between the final Home Switch transition and the Encoder Marker pulse. (Always positive.) [Units = UU/sec]	LREAL	500
FindHomeVelocity	The maximum velocity at which the servo seeks the initial Home Switch transitions during the Find Home cycle when the Home Mode is configured for HOMESW. Also used for the move to the position determined by HomeOffset. If desired, Find Home Velocity can be set to a high value to allow the servo to quickly locate the Home Switch (Always positive.) [Units = UU/sec]	LREAL	2000
Acceleration	The maximum acceleration rate used in the find home sequence (energy is increasing). Maximum acceleration is not necessarily reached. (Always positive) [Units = UU/sec <sup>2</sup> ]	LREAL	100,000

Instance Variable	Description	Allowed Data Types	Initial Value
Deceleration	The maximum deceleration rate used in the find home sequence (energy is decreasing). Maximum deceleration is not necessarily reached. (Always positive) [Units = UU/sec <sup>2</sup> ]	LREAL	100,000
Jerk	The Jerk (rate of change in acceleration) used for the move. (Always positive.) If set to 0, jerk will be unlimited. [Units = UU/sec <sup>3</sup> or %] Note: Not used in DriveControlled HomingMode	LREAL For details, refer to Jerk and JerkUnits in Section 5.3.3.	0
JerkUnits	Selects units for Jerk input: UU/sec <sup>3</sup> or % Note: Not used in DriveControlled HomingMode	MC_JERKUNITS	0
Outputs			
Done	Indicates the Commanded Position has reached the Home Position.	LD: flow Other languages: all except constants	0
Busy	Indicates the function block has been executed on an axis and has not yet completed its action.		1
Active	Indicates that the function block has control of the axis.		0
CommandAborted	Command is aborted by another command.		0
Warning	Indicates that a warning has occurred within the function block.		0
Error	Indicates that an error has occurred within the function block.		0
ErrorID	Error or warning identification.	WORD	0

**Note:** The MC\_Home function block sets its Done output based on Commanded Position. The Actual Position will be different from the Home Position by an amount equal to the Position Error. The InZone axis status flag can be monitored to determine whether the Actual Position is within the In Position Zone relative to the Home Position.

If your application requires axis homing based on Actual Position, the state of the InZone flag can be ANDed with the MC\_Home Done output.

## 6.18.3 Homing Modes

### Drive Controlled (EtherCAT Axes with Motor Feedback Devices only)

In this mode, the EtherCAT drive will perform the homing operation using the Drive Homing Mode selection. Drive Homing Mode can be set in the PMM345 hardware configuration by configuring the Drive Homing Mode and Drive Homing Direction parameters on the Axis (1-4) tab. These parameters will be written to drive parameters HOME.MODE and HOME.DIR each time MC\_Home is executed. If the Drive Homing Mode parameter is set to Disabled, then Drive Controlled homing using MC\_Home is not allowed and homing will need to be performed using PACMotion Workbench.

When MC\_Home is executed the drive will switch to Position Mode and control the motion using parameters input to the MC\_Home function block as well as parameters optionally set on the drive. For this reason, it is recommended that one perform a tuning sequence for the drive position loop.

Restrictions on input parameters in this mode:

- Jerk and JerkUnits are not used in this mode; any values set will be ignored.
- FinalHomeVelocity is only used with Homing Modes that move to an index. Feedback devices that support homing to zero angle can perform the entire homing operation at the FindHomeVelocity as the feedback device knows the location of zero angle and is not searching for it as required for an index.

Restrictions on operations with this mode:

- MC\_SetOverride is not available during this mode.

### Parameters Written to Drive

MC_Home Parameter	Drive Parameter	EtherCAT CoE Index	EtherCAT CoE Subindex
Position	HOME.P	0x607C	0
HomeOffset	Home.DIST	0x3484	0
FinalHomeVelocity	HOME.FEEDRATE	0x6099	2
FindHomeVelocity	HOME.V	0x6099	1
Acceleration	HOME.ACC	0x609A	0
Deceleration	HOME.DEC	0x3524	0

Additional homing parameters can be set on the drive.

When homing to a hard stop HOME.IPEAK and HOME.IPEAKACTIVE can be used to limit the current and HOME.PERRTHRESH can be used to set the amount of position error to detect that a hard stop has been reached.

## Drive Based Homing to HW EOTs

### Home Offset – Typical Usage

The Home Offset parameter specifies how far away from the HW EOT to establish home position. The Home Offset is a signed number typically opposite of the HW EOT utilized to establish home position. For example, the Home Offset for Positive HW EOT should be a negative number while the Home Offset for a Negative HW EOT should be a positive number.

It is noted that the Home Offset value must not be so large as to cause the opposite HW EOT to be triggered during the Home Offset move. If the Home Offset is large enough to trigger the opposite HW EOT then it will follow the atypical case discussed below and is not recommended.

### Home Offset – Atypical Usage (Not recommended)

The home function allows a Home Offset parameter that is in the same direction as the HW EOT with a limitation on the maximum Home Offset position. For example, the home function allows a positive Home Offset to be programmed for a positive HW EOT and vice versa for the negative HW EOT.

In this atypical usage, the maximum Home Offset is determined by the find home velocity and the find home deceleration rate. The home function automatically starts deceleration at the find home deceleration rate when a HW EOT switch is triggered. The axis stops at a position determined by the velocity and deceleration rate. The Home Offset position has no effect on this functionality.

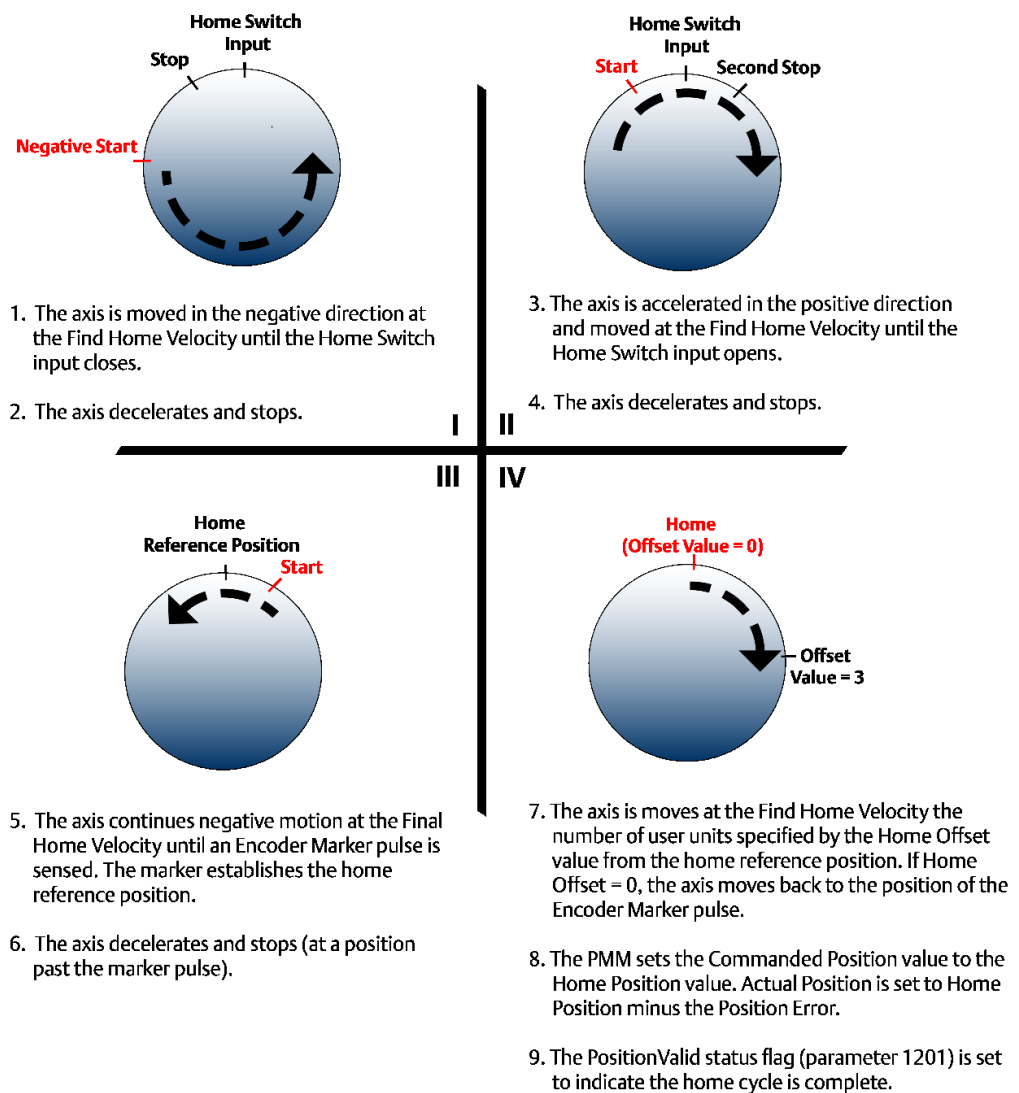
However, if the programmed Home Offset is less than the maximum, the home function will start deceleration prior to the HW EOT limit switch trigger and stop at the programmed Home Offset. If the Home Offset is greater than the maximum, the home function starts deceleration at the HW EOT limit switch and stops at the maximum value. In this case, the Home Offset is not reached due to the HW EOT switch being triggered.

## Limit Switch Reference Pulse Mode (Axes 1—4 External Device Feedback only)

This mode is also called the Absolute Home Switch mode. If the HomingMode is set to LimitSwitchRefPulse, the configured Home Switch input is used to trigger the home cycle to look for the next Encoder Marker pulse. The next Encoder Marker pulse encountered when traveling in the negative direction sets the home position location. An open Home Switch input indicates the axis is on the positive side of the home switch and a closed Home Switch input indicates the axis is on the negative side of the home switch.

An OFF to ON transition of the *Execute* input yields the following home cycle.

**Figure 119: Find Home Routine for Limit Switch Reference Mode**



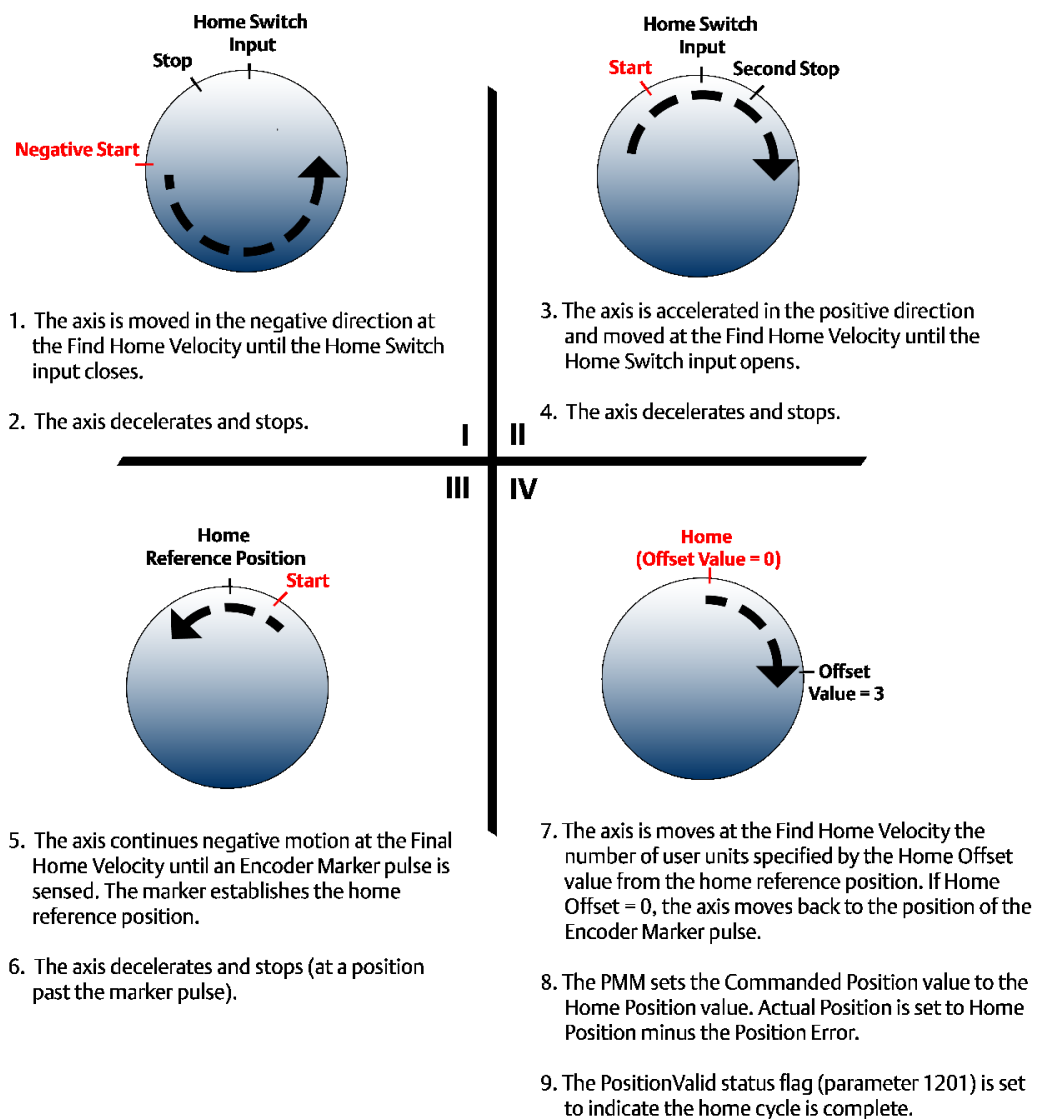
**Output:** The PositionValid status flag (parameter 1201) is set to indicate the home cycle is complete.

### Find Home Routine for Limit Switch Reference Pulse Mode

If initiated from a position on the positive side of the home switch, in which case the home switch is *open* (Logic 0), the Find Home routine starts with step 1 below.

If the MC\_Home function block is executed from a position on the negative side of the home switch, in which case the home switch is *closed* (Logic 1), the routine starts with step 3.

**Figure 120: Find Home Routine for Limit Switch Reference Mode**



## Limit Switch Reference Pulse Example

Many different home switch designs are possible. The switch may be normally open or normally closed, and may be mounted in one of several possible locations. The example given in this section illustrates a common arrangement used for linear axes. In the following picture, the home switch is a normally open proximity switch, mounted near the end of the machine slide's travel range (in the negative direction). The imaginary line that divides the home switch's positive and negative sides is the home switch's operating point, located approximately on the switch's centerline. If the machine slide travels in the negative direction far enough so that the right-hand edge of the home switch CAM causes the home switch to close, the machine slide is considered as having crossed over to the negative side of the home switch. **The home switch CAM is long enough so that while the machine slide is on the negative side of the home switch, it will keep the normally open home switch closed.**

Note the relationships of the home position, the negative over travel position, and the positive stop position. A small amount of distance is provided in the negative direction between the home position and the negative over travel position. This is to allow some working room for adjustment and setup of these positions and for the find home routine, which requires that its final move be in the negative direction.

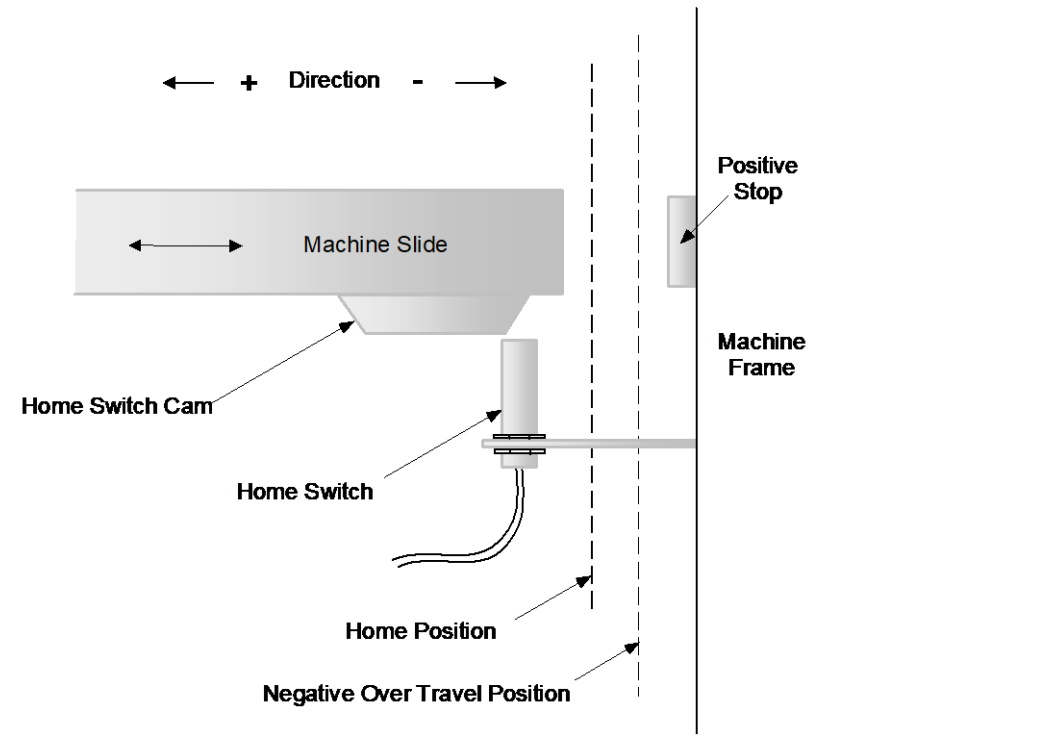
Distance is also provided between the overtravel limit position and the positive stop. Enough distance should be allowed to prevent the machine slide from hitting the positive stop. The correct distance needs to be greater than the worst-case stopping distance required by the machine slide after it reaches the overtravel limit position.

In this example, the machine slide's working range is on the positive side of the home switch. If the Position input parameter was set to 0, this would simplify programming absolute positioning commands since only positive numbers would be used.

Often, the home position needs to be set to an exact distance from a reference point on the machine. To facilitate this adjustment, the home switch CAM could be made with slotted mounting holes that would allow a coarse adjustment of the CAM to bring the calibration to within one turn of the encoder. Then, the small remaining distance would be accurately measured and the value obtained would be entered into the HomeOffset input parameter.

## Limit Switch Ref Pulse Example for Linear Axes

Figure 121: Limit Switch Reference Pulse Example for *Linear* Axes



### Reference Pulse (Move+) and Reference Pulse Negative (Move-) Modes (All Axes External Device Feedback Only)

If Find Home Mode is configured as RefPulse or RefPulseNeg, the first encoder marker pulse encountered when moving in the appropriate direction (positive for RefPulse, negative for RefPulseNeg) after the find home command is given is used to establish the exact location. In this mode, the operator usually jogs the axis to a position close (within one revolution of the encoder) to the home position first, then initiates the find home command. To assist the operator in jogging to the correct position, a set of alignment marks indicating a close proximity to the home position is sometimes placed on the machine and machine axis.



## Find Home Routine for Reference Pulse (Move +) or Reference Pulse Negative (Move –)

When the MC\_Home function block is executed with *Find Home Mode* set to *RefPulse* or *RefPulseNeg*, the following sequence of events occurs:

The axis is accelerated at the Acceleration rate and moved at the configured Final Home Velocity (positive direction for RefPulse, negative direction for RefPulseNeg) until an Encoder Marker pulse is sensed. This marker pulse establishes the home reference position.

The axis is stopped at a position past the marker pulse using the configured Deceleration.

The axis is moved, at the configured Find Home Velocity and with the configured Acceleration, the number of user units specified by the Home Offset value from the home reference position. If Home Offset = 0, the axis moves back to the position of the marker pulse.

The axis is stopped at the configured Deceleration.

The PMM sets the Commanded Position and Actual Position to the configured Home Position value. The PositionValid status flag (parameter 1201) is set ON to indicate the home cycle is complete.

## Home to a Hard Stop Using External Device Feedback

This sample procedure describes a homing operation that is performed without the use of Home Switch and Encoder Marker inputs.

1. Set the following parameter inputs before proceeding:

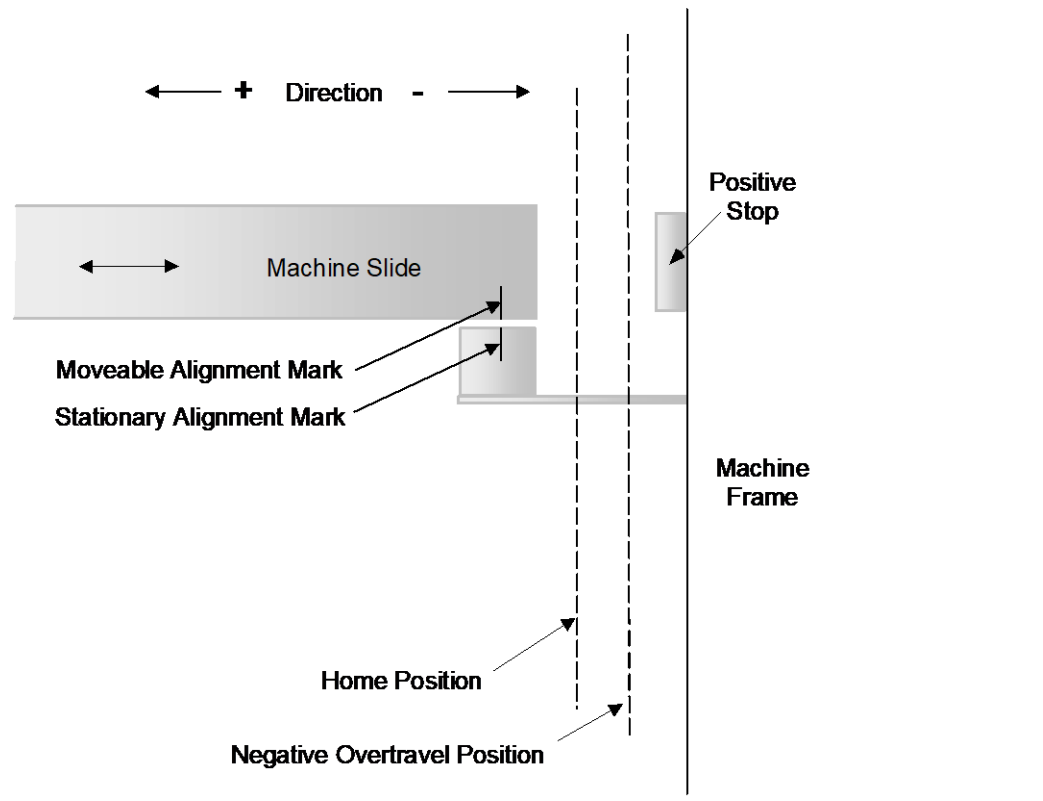
Parameter	Values
Current Limit (PN 1015)	Value not to exceed drive current that will cause damage or mechanical slippage.
Feedback Moving Deadband (PN 1024)	Value not to exceed vibration threshold from servo that could be interpreted as axis moving when encountering hard stop.
Position Lag Monitoring (PN 6)	Value not to be exceed when moving to hard stop to prevent axis from entering position error limit.
Max Position Lag (PN 7)	Value not to be exceed when moving to hard stop to prevent axis from entering position error limit.

2. Capture the starting Actual Position (PN 1300).
3. Jog to the hard stop and quit jogging when Current limit is active.
4. Execute MC\_JogAxis in the direction of the hard stop at a speed that will not damage the machine when the hard stop is encountered and with acceleration low enough that the axis will not exceed the Current Limit while accelerating.
5. Monitor the FeedbackMoving (PN 1224) and CurrentLimitActive (PN 1207) axis status flags. Stop jogging when FeedbackMoving transitions from True to False and CurrentLimitActive is True. If it is possible to home from very close to the hard stop, may need a timeout in the place of Feedback Moving.
6. Capture the current Actual Position (PN 1300).
7. Execute MC\_SetPosition to the desired home position.
8. When the MC\_SetPosition is Done and PositionValid is True, execute an MC\_MoveAbsolute to a position just off the hard stop or begin machine operation.
9. Set CurrentLimit to the desired values for normal machine operation.

### Reference Pulse Negative (Move -) Home Cycle Example

In this example, with the Homing Mode set to *RefPulseNeg*, the operator jogs the axis until the moveable mark on the machine slide lines up with the stationary mark on the alignment plate mounted to the machine frame. (Note that the marks align on the positive side of home position since the Home Position parameter is set to *RefPulseNeg*). Then the operator initiates the find home routine, which causes the axis to move in the negative direction until the marker pulse occurs.

Figure 122: Home Cycle Example (Move-)



## 6.19 MC\_JogAxis

LD	FBD	ST
<p>MC JOGAXIS</p> <p>Axis</p> <p>Enable_Positive</p> <p>Enable_Negative</p> <p>JogVelocity</p> <p>MinJogDistance</p> <p>Acceleration</p> <p>Deceleration</p> <p>Jerk</p> <p>JerkUnits</p> <p>Axis</p> <p>Busy</p> <p>Active</p> <p>CommandAborted</p> <p>Warning</p> <p>Error</p> <p>ErrorID</p>	<p>MC_JOGAXIS_1</p> <p>EN</p> <p>Axis</p> <p>Enable_Positive</p> <p>Enable_Negative</p> <p>JogVelocity</p> <p>MinJogDistance</p> <p>Acceleration</p> <p>Deceleration</p> <p>Jerk</p> <p>JerkUnits</p> <p>ENO</p> <p>Axis</p> <p>Busy</p> <p>Active</p> <p>CommandAborted</p> <p>Warning</p> <p>Error</p> <p>ErrorID</p>	<p>Formal convention:</p> <pre>[instance name](Axis := [input], Enable_Positive := [input], Enable_Negative := [input], JogVelocity := [input], MinJogDistance := [input], Acceleration := [input], Deceleration := [input], Jerk := [input], JerkUnits := [input], Busy =&gt; [output], Active =&gt; [output], CommandAborted =&gt; [output], Warning =&gt; [output], Error =&gt; [output], ErrorID =&gt; [output]);</pre>

This function block jogs an axis forward or backward at the specified velocity, acceleration and jerk. The axis override factors (see MC\_SetOverride) do not apply to axes in the Jogging state, allowing axes in feed hold to be jogged.

MC\_JogAxis is used to provide user-configurable (typically relatively slow speed) movements in the positive or negative direction. The Enable\_Positive and Enable\_Negative inputs select the direction of travel. The MC\_JogAxis instance is enabled if one but not both of the enable inputs is true.

The axis continues to jog as long as the function block instance is enabled. When the enable input goes low, the axis is decelerated until it stops.

The Busy and Active outputs are true as long as the instance is enabled and false when it is not enabled. The CommandAborted output is set true if power to the axis is turned off while the jog is active or if an MC\_Stop is issued to stop the axis.

When the enable input transitions off, all outputs of the instance are cleared.

MC\_JogAxis can be activated under limited circumstances when the axis is in the ErrorStop state. For example, if the axis is outside the software End of Travel limits, MC\_JogAxis can be used to move the axis in the direction that brings it back inside the limits. MC\_Power must be enabled for this feature to work. Once the axis is within the End Of Travel limits

and MC\_JogAxis is no longer Busy/Active, a Reset is required before any motion, including MC\_JogAxis, can be commanded. Refer to 0,

Example: Jog to Software End of Travel with Warning for a UDFB that provides alternate behavior.

MC\_JogAxis does not support buffering commands. Thus, in Jogging state, motion commands are rejected regardless of their buffer mode.

When the enable input transitions off for an instance with valid inputs, the axis is brought to zero velocity. Once the axis has successfully been stopped, all outputs of the instance are cleared.

In the case where the enable input is on and the inputs of the MC\_JogAxis are changed from valid values to invalid values, the Error and ErrorID outputs of the MC\_JogAxis will be set and the axis will be stopped and transitioned to the ErrorStop state.

Execution type: Immediate execution/deferred response.

## Operands

Instance Variable	Description	Allowed Data Types	Initial Value
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as ??? in LD.)	MC_JOGAXIS	NA
Parameter	Description	Allowed Data Types	Initial Value
Input_Output Parameters			
Axis	Axis that receives function block command ( <b>Note:</b> Make all the axis inputs use this text)	AXIS_REF	N/A
Inputs			
Enable_Positive	Generates a relative move in the positive direction when high and a stop when low.	BOOL	0
Enable_Negative	Generates a relative move in the negative direction when high and a stop when low.	BOOL	0
JogVelocity	The maximum jog velocity. Maximum velocity is not necessarily reached. (value is always positive) [Units = UU/sec]	LREAL	0
MinJogDistance	(UU) Minimum distance to jog. If the Enable_Positive or Enable_Negative input goes low, axis motion continues until it has traveled this distance. [Units = UU]	LREAL	0
Acceleration	The maximum jog acceleration rate (energy is increasing). Maximum acceleration is not necessarily reached. (Always positive)[Units = UU/sec <sup>2</sup> ]	LREAL	0

Instance Variable	Description	Allowed Data Types	Initial Value
Deceleration	The maximum jog deceleration rate (energy is decreasing). Maximum deceleration is not necessarily reached. (Always positive)[Units = UU/sec <sup>2</sup> ]	LREAL	0
Jerk	The Jerk (rate of change in acceleration) used for the move. (Always positive.) If set to 0, jerk will be unlimited. [Units = UU/sec <sup>3</sup> or %]	LREAL For details, refer to Jerk and JerkUnits in Section 5.3.3.	0
JerkUnits	Selects units for the Jerk input: UU/sec <sup>3</sup> or %	MC_JERKUNITS.	0
<b>Outputs</b>			
Busy	Indicates the function block is enabled and has not yet completed its action.	LD: flow Other languages: all except constants	1
Active	Indicates that the function block has control of the axis.		0
CommandAborted	True if power is turned off on the axis (the Enable input of MC_Power is set low) or if another JogAxis instance assumes control of the axis.		0
Warning	Indicates that a warning has occurred within the function block.		0
Error	Indicates that an error has occurred within the function block.		0
ErrorID	Error or warning identification.	WORD	0

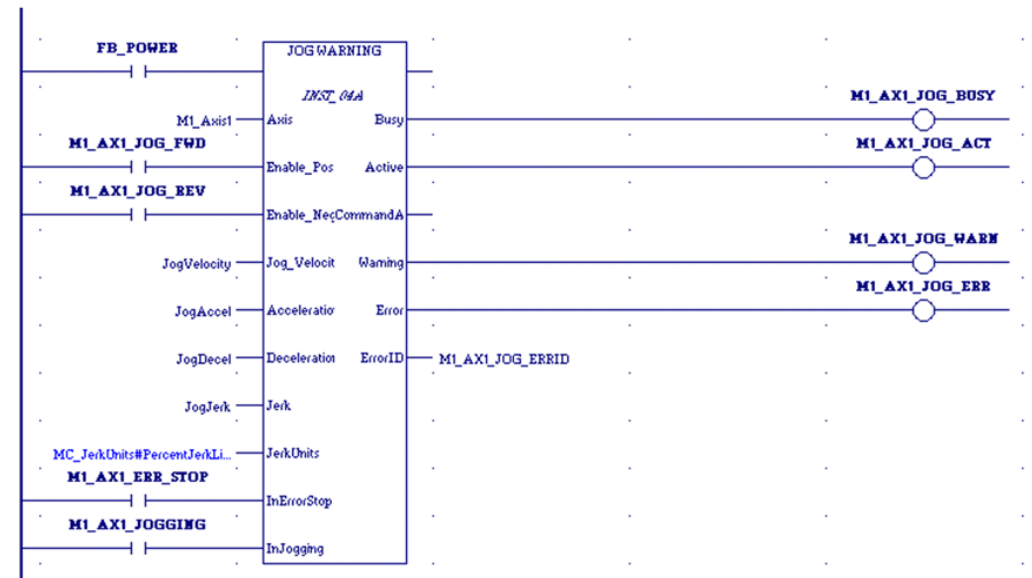
### 6.19.1 Example: Jog to Software End of Travel with Warning

MC\_JogAxis moves at a positive or negative velocity as long as an *Enable* input is held high. If the enable is held high as the axis approaches a Software End-of-Travel (SW-EOT), the PMM will not allow the SW-EOT position to be exceeded. The axis will be stopped at the SW-EOT and transition to *ErrorStop*.

The transition to *ErrorStop* behavior can be changed by creating a user-defined function block (UDFB) to perform the jog motion. The example below implements a version of MC\_JogAxis that generates a Warning rather than an Error and leaves the axis in the *Standstill* state. If the MC\_MoveAbsolute achieves *Done*, then it has reached the SW-EOT. If the MC\_MoveAbsolute is *CommandAborted* by the MC\_Halt, then it did not reach the SW-EOT.

1. First define the UDFB, JogWarning.

Figure 123: Define UDFB JogWarning



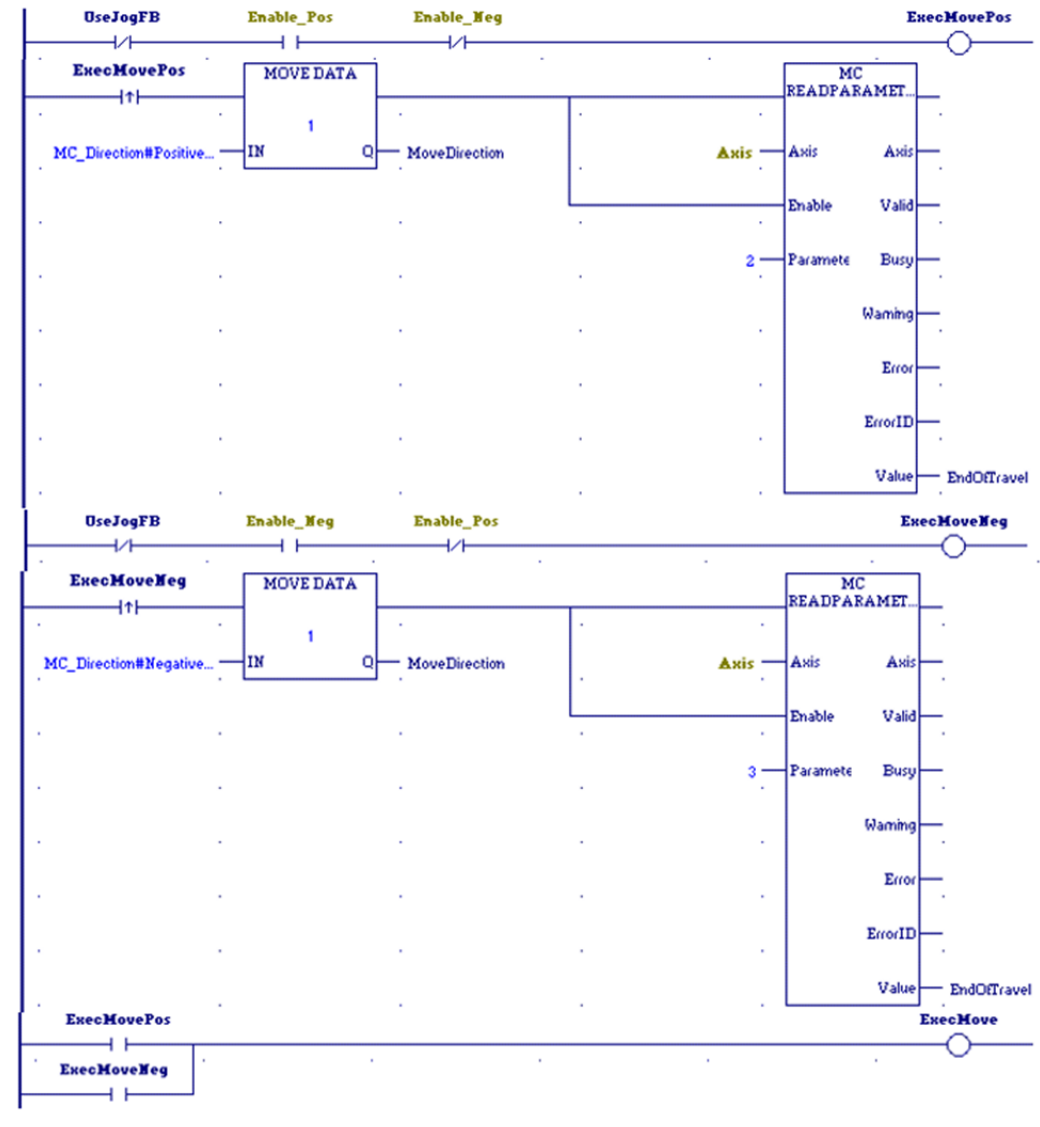
Two state bits are required: InErrorStop and InJogging. These bits must be read using MC\_ReadStatus. The bits allow the MC\_JogAxis function block to be used if the axis is in the ErrorStop state (see step 6).

Figure 124: Monitor InErrorStop and InJogging State Bits



The following logic chooses the direction, finds the SW-EOT position for the current direction and sets the coil used to activate the MC\_MoveAbsolute and MC\_Halt function blocks.

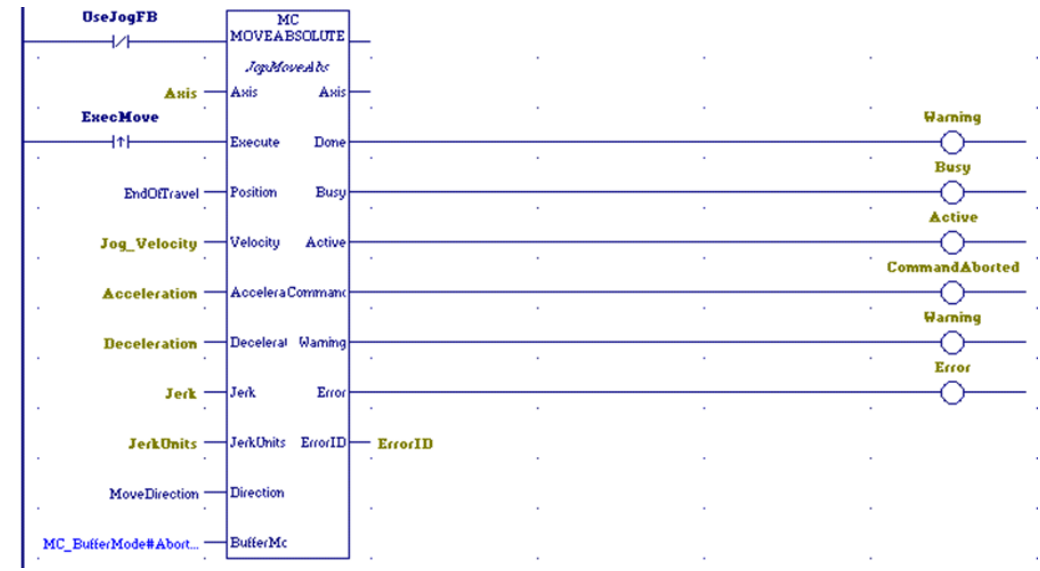
**Figure 125: Set Coil used to Activate the MC\_MoveAbsolute and MC\_Halt function blocks.**





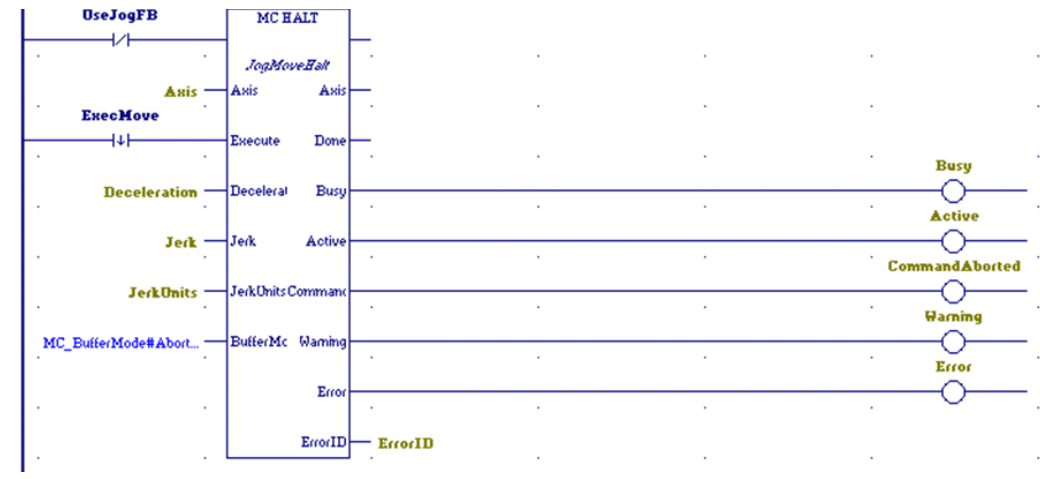
Next, use a POSCON (positive transition contact) to execute a MoveAbsolute in the desired direction to the SW-EOT.

Figure 126: MoveAbsolute in Desired Direction to SW-EOT



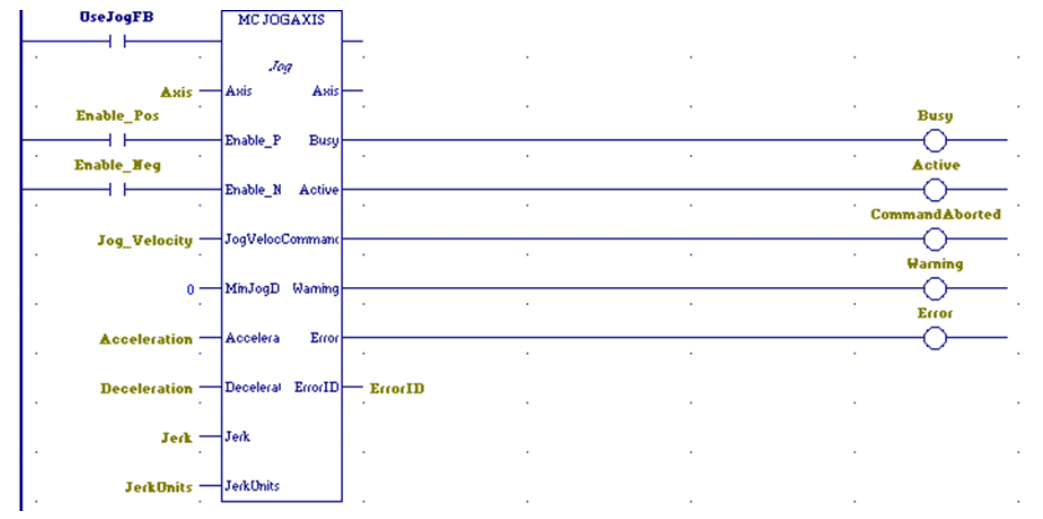
Next, use a NEGCON (negative transition contact) to execute a Halt when the Enable\_\* goes low.

Figure 127: Execute a Halt when Enable\_\* goes Low



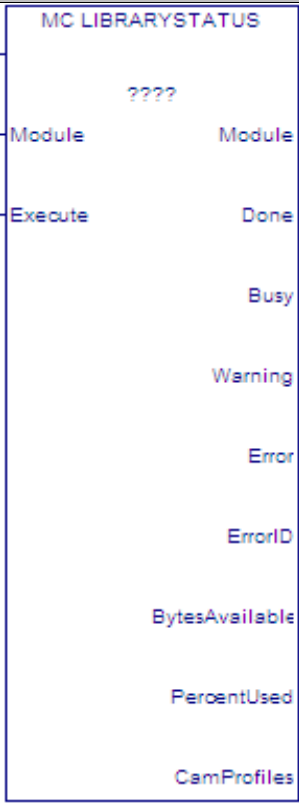
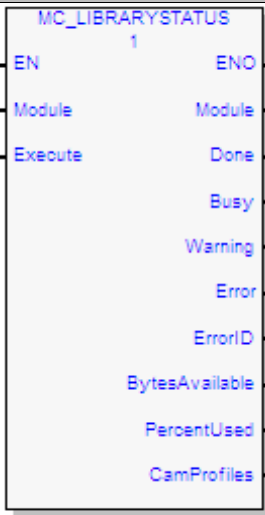
Lastly, if UseJogFB is true, use MC\_JogAxis.

Figure 128: Use MC\_JogAxis If UseJogFB is True



Additional logic can be written to automatically reset the axis if it is jogged from outside a SW-EOT to inside the SW-EOT limit.

## 6.20 MC\_LibraryStatus

LD	FBD	ST
 <p>LD diagram for MC_LIBRARYSTATUS. The diagram shows a vertical box labeled 'MC_LIBRARYSTATUS'. On the left side, there are two input lines: 'Module' and 'Execute'. On the right side, there are eight output lines: 'Module', 'Done', 'Busy', 'Warning', 'Error', 'ErrorID', 'BytesAvailable', 'PercentUsed', and 'CamProfiles'. A '????' is shown in the center of the box.</p>	 <p>FBD diagram for MC_LIBRARYSTATUS. The diagram shows a vertical box labeled 'MC_LIBRARYSTATUS' with a '1' below it. On the left side, there are two input lines: 'EN' and 'Execute'. On the right side, there are eight output lines: 'ENO', 'Module', 'Done', 'Busy', 'Warning', 'Error', 'ErrorID', 'BytesAvailable', 'PercentUsed', and 'CamProfiles'.</p>	<p>Formal convention:        [instance name](Module := [input], Execute := [input], Done =&gt; [output], Busy =&gt; [output], Warning =&gt; [output], Error =&gt; [output], ErrorID =&gt; [output], BytesAvailable =&gt; [output], PercentUsed =&gt; [output], CamProfiles =&gt; [output]);</p>

Each PMM allocates memory for CAM profiles. MC\_LibraryStatus provides the following information on CAM profile memory usage: the number of selected CAM profiles, the total number of bytes available, and the percentage of memory used.

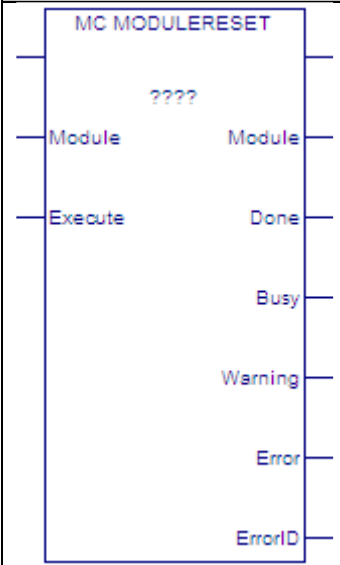
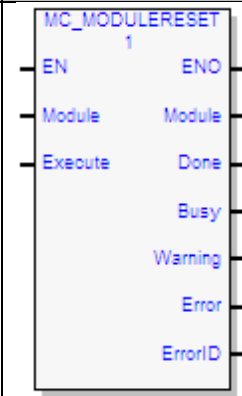
Execution type: Immediate execution/deferred response.

## Operands

Instance Variable	Description	Allowed Data Types	Initial Value
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as ????) in LD.)	MC_LIBRARYSTATUS	NA
Parameter	Description	Allowed Operands	Initial Value
Input_Output Parameters			
Module	Module to be addressed.	MODULE_REF	N/A
Inputs			
Execute	Read library status.	LD: flow Other languages: all except constants	0
Outputs			
Done	Set when the library status is available.	LD: flow Other languages: all except constants	0
Busy	Indicates the function block has been executed and has not yet completed its action.		1
Warning	Indicates that a warning has occurred within the function block.		0
Error	Indicates that an error has occurred within the function block.		0
ErrorID	Error or warning identification.	WORD	0
BytesAvailable <sup>7</sup>	The number of bytes available to hold CAM profiles and motion programs.	DINT	0
PercentUsed <sup>8</sup>	The percentage of space currently used (0–100).	INT	0
CamProfiles <sup>8</sup>	The number of CAM profiles loaded on the module.	INT	0

<sup>7</sup> These outputs are not cleared when Execute transitions low.

## 6.21 MC\_ModuleReset

LD	FBD	ST
 <p>MC_MODULERESET</p> <p>Module</p> <p>Execute</p> <p>Done</p> <p>Busy</p> <p>Warning</p> <p>Error</p> <p>ErrorID</p>	 <p>MC_MODULERESET 1</p> <p>EN ENO</p> <p>Module Module</p> <p>Execute Done</p> <p>Busy</p> <p>Warning</p> <p>Error</p> <p>ErrorID</p>	<p>Formal convention:</p> <p>[instance name](Module := [input], Execute := [input], Done =&gt; [output], Busy =&gt; [output], Warning =&gt; [output], Error =&gt; [output], ErrorID =&gt; [output]);</p>

The MC\_ModuleReset function block is used to clear all errors on a PACMotion module and return any axes in the ErrorStop state to the Standstill state. It does not affect the output of other function block instances.

The Done output is set when no axes in the module are in the ErrorStop state. If the error(s) are not successfully cleared or a new error occurs while clearing the error(s), MC\_ModuleReset returns an error and Done is not set.

If an axis is not in ErrorStop state it continues to operate normally when possible. In some cases, it is necessary to stop functioning axes in order to reset other axes with errors. The stopping method for functioning axes is a normal stop.

MC\_ModuleReset also re-initializes and clears errors on an FTB, if attached and in an error state.

MC\_ModuleReset resets errors on the module. All errors do not take the same time to reset. Thus, the function's execution time can vary based on the error. MC\_ModuleReset has a timeout value of 20 seconds. If MC\_ModuleReset cannot clear the module within this time, it will return an error.

Resetting some errors requires MC\_ModuleReset to stop other axes. The timer does not count down while MC\_ModuleReset is waiting for these axes to stop. Some errors require the user to reset other equipment in the system to clear the error reported by the motion module. A subset of servo drive errors falls into this category. In these situations, after the root cause has been corrected, it may be desirable to resend an MC\_ModuleReset to confirm that the original fault has been corrected and no new faults exist.

Execution type: Immediate execution/deferred response.

## Operands

Instance Variable	Description	Allowed Data Types	Initial Value
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as ??? in LD.)	MC_MODULERESET	NA
Parameter	Description	Allowed Data Types	Initial Value
Input_Output Parameters			
Module	Module to reset.	MODULE_REF	N/A
Inputs			
Execute	The rising edge resets the module.	LD: flow Other languages: all except constants	0
Outputs			
Done	When executed from ErrorStop state indicates Standstill state is reached. If executed from any other state, indicates that no action was taken and a warning was issued.	LD: flow Other languages: all except constants	0
Busy	Indicates the function block has been executed and has not yet completed its action.		1
Warning	Indicates that a warning has occurred within the function block.		0
Error	Indicates that an error has occurred within the function block.		0
ErrorID	Error or warning identification.		WORD

## 6.22 MC\_MoveAbsolute

LD	FBD	ST
<p>MC MOVEABSOLUTE</p> <p>Axis</p> <p>Execute</p> <p>Position</p> <p>Velocity</p> <p>Acceleration</p> <p>Deceleration</p> <p>Jerk</p> <p>JerkUnits</p> <p>Direction</p> <p>BufferMode</p> <p>Axis</p> <p>Done</p> <p>Busy</p> <p>Active</p> <p>CommandAborted</p> <p>Warning</p> <p>Error</p> <p>ErrorID</p>	<p>MC_MOVEABSOLUTE 1</p> <p>EN</p> <p>Axis</p> <p>Execute</p> <p>Position</p> <p>Velocity</p> <p>Acceleration</p> <p>Deceleration</p> <p>Jerk</p> <p>JerkUnits</p> <p>Direction</p> <p>BufferMode</p> <p>ENO</p> <p>Axis</p> <p>Done</p> <p>Busy</p> <p>Active</p> <p>CommandAborted</p> <p>Warning</p> <p>Error</p> <p>ErrorID</p>	<p>Formal convention:</p> <p>[instance name](Axis := [input], Execute := [input], Position := [input], Velocity := [input], Acceleration := [input], Deceleration := [input], Jerk := [input], JerkUnits := [input], Direction := [input], BufferMode := [input], Done =&gt; [output], Busy =&gt; [output], Active =&gt; [output], CommandAborted =&gt; [output], Warning =&gt; [output], Error =&gt; [output], ErrorID =&gt; [output]);</p>

The MC\_MoveAbsolute function block commands the controlled movement of an axis to a specified absolute position relative to the designated home position. While this command is being executed, the axis is in the Discrete Motion state. When the axis commanded position is reached, the Done output will be ON and the axis changes to the Standstill state or begins another move.

Note that the Done output of MC\_MoveAbsolute indicates that the axis commanded position has achieved the position defined by the instruction. The actual position of the axis might still lag the commanded position when the Done output is energized.

This function block can be used on virtual axes as well as physical axes.

Execution type: Immediate execution/deferred response.

## Operands

Instance Variable	Description	Allowed Data Types	Initial Value
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as ??? in LD.)	MC_MOVEABSOLUTE	NA
Parameter	Description	Allowed Data Types	Initial Value
<b>Input_Output Parameters</b>			
Axis	Axis that receives function block command.	AXIS_REF	NA
<b>Inputs</b>			
Execute	Start motion at the rising edge of this input.	LD: flow Other languages: all except constants	0
Position	End position for the move. [Units = UU]	LREAL	0.0
Velocity	The maximum move velocity Maximum velocity not necessarily reached. (Always positive.) [Units = UU/second]	LREAL	0.0
Acceleration	The maximum move acceleration rate (energy is increasing) Maximum acceleration not necessarily reached. (Always positive) [Units = UU/second <sup>2</sup> ]	LREAL	0.0
Deceleration	The maximum move deceleration rate (energy is decreasing). Maximum deceleration is not necessarily reached. (Always positive) [Units = UU/second <sup>2</sup> ]	LREAL	0.0



Instance Variable	Description	Allowed Data Types	Initial Value
Jerk	The Jerk (rate of change in acceleration) used for the move. (Always positive.) If set to 0, jerk will be unlimited. [Units = UU/sec <sup>3</sup> or %]	LREAL For details, refer to Jerk and JerkUnits in Section 5.3.3.	0.0
JerkUnits	Selects units for Jerk input: UU/sec <sup>3</sup> or %	MC_JERKUNITS	0
Direction	One of four values: positive direction, shortest way, negative direction, current direction.	MC_DIRECTION. For details, refer to MC_DIRECTION Data Type in Section 6.22.	0
BufferMode	Defines the axis buffering behavior. Valid modes are Aborting, Buffered and Blending.	MC_BUFFERMODE	0
<b>Outputs</b>			
Done	Commanded position has been reached	LD: flow Other languages: all except constants	0
Busy	Indicates the function block has been executed on an axis and has not yet completed its action.		1
Active	Indicates that the function block has control of the axis.		0
CommandAborted	Command is aborted by another command.		0
Warning	Indicates that a warning has occurred within the function block.		0
Error	Indicates that an error has occurred within the function block. )		0
ErrorID	Error or warning identification.	WORD	0

## MC\_DIRECTION Data Type

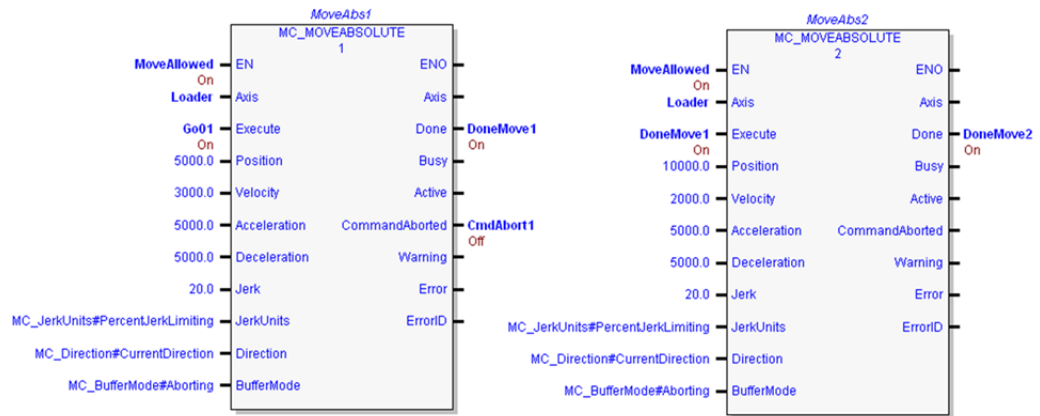
This data type is used with the single-axis MFBs, MC\_MoveVelocity and MC\_MoveAbsolute. It is an enumerated text (ENUM) type with four possible values:

Value	Definition
Positive direction	If Axis Direction is configured for Normal operation, motor shaft rotation will be CCW, looking into the shaft. If Axis Direction is configured for Reverse operation, motor shaft rotation will be CW, looking into the shaft.
Shortest way	Axis moves in the direction that will require the shortest travel to reach the commanded position.
Negative direction	If Axis Direction is configured for Normal operation, motor shaft rotation will be CW, looking into the shaft. If Axis Direction is configured for Reverse operation, motor shaft rotation will be CCW, looking into the shaft.
Current direction	Axis does not change direction.

## 6.22.1 MC\_Move Absolute Example

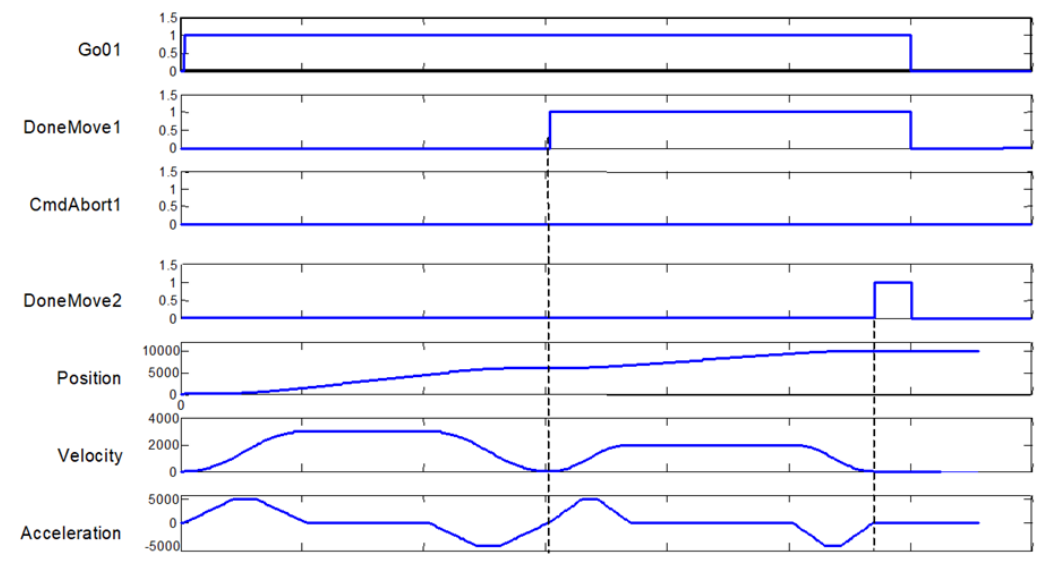
The following block diagram shows two MC\_MoveAbsolute function blocks, MoveAbs1 and MoveAbs2, that are executed one right after the other. MoveAbs1 is executed and runs until it reaches the commanded position of the axis, which is 5000 UU. The Done output, DoneMove1 begins the execution of MoveAbs2. MoveAbs2 executes until the commanded position of the axis reaches 10000 UU.

**Figure 129: MC\_Move Absolute Example**



The following figure shows the timing of the function block input and output parameters in relationship to the axis position, velocity and acceleration.

**Figure 130: Timing of FB Input & Output Parameters in Relationship to Axis Position, Velocity & Acceleration**

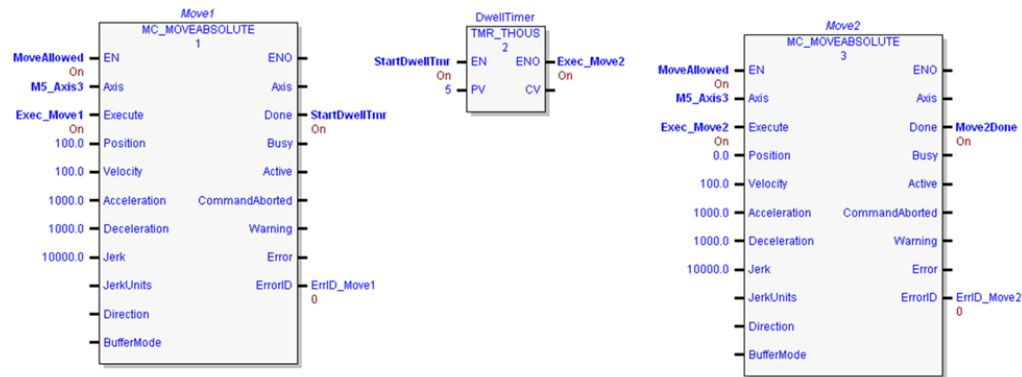


## 6.22.2 MC\_MoveAbsolute Example with Dwell Operation

To add a dwell between motion function blocks, use a timed-interrupt function to monitor the Done output on the first move. When the Done output transitions on, start the timer block for the desired dwell time to trigger the start of the second move.

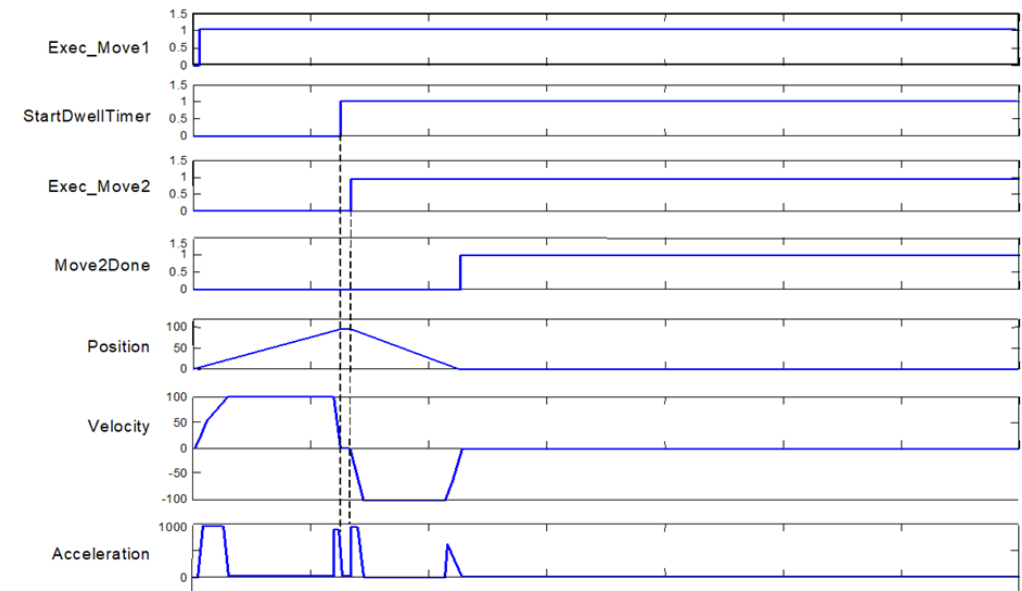
The following logic segment uses an On Delay Timer function block to provide a 5ms delay between the first and second moves.

**Figure 131:**



The following figure shows sample timing of the function block input and output parameters, and typical behavior of axis position, velocity and acceleration.

**Figure 132: Sample Timing of FB Input & Output Parameters, with Typical Behavior of Axis Position, Velocity & Acceleration**



## 6.23 MC\_MoveAdditive

LD	FBD	ST
<p>LD diagram for MC_MOVEADDITIVE. Inputs: Axis, Execute, Distance, Velocity, Acceleration, Deceleration, Jerk, JerkUnits, BufferMode. Outputs: Done, Busy, Active, CommandAborted, Warning, Error, ErrorID.</p>	<p>FBD diagram for MC_MOVEADDITIVE. Inputs: EN, Axis, Execute, Distance, Velocity, Acceleration, Deceleration, Jerk, JerkUnits, BufferMode. Outputs: ENO, Axis, Done, Busy, Active, CommandAborted, Warning, Error, ErrorID.</p>	<p>Formal convention:  [instance name](Axis := [input], Execute := [input], Distance := [input], Velocity := [input], Acceleration := [input], Deceleration := [input], Jerk := [input], JerkUnits := [input], BufferMode := [input], Done =&gt; [output], Busy =&gt; [output], Active =&gt; [output], CommandAborted =&gt; [output], Warning =&gt; [output], Error =&gt; [output], ErrorID =&gt; [output]);</p>

If executed on an axis that is in the Discrete Motion state, this function block commands the axis to move an additional distance relative to the prior commanded position.

If MC\_MoveAdditive is executed on an axis that is in the Continuous Motion or Synchronized Motion state, the specified distance is added to the actual position at the time of the execution.

While this function block is being executed, the axis is in the Discrete Motion state. When the axis position command is reached, the Done output will be ON and the axis goes to the Standstill state or begins another move.

Note that the Done output of MC\_MoveAdditive signifies that the axis commanded position has achieved the position defined by the instruction. It should be understood that the actual position of the axis might still lag the commanded position when the Done output is energized.

Execution type: Immediate execution/deferred response.

## Operands

Instance Variable	Description	Allowed Data Types	Initial Value
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as ???? in LD.)	MC_MOVEADDITIVE	NA
Parameter	Description	Allowed Data Types	Initial Value
Input_Output Parameters			
Axis	Axis that receives function block command.	AXIS_REF	NA
Inputs			
Execute	Start motion at rising edge.	LD: flow Other languages: all except constants	0
Distance	The relative distance to be added to the current move's final destination. [Units = UU]	LREAL	0
Velocity	The maximum move velocity Maximum velocity not necessarily reached. (Always positive.) [Units = UU/second]	LREAL	0
Acceleration	The maximum move acceleration rate (energy is increasing) Maximum acceleration not necessarily reached. (Always positive.) [Units = UU/second <sup>2</sup> ]	LREAL	0
Deceleration	The maximum move deceleration rate (energy is decreasing). Maximum deceleration is not necessarily reached. (Always positive.) [Units = UU/second <sup>2</sup> ]	LREAL	0
Jerk	The Jerk (rate of change in acceleration) used for the move. (Always positive.) If set to 0, jerk will be unlimited. [Units = UU/sec <sup>3</sup> or %]	LREAL <a href="#">For details, refer to Jerk and JerkUnitsin Section 5.3.3.</a>	0
JerkUnits	Selects units for Jerk input: UU/sec <sup>3</sup> or %.	MC_JERKUNITS	0
BufferMode	Defines the axis buffering behavior. Valid modes are Aborting, Buffered and Blending.	MC_BUFFERMODE	0
Outputs			
Done	Commanded position has been reached	LD: flow	0
Busy	Indicates the function block has been executed on an axis and has not yet completed its action.	Other languages: all except constants	1
Active	Indicates that the function block has control of the axis.		0
CommandAborted	Command is aborted by another command.		0

Instance Variable	Description	Allowed Data Types	Initial Value
Warning	Indicates that a warning has occurred within the function block.		0
Error	Indicates that an error has occurred within the function block.		0
ErrorID	Error or warning identification.	WORD	0

### 6.23.1 Example

Figure 133 below shows two MC\_MoveAdditive commands executed one right after the other. As shown in

Figure 134, the initial position for this move is 0. MoveAdd1 is executed and runs until it is done at position 5,000. The rising edge of the Done output, DoneMove1, begins the execution of MoveAdd2. MoveAdd2 executes until it is done, moving an additional 5,000 units, and finishes at position 10,000.

**Figure 133: Two MC\_MoveAdditive commands**

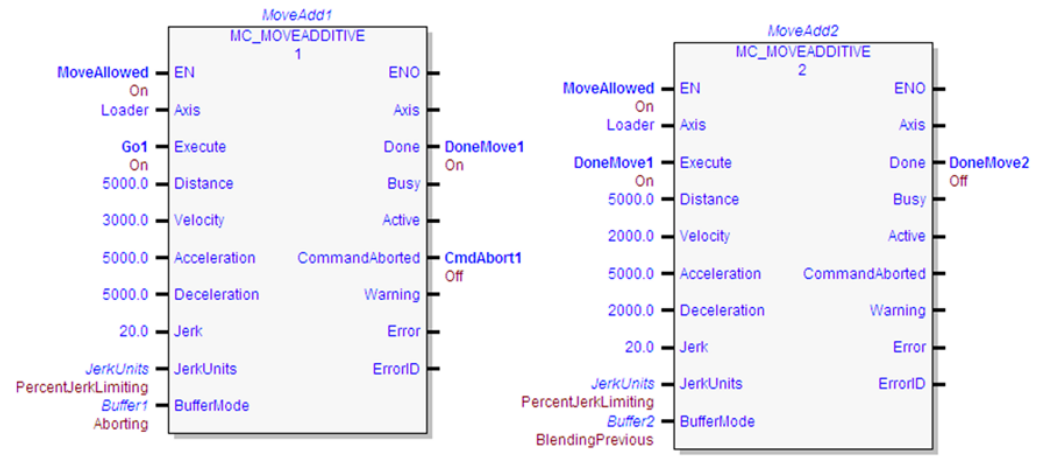
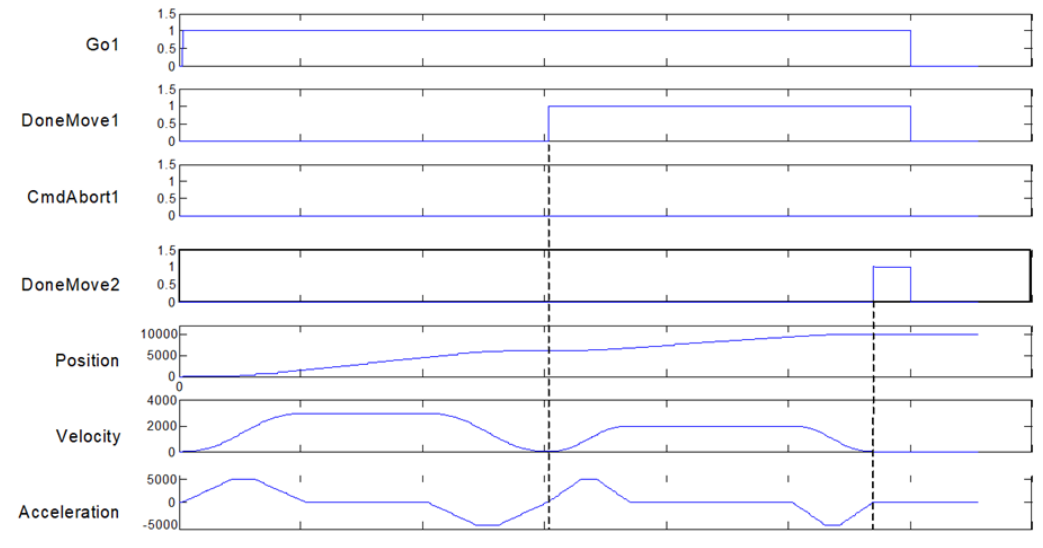




Figure 134: Response to Move Commands



## 6.24 MC\_MoveRelative

LD	FBD	ST
		<p>Formal convention:  [instance name](Axis := [input], Execute := [input], Distance := [input], Velocity := [input], Acceleration := [input], Deceleration := [input], Jerk := [input], JerkUnits := [input], BufferMode := [input], Done =&gt; [output], Busy =&gt; [output], Active =&gt; [output], CommandAborted =&gt; [output], Warning =&gt; [output], Error =&gt; [output], ErrorID =&gt; [output]);</p>

This function block commands an axis to move a specified distance relative to the actual position at the time of the execution. While this function block is being executed, the axis is in the Discrete Motion state. When the axis commanded position is reached, the Done output will be ON and the axis goes to the Standstill state or begins another move.

Note that the Done output of MC\_MoveRelative signifies that the axis commanded position has achieved the position defined by the instruction. The actual position of the axis might still lag the commanded position when the Done output is energized.

