GE Fanuc Intelligent Platforms

PACMotion™

Multi-Axis Motion Controller

User's Manual, GFK-2448A

June 2009

Warnings, Cautions, and 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.

This document is based on information available at the time of its publication. While efforts have been made to be accurate, the information contained herein does not purport to cover all details or variations in hardware or software, nor to provide for every possible contingency in connection with installation, operation, or maintenance. Features may be described herein which are not present in all hardware and software systems. GE Fanuc assumes no obligation of notice to holders of this document with respect to changes subsequently made.

GE Fanuc makes no representation or warranty, expressed, implied, or statutory with respect to, and assumes no responsibility for the accuracy, completeness, sufficiency, or usefulness of the information contained herein. No warranties of merchantability or fitness for purpose shall apply.

The following are trademarks of GE Fanuc Intelligent Platforms:

Alarm Master	Helpmate	ProLoop	Series Six
CIMPLICITY	Logicmaster	PROMACRO	Series Three
CIMPLICITY 90–ADS	Modelmaster	PowerMotion	VersaMax
CIMSTAR	Motion Mate	PowerTRAC	VersaPoint
Field Control	PACMotion	Series 90	VersaPro
GEnet	PACSystems	Series Five	VuMaster
Genius	Proficy	Series One	Workmaster

©Copyright 2008—2009 GE Fanuc Intelligent Platforms
All Rights Reserved

Contents

Introduction 1-1

PACMotion Multi-Axis Motion Controller 1-2
Fiber I/O Terminal Block 1-4
βi, βHVi, and αHVi Series FSSB Servos 1-5
Related Information 1-7
Getting Help 1-7

Getting Started 2-1

Motion System Overview 2-2
PACMotion Programming- Machine Edition 2-2
PMM/RX3i Interface 2-3
Fiber I/O Terminal Block 2-4
Servo Amplifier and Machine Interfaces 2-5
Human-Machine Interfaces 2-5
Basic Installation 2-6
PMM Installation 2-6
FTB Installation 2-8
Basic Configuration 2-12
Connecting the Programmer to the RX3i 2-12
Adding a Motion Controller Module to the Hardware Configuration 2-13
Configuring PMM Settings 2-14
Configuring PMM I/O 2-15
Configuring Axis Parameters 2-16
Motor Type Codes 2-17
Downloading the Configuration to the RX3i 2-18
Axis Powerup and Feedback Device Initialization 2-19
Basic Motion Example – Jogging an Axis 2-20
Using the Synthetic Motor 2-22

Wiring and Connections 3-1

I/O Wiring, Connections, and LEDs 3-2
PMM Faceplate I/O 3-2
Fiber I/O Terminal Block I/O 3-7
Errors Indicated by LEDs 3-13
I/O Circuit Specifications 3-15
Faceplate I/O Circuits 3-15
FTB I/O Circuits 3-18
FSSB Connections 3-25
FSSB Fiber Optic Servo Command Interface Cable 3-25
FTB to PMM Connection 3-26
Grounding the PACMotion System 3-27
Shield Ground Connection 3-28
I/O Cable Grounding 3-29

Configuration 4-1

Connecting the Programmer to the RX3i 4-2
Adding a PMM to the Hardware Configuration 4-3
Configuring PMM Parameters 4-4
Settings 4-5
I/O Function Assignments 4-8
I/O Interrupts 4-18
Axis Configuration Data 4-20
Advanced Data 4-52
Power Consumption 4-52
Terminals 4-52

PACMotion Function Block Operation 5-1

PACMotion Function and Function Block Types 5-2
Behavior of Motion Instructions 5-4
Instance Data 5-4
Immediate Response vs. Deferred Response Function Blocks 5-5
Administrative vs. Motion-Generating Functions and Function Blocks 5-5
Function Block Triggering (Enabled vs. Executed Instructions) 5-6
Function and Function Block Parameters 5-14
Permissives, Constants, Variables or Flow Used With Motion Function Blocks 5-14
EN Input and ENO Output 5-14
Input Parameters 5-15
Reference ID Variables 5-18
Output Parameters 5-20
In_Out Parameters 5-23
Data Types and Structures 5-24
HWC Parameter Linked Data Types 5-24
Enumerated Data Types 5-25
Cam Profile Linked Data Types 5-26
Axis States 5-27
Axis State Diagram 5-29
Synchronized Motion 5-30

PACMotion Instruction Set Reference 6-1

MC_AbortTrigger 6-2
MC_CamFileRead 6-4
MC_CamFileWrite 6-6
MC_CamIn 6-8
Start Mode Mask 6-11
Offset and Scaling 6-12
MC_CamOut 6-16
MC_CamTableDeselect 6-18

MC_CamTableSelect ... 6-20
Periodic (Cam Cycle Execution Mode) ... 6-21
MC_DelayedStart ... 6-22
MC_DigitalCamSwitch ... 6-24
Requirements for Switch and Track Option Selections ... 6-26
Enabling Outputs for DCS Control ... 6-27
MC_DL_Activate ... 6-35
MC_DL_Configure ... 6-37
MC_DL_Delete ... 6-42
MC_DL_Get ... 6-44
Data Logging Example ... 6-45
MC_GearIn ... 6-56
MC_GearIn Example ... 6-59
MC_GearInPos ... 6-61
MC_GearOut ... 6-65
MC_Halt ... 6-67
MC_Home ... 6-69
Homing Modes ... 6-72
MC_JogAxis ... 6-76
Example: Jog to Software End of Travel with Warning ... 6-79
MC_LibraryStatus ... 6-83
MC_ModuleReset ... 6-85
MC_MoveAbsolute ... 6-87
Example ... 6-90
MC_MoveAdditive ... 6-91
Example ... 6-94
MC_MoveRelative ... 6-95
MC_MoveSuperimposed ... 6-97
MC_MoveVelocity ... 6-99
MC_Phasing ... 6-101
MC_Power ... 6-103
MC_Power Input Combinations ... 6-105
Example ... 6-105
MC_ReadActualPosition ... 6-106
MC_ReadActualVelocity ... 6-107
MC_ReadAnalogInput ... 6-108
Input Example ... 6-109
MC_ReadAnalogOutput ... 6-110
MC_ReadAxisError ... 6-112
MC_ReadBoolParameter ... 6-114
Example ... 6-115
MC_ReadBoolParameters ... 6-116
Example ... 6-117

MC_ReadDigitalInput ... 6-118
Example ... 6-119
MC_ReadDigitalOutput ... 6-120
MC_ReadDwordParameters ... 6-122
MC_ReadEventQueue ... 6-124
Example ... 6-126
MC_ReadParameter ... 6-129
Example ... 6-130
MC_ReadParameters ... 6-131
Example ... 6-132
MC_ReadStatus ... 6-133
Axis Status Flags ... 6-135
MC_ReadTorqueCommand ... 6-138
MC_Reset ... 6-139
MC_SetOverride ... 6-141
MC_SetPosition ... 6-144
MC_Stop ... 6-146
MC_SyncStart ... 6-148
MC_TouchProbe ... 6-150
Examples ... 6-152
MC_WriteAnalogOutput ... 6-156
Example ... 6-157
MC_WriteBoolParameter ... 6-158
MC_WriteBoolParameters ... 6-160
MC_WriteDigitalOutput ... 6-162
Example ... 6-164
MC_WriteDwordParameters ... 6-165
MC_WriteParameter ... 6-167
MC_WriteParameters ... 6-169

Electronic Cam Programming ... 7-1

Overview of PACMotion Cam Profile Development ... 7-1
Point Limits in Cam Profiles ... 7-2
Cam Types and Modes for the PMM ... 7-3
Cam Profile Types ... 7-4
Cam Operation Restrictions by Type and Mode ... 7-6
Smoothing and Curve Fitting ... 7-7
Synchronized Motion Function Block Status ... 7-10
CSV Cam File Format ... 7-14
Reference Memory Format for Cam Files ... 7-16
Parameters of type LREAL 8 bytes. ... 7-16
Boolean Parameters ... 7-17

Parameters for Monitoring and Control 8-1

Parameter Number Index 8-2
Axis Parameter Numbers 8-2
Module Parameter Numbers 8-16
I/O Data Reference Numbers 8-21

Diagnostics 9-1

PMM Error IDs 9-2
Accessing Error IDs 9-2
Error ID Format 9-2
Clearing PMM Errors 9-3
Error ID Reference 9-3
CPU Error Codes 9-50
FSSB Servo Alarms 9-55
Digital Servo Alarm Error Codes 9-55
Diagnostics for FSSB Servo Alarms 9-56
PMM Event Queue 9-60
Event Queue Details 9-60
Parameter Errors Caused by Changes in Axis Scaling 9-62
Diagnostic Logic Blocks 9-65
Using DLBs with PACMotion Cam Profiles 9-65

Touch Probe and Digital Cam Switch Accuracy Calculations A-1

Touch Probe Accuracy A-1
External Quadrature Encoder A-1
Motor Encoder A-1
Digital Cam Switch Accuracy A-5
DCS Accuracy at Constant Velocity A-5
DCS Accuracy During Acceleration A-6

Position Feedback Devices B-1

Digital Serial Encoder Resolutions B-1
Digital Serial Encoder Modes B-2
Incremental Encoder Mode Considerations B-2
Absolute Encoder Mode Considerations B-2
External Quadrature Encoder B-5

Startup and Tuning of a FSSB Servo Drive C-1

Validating Axis Direction, Over Travel Switch and Home Switch Inputs C-2
Validating Motor Direction C-2
Validating the Over Travel Limit Switch Inputs C-3
Validating the Home Switch Input C-4
FSSB Servo System Startup Diagnostics C-5

Tuning a FANUC FSSB Servo Drive C-6
Tuning Requirements C-6
Advanced Servo Tuning C-17

Chapter 1 Introduction

The PACMotion Multi-axis Motion Controller (PMM) is a high performance, easy-to-use servo motion control module that is closely integrated with the PACSystems™ RX3i CPU's logic solving 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 PMM's built-in faceplate I/O supports applications with lower I/O counts, while the optional Fiber I/O Terminal Block (FTB) supports larger applications with extensive distributed user configurable digital and analog I/O.

This chapter provides an overview of PMM and FTB features.

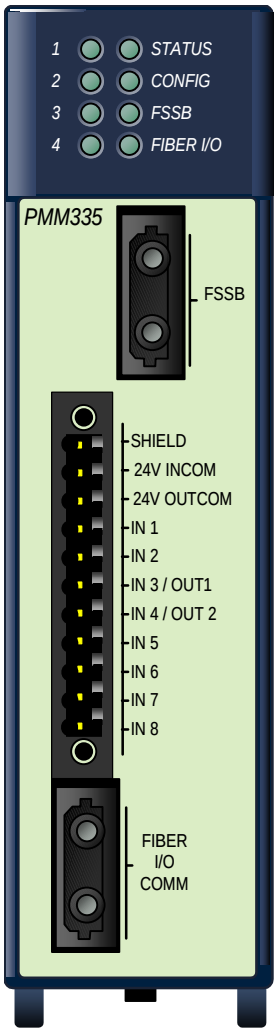
PACMotion Multi-Axis Motion Controller

The PACMotion Multi-Axis Motion Controller (IC695PMM335) 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 PMM335 supports discrete and synchronous motion control yielding a flexible motion controller that scales to fit your requirements.

The PMM335 supports FANUC β*i*, βHV*i* and αHV*i* Series digital servo amplifiers and motors. For a list of supported motors, refer to "Motor Type Codes" in Chapter 2.



PMM335 Features

- Fast motion path (1ms) planning and position update rates (500µs) deliver improved accuracy and faster response to changing control requirements
- Consistent motion update rate regardless of the number of axes
- High reliability FANUC servos improve machine uptime
- High speed synchronization of up to 40 axis 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–up to 100 meters between axes using noise immune fiber cables
- 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

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 may significantly reduce scan time impact
- Digital Cam Switch (PLS) function with multi-track high-speed outputs with microsecond level response
- Reduced downtime with industry leading FANUC servos featuring MTBF ratings in excess of 400,000 hours
- Low MTTR FANUC servos require no tuning or parameter setting

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 Proficy software speed diagnostics and machine time-to-market

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
- Amplifiers and motion I/O can be physically distributed using noise immune fiber optic interfaces
- Virtual (time-based) or real (encoder) master axes over the backplane support advanced cam and electronic gearing applications for flexible electronic line shaft applications

Fiber I/O Terminal Block

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 PMM335. 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.

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.

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 24V outputs.
- Loss of encoder and open wire fault detection on 5V 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.

βi, βHVi, and αHVi Series FSSB Servos

The PMM335 supports the FANUC βis series (230 VAC), βHVis (460 VAC), αHVi and αHVis series servos, and their associated amplifiers. For a list of servo and amplifier models, refer to the *Servo Product Specifications Guide,* GFH-001.

The FANUC 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 (optional battery kit required)
- 24VDC motor holding brake (optional)
- Optional IP67 environmental rating available for most motors (special order)

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. 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 amplifiers through the Fanuc Servo Serial Bus (FSSB). The FSSB interface utilizes the standard FANUC servo communication protocol.

βis and βHVis Series Servo Motors

The β*i*s and βHV*i*s Series Servomotors are built on the superior technology of the α*i*s Series servos. The β*i*s and βHV*is* Series motors conform to international standards (IEC).

A high-resolution absolute encoder is standard with each β*i*s Series servo motor. (Encoder resolution is dependent on motor selected. Consult the *Servo Product Specifications Guide,* GFH-001 for details.)

An optional 24 VDC holding brake is also available with each model.

βi Series Digital Amplifiers

The β*i* and βHVi Series servo amplifier integrates a power supply with the switching circuitry. Therefore, GE Fanuc is able to provide a compact amplifier that is 60% smaller than conventional models. The β*i* Series amplifier has the same height and depth as a GE Fanuc PACSystems controller. This allows efficient panel layout when using the PMM335 motion controller.

The amplifier is designed to conform to international standards.

αHVi and αHVis Series Servo Motors

The αHVi Series servos are high voltage models with an AC voltage input range of 400-480 VAC and include separately mounted shared power supply modules. These power supply modules offer line regeneration capability, which eliminates the need for external power resistors to dissipate regenerated energy during motor deceleration. Depending on the motor rating, up to six amplifiers can be connected to one power supply.

αHVi and αHVis Series Servo Amplifiers

The αHVi series amplifiers are compact, highly efficient amplifiers with a thin modular designed to conserver panel space. Easy maintenance, less cabling, and quick power line connections reduce installation cost and downtime during amplifier replacement.

- Width is reduced by 20% (average) and depth is reduced by 11% to contribute to cabinet downsizing.
- Power loss is reduced by 10% with the latest intelligent power device to provide energy saving.
- High-resolution serial encoder, 1048576 counts (2^{20}) per revolution, for smooth speed control and high position accuracy.
- Continuous torque range of 4-75 Nm.
- Optional 24Vdc motor holding brake.
- Noise immune fiber optic interface from controller to amplifier reduces wiring cost and supports distributed central architectures.
- Absolute feedback is supported with optional encoder battery backup.
- Separately mounted power supply modules supports up to six amplifiers and are line regenerative.

Related Information

PACSystems CPU Reference Manual, GFK-2222

TCP/IP Ethernet Communications for PACSystems, GFK-2224

Station Manager for PACSystems, GFK-2225

Logic Developer Getting Started, GFK-1918

PACSystems RX3i Hardware and Installation Manual, GFK-2314

Servo Products Specifications Guide, GFH-001

AC Servo Motor βis Series - Descriptions Manual, B-65302EN

PACMotion Multi-Axis Motion Controller Datasheet and Important Product Information, GFK-2449

Fiber Terminal Block Datasheet and Important Product Information, GFK-2467

In addition to the documents listed above, product update documents describe product revisions of individual modules. The most recent PACSystems documentation is available on the GE Fanuc website: http://www.gefanuc.com/.

PLCOpen web site, http://www.plcopen.org/

Getting Help

If you purchased this product through a GE Fanuc Authorized Channel Partner, please contact them directly. Otherwise, contact GE Fanuc Customer Care at:

Customer Care Hotline Toll free: 800-GE FANUC (800-433-2682)

International Americas direct dial: 780-401-7700

Other International contact info: www.gefanuc.com

Technical Support Email: support@gefanuc.com

Customer Care Email: customer.care@gefanuc.com

Web Support www.gefanuc.com

Chapter 2 Getting Started

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

Motion System Overview 2-2

Basic Installation 2-6

Basic Configuration 2-12

Axis Powerup and Feedback Device Initialization 2-19

Basic Motion Example – Jogging an Axis 2-20

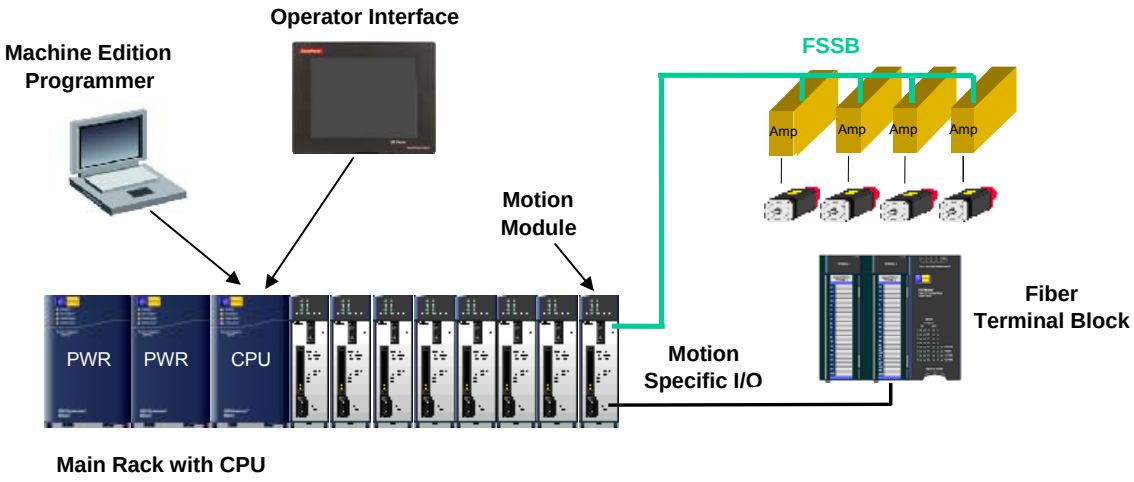
Using the Synthetic Motor 2-22

The PMM requires version 5.9 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.

Motion System Overview

A typical PACMotion system includes the PMM motion controller, a PACSystems RX3i controller, Machine Edition Programmer, motor(s), servo amplifier(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.



PACMotion Programming- Machine Edition

The PMM is hosted by the PACSystems RX3i PAC controller and is programmed and configured using 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 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.

PMM/RX3i Interface

The PMM335 and PACSystems RX3i CPU operate together as one integrated motion control package. The PMM335 communicates with the CPU and other PMM335 modules via the RX3i high-speed PCI-based backplane bus. The RX3i CPU and PMM335 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 PMM335 support multiscan (interrupt driven and time driven) capability and demand driven communications.

Multiscan provides a flexible way to partition the application to achieve the highest performance. The base CPU multiscan 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 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).

Fiber I/O Terminal Block

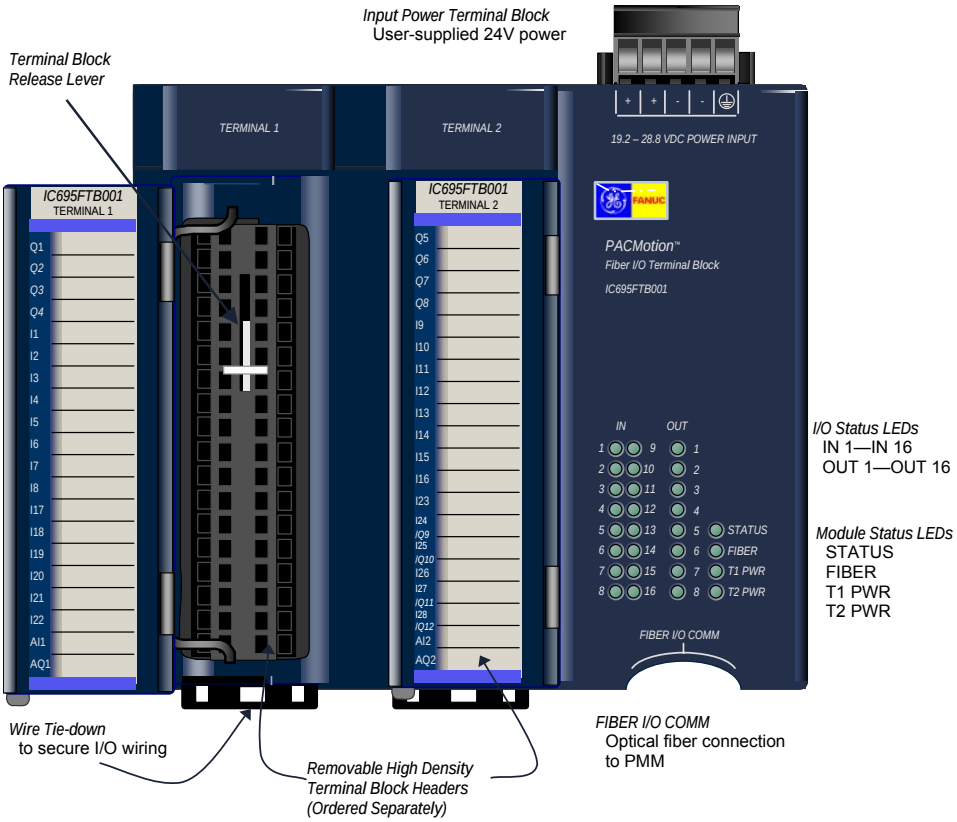
The DIN rail mounted FTB extends the faceplate I/O and allows motion-centric I/O to be distributed up to 100m from the RX3i rack. The FTB contains an external I/O controller that provides a broad range of configurable I/O for the axes controlled by the PMM module.

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:

- Quad 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

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

Because the 5V 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 information on motion system and I/O connections to the FTB, refer to Chapter 3.



Servo Amplifier and Machine Interfaces

The servo amplifier interface is made through an FSSB connector, which provides a noise immune fiber optic interface to the servo amplifier. This interface carries all control and feedback signals necessary for the PMM to control the axis.

PMM I/O

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

FTB/PMM Communications

An FTB identifier can be assigned to each PMM/FTB pair. The transfer of the identifier 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, see Chapter 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 a 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.

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 Chapter 6, “PACMotion Instruction Set Reference.”

Basic Installation

The PMM must be located in a PACSystems RX3i Universal Backplane (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 GE Fanuc.

PMM Installation

Maximum Number of PACMotion Multi-Axis Motion 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 which 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

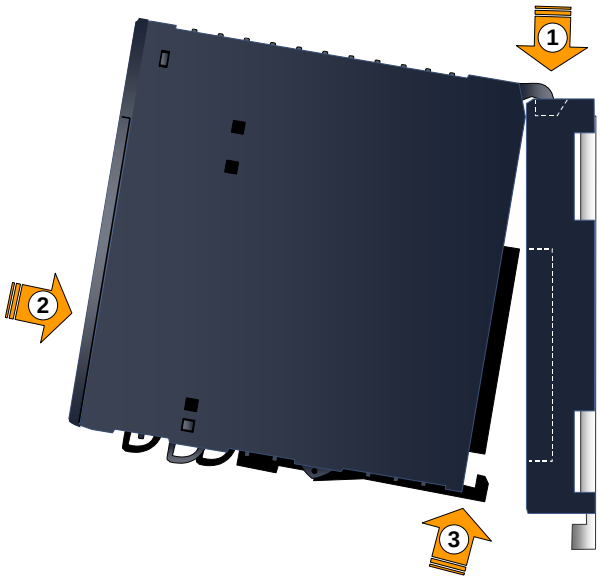
Multifunctional Power Supplies	***Qty***	***+5VDC Power per Power Supply***	***+3.3VDC Power per Power Supply***	***Total +5VDC Power Supplied***	***Total +3.3VDC Power Supplied***	***No. of Slots Used***	***No. of Real Axes***
IC695PSA140	2	6A (30W)	9A (29.7W)	12A (30W)	18A (59.4W)	4	

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	
IC695PMM335	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

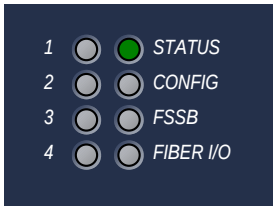
Note: The PMM supports hot insertion and removal. For details on this and other installation and environmental considerations for PACSystems RX3i components, refer to the *PACSystems RX3i System Manual,* GFK-2314.

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 (1).
5. Swing the module down (2) until the module's connector engages the backplane's backplane connector, and the release lever(s) on the bottom of the module snaps into place in the bottom module retainer (3).
6. Visually inspect the module to be sure it is properly seated.
7. Connect faceplate I/O wiring. For detailed wiring information, see Chapter 3.



8. If an FTB is to be used with the PMM, install it as described on page 2-8.
9. Repeat this procedure for each PMM in your RX3i system.

Note: When the PMM initially powers up, the module's STATUS LED should be on and green.

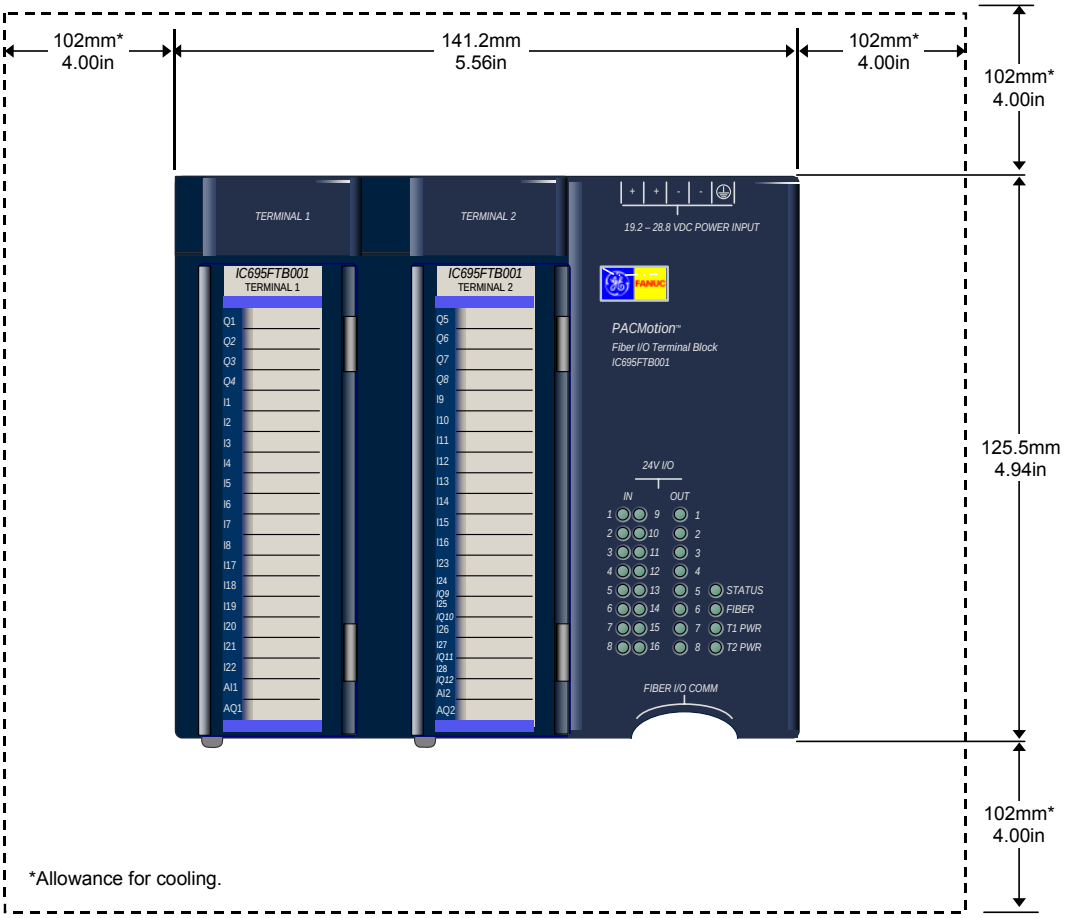


10. Configure the PMM(s) and FTB(s) as described in "Basic Configuration" on page 2-12.

FTB Installation

The FTB can 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.

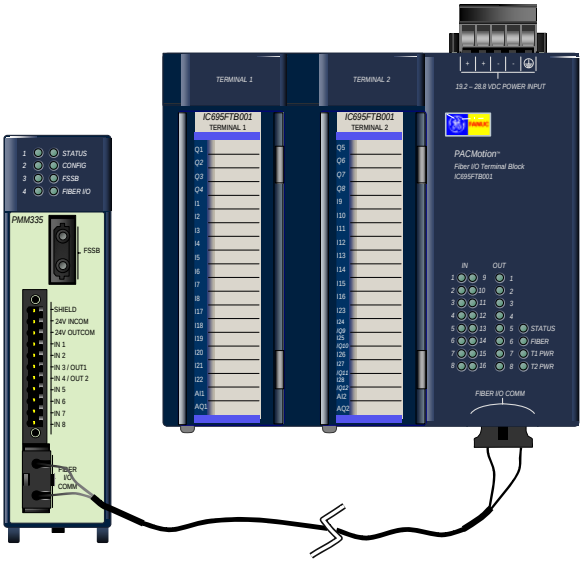
FTB Mounting Dimensions



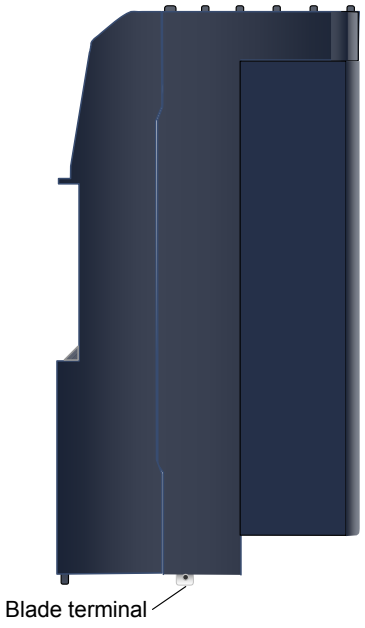
For RX3i environmental requirements, refer to the *PACSystems RX3i System Manual,* GFK-2314.

Installing the FTB

1. Slide the FTB onto the DIN rail.
2. Using an *FSSB Fiber Optic Servo Command Interface Cable,* connect the FIBER I/O COMM ports on the FTB and the PMM. The connectors are keyed and it is possible to insert the connector only with the proper orientation. For cable part numbers, see Chapter 3.



3. Connect the shield ground wire (shipped with the FTB) from the ¼ inch blade terminal at the bottom of the FTB to a panel frame ground.

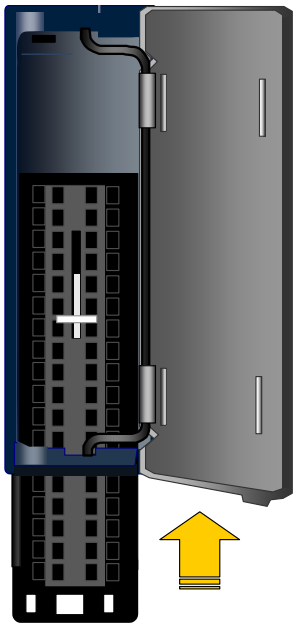


Warning

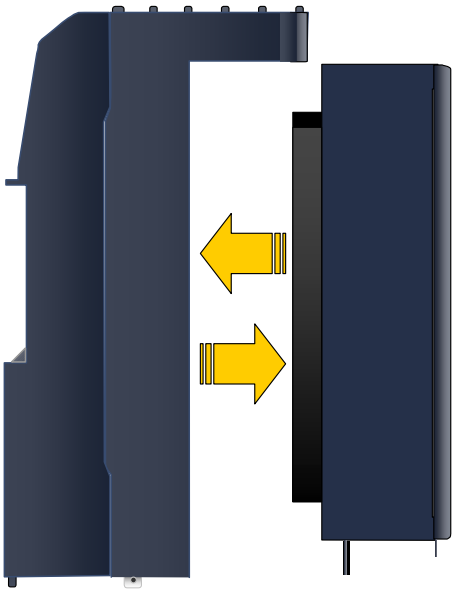
Field power must be turned off when installing or removing a Terminal Block assembly.

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 Chapter 3 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.

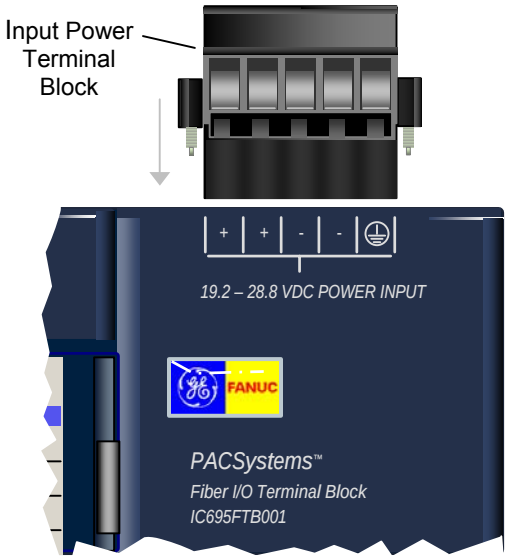
Note: Terminal blocks must be ordered separately from the FTB.



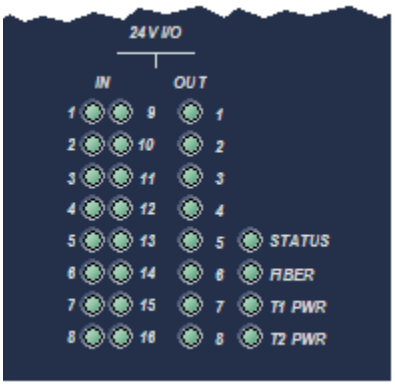
5. Install the terminal blocks on the FTB
   d. Press the terminal block release lever down.
   e. Press the terminal block assembly straight toward the FTB until it is partially seated.
   f. Open the door on the front of the terminal block and push the terminal block release lever up very firmly until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



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.



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



8. Configure the PMM(s) and FTB(s) as described in “Basic Configuration” on page 2-12.

Basic Configuration

This section provides an overview of the configuration process. Refer to Chapter 4 for details on configuration parameters.

For required programmer and CPU revisions, refer to *PACMotion Multi-Axis Controller Datasheet and Important Product Information*, GFK-2449.

1. Connect the programmer to the RX3i.
2. Add a PMM to the hardware configuration, page 2-13
3. Configure PMM Settings, page 2-14
4. Configure PMM I/O, page 2-14
5. Configure axis parameters, page 2-16
6. Download the configuration to the RX3i, page 2-18

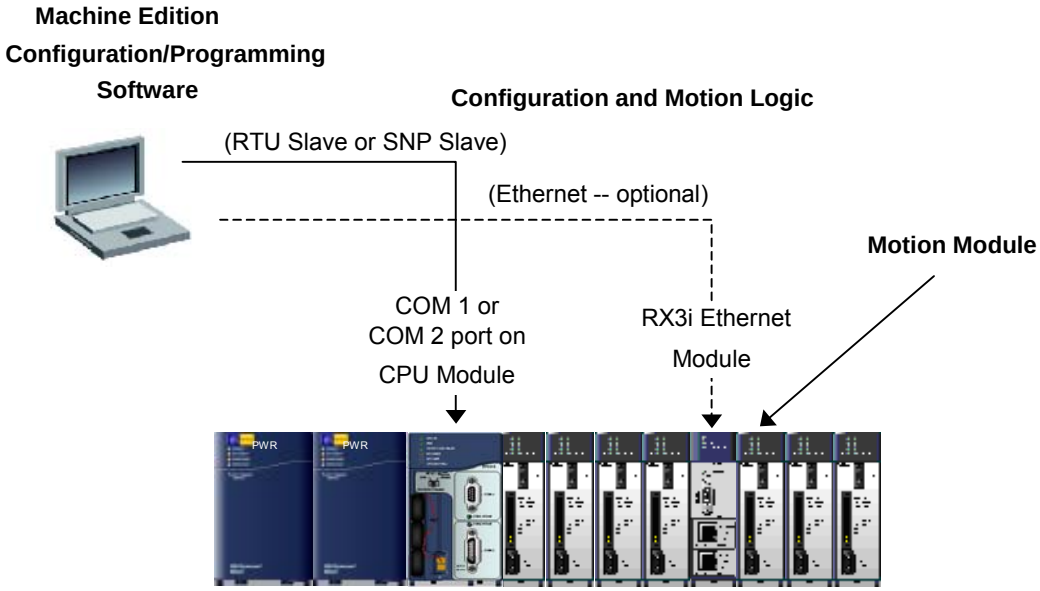
Connecting the Programmer to the RX3i

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 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. 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 *TCP/IP Ethernet Communications for PACSystems,* GFK-2224.

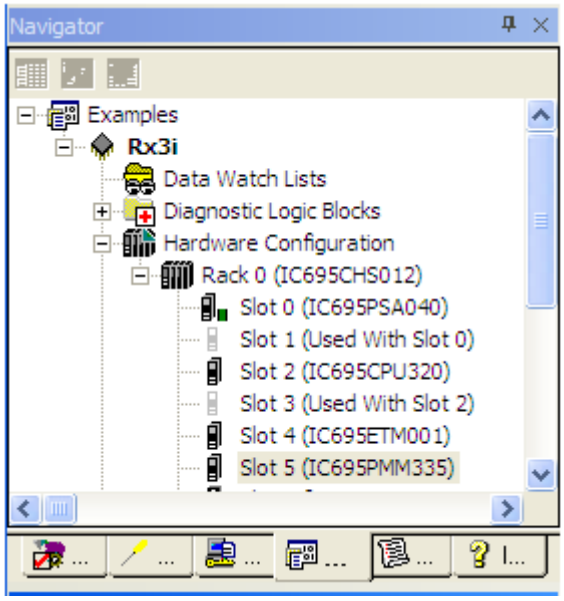
PMM335 Programmer Connection



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.

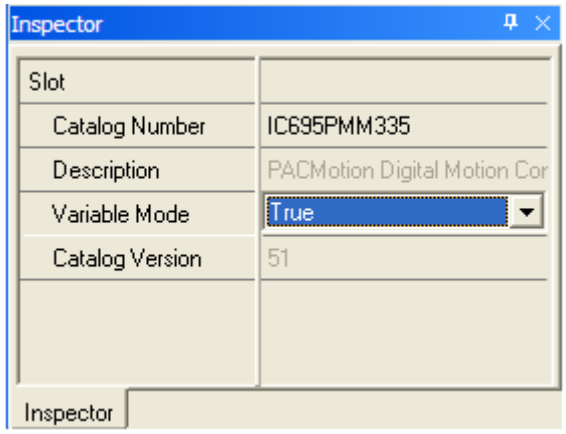
To configure a PMM using the Logic Developer software:



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 PMM335 and click OK.

   This operation adds the PMM335 to the RX3i rack and displays the PMM335 configuration screens that allow you to customize the PMM335 to your particular application.

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.



Configuring PMM Settings

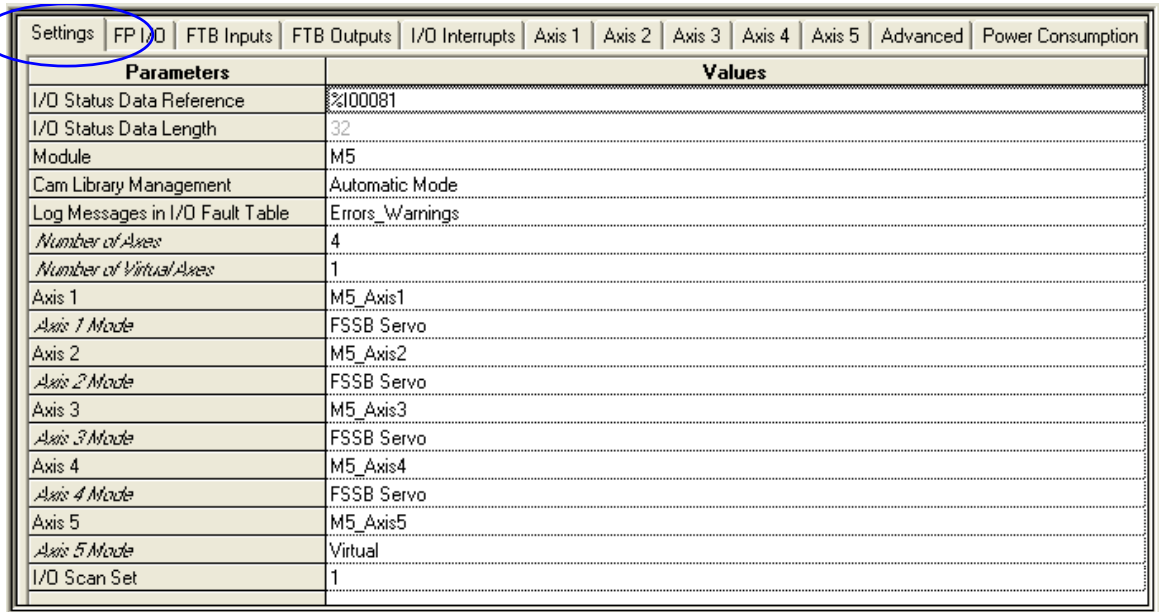
Note: For detailed descriptions of these parameters, refer to Chapter4, “Configuration.”

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.



Parameters	Values
I/O Status Data Reference	%I00081
I/O Status Data Length	32
Module	M5
Cam Library Management	Automatic Mode
Log Messages in I/O Fault Table	Errors_Warnings
Number of Axes	4
Number of Virtual Axes	1
Axis 1	M5_Axis1
Axis 1 Mode	FSSB Servo
Axis 2	M5_Axis2
Axis 2 Mode	FSSB Servo
Axis 3	M5_Axis3
Axis 3 Mode	FSSB Servo
Axis 4	M5_Axis4
Axis 4 Mode	FSSB Servo
Axis 5	M5_Axis5
Axis 5 Mode	Virtual
I/O Scan Set	1

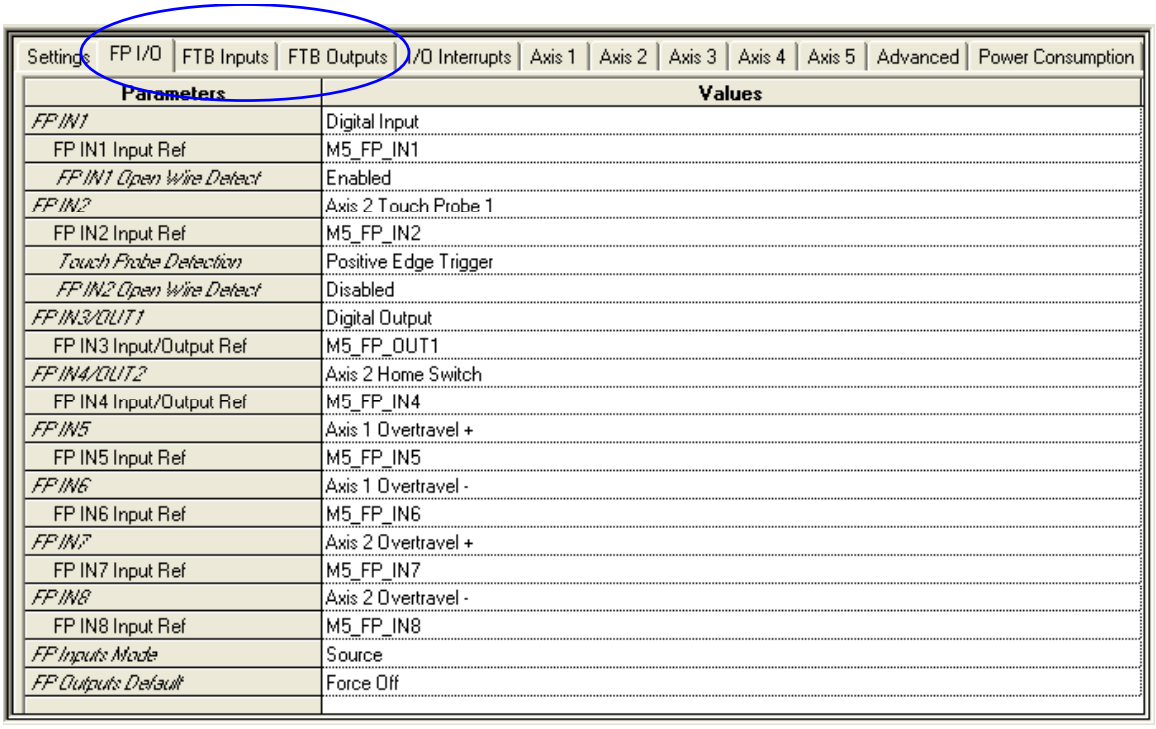
Configuring PMM I/O

The PMM faceplate includes eight built-in, configurable single-ended 24 VDC 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 Chapter 3, “Wiring and Connections.”



Settings | FP I/O | FTB Inputs | FTB Outputs | I/O Interrupts | Axis 1 | Axis 2 | Axis 3 | Axis 4 | Axis 5 | Advanced | Power Consumption

Parameters	Values
FP IN1	Digital Input
FP IN1 Input Ref	M5_FP_IN1
FP IN1 Open Wire Detect	Enabled
FP IN2	Axis 2 Touch Probe 1
FP IN2 Input Ref	M5_FP_IN2
Touch Probe Detection	Positive Edge Trigger
FP IN2 Open Wire Detect	Disabled
FP IN3/OUT1	Digital Output
FP IN3 Input/Output Ref	M5_FP_OUT1
FP IN4/OUT2	Axis 2 Home Switch
FP IN4 Input/Output Ref	M5_FP_IN4
FP IN5	Axis 1 Overtravel +
FP IN5 Input Ref	M5_FP_IN5
FP IN6	Axis 1 Overtravel -
FP IN6 Input Ref	M5_FP_IN6
FP IN7	Axis 2 Overtravel +
FP IN7 Input Ref	M5_FP_IN7
FP IN8	Axis 2 Overtravel -
FP IN8 Input Ref	M5_FP_IN8
FP Inputs Mode	Source
FP Outputs Default	Force Off

Configuring Axis Parameters

Axis operational characteristics are configured on the Axis tabs. For detailed axis configuration information, see Chapter 4.

For initial validation of configuration and system setup, *Position Feedback Source* should be set to *Motor Encoder.*

Set the Motor Type and Motor Encoder Counts per Revolution for the motor that you will be using. A list of motor type codes is provided on page 2-17. (The default Motor Type code 65535 specifies a Synthetic Motor, which can be used to test applications as they are being developed.)

Settings | FP I/O | FTB Inputs | FTB Outputs | I/O Interrupts | Axis 1 | Axis 2 | Axis 3 | Axis 4

Parameters	Values
Stop Axis on FTB Error	Enabled
Position Feedback Source	Motor Encoder
Axis Positioning Mode	Linear
Motor Encoder Mode	Absolute
Motor Encoder Low Battery Alarm	Enabled
Motor Encoder User Units	1.0
Motor Encoder Counts	1
Motor Encoder Counts Per Motor Revolution	65536
Motor Encoder Maximum Positive RPM Limit	8191
Motor Encoder Maximum Negative RPM Limit	8191
External Device	None
Over Travel Limit Switch	Enabled
Axis Direction	Normal
Software End of Travel	Enabled
High Software EOT Limit (uu)	8388607.0
Low Software EOT 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)	1.0E+015
Drive Disable Delay (ms)	100
Motor Type	65535
Motor Velocity Limit (RPM)	4000
Torque Limit (%)	100.0

Motor Type Codes

Either FANUC motor model numbers or motor part numbers can be used to determine the motor type code. FANUC motor part numbers, found on the motor label, are in the form ZA06B-xxxx-yyyy, where xxxx is a four-digit Motor Specification that can be used to look up the Motor Type code configuration value in the following table.

Motor Model	*Motor Specification*	*Motor ID (Type Code)*	*Maximum Speed (rpm)*	*Maximum Encoder Resolution (cts/rev)*
α2/6000HVis	ZA06B-*0212*-B#0#	284	6000	1048576
α4/5000HVis	ZA06B-*0216*-B#0#	266	5000	1048576
α8/6000HVis	ZA06B-*0233*-B#0#	292	6000	1048576
α12/4000HVis	ZA06B-*0239*-B#0#	289	4000	1048576
α22/3000i	ZA06B-*0247*-B#0#	297	3000	1048576
α22/3000Hvi	ZA06B-*0249*-B#0#	299	3000	1048576
α22/4000Hvis	ZA06B-*0266*-B#0#	316	4000	1048576
α30/4000Hvis	ZA06B-*0269*-B#0#	319	4000	1048576
α40/4000Hvis	ZA06B-*0273*-B#0#	323	4000	1048576
α50/3000HVis FAN	ZA06B-*0276*-B#1#	326	3000	1048576
α100/2500HVis FAN	ZA06B-*0286*-B#1#	336	2500	1048576
β0.2/5000iS	ZA06B-*0111*-B#0#	260	5000	65536
β0.3/5000iS	ZA06B-*0112*-B#0#	261	5000	65536
β0.4/5000iS	ZA06B-*0114*-B#0#	280	5000	65536
β0.5/6000iS	ZA06B-*0115*-B#0#	281	6000	65536
β1/6000iS	ZA06B-*0116*-B#0#	282	6000	65536
β2/4000is	ZA06B-*0061*-B#0#	253	4000	131072
β2/4000HVis	ZA06B-*0062*-B#0#	251	4000	131072
β4/4000is	ZA06B-*0063*-B#0#	256	4000	131072
β4/4000HVis	ZA06B-*0064*-B#0#	264	4000	131072
β8/3000is	ZA06B-*0075*-B#0#	258	3000	131072
β8/3000HVis	ZA06B-*0076*-B#0#	267	3000	131072
β12/3000is	ZA06B-*0078*-B#0#	272	3000	131072
β12/3000HVis	ZA06B-*0079*-B#0#	270	3000	131072
β22/2000is	ZA06B-*0085*-B#0#	274	2000	131072
β22/2000HViS	ZA06B-*0086*-B#0#	278	2000	131072
Synthetic Motor*	NA	65535	6000	1048576

***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.

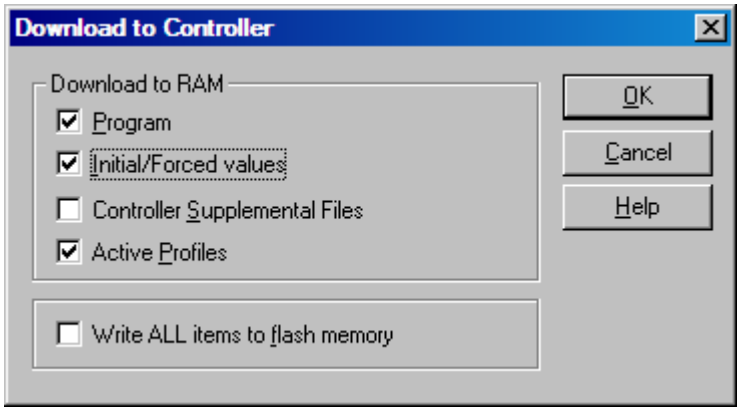
Downloading the Configuration to the RX3i

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 (see page 2-12), 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.



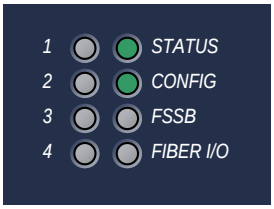
4. In the Download to Controller dialog box, select the items to download and click OK.

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 PMM's CONFIG LED is on and green.



Axis Powerup and Feedback Device Initialization

To successfully execute motion-generating function blocks on an axis, the axis must be powered on by a MC_Power function block and set to a valid initial position by a 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 maintain its commanded position and retain its PositionValid status.

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 ""Function Block Triggering" in Chapter 5.

Hardware Configuration (HWC)

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

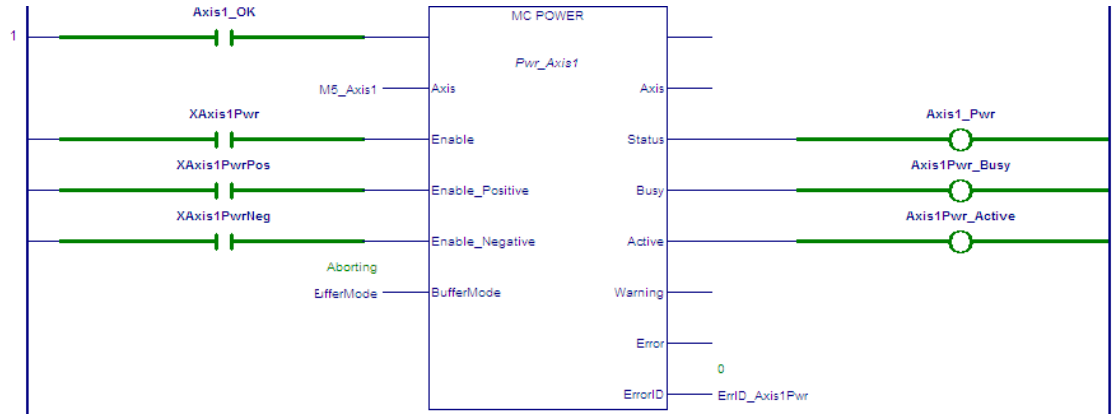
Settings Tab

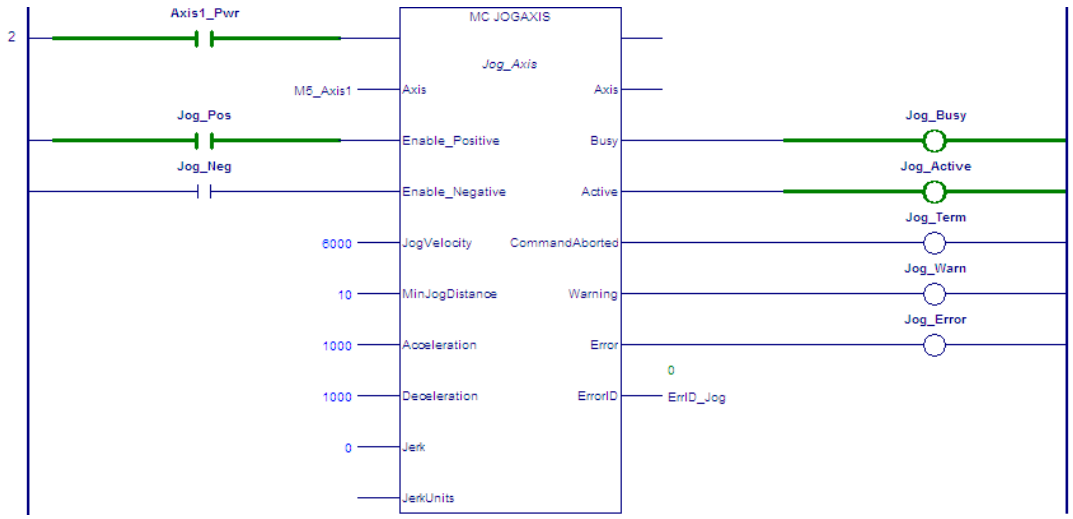
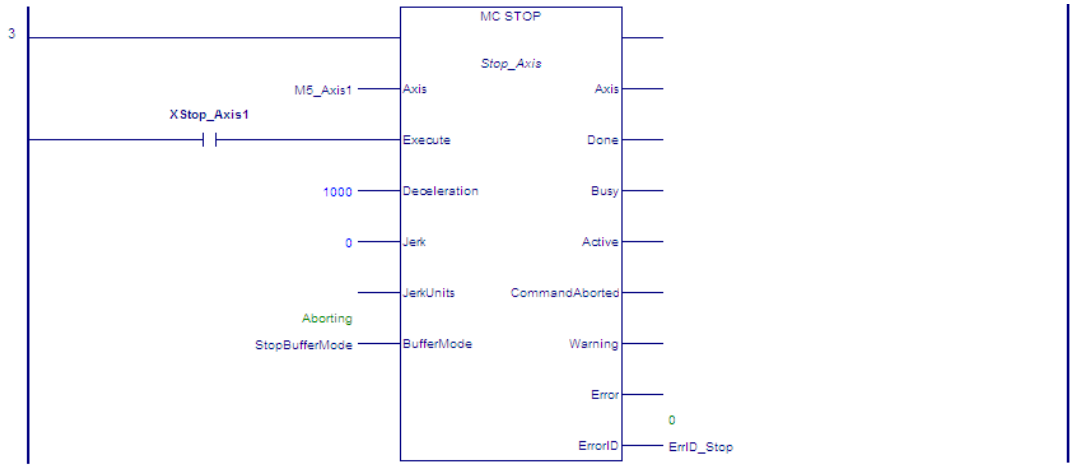
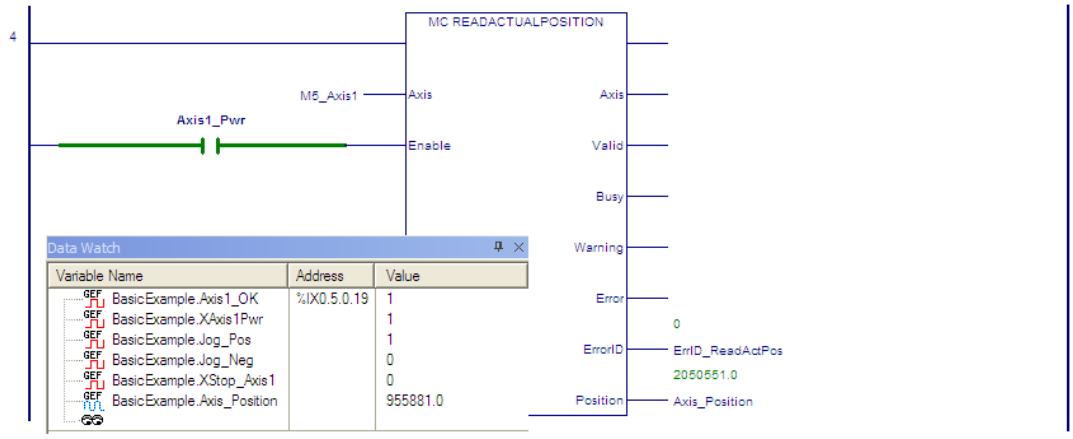
Module: M5

Axis 1: M5_Axis1

Terminals Tab

Axis1_OK is an I/O variable mapped to the Axis 1 flag in the PMM Status Data. When ON, this flag indicates that the axis is enabled in HWC and is not in the ErrorStop state.