Execution type: Immediate execution/deferred response.

## Operands

Instance Variable	Description	Allowed Data Types	Initial Value
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as ??? in LD.)	MC_MOVERELATIVE	NA
Parameter	Description	Allowed Data Types	Initial Value
Input_Output Parameters			
Axis	Axis that receives function block command.	AXIS_REF	NA
Inputs			
Execute	Start motion at rising edge.	LD: flow Other languages: all except constants	0
Distance	Relative distance from the actual position to move. [Units = UU]	LREAL	0
Velocity	The maximum move velocity. Maximum velocity is not necessarily reached. (Always positive.) [Units = UU/second]	LREAL	0
Acceleration	The maximum move acceleration rate (energy is increasing) Maximum acceleration not necessarily reached. (Always positive.) [Units = UU/second <sup>2</sup> ]	LREAL	0
Deceleration	The maximum move deceleration rate (energy is decreasing). Maximum deceleration is not necessarily reached. (Always positive.) [Units = UU/second <sup>2</sup> ]	LREAL	0
Jerk	The Jerk (rate of change in acceleration) used for the move. (Always positive.) If set to 0, jerk will be unlimited. [Units = UU/sec <sup>3</sup> or %]	LREAL <a href="#">For details, refer to Jerk and JerkUnits in Section 5.3.3.</a>	0
JerkUnits	Selects units for Jerk input: UU/sec <sup>3</sup> or %	MC_JERKUNITS	0
BufferMode	Defines the buffering behavior of the axis. Valid modes are Aborting, Buffered and Blending.	MC_BUFFERMODE	0
Outputs			
Done	Commanded position has been reached	LD: flow Other languages: all except constants	0
Busy	Indicates the function block has been executed on an axis and has not yet completed its action.		1
Active	Indicates that the function block has control of the axis.		0
CommandAborted	Command is aborted by another command		0
Warning	Indicates that a warning has occurred within the function block.		0
Error	Indicates that an error has occurred within the function block.		0
ErrorID	Error or warning identification.		WORD

## 6.25 MC\_MoveSuperimposed

LD	FBD	ST
		<p>Formal convention:  [instance name](Axis := [input],  Execute := [input], Distance := [input],  VelocityDiff := [input], Acceleration :=  [input], Deceleration := [input], Jerk :=  [input], JerkUnits := [input], Done =&gt;  [output], Busy =&gt; [output], Active =&gt;  [output], CommandAborted =&gt;  [output], Warning =&gt; [output], Error =&gt;  [output], ErrorID =&gt; [output]);</p>

This function block commands controlled motion of a specified relative distance additional to an existing motion. The existing Motion is not interrupted, but is superimposed by the additional motion.

MC\_MoveSuperimposed can only be executed on the slave axis where an MC\_GearIn is executing.

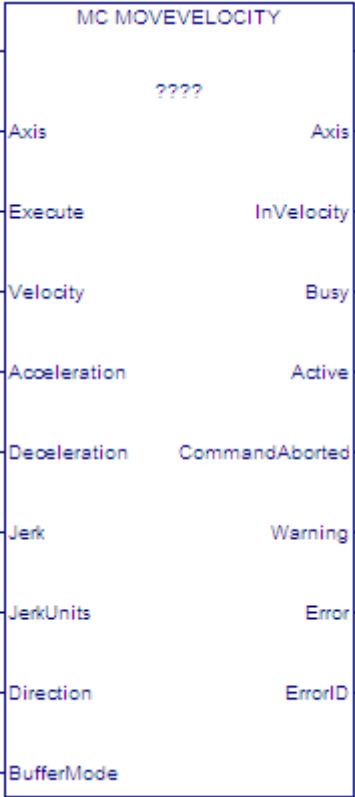

This function block does not operate on the Virtual Axis (Axis 5).

Execution type: Immediate execution/deferred response.

## Operands

Instance Variable	Description	Allowed Data Types	Initial Value	
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as ????) in LD.)	MC_MOVESUPERIMPOSED	NA	
Parameter	Description	Allowed Data Types	Initial Value	
Input_Output Parameters				
Axis	Axis that will perform the motion. Virtual Axis (Axis 5) not supported.	AXIS_REF	N/A	
Inputs				
Execute	Start the motion at rising edge.	LD: flow Other languages: all except constants	0	
Distance	(UU). The relative distance from the current position to the motion's destination.	LREAL	0	
VelocityDiff	Velocity Difference: (UU/second). The value of the maximum velocity difference to the ongoing motion velocity. Depending on Acceleration/Deceleration and the distance to travel, Velocity is not necessarily reached.	LREAL	0	
Acceleration	(UU/second <sup>2</sup> ). The acceleration rate when the energy of the motor is increasing. Depending on the Jerk and the Velocity to accelerate to, Acceleration is not necessarily reached. (Always positive.)	LREAL	0	
Deceleration	(UU/second <sup>2</sup> ). The deceleration rate when the energy of the motor is decreasing. Depending on the Jerk and the Velocity to decelerate to, Deceleration is not necessarily reached. (Always positive.)	LREAL	0	
Jerk	(UU/sec <sup>3</sup> or %) The rate of change in acceleration to be applied to the motion. (Always positive.) If set to 0, jerk will be unlimited.	LREAL For details, refer to Jerk and JerkUnits in Section 5.3.3.	0	
JerkUnits	Selects units for Jerk input: UU/sec <sup>3</sup> or %	MC_JERKUNITS	0	
Outputs				
Done	Commanded position has attained the position specified by the instruction.	LD: flow Other languages: all except constants	0	
Busy	Indicates the function block has been executed and has not yet completed its action.		1	
Active	Indicates that the function block is contributing to the motion of the axis.		0	
CommandAborted	Set to 1 when command is aborted by another command.		0	
Warning	Indicates that a warning has occurred within the function block.		0	
Error	Indicates that an error has occurred within the function block.		0	
ErrorID	Error or warning identification.		WORD	0

## 6.26 MC\_MoveVelocity

LD	FBD	ST
 <p>MC_MOVEVELOCITY</p> <p>Axis</p> <p>Execute</p> <p>Velocity</p> <p>Acceleration</p> <p>Deceleration</p> <p>Jerk</p> <p>JerkUnits</p> <p>Direction</p> <p>BufferMode</p> <p>Axis</p> <p>InVelocity</p> <p>Busy</p> <p>Active</p> <p>CommandAborted</p> <p>Warning</p> <p>Error</p> <p>ErrorID</p>	 <p>MC_MOVEVELOCITY 1</p> <p>EN</p> <p>Axis</p> <p>Execute</p> <p>Velocity</p> <p>Acceleration</p> <p>Deceleration</p> <p>Jerk</p> <p>JerkUnits</p> <p>Direction</p> <p>BufferMode</p> <p>ENO</p> <p>Axis</p> <p>InVelocity</p> <p>Busy</p> <p>Active</p> <p>CommandAborted</p> <p>Warning</p> <p>Error</p> <p>ErrorID</p>	<p>Formal convention:</p> <pre>[instance name](Axis := [input], Execute := [input], Velocity := [input], Acceleration := [input], Deceleration := [input], Jerk := [input], JerkUnits := [input], Direction := [input], BufferMode := [input], InVelocity =&gt; [output], Busy =&gt; [output], Active =&gt; [output], CommandAborted =&gt; [output], Warning =&gt; [output], Error =&gt; [output], ErrorID =&gt; [output]);</pre>

This function block commands a move to the commanded velocity.

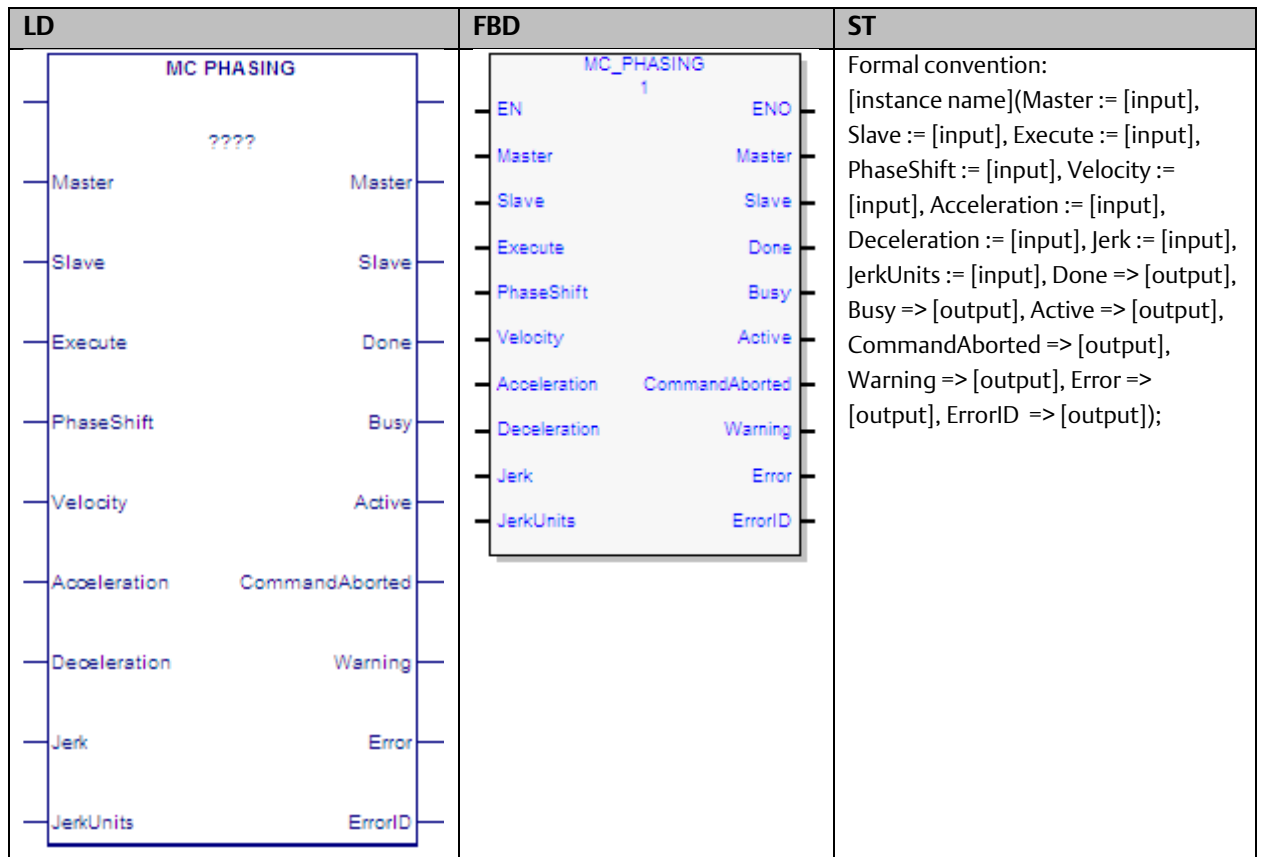
Execution type: Immediate execution/deferred response.

## Operands

Instance Variable	Description	Allowed Data Types	Initial Value
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as ??? in LD.)	MC_MOVEVELOCITY	NA
Parameter	Description	Allowed Data Types	Initial Value
Input_Output Parameters			
Axis	Axis that receives function block command.	AXIS_REF	N/A
Inputs			
Execute	Start motion at rising edge.	LD: flow Other languages: all except constants	0
Velocity	Move's commanded Velocity. When this velocity is reached InVelocity is set. [Units = UU/second]	LREAL	0
Acceleration	The maximum move acceleration rate (energy is increasing) Maximum acceleration not necessarily reached. (Always positive.) [Units = UU/second <sup>2</sup> ]	LREAL	0
Deceleration	The maximum move deceleration rate (energy is decreasing). Maximum deceleration is not necessarily reached. (Always positive.) [Units = UU/second <sup>2</sup> ]	LREAL	0
Jerk	The Jerk (rate of change in acceleration) used for the move. (Always positive.) If set to 0, jerk will be unlimited. [Units = UU/sec <sup>3</sup> or %]	LREAL For details, refer to Jerk and JerkUnits in Section 5.3.3.	0
JerkUnits	Selects units for Jerk input: UU/sec <sup>3</sup> or %	MC_JERKUNITS	0
Direction	One of four enumerated values: positive direction, shortest way <sup>8</sup> , negative direction, current direction.	MC_DIRECTION	0
BufferMode	Defines the behavior of the axis: modes are Aborting, Buffered and Blending.	MC_BufferMode	0
Outputs			
InVel	In Velocity: Commanded velocity has attained the velocity specified by the instruction.	BOOL	0
Busy	Indicates the function block has been executed on an axis and has not yet completed its action.	LD: flow Other languages: all except constants	1
Active	Indicates that the function block has control of the axis		0
CommandAborted	Command is aborted by another command.		0
Warning	Indicates that a warning has occurred within the function block.		0
Error	Indicates that an error has occurred within the function block.		0
ErrorID	Error or warning identification.	WORD	0

<sup>8</sup> **Note:** Shortest way is not supported for this move. If current direction is specified and the direction has not been set, the axis will move in the direction enabled on the MC\_Power instance for the axis. If both directions are enabled on the MC\_Power, an error will be generated.

## 6.27 MC\_Phasing



The MC\_Phasing function block provides dynamic phase shifting capability. In a physical system, one can open the coupling to achieve a phase shift (relative change in the angle of the master and slave shaft positions). An electronically controlled phase shift where both axes are servo controlled is much more flexible. The slave phase can be advanced or delayed to achieve the change, and the rate change is completely programmable.

This function block can be used with CAM slave axes. MC\_Phasing may not be issued until the slave axis is synchronized with the master (the InSync output of the MC\_CamIn instance for the axis is true).

If an MC\_Phasing function block is active and another MC\_Phasing is issued, the first MC\_Phasing is aborted and the second one takes effect immediately.

The Virtual Axis (Axis 5) can be used as a Master input, but not as a Slave input to this function block.

Execution type: Immediate execution/deferred response



## Operands

Instance Variable	Description	Allowed Data Types	Initial Value
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as ??? in LD.)	MC_PHASING	NA
Parameter	Description	Allowed Data Types	Initial Value
Input_Output Parameters			
Master	Reference to master axis.	AXIS_REF	N/A
Slave	Slave Axis reference. Virtual Axis (Axis 5) not supported.	AXIS_REF	N/A
Inputs			
Execute	Start the phasing process at the rising edge.	LD: flow Other languages: all except constants	0
PhaseShift	Phase difference in master (Uu).	LREAL	0
Velocity	Maximum Velocity to reach phase difference (Uu/s)	LREAL	0
Acceleration	Maximum Acceleration to reach phase difference (Uu/s <sup>2</sup> )	LREAL	0
Deceleration	Maximum Deceleration to reach phase difference (Uu/s <sup>2</sup> )	LREAL	0
Jerk	(UU/sec <sup>3</sup> or %) Maximum Jerk to reach phase difference	LREAL For details, refer to Jerk and JerkUnits in Section 5.3.3.	0
JerkUnits	Selects units for Jerk input: UU/sec <sup>3</sup> or %.	MC_JERKUNITS	0
Outputs			
Done	Commanded phasing reached.	LD: flow Other languages: all except constants	0
Busy	Indicates the function block has been executed and has not yet completed its action.		1
Active	Indicates that the MFB is operating on the axis.		0
CommandAborted	Command is aborted by another command.		0
Warning	Indicates that a warning has occurred within the function block.		0
Error	Indicates that an error has occurred within the function block.		0
ErrorID	Identifies the type of error or warning.		WORD

## 6.28 MC\_Power

LD	FBD	ST
		<p>Formal convention:  [instance name](Axis := [input], Enable := [input], Enable_Positive := [input], Enable_Negative := [input], BufferMode := [input], Status =&gt; [output], Busy =&gt; [output], Active =&gt; [output], Warning =&gt; [output], Error =&gt; [output], ErrorID =&gt; [output]);</p>

The MC\_Power function block is used to power an axis on and off and enable motion in the positive and/or negative directions. The function block causes all control loops to be closed and the axis to go to the Standstill state, ready to perform motion commands.

The Enable input on MC\_Power determines whether power is applied to the servo drive. The Status output shows the actual state of the power (based on feedback from the servo drive). The Busy and Active outputs are set true as long as the instance controls the power for the axis. The Warning output is set if warning conditions are encountered. The Error output, if set, indicates an error in the instance and results in the Busy, Active and Warning outputs being set false, indicating that the Status output no longer reflects the power state.

The Busy and Active outputs remain true until another instance of MC\_Power for the same axis receives power flow (see the discussion of multiple instances in Chapter 5, PACMotion Function Block Operation). If an error occurs that prevents power from coming on, the Error and ErrorID outputs of the MC\_Power instance are set and the axis transitions to ErrorStop state. If the error condition is corrected such that power can successfully be applied to the axis, the Error and ErrorID outputs are cleared when the Status output goes true and the axis transitions to Standstill state. Similarly, if a warning condition occurs, the Warning and the ErrorID outputs are set. When the warning condition is cleared, these outputs are cleared. Note that if an error occurs after a warning has been set, the outputs will reflect the error and overwrite the warning.

Execution type: Immediate execution/deferred response.

## Operands

Instance Variable	Description	Allowed Data Types	Initial Value
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as ????) in LD.)	MC_POWER	NA
Parameter	Description	Allowed Data Types	Initial Value
Input/Output Parameters			
Axis	The axis on which the function block is to be executed.	AXIS_REF	N/A
Inputs			
Enable	Turns power on to the axis and enables motion. Refer to Section 6.28.1, MC_Power Input Combinations for details.	LD: flow Other languages: all except constants	0
Enable_Positive	Permits motion in the positive direction. Refer to Section 6.28.1, MC_Power Input Combinations for details..	LD: flow Other languages: all except constants	0
Enable_Negative	Permits motion in the negative direction. Refer to Section 6.28.1, MC_Power Input Combinations for details.	LD: flow Other languages: all except constants	0
Buffer	Defines the behavior of the MC_Power function. Modes are Aborting and Buffered. Blending is not allowed.	MC_BUFFERMODE	Aborting
Outputs			
Status	Indicates axis motion is enabled.	LD: flow	0
Busy	Set when the Enable input is 1 and the function block has not finished executing.	Other languages: all except constants	0
Active	Indicates that the function block has control of the axis.		0
Warning	Indicates that a warning has occurred within the function block.		0
Error	Indicates that an error has occurred within the function block.		0
ErrorID	Error or warning identification.	WORD	0

**Note** that the BufferMode input on a function block is used to control its processing before the instance becomes active (before the Active output on the MFB goes high). Once the Active output on the MFB becomes true, changes to the buffer mode will be ignored until Active goes false again. For MC\_Power that will only occur due to an error or if the instance is superseded by another instance of MC\_Power for the axis. It is not possible to buffer a change to an active MC\_Power instance. Once an instance is active, the application must use multiple instances to accomplish this functionality (and must properly interlock the power flow so that only one instance has power flow at any time). For an example, refer to in Section 5.3.3.

## 6.28.1 MC\_Power Input Combinations

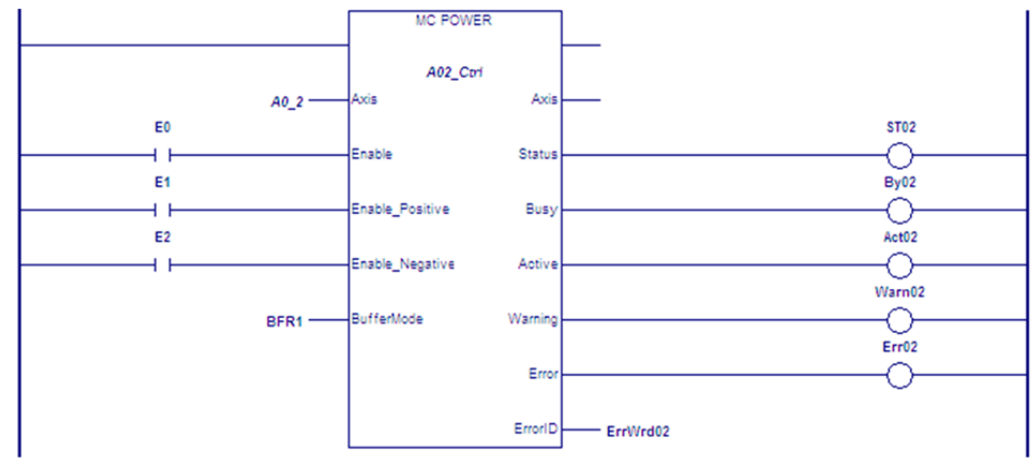
The following table shows possible input combinations and the resulting motion that is enabled. The “x” indicates a value that is ignored.

Value of Enable	Value of Enable Positive	Value of Enable Negative	Result
0	x	x	All motion disabled
1	0	0	Motion enabled in both the positive and negative directions
1	1	0	Motion enabled in the positive direction only
1	0	1	Motion enabled in the negative direction only
1	1	1	Motion enabled in both the positive and negative directions

### Example

In the following sample logic, the MC\_Power function block controls an axis named A0\_2.

**Figure 135: MC\_Power Function Block Example**



## 6.29 MC\_ReadActualPosition

LD	FBD	ST
<p>The Ladder Logic Diagram (LD) shows a single normally open contact labeled 'MC_READACTUALPOSITION'. The contact has three input terminals on the left: 'Axis', 'Enable', and 'Position'. It has seven output terminals on the right: 'Axis', 'Valid', 'Busy', 'Warning', 'Error', 'ErrorID', and 'Position'.</p>	<p>The Function Block Diagram (FBD) shows a function block labeled 'MC_READACTUALPOSITION' with a '1' in the top right corner. It has three input terminals on the left: 'EN', 'Axis', and 'Enable'. It has seven output terminals on the right: 'ENO', 'Axis', 'Valid', 'Busy', 'Warning', 'Error', and 'ErrorID'. The 'Position' output is shown as a text label below the block.</p>	<p>Formal convention:  MC_READACTUALPOSITION(Axis := [input],  Enable := [input], Valid =&gt; [output], Busy =&gt;  [output], Warning =&gt; [output], Error =&gt;  [output], ErrorID =&gt; [output], Position =&gt;  [output]);</p>

The MC\_ReadActualPosition function is used to read the actual axis position. Actual position is a value maintained by the PMM to represent the physical axis position.

If reading Actual Position for a Virtual Axis (Axis 5), valid data will be returned only if an external encoder is used.

Execution type: Immediate execution/immediate response.

## Operands

Parameter	Description	Allowed Data Types	Initial Value
<b>Input_Output Parameters</b>			
Axis	The axis on which the function is to be executed.	AXIS_REF	N/A
<b>Inputs</b>			
Enable	Read the position of an axis while enabled.	LD: flow Other languages: all except constants	N/A
<b>Outputs</b>			
Valid	Set to 1 if valid outputs are available.	LD: flow Other languages: all except constants	0
Busy	Indicates the Enable input is 1 and the function has not finished executing.		1
Warning	Indicates that a warning has occurred within the function block.		0
Error	Indicates that an error has occurred within the function.		0
ErrorID	Error or warning identification.	WORD	0
Position	Actual position in axis' unit (u)	LREAL	0

## 6.30 MC\_ReadActualVelocity

LD	FBD	ST
<p>LD symbol for MC_READACTUALVELOCITY. Inputs: Axis, Enable. Outputs: Valid, Busy, Warning, Error, ErrorID, ActualVelocity.</p>	<p>FBD symbol for MC_READACTUALVELOCITY. Inputs: EN, Axis, Enable. Outputs: ENO, Axis, Valid, Busy, Warning, Error, ErrorID, ActualVelocity.</p>	<p>Formal convention:  MC_READACTUALVELOCITY(Axis := [input],  Enable := [input], Valid =&gt; [output], Busy =&gt;  [output], Warning =&gt; [output], Error =&gt;  [output], ErrorID =&gt; [output], Actual  Velocity =&gt; [output]);</p>

This function returns the actual axis velocity while the Enable input is set. The Valid output is 1 when the Actual Velocity is valid.

If reading Actual Velocity for a Virtual Axis (Axis 5), valid data will be returned only if an external encoder is used.

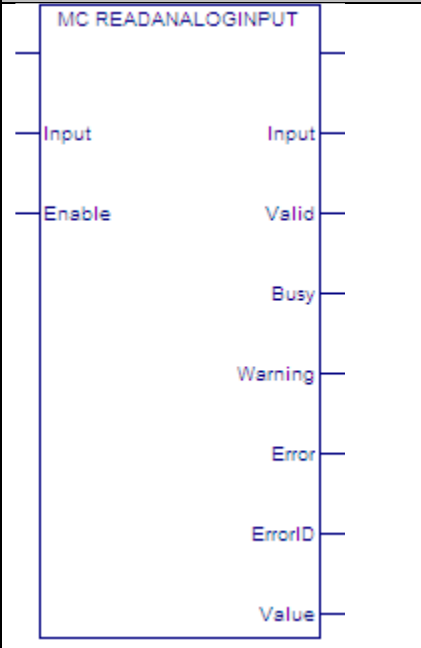
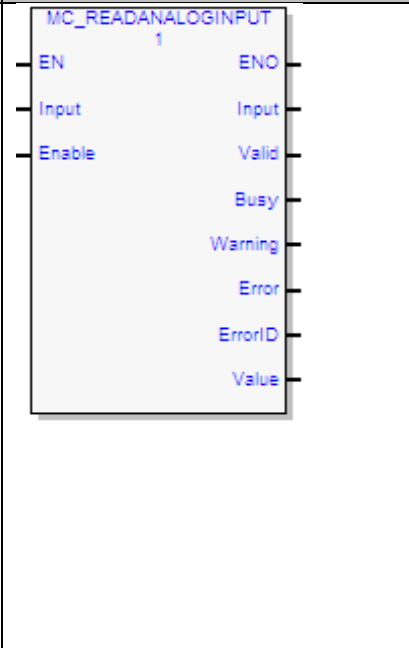
Execution type: Immediate execution/immediate response.

## Operands

Parameter	Description	Allowed Data Types	Initial Value
<b>Input_Output Parameters</b>			
Axis	The axis on which the function is to be executed.	AXIS_REF	N/A
<b>Inputs</b>			
Enable	Read the actual velocity while enabled.	LD: flow Other languages: all except constants	N/A
<b>Outputs</b>			
Valid	Value is available.	LD: flow Other languages: all except constants	0
Busy	Indicates the Enable input is 1 and the function block has not finished executing.	LD: flow Other languages: all except constants	1
Warning	Indicates that a warning has occurred within the function.		0
Error	Indicates that an error has occurred within the function.		0
ErrorID	Error or warning identification.	WORD	0
ActualVelocity	The value of the actual velocity in UU/second.	LREAL	0



## 6.31 MC\_ReadAnalogInput

LD	FBD	ST
 <p>LD symbol for MC_READANALOGINPUT. It is a rectangular box with 'MC_READANALOGINPUT' at the top. On the left side, there are two input terminals labeled 'Input' and 'Enable'. On the right side, there are eight output terminals labeled 'Input', 'Valid', 'Busy', 'Warning', 'Error', 'ErrorID', and 'Value'.</p>	 <p>FBD symbol for MC_READANALOGINPUT. It is a rectangular box with 'MC_READANALOGINPUT' and a small '1' at the top. On the left side, there are three input terminals labeled 'EN', 'Input', and 'Enable'. On the right side, there are eight output terminals labeled 'ENO', 'Input', 'Valid', 'Busy', 'Warning', 'Error', 'ErrorID', and 'Value'.</p>	<p>Formal convention:  MC_READANALOGINPUT(Input := [input], Enable := [input], Valid =&gt; [output], Busy =&gt; [output], Warning =&gt; [output], Error =&gt; [output], ErrorID =&gt; [output], Value =&gt; [output]);</p>

This function provides the ability to monitor the  $\pm 10\text{Vdc}$  single ended analog inputs provided by the FTB. It returns the value of the analog input specified by the parameter Input.

The Input is identified by an I/O data reference number, which is passed to the instruction as part of the INPUT\_REF input variable. These reference numbers cannot be accessed directly by a Parameter Read or Parameter Write instruction. For a list of I/O reference numbers, refer to Section 8.3 I/O Data Reference Numbers.

For specifications and connection details for the analog inputs, refer to Section 3, I/O Wiring, Connections and LED Operation.

Execution type: Immediate execution/immediate response.

## Operands

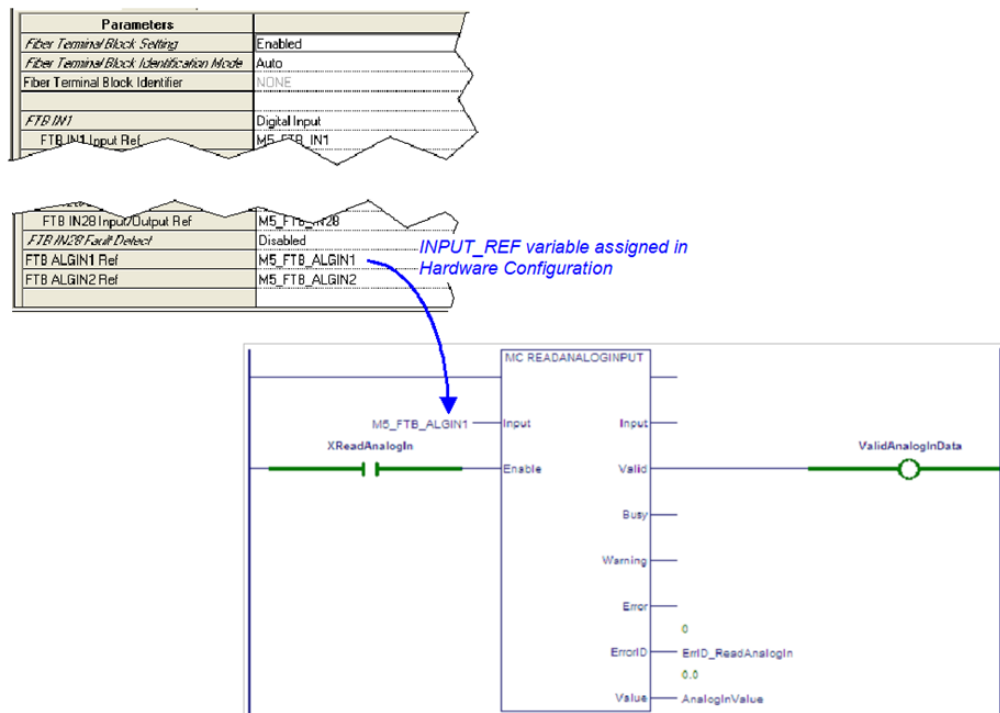
Parameter	Description	Allowed Data Types	Initial Value
<b>Input_Output Parameters</b>			
Input	Reference to the input signal	INPUT_REF defined in the hardware configuration for the module associated with the FTB.	N/A
<b>Inputs</b>			
Enable	Read the analog input while enabled.	LD: flow Other languages: all except constants	0
<b>Outputs</b>			
Valid	Input analog signal value is valid	LD: flow Other languages: all except constants	0
Busy	Indicates the function block is enabled and has not completed its action.		1
Warning	Indicates that a warning has occurred within the function.		0
Error	Indicates that an error has occurred within the function.		0
ErrorID	Error or warning identification.	WORD	0
Value	The value of the selected analog input signal. Units = volts.	LREAL	0

### 6.31.1 Input Example

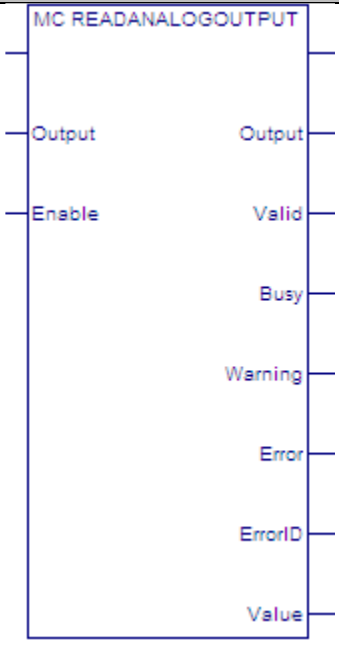
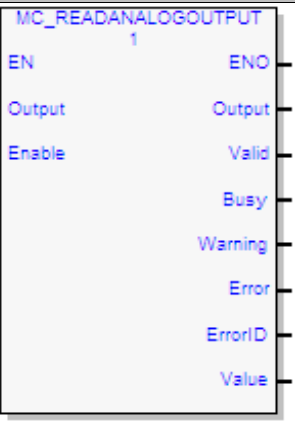
In the following example, the INPUT\_REF variable M5\_FTB\_IN1 has been created on the FTB Inputs tab in the hardware configuration and assigned to the FTB input point FTB IN1.

When XReadAnalogIn is on, the MC\_ReadAnalogInput function block is enabled and the selected input point value is placed in the reference memory location, AnalogInVal, which is assigned to the function's Value output.

**Figure 136: Read Analog Input Function Block Example**



## 6.32 MC\_ReadAnalogOutput

LD	FBD	ST
		<p>Formal convention:  MC_READANALOGOUTPUT(Output := [input], Enable := [input], Valid =&gt; [output], Busy =&gt; [output], Warning =&gt; [output], Error =&gt; [output], ErrorID =&gt; [output], Value =&gt; [output]);</p>

This function provides the ability to monitor the  $\pm 10\text{Vdc}$  single ended analog outputs provided by the FTB. It returns the value of the analog output specified by the parameter Output.

The Output is identified by an I/O data reference number, which is passed to the instruction as part of the OUTPUT\_REF input variable. These reference numbers cannot be accessed directly by a Parameter Read or Parameter Write instruction. For a list of I/O reference numbers, refer to Section 8.3 I/O Data Reference Numbers.

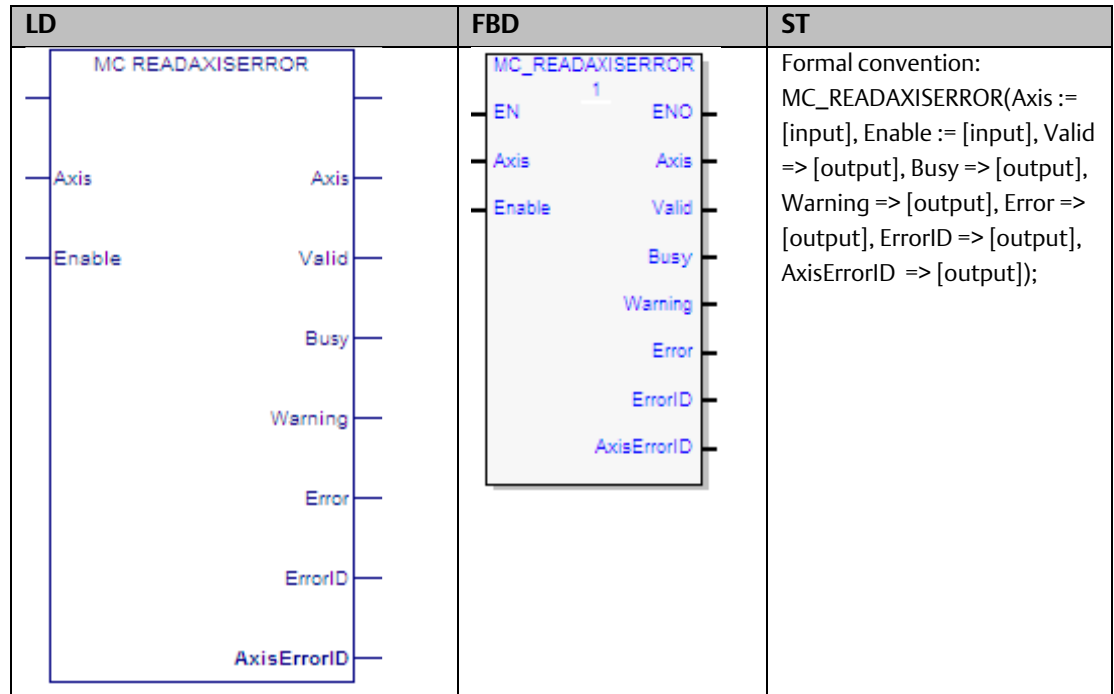
For specifications and connection details for the analog outputs, refer to Section 3, I/O Wiring, Connections and LED Operation.

Execution type: Immediate execution/immediate response.

## Operands

Parameter	Description	Allowed Data Types	Initial Value
<b>Input_Output Parameters</b>			
Output	Reference to the analog signal output.	OUTPUT_REF defined in the hardware configuration for the module associated with the FTB.	N/A
<b>Inputs</b>			
Enable	Read the analog output while enabled.	LD: flow Other languages: all except constants	N/A
<b>Outputs</b>			
Valid	Output analog signal value is valid.	LD: flow	0
Busy	Indicates the function block is enabled and has not completed its action.	Other languages: all except constants	1
Warning	Indicates that a warning has occurred within the function.		0
Error	Indicates that an error has occurred within the function.		0
ErrorID	Error or warning identification.	WORD	0
Value	The value of the selected analog output signal. Units = volts.	LREAL	0

## 6.33 MC\_ReadAxisError



This function is used to read the current axis error or warning. MC\_ReadAxisError returns the most recent, highest severity error or warning message.

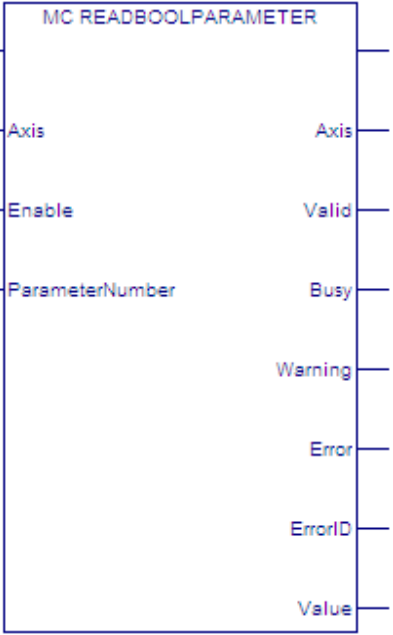
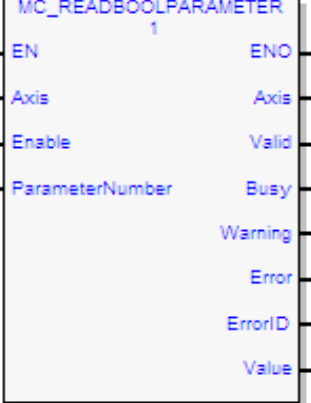
To read multiple errors and warnings, use the MC\_ReadEventQueue function block.

Execution type: Immediate execution/immediate response.

## Operands

Parameter	Description	Allowed Data Types	Initial Value
<b>Input_Output Parameters</b>			
Axis	The axis on which the function is to be executed.	AXIS_REF	N/A
<b>Inputs</b>			
Enable	Read the axis error while enabled.	LD: flow Other languages: all except constants	N/A
<b>Outputs</b>			
Valid	Set to 1 if valid outputs are available.	LD: flow Other languages: all except constants	0
Busy	Indicates the Enable input is 1 and the function block has not finished executing.		1
Warning	Indicates that a warning has occurred within the function.		0
Error	Indicates that an error has occurred within the function.		0
ErrorID	Error or warning identification for function.	WORD	0
AxisErrorID	The value of the axis error. Axis error IDs are 16-bit numbers that encode information about the nature of an axis error. For a numerical list of all Error IDs, refer to Section 9.1.5, Error ID Reference.	All except constants and variables located in %S.	0

## 6.34 MC\_ReadBoolParameter

LD	FBD	ST
 <p>MC READBOOLPARAMETER</p> <p>Axis</p> <p>Enable</p> <p>ParameterNumber</p> <p>Axis</p> <p>Valid</p> <p>Busy</p> <p>Warning</p> <p>Error</p> <p>ErrorID</p> <p>Value</p>	 <p>MC_READBOOLPARAMETER 1</p> <p>EN</p> <p>Axis</p> <p>Enable</p> <p>ParameterNumber</p> <p>ENO</p> <p>Axis</p> <p>Valid</p> <p>Busy</p> <p>Warning</p> <p>Error</p> <p>ErrorID</p> <p>Value</p>	<p>Formal convention: MC_READBOOLPARAMETER(Axis := [input], Enable := [input], ParameterNumber := [input], Valid =&gt; [output], Busy =&gt; [output], Warning =&gt; [output], Error =&gt; [output], ErrorID =&gt; [output], Value =&gt; [output]);</p>

This function returns the value of a Boolean hardware configuration parameter, which is identified by the Axis and ParameterNumber input parameters. To read a module parameter, specify any valid axis on the module.

For a list of parameter numbers, refer to Axis Parameter Number Index in Section 8.1.1.

Execution type: Immediate execution/immediate response.



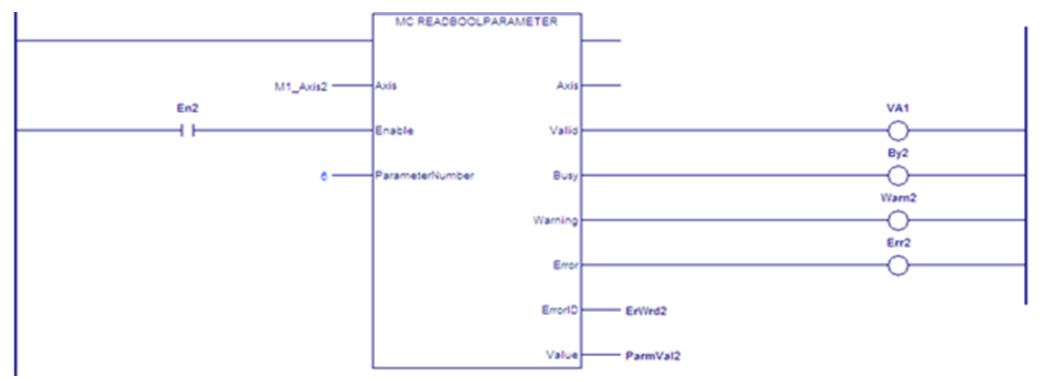
## Operands

Parameter	Description	Allowed Data Types	Initial Value
<b>Input_Output Parameters</b>			
Axis	Axis with parameter to read.	AXIS_REF	N/A
<b>Inputs</b>			
Enable	Read the parameter while enabled.	LD: flow Other languages: all except constants	N/A
ParameterNumber	The parameter to read. This value can be a constant or a mapped variable.	Constant, INT	N/A
<b>Outputs</b>			
Valid	Indicates valid outputs are available	LD: flow Other languages: all except constants	0
Busy	Indicates the function block is enabled and has not completed its action.		1
Warning	Indicates that a warning has occurred within the function.		0
Error	Indicates that an error has occurred within the function.		0
ErrorID	Error or warning identification.	WORD	0
Value	Value of the specified Boolean parameter.	BOOL	0

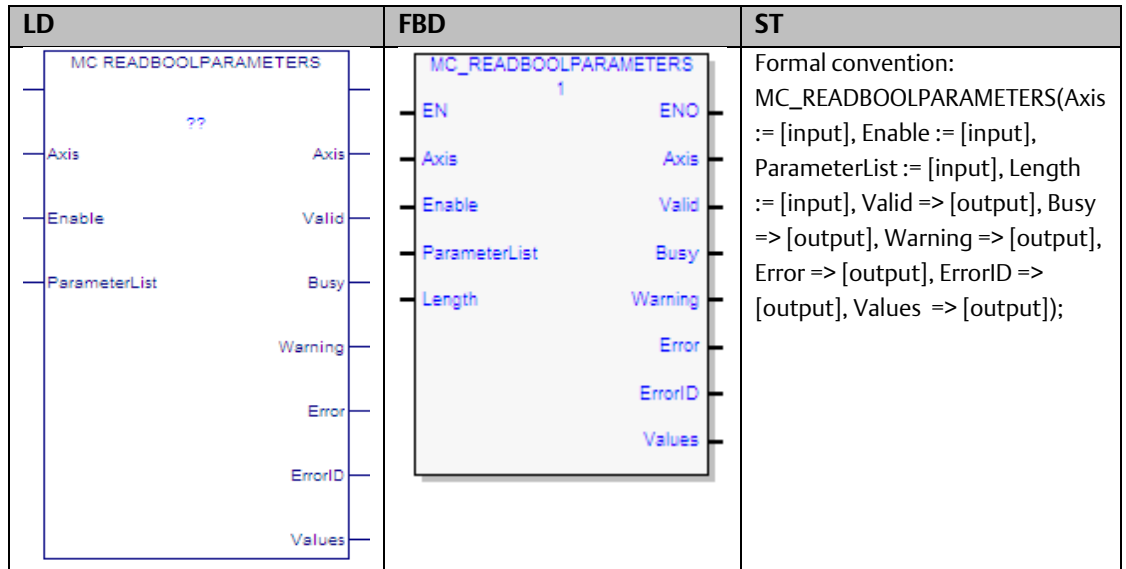
### Example

In the following example, the MC\_ReadBoolParameter function reads the Position Lag Monitoring enable state (parameter 6) of the axis M1\_Axis2 and writes the value to ParmVal2.

**Figure 137: MC\_ReadBoolParameter Function Block Example**



## 6.35 MC\_ReadBoolParameters



This function returns the values of up to 16 Boolean axis parameters specified by the Axis and ParameterList input parameters. To read module parameters, specify any valid axis on the module.

If any parameter in the list is invalid, the function returns an error and no data is returned.

Execution type: Immediate execution/immediate response.

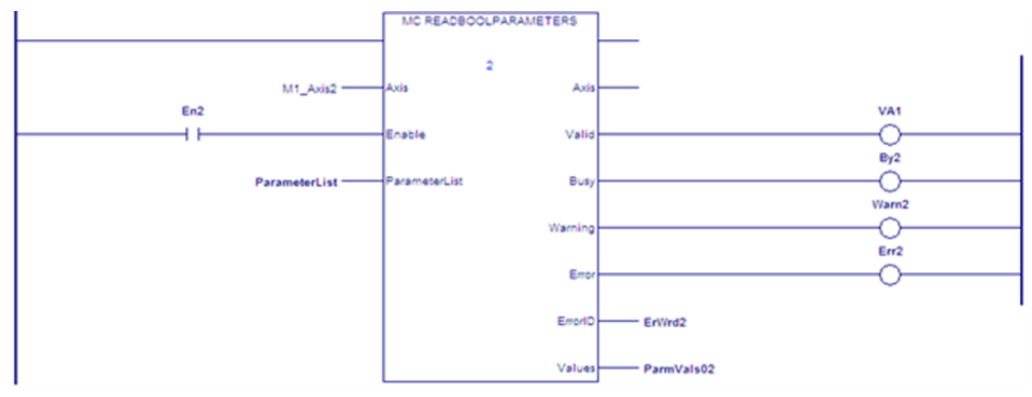
## Operands

Parameter	Description	Allowed Data Types	Initial Value
??	Length. The number of parameters and values to read, starting with the first parameter in the ParameterList array (maximum of 16).	Constant	N/A
<b>Input_Output Parameters</b>			
Axis	Axis with parameter to read.	AXIS_REF	N/A
<b>Inputs</b>			
Enable	Read the parameters while enabled.	LD: flow Other languages: all except constants	N/A
ParameterList	Array of parameter numbers. Must have enough members to accommodate Length. All parameter numbers must specify axis parameters or all must specify module parameters.	INT[ ]	N/A
<b>Outputs</b>			
Valid	Indicates valid outputs are available	LD: flow Other languages: all except constants	0
Busy	Indicates the function block is enabled and has not completed its action.		1
Warning	Indicates that a warning has occurred within the function.		0
Error	Indicates that an error has occurred within the function.		0
ErrorID	Error or warning identification.	WORD	0
Values	Array containing values of the specified parameters. Must have enough elements to accommodate Length.	BOOL[ ]	0

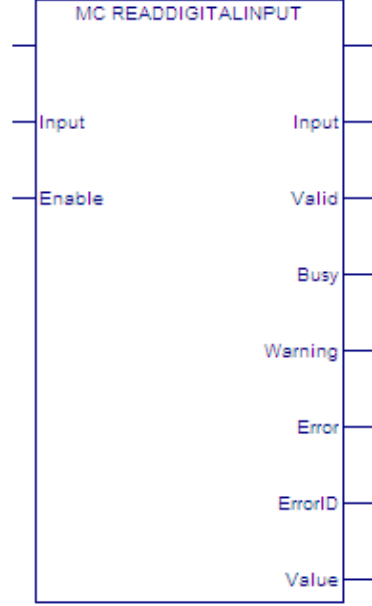
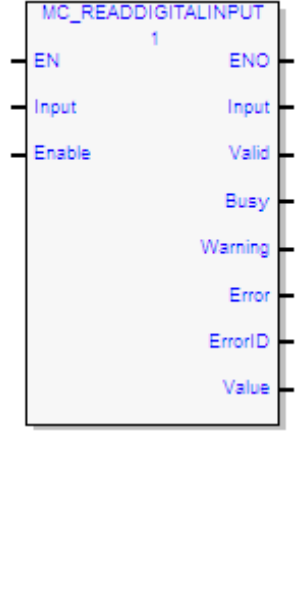
### Example

In the following example, the MC\_ReadBoolParameters function reads the values of two parameters, specified by the array variable ParameterList, for the axis named M1\_Axis2. The function writes the parameter values to the array ParmVals02.

**Figure 138: MC\_ReadBoolParameters Function Block Example**



## 6.36 MC\_ReadDigitalInput

LD	FBD	ST
 <p>MC_READDIGITALINPUT</p> <p>Input</p> <p>Enable</p> <p>Valid</p> <p>Busy</p> <p>Warning</p> <p>Error</p> <p>ErrorID</p> <p>Value</p>	 <p>MC_READDIGITALINPUT</p> <p>1</p> <p>EN</p> <p>Input</p> <p>Enable</p> <p>ENO</p> <p>Input</p> <p>Valid</p> <p>Busy</p> <p>Warning</p> <p>Error</p> <p>ErrorID</p> <p>Value</p>	<p>Formal convention:  MC_READDIGITALINPUT(Input := [input], Enable := [input], Valid =&gt; [output], Busy =&gt; [output], Warning =&gt; [output], Error =&gt; [output], ErrorID =&gt; [output], Value =&gt; [output]);</p>

This function returns the value of the discrete faceplate or FTB input specified by the Input parameter. Specifications and connection details for the discrete I/O points are provided in Chapter 3, I/O Wiring, Connections and LED Operation.

The Input is identified by an I/O data reference number, which is passed to the instruction as part of the INPUT\_REF input variable. These reference numbers cannot be accessed directly by a Parameter Read or Parameter Write instruction. For a list of I/O reference numbers, refer to Section 8.3 I/O Data Reference Numbers.

Execution type: Immediate execution/immediate response.

## Operands

Parameter	Description	Allowed Data Types	Initial Value
<b>Input_Output Parameters</b>			
Input	Reference to the input signal source.	INPUT_REF, defined in the module hardware configuration.	N/A
<b>Inputs</b>			
Enable	Read the digital input while enabled.	LD: flow Other languages: all except constants	N//A
<b>Outputs</b>			
Valid	Input signal value is valid.	LD: flow Other languages: all except constants	0
Busy	Indicates the function block has been executed and has not completed its action.		1
Warning	Indicates that a warning has occurred within the function.		0
Error	Indicates that an error has occurred within the function.		0
ErrorID	Error or warning identification.	WORD	0
Value	The value of the selected input signal.	BOOL	0

### 6.36.1 Example

In the following example, the INPUT\_REF variables M5\_FP\_IN1 and M5\_FP\_IN2 are created in hardware configuration and assigned to the faceplate input points IN1 and IN2.

---

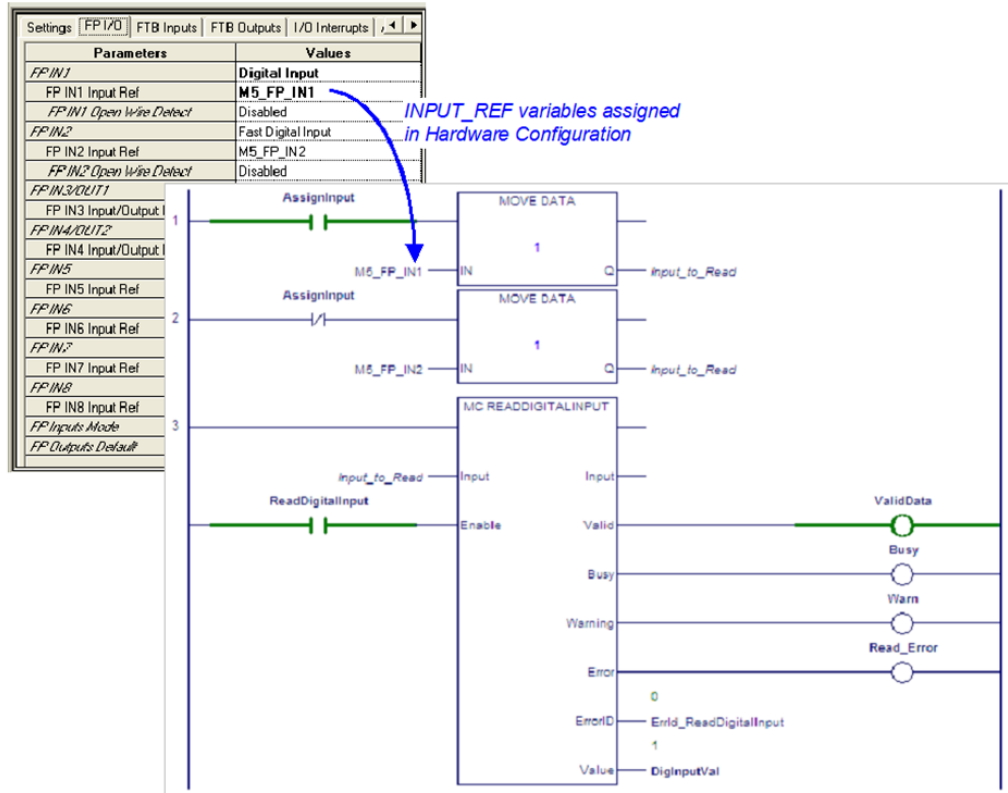
**Note:** To automatically create a variable with data type INPUT\_REF, create it first as the Input parameter to the MC\_ReadDigitalInput function. It can then be used as the IN parameter to the MoveData functions.

---

MoveData functions are used to select which input to read when ReadDigitalInput is on. When AssignInput is on, M5\_FP\_IN1 is copied to the INPUT\_REF variable, Input\_to\_Read so that FP IN1 will be read. When AssignInput is off, M5\_FP\_IN2 is copied to Input\_to\_Read and FP IN2 will be read.

When ReadDigitalInput is on, the MC\_ReadDigitalInput function block is enabled and the selected input point value is placed in the reference memory location, DigInputVal, which is assigned to the function block's Value output.

Figure 139: MC\_ReadDigitalInput Function Block Example



## 6.37 MC\_ReadDigitalOutput

LD	FBD	ST
<p>LD diagram for MC_READDIGITALOUTPUT. The function block is a vertical rectangle with the following inputs on the right side: Output, Valid, Busy, Warning, Error, ErrorID, and Value. The top of the block is labeled 'MC_READDIGITALOUTPUT'.</p>	<p>FBD diagram for MC_READDIGITALOUTPUT. The function block is a vertical rectangle with the following inputs on the right side: EN, Output, Enable, ENO, Output, Valid, Busy, Warning, Error, ErrorID, and Value. The top of the block is labeled 'MC_READDIGITALOUTPUT' and has a '1' below it.</p>	<p>Formal convention:  MC_READDIGITALOUTPUT(Output := [input], Enable := [input], Valid =&gt; [output], Busy =&gt; [output], Warning =&gt; [output], Error =&gt; [output], ErrorID =&gt; [output], Value =&gt; [output]);</p>

This function returns the value of the discrete faceplate or FTB output specified by the Output parameter. For specifications and connection details for the discrete I/O points, refer to Section 3, I/O Wiring, Connections and LED Operation.

The operation of the MC\_ReadDigitalOutput function is similar to that of the MC\_ReadDigitalInput function, for which an example is provided in Section 6.36.1.

Note that the two 24Vdc faceplate outputs, OUT1 and OUT2, share terminals with the inputs, IN3 and IN4. To be used as outputs, these terminals must be configured as such in hardware configuration.

Execution type: Immediate execution/immediate response.

## Operands

Parameter	Description	Allowed Data Types	Initial Value
<b>Input_Output Parameters</b>			
Output	Reference to the signal output.	OUTPUT_REF, defined in the module hardware configuration.	N/A
<b>Inputs</b>			
Enable	Read the digital output while enabled.	LD: flow Other languages: all except constants	N/A
<b>Outputs</b>			
Valid	Output signal value is valid.	LD: flow Other languages: all except constants	0
Busy	Indicates the function block has been executed and has not completed its action.		1
Warning	Indicates that a warning has occurred within the function.		0
Error	Indicates that an error has occurred within the function.		0
ErrorID	Error or warning identification.	WORD	0
Value	The value of the selected output signal.	BOOL	0



## 6.38 MC\_ReadDwordParameters

LD	FBD	ST
		<p>Formal convention:  MC_READDWORDPARAMETERS(Axis  := [input], Enable := [input],  ParameterList := [input], Length :=  [input], Valid =&gt; [output], Busy =&gt;  [output], Warning =&gt; [output], Error  =&gt; [output], ErrorID =&gt; [output],  Values =&gt; [output]);</p>

This function returns the values of up to 16 Dword axis parameters, which are specified by the Axis and ParameterList inputs. To read module parameters, specify any valid axis on the module.

This function is used to read parameters that cannot be expressed as a real value, including packed bits.

If any parameter in the list is invalid, the function returns an error and no data is returned.

---

**Note:** To read DINT parameters, change the Data Type of the variables to DINT instead of DWORD.


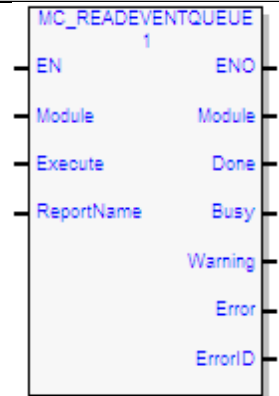
---

Execution type: Immediate execution/immediate response.

## Operands

Parameter	Description	Allowed Data Types	Initial Value
??	Length: The number of parameters and values to read, starting with the first parameter in the ParameterList array (maximum of 16).	Constant	N/A
<b>Input_Output Parameters</b>			
Axis	Axis with parameters to read.	AXIS_REF	N/A
<b>Inputs</b>			
Enable	Read the parameters while enabled.	LD: flow Other languages: all except constants.	N/A
ParameterList	Array of parameter numbers. Must have enough members to accommodate Length. All parameter numbers must specify axis parameters, or all must specify module parameters.	INT[ ]	N/A
<b>Outputs</b>			
Valid	Indicates valid outputs are available.	LD: flow	0
Busy	Indicates the function block is enabled and has not completed its action.	Other languages: all except constants	1
Warning	Indicates that a warning has occurred within the function block.		0
Error	Indicates that an error has occurred within the function block.		0
ErrorID	Error or warning identification.	WORD	0
Values	Array containing values of the specified parameters. Must have enough members to accommodate Length.	LREAL [ ], DWORD[ ], DINT[ ]	0

## 6.39 MC\_ReadEventQueue

LD	FBD	ST
 <p>LD diagram for MC_ReadEventQueue. The block is titled "MC READEVENTQUEUE" and contains "????". It has three input terminals on the left: "Module", "Execute", and "ReportName". It has five output terminals on the right: "Done", "Busy", "Warning", "Error", and "ErrorID".</p>	 <p>FBD diagram for MC_ReadEventQueue. The block is titled "MC_READEVENTQUEUE" with a "1" in the top right corner. It has four input terminals on the left: "EN", "Module", "Execute", and "ReportName". It has seven output terminals on the right: "ENO", "Module", "Done", "Busy", "Warning", "Error", and "ErrorID".</p>	<p>Formal convention:  [instance name](Module := [input], Execute := [input], ReportName := [input], Done =&gt; [output], Busy =&gt; [output], Warning =&gt; [output], Error =&gt; [output], ErrorID =&gt; [output]);</p>

The event queue contains the last 100 PMM events logged. The queue is a FIFO buffer that contains an ordered list indicating the event sequence that has occurred on the module. In addition to errors and warnings, an informational event is queued at power-up and whenever the hardware configuration changes.

Over a power cycle, the most recent 24 events are preserved in non-volatile memory.

This function block copies the current PMM module event queue as an .ELOG file to the RX3i controller.

The file is saved to a file location specified by the function block's Report Name parameter. The PMM logs errors to the I/O fault table in the CPU if communication with the CPU is available. The PMM logs warnings to the fault table if the PMM has been configured to do so.

Execution type: Immediate execution/deferred response.

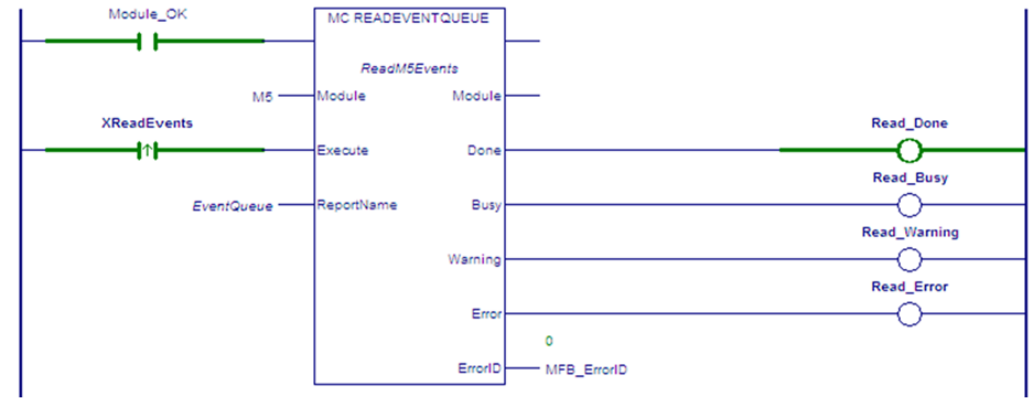
Operands

Instance Variable	Description	Allowed Data Types	Initial Value
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as ??? in LD.)	MC_READEVENTQUEUE	NA
Parameter	Description	Allowed Data Types	Initial Value
Input_Output Parameters			
Module	Module to execute function block	MODULE_REF	N/A
Inputs			
Execute	The rising edge copies the event queue contents to the RX3i CPU.	LD: flow Other languages: all except constants	0
ReportName	Specifies the file reference where the event queue should be reported	EVENTQUEUE_FILE_REF	N/A
Outputs			
Busy	Indicates the function block has been executed and has not yet completed its action.	LD: flow Other languages: all except constants	0
Warning	Indicates that a warning has occurred within the function block.		1
Error	Indicates that an error has occurred within the function block.		0
ErrorID	Indicates the type of error or warning	WORD	0

### 6.39.1 Example

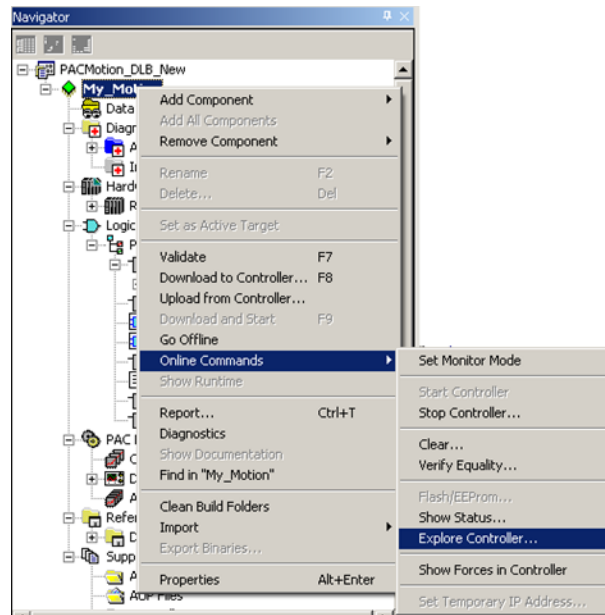
The following example reads the event queue belonging to module M5. When the Execute input transitions ON, the event queue is copied to the CPU.

**Figure 140: MC\_ReadEventQueue Function Block Example**



The ReportName input variable specifies the name of the file, EventQueue.ELOG, that is created in the RX3i controller. To access the report file, browse the files in the controller using the Controller File Explorer.

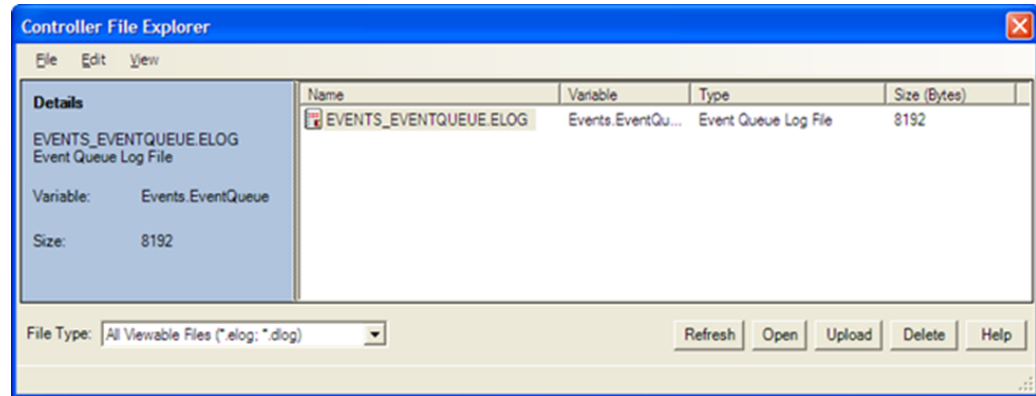
**Figure 141: Accessing the Event Queue Report File**



## Controller File Explorer for ELOG File

To display the Event Queue Log in the InfoViewer, select the .ELOG file and click the Open button.

**Figure 142: Accessing Event Queue Log in InfoViewer**



You can also upload the Event Queue Log to your PC, which creates an HTML version of the file for viewing. For example, if the Report file name is EVENTQUEUE, the uploaded HTML file will be named EVENTQUEUE\_ELOG.html.

The Event Queue Log lists the 100 most recent events by event number. The data for each event includes a time stamp, the module level or axis on which the event occurred, severity, an Event ID, and additional descriptive information. For details on Event Queue entries, refer to Section 9.3, Interpreting Drive Faults and Warnings

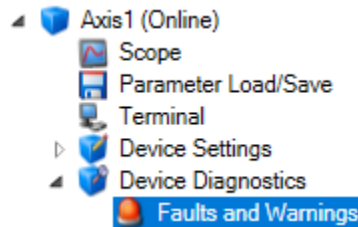
The PMM returns vendor-specific drive faults and warnings, hereafter referred to as drive diagnostics, to the I/O fault table. Vendor-specific drive diagnostics are not returned at the output of a function block. In the I/O fault table, the drive diagnostics are denoted by the Error ID 0xDE.

Diagnostic data denoted by Error ID 0xDE are reported in order of most recent to least recent. Thus, the most recent diagnostic data appears above less recent diagnostics in the I/O fault table, when more than one diagnostic is generated for an event.

The first diagnostic event is reported to the I/O Fault Table. Additional events will not be reported until an MC\_Reset is issued. If desired, all active faults and warnings, along with a fault history, can be obtained via Workbench, by expanding the "Device Diagnostics" tab, then selecting "Faults and Warnings". The most severe active diagnostic is shown on the drive display – codes preceded by 'F' are faults, while codes preceded by 'n' are warnings. For additional information, consult GFK-3168, *PACMotion PSD Installation and User Manual*.

**Note:** the drive itself will respond appropriately to all faults.

Figure 166: Access Diagnostic Information from Workbench the Terminal



## 6.39.2 Drive Faults

The PMM returns vendor-specific drive faults in the form of 16-bit hex numbers. The hexadecimal value can be converted to a decimal number that corresponds to the error table contained in GFK-3168, *PACMotion PSD Installation and User Manual*. The fault code can be found in bytes 6-7 of the Fault Extra Data, corresponding to the entry for vendor-specific drive diagnostic information. A generic EtherCAT drive fault, alerting that an EtherCAT axis is in error, always precedes a vendor-specific drive fault. Drive faults are always logged as error-level events, which must be enabled on the PMM Hardware Configuration settings tab to be logged in the I/O fault table.







## Sample Event Queue Log

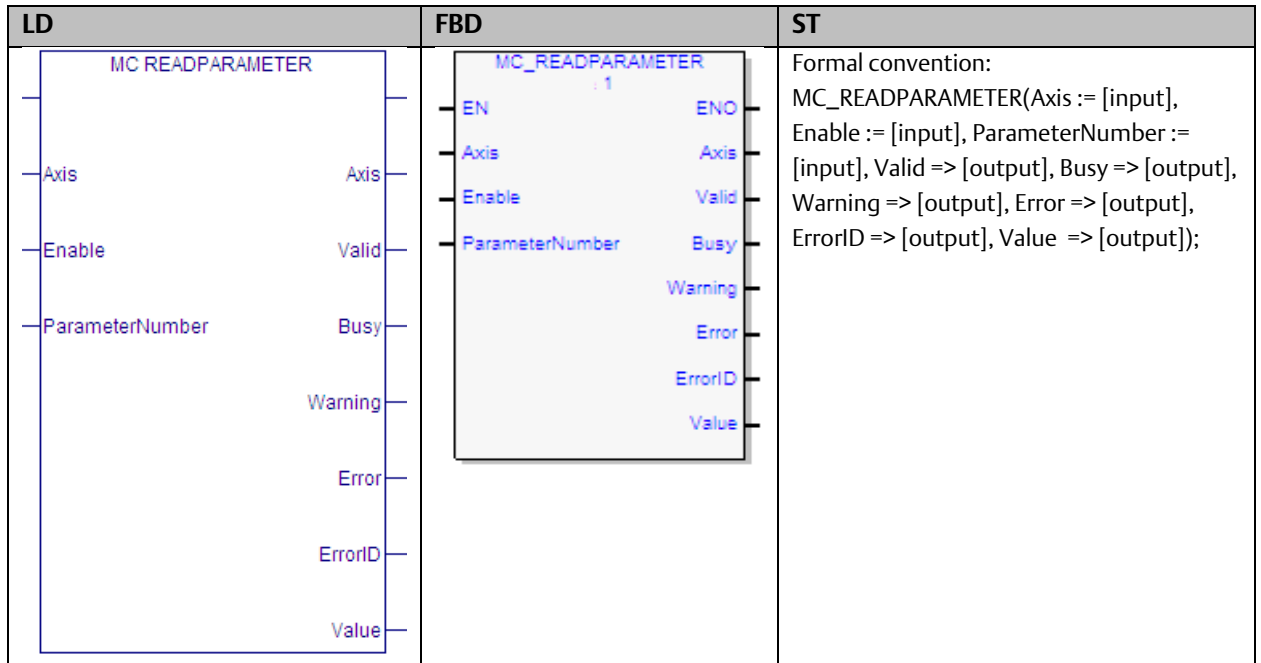
Figure 143: Sample Event Queue Log

ENTS.ELOG

AC Fail Saved Event 
 Informational Event 
 Warning Event 
 Error Event

Number	Relative Time	Event Location	Severity	Event ID	Event Information	Event Data	Response Method	Description
0	0000000000	Axis 1	Informational	0x0343	0x3300	0x0E8A0003	No Stop	Position could not be reached via Direction passed to MC_MoveAbsolute. Direction parameter ignored.
1	0000000000	Axis 1	Informational	0x0319	0x3300	0x0E4E0000	No Stop	Buffering or blending specified with no active command, ignored
2	0000000000	Axis 1	Informational	0x0343	0x3300	0x0E8A0003	No Stop	Position could not be reached via Direction passed to MC_MoveAbsolute. Direction parameter ignored.
23	0000000000	Axis 5	Informational	0x030F	0x3804	0x0A710000	No Stop	PLC mode change aborted function block
24	0000048045	Module	Informational	0x0DC0	0x230F	0x0A4A0008	No Stop	Powerup Event
25	0000593245	Module	Informational	0x0F45	0x2D0F	0x001F0004	No Stop	Assume rack synch mastership
26	0001505132	Axis 1	Informational	0x0330	0x3300	0x0CE80000	No Stop	Function block not allowed in the Disabled state
27	0001692028	Axis 1	Informational	0x0330	0x3300	0x192E0000	No Stop	Function block not allowed in the Disabled state
28	0003099838	Module	Informational	0x0C15	0x210F	0x02B30016	No Stop	Datalog operation aborted due to PLC mode change
29	0003099838	Axis 1	Error	0x530F	0x3820	0x0A780000	Normal Stop	PLC mode change aborted function block
30	0003099838	Axis 2	Informational	0x030F	0x3801	0x0A710000	No Stop	PLC mode change aborted function block

## 6.40 MC\_ReadParameter



This function returns the value of a hardware configuration parameter, which is identified by the Axis and ParameterNumber input parameters. To read a module parameter, specify any valid axis on the module.

For a list of parameter numbers, refer to Axis Parameter Number Index in Section 8.1.1.

If reading Actual Position (PN 1300) or Actual Velocity (PN 10) for a Virtual Axis (Axis 5), valid data will be returned only if an external encoder is used.

Execution type: Immediate execution/immediate response.

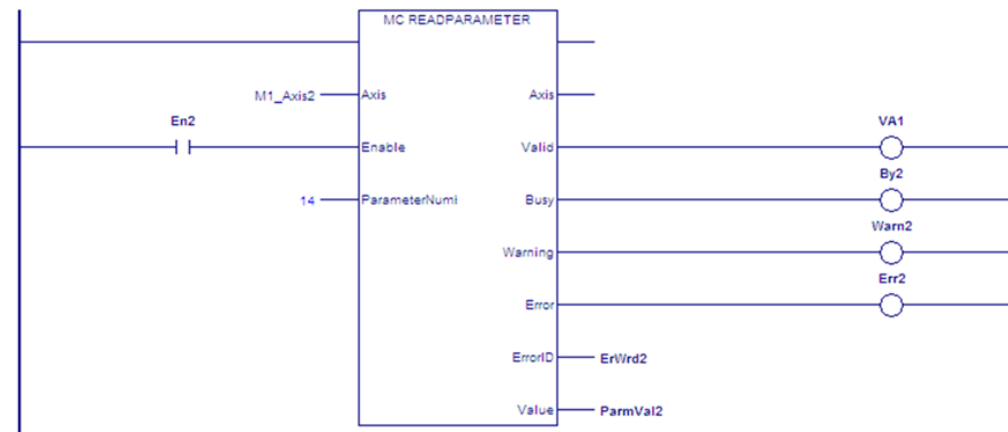
## Operands

Parameter	Description	Allowed Data Types	Initial Value
<b>Input_Output Parameters</b>			
Axis	Axis with parameter to read.	AXIS_REF	N/A
<b>Inputs</b>			
Enable	Read the parameter while enabled.	LD: flow Other languages: all except constants	N/A
ParameterNumber	The parameter to read. This value can be a constant or a mapped variable.	Constant, INT	N/A
<b>Outputs</b>			
Valid	Set to 1 if valid outputs are available	LD: flow	0
Busy	Indicates the function block is enabled and has not completed its action.	Other languages: all except constants	1
Warning	Indicates that a warning has occurred within the function.		0
Error	Indicates that an error has occurred within the function.		0
ErrorID	Error or warning identification.	WORD	0
Value	Value of the specified parameter.	LREAL	0

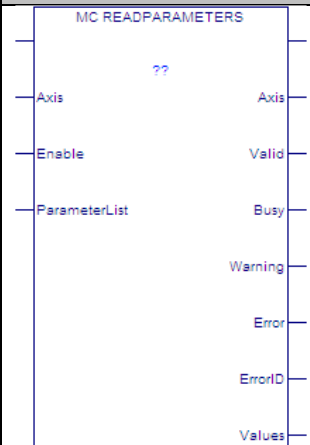
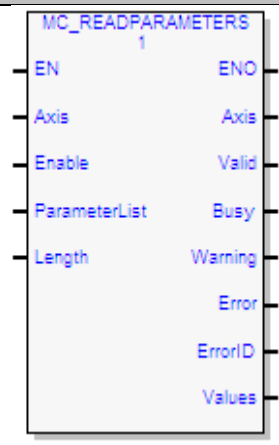
### 6.40.1 Example

In the following example, the MC\_ReadParameter function reads the Deceleration Limit (parameter 14) of the axis named M1\_Axis2 and writes the value to ParmVal2.

**Figure 144: MC\_ReadParameter Function Block Example**



## 6.41 MC\_ReadParameters

LD	FBD	ST
 <p>MC_READPARAMETERS</p> <p>Axis</p> <p>Enable</p> <p>ParameterList</p> <p>Axis</p> <p>Valid</p> <p>Busy</p> <p>Warning</p> <p>Error</p> <p>ErrorID</p> <p>Values</p>	 <p>MC_READPARAMETERS 1</p> <p>EN</p> <p>Axis</p> <p>Enable</p> <p>ParameterList</p> <p>Length</p> <p>ENO</p> <p>Axis</p> <p>Valid</p> <p>Busy</p> <p>Warning</p> <p>Error</p> <p>ErrorID</p> <p>Values</p>	<p>Formal convention: MC_READPARAMETERS(Axis := [input], Enable := [input], ParameterList := [input], Length := [input], Valid =&gt; [output], Busy =&gt; [output], Warning =&gt; [output], Error =&gt; [output], ErrorID =&gt; [output], Values =&gt; [output]);</p>

This function returns the values of up to 16 hardware configuration parameters. The parameters are identified by the Axis and ParameterList input parameters. To read module parameters, specify any valid axis on the module.

For a list of parameter numbers, refer to Axis Parameter Number Index in Section 8.1.1.

If reading Actual Position (PN 1300) or Actual Velocity (PN 10) for a Virtual Axis (Axis 5), valid data will be returned only if an external encoder is used.

Execution type: Immediate execution/immediate response.

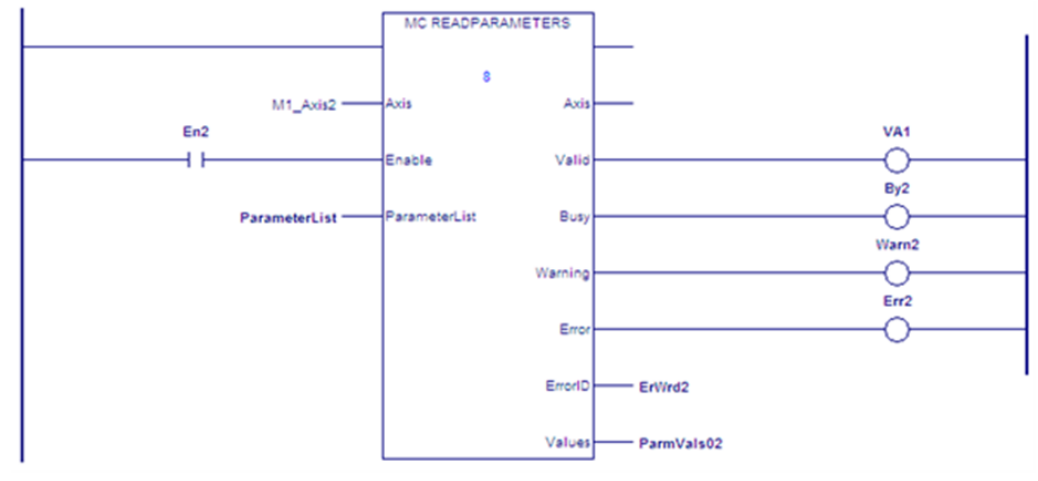
## Operands

Parameter	Description	Allowed Data Types	Initial Value
??	Length. The number of parameter values to read, starting with the first parameter in the ParameterList array (maximum of 16).	Constant	N/A
<b>Input_Output Parameters</b>			
Axis	Axis with parameters to read.	AXIS_REF	N/A
<b>Inputs</b>			
Enable	Read the parameters while enabled.	LD: flow Other languages: all except constants	N/A
ParameterList	Array of parameter numbers. Must have enough elements to accommodate Length. All parameter numbers must specify axis parameters or all must specify module parameters.	INT[ ]	N/A
<b>Outputs</b>			
Valid	Set to 1 if valid outputs are available	LD: flow Other languages: all except constants	0
Busy	Indicates the function block is enabled and has not completed its action.		1
Warning	Indicates that a warning has occurred within the function.		0
Error	Indicates that an error has occurred within the function.		0
ErrorID	Error or warning identification.	WORD	0
Values	Array containing the values of the specified parameters. Must have enough elements to accommodate Length.	LREAL[ ]	0

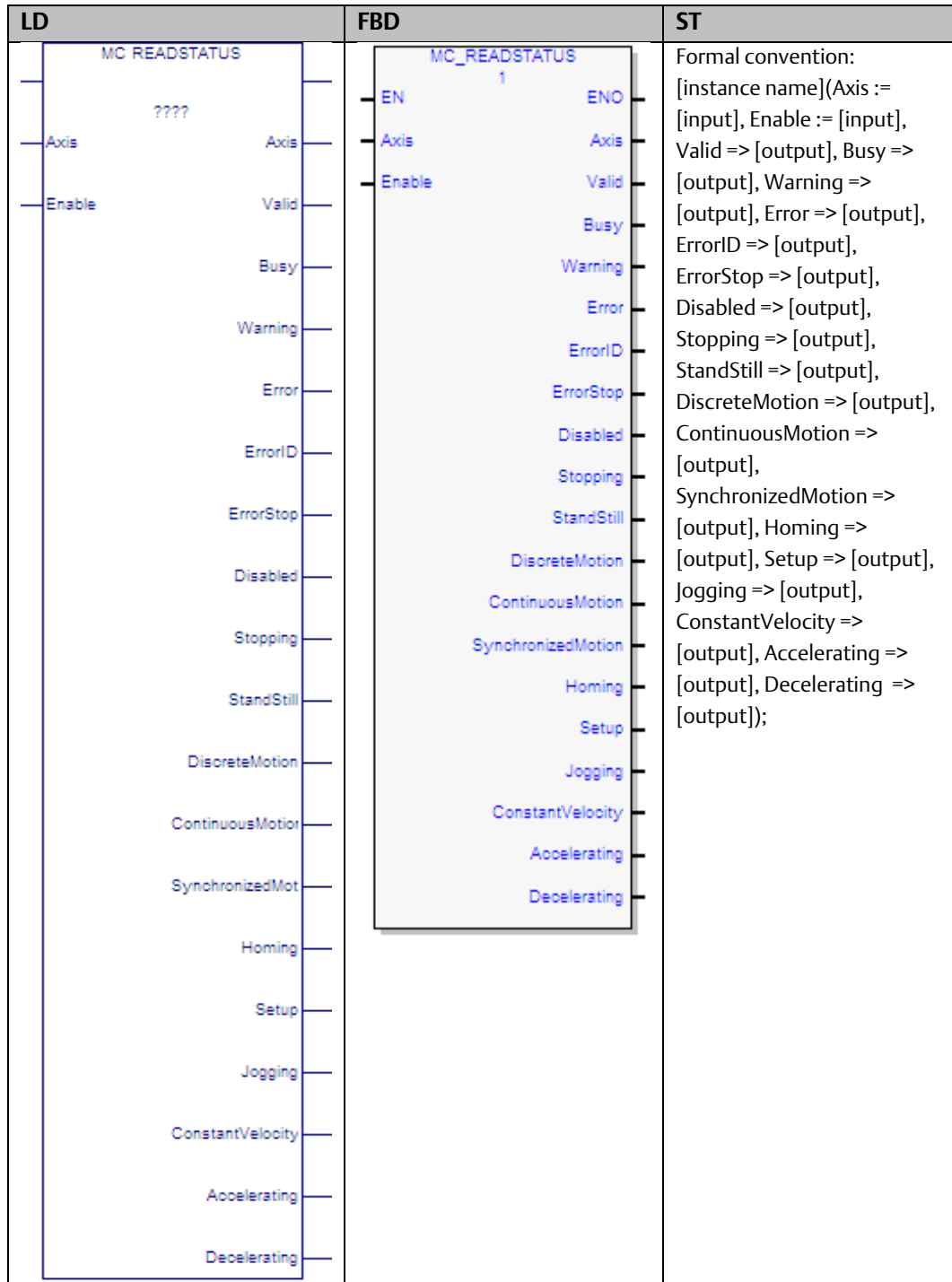
## 6.41.1 Example

In the following example, the MC\_ReadParameters function reads the values of eight parameters, specified by the array variable ParameterList, for the axis named M1\_Axis2. The function writes the parameter values to the array ParmVals02.

**Figure 145: MC\_ReadParameters Function Block Example**



## 6.42 MC\_ReadStatus



This function block returns the current state of the selected axis. For definitions of axis states, refer to Section 5.5 Axis States.

Execution type: Immediate execution/immediate response.



## Operands

Instance Variable	Description	Allowed Data Types	Initial Value
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as ??? in LD.)	MC_READSTATUS	NA
Parameter	Description	Allowed Data Types	Initial Value
<b>Input_Output Parameters</b>			
Axis	The axis on which the function block is to be executed.	AXIS_REF	N/A
<b>Inputs</b>			
Enable	Reads the status of the axis while set to 1.	LD: flow Other languages: all except constants	0
<b>Outputs</b>			
Valid	Set to 1 if valid outputs are available.	LD: flow	0
Busy	Indicates the function block is enabled but has not finished executing.	Other languages: all except constants	1
Warning	Indicates that a warning has occurred within the function block.		0
Error	Indicates that an error has occurred within the function block.		0
ErrorID	Error or warning identification.	WORD	0
ErrorStop	Set to 1 if the axis is in the ErrorStop state.	LD: flow	0
Disabled	Set to 1 if the axis is in the Disabled state.	Other languages: all except constants	0
Stopping	Set to 1 if the axis is currently performing a stop function.		0
Standstill	Set to 1 if the axis is in the Standstill state.		0
DiscreteMotion	Set to 1 if axis is in the Discrete Motion state.		0
ContinuousMotion	Set to 1 if the axis is in the Continuous Motion state.		0
SynchronizedMotion	Set to 1 if the axis is in the Synchronized Motion state.		0
Homing	Set to 1 if the axis is in the Homing state.		0
Setup	Set to 1 if the axis is in the Setup state.		0
Jogging	Set to 1 if the axis is in the Jogging state.		0
ConstantVelocity	Set to 1 if axis is moving at a constant velocity. A velocity of 0 is considered to be constant velocity.		0
Accelerating	Set to 1 if motor is accelerating (increasing energy of the motor).	0	
Decelerating	Set to 1 if motor is decelerating (decreasing energy of the motor.)	0	

## 6.42.1 Axis Status Flags

The following axis status flags can be read as a single DWORD using MC\_ReadDwordParameter(s) or as individual BOOLS using the MC\_ReadBoolParameter(s) functions. The entire 30-bit array of flags can be monitored by reading parameter 1101.

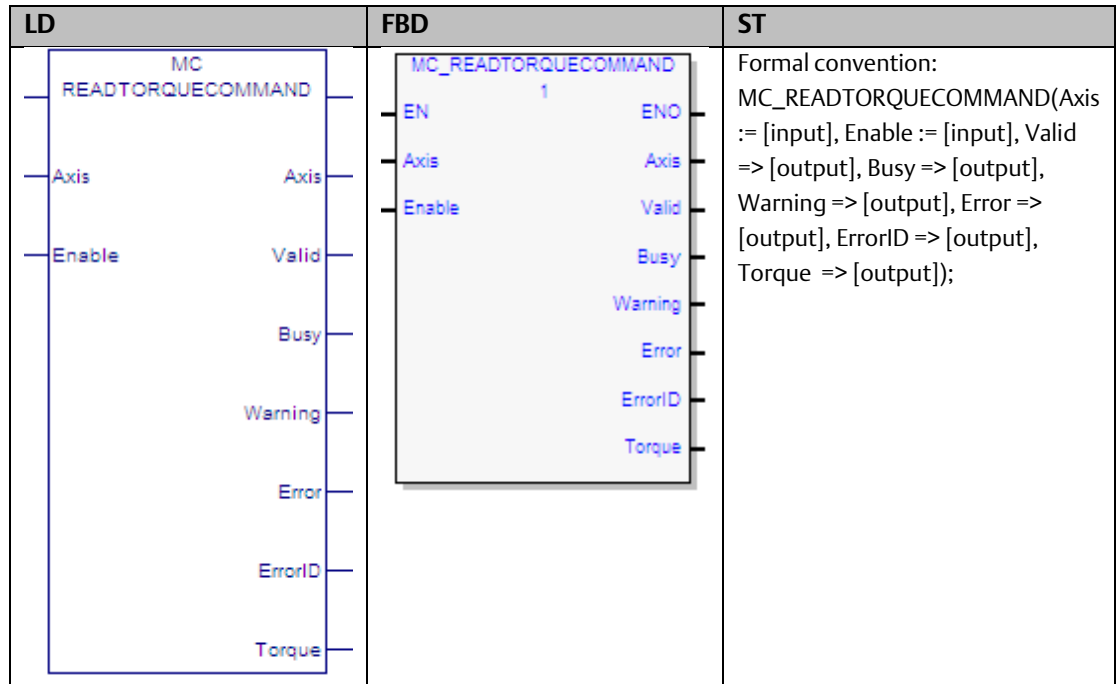
Status	Parameter Number	Flag is set (ON) when:
AxisOK	1200	The axis is ready to receive commands and control a servo. An error condition that stops the axis will turn AxisOK OFF. When AxisOK is OFF, no commands other than MC_Reset will be accepted by the axis. Supported axes: 1–5
PositionValid	1201	A Set Position command or successful completion of a Find Home cycle has initialized the position value. PositionValid must be ON in order to execute a motion program. If the axis is configured to use an absolute feedback digital encoder ( $\beta$ is Series servo with optional encoder battery), PositionValid is set whenever the digital encoder reports a valid absolute position. Supported axes: 1–4 5 Path Gen
DriveEnabled	1202	The Axis has power. DriveEnabled is cleared following power-up or an error condition that stops the axis. Supported axes: 1–4 5 Path Gen
CommandActive	1203	A motion command is executing on the axis. Supported axes: 1–4 5 Path Gen
CommandMoving	1204	The commanded velocity is non-zero and is outside the Command Moving Deadband range specified in HWC. Supported axes: 1–4 5 Path Gen
InZone	1205	Position Error is less than or equal to the In-Position Zone value configured in HWC. Operation of the InZone flag depends only on the Position Error value and is not related to the state of the CommandMoving flag. InZone (ON) can be used in combination with the CommandMoving flag (OFF) to determine when the axis has arrived at its destination. Supported axes: 1–4
MaxPositionLagActive	1206	The absolute value of the position error exceeds the configured <i>MaxPositionLag</i> value and <i>EnablePosLagMonitoring</i> is true. When <i>MaxPositionLagActive</i> is set, Commanded Velocity and Commanded Position are frozen to allow the axis to "catch up" to the Commanded Position. Supported axes: 1–4

Status	Parameter Number	Flag is set (ON) when:
CurrentLimitActive	1207	The commanded current has exceeded the Current Limit setting in HWC.
ServoReady	1208	The servo drive is ready to control motion. Supported axes: 1–4 5 Path Gen
VelocityLimit	1209	The CommandedVelocity (parameter 11) exceeds the MaxVelocitySystem (parameter 8), which is set in HWC. Supported axes: 1–4 5 Path Gen
ErrorStop <sup>9</sup>	1210	An error requiring motion stop has occurred on the axis. Refer refer to Section 5.5, Axis States. Supported axes: 1–4 5 Path Gen
Disabled <sup>10</sup>	1211	The servo drive does not have power. Supported axes: 1–4 5 Path Gen
Stopping <sup>10</sup>	1212	The axis is transitioning from DiscreteMotion, ContinuousMotion or SynchronizedMotion state to Standstill state. Supported axes: 1–4 5 Path Gen
Standstill <sup>10</sup>	1213	The axis is enabled, not moving, and ready to perform motion. Supported axes: 1–4 5 Path Gen
DiscreteMotion <sup>10</sup>	1214	The axis is performing discrete motion. Supported axes: 1–4 5 Path Gen
ContinuousMotion <sup>10</sup>	1215	The axis is performing continuous motion. Supported axes: 1–4 5 Path Gen
SynchronizedMotion <sup>10</sup>	1216	The axis is performing synchronized motion. Supported axes: 1–4 5 Path Gen
Homing <sup>10</sup>	1217	The axis is performing a home cycle. Supported axes: 1–4 5 Path Gen
ConstantVelocity <sup>10</sup>	1218	The axis is moving at a constant velocity. When the axis is at 0 velocity (CommandMoving flag is OFF), the axis is considered to be at ConstantVelocity. Supported axes: 1–4 5 Path Gen
Accelerating <sup>10</sup>	1219	The axis is accelerating (increasing energy of the motor). Supported axes: 1–4 5 Path Gen

<sup>9</sup> Reported by MC\_ReadStatus function block.

Status	Parameter Number	Flag is set (ON) when:
Decelerating <sup>10</sup>	1220	The axis is decelerating (decreasing energy of the motor). Supported axes: 1–4 5 Path Gen
OTPos	1221	Over Travel positive. Over travel limit switch in positive direction is enabled. Default value is set by Over Travel Limit Switch in HWC. Supported axes: 1–4
OTNeg	1222	Over Travel negative. Over travel limit switch in negative direction is enabled. Default value is set by Over Travel Limit Switch in HWC. Supported axes: 1–4
HomeSwitch	1223	Home Switch is enabled. Supported axes: 1–4
FeedbackMoving	1224	The actual velocity is outside the Feedback Moving Deadband range specified in HWC. Supported axes: 1–4 5 External Device
AxisPositioningMode	1225	Axis Positioning Mode: 0 = Linear, 1 = Rotary Supported axes: 1–4 5
Setup <sup>10</sup>	1226	Axis is in a diagnostic or tuning condition. Supported axes: 1–4 5 Path Gen
Jogging <sup>10</sup>	1227	Axis motion is under the control of an MC_JogAxis function block. Supported axes: 1–4 5 Path Gen
AuxPositionValid	1228	Position valid for an external device configured on a virtual axis (Axis 5) or an external device on Axis 1–4 that is not selected as the Position Feedback Source. Set when an MC_SetPosition is executed on Axis 5. May also be set when an MC_SetPosition is executed on the non-position feedback device on an actual axis. For example, if Axis 1 is configured with Motor Encoder as the Position Feedback source, External Quadrature Encoder is configured as the External Device, and an MC_SetPosition specifying External Device on the Encoder input is executed, it will set the AuxPositionValid status on that axis. Supported axes: 1–4 5 External Device
ServoVelocityLimit	1229	The ServoCommandVelocity (parameter 1315) exceeds the Motor Velocity Limit, which is set in HWC. Supported axes: 1–4

## 6.43 MC\_ReadTorqueCommand



This function returns the value of the torque command as long as Enable is set. The Valid output is true when the data output Torque is valid.

---

**Note:** *The value returned by this function is commanded torque, not actual torque.*

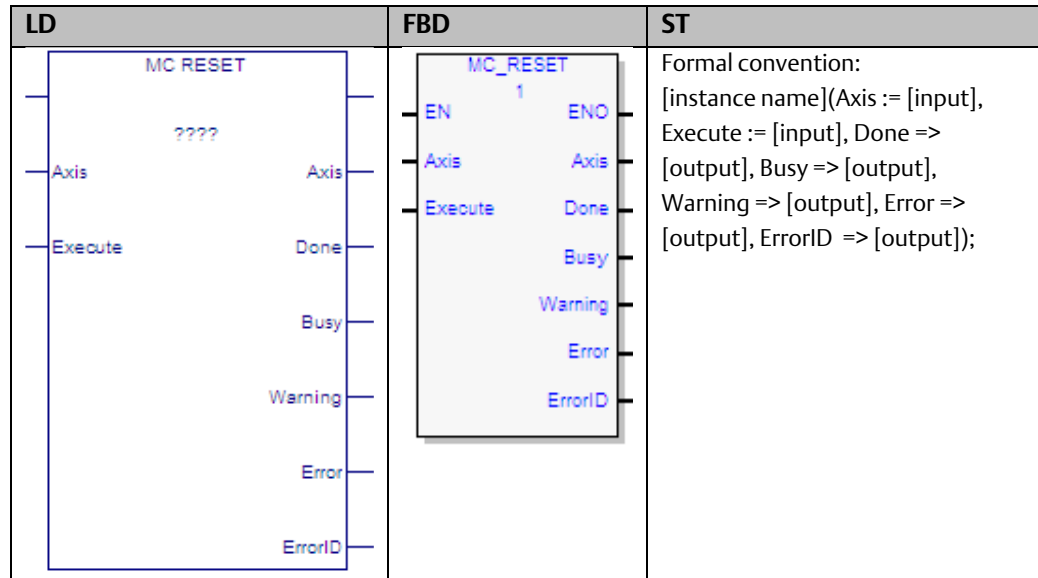
---

Execution type: Immediate execution/immediate response.

## Operands

Parameter	Description	Allowed Data Types	Initial Value
<b>Input_Output Parameters</b>			
Axis	The axis on which the function is to be executed.	AXIS_REF	N/A
<b>Inputs</b>			
Enable	Read the torque command while enabled.	LD: flow Other languages: all except constants	N/A
<b>Outputs</b>			
Valid	Value is available	LD: flow Other languages: all except constants	0
Busy	Indicates Enable is 1 and the function has not finished executing.		1
Warning	Indicates that a warning has occurred within the function.		0
Error	Indicates that an error has occurred within the function.		0
ErrorID	Error or warning identification.	WORD	0
Torque	The value of the commanded torque as a percent of maximum torque (%).	LREAL	0

## 6.44 MC\_Reset



This function block transitions an axis from the state ErrorStop to Standstill by resetting all internal axis related errors. It does not affect the output of the function block instances. Note that this is not a hardware reset.

If an error is not successfully cleared or a new error occurs while clearing the error, the axis remains in the ErrorStop state and the error is returned in the MC\_Reset function block's ErrorID output. In this case, the Done output is not set.

If MC\_Reset is executed in a state other than ErrorStop no action is taken. The Done and Warning outputs are set.

---

**Note:** To clear all errors on a PACMotion module or to clear FTB errors, use the MC\_ModuleReset function block.

---

Execution type: Immediate execution/deferred response.

## Operands

Instance Variable	Description	Allowed Data Types	Initial Value
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as ????) in LD.)	MC_RESET	NA
Parameter	Description	Allowed Data Types	Initial Value
<b>Input_Output Parameters</b>			
Axis	The axis on which the function block is to be executed.	AXIS_REF	N/A
<b>Inputs</b>			
Execute	The rising edge resets the axis.	LD: flow Other languages: all except constants	0
<b>Outputs</b>			
Done	When executed from ErrorStop state indicates Standstill state is reached. If executed from any other state, indicates that no action was taken and a warning was issued.	LD: flow Other languages: all except constants	0
Busy	Indicates the function block has been executed on an axis and has not yet completed its action.		1
Warning	Indicates that a warning has occurred within the function block.		0
Error	Indicates that an error has occurred within the function block.		0
ErrorID	Error or warning identification.	WORD	0



## 6.45 MC\_SetOverride

LD	FBD	ST
		<p>Formal convention:  [instance name](Axis :=  [input], Enable := [input],  VelFactor := 1.0,  AccFactor := 1.0,  JerkFactor := 1.0, Enabled  =&gt; [output], Busy =&gt;  [output], Warning =&gt;  [output], Error =&gt;  [output], ErrorID =&gt;  [output]);</p>

This function block sets override factors for an axis and all functions that are working on that axis. Each override parameter is multiplied by the commanded velocity, acceleration, deceleration and jerk of the move function block. When enabled, the override factors can change continuously. When disabled, the last override factors are used until the next MC\_SetOverride is enabled.

MC\_SetOverride does not influence the state diagram of the axis.

This function block does not act on axes in the Synchronized Motion or Jogging states.

The Enable input on the MC\_SetOverride controls when the override factors that are input to the instance are transferred to the axis.

When the Enable input is high, the Enabled and Busy outputs will be on. The Warning output is on if warning conditions are encountered. The Error output, if on, indicates an error in the instance and will result in the Enabled, Busy and Warning outputs being set off.

When the Enable input transitions off for an instance with valid inputs, all outputs of the instance are cleared.

In the case where the Enable input is on and the inputs of the MC\_SetOverride are changed from valid values to invalid values, the Error and ErrorID outputs of the MC\_SetOverride will be set on and the override will be latched at the last valid value specified by the MC\_SetOverride (as if the Enable had been set off).

In addition, when the Velocity, Acceleration or Jerk input to a motion function block is multiplied by the current override factor, the result must be greater than or equal to the fixed low limits:

$$VEL_{min} = 1/10 \text{ rpm} = 1/600 \text{ rev/sec}$$

$$ACC_{min} = 1/6000 \text{ rev/sec}^2$$

$$\text{JERKmin} = 1/60000 \text{ rev/sec}^3$$

---

**Note:** *The JerkFactor override value is only used for jerk values that specify jerk in Uu/sec<sup>3</sup>. The factor does not apply to jerk that is specified in percent.*

---

Applying new set override values to moves that are currently in progress can cause a particular path to no longer be achievable. For example, if a move is blending high where the second move has the higher velocity, there is a point in the move sequence where a set override that reduces the acceleration factor can make the blending fail. The failure occurs because the move (due to the set override) no longer has the acceleration required to reach the blending position/velocity.

### CAUTION


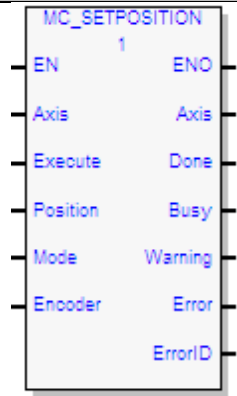
If an axis is in Discrete Motion state, reducing the AccFactor and/or JerkFactor can lead to potential position overshoot.

---

## Operands

Instance Variable	Description	Allowed Data Types	Initial Value
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as ??? in LD.)	MC_SETOVERRIDE	NA
Parameter	Description	Allowed Data Types	Initial Value
<b>Input_Output Parameters</b>			
Axis	Axis that receives function block command.	AXIS_REF	N/A
<b>Inputs</b>			
Enable	Updates the override factors while enabled. The override factor is valid until a new override is set.	LD: flow Other languages: all except constants	0
VelFactor	Velocity Factor: New override factor for the commanded velocity. Range: 0.0, 0.01–1.20 When VelFactor is set to 0.0, the axis stops without going to the Standstill state. (This type of stop is referred to as a feed hold.) Values less than 0.0 are not allowed. VelFactor can be changed at any time and acts directly on the ongoing motion until a new motion command is received. If VelFactor is set to a value greater than 1.0 (100%) that would cause the actual velocity of the axis to exceed the Max Velocity System limit, an error will be generated.	LREAL	1
AccFactor	Acceleration Factor: New override factor for the commanded acceleration. The AccFactor acts on acceleration and deceleration. Range: 0.01–1.20	LREAL	1
JerkFactor	New override factor for the commanded jerk. Range: 0.01–1.20 Only applies to jerk specified in Uu/sec <sup>3</sup> . Does not apply when jerk is specified in percent.	LREAL	1
<b>Outputs</b>			
Enabled	Signals that the override factor(s) is (are) set successfully.	LD: flow Other languages: all except constants	0
Busy	The function block is enabled and has not yet completed its action.		1
Warning	Indicates that a warning has occurred within the function block.		0
Error	Indicates that an error has occurred within the function block.		0
ErrorID	Error or warning identification.	WORD	0

## 6.46 MC\_SetPosition

LD	FBD	ST
 <p>LD diagram showing the MC_SetPosition block. The block is titled "MC SETPOSITION" and contains a "????". The inputs on the left are Axis, Execute, Position, Mode, and Encoder. The outputs on the right are Axis, Done, Busy, Warning, Error, and ErrorID.</p>	 <p>FBD diagram showing the MC_SetPosition block. The block is titled "MC_SetPOSITION" and contains a "1". The inputs on the left are EN, Axis, Execute, Position, Mode, and Encoder. The outputs on the right are ENO, Axis, Done, Busy, Warning, Error, and ErrorID.</p>	<p>Formal convention:  [instance name](Axis := [input], Execute := [input], Position := [input], Mode := [input], Encoder := [input], Done =&gt; [output], Busy =&gt; [output], Warning =&gt; [output], Error =&gt; [output], ErrorID =&gt; [output]);</p>

MC\_SetPosition function block establishes or recalibrates axis position. It can either be used to establish a new coordinate system or shift the axis coordinate system. This function block does not cause any axis movement.

MC\_SetPosition can be used during motion with commanded and actual position moved into the new coordinate system.

MC\_SetPosition can be used in one of two modes, absolute or relative. In absolute mode, the actual position is changed to the value of the Position input with commanded position updated to not cause a change in position error. This establishes a new coordinate system. In relative mode, the actual position and commanded position are incremented by the value of the Position input. This shifts the coordinate system and does not change position error.

The position can be changed for the configured feedback source or for a specific encoder. The feedback source is selected in the hardware configuration as either Motor Encoder or External Device. For a virtual axis, the only feedback source allowed is External Device.

Execution type: Immediate execution/deferred response.

## 6.46.1 Synchronous State Operation

When the axis is in a synchronous state as either the master or slave for a CAM or a gear, MC\_SetPosition operates as follows:

**MC\_SetPosition cannot be used on a slave axis.** Instead, Offsets, Phasing, and Superimposed moves are recommended methods for creating adjustments to a Slave axis.

**MC\_SetPosition on Master Axis:** If the value input for Position causes the position error limit to be exceeded, the slave will generate an error. Small corrections (such as compensating for mechanical realities like slip) should be possible while the CAM is engaged. Use of *Absolute Set Position* is not recommended while a slave is active.

### Operands

Instance Variable	Description	Allowed Data Types	Initial Value
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as ??? in LD.)	MC_SETPOSITION	NA
Parameter	Description	Allowed Data Types	Initial Value
Input/Output Parameters			
Axis	The axis on which the function block is to be executed.	AXIS_REF	N/A
Inputs			
Execute	The rising edge sets the axis position.	LD: flow Other languages: all except constants	0
Position <sup>10</sup>	(UU) In absolute mode, specifies the new actual position of the axis. In relative mode, specifies the distance the actual position/commanded position of the axis is to be incremented.	LREAL	0.0
Mode	Positioning mode. 0 = Absolute 1 = Relative	BOOL	0
Encoder	Identifies the encoder for which the position is changed: Axis Feedback Source, Motor Encoder, or External Device.	MC_ENCODER	AxisFeedback Source
Outputs			
Done	Position has new value.	LD: flow Other languages: all except constants	0
Busy	Indicates the function block has been executed on an axis and has not yet completed its action.		1
Warning	Indicates that a warning has occurred within the function block.		0
Error	Indicates that an error has occurred within the function block.		0
ErrorID	Error or warning identification.	WORD	0

<sup>10</sup> Refer to Valid Values for Position Input Parameter above.

---

**Note:** *Axis 5 consists of a Path Generator and optional External Device. The position must be set independently for each part. When executing MC\_SetPosition for the Path Generator the Encoder input must be Axis Feedback Source. When executing MC\_SetPosition for the External Device the Encoder input must be External Device.*

---

### Valid Values for Position Input Parameter

External Device Low Position Limit  $\leq$  Position  $<$   
(External Device Low Position Limit + External Device Position Range)

or

Motor Encoder Low Position Limit  $\leq$  Position  $<$   
(Motor Encoder Low Position Limit + Motor Encoder Position Range)

## 6.47 MC\_Stop

LD	FBD	ST
		<p>Formal convention:  [instance name](Axis := [input], Execute := [input], Deceleration := [input], Jerk := [input], JerkUnits := [input], BufferMode := [input], Done =&gt; [output], Busy =&gt; [output], Active =&gt; [output], CommandAborted =&gt; [output], Warning =&gt; [output], Error =&gt; [output], ErrorID =&gt; [output]);</p>

This function block commands a controlled motion stop using programmed Deceleration and Jerk values and aborts any ongoing function block execution. It is important to specify a deceleration rate that results in a satisfactory stopping distance.

When MC\_Stop is executed, the axis transitions to the Stopping state and remains in that state as long as the Execute input is true. The axis does not execute any motion generating commands while in the stopping state. When Execute goes false, the axis transitions to the Standstill state.

While the Execute input is true, the axis remains in the Stopping state and does not execute any other commands. With the Done output set, the axis is transferred to the Standstill state.

Execution type: Immediate execution/deferred response.

## Operands

Instance Variable	Description	Allowed Data Types	Initial Value
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as ????) in LD.)	MC_STOP	NA
Parameter	Description	Allowed Data Types	Initial Value
<b>Input_Output Parameters</b>			
Axis	Axis that receives function block command.	AXIS_REF	N/A
<b>Inputs</b>			
Execute	Start the action at rising edge. Axis remains in Stopping state and does not execute additional commands while Execute input remains true.	LD: flow Other languages: all except constants	0
Deceleration	The maximum move deceleration rate. (energy is decreasing) Maximum deceleration is not necessarily reached. (Always positive.)[Units = UU/second <sup>2</sup> ]	LREAL	100,000
Jerk	The Jerk (rate of change in acceleration) used for the move. (Always positive.) If set to 0, jerk will be unlimited. [Units = UU/sec <sup>3</sup> or %]	LREAL For details, refer to Jerk and JerkUnits in Section 5.3.3.	0
JerkUnits	Selects units for Jerk input: (UU/sec <sup>3</sup> or %)	MC_JERKUNITS	0
BufferMode	Defines the axis buffering behavior. Valid modes are Aborting, Buffered, Blending.	MC_BufferMode	0
<b>Outputs</b>			
Done	Zero velocity reached.	LD: flow Other languages: all except constants	0
Busy	Indicates the function block has been executed on an axis and has not yet completed its action.		1
Active	Indicates that the function block has control of the axis.		0
Warning	Indicates that a warning has occurred within the function block.		0
Error	Indicates that an error has occurred within the function block.		0
ErrorID	Error or warning identification.	WORD	0



## 6.48 MC\_SyncStart

LD	FBD	ST
		<p>Formal convention:  [instance name](AxisSize := [input], Execute := [input], SyncTime := [input], AbsoluteStart := [input], Length := [input], Done =&gt; [output], Busy =&gt; [output], Warning =&gt; [output], Error =&gt; [output], ErrorID =&gt; [output]);</p>

The MC\_SyncStart function block identifies axes to be started at the same time and specifies how much time can elapse before the motion must start. The function block is communicated to each of the axes specified and indicates that the next MFB received is the motion that should start in sync. The axes communicate with each other to coordinate the start of motion.

Before synchronously starting motion, each axis verifies that it is in a valid state for performing a motion and that it has received a valid MFB to execute. The axes wait the amount of time specified by SyncTime for all axes to be ready.

Valid axis states for performing motion are: Standstill, DiscreteMotion, ContinuousMotion, and SynchronizedMotion. The current axis state can be read using MC\_ReadStatus.

The valid MFBs are: MC\_MoveAbsolute, MC\_MoveRelative, MC\_MoveAdditive, MC\_MoveSuperImposed, MC\_MoveVelocity, MC\_CamIn, MC\_GearIn, and MC\_GearInPos.

MC\_SyncStart will be aborted if an MC\_Stop, MC\_Reset, MC\_Power, or MC\_ModuleReset function block is executed on an axis or module associated with the MC\_SyncStart instruction.

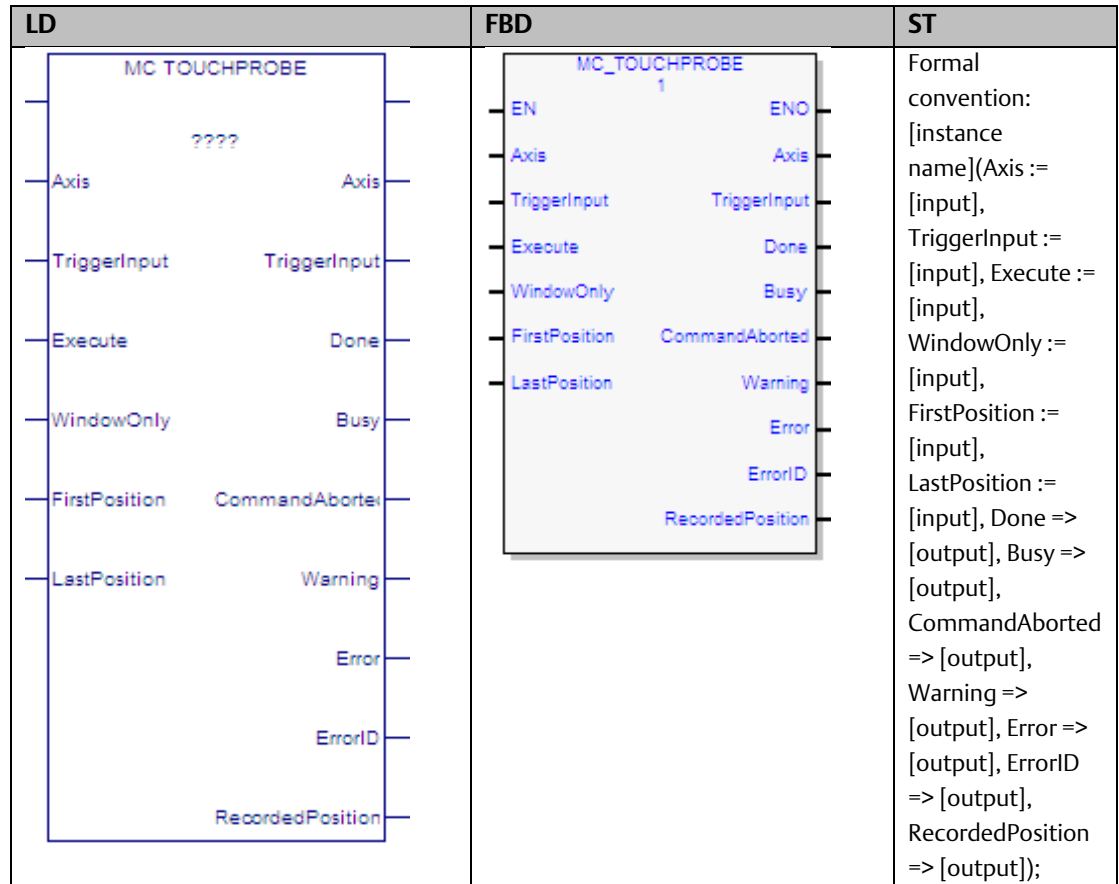
MC\_SyncStart guarantees only that the MFBs are executed at the same time. If an error occurs on an axis as it begins to execute the function block, that axis enters the Errorstop state and all other axes continue the motion.

Execution type: Immediate execution/deferred response.

## Operands

Instance Variable	Description	Allowed Data Types	Initial Value
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as ????) in LD.)	MC_SYNCSTART	NA
Parameter	Description	Allowed Data Types	Initial Value
??	Length. The number of axes to synchronize (maximum of 8).	Constant	1
Input_Output Parameters			
AxisArray	Array of axes to be synchronized (maximum of 8).	AXIS_REF[ ]	N/A
Inputs			
Execute	Execute SyncStart function block	LD: flow Other languages: all except constants	0
SyncTime	The amount of time (in ms) that can elapse between the execution of the SyncStart function block and when the motion must start. If motion is not able to start on all axes in this amount of time an error occurs. The time is specified in milliseconds. Minimum: 5 Maximum: 65535 A value of 0 indicates a time limit of five minutes and is recommended for use during manual testing.	UINT	0
AbsoluteStart	If set to 1 the motion should start at exactly the time specified by Sync Time. If set to 0 the motion will start as soon as can be coordinated.	BOOL	0
Outputs			
Done	Axes have started in sync	LD: flow	0
Busy	Indicates the function block has been executed and has not yet completed its action.	Other languages: all except constants	0
Warning	Signals that warning has occurred within function block		0
Error	Signals that error has occurred within Function block		0
ErrorID	Indicates the type of error or warning	WORD	0

## 6.49 MC\_TouchProbe



The PMM provides the ability to capture the actual axis position in response to a touch probe event.

Up to two touch probe inputs can be monitored per axis. On EtherCAT axes using the PSD family of drives, the touch probe inputs are assigned to drive inputs. The drive input and trigger edge can be configured on the Axis Tab of the PMM345 Hardware Configuration using the Touchprobe x Drive Input and Touchprobe x Detection parameters. On analog or synthetic axes the touch probe inputs are assigned to faceplate or FTB digital inputs.

Touch probe inputs can be configured to trigger on the positive (rising) edge or the negative (falling) edge of the pulse. For faceplate or FTB touch probes the minimum pulse width required is 3µs. When a touch probe input is triggered, the axis actual position is captured. The position capture resolution is ±1 count with an additional 10µs of variance for the touch probe input filter delay. The actual error seen is dependent upon servo acceleration and touch probe input filtering/sampling. For details on acceleration and velocity compensations, refer to Appendix A-1.

The MC\_TouchProbe function block is used to record axis position when a touch probe trigger event occurs. This function block provides an optional windowing feature that specifies upper and lower axis position boundaries for recording the touch probe event.

The function captures a single touch probe event. The function block can be re-executed to record additional touch probe events. Only one instance of MC\_TouchProbe can be executed at a time for each Trigger Input. Additional instances abort the previous instance.

If the windowing option is enabled, the touch probe is recorded only when the axis is between the First Position and Last Position specified.

If operating on a Virtual Axis (Axis 5), valid data will be returned only if an external encoder is used.

The MC\_AbortTrigger function block is used to terminate the operation of an MC\_TouchProbe function block.

Execution type: Immediate execution/deferred response.

## Operands

Instance Variable	Description	Allowed Data Types	Initial Value
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as ??? in LD.)	MC_TOUCHPROBE	NA
Parameter	Description	Allowed Data Types	Initial Value
Input_Output Parameters			
Axis	The axis on which the function block is to be executed.	AXIS_REF	N/A
TriggerInput	Reference to the strobe to be used as the trigger signal source. A value of 1 indicates Touch Probe 1; a value of 2 indicates Touch Probe 2.	TRIGGER_REF	N/A
Inputs			
Execute	The rising edge activates the TouchProbe.	LD: flow Other languages: all except constants	0
WindowOnly	If ON, only use the window to accept trigger events. This option is not supported on EtherCAT axes.	BOOL	0
FirstPosition	Start position from where trigger events are accepted in user units (UU). Value included in window.	LREAL	0
LastPosition	Stop position of the window in user units (UU). Value included in window.	LREAL	0
Outputs			
Done	Strobe position has been captured.	LD: flow Other languages: all except constants	0
Busy	Indicates the function block has been executed on an axis and has not yet completed its action.		1
CommandAborted	Command is aborted by another command		0

Instance Variable	Description	Allowed Data Types	Initial Value
Warning	Indicates that a warning has occurred within the function block.		0
Error	Indicates that an error has occurred within the function block.		0
ErrorID	Error or warning identification.	WORD	0
RecordedPosition	(UU) Actual position where the strobe occurred.	LREAL	0

## 6.49.1 Pre-trigger Replaces Window Only on PSD Drive Family

The windowing mode is not supported on EtherCAT drives. If WindowOnly is enabled MC\_Touchprobe will return an error. Instead the drives support a pre-trigger that can be used as a condition that must be met for the trigger to capture position. One option to replicate the WindowOnly mode is to use a Digital Cam Switch to set the pre-trigger condition. For additional information on configuring the pre-trigger, consult GFK-3168, *PACMotion PSD Installation and User Manual*.

## 6.49.2 Examples

### Touch Probe Operation with Rotary Axis

The following example illustrates the use of the touch probe function to perform a windowing strobe function with a rotary axis.

When the MC\_TouchProbe function block is active and the WindowOnly input is OFF, the first touch probe event will capture the current Axis 1 motor position regardless of its value.

When the MC\_TouchProbe function block is active and the WindowOnly input is ON, the first touch probe event will capture the current Axis 1 motor position only if the current motor position is inside the window specified by the FirstPosition and LastPosition inputs.

### Hardware Configuration

For this example, the Axis 1 TouchProbe 1 function is assigned to the PMM faceplate input FPIN1 on the FP I/O tab and is configured to trigger on the Positive Edge of an incoming pulse.

**Figure 146: Axis 1 TouchProbe 1 Hardware Configuration (FP I/O tab)**

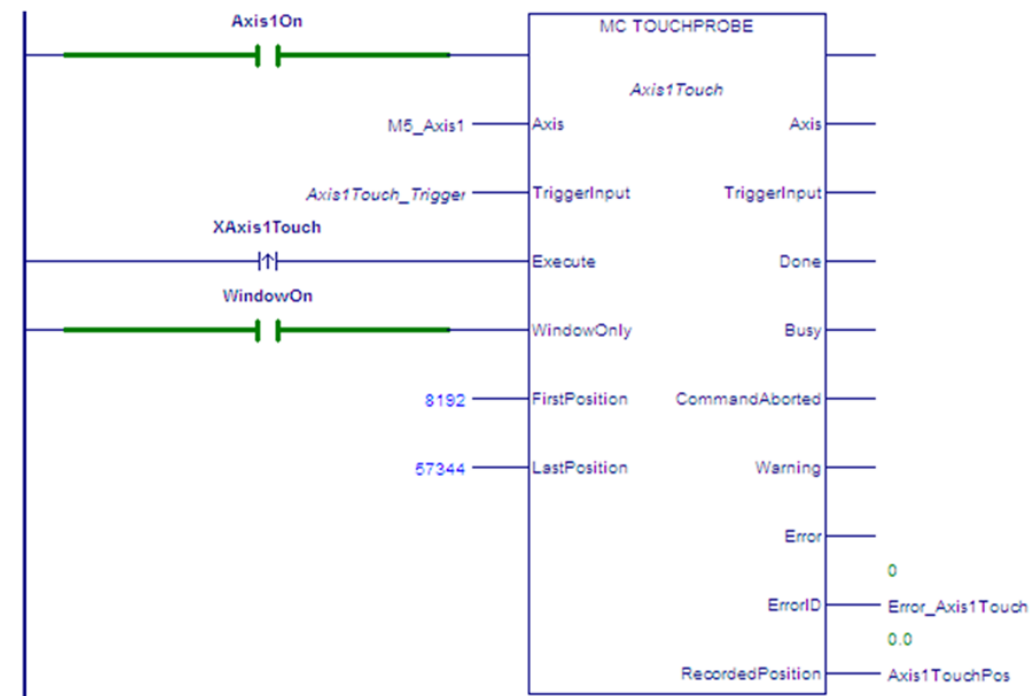
Settings		FP I/O	FTB Inputs	FTB Outputs	I/O Interrupts	Axis 1	Axis 2	Axis 3	Axis 4	Advanced
Parameters						Values				
<i>FPIN1</i>						Axis 2 Touch Probe 1				
FP IN1 Input Ref						M4_FP_IN1_1				
<i>Touch Probe Detection</i>						Positive Edge Trigger				
<i>FPIN1 Open Wire Detect</i>						Disabled				

## MC\_TouchProbe Function Block

The function block acts on Axis 1 of module M5. Once the Execute input transitions high, the MC\_TouchProbe function remains active on Axis 1 until it is terminated by either the TriggerInput becoming active or an MC\_TriggerAbort function block being applied to Axis 1. When executed, the instance data RecordedPosition is initialized to a value of 0.

The TriggerInput is a UINT value that determines which of the two TouchProbe inputs will trigger the position capture. In this case, Axis1Touch\_Trigger is 1, so TouchProbe1 will be used. When WindowOn is on, the axis position is captured only if the TouchProbe1 faceplate input goes high while the axis position is between 8,192 and 57,344 UU.

Figure 147: MC\_TouchProbe Function Block Example



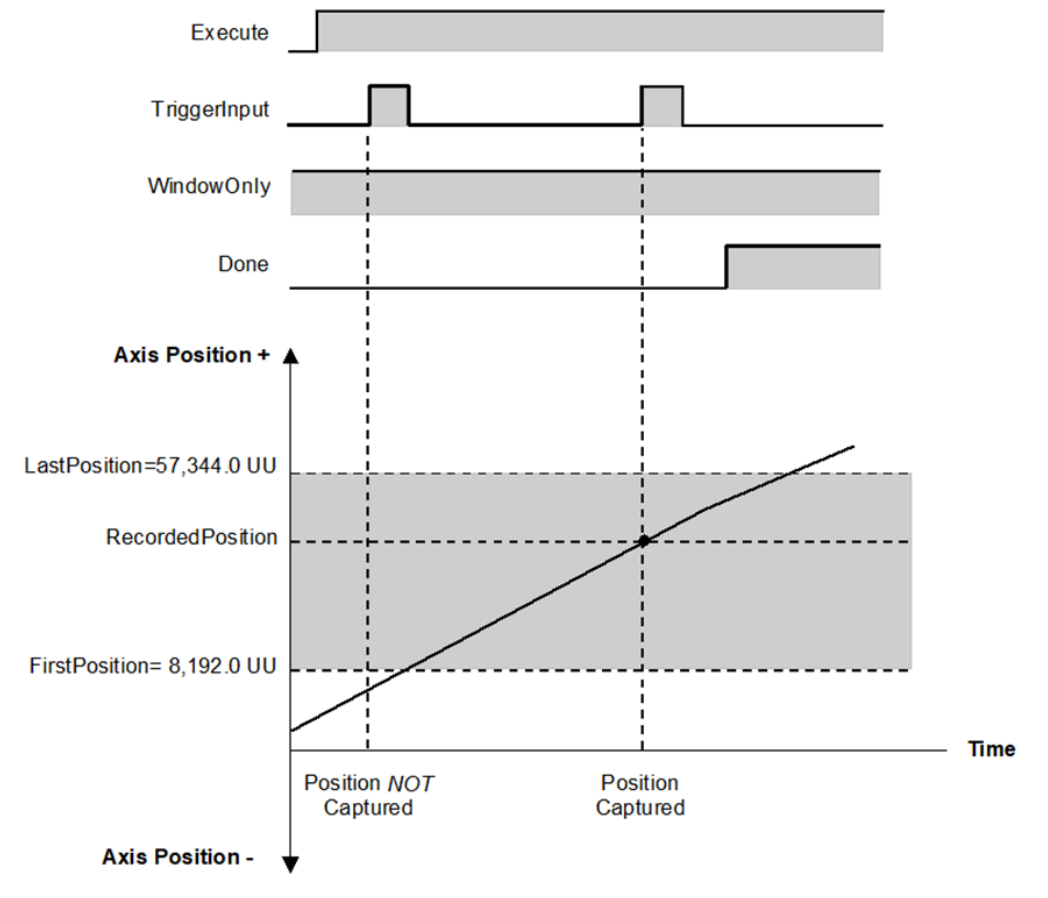
### Timing Diagram for TouchProbe Example with WindowOnly Input On

For both linear and rotary axes, when the WindowOnly input is on, the axis position is captured only if the touch probe event occurs while the axis position is within the window range defined by FirstPosition and LastPosition.

For the purpose of defining the window, positions in the positive direction are greater than those in the negative direction. The window itself is non-directional, meaning that the axis can enter it while moving in either direction.

In the following example, the FirstPosition is less than the LastPosition and the axis is moving in the positive direction. The axis could be rotary or linear. For examples of window operation with other axis modes, refer to Window Operation with Rotary Axes below.

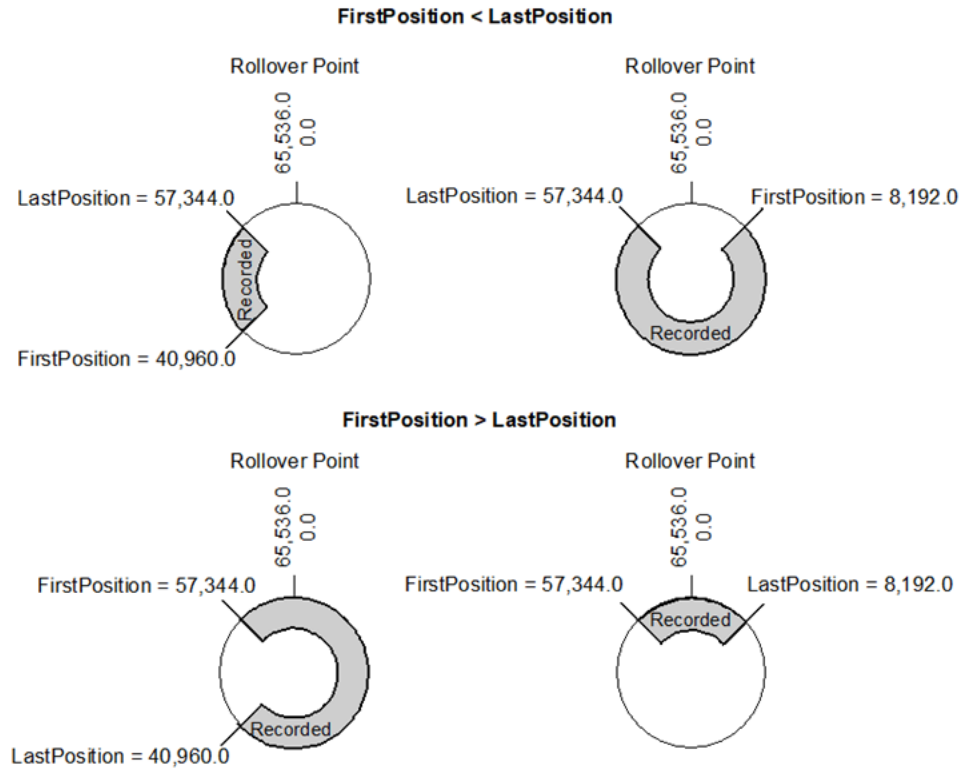
Figure 148: Timing Diagram for Touch Probe Operation (Windowing Enabled)



## Window Operation with Rotary Axes

For these examples, the High Position Limit is 65,536.0. When moving in either direction, the actual position of the axis rolls over at this point.

**Figure 149: Touch Probe Operation with Rotary Axis (Windowing Enabled) Noting Rollover**



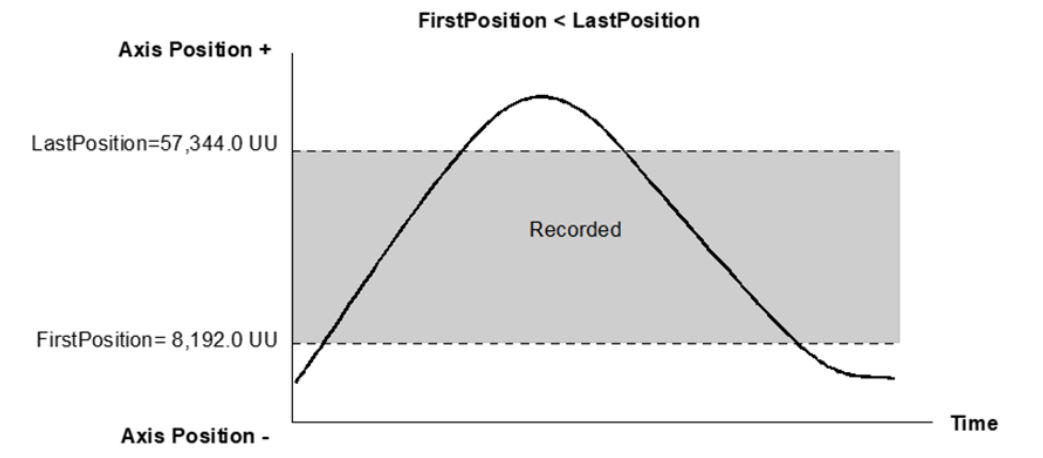
Adapted from the PLCOpen TC2 Task Force Motion Control Part 2 —  
Extensions — V.1.0 PLCOpen — 2002, 2005  
[www.plcopen.org](http://www.plcopen.org)



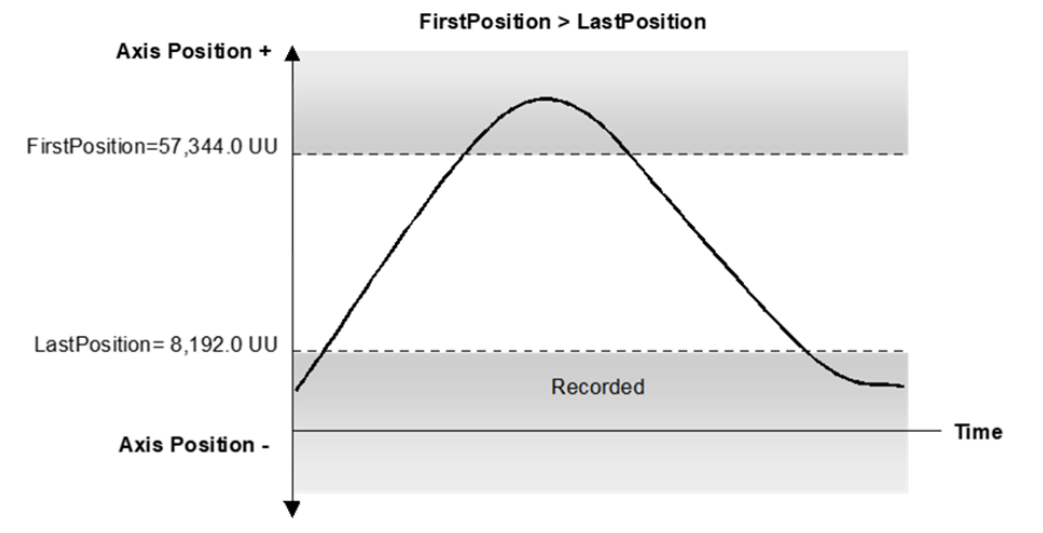
## Window Operation with Linear Axes

Since a linear axis does not roll over, it enters the Recorded window more than once to change direction, as shown in the following example, where the FirstPosition is less than the LastPosition.

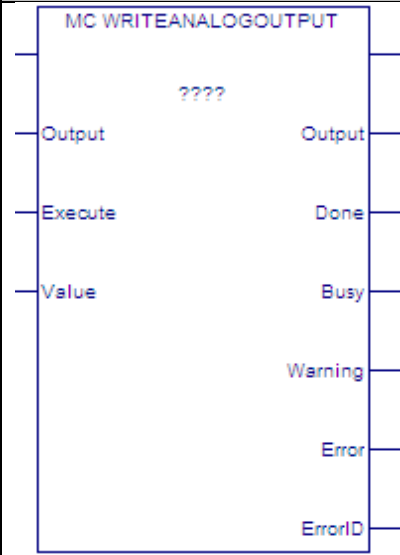
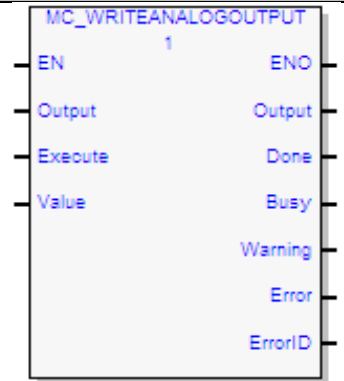
**Figure 150: Touch Probe Operation with Linear Axis (Windowing Enabled) First Position < Last Position**



**Figure 151: Touch Probe Operation with Linear Axis (Windowing Enabled) First Position > Last Position**



## 6.50 MC\_WriteAnalogOutput

LD	FBD	ST
 <p>MC_WRITEANALOGOUTPUT</p> <p>????</p> <p>Output</p> <p>Execute</p> <p>Value</p> <p>Output</p> <p>Done</p> <p>Busy</p> <p>Warning</p> <p>Error</p> <p>ErrorID</p>	 <p>MC_WRITEANALOGOUTPUT</p> <p>1</p> <p>EN</p> <p>Output</p> <p>Execute</p> <p>Value</p> <p>ENO</p> <p>Output</p> <p>Done</p> <p>Busy</p> <p>Warning</p> <p>Error</p> <p>ErrorID</p>	<p>Formal convention: [Instance name](Output := [input], Execute := [input], Value := [input], Done =&gt; [output], Busy =&gt; [output], Warning =&gt; [output], Error =&gt; [output], ErrorID =&gt; [output]);</p>

This function block provides the ability to control the two  $\pm 10\text{Vdc}$  single-ended analog outputs provided by the FTB. It writes a value once to the specified analog output.

The Output is identified by an I/O data reference number, which is passed to the instruction as part of the OUTPUT\_REF input variable. These reference numbers cannot be accessed directly by a Parameter Read or Parameter Write instruction. For a list of I/O reference numbers, refer to Section 8.3 I/O Data Reference Numbers. For specifications and connection details for the analog outputs, refer to Section 3, I/O Wiring, Connections and LED Operation.

Execution type: Immediate execution/deferred response.

## Operands

Instance Variable	Description	Allowed Data Types	Initial Value
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as ??? in LD.)	MC_WRITEANALOGOUTPUT	NA
Parameter	Description	Allowed Data Types	Initial Value
Input_Output Parameters			
Output	Reference ID of the output signal to write to.	OUTPUT_REF defined in the hardware configuration for the module associated with the FTB.	N/A
Inputs			
Execute	The rising edge Writes the value of the selected output.	LD: flow Other languages: all except constants	0
Value	The value to be written to the selected analog output. Units = volts. Range = $\pm 10$ Vdc.	LREAL	0
Outputs			
Done	Writing of the output signal value is done	LD: flow Other languages: all except constants	0
Busy	Indicates the function block has been executed and has not yet completed its action.		1
Warning	Indicates that a warning has occurred within the function block.		0
Error	Indicates that an error has occurred within the function block.		0
ErrorID	Error or warning identification.	WORD	0

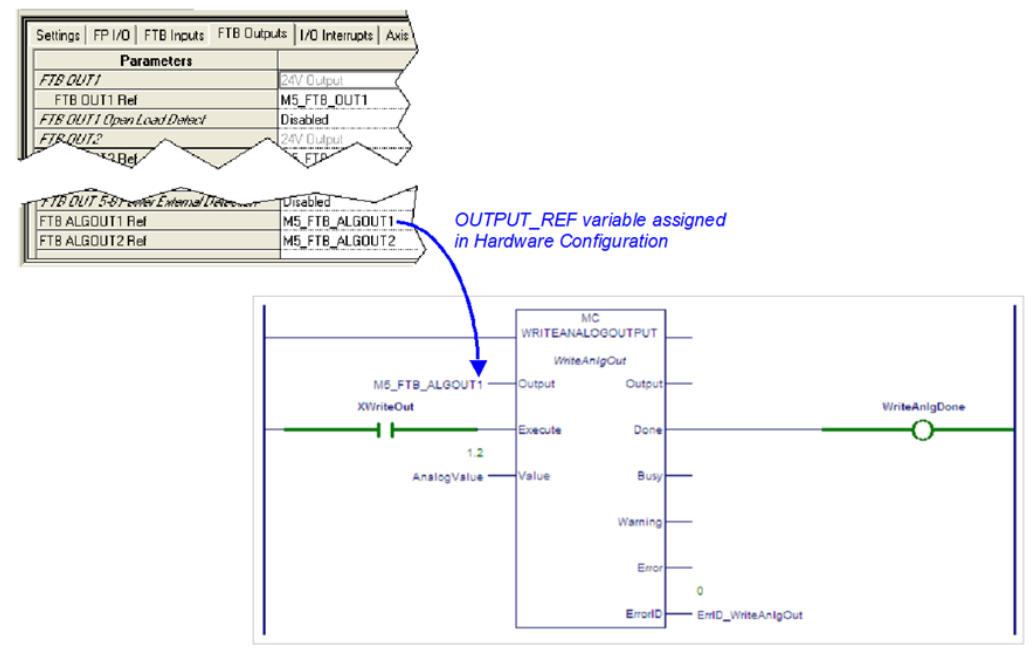
## 6.50.1 Example

In the following example, the OUTPUT\_REF variable M5\_FTБ\_ALGOUT1 has been created on the FTБ Outputs tab in the hardware configuration and assigned to the FTБ analog output ALGOUT1.

When XWriteOut transitions high, the MC\_WriteAnalogOutput function block is executed and the value of AnalogValue is written to the analog output ALGOUT1.

Note that MC\_Write\_AnalogOutput operates only on the rising edge of the Execute input. To perform the write operation again, XWriteOut must first transition low.

**Figure 152: MC\_WriteAnalogOutput Function Block Example**



## 6.51 MC\_WriteBoolParameter

LD	FBD	ST
<p>LD diagram for MC_WRITEBOOLPARAMETER. The block is a rectangle with the title 'MC_WRITEBOOLPARAMETER' at the top. Inside, there is a placeholder '????'. On the left side, there are four input terminals labeled 'Axis', 'Execute', 'ParameterNumber', and 'Value'. On the right side, there are five output terminals labeled 'Axis', 'Done', 'Busy', 'Warning', 'Error', and 'ErrorID'.</p>	<p>FBD diagram for MC_WRITEBOOLPARAMETER. The block is a rectangle with the title 'MC_WRITEBOOLPARAMETER' at the top and a '1' below it. On the left side, there are four input terminals labeled 'EN', 'Axis', 'Execute', and 'ParameterNumber'. On the right side, there are five output terminals labeled 'ENO', 'Axis', 'Done', 'Busy', 'Warning', 'Error', and 'ErrorID'.</p>	<p>Formal convention:  [instance name](Axis := [input], Execute := [input], ParameterNumber := [input], Value := [input], Done =&gt; [output], Busy =&gt; [output], Warning =&gt; [output], Error =&gt; [output], ErrorID =&gt; [output]);</p>

This function block modifies the value of a Boolean axis parameter, which is identified by the Axis and ParameterNumber input parameters. To write a module parameter, specify any valid axis on the module.

For a list of parameter numbers, refer to Axis Parameter Number Index in Section 8.1.1.

Execution type: Immediate execution/deferred response.

## Operands

Instance Variable	Description	Allowed Data Types	Initial Value
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as ??? in LD.)	MC_WRITEBOOLPARAMETER	NA
Parameter	Description	Allowed Data Types	Initial Value
<b>Input_Output Parameters</b>			
Axis	Axis with parameter to write.	AXIS_REF	N/A
<b>Inputs</b>			
Execute	Write the value of the parameter at rising edge.	LD: flow Other languages: all except constants	0
ParameterNumber	The parameter to write. can be a constant or a mapped variable.	INT	0
Value	Boolean value to write to the parameter.	BOOL	0
<b>Outputs</b>			
Done	Parameter successfully written.	LD: flow	0
Busy	Indicates the function block has been executed and has not completed its action.	Other languages: all except constants	1
Warning	Indicates that a warning has occurred in the MFB.		0
Error	Indicates that an error has occurred in the MFB.		0
ErrorID	Error or warning identification.	WORD	0

## 6.52 MC\_WriteBoolParameters

LD	FBD	ST
		<p>Formal convention:  [instance name](Axis :=  [input], Execute :=  [input], ParameterList  := [input], Values :=  [input], Length :=  [input], Done =&gt;  [output], Busy =&gt;  [output], Warning =&gt;  [output], Error =&gt;  [output], ErrorID =&gt;  [output]);</p>

This function block modifies multiple (up to 16) Boolean axis parameter values, which are specified by the Axis and ParameterList inputs.