Comments	*Sample Logic*
To perform motion, an axis must be enabled by a MC_Power function block. If Enable_Positive and Enable_Negative have the same state, motion is enabled in both directions.	

Comments	Sample Logic
To move the axis, apply power to the MC_JogAxis function block and to either the Enable_Positive or the Enable_Negative input.	
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.	
Axis1_Pwr is controlled by the Status output of the MC_Power function block. When *Axis1_Pwr* is on, the axis' actual position is output to *Axis_Position.*	

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 out 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 as such requires tuning to achieve a low Position Error.

To use the synthetic motor with an axis, set the axis hardware configuration parameter, *Motor Type* to 65535. 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 Chapter 4, “Configuration.”

Chapter 3

Wiring and Connections

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

I/O Wiring, Connections, and LEDs 3-2

PMM Faceplate 3-2

FTB 3-7

I/O Circuit Specifications 3-15

FSSB Connections 3-25

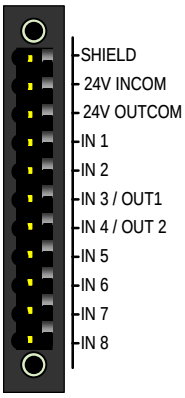
FTB to PMM Connection 3-26

Grounding the PACMotion System 3-27

I/O Wiring, Connections, and LEDs

PMM Faceplate I/O

The faceplate provides eight digital I/O inputs, consisting of six general-purpose 24V inputs, and two high-speed 24V source/sink inputs. Two of the I/O points can be configured as general-purpose 24V inputs or low current (30 mA) 24V source/sink output drivers. The inputs and outputs are in a single group with power and common terminals. The Machine Edition 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 24V 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 faceplate I/O requires 24V user-supplied power.

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.

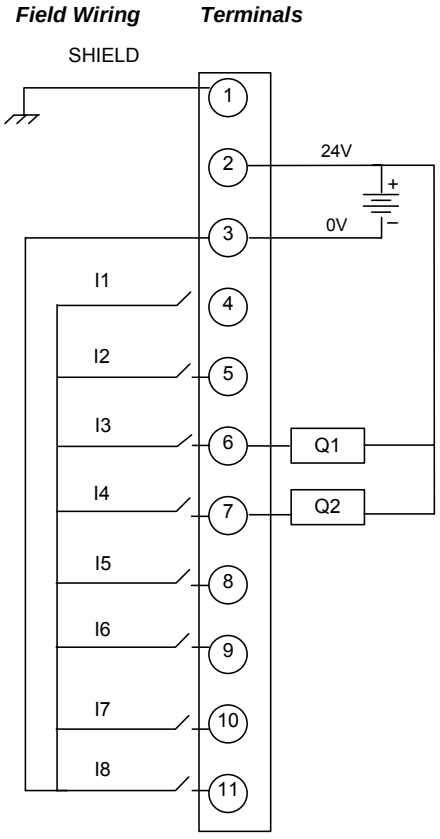
For specifications of I/O circuits, see page 3-15.

PMM Faceplate Wiring Diagrams and Pin Assignments

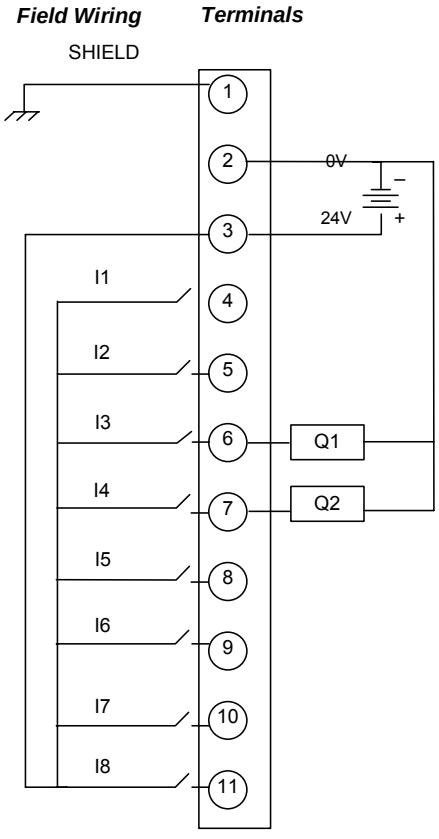
Faceplate I/O Connector Pin Assignments

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

Sink Mode Wiring

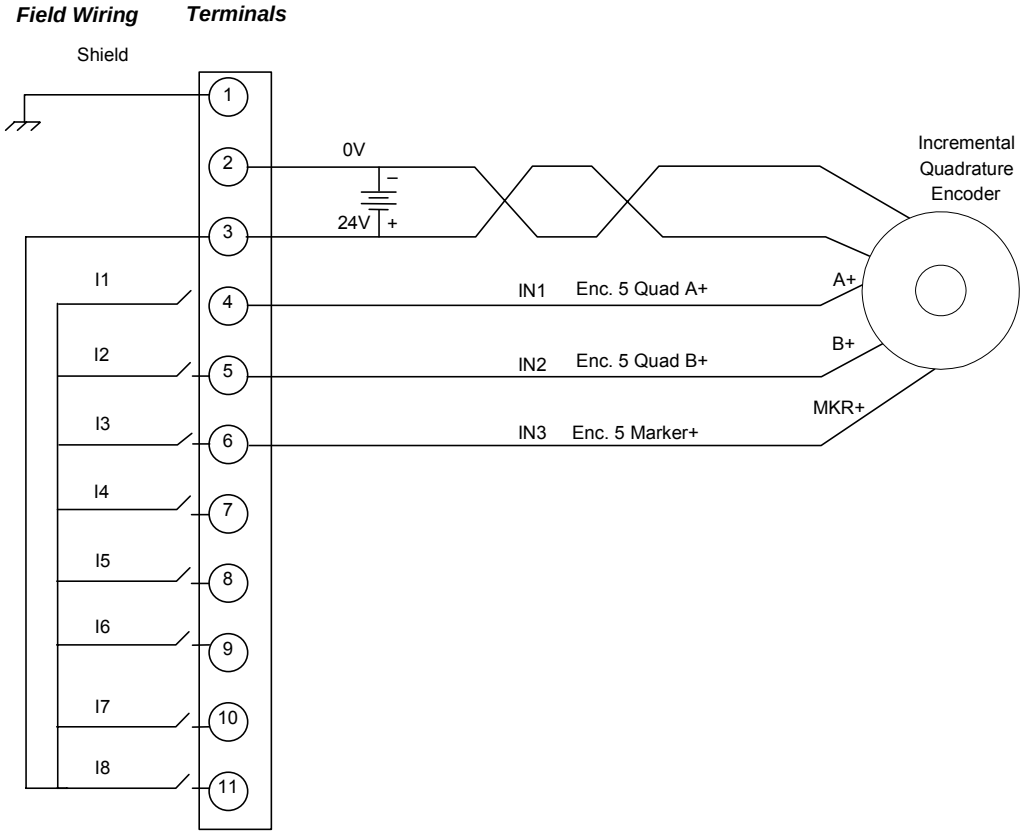


Source Mode Wiring



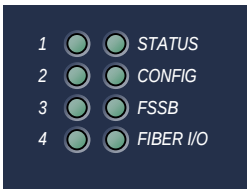
Typical External Encoder Connection

When a quadrature encoder is used with the faceplate, the quadrature totem pole drivers are powered from the 24V 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.



PMM335 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 FSSB and FTB communications links.



Axis LEDs

LED	LED State	PMM335 Operating State
1 (Axis 1) 2 (Axis 2) 3 (Axis 3) 4 (Axis 4)	Green ON	The Axis drive corresponding to this LED is enabled.
	Red ON	A normal stop error has occurred on this axis.
	Red, flashing 500ms interval	A fast stop error has occurred on this axis.

PMM Status LEDs

LED	LED State	PMM335 Operating State
STATUS	OFF	The PMM does not have power.
	Green ON	The PMM335 is functioning properly and there are no errors on the module.
	Green, flashing 1 second interval	Indicates that a *Warning* or an *Error* not requiring a stop has occurred.
	Green, flashing 500ms interval	Indicates that an *Error* requiring a fast or normal stop has occurred.
	Green, flashing blink code; CONFIG and axis LEDs OFF	The blink pattern indicates that a fatal initialization error has occurred on the module while powering up. Record the number of blinks in the sequence and contact GE Fanuc 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, flashing	The PMM has not yet received a configuration from the programmer.
	Amber ON	The PMM is in boot mode.
	Amber, flashing	The PMM received an invalid configuration from the programmer.

Communications Status LEDs

LED	LED State	PMM335 Operating State
FSSB	OFF	FSSB communications is inactive.
	Green ON	FSSB communications is active.
	Green, flashing	FSSB setup is in progress or servo amplifiers not yet available. Axes can be used with synthetic motor. This is normal operation.
	Red ON	FSSB communication link has failed.
FTB	OFF	Fiber I/O communications is inactive.
	Green ON	Fiber I/O communications is active.
	Green flashing	Fiber I/O configuration is in progress.
	Red ON	Fiber I/O communication link has failed.
	Alternately flashing 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.

Fiber I/O 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 5V differential inputs includes open wire and loss of encoder power. Fault detection is enabled or disabled in hardware configuration.

The Machine Edition 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 on page 3-8.

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 5V and analog I/O lines to provide noise immunity.

IC695FTB001 Fiber I/O Terminal Block 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”)
24V Outputs (differential)	8, 2 groups of 4 optically isolated @ 4A max per group
24V Inputs (general purpose)	16, 4 groups of 4 optically isolated
5V Input/Outputs	4, differential inputs or 4, differential outputs
5V Inputs	8, differential (6 can be used as single ended)
Analog Inputs	2, ±10V differential
Analog Outputs	2, ±10V single-ended
5V Power Output	0.5A max.
Quad Encoder Open Circuit Detection	Yes
Encoder Power output	5V at 0.5A max

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

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*	
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*	
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

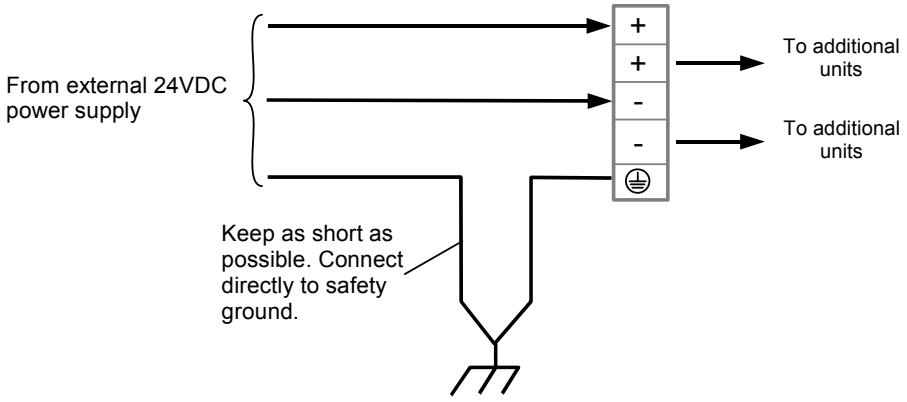
* For fiber optic lengths less than 10 meters, unsheathed cable is recommended. For fiber optic lengths from 10 to 100 meters, sheathed cable is required.

Warning

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

FTB Input Power

Each terminal accepts one AWG #14 (2.1mm^2) or two AWG #16 (1.3mm^2) 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.

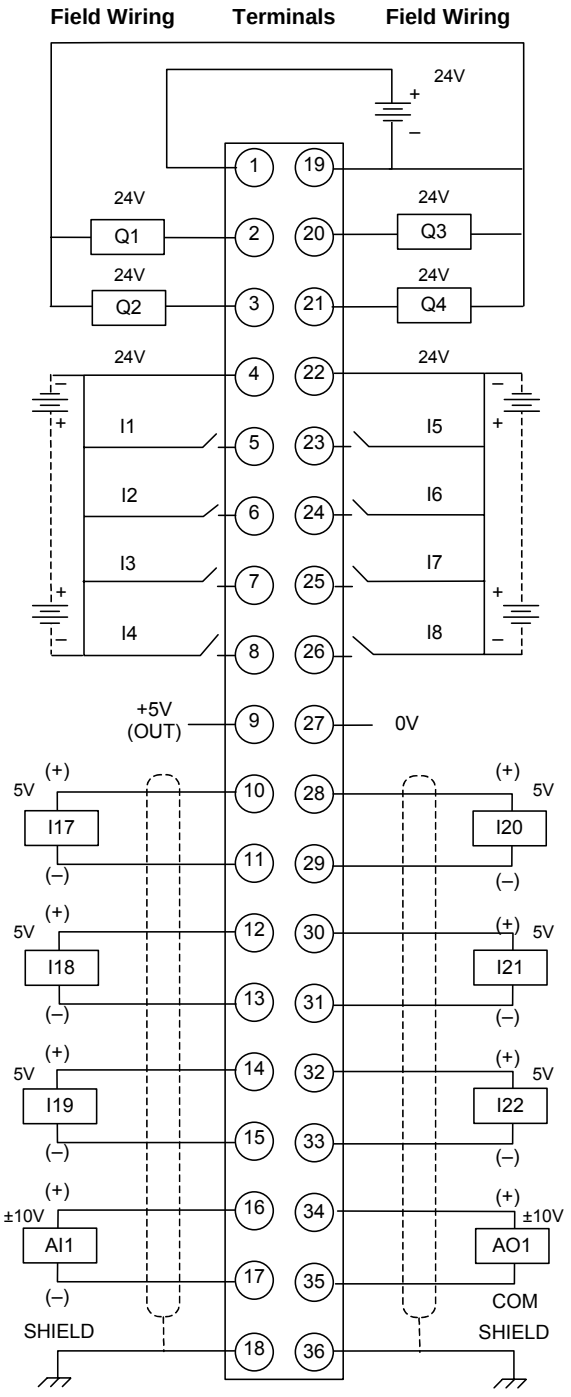


Fiber I/O Terminal Block IC695FTB001 Wiring Diagrams and Pin Assignments

FTB Terminal 1 Pin Assignments

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

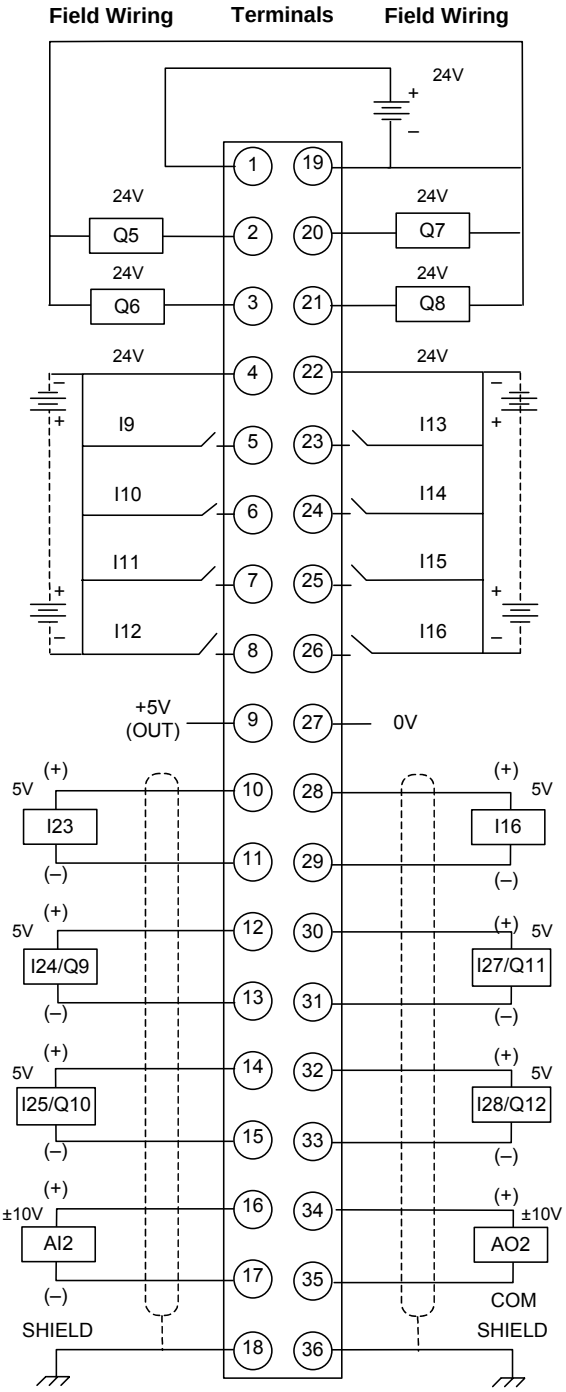
FTB Terminal 1 Wiring Diagram



FTB Terminal 2 Pin Assignments

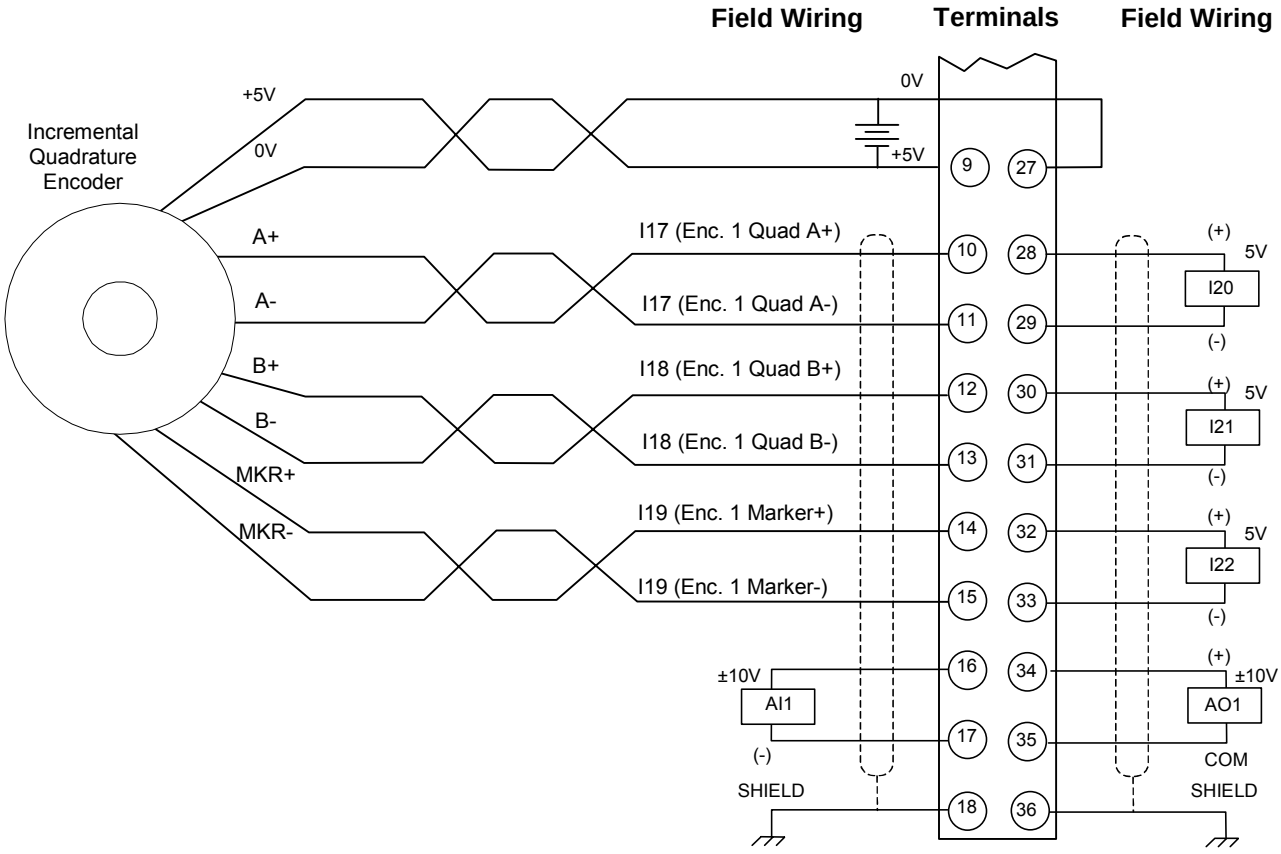
Pin	Circuit Identifier	Circuit Type	Default Circuit Function
1	24V+	24V Output	Q5—Q8 Power
2	Q5	24 VDC (ESCP) Output	24V Output
3	Q6		24V Output
4		24V+	I9—I12 Common
5	I9	24 VDC Input	Digital Input
6	I10		Digital Input
7	I11		Digital Input
8	I12		Digital Input
9	+5V (OUT)	+5V OUT	External Power
10	I23+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

FTB Terminal 2 Wiring Diagram



Typical External Differential Encoder Connection for FTB

When a differential quadrature encoder is used with the FTB, the quadrature drivers are powered from the 5V 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.

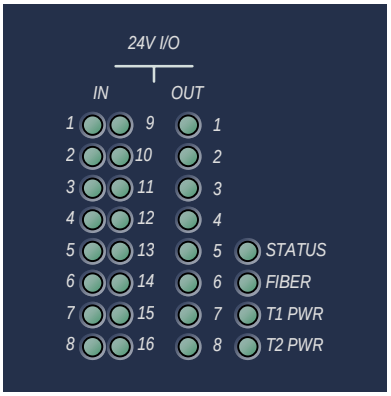


FTB LED Operation

The FTB has one LED for each of the 24V 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.

24V 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



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, flashing	I/O link configuration data being transferred.
	Alternately flashing 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 24V 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 24V 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.

Errors Indicated by LEDs

Use the MC_ReadAxisError or MC_ReadEventQueue function block to obtain ErrorIDs that are reported in Axis Error Codes.

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. Correct the source of the error. Then use MC_Reset to transition the axis out of the ErrorStop state.
Axis LED red, flashing 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.	
STATUS LED green, flashing 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, flashing 500ms interval	An *Error* requiring a fast or normal stop has occurred.	
STATUS LED green, flashing blink code; CONFIG and axis LEDs OFF	A fatal initialization error has occurred on the module while powering up. 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 GE Fanuc technical support for additional information.
STATUS LED amber	A severe module hardware error or watchdog timeout has occurred.	Contact GE Fanuc with the fault table content and event queue, captured as soon after the error occurred as possible, from the PMM generating the error. GE Fanuc support personnel may also ask for copies of the logic and/or HWC for the system experiencing the error.
CONFIG LED is green, flashing	The PMM has not yet received a configuration from the programmer.	Store a configuration to the RX3i.
CONFIG LED is amber, flashing	The PMM received an invalid configuration from the programmer.	Store a valid configuration to the RX3i.
FSSB LED green, flashing	FSSB setup is in progress.	Normal operation. Servo amplifiers not yet available. Axes can be used with synthetic motor.
FSSB LED red, ON	FSSB communication link has failed.	Check the Axis Error Code. Correct the source of the error, then use MC_ModuleReset to clear all errors on the module.
FTB LED red ON	Fiber I/O communication link has failed.	Check the Axis Error Code.
FTB LED alternately flashing 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.

FTB LEDs

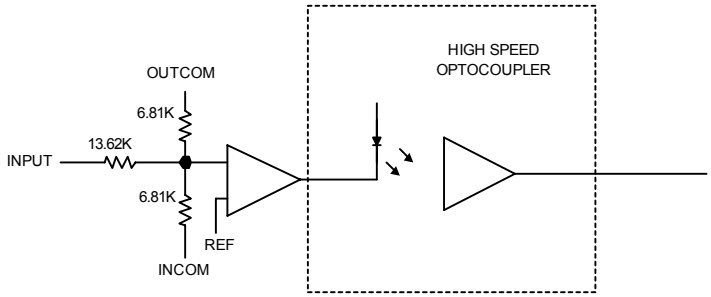
Symptom	*Possible Causes*	*Suggested Correction*
24V IO Out1 — Out 8 LED is red, ON	The specified 24-volt output has open load error.	Check wiring, or disable open load fault detection in HWC.
STATUS LED is amber, ON	Configuration not yet received.	Download a valid configuration to the RX3i.
FIBER LED alternately flashing 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 GE Fanuc with the fault table content and event queue, captured as soon after the error occurred as possible, from the PMM generating the error. GE Fanuc 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.

I/O Circuit Specifications

Faceplate I/O Circuits

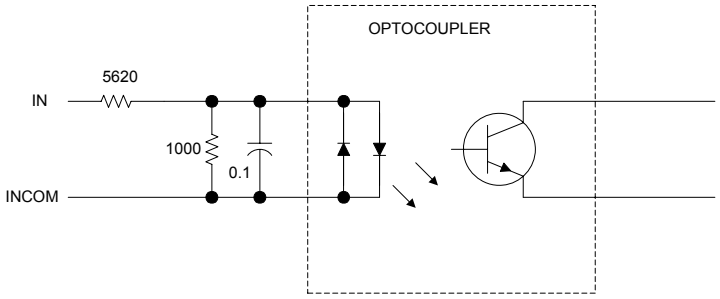
Faceplate 24V High Speed Inputs

Circuit Identifiers	IN1, IN2
IO Type	Optically isolated 24V 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 Kohms 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 24V supply
Logic 1 Threshold	65% Typical, 75% Max percentage of 24V 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



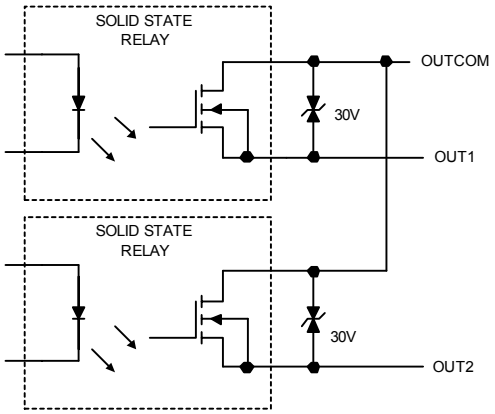
Faceplate 24V General Purpose Inputs

Circuit Identifiers	IN3, IN4, IN5, IN6, IN7, IN8
IO Type	Optically isolated 24V Source/Sink Inputs
Circuit Type	Source/Sink
Input Impedance	5.62 Kohms to INCOM
Maximum Input Voltage	±30.0V (referenced to INCOM)
Logic 0 Threshold	±6.0V maximum (referenced to INCOM)
Logic 1 Threshold	±18.0V minimum (referenced to INCOM)
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. 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.



Faceplate 24V General Purpose Outputs

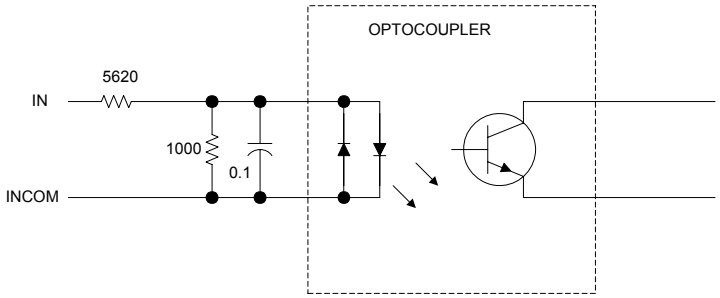
Circuit Identifier	OUT1, OUT2
IO Type	Dual Optically isolated 24V Source/Sink Outputs both connected to OUTCOM. Source/Sink operation depends on relative polarity of power applied to INCOM and OUTCOM.
Circuit Type	Solid State Relay (SSR) with electronic short circuit protection.
Power Supply Voltage	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 µA 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.



FTB I/O Circuits

FTB 24V General Purpose Inputs

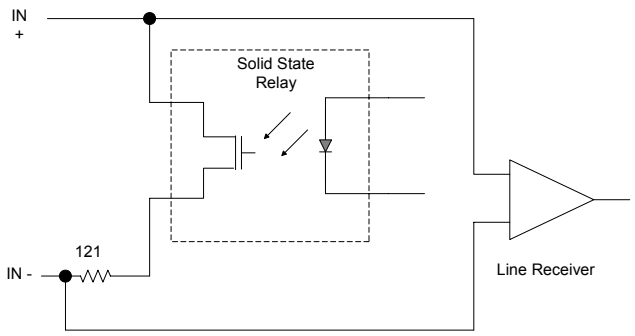
Circuit Identifiers	I1 – I16
IO Type	Optically isolated 24V Source/Sink Inputs
Circuit Type	Source/Sink
Input Impedance	5.62 Kohms to Input COM @24 VDC
Maximum Input Voltage	±30.0V (referenced to Input COM)
Logic 0 Threshold	±6.0V maximum (referenced to Input COM)
Logic 1 Threshold	±18.0V minimum (referenced 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.



FTB Differential/Single Ended 5V Inputs

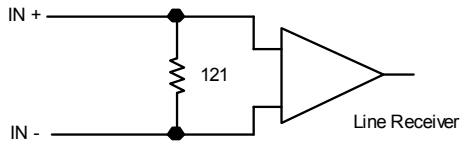
These inputs use a solid state relay in series with the differential termination resistor. Configuring the input for single-ended operation opens the 121Ω termination resistor. With single ended operation, only the + input is connected.

Circuit Identifiers	I17 – I22
IO Type	Differential or single-ended 5V input (configurable)
Circuit Type	RS422 / RS485 Line Receiver with fault detection
Input Impedance (Differential)	121 ohms
Input Impedance (Single Ended)	90K ohms minimum (+ input)
Maximum Input Voltage	±20V differential +/-10V common mode
Logic 0 Threshold	-0.475V max differential 1.2V single ended (typical)
Logic 1 Threshold	+0.475V max differential 1.5V 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 “Touch Probe Accuracy” appendix
Notes:	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 5V TTL or CMOS logic.



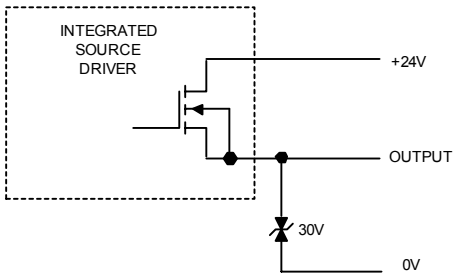
FTB Differential 5V Inputs

Circuit Identifiers	I23 – I28
IO Type	Differential 5V input
Circuit Type	RS422 / RS485 Line Receiver with fault detection
Input Impedance (Differential)	121 ohms
Input Impedance (Single Ended)	90K ohms minimum (+ input)
Maximum Input Voltage	±20V differential +/-10V common mode
Logic 0 Threshold	-0.475V max differential
Logic 1 Threshold	+0.475V max differential
Input Filtering	100ns typical
Fault Detect Filtering	100µs typical
Maximum quadrature input frequency (count rate is 4x input frequency)	2.50MHz (differential)
Quadrature tolerance at maximum input frequency	90 degrees ±45 degrees
Touch Probe Response (at constant velocity)	Minimum Pulse Width: 200ns (differential) Position Capture Accuracy: Refer to Touch Probe Accuracy Appendix
Notes:	Use terminal block 0V pins for common mode reference. Inputs can be driven by 5V TTL or CMOS logic.



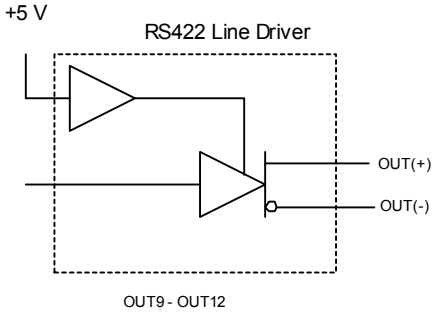
FTB 24V General Purpose Outputs

Circuit Identifiers	Q1 – Q8
IO Type	Optically isolated 24V source outputs, four per group
Circuit Type	Source (Open Drain pullup to +24V 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 +24V 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 24V across output
Turn Off / Turn On Delay	100 µs max driver delay



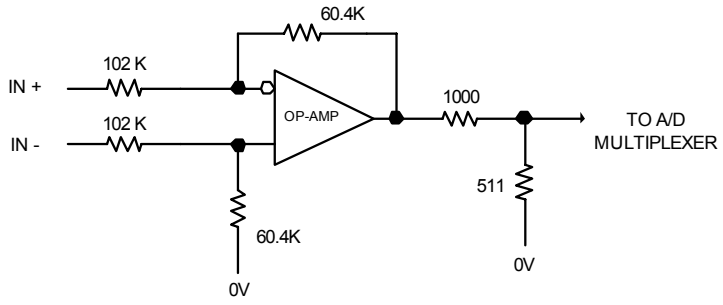
FTB 5V Differential Outputs

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



FTB Differential ±10V Analog Inputs

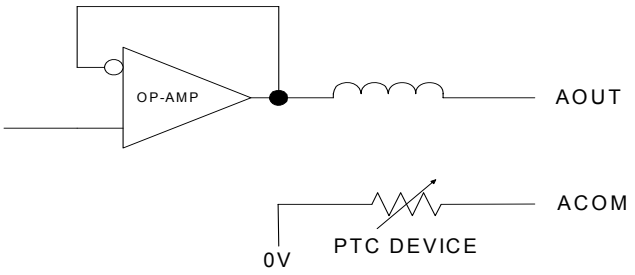
Circuit Identifiers	AI1, AI2
IO Type	Differential ±10.0V analog inputs
Circuit Type	Differential Analog input
Input Impedance (Differential)	204 Kohms
Input Impedance (Common mode)	102 Kohms common mode with respect to FTB connector 0V
Maximum Input Voltage	±15V common mode ±20V differential
Resolution	14 bits
Linearity	12 bits
Input Offset	±5.0 millivolts max
Gain Factor	10.0V = 10.000 (reported to the RX3i as floating point data)
Gain Accuracy*	±1.0 %
Update Rate	1 kHz
Note:	Use terminal block 0V pin for common mode reference.
*	In the presence of severe RF interference, the accuracy may be degraded to ±5.0%



FTB Single Ended ±10V Analog Outputs

Circuit Identifiers	AO1, AO2
IO Type	Single Ended Analog Output
Circuit Type	Op Amp output
Load Impedance	2K ohms minimum
Output Current	5 mA maximum
Resolution	12 bits
Linearity	12 bits
Output Offset Voltage	± 5.0 millivolts max
Gain Factor	10.000 floating point data from the RX3i = 10.0V output
Gain Accuracy*	±2.0 %
Update Rate	1 kHz
*	In the presence of severe RF interference, the accuracy may be degraded to ±5.0%

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 +10V and the negative differential input to AO COM.



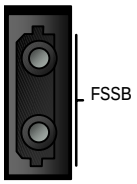
FTB +5V Power

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

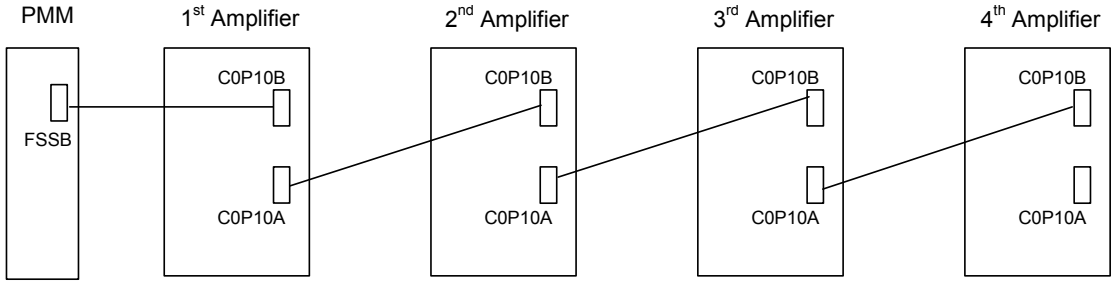
Circuit Identifier	+5V (OUT)
I/O Type	+5V Encoder Power
Circuit Type	+5V Power with Electronic Short Circuit Protection
Output Voltage	4.70V to 5.20V at 0.5 amp
Output Current	0.5 amp maximum (Per terminal block and total for both FTB terminal blocks.)

FSSB Connections

The PMM's front panel contains a Fanuc Serial Servo Bus (FSSB) connector that is used to communicate control data between the PMM control and the servo amplifier. The data includes motor command signals, feedback data, and diagnostics.



Each FANUC amplifier has two FSSB connectors, an optical transceiver labeled C0P10A and a second transceiver labeled C0P10B. The cable from the PMM to the first amplifier must be connected to the COP10B connector on the amplifier. Proper system operation requires that the FSSB cables be installed on the proper connectors as shown below.

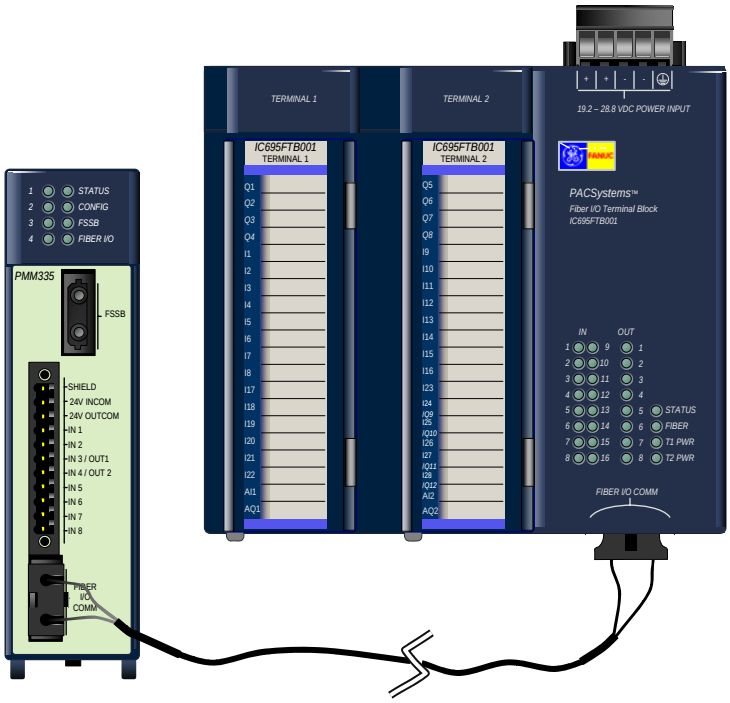


FSSB Fiber Optic Servo Command Interface Cable

The FSSB optical cables are used to interface up to four amplifiers to the PMM motion controller and to connect the PMM to the FTB. The fiber optic cables are available in various lengths and in two styles. For a list of fiber optic cables, see "Terminal Header and Cable Options" on page 3-8.

FTB to PMM Connection

Any FSSB cable listed in “Terminal Header and Cable Options” on page 3-8 can be used for the fiber optic connection between the PMM and the FTB.

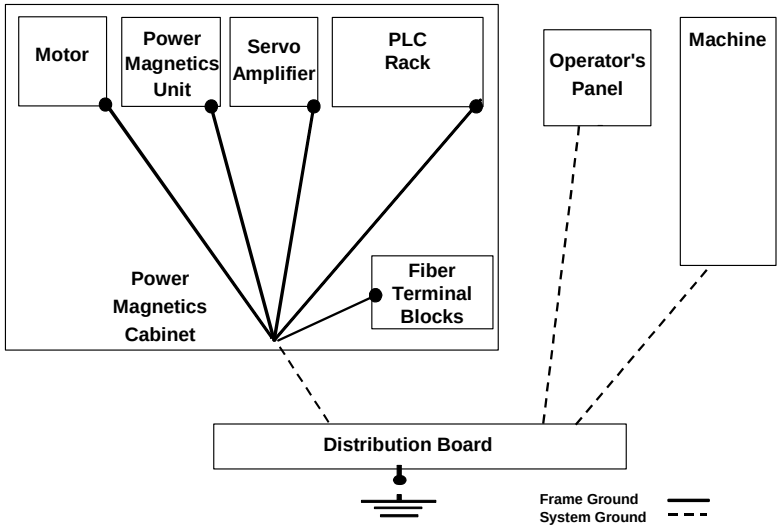


Grounding the PACMotion System

The motion system must be properly grounded. Many problems occur simply because this practice is not followed. To properly ground your system, you should follow these guidelines:

- The grounding resistance of the system ground should be 100 ohms or less (class 3 grounding).
- If an FTB is used, the ¼ inch blade terminal at the bottom of the block must be connected to the panel frame ground.
- 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. The following figure illustrates the grounding systems.
- The amplifier 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 “I/O Cable Grounding” on page 3-29.

System Grounding Connections



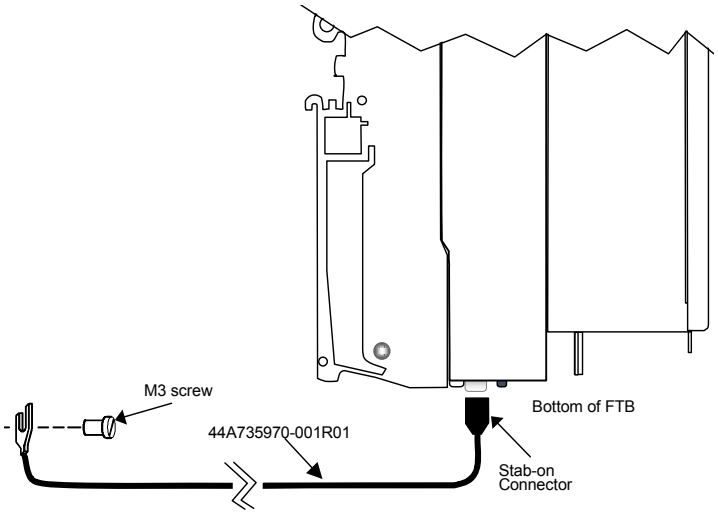
Grounding Systems

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.

Shield Ground Connection

Fiber I/O Terminal Block (FTB)

The FTB's faceplate shield 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.



I/O Cable Grounding

Properly routing signal cables, amplifier 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 100 ohm resistance to an earth ground. Consult local electrical codes and install in conformance to local regulations.

The specifications for completing βi, βHVi and αHVi Series Digital Servo amplifier installation and wiring, including amplifier grounding are described in the manual *Servo Product Specification Guide, GFH-001.*

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

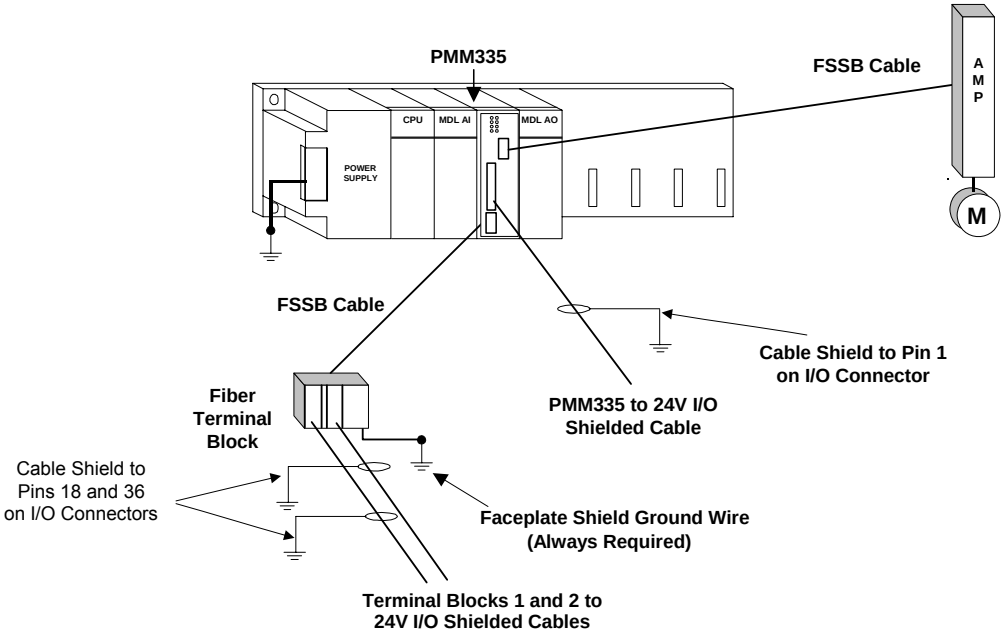
Group	Signal	Action
A	Amplifier input power Motor Power	Separate a minimum 10cm from group “B” signals by bundling separately or use electromagnetic shielding (grounded steel plate). Use noise protector for MCC.
B	FTB to 5 Volt I/O Terminal cable PMM or FTB to 24 Volt I/O Terminal cable Encoder feedback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

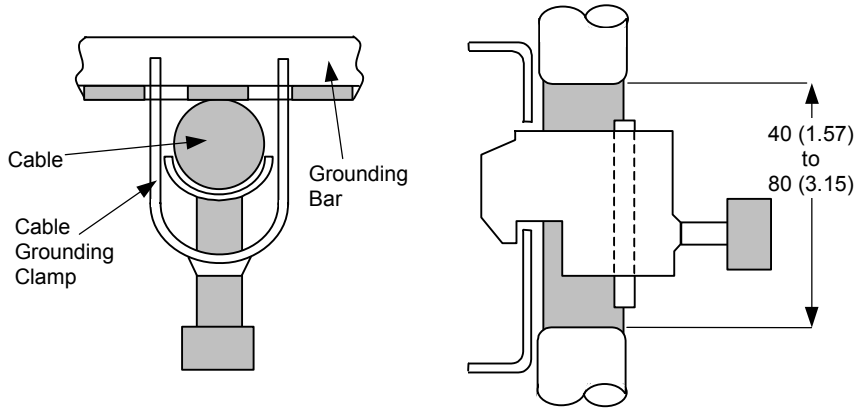
1. The PMM faceplate ground wire must be connected to a reliable panel ground.
2. The PMM I/O connector provides ground via pin 1. A short ground wire must be connected from each of these terminals to a reliable panel ground.
3. If an FTB is used, the FTB faceplate ground wire must be connected to a reliable panel ground.
4. 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.

I/O Cable Grounding Diagram

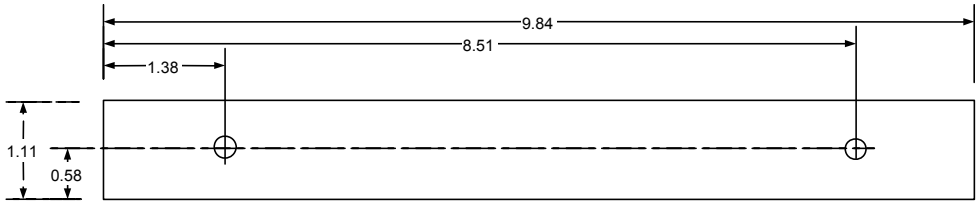


The Digital Servo amplifier encoder feedback cable requires a ZA99L-0035-0001 Cable Shield Grounding Clamp and one of the 11 available slots on a Z44B295864-001 Grounding Bar at the amplifier end of the cable. This clamp arrangement serves as a mechanical strain relief and as cable shield ground. The outer insulation of the Digital servo amplifier cable must be removed to expose the cable shield in the contact area of the clamp.

Z44B295864-001 Grounding Bar, Side View Dimensions



Z44B295864-001 Grounding Bar Dimensions, Rear View Showing Mounting Holes



5. 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.

Chapter 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 Logic Developer software version 5.9 or later.

Connecting the Programmer to the RX3i 4-2

Adding a PMM to the Hardware Configuration 4-3

Configuring PMM Parameters 4-4

- Settings 4-5
- I/O Function Assignments 4-8
  - FP Inputs 4-8
  - FTB Inputs 4-10
  - FTB Outputs 4-13
  - I/O Functions Summary 4-16
- I/O Interrupts 4-18
- Axis Configuration Data 4-20
  - Axis 1—Axis 4 4-25
  - Axis 5 4-32
- Advanced Data 4-52
- Power Consumption 4-52
- Terminals 4-52

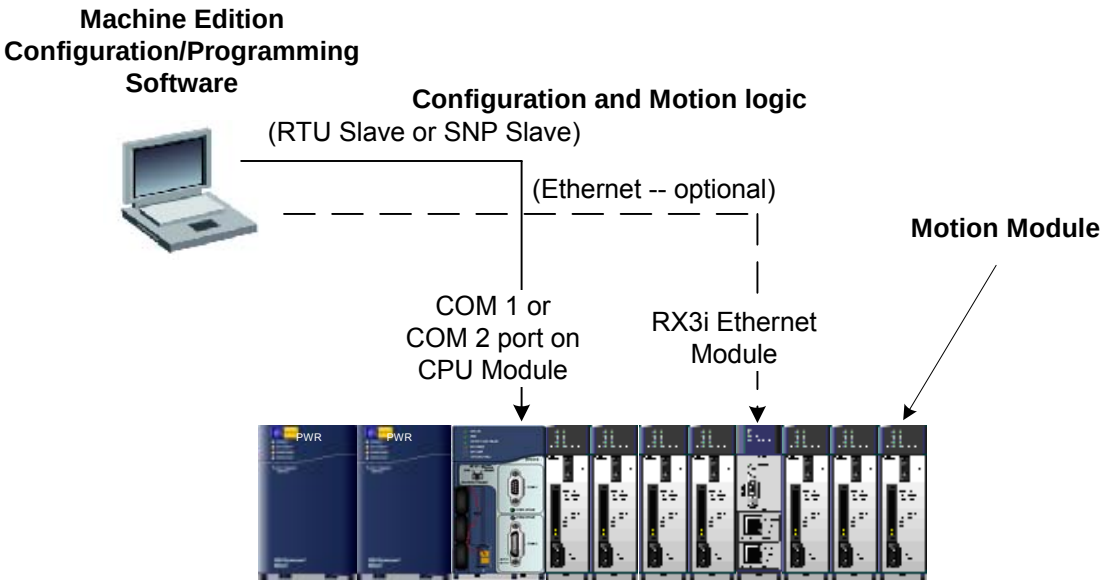
Connecting the Programmer to the RX3i

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 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 *TCP/IP Ethernet Communications for PACSystems,* GFK-2224.

PMM335 Programmer Connection



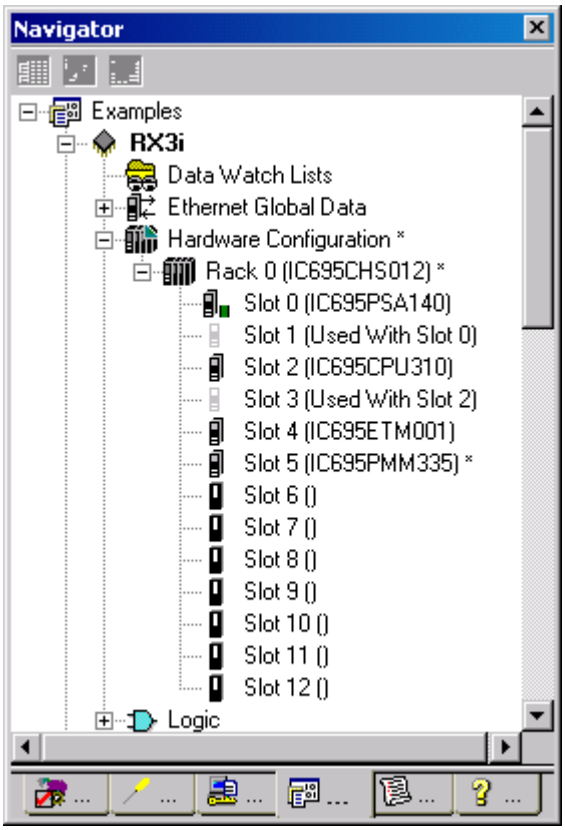
Adding a PMM to the Hardware Configuration

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 Machine Edition software:

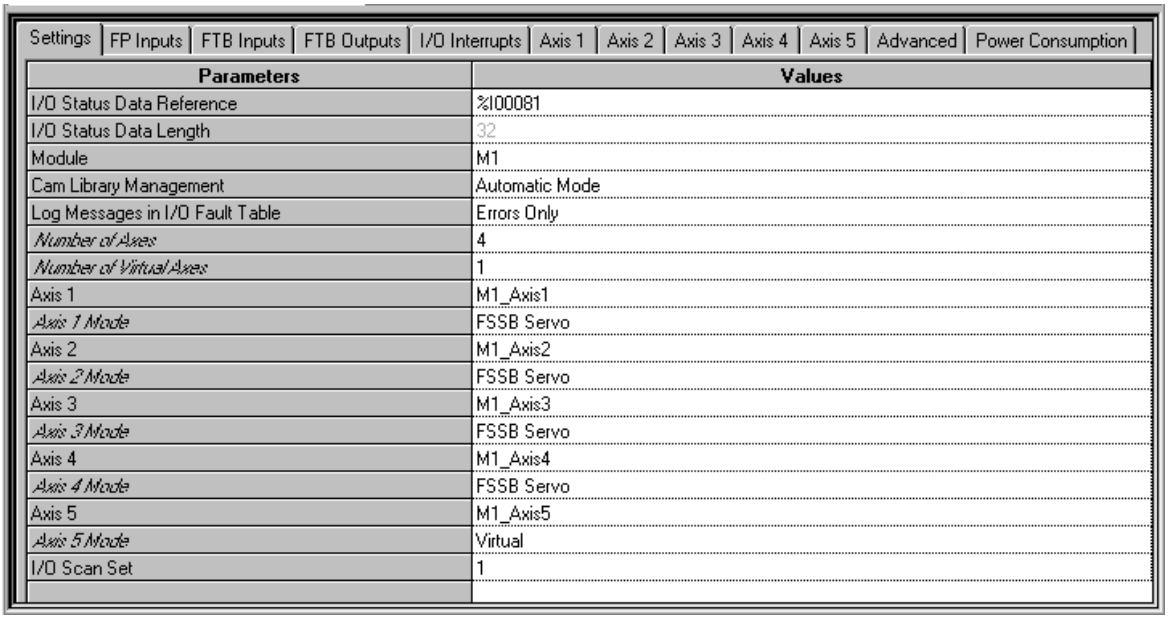
1. Create or open a project containing an RX3i target.
2. In the Navigator window, expand the Hardware Configuration.
3. If necessary, replace the power supply and/or CPU with the models that will be used in your application. To replace a module, right click and choose Replace Module.
4. Right click 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 PMM335 and click OK.

   This operation adds the PMM335 to the RX3i rack and displays the PMM335 configuration screens that allow you to customize the PMM335 to your particular application.



Configuring PMM Parameters

The 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 *Proficy Logic Developer Getting Started,* GFK-1918.



Settings | FP Inputs | FTB Inputs | FTB Outputs | I/O Interrupts | Axis 1 | Axis 2 | Axis 3 | Axis 4 | Axis 5 | Advanced | Power Consumption

Parameters	Values
I/O Status Data Reference	%I00081
I/O Status Data Length	32
Module	M1
Cam Library Management	Automatic Mode
Log Messages in I/O Fault Table	Errors Only
Number of Axes	4
Number of Virtual Axes	1
Axis 1	M1_Axis1
Axis 1 Mode	FSSB Servo
Axis 2	M1_Axis2
Axis 2 Mode	FSSB Servo
Axis 3	M1_Axis3
Axis 3 Mode	FSSB Servo
Axis 4	M1_Axis4
Axis 4 Mode	FSSB Servo
Axis 5	M1_Axis5
Axis 5 Mode	Virtual
I/O Scan Set	1

Parameters	***Function***	***Page***
Settings	Contains the status data reference assignment, axis selection and other module-level data.	4-5
I/O Function Assignments		4-8
FP Inputs	Used for configuring the PMM's faceplate inputs and outputs.	4-8
FTB Inputs	Used for configuring the Fiber I/O Terminal Block (FTB) inputs.	4-10
FTB Outputs	Used for configuring the FTB outputs.	4-13
I/O Interrupts	Used to specify interrupts that will trigger execution of a logic block in the CPU.	4-16
Axis Configuration	Used for configuring axis operational characteristics.	4-18
Axis 1—Axis 4	The Number of Axes parameter on the Settings tab determines how many axis tabs are displayed.	4-25
Axis 5	Displayed when *Number of Virtual Axes* on the Settings tab is set to 1.	4-32
Advanced	Contains advanced tuning data for axes 1 — 4.	4-52
Power Consumption	Lists PMM power required from backplane supply.	4-52
Terminals	Appears only 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.	4-52

***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 Chapter 8.

Settings

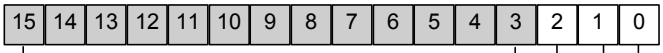
The Settings tab contains configuration information that allows you to define basic module operation. The configuration parameters in the Settings tab are described in the table below.

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 “PMM Status Data” on page 4-7. 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: M*x*, 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. ***Note*:** Error and Warning messages are recorded in the PMM’s event queue, regardless of whether they are logged in the fault table. 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	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. Default: M*x*_Axis*y*, where *x* is the module number and y is the axis number.
Axis 1 Mode, Axis 2 Mode, Axis 3 Mode, Axis 4 Mode	FSSB Servo selects the fiber optic digital command interface to the FANUC digital servo drives. Axes 1 through 4 must be enabled (set to FSSB Servo) in sequential order, starting with Axis 1. An axis can be disabled if you need to temporarily remove it from the configuration. Choices: Disabled, FSSB Servo Default: FSSB Servo
Axis 5 Mode	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. Choices: Disabled, Virtual Default: Virtual
I/O Scan Set	Assigns the module I/O status data to a scan set defined in the CPU configuration. The scan set determines how often the data will be polled by the RX3i. For more information, see the *PACSystems CPU Reference Manual,* GFK-2222. Range: 1 to 32 Default: 1 (Read status every I/O sweep.)

PMM Status Data

Least Significant Word



Status (parameter no.)

Reserved - always 0

Module Interrupt 3 (2003)

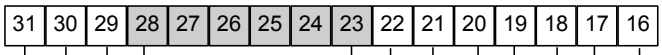
Module Interrupt 2 (2002)

Module Interrupt 1 (2001)

Set ON by the host controller CPU when it receives an interrupt from the PMM for the corresponding module interrupt.

Module interrupt bits are scanned as FALSE by the CPU, either during a normal I/O scan or a scan that is forced by a DO_IO or SCAN_SET_IO function. They are set for a maximum of one scan, and only during the scan in which the interrupt notification came in to the CPU.

Most Significant Word



Status (parameter no.)

Module Present (2032)

Set ON when the PMM is present.

This bit does not indicate that the module is capable of performing motion commands. The Axis OK and Module OK status bits must be used as appropriate to qualify enabling of function blocks in the application.

Module Synchronized (2031)

When ON, indicates that the module is synchronized with other PMMs in the backplane.

New Config Received (2030)

Set ON when the module has received and accepted a hardware configuration.

Reserved (always 0

Axis 5 OK (2023/1200)

Axis 4 OK (2022/1200)

Axis 3 OK (2021/1200)

Axis 2 OK (2020/1200)

Axis 1 OK (2019/1200)

Set ON when the corresponding axis is capable of performing motion (the axis is enabled in the hardware configuration and is not in the ErrorStop state).

Note: Axis OK status bits must be used to qualify enabling of function blocks that command axis motion, except for MC_JogAxis.

FTB OK (2018)

Set ON when the FTB is present and actve.

Module OK (2017)

Set ON when the module is capable of performing motion. This bit is set after the hardware configuration is successfully downloaded to the module, the DSCM is configured, the FTB is OK, and the module is synchronized.

Note: The Module OK status bit must be used to qualify enabling of motion function blocks in the application.

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 “Faceplate and FTB I/O Function Descriptions” on page 4-14.

The tables on pages 4-16 and 4-17 identify the functions that can be assigned to the I/O points on the PMM faceplate and on the FTB.

Notes: 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. *Note:* If IN2 is configured as Axis 5 Encoder Channel B, IN1 must be configured as Axis 5 Channel A.	3000 (Read)
FP IN1 Input Ref	Symbolic variable of type INPUT_REF, associated with this FP input. Default: M*x*_FP_IN*x*	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. *Note:* If FP IN1 is configured as Axis 5 Encoder Channel A, FP IN2 must be configured as Axis 5 Channel B.	3001 (Read)
FP IN2 Input Ref	Symbolic variable of type INPUT_REF, associated with this FP input. Default: M*x*_FP_IN*x*	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)
FP IN3 Input/Output Ref	Symbolic variable of type INPUT_REF, associated with this FP input. Default: M*x*_FP_IN*x* or M*x*_FP_OUT*x*	NA

Configuration Parameter	*Description*	*Parameter Number*
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: M*x*_FP_IN*x* or M*x*_FP_OUT*x*	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: M*x*_FP_IN*x*	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: M*x*_FP_IN*x*	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: M*x*_FP_IN*x*	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: M*x*_FP_IN*x*	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. ***Note:*** When communication with the CPU is lost, FTB outputs are set to 0. 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 I/O Terminal Block. Four differential 5V 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 page 4-17.

For explanations of these functions, see "Faceplate and FTB I/O Function Descriptions" on page 4-14.

Notes: 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 can not 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. Set to Enabled when an FTB will be connected to the PMM. 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 Input Ref — FTB IN16 Input Ref	Symbolic variable of type INPUT_REF, associated with this FTB input. Default: Mx_FTB_IN*x*	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_IN*x*	NA
FTB IN17 Mode — FTB IN23 Mode	Specifies the communications connection method for the 5VDC inputs FTB IN17 through FTB IN23. 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: M*x*_FTB_IN*x*	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 24V outputs on the Fiber I/O Terminal Block.

Configuration Parameter	***Description***	***Parameter Number***
FTB OUT1—FTB OUT8	24V Output.	3160—3137 (Read/Write)
FTB OUT1 Ref — FTB OUT8 Ref	Variable name associated with this FTB output. Default: M*x*_FTB_OUT*y*. Where *x* 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. ***Note:*** 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	Selects the signal that is output to analog output 1.	3288 (Read/Write)
FTB ALGOUT2 Ref	Selects the signal that is output to analog output 2.	3289 (Read/Write)

Faceplate and FTB I/O Function Descriptions

For details on the wiring of I/O connections, refer to Chapter 3, “Installation and Wiring.” For a summary of I/O functions that can be assigned to the faceplate and FTB I/O points, refer to pages 4-16 and 4-17.

Fast Digital Input

A 24VDC input with high speed input filtering.

Input Type	Input Filtering
FTB 24VDC	500µs maximum
Faceplate General Purpose 24VDC	500µs maximum
Faceplate High Speed 24VDC	1µs 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 Touch Probe 1 – Axis 5 Touch Probe 1 and Axis 1 Touch Probe 2 – Axis 5 Touch Probe 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.

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

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 Device Velocity

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 2 Overtravel +, Axis 3 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, see on page Over Travel Limit Switch on page 4-41.

Axis 1 Overtravel –, Axis 2 Overtravel –, Axis 3 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.

Home Switch 1—Home Switch 4

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.

Digital Output

General-purpose 24V output. (OUT1 and OUT2 on the PMM faceplate.)

Fast Digital Output

5-Volt differential output used for general purpose signaling.

24V Output

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

Faceplate Digital I/O Functions Summary

Legend:

Y	Allowed
A	Allowed on all inputs
	Not allowed

Notes:

- Except for Fast Digital Input, Digital Input and Digital Output, each function can be applied to only one FP or FTB point at a time.
- 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 FP and FTB inputs.
- If an input is selected as an I/O Interrupt Source, a Digital Output or Axis Encoder input cannot be assigned to that point.

Function	IN1	IN2	IN3/OUT1	IN4/OUT2	IN5	IN6	IN7	IN8
Fast Digital Input	Y	Y	Y	Y	Y	Y	Y	Y
Digital Input	Y	Y	Y	Y	Y	Y	Y	Y
Axis 1 Touch Probe 1	A	A	A	A	A	A	A	A
Axis 1 Touch Probe 2	A	A	A	A	A	A	A	A
Axis 2 Touch Probe 1	A	A	A	A	A	A	A	A
Axis 2 Touch Probe 2	A	A	A	A	A	A	A	A
Axis 3 Touch Probe 1	A	A	A	A	A	A	A	A
Axis 3 Touch Probe 2	A	A	A	A	A	A	A	A
Axis 4 Touch Probe 1	A	A	A	A	A	A	A	A
Axis 4 Touch Probe 2	A	A	A	A	A	A	A	A
Axis 5 Touch Probe 1	A	A	A	A	A	A	A	A
Axis 5 Touch Probe 2	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 Overtravel +,	Y		Y		Y		Y	
Axis 2 Overtravel +	Y		Y		Y		Y	
Axis 3 Overtravel +	Y		Y		Y		Y	
Axis 4 Overtravel +	Y		Y		Y		Y	
Axis 1 Overtravel –		Y		Y		Y		Y
Axis 2 Overtravel –		Y		Y		Y		Y
Axis 3 Overtravel –		Y		Y		Y		Y
Axis 4 Overtravel –		Y		Y		Y		Y
Axis 1 Home Switch	Y	Y	Y	Y	Y	Y	Y	Y
Axis 2 Home Switch	Y	Y	Y	Y	Y	Y	Y	Y
Axis 3 Home Switch	Y	Y	Y	Y	Y	Y	Y	Y
Axis 4 Home Switch	Y	Y	Y	Y	Y	Y	Y	Y
Digital Output			Y	Y				
Fast Digital Output								
24V Output								

FTB Digital I/O Functions Summary

Function	IN1 — 16	IN17	IN18	IN19	IN20	IN21	IN22	IN23	IN24/Out9	IN25/Out10	IN26	IN27/Out11	IN28/Out12	Out1—8
Fast Digital Input	Y	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	
Digital Input	**Y**													
Axis 1 Touch Probe 1	A	A	A	A	A	A	A	A	A	A	A	A	A	
Axis 1 Touch Probe 2	A	A	A	A	A	A	A	A	A	A	A	A	A	
Axis 2 Touch Probe 1	A	A	A	A	A	A	A	A	A	A	A	A	A	
Axis 2 Touch Probe 2	A	A	A	A	A	A	A	A	A	A	A	A	A	
Axis 3 Touch Probe 1	A	A	A	A	A	A	A	A	A	A	A	A	A	
Axis 3 Touch Probe 2	A	A	A	A	A	A	A	A	A	A	A	A	A	
Axis 4 Touch Probe 1	A	A	A	A	A	A	A	A	A	A	A	A	A	
Axis 4 Touch Probe 2	A	A	A	A	A	A	A	A	A	A	A	A	A	
Axis 5 Touch Probe 1	A	A	A	A	A	A	A	A	A	A	A	A	A	
Axis 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 Overtravel +	Y													
Axis 2 Overtravel +	Y													
Axis 3 Overtravel +	Y													
Axis 4 Overtravel +	Y													
Axis 1 Overtravel –	Y													
Axis 2 Overtravel –	Y													
Axis 3 Overtravel –	Y													
Axis 4 Overtravel –	Y													
Axis 1 Home Switch	Y													
Axis 2 Home Switch	Y													
Axis 3 Home Switch	Y													
Axis 4 Home Switch	Y													
Digital Output														
Fast Digital Output									Y	Y		Y	Y	
24V Output														Y

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	**Disabled** - Interrupts are not used. **FP Input** - 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. **FTB Input** - 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. **Timed** - 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	
I/O Interrupt *n* Source ID	Available when I/O Interrupt n Config is set to FP Input or FTB Input. 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 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	
I/O Interrupt *n* Detection	Available when I/O Interrupt n Config is set to FP Input or FTB Input Negative Edge Trigger Positive Edge Trigger Default: Positive Edge Trigger Note that the maximum frequency for each configured I/O interrupt is once per 2 mS. Interrupts occurring at a faster rate will be ignored and a corresponding fault will be generated.	
I/O Interrupt *n* Interval (ms)	Available when I/O Interrupt n Config is set to Timed. Specifies the interval at which the interrupt will be generated. For example, a value of 2 results in the interrupt occurring every 2 ms. Range: 2.0—40.0 Default: 10.0 ms	

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 Chapter 3.

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 starting 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 page 4-7.

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 Chapter 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 Axis 1— 4 parameters, see page 4-25. Axis 5 parameters are summarized on page 4-32.

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 Chapter 8.

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” on page 4-22.

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 ≤ 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 millimeter and 254 user units would represent 25.4 millimeters (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.

1. Determine the number of User Units per spur gear revolution:

   14.336 inch pitch diameter × π (pi) = 45.0378 inches circumference

   45.0378 inches / 0.001 inch desired programming units = 45037.8 User Units per revolution of spur gear

2. Determine the number of encoder counts per spur gear revolution:

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

3. Check the value of the User Units to Counts ratio. The two numbers and their ratio must be within their allowed ranges, listed on page 4-20.

   45037.8 User Units / 160,000 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 on page 4-23. 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**	4.0×10^{10}	Counts
Maximum Velocity	**MaxVelLim**	1.1×10^{8}	Counts/Sec
Maximum Acceleration	**MaxAccLim**	1.0×10^{12}	Counts/Sec^2
Maximum Jerk	**MaxJerkLim**	1.0×10^{15}	Counts/Sec^3

Formulas for Computing Maximum Data Limits in User Units

Feedback Source	*Limit*	*Maximum Data Limit*	*Formula*
Motor Encoder	Position	MaxPosnUuMtr =	$\text{MaxPosnLim} * \frac{\text{MotorEncoderUserUnits}}{\text{MotorEncoderCounts}}$
	Velocity	MaxVelUuMtr =	$\text{MaxVelLim} * \frac{\text{MotorEncoderUserUnits}}{\text{MotorEncoderCounts}}$
	Acceleration	MaxAccUuMtr =	$\text{MaxAccLim} * \frac{\text{MotorEncoderUserUnits}}{\text{MotorEncoderCounts}}$
	Jerk	MaxJerkUuMtr =	$\text{MaxJerkLim} * \frac{\text{MotorEncoderUserUnits}}{\text{MotorEncoderCounts}}$
External Quadrature Encoder	Position	MaxPosnUuExt=	$\text{MaxPosnLim} * \frac{\text{ExternalDeviceUserUnits}}{\text{ExternalDeviceCounts}}$
	Velocity	MaxVelUuExt =	$\text{MaxVelLim} * \frac{\text{ExternalDeviceUserUnits}}{\text{ExternalDeviceCounts}}$
	Acceleration	MaxAccUuExt =	$\text{MaxAccelLim} * \frac{\text{ExternalDeviceUserUnits}}{\text{ExternalDeviceCounts}}$
	Jerk	MaxJerkUuExt=	$\text{MaxJerkLim} * \frac{\text{ExternalDeviceUserUnits}}{\text{ExternalDeviceCounts}}$
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	$= 0.1 \times \frac{\text{MotorEncoderCountsperRev} * \text{Motor EncoderUu}}{\text{MotorEncoderCounts}}$
	Velocity	MinVelUuMtr	$= \frac{1}{600} \times \frac{\text{Motor Encoder Counts per Revolution} * \text{Motor Encoder Uu}}{\text{MotorEncoderCounts}}$
	Acceleration	MinAccUuMtr	$= 0.1 \times$ MinVelUuMtr
	Jerk	MinJerkUuMtr	$= 0.1 \times$ MinAccUuMtr
External Quadrature Encoder*	Position	MinPosnUuExt	$= 0.1 \times \frac{\text{External Device CountsperRev} * \text{ExternalDeviceUu}}{\text{External Device Counts}}$
	Velocity	MinVelUuExt	$= \frac{1}{600} \times \frac{\text{External Device CountsperRev} * \text{ExternalDeviceUu}}{\text{External Device Counts}}$
	Acceleration	MinAccUuExt	$= 0.1 \times$ MinVelUuExt
	Jerk	MinJerkUuExt	$= 0.1 \times$ MinAccUuExt
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	$= \frac{1}{60000} \times$ Command Counts per Motor Revolution $\times$ Command Position Resolution

* *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	$Revs = UU * \frac{1}{Motor\ Encoder\ Counts\ per\ Rev} * \frac{Motor\ Encoder\ Counts}{Motor\ Encoder\ User\ Units}$
	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	$Revs = UU * \frac{1}{External\ Device\ Counts\ per\ Rev} * \frac{External\ Device\ Counts}{External\ Device\ User\ Units}$
	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	$RPM = UU/sec * \frac{1}{External\ Device\ Counts\ per\ Rev} * \frac{External\ Device\ Counts}{External\ Device\ User\ Units} * 60$
Axis 5 Path Generator	Revs to UU	$UU = Revs * Command\ Counts\ per\ Motor\ Rev * Command\ Position\ Resolution$
	UU to Revs	$Revs = UU * \frac{1}{Command\ Counts\ per\ Motor\ Rev} * \frac{1}{Command\ Position\ Resolution}$
	RPM to UU/sec	$UU/sec = RPM * Command\ Counts\ per\ Motor\ Rev * Command\ Position\ Resolution * \frac{1}{60}$
	UU/sec to RPM	$RPM = UU/sec * \frac{1}{Command\ Counts\ per\ Motor\ Rev} * \frac{1}{Command\ Position\ Resolution} * 60$

Axis 1 through Axis 4 Parameters

For virtual axis (Axis 5) parameters, see page 4-32.

Configuration Parameter	***Description***	***Units***	***Page 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	4-35	NA
Position Feedback Source	Selects the position feedback source for the axis. Choices: Motor Encoder, External Device Default: Motor Encoder	NA	4-36	NA
Axis Positioning Mode	Selects the positioning mode for the axis. Choices: Linear, Rotary Default: Linear	N/A	4-36	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	4-36	NA
Motor Encoder Low Battery Alarm	If Motor Encoder Mode is set to Absolute, enables or disables the motor encoder low battery alarm. Choices: Enabled, Disabled Default: Enabled	NA	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. ***Note:*** User Units must be ≤ Counts. Range: 0.000001 — 131,072 Default: 1.0	N/A	4-36	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	4-36	1001 (Read/Write)

Configuration Parameter	Description	Units	Page 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: MinPosnUuMtr* High Limit: MaxPosnUuMtr* *Motor Encoder Low Position Limit + Motor Encoder Position Range ≤ MaxPosnUuMtr* Default: 16777216.0	Uu	4-36	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: -MaxPosnUuMtr* Low Limit: MaxPosnUuMtr* - MinPosnUuMtr* *–MaxPosnUuMtr ≤ Motor Encoder Low Position Limit ≤ MaxPosnUuMtr* Default: -8388608	Uu	4-37	1003 (Read/Write)
Motor Encoder Counts per Motor Revolution	Selects encoder resolution. Range: 100 — 1,048,576 Must be ≤ the maximum encoder resolution supported by the *Motor Type.* Default: 8192.0	Cts	4-37	NA
Motor Encoder Maximum Positive RPM Limit	Motor encoder maximum positive RPM limit. Range: 10 — 10,000 Default: 8191	RPM	4-37	NA
Motor Encoder Maximum Negative RPM Limit	Motor encoder maximum negative RPM limit. Range: 10 — 10,000 Default: 8191	RPM	4-37	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	4-38	NA

Configuration Parameter	Description	Units	Page Ref.	Parameter Number
External Device User Units	Available only if External Device is set to External Quadrature Encoder. 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. ***Note:*** User Units must be ≤ Counts. Range: 0.000001 — 65,536.0 Default: 1.0	N/A	4-40	1004 (Read/Write)
External Device Counts	Available only if External Device is set to External Quadrature Encoder. 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	4-40	1005 (Read/Write)
External Device Position Range	Available if External Device is set to External Quadrature Encoder and 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: MinPosnUuExt* High limit: MaxPosnUuExt* *External Device Low Position Limit + External Device Position Range ≤ MaxPosnUuExt* Default: 16,777,216.0	Uu	4-40	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: -MaxPosnUuExt* — MaxPosnUuExt* *–MaxPosnExt ≤ Command Low Position Limit ≤ MaxPosnExt* Default: -8,388,608	Uu	4-40	1007 (Read/Write)
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. Default: 8,192	Counts	4-40	NA

Configuration Parameter	Description	Units	Page Ref.	Parameter Number
Over Travel Limit Switch	Over travel Limit Switch Enable / Disable Ignored by axes using Motor Type 65535. Choices: Enabled, Disabled Default: Enabled	N/A	4-41	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	4-41	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	4-41	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* 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	4-42	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* 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	4-42	3 (Read/Write)
Max Velocity System	Specifies the maximum axis velocity. Low limit: 0.1 High limit: Maximum velocity of Motor Type Default: 4,000.0	RPM	4-42	8 (Read)
Max Acceleration System	Specifies the maximum rate of velocity increase. Range: 0.01 — 60E12 / Feedback Device Counts per Rev Default: 40,000	RPM/sec	4-42	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	4-43	14 (Read)

Configuration Parameter	Description	Units	Page Ref.	Parameter Number
Max Jerk	Specifies the maximum rate of change in acceleration or deceleration. Low limit: MinJerkUu* High limit: MaxJerkUu* Default: 100,000,000,000,000.0 (1E14)	Uu/sec^3	4-43	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	4-43	NA
Motor Type	Motor type code identifying the motor attached to the axis. Range: 0—65535 Default: 65535	N/A	4-43	NA
Motor Velocity Limit	Specifies the maximum speed of the motor. Range: 60 — 1.1 x max. velocity of configured Motor Type Default: 4,000	RPM	4-45	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	%	4-45	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	4-45	6 (Read/Write)
Max Position Lag	Maximum position lag allowed when controlling the servo. *Position Lag = Commanded Position - Actual Position* Low Limit = 0.001 × MaxVelocitySystem (RPM) × 1/60 min/s × Counts per Rev × UU/Counts High Limit = 100 × (Counts per /Rev) × (Uu/Counts) *Must be ≤ Max Position Error × Counts per Rev × Uu/Count.* Default: 500,000.0 *Range is determined by the Counts per Rev, Uu and counts for the configured Position Feedback Source.*	Uu	4-45	7 (Read/Write)

Configuration Parameter	Description	Units	Page Ref.	Parameter Number
Max Position Error	An absolute value used to determine when the servo is out of sync and should be stopped. Low limit = 0.002 × MaxVelocitySystem (RPM) × 1/60 High limit = 100 Default: 10	Motor Revs	4-46	1016 (Read)
In Position Zone	Maximum allowed position error. If the position error is less than this value, the axis is considered to be in the position zone. Range: 0.0—60000.0 × (Uu/Counts) *Range is determined by the Uu and counts for the configured Position Feedback Source.* Default: 10.0	Uu	4-46	1008 (Read/Write)
Position Loop Time Constant	Determines the response speed of the closed position loop. Range: 0.0—1,0000.0 Default: 100.0	ms	4-47	1009 (Read/Write)
Velocity Feedforward	Specifies the percentage of commanded velocity that is added to the PMM's position loop velocity command output. Range: 0.0–120.0 Default: 0.0	%	4-47	1010 (Read/Write)
Load Inertia Ratio (256 = 1.1)	Specifies the gain applied to the velocity control loop to match load inertia to motor inertia. Ignored by axes using Motor Type 65535. Range: 0–4096 Default: 128	NA	4-48	10032 (Read/Write)
Error Stop Deceleration	Maximum deceleration allowed for a normal stop. Low limit: MinAccUu* High limit: The lesser of *MaxAccUu* or *Max Deceleration System* in Uu/sec^2 Default: 1,000,000.0	Uu/sec^2	4-48	1013 (Read/Write)
Error Stop Jerk	Maximum jerk allowed for a normal stop. Low limit: MinJerkUu* High limit: The lesser of MaxJerkLim or MaxJerkUu* Default: 10,000,000.0	Uu/sec^3	4-48	1014 (Read/Write)
Master Axis Velocity Filter	Specifies filter width of the master axis velocity signal. Valid values are 1, 2, 4, 8, 16 and 32. Default: 8	ms	4-48	1321 (Read/Write)

Configuration Parameter	Description	Units	Page Ref.	Parameter Number
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	4-49	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	4-49	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	4-50	1312 (Read/Write)
Disabled Direction Deadband	Specifies a deadband range in the direction not enabled by MC_POWER. Low limit: 0 High limit: MinPosnUu*	Uu	4-51	1313 (Read/Write)

*For definitions of MaxAccUu, MaxPosnUu, MaxVelUu and MaxJerkUu, see “Computing Data Limits“ on page 4-22.

Axis 5 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***	***Page***	***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	4-35	NA
Axis Positioning Mode	Selects the linear or rotary positioning mode for the axis 5 path generator. Choices: Linear, Rotary Default: Linear	NA	4-36	NA
Command Position Resolution	Specifies the resolution of the Axis 5 path generator. Range: 0.0000001 – 1.0 Default: 1.0	Uu	4-51	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* High limit: MaxPosnCmd* *Command Low Position Limit + Command Position Range must be ≤ MaxPosnCmd* Default: 1,6777,216	Uu	4-51	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* High Limit: MaxPosnCmd* - MinPosnCommand* *-MaxPosnCmd ≤ Command Low Position Limit ≤ MaxPosnCmd* Default: -8,388,608	Uu	4-51	1023
Command Counts Per Motor Revolution	Axis 5 Path Generator resolution Choices: 65,536, 131,072, 1,048,576 Default: 65,536	Cts	4-51	NA
External Device	Selects an optional external feedback source. Choices: None, External Quadrature Encoder Default: None.	NA	4-38	NA

Configuration Parameter	Description	Units	Page	Parameter Number
External Device User Units	Used to calculate the Uu/Cts ratio, a scaling factor that relates user programming units to encoder counts. ***Note:*** User Units must be ≤ Counts. Range: 0.000001 — 65,536.0 Default: 1.0	N/A	4-40	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	4-40	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 × (External encoder Uu/Counts) High limit: MaxPosnUuExt* Default: 16777216.0	Uu	4-40	1006 (Read/Write)
External Device Low Position Limit	Specifies the lower limit of the axis actual position. Low Limit: -MaxPosnUuExt* High Limit: MaxPosnUuExt* - MinPosnUuExt* Default: -8388608	Uu	4-40	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	4-40	NA
Axis Direction	Defines the positive axis direction as counter-clockwise motor direction (Normal) or clockwise motor direction (Reverse). Choices: Normal, Reverse Default: Normal	NA	4-41	NA
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	4-41	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 ≥ Low Software EOT Limit* Default: +8388607.0	Uu	4-42	2 (Read/Write)

Configuration Parameter	Description	Units	Page	Parameter Number
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 ≤ High Software EOT Limit* Default: -8388608.0	Uu	4-42	3 (Read/Write)
Max Velocity System	Specifies the maximum Path Generator axis velocity. Range: 0.1 — 6,000 Default: 4,000.0	RPM	4-42	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^2	4-42	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	4-43	14 (Read)
Max Jerk	Specifies the maximum Path Generator rate of change in acceleration or deceleration. Range: MinJerkCmd* — MaxJerkCmd* Default: 100,000,000,000,000.0 (1E15)	Uu/sec^3	4-43	16 (Read/Write)
Error Stop Deceleration	Maximum Path Generator deceleration allowed for a normal stop. Low limit: MinAccUu High limit: The lesser of *MaxAccCmd** or *Max Deceleration System* in Uu/sec^2 Default: 1,000,000.0	Uu/sec^2	4-48	1013 (Read/Write)
Error Stop Jerk	Maximum Path Generator jerk allowed for a normal stop. Low limit: MinJerkUu* High limit: The lesser of MaxJerkLim or MaxJerkCmd* Default: 10,000,000.0	Uu/sec^3	4-48	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	4-49	1024 (Read/Write)

*For definitions of MaxAccUu, MaxPosnUu, MaxVelUu and MaxJerkUu, see “Computing Data Limits“ on page 4-22.

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 5V input (unless input is for quadrature encoder)	Generates a warning for the associated axis.
Open load on any 24V output	
Short circuit on any 24V 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 5V 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 5V 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 5V inputs are used for slave axis 1—4 quadrature feedback.	Generates a fast stop error for the associated axis.
An FTB communications fault if 5V 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 on pages 4-22 and 4-23.

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.

Motor Encoder Mode

Selects the Incremental or Absolute mode for the Fanuc serial 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, see “Digital Encoder Modes” in 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” on page 4-20.

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 2 ms).

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 Counts per Motor Revolution

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

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

If set to External Quadrature Encoder, you can control the servo position loop of the axis from a position feedback device (external quadrature encoder) instead of the position encoder mounted on the motor. The standard motor feedback from the FANUC serial encoder will be used for motor commutation and feedback for the remaining loops.

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

The PMM supports up to five external quadrature encoders with a Marker input (also called the *index* or *Z* channel) for each. If a Find Home Cycle is to be used, the encoder marker signal input channel must be configured on the FTB Inputs tab (or FP Inputs tab if Axis 5) and connected to the external quadrature encoder.

The external quadrature encoder must be wired in phase with the encoder that is in use on the FANUC motor. A procedure for verifying that the encoders are wired correctly is provided on page 4-40.

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: Ext Quadrature Encoder when the external quadrature encoder is not mechanically connected is a potentially hazardous situation.

Note: If Position Feedback Source is set to Motor Encoder, and External Quadrature Encoder is selected, 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.

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.

PMM335 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 24V 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 on pages 4-16 and 4-17.

Verifying Correct Quadrature Encoder/Digital Encoder Phase

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

1. Wire the external quadrature encoder to Axis 1.
2. Set the *Position Feedback Source* parameter to *External Device.*
3. Set the *External Device* parameter to *External Quadrature Encoder.*
4. Turn off the position loop by setting the *Position Loop Time Constant* to zero.
5. Set velocity feed-forward to 100%.
6. Use the MC_JogAxis function block to jog the axis.
7. 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.
8. 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” on page 4-20.

External Device Position Range

Specifies the axis’ actual position range.

External Device Low Position Limit

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

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.

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 milliseconds 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 (motor type 65535) 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 FANUC digital 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, see “Computing Data Limits” on page 4-22.

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.

Motor Type

Selects the type of FANUC AC servomotor to be used with the PMM. The PMM stores setup motor parameter tables for each of the FANUC motors supported.

Either FANUC motor model numbers or motor part numbers can be used to determine the motor type code. FANUC motor part numbers, found on the motor label, are in the form ZA06B-*xxxx*-yyyy, where *xxxx* is a four-digit motor type specification that can be used to look up the Motor Type code configuration value. For a listing of motor types, refer to “Motor Type Codes” on page 4-44.

The default value of 65535 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.

Using Motor Type to Temporarily Prevent FSSB Errors

Any fault that causes Axis OK to be off disables motion command processing for that axis. If a configured servo drive is not present and powered, all of the axes will be faulted and unable to accept motion commands. In this case, the Motor Type of the affected axis can be temporarily set to 65535. This disables servo loop processing and FSSB communication checks, but prevents errors and permits processing function blocks that are supported by the synthetic motor.

Motor Type Codes

Either FANUC motor model numbers or motor part numbers can be used to determine the motor type code. FANUC motor part numbers, found on the motor label, are in the form ZA06B-xxxx-yyyy, where xxxx is a four-digit Motor Specification that can be used to look up the Motor Type code configuration value in the following table.

Motor Model	*Motor Specification*	*Motor ID (Type Code)*	*Maximum Speed (rpm)*	*Maximum Encoder Resolution (cts/rev)*
α2/6000HVis	ZA06B-*0212*-B#0#	284	6000	1048576
α4/5000HVis	ZA06B-*0216*-B#0#	266	5000	1048576
α8/6000HVis	ZA06B-*0233*-B#0#	292	6000	1048576
α12/4000HVis	ZA06B-*0239*-B#0#	289	4000	1048576
α22/3000i	ZA06B-*0247*-B#0#	297	3000	1048576
α22/3000Hvi	ZA06B-*0249*-B#0#	299	3000	1048576
α22/4000Hvis	ZA06B-*0266*-B#0#	316	4000	1048576
α30/4000Hvis	ZA06B-*0269*-B#0#	319	4000	1048576
α40/4000Hvis	ZA06B-*0273*-B#0#	323	4000	1048576
α50/3000HVis FAN	ZA06B-*0276*-B#1#	326	3000	1048576
α100/2500HVis FAN	ZA06B-*0286*-B#1#	336	2500	1048576
β0.2/5000iS	ZA06B-*0111*-B#0#	260	5000	65536
β0.3/5000iS	ZA06B-*0112*-B#0#	261	5000	65536
β0.4/5000iS	ZA06B-*0114*-B#0#	280	5000	65536
β0.5/6000iS	ZA06B-*0115*-B#0#	281	6000	65536
β1/6000iS	ZA06B-*0116*-B#0#	282	6000	65536
β2/4000is	ZA06B-*0061*-B#0#	253	4000	131072
β2/4000HVis	ZA06B-*0062*-B#0#	251	4000	131072
β4/4000is	ZA06B-*0063*-B#0#	256	4000	131072
β4/4000HVis	ZA06B-*0064*-B#0#	264	4000	131072
β8/3000is	ZA06B-*0075*-B#0#	258	3000	131072
β8/3000HVis	ZA06B-*0076*-B#0#	267	3000	131072
β12/3000is	ZA06B-*0078*-B#0#	272	3000	131072
β12/3000HVis	ZA06B-*0079*-B#0#	270	3000	131072
β22/2000is	ZA06B-*0085*-B#0#	274	2000	131072
β22/2000HViS	ZA06B-*0086*-B#0#	278	2000	131072
Synthetic Motor*	NA	65535	6000	1048576

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 velocity refer to the Rated Speed column in the table on page 4-44.

Torque Limit

Specifies the maximum allowed torque, in percent of available torque, to be produced by the servo motor at commanded velocity. To determine the actual value of torque output available at a given velocity, refer to the motor torque curve in the appropriate servo motor manual. A simple example of the use of Torque Limit would be to prevent over-tightening on a machine.

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:

Low limit = 0.001 × MaxVelocitySystem (RPM) × (1/60 min per sec)
× feedback encoder counts per rev × uu/cts

High limit = 100 × feedback encoder counts per rev × 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:

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

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:

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

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

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 (milliseconds) 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 100msec, the Position Loop Time Constant should be between 25 to 50msec. To maintain system stability, use the largest value possible.

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

Bandwidth (radians/sec) = 1000 / Position Loop Time Constant (ms)

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

Gain (ipm/mil) = 60 / 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 an non-zero value. The servo system capabilities will determine the optimum value. Refer to Appendix C, ***Startup and Tuning of a FSSB Servo Drive*** for more information. If *Velocity Feedforward* is changed, *Max Position Lag* may require adjustment.

Load Inertia Ratio

Applies a gain to the velocity control loop to match load inertia (J_L) to motor inertia (J_M) for a FANUC Digital Servo. 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.

Beta *is* motors have higher velocity loop bandwidth than other motors. If load resonance or high frequency oscillation (above 200 Hz) is experienced, the load inertia ratio should be set to a value in the range of ⅓ to ½ the value calculated by the formula.

Refer to "Resonance Elimination Filters" in Appendix D for information about optional parameter settings that may be useful when large inertial loads are driven by a motor.

This parameter does not apply to axes using Synthetic Motors (motor type 65535).

Error Stop Deceleration

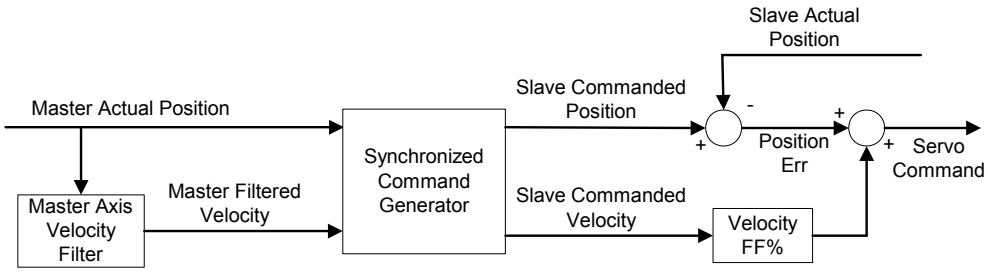
Specifies the maximum deceleration allowed when the axis undergoes an error stop. *Normal Stop Deceleration* must be ≤ *Deceleration Limit*.

Error Stop Jerk

Specifies the maximum jerk allowed during an error stop of the axis. *Error Stop Jerk* must be ≤ *Max Jerk*.

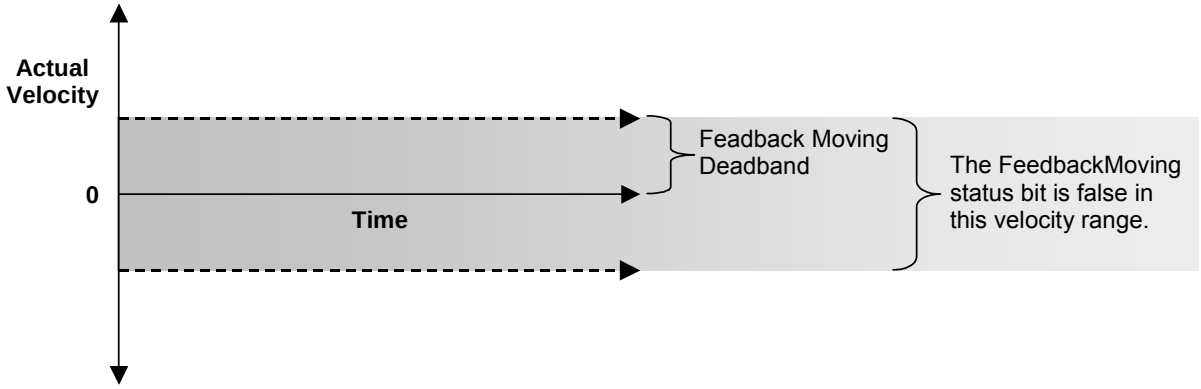
Master Axis Velocity Filter

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.



Feedback Moving Deadband

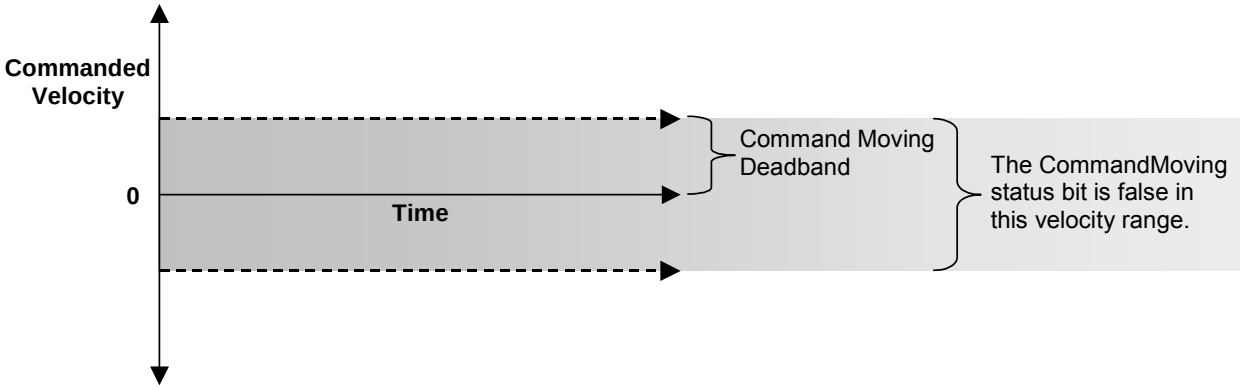
Specifies a deadband range in UU/sec for setting the actual motor velocity status bit, FeedbackMoving. When the actual motor velocity is within this range, the axis status bit, FeedbackMoving is ON.



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 axis' commanded velocity, this deadband introduces lag in setting the these bits. Unless the axis is using synchronous motion (the axis is following a master), Command Moving Deadband should be set to zero.

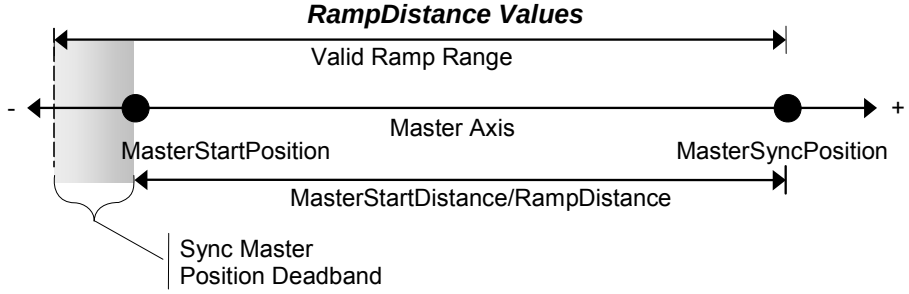


Sync Master Position Deadband

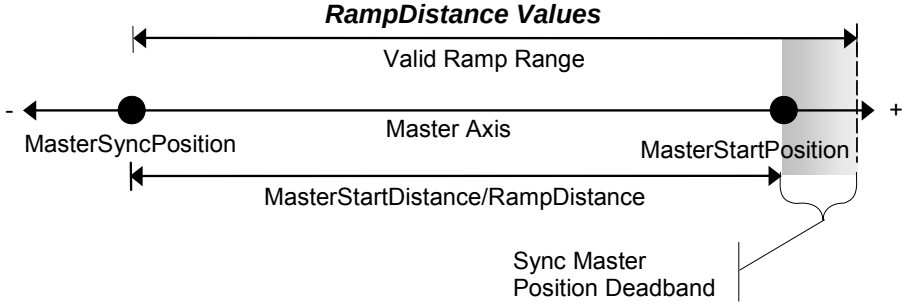
Specifies a positional deadband, in Master Axis Uu, to be applied to the Master's observed position by this axis when it is a Slave. The deadband may be applied to master positions such as the Master Cam Rollover position and the start position of Ramp onto a CamIn or GearInPos operation.

For MC_GearInPos: *MasterStartPosition = MasterSyncPosition – MasterStartDistance*
For MC_CamIn: *MasterSyncPosition = Current Master Position + RampDistance*

Positive MasterStartDistance and RampDistance Values

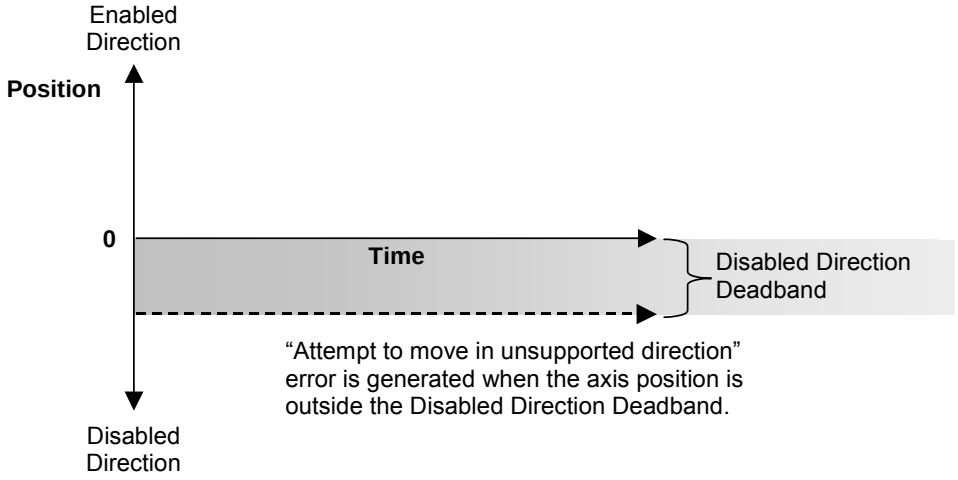


Negative MasterStartDistance and RampDistance Values



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.



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.

Advanced Data

The Advanced tab allows up to 32 parameters and associated data to be entered for the axes.

This tab can be used to set the parameters described in “Advanced Servo Tuning” in Appendix C. Refer to Appendix C for details on setting these parameters.

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.

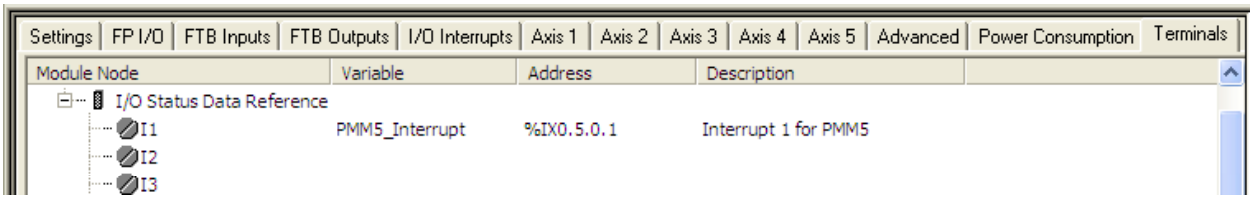
Power Consumption

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

Terminals

This tab is displayed when the module’s Variable Mode property is set to True. 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 page 4-7.



Chapter 5

PACMotion Function Block Operation

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

- PACMotion Function and Function Block Types 5-2
- Behavior of Motion Instructions 5-4
  - Instance Data 5-4
  - Immediate Response vs. Deferred Response 5-5
  - Administrative vs. Motion-Generating Instructions 5-5
  - Function Block Triggering (Enabled vs. Executed Instructions) 5-6
- Function and Function Block Parameters 5-14
  - Permissives, Constants, Variables or Flow used with MFBs 5-14
  - EN Input and ENO Output 5-14
  - Input Parameters 5-15
  - Reference ID Variables 5-18
  - Output Parameters 5-20
  - In_Out Parameters 5-23
- Data Types and Structures 5-24
- Axis States 5-27
- Synchronized Motion 5-30

For details about the use and operation of specific instructions, refer to Chapter 6.

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 the use of these function blocks, see Chapter 6, "PACMotion Instruction Set Reference."

Reset	Clear axis errors	MC_Reset
	Clear all errors on a module	Module Reset
Axis Powerup 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
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, see page 5-27.	MC_ReadStatus
	Read the ErrorID of the current axis error or warning.	MC_ReadAxisError
	Read an axis' actual position.	MC_ReadActualPosition
	Read an axis' actual velocity	MC_ReadActualVelocity
	Read an axis' commanded torque.	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
	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

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).

Instance Data

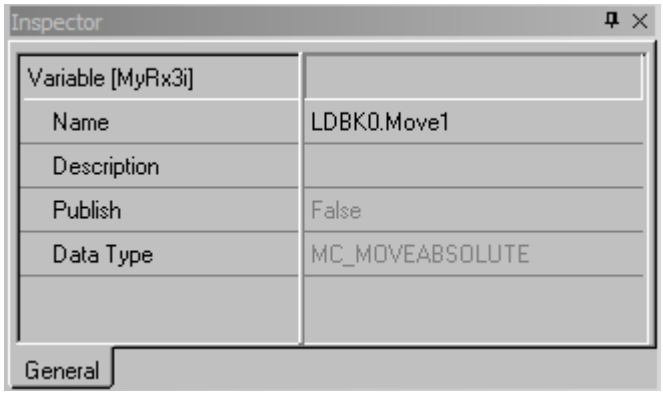
Standard function blocks have instance data consisting of a structure variable. Instance data is located in RX3i memory and is automatically allocated when a function block is stored to the RX3i for the first time. Each function block used in the application program has its own instance data. The values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

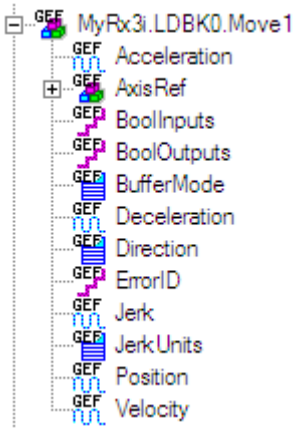
Each MFB has a unique data type that contains the instance data for that particular function block. For example, an MC_MoveAbsolute function block has a corresponding data type MC_MoveAbsolute that contains elements for input and output parameters as well as internal data.

For additional information on the use of instance data by function blocks, refer to "Program Organization" in the *PACSystems CPU Reference Manual,* GFK-2222.

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 "Output Parameters" on page 5-20.





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 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.

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 page 5-29.

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 “Output Parameters” on page 5-20.

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.

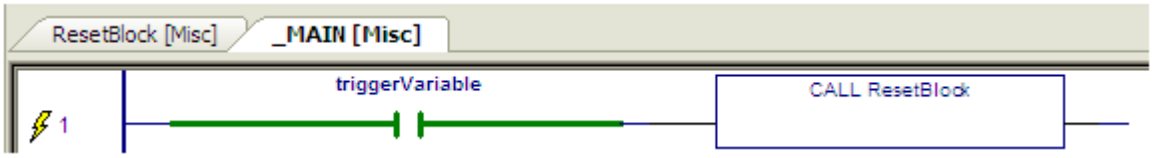
Permissive Logic Examples for Executed Function Blocks

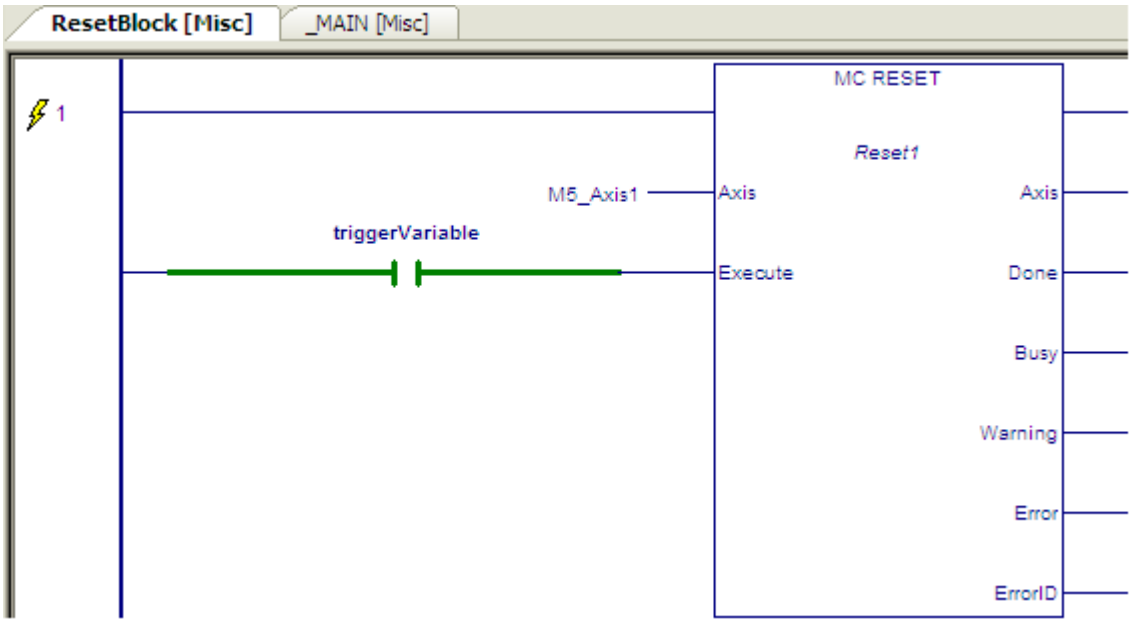
To start the execution of an edge-triggered MFB instance, you must ensure that the last time the MFB instance was called, its Execute input was FALSE and then call the MFB with a TRUE value for the Execute input.

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.

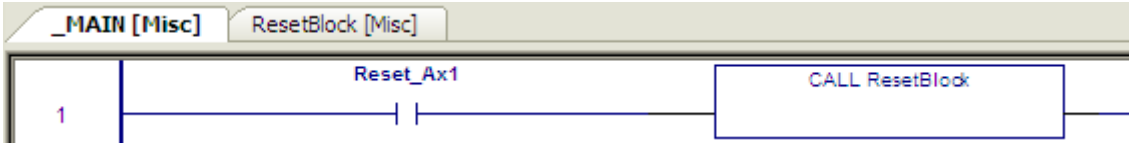
Not Recommended





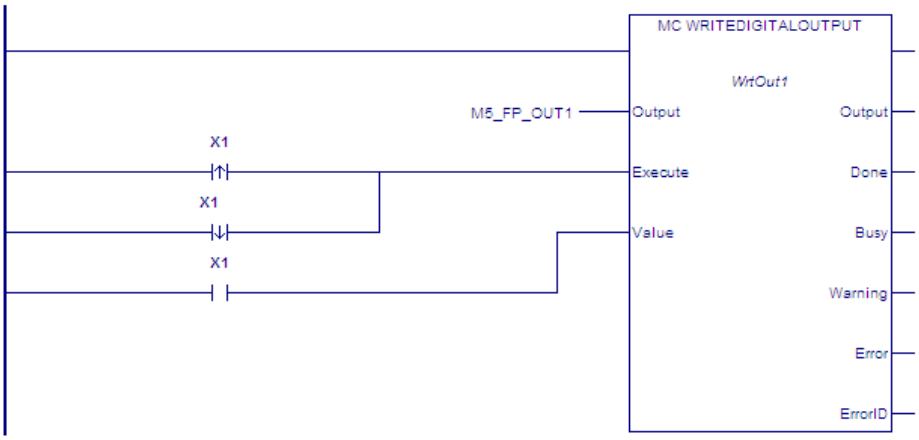
Recommended

Calling *ResetBlock* with a different variable allows the MC_Reset to be called multiple times.

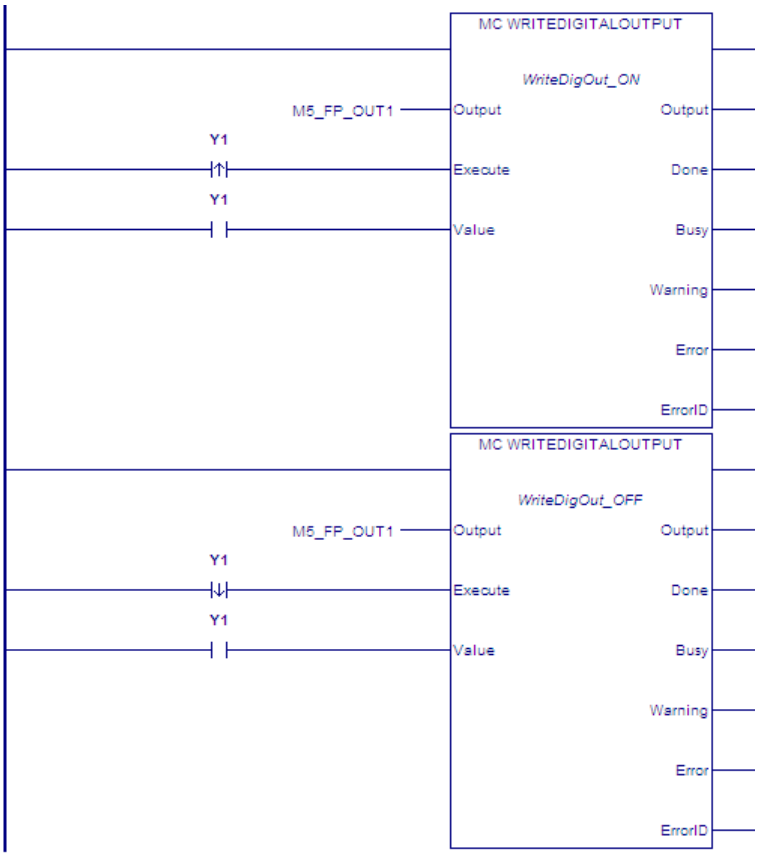


Example: Writing to a Digital Output

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.



The following sample logic accomplishes the result of writing the value of Y1 to M5_FP_OUT1 whenever the value of Y1 changes.



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.

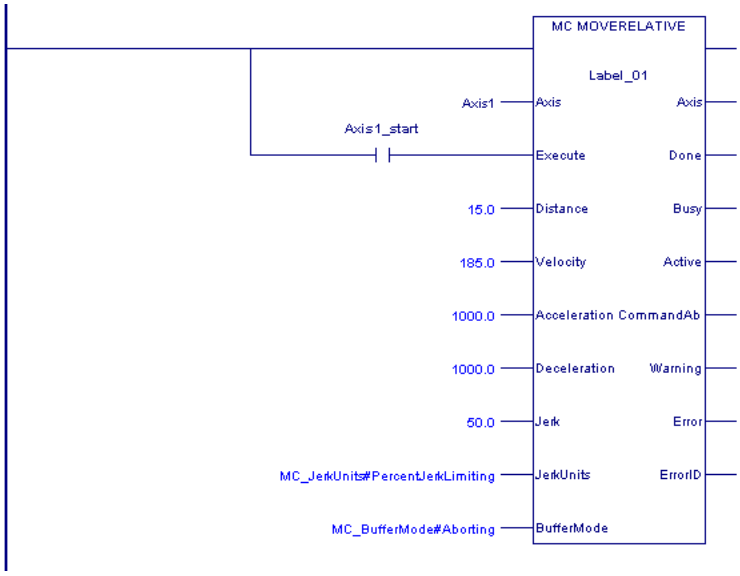
Calling an Executed Motion Function Block from an Interrupt Block

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.

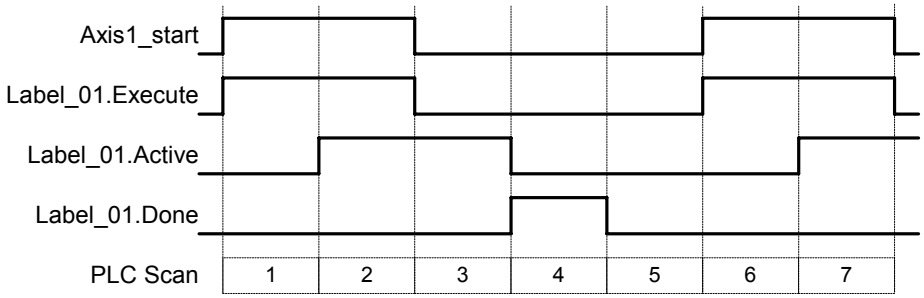
Incorrect Implementation of Interrupt MFB Call

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 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.