To write module parameters, specify any valid axis on the module.

For a list of parameter numbers, refer to Axis Parameter Number Index in Section 8.1.1.

**Note:** *If a write fails or is invalid for any of the parameters, none of the parameters are updated.*

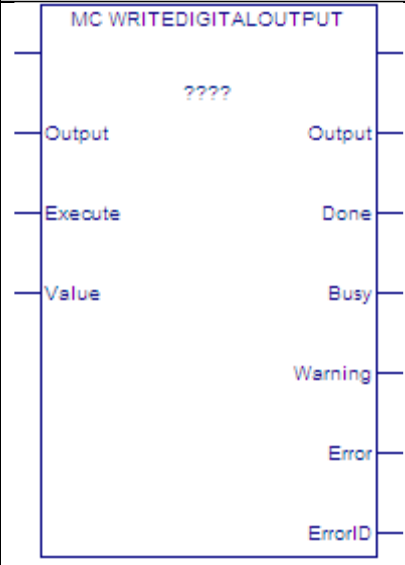
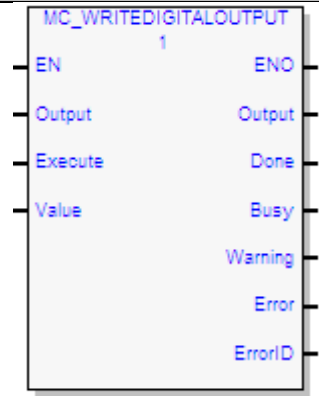
Execution type: Immediate execution/deferred response.

## Operands

Instance Variable	Description	Allowed Data Types	Initial Value
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as ??? in LD.)	MC_WRITEBOOLPARAMETERS	
Parameter	Description	Allowed Data Types	Initial Value
??	Length: The number of parameter values to write, starting with the first parameter in the ParameterList array (maximum of 16).	Constant	
Input_Output Parameters			
Axis	Axis with parameter to write.	AXIS_REF	N/A
Inputs			
Execute	Write the value of the parameter at rising edge.	LD: flow Other languages: all except constants	0
ParameterList	Array of parameter numbers with enough members to accommodate Length.	INT[]	0
Values	New Boolean values of the specified parameters. Must have enough members to accommodate Length.	BOOL[]	0
Outputs			
Done	Parameter successfully written	LD: flow Other languages: all except constants	0
Busy	Indicates the function block has been executed and has not completed its action.		1
Warning	Indicates that a warning has occurred within the function block.		0
Error	Indicates that an error has occurred within the function block.		0
ErrorID	Error or warning identification.	WORD	0



## 6.53 MC\_WriteDigitalOutput

LD	FBD	ST
		<p>Formal convention: [instance name](Output := [input], Execute := [input], Value := [input], Done =&gt; [output], Busy =&gt; [output], Warning =&gt; [output], Error =&gt; [output], ErrorID =&gt; [output]);</p>

This function block writes a value to the discrete FTB or Faceplate output once (with Execute), specified by the Output parameter.

The Output is identified by an I/O data reference number, which is passed to the instruction as part of the OUTPUT\_REF input variable. These reference numbers cannot be accessed directly by a Parameter Read or Parameter Write instruction. For a list of I/O reference numbers, refer to Section 8.3 I/O Data Reference Numbers.

For specifications and connection details for the discrete I/O points, refer to Section 3, I/O Wiring, Connections and LED Operation

Execution type: Immediate execution/deferred response.

## Operands

Instance Variable	Description	Allowed Data Types	Initial Value
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as ??? in LD.)	MC_WRITEDIGITALOUTPUT	NA
Parameter	Description	Allowed Data Types	Initial Value
<b>Input_Output Parameters</b>			
Output	Reference to the signal outputs.	OUTPUT_REF, defined in the module hardware configuration.	N/A
<b>Inputs</b>			
Execute	The rising edge writes the value of the selected output.	LD: flow Other languages: all except constants	0
Value	The value to be written to the selected output.	BOOL	0
<b>Outputs</b>			
Done	Writing of the output signal value is done.	LD: flow	0
Busy	Indicates the function block has been executed and has not yet completed its action.	Other languages: all except constants	1
Warning	Indicates that a warning has occurred within the function block.		0
Error	Indicates that an error has occurred within the function block.		0
ErrorID	Error or warning identification.	WORD	0

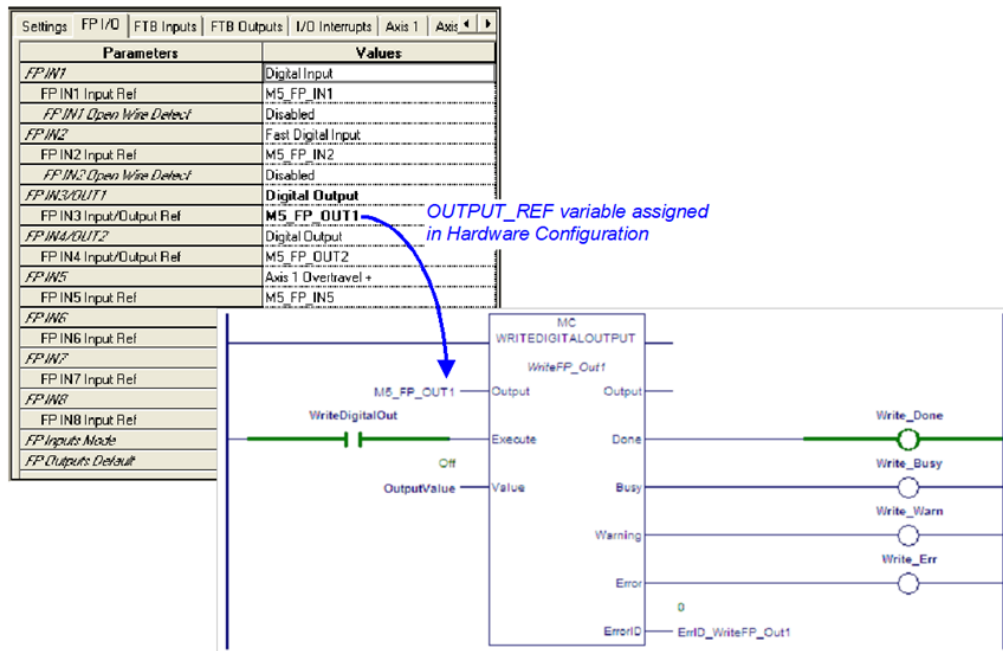
## 6.53.1 Example

In the following example, the value in OutputValue is written to the faceplate output OUT1.

Because each faceplate 24Vdc output shares a terminal with an input point, the terminal must first be configured as a Digital Output. For this example, the OUTPUT\_REF variable M5\_FP\_OUT1 is assigned to the faceplate output point OUT1 in hardware configuration and is input to the MC\_WriteDigitalOutput function block's Output parameter.

When WriteDigitalOut transitions high, the function block is executed and the value of OutputValue is written to the OUT1 faceplate point. Note that MC\_Write\_DigitalOutput it operates only on the rising edge of the Execute input. To perform the write operation again, WriteDigitalOut must first transition low.

**Figure 153: MC\_Write\_DigitalOutput Function Block Example**



## 6.54 MC\_WriteDwordParameters

LD	FBD	ST
<p>MC_WRITEWORDPARAMETERS</p> <p>Axis</p> <p>Execute</p> <p>ParameterList</p> <p>Values</p> <p>Axis</p> <p>Done</p> <p>Busy</p> <p>Warning</p> <p>Error</p> <p>ErrorID</p>	<p>MC_WRITEWORDPARAMETERS</p> <p>1</p> <p>EN</p> <p>Axis</p> <p>Execute</p> <p>ParameterList</p> <p>Values</p> <p>Length</p> <p>ENO</p> <p>Axis</p> <p>Done</p> <p>Busy</p> <p>Warning</p> <p>Error</p> <p>ErrorID</p>	<p>Formal convention: [instance name](Axis := [input], Execute := [input], ParameterList := [input], Values := [input], Length := [input], Done =&gt; [output], Busy =&gt; [output], Warning =&gt; [output], Error =&gt; [output], ErrorID =&gt; [output]);</p>

This function block modifies multiple (up to 16) Dword (32-bit) parameter values, which are specified by the Axis and ParameterList inputs. To write module parameters, specify any valid axis on the module.

This function block is used to write parameters that cannot be expressed as a real value including packed bits. DINT parameters can be written to by changing the Data Type of the variables to DINT, instead of DWORD.

**Note:** *If a write fails or is invalid for any of the parameters, none of the parameters are updated.*

Execution type: Immediate execution/deferred response.

## Operands

Instance Variable	Description	Allowed Data Types	Initial Value
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as ??? in LD.)	MC_WRITEDWORDPARAMETERS	N/A
Parameter	Description	Allowed Data Types	Initial Value
??	Length: The number of parameter values to write, starting with the first parameter in the ParameterList array (maximum of 16).	Constant	N/A
Input_Output Parameters			
Axis	Axis with parameter to write.	AXIS_REF	N/A
Inputs			
Execute	Write the value of the parameter at rising edge.	LD: flow Other languages: all except constants	0
ParameterList	Array of parameter numbers. Must have enough members to accommodate Length.	INT[]	0
Value	New values of the specified parameters. Must have enough members to accommodate Length.	DWORD[ ], DINT[ ]	0
Outputs			
Done	Parameter successfully written	LD: flow Other languages: all except constants	0
Busy	Indicates the function block has been executed and has not completed its action.		1
Warning	Indicates that a warning has occurred within the function block.		0
Error	Indicates that an error has occurred within the function block.		0
ErrorID	Error or warning identification.	WORD	0

## 6.55 MC\_WriteParameter

LD	FBD	ST
		<p>Formal convention:  [instance name]{Axis := [input], Execute := [input], ParameterNumber := [input], Value := [input], Done =&gt; [output], Busy =&gt; [output], Warning =&gt; [output], Error =&gt; [output], ErrorID =&gt; [output]};</p>

This function block modifies the value of an axis parameter, which is identified by the Axis and ParameterNumber input parameters. To write a module parameter, specify any valid axis on the module.

MC\_WriteParameter can be used to change an application limit, which affects axis motion. Application limit parameters are Velocity Limit (MaxVelAppl - PN9), Acceleration Limit (MaxAccelerationAppl -PN13) and Deceleration Limit (MaxDecelerationAppl -PN15).

If you change an application limit for an axis that is executing a move command, the move will be executed using the existing limit. The new limit will be imposed when the next move command is executed on that axis.

Application limits must not exceed the system limits, which are set for each axis in HWC.

For a list of parameter numbers, refer to Axis Parameter Number Index in Section 8.1.1.

Execution type: Immediate execution/deferred response.

## Operands

Instance Variable	Description	Allowed Data Types	Initial Value
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as ??? in LD.)	MC_WRITEPARAMETER	
Parameter	Description	Allowed Data Types	Initial Value
<b>Input_Output Parameters</b>			
Axis	Axis with parameter to write.	AXIS_REF	N/A
<b>Inputs</b>			
Execute	Write the value of the parameter at rising edge.	LD: flow Other languages: all except constants	0
ParameterNumber	The parameter to write. This value can be a constant or a mapped variable.	INT	0
Value	New value of the specified parameter.	LREAL	0
<b>Outputs</b>			
Done	Parameter successfully written.	LD: flow	0
Busy	Indicates the function block has been executed and has not completed its action.	Other languages: all except constants	1
Warning	Indicates that a warning has occurred within the function block.		0
Error	Indicates that an error has occurred within the function block.		0
ErrorID	Error or warning identification.	WORD	0

## 6.56 MC\_WriteParameters

LD	FBD	ST
		<p>Formal convention:  [instance name](Axis := [input], Execute := [input], ParameterList := [input], Values := [input], Length := [input], Done =&gt; [output], Busy =&gt; [output], Warning =&gt; [output], Error =&gt; [output], ErrorID =&gt; [output]);</p>

This function block modifies multiple (up to 16) axis parameter values, which are specified by the Axis and ParameterList inputs. To write module parameters, specify any valid axis on the module.

MC\_WriteParameters can be used to change application limits, which affect axis motion. Application limit parameters are Velocity Limit (9), Acceleration Limit (13) and Deceleration Limit (15).

If you change application limits for an axis that is executing a move command, the move will be executed using the existing limits. The new limits will be imposed when the next move command is executed on that axis.

Application limits must not exceed the system limits, which are set for each axis in HWC.

**Note:** *If a write fails or is invalid for any of the parameters, none of the parameters are updated.*

Execution type: Immediate execution/deferred response.



## Operands

Instance Variable	Description	Allowed Data Types	Initial Value
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as ??? in LD.)	MC_WRITEPARAMETERS	
Parameter	Description	Allowed Data Types	Initial Value
??	Length. The number of parameter values to write, starting with the first parameter in the ParameterList array (maximum of 16).	Constant	N/A
Input_Output Parameters			
Axis	Axis with parameter to write.	AXIS_REF	N/A
Inputs			
Execute	Write the value of the parameter at rising edge.	LD: flow Other languages: all except constants	0
ParameterList	Array of parameter numbers. Must have enough members to accommodate Length.	INT[ ]	0
Values	New values of the specified parameters. Must have enough members to accommodate Length.	LREAL[ ]	0
Outputs			
Done	Parameter successfully written	LD: flow Other languages: all except constants	0
Busy	Indicates the function block has been executed and has not completed its action.		1
Warning	Indicates that a warning has occurred within function block.		0
Error	Indicates that an error has occurred within function block.		0
ErrorID	Indicates the type of error or warning	WORD	0

# Section 7 Electronic CAM Programming

Topics covered:

Section 7.1 Overview of PACMotion CAM Profile Development

Section 7.2 CAM Types and Modes for the PMM

Section 7.3 CAM Operation Restrictions by Type and Mode

Section 7.4 Smoothing and Curve Fitting

Section 7.5 Calculating Slave Axis Velocity and Acceleration

Section 7.6 Synchronized Motion Function Block Status

Section 7.7 CSV CAM File Format

Section 7.8 Reference Memory Format for CAM Files

## 7.1 Overview of PACMotion CAM Profile Development

The CAM profile is typically generated off line with the PACMotion CAM tool in Logic Developer to take advantage of the built-in curve-fit tools and the graphic display. Point data can be fit using 1st, 2nd, 3rd or 5th degree spline curve fitting. A CAM profile (CAM table) can be divided into sectors with a different curve-fit degree for each sector.

In an RX3i target there are two areas where CAM profiles can be stored, the profile library, which is not restricted to size (except disk space on your PC), and the active profiles node, which will store to the RX3i CPU on a project download. There is a limit of 2048 active profiles at a time.

Logic Developer allows .csv import and export of the CAM data profile to the profile library. Many CAD packages' output can be converted into .csv format.

An additional feature of the PACMotion system is that the RX3i CPU can read or write profile information between CPU memory and a PMM module. This function allows a CAM profile created off-line to be pulled into CPU memory and made available for editing via an operator interface then stored back to the PMM for execution.

During a download to the RX3i, a checkbox in Logic Developer will confirm that you want to write the Active Profiles to the RX3i CPU profile library. Once a profile is in the CPU active library it can be selected to operate on any PMM module in the RX3i system using the MC\_CamTableSelect instruction. A maximum of 256 profiles can be selected for a single PMM module.

The profiles stored (selected) to a given PMM are lost if the rack is power cycled, however they remain in the CPU library. If the number of profiles selected to run on a single PMM module exceeds the maximum limit, one of two actions will occur, depending on the CAM Library Management parameter in Hardware Configuration. In Automatic file management mode, the oldest file, in the order of MC\_CamTableSelect executions, on the PMM will be removed and the newly selected profile will take its place. In Manual management mode, the application logic must load and unload the files by name to avoid an overflow error.

Once a profile has been selected it is active in a given PMM module and the MC\_CamIn instruction will start it running.

The separation of MC\_CamTableSelect and MC\_CamIn functions provides initial error checking before the CAM is engaged and allows cams to be switched out on the fly.

Additional checks occur when you engage a profile (with MC\_CamIn), after the profile has been selected. These checks occur on MC\_CamIn because, for a given profile, it could be valid to engage certain master/slave axis pairs but not others

The MC\_CamTableDeselect function block deletes a CAM profile from the specified PMM to free memory. The MC\_CamOut function block disengages a slave axis from the master.

For details on the CAM function blocks, refer to Section 6, PACMotion Instruction Set Reference.

## 7.1.1 Point Limits in CAM Profiles

- CAM profiles containing exactly one linear sector may contain up to 5000 points. This is compatible with the DSM motion controllers.
- CAM profiles using a combination of only linear, quadratic, or cubic sectors (without any quintic sectors) may contain up to 4096 points.
- CAM profiles containing only quintic spline sectors may have up to 2048 points.
- CAM profiles containing quintic spline sectors (with sectors containing first and second derivative data) have a variable maximum point count, which is limited by space. A quintic point pair takes up twice the space as a non-quintic pair. Therefore, the total number of non quintic points plus twice the number of quintic points must be less than 4096.
- $\text{Non-Quintic Points} + 2 * \text{Quintic Points} \leq 4096$

## 7.2 CAM Types and Modes for the PMM

A wide range of applications can be achieved by using the CAM types and modes of CAM operation.

The modes and axis configuration may be changed by the application logic; however, the restrictions of operation must always be adhered to. For a summary of restrictions, refer to Section 7.3, CAM Operation Restrictions by Type and Mode.

Type or Mode	Choices	How Selected	Comments
CAM Profile Type	Linear Cyclic Circular Cyclic Non-Cyclic	CAM Type property of the stored CAM Profile	Assigned when CAM Profile is defined.
CAM Axis Mode	Relative Absolute	StartMode input to the MC_CamIn function block. Refer to Section Start Mode Mask.	The master and slave axis are separately configurable; however, the selected CAM type may constrain which combinations are valid.
CAM Cycle Execution Mode	Periodic Non-periodic.	Periodic input to the MC_CamTableSelect function block.	A periodic CAM continues executing until stopped with the MC_CamOut instruction or an error condition. A non-periodic CAM executes until the master exits the upper or lower limit of the profile.
Axis Position Mode	Linear Rotary	Hardware configuration for the axis and/or via Parameter numbers.	A <b>Linear</b> axis always stays within bounds of position determined by the upper and lower position limit settings. A <b>Rotary</b> axis establishes a new modulus by rolling over at the active high or low position limits. In Rotary mode, the Position Range and Low Position Limit specified for the feedback source will affect the CAM operation.

## 7.2.1 CAM Profile Types

When defining the profiles, the master position data in the CAM profile must monotonically increase. For example, if the first master position is 1 the next master position point has to be > 1. The master source movement may be unidirectional or bi-directional and the slave axis will track it.

CAM profiles can be one of the following types:

1. Non-Cyclic CAM
2. Linear Cyclic CAM
3. Circular Cyclic CAM

Whenever a CAM profile is defined, it is assigned one of these types.

### Non-Cyclic CAM

A Non-Cyclic CAM has a unique non-repeating profile for the whole range of Master position values. The CAM exits when either boundary of the CAM profile is reached. The CAM can also exit if an external event is configured to trigger a conditional Jump. The User Units to Counts ratio specified for the Master and Slave axes when configuring a Non-Cyclical CAM must match the User Units:Counts ratio specified for the corresponding axes in Hardware Configuration. Also, the maximum and minimum position values for the slave and master axes must lie within the High/Low position limits specified for the corresponding axes in Hardware Configuration.

### Linear Cyclic CAM

A Linear Cyclic CAM has a profile that keeps repeating until an event causes it to exit. Furthermore, the numerical and physical end points of the CAM slave axis are the same as the starting point of the cycle. A reciprocating crankshaft is an example of a Linear Cyclic CAM. The User Units to Counts ratio specified for the master and slave axes when configuring a CAM profile must match the User Units per Counts value for the corresponding axes in Hardware Configuration.

**Constraint:** The first and last slave point must be the same for a Linear Cyclic CAM. The CAM Editor will not display the option for Linear Cyclic in the CAM Type field unless this constraint is satisfied by the data in the CAM table.

---

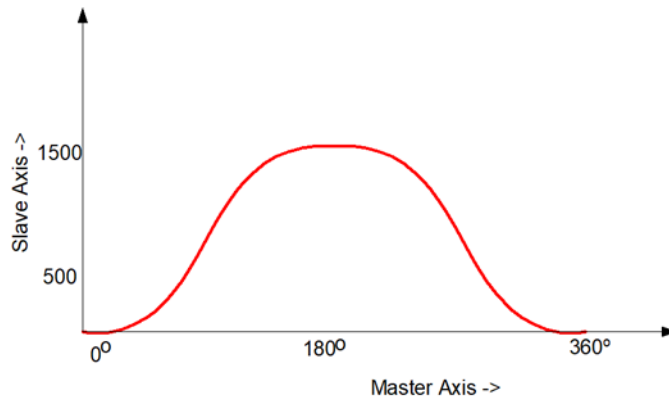
**Note:** For a Cyclic CAM where the Master will be selected as Absolute, the Master Axis Position Range in the Hardware Configuration must be set up according to the master rollover points in the CAM profile. The master axis Low Limit must equal the first master position in the profile. The master High Position Limit must be equal to the Last Master Position in user units. This is because a cyclic profile's first and last point are the same on the physical device.

For a Linear Cyclic CAM, the master axis rolls over at the profile's end points but the slave axis does not. The maximum and minimum profile values for the slave axis must lie within the High/Low limits specified for the corresponding axis in Hardware Configuration.

---

## Linear Cyclic CAM Example

Figure 154:



## Circular Cyclic CAM

A Circular Cyclic CAM has a profile that keeps repeating until an event causes it to exit. Furthermore, a Circular Cyclic CAM has different numerical start and end slave axis positions. Both the master axis and the slave axis roll over at the profile end points. A rotary knife is an example of a Circular Cyclic CAM.

**Constraint:** The **entire** slave profile (including interpolated values) must lie between the minimum and maximum slave position limits, where the minimum and maximum slave limits are defined as follows:

Minimum Slave Value Maximum Slave Value Condition

First Slave Point Last Slave Point Last Point  $\geq$  First Point

Last Slave Point First Slave Point Last Point  $\leq$  First Point

## 7.3 CAM Operation Restrictions by Type and Mode

The following table indicates the valid selections and combinations of CAM profile type and operational modes.

Axis Mode	Axis Position Mode	Axis	Execution Mode	CAM Profile Type		
				Non-Cyclic	Linear-Cyclic	Circular-Cyclic
Absolute	Linear	Master	periodic	Not Allowed. <sup>11</sup>	Not Allowed. <sup>12</sup>	Not Allowed. <sup>12</sup>
			non-periodic	Master Profile end positions must be within or equal to Axis End positions	Master Profile end positions must be within or equal to Axis End positions	Master Profile end positions must be within or equal to Axis End positions
		Slave	periodic	First slave position must match last slave position; All Slave Profile must be between Slave Axis End positions	All Slave Profile must be between Slave Axis End positions	Not Allowed. <sup>12</sup>
			non-periodic	Slave Profile boundary positions must be within or equal to Slave Axis end positions	Slave Profile boundary positions must be within or equal to Slave Axis end positions	Slave Profile boundary positions must be within or equal to Slave Axis end positions
	Rotary	Master	periodic	Master Axis Range must match scaled Master Profile Range; Master Offsets allowed;	Master Axis Range must match scaled Master Profile Range; Master Offsets allowed;	Master Axis Range must match scaled Master Profile Range; Master Offsets allowed.
			non-periodic	Scaled Master Profile Range must be less than or equal to Master Axis Range	Scaled Master Profile Range must be less than or equal to Master Axis Range	Scaled Master Profile Range must be less than or equal to Master Axis Range
		Slave	periodic	Slave Axis Range must A) match scaled Slave Profile range or, B) first slave position must match last slave position; Slave Offsets allowed	Scaled Slave Profile Range must be less than or equal to the Slave Axis Range; Slave Offsets allowed	Slave Axis Range must be equal to the Scaled Slave Profile Range; Slave Offsets allowed
			non-periodic	Scaled Slave Profile Range must be less than or equal to Slave Axis Range; Slave Offsets allowed	Scaled Slave Profile Range must be less than or equal to Slave Axis Range; Slave Offsets allowed	Scaled Slave Profile Range must be less than or equal to Slave Axis Range; Slave Offsets allowed

<sup>11</sup> This combination of inputs to MC\_CamTableSelect is allowed. In general, these types of checks occur when you engage a profile (with MC\_CamIn), after the profile has been selected. These checks occur on MC\_CamIn because, for a given profile, it could be valid to engage on certain master/slave axis pairs but not on others



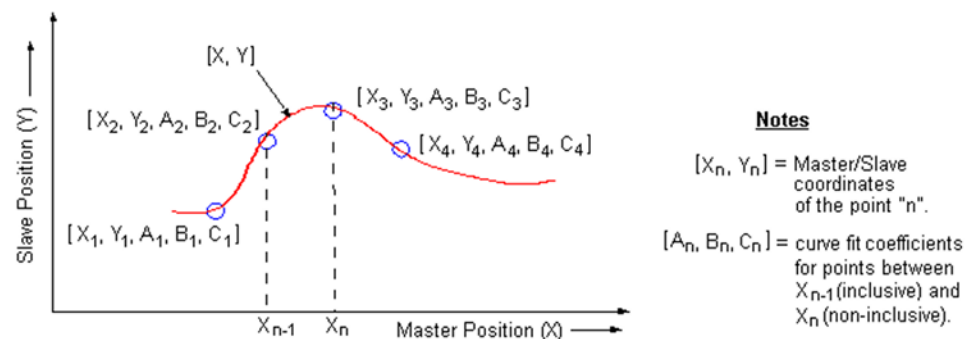
Relative	Linear	Master	periodic	Master Profile End positions must be within Master Axis End positions;	Master Profile end positions must be within Master Axis end positions;	Master Profile end positions must be within Master Axis end positions;
			non-periodic	Master Profile End positions must be within Master Axis End positions	Master Profile End positions must be within Master Axis End positions	Master Profile End positions must be within Master Axis End positions
		Slave	periodic	Slave Profile Boundary positions must be within Slave Axis End positions.	Slave Profile Boundary positions must be within Slave Axis End positions.	Slave Profile Boundary positions must be within Slave Axis End positions.
			non-periodic	Slave Profile Boundary positions must be within Slave Axis End positions.	Slave Profile Boundary positions must be within Slave Axis End positions.	Slave Profile Boundary positions must be within Slave Axis End positions.
	Rotary	Master	periodic	No Limitations	No Limitations	No Limitations
			non-periodic	No Limitations	No Limitations	No Limitations
		Slave	periodic	No Limitations	No Limitations	No Limitations
			non-periodic	No Limitations	No Limitations	No Limitations

## 7.4 Smoothing and Curve Fitting

The CAM editor employs spline polynomial curve fitting to define segments, which are regions of a profile that fall between user-defined points. This approach reduces the memory required for profile storage on the motion module while providing accurate and smooth motion trajectories. Without this smoothing scheme, a large number of data points, requiring a large amount of memory, would be needed to define each profile.

A CAM profile is defined with a minimum number of actual data points. After these points are defined, they are grouped into sectors; a profile is composed of one or more sectors. For each sector, you specify the curve-fit degree (1, 2, 3 or 5). The higher the degree, the smoother the curve-fit is. The curve-fit degree is the degree of the polynomial curves used to define the regions of the sector not specified by user defined points. Unique curve-fit polynomial coefficients are generated for each segment of a sector (that is, between each pair of user-defined points). The coefficients of the polynomials are calculated to include the user-defined points and to match the slope of the profile on either side of a user-defined point (except for 1st and 5th degree sectors).

**Figure 155: Spline Polynomial Curve Fitting**



The polynomial curves for a position profile are described by the following function:

$$Y(X) = A_{n-1}(X_n - X_{n-1})^5 + B_{n-1}(X_n - X_{n-1})^4 + C_{n-1}(X_n - X_{n-1})^3 + D_{n-1}(X_n - X_{n-1})^2 + E_{n-1}(X_n - X_{n-1}) + Y_{n-1}$$

Where:

$Y$  = slave position value for a master position  $X$ .

$X_{n-1}$  = master position value at point  $n-1$ .

$Y_{n-1}$  = slave position value at point  $n-1$ .

$A_{n-1}, B_{n-1}, C_{n-1}, D_{n-1}, E_{n-1}$  = curve-fit coefficients at point  $n-1$ .

**Note:** For a given master position  $X$ , that lies between  $X_{n-1}$  and  $X_n$ , the coefficients  $A, B, C, D$ , and  $E$  are selected for the point corresponding to  $X_{n-1}$ .

For a second-degree curve-fit, the  $A, B$ , and  $C$  coefficients are zero, and for a first-degree curve-fit, the  $A, B, C$ , and  $D$  coefficients are zero.

## 7.5 Calculating Slave Axis Velocity and Acceleration

When the profile is defined in the PME CAM Editor, graphs called Velocity and Acceleration are available. However, these are equivalent to the slave axis velocity and acceleration when the master is moving a constant velocity of 1 user-unit/second. In most applications, this is not the case. The operating slave's velocity and acceleration are functions of the master velocity and acceleration. The graphs are better considered as the first and second derivatives of the profile function.

The CAM Profile (or CAM Curve) defines the position of a slave with respect to the position of its master.

$$f \equiv \text{CamCurve}; \quad x \equiv \text{MasterPsn}; \quad y \equiv \text{SlavePsn}$$

$$y = f(x)$$

The behavior of a master axis can be described as a function of time, which means the slave axis can also be described as a function of time. The equations below detail the relationship.

$$t \equiv \text{time}; \quad \frac{df}{dx} \equiv \text{profile first derivative}; \quad \frac{dx}{dt} \equiv \text{master velocity}$$

$$\frac{d^2 f}{dx^2} \equiv \text{profile second derivative}; \quad \frac{d^2 x}{dt^2} \equiv \text{master acceleration}$$

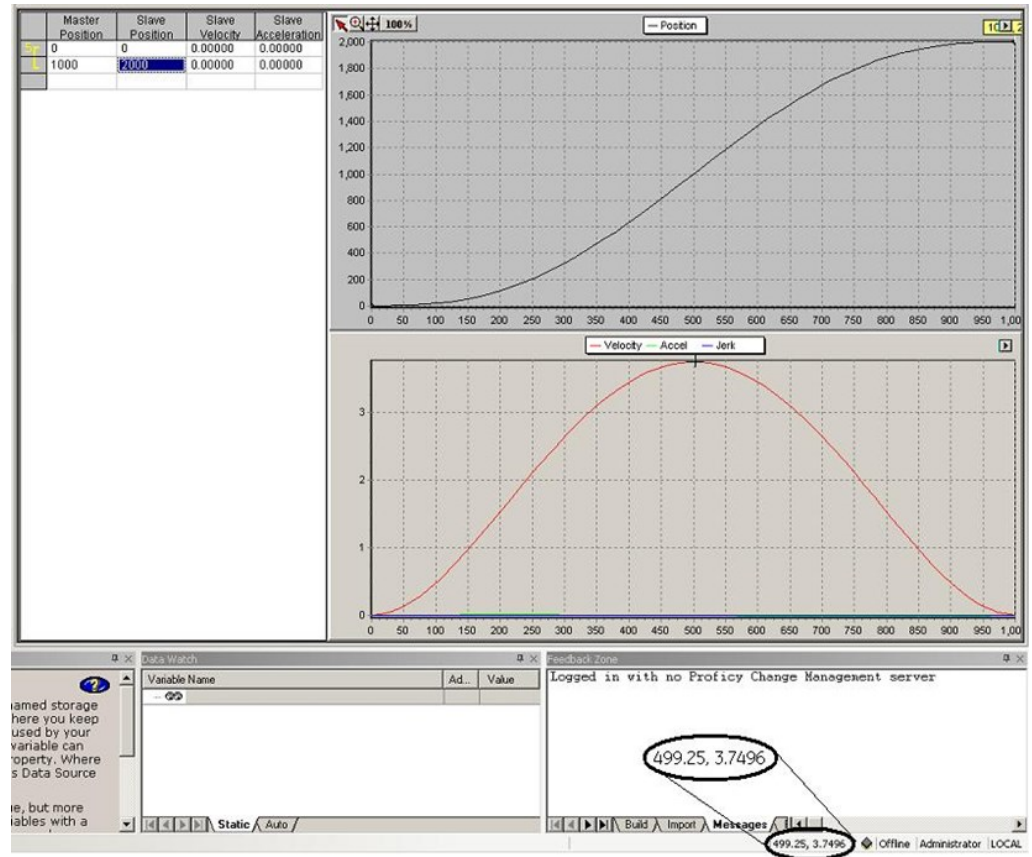
$$\text{Position} \Rightarrow y(t) = f(x(t))$$

$$\text{Velocity} \Rightarrow \frac{dy}{dt} = \frac{df}{dx} \frac{dx}{dt}$$

$$\text{Acceleration} \Rightarrow \frac{d^2 y}{dt^2} = \frac{df}{dx} \frac{d^2 x}{dt^2} + \frac{d^2 f}{dx^2} \left[ \frac{dx}{dt} \right]^2$$

If the master axis is moving at a constant velocity, the peak slave axis velocity is easy to determine. Find the peak of Velocity graph, which gives you the profile first derivative. Then multiply that by the master velocity to get the slave velocity. In the screen shot below, the cursor is positioned at the peak of the Velocity graph and the x-y cursor positions are displayed in the bottom right (inside the oval). If the master axis units are programmed in revolutions and it travels at 1200RPM, it is moving at 20 revs/sec. Letting the slave axis be programmed in inches, the slave velocity at master position 499.25 is  $20 * 3.7496 \sim 5.3$  inches/sec.

Figure 156: Slave Position and Velocity when Following a Constant-Velocity Master



## Blending Sectors

The process applied to blend adjacent sectors depends on their curve-fit degrees. The following descriptions cover the possible combinations.

Of the four curve-fit degrees allowed, two of the methods, 1st-degree and 5th-degree, are completely defined by the data provided, so there are no boundaries to set. Henceforth these are denoted as complete sectors.

2nd-degree sectors have one boundary, which is set as follows:

For Non-Cyclic profiles, if the sector is the first sector, the start boundary is set to the value chosen for the Starting First or Second Derivative.

For both Cyclic and Non-Cyclic types, for internal sectors, if a 2nd-degree sector follows a complete sector, the starting first derivative of the 2nd-degree is set equal to the ending first derivative of the complete sector.

If a Quadratic sector follows another 2nd-degree sector, the sectors are built as if they were one sector.

If a 2<sup>nd</sup>-degree sector follows a 3<sup>rd</sup>-degree sector and a complete sector follows the 2<sup>nd</sup>-degree sector, the ending first derivative of the 2<sup>nd</sup>-degree sector is set equal to the starting first derivative of the complete sector.

If a 2<sup>nd</sup>-degree sector follows a 3<sup>rd</sup>-degree sectors and is either the last sector is followed by a 3<sup>rd</sup>-degree sector, the starting first derivative of the 2<sup>nd</sup>-degree sector is set by fitting a quadratic using three points, with the 2<sup>nd</sup>-degree sector's first point in the middle.

3<sup>rd</sup>-degree sectors have two boundaries, which are set as follows:

If the sector is the first sector, the start boundary is set to the value chosen for the Starting First or Second Derivative. If the sector is the last sector, the end boundary is set to the value chosen for the Ending First or Second Derivative. If there is only one sector, the Starting and Ending Boundaries will be set.

For internal sectors, if a 3<sup>rd</sup>-degree sector follows any non-3<sup>rd</sup>-degree sector, the starting first derivative is set equal to the ending first derivative of the preceding sector. If a 3<sup>rd</sup>-degree sector follows a 3<sup>rd</sup>-degree sector, the sectors are built as if they were one sector. If a 3<sup>rd</sup>-degree sector is followed by any non-3<sup>rd</sup>-degree sector, the ending first derivative is set equal to the starting first derivative of the following sector.

## Boundary Conditions

For non-cyclic profiles, it may be necessary to define some condition at the start and end of the profile to calculate curve-fit polynomial coefficients. If the first sector of the profile is quadratic or cubic, a start condition is required. If the last sector of a profile is cubic and end condition is required.

For quadratic sectors, the start boundary condition can be:

- The numerical value of the 1st derivative (Slope) of the profile.
- The numerical value of the 2nd derivative of the profile.
- Based on a default calculation.

For cubic sectors, the end boundary condition can also be set.

The default calculations are as follows:

- **Start Boundary.** The slope at the start point of the profile is calculated by temporarily fitting a polynomial curve to the first three (2<sup>nd</sup> degree sector) or four points (3<sup>rd</sup> degree sector) and calculating the slope of the temporary polynomial at the first point.
- **End Boundary.** The slope at the end point of the profile is calculated by temporarily fitting a polynomial curve to the last four points (3<sup>rd</sup> degree sector) and calculating the slope of the temporary polynomial at the end point.

## 7.6 Synchronized Motion Function Block Status

The Synchronized Motion Function blocks, MC\_CamIn and MC\_GearInPos can have the following statuses: they can be Pending, Ramping, or InSync. The MC\_GearIn function block status is never Pending, but can be Ramping or InGear.

For details on the operation of these function blocks, refer to Section 6, PACMotion Instruction Set Reference.

### 7.6.1 Pending

Synchronized MFBs: MC\_GearInPos and MC\_CamIn

A MC\_CamIn or MC\_GearInPos MFB is Pending when it is Busy but is not Active. While the function block is Pending, any previous motion on the axis continues.

The function block is no longer Pending when:

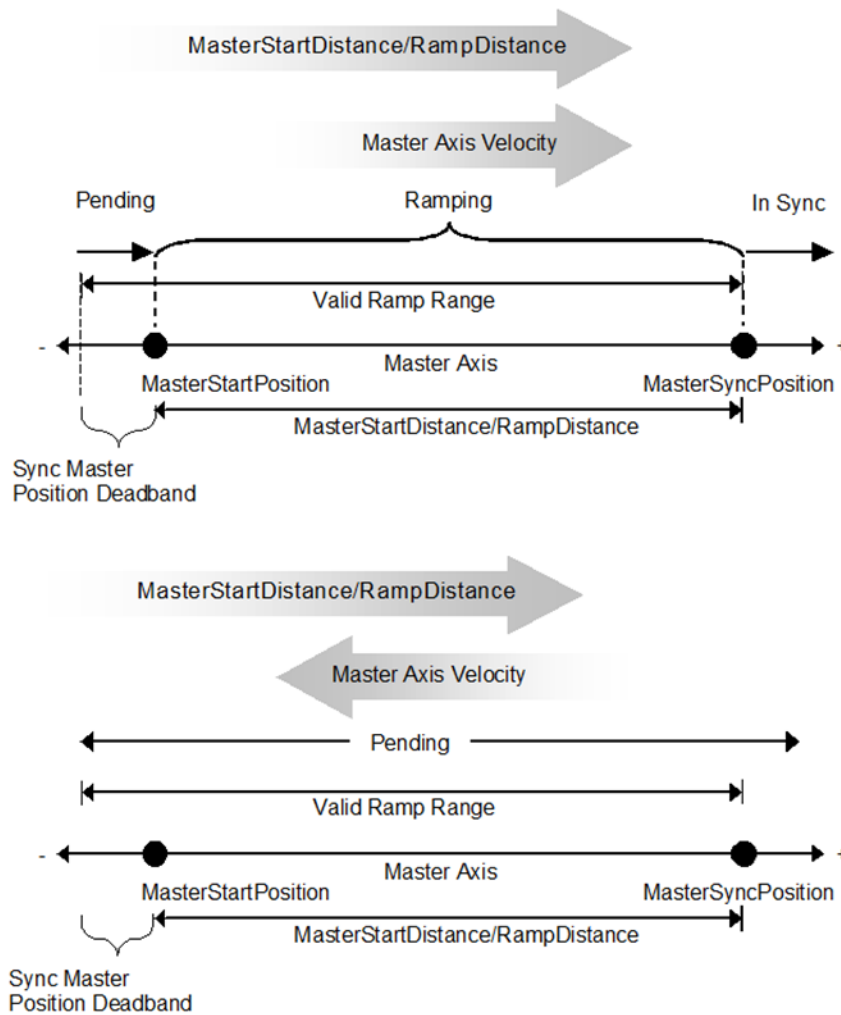
- a. the master axis motion triggers Ramping to begin, causing the MC\_CamIn or MC\_GearInPos to become Active
- b. MC\_CamIn or MC\_GearInPos is aborted by another MFB, or
- c. when the RX3i CPU transitions from Run to Stop mode.

The master axis motion triggers the transition of a slave axis from Pending to Ramping under the following conditions:

- For MC\_GearInPos, the transition is triggered by position and direction. Ramping begins when the master axis crosses the MasterStartPosition in the same direction as the sign of the MasterStartDistance. (The master axis velocity has the same sign as MasterStartDistance.)
- For MC\_CamIn, the transition is triggered only by direction. Ramping begins when the sign of the master's velocity matches the sign of the RampDistance. At this point, the position of the current master axis becomes the MasterStartPosition.

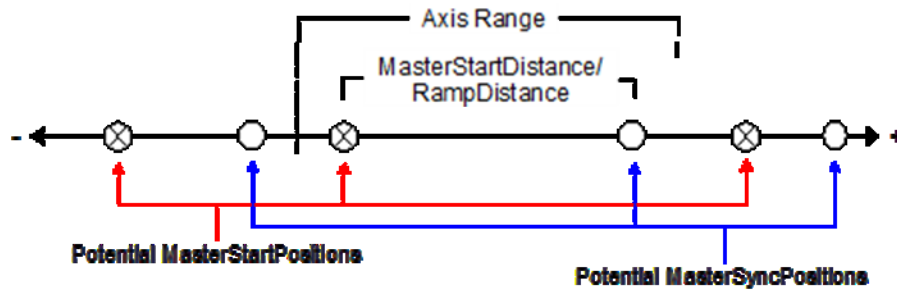
## Master Axis Direction Effect on Pending

Figure 157: Master Axis Direction Effect on Pending



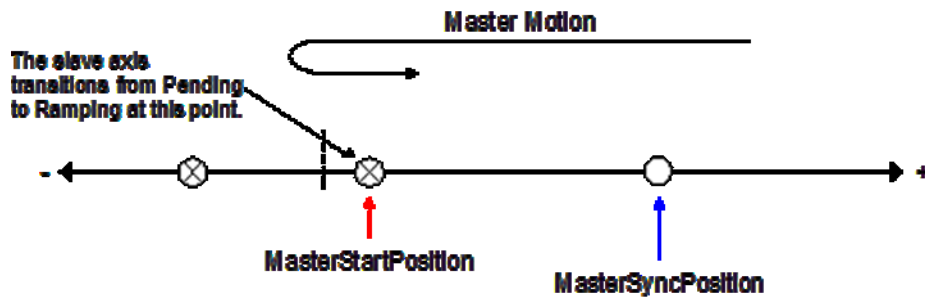
When the Axis Positioning Mode of a master axis is Rotary, there are additional considerations. When the direction of the master velocity is opposite that of the RampDistance (MC\_CamIn), or has not yet crossed the MasterStartPosition (MC\_GearInPos), the MasterSyncPosition and MasterStartPosition are potential.

Figure 158: Potential MasterSyncPosition & MasterStartPosition in Rotary Axis Positioning Mode



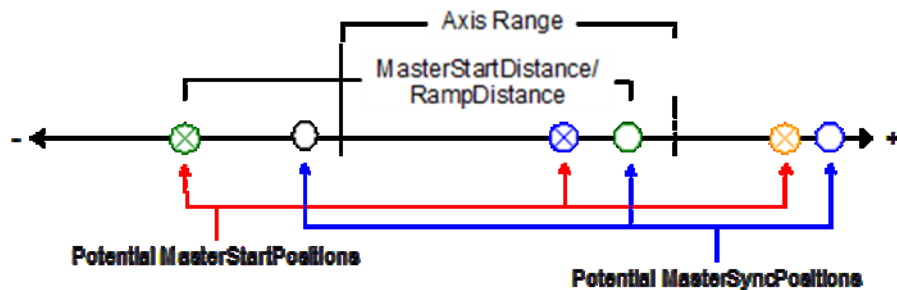
Once the trigger condition is met, the positions are no longer potential, but become fixed.

Figure 159: Fixed MasterSyncPosition & MasterStartPosition in Rotary Axis Positioning Mode



If the MasterStartDistance is larger than the Axis Range of the master, the master will pass potential MasterSyncPositions but will ignore them.

Figure 160: Master Ignoring Potential MasterSyncPositions



If, on a rotary axis, the MasterStartDistance is an integer multiple of the Axis Range of the master, the rotary MasterStartPosition is effectively the same as the MasterSyncPosition.



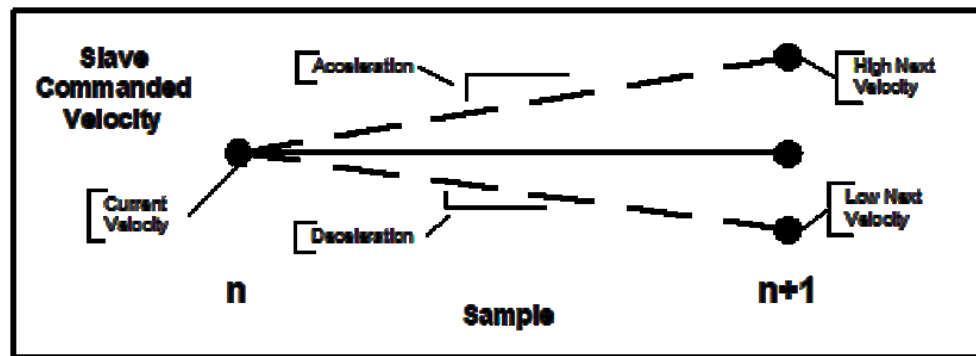
## 7.6.2 Ramping

Synchronized MFBs: MC\_GearIn, MC\_GearInPos and MC\_CamIn

While ramping, the function block's Active and Busy outputs are on, but InSync (or InGear) is not. The slave axis is attempting to synchronize its motion with the master axis, but has not yet achieved synchronization.

MC\_GearIn will use the Acceleration and Deceleration inputs to command the slave axis towards synchronization until the Slave's Commanded Velocity is "close enough" to the target velocity. The Commanded Velocity is "close enough" when the target velocity (which is the velocity of the master times the gear ratio) is between the High Next Velocity and the Low Next Velocity as in Figure 161 below.

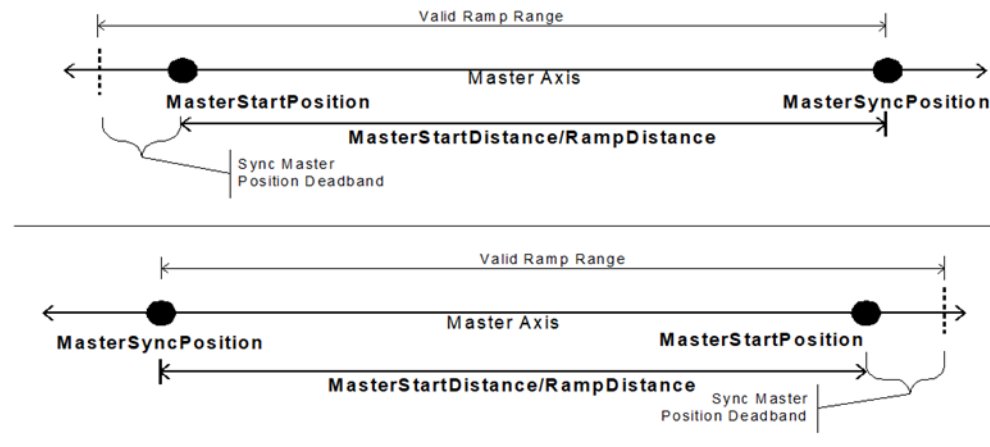
Figure 161: MC\_GearIn Commanding Slave Axis towards Synchronization



MC\_CamIn and MC\_GearInPos ramp the slave to a position when the master is at a specific position and specific velocity relative to the master's velocity at that position. During ramping, the master can move forward, backward, or come to a stop. To achieve this, a curve is continuously fit between the current commanded slave position and velocity to the desired position and velocity.

The commanded velocity of the slave and its accelerations are limited by the application limits of the slave while ramping.

Figure 162: Sync Master Position Deadband



### Limiting Slave Movement During GearInPos or CamIn Ramping

During ramping, the slave attempts to minimize the acceleration it must undergo. If backup is allowed (RampMode or SyncMode is set to 1), a consequence of this is that the ramp may move the slave backward, away from the synchronization position, or past the synchronization position so that it can be moving in the opposite direction when it does synchronize.

In some situations, for example when the master reverses direction while ramping with Backup Allowed, the slave may exhibit unexpected behavior. To prevent this behavior RampMode (or SyncMode) should be set to 0 (No Backup Allowed).

For some CAM applications, it is desirable that the Slave never exceeds the boundaries of the profile. This is the default behavior when Ramp Mode 0.

## 7.6.3 InSync

Synchronized MFBs: MC\_GearIn, MC\_GearInPos and MC\_CamIn

When the function block's Busy, Active, and InSync (InGear) outputs are on, the slave axis motion is synchronized with that of the master axis.

The MC\_CamIn and MC\_GearInPos function blocks set their InSync output on when synchronization is reached.

The MC\_GearIn function block sets its InGear output to indicate synchronization.

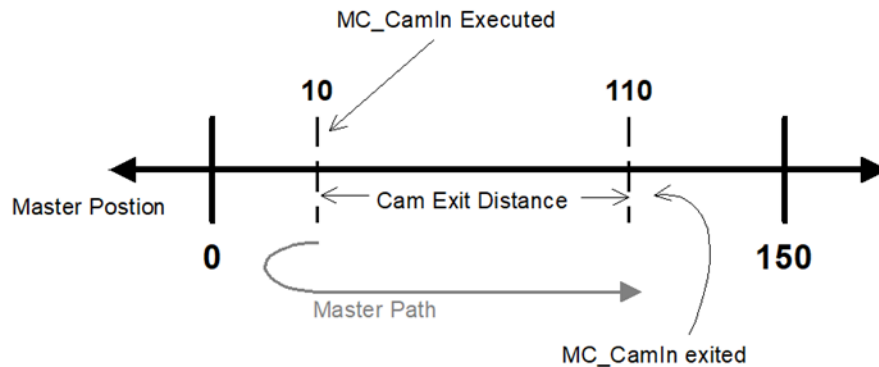
## CAM Exit Distance

Use CAM Exit Distance to exit a CAM profile after the master has travelled a specified distance following CAM engagement (MC\_CamIn becomes Active). For CAM profiles selected as non-periodic (see MC\_CamTableSelect), the profile will exit as usual at the profile boundaries if the CAM Exit Distance is larger than movement of the master axis necessary to exit the profile. For periodic profiles, the profile will repeat until the master has travelled the CAM Exit Distance and then exit immediately. It is now allowed to Buffer a Motion Function Block after a CAM profile selected as periodic if there is a non-zero CAM Exit Distance.

Axis parameter, PN 1333, the CAM Exit Distance, may take on any valid LREAL and the sign indicates the direction of movement of the master axis. The parameter should be written/read on the slave axis of an MC\_CamIn. The parameter is applied when an MC\_CamIn is executed and the parameter is set to zero. Clearing the parameter prevents accidental reuse and allows a buffered MC\_CamIn Function Block to have a CamExitDistance applied.

In the following example, a CAM is entered (MC\_CamIn executed) with the master position at 10 and the CAM Exit Distance parameter is 100, letting the master bounds of the profile be 0 and 150. The CAM will exit when the master has moved 100 units in the positive direction. The master path can move in the positive and negative directions and, as long as it does not exceed the profile bounds, will exit at 110. In the example master path, the master moves 5 units in the negative direction then moves in the positive direction to position 110 where the exit occurs.

Figure 163: MC\_CamIn Execution and Exit



Note that the profile bounds of a non-periodic profile will still be observed. Thus, if the master were to move past position zero in the negative direction, the CAM would exit. Due to the sampled nature of the system, the master position may not be exactly at the exit position at a sample time, so the profile will exit on the sample after the master has moved past the exit position. This is the same as exiting a non-periodic profile at a profile boundary. The worst case is the master velocity times the path planning rate of 1ms. For a master velocity of 5000RPM this translates to 30° of master axis shaft movement beyond the exit position.

## CAM Load Data

The slave position of a CAM profile can be determined from the master position without requiring motion of the slave. The *Load Data* bit of the StartMode input (refer to Section 6.4) returns the position and relative velocity data in the slave axis parameters, PN 1331 and PN 1332 (refer to Axis Parameter Number Index in Section 8.1.1 for parameter definitions). When this bit is set, the function block may only be called with the Slave in the *Standstill* state.

When the Load Data bit is logic 1, the MC\_CamIn does not attempt to engage the axis or change states, but instead will use the settings (Scaling, Offsets, etc.) to determine the slave's position and relative velocity that corresponds to the Master's current position based on the specified CAM profile. These values can then be used with other motion commands to control movement of the slave axis to the synchronized slave position.

The parameter data is valid on a negative transition of the MC\_CamIn *Busy* output bit with no Errors.

## Basic Use

---

**Note:** *The PMM may respond more quickly than the CPU logic sweep, so the positive transition of Busy should be latched immediately after the MC\_CamIn FB is executed allowing the negative transition to be detected.*

---

Using CAM Load Data takes three steps.

1. With the stationary master at the desired position, setup the MC\_CamIn function block with all inputs, set the Load Data bit in StartMode to “1”, and execute the MC\_CamIn command.
2. When Busy transitions false (with no errors) the parameters contain valid data. Read the Slave Position from parameter PN 1331 and execute a move to that position (e.g. MoveAbsolute) on the Slave axis.
3. Set the Load Data bit to “0” on the MC\_CamIn. When that move completes, execute the MC\_CamIn.

## Advanced Use

Custom ramps can be created that allow synchronizing slave axes to master axes under a wide variety of situations. Run-time calculations of ramps should be created while the master is stationary, which may require this to be performed as an initialization step.

Relative velocity can be used to create a custom ramp onto a moving master. The slave velocity can be calculated (slave velocity = master velocity \* relative velocity).

If a slave position is needed at a master position that is not the current one, the MasterOffset parameter can be used to offset the current master position. The MasterOffset parameter will then need to be changed back when MC\_CamIn is executed to actually engage the CAM.

If the master mode is absolute, then the MasterOffset can be used to test master positions at the MasterOffset distance from the current master position.

If the master mode is relative, then the MasterOffset moves from the *left side* of the profile.

## 7.7 CSV CAM File Format

You can export a CAM profile to a comma separated variable (CSV) file. Also, a CSV-formatted CAM profile can be imported into your project and then edited using the CAM profile editor.

The file has three parts, the File Header, the Sector Header, and Sector Bodies. There is one File Header, Number of Sectors Sector Headers, and Number of Points, and each of the Number of Points in Sector has a Sector Body. A single CSV file may contain multiple profiles. The start of a new profile begins with the PROFILE\_NAME field. The following example shows the layout of a CSV file containing two profiles.

**File Header**  
Sector Header  
Sector Body  
...  
Sector Body  
Sector Header  
Sector Body  
...  
Sector Body  
Sector Header  
Sector Body  
...  
Sector Body  
**File Header**  
Sector Header  
Sector Body  
...  
Sector Body

## 7.7.1 File Header Format

Field #1, Identifier	Field #2, Data	Field #3, Options
PROFILE_NAME-	Up to 31 Characters-	
PROFILE_DESCRIPTION	Up to 255 Characters	
CAM_TYPE	CYCLIC_LINEAR/CYCLIC_CIRCULAR/NON_CYCLIC	
NUM_POINT_PAIRS	Positive Integer 2 to 4096 ((5000 if only one linear sector)	
NUM_SECTORS	Positive Integer 1 to 100	
NORMALIZED	YES/NO	
START_FIRST_DERIVATIVE	Signed Decimal	DEFAULT/USER
START_SECOND_DERIVATIVE	Signed Decimal	DEFAULT/USER
END_FIRST_DERIVATIVE	Signed Decimal	DEFAULT/USER
END_SECOND_DERIVATIVE	Signed Decimal	DEFAULT/USER
MASTER_RANGE	Signed Decimal	
MASTER_LOW_LIMIT	Signed Decimal	
SLAVE_RANGE	Signed Decimal	
SLAVE_LOW_LIMIT	Signed Decimal	

## Sector Header Format

Field #1	Field #2	Field #3
Number of Points in Sector (Integer, must be positive)	1/2/3/5 Degree of Curve Fit	YES/NO (Online Correction Enabled)

## CSV Sector Body (repeated for each Sector) Format

Field #1	Field #2	Field #3	Field #4
Master Position	Slave Position	Slave 1 <sup>st</sup> Derivative (Type 5)	Slave 2 <sup>nd</sup> Derivative (Type 5)

## 7.8 Reference Memory Format for CAM Files

A CAM file export-import format is defined below, which is capable of specifying the data that is to be imported into a file. Data that is exported from a file will be exported to reference memory in the same export-import format.

### 7.8.1 Parameters of type LREAL 8 bytes

Structure	Field	Contents
Names	Profile Name	Array of 32 (ASCII) BYTES (16 words). The last byte must be NULL(0x00). If Profile Name is not able to use up the entire 31 bytes reserved for it, NULL characters are filled in the remaining spare bytes.
	Description	Array of 256 (ASCII) BYTES (128 WORDS). The last byte must be NULL (0x00). If description name is not able to use up the entire 255 bytes reserved for it, then NULL characters are filled in the remaining spare bytes.
Boolean Parameters	Boolean Parameters	DWORD (See Boolean Parameters description below)
Enumeration	CAM Type	DWORD: NON_CYCLIC (1), CYCLIC_LINEAR(2), CYCLIC_CIRCULAR(3)
Boundary Conditions	Initial boundary first derivative	LREAL
	Initial boundary second derivative	LREAL
	Final boundary first derivative	LREAL
	Final boundary second derivative	LREAL
Number of position pairs		UINT16
Number of Sectors		UINT16
Normalization Information	Master Axis Range	LREAL
	Master Axis Low Limit	LREAL
	Slave Axis Range	LREAL
	Slave Axis Low Limit	LREAL
Data Table	Sector Header	Sector Header Indicator (LREAL, NaN) This format requires a separator between sectors. The sector header indicator is defined to be 0xFFFF FFFF FFFF FFFF.
		Number of position pairs in sector (UINT16)



Structure	Field	Contents	
		Curve-fit type (linear, quadratic, cubic, w/ 1 <sup>st</sup> der., w/ 2 <sup>nd</sup> der.) (BYTE)	
		Online-Correction (BYTE)	
	1 <sup>st</sup> position pair in sector	1 <sup>st</sup> Master position (LREAL)	
		1 <sup>st</sup> Slave position (LREAL)	
		(IF Curve fit 4 or 5) 1 <sup>st</sup> Slave relative velocity (LREAL)	
		(IF Curve fit 5) 1 <sup>st</sup> Slave relative acceleration (LREAL)	
	...	(data for intermediate position pairs in sector)	
	Last position pair in sector	Last Master position (LREAL)	
		Last Slave position (LREAL)	
		(IF Curve fit 4 or 5) last Slave relative velocity (LREAL)	
		(IF Curve fit 5) last Slave relative acceleration (LREAL)	
			(Sector Header and Position Pair pattern repeated for a total of 'Number of Sectors' times)

## 7.8.2 Boolean Parameters

Description of the available Boolean parameters

Bit#	Description
0	Initial Boundary First Derivative Type: 0 Default, 1-User Defined
1	Initial Boundary Second Derivative Type: 0 Default, 1-User Defined
2	Final Boundary First Derivative Type: 0 Default, 1-User Defined
3	Final Boundary Second Derivative Type: 0 Default, 1-User Defined
4	Normalized: 0 - Not Normalized, 1 - Normalized
5-31	Reserved

# Section 8 Parameters for Monitoring and Control

A number of Motion functions and function blocks make use of parameter numbers or I/O data reference numbers to access configuration, status, and I/O data on the PMM. This chapter provides a reference to the parameters and I/O data reference numbers used to access this data.

Most parameters are associated with an individual axis; however, some are associated with the module. I/O data reference numbers are associated with the PMM faceplate and FTB inputs and outputs and are used as input variables to the read/write digital I/O function blocks.

Parameters can be accessed using the MC\_ReadParameter, MC\_ReadBoolParameter, MC\_WriteParameter and MC\_WriteBoolParameter functions.

Topics covered:

Section 8.1 Axis Parameter Numbers

Section 8.2 Module Parameter Numbers

Section 8.3 I/O Data Reference Numbers

## 8.1 Axis Parameter Numbers

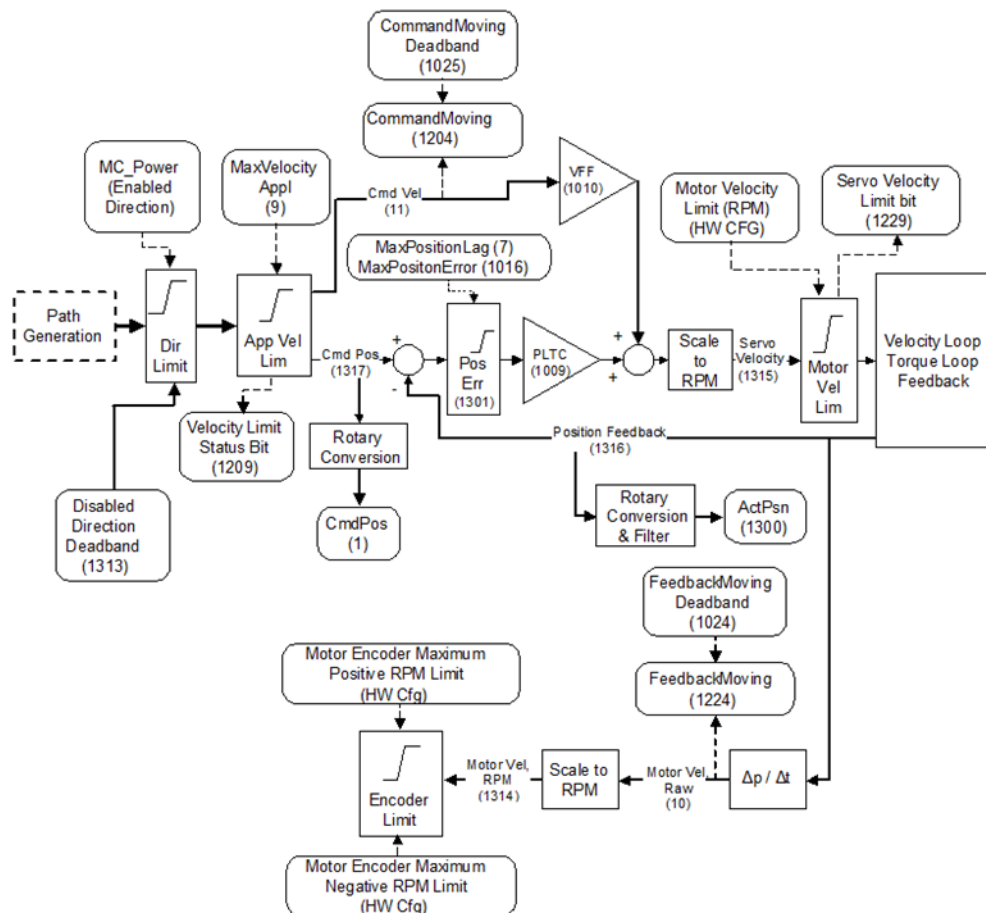
Some parameter numbers correspond to hardware configuration parameters. For additional information on these parameters, refer to Section 4, Configuration. These parameters are initialized to the corresponding values in the Hardware Configuration that is stored to the module. Some Hardware Configuration parameters that are defined in Revs, RPMs or RPM/sec are returned as equivalent User Unit values to facilitate usage in program logic.

Some parameters are dependent on the PMM configuration, and Motion Function Blocks reading or writing to them will fail if the request is not consistent with the configuration. For example, it is invalid to read or write parameter 1004 (ExternalDeviceUserUnits) if no External Device is configured.

### 8.1.1 Position Loop Overview

The following diagram summarizes axis parameters that affect the position loop.

**Figure 164: Position Loop Diagram showing Axis Parameter Interaction**



## 8.1.2 Axis Parameter Number Index

**Note:** The designation UU indicates User Units, a scaling factor that relates engineering units to encoder revolutions and is defined in the HWC. For a sample calculation of User Units, refer to Section 4.3.5, Axis Configuration Data.

Parm	Parameter Name	Description	Data Type	Read/Write	Units
1	CommandedPosition	The instantaneous position commanded by the PMM's internal path generator. Supported axes: 1–4 5 Path Gen	LREAL	Read	UU
2	SWLimitPos	Software end of travel – positive direction Supported axes: 1–4 5 Path Gen	LREAL	Read/Write <sup>12</sup>	UU
3	SWLimitNeg	Software end of travel – negative direction Supported axes: 1–4 5 Path Gen	LREAL	Read/Write <sup>13</sup>	UU
4	EnableLimitPos	Enable software end of travel in positive direction. Initial value is the Software End of Travel parameter in HWC. Supported axes: 1–4 5 Path Gen	BOOL	Read/Write <sup>13</sup>	NA
5	EnableLimitNeg	Enable software end of travel in negative direction. Initial value is the Software End of Travel parameter in HWC. Supported axes: 1–4 5 Path Gen	BOOL	Read/Write <sup>13</sup>	NA
6	EnablePosLagMonitoring	Enable position error monitoring specified by Max Position Lag. Supported axes: 1–4	BOOL	Read/Write	NA
7	MaxPositionLag	Maximum position error Supported axes: 1–4	LREAL	Read/Write	UU

<sup>12</sup> Parameter can be changed only if the axis is in the Disabled state.

Parm	Parameter Name	Description	Data Type	Read/Write	Units
8	MaxVelocitySystem	Maximum allowed velocity of the axis in the motion system. Value is set as RPM in HWC, converted to UU/sec. Supported axes: 1–4 5 Path Gen	LREAL	Read	UU/sec
9	MaxVelocityAppl	Maximum allowed velocity of the axis in the application. This parameter may not exceed the MaxVelocitySystem. MaxVelocityAppl is initialized with the MaxVelocitySystem value and can be later reduced using an MC_WriteParameter function. Axis 1–4 Limits: Low limit: $0.1 * \text{cts/rev} * \text{Uu/cts} * 1/60$ High limit: Max Velocity System in Uu/sec Initial value: Max Velocity System in UU/sec Axis 5 Limits: Low limit: $0.1 * \text{Command Counts Per Motor Revolution} * \text{Command Position Resolution} * 1/60$ Initial value: Max Velocity System in UU/sec Supported axes: 1–4 5 Path Gen	LREAL	Read/Write <sup>13</sup>	UU/sec
10	ActualVelocity	The velocity of the axis reported by the position feedback device. Supported axes: 1–4 5 External Device	LREAL	Read	UU/sec
11	CommandedVelocity	The instantaneous velocity commanded by the PMM's internal path generator. Supported axes: 1–4 5 Path Gen	LREAL	Read	UU/sec

Parm	Parameter Name	Description	Data Type	Read/Write	Units
12	MaxAccelerationSystem	Maximum allowed acceleration of the axis in the motion system. Value is set as RPM/sec in HWC; converted to UU/sec <sup>2</sup> . Supported axes: 1–4 5 Path Gen	LREAL	Read	UU/se c <sup>2</sup>
13	MaxAccelerationAppl	Maximum allowed acceleration of the axis in the application. MaxAccelerationAppl is initialized with the MaxAccelerationSystem value and can be later reduced using an MC_WriteParameter function. Axis 1–4 Limits: Low limit: 0.01 * cts/rev * Uu/cts * 1/60 High limit: Max Acc System (in Uu/sec <sup>2</sup> ) Initial value: MaxAccelerationSystem (in Uu/sec <sup>2</sup> ) Axis 5 Limits: Low limit: 0.01 * Command Counts Per Motor Revolution * Command Position Resolution * 1/60 Initial value: MaxAccelerationSystem (in Uu/sec <sup>2</sup> ) Supported axes: 1–4 5 Path Gen	LREAL	Read/Write <sup>13</sup>	UU/se c <sup>2</sup>
14	MaxDecelerationSystem	Maximum allowed deceleration of the axis in the motion system. Initial value is set as RPM/sec in HWC; converted to UU/sec <sup>2</sup> . Supported axes: 1–4 5 Path Gen	LREAL	Read	UU/se c <sup>2</sup>

Parm	Parameter Name	Description	Data Type	Read/Write	Units
15	MaxDecelerationAppl	<p>Maximum deceleration (application). Maximum allowed deceleration of the axis in the application. MaxDecelerationAppl is initialized with the MaxDecelerationSystem value and can be later reduced using an MC_WriteParameter function.</p> <p>Axis 1–4 Limits:                      Low limit = <math>0.01 * \text{cts/rev} * \text{cts/Uu} * 1/60</math>                      High limit = Max Decel System (in Uu/sec<sup>2</sup>),                      Initial value is MaxDecelerationSystem (in Uu/sec<sup>2</sup>)</p> <p>Axis 5 Limits:                      Low limit: <math>0.01 * \text{Command Counts Per Motor Revolution} * \text{Command Position Resolution} * 1/60</math>                      Initial value is MaxDecelerationSystem (in Uu/sec<sup>2</sup>)</p> <p>Supported axes: 1–4 5 Path Gen</p>	LREAL	Read/Write <sup>13</sup>	UU/se c <sup>2</sup>
16	Maxjerk	<p>Maximum allowed jerk of the axis.</p> <p>Supported axes: 1–4 5 Path Gen</p>	LREAL	Read/Write <sup>13</sup>	UU/se c <sup>3</sup>
1000	MotorEncoderUserUnits (Axes 1–4) Refer also to Considerations When Changing Scaling in Section 8.1.1.	<p>Motor encoder user units. When this parameter is changed, all configuration parameters dependent on this scaling are validated against the configuration limits. If any parameter fails validation, the function block will report an error and the change will not be applied.</p> <p>Supported axes: 1–4 5 Path Gen</p>	LREAL	Read/Write <sup>13,13</sup>	UU

<sup>13</sup> Writing this parameter clears the Position Valid status for the associated feedback device. Therefore, a Find Home or Set Position command is required before point-to-point motion is allowed.

Parm	Parameter Name	Description	Data Type	Read/Write	Units
1000	CommandPositionResolution (Axis 5) Refer also to Considerations When Changing Scaling in Section 8.1.1.	Specifies the resolution of the Axis 5 path generator. When this parameter is changed, all configuration parameters dependent on this scaling are validated against the configuration limits. If any parameter fails validation, the function block will report an error and the change will not be applied.	LREAL	Read/Write <sup>13, 14</sup>	UU
1001	MotorEncoderCounts Refer also to Considerations When Changing Scaling in Section 8.1.1.	Motor encoder counts. When this parameter is changed, all configuration parameters dependent on this scaling are validated against the configuration limits. If any parameter fails validation, the function block will report an error and the change will not be applied. Supported axes: 1–4	DWORD	Read/Write <sup>13, 14</sup>	NA
1002	MotorEncoderPositionRange	Motor encoder position range when Axis Positioning Mode is Rotary, or Motor Encoder is not the Position Feedback Source. Supported axes: 1–4	LREAL	Read/Write <sup>13, 14</sup>	UU
1003	MotorEncoderLowPosition Limit	Motor encoder low position limit when Axis Positioning Mode is Rotary, or Motor Encoder is not the Position Feedback Source. Supported axes: 1–4	LREAL	Read/Write <sup>13, 14</sup>	UU
1004	ExternalDeviceUserUnits Refer also to Considerations When Changing Scaling in Section 8.1.1.	External encoder user units. When this parameter is changed, all configuration parameters dependent on this scaling are validated against the configuration limits. If any parameter fails validation, the function block will report an error and the change will not be applied. Supported axes: 1–4 5 External Device	LREAL	Read/Write <sup>13, 14</sup> (Valid only if an External Device has been configured.)	NA



Parm	Parameter Name	Description	Data Type	Read/Write	Units
1005	ExternalDeviceCounts Refer also to Considerations When Changing Scaling in Section 8.1.1.	External encoder counts. When this parameter is changed, all configuration parameters dependent on this scaling are validated against the configuration limits. If any parameter fails validation, the function block will report an error and the change will not be applied. Supported axes: 1–4 5 External Device	DWORD	Read/Write <sup>13,14</sup> (Valid only if an External Device has been configured.)	NA
1006	ExternalDevicePositionRange	External encoder position range when Axis Positioning Mode is Rotary or External Device is not the Position Feedback Source. Supported axes: 1–4 5 External Device	LREAL	Read/Write <sup>13,14</sup>	UU
1007	ExternalDeviceLowPositionLimit	External encoder low position limit when Axis Positioning Mode is Rotary or External Device is not the Position Feedback Source. Supported axes: 1–4 5 External Device	LREAL	Read/Write <sup>13,14</sup>	UU
1008	InPositionZone	When Position Error is less than or equal to the active In Position Zone, the InZone status (parameter 1205) will be ON. Supported axes: 1–4	LREAL	Read/Write	UU
1009	PositionLoopTimeConstant	Response speed of the closed position loop. Supported axes: 1–4	LREAL	Read/Write	ms
1010	VelocityFeedforward	Percentage of Commanded Velocity that is added to the PMM's position loop velocity command output. Supported axes: 1–4	LREAL	Read/Write	%

Parm	Parameter Name	Description	Data Type	Read/Write	Units
1012	MasterPhaseOffset	Phase offset applied to master axis position by slaves in Synchronized Motion state. This parameter is cleared after aborting a phasing move with a Run to Stop transition. After transitioning the RX3i back to Run Mode, the MasterPhaseOffset value will be the total phase offset that was reached before the axis was stopped. Supported axes: 1–4	LREAL	Read	UU
1013	ErrorStopDeceleration	Maximum deceleration allowed during an Error Stop. Supported axes: 1–4 5 Path Gen	LREAL	Read/Write	UU/sec <sup>2</sup>
1014	ErrorStopJerk	Maximum jerk allowed during an Error Stop. Supported axes: 1–4 5 Path Gen	LREAL	Read/Write	UU/sec <sup>3</sup>
1015	TorqueLimit	Specifies the maximum allowed torque, in percent of available torque, to be produced by the servomotor at commanded velocity. Supported axes: 1–4	LREAL	Read/Write	%
1016	MaxPositionError	An absolute value used to determine when the servo is out of synch and should be stopped. Initial value is set in HWC and converted to User Units. Supported axes: 1–4	LREAL	Read/Write	UU
1022	CommandPositionRange (Axis 5 only)	Specifies the range of values allowed for the virtual axis Commanded Position when Axis Positioning Mode is Rotary. Supported axes: 1–4 5 Path Gen	LREAL	Read/Write <sup>13</sup>	UU
1023	CommandLowPositionLimit (Axis 5 only)	Specifies the lower limit of the virtual axis Commanded Position when Axis Positioning Mode is Rotary. Supported axes: 1–4 5 Path Gen	LREAL	Read/Write <sup>13</sup>	UU

Parm	Parameter Name	Description	Data Type	Read/Write	Units
1024	FeedbackMovingDeadband	Specifies a deadband range for the Actual Velocity in which axis movement will not set the FeedbackMoving status. Supported axes: 1–4 5 External Device	LREAL	Read/Write	UU/Sec
1025	CommandMovingDeadband	Specifies a deadband range for Commanded Velocity, beyond which the CommandMoving status is set. Supported axes: 1–4 5 Path Gen	LREAL	Read/Write	UU/Sec
1100	AxisErrorCode	The most recent, highest-severity Response Type Axis Error ID. Refer to Section 9.1.5, Error ID Reference for details on Error IDs. Supported axes: 1–4 5 Path Gen	WORD	Read	NA
1101	AxisStatus	Array of parameter numbers 1200 to 1229.	DWORD	Read	NA
1200	AxisOK	Axis status. Refer to Section 6.42.1	BOOL	Read	NA
1201	PositionValid	Axis status. Refer to Section 6.42.1	BOOL	Read	NA
1202	DriveEnabled	Axis status. Refer to Section 6.42.1	BOOL	Read	NA
1203	CommandActive	Axis status. Refer to Section 6.42.1	BOOL	Read	NA
1204	CommandMoving	Axis status. Refer to Section 6.42.1	BOOL	Read	NA
1205	InZone	Axis status. Refer to Section 6.42.1	BOOL	Read	NA
1206	MaxPositionLagActive	Axis status. Refer to Section 6.42.1	BOOL	Read	NA
1207	TorqueLimitActive	Axis status. Refer to Section 6.42.1	BOOL	Read	NA
1208	ServoReady	Axis status. Refer to Section 6.42.1	BOOL	Read	NA
1209	VelocityLimit	Axis status. Refer to Section 6.42.1	BOOL	Read	NA
1210	ErrorStop	Axis status. Refer to Section 6.42.1	BOOL	Read	NA

Parm	Parameter Name	Description	Data Type	Read/Write	Units
1211	Disabled	Axis status. Refer to Section 6.42.1	BOOL	Read	NA
1212	Stopping	Axis status. Refer to Section 6.42.1	BOOL	Read	NA
1213	Standstill	Axis status. Refer to Section 6.42.1	BOOL	Read	NA
1214	DiscreteMotion	Axis status. Refer to Section 6.42.1	BOOL	Read	NA
1215	ContinuousMotion	Axis status. Refer to Section 6.42.1	BOOL	Read	NA
1216	SynchronizedMotion	Axis status. Refer to Section 6.42.1	BOOL	Read	NA
1217	Homing	Axis status. Refer to Section 6.42.1	BOOL	Read	NA
1218	ConstantVelocity	Axis status. Refer to Section 6.42.1	BOOL	Read	NA
1219	Accelerating	Axis status. Refer to Section 6.42.1	BOOL	Read	NA
1220	Decelerating	Axis status. Refer to Section 6.42.1	BOOL	Read	NA
1221	OTPos	Over Travel, positive Axis Status. Refer to Section 6.42.1.	BOOL	Read	NA
1222	OTNeg	Over travel negative Axis Status. Refer to Section 6.42.1.	BOOL	Read	NA
1223	HomeSwitch	Axis status. Refer to Section 6.42.1	BOOL	Read	NA
1224	FeedbackMoving	Axis status. Refer to Section 6.42.1.	BOOL	Read	NA
1225	AxisPositioningMode	0 = Linear, 1 = Rotary Set in HWC. Supported axes: 1–4 5	BOOL	Read	NA
1226	Setup	Axis status. Returned in MC_ReadStatus. Refer to Refer to Section 6.42.1	BOOL	Read	NA
1227	Jogging	Axis status. Returned in MC_ReadStatus Axis status. Refer to Refer to Section 6.42.1.	BOOL	Read	NA
1228	AuxPositionValid	A Axis status. Refer to Refer to Section 6.42.1	BOOL	Read	NA
1229	ServoVelocityLimit	Axis status. Refer to Refer to Section 6.42.1	BOOL	Read	NA

Parm	Parameter Name	Description	Data Type	Read/Write	Units
1230–1231	Reserved	Reserved	NA	NA	NA
1300	ActualPosition	Actual position of the axis, as reported by the configured Position Feedback Device. Supported axes: 1–4 5 External Device	LREAL	Read	UU
1301	PositionError	Position Error = Commanded Position – Actual Position Supported axes: 1–4	LREAL	Read	UU
1302	CommandedAcceleration	Commanded acceleration of the axis. Supported axes: 1–4 5 Path Gen	LREAL	Read	UU/sec <sup>2</sup>
1304	TorqueCommand	Commanded torque of the axis. Supported axes: 1–4	LREAL	Read	%
1305	ActualCurrent	Percent of available torque being used by the motor. Supported axes: 1–4	LREAL	Read	%
1306	MotorEncoderPosition	Motor encoder position. Supported axes: 1–4	LREAL	Read	UU
1307	MotorEncoderVelocity	Motor encoder velocity. Supported axes: 1–4	LREAL	Read	UU/sec
1308	ExternalDevicePosition	External encoder position. Supported axes: 1–4 5 External Device	LREAL	Read	UU
1309	ExternalDeviceVelocity	External encoder velocity. Supported axes: 1–4 5 External Device	LREAL	Read	UU/sec
1310	MasterAxisVelocity	The master axis velocity, which is filtered at the rate defined by Master Axis Velocity Filter in HWC Supported axes: 1–4	LREAL	Read	UU/sec

Parm	Parameter Name	Description	Data Type	Read/Write	Units
1311	ForceServoVelocity	Forces servo to the specified RPM for the duration specified by the ForceServoVelocityTimeout (parameter 1320). The axis must be in standstill state. Forced velocity must not exceed the maximum application velocity, MaxVelocityAppl (parameter 9). Supported axes: 1–4 5 Path Gen	LREAL	Read/Write	RPM
1312	SyncMasterPositionDeadband	Valid range is 0.0 to 60,000.0 Default = 0.0. Specifies a positional deadband to be applied to the Master's observed position by this axis when it is a Slave. It may be applied to master positions such as the Master CAM Rollover position and the start position of Ramp on to a CamIn or GearInPos. Supported axes: 1–4	LREAL	Read/Write	Master Axis UU
1313	DisabledDirectionDeadband	Deadband in the direction not enabled by MC_POWER after which an error is generated. High limit = 0.1 motor revolution = 0.1 * Count/rev * Uu/Counts Supported axes: 1–4	LREAL	Read/Write	UU
1314	ServoActualVelocity	The servo velocity in RPM as reported by the Position Feedback Device. Supported axes: 1–4 5 External Device	LREAL	Read	RPM
1315	ServoCommandedVelocity	The velocity in RPMs commanded to the servo. Supported axes: 1–4 5 Path Gen	LREAL	Read	RPM
1316	ActualPositionUnadjusted	Actual Position, unadjusted for rotary mode. Supported axes: 1–4 5 External Device	LREAL	Read	UU

Parm	Parameter Name	Description	Data Type	Read/Write	Units
1317	CommandedPositionUnadjusted	Commanded Position, unadjusted for rotary mode. Supported axes: 1–4 5 Path Gen	LREAL	Read	UU
1318	PhaseAdvanceMinVelocity	The minimum velocity at which Phase Advance will be applied to the master position and velocity data. This parameter is used to avoid overshooting a position when the master is stopping. Supported axes: 1–4	LREAL	Read/Write <sup>13,14</sup>	Master axis Uu/sec
1320	ForceServoVelocityTimeout	Valid range is 0ms to 10000ms. A value of 0 disables motion. Supported axes: 1–4 5 Path Gen	DWORD	Read/Write	ms
1321	MasterAxisVelocityFilter	Specifies filter width of the master axis velocity signal. Valid values are 1, 2, 4, 8, 16, 32, 64, 128 and 256 0 = Disabled Supported axes: 1–4	DWORD	Read/Write <sup>13</sup>	ms
1322	TorqueCommandFilter	Analog Servo Torque Mode only. Sets the torque command filter cutoff frequency 3db point. Allows you to activate a low pass filter for the velocity regulator output. 0 = Disabled Range: 60–400 Supported axes: 1–4	DWORD	Read/Write <sup>13</sup>	Hz

Parm	Parameter Name	Description	Data Type	Read/Write	Units
1323	MinimumVelocityOutput	Analog Servo Velocity Mode only. The minimum velocity output for an analog servo. Any non-zero velocity command will be at least the minimum velocity. The Minimum Velocity Output should be increased until the servo can pull in to $\pm 1$ count of position error. The recommended starting value is 5–10mv When set to 0, the axis may finish a move with non-zero position error. Range 0–1000. Supported axes: 1–4	DWORD	Read/Write	mV
1330	LostPositionAccumulation	Lost position accumulated due to exceeding velocity limit. Supported axes: 1–4	LREAL	Read	UU
1331	SlaveSyncPosition	Synchronized slave position: The absolute position at which the slave axis is on the CAM profile for the current master position. Returned when the GetData bit (on MC_CamIn input StartMode) is enabled. Supported axes: 1–4	LREAL	Read	Slave axis UU
1332	SlaveRelativeVelocity	Synchronized slave relative velocity: The ratio of velocities at the Synchronized Slave Position: Slave Velocity / Master Velocity Returned when the GetData bit (on MC_CamIn input StartMode) is enabled. Supported axes: 1–4	LREAL	Read	NA



Parm	Parameter Name	Description	Data Type	Read/Write	Units
1333	CamExitDistance	CAM exit distance: The distance over which the master may travel, at which point the CAM exits. Applied to the slave axis of an MC_CamIn function block. Sign indicates direction of movement of the master axis. Supported axes: 1–4	LREAL	Read/Write	Master axis UU
1400	OvertravellimitSwitchEn	Hardware switch enable. 0 = disabled 1 = enabled Default value is set by Overtravel Limit Switch in HWC. Ignored by axes using Synthetic Motors (Type Synthetic). Supported axes: 1–4	BOOL	Read/Write	NA
10006	VelocityLoopGain1, Integral (PK1V)	Parameter used with Analog Servo Torque Mode. Write/Read with the MC_WriteDwordParameters and MC_ReadDwordParameters by changing the Data Type of the variables to DINT, instead of DWORD. For additional information, refer to Appendix Section C-3.2, Advanced FSSB Servo Tuning. Ignored by axes using Synthetic Motors (Type Synthetic). Supported axes: 1–4	DINT	Read/Write	NA
10007	VelocityLoopGain2, Proportional (PK2V)	Parameter used with Analog Servo Torque mode. See comments for parameter 10006. Ignored by axes using Synthetic Motors (Type Synthetic). Supported axes: 1–4	DINT	Read/Write	NA

Parm	Parameter Name	Description	Data Type	Read/Write	Units
10008	VelocityLoopGain3, Integral Decay (PK3V)	Parameter used Analog Servo Torque mode. See comments for parameter 10006. Ignored by axes using Synthetic Motors (Type Synthetic).  Supported axes: 1–4	DINT	Read/Write	NA
10031	TorqueCommandFilter (TCMD)	Parameter used with Analog Servo Torque mode. See comments for parameter 10006. Ignored by axes using Synthetic Motors (Type Synthetic).  Supported axes: 1–4	DINT	Read/Write	NA
10032	LoadInertiaRatio	Parameter used with Analog Servo Torque mode. See comments for parameter 10006. Ignored by axes using Synthetic Motors (Type Synthetic).  Supported axes: 1–4	DINT	Read/Write	NA

## Considerations When Changing Scaling

The MotorEncoderUserUnits (parameter 1000), MotorEncoderCounts (parameter 1001), ExternalDeviceUserUnits (parameter 1004), and ExternalDeviceCounts (parameter 1005) provide the ability to change axis feedback device scale factors from program logic. Changing a scale factor for the configured Position Feedback Device impacts all scaled parameters for that axis. Therefore, all parameters dependent on the scaling are validated against the protective limit parameters (Max Velocity System, Max Acceleration System, Max Deceleration System, Max Position Error, Error Stop Deceleration, Error Stop Jerk) using the modified scale factor. If the new scale factor would invalidate any parameter, the function block will return a failure status and the new scale factor is rejected. It may be necessary to temporarily change some dependent parameters to values that are valid with both the original and new scaling, and then change them to the final value after the scaling change has been completed. Section 9.5.1 Parameter Errors Caused by Changes in Axis Scaling discusses how to identify which dependent parameter has failed. It is only permitted to change the UserUnits and Counts parameters for the Axis Feedback Source in the Disabled state. The UserUnits and Counts for the Non-Feedback Source can be changed in the Disabled, Standstill, and ErrorStop states.

## 8.2 Module Parameter Numbers

Parameter Number	Parameter Name	Description	Data Type	Read/Write
2000	ModuleScanData1	Array of parameters 2001 through 2032.	DWORD	Read
2001	Interrupt 1	Interrupt flags are set in RX3i CPU memory only by the occurrence of an interrupt. These interrupt parameters are always scanned/read as zero, except in the corresponding CPU interrupt block.	BOOL	Read
2002	Interrupt 2		BOOL	Read
2003	Interrupt 3		BOOL	Read
2017	ModuleOk	Module status.	BOOL	Read
2018	FTBok	FTB status. Wait for this bit to be set before attempting to read or write I/O on the FTB. This should take no more than 1 second after hardware configuration has been received by the PMM.	BOOL	Read
2019	Axis1Ok	Also, axis parameter number 1200.	BOOL	Read
2020	Axis2Ok	Also, axis parameter number 1200.	BOOL	Read
2021	Axis3Ok	Also, axis parameter number 1200.	BOOL	Read
2022	Axis4Ok	Also, axis parameter number 1200.	BOOL	Read
2023	Axis5Ok	Also, axis parameter number 1200.	BOOL	Read
2030	NewConfigurationReceived	Set when a valid hardware configuration has been received by the module.	BOOL	Read/Write
2031	ModuleSynchronized	Set when PMM is synchronized with other PMMs in a rack.	BOOL	Read
2032	ModulePresent	Module is powered up and operating.	BOOL	Read

Parameter Number	Parameter Name	Description	Data Type	Read/Write
2100	ModuleStatusCode	The most recent, highest severity error code that applies to the module. Refer to Section 9.1.5, Error ID Reference for definitions of error IDs.	DWORD	Read
2101	PmmFirmwareRevision	Rx3i Motion Controller firmware version	DWORD	Read
2102	PmmFirmwareBuildId	Rx3i Motion Controller firmware build ID	DWORD	Read
2104	AnalogOutput1Source	Determines the signal that is output to analog output 1.  Analog output 1 source:  0 = MC_WriteAnalogOutput value (default) 10 = Axis 1 Torque Command 11 = Axis 1 Motor Actual Current 15 = Axis 1 Servo Actual Velocity 16 = Axis 1 Servo Commanded Velocity 20 = Axis 2 Torque Command 21 = Axis 2 Motor Actual Current 25 = Axis 2 Servo Actual Velocity 26 = Axis 2 Servo Commanded Velocity 30 = Axis 3 Torque Command 31 = Axis 3 Motor Actual Current	DWORD	Read/Write (Read-only if axis is used as Analog Servo control)
	Analog Output Scaling			
	Torque Command	8.00V = 100%		
	Motor Actual Current	8.00V = 100%		
	Servo Actual Velocity	10.00V / 8000R		
	Servo Commanded Velocity	10.00V / 8000R		

Parameter Number	Parameter Name	Description	Data Type	Read/Write
		<p>35 = Axis 3 Servo Actual Velocity</p> <p>36 = Axis 3 Servo Commanded Velocity</p> <p>40 = Axis 4 Torque Command</p> <p>41 = Axis 4 Motor Actual Current</p> <p>45 = Axis 4 Servo Actual Velocity</p> <p>46 = Axis 4 Servo Commanded Velocity</p>		
2105	AnalogOutput2Source	<p>Determines the signal that is output to analog output 2.</p> <p>Analog output 2 source:</p> <p>valid values same as Analog Output 1 Source.</p>	DWORD	Read/Write (Read-only if axis is used as Analog Servo control)
2106	TimedInterruptAcknowledgeTimeout	<p>Time to wait (in ms) before checking that a timed interrupt sent to the CPU was acknowledged.</p> <p>Valid range is 0, 2ms to 40ms.</p> <p>0 (default) = Do not check for acknowledgement</p>	DWORD	Read/Write
2107	Array of faceplate input values. Same as parameters 3000 through 3007.	<p>Faceplate Inputs IN1 – IN8</p> <p>bit (8-31) - reserved</p> <p>bit 7 - FP 24Vdc Input 8</p> <p>bit 6 - FP 24Vdc Input 7</p> <p>bit 5 - FP 24Vdc Input 6</p> <p>bit 4 - FP 24Vdc Input 5</p> <p>bit 3 - FP 24Vdc Input 4</p>	DWORD	Read

Parameter Number	Parameter Name	Description	Data Type	Read/Write
		bit 2 - FP 24Vdc Input 3 bit 1 - FP 24Vdc Input 2 bit 0 - FP 24Vdc Input 1		
2108	Array of FTB input values. Same as parameters 3032 through 3059.	FTB Inputs IN1 – IN28 bits (28-31) - reserved bit 27 - FTB 5Vdc Input 28 bit 26 - FTB 5Vdc Input 27 bit 25 - FTB 5Vdc Input 26 bit 24 - FTB 5Vdc Input 25 bit 23 - FTB 5Vdc Input 24 bit 22 - FTB 5Vdc Input 23 bit 21 - FTB 5Vdc Input 22 bit 20 - FTB 5Vdc Input 21 bit 19 - FTB 5Vdc Input 20 bit 18 - FTB 5Vdc Input 19 bit 17 - FTB 5Vdc Input 18 bit 16 - FTB 5Vdc Input 17 bit 15 - FTB 24Vdc Input 16 bit 14 - FTB 24Vdc Input 15 bit 13 - FTB 24Vdc Input 14 bit 12 - FTB 24Vdc Input 13 bit 11 - FTB 24Vdc Input 12 bit 10 - FTB 24Vdc Input 11 bit 9 - FTB 24Vdc Input 10 bit 8 - FTB 24Vdc Input 9 bit 7 - FTB 24Vdc Input 8 bit 6 - FTB 24Vdc Input 7 bit 5 - FTB 24Vdc Input 6	DWORD	Read

Parameter Number	Parameter Name	Description	Data Type	Read/Write
		bit 4 - FTB 24Vdc Input 5 bit 3 - FTB 24Vdc Input 4 bit 2 - FTB 24Vdc Input 3 bit 1 - FTB 24Vdc Input 2 bit 0 - FTB 24Vdc Input 1		
2109	Array of faceplate output values. Same as parameters 3128 and 3129.	FP Outputs OUT1 - OUT2  bits (31-2) - reserved bit 1 - FP Output2 bit 0 - FP Output1	DWORD	Read/Write
2110	Array of FTB output values. Same as parameters 3160 through 3171.	FTB Outputs OUT1 - OUT12  bits (31-24) - reserved bit 11 - FTB Output12 bit 10 - FTB Output11 bit 9 - FTB Output10 bit 8 - FTB Output9 bit 7 - FTB Output8 bit 6 - FTB Output7 bit 5 - FTB Output6 bit 4 - FTB Output5 bit 3 - FTB Output4 bit 2 - FTB Output3 bit 1 - FTB Output2 bit 0 - FTB Output1	DWORD	Read/Write (Read-only if used as Analog Servo Drive Enable or Reset.)
2111	Faceplate 24Vdc fault status	bits 31 - FP Output Overcurrent  bits (2-30) – reserved bit 1 - FP Input 2 Open Wire bit 0 - FP Input 1 Open Wire	DWORD	Read
2112	FTB 5Vdc input fault status	bit 31 - Encoder Power Fault bits (28–30) - reserved bit 27 - FTB Input 28 Open Wire bit 26 - FTB Input 27 Open Wire bit 25 - FTB Input 26 Open Wire bit 24 - FTB Input 25 Open Wire bit 23 - FTB Input 24 Open Wire bit 22 - FTB Input 23 Open	DWORD	Read

Parameter Number	Parameter Name	Description	Data Type	Read/Write
		Wire bit 21 - FTB Input 22 Open Wire bit 20 - FTB Input 21 Open Wire bit 19 - FTB Input 20 Open Wire bit 18 - FTB Input 19 Open Wire bit 17 - FTB Input 18 Open Wire bit 16 - FTB Input 17 Open Wire bits (0-15) - reserved		
2113	FTB 24Vdc output fault status	bit 31 - Term 2 Power Fault bit 30 - Term 1 Power Fault bits (24-29) reserved bit 23 - FTB Output 8 Open Load bit 22 - FTB Output 7 Open Load bit 21 - FTB Output 6 Open Load bit 20 - FTB Output 5 Open Load bit 19 - FTB Output 4 Open Load bit 18 - FTB Output 3 Open Load bit 17 - FTB Output 2 Open Load bit 16 - FTB Output 1 Open Load bits (8-15) - reserved bit 7 - FTB Output 8 Over Temp bit 6 - FTB Output 7 Over Temp bit 5 - FTB Output 6 Over Temp bit 4 - FTB Output 5 Over Temp bit 3 - FTB Output 4 Over Temp bit 2 - FTB Output 3 Over Temp bit 1 - FTB Output 2 Over Temp	DWORD	Read



Parameter Number	Parameter Name	Description	Data Type	Read/Write
		bit 0 - FTB Output 1 Over Temp		
2114	FpDigitalOutputSource	1 - only a Digital CAM Switch function block can control the associated output  0 - only the RX3i CPU can control the associated output using Write Digital Output or Write Parameter 2109  bit 1 - FP Output 2 bit 0 - FP Output 1	DWORD	Read/Write
2115	FtbDigitalOutputSource	1 - only a Digital CAM Switch function block can control the associated output  0 - only the RX3i CPU can control the associated output using Write Digital Output or Write Parameter 2110  bit 11 - FTB Output 12 bit 10 - FTB Output 11 bit 9 - FTB Output 10 bit 8 - FTB Output 9 bit 7 - FTB Output 8 bit 6 - FTB Output 7 bit 5 - FTB Output 6 bit 4 - FTB Output 5 bit 3 - FTB Output 4 bit 2 - FTB Output 3 bit 1 - FTB Output 2 bit 0 - FTB Output 1	DWORD	Read/Write
2500–2509	TenMostRecentEvents	Contain the ten most recent axis and module events in chronological order, where 2500 contains the most recent event. Includes events preserved over a power cycle in non-volatile memory.	DWORD	Read

## 8.3 I/O Data Reference Numbers

I/O data reference numbers are associated with the PMM faceplate and FTB inputs and outputs. They are used as input variables to the read/write digital I/O function blocks, MC\_ReadDigitalInput, MC\_WriteDigitalInput, MC\_ReadDigitalOutput, and MC\_WriteDigitalOutput. The reference numbers are accessed through the OUTPUT\_REF or INPUT\_REF variable inputs to these instructions.

---

**Note:** These reference numbers are not parameters that can be accessed using the Read/Write parameter instructions, MC\_ReadBoolParameter, MC\_WriteBoolParameter, etc.

---

Reference Number	Parameter Name	Description	Data Type	Read/Write
<b>Faceplate Digital Inputs</b>				
3000	Fp24Vin1	Value of the 24Vdc high-speed input	BOOL	Read
3001	Fp24Vin2		BOOL	Read
3002	Fp24Vin3	Value of the 24Vdc general purpose input In3 and In4 share the same physical faceplate points with Out1 and Out2, respectively. Each point is configured as an input or an output in HWC.	BOOL	Read
3003	Fp24Vin4		BOOL	Read
3004	Fp24Vin5		BOOL	Read
3005	Fp24Vin6		BOOL	Read
3006	Fp24Vin7		BOOL	Read
3007	Fp24Vin8		BOOL	Read
<b>Fiber Terminal Block Digital Inputs</b>				
3032	FtIn1	Value of the 24Vdc input	BOOL	Read
3033	FtIn2		BOOL	Read
3034	FtIn3		BOOL	Read
3035	FtIn4		BOOL	Read
3036	FtIn5		BOOL	Read
3037	FtIn6		BOOL	Read
3038	FtIn7		BOOL	Read
3039	FtIn8		BOOL	Read
3040	FtIn9		BOOL	Read
3041	FtIn10		BOOL	Read
3042	FtIn11		BOOL	Read
3043	FtIn12		BOOL	Read
3044	FtIn13		BOOL	Read
3045	FtIn14		BOOL	Read
3046	FtIn15		BOOL	Read
3047	FtIn16		BOOL	Read
3048	FtIn17	Value of the 5 Vdc input	BOOL	Read
3049	FtIn18		BOOL	Read
3050	FtIn19		BOOL	Read
3051	FtIn20		BOOL	Read

Reference Number	Parameter Name	Description	Data Type	Read/Write
3052	FtIn21		BOOL	Read
3053	FtIn22		BOOL	Read
3054	FtIn23		BOOL	Read
3055	FtIn24	Value of the 5 Vdc input. The IN24 input and OUT9 output share the same physical point on the FTB. You must configure the point as in input or an output in HWC.	BOOL	Read
3056	FtIn25	Value of the 5 Vdc input. The IN25 input and OUT10 output share the same physical point on the FTB. You must configure the point as in input or an output in HWC.	BOOL	Read
3057	FtIn26	Value of the 5 Vdc input	BOOL	Read
3058	FtIn27	Value of the 5 Vdc input. The IN27 input and OUT11 output share the same physical point on the FTB. You must configure the point as in input or an output in HWC.	BOOL	Read
3059	FtIn28	Value of the 5 Vdc input. The IN28 input and OUT12 output share the same physical point on the FTB. You must configure the point as in input or an output in HWC.	BOOL	Read
<b>Faceplate Digital Outputs</b>				
3128	Fp24FVOut1	The 24Vdc Out1 and 24Vdc In3 share the same physical faceplate point. You must configure the point as an input or an output in HWC.	BOOL	Read/Write
3129	Fp24FVOut2	The 24Vdc Out2 and 24Vdc In4 share the same physical faceplate point. You must configure the point as an input or an output in HWC	BOOL	Read/Write
<b>Fiber Terminal Block Digital Outputs</b>				
3160	FtbOut1	Value of the 24Vdc output	BOOL	Read/Write (Read-only if used as Analog Servo Drive Enable or Reset.)
3161	FtbOut2		BOOL	
3162	FtbOut3		BOOL	
3163	FtbOut4		BOOL	
3164	FtbOut5		BOOL	
3165	FtbOut6		BOOL	
3166	FtbOut7		BOOL	
3167	FtbOut8		BOOL	
3168	FtbOut9	Value of the 24Vdc output. The IN24 input and OUT9 output share the same physical point on the FTB. You must configure the point as an input or output in HWC.	BOOL	Read/Write (Read-only if used as

Reference Number	Parameter Name	Description	Data Type	Read/Write
3169	FtbOut10	Value of the 24Vdc output. The IN25 input and OUT10 output share the same physical point on the FTB. You must configure the point as an input or output in HWC.	BOOL	Analog Servo Drive Enable or Reset.)
3170	FtbOut11	Value of the 24Vdc output. The IN27 input and OUT11 output share the same physical point on the FTB. You must configure the point as an input or output in HWC.	BOOL	
3171	FtbOut12	Value of the 24Vdc output. The IN24 input and OUT12 output share the same physical point on the FTB. You must configure the point as an input or output in HWC.	BOOL	
<b>Fiber Terminal Block Analog Inputs</b>				
3256	FtbAlgIn1	Value of the ±10 V analog input.	REAL64	Read
3257	FtbAlgIn2		REAL64	
<b>Fiber Terminal Block Analog Outputs</b>				
3288	FtbAlgOut1	Value of the ±10 V analog output.	REAL64	Read/Write (Read-only if used for Analog Servo control.)
3289	FtbAlgOut2		REAL64	

# Section 9    Diagnostics

Basic diagnostic tools for PACMotion include:

Section 9.1 PMM Error IDs

Section 9.2 CPU Error Codes

Section 9.3 Interpreting Drive Faults and Warnings

Section 9.4 PMM Event Queue

Section 9.5 Accessing the Ten Most Recent Events

Section 9.6 Diagnostic Logic Blocks

An additional diagnostic tool, *Data Logging*, is discussed in Section 6.10, *MC\_DL\_Activate*; Section 6.11, *MC\_DL\_Configure*; Section 6.12, *MC\_DL\_Delete*; and Section 6.13, *MC\_DL\_Get*.

## 9.1 PMM Error IDs

The PMM returns ErrorIDs in the form of 16-bit numbers that are structured to encode information about error severity and cause. ErrorIDs are returned in the Error output of motion function blocks (MFBs), in the module status, or in the axis error code. If an entry is generated in the I/O Fault Table, the first 2 bytes in the Fault Extra Data correspond to the Error ID.

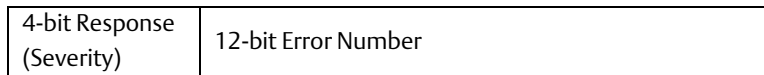
### 9.1.1 Accessing Error IDs

The MC\_ReadAxisError function is used to read a current axis error or warning that is not related to the execution of functions or function blocks. This function returns the highest severity error or warning message. To view multiple errors and the error sequence, use the MC\_ReadEventQueue function block.

For a listing of error codes with causes and corrective actions, refer to Section 9.1.5, Error ID Reference.

### 9.1.2 Error ID Format

The Error IDs returned from motion functions and function blocks are 16-bit numbers structured to encode information about the nature and the effect (response) of the error or warning. The figure below shows the encoding of the Error IDs.



An error takes precedence over a warning. A function block can successfully complete with a warning, but not with an error.

The following Response values indicate the effect of the error on the motion.

Response	Response Type	Effect of Error on Motion
0	Warning (No Stop)	The requested function was not performed, but the error had no impact on the axis motion.
5	Normal Stop, Axis Level	The error caused a normal stop on the axis. Motion is stopped on the axis and the axis enters the ErrorStop state. MC_Reset must be issued on the axis to clear the error condition and return the axis to the Standstill state before motion can be restarted on the axis.
6	Fast Stop	The error caused a fast stop on the axis. Motion is stopped on the axis and the axis enters the ErrorStop state. Before motion can be restarted on the axis, MC_Reset must be issued on the axis to clear the error condition and return the axis to the Standstill state. Note that processing MC_Reset for such errors may cause other axes that are active to be forced into a normal stop in order to reset the error.

The Error Number field gives detailed information about the error or warning. For Error Number definitions, refer to Section 9.1.5, Error ID Reference.

### 9.1.3 Clearing PMM Errors

After correcting the cause of an axis error, the error must be cleared using an MC\_Reset or MC\_ModuleReset function block. MC\_Reset clears only the errors on the specified axis. MC\_ModuleReset clears errors on all axes, as well as module level errors.

An MC\_ModuleReset clears module-level errors, such as those reported in the Module Status Code (parameter no. 2100), on the target PMM. FTB errors must be cleared by executing MC\_ModuleReset on the PMM that is connected to the FTB.

### 9.1.4 I/O Fault Table

Errors that are reported to the I/O Fault Table include the Error ID in the first 2 bytes of Fault Extra Data.

In this example, the 24Vdc power was removed from the servo drive while the servo was powered-on. As can be seen in the table (Figure 165), this causes three faults to be generated. To find more information about any of these faults, search this chapter for the last 3 nibbles of the Error ID, such as 0D6.

Figure 165: I/O Fault Table Example

Loc	CIRC No.	Variable Name	Ref. Address	Fault Category	Fault Type	Date/Time																												
0.3	n/a		%I 00081	Motion Module fault	Axis warning or error	01-01-2000 00:05:33																												
<table border="1"> <thead> <tr> <th>I/O Bus</th> <th>Bus Address</th> <th>Point Address</th> <th>Group</th> <th>Action</th> <th>Category</th> <th>Fault Type</th> </tr> </thead> <tbody> <tr> <td>n/a</td> <td>n/a</td> <td>n/a</td> <td>10</td> <td>2:Diagnostic</td> <td>32</td> <td>3</td> </tr> <tr> <td colspan="7">Fault Extra Data: 62 64 27 20 17 97 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00</td> </tr> <tr> <td colspan="7">Fault Description: Axis position is not valid</td> </tr> </tbody> </table>							I/O Bus	Bus Address	Point Address	Group	Action	Category	Fault Type	n/a	n/a	n/a	10	2:Diagnostic	32	3	Fault Extra Data: 62 64 27 20 17 97 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00							Fault Description: Axis position is not valid						
I/O Bus	Bus Address	Point Address	Group	Action	Category	Fault Type																												
n/a	n/a	n/a	10	2:Diagnostic	32	3																												
Fault Extra Data: 62 64 27 20 17 97 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00																																		
Fault Description: Axis position is not valid																																		
0.3	n/a		%I 00081	Motion Module fault	Axis warning or error	01-01-2000 00:05:32																												
<table border="1"> <thead> <tr> <th>I/O Bus</th> <th>Bus Address</th> <th>Point Address</th> <th>Group</th> <th>Action</th> <th>Category</th> <th>Fault Type</th> </tr> </thead> <tbody> <tr> <td>n/a</td> <td>n/a</td> <td>n/a</td> <td>10</td> <td>2:Diagnostic</td> <td>32</td> <td>3</td> </tr> <tr> <td colspan="7">Fault Extra Data: 60 d6 27 a0 10 02 40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00</td> </tr> <tr> <td colspan="7">Fault Description: Servo link error - see module event queue for detailed description</td> </tr> </tbody> </table>							I/O Bus	Bus Address	Point Address	Group	Action	Category	Fault Type	n/a	n/a	n/a	10	2:Diagnostic	32	3	Fault Extra Data: 60 d6 27 a0 10 02 40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00							Fault Description: Servo link error - see module event queue for detailed description						
I/O Bus	Bus Address	Point Address	Group	Action	Category	Fault Type																												
n/a	n/a	n/a	10	2:Diagnostic	32	3																												
Fault Extra Data: 60 d6 27 a0 10 02 40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00																																		
Fault Description: Servo link error - see module event queue for detailed description																																		
0.3	n/a		%I 00081	Motion Module fault	Axis warning or error	01-01-2000 00:05:32																												
<table border="1"> <thead> <tr> <th>I/O Bus</th> <th>Bus Address</th> <th>Point Address</th> <th>Group</th> <th>Action</th> <th>Category</th> <th>Fault Type</th> </tr> </thead> <tbody> <tr> <td>n/a</td> <td>n/a</td> <td>n/a</td> <td>10</td> <td>2:Diagnostic</td> <td>32</td> <td>3</td> </tr> <tr> <td colspan="7">Fault Extra Data: 60 c0 38 20 0e 49 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00</td> </tr> <tr> <td colspan="7">Fault Description: Servo error - see module event queue for detailed description</td> </tr> </tbody> </table>							I/O Bus	Bus Address	Point Address	Group	Action	Category	Fault Type	n/a	n/a	n/a	10	2:Diagnostic	32	3	Fault Extra Data: 60 c0 38 20 0e 49 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00							Fault Description: Servo error - see module event queue for detailed description						
I/O Bus	Bus Address	Point Address	Group	Action	Category	Fault Type																												
n/a	n/a	n/a	10	2:Diagnostic	32	3																												
Fault Extra Data: 60 c0 38 20 0e 49 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00																																		
Fault Description: Servo error - see module event queue for detailed description																																		

## 9.1.5 Error ID Reference

This section lists error IDs for errors or warnings that could occur within motion control functions and function blocks, with recommended corrective actions.

Error Format	Response Type
0xxx	No Stop
5xxx	Normal Stop
6xxx	Fast Stop

### Error IDs

Error No. (Hex)	Description	Cause	Recommended Correction
x024	Velocity is greater than application velocity limit.	The application velocity maximum has been exceeded. Commanded velocity is clamped to the application velocity maximum.	Increase the application velocity maximum or reduce the velocity of the motion executing on the axis. When in synchronized motion (SyncMotion), this includes motion of the master and the applicable CAM profile or gear ratio as well as any executing MC_Phasing or MC_SuperImposed function blocks.
x030	MC_Home issued when power not on at axis (drive not enabled).	An MC_Home function block was issued by the application when the axis was in the Disabled state.	Change the application to enable the axis power using MC_Power before issuing the MC_Home. The axis must be in the Standstill state before homing.
x032	MC_Home issued while Force Digital Servo Velocity is active.	An MC_Home function block was issued by the application when the axis was in the Setup state due to a Force Digital Servo Velocity command.	Change the application to wait until the servo velocity is not forced before performing the MC_Home. The axis must be in the Standstill state before homing.
x033	MC_Home issued while jogging.	The application attempted to invoke an MC_Home function block instance when an MC_JogAxis was active.	Disable both the Enable_Forward and the Enable_Backward parameters of any MC_JogAxis function block instance active on the axis before invoking the MC_Home MFB instance.
x034	MC_Home while in non-synchronized motion	The application attempted to invoke an MC_Home function block instance with the axis in non-synchronized motion (Discrete Motion or Continuous Motion state).	The MC_Home function block instance may be invoked only in the Standstill state. Ensure the axis is in Standstill state before issuing the MC_Home function block.
x035	MC_Home while in synchronized motion	The application attempted to invoke an MC_Home function block instance with the axis in synchronous motion (Synchronized Motion state).	The MC_Home function block instance may be invoked only in the Standstill state. Ensure the axis is in Standstill state before issuing the MC_Home.



Error No. (Hex)	Description	Cause	Recommended Correction
x039	MC_MoveVelocity when power not on at axis (drive not enabled)	The application attempted to invoke an MC_MoveVelocity function block instance with no power applied to the axis (either no MC_Power instance active or the Enable input to the active MC_Power set FALSE).	Hold the Enable parameter to the MC_Power function block for the axis TRUE before invoking the MC_MoveVelocity function block instance. Change the application to enable the axis power using MC_Power before issuing the MC_MoveVelocity.
x03B	MC_MoveVelocity when homing	The application attempted to invoke an MC_MoveVelocity function block instance while in the Homing state.	Allow the active MC_Home to finish before enabling the MC_MoveVelocity function block.
x03C	MC_MoveVelocity while jogging.	The application attempted to invoke an MC_MoveVelocity function block instance when an MC_JogAxis function block was active.	Disable both the Enable_Forward and the Enable_Backward parameters of any MC_JogAxis function block instance active on the axis before invoking the MC_MoveVelocity instance.
x040	MC_JogAxis while homing.	An MC_JogAxis function block instance was invoked while in the Homing state.	Allow the active MC_Home to finish before enabling the jog.
x045	MC_JogAxis while Force Digital Servo Velocity is active.	The application attempted to invoke an MC_JogAxis function block instance when the force servo velocity function is active.	Write the Force Servo Velocity (1311) parameter to disable the force servo velocity function and then enable the MC_JogAxis function block.
x046	MC_JogAxis while in motion and feed hold not active.	The application attempted to invoke an MC_JogAxis function block instance when the axis was in motion, i.e. in the Discrete Motion or Continuous Motion states.	Either wait for the axis to enter the Standstill state or issue an MC_SetOverride with the VelFactor parameter set to zero to assert a feed hold on the axis before issuing the jog. Note that jog may not be issued with the axis in Synchronized Motion state, since an MC_SetOverride cannot be used to assert a feed hold in Synchronized Motion state.
x047	Force Digital Servo Velocity attempted while jogging.	The application attempted to force the servo velocity by writing the Force Servo Velocity (1311) parameter while the axis was jogging.	Disable both the Enable_Forward and the Enable_Backward parameters of any MC_JogAxis function block instance active on the axis before writing the parameter.
x049	Force servo velocity during non-synchronous motion.	The application attempted to force the servo velocity by writing the Force Servo Velocity (1311) parameter while the axis was in the Discrete or Continuous Motion state.	Have any active function block instance on the axis complete or abort before writing the parameter.

Error No. (Hex)	Description	Cause	Recommended Correction
x04A	Force servo velocity during synchronized motion.	The application attempted to force the servo velocity by writing the Force Servo Velocity (1311) parameter while the axis was in the Synchronized Motion state.	Disengage the slave axis from the master and bring the axis to a stop before writing the parameter.
x051	MC_SetPosition absolute position out of range.	The specified position value for an absolute MC_SetPosition would result in an invalid position.	Set the Position parameter value to a value within the valid range or change the HWC to extend the range.
x052	MC_SetPosition state error.	MC_SetPosition with relative mode executed without position valid.	Obtain a valid position by executing an absolute MC_SetPosition or MC_Home before executing MC_SetPosition relative.
X054	MC_SetPosition	An MC_SetPosition instance was executed on an axis that does not have valid encoder feedback.	Examine the the fault table and event queue for communication or drive faults that could result in invalid encoder feedback.
x056	Move past positive end of travel.	The commanded position for a move would exceed the end of travel limit in the positive direction.	Either change the position for the function block in error or change the High Software EOT limit in HWC. Use an MC_JogAxis to return the axis to an allowed position.
x057	Move past negative end of travel.	The commanded position for a move would exceed the end of travel limit in the negative direction.	Either change the position for the function block in error or change the Low Software EOT limit in HWC. Use an MC_JogAxis to return the axis to an allowed position.
x058	Absolute encoder position greater than High Software EOT limit.	On power-up or reconfiguration the position of an absolute encoder has been moved beyond the positive end of travel for the axis.	Move the axis (using MC_JogAxis for example) within the end of travel limit or change the hardware configuration so that the axis position is within the end of travel limit.
x059	Absolute encoder position less than Low Software EOT limit.	On power-up or reconfiguration the position of an absolute encoder has been moved beyond the negative end of travel for the axis.	Move the axis (using MC_JogAxis for example) within the end of travel limit or change the hardware configuration so that the axis position is within the end of travel limit.
x05B	Drive disabled (MC_Power Enable set false). Axis motion not possible.	The Enable input on the MC_Power instance for the axis has transitioned to FALSE. Any MFB in progress or pending (e.g. waiting to be blended into the result of the active move) for the axis will be terminated with this error.	If unexpected, examine the logic for the cause of setting the MC_Power Enable input FALSE and correct this logic so that the Enable input is held TRUE while commands are active.
x05F	Internal error on motion module.	Internal error.	Contact Technical Support.

Error No. (Hex)	Description	Cause	Recommended Correction
X068	(Online Correction Enabled) CAM Velocity Command Limited due to Velocity Limit violation	The Online Correction feature is Enabled for this CAM-curve sector in the CAM Profile editor. The commanded velocity from the CAM is greater than the maximum velocity application of the axis (MaxVelocityAppl) and has been limited to the MaxVelocityAppl value. This can cause the position error to grow.	If this is not the desired behavior, make one of the following changes: disable the Online Correction feature (will cause axis to normal stop if max velocity is exceeded), reduce master velocity, or modify CAM profile shape to reduce slave velocity.
X069	(Online Correction Disabled) CAM velocity command above configured axis Velocity Limit	The Online Correction feature is Disabled for this CAM-curve sector in the CAM Profile editor. The commanded velocity from the CAM is greater than the maximum velocity application of the axis (MaxVelocityAppl), causing the axis to normal stop.	If this is not the desired behavior, make one of the following changes: enable the Online Correction feature (will cause the axis to limit velocity and generate warning if max velocity is exceeded), reduce master velocity, or modify CAM profile shape to reduce slave velocity.
x0A0	Hardware limit switch encountered in the positive direction.	The move encountered the hardware limit switch in the positive direction.	Jog the axis off the limit switch and change the range of the move to be within the hardware limits.
x0A1	Hardware limit switch encountered in the negative direction.	The move encountered the hardware limit switch in the negative direction.	Jog the axis off the limit switch and change the range of the move to be within the hardware limits.
X0A2	EtherCAT cyclic data synchronization error.	EVENTID_PDO_SYNC_ERROR	
x0A8	Max Position Error exceeded.	The servo position error has exceeded the Max Position Error (UU) value in the axis configuration. The error can be caused by <ul style="list-style-type: none"> <li>1. Defective or incorrectly wired encoder used for external position feedback</li> <li>2. Incorrect settings for Position Lag Monitoring, Max Position Lag (UU) or Max Position Error (UU)</li> <li>3. Motor Velocity Limit (RPM) set lower than the commanded servo velocity when Position Lag Monitoring is Disabled</li> </ul>	Ensure the feedback source for the axis is properly configured and is connected and working properly. <ul style="list-style-type: none"> <li>1. Check for a defective or incorrectly wired encoder used for external position feedback.</li> <li>2. Review the settings for Position Lag Monitoring, Max Position Lag (UU) and Max Position Error (UU).</li> <li>3. Review the configuration software setting for Motor Velocity Limit (RPM).</li> </ul>

Error No. (Hex)	Description	Cause	Recommended Correction
X0C0	Servo Drive Fault	A fault was detected by the servo drive.	Check the servo drive display for fault code.
x0D9, x0DA, x0DC	PMM Module hardware error.	A hardware error was detected on the module.	Contact Technical Support.
x0DD	EtherCAT acyclic message failed.	A mailbox message failed to send or received an error response.	The most likely cause is attempting to write a drive parameter to an invalid value. Check the function block sending drive parameters (i.e. MC_Home) for an error indicating which parameter failed to write. Contact technical support if the cause of the fault is unclear.
x0DE	Error or warning occurred on PACMotion Drive	The PACMotion Drive has reported an error or warning.	See section 9.3 for details on interpreting the specific cause of this event.
x0DF	PACMotion Drive entered the fault state.	An error has occurred on the PACMotion Drive causing it to enter a fault state.	When available an additional event x0DE will be logged with additional details reported by the PACMotion Drive.
x0E0	EtherCAT Initialization Failure	Failed to connect to all EtherCAT devices and be ready to start operation in the expected time.	Verify that the number of EtherCAT axes configured in the hardware configuration matches the number of drives powered on and connected to the EtherCAT network. Check for other communication specific errors.
x0E1	EtherCAT Communication lost	Communication has been lost with one or more axes. Any axes in motion will be stopped.	Check for disconnected network cable or a drive that has lost power.
x0E2	EtherCAT Network State Error	The EtherCAT network has changed to a non-operational state due to loss of a device or loss of synchronization with a device.	Check for disconnected network cable or a drive that has lost power. Check for faults reported by a drive.
x0E3	EtherCAT Device State Error	A drive has transitioned to a non-operational state.	Check for specific faults reported by a drive or for related communication errors.
x0E4	Timeout Reading Drive Parameters	Unable to read drive error or warning via EtherCAT.	Use PACMotion Workbench to read the drive error or warning. Error code is also displayed on the drive faceplate LCD.
x0E5	Motor encoder velocity > Motor Encoder Maximum Positive Limit.	Motor encoder velocity in the positive direction exceeded the rpm limit set by HWC.	Verify commanded velocity and Limit are expected values and modify as necessary.
x0E6	Motor encoder velocity < Motor Encoder Maximum Negative Limit.	Motor encoder velocity in the negative direction exceeded the rpm limit set by HWC.	Verify commanded velocity and Limit are expected values and modify as necessary.
X0E7	Home Sequence did not complete	Re-home the motor	Check drive home parameters are set correctly per MC_HOME function block instructions

Error No. (Hex)	Description	Cause	Recommended Correction
x0FB	Control loop execution exceeds specified maximum time.	The control loop exceeded its specified time limit.	Contact Technical Support.
x0FC	Control loop warning time limit exceeded.	The control loop exceeded its warning threshold.	Contact Technical Support.
x0FD	Severe software error.	Internal error.	Contact Technical Support.
x0FE	Unrecognized encoder.	An unrecognized encoder is attached.	This error can indicate a defective encoder cable – check cable. If cable checks out correctly, contact Technical Support.
x101	Invalid system state to process function block.	The MFB instance cannot be processed because the state of the PMM or the state of the RX3i controller does not support the command.	Refer to other diagnostic tools such as Axis Status, Module Status, and Controller and I/O Fault tables for further diagnostics.
x103	Unable to process additional function block commands because list is full.	Internal error.	Contact Technical Support.
x104	Delay in sending function block responses.	Response queue is full.	None. This is an informational ErrorID.
x20A	DCS invalid track number.	Invalid inputs to MC_DigitalCamSwitch function block.  <b>Note:</b> <i>If error causes an active DCS to be aborted then a Normal Stop on the axis will occur.</i>	Correct the DCS track information.
x20B	DCS overlapping switch points.	Invalid inputs to MC_DigitalCamSwitch function block.  <b>Note:</b> <i>If error causes an active DCS to be aborted then a Normal Stop on the axis will occur.</i>	Correct switch point information.
x20C	DCS invalid axis direction.	Invalid inputs to MC_DigitalCamSwitch function block.  <b>Note:</b> <i>If error causes an active DCS to be aborted then a Normal Stop on the axis will occur.</i>	Correct AxisDirection input on DCS.

Error No. (Hex)	Description	Cause	Recommended Correction
x20D	DCS aborted due to write of DCS mask.	Application has written to the parameters (2114 or 2115) that control whether the DCS uses a point. This action aborts all active DCS functions.  <b>Note:</b> <i>If error causes an active DCS to be aborted then a Normal Stop on the axis will occur.</i>	Do not write the parameters (PN2114 or PN2115) that determine whether a point is controlled by a DCS while an MC_DigitalCamSwitch is active.
x20F	DCS aborted; axis position is invalid.	The position is no longer valid causing the DCS function to terminate.	Check application to assure that home cycles or other operations that can invalidate Position Valid do not occur while a DCS is active.
x210	DCS instance was superseded by another.	A DCS controlling outputs for an axis was terminated because another instance of DCS was executed for the same axis.	This may be desired behavior. If not, change the application to use only one instance of DCS per axis.
x211	DCS output changed.	Either an Output Reference Input has changed, or the same Output Reference is specified for more than one DCS output on the same module in the system.	Ensure the same Output Reference is not specified for more than one Output for all MC_DigitalCamSwitch function blocks for a module.
x212	DCS aborted: FP output faulted.	A faceplate output associated with a DCS has faulted.	Correct fault condition on faceplate output.
x213	DCS aborted: FTB output faulted.	An FTB output associated with a DCS has faulted.	Correct fault condition on FTB output.
x214	DCS aborted: FTB faulted.	FTB associated with a DCS has faulted.	Correct FTB fault condition. Ensure the FTB is connected.
x215–x217	Configuration errors detected by set position operation.	Internal error.	Contact Technical Support.
x218	Cannot set position for a real axis external device; it is not configured.	There is no external encoder configured for the axis.	Configure an external encoder if one is needed or change the Encoder input to the MC_SetPosition function block.
x219	Servo Control Board not ready.	The Servo Control Board has either not been configured or an error has occurred that prevents it from controlling motion.	After power-up, allow time for the Servo Control Board to be configured by waiting for the axis OK bit. If an error has occurred on the Servo Control board, refer to that error for the method to clear the error.
x220	Touch Probe invalid feedback source.	An invalid position source has been chosen for the touch probe.	Change the position source for the touch probe to a valid source.

Error No. (Hex)	Description	Cause	Recommended Correction
x221	Specified trigger inactive; nothing to abort.	MC_TouchProbe not executed or Done on the axis specified.	This may be desired behavior. If not, check inputs on MC_AbortTrigger and MC_TouchProbe function blocks.
x222	Active Touch Probe aborted due to axis position going invalid.	Position was marked from valid to invalid with an active touch probe.	Disable touch probe prior to action that causes position to go invalid. Check cables to assure wiring issue did not cause position to go invalid
x224, x225	Internal error.	Internal error.	Contact Technical Support.
x228	Unable to communicate with FTB.	An FTB has been configured but is not communicating with the PMM.	Check power to FTB. Check fiber cable connecting FTB and PMM.
x229	Configured FTB unable to communicate.	An FTB has been configured but is not communicating with the PMM.	Check power to FTB. Check fiber cable connecting FTB and PMM.
x22A	FTB module hardware error.	FTB hardware is incompatible with PMM.	Contact Technical Support.
x22B	FTB module hardware error (stop axis).	FTB hardware is incompatible with PMM.	Contact Technical Support.
x22C	FTB identifier mismatch.	The FTB Identifier programmed on the attached FTB does not match the Identifier in the hardware configuration on the PMM.	Check that the correct FTB is connected to the faulting PMM. If so, correct the configured Identifier.
x22D	FTB identifier mismatch (stop axis)	The FTB Identifier programmed on the attached FTB does not match the Identifier in the hardware configuration on the PMM.	Check that the correct FTB is connected to the faulting PMM. If so, correct the configured Identifier.
x22E	Loss of FTB encoder power.	Short circuit or excessive current draw.	Check wiring and power specifications.
x22F	Overcurrent detected on faceplate output(s).	Short circuit on one or both faceplate outputs.	Check wiring.
x230	Open wire detected on faceplate input.	The specified 24Vdc input has an open wire error.	Check input wiring or disable open wire fault detection in HWC.
x232	Open wire detected on FTB input.	The specified 24Vdc input has an open wire error.	Check input wiring or disable open wire fault detection in HWC.
x23E	Open load detected on FTB output.	The specified 24Vdc output has an open load error.	Check wiring or disable open load fault detection in HWC.
x246	Power failure for FTB outputs 1–4.	There is no external power source for the 24Vdc outputs.	Check wiring or disable FTB OUT 1–4 Power External Connection detection in HWC.

Error No. (Hex)	Description	Cause	Recommended Correction
x247	Power failure for FTB outputs 5–8.	There is no external power source for the 24Vdc outputs.	Check wiring or disable FTB OUT 5–8 Power External Connection detection in HWC.
x248	Synthetic motors must be on higher numbered axes than real motors.	All synthetic motors (configured as drive type Synthetic) must be on higher-numbered axes than any real motor.	Adjust the Drive Type settings on the Axis tabs in HWC. The default drive type of Synthetic must be on higher numbered axes than real motors.
x249	Real servos must be contiguous starting at Axis 1.	All real motors must be configured on lower numbered axes than any synthetic motor (configured as drive type Synthetic).	Adjust the Drive Type settings on the Axis tabs in HWC. The default drive type of Synthetic must be on higher numbered axes than real motors.
x24B	Motor Velocity Limit is higher than supported by this drive type.	Motor Velocity Limit is set higher than supported by this drive type.	Reduce the Motor Velocity Limit to a value less than or equal to $1.1 \times$ the maximum velocity supported by the drive type.
x251	Positive Software End of Travel value from HWC is invalid.	Internal error.	Contact Technical Support.
x252	Negative Software End of Travel value from HWC is invalid.	Low Software EOT Limit is less than the Low Position Limit. The Low Software EOT Limit has been internally set equal to the Low Position Limit.	Adjust the Low Software EOT Limit, or the feedback source Low Position Limit parameter for this axis.
x253	External Device error	A quadrature error has occurred on the external feedback device.	Check external feedback device wiring.
x254	External Device error (stop axis)	A quadrature error has occurred on the external feedback device.	Check external feedback device wiring.
x255	A faceplate input function was assigned multiple times when not allowed	There is an error in the Faceplate I/O configuration.	Contact Technical Support.
x256	The Encoder B channel faceplate input must follow the Encoder A channel input.	There is an error in the Faceplate I/O configuration.	Contact Technical Support.
x257	IO Interrupt triggered too fast; Interrupt dropped.	An interrupt has exceeded the maximum frequency for I/O interrupts.	Ensure that interrupts do not exceed the maximum frequency, 0.2ms.
x258	Too many IO Interrupts waiting to be processed	I/O Interrupts are occurring faster than they are being processed.	Contact Technical Support.
x259	IO Interrupt was rejected by the RX3i controller. Will try to send again.	I/O Interrupt was refused by the system CPU.	Contact Technical Support.



Error No. (Hex)	Description	Cause	Recommended Correction
x25A	Interrupts suspended in the RX3i controller. Will try to send IO Interrupt later.	An I/O interrupt occurred while interrupts were suspended by the CPU via the SVC_REQ 32 function.	This warning notification may be prevented by disabling SVC_REQ 32, or preventing the interrupt while the service request is enabled.
x25B–x25D	Timed interrupt internal errors.	Internal error.	Contact Technical Support.
x25E	Interrupts suspended in the RX3i controller; Timed interrupt dropped.	A timed interrupt occurred while interrupts were suspended by the CPU via the SVC_REQ 32 function.	This error is a warning notification. It may be prevented by disabling SVC_REQ 32, or preventing the interrupt while the service request is enabled.
x25F	Interrupts suspended in RX3i CPU; Timed Interrupt dropped (stop).	A timed interrupt occurred while Interrupts were suspended by the CPU via the SVC_REQ 32 function.	A timeout occurred waiting for the CPU to service a Timed Interrupt. Interrupts should be disabled or suspended for less than the timeout period.
x260	A Timed Interrupt was dropped.	A timed interrupt occurred while Interrupts were suspended by the CPU via SVC_REQ 32.	A timeout occurred waiting for the CPU to service a Timed Interrupt. Interrupts should be disabled or suspended for less than the timeout period.
x261–x263	Interrupt internal errors.	Internal errors.	Contact Technical Support.
x264	Position is no longer valid. (PositionValid axis status bit is OFF.)	Feedback device is no longer referenced to known position	Assure that position referencing cycles are accruing at the correct time in the machine cycle. Check input wiring and feedback devices.
x265–x267	Interrupt internal errors.	Internal errors.	Contact Technical Support.
x26D	Synchronization lost.	Module has lost synchronization.	Contact Technical Support.
x26E	Short circuit detected on FTB output.	A short circuit has been detected on the specified 24Vdc output.	Check output wiring.
x274	Encoder channel open wire error (FTB).	An open wire error has occurred on an external encoder input on the FTB I/O	Check input wiring.
x275	Encoder channel open wire error (FTB) – stop axis.	An open wire error has occurred on an external encoder input on the FTB I/O.	Check input wiring.
x276	Encoder channel open wire error (faceplate).	An open wire error has occurred on an external encoder input on the faceplate I/O.	Check input wiring.
x277	Encoder channel open wire error (stop axis).	An open wire error has occurred on an external encoder input.	Check input wiring.
x278	Unable to access encoder inputs.	External encoder data lost due to FTB communication errors.	Check FTB fiber cable and connections.
x279	Unable to access encoder inputs (stop axis).	External encoder data lost due to FTB communication errors.	Check FTB fiber cable and power connections.

Error No. (Hex)	Description	Cause	Recommended Correction
x27A	Unable to set absolute position; encoder did not retain position.	The encoder battery and capacitor failed to retain the encoder position.	Replace the encoder battery or do not turn off the servo drive power for more than a few minutes (capacitor can maintain position for a short time) to prevent this error in the future. Use MC_Home or MC_SetPosition to re-establish a valid position.
x27B	Unable to set absolute position; previous hardware configuration changed.	One of the following hardware configuration parameters has changed preventing absolute position from being restored: Motor Encoder User Units, Motor Encoder Counts, Motor Encoder Counts Per Motor Revolution, or Axis Direction.	This error will only occur one time after one of the hardware configuration parameters listed have changed. Use MC_Home or MC_SetPosition to re-establish a valid position.
x27C	Unable to set absolute position, motor has moved more than 1024 revolutions	While the module was not operating the motor has turned more than 16384 revolutions or the motor encoder has been replaced. The absolute position cannot be restored.	Use MC_Home or MC_SetPosition to re-establish a valid position.
x27D	Servo velocity command clamped to Motor Velocity Limit.	The servo velocity command has exceeded and subsequently clamped to the Motor Velocity Limit.	Increase the Motor Velocity Limit. Reduce the Max Velocity Application (parameter no. 9) to clamp Commanded Velocity to a lower value.
x27E	Servo velocity command clamped to Motor Velocity Limit multiple times.	The servo velocity command has exceeded and subsequently clamped to the Motor Velocity Limit multiple times during the past minute. This error will be logged once per minute until the limit is not exceeded for a period of one minute.	Increase the Motor Velocity Limit. Reduce the Max Velocity Application in order to clamp Commanded Velocity to a lower value.
x27F	DCS new command already pending, commands for same axis coming too fast	Multiple DCS commands sent to the same axis too quickly. A new DCS command must complete processing before another one is sent.	Delay several milliseconds between DCS commands to the same axis. One normally would not want to send back-to-back DCS commands to the same axis.
x280	Position-compensated switch point overlapped another switch point.	A DCS position-compensated switch point overlapped another switch point.	Adjust DCS switch points so they will not overlap at the highest velocities expected.
x281	Duration switch point overlapped another switch point.	A time-based (Duration) switch point overlapped another switch point.	Adjust DCS switch points so they will not overlap at the highest velocities expected.
x282	Hysteresis and On/Off Compensation not allowed together.	Attempt to use Hysteresis and On/Off Compensation together.	Do not use Hysteresis and On/Off Compensation on the same DCS track.

Error No. (Hex)	Description	Cause	Recommended Correction
x283	Hysteresis and Duration switch points not allowed together.	Attempt to use a Hysteresis track option with time-based (Duration) switch point(s).	Use Hysteresis only with position-based switch points.
x284	Hysteresis and single direction switch points not allowed together.	Attempt to use Hysteresis track option and single direction switch point together.	Use Hysteresis only with switches configured for both directions (AxisDirection = 0).
x285	On/Off Compensation and Duration switch points not allowed together.	Attempt to use On/Off Compensation and time-based (Duration) switch point together.	Do not use On/Off Compensation with time-based switch points.
x286	DCS aborted due to change in scaling, low position limit, or position range.	DCS was aborted due to change in an axis operational parameter value such as scaling, low position limit, or position range.	Do not change axis parameters while a DCS is active.
x287	Negative On/Off compensation and Rotary axis mode not allowed together.	Attempt to use Negative On/Off compensation and Rotary axis mode together (includes Axis 5 with Actual Position).	DCS tracks must have On Compensation and Off Compensation values $\geq 0$ when used with a Rotary axis.
X288	Unable to set absolute position, position valid never established for axis	PMM is configured for absolute position mode but position valid has never been set on the PMM	Perform a home cycle to establish a valid position
X289	Unable to set absolute position, drive home position not set for axis	PMM is configured for absolute position mode but the drive itself has never completed a home cycle.	Perform a home cycle to establish a valid position
X290	Motor encoder does not support absolute position	PMM is configured for absolute position mode, but the connected motor does not have a multi-turn absolute encoder connected.	Configure the axis using Incremental position mode or replace the motor with a multi-turn absolute encoder.
x300	Timeout while waiting for power feedback from the servo drive	The MC_Power function block outputs this warning if the Enable input of the function block is set to TRUE indicating that power is to be applied at the servo drive and 500ms has elapsed without feedback indicating that the servo drive is powered up.	If the axis state does not transition to Standstill, there is a problem turning the servo drive power on. Verify that the proper level of AC voltage is supplied to the servo drive and that the servo drive E-Stop has not been activated. Examine the fault table and the event queue for errors indicating the nature of the problem powering up the servo drive.

Error No. (Hex)	Description	Cause	Recommended Correction
x301	Current axis state does not allow function block	The PLCopen state does not support execution of the function block receiving the error.	Use the PLCopen state diagram to determine the states that allow this the function block to be issued. Change the application to have the axis in the proper state when the function block is used.
x303	MC_Reset for an axis issued before an active MC_Reset has completed.	An MC_Reset can take significant time to complete processing if for example it has to clear a servo controller error. Any MC_Reset issued while another MC_Reset has not yet completed will receive this error.	Examine the application and assure that any MC_Reset completes (gets a Done or Error output) before issuing another MC_Reset.
x304	Servo unable to reset.	During the processing of an MC_Reset, the axis hardware indicated an error such that the reset could not complete successfully.	Examine the fault table and the event queue for errors indicating the nature of the problem resetting the axis.
x305	Axis in position lag limit.	This warning is output to a function block if the axis reaches the configured Max Position Lag when Position Lag Monitoring is enabled.	If position lag is expected in the application, this warning can be ignored. If Max Position Lag is set such that it should not be encountered by the application, or if this warning is followed by a Position Error Limit error, then look into what is driving position error in the axis. Tuning the servo may help. If the servo is properly tuned, ensure the motor is properly sized for the application. Also look for external interference with the motor that may be causing the position lag.
x306	Function block attempting motion when axis position not valid.	The motion function block was issued before the axis position was valid.	Axis position can be made valid using a home cycle under the control of MC_Home or MC_SetPosition with an absolute position. If this error is received on an absolute axis, check the fault table and event queue for battery errors or other errors from the axis.
x307	Initializing path generation for axis failed.	An internal error occurred while initializing path generation for the axis.	Contact Technical Support.

Error No. (Hex)	Description	Cause	Recommended Correction
x308	Attempt to move in direction not allowed by the application.	Movement in a direction not allowed by the direction inputs of the MC_Power that is active on the axis has exceeded the Disabled Direction Deadband.	If motion is desired, change the direction constraints on the MC_Power (the Enable_Positive and Enable_Negative inputs) to support the specified direction. If the problem occurs when a CAM or Gear function block is used with an external encoder master axis, consider increasing the value of the Disabled Direction Deadband configuration item.
x309	MC_SetPosition failed in updating positions.	Processing an MC_SetPosition encountered an error in updating the position for the encoder in question.	Examine the fault table and the event queue for errors indicating the nature of the problem setting the axis position.
X30A	Buffered MC_Power instance aborted by another buffered command	MC_Power function pending buffered command was aborted by another function block	Examine application for instances where this can occur and correct as appropriate
x30B	Move Superimposed not supported on active command.	An MC_MoveSuperimposed was issued when the active motion command on the axis was not MC_GearIn. At this time, an MC_MoveSuperimposed is supported only if an MC_GearIn is active.	Examine the application and ensure that MC_MoveSuperimposed is used only when MC_GearIn is active.
X30C	MC_Phasing not supported on active command.	MC_Phasing can only be used when an MC_CamIn is active. An MC_Phasing command was issued when no MC_CamIn was active.	Examine the application and ensure that MC_Phasing is only used when an MC_CamIn is active.
X30D	Timeout while waiting for axis to reset.	An MC_Reset failed to complete its processing within the 10 second timeout.	This is an internal error that should never be encountered. If it occurs, contact Technical Support.
X30E	Master axis on new command does not match active master.	Indicates that an MC_Phasing instance was used with a master axis that was not the master axis currently being followed by the CAM.	The phasing will be applied to the currently active master axis. If this is not the desired behavior, check the application to ensure that the master for the CAM and the master for the phasing are the same.

Error No. (Hex)	Description	Cause	Recommended Correction
X30F	An RX3i controller mode change aborted a function block.	Motion will stop any time the RX3i controller goes to Stop or to Run/Output Disabled mode. This error will be given on all active MFBs when that occurs.	These errors are due to a change in the status of the RX3i controller. Stop all active motion commands before transitioning the RX3i to Stop mode.  <b>Note:</b> <i>An MC_Reset or MC_ModuleReset must be performed in order for motion to resume after the RX3i returns to Run/Output Enabled mode.</i>
X310	Function block terminated due to error stop on the axis.	An error caused the axis to go to ErrorStop state using a normal error stop.	Examine the fault table and the event queue for the errors that sent the axis into normal error stop.
X311	Function block terminated due to fast stop on the axis	An error caused the axis to go to ErrorStop state using a fast error stop.	Examine the fault table and the event queue for the errors that sent the axis into fast error stop.
X313	Error disengaging a CAM	An MC_CamOut encountered an internal error when disengaging the CAM or gear.	Contact Technical Support.
X314	The specified buffer mode is not valid with the active command.	The BufferMode specified is not allowed with the active function block.	For a summary of buffer modes that can be used with each function block, refer to Buffer Mode in Section 5.3.3 Valid.
X315	Cannot reach specified velocity at specified position.	The move cannot reach the specified velocity within the distance constraint set by the position.	Increase the acceleration or deceleration and jerk, decrease the final velocity, or increase the distance for the move.
X316	Insufficient distance to accelerate to meet end conditions of move.	The move cannot reach its end conditions within the distance available.	Increase the acceleration or deceleration and jerk, decrease the final velocity, or increase the distance for the move.
X317	Backup was required to reach the specified position.	In order to arrive at the specified position, the axis overshot its commanded position and then reversed to attain the final position.	If backup is not desired, adjust the velocity, acceleration, deceleration and jerk so that the axis can reach the desired position without overshooting.