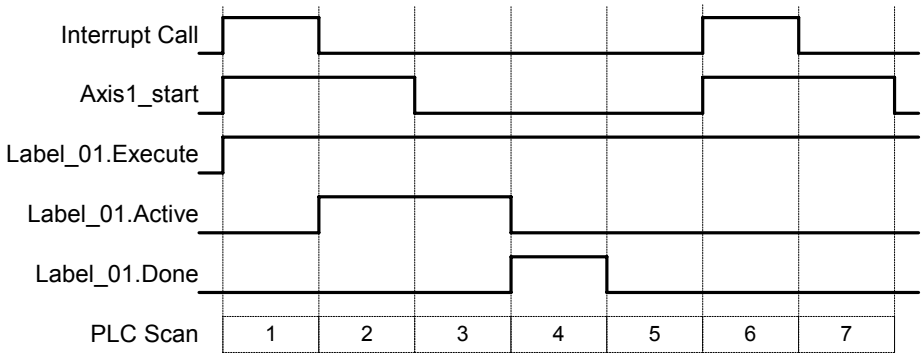
Timing Diagram, Conventional Logic Call

The response of the move command shown on page on page 5-11 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 is *Axis1_start* re-triggered in state 6 a new move command is initiated as expected.



Timing Diagram, Interrupt Logic Call

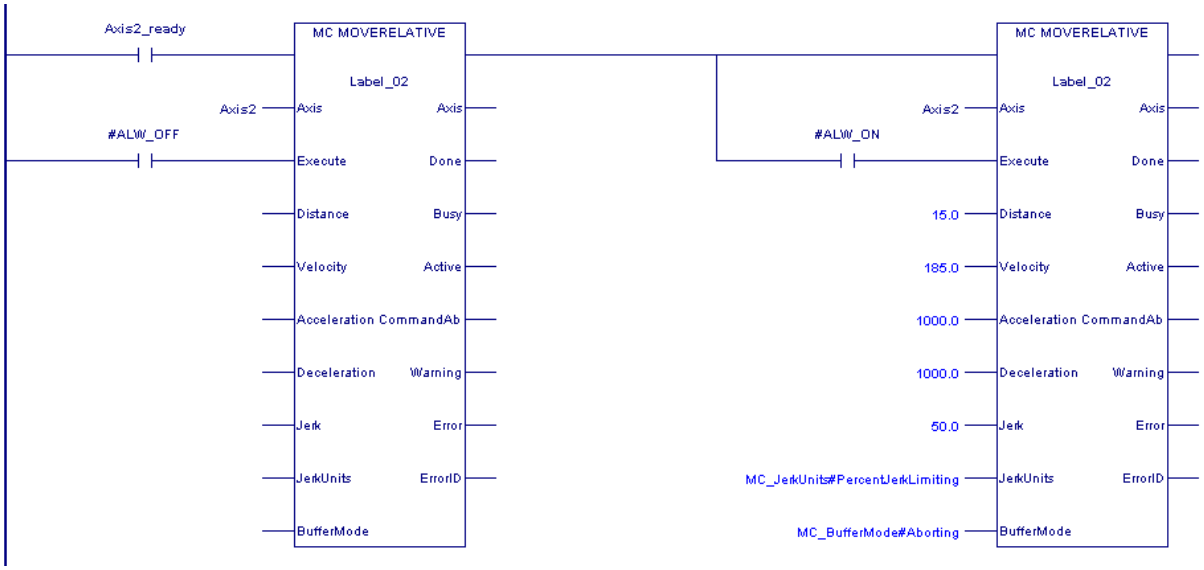
The response of the same move command when called from an interrupt block is detailed in the following timing diagram. In this example the interrupt block is only processed during steps 1 and 6. A result of the interrupt call is that the instance input data is not updated when *Axis1_start* transitions OFF. Thus no positive transition of *Label_01.Execute* occurs in step 6 (and no new move is initiated).



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 example, “Recommended Implementation of Interrupt MFB Call” on page 5-13.

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.



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.

Function and Function Block Parameters

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 CPU Reference Manual,* GFK-2222.

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.

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 function blocks 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 page 5-6.

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

Execute

The Execute command 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 page 5-7.

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	—
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	—
MC_SyncStart	—
MC_DelayedStart	—
MC_MoveSuperimposed	—

— Buffering not allowed after this command.

Buffer Mode Handling

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

Active Command	***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	2	2	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	2	2	4	2	All	All	All	All	2	4
MC_MoveAdditive	2	2	2	4	2	All	All	All	All	2	4
MC_MoveRelative	2	2	2	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 - 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.

— - 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.

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.”

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

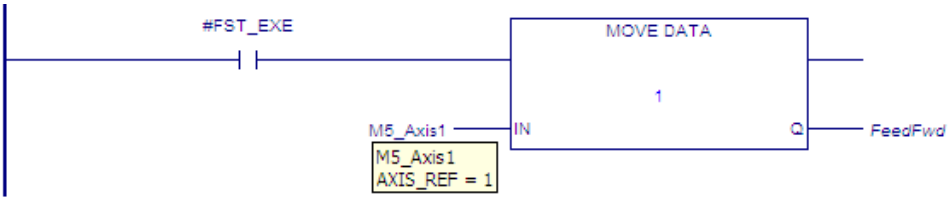
System RIVs and User-created RIVs can have the data types listed in the following table.

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.

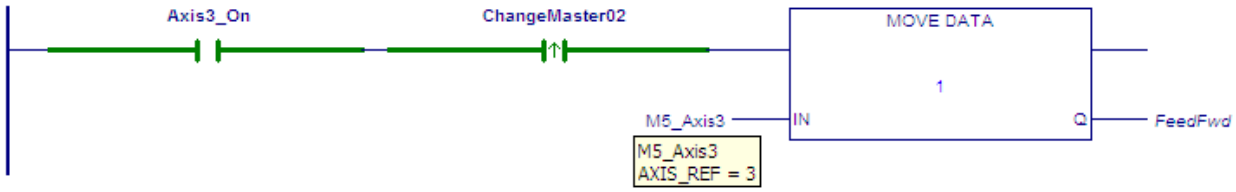
Example: Initializing and Changing a User-Created RIV

In the following sample logic, the System RIV, M5_Axis1, has been automatically assigned a value of 1. When 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.



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.



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" on page 5-6.

For details on output parameter behavior specific to edge-triggered function blocks, refer to "Outputs of Executed (Edge-Triggered) Function Blocks" on page 5-10.

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 Chapter 9, “Diagnostics.”

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.

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.

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 “Parameter Number Index” in Chapter 8.

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 "Parameter Number Index" in Chapter 8.

Enumerated Data Types

Enumerated data types are:

DL_OperatingMode DL_PostSample DL_SamplingRate DL_TriggerMode	Used by MC_DL_Configure function block. See discussion in Chapter 6.
MC_BufferMode	Used by many function blocks. See discussion on page 5-15.
MC_Direction	Used by MC_MoveVelocity and MC_MoveAbsolute. See discussion in Chapter 6.
MC_Encoder	Used by MC_SetPosition function block. See discussion in Chapter 6.
MC_HomingMode	Used by MC_Home function block. See discussion in Chapter 6.
MC_JerkUnits	Used by function blocks that command motion. See "Jerk and Jerk Units" on page 5-18.
MC_PositionSource	Used by MC_CamIn, MC_DigitalCamSwitch, MC_GearIn and MC_GearInPos function blocks. See discussion in Chapter 6.
MC_SyncMode	Used by MC_GearInPos function block. See discussion in Chapter 6.

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.

Axis States

When an axis is enabled (the amplifier has power) it is always in one of the states defined by the state machine shown on page 5-29. 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 amplifier does not have power.

Notes for Axis State Diagram (page 5-29)

Note 1: 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.

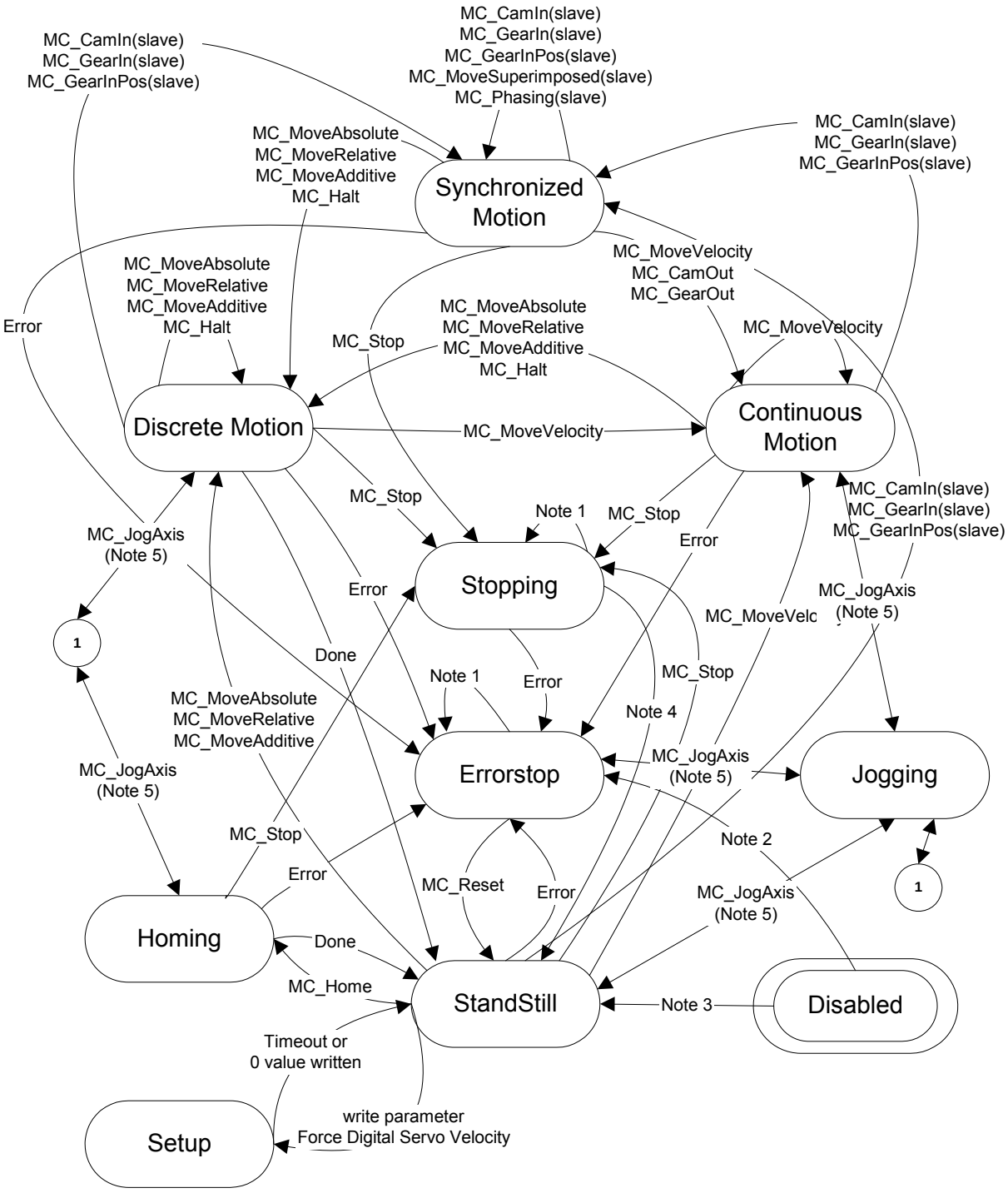
Note 2: MC_Power.Enable is TRUE and there is an error in the Axis

Note 3: MC_Power.Enable is TRUE and there is no error in the Axis

Note 4: MC_Stop.Done is TRUE and MC_Stop.Execute is FALSE

Note 5: 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).

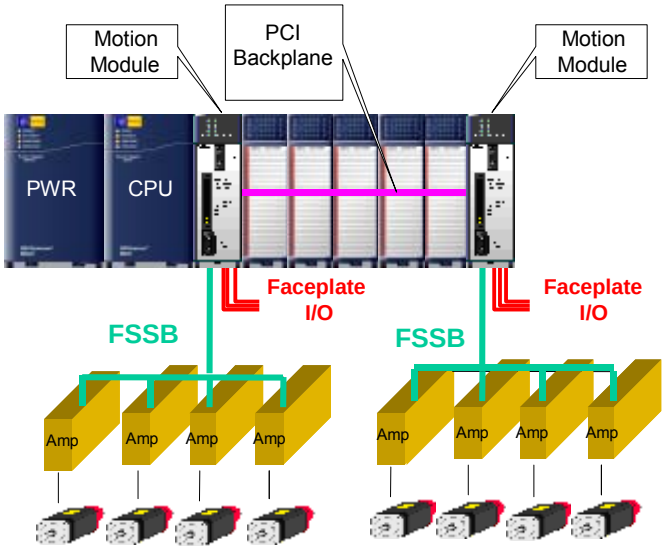
Axis State Diagram



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.



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 “Synchronized Motion Function Block Status” in Chapter 7.

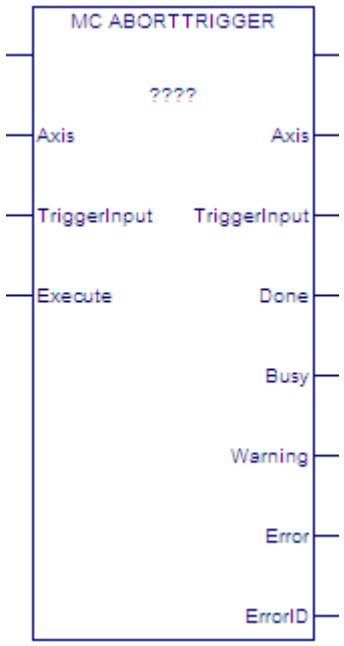
Chapter 6

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.

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

MC_AbortTrigger

LD	FBD	ST
		Formal convention: [instance name](Axis := [input], TriggerInput := [input], Execute := [input], Done => [output], Busy => [output], Warning => [output], Error => [output], ErrorID => [output]);

This function block is used to terminate an MC_TouchProbe function block operating on the specified Axis. The MC_TouchProbe function block is described on page 6-150.

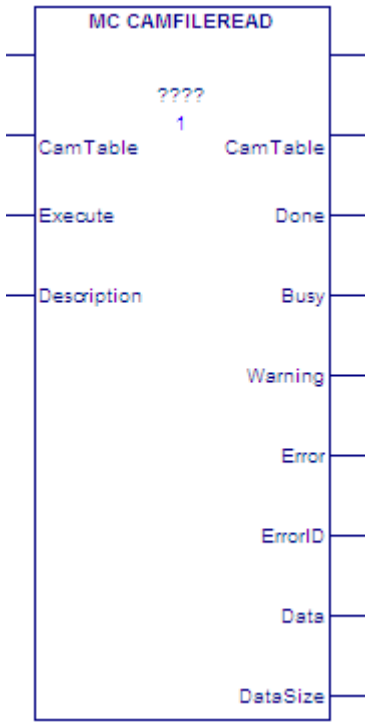
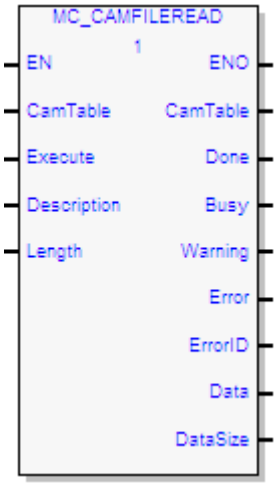
Execution type: Immediate execution/deferred response.

Operands

Parameter	Description	Allowed Data Types	Initial Value
????	Instance variable name.	MC_ABORTTRIGGER	N/A
	Input_Output Parameters		
Axis	The axis on which the function block is to be executed.	AXIS_REF	N/A
	Inputs		
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

Outputs			
Done	Trigger functionality has been aborted.	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

MC_CamFileRead

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

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 two milliseconds 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.

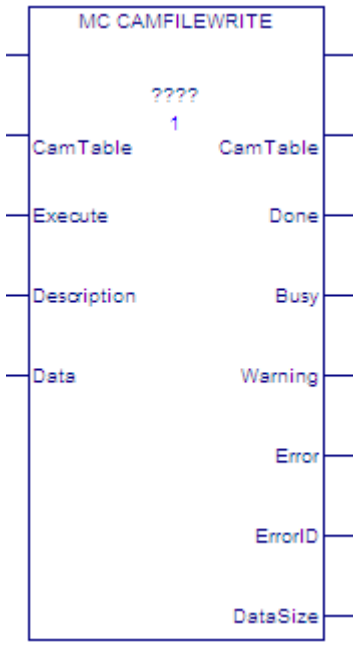
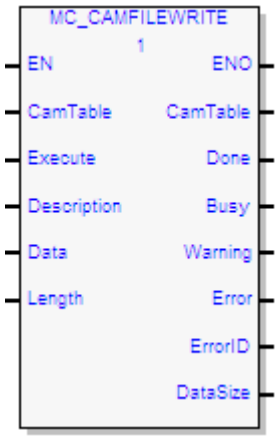
Execution type: Immediate execution/deferred response

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
Input_Output Parameters			
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
Outputs			
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

MC_CamFileWrite

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

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 two milliseconds 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.

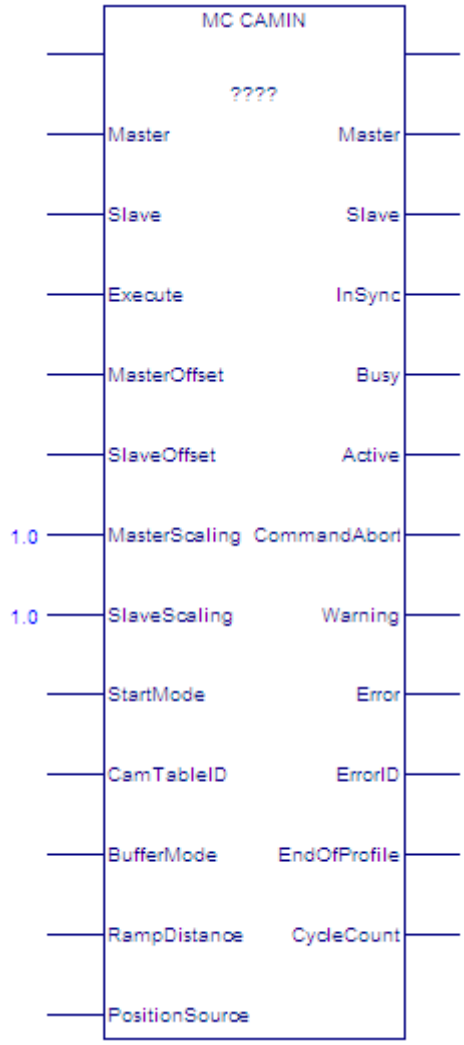
Execution type: Immediate execution/deferred response

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
Input_Output Parameters			
CamTable	The name of the file to be written.	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 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
Outputs			
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

MC_CamIn

LD	FBD	ST
		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 => [output], Busy => [output], Active => [output], CommandAborted => [output], Warning => [output], Error => [output], ErrorID => [output], EndOfProfile => [output], CycleCount => [output]);

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.

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 master axis' configured *Low Limit, Range* or *End of Travel* limits (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

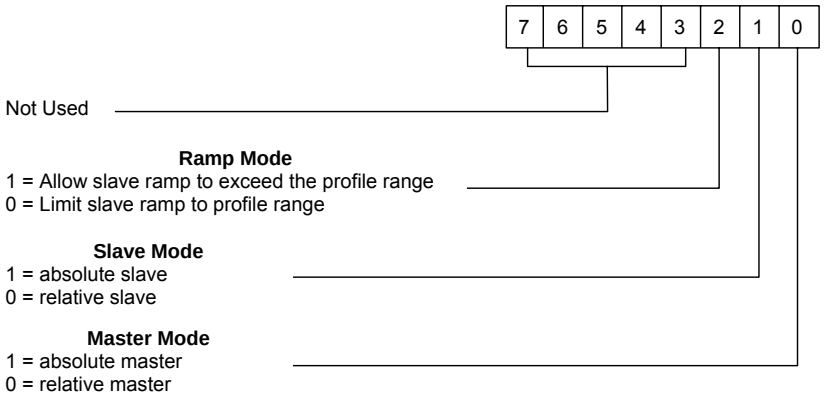
Operands

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***
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 at rising edge.	LD: flow Other languages: all except constants	0
MasterOffset	Offset of master table. See "Offset and Scaling" on page 6-12.	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.	DWORD. See "Start Mode Mask" on page 6-11.	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

Parameter	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 Actual Position is selected and Axis 5 is the master, Axis 5 must use an External Encoder.	MC_PositionSource	0
Outputs			
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

Start Mode Mask

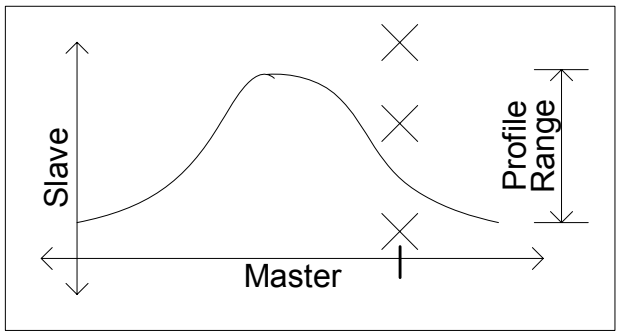


Ramp Mode

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 exceeds the boundaries of the profile. This is the default behavior when Ramp Mode is set to 0.



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.

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.

$$f \equiv CamFunction, x \equiv MasterPsn, y \equiv SlavePsn$$

$$y = SlaveScaling * f\left(\frac{x - MasterOffset}{MasterScaling}\right) + SlaveOffset$$

Using the Scaling and Offset to calculate the Lower and the Upper limits of the profile:

$$y = f(x), \quad x_{\min} \le x \le x_{\max}, \quad y_{\min} \le y \le y_{\max}$$

O = Offset; S=Scaling

$$S_x x_{\min} + O_x \le x \le S_x x_{\max} + O_x, \quad S_x > 0$$
$$S_x x_{\max} + O_x \le x \le S_x x_{\min} + O_x, \quad S_x < 0$$

$$S_y y_{\min} + O_y \le y \le S_y y_{\max} + O_y, \quad S_y > 0$$
$$S_y y_{\max} + O_y \le y \le S_y y_{\min} + O_y, \quad S_y < 0$$

Absolute

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.

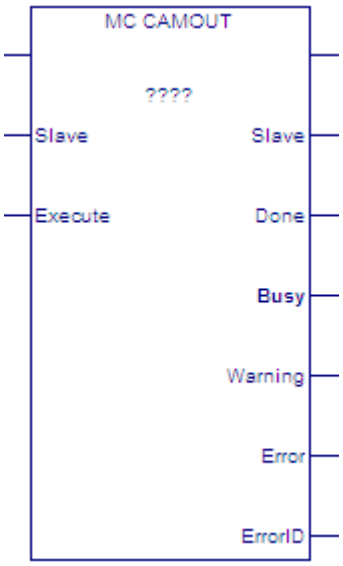
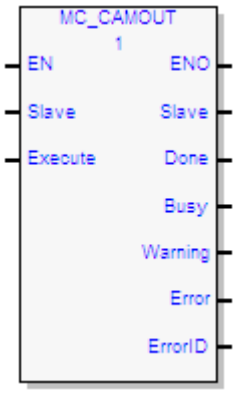
For example, with the following offsets, we get the following points:

MasterOffset: 0; SlaveOffset: 50

Master	Slave
0	-100
1000	150
2000	400

The Slave is currently at 100, but it the profile requires it to be at 150. In this case, Ramping is probably required. See “Synchronized Motion Function Block Status” in Chapter 7 for more on Ramping.

MC_CamOut

LD	FBD	ST
		Formal convention: [instance name](Slave := [input], Execute := [input], Done => [output], Busy => [output], Warning => [output], Error => [output], ErrorID => [output]);

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

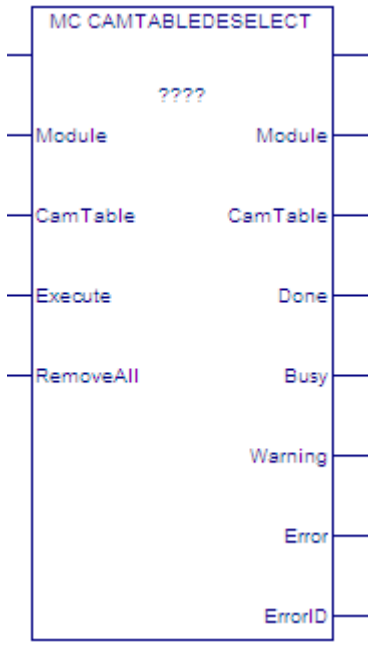
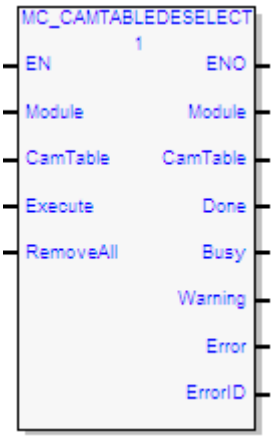
Operands

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*
	Input_Output Parameters		
Slave	Reference to slave axis. Virtual Axis (Axis 5) not supported.	AXIS_REF	N/A

Parameter	Description	Allowed Data Types	Initial Value
Inputs			
Execute	Start to disengage the slave from the master.	LD: flow Other languages: all except constants	0
Outputs			
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

MC_CamTableDeselect

LD	FBD	ST
		Formal convention: [instance name](Module := [input], CamTable := [input], Execute := [input], RemoveAll := [input], Done => [output], Busy => [output], Warning => [output], Error => [output], ErrorID => [output]);

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.

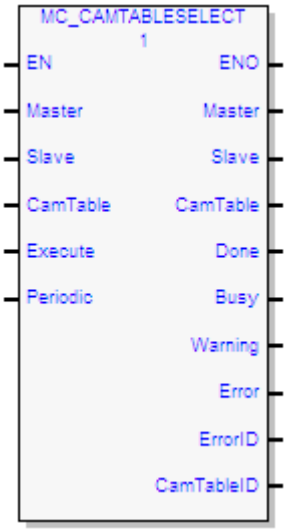
Execution type: Immediate execution/deferred response

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*
	Input_Output Parameters		
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	
	Outputs		
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	0

MC_CamTableSelect

LD	FBD	ST
		Formal convention: [instance name](Master := [input], Slave := [input], CamTable := [input], Execute := [input], Periodic := [input], Done => [output], Busy => [output], Warning => [output], Error => [output], ErrorID => [output], CamTableID => [output]);

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 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 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

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.

MC_DelayedStart

LD	FBD	ST
		Formal convention: [instance name](AxisArray := [input], Execute := [input], DelayTimes := [input], StartTime := [input], AbsoluteStart := [input], Length := [input], Done => [output], Busy => [output], Warning => [output], Error => [output], ErrorID => [output]);

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

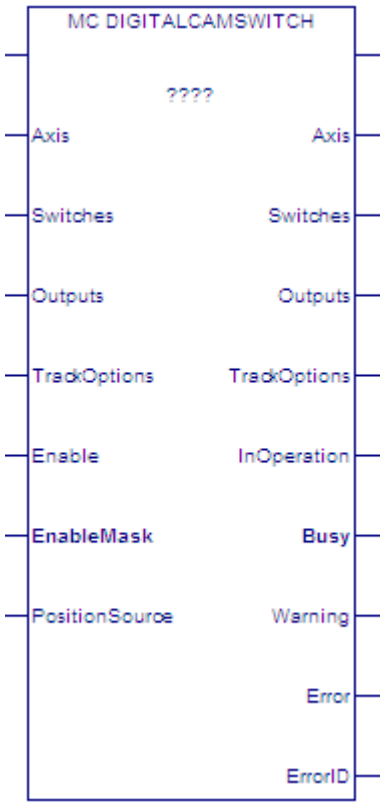
Operands

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

Parameter	*Description*	*Allowed Data Types*	*Initial Value*
	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 milliseconds 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 Other languages: all except constants	0
Busy	Indicates the function block has been executed and has not yet completed its action.		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

MC_DigitalCamSwitch

LD	FBD	ST
		Formal convention: [instance name](Axis := [input], Switches := [input], Outputs := [input], TrackOptions := [input], Enable := [input], EnableMask := [input], PositionSource := [input], InOperation => [output], Busy => [output], Warning => [output], Error => [output], ErrorID => [output]);

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 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. Any axis may have a DCS command; however 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, see page 6-27.	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, see page 6-32.	DCS_OUTPUTs	NA
TrackOptions	Reference to structure containing Hysteresis and ON and OFF compensations for each output/track. For details, see page 6-33.	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

MC_DigitalCamSwitch

Parameter	*Description*	*Allowed Data Types*	*Initial Value*
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 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 block.		0
Error	Indicates that an error has occurred within the function block.		0
ErrorID	Error or warning identification.	WORD	0

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.

Enabling Outputs for DCS Control

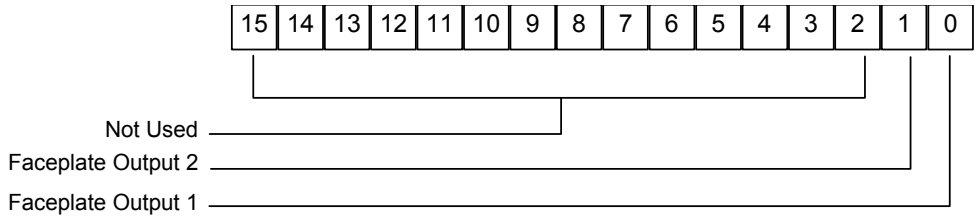
Before you can use DCS, outputs must be enabled for DCS control by writing the appropriate mask values to parameter 2114 for faceplate digital outputs and to parameter 2115 for FTB outputs. Any of the Digital Output Sources may be *enabled* for DCS control, but only four may be controlled at a time.

DCS Mask Values

To enable a digital output for use with the DCS, its corresponding mask bit must be set to 1.

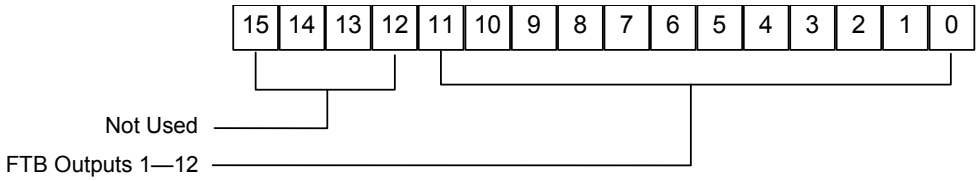
Parameter 2114 – Faceplate Digital Output Source

If a Faceplate digital output is used, it must be configured as *Digital Output* in HWC.



Parameter 2115 – FTB Digital Output Source

If an FTB output 9—12 is used, it must be configured as *Fast Digital Output* in HWC.



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 discussed on page 6-32 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:

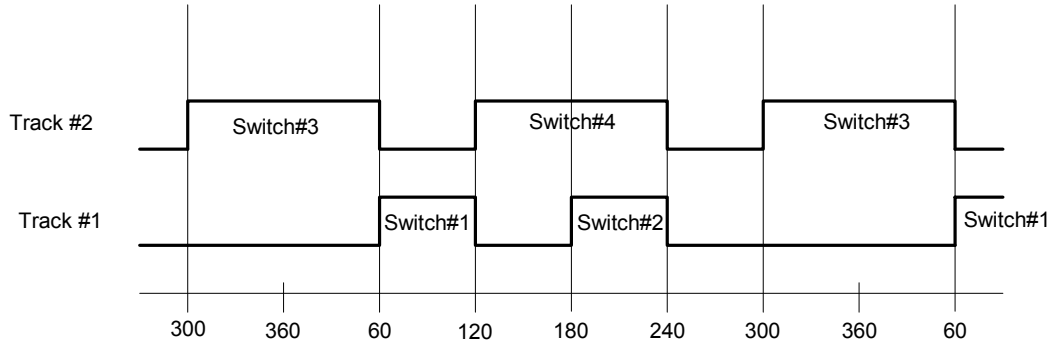
Element	Type	Values
Switch1 through Switch 8	SWITCH	
AxisDirection	INT	0 = both 1= positive 2= negative
CamSwitchMode	INT	0 = position-based 1 = time-based
Duration	LREAL	Time value. Applies only to the time-based CamSwitchMode.
FirstOnPosition	LREAL	Lower boundary where the switch is ON.
LastOnPosition	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

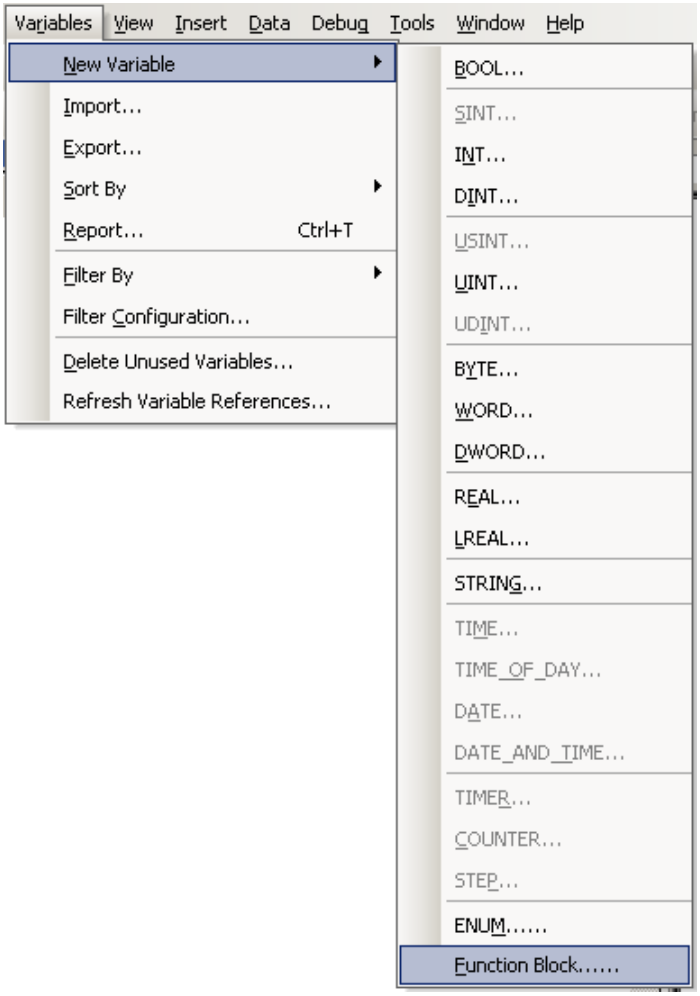
This example illustrates how to set up two tracks with two switch points each. In this example, the *Axis Position Mode* is *Rotary* (set up in the axis hardware configuration) and the positions are programmed in degrees. The application requires the following switch patterns.

	AxisDirection	CamSwitchMode	Duration	FirstOnPosition	LastOnPosition	TrackNumber
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

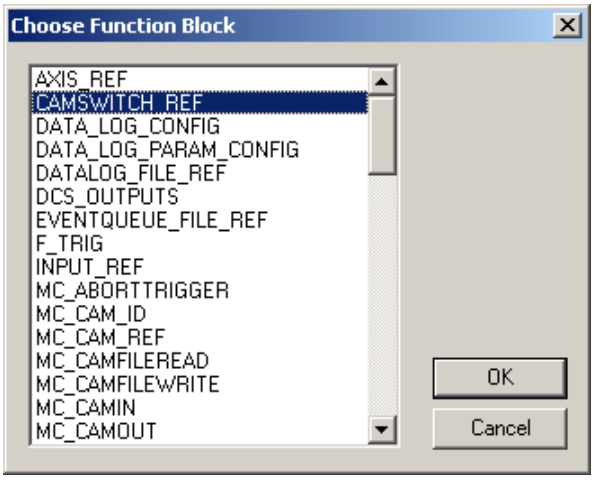
These switch patterns correspond to the following switching sequences.



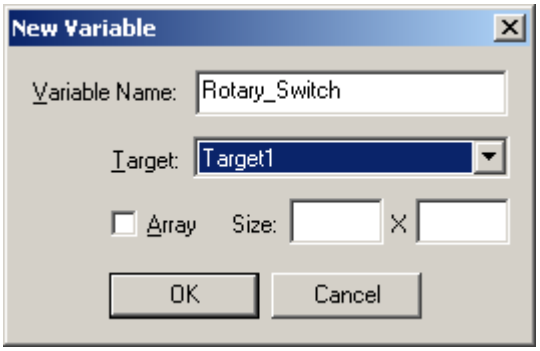
To achieve this switch pattern, first create a new variable called Rotary_Switch with the type CAMSWITCH_REF. To perform this step, select *Variables* from the main menu and the sub menu *NewVariable* followed by the selection *Function Block*.



The Choose Function Block dialog box appears, which allows you to select the type for the variable.

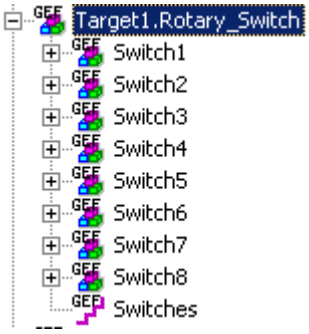


Click OK and then give the variable the name Rotary_Switch as shown below

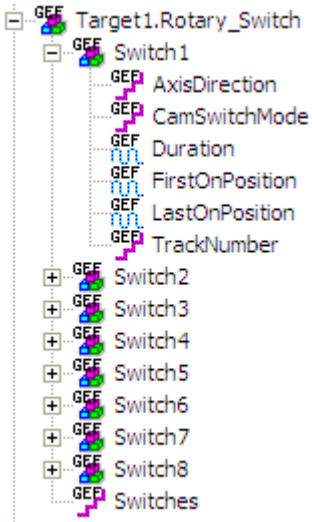


The Switches input on the digital cam switch contains a structure that defines the switch points and the tracks association. (The Outputs input will be used to pair a cam switch track to a physical output.)

This places a new variable called Rotary_Switch in the variable list. Navigate to this variable in the Variables tab in the Navigator pane.



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:



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.

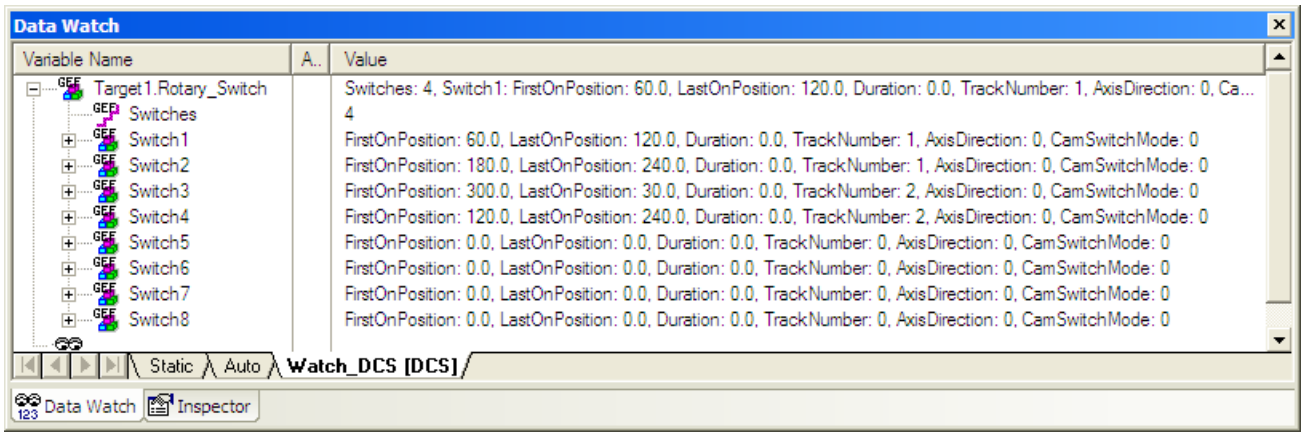
FirstOnPosition – Lower boundary where the digital cam switch is to activate. If FirstOnPosition > LastOnPosition, the switch is inverted. The means the switch is off during the interval instead of on.

LastOnPosition- Upper boundary where cam switch is on.

TrackNumber- Track number to associate with the switch (1–4).

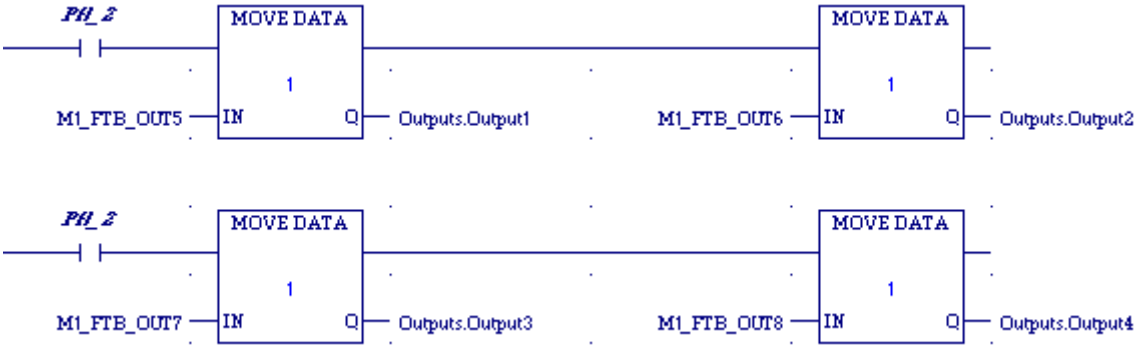
Set the *Switches* element to the number of switches that will be activated in the DCS.

Once the switching pattern definition is complete, you can view the variables in the Data Watch window to check that the definition is correct.

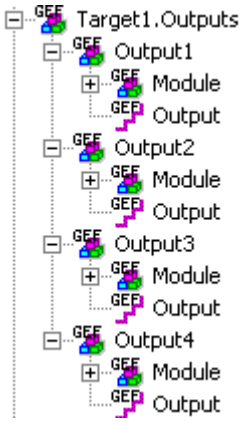


Outputs

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_FTB_OUT5, M1_FTB_OUT6, M1_FTB_OUT7, and M1_FTB_OUT8) is shown below.

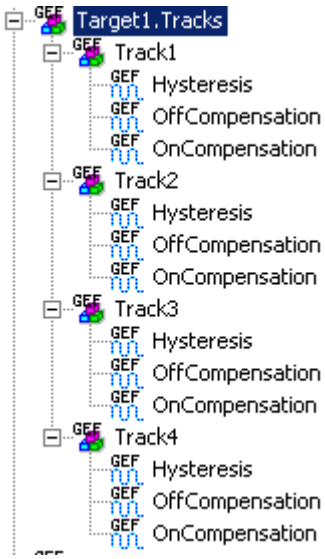


This moves the references into the Outputs array shown below and associates the four outputs with the tracks. In this case track 1 is linked to M1_FTB_OUT5, track 2 is linked to M1_FTB_OUT6, track 3 is linked to M1_FTB_OUT7, and track 4 is linked to M1_FTB_OUT8 via the DCS function block. An output can be controlled by only one Digital Cam Switch command at a time.



TrackOptions

The TrackOptions structure allows you to apply Hysteresis or On/Off compensation to each track.



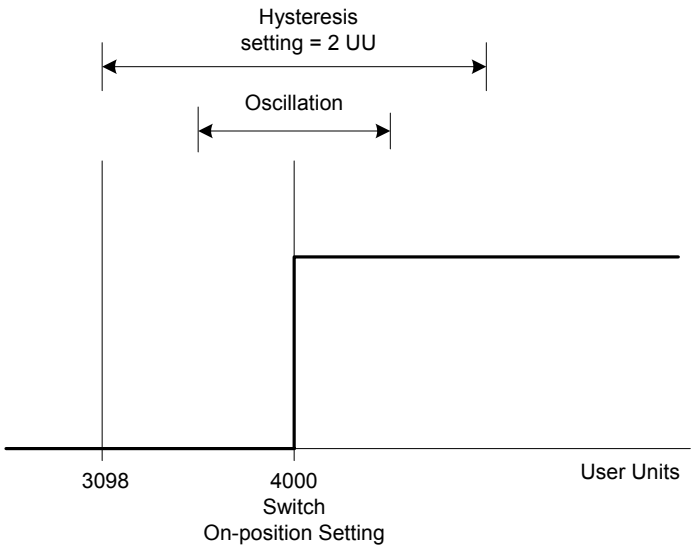
Note: Hysteresis and On/Off compensation cannot be used on the same track. Only one type of compensation is allowed per track.

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, 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 PositionSource 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.

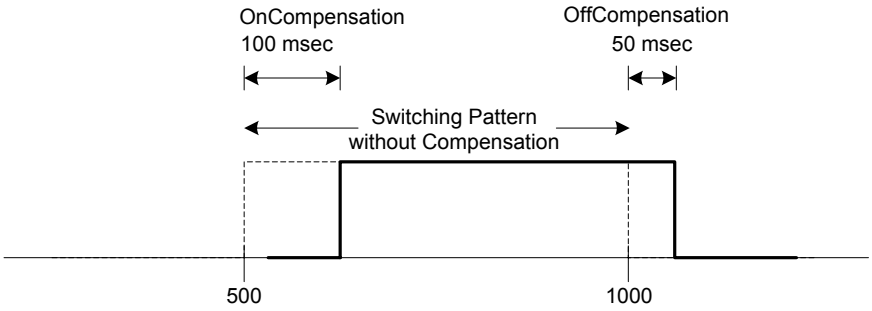
Example Hysteresis Setting



OffCompensation – The off compensation 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- The on compensation 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.

On/Off Compensation Example



Note: The actual position at which the switch turns On/OFF is velocity dependent

Acceleration or deceleration may affect the accuracy of the compensation. For details, refer to “Digital Cam Switch Accuracy” in Appendix A.

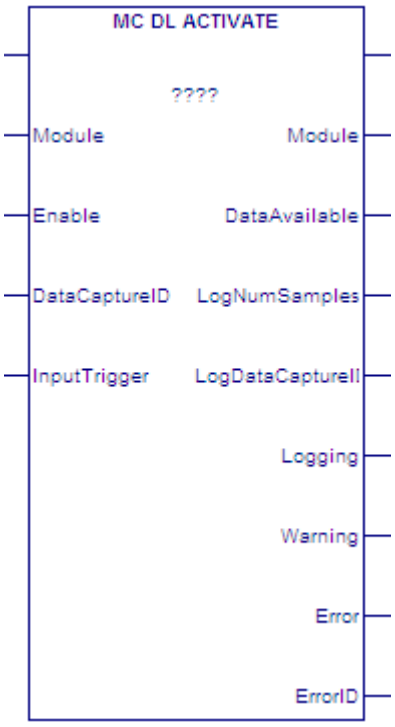
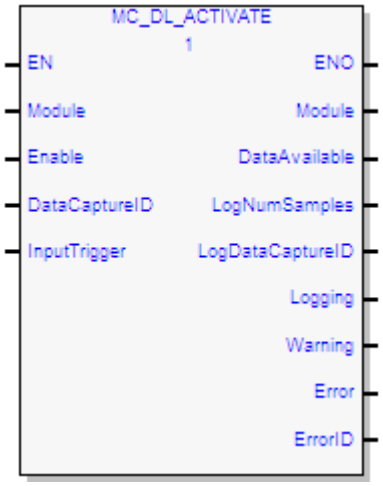
Enable

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).

MC_DL_Activate

LD	FBD	ST
		Formal convention: [instance name](Module := [input], Enable := [input], DataCaptureID := [input], InputTrigger := [input], DataAvailable => [output], LogNumSamples => [output], LogDataCaptureID => [output], Logging => [output], Warning => [output], Error => [output], ErrorID => [output]);

This function block is used to start data logging on the module. Its operation is controlled by the MC_DL_Configure function block, described on page 6-37.

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, see page 6-45.

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.

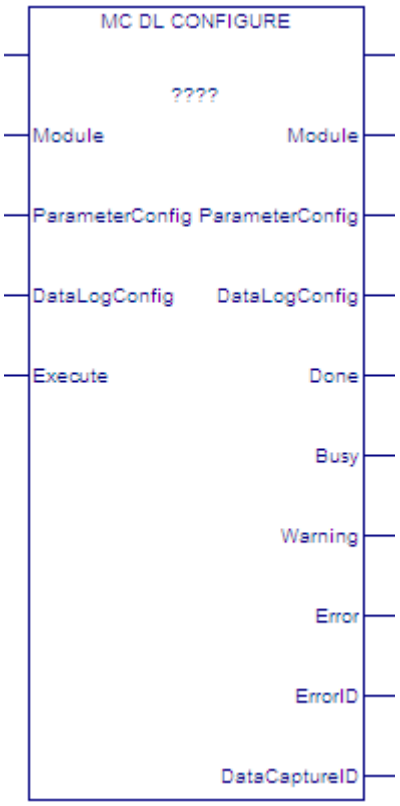
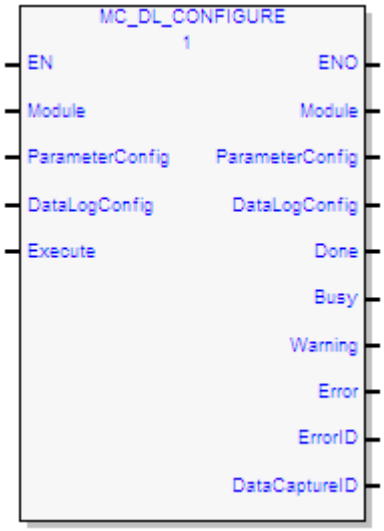
Execution type: Immediate execution/deferred response.

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***
Input_Output Parameters			
Module	Identifies the module to work upon.	MODULE_REF	N/A
Inputs			
Enable	Start logging data	LD: flow Other languages: all except constants	0
DataCaptureID	Identifies the Data Logging configuration output of the MC_DL_CONFIGURE function block (page 6-37).	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
Outputs			
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

MC_DL_Configure

LD	FBD	ST
		Formal convention: [instance name](Module := [input], ParameterConfig := [input], DataLogConfig := [input], Execute := [input], Done => [output], Busy => [output], Warning => [output], Error => [output], ErrorID => [output], DataCaptureID => [output]);

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, see “Data_Log_Param_Config Data Structure” on page 6-39.

The *DataLogConfig* specifies the Number of Samples, Sampling Rate, Operating Mode, Trigger Mode and Post Sample Percentage. For details, see “Data_Log_Config Data Structure” on page 6-39.

Data logging is initiated by the MC_DL_Activate function block, described on page 6-35.

For an example of data logging operation, see page 6-45.

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.	DATA_LOG_PARAM_CONFIG See “Data_Log_Param_Config Data Structure” on page 6-39.	N/A
DataLogConfig	Data Logger configuration.	DATA_LOG_CONFIG See “Data_Log_Config Data Structure” on page 6-39.	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, see “Parameter Number Index” in Chapter 8.

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 shown on page 6-41.
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 being 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.)
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:

MaxNumberSamples = 2Mbyte / (DataSetSize + TimeStampSize)

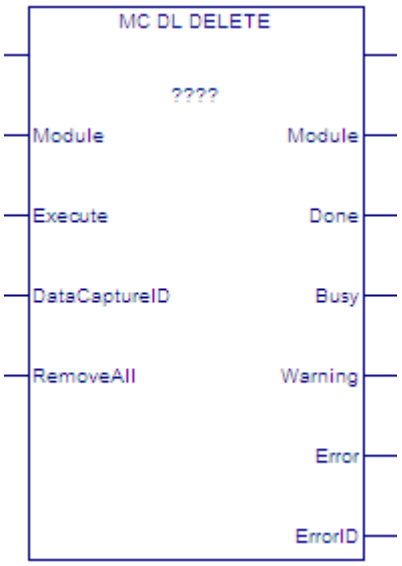
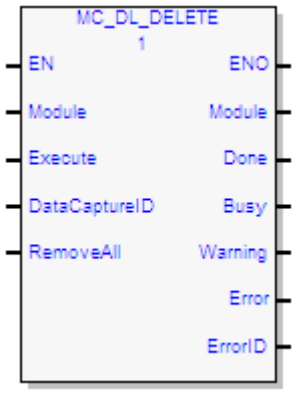
- DataSetSize is based on the number of parameters to log, the parameter type and the number of axes.
- TimeStampSize is 4, 32bit number for indicating the timestamp of every position loop interval.
- The maximum space available for data logging is 2MB (2097152 bytes)

Example for calculating the maximum number of samples:

- Two real parameters (8 bytes)
- Two axes

$$\text{MaxNumberSamples} = \frac{[20971526\text{bytes}]}{[8\text{bytes} * 2 * 2] + 4\text{bytes}} \rightarrow 58254\text{samples}$$

MC_DL_Delete

LD	FBD	ST
		Formal convention: [instance name](Module := [input], Execute := [input], DataCaptureID := [input], RemoveAll := [input], Done => [output], Busy => [output], Warning => [output], Error => [output], ErrorID => [output]);

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.

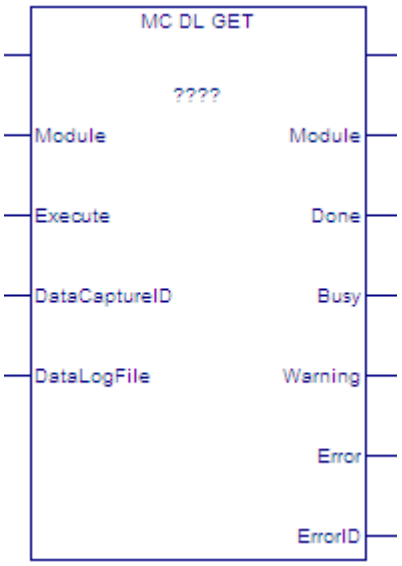
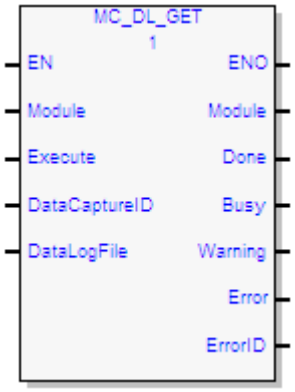
Operands

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
Input_Output Parameters			
Module	Identifies the module to work upon.	MODULE_REF	N/A
Inputs			
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 (page 6-37).	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	

Outputs			
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

MC_DL_Get

LD	FBD	ST
		Formal convention: [instance name](Module := [input], Execute := [input], DataCaptureID := [input], DataLogFile := [input], Done => [output], Busy => [output], Warning => [output], Error => [output], ErrorID => [output]);

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 milliseconds 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
Input_Output Parameters			
Module	Identifies the module to work upon.	MODULE_REF.	N/A

Inputs			
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 (page 6-37).	WORD	0
DataLogFile	Specifies a file reference to the stored data log.	DATALOG_FILE_REF	0
Outputs			
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

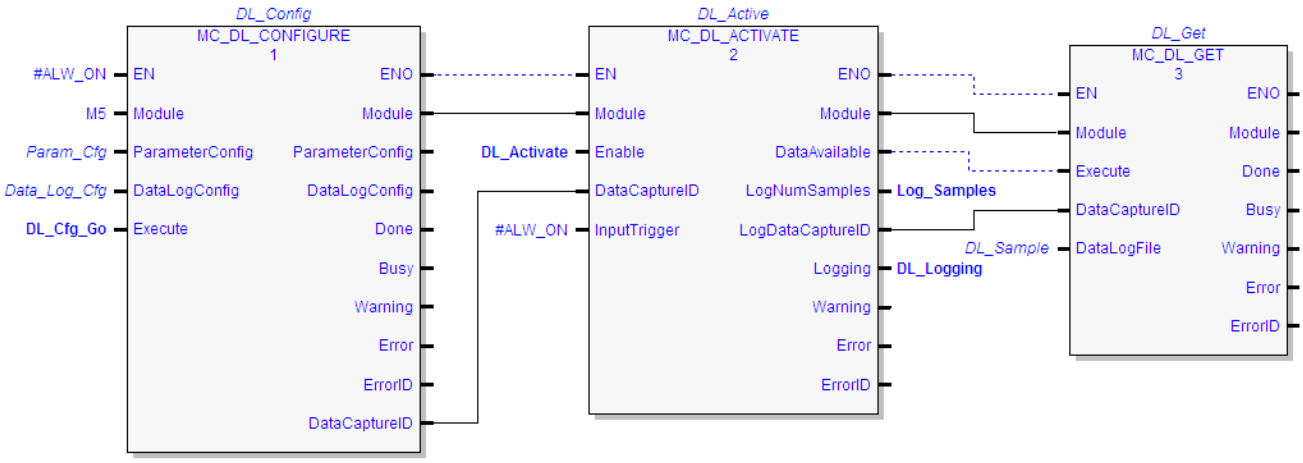
Data Logging Example

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:

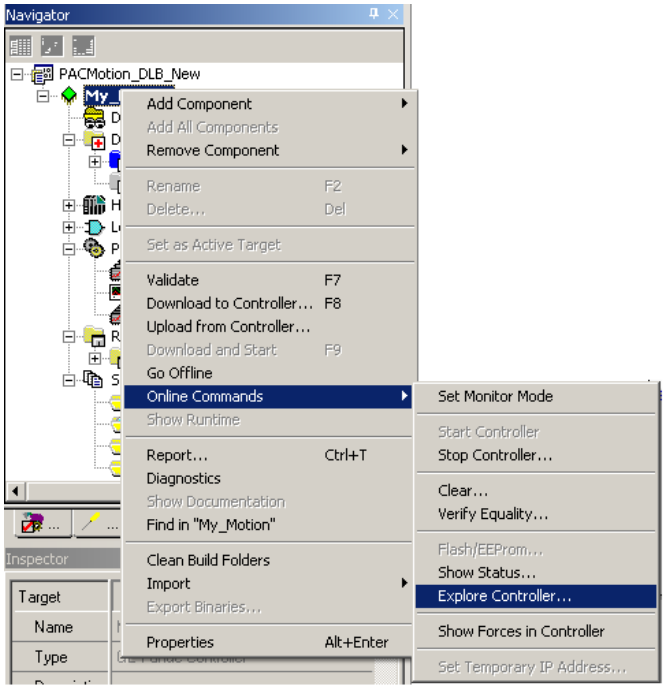
1. Configure the information to be logged using the MC_DL_Configure function block.
2. Activate the data logging session using the MC_DL_Activate function block.
3. Wait for the data logging to trigger and complete capture in the PMM module. Completion of the data logging can be determined by monitoring 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.