Error No. (Hex)	Description	Cause	Recommended Correction
X318	Jerk constraint could not be maintained due to other constraints.	This warning indicates that jerk was internally modified in order to reach the specified position and meet the specified velocity and acceleration/deceleration constraints.	<p>If transitioning from Synchronous motion state to a non-synchronous state, the commanded jerk may be too small relative to the initial acceleration. Transitioning at a time when commanded acceleration is smaller or increasing commanded jerk can remedy this.</p> <p>If aborting a jerk-limited command with one that has a reduced jerk value, the remainder of the move may continue to use the original jerk so that the move does not exceed other move constraints (for example maximum velocity). Increasing the aborting jerk to a value greater than or equal to the original jerk or aborting the first command sooner can prevent this warning. If jerk-limited motion is not necessary for your application, using unlimited jerk (Jerk = 0) will prevent this warning.</p>
X319	Blending specified with no active command, ignored.	The BufferMode parameter of the function block specified blending and there was no active command on the axis to blend to. The buffer mode is ignored in this case.	If blending was expected, examine the application to find out why the command that was expected to be active was not. The command may have completed in error. The fault table and event queue may be used to determine what error occurred.
X31A	Master position not available to slave.	The CamIn, GearIn, or GearInPos function block specified a slave axis that could not be commanded because the position information about the master was unavailable or invalid.	<p>Ensure the master axis position is valid. This can be determined by monitoring the Position Valid flag (PN1201) for the master axis. If Axis 5 is the master, also monitor the Aux Position Valid flag (PN1228). For descriptions of these parameters, refer to Section 6.42.1.</p> <p>Ensure that the module containing the master axis is present and operational (check the fault table for a loss of module and a module present input from the master module)</p>

Error No. (Hex)	Description	Cause	Recommended Correction
x31B	Master velocity not available to slave.	The slave axis could not be commanded because the velocity information from the master was unavailable or invalid.	Ensure the master axis position is valid. If not make the master position valid. Ensure the module containing the master axis is present and operational (check the fault table for a loss of module and a module present input from the master module)
x31C	Slave axis failed to transition into synchronized motion.	The slave axis attempted to transition into the Synchronized Motion state and failed.	Contact Technical Support.
X31D	Slave axis failed to transition from synchronized motion.	The slave axis attempted to transition out of the Synchronized Motion state and failed.	Contact Technical Support.
X31F	Slave axis failed to synchronize with its master.	Ramp distance has been transited and the master is at the master sync position and the slave has been unable to reach the slave sync position.	Increase the ramp distance or master start distance. Increase the velocity, acceleration or deceleration on the slave axis to allow it to ramp faster.
X320	A synchronous motion command failed to ramp because the master violated conditions or the slave was too limited to keep up.	The slave failed to synchronize because:  The master is linear, and already at the master sync position but moving away from it.  The master is linear and already at the Master Sync Position, but the master start distance isn't zero.	If you expect the linear master to already be at the Master Sync Position before executing MC_GearInPos, you can execute with Master Start Distance of zero. Back the master up or set the Master Sync Position further out. A linear master requires the master to be some distance away from and moving toward the master sync position. The slave will not move to the Slave Sync Position unless the master is moving (even if the master is already at the Master Sync Position).
X321	The end of the ramp (master position + ramp distance) was found to be off the CAM profile.	The end of the ramp (master position + ramp distance) was found to be off the CAM profile.	Reposition the master so that the master position + ramp distance is within the bounds of the CAM profile. Make the ramp distance smaller. Make the profile larger.
X322	A synchronous motion command failed because the master backed up past its initial position while ramping	If the function block begins ramping and the master changes direction and continues beyond the master sync position dead band from the master start position.	Change the master to not back up beyond the dead band limit. Increase the dead band to allow the backup that is occurring.



Error No. (Hex)	Description	Cause	Recommended Correction
X323	On MC_CamIn, the slave was limited to stay within the bounds of the profile while ramping.	The StartMode input specifies Limit Slave Ramp to Profile Range and the slave is outside the profile range or trying to leave the profile range.	Specify a different start mode. Position the slave axis so that the slave stays within the profile range. Adjust the ramp constraints so that the slave synchronizes within the profile.
X324	Master axis cannot be changed while slave is in Synchronized Motion	The axis was in synchronous motion and another synchronous motion command was received specifying a different master.	Change the application to use the same master or exit the Synchronous Motion state before engaging the new master.
X325	Discarding an update to a Stop command that is no longer executing.	An error terminated an MC_Stop at the time the Execute input of the MC_Stop was transitioning low.	Information only; no action required.
X326	A pending synchronous motion command was terminated without running.	A new MFB was executed while a synchronous motion command was pending.	This may be intended operation. If it is not, restructure the application so that either the synchronous motion command acts immediately, or the following command is conditioned on the synchronous motion command and begins ramping before it executes.
X327	A synchronous command will not engage because the master is moving in the wrong direction.	A pending synchronous motion command has determined that the master will have to turn around in order to engage.	Reverse the master direction.
X328	A Buffered or Blended FB terminated without running.	A buffered/blended command was terminated before it ever controlled the axis. This can occur if a subsequent command is issued before the buffered/blended command assumes control over the axis.	This may be intended operation. If it is not, restructure the application to avoid terminating the pending/buffered/blended command.
X329	MC_CamIn with zero ramp distance but master/slave not already synchronized	MC_CamIn specified a ramp distance of zero and the position of the slave is not on the profile such that the master and slave are in sync.	Move the slave to the proper place (within tolerance based on application maximums for velocity and acceleration of the slave axis) or specify a non-zero ramp distance.
X32A	Axis power was forced off due to an ErrorStop.	An active MC_Power has enabled power to the axis, but the axis is disabled due to an error.	Correct the cause of the ErrorStop and then issue an MC_Reset. This should cause the drive to be re-enabled and to transition to the Standstill state.

Error No. (Hex)	Description	Cause	Recommended Correction
X32B	MC_MoveVelocity has reached the limit of the axis and has stopped.	An MC_MoveVelocity has reached either the software end of travel or the position limit of the axis. The axis has stopped.	If the axis is linear and EOTs are enabled, assure that something controls the motion before reaching EOT. If the axis is rotary or EOTs are disabled, a warning output is given when the axis reaches half the possible range for the move. Use MC_SetPosition in Absolute mode to reset the coordinate system at a convenient point before reaching EOT.
X32C	Blending specified but not supported by the command; treated as buffered.	A buffer mode specifying blending was used and blending cannot be performed. This error may occur if either the command being issued or the command that is active does not support blending. The buffer mode has been treated as Buffered.	This warning may indicate an attempt to blend to an unexpected command. It may not be an error but should be investigated.
X32D	Axis has reached halfway to the end of the range during an MC_MoveVelocity	Motion has reached halfway to either the software end of travel or the position limit of the axis.	This warning may be used to trigger resetting the coordinate system for a rotary axis as described for error x32B.
x32E	Unrecoverable calculation error.	An unrecoverable calculation error has occurred. This may be caused if an in-process move is interrupted by a new move with a Jerk command value too small to allow the new move to complete without exceeding its maximum velocity.	This condition is indicated in the complete fault code by a value of ff c9 in the location shown below. Sample fault code 63 2e 37 21 01 a5 ff c9 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 To correct this problem, increase the Jerk command value in the new move so that the move can complete successfully. For other fault codes, contact Technical Support.
X32F	Jog when axis is at EOT and jog direction not away from EOT.	The axis is on a hardware or software EOT and an MC_JogAxis is enabled in the direction of the active EOT.	Change the input to the jog to command motion back inside the Software EOT limits.
X330	Function block not allowed in the Disabled state.	A function block was issued in the Disabled state and the PLCopen state machine does not support its use there.	Refer to Section 5.5, Axis States to determine which states allow the function block to be issued. Change the application logic to place the axis in the proper state before issuing the function block in error.

Error No. (Hex)	Description	Cause	Recommended Correction
X331	Function block not allowed in the Standstill state.	A function block was issued in the Standstill state and the PLCopen state machine does not support its use there.	Refer to Section 5.5, Axis States to determine which states allow the function block to be issued. Change the application logic to place the axis in the proper state before issuing the function block in error.
X332	Function block not allowed in the Homing state.	A function block was issued in the Homing state and the PLCopen state machine does not support its use there.	Refer to Section 5.5, Axis States to determine which states allow the function block to be issued. Change the application logic to place the axis in the proper state before issuing the function block in error.
X333	Function block not allowed in the ErrorStop state.	A function block was issued in the ErrorStop state and the PLCopen state machine does not support its use there.	Refer to Section 5.5, Axis States to determine which states allow the function block to be issued. Change the application logic to place the axis in the proper state before issuing the function block in error.
X334	Function block not allowed in the Stopping state.	A function block was issued in the Stopping state and the PLCopen state machine does not support its use there.	Refer to Section 5.5, Axis States to determine which states allow the function block to be issued. Change the application logic to place the axis in the proper state before issuing the function block in error.
X335	Function block not allowed in the Discrete Motion state.	A function block was issued in the Discrete Motion state and the PLCopen state machine does not support its use there.	Refer to Section 5.5, Axis States to determine which states allow the function block to be issued. Change the application logic to place the axis in the proper state before issuing the function block in error.
X336	Function block not allowed in the Continuous Motion state.	A function block was issued in the Continuous Motion state and the PLCopen state machine does not support its use there.	Refer to Section 5.5, Axis States to determine which states allow the function block to be issued. Change the application logic to place the axis in the proper state before issuing the function block in error.
X337	Function block not allowed in the Synchronous Motion state.	A function block was issued in the Synchronous Motion state and the PLCopen state machine does not support its use there.	Refer to Section 5.5, Axis States to determine which states allow the function block to be issued. Change the application logic to place the axis in the proper state before issuing the function block in error.
X338	Function block not allowed in the Setup state.	A function block was issued in the Setup state and the PLCopen state machine does not support its use there.	Refer to Section 5.5, Axis States to determine which states allow the function block to be issued. Change the application logic to place the axis in the proper state before issuing the function block in error.

Error No. (Hex)	Description	Cause	Recommended Correction
X339	Function block not allowed in the Jogging state.	A function block was issued in the Jogging state and the PLCopen state machine does not support its use there.	Refer to Section 5.5, Axis States to determine which states allow the function block to be issued. Change the application logic to place the axis in the proper state before issuing the function block in error.
X33A	MC_CamIn aborted due to DLB stop.	An MC_CamIn in a DLB was terminated due to the DLB being stopped.	This is an expected error if an MC_CamIn is active when a DLB is stopped.
X33B	An MC_CamIn from the application may not use a CAM ID from a DLB.	An MC_CamIn in the main logic referenced a CAM ID from a DLB. This reference is not allowed.	The application should perform an MC_CamTableSelect for the desired CAM and use the returned CAM ID. To prevent possible conflicts, DLB logic should not reuse CAM ID variables from the main logic.
X33C	MC_CamOut cannot be executed while MC_CamIn is not the active command.	An MC_CamOut was issued while the axis was in Synchronous Motion state but the motion was controlled by either an MC_GearIn or an MC_GearInPos.	Change the application to use MC_GearOut to disengage the synchronous motion
x33D	MC_GearOut cannot be executed while MC_GearIn or MC_GearInPos is not the active command.	An MC_GearOut was issued while the axis was in Synchronous Motion state but the motion was controlled by an MC_CamIn.	Change the application to use an MC_CamOut to disengage the synchronous motion.
X33E	Master position is not within CAM profile range.	On an MC_CamIn, master position is not on profile (occurs when ramp distance = 0 and master position is outside profile range).	If using a ramp distance of zero, position the master within the master's CAM profile range. If you wish to ramp, specify a ramp distance such that the master's position plus the ramp distance is on the profile.
X33F	Master's axis configuration (low limit, range, or EOTs) changed while pending or engaged on a CAM or gear.	On a synchronous motion command, the master's axis configuration changed after the MC_CamIn or MC_GearInPos was executed. This means either the master's low limit/range or software EOTs changed while a synchronous command was active or pending.	If you must change the master's hardware configuration, disengage the synchronous command first. Execute an MC_CamOut, MC_GearOut, or abort the slave axis with another command before changing the master's axis configuration.
X340	CAM master is too fast or profile too small. Crossed master profile in a single sample.	An axis on a CAM transited the CAM profile in a single sample period. Either the profile is too small or the master was moving faster than anticipated.	Slow the master or change the profile.
X341	MC_Phasing not allowed on an MC_CamIn that is ramping (not InSync)	MC_Phasing was not permitted because the underlying MC_CamIn was not yet synchronized.	Wait to execute MC_Phasing until MC_CamIn is InSync.

Error No. (Hex)	Description	Cause	Recommended Correction
X342	Direction constraints will be ignored until axis reaches Disabled, Standstill or ErrorStop states	This warning occurs when the direction constraints on an MC_Power function block were changed while the axis was in motion. The new direction constraints will be applied only when the axis stops moving.	This may not be an error but may instead be the intended operation of the application. Confirm that the change in the direction constraints can wait until axis motion stops.
X343	Position could not be reached via Direction passed to MC_MoveAbsolute . Direction parameter ignored.	For MC_MoveAbsolute on an axis configured with a Linear positioning mode, the direction to move is implied by the destination position. (The Direction parameter on the MC_MoveAbsolute is superfluous when the axis is configured in Linear mode.) Therefore, if the Direction parameter implies a direction opposite from what is required to move to the destination position, the Direction parameter is ignored, this warning is given, and the axis is moved toward the destination position.	Change the Direction input to MC_MoveAbsolute to ShortestWay, which will never result in this warning. Alternatively, specify the correct direction in the call to MC_MoveAbsolute.
X344	RPM specified for Force Digital Servo Velocity exceeds motor limit	RPM in parameter 1311 to force digital servo velocity exceeds the configured maximum for the motor.	Reduce the specified RPM in the parameter or change the hardware configuration to increase the motor RPM
x345	Commanded velocity exceeds application limit, clamped at application limit.	The commanded velocity is greater than the application maximum velocity.	Often an MC_SetOverride with greater than 100% velocity factor may be the cause. The other possible cause is a change to the application maximum velocity. In either case either decrease the override percent or increase the maximum.
X346	Commanded acceleration or deceleration exceeds application limit, clamped at application limit.	The commanded acceleration or deceleration is greater than the application maximum acceleration.	Often an MC_SetOverride with greater than 100% acceleration factor may be the cause. The other possible cause is a change to the application maximum acceleration. In either case either decrease the override percent or increase the maximum.
X347	Jerk exceeds application limit, clamped at application limit.	The commanded jerk is greater than the configured jerk limit.	An MC_SetOverride with greater than 100% jerk factor may be the cause. The other possible cause is a change to the maximum jerk. In either case either decrease the override percent or increase the maximum.

Error No. (Hex)	Description	Cause	Recommended Correction
X348	Attempt to move past absolute limits.	Attempt to position past the largest value possible.	Change coordinate system of the application such that all required moves are within the maximum range of the axis.
X349	Error dispatching buffered move, axis stopped.	A buffered move was dispatched and was unable to successfully take control of the axis due to an error.	Investigate the error on the buffered move to determine why it could not run successfully. Correct that error so that this error will not occur.
X34A	MC_Power instance was superseded by another.	An active MC_Power instance has been superseded by another. Generally, this is caused by inadvertently using the same instance name or axis reference on two instances.	Change the logic to use only a single instance of MC_Power unless intentionally superseding one instance with another. Superseding one instance of MC_Power with another is not recommended.
X34B	MC_SetOverride instance was superseded by another.	An active MC_SetOverride instance has been superseded by another. Generally, this is caused by inadvertently using the same instance name or axis reference on two instances.	Change the logic to use only a single instance of MC_SetOverride unless intentionally superseding one instance with another. Superseding one instance of MC_SetOverride with another is not recommended.
X34C	MC_JogAxis instance was superseded by another.	An active MC_JogAxis instance has been superseded by another. Generally, this is caused by inadvertently using the same instance name or axis reference on two instances.	Change the logic to use only a single instance of MC_JogAxis unless intentionally superseding one instance with another. Superseding one instance of MC_JogAxis with another is not recommended.
X34D	MC_ModuleReset failed to reset an axis because the axis was not stopped with the power turned off.	An MC_ModuleReset attempted to reset an axis that was in error or had been disabled but the axis was not yet at zero velocity with the power off. The axis in question was not reset and remains in ErrorStop state.	Allow all axes that are disabled or in error to have time to stop and power off before issuing the MC_ModuleReset, or individually reset any axes that are still in ErrorStop.
X34E	MC_SetPosition aborted by a subsequent MC_SetPosition.	An active MC_SetPosition instance has been superseded by another. Generally, this is caused by issuing a second MC_SetPosition before the Done output from a prior MC_SetPosition has gone true.	This may be an expected error if two MC_SetPosition instances are activated in rapid succession. If it is unexpected, the second MC_SetPosition should be interlocked with the Done output of the first.
X34F	MC_SetPosition timeout due to other commands at path generator.	An MC_SetPosition could not be processed due to other commands at the path generator. It was retried a number of times before this error was given.	This error occurs only if moves are occurring for the axis every sample period for a large number of sample periods. Change the application so that no moves are issued until the Done output of the set position (MC_SetPosition) is true.

Error No. (Hex)	Description	Cause	Recommended Correction
X350	Reset of an axis failed because the servo was not ready.	An MC_Reset was issued and the servo controller remained in error after the reset was processed.	Correct the underlying issue with the servo and reissue the MC_Reset.
X351	Reset of an axis failed because a required FTB was not ready.	An MC_ModuleReset was issued and the FTB was not operational at the time of the reset.	Make the FTB operational and re-issue the MC_ModuleReset or change the hardware configuration so that the FTB is not required.
X352	Overrides are ignored while the axis is in Synchronized Motion state.	The axis has entered Synchronized Motion state and has an enabled MC_SetOverride.	Disable the MC_SetOverride when axis is in the synchronous state.
X353	MC_GearInPos with zero Master Start Distance. Master stopped at MasterSyncPosition but slave not already at SlaveSyncPosition.	MC_GearInPos executed with 0 MasterStartDistance. The master's position is exactly MasterSyncPosition and the master axis is not moving. The slave's position is too far away from SlaveSyncPosition to be considered InSync and GearInPos cannot ramp because there is no distance between the master's current position and MasterSyncPosition.	If intending to use MC_GearInPos without ramping, set both the master and slave positions to their Sync positions before executing the function block. If this error occurs even though the slave axis position is close to the SlaveSyncPosition, you can increase the positional tolerance by increasing the velocity input to the function block.
X354	A non-zero HomeOffset is not allowed when the encoder is not used for feedback.	A HomeOffset parameter is set non-zero for an MC_Home applied to a virtual axis.	Change the application to use a zero-home offset. Since the encoder is not used for feedback on the virtual axis, a home offset cannot be applied.
X355	The axis was in position lag error at some point during a controlled stop. As a result, the PMM firmware had to take special actions to ensure that the axis is stopped quickly despite this condition.	Position lag was encountered while attempting to stop the axis in a controlled manner (e.g. Normal ErrorStop, MC_Stop, MC_Halt, etc.)	This is an informational note that provides additional information on the Position Lag encountered by the axis. The fact that the axis experienced Position Lag was separately recorded. See that item for what to do about Position Lag.
X356	The CAM profile used by this MC_CamIn has been modified or deleted on the controller since it was selected.	MC_CamIn was executed using a CAM profile that has been modified or deleted from the controller since it was selected on the PMM.	If this is expected operation, ignore the error. If you intended to engage the modified CAM profile, re-execute MC_CamTableSelect and re-engage MC_CamIn.
X357	Servo errors were cleared, but new errors are now present.	An MC_Reset was issued and upon completion detected servo controller errors that do not cause an axis to stop motion.	Examine the fault table and the event queue for errors that indicate a Servo Controller problem but do not stop motion.

Error No. (Hex)	Description	Cause	Recommended Correction
X358	MC_GearInPos executed with an invalid Master Start Position.	MC_GearInPos executed with an invalid Master Start Position (MasterSyncPos – MasterStartDistance). Example: Master Start Position outside the EOT limits of a linear master.	Examine High and Low Software EOT parameters, MasterSyncPosition and MasterStartPosition inputs. If MasterSyncPosition minus MasterStartDistance is outside either EOT, make appropriate adjustments and re-execute MC_GearInPos.
X359	Unable to plan command acceleration much greater than deceleration.	The acceleration on a command so far exceeded the deceleration that the command accelerates to a speed such that the deceleration cannot stop the axis within the specified distance for the move.	Increase deceleration and/or distance
x35A	New SetOverride inputs applied for more than the maximum number of consecutive sample periods.	Possibly the inputs on an instance of SetOverride are being changed such that the SetOverride is scanned in the controller with new inputs on many consecutive sample periods of the PMM. Alternatively, multiple instances of SetOverride are being executed such that a new one arrives each consecutive sample period for many sample periods.	Decrease the frequency of changes to SetOverride.
X35B	Jog attempted while MC_Reset /MC_ModuleReset was still in-progress.	A jog was initiated while an MC_ModuleReset or MC_Reset was still being processed. Note that it is possible for this to occur when the MC_Reset is on a different axis than the jog.	Ensure any previous MC_Reset or MC_ModuleReset has completed before initiating the jog.
X35C	Move requires a backup to reach commanded position but would cause motion in direction that is not allowed	The move was unable to reach its final commanded position due to direction limitation on MC_Power.	Allow movement in both directions or ensure that the move's inputs and initial values do not require the axis to back up to reach its final position.
X35D	Maximum acceleration time has been exceeded	Maximum time allowed during acceleration phase of move has been exceeded.	Increase acceleration and/or jerk.
X35E	Maximum deceleration time has been exceeded.	Maximum time allowed during deceleration phase of move has been exceeded.	Increase deceleration and/or jerk.



Error No. (Hex)	Description	Cause	Recommended Correction
X35F	Initial conditions for blend preclude planning the blend.	The move cannot be planned successfully due to the initial velocity and blending velocity constraints.	This error typically occurs when a move that requires blending is currently in progress and a set override command is issued that reduces the current moves' maximum velocity to a value less than the blending velocity. Either change the blending mode or execute the set override before the first move in the blending sequence starts.
X360	MC_SetOverride has been disabled since the axis was disabled or in error.	Override percentages are not active during an ErrorStop or while the axis is Disabled. These values do not persist through the ErrorStop or Disabled states.	Clear the error and reset the axis (if applicable). Power-on the axis. Reapply the override by executing MC_SetOverride with its Enable input ON.
X361	A blended move cannot change the axis direction.	A move that would change the axis direction has been executed in one of the Blending modes. A Blending move that changes axis direction is not allowed.	If it is necessary to change motion direction, use the Aborting or Buffered value for the MC_BufferMode input of the buffered function block.
X362	An MC_SetPosition was attempted that would make the active move infeasible.	An MC_SetPosition was executed with parameters that would cause invalid movement.	Change the Position input on MC_SetPosition or the constraints of the move (for example, enable both directions on MC_Power).
X363	Jerk value too small to complete move without exceeding constraints.	An in-process move was interrupted by a new move that has a Jerk input value too small to allow completion without exceeding its maximum velocity.	Increase the value of the Jerk input parameter.
X402	High Software EOT is less than the current commanded position.	The High Software EOT was changed to a value less than the current commanded position.	Modify the High Software EOT to a value greater than the current commanded position or move the commanded position to a lower value before writing the High Software EOT.
X403	Low Software EOT is greater than the current commanded position.	The Low Software EOT was written to a value greater than the current commanded position.	Modify the Low Software EOT to a value less than the current commanded position or move the commanded position to a higher value before writing the Low Software EOT.

Error No. (Hex)	Description	Cause	Recommended Correction
X404	Parameter value invalid with the I/O configuration in HWC.	<p>A write parameter command is being used to write an invalid value, based on the current hardware configuration.</p> <p>Examples:</p> <p>Attempting to use MC_WriteDWordParameter to write an FTB or Faceplate output point when that point is configured as an input.</p> <p>Attempting to use a write parameters command to write a max position lag that is greater than the max position error.</p> <p>An MC_Reset or MC_ModuleReset must be performed in order for motion to resume after the RX3i returns to Run/Output Enabled mode.</p> <hr/> <p><b>Note:</b> <i>If, when writing PN2114 and PN2115, any configurable I/O is configured as an Input, the Warning output will be active along with ErrorID of x404 on the function block output. This is normal operation; it is simply letting you know that there is at least one Input involved.</i></p>	Modify the hardware configuration or change the parameter values accordingly.
X405	Invalid Output point mask	The user has specified that an invalid output point should be turned on using the write parameter interface. For example, to turn on the two faceplate outputs, the user specifies a mask of 0x03, If the user specifies a mask of 0x0F for the faceplate inputs, only bits 0 & 1 are valid; bits 2 & 3 are referencing invalid output points.	Modify the output mask that has been specified.

Error No. (Hex)	Description	Cause	Recommended Correction
X406	A floating-point argument to a function block is NaN	The floating-point value not a number (NaN) was used as an input argument to a function block. This value is invalid.	Ensure that floating-point computations used as input to the function block do not yield a NaN result. (Note that NaN cannot be entered as a numeric literal, so it must result from computation in the application).
X407	Axis parameter is used more than once in Axis Array.	A function block instance has an Axis Array that uses an Axis parameter more than once.	Axes can only be used once in an Axis Array. Remove the duplicate entries.
X408	Invalid function block Master Axis parameter.	A function block instance has an invalid Master Axis parameter.	Ensure the Master input parameter is linked to a configured axis that it is different from the Slave axis.
X409	Invalid function block Axis parameter.	A function block instance has an invalid Axis parameter.  <i>Note: For MC_DigitalCamS witch: Error occurred when there was already an active DCS function block running on the axis. This results in the error being a Normal Stop error.</i>	Ensure the Axis input parameter value is linked to a configured axis.
X40B	Invalid function block BufferMode parameter.	A function block instance has an invalid BufferMode parameter.	Select a valid MC_BufferMode value for this function block.
X40C	Invalid function block Length parameter.	A function block instance has an invalid Length parameter.	The maximum length allowed is 16. Select a length between 1 and 16 that corresponds to the number of parameters being written.
X40D	Invalid function block Output parameter.	A function block instance has an invalid Output parameter.	Ensure the Output parameter is configured in the HWC for this module.
X40E	Invalid function block Position parameter.	A function block was executed with an invalid Position parameter.	Ensure the position range is configured correctly in HWC. For a linear axis, the valid range is between the High and Low Software EOT values, if enabled. For a rotary axis, the valid range is defined by the Low Position Limit and the Position Range
x40F	Invalid MC_SetOverride block VelFactor parameter.	A function block was executed with an invalid VelFactor parameter.	The valid range is 0, or 0.01 to 1.2. Select a value in this range.
X410	Invalid MC_SetOverride block AccFactor parameter.	A function block was executed with an invalid AccFactor parameter.	The valid range is 0.01 to 1.2. Select a value in this range.

Error No. (Hex)	Description	Cause	Recommended Correction
X411	Invalid MC_SetOverride JerkFactor parameter.	A function block was executed with an invalid JerkFactor parameter.	The valid range is 0.01 to 1.2. Select a value in this range.
X412	Invalid function block Module parameter.	A function block instance has an invalid Module parameter.	Ensure the Module input parameter is linked to a configured PMM.
X413	Invalid function block File Reference parameter.	A function block instance has an invalid File Reference parameter.	File names must be between 1 and 32 characters in length.
X414	Invalid MC_TouchProbe TriggerInput parameter.	An MC_TouchProbe function block instance has an invalid TriggerInput parameter.	The Trigger Input must be configured as a Touch Probe input in the HWC.
X415	Invalid MC_DigitalCamSwitch Switches parameter	An MC_DigitalCamSwitch function block instance has an invalid Switches parameter. If this error occurred when there was already an active DCS function block running on the axis, a Normal Stop will occur.	Each component of the Switches input must be valid. Refer to the MC_DigitalCamSwitch function block discussion for details on the correct format for these inputs.
X416	Invalid MC_DigitalCamSwitch TrackOptions parameter	An MC_DigitalCamSwitch function block instance has an invalid TrackOptions parameter.	Each component of the TrackOptions input must be valid. Refer to the MC_DigitalCamSwitch function block discussion for details on the correct format for these inputs.
X417	Invalid MC_DigitalCamSwitch EnableMask parameter	An MC_DigitalCamSwitch function block instance has an invalid EnableMask parameter. This error occurred when there was already an active DCS function block running on the axis, resulting in a Normal Stop.	Only bits 0–3 of the EnableMask are valid to be set. Bits 4 to 32 must be set to 0.
X418	Invalid function block PositionSource parameter.	A function block instance has an invalid PositionSource parameter.	Valid PositionSource values are Actual Position and Commanded Position.
X419	Invalid ParameterConfig input to the MC_DL_Configure function block.	A function block instance has an invalid ParameterConfig parameter.	Each of the parameters must be a valid parameter for the axis or module.
X41A	Invalid DataLogConfig input to the MC_DL_Configure function block.	A function block instance has an invalid DataLogConfig parameter.	Each component of the DataLogConfig must be valid. Refer to the MC_DigitalCamSwitch function block discussion for details.
X41B	Invalid DataCaptureID parameter.	A function block instance has an invalid DataCaptureID parameter.	The DataCaptureID must be returned from an MC_DL_Configure that was executed on the same module.

Error No. (Hex)	Description	Cause	Recommended Correction
X41C	Invalid function block Velocity parameter	A function block was executed with an invalid Velocity parameter while the function block was active.	The minimum velocity allowed is 1/10 rpm or 1/600 rev/sec and the maximum velocity is specified by the MaxVelocityAppl parameter.
X41D	Invalid function block Acceleration parameter.	A function block was executed with an invalid Acceleration parameter.	The minimum acceleration allowed is 1/6000 rev/sec <sup>2</sup> . The maximum acceleration is specified by the MaxAccelerationAppl parameter.
X41E	Invalid function block Deceleration parameter.	A function block was executed with an invalid Deceleration parameter.	The minimum deceleration allowed is 1/6000 rev/sec <sup>2</sup> . The maximum deceleration is specified by the MaxDecelerationAppl parameter.
X41F	Invalid function block Jerk parameter.	A function block was executed with an invalid Jerk parameter.	If the JerkUnits is UU/sec <sup>3</sup> the minimum jerk allowed is 1/60000 rev/sec <sup>3</sup> and the maximum jerk is specified by the Max Jerk parameter. A value of 0 is allowed and is a special case indicating unlimited jerk. If the JerkUnits is Percent the allowed range is 0% to 100%.
X420	Invalid function block JerkUnits parameter.	A function block instance has an invalid JerkUnits parameter.	Valid JerkUnits are: UserUnitsperSecondCubed (UU/sec <sup>3</sup> ) and PercentJerkLimiting
x421	Invalid function block Direction parameter.	A function block instance has an invalid Direction parameter.	Valid Directions are Positive, Negative, Current Direction and Shortest Way. Current Direction is valid only if a previous move has been executed to establish the current direction.
X422	Invalid HomingMode parameter for MC_Home.	An MC_Home function block was executed with an invalid HomingMode parameter.	Valid Homing Modes for Axes 1–4 are DriveControlled, LimitSwitchRefPulse, RefPulse and RefPulseNeg. The Virtual Axis supports only RefPulse and RefPulseNeg. For DriveControlled make sure a valid Drive Homing Mode is selected in the Axis hardware configuration.
X423	Invalid MC_Home HomeOffset parameter.	A function block was executed with an invalid HomeOffset parameter.	The valid range for HomeOffset is –1000000 to 1000000. The value should not cause the position to move outside the valid position range.

Error No. (Hex)	Description	Cause	Recommended Correction
X424	Invalid MC_JogAxis MinJogDistance parameter.	A function block was executed with an invalid MinJogDistance parameter while the function block was active. The axis will be normal stopped.	MinJogDistance can be any value greater than or equal to zero that does not cause the position to move outside the valid position range.
X426	Invalid MC_CamIn MasterOffset parameter.	A function block was executed with an invalid MasterOffset parameter.	Valid MasterOffset range -4E10 to 4E10
x427	Invalid MC_CamIn SlaveOffset parameter.	A function block was executed with an invalid SlaveOffset parameter.	Valid SlaveOffset range -4E10 to 4E10
x428	Invalid MC_CamIn MasterScaling parameter.	A function block was executed with an invalid MasterScaling parameter.	Valid MasterScaling range -8E10 to 8E10, except may not be equal to zero.
X429	Invalid MC_CamIn SlaveScaling parameter.	A function block was executed with an invalid SlaveScaling parameter.	Valid SlaveScaling range -8E10 to 8E10.
X42A	Invalid MC_CamIn CamTableID parameter.	A function block was executed with an invalid CamTableID parameter.	Use a valid CamTableID output from an MC_CamTableSelect function block. Check MC_CamTableSelect parameters. Download Active profiles to the RX3i controller.
X42B	Invalid function block Distance parameter	A function block was executed with an invalid Distance parameter.	Either the distance specified is outside the possible distance for the type of axis or the value is an invalid floating-point number (i.e. NaN).
X42C	Invalid function block RatioNumerator parameter.	A function block was executed with an invalid RatioNumerator parameter.	The valid range for RatioNumerator is -32768 to 32767.
X42D	Invalid function block RatioDenominator parameter.	A function block was executed with an invalid RatioDenominator parameter.	The valid range for RatioDenominator is 1 to 32767.
X42E	Invalid SyncMode input to MC_GearInPos.	An MC_GearIn function block was executed with an invalid SyncMode parameter.	The valid values for SyncMode are BackupAllowed and NoBackupAllowed.
X42F	Invalid function block Length parameter.	A function block was executed with an invalid Length parameter.	The valid Length for MC_SyncStart and MC_DelayedStart is 1 to 8.
X430	Invalid DelayTimes input to MC_DelayedStart.	A function block was executed with an invalid DelayTimes parameter.	DelayTimes must be greater than or equal to 0.
X431	Invalid MC_Home FindHomeVelocity parameter.	An MC_Home function block was executed with an invalid FindHomeVelocity parameter.	The minimum FindHomeVelocity allowed is 0.1 RPM. The maximum FindHomeVelocity is specified by the MaxVelocityAppl parameter.

Error No. (Hex)	Description	Cause	Recommended Correction
X432	Invalid MC_Home FinalHomeVelocity parameter.	An MC_Home function block was executed with an invalid FinalHomeVelocity parameter.	The minimum FinalHomeVelocity allowed is 0.1 RPM. The maximum FinalHomeVelocity is specified by the MaxVelocityAppl parameter.
X433	Invalid MC_SetPosition Encoder parameter.	A function block was executed with an invalid Encoder parameter.	Valid Encoder values are Feedback Source, Motor Encoder or External Device. External Device is valid only if there is an external encoder. For Axis 5, only External Device is supported.
X434	Invalid MC_DigitalCamSwitch Outputs parameter	An MC_DigitalCamSwitch function block instance has an invalid Outputs parameter.  <i>Note: For MC_DigitalCamSwitch: the error occurred when there was already an active DCS function block running on the axis. This results in the error being a Normal Stop error.</i>	The array of OutputRefs must correspond to the EnableMask.
X435	Error occurred on an axis to be sync started.	An error has occurred on one of the axis specified in MC_SyncStart or MC_DelayedStart.	Check the axis error codes for the axes for the axes in the MC_SyncStart or MC_DelayedStart for the specific error that occurred causing the sync command to error.
X436	Timeout occurred waiting for a motion command.	The SyncTime for an MC_SyncStart or the StartTime for an MC_DelayedStart elapsed before each axis to be sync started had a valid command to start.	Either the motion commands to be sync started need to be executed sooner or a longer timeout needs to be set.
X437	While waiting to be sync started a new motion command replaced this command.	A new motion command was executed on this axis that replaced this command. The new command will now be started with the MC_SyncStart or MC_DelayedStart command.	If this was not expected check the timing of when the motion commands were executed.
X438	Error occurred on an axis to be sync started after it was too late to abort the start.	An error occurred on an axis in an MC_SyncStart or MC_DelayedStart command after the communication to start the command has already been sent.	Monitor the axes in an MC_SyncStart or MC_DelayedStart and if one axis fails take the necessary action on the other axes.

Error No. (Hex)	Description	Cause	Recommended Correction
X439	The sync start command was aborted by another command	Another command has aborted the MC_SyncStart or MC_DelayedStart command.	An MC_Power Enable = False, MC_Reset, MC_ModuleReset, and MC_Stop can abort an MC_SyncStart or MC_DelayedStart
x43A	Invalid MC_GearInPos MasterSyncPosition; outside range of master.	A function block was executed with an invalid MasterSyncPosition parameter.	For linear axis select a position inside the end of travel limits. For rotary axis select a position between the Low Limit and the Low Limit + Position Range.
X43B	Invalid MC_GearInPos SlaveSyncPosition; outside range of slave.	A function block was executed with an invalid SlaveSyncPosition parameter.	For linear axis select a position inside the end of travel limits. For rotary axis select a position between the Low Limit and the Low Limit + Position Range.
X43C	Invalid MC_CamIn StartMode parameter.	A function block instance has an invalid StartMode parameter.	Bit 0 indicates if Master is Relative or Absolute. Bit 1 indicates if Slave is Relative or Absolute. Bit 3 allows ramps to exceed the slave range in the profile. Setting any other bit is invalid.
X43D	Analog value out of range (+/- 10Vdc), clamped at limit.	MC_WriteAnalogOutput function block has attempted to set output greater than 10 or less than -10.	Use a value between 10 and -10.
X43E	Attempt to access FTB I/O failed. FTB failed or not configured.	MC_WriteAnalogOutput function block attempted when FTB has failed or is not present.	Check FTB status before issuing the MC_WriteAnalogOutput.
X43F	Invalid parameter number specified as input to function block.	An MFB has attempted to access an unsupported parameter.	Check parameter number.
X440	The application attempted to write a read-only parameter.	The MFB has attempted to write to a read-only parameter.	If possible, change the parameter value in Hardware Configuration.
X441	Parameter type does not match function block used.	An incorrect MFB is being used to access the specified parameter.	Check the parameter type and use the appropriate MFB.
X442	NaN assigned to a Real using a write parameter(s) function block.	MC_WriteParameter or MC_WriteParameters has attempted to set an invalid value.	Ensure that floating-point computations used as input to the function block do not yield a NaN result. (Note that NaN cannot be entered as a numeric literal, so it must result from computation in the application).
X443	Invalid digital output reference used to write digital output.	MC_WriteDigitalOutput has specified an invalid output reference.	Check the OUTPUT_REF value.
X444	Invalid analog output reference used to write analog output.	MC_WriteAnalogOutput has specified an invalid output reference.	Check the OUTPUT_REF value.



Error No. (Hex)	Description	Cause	Recommended Correction
X446	Parameter value is invalid.	An MFB has attempted to set a parameter to a value that is out of range.	Check valid ranges for the parameter. In some cases, an attempted scaling change (user units or counts) can make a scaled parameter invalid. To identify the dependent parameter that has failed, download the event queue and find the event corresponding to the failed scaling change function block. For details, refer to Section 9.5.1 Parameter Errors Caused by Changes in Axis Scaling.
X447	Invalid axis for requested operation.	The MFB has attempted to access a parameter that is not supported on the specified axis.	Check valid axes for the parameter. Note that many parameters are not available on virtual axes.
X448	Axis must be in Disabled, Standstill, or ErrorStop state.	An MFB has attempted to set a parameter on an axis that is not in a valid state to change that parameter. Some parameters cannot be set unless the axis is in Disabled, Standstill, or ErrorStop state.	Ensure axis is in a valid state before attempting parameter write.
X449	Invalid parameter when Positioning Mode is Rotary.	An MFB has attempted to set or enable End Of Travel (EOT) on an axis in Rotary positioning mode.	Change Low Limit and Range parameters or configure Axis Positioning Mode as Linear.
X44A	Invalid parameter when Positioning Mode is Linear.	An MFB has attempted to set or enable Low Limit or Range on an axis in Linear positioning mode.	Change EOT parameters or configure Axis Positioning Mode as Rotary.
X44B	Low Position Limit + Position Range exceed maximum position.	An MFB has changed Low Limit or Range parameter such that the sum is greater than the scaled maximum position.	Check that the Low position limit plus the specified position range does not exceed the maximum PMM position range of $(4e10 \times \text{UU:Counts ratio})$ .
X44C	Specified limit exceeds maximum position.	An MFB attempted to set a Low Limit greater than the scaled maximum position.	Check that the High Limit does not exceed the maximum PMM position range of $(4e10 \times \text{UU:Counts ratio})$ .
X44D	High limit must be greater than low limit.	An MFB attempted to set a High Limit lower than the Low Limit.	Check that the High Limit value is greater than the Low Limit Value
x44E	Over travel limit is disabled.	A disabled EOT has been changed.	This is an informational warning. The new value will be used if a subsequent parameter write enables the EOT.
X44F	Parameter change makes commanded position invalid.	A function block attempted to enable an EOT that would make the current Commanded Position invalid.	Change the EOT or move or set position to within the desired range.

Error No. (Hex)	Description	Cause	Recommended Correction
X450	Invalid Max Application Velocity.	MFB attempted to set a Max Application Velocity greater than the configured Max Velocity System.	Set a lower Max Application Velocity or raise Max Velocity System in HWC.
X451	Invalid Max Application Acceleration.	A function block attempted to set a Max Application Acceleration that is <= zero or greater than the configured Max Acceleration System	Set a non-zero positive Max Application Acceleration, decrease it, or raise Max Acceleration System in Hardware Configuration.
X452	Max Application Deceleration cannot exceed Max System Deceleration.	An MFB attempted to set a Max Application Deceleration greater than the configured Max Deceleration System.	Set a lower Max Application Deceleration or raise Max Deceleration System in HWC.
X456	Parameter change invalidates dependent configuration parameter.	Applying the specified scaling parameter would make one or more scaled parameters invalid.	An attempted scaling change (user units or counts) would make a scaled parameter invalid. To identify the dependent parameter that has failed, download the event queue and find the event corresponding to the failed scaling change function block. For details, refer to Section 9.5.1 Parameter Errors Caused by Changes in Axis Scaling
X457	No external encoder is configured for this axis.	Attempt to change External Encoder parameter for an axis that does not have an External Encoder configured.	Configure an external encoder if one is needed by the application and has not been configured for the axis. If the axis is a virtual axis, an external encoder must be configured if an MC_Home is issued. Check axis reference input.
x458	Invalid scaling. User Units must be less than or equal to Counts	The attempted parameter change would make User Units less than counts for the specified device.	Check scaling values.
x459	WriteAnalogOutput value is not the active analog source.	Attempted to write an analog output that is configured to output velocity or torque.	Set analog output source to Application Write.
x45A	Unable to write SCB parameter; write already in progress.	Another Servo Controller Board parameter write is in progress on the same axis.	Wait until the current write completes before beginning a new write.
x45B	Invalid PK1V Velocity Loop Gain 1 (Integral)	An MC_WriteDwordParameters was issued with an invalid value for PK1V.	The valid range for PK1V is 0 to 32767. Refer to Appendix Section C3.2, Advanced FSSB Servo Tuning for details on how to use this parameter.

Error No. (Hex)	Description	Cause	Recommended Correction
x45C	Invalid PK2V Velocity Loop Gain 2 (Proportional)	An MC_WriteDwordParameters was issued with an invalid value for PK2V.	The valid range for PK2V is -32767 to 0. Refer to Appendix Section C-3.2, Advanced FSSB Servo Tuning for details on how to use this parameter.
x45D	Invalid PK3V Velocity Loop Gain 3 (Integral Decay)	An MC_WriteDwordParameters was issued with an invalid value for PK3V.	The valid range for PK3V is 0 to 32767. Refer to Appendix Section C3.2, Advanced FSSB Servo Tuning for details on how to use this parameter.
x45E	Invalid TCMD Filter.	An MC_WriteDwordParameters was issued with an invalid value for FILTER TCMD.	The valid range for FILTER TCMD is 0 to 2810. Refer to Appendix Section C-3.2, Advanced FSSB Servo Tuning for details on how to use this parameter
x45F	Invalid LDINT (Load Inertia Ratio).	An MC_WriteDwordParameters was issued with an invalid value for LDINT (Load Inertia Ratio).	The valid range for LDINT is 0 to 4096. Refer to Appendix Section C for details on how to use this parameter.
x46C	Axis is in an invalid state to execute this function block.	MC_SyncStart and MC_DelayedStart cannot be executed while the axis is in ErrorStop, Stopping, Homing, or Disabled states.	Change the state to a valid state by clearing the error, completing the Stop or Home cycle, or by turning on power.
x46D	Invalid MC_CamIn MasterScaling parameter equal to zero.	A function block was issued with an invalid MasterScaling parameter equal to zero.	Valid MasterScaling range -8E10 to 8E10, except may not be equal to zero.
x46E	Invalid MC_CamIn Ramp Distance parameter.	A function block was issued with an invalid Ramp Distance parameter.	Valid Ramp Distance range is -4E10 to 4E10.
x46F	Axis position is invalid; motion command was suppressed.	Command was issued while axis position was invalid. For MC_DigitalCamSwitch: Error occurred when there was already an active DCS function block running on the axis. This results in the error being a Normal Stop error.	Ensure position is valid before issuing command.
x470	Invalid SyncTime or StartTime parameter.	MC_SyncStart was executed with an invalid SyncTime or MC_DelayedStart was executed with an invalid StartTime.	A value of zero is a special case that specifies a sync time of 5 minutes. Other values must be greater than or equal to 5 ms.
x471	Invalid Master Start Distance parameter	MC_GearInPos was executed with an invalid Master Start Distance.	Valid Master Start Distance range is -4E10 to 4E10

Error No. (Hex)	Description	Cause	Recommended Correction
x472	Invalid DCS Output Source mask specified	Value(s) written to parameter(s) 2114 and/or 2115 have specified masks with too many or invalid bits set.	No more than a total of 4 bits can be set in parameters 2114 and 2115. Valid bits in parameter 2114 are 0–1, parameter 2115 are 0–11.
x473	An attempt was made to change the output value using PN2109 and/or PN2110 of an output selected to be a DCS.	User has attempted to assign a value to a DCS Output using PN2109 or PN2110.	Modify the bit values for PN2109 and/or PN2110 so that DCS mask bits are not set.
x474	Invalid feedback moving deadband.	User has attempted to write an invalid value for PN1024.	Valid range is defined by configuration limits.
x475	Invalid command moving deadband.	User has attempted to write an invalid value for PN1025.	Valid range is defined by configuration limits.
x476	Invalid phase shift.	The value of the PhaseShift input to MC_Phasing was not within the valid range.	Correct the PhaseShift input. Valid phase shift range is -4E10 to 4E10.
x477	Selected parameter for MC_ReadParameter(s) is currently disabled.	Selected parameter is not active in the hardware configuration.	Change the hardware configuration to activate the parameter.
x478	MC_Home on axis with no configured marker input.	MC_Home issued on an axis that does not have an input configured as the marker for the axis.	In hardware configuration, select the marker input for the axis. Home cannot be performed without a marker.
x479	MC_Home using a home-to-switch (Limit Switch Ref Pulse) homing mode on an axis with no configured home switch input.	MC_Home issued with Limit Switch Ref Pulse as the homing mode, but no home switch input was configured for the axis.	In hardware configuration, select a home switch for the axis or use a different value for the HomingMode input on the MC_Home command.
x480	DCS aborted due to changed command position resolution.	Changed command position resolution while DCS command active.	Do not change command position resolution while DCS command is active.
x481	DCS aborted due to changed command position range.	Changed command position range while DCS command active.	Do not change command position range while DCS command active.
x482	DCS aborted due to changed command position low limit.	Changed command position low limit while DCS command active.	Do not change command position low limit while DCS command active.
x483	DCS aborted due to changed motor encoder user units or counts.	Changed motor encoder user units or counts while DCS command active.	Do not change motor encoder user units or counts while DCS command active.
x484	DCS aborted due to changed motor encoder position range.	Changed motor encoder position range while DCS command active.	Do not change motor encoder position range while DCS command active.

Error No. (Hex)	Description	Cause	Recommended Correction
x485	DCS aborted due to changed motor encoder position low limit.	Changed motor encoder position low limit while DCS command active.	Do not change motor encoder position low limit while DCS command active.
x486	DCS aborted due to changed external encoder user units or counts.	Changed external encoder user units or counts while DCS command active.	Do not change external encoder user units or counts while DCS command active.
x487	DCS aborted due to changed external encoder position range.	Changed external encoder position range while DCS command active.	Do not change external encoder position range while DCS command active.
x488	DCS aborted due to changed external encoder position low limit.	Changed external encoder position low limit while DCS command active.	Do not change external encoder position low limit while DCS command active.
x48C	Axis must be in Disabled state.	Attempted to change UU:Cts scaling via MC_WriteParameter(s) or MC_WriteDWORDParameter(s) when axis not in Disabled state.	Before attempting to change this parameter, remove power from the drive and transition the axis to the Disabled state by lowering the Enable input to MC_Power.
x48E	Invalid output reference configured as drive status for analog servo	User has attempted to write a value to a digital output that is controlling an analog servo.	Digital outputs that are being used as Analog Servo Drive Enable or Analog Servo Drive Reset cannot be written by application logic. Select a different digital output.
x48F	Invalid analog output reference configured for analog servo control	User has attempted to write a value to an analog output that is controlling an analog servo.	Analog outputs that are being used to drive analog servos cannot be written by application logic. Select a different analog output.
x490	Invalid torque command filter	User has attempted to write an invalid value for PN1322	This parameter is only valid for axes configured as AnalogTorqueMode. Valid values are 0 and 60 to 400.
x491	Invalid minimum velocity output	User has attempted to write an invalid value for PN1323	This parameter is only valid for axes configured as AnalogVelocityMode. The valid range is 0 to 1000.
X492	Invalid MC_Home Mode Parameter	Invalid Mode parameter detected on MC_Home function block	Check MC_Home function block for correct mode parameter
X493	Invalid MC_Home direction parameter	Invalid Direction parameter detected on MC_Home function block	Check MC_Home function block for correct Direction parameter
x800	CAM library is full.	There is no room on the motion module to store CAM profiles.	Remove specific profiles from the motion module by executing MC_CamTableDeselect or, in HWC, configure the module to use Automatic CAM Library Management mode.

Error No. (Hex)	Description	Cause	Recommended Correction
x801	CAM Table Select failed	The specified CAM file does not exist on the RX3i.	Verify that Active profiles were downloaded to the RX3i. Verify the MC_CamRef input to the MC_CamTableSelect function block is correct. The file name should match the name of the CAM profile you're trying to select.
x802	CAM profile build failed.	While building the CAM profile, a math error was encountered.	Verify the CAM profile is correct. If the CAM profile has been modified by MC_CamFileRead or MC_CamFileWrite operations, portions of it may have been written improperly. If verifying that the profile on the RX3i controller is what you expect and re-selecting it does not correct the problem, contact Technical Support.
x803	CAM profile not found on PMM.	The CAM profile specified by MC_CamTableSelect does not exist on the PMM.	Verify that it has been selected and check to see if it could have been removed by an MC_CamTableDeselect, RemoveAll or from automatic CAM library management.
x804	Cannot use MC_CamTableSelect to update a profile that is engaged.	MC_CamTableSelect failed because the specified profile is a different version of a profile that is already engaged and in use on the PMM.	Disengage the CAM profile by executing MC_CamOut, then re-execute MC_CamTableSelect to update the version of this profile on the module.
x809	CAM profile not built.	The specified CAM profile exists on the module but is not ready to be used on an MC_CamIn.	Reselect the CAM profile before executing this MC_CamIn.
x80A	Cannot use MC_CamTableDeselect on an engaged profile.	The specified CAM profile is engaged (in use by an active MC_CamIn).	If you want to keep the CAM engaged, then either do not execute MC_CamTableDeselect or ignore this error. To disengage the CAM and remove the profile, execute MC_CamOut (or abort MC_CamIn with another function block), then remove the profile with MC_CamTableDeselect.
x80B	Max number of CAM profiles reached.	The max number of CAM profiles has been reached (256 profiles have been selected and the module is in 'Manual CAM Library Management mode'.	Remove some profiles and reselect. Or reconfigure the module to use Automatic CAM Library Management mode.
x80C	CAM profile CRC error.	Internal software/hardware error.	Contact Technical Support.

Error No. (Hex)	Description	Cause	Recommended Correction
x80D	MC_CamTableSelect aborted.	Another MC_CamTableSelect for the same profile has aborted this one, which was in progress (Busy, not Done yet).	It is only necessary to do one MC_CamTableSelect per CAM per module. It isn't necessary to select the same profile on multiple slave axes, as long as the slaves are on the same module.
x80E	CAM profile version not supported.	Invalid combination of software/firmware versions.	Verify build numbers of CPU firmware, PMM firmware, and Programmer.
x810	MC_CamTableSelect aborted due to an RX3i controller state transition.	The RX3i controller went to stop mode while MC_CamTableSelect was busy. The operation was aborted.	The CAM profile was not successfully selected. It will have to be re-selected when you go back to run or I/O enabled mode.
x811	MC_CamTableDeselect aborted due to an RX3i controller state transition	The RX3i controller went to stop mode while MC_CamTableDeselect was busy. The operation was aborted.	None, this may be ignored. Realize that the CAM profile was not successfully deselected. It will have to be deselected when you go back to run or I/O enabled mode.
x812	The CAM profile does not fit on the master axis.	After applying the master scaling factor to the profile, the profile range doesn't match the master axis range.	Adjust the profile range, scaling factor, or master axis range. For more information on profile types and boundary conditions, refer to Section 7.3, CAM Operation Restrictions by Type and Mode
x813	The CAM profile does not fit on the slave axis.	After applying the slave scaling factor to the profile, the profile range doesn't match the slave axis range.	Adjust the profile range, scaling factor, or master axis range. For more information on profile types and boundary conditions, refer to Section 7.3, CAM Operation Restrictions by Type and Mode.
x814	The CAM profile is invalid.	The CAM profile is not formatted properly.	Try reverting to a previous version of the CAM profile if possible. Restore the PME project or re-inspect any CamFileRead/Writes. If these steps do not resolve the problem, contact Technical Support.
x815	Absolute Master, Periodic Profiles are not allowed on Linear Masters	An MC_CamIn was executed on a linear master with a periodic CAM profile using Absolute Master mode. This is not permitted.	You may try using Relative Master mode or using a series of non-periodic profiles. For more information on profile types and boundary conditions, refer to Section 7.3, CAM Operation Restrictions by Type and Mode

Error No. (Hex)	Description	Cause	Recommended Correction
x816	Absolute Slave, Periodic Profiles are not allowed on Linear Slaves.	An MC_CamIn was executed on a linear slave with a periodic circular cyclic CAM profile using Absolute Slave mode. This is not permitted.	You may try using Relative Slave mode or using a different profile type. For more information on profile types and boundary conditions, refer to Section 7.3, CAM Operation Restrictions by Type and Mode.
x818	CAM master data received was not monotonic.	The series of master positions in the CAM profile are not increasing.	View the CAM profile; verify there are not two identical master positions or that master positions never decrease.
x819	CAM profile is normalized, but master data does not match.	The profile is marked as being normalized, but the master's positions are not from 0 to 1.	View the CAM profile and verify the master positions are from 0 to 1. Either mark the profile as non-normalized or normalize the master/slave position pairs.
x81A	CAM master data does not match CAM profile's master low limit and range.	The master positions do not cover the profile range specified from the master low limit to the master profile low limit + master profile range.	View the CAM profile; verify the master positions are within the defined master profile range. Either adjust the master positions or the master profile range.
x81B	CAM profile is normalized, but slave data does not match.	The profile is marked as being normalized, but the slave's positions are not from 0 to 1.	Review the CAM profile and verify the slave positions are between 0 and 1. Either mark the profile as non-normalized, or normalize the master/slave position pairs.
x81C	CAM slave data does not match CAM profile's slave low limit and range.	The slave positions do not cover the profile range specified from the slave low limit to the slave profile low limit + slave profile range.	View the CAM profile; verify the slave positions are within the defined slave profile range. Adjust either the slave positions or the slave profile range.
x81D	MC_CamTableSelect aborted due to DLB stop	An MC_CamTableSelect from the DLB that was in progress was aborted due to the DLB stop.	This error can be ignored. The CAM ID this MC_CamTableSelect would have produced would no longer be valid after the DLB had been terminated.
x81E	CAM profile slave start and end positions are not equal.	With a periodic absolute linear slave, a non-cyclic profile must have the slave's first and last positions match.	Use a different profile type if the slave's start and end positions cannot match. Refer to Section 7.3, CAM Operation Restrictions by Type and Mode for more information on profile types and boundary conditions.
x81F	MC_CamTableSelect from application may not use CAM profile from DLB.	An MC_CamTableSelect from the application has used the MC_CamRef of a CAM profile stored with the DLB. This is not allowed.	Profiles stored with the DLB may only be selected from the DLB.
x881–x888	Internal error.	Internal error.	Contact Technical Support.
x889	No registrant for COMMREQ received.	An invalid COMMREQ was sent to the PMM.	Ensure the rack and slot specified in the COMMREQs address the desired module.



Error No. (Hex)	Description	Cause	Recommended Correction
x88A–x8A0	Internal error.	Internal error.	Contact Technical Support.
x8A1	Wait for axes to stop before clearing configuration or downloading new configuration.	Axis servo is still powered on when clear of configuration or download of configuration is attempted.	Wait until axis servo turns off before clearing configuration or downloading new hardware configuration. If axis is configured as Analog Velocity Mode or Analog Torque Mode, verify that Drive Input Status is configured for the type of Drive Ready Status provided by the servo drive.
xC00	Cannot trigger data logging before activating a configuration.	A data logging session with post-trigger or pre-trigger trigger modes has been initiated by setting the InputTrigger input of MC_DL_Activate high prior to activating the data logging session with the Enable Input of MC_DL_Activate.	Turn on the Enable input of MC_DL_Activate and then turn on the InputTrigger of MC_DL_Activate to start the data logger.
xC01	Cannot delete a currently active configuration.	An MC_DL_Delete request was issued for a data logging session that is still in progress.	Deactivate the data logging session by setting the Enable input of MC_DL_Activate low, or Allow the data logging session to complete before deleting the datalog configuration.
xC03	Cannot activate the configuration being requested.	An MC_DL_Activate has attempted to activate a data logging configuration while one is already in progress. Only one datalog configuration can be active.	Deactivate the active configuration.
xC04	Cannot retrieve an active datalog configuration or retrieval is already in progress	An MC_DL_Get attempted retrieval of the data log before deactivating the data logging session.	Deactivate the data logging session by setting the Enable input of MC_DL_Activate OFF and then retrieve the datalog with MC_DL_Get.
xC05	The maximum number of datalog configurations supported has been reached.	The maximum number of configurations allowed is 10.	Delete one or more configurations before executing MC_DL_Configure again.
xC06	The number of samples specified in the datalog configuration exceeds the maximum.	The number of samples specified for the MC_DL_Configure will create a datalog file that is greater than the maximum space of 2MB allocated for data logging.	Reduce the number of samples specified in the datalog configuration.
xC08	An invalid data capture ID was specified on the function block.	The data capture ID specified is not associated with a datalog configuration present on the module. Data capture IDs are created by the DataCaptureID output of the MC_DL_Configure function block.	Use a valid DataCaptureID. Ensure the data capture ID is not associated with a configuration that has been deleted by the MC_DL_Delete function block.

Error No. (Hex)	Description	Cause	Recommended Correction
xC09	Zero samples specified in the datalog configuration.	The MC_DL_Configure specifies zero samples, which cannot be logged based on the number of parameters being logged.	Change the number of samples in DataLogConfig to a non-zero number.
xC0A	Trigger mode and operating mode combination not valid.	The Data Logger operating mode is single, and the trigger mode is pre-trigger or combined trigger mode. Pre-trigger and combined trigger modes are not available in single mode operation.	Change the operating mode to circular mode to use pre-trigger and combined trigger modes.
xC0B	No configurations are present on the module to be deleted, activated or retrieved.	There are no Data Logger configurations present on the module that can be used for data logging operations.	Execute MC_DL_Configure to create a configuration and then execute the operation.
xC0D	No parameters specified in the datalog configuration.	The datalog configuration has no parameters specified.	Add the parameters to be logged in the datalog ParamConfig reference structure on the MC_DL_Config function block input.
xC0E	An invalid or unsupported parameter number was specified in the datalog configuration.	One or more parameters specified in the datalog configuration are invalid.	Check that all the parameters specified in the datalog configuration are in the parameter list and supported.
xC0F	An invalid or unsupported parameter type was specified in the datalog configuration.	Internal module error.	Contact Technical Support.
xC10	A file transfer failure occurred while sending the datalog file to the CPU	No user memory available. (This condition is more likely to occur with the CPE305 than with other CPU models.)	Upload Data Logs from the CPU or CPE and delete the logs from the CPU/CPE.  and/or  To reduce the log file size, reduce the following in the Datalog configuration:  a. the number of samples b. the sample rate, c. the number of parameters logged
xC11	The datalog file being requested is no longer available on the module	The datalog file has been over-written with a new file. The data specified by MC_DL_Get no longer exists on the module.	The datalog file associated with the data capture that is been requested no longer exists. Log the data using the desired DataCaptureID, and then execute MC_DL_Get to retrieve the data.
xC12	The datalog file being retrieved has no data.	An MC_DL_Get has been executed, but no data has been logged yet.	Log some data and then re-execute MC_DL_Get to retrieve the logged data.

Error No. (Hex)	Description	Cause	Recommended Correction
xC13	A new MC_DL_Activate function block instance is now in control.	More than one MC_DL_Activate is attempting to initiate a data logging session. It is recommended that one instance of MC_DL_Activate should have power flow and used in an application.	Only use one instance of MC_DL_Activate for data logging.
xC14	The data logger command buffer is full; the maximum number of commands is 10.	The application logic is sending many data logger function block commands to the module consecutively. This can happen in cases where a flood of data logger commands is sent consecutively to the PMM.	Check the application logic for conditions where a flood of commands is being sent to the PMM. Contact Emerson for further assistance, providing the contents of the fault table and event queue.
xC15	Datalog operation aborted due to an RX3i controller mode change.	An RX3i controller mode change caused the data logger function block to be aborted, examples of mode transitions include Stop Mode Transition, DLB termination, and I/O Disabled transitions.	None. An RX3i controller mode transition caused termination of the function block.
xC16	Number of parameters in the datalog configuration cannot be supported at configured sample rate.	Too many parameters specified by MC_DL_Configure for this sampling rate. At faster sampling rates of 250us and 500us, the full set of 48 parameters cannot be logged. At 250us, up to 24 parameters can be logged. At 500us, up to 30 parameters can be logged.	Reduce the number of parameters configured to be logged. The first 24 or 30 parameters will be logged for 250us and 500us respectively.
xC41	Backplane startup services failed.	Internal error.	Contact Technical Support.
xCC0	System Manager detected internal failure.	Internal error.	Contact Technical Support.
xCC1	Call to perform module reset operation failed.	An MC_ModuleReset function block issued by the application failed to reset the module.	Contact Technical Support.
xCC2	Attempt to reset the SCB failed.	An error occurred on the Servo Controller Board, which prevents resetting this device.	Examine the fault table and the event queue for errors indicating the nature of the problem with Servo Control Board.
xCC3	Attempt to reset the FTB failed.	An error occurred on the FTB, which prevents resetting this device.	Examine the fault table and the event queue for errors indicating the nature of the problem with the FTB.
xCC4	Module synchronization failed.	An error occurred attempting to synchronize this module with other motion modules in the system.	Contact Technical Support.
xCC5	Currently active reset command aborted.	An MC_ModuleReset function block issued by the application is aborted when a new MC_ModuleReset is issued.	Examine the application and assure that any MC_ModuleReset completes (gets a Done or Error output) before issuing another MC_ModuleReset.

Error No. (Hex)	Description	Cause	Recommended Correction
xCC7	System Manager received invalid command.	Internal error.	Contact Technical Support.
xCC8	System Manager entered error state.	A system error has been detected that prevents the module from performing motion.	Contact Technical Support.
xCC9	System Manager unable to configure the SCB.	An error occurred while attempting to reset the Servo Controller Board after the module received a new HWC file from the RX3i CPU.	Examine the fault table and the event queue for errors indicating the nature of the problem with the Servo Control Board.
xCCA	System Manager detected normal stop failure.	An error occurred during a normal error stop.	Examine the fault table and the event queue for errors that sent the axis into normal error stop.
xCCB	System Manager detected fast stop failure.	An error occurred during a fast stop.	Examine the fault table and the event queue for errors that sent the axis into fast error stop.
xCCC	The axis drive was never disabled.	The system detected that an axis drive was never disabled after the drive disable delay expired.	Examine the fault table and the event queue for errors indicating the nature of the problem.
xCCD	System Manager detected invalid commanded jerk.	The system detected that the commanded jerk is greater than the configured maximum jerk during a stop operation.	Examine the application for instances where jerk is specified, or where jerk may be modified, to insure it does not exceed the configured maximum. If no direct cause is found and the condition persists, contact Technical Support.
xCCE	SCB errors were cleared, but new errors are now present.	An MC_ModuleReset was requested, and upon completion, detected servo controller board errors that do not cause an axis to stop motion.	Examine the fault table and the event queue for errors that indicate a Servo Controller Board problem, but do not stop motion.
xCCF	Wait for axes to stop before clearing configuration status.	Clear of configuration status attempted when Axis in motion.	Stop axis before clearing configuration status.
xCD2	The axis was normal stopped to reset the module.	An MC_Reset was executed on another axis to clear an PM EtherCAT Servo Alarm error. Clearing that error required that this axis also be reset.	Disable MC_Power on all axes before executing MC_Reset to clear an PM EtherCAT Servo Alarm
xCD3	The module is being shut down because of a system error. The axis was fast stopped	Internal errors – Event Queue may have additional data concerning source.	Contact Technical Support.
xCD4	Module was unable to clear an error	An MC_ModuleReset was executed and was unable to reset the module due to an error condition that is still present.	Examine the fault table and event queue for fault conditions that are still present on the module.

Error No. (Hex)	Description	Cause	Recommended Correction
xD00–xD09	Internal hardware error	Internal hardware error.	Contact Technical Support.
xD0A	Module exceeded high temperature threshold.	The temperature of the module exceeded the high temperature threshold. (Temp $\geq$ 55°C).	Check temperature levels in operating environment.
xD0B	Module exceeded warm temperature threshold.	The temperature of the module exceeded the warm temperature threshold. (50C $\geq$ temp > 55C).	Check temperature levels in operating environment.
xD0C	Module temperature now OK.	The temperature of the module is now OK.	Informational message.
xD80–xD84	Internal System Software error.	Internal error.	Contact Technical Support.
xDC0	Power-up Event.	This event is generated every time the PMM is reset or powered-up.	None. This is an informational message.
xDC1	Invalid EventID No Stop Response.	Internal error.	Contact Technical Support.
xDC2	Invalid EventID Normal Stop Response.	Internal error.	Contact Technical Support.
xDC3	Invalid Event ID Fast Stop Response	Internal error.	Contact Technical Support.
xDC4	An error occurred while sending the Event Queue file to the CPU.	Internal error.	Contact Technical Support.
xDC5	Cannot process multiple MC_ReadEventQueue function blocks.	More than one MC_ReadEventQueue function block has been executed.	Only execute one MC_ReadEventQueue at a time.
xE00	Control loop execution time exceeded warning limit	Motion Module execution time reached warning limit	Informational message
xE01	Control loop execution time exceeded error limit	Motion Module execution time reached error limit	Contact Technical Support
xF40	MC_ModuleReset successfully performed.	User has performed an MC_ModuleReset.	Informational message.
xF41	MC_Reset successfully performed.	User has performed an MC_Reset.	Informational message.
xF42	PMM Module hardware error	Internal Error	Contact Technical Support
xF43	Module Synchronized with rack synch source	Module is synchronized with rack synch source.	Informational message. No action is necessary.
xF44	Module scanning for rack synch source	Module is scanning for rack synch source.	Informational message. No action is necessary.

Error No. (Hex)	Description	Cause	Recommended Correction
xF45	Module is Rack Synch Source	Module is serving as rack synch source.	Informational message. No action is necessary.
xF46	Multiple rack synch source modules detected.	Multiple rack synch sources detected.	Contact Technical Support.
xF47	Internal Event logged to Event Queue.	Internal Event – informational only – event logged to event queue	Informational message. No action is necessary.
xF48	An unexpected T1 interrupt has occurred.	An unexpected T1 interrupt has occurred.	Informational message. No action is necessary.

## 9.2 CPU Error Codes

This section lists motion-related errors that are reported in the Controller Fault Table.

Error ID (Hex)	Description	Cause	Recommended Correction
xF81	Invalid parameter data in PMM.	Parameter information in PMM was in an invalid state when an MC_Read-type function or function block attempted to read it.	If problem does not correct itself by next execution of function, check for an error using MC_ReadAxisError or by examining event queue log.
xF82	General file handling error.	During execution of an MC_CamFileRead or MC_CamFileWrite, a problem was detected with the associated CAM profile file. This most likely indicates a problem with the format of the data in the CAM profile file.	Store a known good copy of the CAM profile file to the RX3i controller and try the operation again.
xF83	Another CAM file function block instance is in progress.	The application has tried to start the execution of an MC_CamFileRead or MC_CamFileWrite function block while the execution of another instance of an MC_CamFileRead or MC_CamFileWrite was still in progress. Only one MC_CamFileRead or MC_CamFileWrite execution can be active at any given time.	Change application logic so it does not attempt to start an MC_CamFileRead or MC_CamFileWrite function block while one is already in progress.
xF85	File not found	The execution of an MC_CamFileRead or MC_CamFileWrite was started but had to be aborted because the CAM profile file was not found.	Ensure the name of the CAM profile file is correct and the CAM profile has been stored to the RX3i controller.
xF86	File already opened for writing	The application attempted to execute an MC_CamFileRead or MC_CamFileWrite, but the specified CAM profile file was already being written by another operation.	Ensure the specified CAM profile is not being written by another MC_CamFileWrite and is not being stored to the RX3i controller when attempting to execute an MC_CamFileRead or MC_CamFileWrite.
xF87	File type unsupported	The application attempted to execute an MC_CamFileRead or MC_CamFileWrite, but the file specified by the MC_CAM_REF input was not a CAM profile file.	Correct the MC_CAM_REF input so that it refers to a CAM profile file.