Sample Logic to Configure, Activate, and Get a Data Log



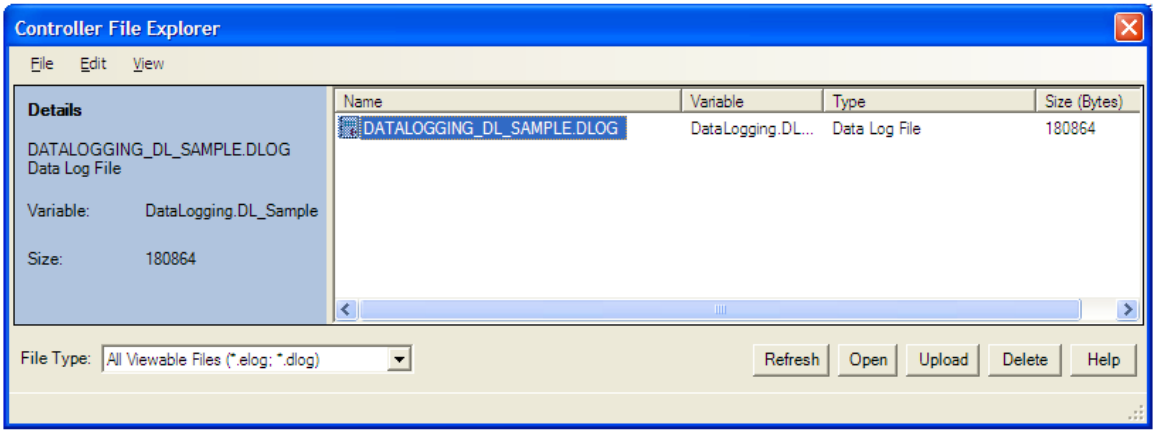
Uploading Data Logging (DLOG) Files from PACSystems Controller

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.



If you select the Online Commands > Explore Controller… menu item, the Controller File Explorer dialog is presented.

Controller File Explorer

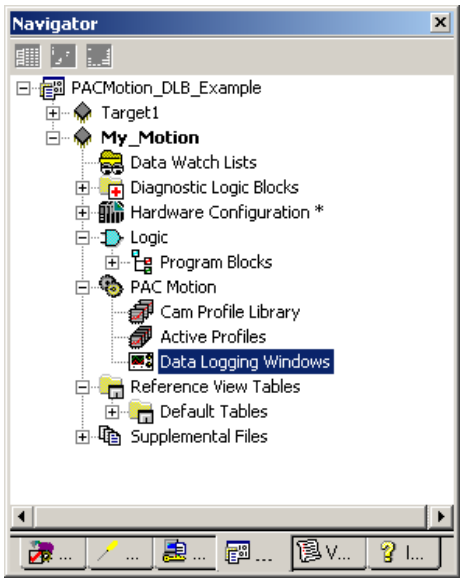


You can open, upload, or delete files from the list of dlog files that exist on the controller. The list shows all the dlog files that are available. To update the list of files, click the Refresh button. To create a Data Logging Session and upload the DLOG file to the project, click the Open button. 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

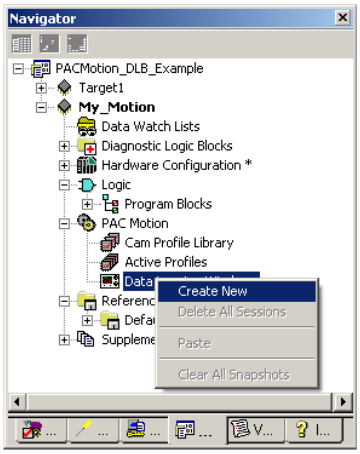
Once files have been uploaded from the controller, the logged data may be displayed. The DLOG information for the PMM is managed for the controller target in the Project Navigator using the Data Logger Window node highlighted below.

Data Logging Windows Node

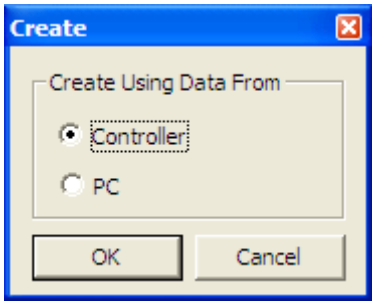


Managing and Viewing the Data Logging Information

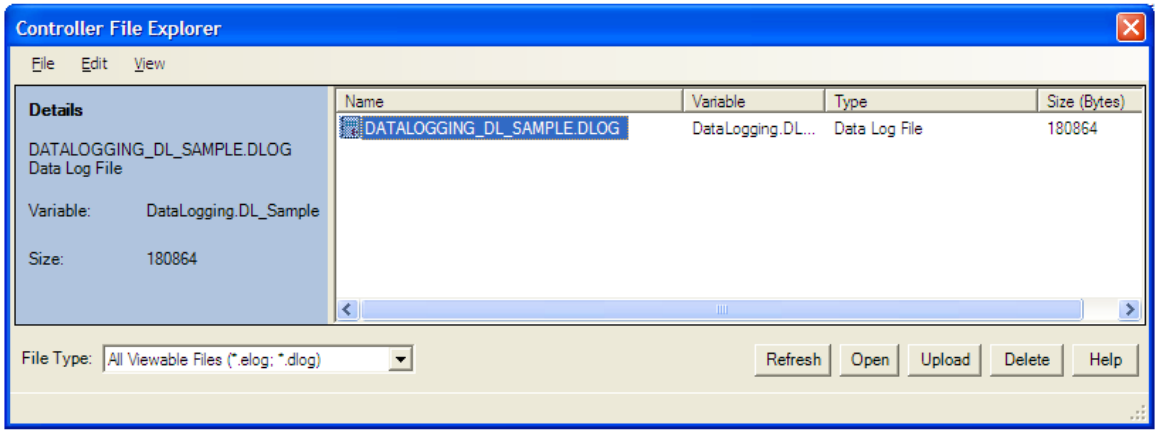
1. Create a new Data Logging Window Session for a DLOG File using the Create New session menu item –or- Select the Open button from the Explore Controller dialog:



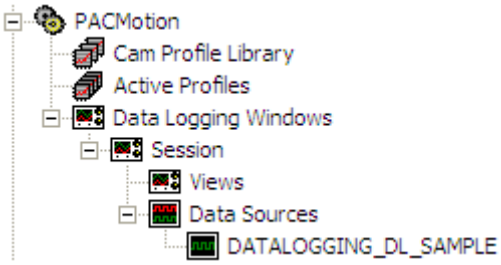
2. If using the Create New menu, specify the Data Logging data source.



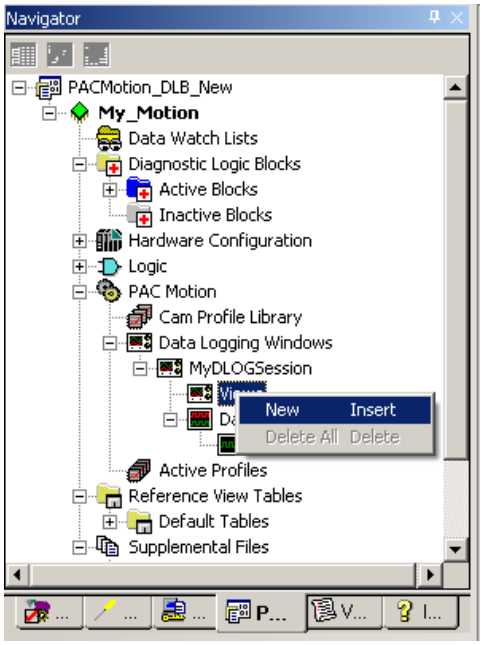
3. Select the DLOG file that you want to view and select Open.



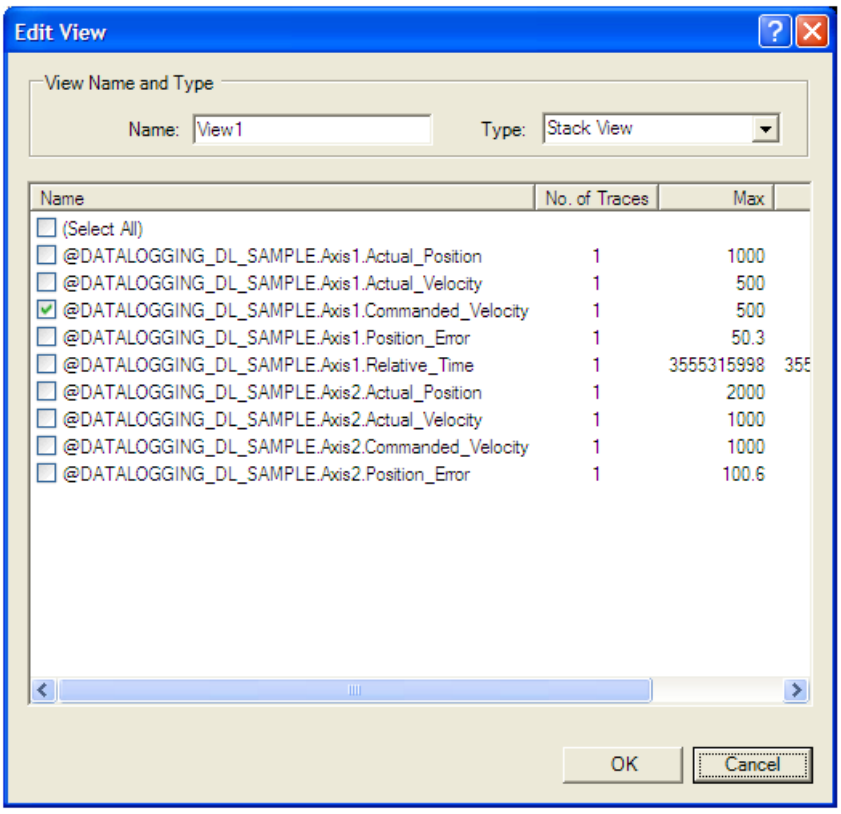
The session node now has two child nodes, one to manage the **Views** of the Data and a second to manage the **Data Sources**.



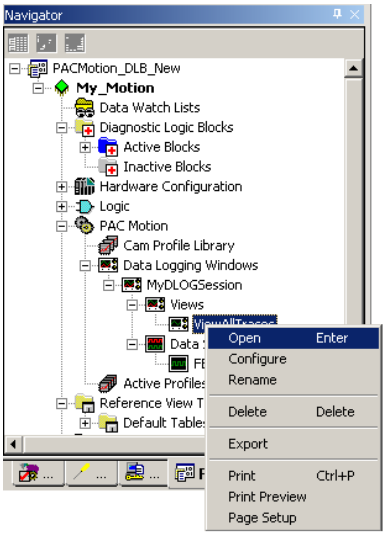
4. To create a View to display the data logging information, selecting the Views node and right-click the New menu item.



5. The Edit View dialog lets you define the View's Name (View1 is the default name), select which signals to trace, and select the general display Type (Stack View or Tab View).

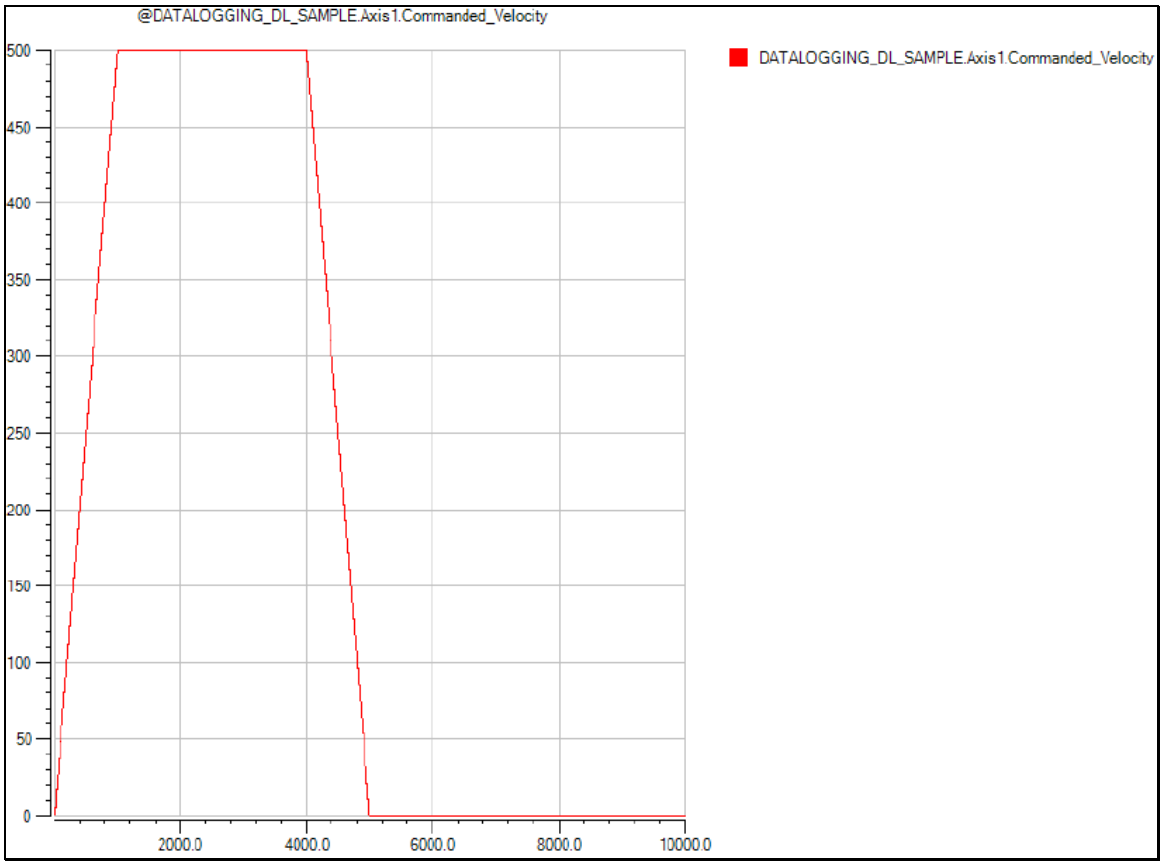


6. Once the View has been defined, you can display the view using the Open menu, or by pressing the Enter key:



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.

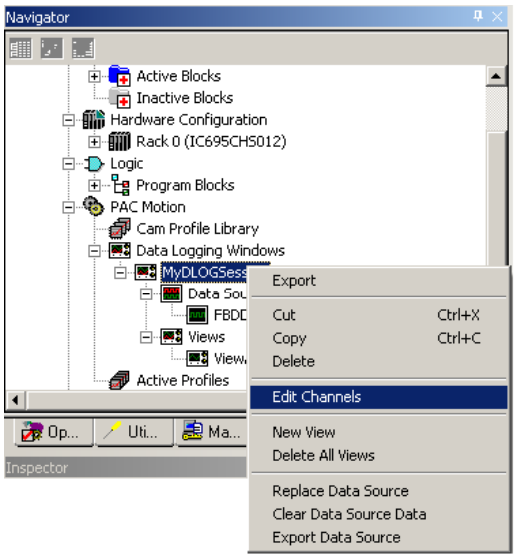
Sample Data Logging Window View With One Trace



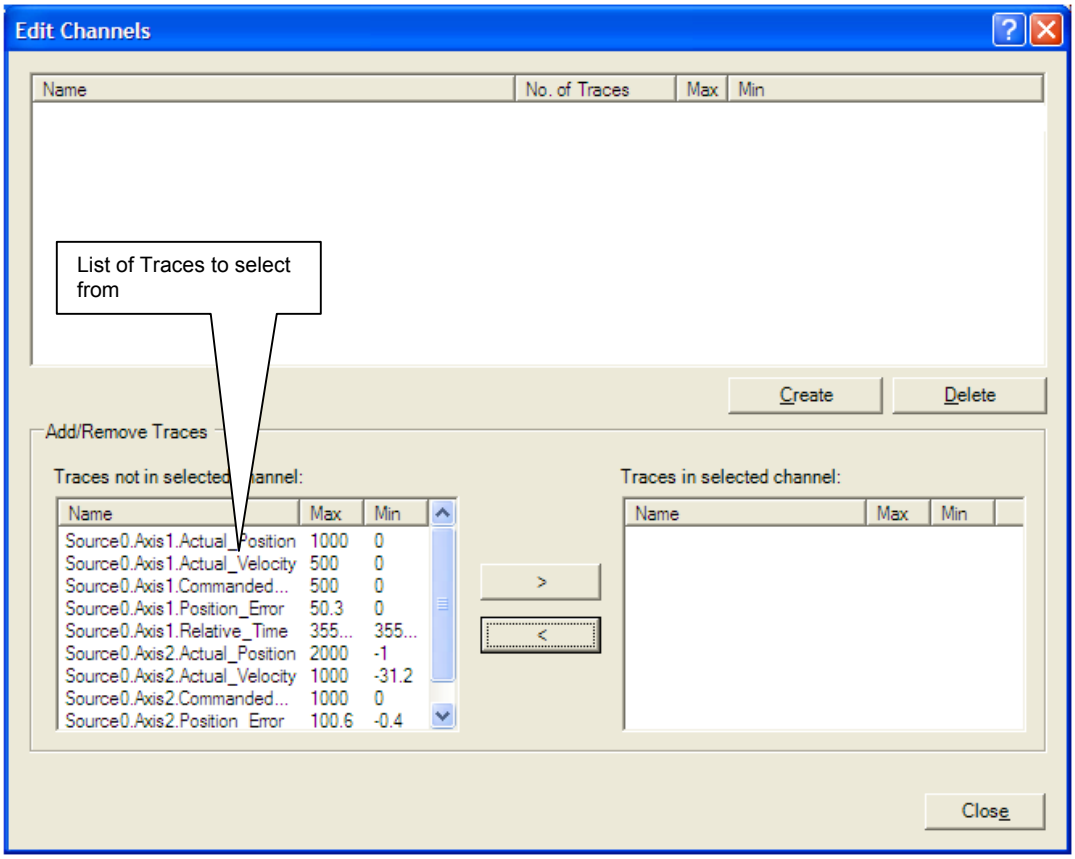
Defining Data Logging Channels

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.

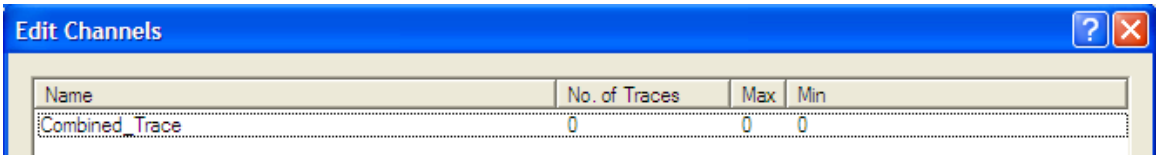


The **Edit Channels** dialog is displayed:

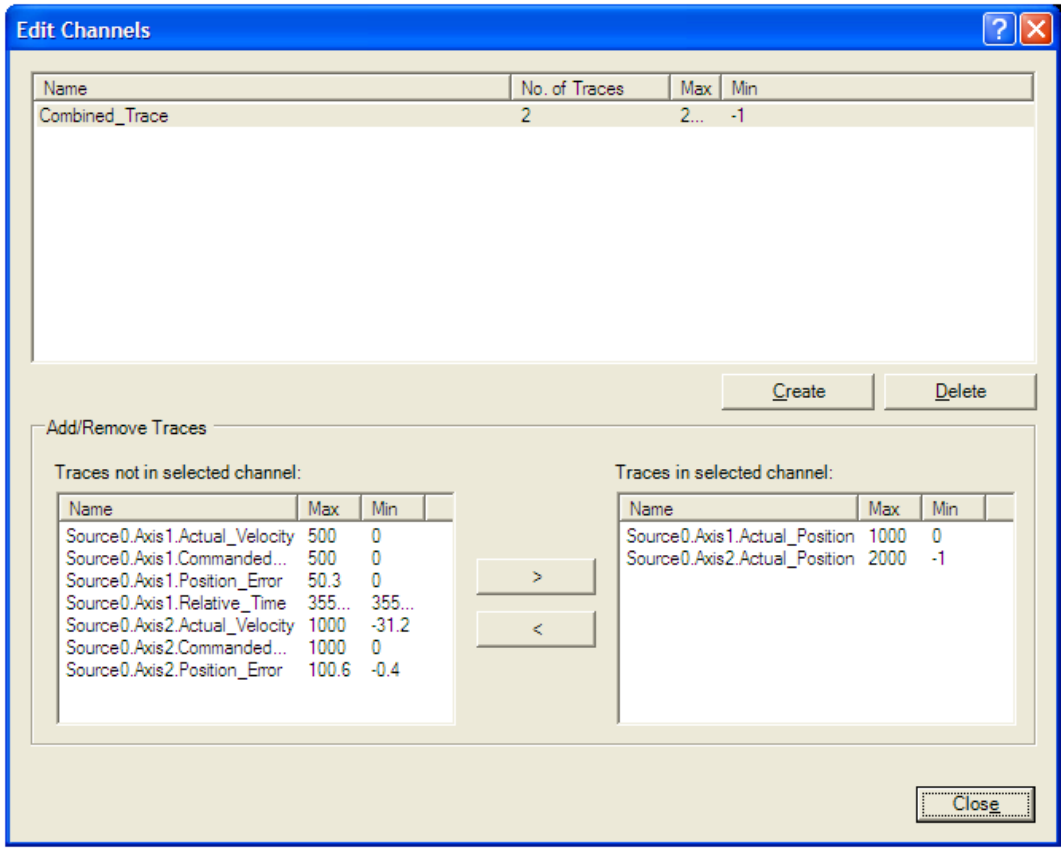


To define a Channel, click the Create button.

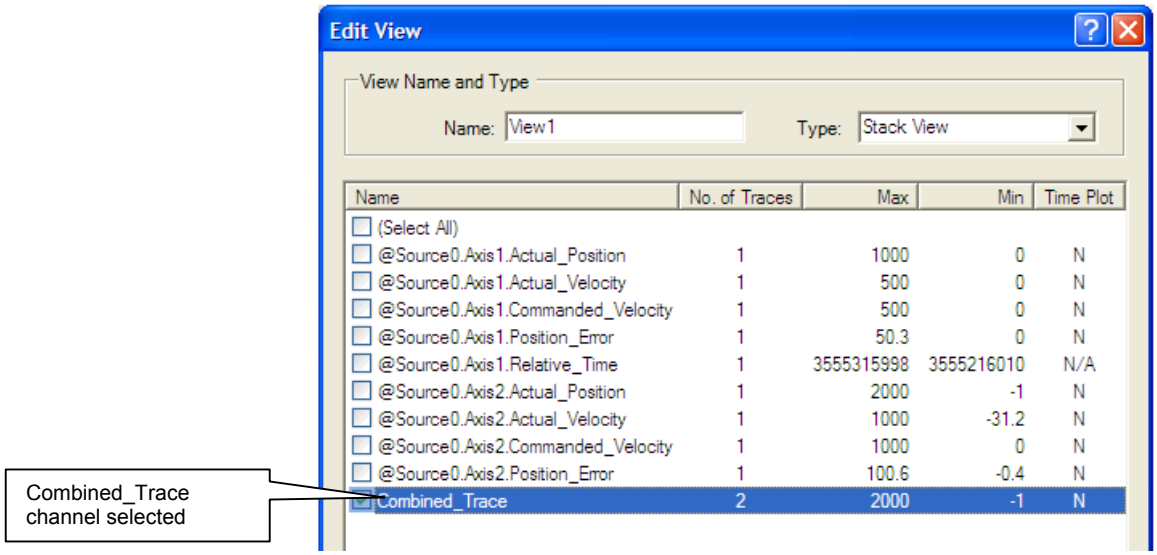
Give the new Channel a descriptive name, such as Combined_Trace.



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.

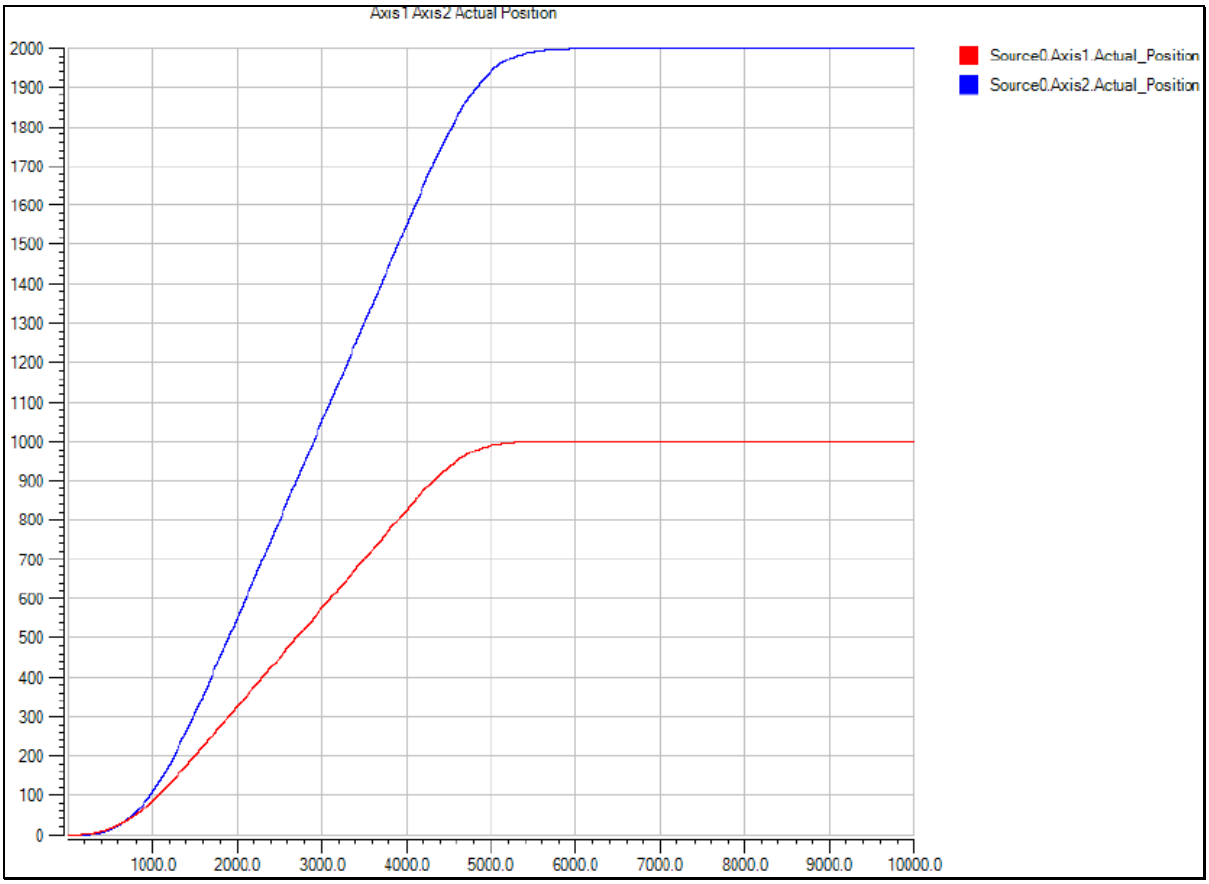


Now, when you create a View you can select the Combined_Trace Channel:



This will result in a display with combined traces, similar to the following sample trace.

Channel View With Two Traces



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

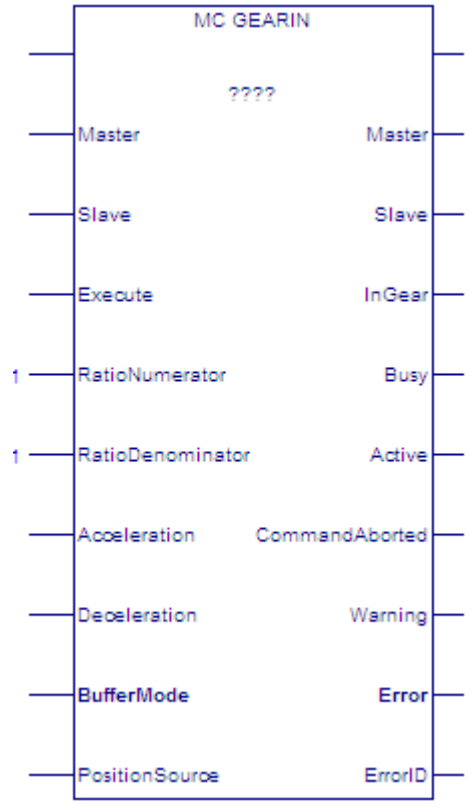
  Zoom– for zooming in the X and Y axes

- 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

MC_GearIn

LD	FBD	ST
		Formal convention: [instance name](Master := [input], Slave := [input], Execute := [input], RatioNumerator := 1, RatioDenominator := 1, Acceleration := [input], Deceleration := [input], BufferMode := [input], PositionSource := [input], InGear => [output], Busy => [output], Active => [output], CommandAborted => [output], Warning => [output], Error => [output], ErrorID => [output]);

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.

Gear velocity =
RatioNumerator/RatioDenominator × 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

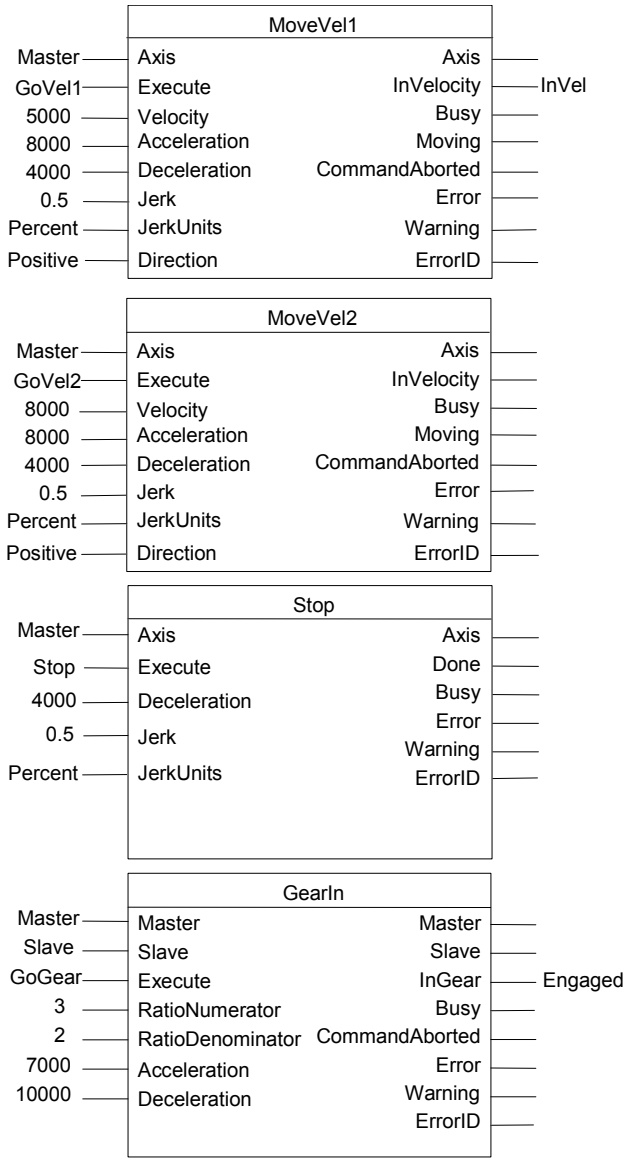
Parameter	*Description*	*Allowed Data Types*	*Initial Value*
Input_Output Parameters			
Master	Reference to master axis.	AXIS_REF	N/A
Slave	Reference to slave axis. Virtual Axis (Axis 5) not supported.	AXIS_REF	N/A
Inputs			
Execute	Start the gearing process at the rising edge.	LD: flow Other languages: 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^2). 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^2). 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

Parameter	Description	Allowed Data Types	Initial Value
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
Outputs			
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

MC_GearIn Example

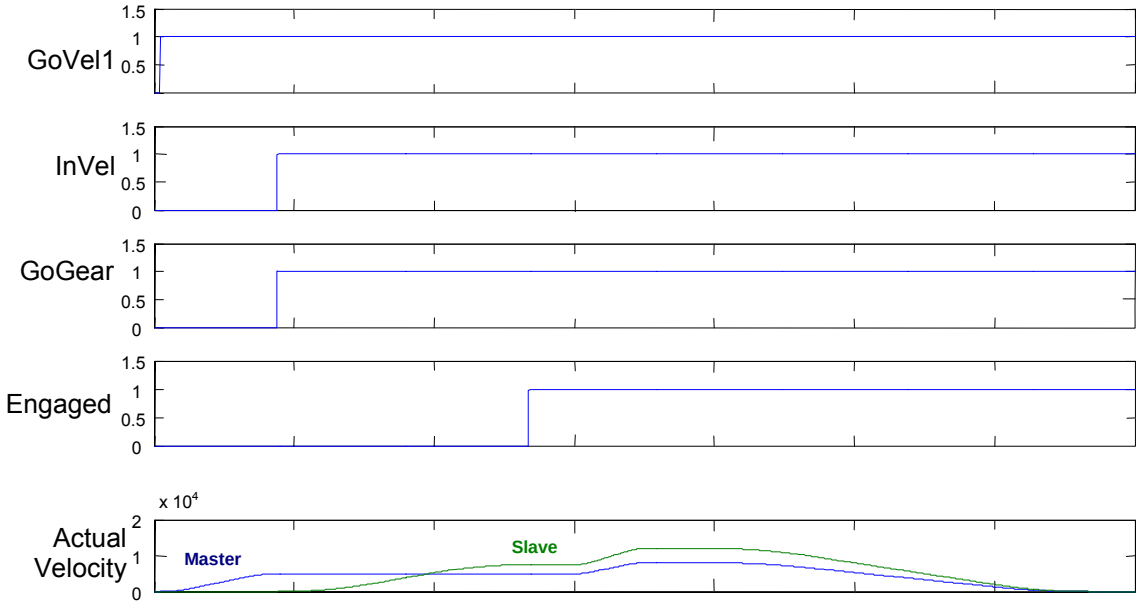
In this example, two MC_MoveVelocity function blocks are executed on a Master axis. MC_GearIn is executed and the Slave axis ramps up to the velocity of the Master axis and follows it with a 3/2 gear ratio. The GoGear input starts the execution of the MC_GearIn and the Engaged output indicates when the Slave is in gear.

MC_GearIn Example Block Diagram

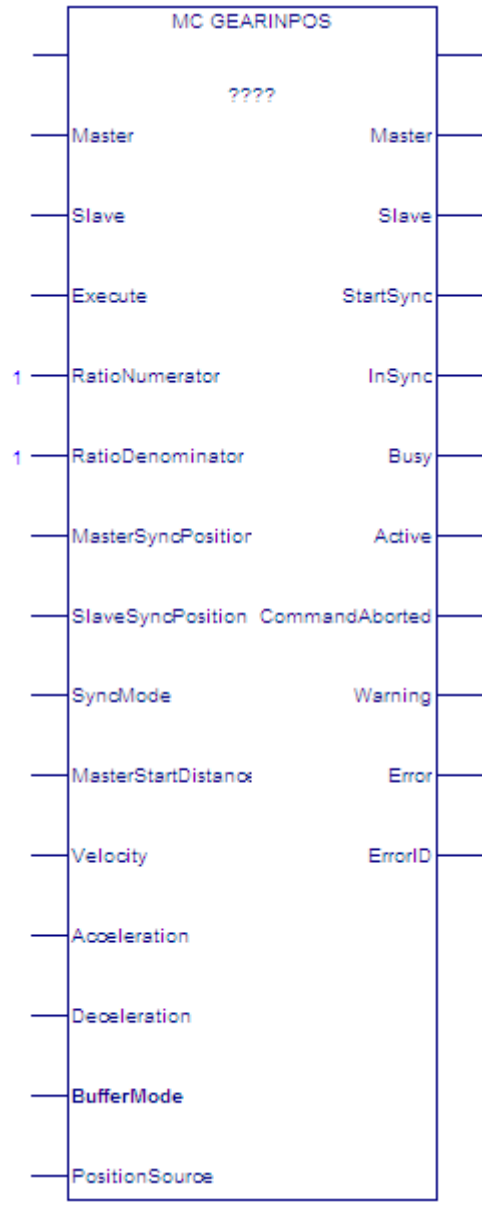


MC_GearIn

MC_GearIn Example Timing Diagram



MC_GearInPos

LD	FBD	ST
		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 => [output], InSync => [output], Busy => [output], Active => [output], CommandAborted => [output], Warning => [output], Error => [output], ErrorID => [output]);

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.

Start Position = MasterSyncPosition – 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 "Synchronized Motion Function Block Status" in Chapter 7.

Changes to the master axis' configured *Low Limit, Range* or *End of Travel* limits (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

Parameter	*Description*	*Allowed Data Types*	*Initial Value*
	Inputs		
Execute	Starts the gearing process at the rising edge.	LD: flow Other languages: all except constants	0
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	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. **Note:** 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	LREAL	0
Deceleration	Maximum Deceleration during the time difference StartSync and InSync.	LREAL	0
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

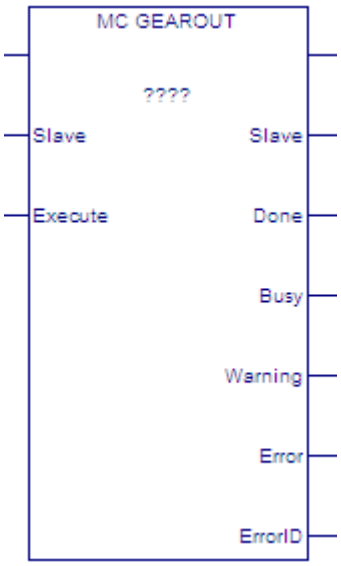
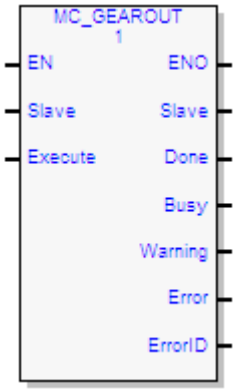
Parameter	Description	Allowed Data Types	Initial Value
PositionSource	Specifies the source on the master to follow. Actual position: Source is the configured feedback device. Commanded position: Source is the instantaneous position generated by the PMM's internal path generator. If Actual Position is selected and Axis 5 is the master, Axis 5 must use an External Encoder.	MC_PositionSource	0
Outputs			
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.

MC_GearOut

LD	FBD	ST
		Formal convention: [instance name](Slave := [input], Execute := [input], Done => [output], Busy => [output], Warning => [output], Error => [output], ErrorID => [output]);

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

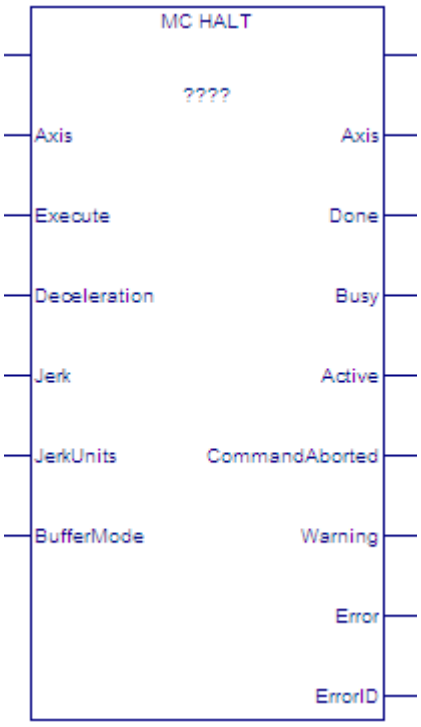
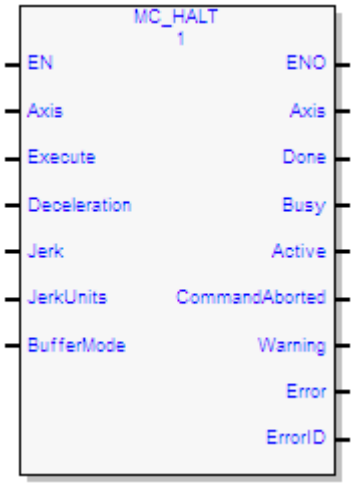
Instance Variable	Description	Allowed Data Types	Initial Value
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as *????* in LD.)	MC_GEAROUT	NA

Parameter	Description	Allowed Data Types	Initial Value
Input_Output Parameters			
Slave	Reference to slave axis. Virtual Axis (Axis 5) not supported.	AXIS_REF	N/A

MC_GearOut

Parameter	*Description*	*Allowed Data Types*	*Initial Value*
	Inputs		
Execute	Start to disengage the slave from the master.	LD: flow Other languages: all except constants	0
	Outputs		
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.		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

MC_Halt

LD	FBD	ST
		Formal convention: [instance name](Axis := [input], Execute := [input], Deceleration := [input], Jerk := [input], JerkUnits := [input], BufferMode := [input], Done => [output], Busy => [output], Active => [output], CommandAborted => [output], Warning => [output], Error => [output], ErrorID => [output]);

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 is 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.

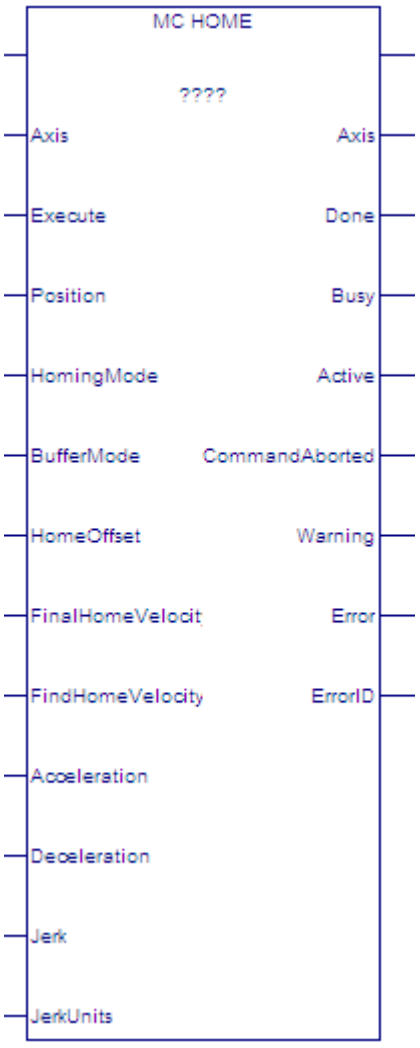
Execution type: Immediate execution/deferred response.

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^2]	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^3 or %)]	LREAL For details, see "Jerk and JerkUnits" in Chapter 5.	0
JerkUnits	Selects units for Jerk input: UU/sec^3 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 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 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_Home

LD	FBD	ST
		Formal convention: [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 => [output], Busy => [output], Active => [output], CommandAborted => [output], Warning => [output], Error => [output], ErrorID => [output]);

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.

The axis must be powered on during an entire home cycle. Note that 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. No motion function blocks that cause motion can be executed unless the PositionValid bit is ON. 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.

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 three modes, selected by the HomingMode input. For details and examples of the three homing modes, refer to page 6-72.

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.

Axis 5 does not support this mode.

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.

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***
Input_Output Parameters			
Axis	Axis that receives function block command.	AXIS_REF	N/A
Inputs			
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: LimitSwitchRefPulse – Axes 1–4 only RefPulse and RefPulseNeg – All axes	MC_HomingMode For details, refer to page 6-71.	0
BufferMode	Defines the behavior of the axis. Allowed modes are Aborting, Buffered, and Blending.	MC_BUFFERMODE	0

Parameter	Description	Allowed Data Types	Initial Value
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^2]	LREAL	100,000
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^2]	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^3 or %)]	LREAL For details, see "Jerk and JerkUnits" in Chapter 5.	0
JerkUnits	Selects units for Jerk input: UU/sec^3 or %	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.

Homing Modes

Limit Switch Reference Pulse Mode (Axes 1—4 motor feedback devices only)

This mode is also called the “Absolute Home Switch” mode. If the *HomingMode* is set to *LimitSwitchRefPuls*e, 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.

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.

1. The axis is moved in the negative direction at the *Find Home Velocity* until the Home Switch input closes.
2. The axis decelerates and stops.
3. The axis is accelerated in the positive direction and moved at the *Find Home Velocity* until the Home Switch input opens.
4. The axis decelerates and stops.
5. The axis accelerates in the negative direction and moved at the *Final Home Velocity* until the Home Switch input closes.
6. The axis continues negative motion at the *Final Home Velocity* until an Encoder Marker pulse is sensed. The marker establishes the home reference position.
7. The axis decelerates and stops (at a position past the marker pulse).
8. The axis is moves at the *Find Home Velocity* 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 Encoder Marker pulse.
10. The PMM sets the *Commanded Position* value to the *Home Position* value. *Actual Position* is set to *Home Position – Position Error*.

    The *PositionValid* status flag (parameter 1201) is set to indicate the home cycle is complete.

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.***

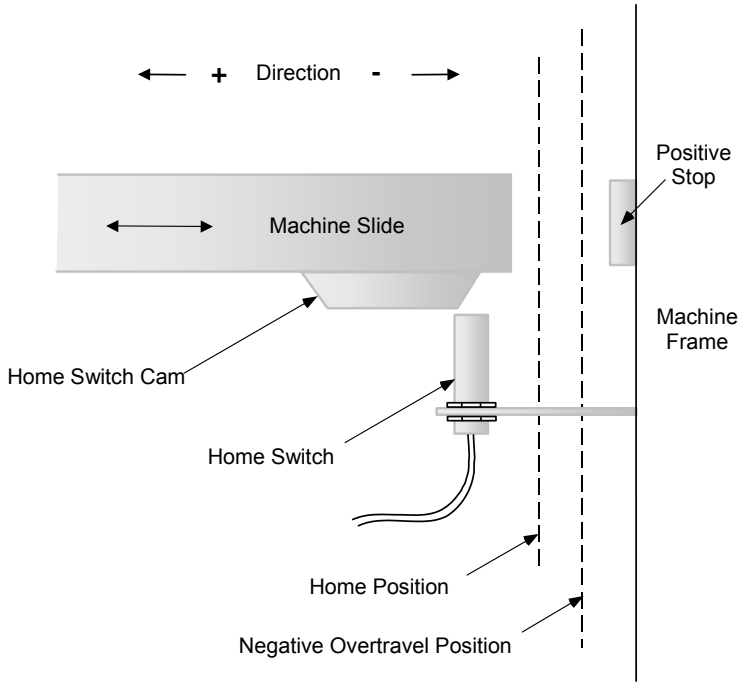
Note the relationships of the home position, the negative overtravel position, and the positive stop position. A small amount of distance is provided in the negative direction between the home position and the negative overtravel position. This is to allow some working room for adjustment and setup of 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



Reference Pulse (Move+) and Reference Pulse Negative(Move–) Modes (All Axes)

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.

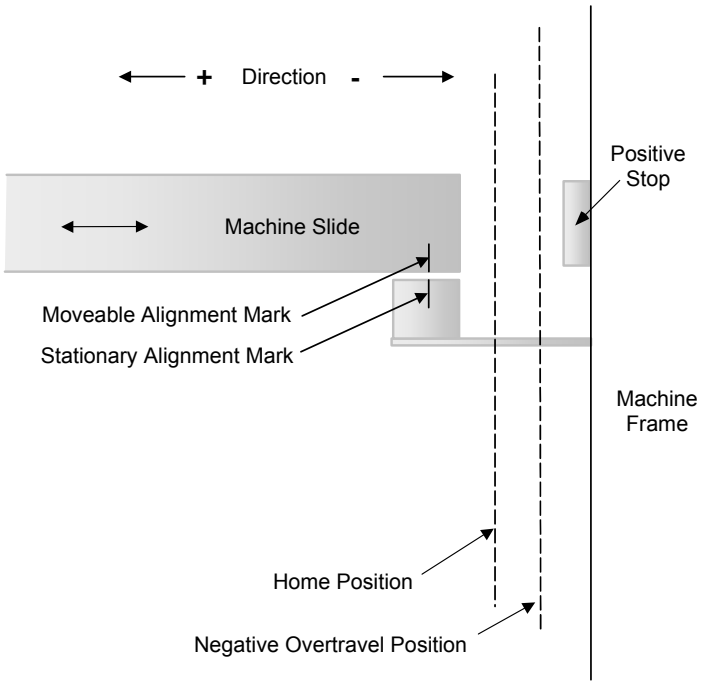
Find Home Routine for Reference Pulse (Move +) or Reference Pulse Negative (Move –)

When the MC_Home function block is executed with *Find Home Mode* set to *RefPulse* or *RefPulseNeg*, the following sequence of events occurs:

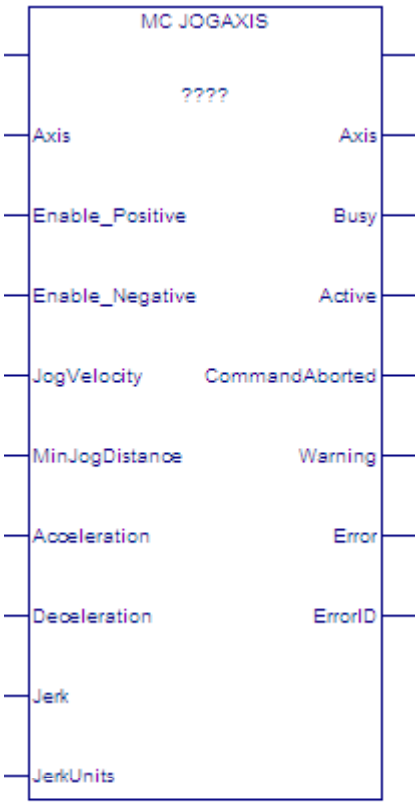
1. The axis is accelerated at the *Acceleration* rate and moved at the configured *Final Home Velocity* (positive direction for RefPulse, negative direction for RefPulseNeg) until an Encoder Marker pulse is sensed. This marker pulse establishes the home reference position.
2. The axis is stopped at a position past the marker pulse using the configured *Deceleration*.
3. 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.
4. The axis is stopped at the configured *Deceleration*.
5. 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.

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.



MC_JogAxis

LD	FBD	ST
		Formal convention: [instance name](Axis := [input], Enable_Positive := [input], Enable_Negative := [input], JogVelocity := [input], MinJogDistance := [input], Acceleration := [input], Deceleration := [input], Jerk := [input], JerkUnits := [input], Busy => [output], Active => [output], CommandAborted => [output], Warning => [output], Error => [output], ErrorID => [output]);

This function block jogs an axis forward or backward at the specified velocity, acceleration and jerk.

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. See the “Jog to Software End of Travel with Warning” example 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 (NOTE: 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^2]	LREAL	0
Deceleration	The maximum jog deceleration rate (energy is decreasing). Maximum deceleration is not necessarily reached. (Always positive.)[Units = UU/sec^2]	LREAL	0

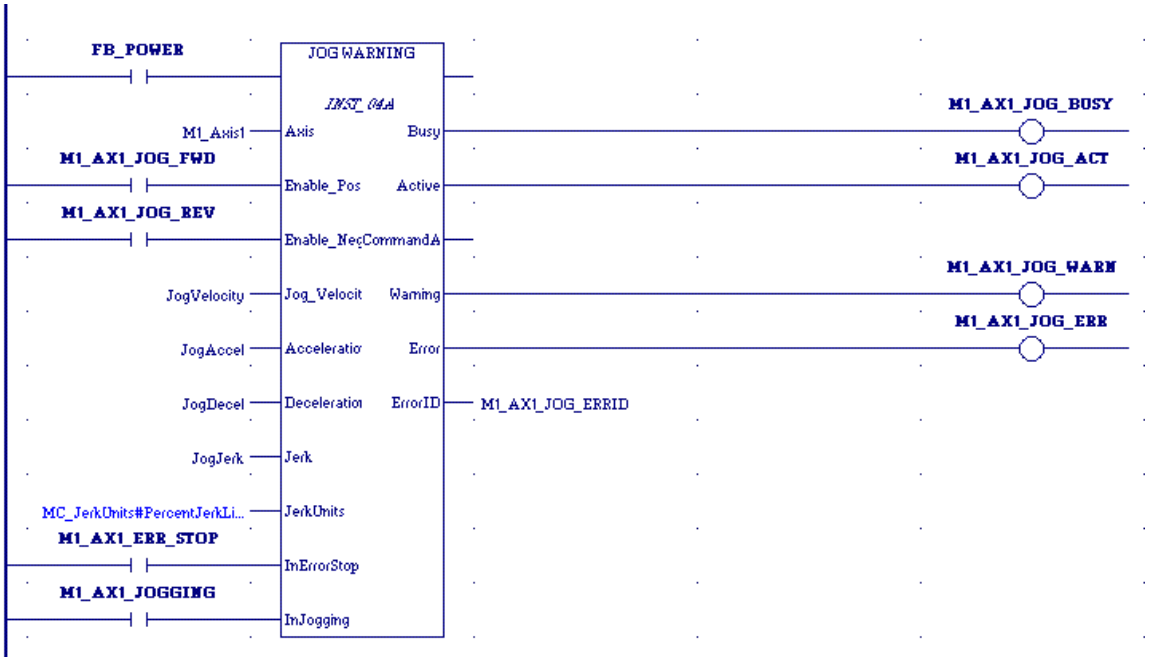
Jerk	The Jerk (rate of change in acceleration) used for the move. (Always positive.) If set to 0, jerk will be unlimited. [Units = UU/sec^3 or %)]	LREAL For details, see "Jerk and JerkUnits" in Chapter 5.	0
JerkUnits	Selects units for the Jerk input: UU/sec^3 or %	MC_JERKUNITS.	0
Outputs			
Busy	Indicates 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

Example: Jog to Software End of Travel with Warning

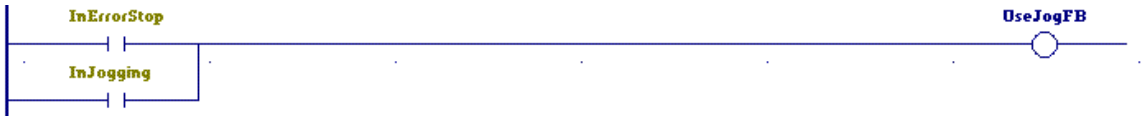
MC_JogAxis moves at a positive or negative velocity as long as an Enable input is held high. If the enable is held high as the axis approaches a Software End-of-Travel (SW-EOT), the PMM will not allow the SW-EOT position to be exceeded. The axis will be stopped at the SW-EOT and transition to ErrorStop.

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.

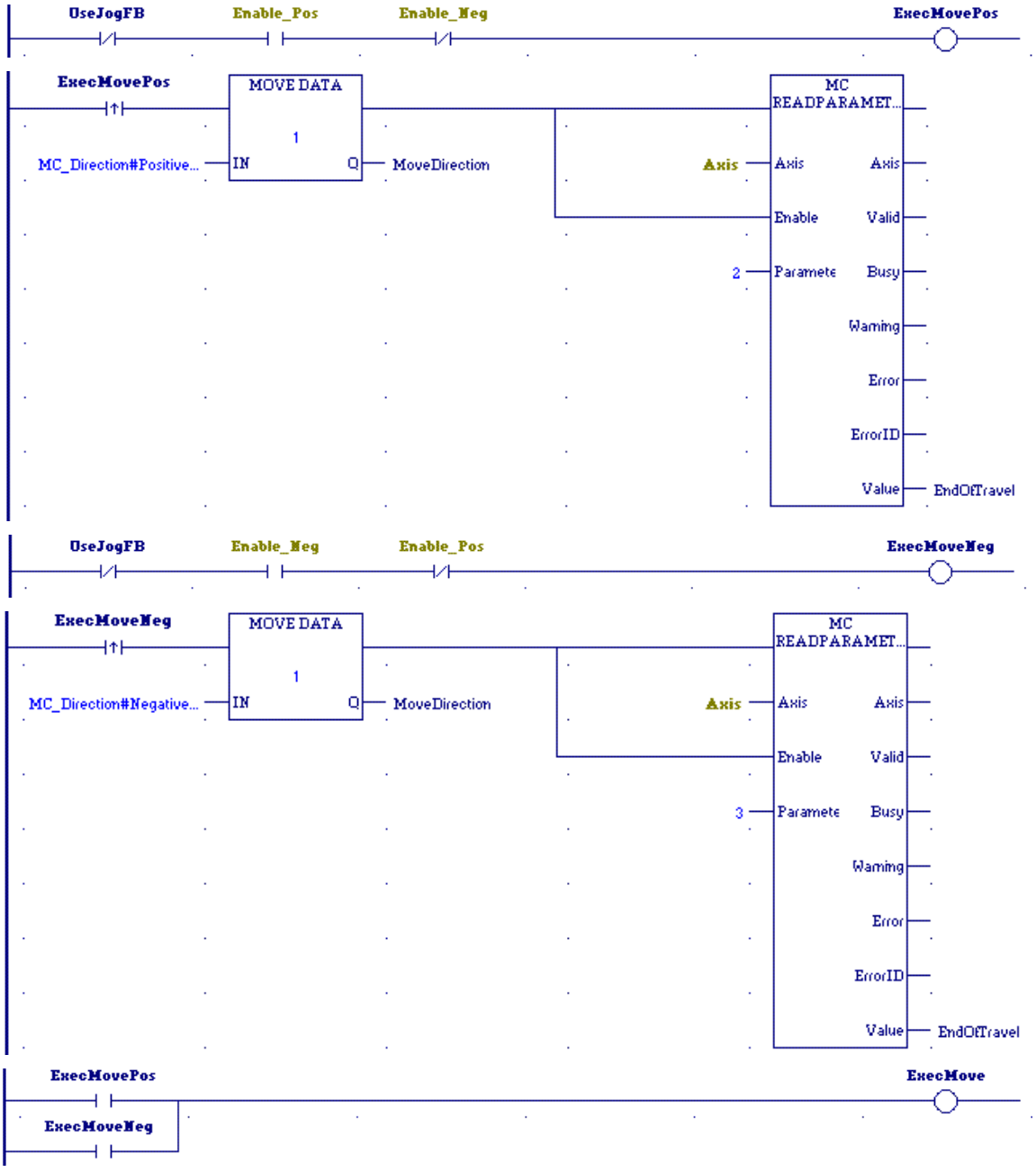
1. First define the UDFB, *JogWarning.*



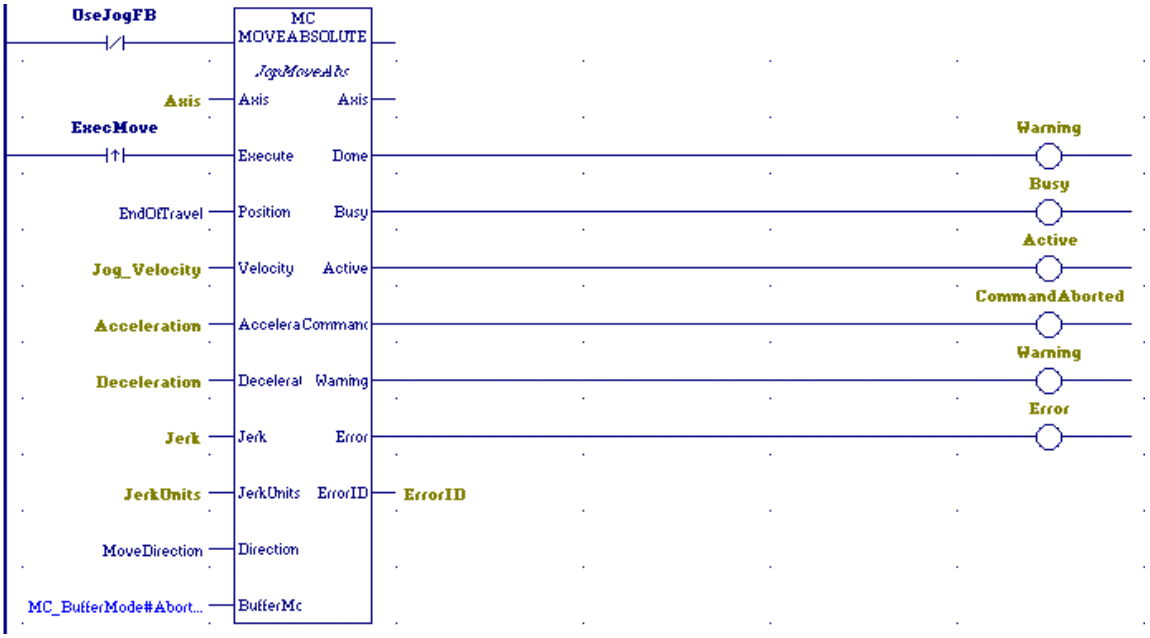
2. Two state bits are required: InErrorStop and InJogging. These bits must be read using MC_ReadAxisStatus. The bits allow the MC_JogAxis function block to be used if the axis is in the ErrorStop state (see step 6).



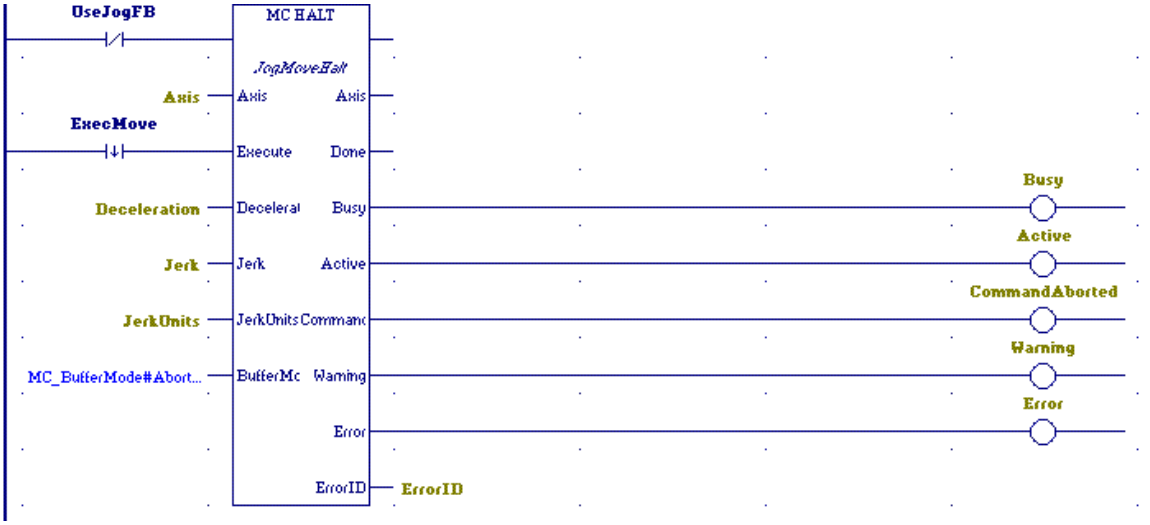
3. The following logic chooses the direction, finds the SW-EOT position for the current direction and sets the coil used to activate the MC_MoveAbsolute and MC_Halt function blocks.



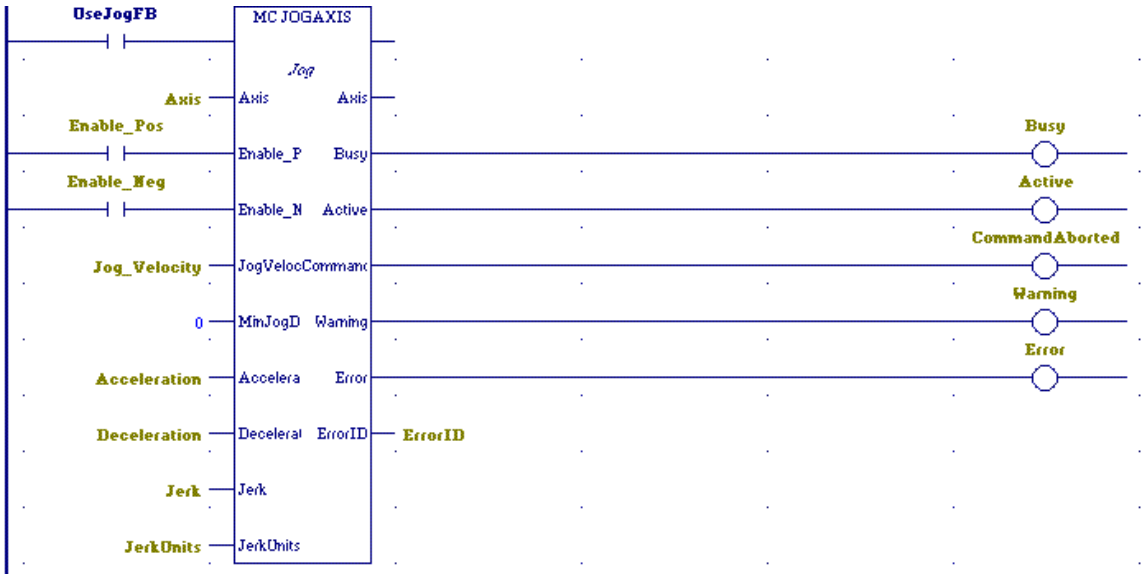
4. Next, use a POSCON (positive transition contact) to execute a MoveAbsolute in the desired direction to the SW-EOT.



5. Next, use a NEGCON (negative transition contact) to execute a Halt when the Enable_* goes low.

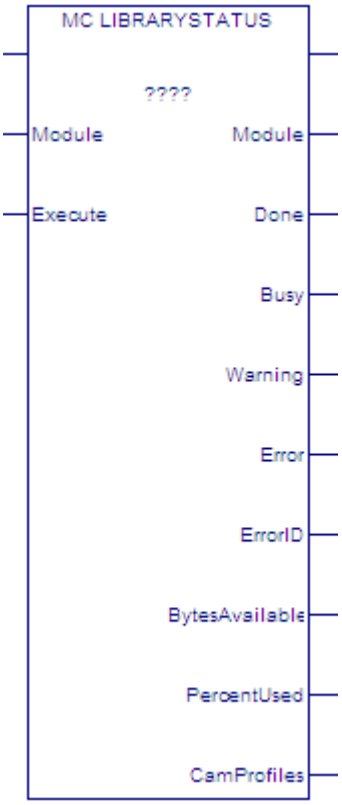
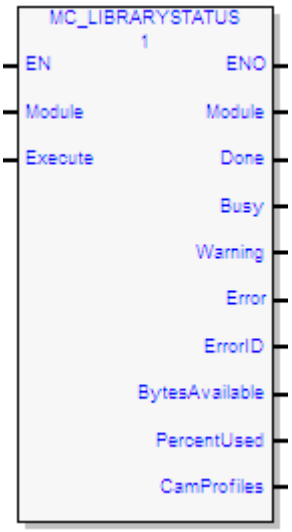


6. Lastly, if UseJogFB is true, use MC_JogAxis.



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.

MC_LibraryStatus

LD	FBD	ST
		Formal convention: [instance name](Module := [input], Execute := [input], Done => [output], Busy => [output], Warning => [output], Error => [output], ErrorID => [output], BytesAvailable => [output], PercentUsed => [output], CamProfiles => [output]);

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

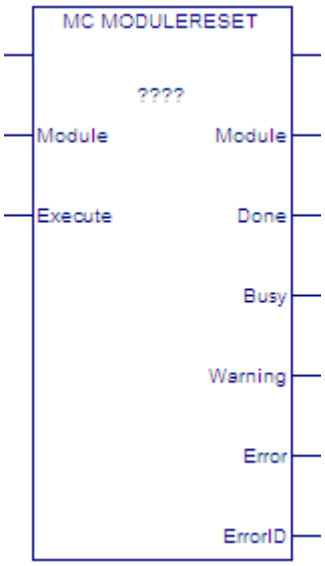
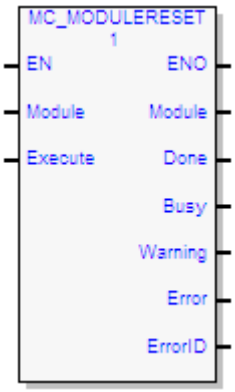
Instance Variable	***Description***	***Allowed Data Types***	***Initial Value***
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as *????* in LD.)	MC_LIBRARYSTATUS	NA
Parameter	***Description***	***Allowed Operands***	***Initial 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

MC_LibraryStatus

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*	The number of bytes available to hold cam profiles and motion programs.	DINT	0
PercentUsed*	The percentage of space currently used (0—100).	INT	0
CamProfiles*	The number of cam profiles loaded on the module.	INT	0

* These outputs are *not* cleared when Execute transitions low.

MC_ModuleReset

LD	FBD	ST
		Formal convention: [instance name](Module := [input], Execute := [input], Done => [output], Busy => [output], Warning => [output], Error => [output], ErrorID => [output]);

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 amplifier 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.

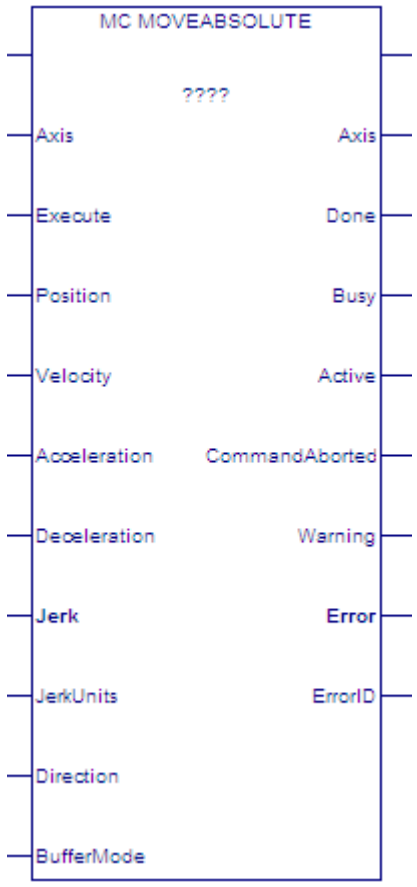
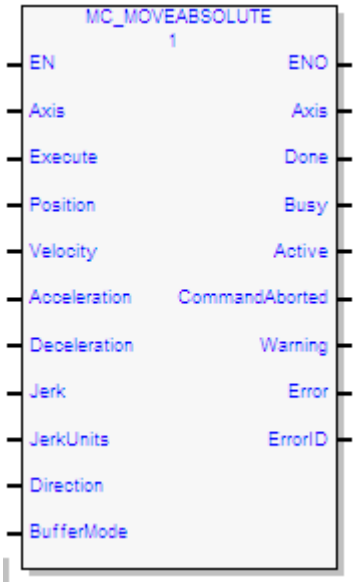
MC_ModuleReset

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	0

MC_MoveAbsolute

LD	FBD	ST
		Formal convention: [instance name](Axis := [input], Execute := [input], Position := [input], Velocity := [input], Acceleration := [input], Deceleration := [input], Jerk := [input], JerkUnits := [input], Direction := [input], BufferMode := [input], Done => [output], Busy => [output], Active => [output], CommandAborted => [output], Warning => [output], Error => [output], ErrorID => [output]);

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*
	Input_Output Parameters		
Axis	Axis that receives function block command.	AXIS_REF	NA
	Inputs		
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^2]	LREAL	0.0
Deceleration	The maximum move deceleration rate (energy is decreasing). Maximum deceleration is not necessarily reached. (Always positive.) [Units = UU/second^2]	LREAL	0.0
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^3 or %)]	LREAL For details, see “Jerk and JerkUnits” in Chapter 5.	0.0
JerkUnits	Selects units for Jerk input: UU/sec^3 or %	MC_JERKUNITS	0
Direction	One of four values: positive direction, shortest way, negative direction, current direction.	MC_DIRECTION. For details, see page 6-89.	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

Parameter	Description	Allowed Data Types	Initial Value
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
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

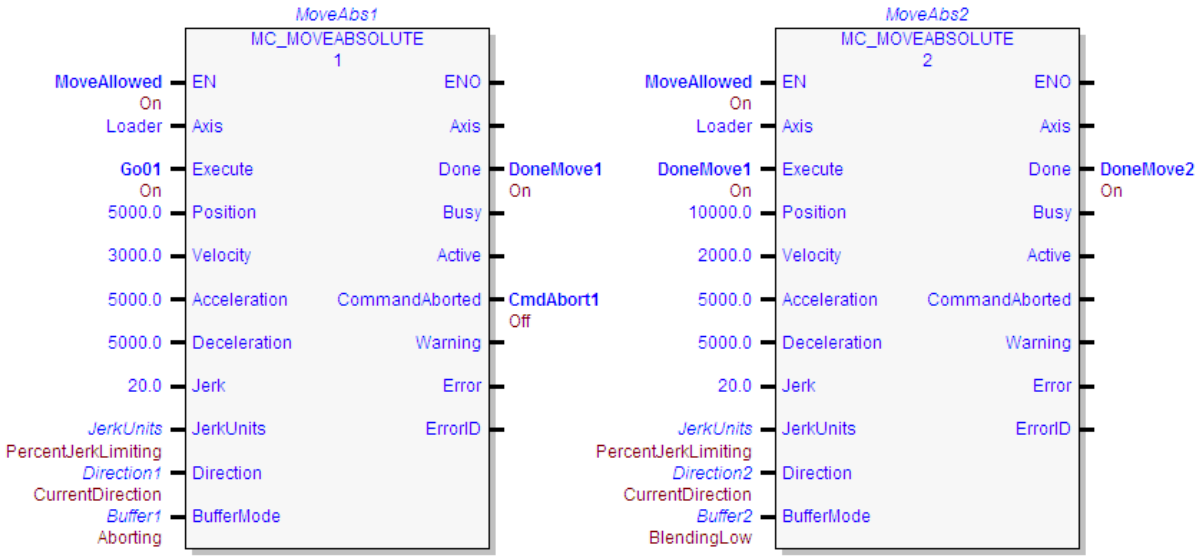
MC_DIRECTION Data Type

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:

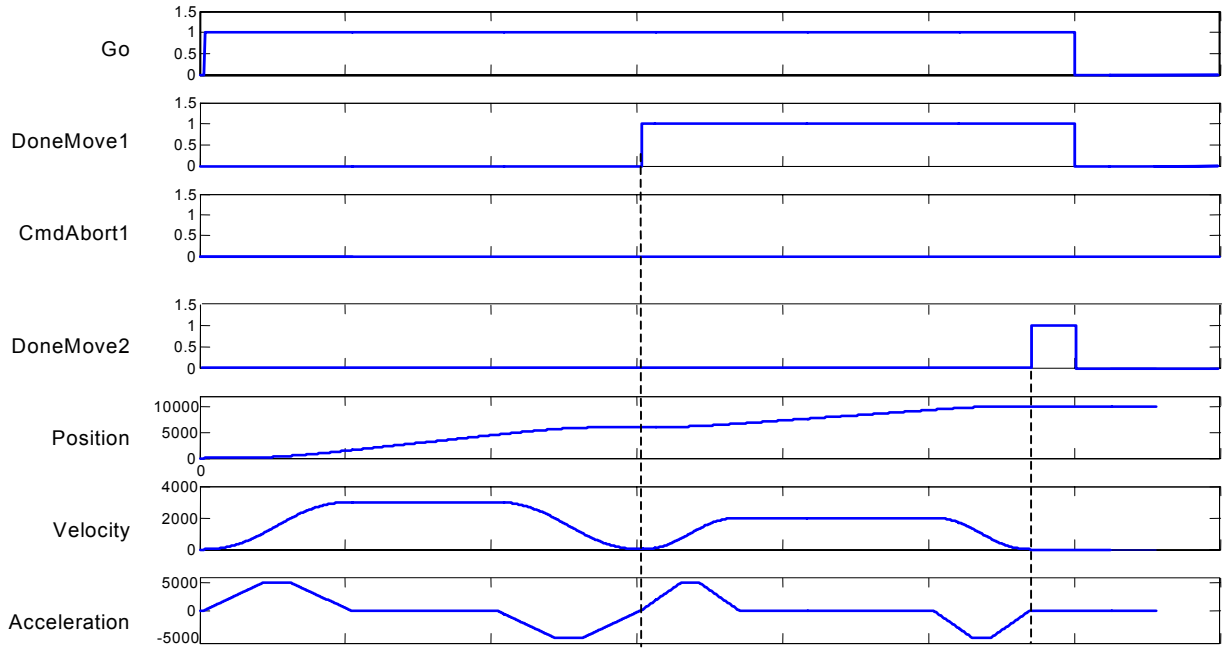
Value	Definition
Positive direction	If Axis Direction is configured for Normal operation, motor shaft rotation will be CCW, looking into the shaft.
	If Axis Direction is configured for Reverse operation, motor shaft rotation will be CW, looking into the shaft.
Shortest way	Axis moves in the direction 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.

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 axis' commanded position, which is 5000 UU. The Done output, DoneMove1 sets Go2, which begins the execution of MoveAbs2. MoveAbs2 executes until the axis' commanded position reaches 10000 UU.



The following figure shows the timing of the function block input and output parameters in relationship to the axis position, velocity and acceleration.



MC_MoveAdditive

LD	FBD	ST
		Formal convention: [instance name](Axis := [input], Execute := [input], Distance := [input], Velocity := [input], Acceleration := [input], Deceleration := [input], Jerk := [input], JerkUnits := [input], BufferMode := [input], Done => [output], Busy => [output], Active => [output], CommandAborted => [output], Warning => [output], Error => [output], ErrorID => [output]);

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 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

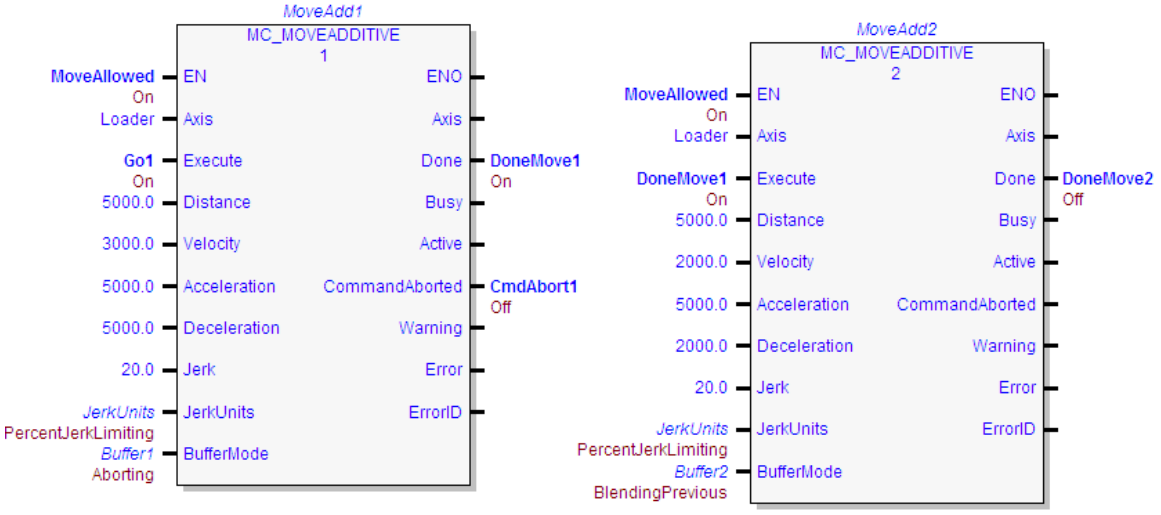
Instance Variable	*Description*	*Allowed Data Types*	*Initial Value*
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as *????* in LD.)	MC_MOVEADDITIVE	NA

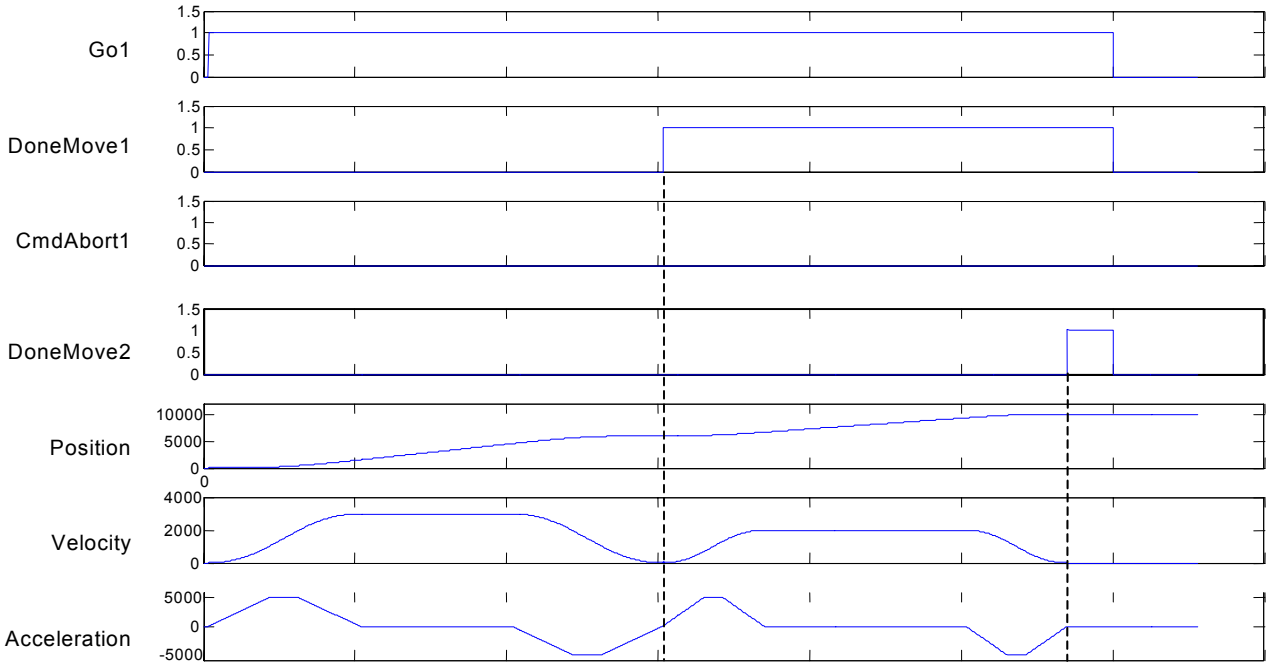
Parameter	*Description*	*Allowed Data Types*	*Initial Value*
	Input_Output Parameters		
Axis	Axis that receives function block command.	AXIS_REF	NA
	Inputs		
Execute	Start motion at rising edge.	LD: flow Other languages: all except constants	0
Distance	The relative distance to be added to the current 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^2]	LREAL	0
Deceleration	The maximum move deceleration rate(energy is decreasing). Maximum deceleration is not necessarily reached. (Always positive.)[Units = UU/second^2]	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^3 or %)]	LREAL For details, see "Jerk and JerkUnits" in Chapter 5.	0
JerkUnits	Selects units for Jerk input: UU/sec^3 or %.	MC_JERKUNITS	0
BufferMode	Defines the axis buffering behavior. Valid modes are Aborting, Buffered and Blending.	MC_BUFFERMODE	0

Parameter	*Description*	*Allowed Data Types*	*Initial Value*
	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	0

Example

The following figure shows two MC_MoveAdditive commands executed one right after the other. 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.





MC_MoveRelative

LD	FBD	ST
		Formal convention: [instance name](Axis := [input], Execute := [input], Distance := [input], Velocity := [input], Acceleration := [input], Deceleration := [input], Jerk := [input], JerkUnits := [input], BufferMode := [input], Done => [output], Busy => [output], Active => [output], CommandAborted => [output], Warning => [output], Error => [output], ErrorID => [output]);

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.

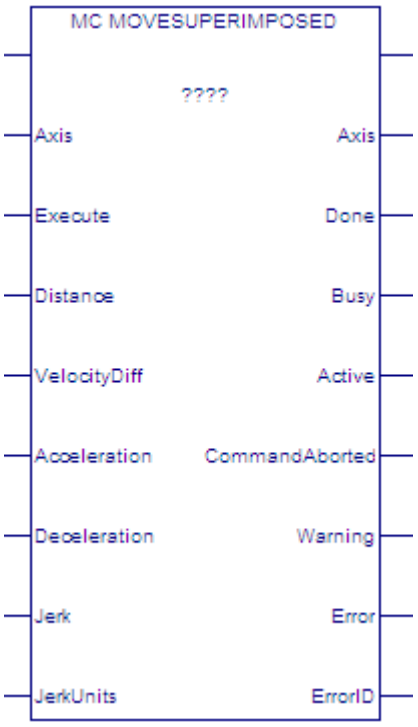
Note that the Done output of MC_MoveRelative signifies that the axis commanded position has achieved the position defined by the instruction. The actual position of the axis might still lag the commanded position when the Done output is energized.

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^2]	LREAL	0
Deceleration	The maximum move deceleration rate (energy is decreasing). Maximum deceleration is not necessarily reached. (Always positive.)[Units = UU/second^2]	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^3 or %)]	LREAL For details, see "Jerk and JerkUnits" in Chapter 5.	0
JerkUnits	Selects units for Jerk input: UU/sec^3 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	0

MC_MoveSuperimposed

LD	FBD	ST
		Formal convention: [instance name](Axis := [input], Execute := [input], Distance := [input], VelocityDiff := [input], Acceleration := [input], Deceleration := [input], Jerk := [input], JerkUnits := [input], Done => [output], Busy => [output], Active => [output], CommandAborted => [output], Warning => [output], Error => [output], ErrorID => [output]);

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

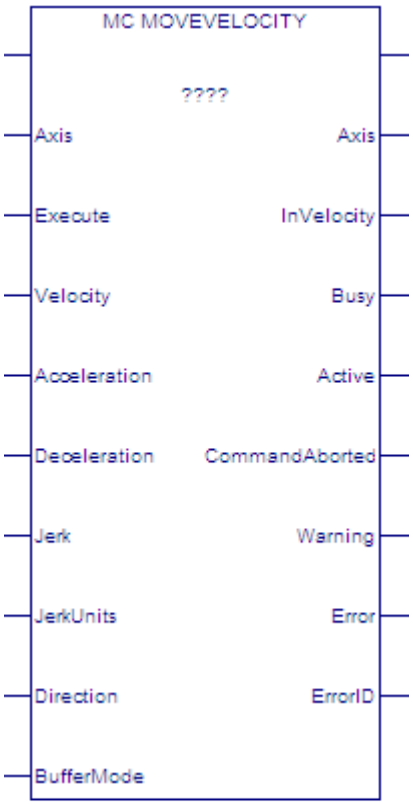
Instance Variable	Description	Allowed Data Types	Initial Value
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as *????* in LD.)	MC_MOVESUPERIMPOSED	NA

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

MC_MoveSuperimposed

Parameter	*Description*	*Allowed Data Types*	*Initial Value*
	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^2). 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^2). 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^3 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, see "Jerk and JerkUnits" in Chapter 5.	0
JerkUnits	Selects units for Jerk input: UU/sec^3 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

MC_MoveVelocity

LD	FBD	ST
		Formal convention: [instance name](Axis := [input], Execute := [input], Velocity := [input], Acceleration := [input], Deceleration := [input], Jerk := [input], JerkUnits := [input], Direction := [input], BufferMode := [input], InVelocity => [output], Busy => [output], Active => [output], CommandAborted => [output], Warning => [output], Error => [output], ErrorID => [output]);

This function block commands a move to the commanded velocity.

Execution type: Immediate execution/deferred response.

Operands

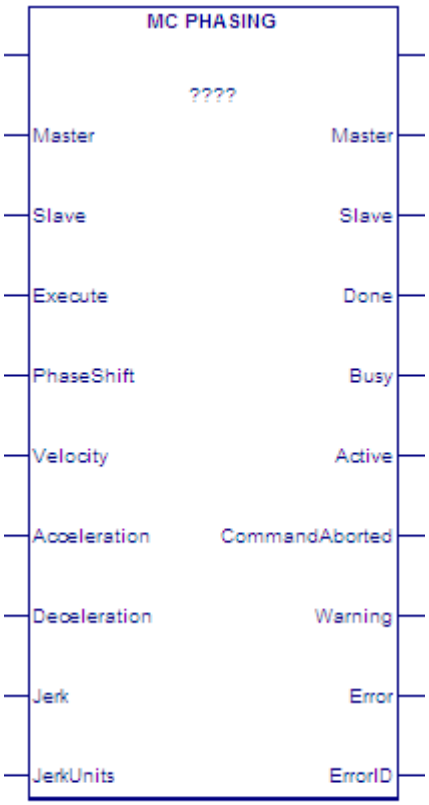
Instance Variable	Description	Allowed Data Types	Initial Value
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as *????* in LD.)	MC_MOVEVELOCITY	NA

Parameter	Description	Allowed Data Types	Initial Value
Input_Output Parameters			
Axis	Axis that receives function block command.	AXIS_REF	N/A

MC_MoveVelocity

Parameter	*Description*	*Allowed Data Types*	*Initial Value*
	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^2]	LREAL	0
Deceleration	The maximum move deceleration rate (energy is decreasing). Maximum deceleration is not necessarily reached. (Always positive.) [Units = UU/second^2]	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^3 or %)]	LREAL For details, see "Jerk and JerkUnits" in Chapter 5.	0
JerkUnits	Selects units for Jerk input: UU/sec^3 or %	MC_JERKUNITS	0
Direction	One of four enumerated values: positive direction, shortest way, negative direction, current direction.	MC_DIRECTION 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.	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

MC_Phasing

LD	FBD	ST
		Formal convention: [instance name](Master := [input], Slave := [input], Execute := [input], PhaseShift := [input], Velocity := [input], Acceleration := [input], Deceleration := [input], Jerk := [input], JerkUnits := [input], Done => [output], Busy => [output], Active => [output], CommandAborted => [output], Warning => [output], Error => [output], ErrorID => [output]);

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.

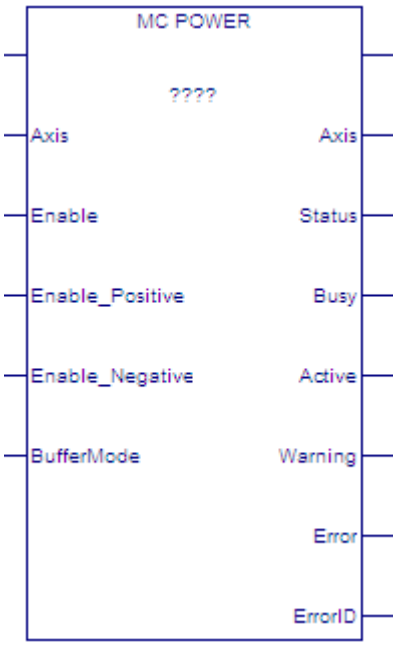
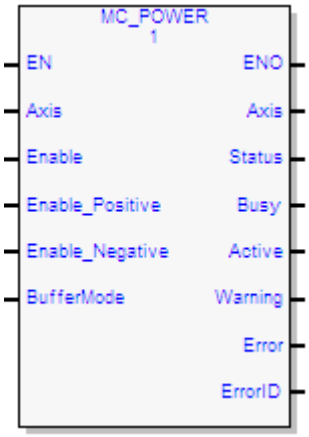
Execution type: Immediate execution/deferred response

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^2)	LREAL	0
Deceleration	Maximum Deceleration to reach phase difference (Uu/s^2)	LREAL	0
Jerk	(UU/sec^3 or %) Maximum Jerk to reach phase difference	LREAL For details, see “Jerk and JerkUnits” in Chapter 5.	0
JerkUnits	Selects units for Jerk input: UU/sec^3 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	0

MC_Power

LD	FBD	ST
		Formal convention: [instance name](Axis := [input], Enable := [input], Enable_Positive := [input], Enable_Negative := [input], BufferMode := [input], Status => [output], Busy => [output], Active => [output], Warning => [output], Error => [output], ErrorID => [output]);

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 amplifier. The Status output shows the actual state of the power (based on feedback from the amplifier). 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). 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. See “MC_Power Input Combinations” table for details.	LD: flow Other languages: all except constants	0
Enable_Positive	Permits motion in the positive direction. See “MC_Power Input Combinations” table for details.	LD: flow Other languages: all except constants	0
Enable_Negative	Permits motion in the negative direction. See “MC_Power Input Combinations” table 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 Other languages: all except constants	0
Busy	Set when the Enable input is 1 and the function block has not finished executing.		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 superceded 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 “Buffering MC_Power” in Chapter 5.

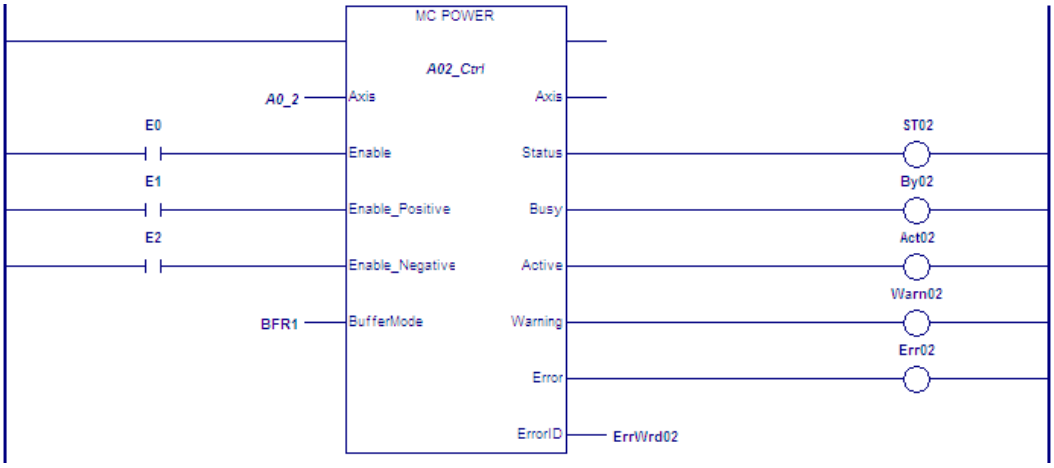
MC_Power Input Combinations

The following table shows possible input combinations and the resulting motion that is enabled. The “x” indicates a value that is ignored.

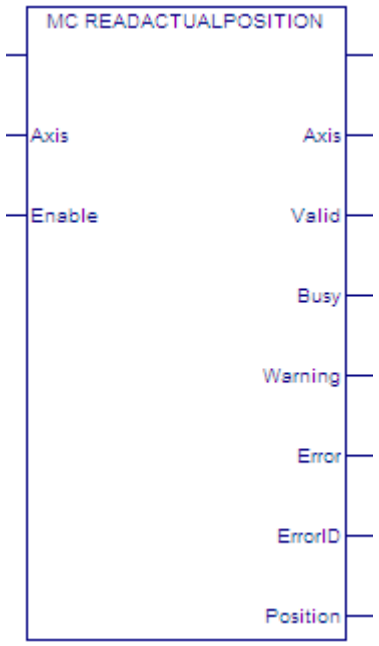
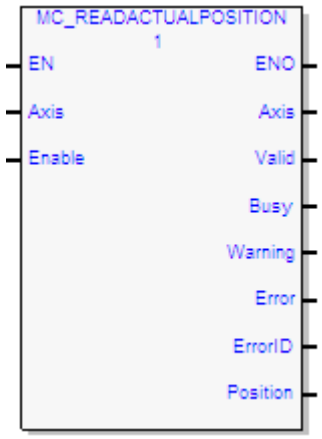
Value of Enable	Value of Enable Positive	Value of Enable Negative	Result
0	x	x	All motion disabled
1	0	0	Motion enabled in both the positive and negative directions
1	1	0	Motion enabled in the positive direction only
1	0	1	Motion enabled in the negative direction only
1	1	1	Motion enabled in both the positive and negative directions

Example

In the following sample logic, the MC_Power function block controls an axis named A0_2.



MC_ReadActualPosition

LD	FBD	ST
		Formal convention: MC_READACTUALPOSITION(Axis := [input], Enable := [input], Valid => [output], Busy => [output], Warning => [output], Error => [output], ErrorID => [output], Position => [output]);

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.

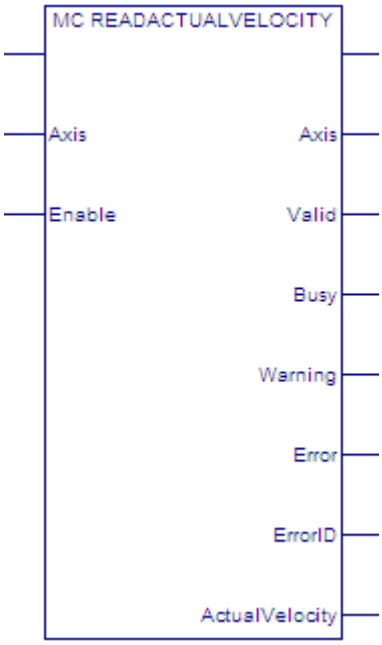
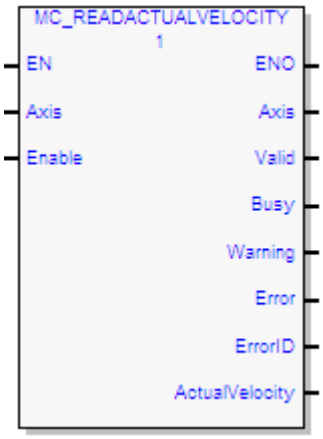
If reading Actual Position for a Virtual Axis (Axis 5), valid data will be returned only if an external encoder is used.

Execution type: Immediate execution/immediate response.

Operands

Parameter	Description	Allowed Data Types	Initial Value
Input_Output Parameters			
Axis	The axis on which the function is to be executed.	AXIS_REF	N/A
Inputs			
Enable	Read the axis' position while enabled.	LD: flow Other languages: all except constants	N/A
Outputs			
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

MC_ReadActualVelocity

LD	FBD	ST
		Formal convention: MC_READACTUALVELOCITY(Axis := [input], Enable := [input], Valid => [output], Busy => [output], Warning => [output], Error => [output], ErrorID => [output], ActualVelocity => [output]);

This function returns the actual axis velocity while the Enable input is set. The Valid output is 1 when the Actual Velocity is valid.

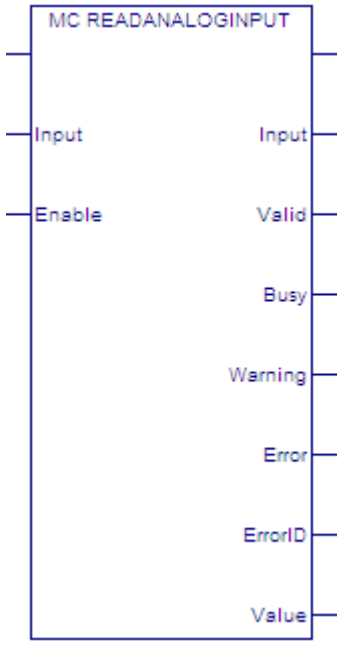
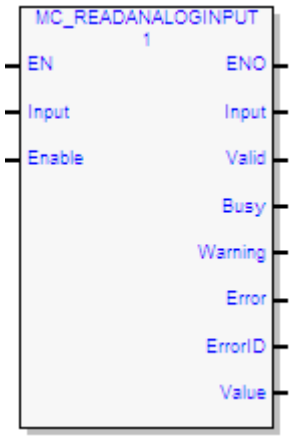
If reading Actual Velocity for a Virtual Axis (Axis 5), valid data will be returned only if an external encoder is used.

Execution type: Immediate execution/immediate response.

Operands

Parameter	Description	Allowed Data Types	Initial Value
	Input_Output Parameters		
Axis	The axis on which the function is to be executed.	AXIS_REF	N/A
	Inputs		
Enable	Read the actual velocity while enabled.	LD: flow Other languages: all except constants	N/A
	Outputs		
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.		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

MC_ReadAnalogInput

LD	FBD	ST
		Formal convention: MC_READANALOGINPUT(Input := [input], Enable := [input], Valid => [output], Busy => [output], Warning => [output], Error => [output], ErrorID => [output], Value => [output]);

This function provides the ability to monitor the ±10V 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 Chapter 8.

For specifications and connection details for the analog inputs, see Chapter 3, “Wiring and Connections.”

Execution type: Immediate execution/immediate response.

Operands

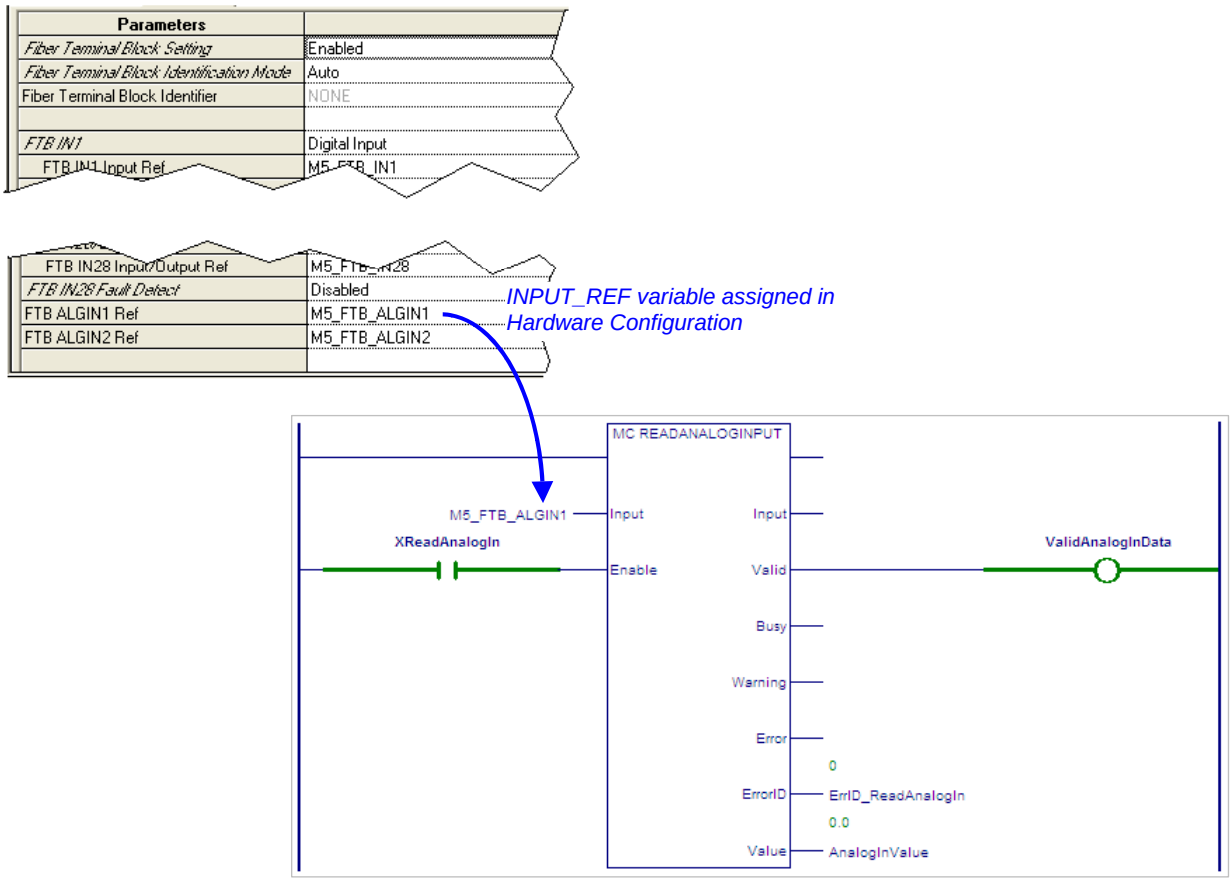
Parameter	Description	Allowed Data Types	Initial Value
	Input_Output Parameters		
Input	Reference to the input signal	INPUT_REF defined in the hardware configuration for the module associated with the FTB.	N/A
	Inputs		
Enable	Read the analog input while enabled.	LD: flow Other languages: all except constants	0

Outputs			
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

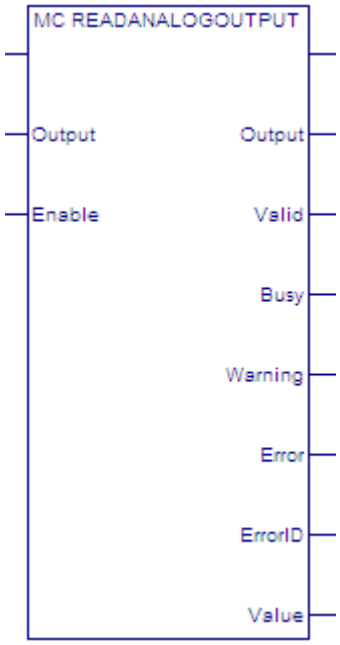
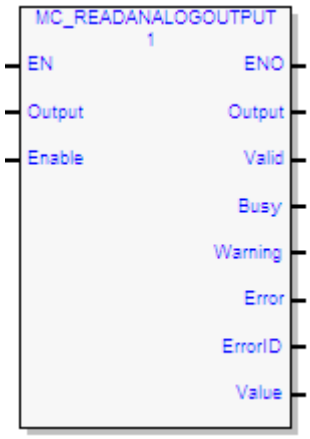
Input Example

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.



MC_ReadAnalogOutput

LD	FBD	ST
		Formal convention: MC_READANALOGOUTPUT(Output := [input], Enable := [input], Valid => [output], Busy => [output], Warning => [output], Error => [output], ErrorID => [output], Value => [output]);

This function provides the ability to monitor the ±10V 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 Chapter 8.

For specifications and connection details for the analog outputs, see Chapter 3, “Wiring and Connections.”

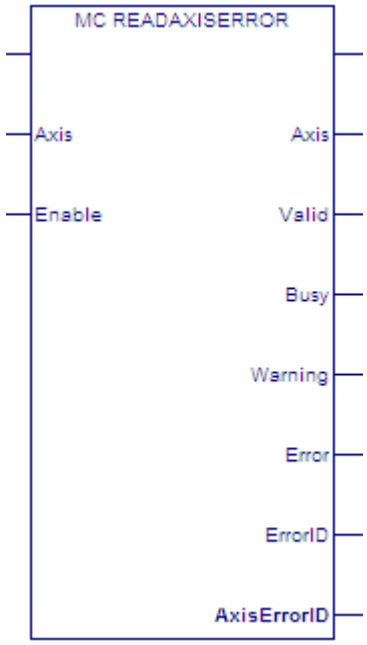
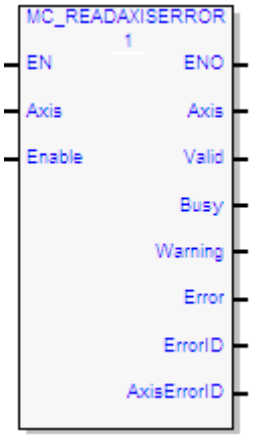
The operation of the MC_ReadAnalogOutput function is similar to that of the MC_ReadAnalogInput function, for which an example is provided on page 6-109.

Execution type: Immediate execution/immediate response.

Operands

Parameter	Description	Allowed Data Types	Initial Value
	Input_Output Parameters		
Output	Reference to the analog signal output.	OUTPUT_REF defined in the hardware configuration for the module associated with the FTB.	N/A
	Inputs		
Enable	Read the analog output while enabled.	LD: flow Other languages: all except constants	N/A
	Outputs		
Valid	Output 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 output signal. Units = volts.	LREAL	0

MC_ReadAxisError

LD	FBD	ST
		Formal convention: MC_READAXISERROR(Axis := [input], Enable := [input], Valid => [output], Busy => [output], Warning => [output], Error => [output], ErrorID => [output], AxisErrorID => [output]);

This function is used to read the current axis error or warning. MC_ReadAxisError returns the highest priority error or warning message.

To read multiple errors and warnings, use the MC_ReadEventQueue function block.

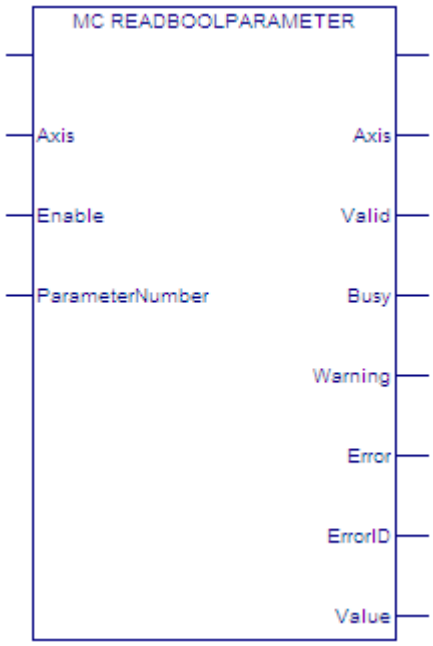
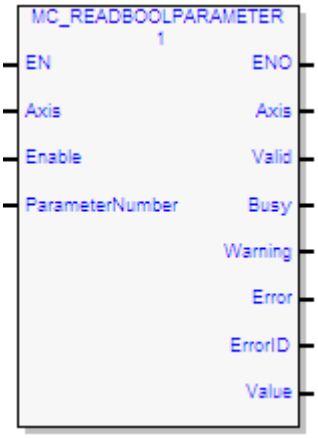
Execution type: Immediate execution/immediate response.

Operands

Parameter	Description	Allowed Data Types	Initial Value
Input_Output Parameters			
Axis	The axis on which the function is to be executed.	AXIS_REF	N/A
Inputs			
Enable	Read the axis error while enabled.	LD: flow Other languages: all except constants	N/A

Parameter	*Description*	*Allowed Data Types*	*Initial Value*
	Outputs		
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 Chapter 9 “Diagnostics.”	All except constants and variables located in %S.	0

MC_ReadBoolParameter

LD	FBD	ST
		Formal convention: MC_READBOOLPARAMETER(Axis := [input], Enable := [input], ParameterNumber := [input], Valid => [output], Busy => [output], Warning => [output], Error => [output], ErrorID => [output], Value => [output]);

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, see “Parameter Number Index” in Chapter 8.

Execution type: Immediate execution/immediate response.

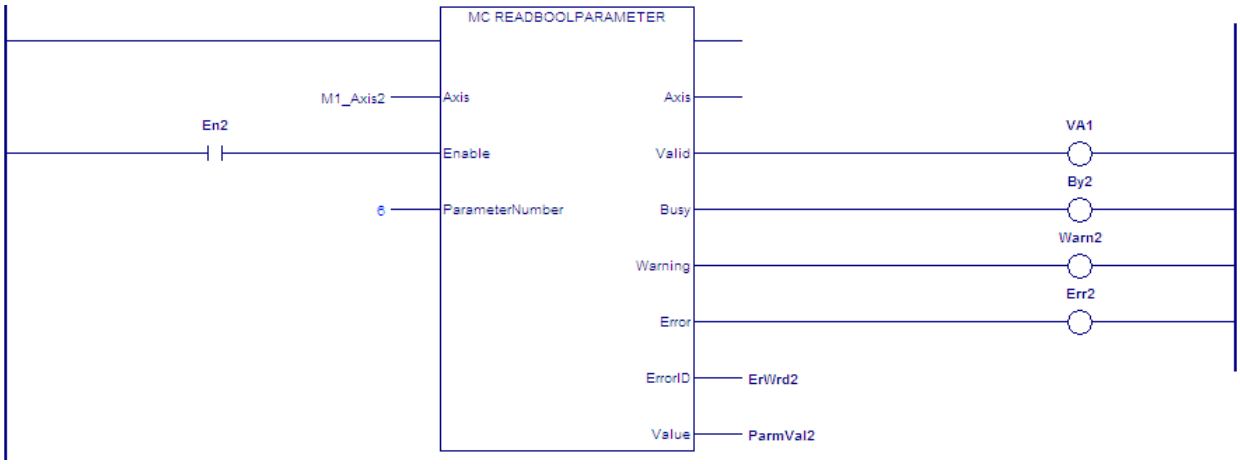
Operands

Parameter	Description	Allowed Data Types	Initial Value
	Input_Output Parameters		
Axis	Axis with parameter to read.	AXIS_REF	N/A
	Inputs		
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

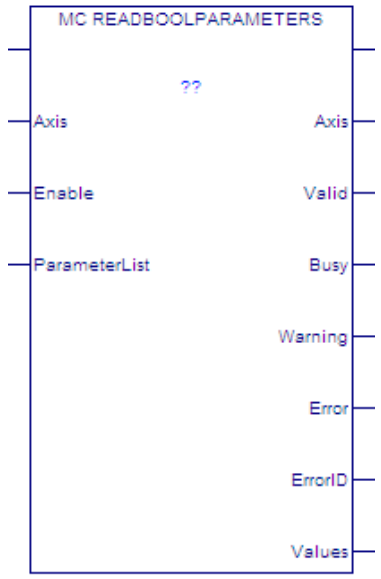
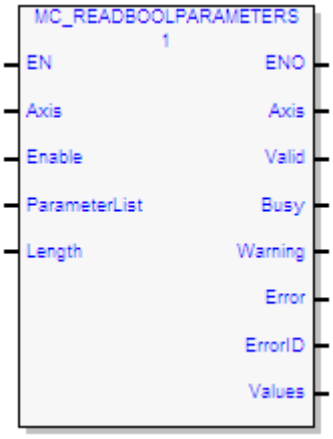
Outputs			
Valid	Indicates valid outputs are available	LD: flow Other languages: all except constants	0
Busy	Indicates the function block is enabled and has not completed its action.		1
Warning	Indicates that a warning has occurred within the function.		0
Error	Indicates that an error has occurred within 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.*



MC_ReadBoolParameters

LD	FBD	ST
		Formal convention: MC_READBOOLPARAMETERS(Axis := [input], Enable := [input], ParameterList := [input], Length := [input], Valid => [output], Busy => [output], Warning => [output], Error => [output], ErrorID => [output], Values => [output]);

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.