Error ID (Hex)	Description	Cause	Recommended Correction
xF88	Data size not sufficient	MC_CamFileWrite: The data being read from the Data parameter into the CAM profile file exceeded the bounds of the Data parameter or exceeded the bounds of its reference memory. MC_CamFileRead: Insufficient memory in the Data parameter.	Increase the size of the Data parameter or of the reference memory.
xF89	File already opened for reading	The application attempted to execute an MC_CamFileWrite on a specified CAM profile, while the CAM profile file was being read by another operation.	Ensure the specified CAM profile is not concurrently being read by an MC_CamFileRead or an MC_CamTableSelect and is not concurrently being loaded from the RX3i controller to a Logic Developer target.
xF8A	User specified a CAM description that is not supported.	The Description input to MC_CamFileRead and MC_CamFileWrite must specify that only one element of the CAM profile file is being read or written, and that that element is the entire file.	Change the data in the Description parameter to indicate that the whole CAM profile is being read or written.
xF8B	CAM profile in reference memory would exceed 128KB limit of CAM file.	For an MC_CamFileWrite, the amount of data being written to the CAM profile file from data in reference memory would exceed the CAM profile file's size limit of 128K bytes.	Reduce the amount of data being written to the CAM profile file.
<p><b>Note:</b> For Error Codes xF8C to xF8F, the problem could be caused by extra or missing CAM profile data unintentionally written or not written to reference memory. This would cause the data coming after it in reference memory to be misinterpreted. It could be difficult to find the source of a problem like this. In this case, the problem with the contents of reference memory occurred at a lower reference address than the specific problem reported.</p>			<p>Perform the corrective action(s) listed for the error. If the problem persists, use MC_CamFileRead to obtain a known good CAM profile. For details on the file structure, refer to Section 7.7 CSV CAM File Format</p>
xF8C	Number of sectors in header doesn't match actual number of sectors in profile or is out of range	For an MC_CamFileWrite, the number of sectors being written from reference memory to the CAM profile file is outside the valid range.	Correct the data in reference memory so that the number of sectors is within range. The current valid range is 1 to 100.



Error ID (Hex)	Description	Cause	Recommended Correction
xF8D	Number of points in header doesn't match actual number of points in profile or is out of range	For an MC_CamFileWrite, the number of point pairs being written is outside the valid range allowed. (The current valid range is 2 to 4096 point pairs, unless there is only one linear sector, in which case the valid range is 2 to 5000). Or, the sum of the point pairs from each sector does not match the total number of point pairs specified in the CAM profile file header.	Correct the CAM profile data in memory so that the number of point pairs is valid.
xF8E	Expected to find a NaN sector header indicator in reference memory to signify start of new sector, reference memory format invalid.	For an MC_CamFileWrite, an expected NaN sector header to signify the start of a new sector was missing in reference memory.	Revise the contents of reference memory to include the NaN sector header.
xF8F	Unsupported curve fit type found while parsing sectors.	For an MC_CamFileWrite, an unsupported curve-fit type was found in the CAM file data being read from reference memory. The supported curve-fit types are linear (1), quadratic spline (2), cubic spline (3), and quintic (5). Quartic (4) is not supported.	Correct the CAM profile data in memory so that only valid curve-fit types are supported.
xF91	MC_ReadParam(s) Error - Undefined parameter within valid range	An undefined parameter number was passed into an MC_ReadParameter, MC_ReadBoolParameter, MC_ReadParameters, MC_ReadBoolParameters, or MC_ReadDwordParameters function. The parameter number was within the valid range of parameter numbers for the function, but was not a defined parameter.	Ensure all parameter numbers passed into the function are defined.

Error ID (Hex)	Description	Cause	Recommended Correction
xF92	MC_ReadParameter(s) Error - Parameter out of range	A parameter number was passed into an MC_ReadParameter(s), MC_ReadBoolParameter(s), or MC_ReadDwordParameters function that was outside the range of valid parameter numbers for the function. Note that there are distinct ranges of numbers used for LREAL parameters, BOOL parameters, and DWORD parameters. For example, a parameter number in the LREAL parameter range may have been passed to an MC_ReadBoolParameter.	Ensure all parameter numbers passed into a function are in the correct range of parameter numbers for that function.
xF93	MC_ReadParameter(s) Error - Mix of Axis and Module IO parameters in request	For MC_ReadParameters, MC_ReadBoolParameters, or MC_ReadDwordParameters, some of the parameter numbers passed into the function specified axis parameters while others specified module parameters. For these functions, all parameter numbers must specify axis parameters or all must specify module parameters.	Use only axis parameter numbers or only module parameter numbers in a given call of these functions.
xF94	MC_ReadParameter(s) Error - Mix of data types in request	For MC_ReadParameters, MC_ReadBoolParameters, or MC_ReadDwordParameters, one or more of the parameter numbers passed into the function represents parameters that of a different size than that expected by the function (LREAL, BOOL, or DWORD).	Ensure all parameter numbers represent the correct data type.
xF95	General MC_ReadParameter(s) error	An error was detected during execution of an MC_Read function or an MC_ReadStatus function block. This error indicates a condition other than those indicated by errors 0x0F81 or xF91 – xF9.	Check parameter numbers. If it is correct, consult the MC_ReadAxisError output and Event Queue logs. If error cannot be corrected, contact Technical Support.

Error ID (Hex)	Description	Cause	Recommended Correction
xF96	Read IO Error - Undefined IO number within valid range.	For MC_ReadDigitalInput, MC_ReadDigitalOutput, MC_ReadAnalogInput, MC_ReadAnalogOutput, this error indicates that the input number or output number, although within the valid range of input or output numbers, is not a defined input or output.	Correct the input or output number.
xF97	Read IO Error - IO number out of range.	For MC_ReadDigitalInput, MC_ReadDigitalOutput, MC_ReadAnalogInput, MC_ReadAnalogOutput, this error indicates that the input number or output number was not within the valid range of input or output numbers.	Correct the input or output number so that it is within the valid range.
xF9A	Read IO Error - Analog IO number given when Digital IO number expected.	For MC_ReadDigitalInput or MC_ReadDigitalOutput, this error indicates that an analog input or output number was given rather than a digital input or output number.	Replace the incorrect analog input or output number with the desired digital input or output number.
xF9B	Read IO Error - Digital IO number given when Analog IO number expected.	For MC_ReadAnalogInput or MC_ReadAnalogOutput, this error indicates that a digital input or output number was given rather than an analog input or output number.	Replace the incorrect digital input or output number with the desired analog input or output number.
xF9C	General MC_Read IO error.	For MC_ReadDigitalInput, MC_ReadDigitalOutput, MC_ReadAnalogInput or MC_ReadAnalogOutput, this error indicates a condition was detected other than those covered by error codes xF96 – xF9B.	Ensure the input or output number is correct for the function. If it is correct, consult the MC_ReadAxisError output and Event Queue logs. If error cannot be corrected, contact Technical Support.
xF9D	MC_CamTableSelect - CAM file not found.	When attempting to execute an MC_CamTableSelect, the CAM profile file specified by the MC_CAM_REF input argument was not found in the RX3i controller's memory.	Store the specified CAM profile file to the RX3i controller.

Error ID (Hex)	Description	Cause	Recommended Correction
xFA2	Uninitialized Axis, Module, or CAM Table variable.	An AXIS_REF, MODULE_REF, INPUT_REF, OUTPUT_REF, or MC_CAM_REF variable has never been given a value.	For AXIS_REF, MODULE_REF, INPUT_REF, and OUTPUT_REF variables that are not generated as part of the PMM hardware configuration, ensure these variables are initialized in the application logic using MOVE_DATA before they are used on a motion function or function block. For MC_CAM_REF variables that are not associated with a new CAM profile, ensure they are initialized in the application logic using MOVE_DATA before they are used on a MFB. When downloading the project to the controller, select Download Active Profiles.
xFA4	Motion module not available.	The application attempted to execute a motion function or function block on a motion module that has not been configured.	Configure the motion axis and/or motion module or change the AXIS_REF/MODULE_REF input to the function block to one that references a configured axis or module.
xFA5	Motion Module lost.	Communication with the Motion Module was lost.	Ensure the PMM is in the rack and is properly seated. Observe LEDs on the PMM. If all appears OK but this error is still seen, (at your discretion) remove the PMM from the rack and reinsert it.

## 9.3 Interpreting Drive Faults and Warnings

The PMM returns vendor-specific drive faults and warnings, hereafter referred to as drive diagnostics, to the I/O fault table. Vendor-specific drive diagnostics are not returned at the output of a function block. In the I/O fault table, the drive diagnostics are denoted by the Error ID 0xDE.

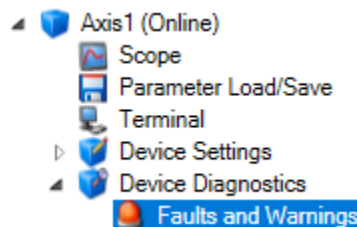
Diagnostic data denoted by Error ID 0xDE are reported in order of most recent to least recent. Thus, the most recent diagnostic data appears above less recent diagnostics in the I/O fault table, when more than one diagnostic is generated for an event.

The first diagnostic event is reported to the I/O Fault Table. Additional events will not be reported until an MC\_Reset is issued. If desired, all active faults and warnings, along with a fault history, can be obtained via Workbench, by expanding the “Device Diagnostics” tab, then selecting “Faults and Warnings”. The most severe active diagnostic is shown on the drive display – codes preceded by ‘F’ are faults, while codes preceded by ‘n’ are warnings. For additional information, consult GFK-3168, *PACMotion PSD Installation and User Manual*.

Note: the drive itself will respond appropriately to all faults.

---

**Figure 166: Access Diagnostic Information from Workbench the Terminal**



---

### 9.3.1 Drive Faults

The PMM returns vendor-specific drive faults in the form of 16-bit hex numbers. The hexadecimal value can be converted to a decimal number that corresponds to the error table contained in GFK-3168, *PACMotion PSD Installation and User Manual*. The fault code can be found in bytes 6-7 of the Fault Extra Data, corresponding to the entry for vendor-specific drive diagnostic information. A generic EtherCAT drive fault, alerting that an EtherCAT axis is in error, always precedes a vendor-specific drive fault. Drive faults are always logged as error-level events, which must be enabled on the PMM Hardware Configuration settings tab to be logged in the I/O fault table.



### 9.3.2 Drive Warnings

The PMM returns vendor-specific drive warnings in the form of 16-bit hex numbers. The hexadecimal value can be converted to a decimal number that corresponds to the error table contained in Section 7.3 of the *PACMotion Installation and User Manual*. The warning code can be found in bytes 4-7 of the Fault Extra Data, corresponding to the entry for vendor-specific drive diagnostic information. Drive warnings are logged as warning-level events, which must be enabled on the PMM Hardware Configuration settings tab to be logged in the I/O fault table.

**Figure 169: Vendor-Specific Drive Warning Example**

0.5	n/a	%I 00225	Motion Module fault	Axis warning or error	01-07- 2020 10:26:33		
L	I/O Bus	Bus Address	Point Address	Group	Action	Category	Fault Type
	n/a	n/a	n/a	10	2:Diagnostic	32	3
	Fault Extra Data	00de63100000025a0000000000000000000000000000					
	Fault Description						

The 32-bit warning code can be found in the Event Data field of the Event Queue Log. The lower 16-bits of the associated informational event contain the lower 16-bits of the warning code.

**Figure 170: Vendor-Specific Drive Warning, View from PMM Event Queue**

Axis 1	Warning	0x00DE	0x6310	0x0000025A	No Stop	Vendor-specific drive error. See vendor datasheet for description.
Axis 1	Warning	0x00DE	0x6310	0x0000025A	No Stop	Vendor-specific drive error. See vendor datasheet for description.

## 9.4 PMM Event Queue

The event queue contains the last 100 PMM events logged on the module. The queue is a FIFO buffer that contains an ordered list indicating the event sequence that has occurred on the module. In addition to errors and warnings, an informational event is queued at power-up and whenever the hardware configuration changes.

Over a power cycle, the most recent 24 events are preserved in non-volatile memory.

The MC\_ReadEventQueue function block is used to copy the current PMM event queue to a file in the RX3i controller. For details on accessing the event queue, refer to the discussion in Section 6.39, MC\_ReadEventQueue

The ten most recent events are recorded in parameters 2500–2509, which can be accessed by the MC\_ReadDWordParameters function block. For details, refer to Section 9.5, Accessing the Ten Most Recent Events.

### 9.4.1 Event Queue Details

Relative Time	Time stamp of event. The time stamp is a counter that starts running at power-up. The resolution of the timer/counter is 50µsec. Thus, a value of 25 for relative time is $25 \times 50\mu\text{sec} = 1250\mu\text{sec}$ .
Event Location	Module level or axis on which the event occurred.
Severity	Informational: Not a fault. Event is recorded only to provide information. Warning: Indicates a fault that is not severe enough to require motion stop. Error: Indicates a fault that is severe enough to require motion stop. A Stored event is an event that was stored from a previous power cycle.
Event ID	Event Error ID. For failed motion function blocks, Event ID is the same as the function block ErrorID output.
Event Information	Internal value used by Tech Support for additional diagnostics.
Event Data	If the Event ID is 0x0446 or 0x0456 this data may be used to determine the parameter that caused the failure. For details, refer to Section 9.5.1. Otherwise, this is an internal value used by Tech Support for additional diagnostics.
Response Method	No Stop: An event occurred that did not result in stopping the axis. Normal Stop: An event occurred that has caused a particular axis on the module to stop at a controlled deceleration rate. Fast Stop: An event occurred that that has caused a particular axis on the module to stop at a non-controlled deceleration rate.
Description	Text description of the event.



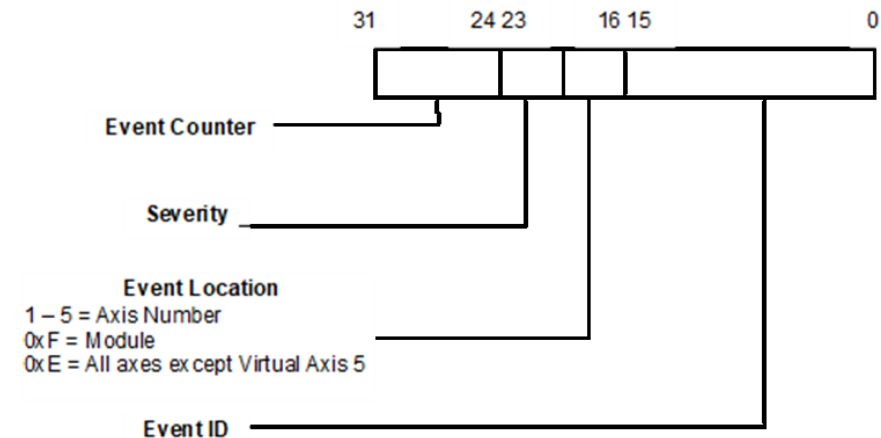
## 9.5 Accessing the Ten Most Recent Events

Parameters 2500–2509 contain the ten most recent axis and module events in chronological order, with the most recent event in parameter 2500. The parameters include events preserved over a power cycle in non-volatile memory. These parameters are accessed using the MC\_ReadDWordParameters function block.

The parameter data format is shown below.

### Recent Event Parameter Data

Figure 171: Bit Assignments for Recent Event Parameter Data



## Sample Event Queue Log

Figure 172: Sample Event Queue Log

EVENTS\_EVENTQUEUE.ELOG

SFRAM Event 
 Informational Event 
 Warning Event 
 Error Event

Number	Relative Time	Event Location	Severity	Event ID	Event Information	Event Data	Response Method	Description
0	0000000000	Module	Informational	0x0F45	0x2D0F	0x001F0004	No Stop	Assume rack synch mastership
1	0000000000	Axis 1	Informational	0x0421	0x3300	0x13220003	No Stop	Invalid function block Direction parameter
2	0000000000	Axis 1	Informational	0x032D	0x3400	0x06750000	No Stop	Axis has reached half way to the end of the range during an MC_MoveVelocity
3	0000000000	Axis 1	Error	0x532B	0x3420	0x069B0000	Normal Stop	MC_MoveVelocity has reached the limit of the axis and has stopped
4	0000000000	Axis 1	Informational	0x032A	0x3400	0x03A40003	No Stop	Axis power forced off due to an ErrorStop
5	0000000000	Module	Informational	0x0F40	0x3A0F	0x08930000	No Stop	MC_Module_Reset successfully performed
⋮								
74	0089115077	Axis 1	Informational	0x0306	0x3300	0x128B0000	No Stop	Function block attempting motion when axis position not valid
75	0089176776	Axis 2	Warning	0x0318	0x3711	0x0144007F	No Stop	Jerk constraint could not be maintained due to other constraints
76	0091434507	Module	Informational	0x0C15	0x210F	0x02B30016	No Stop	Datalog operation aborted due to PLC mode change
77	0091434507	Axis 1	Informational	0x530F	0x3600	0x0A5F0000	Normal Stop	PLC mode change aborted function block
78	0091434507	Axis 2	Informational	0x530F	0x3601	0x0A5F0000	Normal Stop	PLC mode change aborted function block
79	0091434507	Axis 3	Informational	0x030F	0x3602	0x0A590000	No Stop	PLC mode change aborted function block
80	0091434507	Axis 4	Informational	0x030F	0x3603	0x0A590000	No Stop	PLC mode change aborted function block
81	0091434507	Axis 5	Informational	0x030F	0x3604	0x0A590000	No Stop	PLC mode change aborted function block
82	0091434507	Axis 1	Informational	0x530F	0x3600	0x0A5F0000	Normal Stop	PLC mode change aborted function block
83	0091434507	Axis 2	Informational	0x530F	0x3601	0x0A5F0000	Normal Stop	PLC mode change aborted function block
84	0091434507	Axis 3	Informational	0x030F	0x3602	0x0A590000	No Stop	PLC mode change aborted function block
85	0091434507	Axis 4	Informational	0x030F	0x3603	0x0A590000	No Stop	PLC mode change aborted function block
86	0091434507	Axis 5	Informational	0x030F	0x3604	0x0A590000	No Stop	PLC mode change aborted function block
87	0092438906	Module	Informational	0x0F45	0x2D0F	0x001F0004	No Stop	Assume rack synch mastership
88	0092934814	Axis 1	Informational	0x0421	0x3300	0x13220003	No Stop	Invalid function block Direction parameter
89	0094572096	Axis 1	Informational	0x032D	0x3400	0x06750000	No Stop	Axis has reached half way to the end of the range during an MC_MoveVelocity
90	0094721956	Axis 1	Error	0x532B	0x3420	0x069B0000	Normal Stop	MC_MoveVelocity has reached the limit of the axis and has stopped
91	0094723976	Axis 1	Informational	0x032A	0x3400	0x03A40003	No Stop	Axis power forced off due to an ErrorStop

## 9.5.1 Parameter Errors Caused by Changes in Axis Scaling

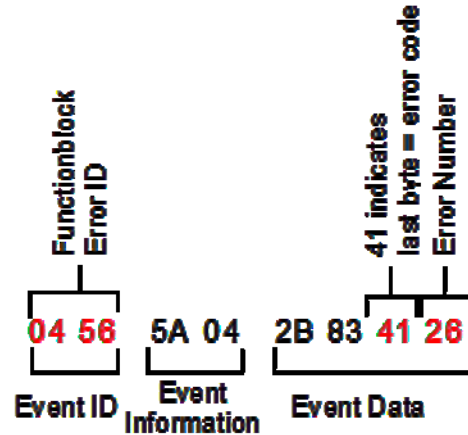
Using an MC\_WriteParameter(s) function block to change axis scaling (User Units, Counts, or Command Position Resolution) may invalidate configuration parameters that are scaled, causing the Motion Function Block to fail and the scaling change to be rejected.

To identify the dependent parameter that has failed, download the event queue and find the event corresponding to the failed scaling change function block. In some cases, the Event Data field may identify the error. If the third byte is 41, the least significant byte indicates the Error Number in the Scaling-Dependent Parameter Errors table below. It may be necessary to change the failed parameter before changing the scaling.

### Example:

The function block returns an error code 0x456 (Parameter change invalidates dependent configuration parameter.). The Event ID, Event Information and Event Data corresponding to the failed write parameter MFB are shown in the following example. The Event ID corresponds to the function block Error ID (0456). Since the third byte of the Event Data is 41, the least significant byte, 26, is the Error Number. Thus, Max Jerk is the parameter that was invalidated.

Figure 173: Error Code Example and Related Fields



## Scaling-Dependent Parameter Errors

Error Number	Failed Dependent Parameter	Likely Cause	Scaling	Scaling Parameter Numbers <sup>14</sup>
0x0B	Motor Encoder Position Range	Motor Encoder Position Range < Minimum range (0.1 rev)	Motor Encoder	1000, 1001
0x0C	Motor Encoder Position Range	Motor Encoder Position Range + Motor Encoder Low Position > Maximum Position	Motor Encoder	1000, 1001
0x0D	Motor Encoder Low Position Limit	Motor Encoder Low Position < Minimum range (0.1 rev)	Motor Encoder	1000, 1001
0x0E	Motor Encoder Low Position Limit	Motor Encoder Position Range + Motor Encoder Low Position > Maximum Position	Motor Encoder	1000, 1001
0x14	External Device Position Range	External Device Position Range < Minimum range (0.1 rev)	External Device	1004,1005
0x15	External Device Position Range	External Device Position Range + External Device Low Position > Maximum Position	External Device	1004,1005
0x16	External Device Low Position Limit	External Device Low Position < Minimum range (0.1 rev)	External Device	1004,1005
0x17	External Device Low Position Limit	External Device Position Range + External Device Low Position > Maximum Position	External Device	1004,1005
0x1A	High Software EOT Limit, Low Software EOT Limit	Attempted to change Software EOT Limit on axis configured for Rotary Axis Positioning Mode	Motor Encoder External Device Command Position Resolution	1000, 1001, 1004, 1005
0x1B	High Software EOT Limit	Attempted to set High Software EOT > Maximum Position	Motor Encoder External Device Command Position Resolution	1000, 1001, 1004, 1005
0x1C	Low Software EOT Limit	Attempted to set Low Software EOT > -Maximum Position	Motor Encoder External Device Command Position Resolution	1000, 1001, 1004, 1005
0x1D	High Software EOT Limit, Low Software EOT Limit	Attempted change would make High Software EOT <= Low Software EOT	Motor Encoder External Device Command Position Resolution	1000, 1001, 1004, 1005
0x1E	Command Position Range	Command Position Range < Minimum range (0.1 rev)	Command Position Resolution	1000
0x1F	Command Position Range	Command Position Range + Motor Encoder Low Position > Maximum Position	Command Position Resolution	1000
0x20	Command Low Position Limit	Command Low Position < Minimum range (0.1 rev)	Command Position Resolution	1000

<sup>14</sup> Refer to Scaling Parameters section below.

Error Number	Failed Dependent Parameter	Likely Cause	Scaling	Scaling Parameter Numbers <sup>14</sup>
0x21	Command Low Position Limit	Command Position Range + Motor Encoder Low Position > Maximum Position	Command Position Resolution	1000
0x26	Max Jerk	Max Jerk exceeds high or low limit	Motor Encoder External Device Command Position Resolution	1000, 1001, 1004, 1005
0x30	Error Stop Deceleration	Error Stop Deceleration exceeds high or low limit	Motor Encoder External Device Command Position Resolution	1000, 1001, 1004, 1005
0x31	Error Stop Jerk	Error Stop Jerk exceeds high or low limit	Motor Encoder External Device Command Position Resolution	1000, 1001, 1004, 1005
0x33	Feedback Moving Deadband	Feedback Moving Deadband exceeds high or low limit	Motor Encoder External Device Command Position Resolution	1000, 1001, 1004, 1005
0x34	Command Moving Deadband	Command Moving Deadband exceeds high or low limit	Motor Encoder External Device	1000, 1001, 1004, 1005

### Scaling parameters

- 1000 MotorEncoderUserUnits (Axes 1–4)  
CommandPositionResolution (Axis 5)
- 1001 MotorEncoderCounts
- 1004 ExternalDeviceUserUnits
- 1005 ExternalDeviceCounts

For parameter details, refer to Axis Parameter Number Index in Section 8.1.1.

## 9.6 Diagnostic Logic Blocks

A Diagnostic Logic Block (DLB) is a block of Ladder Diagram logic that can be downloaded to the controller for independent execution. These blocks are useful tools for interacting with an application that is running in the PACSystems controller.

DLBs are intended to accomplish a specific task that is temporary in nature, such as diagnosing the source of a problem or testing tuning parameters. When you have finished using a DLB, it should be removed from the host controller. At this point the application logic and its variable allocation return to what it was before the DLB was downloaded.

Note that, although the DLB is removed from the controller, changes the DLB made to the system are not removed. For example, if a DLB changes a hardware parameter using an MC\_WriteParameter function block, the parameter does not return to its previous value when the DLB is removed. Similarly, if an MC\_Reset is used to clear errors, the errors remain cleared.

### **⚠ CAUTION**

Do not use a DLB as a permanent part of a production application, because a DLB is stopped and deleted from RX3i memory if Logic Developer loses its Programmer-mode connection with the RX3i. This could happen if the programmer's communications cable is disconnected or if a second programmer connects serially to the same RX3i and establishes a Programmer-mode session.

When a DLB is downloaded, you are given the option of storing initial values or clearing memory for local variables. For a PACMotion application, it is recommended that you choose to store initial values instead of clearing memory for local symbolic variables.

For details on using DLBs, refer to the PACSystems RX3i and RSTi-EP CPU Programmer's Reference Manual , GFK-2950.

### 9.6.1 Using DLBs with PACMotion CAM Profiles

A DLB can have a maximum of three associated CAM profiles. The DLB CAM profile files do not count against the application's limit on the number of CAM profile files.

When copied, associated CAM profiles are copied with the block, so that the CAM profile will still be available. If a DLB containing CAM profiles is pasted or dragged into normal logic, the CAM profiles will be added under the CAM Profile Library node in the Navigator, which is found under the PACMotion node.

Each CAM profile associated with a DLB has its own menu and properties, which is the same as if the CAM profile was under the CAM Profile Library node. If a new CAM profile is added to a DLB that is currently on the controller, the entire DLB will require downloading in order to put the new profile on the controller.

When a DLB is downloaded, you are given the option of storing initial values or clearing memory for local variables. If another DLB is already stored on the controller it will be removed before the selected DLB is downloaded. For a PACMotion application, it is recommended that you choose to store download initial values instead of clearing memory for local variables.

# Appendix A: Touch Probe and Digital CAM Switch Accuracy Calculations

Topics covered:

Touch Probe Accuracy

External Quadrature Encoder



## A-1 Touch Probe Accuracy

The Touch Probe function utilizes two different position source types to capture position. The first is an external quadrature encoder while the second is the motor mounted encoder. Each of these devices is handled by the function block to maximize the touch probe accuracy. The specifics are discussed in the sections below.

For details on the Touch Probe function blocks, refer to Section 6.49, MC\_TouchProbe.

### A-1.1 External Quadrature Encoder

When using an external quadrature encoder, a touch probe event causes the quadrature counter value to be immediately latched into a holding register. This means that the position capture inaccuracies are based primarily on the input filtering, which varies from 100ns to 0.5ms depending on the input type. Because the data is captured via hardware, the number of counts that can occur during the hardware input filter delay determines the accuracy in encoder counts.

### A-1.2 Motor Encoder

The PMM345/PSD411 combination supports several different motor encoder types. This includes serial encoders where the position information is sampled. To yield highly accurate touch probes when paired with a serial encoder, the PMM345/PSD411 utilizes the drive's Position Capture function. The Position Capture function utilizes dedicated hardware and interpolation to enhance the accuracy of the sampled serial encoder. Details on the accuracy are contained in the PSD manuals (see section 1.2.2).

#### **WARNING**

Note that user wiring and the type of device used for the touch probe input may also cause inaccuracies in the touch probe value.

---

## A-2 Digital CAM Switch Accuracy

- The Digital CAM Switch (DCS) function controls digital outputs based on Actual or Commanded position of an axis. The DCS function has the following timing characteristics:
- DCS input positions are read every 1.0ms by the PMM firmware and used to control hardware timers that turn the DCS outputs on or off.
- The positions available to the DCS firmware depend upon the feedback device. For External encoders the position is 125 us old. For motor based encoders the position is 400 us old.
- The DCS hardware output timers have a range of 0 to 1.0ms. A timer value of zero causes a DCS output to change state 500 us plus capture delay. after the associated input position was captured by hardware. A timer value of 1.0ms causes the DCS output to change state 1 msec plus capture delay after the associated input position was captured by hardware.
- Therefore, the DCS firmware extrapolates or looks ahead by the position capture delay plus 1 ms to control a digital output.

For details on the function blocks related to Digital CAM Switch operation, refer to Section 6.9 MC\_DigitalCamSwitch.

### A-2.1 DCS Accuracy at Constant Velocity

The DCS position extrapolation has no error if the position source is running at constant velocity. In that case the only inaccuracies in DCS output switching are caused by:

1. Output device delay:

Output Device	Delay
Fiber Terminal Block 5Vdc output	< 1 us
Fiber Terminal Block 24Vdc output	< 100 us
PMM faceplate output	0.6 ms - 3.6 ms

2. Position source encoder resolution and speed. The additional DCS inaccuracy as a function of encoder resolution and speed =  $1 / (\text{encoder speed in cts/sec})$ .

For example, if a DCS function is used with a position encoder moving at 100,000 cts/sec, the additional DCS inaccuracy is  $1 / 100,000 = 0.00001$  seconds = 10us. This inaccuracy occurs because the encoder only provides new position data every 10us.

## A-2.2 DCS Accuracy During Acceleration

The DCS position extrapolation has an additional error if the position source is accelerating or decelerating. The error during acceleration depends on the position source acceleration and velocity.

### Equations

Constant Velocity (Zero Acceleration):

$$\text{Velocity} = \text{constant}$$

$$\text{Position change} = \text{velocity} * \text{time}$$

Linear Acceleration:

$$\text{Velocity} = \text{Acceleration} * \text{time}$$

$$\text{Position change} = 0.5 * \text{Acceleration} * \text{time}^2$$

### DCS Extrapolation (Look Ahead) Error Calculations

1. DCS firmware calculates the velocity of the position source as the position change during the previous 1.0 ms interval. If the position source is linearly accelerating, the velocity calculated by DCS is incorrect by 0.5 ms – it will be the velocity that actually existed 0.5 ms earlier. The worst-case error (EP1) in extrapolated position due to the velocity calculation error equals  $\text{Acceleration} * 0.5\text{ms} * (1.5\text{ms} + \text{position capture delay})$ . For external encoders this equals:

$$\text{EP1} = A * 0.5\text{ms} * 1.625\text{ms}$$

2. Because rapidly changing acceleration cannot be accurately measured over a 1.0 ms interval, the DCS extrapolation calculations do not include an acceleration term. Therefore, an additional error term EP2 in extrapolated position exists and is equal to  $0.5 * A * (1.5\text{ms} + \text{position capture delay})^2$ . For External encoders the resulting equation is:

$$\text{EP2} = 0.5 * A * (1.625\text{ms})^2$$

The total error in extrapolated position  $\text{EP}_{\text{total}} = \text{EP1} + \text{EP2}$

$$= [A * 0.5\text{ms} * 1.625\text{ms}] + [0.5 * A * (1.625\text{ms})^2]$$

$$= [0.5 * A * 1.0\text{ms} * 1.625\text{ms}] + [0.5 * A * 1.625\text{ms} * 1.625\text{ms}]$$

$$= [0.5 * A * 1.625\text{ms}] * [1.0\text{ms} + 1.625\text{ms}]$$

$$= [0.5 * A * 1.625\text{ms}] * [2.625\text{ms}]$$

$$\text{EP}_{\text{total}} = 2.133\text{e-6} * A$$

where A is acceleration in position units/sec/sec

3. As stated previously, the DCS calculations use a velocity that is effectively 0.5ms old. The actual velocity of the position input at the worst case 1.5 plus position capture delay ms extrapolation point is the measured DCS velocity plus the velocity change over an acceleration time of  $(0.5\text{ms} + 1.5\text{ms} + \text{position capture delay})$ . For External Encoders this is:

$$V_{actual} = V_{meas} + (2.125e-3 * A)$$

where:

A = acceleration in position units/sec/sec

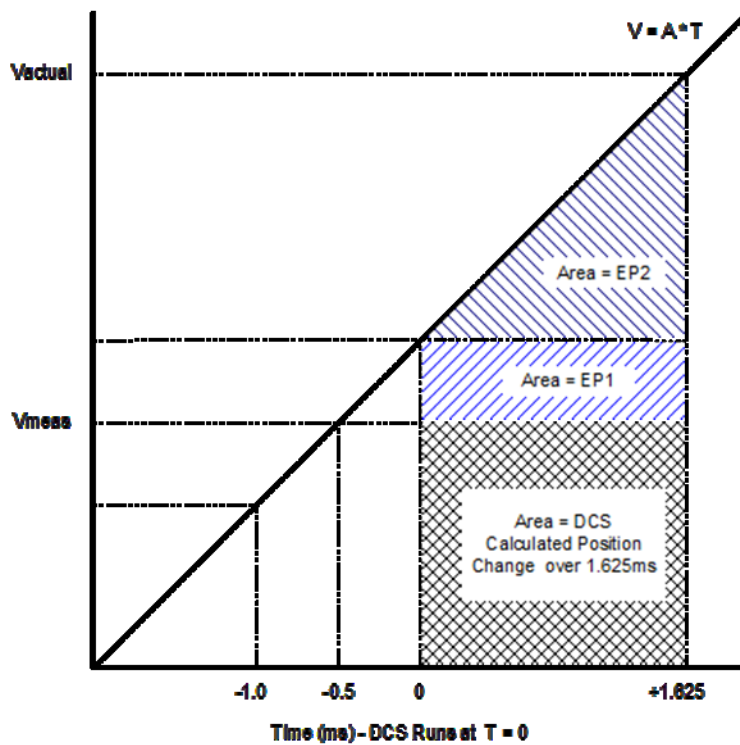
V<sub>meas</sub> = velocity in position units/sec

## DCS Calculated Position Change and Position Error Terms

The following graph of velocity vs time shows:

- Velocities V<sub>meas</sub> and V<sub>actual</sub>
- Area representing DCS calculated position change over 1.625ms (value for External Encoders)
- Areas representing DCS position error terms EP1 and EP2

**Figure 174: Graph showing DCS Calculated Position Change & Position Error Terms**



4. Dividing the worst-case extrapolation error in position (EP<sub>total</sub>) by the actual velocity at the 1.625ms extrapolation point (V<sub>actual</sub>) provides the DCS extrapolation error in units of time:

$$E_{time} \text{ (seconds)} = EP_{total} / V_{actual}$$

## DCS Acceleration Error Examples

The following table shows the maximum DCS acceleration error in units of position and time for several values of acceleration and velocity using an External Encoder.

The table shows that DCS extrapolation error increases with higher acceleration and decreases with higher velocity. The worst-case error occurs at high acceleration and low velocity.

<b>DCS Error vs Position Source Acceleration and Velocity (example uses External Encoder)</b>						
	<b>Example 1</b>	<b>Example 2</b>	<b>Example 3</b>	<b>Example 4</b>	<b>Example 5</b>	<b>Example 6</b>
Encoder Resolution (cts/rev)	65,536	65,536	65,536	65,536	65,536	65,536
Acceleration (RPM/sec)	80,000	40,000	10,000	10,000	4,000	40,000
Acceleration time to 4000 RPM (sec)	0.05	0.10	0.40	0.40	1.00	0.10
Acceleration (cts/sec/sec)	87,381,333	43,690,667	10,922,667	10,922,667	4,369,067	43,690,667
Velocity measured by DCS (RPM)	1,200	1,200	400	2,400	10	80
Velocity measured by DCS (cts/sec)	10,720	1,310,720	436,907	2,621,440	10,923	87,381
Vactual	1,496,405	1,403,563	460,117	2,644,651	20,207	180,224
<b>E<sub>total</sub></b> = DCS Extrapolation Error (cts)	186	93	23	23	9	93
<b>E<sub>time</sub></b> = DCS Extrapolation Error (us)	125	66	51	9	461	517

## Using DCS Acceleration Error Formulas

If the DCS position source acceleration and velocity are known, the equations for  $E_{Ptotal}$ ,  $V_{meas}$  and  $E_{time}$  can be used to calculate the worst case DCS error. The example uses the delay associated with an External Encoder.

$$V_{actual} = V_{meas} + (2.125e-3 * A)$$

$$E_{Ptotal} = 2.133e-6 * A$$

$$E_{time} \text{ (seconds)} = E_{Ptotal} / V_{actual}$$

where

A = acceleration in position units/sec/sec

$V_{meas}$  = velocity in position units/sec

### Calculation Example

DCS position source acceleration = 100 user units/sec/sec

DCS position source velocity = 8 user units/sec

$$V_{actual} = 8 + (2.125e-3 * 100) = 8.2125 \text{ user units/secFUserUnit}$$

$$E_{Ptotal} = 2.133e-6 * 100 = 213.3e-6 \text{ user units}$$

$$E_{time} = 213.3e-6 / 8.2125 = 25.97 \mu\text{s}$$

---

**Note:** The total DCS error is the sum of the acceleration related error and the errors discussed in Appendix C-1.

---

# Appendix B: Position Feedback Devices

## Topics covered:

Digital Serial Encoders

External Quadrature Encoders

## B-1 Digital Serial Encoders

### B-1.1 Digital Serial Encoder Modes

The motor encoder selected and the mode selected in the PMM345 hardware configuration determine if the encoder will be run in incremental or Absolute mode.

#### Incremental Encoder Mode Considerations

The digital serial encoder can be used as an incremental encoder with no revolution counts retained through a power cycle. The equivalent of a marker pulse occurs once each motor shaft revolution. All Home Modes and Set Position commands reference the axis and set the Position Valid flag upon successful completion. The configured End of Travel limits are valid and the Actual Position reported by the PMM will wrap from high to low count or from low to high count values. This is an excellent mode for continuous applications that will always operate via incremental moves, in the same direction. The Home Offset and Home Position inputs to the MC\_Home function block allow simple referencing to the desired location.

#### Absolute Encoder Mode Considerations

The PSD family offers absolute encoders. The encoders maintain absolute position via a mechanical clockworks inside the encoder while the power is off. Reference the motor documentation for additional details (see 1.2.2 for details). A Find Home cycle or Set Position command must be performed initially or hardware configuration is changed in such a way that position is no longer valid. Example hardware configuration change would be changing the application scaling (uu/cnts). Motor Encoder Mode must be set to Absolute in the hardware configuration for proper operation.

#### Absolute Encoder Mode - Position Initialization

When a system is first powered up in Absolute Encoder mode, a position offset for the encoder must be established. Using the Find Home cycle or the Set Position command can accomplish this.

## Find Home Cycle - Absolute Encoder Mode

The Find Home function operates as describe in section 6.18. The Home Offset and Home Position parameters function the same as in Incremental Encoder mode. At the completion of the Home Cycle, the Actual Position is set to the configured Home Position value. The PMM internally calculates the encoder Absolute Feedback Offset needed to produce the commanded Home Position at the completion of the Home Cycle.

Once an absolute position is established by successful completion of a Find Home cycle, the PMM sets the Position Valid flag.

---

**Note:***If the Position Valid flag is set before initiating a Home Cycle, the Home Cycle clears Position Valid and then sets Position Valid again when the cycle completes. If the Home cycle is halted by an MC\_Stop command, Position Valid will remain off.*

---

## Set Position Command - Absolute Encoder Mode

The Set Position command functions the same way as in incremental encoder mode. At the completion of the Set Position operation, Actual Position is set to the Set Position value. The PMM internally calculates the encoder Absolute Feedback Offset needed to produce the commanded Set Position value.

Once an absolute position is established by a Set Position command, the PMM initializes Actual Position and sets the Position Valid bit.

## Absolute Encoder Mode - PMM Power-Up

The absolute encoder has a mechanical works that keeps track of position when power is off.

The PMM completes its power-on diagnostics, and, when configured for absolute encoder mode, interrogates the referenced status of the encoder. A valid referenced status from the encoder signals the PMM to read the encoder absolute position. The PMM reports the Actual Position as the sum of the encoder position and the Absolute Feedback Offset established by the initial Find Home cycle or Set Position command.

When the PMM is configured for absolute encoder there are situations in which the actual position cannot be auto-restored:

- Encoder has certain internal alarms.
- The latest hardware configuration has a change from the previous configuration such as scaling or positioning mode.

In these situations, Position Valid will not be automatically set. A Set Position or Home Cycle will be required to set Position Valid. Commanded position and absolute position are set to the position default value, which is 0 if allowed by the Low Limit/Range (rotary mode) or Software EOT limits. If 0 is not within the limits, the default position is:

Rotary Mode:  $\text{Low Limit} + 0.5 * \text{Range}$

Linear Mode:  $\text{Negative SW EOT} + 0.5 * (\text{Positive SW EOT} - \text{Negative SW EOT})$



## B-2 External Quadrature Encoders

External Quadrature Encoders provide three output signals to the PMM: Channel A, Channel B, and optionally Channel Z (Marker). The Channel A and Channel B signals transition as the encoder turns, allowing the PMM to count the number of signal transitions and calculate the encoder position and direction of rotation.

External Quadrature Encoders are incremental feedback devices; they do not provide a continuous indication of absolute shaft angle as the input shaft rotates. For this reason, the PMM's *Actual Position* status word must be initialized with a known physical position before positioning control is allowed. This position alignment can be accomplished using the *Set Position* or the *Home* cycle. The home cycle makes use of the encoder marker channel, which is a pulse produced once per revolution at a known encoder shaft angle. Successful completion of the *Find Home* cycle or a *Set Position* command causes the PMM to set the axis *Position Valid* flag. *Position Valid* must be set before most motion function blocks are allowed to execute.

---

**Note:** Only incremental quadrature encoders are supported for master axes.

---

### B-2.1 Example: Connecting an External Encoder to Axis 5

External encoders attached to hand wheels, conveyors, or other position sources can be connected to Axis 5 so that it can act as a master source for another axis. In this example, a single-ended encoder is configured, wired to the FTB, and programmed to be the master to a slave axis wired to the FTB.

#### Step 1: Configuration

In the Axis 5 Tab of the PMM345 hardware configuration, set the *External Device* to *External Quadrature Encoder*. Now, for the device, set user units, counts, range, and low position limit as appropriate. (Section 4.3.5, Axis Configuration Data, describes the process in detail.) Here, a 4096 counts-per-revolution encoder (1024 pulses/channel per revolution) is used and programmed in revolutions. External Quadrature Encoders are always treated by the PMM as *Rotary*. The range is set to 5000 revolutions at which point it will rollover to zero. The Axis Direction should also be set. Because no servo is being controlled, either direction may be chosen. In this case, *Reverse* is chosen.

---

**Note:** Axis 5 contains two logical masters: an External Master – as being configured here – and a Virtual Master, the commanded position of which may be followed by any slave axis in the system. All of the Command parameters, including Max Velocity and Axis Positioning Mode, apply to the Virtual Master while all of the Device parameters, including Axis Direction and Feedback Moving Deadband, apply to the External Master. The two masters are not connected in any way.

---

**Figure 175: PME Axis 5 Tab of PMM345 used to Define External Quadrature Encoder as External Device**

Parameters	Values
Stop Axis on FTB Error	Disabled
Axis Positioning Mode	Rotary
Command Position Resolution (uu)	0.666666667
Command Position Range (uu)	11184810.67
Command Low Position Limit (uu)	-5592405.333
Command Counts Per Motor Revolu...	65,536
<b>External Device</b>	<b>External Quadrature Encoder</b>
External Device User Units	1.0
External Device Counts	<b>4096</b>
External Device Position Range...	<b>5000.0</b>
External Device Low Position Li...	<b>0.0</b>
Axis Direction	<b>Reverse</b>
Software End of Travel	Disabled
Max Velocity System (RPM)	4000.0
Equivalent Velocity (uu/sec)	2912711.11256747
Max Acceleration System (RPM/sec)	915527343.75 <span style="border: 1px solid black; padding: 2px;">Non-editable Value</span>
Equivalent Acceleration (uu/sec...	666666667000.0
Max Deceleration System (RPM/sec)	915527343.75
Equivalent Deceleration (uu/sec...	666666667000.0
Max Jerk (uu/sec**3)	666666666666667.0
Error Stop Deceleration (uu/sec**2)	66666.6666666667
Error Stop Jerk (uu/sec**3)	666666.666666667
Feedback Moving Deadband (uu/s...	333.333333333333

Next, select the encoder inputs in the PMM345 hardware configuration. Note that the Mode for these must be set to Single-ended. The Section FTB I/O Functions Summary in Chapter 4 lists inputs for Axis 5 encoder signals. In the example below IN17, IN18, and IN19 are selected for the A, B, and Z channels respectively.

**Figure 176: Select Encoder Inputs in the PMM345 HWC**

Parameters	Values
FTB IN11 Input Ref	M1_FT_B_IN11
<i>FTB IN12</i>	Digital Input
FTB IN12 Input Ref	M1_FT_B_IN12
<i>FTB IN13</i>	Digital Input
FTB IN13 Input Ref	M1_FT_B_IN13
<i>FTB IN14</i>	Digital Input
FTB IN14 Input Ref	M1_FT_B_IN14
<i>FTB IN15</i>	Digital Input
FTB IN15 Input Ref	M1_FT_B_IN15
<i>FTB IN16</i>	Digital Input
FTB IN16 Input Ref	M1_FT_B_IN16
<b>FTB IN17</b>	<b>Axis 5 Encoder A Channel</b>
FTB IN17 Input Ref	M1_FT_B_IN17
<i>FTB IN17 Mode</i>	<b>Single Ended</b>
<i>FTB IN17 Fault Detect</i>	Disabled
<b>FTB IN18</b>	<b>Axis 5 Encoder B Channel</b>
FTB IN18 Input Ref	M1_FT_B_IN18
<i>FTB IN18 Mode</i>	<b>Single Ended</b>
<i>FTB IN18 Fault Detect</i>	Disabled
<b>FTB IN19</b>	<b>Axis 5 Encoder Marker</b>
FTB IN19 Input Ref	M1_FT_B_IN19
<i>FTB IN19 Mode</i>	Differential
<i>FTB IN19 Fault Detect</i>	Disabled
<i>FTB IN20</i>	Fast Digital Input
FTB IN20 Input Ref	M1_FT_B_IN20
<i>FTB IN20 Mode</i>	Differential
<i>FTB IN20 Fault Detect</i>	Enabled
<i>FTB IN21</i>	Fast Digital Input
FTB IN21 Input Ref	M1_FT_B_IN21
<i>FTB IN21 Mode</i>	Differential
<i>FTB IN21 Fault Detect</i>	Enabled
<i>FTB IN22</i>	Fast Digital Input
FTB IN22 Input Ref	M1_FT_B_IN22

## Step 2: Wiring

Section 3.2.4, FTB Wiring Diagrams and Pin Assignment maps the pins to the inputs. (Refer to Section 3.2.6, Typical Single-Ended Encoder Connection for FTB. (For a wiring example of a differential encoder.)

Since this is a single-ended encoder, only the “+” side should be connected. Refer to Section 3.2.6, Typical Single-Ended Encoder Connection for FTB.

---

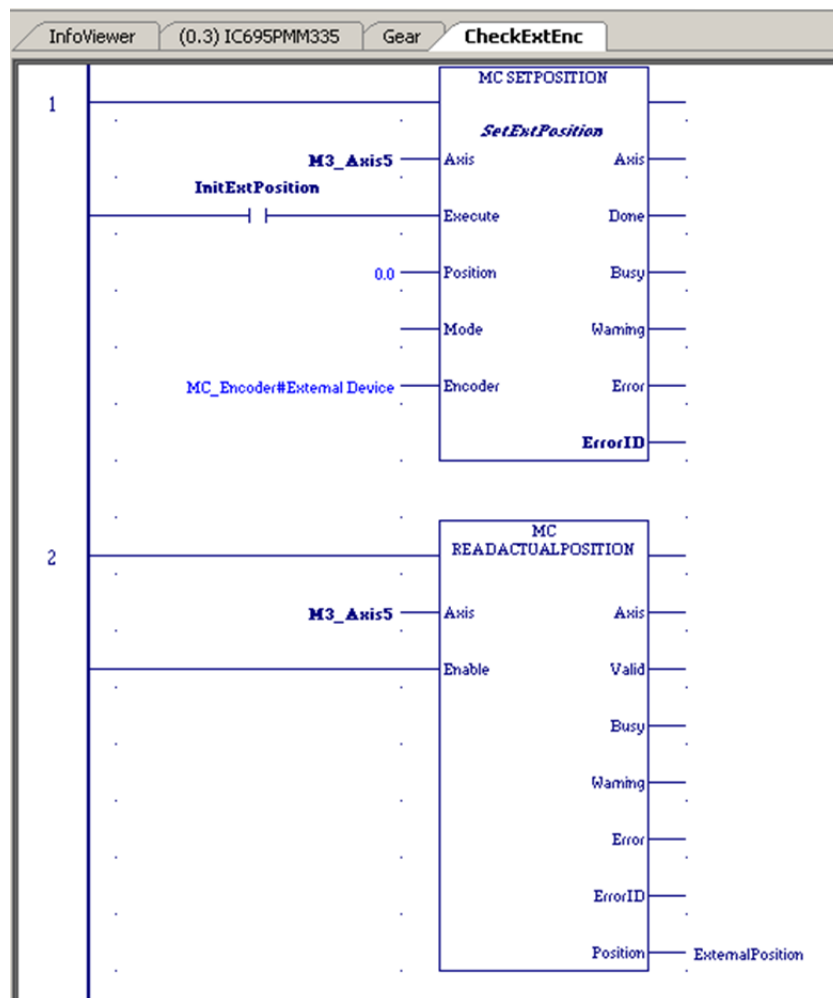
**Note:** For grounding and noise reduction recommendations, refer to Section 3.7 Grounding the PACMotion System.

---

## Step 3: Basic Checking

The external encoder should be assigned a known position using the MC\_SetPosition FB. The Encoder input must be MC\_Encoder#ExternalDevice. One simple block can now be executed to check if the encoder is connected properly. Download and Run a program with the MC\_ReadActualPosition FB with Axis 5 as the Axis Input. Toggle On InitExtPosition, and then move the encoder to observe the position change.

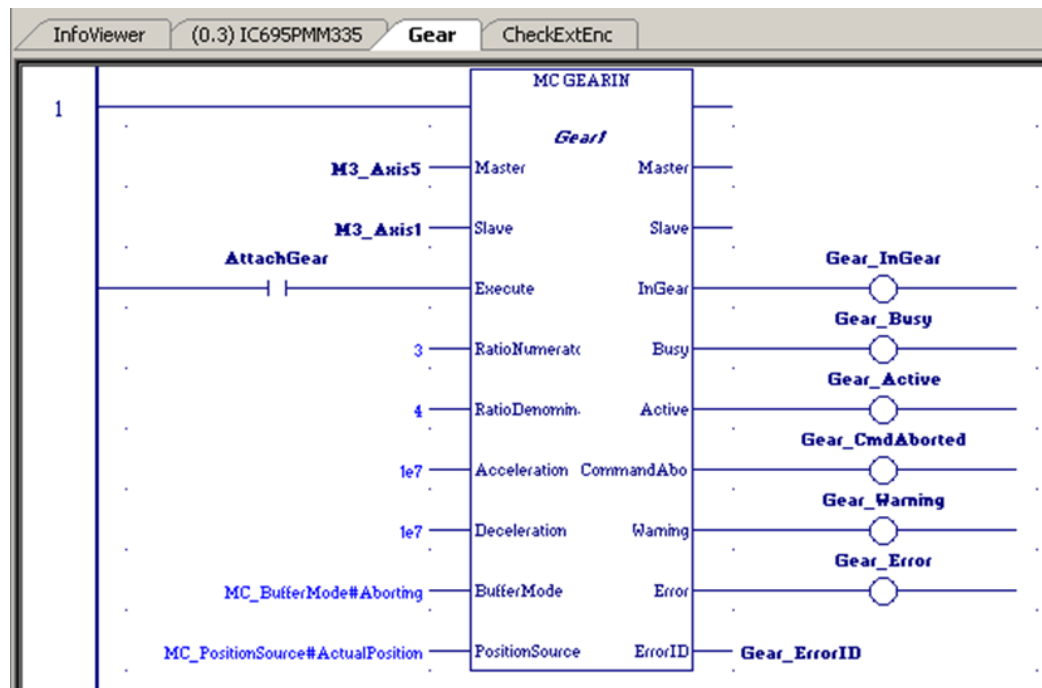
**Figure 177: Ladder Logic to Check External Encoder Connection & Set-up**



### Step 4: Programming a Gear

To program a simple velocity-follower (gear), Axis 1 is geared to the external master on Axis 5 at a 3:4 ratio. Then axis 1 must be powered and set to PositionValid (using either MC\_Home or MC\_SetPosition). The PositionSource input must be MC\_PositionSource#ActualPosition for the external master to be followed. Toggle On AttachGear and move the encoder. Axis 1 should follow at a 3:4 ratio.

**Figure 178: Ladder Logic to Program a Simple Velocity-Follower (Gear)**



# Appendix C: Tuning Digital and Analog Servo Systems

This chapter introduces the basics required for tuning a servo drive. The methods shown in this introduction may not work in all applications; the approach should be modified based on the application.

Before tuning the drive, first make sure that the parameters discussed in Section 4.3, Configuring PMM Parameters have been reviewed and properly set. This appendix provides a procedure for validating Axis Direction and the operation of Over Travel and Home Switch inputs.

The Data Logging Window can be used to display and measure the necessary signal waveforms. For information on setting up and using the Data Logging feature, refer to Section 6.10, MC\_DL\_Activate; Section 6.12, MC\_DL\_Delete; and Section 6.10 MC\_DL\_Activate. Additionally, the PACMotion Workbench provides a “Scope” function to allow gathering information while performing tuning for the drive specifically.

Topics covered:

C-1 Validating Axis Direction, Over Travel Switch and Home Switch Inputs

C-2 Forcing Servo Velocity

C-3 Tuning a PM EtherCAT Servo

C-4 Tuning an Analog, Velocity-Controlled Drive

C-5 Tuning an Analog, Torque-Controlled Drive

## C-1 Validating Axis Direction, Over Travel Switch and Home Switch Inputs

### CAUTION

Until the axis direction configuration has been validated, it is recommended that the motor be uncoupled from the machine.

### C-1.1 Validating Motor Direction

To determine whether Axis Direction is correctly configured for the axis:

1. With the axis Position Feedback Source configured as Motor Encoder, use an MC\_JogAxis function block to jog the motor in the **positive** direction at low speed, while observing the motor direction of rotation.
2. If the motor rotates in the direction defined as **positive** for the machine, Axis Direction is configured correctly. If the motor rotates in the wrong direction, change the Axis Direction configuration parameter to Reverse. Store the configuration and repeat the jog test to make sure Axis Direction is now configured correctly.

### C-1.2 If an external quadrature encoder feedback device will be used for the motor

1. Temporarily keep the Position Feedback Source configured as Motor Encoder and use an MC\_ReadParameter function block to monitor parameter 1309, External Device Velocity. Jog the motor in the positive direction and confirm that External Device Velocity is a positive value.
2. If External Device Velocity has the wrong polarity, change the external quadrature encoder feedback phase by swapping the A and B phase signals at the terminal block. Repeat the jog test to make sure External Device Velocity has the correct polarity.
3. Confirm that the configuration for External Device Counts Per Motor Revolution is set correctly. The value must take into consideration the resolution of the quadrature encoder in counts per revolution (x4 mode) and the gear ratio between the motor and external encoder.

#### Example:

An encoder with a 2048-line optical disc produces 8192 counts/revolution in x4 mode. If the encoder is geared to turn  $\frac{1}{2}$  as fast as the motor, the External Device Counts Per Motor Revolution must be set to 4096.

4. Set Position Feedback Source to External Device in hardware configuration. Store the configuration and jog the motor again to confirm that the position loop is operating correctly with the external feedback device.

## C-1.3 Validating the Over Travel Limit Switch Inputs

If Over Travel Limit switches are not used, the Over Travel Limit Switch parameter must be set to Disabled in the hardware configuration.

If Over Travel Limit switches are used (Over Travel Limit Switch is Enabled in hardware configuration), wire them to the correct input points (refer to Section 3, I/O Wiring, Connections and LED Operation). The Over Travel Limit switches must be assigned to the correct PMM faceplate or FTB input point in hardware configuration.

---

**Note:** *The Over Travel Limit switches operate in the fail-safe mode, therefore each switch must be on and current must flow in the associated input while the axis is not at or beyond the overtravel position. Otherwise an alarm condition will be reported, and the servo drive will not operate.*

---

Follow these steps to test the Over Travel Limit switches and wiring:

1. Close the positive and negative Over Travel Limit switches.
2. Enable the servo drive using the MC\_Power function block. Use an MC\_ReadAxisError function block to read Parameter 1100, Axis Error Code.
3. Open the positive Over Travel switch and confirm that the servo drive disables. Error code 0x60a0 should be reported in parameter 1100, Axis Error Code.
4. Close the positive Over Travel switch. Use an MC\_Reset function block to clear the Over Travel error and allow MC\_Power to re-enable the servo drive. Open the negative Over Travel switch and confirm that the servo drive disables. Error code 0x60a1 should be reported in parameter 1100, Axis Error Code.
5. To diagnose wiring problems with the Over Travel Limit switches, use an MC\_ReadDwordParameters function block to monitor the bits in parameters 2107 (PMM Faceplate Inputs 1-8) and 2108 (FTB Inputs 1-8).

## C-1.4 Validating the Home Switch Input

If a Home Switch is used, wire it to the correct PMM faceplate, FTB input point or Servo Drive Input (refer to Chapter 3, I/O Wiring, Connections and LED Operation and the Servo Drive Manual). The Home Switch must be assigned to the correct PMM faceplate, FTB input or Servo Drive Input point in hardware configuration.

The Home Switch must be actuated so that it is **Always On** (closed) when the axis is on the negative side of home and **Always Off** (open) when the axis is on the positive side of home. Typically, the Home Switch is mounted at or near one end of the axis travel. It is important to verify the operation of the Home Switch before attempting a home cycle.

**Follow these steps to verify the Home Switch wiring and operation:**

1. Use an MC\_ReadBoolParameter function block to read Parameter 1223, Home Switch.
2. Confirm that Parameter 1223, Home Switch = 1 when the axis is on the negative side of home.

3. Confirm that Parameter 1223, Home Switch = 0 when the axis is on the positive side of home.
4. If an external encoder is configured for position feedback, confirm that the encoder marker channel is wired to an FTB 5Vdc input and the associated FTB hardware configuration is correctly set.
5. The motor encoder or external encoder reference pulse operation should be checked before exercising the Home Switch in a home cycle. For details of the MC\_Home function block, refer to Section 6.18 MC\_Home. To confirm that the encoder reference pulse operates correctly, program an MC\_Home function block with the Homing Mode set to RefPulse. The Find Home Velocity and Final Home Velocity parameters for MC\_Home should initially be set to a low value (equivalent of 5–10 RPM).
6. Execute the MC\_Home function block. The axis should move in the positive direction no more than one revolution before the encoder reference point is located and the home cycle completes.
7. If the axis moves more than one revolution and does not find the encoder reference, remove the Enable from MC\_Power or use an MC\_Halt function block to stop axis motion. If an external encoder is used for position feedback, review the encoder marker channel wiring and associated FTB input configuration.

---

**Note:** Do not proceed to the next step until MC\_Home operates correctly with the Homing Mode set to RefPulse.

---

8. Change the Homing Mode parameter to Limit Switch Reference Pulse. Execute the MC\_Home function block. The axis should move toward the Home Switch at the configured Find Home Velocity. If the axis initially moves away from the Home Switch, remove the Enable from MC\_Power or use an MC\_Halt function block to stop axis motion. Review the Home Switch wiring and operation as described in steps 1–3.
9. If necessary, adjust the MC\_Home parameters and the location of the Home Switch for consistent operation. The Home Switch must be mounted so that the final Home Switch off to on transition occurs at least 10 ms before the encoder reference point is encountered.

## C-1.5 PM EtherCAT Servo System Start-up Diagnostics

1. A powered servo drive corresponding to every axis enabled as PM EtherCat Servo on the Settings tab of the PMM hardware configuration must be present in the EtherCAT network.
2. The default PMM configuration for the Over Travel Limit Switch inputs is Enabled. Therefore, a logic high signal must be applied to the overtravel inputs or the axis will not operate. If overtravel inputs are not used, Over Travel Limit Switch inputs should be set to Disabled.
3. The axis must be enabled by an MC\_Power function block and not be in the ErrorStop state, or no motion other than Jog will be allowed. The corresponding Axis OK bit in the module status data will be ON. To monitor axis status, use an MC\_ReadStatus function block.
4. If an axis is in ErrorStop state, the error condition must be corrected and the error cleared using an MC\_Reset function block. A MC\_Reset transitions the axis from ErrorStop to Standstill state.



5. The CONFIG LED must be ON or the PMM will not respond to PACSystems CPU commands. If the LED is flashing green, a valid configuration has not been received from the PACSystems CPU. If the LED is flashing amber, an invalid configuration is indicated. Also check the fault tables for reported configuration errors.

## C-2 Forcing Servo Velocity

The Force Servo Velocity (FSV) parameter sends a velocity command directly to the velocity control loop, bypassing the path generation and position loops. When FSV is active, the axis enters the Setup state, which can be verified using MC\_ReadStatus.

To use FSV, the axis must be in the Standstill state with the PositionValid (PN 1201) axis status flag set True. The FSV can then be commanded by first writing a timeout specifying the move duration followed by writing the velocity of the move.

ForceServoVelocityTimeout (PN 1320) is a DWORD parameter that is used to specify how long (in ms) the FSV move should last. The valid range is 0ms to 10000ms.

ForceServoVelocity (PN 1311) is an LREAL parameter that sets the desired servo velocity in RPM. The forced velocity cannot exceed the MaxVelocityAppl (PN 9). The axis enters the Setup state and the timeout counter starts when ForceServoVelocity is written. The axis returns to the Standstill state when the timeout expires or a value of zero is written to the ForceServoVelocityTimeout parameter.

While the axis in the Setup state it is allowed to rewrite the velocity to either extend the length of the FSV or with a value of zero to immediately stop motion.

FSV cannot be aborted by other moves.

### **⚠ CAUTION**

While in the Setup state, the servo does not check for Software End of Travel limits. Hardware Over Travel Limit Switches are still enforced.

## C-3 Tuning a PM EtherCAT Servo

There is one control loop in the PMM that require tuning - the position loop. When tuning the servo, always adjust the velocity loop first, followed by the position loop. The PACMotion Workbench tool contains autotuning functions to allow the user to tune the velocity and current loop. This requires directly connecting the PACMotion Workbench tool to the top Ethernet port on the servo drive. Once PACMotion Workbench is connected the user can utilize the autotuning tools to correctly tune the drive control loops. It is suggested to tune the servo drive position loop as well. This loop is used during the homing procedures that the servo drive controls.

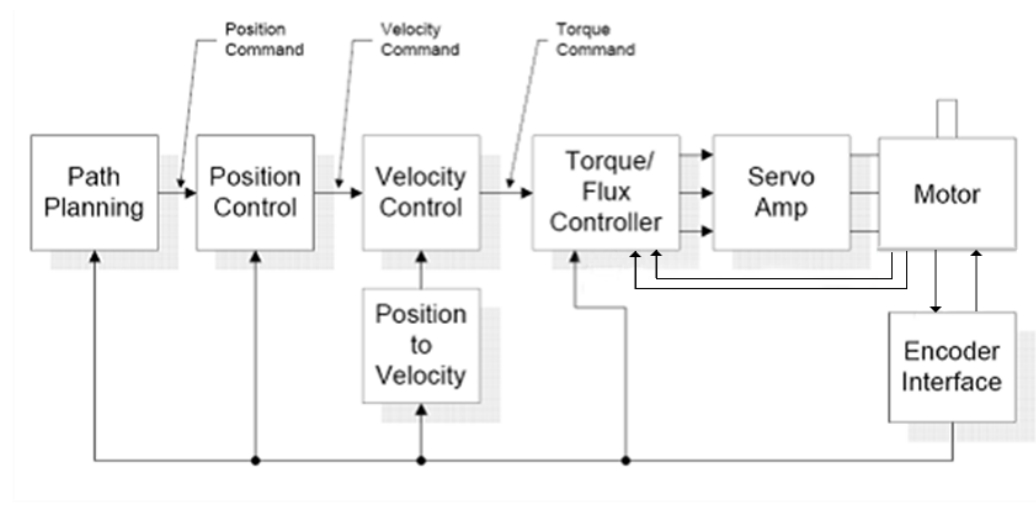
## C-3.1 Tuning Requirements

The module has two main parameters that are adjusted during position loop tuning. The parameters are Position Loop Time Constant and Velocity Feed Forward.

The approach to tuning the control loops is to tune the inner control loops first. In this example, the inner control loop that requires tuning is the velocity loop. As shown in Figure 179, the position loop is the outer loop and sends velocity commands to the velocity loop.

### Control Loops Block Diagram

Figure 179: Control Loops Block Diagram



### Tuning the Velocity Loop

The PACMotion Workbench tool contains autotuning functions to allow the user to tune the velocity loop. This requires directly connecting the PACMotion Workbench tool via Ethernet servo drive. Once PACMotion Workbench is connected the user can utilize the autotuning tools to correctly tune the drive control loops. Reference Servo Drive documentation (see section 1.2.2) and Workbench help for additional details on this topic.

## Tuning the Position Loop

The first step in adjusting the position loop tuning is to ensure that the velocity loop is stable and has response suitable to the application. Refer to the previous section for methods of setting the velocity loop.

### Preliminary Position Loop Settings for Tuning Session

Set the User Unit and Counts configuration parameters to values appropriate to the axis mechanical configuration. See the discussion and examples in the Configuration chapter for details.

If using an external quadrature encoder for the position loop feedback, set the External Device User Units and Counts values as described in the Configuration chapter.

Set Velocity Feedforward % to zero.

Set the Max Position Error to 10 Motor Revolutions.

### Setting the Position Loop Gain

The position control loop is primarily a “P” (Proportional) algorithm with optional Velocity Feed Forward. Begin tuning the position loop by setting the proportional gain (Position Loop Time Constant) to provide a stable response with sufficient gain (bandwidth) to meet the motion profile requirements. The default Position Loop Time Constant is set in the Axis Configuration (refer to Section 4.3.5, Axis Configuration Data). Position Loop Time Constant can also be modified using an MC\_WriteParameter function block to change parameter number 1009.

A typical value for Position Loop Time Constant in high performance systems is 30-60 ms. There are two suggested methods of setting the Position Loop Time Constant.

#### Position Loop Proportional Gain Method 1

Calculating the position loop proportional gain assumes that the mechanical design of the machine will have sufficient bandwidth to remain stable and that any resonant frequencies are higher than the bandwidth required by the motion profile.

### Terminology

A large mismatch between the load and motor inertia can cause a resonance in the system. Resonance is oscillatory behavior caused by mechanical limitations and aggravated by gearing backlash or torsion windup of mechanical members like couplings or shafts. Resonance is eliminated by improving the mechanics, reducing load/motor inertia mismatch or reducing servo gains (reduce performance).

Bandwidth is a figure of merit used to compare control system or mechanical performance. As the frequency of command increases, the system response will begin to lag. The bandwidth is defined as the frequency range over which system response (gain) is at least 70% (-3 decibels) of the desired command.

## High Bandwidth

- Allows the servo to more accurately reproduce the desired motion
- Allows accurate following of sharp corners in motion paths and high machine cycle rates
- Rejects torque disturbances from mechanics or outside influences improving system accuracy
- Can expose machine resonance, which occur at frequencies near or below the bandwidth

The response of a proportional only system is an exponential rise. A time constant for an exponential curve represents 68% of the remaining rise. For instance, starting at zero velocity, the response of the position loop to a change in command will require one time constant to reach 68% of the commanded velocity. The 2nd time constant will reduce 68% of the remaining command. Subsequent time constants will reduce 68% of remaining command. For example,  $100\% - 68\% \text{ (one time constant)} = 32\%$ ,  $32\% (68\%) = 21.8\%$ ,  $68\% \text{ (first time constant)} + 21.8\% \text{ (second time constant)} = 89.8\%$ . Two-time constants eliminate 89.8% of the command. Three-time constants will account for 96.7% of the rise in command. Four-time constants account for 98.9% of the rise. Typically, three-time constants are sufficient for most motion applications.

You can use your knowledge of time constants to predict the required system response. For instance, if the fastest acceleration required in the motion profiles must occur within 200 ms, the 200 ms response to the change in command will be 98.9% complete in three-time constants. Dividing the 200 ms by 3 results in a time constant of about 67 ms. **The Position Loop Time Constant parameter represents the duration of one time constant in ms.** In the example above, the duration of the time constant is 67 ms.

### Position Loop Proportional Gain Method - Method 2

This procedure is similar to the Tuning the Velocity Loop method described in Appendix Section C-3. Using the Data Logging Window to monitor the Servo Actual Velocity, lower the Position Loop Time Constant value (thus increasing gain) to obtain the performance characteristics desired.

## Optimizing Velocity Feedforward

Before setting Velocity Feedforward, the Position Loop Time Constant should be set as previously described. The servo system capabilities will determine the optimum value of Velocity Feedforward. The default Velocity Feedforward percentage is set in the Axis Configuration (refer to Section 4.3.5, Axis Configuration Data). Velocity Feedforward percentage can also be modified using an MC\_WriteParameter function block to change parameter number 1010.

Start with 100% Velocity Feedforward. In many systems, 100% is the optimum Velocity Feedforward value. With the servo at constant velocity, 100% feedforward causes Position Error to stay close to zero. The servo will follow the path generator command very closely as long as the path generator commanded acceleration does not exceed the acceleration (torque) capability of the servo.

If excessive position overshoot occurs when stopping, reduce Velocity Feedforward in 1% increments until the overshoot is acceptable.

---

**Note:** *If Velocity Feedforward is changed, Max Position Lag may require adjustment. Max Position Lag should normally be set to a value 10% to 20% higher than the highest position lag encountered under normal servo operation. It must be less than or equal to Max Position Error. For details, refer to Section 4, Configuration.*

---

## C-4 Tuning an Analog, Velocity-Controlled Drive

### C-4.1 Wiring and Configuration

1. Connect the motor to the analog velocity interface servo drive according to the manufacturer's recommendations.
2. Connect the Drive Enable and Analog Servo Control outputs to the servo drive.
3. If the servo drive provides a reset input, connect the Analog Axis Reset output to the servo drive.
4. Connect the External Quadrature Encoder inputs to the FTB.

---

**Note:** *If these connections are incorrect or there is slippage in the coupling to the Feedback Device, an **Out of Sync** error condition (x26D) can occur when motion is commanded.*

---

5. Connect the servo drive Ready output (if available) to the Drive Status input. This signal must turn on when the servo drive is ready to control the servo. If the servo drive does not provide a suitable Ready output, the Drive Status input can be disabled in the module configuration.
6. If a Home switch is used (24Vdc), wire it to the correct FTB input. The Home switch must be wired so that it is ALWAYS ON when the axis is on the negative side of home and ALWAYS OFF when the axis is on the positive side of home.
7. Use the configuration software to set the desired parameters. Store the configuration to the host controller.