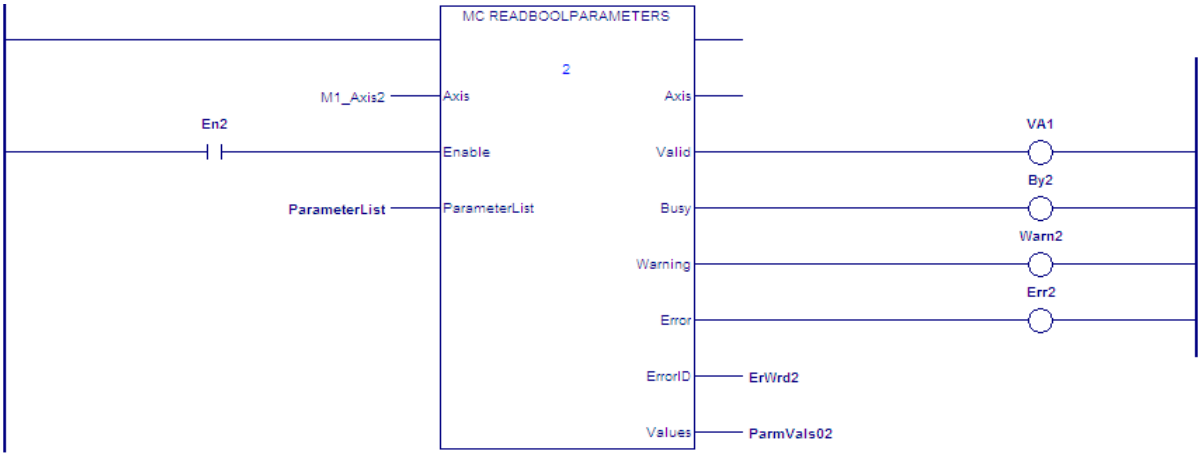
Operands

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
	Input_Output Parameters		
Axis	Axis with parameter to read.	AXIS_REF	N/A
	Inputs		
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

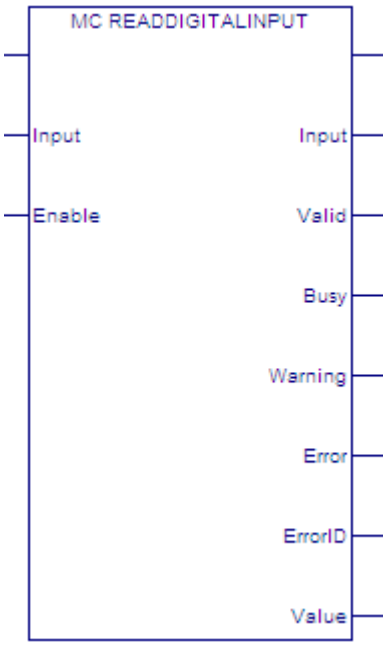
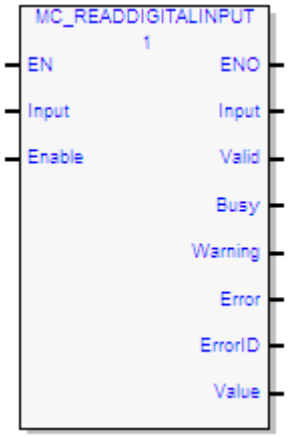
Outputs			
Valid	Indicates valid outputs are available	LD: flow Other languages: all except constants	0
Busy	Indicates the function block is enabled and has not completed its action.		1
Warning	Indicates that a warning has occurred within the function.		0
Error	Indicates that an error has occurred within 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.*



MC_ReadDigitalInput

LD	FBD	ST
		Formal convention: MC_READDIGITALINPUT(Input := [input], Enable := [input], Valid => [output], Busy => [output], Warning => [output], Error => [output], ErrorID => [output], Value => [output]);

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, “Wiring and Connections.”

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 Chapter 8.

Execution type: Immediate execution/immediate response.

Operands

Parameter	Description	Allowed Data Types	Initial Value
Input_Output Parameters			
Input	Reference to the input signal source.	INPUT_REF, defined in the module hardware configuration.	N/A
Inputs			
Enable	Read the digital input while enabled.	LD: flow Other languages: all except constants	N//A
Outputs			
Valid	Input signal value is valid.	LD: flow Other languages: all except	0
Busy	Indicates the function block has been executed and has not completed its action.		1

Parameter	Description	Allowed Data Types	Initial Value
Warning	Indicates that a warning has occurred within the function.	constants	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

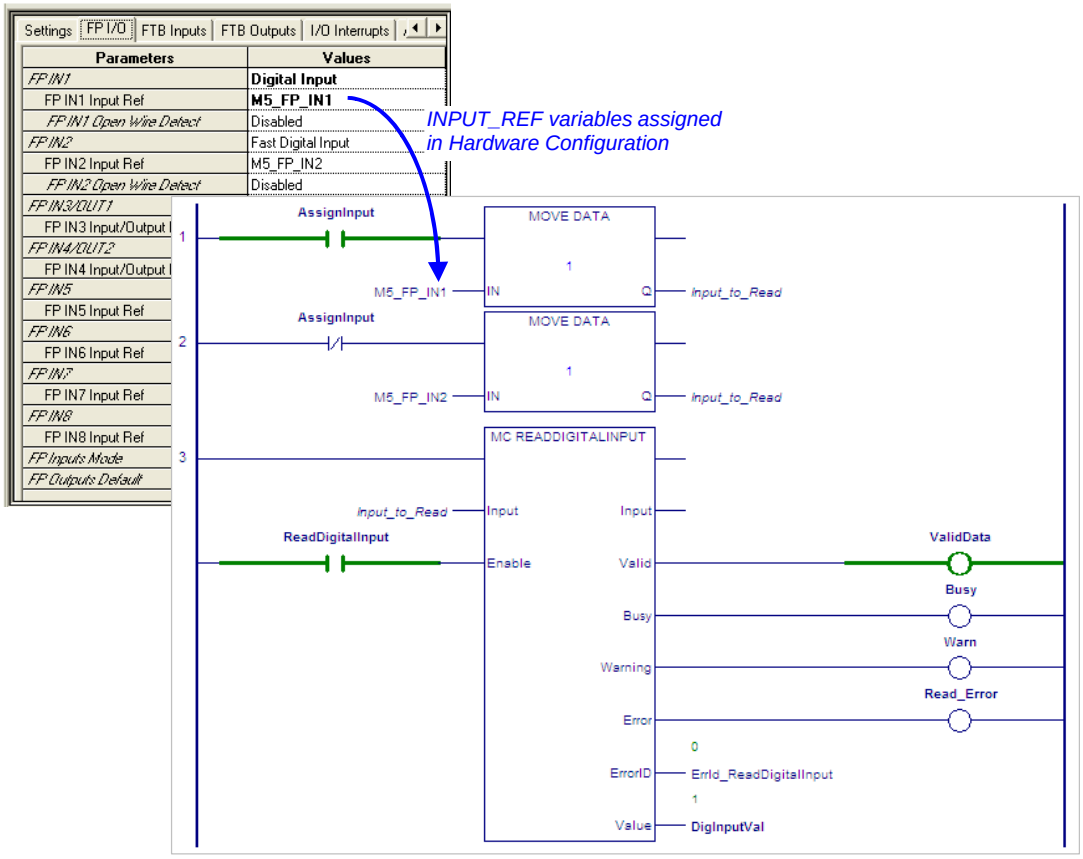
Example

In the following example, the INPUT_REF variables *M5_FP_IN1* and *M5_FP_IN1* are created in hardware configuration and assigned to the faceplate input points IN1 and IN2.

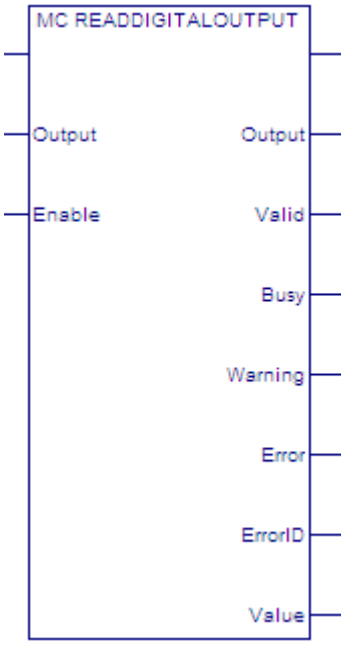
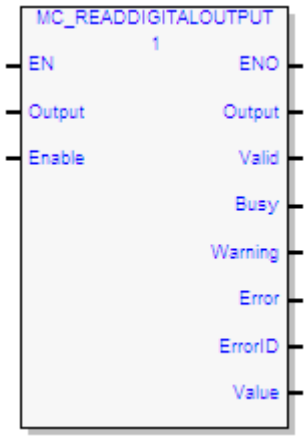
Note: To automatically create a variable with data type INPUT_REF, create it first as the Input parameter to the MC_ReadDigitalInput function. It can then be used as the IN parameter to the MoveData functions.

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.



MC_ReadDigitalOutput

LD	FBD	ST
		Formal convention: MC_READDIGITALOUTPUT(Output := [input], Enable := [input], Valid => [output], Busy => [output], Warning => [output], Error => [output], ErrorID => [output], Value => [output]);

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, see Chapter 3, “Wiring and Connections.”

The operation of the MC_ReadDigitialOutput function is similar to that of the MC_ReadDigitalInput function, for which an example is provided on page 6-119.

Note that the two 24V 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.

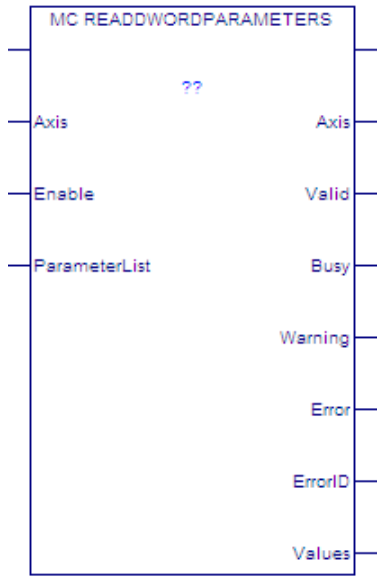
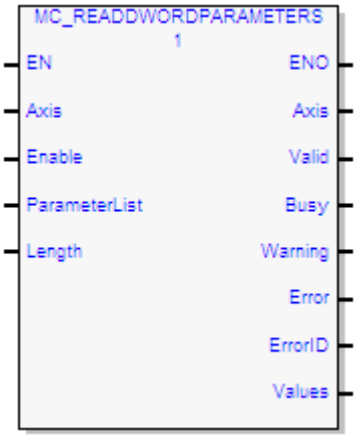
Execution type: Immediate execution/immediate response.

Operands

Parameter	Description	Allowed Data Types	Initial Value
Input_Output Parameters			
Output	Reference to the signal output.	OUTPUT_REF, defined in the module hardware configuration.	N/A
Inputs			
Enable	Read the digital output while enabled.	LD: flow Other languages: all except constants	N/A

Outputs			
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

MC_ReadDwordParameters

LD	FBD	ST
		Formal convention: MC_READDWORDPARAMETERS(Axis := [input], Enable := [input], ParameterList := [input], Length := [input], Valid => [output], Busy => [output], Warning => [output], Error => [output], ErrorID => [output], Values => [output]);

This function returns the values of up to 16 Dword axis parameters, which are specified b y 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.

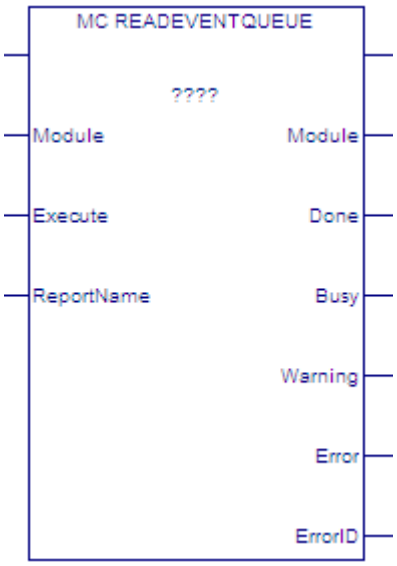
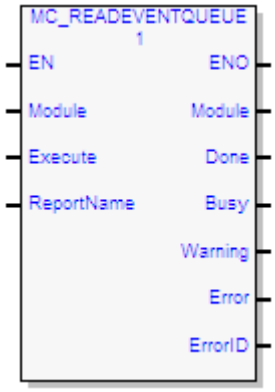
Execution type: Immediate execution/immediate response.

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
Input_Output Parameters			
Axis	Axis with parameters to read.	AXIS_REF	N/A

Inputs			
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
Outputs			
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 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

MC_ReadEventQueue

LD	FBD	ST
		Formal convention: [instance name](Module := [input], Execute := [input], ReportName := [input], Done => [output], Busy => [output], Warning => [output], Error => [output], ErrorID => [output]);

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.

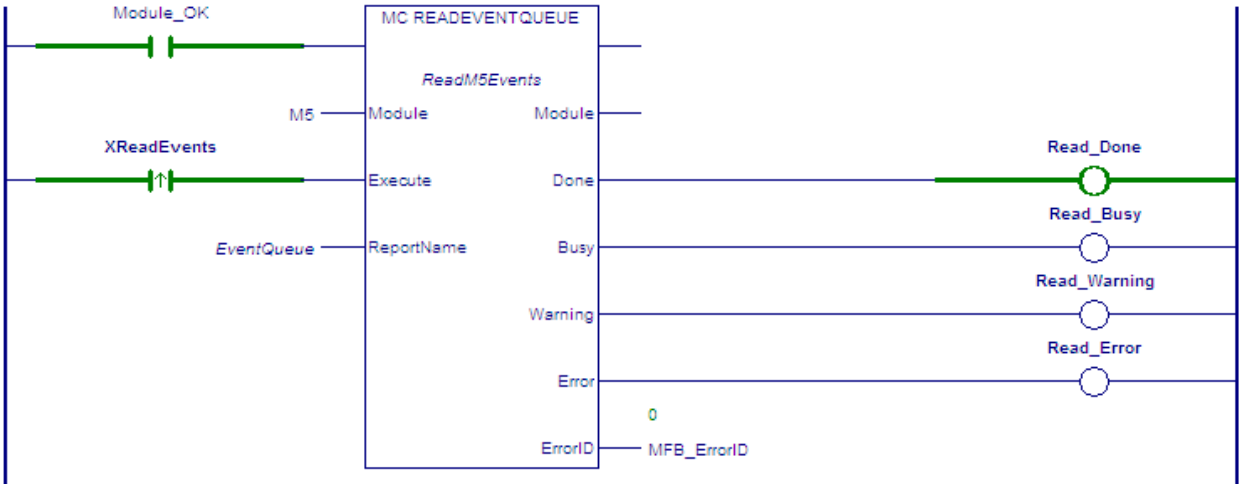
Operands

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

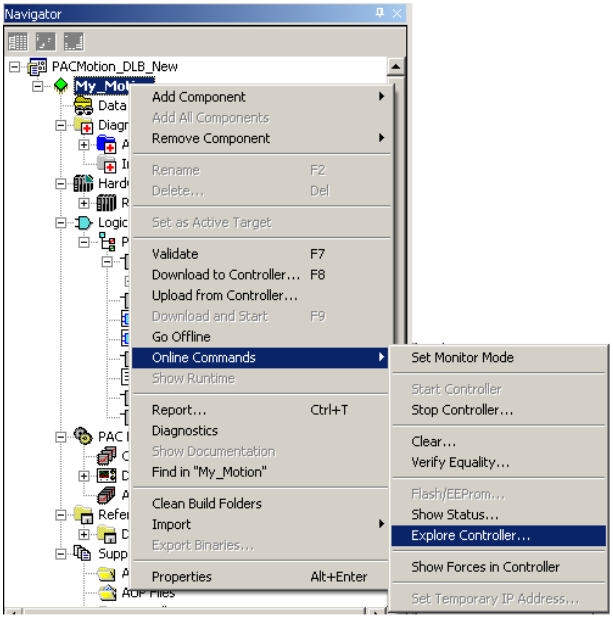
Parameter	Description	Allowed Data Types	Initial Value
	Input_Output Parameters		
Module	Module to execute function block	MODULE_REF	N/A
	Inputs		
Execute	The rising edge copies the event queue contents to the RX3i CPU.	LD: flow Other languages: all except constants	0
ReportName	Specifies the file reference 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

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.

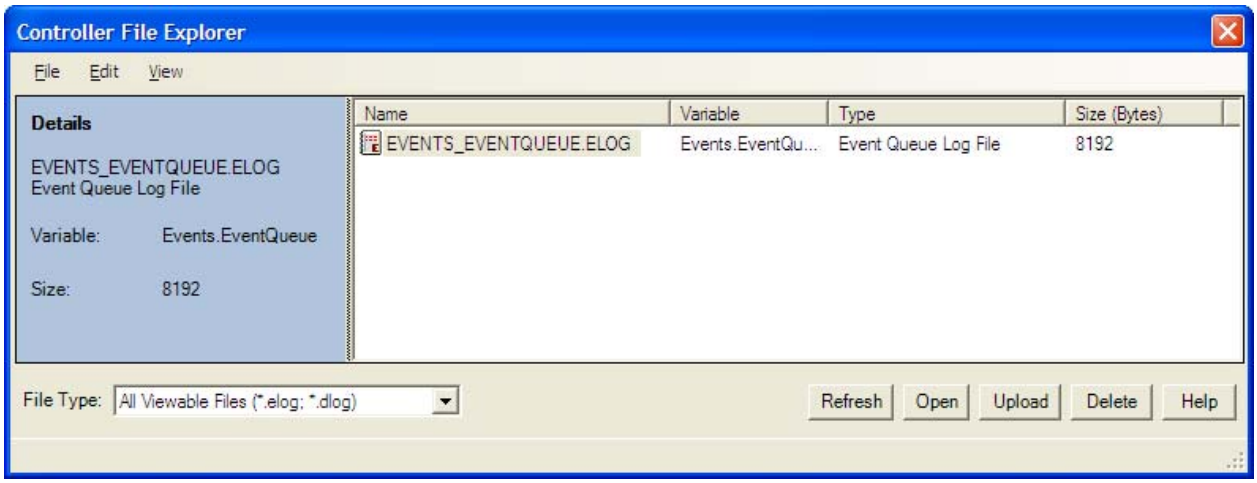


The ReportName input variable specifies the name of the file, EventQueue.ELOG, that is created in the RX3i controller. To access the report file, browse the files in the controller using the Controller File Explorer.



Controller File Explorer for ELOG File

To display the Event Queue Log in the InfoViewer, select the .ELOG file and click the Open button.



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 “PMM Event Queue” in Chapter 9, “Diagnostics.”

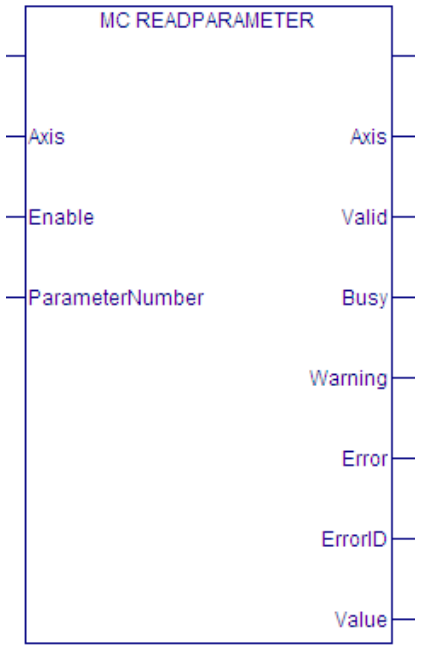
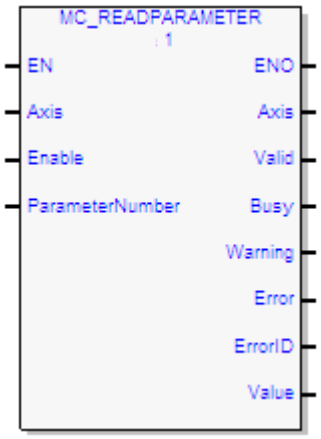
Sample Event Queue Log

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	0000046045	Module	Informational	0x0DC0	0x230F	0x0A4A0008	No Stop	Powerup Event
25	0000893246	Module	Informational	0x0F45	0x2D0F	0x001F0004	No Stop	Assume rack synch mastership
26	0001505132	Axis 1	Informational	0x0330	0x3300	0x0CE60000	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	0003099836	Module	Informational	0x0C15	0x210F	0x02B30016	No Stop	Datalog operation aborted due to PLC mode change
29	0003099836	Axis 1	Error	0x530F	0x3820	0x0A780000	Normal Stop	PLC mode change aborted function block
30	0003099836	Axis 2	Informational	0x030F	0x3801	0x0A710000	No Stop	PLC mode change aborted function block

MC_ReadParameter

LD	FBD	ST
		Formal convention: MC_READPARAMETER(Axis := [input], Enable := [input], ParameterNumber := [input], Valid => [output], Busy => [output], Warning => [output], Error => [output], ErrorID => [output], Value => [output]);

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, see “Parameter Number Index” in Chapter 8.

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

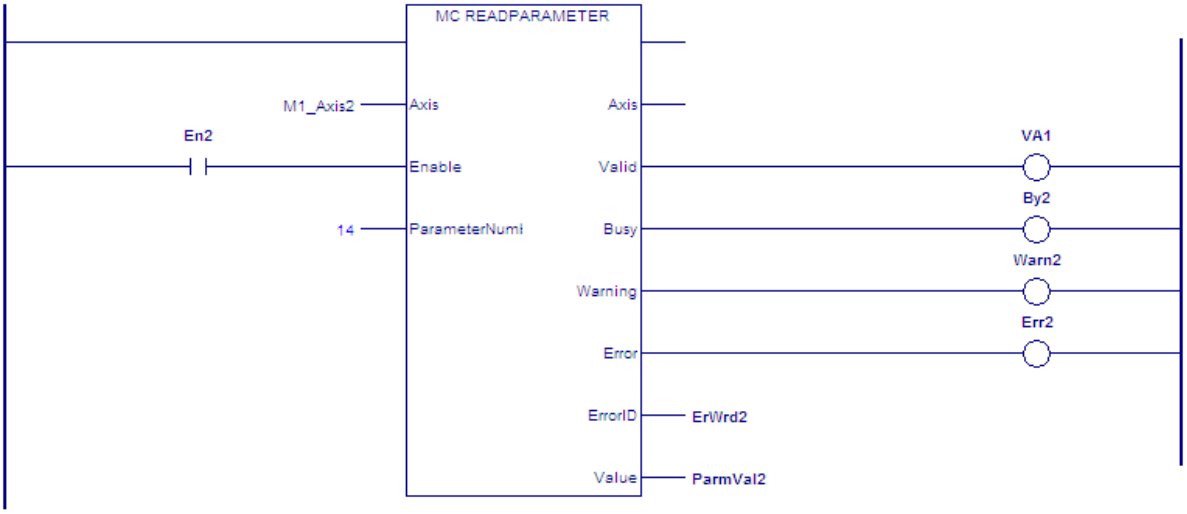
Parameter	Description	Allowed Data Types	Initial Value
Input_Output Parameters			
Axis	Axis with parameter to read.	AXIS_REF	N/A
Inputs			
Enable	Read the parameter while enabled.	LD: flow Other languages: all except constants	N/A
ParameterNumber	The parameter to read. This value can be a constant or a mapped variable.	Constant, INT	N/A

MC_ReadParameter

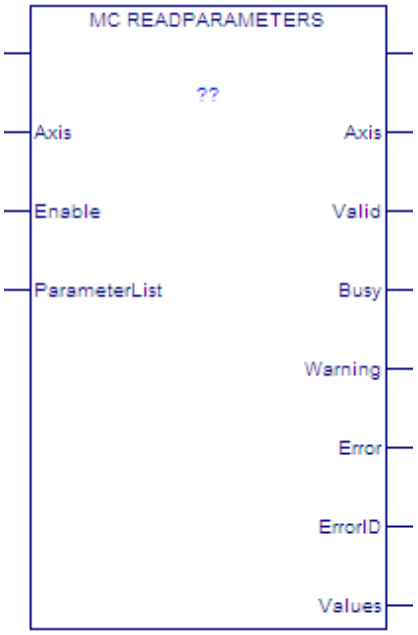
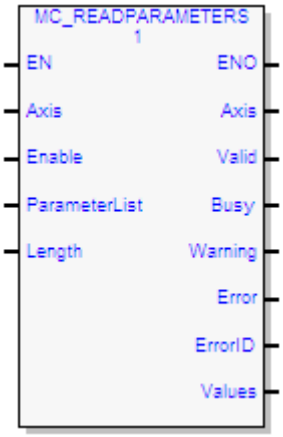
Outputs			
Valid	Set to 1 if valid outputs are available	LD: flow Other languages: all except constants	0
Busy	Indicates the function block is enabled and has not completed its action.		1
Warning	Indicates that a warning has occurred within the function.		0
Error	Indicates that an error has occurred within the function.		0
ErrorID	Error or warning identification.	WORD	0
Value	Value of the specified parameter.	LREAL	0

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*.



MC_ReadParameters

LD	FBD	ST
		Formal convention: MC_READPARAMETERS(Axis := [input], Enable := [input], ParameterList := [input], Length := [input], Valid => [output], Busy => [output], Warning => [output], Error => [output], ErrorID => [output], Values => [output]);

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, see “Parameter Number Index” in Chapter 8.

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

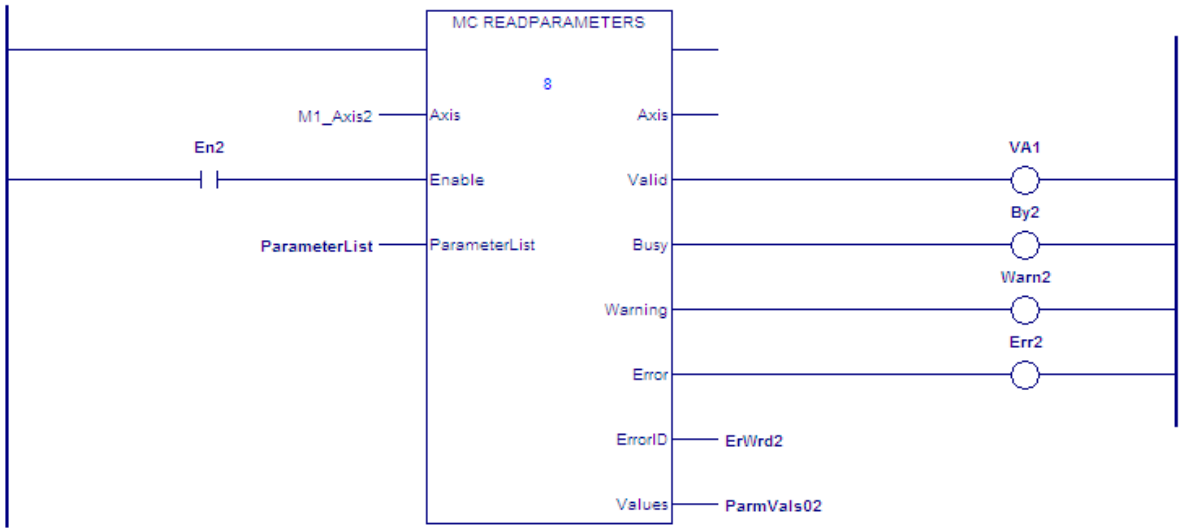
Parameter	Description	Allowed Data Types	Initial Value
??	Length. The number of parameter values to read, starting with the first parameter in the ParameterList array (maximum of 16).	Constant	N/A
	Input_Output Parameters		
Axis	Axis with parameters to read.	AXIS_REF	N/A
	Inputs		
Enable	Read the parameters 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

MC_ReadParameters

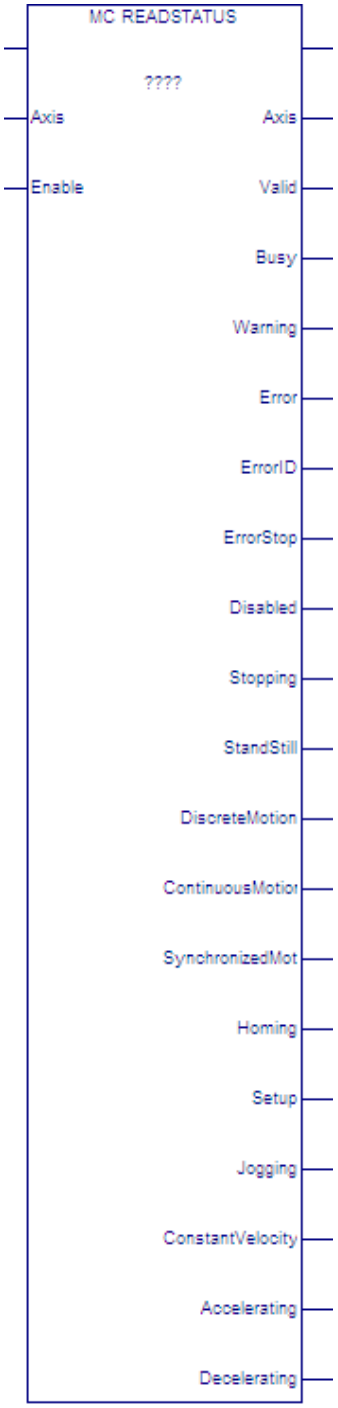
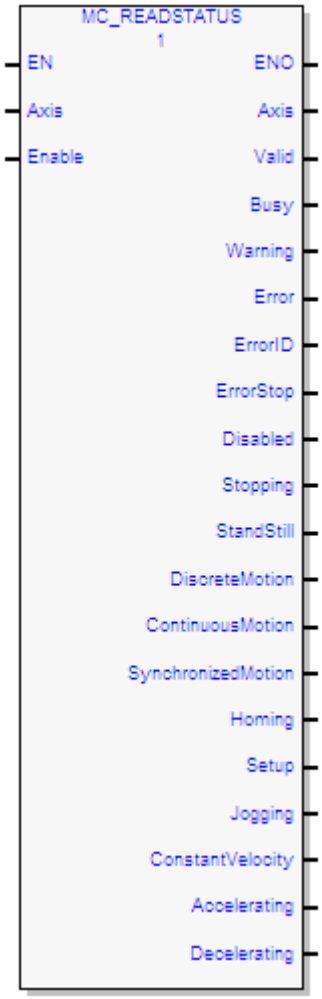
Outputs			
Valid	Set to 1 if valid outputs are available	LD: flow Other languages: all except constants	0
Busy	Indicates the function block is enabled and has not completed its action.		1
Warning	Indicates that a warning has occurred within the function.		0
Error	Indicates that an error has occurred 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

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.*



MC_ReadStatus

LD	FBD	ST
		Formal convention: [instance name](Axis := [input], Enable := [input], Valid => [output], Busy => [output], Warning => [output], Error => [output], ErrorID => [output], ErrorStop => [output], Disabled => [output], Stopping => [output], StandStill => [output], DiscreteMotion => [output], ContinuousMotion => [output], SynchronizedMotion => [output], Homing => [output], Setup => [output], Jogging => [output], ConstantVelocity => [output], Accelerating => [output], Decelerating => [output]);

This function block returns the current state of the selected axis. For definitions of axis states, refer to “Axis States” in Chapter 5.

Execution type: Immediate execution/immediate response.

MC_ReadStatus

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*
Input_Output Parameters			
Axis	The axis on which the function block is to be executed.	AXIS_REF	N/A
Inputs			
Enable	Reads the status of the axis while set to 1.	LD: flow Other languages: all except constants	0
Outputs			
Valid	Set to 1 if valid outputs are available.	LD: flow Other languages: all except constants	0
Busy	Indicates the function block is enabled but 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 block.		0
ErrorID	Error or warning identification.	WORD	0
ErrorStop	Set to 1 if the axis is in the ErrorStop state.	LD: flow Other languages: all except constants	0
Disabled	Set to 1 if the axis is in the Disabled state.		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

Axis Status Flags

The following axis status flags can read as 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 (βis 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 *MaxPositionLag* value and *EnablePosLagMonitoring* is true. When *MaxPositionLagActive* is set, *Commanded Velocity* and *Commanded Position* are frozen to allow the axis to ”catch up” to the *Commanded Position*. Supported axes: 1–4
TorqueLimitActive	1207	The commanded torque has exceeded the Torque Limit setting in HWC.
ServoReady	1208	The servo amplifier 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

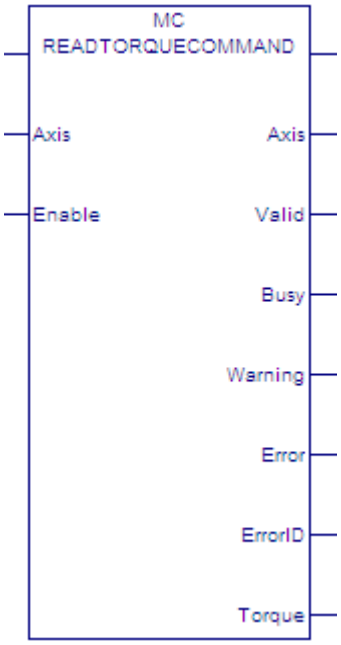
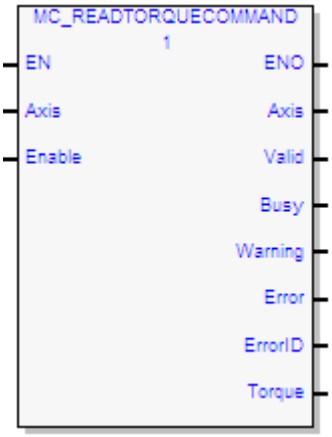
MC_ReadStatus

Status	*Parameter Number*	*Flag is set (ON) when:*
ErrorStop*	1210	An error requiring motion stop has occurred on the axis. Also see “Axis States” in Chapter 5. Supported axes: 1–4 5 Path Gen
Disabled*	1211	The amplifier does not have power. Supported axes: 1–4 5 Path Gen
Stopping*	1212	The axis is transitioning from DiscreteMotion, ContinuousMotion or SynchronizedMotion state to Standstill state. Supported axes: 1–4 5 Path Gen
Standstill*	1213	The axis is enabled, not moving, and ready to perform motion. Supported axes: 1–4 5 Path Gen
DiscreteMotion*	1214	The axis is performing discrete motion. Supported axes: 1–4 5 Path Gen
ContinuousMotion*	1215	The axis is performing continuous motion. Supported axes: 1–4 5 Path Gen
SynchronizedMotion*	1216	The axis is performing synchronized motion. Supported axes: 1–4 5 Path Gen
Homing*	1217	The axis is performing a home cycle. Supported axes: 1–4 5 Path Gen
ConstantVelocity*	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*	1219	The axis is accelerating (increasing energy of the motor). Supported axes: 1–4 5 Path Gen
Decelerating*	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

Status	Parameter Number	Flag is set (ON) when:
HomeSwitch	1223	Home switch is enabled. Supported axes: 1–4
FeedbackMoving	1224	The actual velocity is within the *Feedback Moving Deadband* range specified in HWC. Supported axes: 1–4 5 External Device
Reserved	1225	Reserved.
Setup*	1226	Axis is in a diagnostic or tuning condition. Supported axes: 1–4 5 Path Gen
Jogging*	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

*Reported by MC_ReadStatus function block.

MC_ReadTorqueCommand

LD	FBD	ST
		Formal convention: MC_READTORQUECOMMAND(Axis := [input], Enable := [input], Valid => [output], Busy => [output], Warning => [output], Error => [output], ErrorID => [output], Torque => [output]);

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.

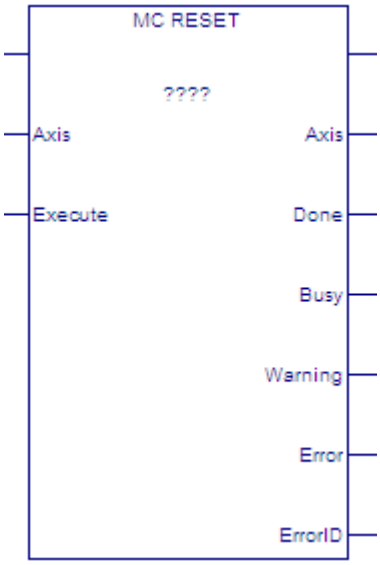
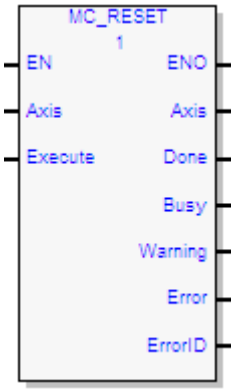
Note: The value returned by this function is commanded torque, not actual torque.

Execution type: Immediate execution/immediate response.

Operands

Parameter	Description	Allowed Data Types	Initial Value
	Input_Output Parameters		
Axis	The axis on which the function is to be executed.	AXIS_REF	N/A
	Inputs		
Enable	Read the torque command while enabled.	LD: flow Other languages: all except constants	N/A
	Outputs		
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

MC_Reset

LD	FBD	ST
		Formal convention: [instance name](Axis := [input], Execute := [input], Done => [output], Busy => [output], Warning => [output], Error => [output], ErrorID => [output]);

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

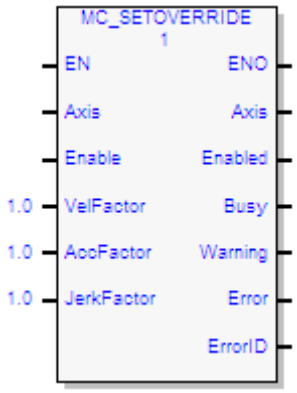
Instance Variable	Description	Allowed Data Types	Initial Value
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as *????* in LD.)	MC_RESET	NA

Parameter	Description	Allowed Data Types	Initial Value
Input_Output Parameters			
Axis	The axis on which the function block is to be executed.	AXIS_REF	N/A

MC_Reset

Parameter	*Description*	*Allowed Data Types*	*Initial Value*
	Inputs		
Execute	The rising edge resets the axis.	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 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

MC_SetOverride

LD	FBD	ST
		Formal convention: [instance name](Axis := [input], Enable := [input], VelFactor := 1.0, AccFactor := 1.0, JerkFactor := 1.0, Enabled => [output], Busy => [output], Warning => [output], Error => [output], ErrorID => [output]);

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 slave axes (axes in the Synchronized Motion state).

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:

VELmin = 1/10 rpm = 1/600 rev/sec

ACCmin = 1/6000 rev/sec^2

JERKmin = 1/60000 rev/sec^3

Note: The JerkFactor override value is only used for jerk values that specify jerk in uu/sec^3. 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.

Execution type: Immediate execution/deferred response.

Caution

If an axis is in Discrete Motion state, reducing the AccFactor and/or JerkFactor can lead to potential position overshoot.

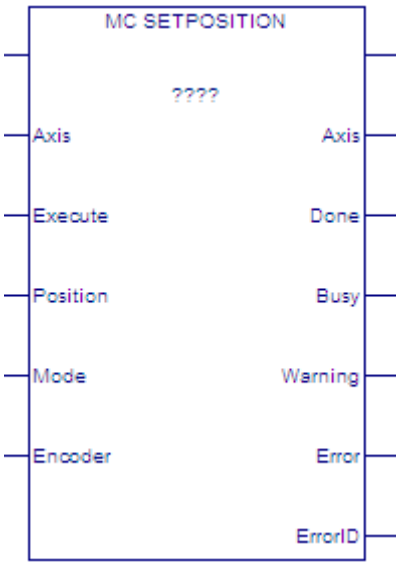
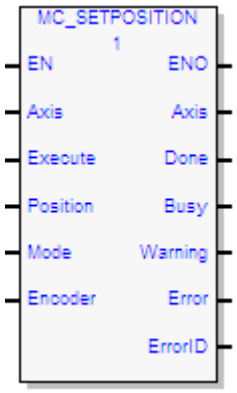
MC_SetOverride Operands

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*
	Input_Output Parameters		
Axis	Axis that receives function block command.	AXIS_REF	N/A
	Inputs		
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 hol*d.) 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 axis' actual velocity to exceed the Max Velocity System limit, an error will be generated.	LREAL	1

Parameter	Description	Allowed Data Types	Initial Value
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^3. Does not apply when jerk is specified in percent.	LREAL	1
Outputs			
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

MC_SetPosition

LD	FBD	ST
		Formal convention: [instance name](Axis := [input], Execute := [input], Position := [input], Mode := [input], Encoder := [input], Done => [output], Busy => [output], Warning => [output], Error => [output], ErrorID => [output]);

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.

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*	(UU) In absolute mode, specifies the axis' new actual position. In relative mode, specifies the distance the axis' actual position/commanded position 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

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.

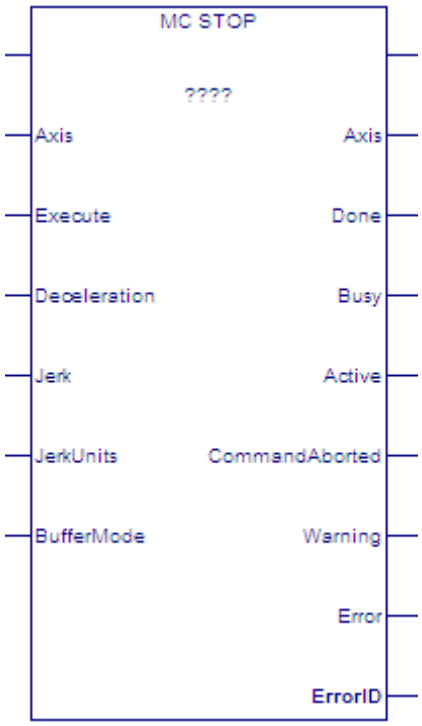
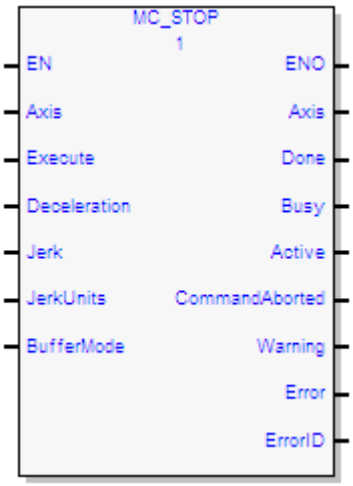
**** Valid Values for Position Input Parameter***

External Device Low Position Limit ≤ *Position* <
(External Device Low Position Limit + External Device Position Range)

or

Motor Encoder Low Position Limit ≤ *Position* <
(Motor Encoder Low Position Limit + Motor Encoder Position Range)

MC_Stop

LD	FBD	ST
		Formal convention: [instance name](Axis := [input], Execute := [input], Deceleration := [input], Jerk := [input], JerkUnits := [input], BufferMode := [input], Done => [output], Busy => [output], Active => [output], CommandAborted => [output], Warning => [output], Error => [output], ErrorID => [output]);

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.

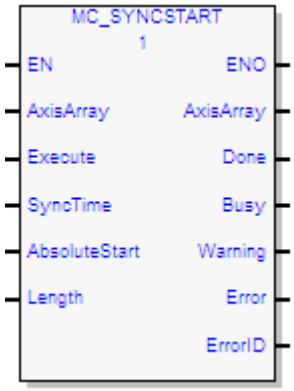
Execution type: Immediate execution/deferred response.

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*
	Input_Output Parameters		
Axis	Axis that receives function block command.	AXIS_REF	N/A
	Inputs		
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^2]	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^3 or %)]	LREAL For details, see "Jerk and JerkUnits" in Chapter 5.	0
JerkUnits	Selects units for Jerk input: (UU/sec^3 or %)	MC_JERKUNITS	0
BufferMode	Defines the axis buffering behavior. Valid modes are Aborting, Buffered, Blending.	MC_BufferMode	0
	Outputs		
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

MC_SyncStart

LD	FBD	ST
		Formal convention: [instance name](AxisArray := [input], Execute := [input], SyncTime := [input], AbsoluteStart := [input], Length := [input], Done => [output], Busy => [output], Warning => [output], Error => [output], ErrorID => [output]);

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.

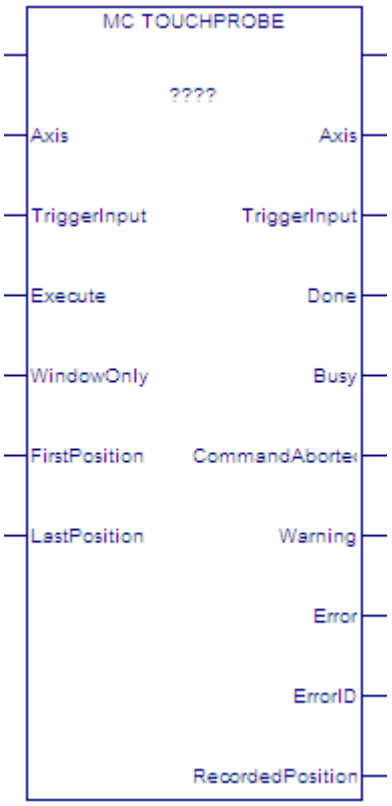
Execution type: Immediate execution/deferred response.

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 milliseconds 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 Other languages: all except constants	0
Busy	Indicates the function block has been executed and has not yet completed its action.		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

MC_TouchProbe

LD	FBD	ST
		Formal convention: [instance name](Axis := [input], TriggerInput := [input], Execute := [input], WindowOnly := [input], FirstPosition := [input], LastPosition := [input], Done => [output], Busy => [output], CommandAborted => [output], Warning => [output], Error => [output], ErrorID => [output], RecordedPosition => [output]);

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 and 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. 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 B, “Touch Probe Accuracy Calculations.” 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

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.	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
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

Examples

Touch Probe Operation with Rotary Axis

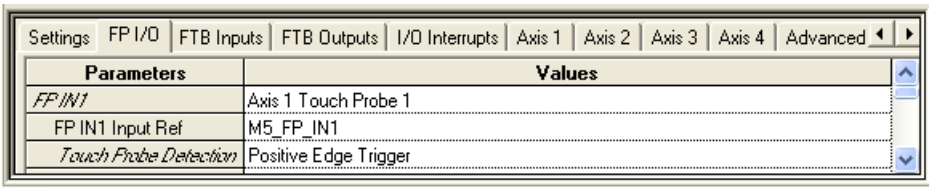
The following example illustrates the use of the touch probe function to perform a windowing strobe function with a rotary axis.

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

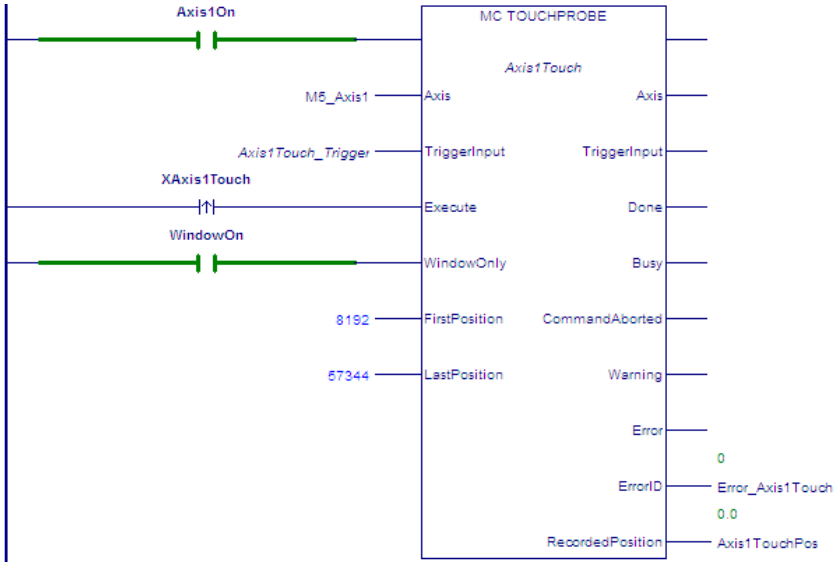
For this example, the Axis 1 TouchProbe 1 function is assigned to the PMM faceplate input FPIN1 on the FP I/O tab and is configured to trigger on the Positive Edge of an incoming pulse.



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.

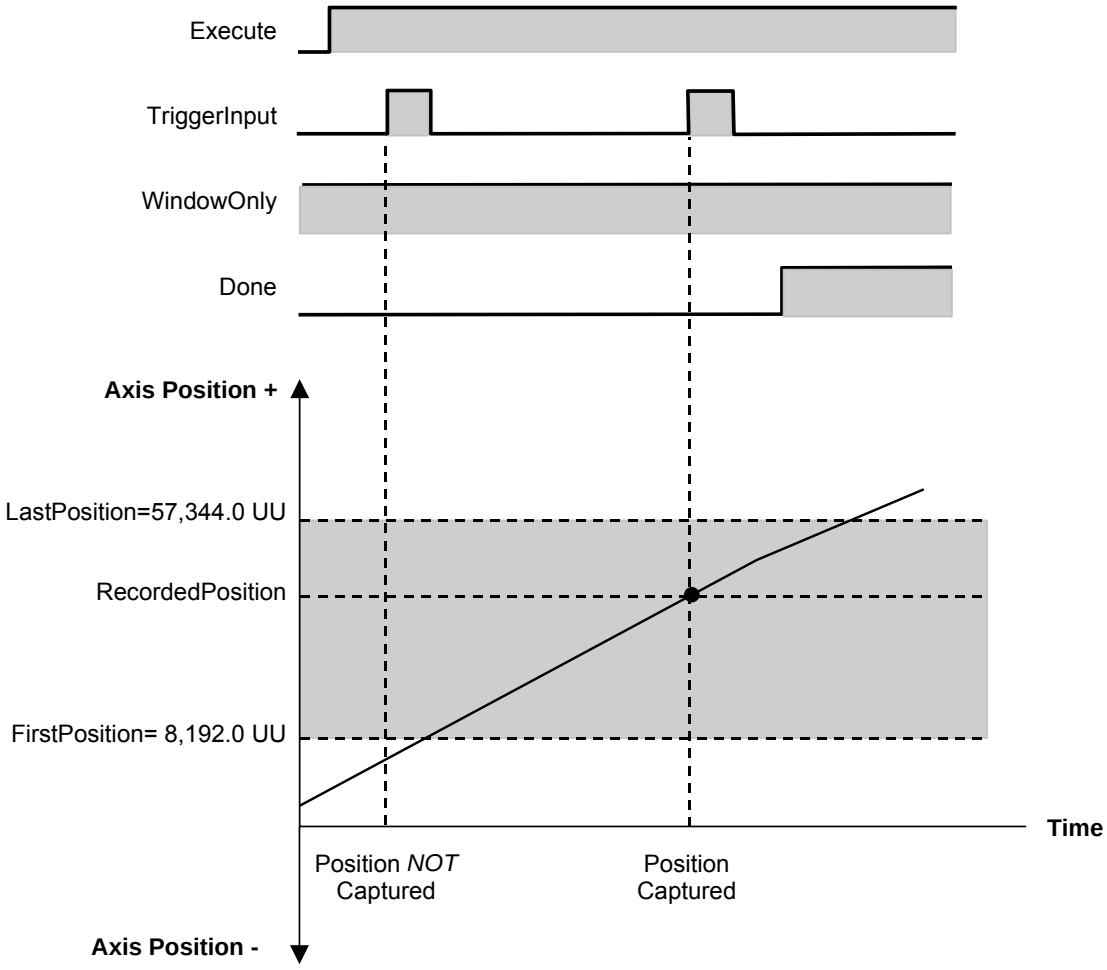


Timing Diagram for TouchProbe Example with WindowOnly Input On

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.

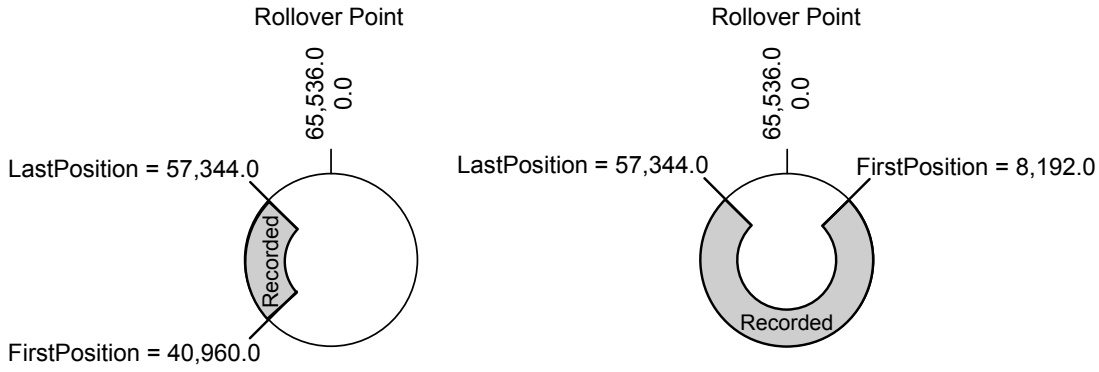
In the following example, the FirstPosition is less than the LastPosition and the axis is moving in the positive direction. The axis could be rotary or linear. For examples of window operation with other axis modes, see page 6-154.



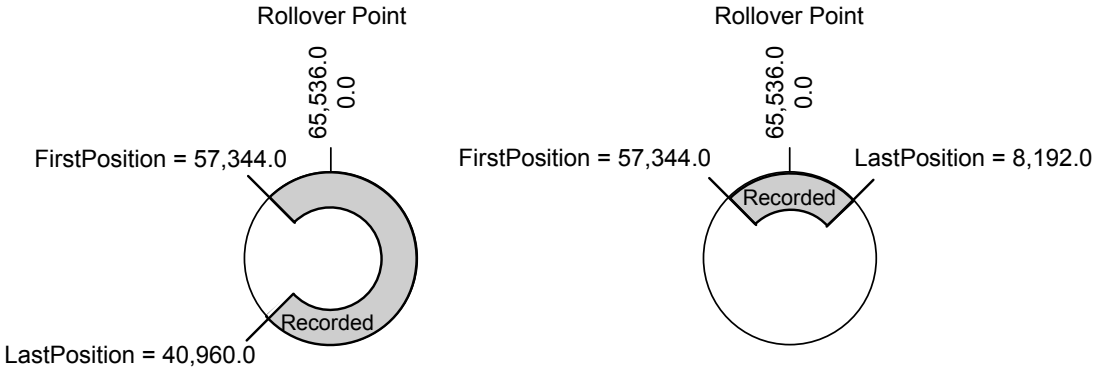
Window Operation with Rotary Axes

For these examples, the High Position Limit is 65,536.0. When moving in either direction, the axis' actual position rolls over at this point.