Parameter	Config Tab	Description
Analog Mode	Settings tab	Must be set to Analog Servo Velocity Mode. This is not the default value.
Drive Status Input	Axis tab	If the analog servo drive provides a drive status signal, select the appropriate type of feedback. (Requires an Analog Axis Drive Status input to be configured on the FTB Inputs tab.)
Max Velocity System	Axis tab	Determines the maximum velocity the servo will be commanded to run. In the early tuning stages, it is advisable to set this value relatively low. This will allow the system to be brought up in stages. Once basic operation and tuning has been verified, the maximum value can be raised to the value that is determined by either the process limitations or servo drive/motor set.
Torque Limit	Axis tab	Determines the maximum analog torque command that will be sent to the servo drive. In the early tuning stages, it is advisable to set this value relatively low. Once basic operation is verified, the torque limit value can then be set to the value desired for the application.
External Device Counts per Motor Revolution	Axis tab	For correct analog mode operation, this value must be set to the number of quadrature encoder counts generated by the motor feedback device per revolution. You can determine the value from the feedback device specification. As a double check, you may wish to connect the feedback device to the FTB and manually rotate the motor shaft one revolution. The actual position value of the axis reported by PN1300 should closely match (variations are caused by the accuracy of manually turning the shaft one revolution) the value placed in this parameter.
FTBInx	FTB Inputs tab	Select FTB inputs for Encoder A and Encoder B Channels. If the following signals will be used, assign inputs for them: Analog Axis Drive Status, Touch Probes, Home Switch and Overtravel.
FTBOUTx	FTB Outputs tab	Select FTB outputs for Analog Axis Drive Enable and Analog Axis Reset (if used).
FTBALGOUT x	FTB Outputs tab	Assign an Analog Servo Control output for the axis.

---

**Note:** For proper servo operation, **External Device Counts per Motor Revolution** *MUST* be set to the correct value for the servo drive/motor set. If this value is not set correctly, instabilities can result.

---

## C-4.2 Validating Axis Configuration and Servo Drive Settings

1. Turn on the Analog Axis Drive Enable output. Use MC\_WriteAnalogOutput to write a value of 0 to the FTB ALGOUT output. Confirm that the servo drive is enabled (the motor should exhibit holding torque). If the motor stops moving, adjust the servo drive command offset adjustment until the motor stops moving.

---

**Note:** *The Analog Axis Drive Enable output must be maintained ON for the MC\_WriteAnalogOutput command to function.*

---

2. Use MC\_WriteAnalogOutput to write a value of +1.0V to the FTB ALGOUT output. Confirm that the motor moves in the positive direction as set on the servo drive and the Actual Velocity reported is positive. If the motor moves in the wrong direction, consult the servo drive manufacturer's instructions for corrective action. If the motor moves in the positive direction but the PMM reports that Actual Velocity is negative, the encoder channel A and channel B inputs must be swapped.
3. Record the actual motor velocity reported by the PMM with a 1.0V velocity command. Multiply this velocity by 10 and change the Motor Velocity at 10 Volts configuration parameter, if necessary. Initially set the Pos Loop Time Constant configuration parameter to a high value (typically 100ms).
4. Use the MC\_JogAxis function block to move the axis. Confirm that the servo moves in the proper direction and that the Actual Velocity reported by the PMM matches the JogVelocity.
5. With the Drive Enable output ON and no servo motion commanded, adjust the servo drive command offset adjustment for zero Position Error. The integrator should be OFF during this process.
6. Check for proper operation of the Over Travel Limit Switch inputs. For steps to perform this procedure, refer to C-1.3 *Validating the Over Travel Limit Switch Inputs*.
7. Check for proper operation of the Find Home cycle by executing an MC\_Home function block on the axis. For steps to perform this procedure, refer to Appendix *Validating the Home Switch Input*.
8. Monitor servo performance and use the MC\_JogAxis function block to move the analog servo motor in each direction. The Position Loop Time Constant can be temporarily modified by writing a value to PN1009. For most systems, the Position Loop Time Constant can be reduced until some servo instability is noted, then increased to a value approximately 50% higher. Once the correct time constant is determined, the PMM configuration should be updated using the configuration software. Velocity Feedforward can also be set to a non-zero value (typically 90 to 100%) for optimum servo response.

---

**Note:** *For proper servo operation, the Configuration entry for **Velocity at Max Cmd** MUST be set to the actual servo velocity (in User Units/sec) caused by a 10Vdc velocity command to the servo drive.*

---



## C-5 Tuning an Analog, Torque-Controlled Drive

There are two control loops that require tuning - the velocity loop and the position loop. When tuning the servo, always adjust the velocity loop first, followed by the position loop.

PMM firmware version 1.00 or later is required to use the Analog Servo Torque mode.

### C-5.1 Wiring and Configuration

1. Connect the motor to the analog torque interface servo drive according to the manufacturer's recommendations.

---

**Note:** The servo drive must be configured to accept voltage ( $\pm 10\text{Vdc}$ ) that corresponds to motor torque.

---

2. Connect the Drive Enable and Analog Servo Control outputs to the servo drive.
3. If the servo drive provides a reset input, connect the Analog Axis Reset output to the servo drive.
4. Connect the External Quadrature Encoder inputs to the FTB.

---

**Note:** If these connections are incorrect or there is slippage in the coupling to the Feedback Device, an Out of Sync error condition (x26D) can occur when motion is commanded.

---

5. Connect the servo drive Ready output (if available) to the Drive Status. This signal must turn on when the drive is ready to control the servo.

---

**Note:** Incorrect Drive Status configuration or wiring will cause error code x0C0, Servo Not Ready, to be reported.

If Drive Ready is selected in the module configuration, the Drive Status input must be turned on within 500ms after the Drive Enable turns on and off within 500ms after Drive Enable is turned off, or an error will occur.

If Drive Available is selected, the Drive Status input must remain on while Drive Enable is on, or an error will occur. If the servo drive does not provide a suitable Ready output, this input to the FTB must be connected to 0V or the Drive Status input can be disabled in the module configuration.

---

6. If a Home switch is used (24Vdc), wire it to the correct FTB input. The Home switch must be wired so that it is always ON when the axis is on the negative side of home and always OFF when the axis is on the positive side of home.
7. Use the configuration software to set the desired parameters. Store the configuration to the host controller. You will need to set the following specific parameters:

Parameter	Config Tab	Description
Analog Mode	Settings tab	Must be set to Analog Servo Torque Mode. This is not the default value.
Drive Status Input	Axis tab	If the analog servo drive provides a drive status signal, select the appropriate type of feedback. (Requires an Analog Axis Drive Status input to be configured on the FTB Inputs tab.)
Max Velocity System	Axis tab	Determines the maximum velocity the servo will be commanded to run. In the early tuning stages, it is advisable to set this value relatively low. This will allow the

Parameter	Config Tab	Description
		system to be brought up in stages. Once basic operation and tuning has been verified, the maximum value can be raised to the value that is determined by either the process limitations or servo drive/motor set.
Torque Limit	Axis tab	Determines the maximum analog torque command that will be sent to the servo drive. In the early tuning stages, it is advisable to set this value relatively low. Once basic operation is verified, the torque limit value can then be set to the value desired for the application.
External Device Counts per Motor Revolution	Axis tab	For correct analog mode operation, this value must be set to the number of quadrature encoder counts generated by the motor feedback device per revolution. You can determine the value from the feedback device specification. As a double check, you may wish to connect the feedback device to the FTB and manually rotate the motor shaft one revolution. The actual position value of the axis reported by PN1300 should closely match (variations are caused by the accuracy of manually turning the shaft one revolution) the value placed in this parameter.
Velocity Loop Proportional Gain	Axis tab	The parameter is only used in analog torque mode. The proportional gain is multiplied by the velocity error (velocity command - velocity feedback) to generate the portion of the torque command due to the proportional term. Correctly setting this value will determine how well the velocity regulator performs in the control system. The following sections will discuss how to set this value.
Velocity Regulator Proportional Gain	Axis tab	The parameter is only used in analog torque mode. The integral gain is the term multiplied by the area of the velocity error (velocity command - velocity feedback) to generate the portion of the torque command due to the integral term. Correctly setting this value will determine how well the velocity regulator performs in the control system. The following sections will discuss how to set this value.
FTBInx	FTB Inputs tab	Select FTB inputs for Encoder A and Encoder B Channels. If the following signals will be used, assign inputs for them: Analog Axis Drive Status, Touch Probes, Home Switch and Overtravel.
FTBOUTx	FTB Outputs tab	Select FTB outputs for Analog Axis Drive Enable and Analog Axis Reset (if used).
FTBALGOUT x	FTB Outputs tab	Assign an Analog Servo Control output for the axis.

**Note:** For proper servo operation, **External Device Counts per Motor Revolution** MUST be set to the correct value for the servo drive/motor set. If this value is not set correctly instabilities can result.

- Turn on the Analog Axis Drive Enable output. Send a Force Servo Velocity command (PN1311) with a value of 0 RPM. Confirm that the servo drive is enabled (the motor

should exhibit holding torque). If the motor moves, adjust the servo drive command offset adjustment until the motor stops moving

For information on the operation of the FSV parameter, refer to Appendix

## C-5.2 Verifying Basic Analog Control Functions

### **⚠ CAUTION**

Make sure that the motor shaft is not connected to the load when first performing the following operation.

1. Send a Force Servo Velocity command (PN1311) equal to 10 RPM. Confirm that the motor moves in the desired positive direction as set on the servo drive and the Actual Velocity reported by the PMM is positive. If the motor moves in the wrong direction, consult the servo drive manufacturer's instructions for corrective action. If the motor moves in the positive direction but the PMM reports that Actual Velocity is negative, the encoder channel A and channel B inputs must be swapped.

For information on the operation of the FSV parameter, refer to Appendix

2. With the Axis powered on (Axis status is On) and no servo motion commanded, adjust the servo drive so no motion is generated. The Velocity Loop Integral Gain MUST be set to 0 to properly complete this step.
3. Check for proper operation of the Over Travel Limit Switch inputs. For steps to perform this procedure, refer to Appendix C-1.
4. Check for proper operation of the Find Home cycle by executing an MC\_Home function block on the axis. For steps to perform this procedure, refer to Appendix C-1.
5. Once correct basic operation has been achieved, the velocity loop requires tuning. Appendix C-1 contains a basic procedure for tuning the loop.

---

**Note:** *The tuning procedure for Torque Mode velocity regulators is different from Digital Mode Velocity regulators. Do not proceed to tuning the Position Loop until the velocity loop tuning is complete.*

---

## C-5.3 Tuning the Torque Mode Velocity Loop

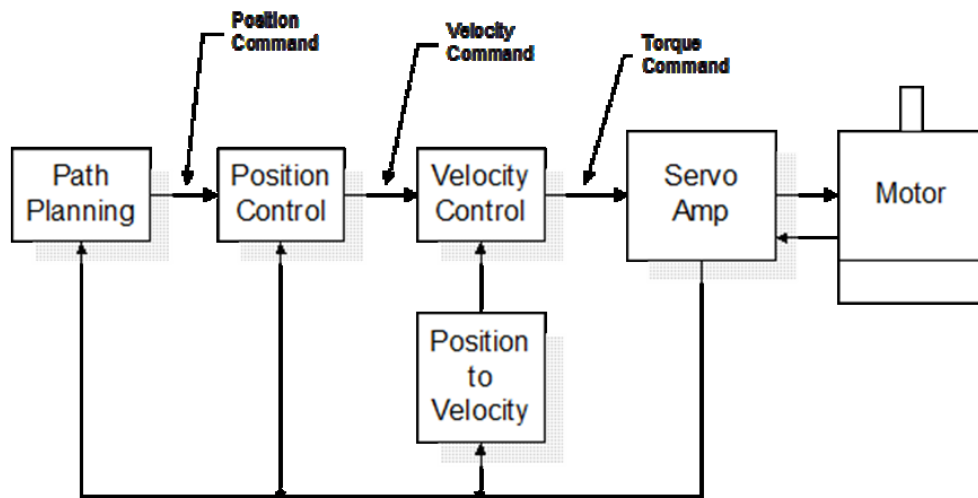
### Tuning Requirements

Three main parameters are adjusted when tuning a Torque mode servo: Load Inertia Ratio, Position Loop Time Constant and Velocity Feed Forward.

The proper method to tune the velocity loop is to separate the velocity loop from the position loop. To achieve this separation, a method must be used to directly send velocity commands without using the position loop control. The PMM module has several modes that will allow the user to send a velocity command directly to the velocity loop. Two methods are as follows.

### Analog Mode Torque Interface Control Loops Block Diagram

Figure 180: Analog Mode Torque Interface Control Loops Block Diagram



### Sending a Velocity Command to the Velocity Loop

#### Method #1

Use the Force Servo Velocity parameter (PN1311) to send a velocity command directly to the velocity loop. This command is different from the MC\_JogAxis or MC\_MoveVelocity function blocks, which use the position loop to generate the command. This is important since the position loop should not be interacting with the velocity loop at this point in the tuning process. The Force Servo Velocity parameter allows you to generate a step change in the velocity. The velocity command step is then used to generate the velocity loop step response.

## ⚠ CAUTION

Note that, when a velocity command step change is performed, the acceleration is limited only by the bandwidth of the velocity loop. In some applications, this can cause damage to the controlled device due to the high acceleration rate.

For additional information on the operation of the FSV parameter, refer to Appendix C-2 Forcing Servo Velocity.

## Method #2

In some applications, method #1 introduces too large a shock to the device under control. In these cases, another method to generate a velocity command is needed. The method requires that the user set the position loop to an open loop configuration. The position loop is set to open loop by setting the **Position Loop Time Constant** to zero and the **Velocity Feedforward Gain** to 100 percent. You can then use MC\_JogAxis or MC\_MoveVelocity function blocks to generate velocity commands to the servo drive.

## Tuning the Velocity Regulator

The following procedure tunes the velocity regulator. It is suggested that initially, this be done with the motor **not** connected to the driven load. The tuning associated with the load will be performed in a later step.

1. Choose the method to introduce velocity command to the velocity loop. Method #1 and Method #2 (above) are examples of ways to perform this task.
2. Use the Data Logger Window to display the Servo Actual Velocity parameter (PN1314) and the Torque Command parameter (PN1304).
3. The first parameter that needs to be adjusted is the Velocity Loop Proportional Gain (PN10007). The velocity loop proportional gain is multiplied by velocity error (velocity command - velocity feedback) to generate the portion of the torque command due to the proportional term. The proportional term should be set low to begin the process. Depending on the bandwidth of the controlled servo drive, the default value of 1500 may be a good starting point. However, if the servo drive has a low bandwidth or is very sensitive to changes in the torque command, the initial value may need to be set lower. The tuning procedure will allow you to iterate to get the final value. Thus, if there is any concern start with a very low value (100 for example).
4. Generate a velocity command step change. At this point the step change should be relatively small compared to the full speed of the machine. A good start is 10% to 20% of the rated machine speed.
5. Observe the Servo Actual Velocity and Torque Command data in the Data Logger Window. The objective is to obtain a critically damped velocity loop response. There will most likely be a steady state error in the velocity at this point. This is expected at this point in the tuning process. The velocity integral term will be introduced in steps that follow to cancel this error. Pay particular attention to the first peak that occurs and any oscillations that are occurring in the velocity signal.
6. Increase the **Velocity Loop Proportional** Gain in small steps and repeat steps 4 and 5 until the desired response is achieved. Depending on the application this may be a critically damped system or may have a slight overshoot. As a general rule, the lower the **Velocity Loop Proportional** Gain value that meets the system requirements the more robust the control. You should carefully observe the velocity feedback signal. In

some applications, running the **Velocity Loop Proportional** Gain high enough to create instability can cause machine damage. If oscillations are observed in the Motor Velocity feedback signal prior to this point, decrease the **Velocity Loop Proportional** Gain.

7. The next parameter to be adjusted is the **Velocity Loop Integral Gain (PN10006)**. The Velocity Loop integral gain is the term multiplied by the area of the velocity error (velocity command - velocity feedback) to generate the portion of the torque command due to the integral term. The integral gain term is typically used to compensate for steady state error in velocity.
8. Choose the method to introduce velocity command to the velocity loop. Method #1 and Method #2 (page 493) are examples of ways to perform this task.
9. Connect an oscilloscope to the analog outputs for Motor Velocity from the servo drive.
10. To begin the tuning process the **Velocity Loop Integral** Gain should be set to 0. The tuning procedure will be to slowly increase this value until steady state error is eliminated without incurring large overshoot or excessive ringing in the response.
11. Generate a velocity command step change. At this point the step change should be relatively small compared to the full speed of the machine. A good start is 10% to 20% of the rated machine speed.
12. Observe the Motor Velocity on the oscilloscope. The objective is to eliminate steady state error without introducing excessive overshoot or ringing. While tuning the integral term, pay particular attention to any oscillations that occur in the response. Excessive oscillations are an indication of instability in the control loop due to excessive integral gain.
13. Increase the Velocity Loop Integral Gain in small increments and repeat steps 11 and 12 until the desired response is achieved. Depending on the application this may be a critically damped system or may have a slight overshoot. As a general rule, the lower the Velocity Loop Integral Gain value that meets the system requirements the more robust the control. Carefully observe the velocity feedback signal. In some applications, running the Velocity Loop Integral Gain high enough to create instability can cause machine damage. If oscillations are observed in the Motor Velocity feedback signal prior to this point, decrease the Velocity Loop Integral Gain. The basic velocity loop is tuned at this point. The next step will be to connect the motor to the load and adjust the Load Inertia Ratio parameter to adjust for the motor load.
14. With the base Velocity Loop tuned, connect the motor to the load. The Load Inertia Ratio parameter (PN10032) adjusts the velocity loop response to compensate for the load. Specifically, the Load Inertia Ratio parameter adjusts the velocity loop bandwidth. As a starting point use the following formula shown below.

## Equation 2

$$\text{Load Inertia Ratio} = \frac{J_L}{J_M} \cdot 256$$

Where :

$J_L$  = Load Inertia

$J_M$  = Motor Inertia

The Load Inertia Ratio calculated above in many cases will not need to be altered. However, due to the application (for example, machine resonance) the value may need to be adjusted. To tune the Load Inertia Ratio the following procedure can be used:

15. Choose the method to introduce velocity command to the velocity loop. Method #1 and Method #2 (page 493) are examples of ways to perform this task.
16. Use the Data Logger Window to display the Servo Actual Velocity parameter (PN1314).
17. Set the Load Inertia Ratio to 0. This is a conservative approach. If the application is known to not have resonant frequencies from zero to approximately 250 Hz, you can start with a higher value, but do not exceed the value calculated in equation 1 at this point.
18. Generate a velocity command step change. At this point the step change should be relatively small compared to the full speed of the machine. A good start is 10% to 20% of the rated machine speed.
19. Observe the Servo Actual Velocity and Torque Command data in the Data Logger Window. The objective is to obtain a critically damped velocity loop response. Pay particular attention to any oscillations that are occurring in the velocity feedback signal.
20. Increase the Load Inertia Ratio in small steps and repeat steps 4 and 5 until instability in the Servo Actual Velocity feedback signal is observed. Once this point is reached, decrease the Load Inertia Ratio by at least 15%. As a general rule, the lower the Load Inertia Ratio value that meets the system requirements the more robust the control. You should carefully observe the velocity feedback signal. In some applications, running the Load Inertia Ratio high enough to create instability can cause machine damage. If in doubt, adjust the Load Inertia Ratio to be no greater than the value calculated in equation 1. If oscillations are observed in the Servo Actual Velocity feedback signal prior to this point, decrease the Load Inertia Ratio and continue with step 21 below.
21. The velocity loop is tuned at this point. However, the robustness of the loop must be checked. To perform this test, introduce velocity command steps in increments of 20% Rated Machine Speed, 40% Rated Machine Speed, 60% Rated Machine Speed, 80% Machine Rated Speed, and 100% Rated Machine Speed. Observe the Motor Velocity and Torque Command signals for any instability. If an instability or resonance is observed, reduce the Load Inertia Ratio and repeat the test.

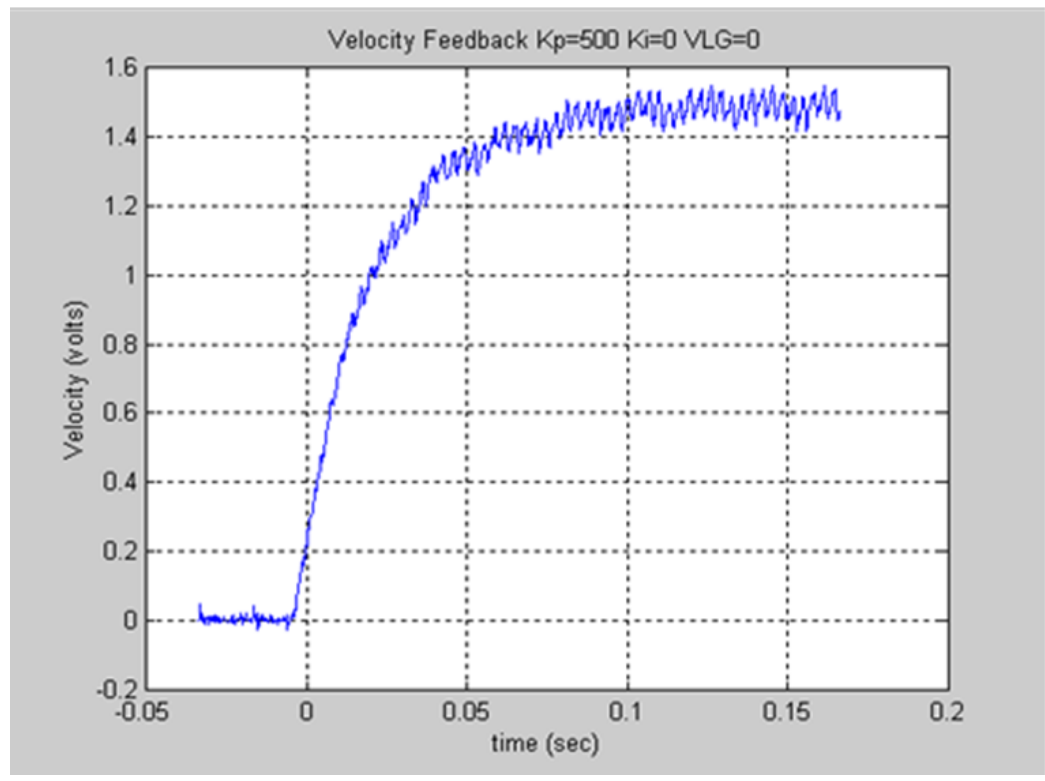
## Sample Velocity Loop Tuning Session

A sample velocity loop tuning session, which adjusts the Velocity Loop Proportional Gain is shown in the plots that follow.

### 1. Tuning the Velocity Loop Proportional Gain

In the following figure, the system has a relatively slow response. Also based on the desired velocity, there is a steady state error. In this case, the Velocity Loop Proportional Gain can be increased to help generate a faster response.

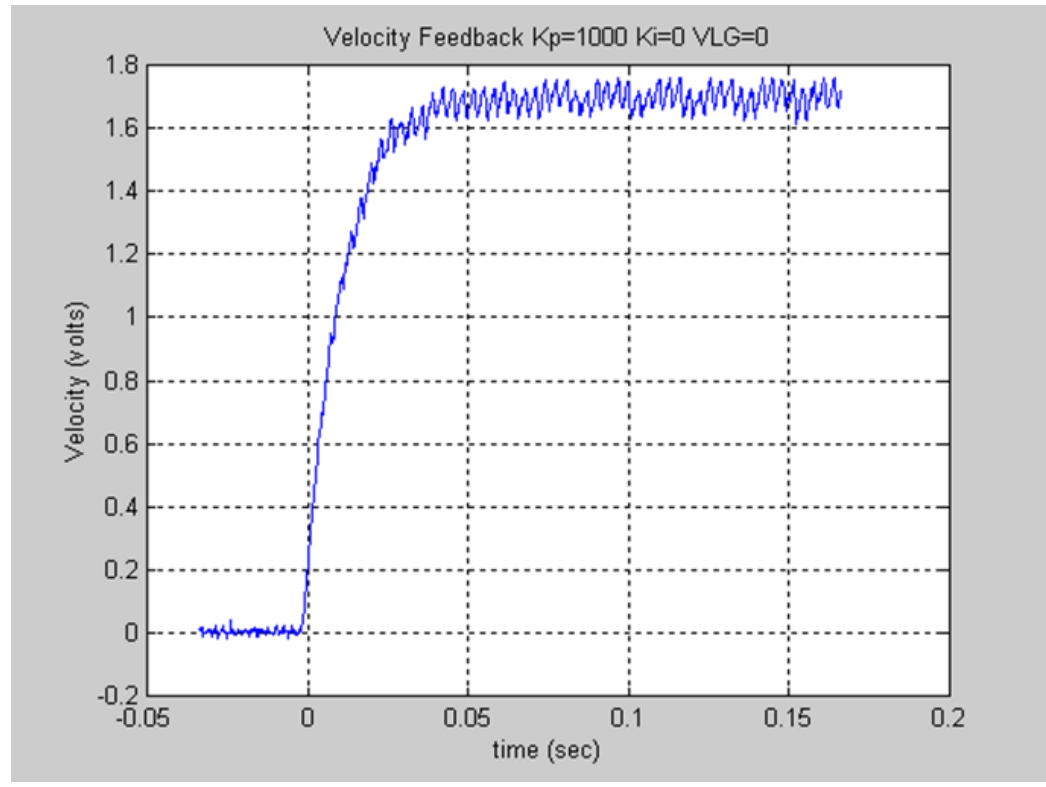
**Figure 181: Velocity Loop Step Response Velocity Feedback vs. Time (Kp=500 Ki=0 VLG=0)**





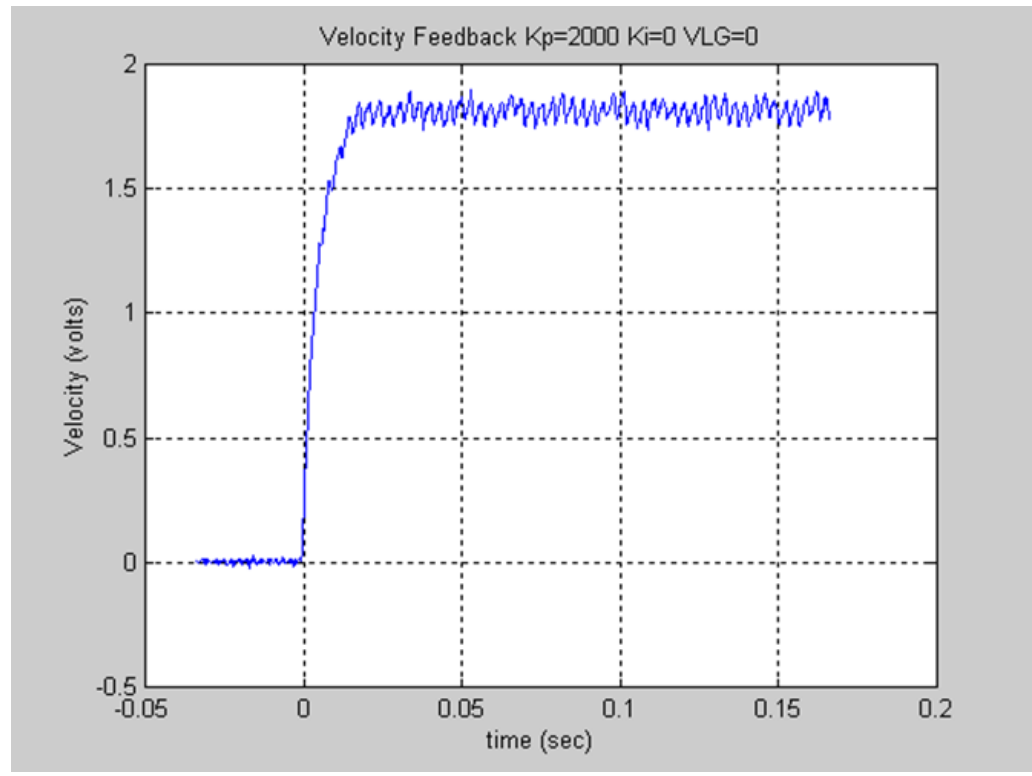
The **Velocity Loop Proportional Gain** has been increased in the figure below. The rise time has been decreased. However, the system can still be enhanced by adding additional Velocity Loop Proportional Gain. The steady state error is still present.

**Figure 182: Velocity Loop Step Response Velocity Feedback vs. Time (Kp=1000 Ki=0 VLG=0)**



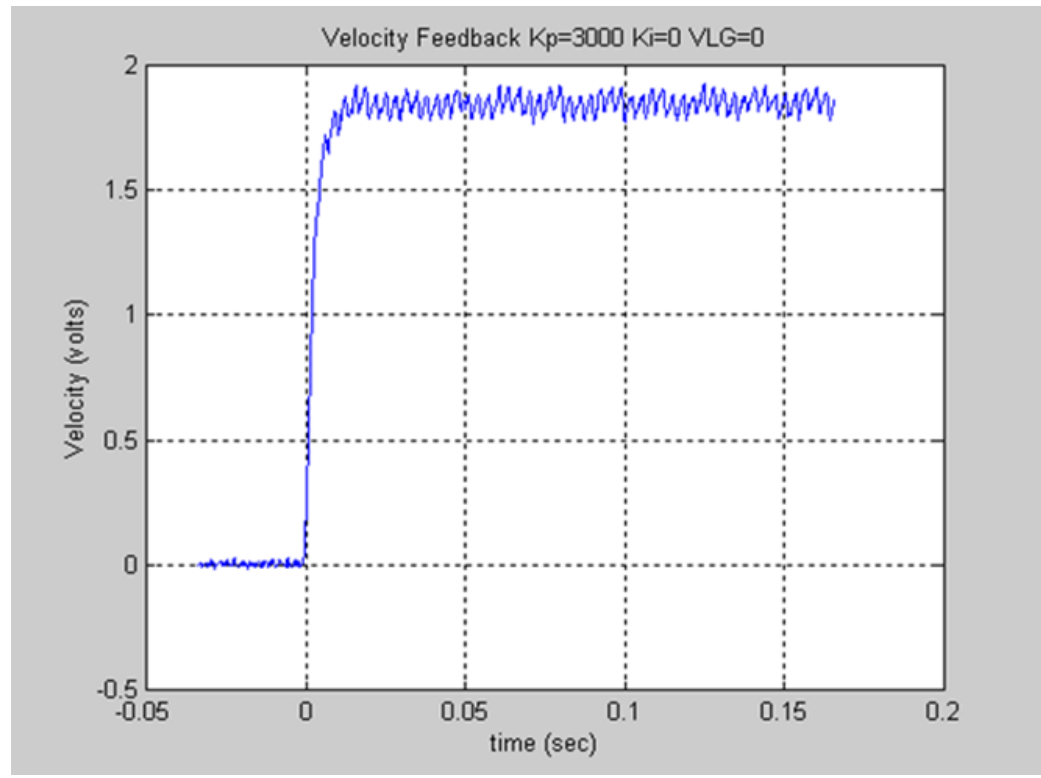
The Velocity Loop Proportional Gain has been increase again. The response shown below is starting to look very acceptable. However, the rise time can be improved further.

**Figure 183: Velocity Loop Step Response Velocity Feedback vs. Time (Kp=2000 Ki=0 VLG=0)**



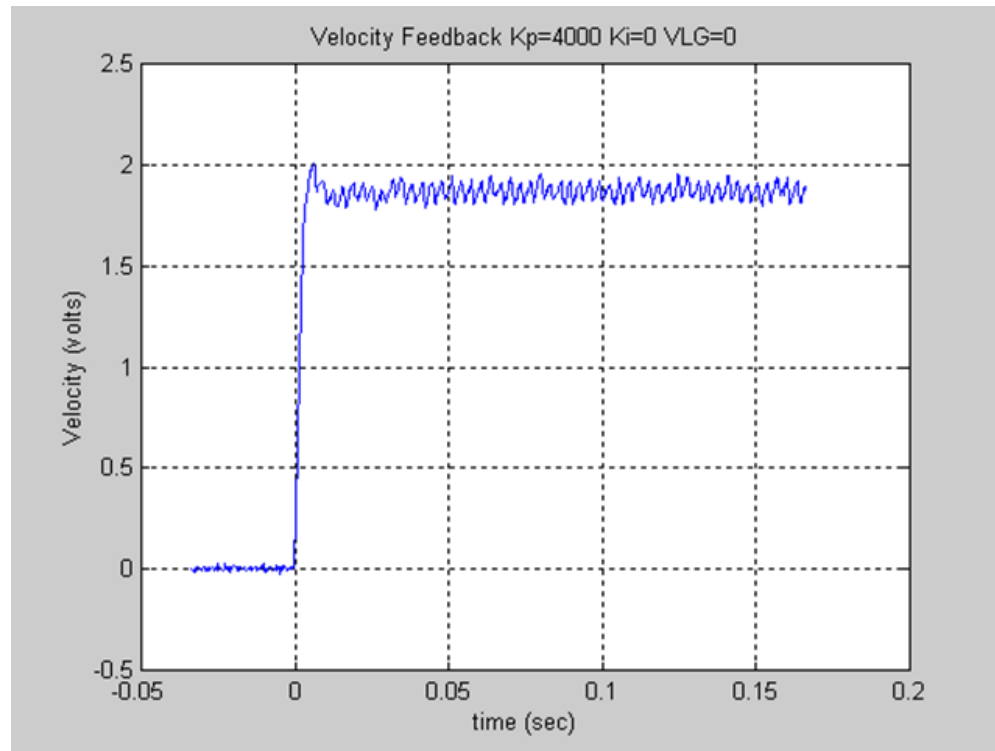
The response shown below is looking very good. Note the slight peak in the response. To experiment with the response, the Velocity Loop Proportional Gain will be increased more.

**Figure 184: Velocity Loop Step Response Velocity Feedback vs. Time (Kp=3000 Ki=0 VLG=0)**



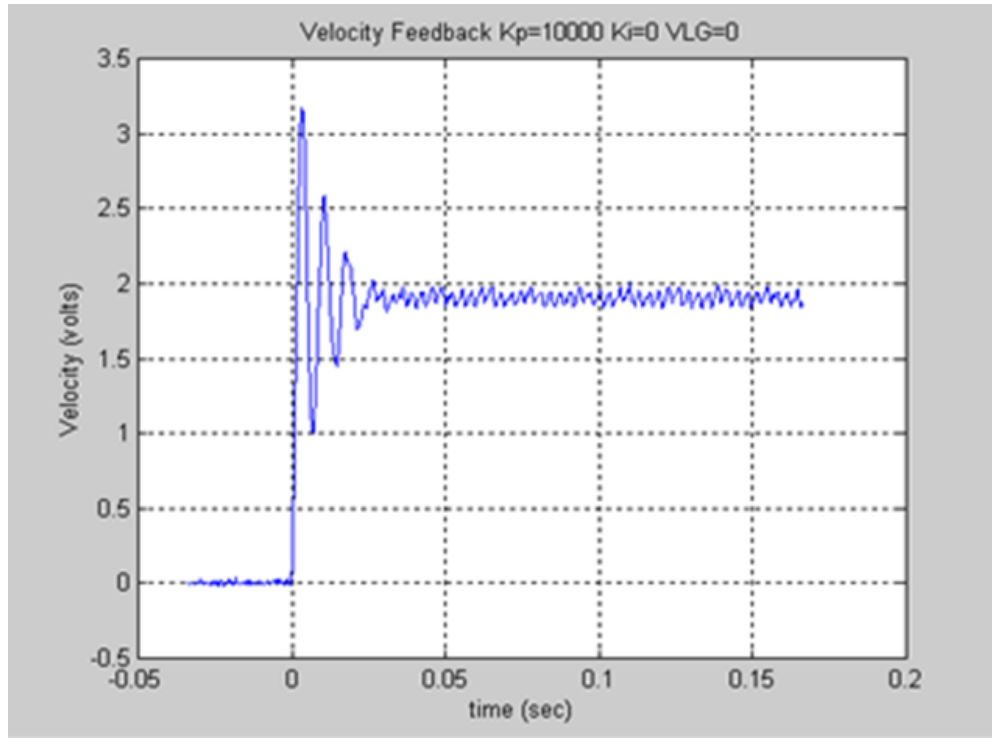
The response shown in the figure has a slight overshoot. This or the previous response would be very acceptable in many applications. However, the tuning should be determined based upon the machine abilities. The plots are shown for reference only.

**Figure 185: Velocity Loop Step Response Velocity Feedback vs. Time (Kp=4000 Ki=0 VLG=0)**



The plot shown below represents an unacceptable response. The loop is exhibiting signs of instability. Note the Overshoot and ringing following the first peak. The Velocity Loop Proportional Gain should be significantly decreased to achieve a more stable response.

**Figure 186: Velocity Loop Step Response Velocity Feedback vs. Time (Kp=10000 Ki=0 VLG=0)**



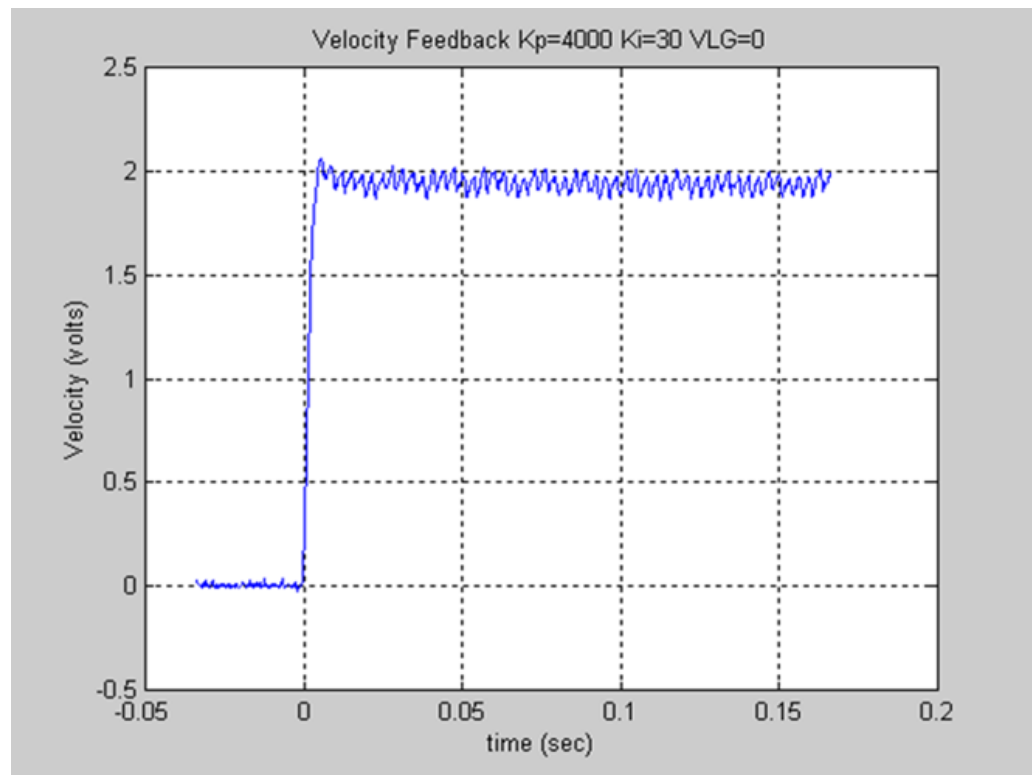
For this exercise, the response shown corresponding to the Velocity Loop Proportional Gain (Kp) =4000 will be chosen as the desired response for the system. This value will be used when tuning the Velocity Loop Integral Gain.

## 2. Tuning the Velocity Loop Integral Gain

The **Velocity Loop Integral Gain** should be set initially to 0. You can make a small change to the value and observe the response.

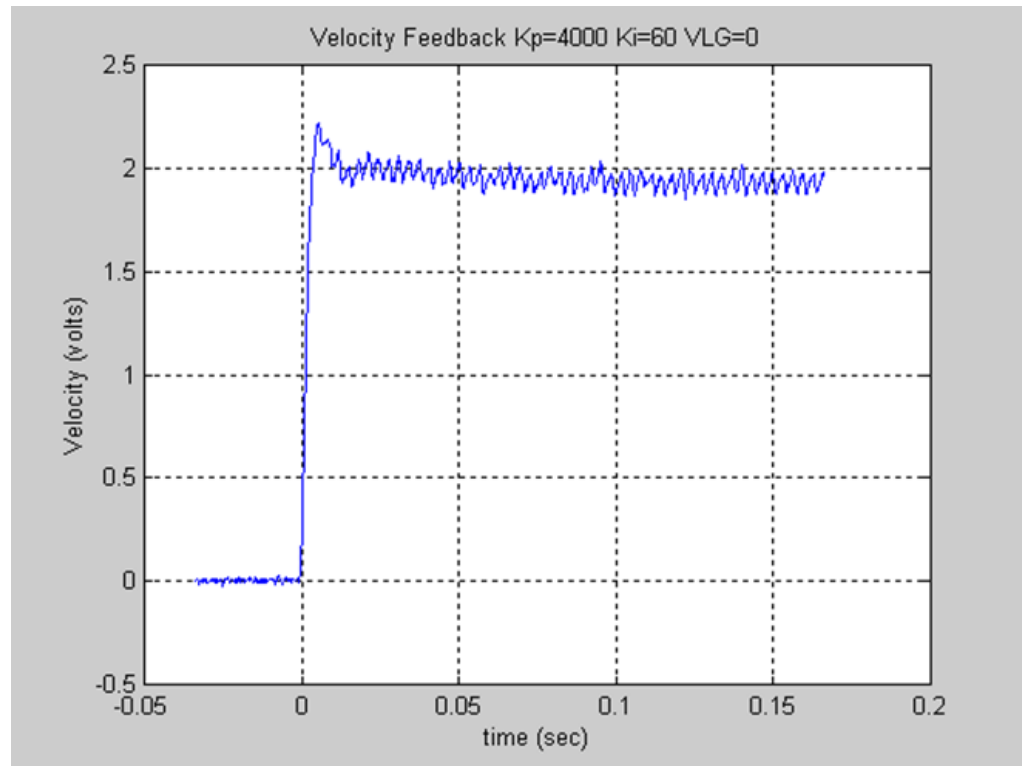
The response shown below indicates that the Velocity Loop Integral Gain has resulted in a more desirable response. Specifically, the steady state error is being reduced.

**Figure 187: Velocity Loop Step Response Velocity Feedback vs. Time (Kp=4000 Ki=30 VLG= 0)**



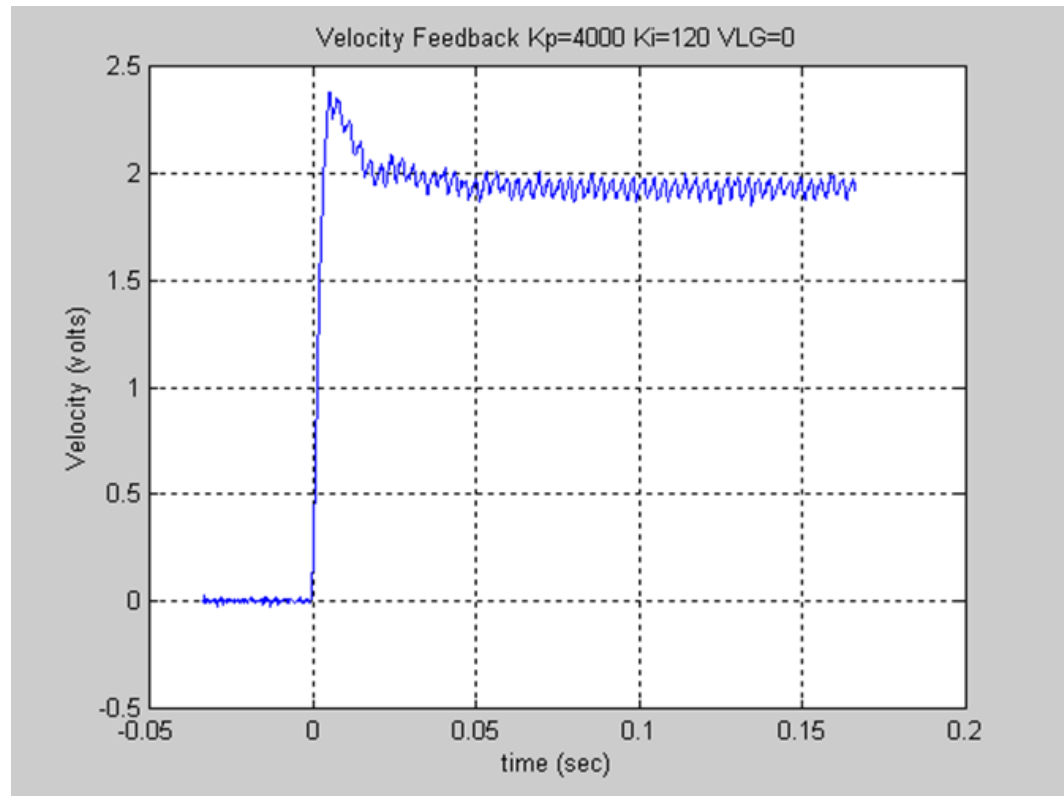
When you increase the Velocity Loop Integral Gain further, the beginning of an overshoot due to the integral gain. However, the responses in the previous figure and the following one are both acceptable. The final values chosen depend on the capabilities of the driven load. In general, the lower the Velocity Loop Proportional Gain and Velocity Loop Integral Gain that meet the system requirements, the more robust the control.

**Figure 188: Velocity Loop Step Response Velocity Feedback vs. Time (Kp=4000 Ki=60 VLG=0)**



The response shown below illustrates too much Velocity Loop Integral Gain and in most applications, this would be considered unacceptable.

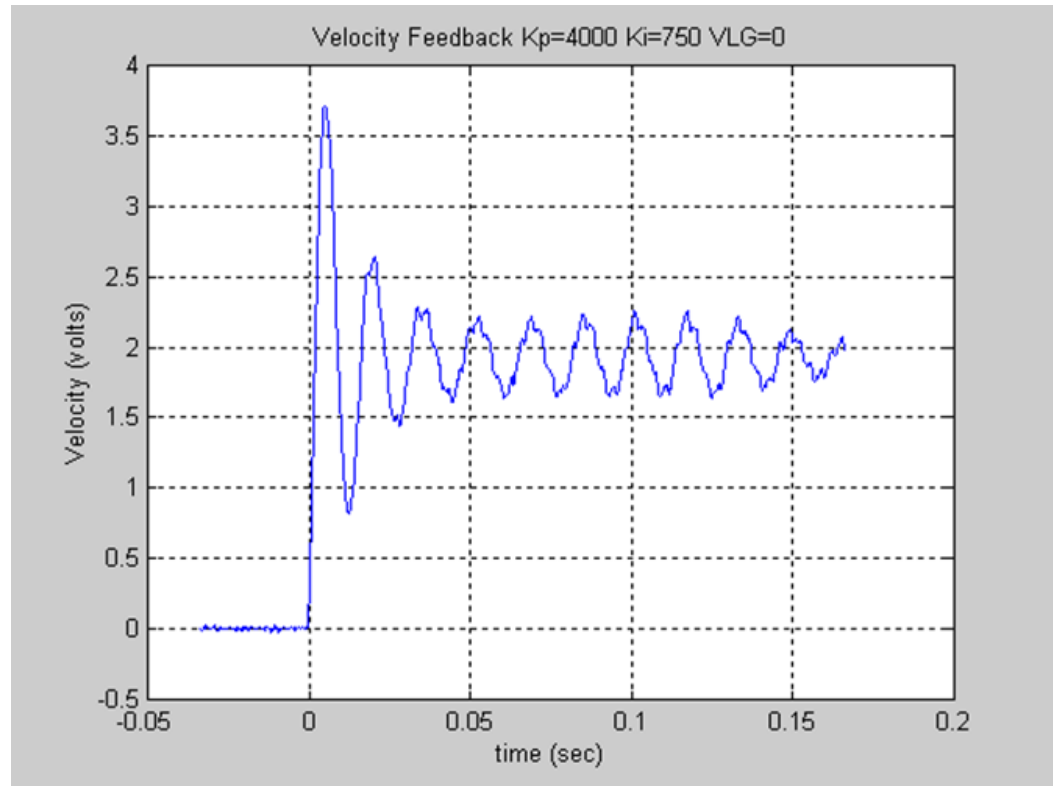
**Figure 189: Velocity Loop Step Response Velocity Feedback vs. Time (Kp=4000 Ki=120 VLG=0)**





The result shown below represents a marginally stable system. In this response, there is not only a significant overshoot, but also a ringing in the velocity response that is slowly being damped out. The response is unacceptable.

**Figure 190: Velocity Loop Step Response Velocity Feedback vs. Time (Kp=4000 Ki=7500 VLG= 0)**

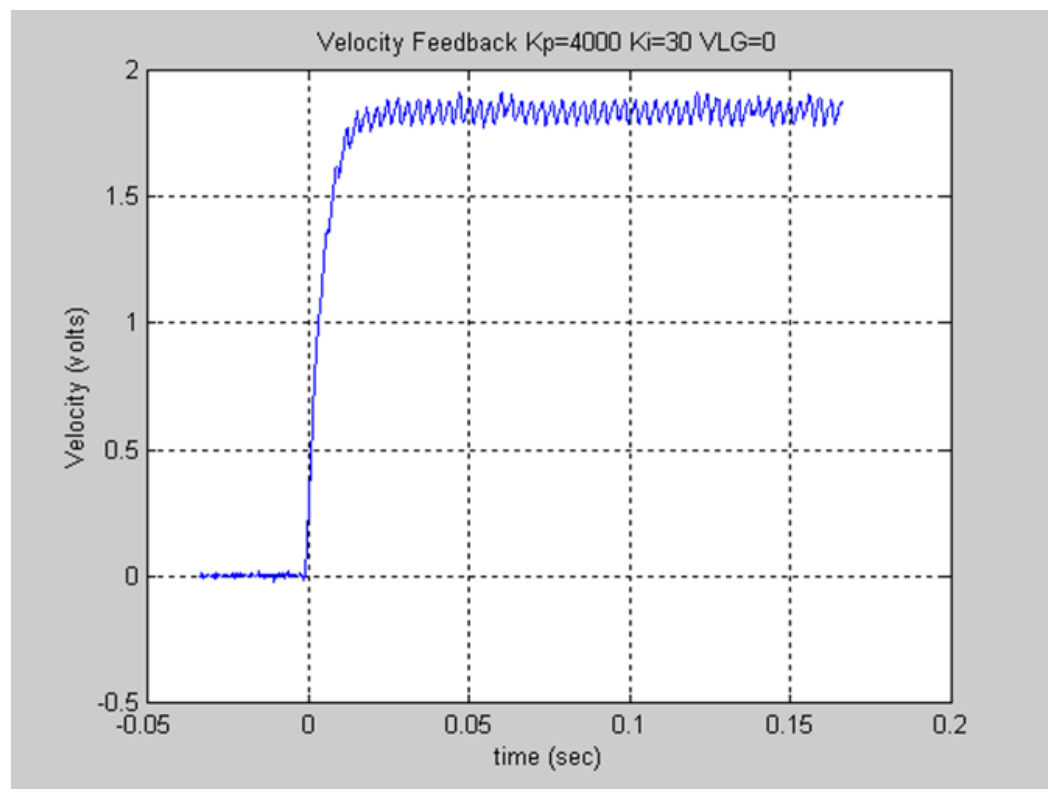


### 3. Tuning the Load Inertia Ratio

The next step in the tuning process is to connect the motor to the load and then adjust the control to achieve the desired performance. The Load Inertia Ratio parameter allows you to adjust the controller parameters to account for the motor load. As in the procedure above, start with the Load Inertia Ratio equal to zero.

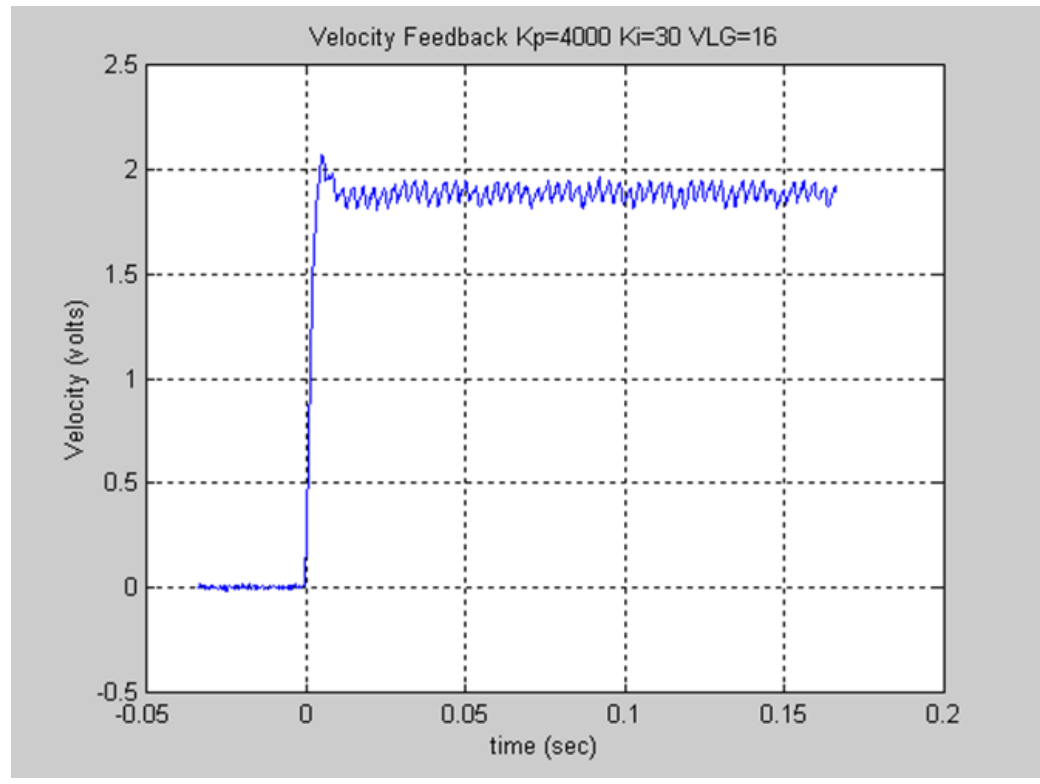
The figure below shows the motor velocity response with a load connected to the motor and the motor tuned per the exercise above. The performance is acceptable, but by increasing the Load Inertia Ratio the rise time can be decreased.

**Figure 191: Velocity Loop Step Response Velocity Feedback vs. Time (Kp=4000 Ki=30 VLG= 0)**



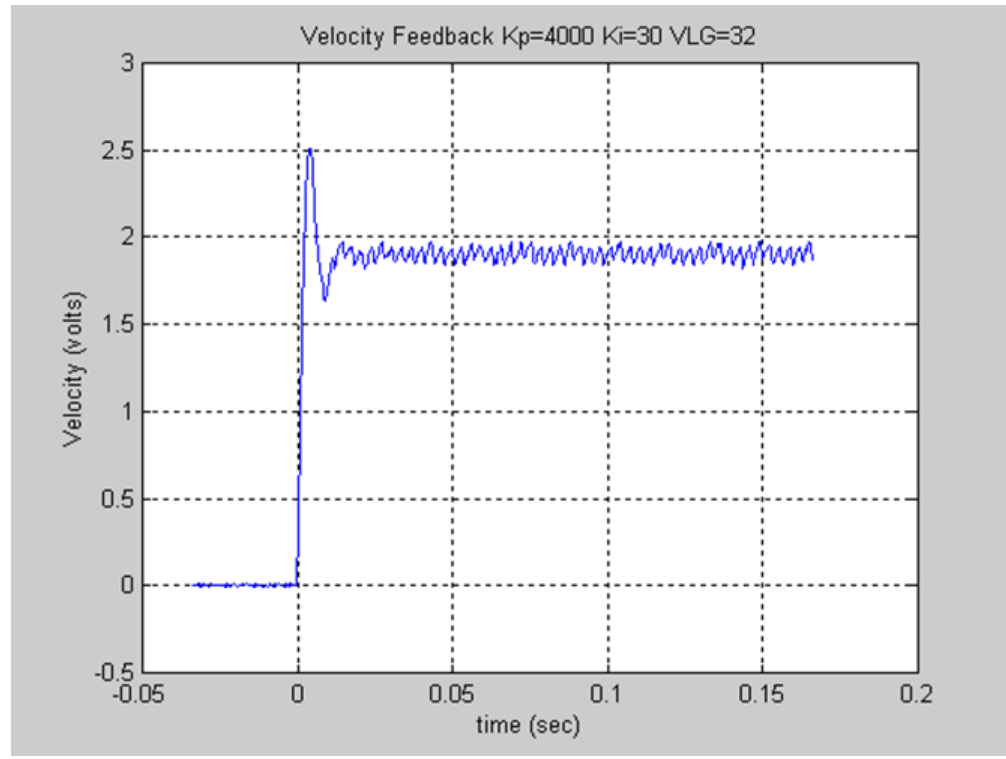
The response shown below is acceptable. The response has a slight overshoot but no sustained oscillation or ringing.

**Figure 192: Velocity Loop Step Response Velocity Feedback vs. Time (Kp=4000 Ki=30 VLG=16)**



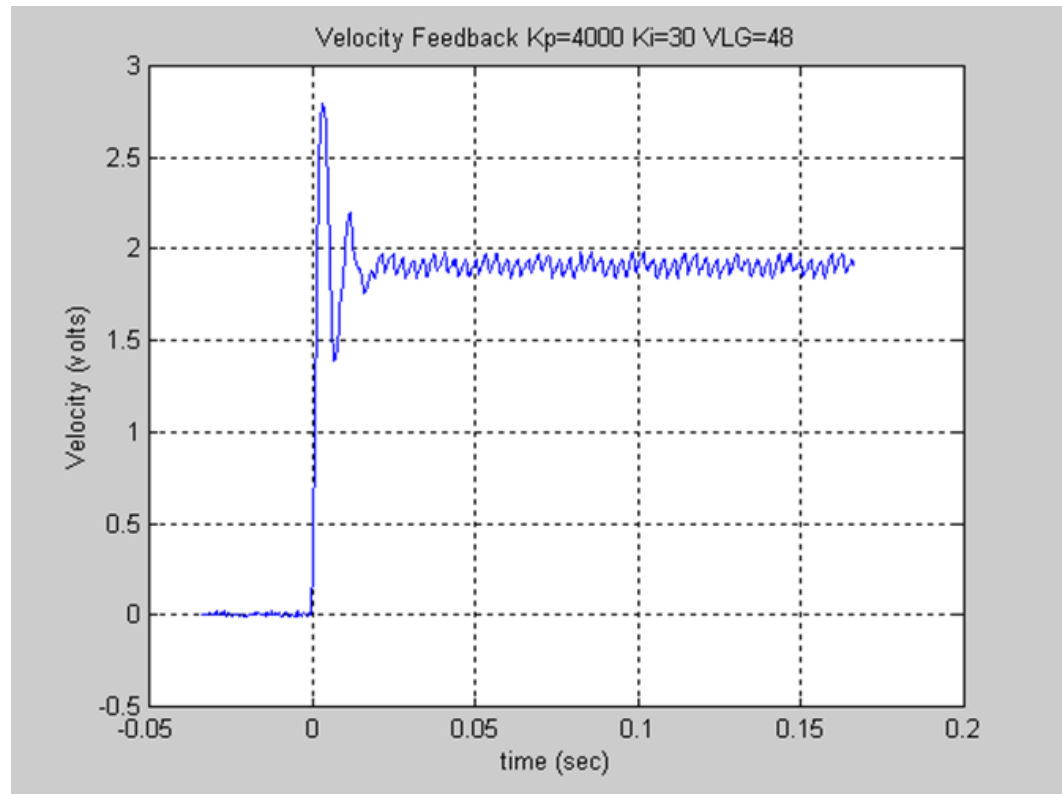
The response shown below has a rather large overshoot, however there are no adverse effects beyond the initial overshoot and oscillation. The overshoot indicates that the user may wish to reduce the Velocity Loop Gain.

**Figure 193: Velocity Loop Step Response Velocity Feedback vs. Time (Kp=4000 Ki=30 VLG = 32)**



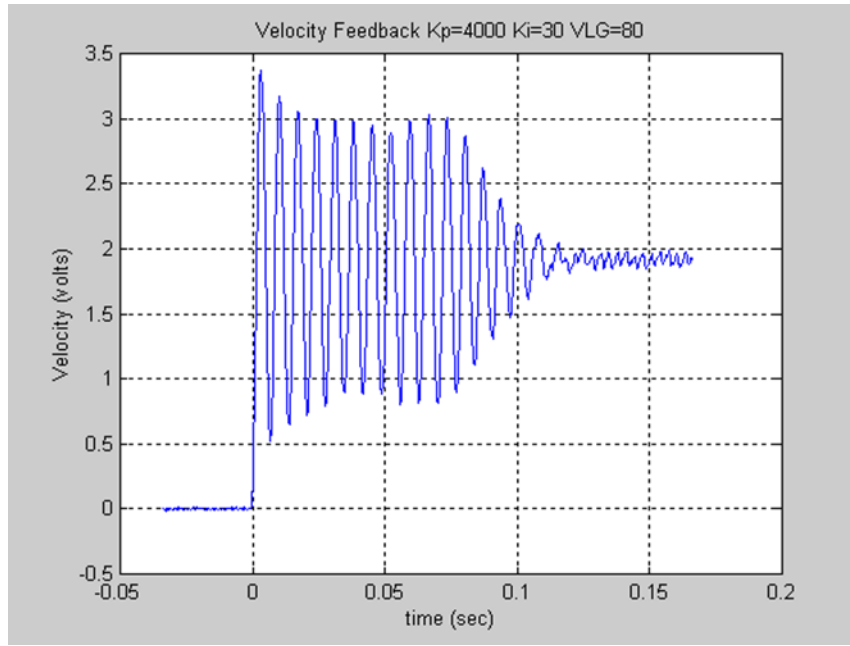
The response shown below exhibits an overshoot and notable ringing in the response. This response is starting to indicate that the Load Inertia Ratio is greater than necessary.

**Figure 194: Velocity Loop Step Response Velocity Feedback vs. Time (Kp=4000 Ki=30 VLG=48)**



The response shown below represents a marginally stable system. The Load Inertia Ratio is significantly too large. Notice the significant overshoot and sustained ringing in the response. This response would not be acceptable.

**Figure 195: Velocity Loop Step Response Velocity Feedback vs. Time (Kp=4000 Ki=30 VLG= 80)**



## C-5.4 Position Loop Tuning

Once the velocity regulators have been tuned, the position loop tuning and setup can be completed. Initially set the Pos Loop Time Constant configuration parameter to a high value (typically 100ms in the configuration).

Use the MC\_JogAxis function block to move the axis. Confirm that the servo moves in the proper direction and that the Actual Velocity reported by the PMM matches the JogVelocity.

Check for proper operation of the Find Home cycle by executing an MC\_Home function block on the axis. The axis should move toward the Home Switch at the Find Home Velocity, then seek the Encoder Marker at the configured Final Home Velocity. If necessary, adjust the configured velocities and the location of the Home Switch for consistent operation. The final Home Switch transition MUST occur at least 10 ms before the Encoder Marker Pulse is encountered. The physical location of Home Position can then be adjusted by changing the Home Offset input to the MC\_Home function block.

If necessary, adjust the configured velocities and the location of the Home Switch for consistent operation. The Home Switch must be mounted so that the final Home Switch off to on transition occurs at least 10ms before the encoder reference point is encountered.

Monitor servo performance and use the MC\_JogAxis function block to move the analog servo motor in each direction. The Position Loop Time Constant can be temporarily modified by writing a value to PN1009. For most systems, the Position Loop Time Constant can be reduced until some servo instability is noted, then increased to a value approximately 50% higher. Once the correct time constant is determined, the PMM configuration should be updated using the configuration software. Velocity Feedforward can also be set to a non-zero value (typically 90 to 100%) for optimum servo response.

---

**Note:** For proper servo operation, the Configuration entry for **Max Velocity System** must be set to the maximum servo velocity that the system or process allows.

---

## C-5.5 Advanced Analog Servo (Torque Mode) Tuning

Advanced Servo Tuning makes use of the digital servo parameters listed in the following table. These parameters can be set in the hardware configuration Advanced Tab (refer to Section 4.3.6, Advanced Parameters or using the MC\_WriteParameter command (refer to Section 6.55).

### Digital Servo Parameters

Servo Parameter	Min. Value	Max. Value	Description
10006	0	32767	PK1V, Velocity Loop Gain 1 (Integral)
10007	-32768	0	PK2V, Velocity Loop Gain 2 (Proportional)
10008	0	32767	PK3V, Velocity Loop Gain 3 (Integral Decay)
10031	0	2810	TCMD Torque Command Filter

The following simplified block diagram of the digital servo velocity loop shows these elements:

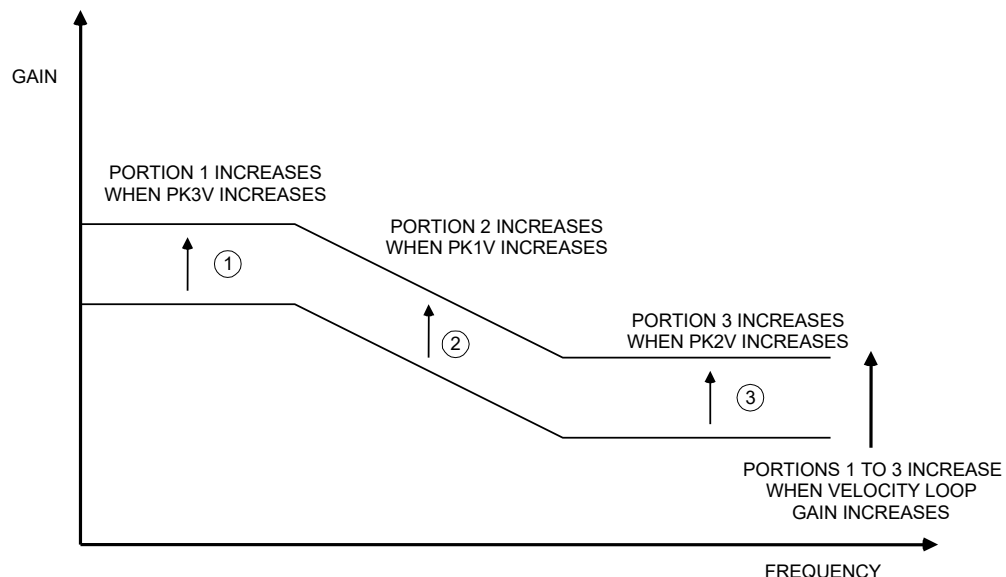
- Velocity loop gain parameters PK1V, PK2V, PK3V
- Torque Command Filter
- Resonance Filters

### Velocity Loop Gain Parameters PK1V, PK2V, PK3V

These parameters set the gain of the velocity loop in three frequency bands.

### Frequency Ranges for PK1V-PK3V

**Figure 196: Frequency Ranges for PK1V, PK2V & PK3V**





- **PK1V** (Integral Gain) adjusts mid-frequency velocity loop gain
- **PK2V** (Proportional Gain) adjusts high-frequency velocity loop gain
- **PK3V** (Integral Gain Decay) adjusts low-frequency velocity loop gain

Setting the proportional and integral gains is discussed above. Once the base values for these are determined the Load Inertia Ratio parameter can be used to adjust for different motor loads

**Note:** The Load Inertia Ratio setting changes velocity loop gain over the entire low to high frequency range. For information on setting Load Inertia Ratio.

Under some conditions PK3V (Integral Gain Decay) can be changed to prevent the servo from jumping when the servo torque limit has been temporarily set to zero, then set back to a non-zero value. The time constant *Tau* of integral decay is set according to this equation:

$$\text{Tau (seconds)} = 1.0 \text{ ms} / \ln (32768/\text{PK3V})$$

where ln = natural logarithm

The default value of PK3V is 0. This means that the integral gain does not decay, therefore PK1V will set both the low and mid frequency gains of the velocity loop. If PK3V is changed, typical values are in the range 25000 to 32760. The following table lists the integral decay time constant for several PK3V values.

### Integral Gain Decay Time Constants

PK3V	Integral Decay Time Constant
32760	4.09 seconds
32740	1.17 seconds
32700	0.481 seconds
32600	0.195 seconds
32000	42 ms
30000	11 ms
25000	3.7 ms

### Torque Command Filter

As shown in the table below, the TCMD filter is a low pass filter in series with the torque command to the servo torque loop. Under some conditions the TCMD filter can be used to reduce mid-frequency machine resonances at 100 HZ or greater. Usually the Resonance Elimination Filters, discussed below, will provide better performance than the TCMD filter. If the TCMD filter is used, a suggested setting for the cutoff frequency is ½ the frequency of the machine resonance.

The default value for the TCMD FILTER is 0, which means that the filter is disabled. The following table lists the TCMD filter cutoff frequency for representative values of the FILTER parameter. Values greater than 2400 are not recommended because they reduce the velocity loop gain at such a low frequency that machine resonances may increase rather than decrease.

### TCMD Filter Cutoff Frequencies

Filter Parameter	Cutoff Frequency (Hz)	Filter Parameter	Cutoff Frequency (Hz)
2810 **	60	1700	140
2723 **	65	1596	150
2638 **	70	1499	160
2557 **	75	1408	170
2478 **	80	1322	180
2401 **	85	1241	190
2327	90	1166	200
2255	95	1028	220
2185	100	907	240
2052	110	800	260
1927	120	705	280
1810	130	622	300

# General Contact Information

Home link: <http://www.emerson.com/industrial-automation-controls>

Knowledge Base: <https://www.emerson.com/industrial-automation-controls/support>

## Technical Support

### Americas

Phone: 1-888-565-4155

1-434-214-8532 (If toll free option is unavailable)

Customer Care (Quotes/Orders>Returns): [customercare.mas@emerson.com](mailto:customercare.mas@emerson.com)

Technical Support: [support.mas@emerson.com](mailto:support.mas@emerson.com)

### Europe

Phone: +800-4444-8001

+420-225-379-328 (If toll free option is unavailable)

Customer Care (Quotes/Orders>Returns): [customercare.emea.mas@emerson.com](mailto:customercare.emea.mas@emerson.com)

Technical Support: [support.mas.emea@emerson.com](mailto:support.mas.emea@emerson.com)

### Asia

Phone: +86-400-842-8599

+65-6955-9413 (All other Countries)

Customer Care (Quotes/Orders>Returns): [customercare.cn.mas@emerson.com](mailto:customercare.cn.mas@emerson.com)

Technical Support: [support.mas.apac@emerson.com](mailto:support.mas.apac@emerson.com)

Any escalation request should be sent to: [mas.sfdcescalation@emerson.com](mailto:mas.sfdcescalation@emerson.com)

**Note:** If the product is purchased through an Authorized Channel Partner, please contact the seller directly for any support.

Emerson reserves the right to modify or improve the designs or specifications of the products mentioned in this manual at any time without notice. Emerson does not assume responsibility for the selection, use or maintenance of any product. Responsibility for proper selection, use and maintenance of any Emerson product remains solely with the purchaser.

© 2020 Emerson. All rights reserved.

Emerson Terms and Conditions of Sale are available upon request. The Emerson logo is a trademark and service mark of Emerson Electric Co. All other marks are the property of their respective owners.