FirstPosition < LastPosition



FirstPosition > LastPosition

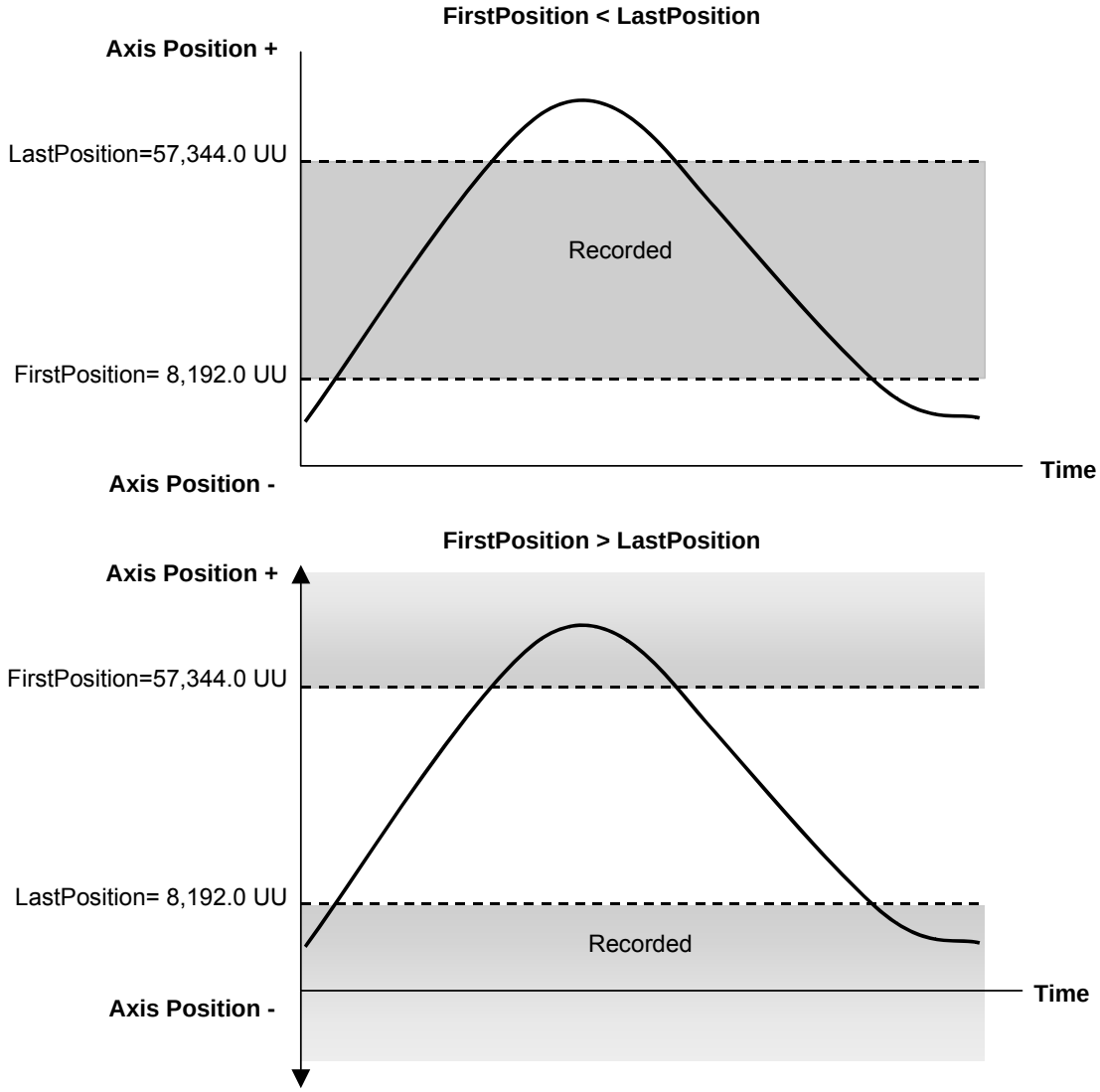


Adapted from the PLCOpen TC2 Task Force Motion Control Part 2 — Extensions — V.1.0 PLCOpen — 2002, 2005
www.plcopen.org

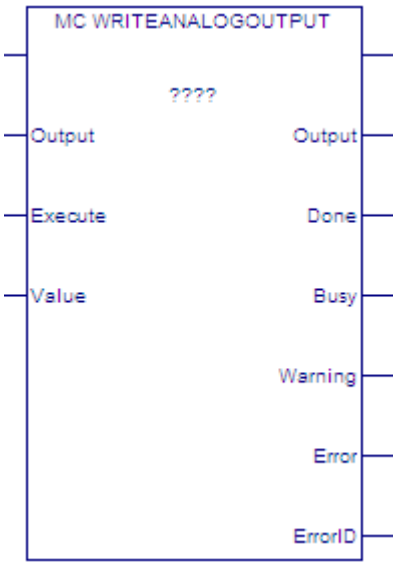
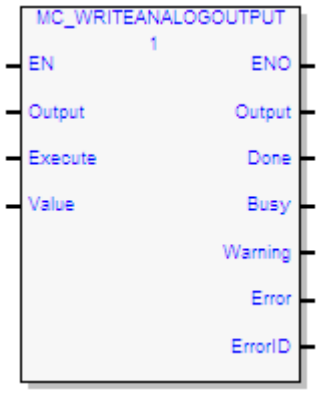
0

Window Operation with Linear Axes

Since a linear axis does not roll over, it enters the “Recorded” window more than once is to change direction, as shown in the following example, where the FirstPosition is less than the LastPosition.



MC_WriteAnalogOutput

LD	FBD	ST
		Formal convention: [Instance name](Output := [input], Execute := [input], Value := [input], Done => [output], Busy => [output], Warning => [output], Error => [output], ErrorID => [output]);

This function block provides the ability to control the two ±10V 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 Chapter 8.

For specifications and connection details for the analog outputs, see Chapter 3, “Wiring and Connections.”

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

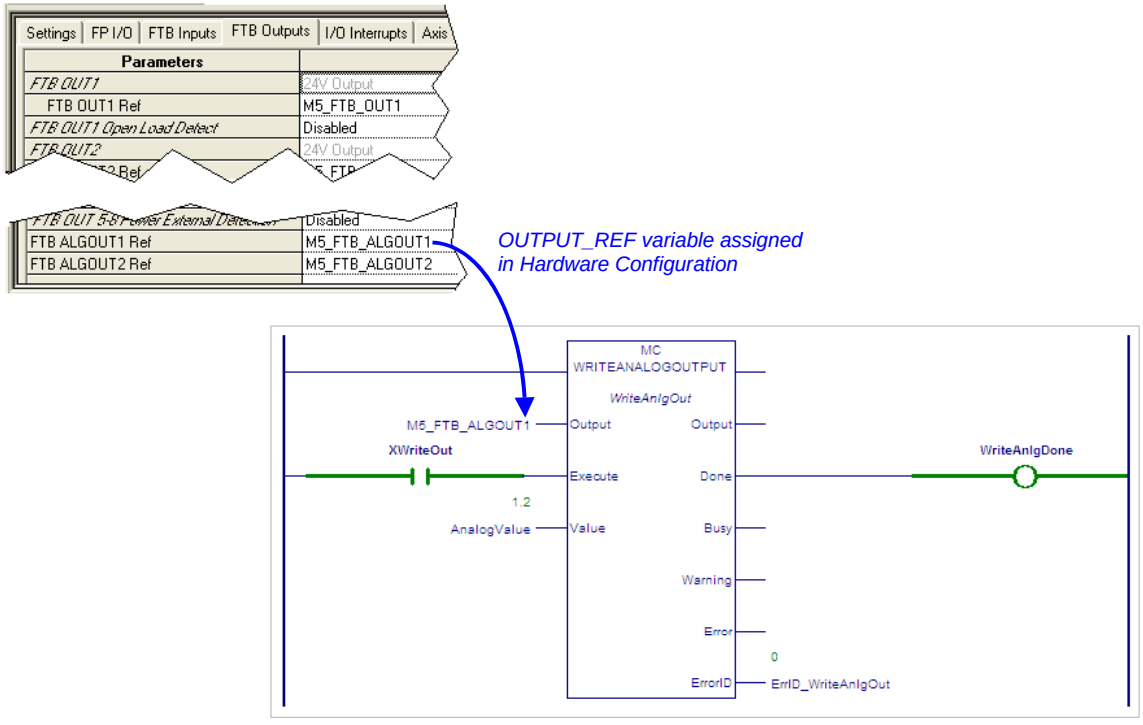
Inputs			
Execute	The rising edge Writes the value of the selected output.	LD: flow Other languages: all except constants	0
Value	The value to be written to the selected analog output. Units = volts. Range = ±10V.	LREAL	0
Outputs			
Done	Writing of the output signal value is done	LD: flow Other languages: 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

Example

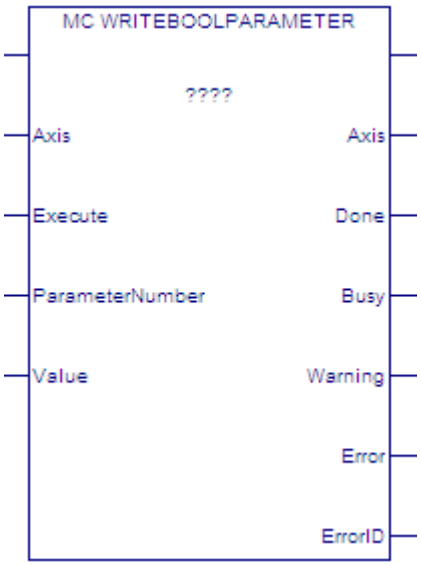
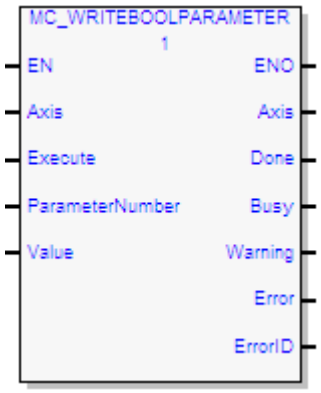
In the following example, the OUTPUT_REF variable *M5_FTB_ALGOUT1* has been created on the FTB Outputs tab in the hardware configuration and assigned to the FTB analog output ALGOUT1.

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.



MC_WriteBoolParameter

LD	FBD	ST
		Formal convention: [instance name](Axis := [input], Execute := [input], ParameterNumber := [input], Value := [input], Done => [output], Busy => [output], Warning => [output], Error => [output], ErrorID => [output]);

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, see “Parameter Number Index” in Chapter 8.

Execution type: Immediate execution/deferred response.

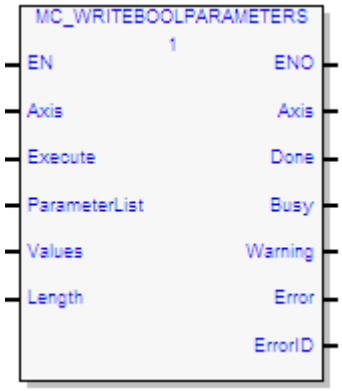
Operands

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
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
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

Parameter	*Description*	*Allowed Data Types*	*Initial Value*
	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 in the MFB.		0
Error	Indicates that an error has occurred in the MFB.		0
ErrorID	Error or warning identification.	WORD	0

MC_WriteBoolParameters

LD	FBD	ST
		Formal convention: [instance name](Axis := [input], Execute := [input], ParameterList := [input], Values := [input], Length := [input], Done => [output], Busy => [output], Warning => [output], Error => [output], ErrorID => [output]);

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, see “Parameter Number Index” in Chapter 8.

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.

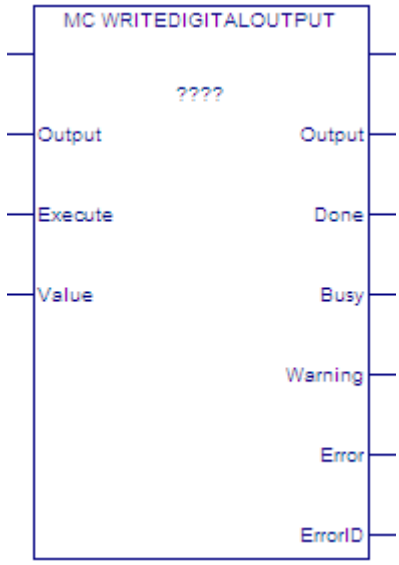
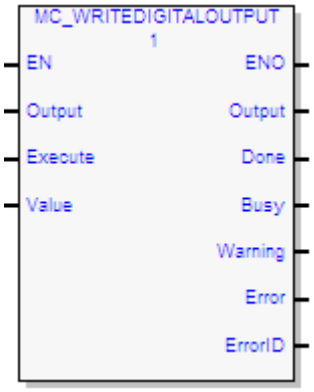
Operands

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

MC_WriteDigitalOutput

LD	FBD	ST
		Formal convention: [instance name](Output := [input], Execute := [input], Value := [input], Done => [output], Busy => [output], Warning => [output], Error => [output], ErrorID => [output]);

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 Chapter 8.

For specifications and connection details for the discrete I/O points, see Chapter 3, “Wiring and Connections.”

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
Input_Output Parameters			
Output	Reference to the signal outputs.	OUTPUT_REF, defined in the module hardware configuration.	N/A

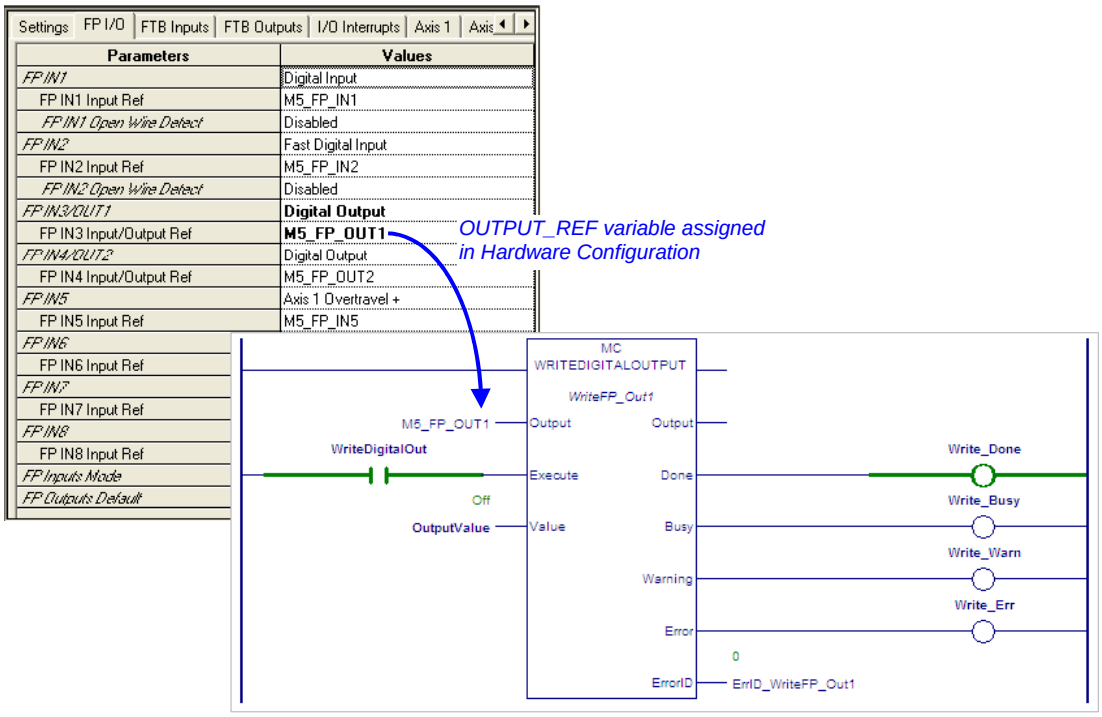
Parameter	*Description*	*Allowed Data Types*	*Initial Value*
	Inputs		
Execute	The rising edge writes the value of the selected output.	LD: flow Other languages: all except constants	0
Value	The value to be written to the selected output.	BOOL	0
	Outputs		
Done	Writing of the output signal value is done.	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

Example

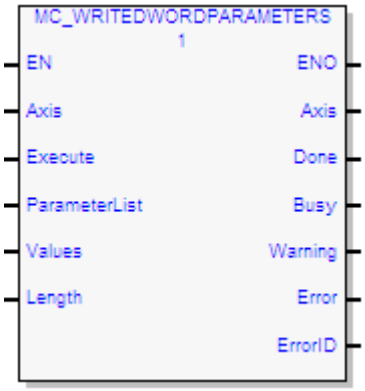
In the following example, the value in *OutPutValue* is written to the faceplate output OUT1.

Because each faceplate 24V 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.



MC_WriteDwordParameters

LD	FBD	ST
		Formal convention: [instance name](Axis := [input], Execute := [input], ParameterList := [input], Values := [input], Length := [input], Done => [output], Busy => [output], Warning => [output], Error => [output], ErrorID => [output]);

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

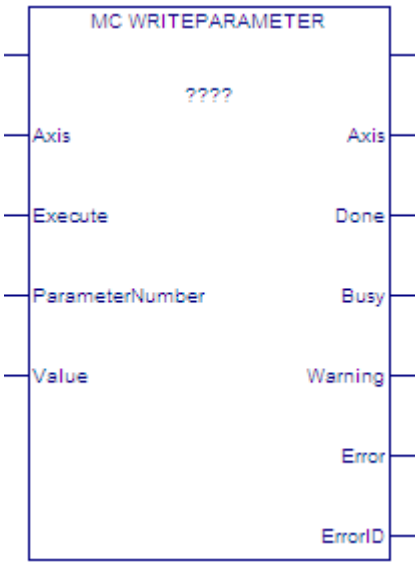
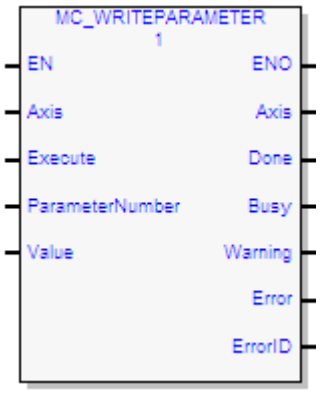
Instance Variable	Description	Allowed Data Types	Initial Value
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as *????* in LD.)	MC_WRITEDWORDPARAMETERS	N/A

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

MC_WriteDwordParameters

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

MC_WriteParameter

LD	FBD	ST
		Formal convention: [instance name](Axis := [input], Execute := [input], ParameterNumber := [input], Value := [input], Done => [output], Busy => [output], Warning => [output], Error => [output], ErrorID => [output]);

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 (MaxAcclerationAppl -PN13) and Deceleration Limit (MaxDecelerationAppl - 15).

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, see “Parameter Number Index” in Chapter 8.

Execution type: Immediate execution/deferred response.

Operands

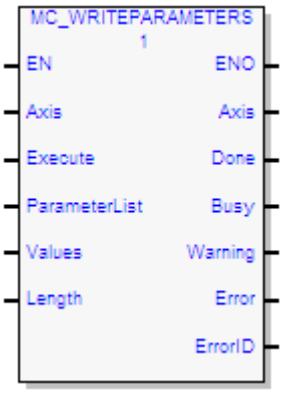
Instance Variable	Description	Allowed Data Types	Initial Value
[Instance Variable Name]	Structure variable containing the internal data for the function block instance. (Initially displayed as *????* in LD.)	MC_WRITEPARAMETER	

Parameter	Description	Allowed Data Types	Initial Value
Input_Output Parameters			
Axis	Axis with parameter to write.	AXIS_REF	N/A

MC_WriteParameter

Parameter	*Description*	*Allowed Data Types*	*Initial Value*
	Inputs		
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
	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

MC_WriteParameters

LD	FBD	ST
		Formal convention: [instance name](Axis := [input], Execute := [input], ParameterList := [input], Values := [input], Length := [input], Done => [output], Busy => [output], Warning => [output], Error => [output], ErrorID => [output]);

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	

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

Chapter 7

Electronic Cam Programming

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 Chapter 6.

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.

  Non-Quintic Points + 2 * Quintic Points ≤ 4096

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 “Cam Operation Restrictions by Type and Mode” on page 7-6.

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. See “Start Mode Mask” in Chapter 6.	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 ***Linear*** axis always stays within bounds of position determined by the upper and lower position limit settings. A ***Rotary*** 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.

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.

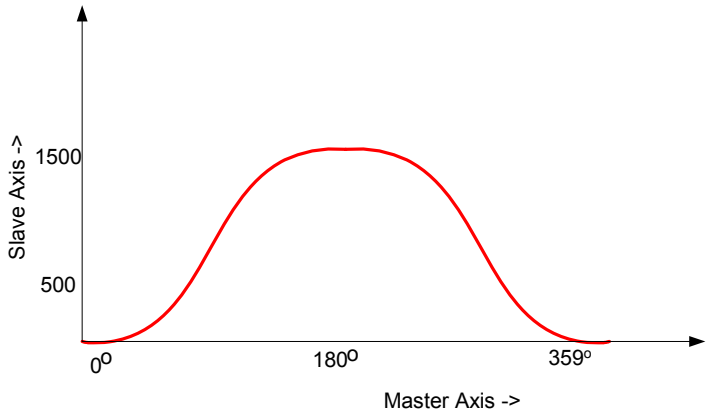
Constraint: The first and last slave point must be the same for a Linear Cyclic Cam. The Cam Editor will not display the option for “Linear Cyclic” in the “Cam Type” field unless this constraint is satisfied by the data in the cam table.

Notes:

1. 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 – 1) in user units. This is because a cyclic profile's first and last point are the same on the physical device.

2. 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



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 > First Point

Last Slave Point First Slave Point Last Point < First Point

Cam Operation Restrictions by Type and Mode

The following table indicates the valid selections and combinations of cam profile type and operational modes.

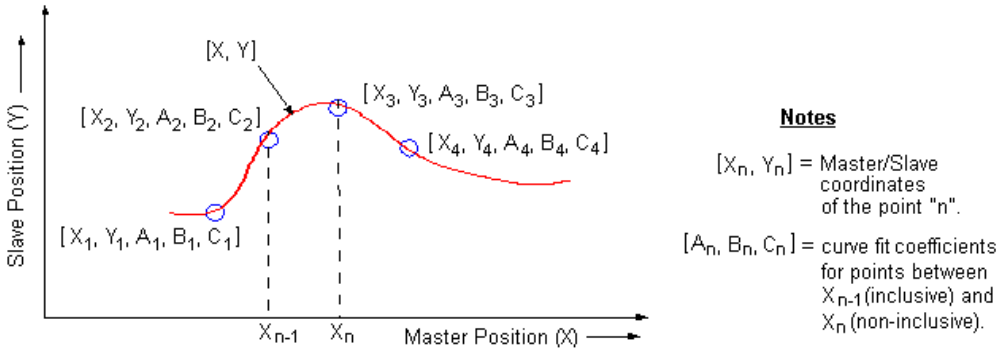
Axis Mode	Axis Position Mode	Axis	Execution Mode	Cam Profile Type		
				non-cyclic	linear-cyclic	circular-cyclic
Absolute	Linear	Master	periodic	Not Allowed.*	Not Allowed.*	Not Allowed.*
			non-periodic	Master Profile end positions must be within or equal to Axis End positions	Master Profile end positions 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.*
			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
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

* 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.

Smoothing and Curve Fitting

The cam editor employs spline polynomial smoothing 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).



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.

Notes:

- 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.

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 2nd-degree sector follows a 3rd-degree sector and a complete sector follows the 2nd-degree sector, the ending first derivative of the 2nd-degree sector is set equal to the starting first derivative of the complete sector.

If a 2nd-degree sector follows a 3rd-degree sectors and is either the last sector is followed by a 3rd-degree sector, the starting first derivative of the 2nd-degree sector is set by fitting a quadratic using three points, with the 2nd-degree sector's first point in the middle.

3rd-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 of Second Derivative. If there is only one sector, the Starting and Ending Boundaries will be set.

For internal sectors, if a 3rd-degree sector follows any non-3rd-degree sector, the starting first derivative is set equal to the ending first derivative of the preceding sector. If a 3rd-degree sector follows a 3rd-degree sector, the sectors are built as if they were one sector. If a 3rd-degree sector is followed by any non-3rd-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 the profile is cubic and end condition is required.

For quadratic sectors, the start boundary condition can be:

- The numerical value of the profile's 1st derivative (Slope).
- The numerical value of the profile's 2nd derivative.
- 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 (2nd degree sector) or four points (3rd 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 (3rd degree sector) and calculating the slope of the temporary polynomial at the end point.

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 In Gear.

For details on the operation of these function blocks, refer to Chapter 6.

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

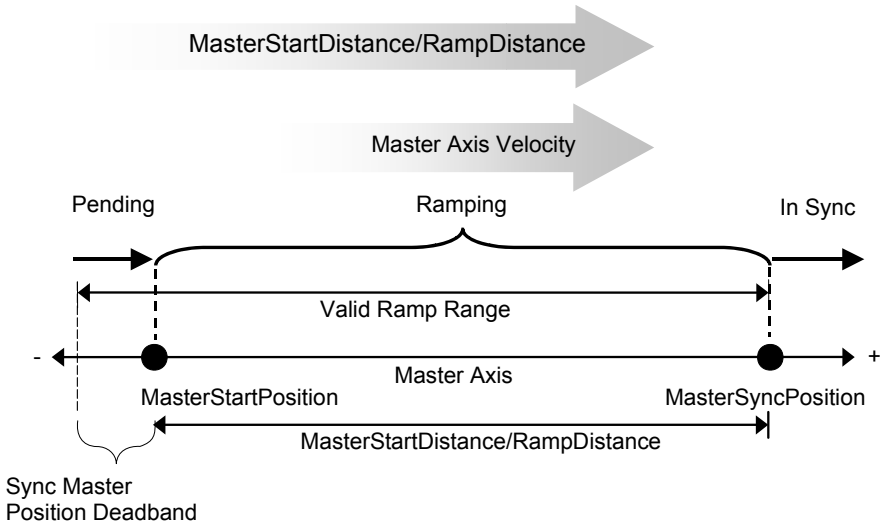
B) MC_CamIn or MC_GearInPos is aborted by another MFB, or

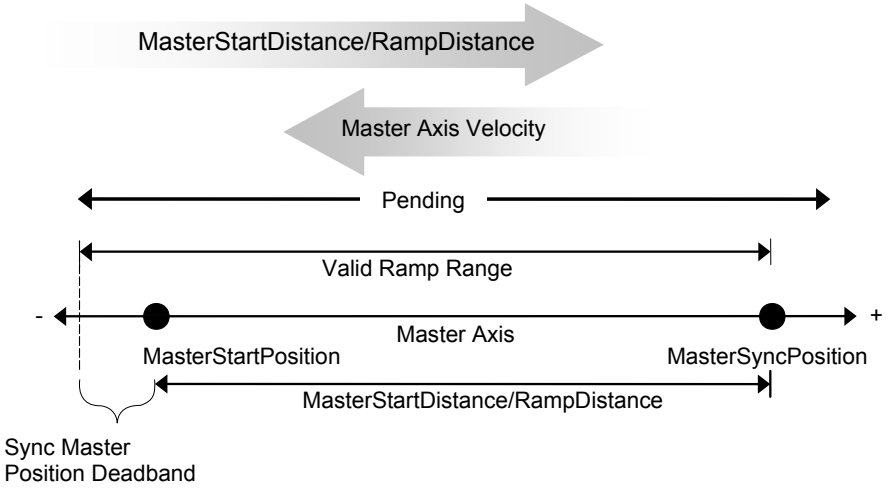
C) when the RX3i CPU transitions from Run to Stop mode.

The master axis motion triggers the slave axis' transition from Pending to Ramping under the following conditions:

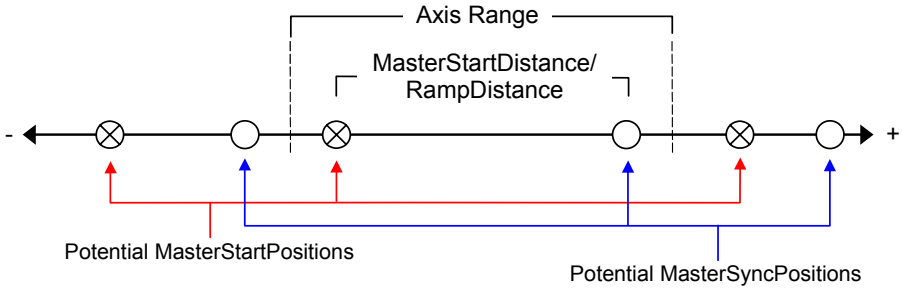
- For MC_GearInPos, the transition is triggered by position and direction. Ramping begins when the master axis crosses the MasterStartPosition in the same direction as the sign of the MasterStartDistance. (The master axis velocity has the same sign as MasterStartDistance.)
- For MC_CamIn, the transition 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 current master axis' position becomes the MasterStartPosition.

Master Axis Direction Effect on Pending

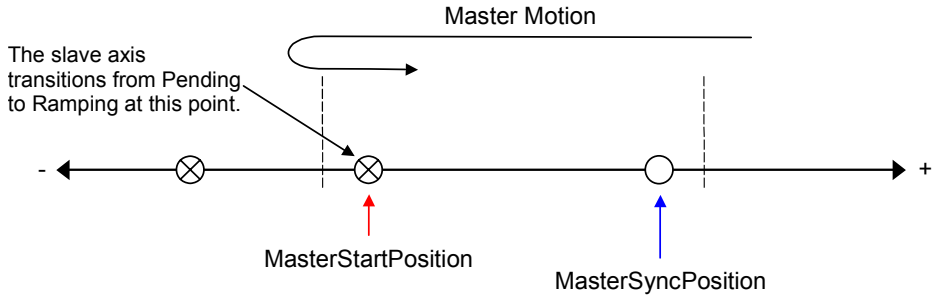




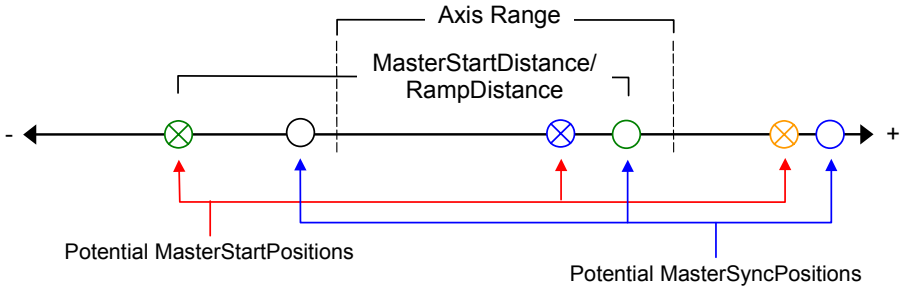
When the master's Axis Positioning Mode is Rotary, there are additional considerations. When the master velocity's direction is opposite that of the RampDistance (MC_CamIn) or has not yet crossed the MasterStartPosition (MC_GearInPos), the MasterSyncPosition and MasterStartPosition are "potential."



Once the trigger condition is met, the positions are no longer potential, but become “fixed.”



If the MasterStartDistance is larger than the master’s Axis Range, the master will pass potential MasterSyncPositions but will ignore them.



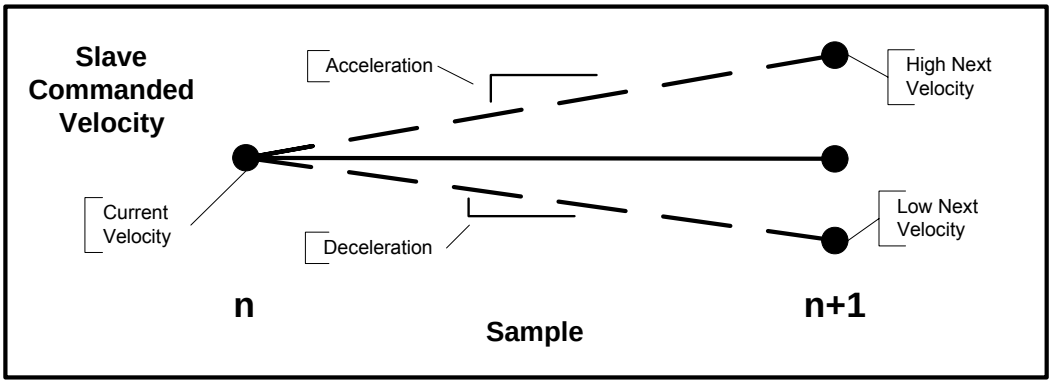
If, on a rotary axis, the MasterStartDistance is an integer multiple of the master’s Axis Range, the rotary MasterStartPosition is effectively the same as the MasterSyncPosition.

Ramping

Synchronized MFBs: MC_GearIn, MC_GearInPos and MC_CamIn

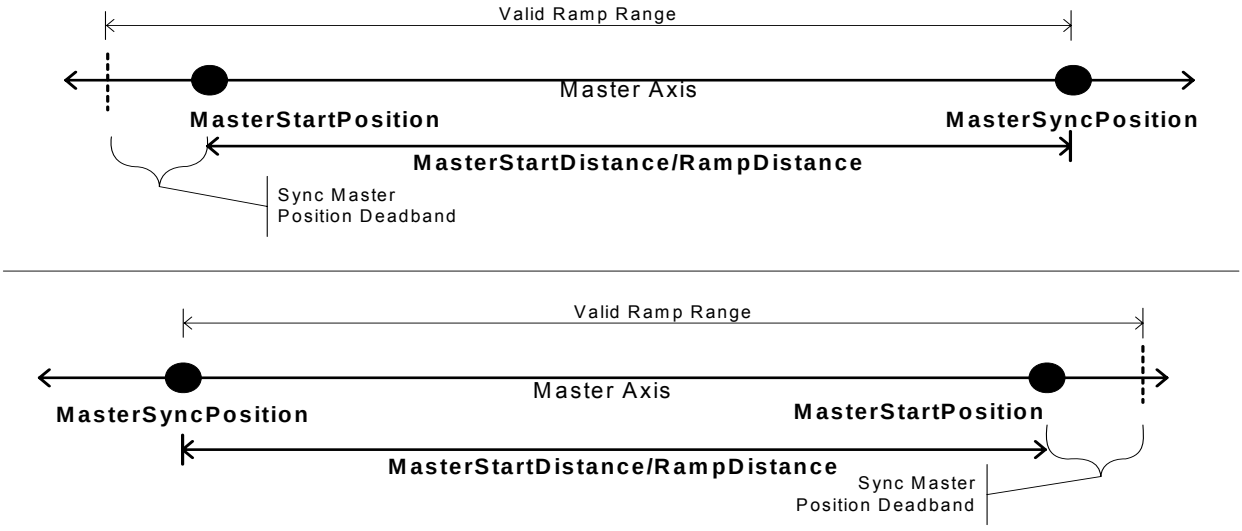
While ramping, the function block’s Active and Busy outputs are on, but InSync (or InGear) is not. The slave axis is attempting to synchronize its motion with the master axis, but has not yet achieved synchronization.

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 master's velocity times the gear ratio) is between the High Next Velocity and the Low Next Velocity as in the figure below.



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 slave's commanded velocity and accelerations are limited by the slave axis' application limits while ramping.



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.

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.

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

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 1st Derivative (Type 5)	Slave 2nd Derivative (Type 5)

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.

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

Structure	Field	Contents
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)
		Curve-fit type (linear, quadratic, cubic, w/ 1st der., w/ 2nd der.) (BYTE)
		Online-Correction (BYTE)
	1st position pair in sector	1st Master position (LREAL)
		1st Slave position (LREAL)
		(IF Curve fit 4 or 5) 1st Slave relative velocity (LREAL)
		(IF Curve fit 5) 1st 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)	

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 DerivativeType: 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

Chapter 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.

Parameter Number Index 8-2

I/O Data Reference Numbers 8-21

Parameter Number Index

Most parameters are associated with an individual axis, however some are associated with the module. Many of the parameters have default values set by the hardware configuration (HWC). Parameters can be accessed using the MC_ReadParameter, MC_ReadBoolParameter, MC_WriteParameter and MC_WriteBoolParameter functions.

Axis parameter numbers are listed below. Module parameter numbers can be found on page 8-16.

Axis Parameter Numbers

Some parameter numbers correspond to hardware configuration parameters. For additional information on these parameters, refer to Chapter 4, “Configuration.” These parameters are initialized to the corresponding values in the Hardware Configuration stored to the module. Some Hardware Configuration parameters 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.

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, see “Axis Configuration Data” in Chapter 4.

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[1]	UU
3	SWLimitNeg	Software end of travel – negative direction Supported axes: 1–4 5 Path Gen	LREAL	Read/Write[1]	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[1]	NA

Parm.	Parameter Name	Description	Data Type	Read/Write	Units
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[1]	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
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 * cts/rev * 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 * Command Counts Per Motor Revolution * Command Position Resolution * 1/60 Initial value: Max Velocity System in UU/sec Supported axes: 1–4 5 Path Gen	LREAL	Read/Write[1]	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^2. Supported axes: 1–4 5 Path Gen	LREAL	Read	UU/sec^2
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^2) Initial value: MaxAccelerationSystem (in uu/sec^2) *Axis 5 Limits:* Low limit: 0.01 * Command Counts Per Motor Revolution * Command Position Resolution * 1/60 Initial value: MaxAccelerationSystem (in uu/sec^2) Supported axes: 1–4 5 Path Gen	LREAL	Read/Write[1]	UU/sec^2
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^2. Supported axes: 1–4 5 Path Gen	LREAL	Read	UU/sec^2

Parm.	Parameter Name	Description	Data Type	Read/Write	Units
15	MaxDecelerationAppl	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. *Axis 1–4 Limits:* Low limit = 0.01 * cts/rev * cts/uu * 1/60 High limit = Max Decel System (in uu/sec^2), Initial value is MaxDecelerationSystem (in uu/sec^2) *Axis 5 Limits:* Low limit: 0.01 * Command Counts Per Motor Revolution * Command Position Resolution * 1/60 Initial value is MaxDecelerationSystem (in uu/sec^2) Supported axes: 1–4 5 Path Gen	LREAL	Read/Write[1]	UU/sec^2
16	MaxJerk	Maximum allowed jerk of the axis. Supported axes: 1–4 5 Path Gen	LREAL	Read/Write[1]	UU/sec^3
1000	MotorEncoderUserUnits (Axes 1–4) Also see “Considerations When Changing Scaling” on page 8-15.	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. Supported axes: 1–4 5 Path Gen	LREAL	Read/ Write[1, 2]	UU
1000	CommandPositionResolution (Axis 5) Also see “Considerations When Changing Scaling” on page 8-15.	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[1, 2]	UU
1001	MotorEncoderCounts Also see “Considerations When Changing Scaling” on page 8-15.	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[1, 2]	NA

Parm.	Parameter Name	Description	Data Type	Read/Write	Units
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[1, 2]	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[1, 2]	UU
1004	ExternalDeviceUserUnits Also see “Considerations When Changing Scaling” on page 8-15.	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[1, 2] (Valid only if an External Device has been configured.)	NA
1005	ExternalDeviceCounts Also see “Considerations When Changing Scaling” on page 8-15.	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[1, 2] (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[1, 2]	UU
1007	ExternalDeviceLow PositionLimit	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[1, 2]	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

Parm.	Parameter Name	Description	Data Type	Read/Write	Units
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	%
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^2
1014	ErrorStopJerk	Maximum jerk allowed during an Error Stop. Supported axes: 1–4 5 Path Gen	LREAL	Read/Write	UU/sec^3
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[1]	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[1]	UU
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

Parm.	Parameter Name	Description	Data Type	Read/Write	Units
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 priority Response Type axis Error ID. See Chapter 9 for details on Error IDs. Supported axes: 1–4 5 Path Gen	DWORD	Read	NA
1101	AxisStatus	Array of parameter numbers 1200 to 1229.	DWORD	Read	NA
1200	AxisOK	Axis status. See “Axis Status Flags” in Chapter 6.	BOOL	Read	NA
1201	PositionValid	Axis status. See “Axis Status Flags” in Chapter 6.	BOOL	Read	NA
1202	DriveEnabled	Axis status. See “Axis Status Flags” in Chapter 6.	BOOL	Read	NA
1203	CommandActive	Axis status. See “Axis Status Flags” in Chapter 6.	BOOL	Read	NA
1204	CommandMoving	Axis status. See “Axis Status Flags” in Chapter 6.	BOOL	Read	NA
1205	InZone	Axis status. See “Axis Status Flags” in Chapter 6.	BOOL	Read	NA
1206	MaxPositionLagActive	Axis status. See “Axis Status Flags” in Chapter 6.	BOOL	Read	NA
1207	TorqueLimitActive	Axis status. See “Axis Status Flags” in Chapter 6.	BOOL	Read	NA
1208	ServoReady	Axis status. See “Axis Status Flags” in Chapter 6.	BOOL	Read	NA
1209	VelocityLimit	Axis status. See “Axis Status Flags” in Chapter 6.	BOOL	Read	NA
1210	ErrorStop	Axis State. See “Axis Status Flags” in Chapter 6.	BOOL	Read	NA
1211	Disabled	Axis State. See “Axis Status Flags” in Chapter 6.	BOOL	Read	NA
1212	Stopping	Axis State. See “Axis Status Flags” in Chapter 6.	BOOL	Read	NA
1213	Standstill	Axis State. See “Axis Status Flags” in Chapter 6.	BOOL	Read	NA
1214	DiscreteMotion	Axis State. See “Axis Status Flags” in Chapter 6.	BOOL	Read	NA
1215	ContinuousMotion	Axis State. See “Axis Status Flags” in Chapter 6.	BOOL	Read	NA
1216	SynchronizedMotion	Axis State. See “Axis Status Flags” in Chapter 6.	BOOL	Read	NA

Parm.	Parameter Name	Description	Data Type	Read/Write	Units
1217	Homing	Axis State. See “Axis Status Flags” in Chapter 6.	BOOL	Read	NA
1218	ConstantVelocity	Axis status. See “Axis Status Flags” in Chapter 6.	BOOL	Read	NA
1219	Accelerating	Axis status. See “Axis Status Flags” in Chapter 6.	BOOL	Read	NA
1220	Decelerating	Axis status. See “Axis Status Flags” in Chapter 6.	BOOL	Read	NA
1221	OTPos	Over Travel, positive Axis Status. See “Axis Status Flags” in Chapter 6.	BOOL	Read	NA
1222	OTNeg	Over travel negative Axis Status. See “Axis Status Flags” in Chapter 6.	BOOL	Read	NA
1223	HomeSwitch	Axis status. See “Axis Status Flags” in Chapter 6.	BOOL	Read	NA
1224	FeedbackMoving	Axis status. See “Axis Status Flags” in Chapter 6.	BOOL	Read	NA
1225	AxisPositioningMode	0 = Linear, 1 = Rotary Set in HWC.	BOOL	Read	NA
1226	Setup	Axis status. Returned in MC_ReadStatus. See “Axis Status Flags” in Chapter 6.	BOOL	Read	NA
1227	Jogging	Axis status. Returned in MC_ReadStatus. See “Axis Status Flags” in Chapter 6.	BOOL	Read	NA
1228	AuxPositionValid	Axis status. See “Axis Status Flags” in Chapter 6.	BOOL	Read	NA
1229	ServoVelocityLimit	Axis status. See “Axis Status Flags” in Chapter 6.	BOOL	Read	NA
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^2
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	%

Parm.	Parameter Name	Description	Data Type	Read/Write	Units
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
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

Parm.	Parameter Name	Description	Data Type	Read/Write	Units
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
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[1, 2]	Master axis uu/sec
1320	ForceServoVelocityTimeout	Valid range is 0ms to10000ms. 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 and 32. 0 = *Disabled.* Supported axes: 1–4	DWORD	Read/Write[1]	ms
1330	LostPositionAccumulation	Lost position accumulated due to exceeding velocity limit. Supported axes: 1–4	LREAL	Read	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 65535). Supported axes: 1–4	BOOL	Read/Write	NA

Parm.	Parameter Name	Description	Data Type	Read/Write	Units
10006	VelocityLoopGain1, Integral (PK1V)	DSCM Parameter. 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, see "Advanced Servo Tuning" in Appendix C. Ignored by axes using Synthetic Motors (Type 65535). Supported axes: 1–4	DINT	Read/Write	NA
10007	VelocityLoopGain2, Proportional (PK2V)	DSCM Parameter. See comment for parameter 10006. Ignored by axes using Synthetic Motors (Type 65535). Supported axes: 1–4	DINT	Read/Write	NA
10008	VelocityLoopGain3, Integral Decay (PK3V)	DSCM Parameter. See comment for parameter 10006. Ignored by axes using Synthetic Motors (Type 65535). Supported axes: 1–4	DINT	Read/Write	NA
10019	Current Loop Compensation (EMFCMP)	This parameter should only be changed if an overcurrent alarm (BE error) occurs with Beta 0.5/6000*is* or Beta 1/6000*is* motors. The alarm may occur when torque limited acceleration is commanded to negative velocities faster than -5000 rpm. To prevent the error from occurring, reduce the acceleration rate so the motor does not operate in torque limit. If the acceleration cannot be reduced, enter Hardware Configuration Advanced Tab parameter 10019 (EMFCMP) with data value = 0. **Do not use the EMFCMP parameter for motors other than Beta 0.5/6000*is* or Beta 1/6000*is*. The only EMFCMP data value that should be used with these motors is zero.** For additional information, see "Advanced Servo Tuning" in Appendix C. Supported axes: 1–4	DINT	Read/Write	NA
10031	TorqueCommandFilter (TCMD)	DSCM Parameter. See comment for parameter 10006. Ignored by axes using Synthetic Motors (Type 65535). Supported axes: 1–4	DINT	Read/Write	NA

Parm.	Parameter Name	Description	Data Type	Read/Write	Units
10032	LoadInertiaRatio	DSCM Parameter. See comment for parameter 10006. Ignored by axes using Synthetic Motors (Type 65535). Supported axes: 1–4	DINT	Read/Write	NA
10086	DecelerationTorqueLimit (DETQLM)	DSCM Parameter. See comment for parameter 10006. Ignored by axes using Synthetic Motors (Type 65535). Supported axes: 1–4	DINT	Read/Write	NA
10256	DisturbanceInputFunction (BITPA20)	DSCM Parameter. See comment for parameter 10006. Ignored by axes using Synthetic Motors (Type 65535).	DINT	Read/Write	NA
10296	DisturbanceInputGain (DI_GAIN)	DSCM Parameter. See comment for parameter 10006. Ignored by axes using Synthetic Motors (Type 65535). Supported axes: 1–4	DINT	Read/Write	NA
10297	DisturbanceInputStartFrequency (S_FREQ)	DSCM Parameter. See comment for parameter1 0006. Ignored by axes using Synthetic Motors (Type 65535). Supported axes: 1–4	DINT	Read/Write	NA
10298	DisturbanceInputEndFrequency (E_FREQ)	DSCM Parameter. See comment for parameter 10006. Ignored by axes using Synthetic Motors (Type 65535). Supported axes: 1–4	DINT	Read/Write	NA
10299	NumberDisturbanceInputPoints (MPOINTS)	DSCM Parameter. See comment for parameter 10006. Ignored by axes using Synthetic Motors (Type 65535). Supported axes: 1–4	DINT	Read/Write	NA
10330	ResonanceFilter2Center Frequency (REF2C)	DSCM Parameter. See comment for parameter 10006. Ignored by axes using Synthetic Motors (Type 65535). Supported axes: 1–4	DINT	Read/Write	NA
10331	ResonanceFilter2BandWidth (REF2W)	DSCM Parameter. See comment for parameter 10006. Ignored by axes using Synthetic Motors (Type 65535). Supported axes: 1–4	DINT	Read/Write	NA

Parm.	Parameter Name	Description	Data Type	Read/Write	Units
10332	ResonanceFilter2Damping (REF2D)	DSCM Parameter. See comment for parameter 10006. Ignored by axes using Synthetic Motors (Type 65535). Supported axes: 1–4	DINT	Read/Write	NA
10333	ResonanceFilter3Center Frequency (REF3C)	DSCM Parameter. See comment for parameter 10006. Ignored by axes using Synthetic Motor Type 65535. Supported axes: 1–4	DINT	Read/Write	NA
10334	ResonanceFilter3BandWidth (REF3W)	DSCM Parameter. See comment for parameter 10006. Ignored by axes using Synthetic Motor Type 65535. Supported axes: 1–4	DINT	Read/Write	NA
10335	ResonanceFilter3Damping (REF3D)	DSCM Parameter. See comment for parameter 10006. Ignored by axes using Synthetic Motor Type 65535. Supported axes: 1–4	DINT	Read/Write	NA

[1] Parameter can be changed only if the axis is in the Disabled state.

[2] 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-point motion is allowed.

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. “Parameter Errors Caused by Changes in Axis Scaling” in Chapter 9 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.

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 parameter number 1200.	BOOL	Read
2020	Axis2Ok	Also parameter number 1200.	BOOL	Read
2021	Axis3Ok	Also parameter number 1200.	BOOL	Read
2022	Axis4Ok	Also parameter number 1200.	BOOL	Read
2023	Axis5Ok	Also 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
2100	ModuleStatusCode	The most recent, highest severity error code that applies to the module. See Chapter 9, “Diagnostics” 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

<table>
<tr><th>Parameter Number</th><th>Parameter Name</th><th>Description</th><th>Data Type</th><th>Read/Write</th></tr>
<tr><td>2104</td><td>AnalogOutput1Source

Analog Output Scaling<table><tr><td>Torque Command</td><td>8.00V = 100%</td></tr><tr><td>Motor Actual Current</td><td>8.00V = 100%</td></tr><tr><td>Servo Actual Velocity</td><td>10.00V / 8000RPM</td></tr><tr><td>Servo Commanded Velocity</td><td>10.00V / 8000RPM</td></tr></table></td><td>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
35 = Axis 3 Servo Actual Velocity
36 = Axis 3 Servo Commanded Velocity
40 = Axis 4 Torque Command
41 = Axis 4 Motor Actual Current
45 = Axis 4 Servo Actual Velocity
46 = Axis 4 Servo Commanded Velocity</td><td>DWORD</td><td>Read/Write</td></tr>
<tr><td>2105</td><td>AnalogOutput2Source</td><td>Determines the signal that is output to analog output 2.
Analog output 2 source:
valid values same as Analog Output 1 Source.</td><td>DWORD</td><td>Read/Write</td></tr>
<tr><td>2106</td><td>TimedInterruptAcknowledgeTimeout</td><td>The number of milliseconds to wait before checking that a timed interrupt sent to the CPU was acknowledged.
Valid range is 0, 2ms to 40ms.
0 (default) = Do not check for acknowledgement</td><td>DWORD</td><td>Read/Write</td></tr>
</table>

Parameter Number	Parameter Name	Description	Data Type	Read/Write
2107	Array of faceplate input values. Same as parameters 3000 through 3007.	Faceplate Inputs IN1 – IN8 bit (8-31) - reserved bit 7 - FP 24V Input 8 bit 6 - FP 24V Input 7 bit 5 - FP 24V Input 6 bit 4 - FP 24V Input 5 bit 3 - FP 24V Input 4 bit 2 - FP 24V Input 3 bit 1 - FP 24V Input 2 bit 0 - FP 24V Input 1	DWORD	Read
2108	Array of FTB input values. Same as parameters 3032 through 3059.	FTB Inputs IN1 – IN28 bits (28-31) - reserved bit 27 - FTB 5V Input 28 bit 26 - FTB 5V Input 27 bit 25 - FTB 5V Input 26 bit 24 - FTB 5V Input 25 bit 23 - FTB 5V Input 24 bit 22 - FTB 5V Input 23 bit 21 - FTB 5V Input 22 bit 20 - FTB 5V Input 21 bit 19 - FTB 5V Input 20 bit 18 - FTB 5V Input 19 bit 17 - FTB 5V Input 18 bit 16 - FTB 5V Input 17 bit 15 - FTB 24V Input 16 bit 14 - FTB 24V Input 15 bit 13 - FTB 24V Input 14 bit 12 - FTB 24V Input 13 bit 11 - FTB 24V Input 12 bit 10 - FTB 24V Input 11 bit 9 - FTB 24V Input 10 bit 8 - FTB 24V Input 9 bit 7 - FTB 24V Input 8 bit 6 - FTB 24V Input 7 bit 5 - FTB 24V Input 6 bit 4 - FTB 24V Input 5 bit 3 - FTB 24V Input 4 bit 2 - FTB 24V Input 3 bit 1 - FTB 24V Input 2 bit 0 - FTB 24V Input 1	DWORD	Read

Parameter Number	Parameter Name	Description	Data Type	Read/Write
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 bit (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
2111	Faceplate 24V fault status	bit 31 - FP Output Overcurrent bit 2-30) - reserved bit 1 - FP Input 2 Open Wire bit 0 - FP Input 1 Open Wire	DWORD	Read
2112	FTB 5V 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 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	DWORD	Read

Parameter Number	Parameter Name	Description	Data Type	Read/Write
2113	FTB 24V 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 bit 0 - FTB Output 1 Over Temp	DWORD	Read
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

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***
Faceplate Digital Inputs				
3000	Fp24Vin1	Value of the 24V high speed input	BOOL	Read
3001	Fp24Vin2		BOOL	Read
3002	Fp24Vin3	Value of the 24V general purpose input	BOOL	Read
3003	Fp24Vin4	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
3004	Fp24Vin5		BOOL	Read
3005	Fp24Vin6		BOOL	Read
3006	Fp24Vin7		BOOL	Read
3007	Fp24Vin8		BOOL	Read
Fiber Terminal Block Digital Inputs				
3032	FtbIn1	Value of the 24 VDC input	BOOL	Read
3033	FtbIn2		BOOL	Read
3034	FtbIn3		BOOL	Read
3035	FtbIn4		BOOL	Read
3036	FtbIn5		BOOL	Read
3037	FtbIn6		BOOL	Read
3038	FtbIn7		BOOL	Read
3039	FtbIn8		BOOL	Read
3040	FtbIn9		BOOL	Read
3041	FtbIn10		BOOL	Read
3042	FtbIn11		BOOL	Read
3043	FtbIn12		BOOL	Read
3044	FtbIn13		BOOL	Read
3045	FtbIn14		BOOL	Read
3046	FtbIn15		BOOL	Read
3047	FtbIn16		BOOL	Read

Reference Number	Parameter Name	Description	Data Type	Read/Write
3048	FtbIn17	Value of the 5 VDC input	BOOL	Read
3049	FtbIn18		BOOL	Read
3050	FtbIn19		BOOL	Read
3051	FtbIn20		BOOL	Read
3052	FtbIn21		BOOL	Read
3053	FtbIn22		BOOL	Read
3054	FtbIn23		BOOL	Read
3055	FtbIn24	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	FtbIn25	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	FtbIn26	Value of the 5 VDC input	BOOL	Read
3058	FtbIn27	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	FtbIn28	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
Faceplate Digital Outputs				
3128	Fp24FVOut1	The 24V Out1 and 24V 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 24V Out2 and 24V In4 share the same physical faceplate point. You must configure the point as an input or an output in HWC	BOOL	Read/Write

Reference Number	Parameter Name	Description	Data Type	Read/Write
Fiber Terminal Block Digital Outputs				
3160	FtbOut1	Value of the 24 VDC output	BOOL	Read/Write
3161	FtbOut2		BOOL	Read/Write
3162	FtbOut3		BOOL	Read/Write
3163	FtbOut4		BOOL	Read/Write
3164	FtbOut5		BOOL	Read/Write
3165	FtbOut6		BOOL	Read/Write
3166	FtbOut7		BOOL	Read/Write
3167	FtbOut8		BOOL	Read/Write
3168	FtbOut9	Value of the 24 VDC 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
3169	FtbOut10	Value of the 24 VDC 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	Read/Write
3170	FtbOut11	Value of the 24 VDC 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	Read/Write
3171	FtbOut12	Value of the 24 VDC 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	Read/Write
Fiber Terminal Block Analog Inputs				
3256	FtbAlgIn1	Value of the ±10 V analog input.	REAL64	Read
3257	FtbAlgIn2		REAL64	Read
Fiber Terminal Block Analog Outputs				
3288	FtbAlgOut1	Value of the ±10 V analog output.	REAL64	Read/Write
3289	FtbAlgOut2		REAL64	Read/Write

Chapter 9

Diagnostics

Basic diagnostic tools for PACMotion include:

- PMM Error IDs 9-2
  - Accessing Error IDs 9-2
  - Error ID Format 9-2
  - Error ID Reference 9-3
- CPU Error Codes 9-50
- FSSB Servo Alarms 9-55
- PMM Event Queue 9-60
- Diagnostic Logic Blocks 9-65

An additional diagnostic tool, *Data Logging,* is discussed in Chapter 6.

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.

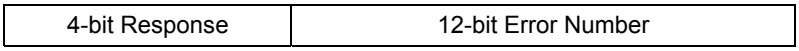
Accessing Error IDs

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 priority 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 page 9-3.

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 “Error ID Reference” on page 9-3.

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.

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.

Error No. (Hex)	Description	Cause	Recommended Correction
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.
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.

Error No. (Hex)	Description	Cause	Recommended Correction
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.
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.
x053	MC_SetPosition absolute digital encoder has not passed through the zero reference	An MC_SetPosition instance was issued for an encoder that has not yet gone through the zero reference point.	After initial power up, the encoder must be rotated past the reference point (up to one revolution) before MC_SetPosition will succeed.
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.

Error No. (Hex)	Description	Cause	Recommended Correction
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 GE Fanuc.
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.
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 1. Defective or incorrectly wired encoder used for external position feedback 2. Incorrect settings for *Position Lag Monitoring, Max Position Lag (UU*) or *Max Position Error (UU)* 3. *Motor Velocity Limit (RPM)* set lower than the commanded servo velocity when *Position Lag Monitoring* is Disabled	Ensure the feedback source for the axis is properly configured and is connected and working properly. 1. Check for a defective or incorrectly wired encoder used for external position feedback. **2.** Review the settings for *Position Lag Monitoring*, *Max Position Lag (UU*) and *Max Position Error (UU).* 3. Review the configuration software setting for *Motor Velocity Limit (RPM).*
x0AB	FSSB interface board not present or failed	Hardware error.	Contact GE Fanuc.
x0AC	FSSB setup failed	FSSB setup failed.	The FSSB setup has failed. Check that the amplifiers are connected via FSSB and that they are powered on. Also verify that the correct number of amplifiers is connected.

Error No. (Hex)	Description	Cause	Recommended Correction
x0B0	Over voltage on DC link.	The servo has experienced an over voltage condition on the DC link. This alarm occurs if the high voltage DC level (DC LINK) is abnormally high. 1. AC voltage supply to the amplifier is too high. 2. External regeneration resistor is wired incorrectly. 3. Regeneration resistor is not capable of dissipating excess generated voltage.	Perform the following checks: 1. The AC voltage supplied to the amplifier must not be higher than the rated input voltage. 2. Carefully check the connections of the regeneration resistor to the amplifier. Check that the resistance of the regeneration resistor is within 20% of the rated value. Replace the regeneration unit if the resistance is out of tolerance. 3. Review the calculations for selecting the regenerative discharge unit and replace with a resistor of higher wattage rating as needed. For details, refer to “FSSB Servo Alarms” on page 9-55.
x0B1	Low voltage control power.	The control voltage used to operate the low voltage circuitry in the amplifier is too low.	For details, refer to “FSSB Servo Alarms” on page 9-55.
x0B2	Servo DC link charge fault.	The main circuit capacitor on the PSM (Power Supply Module for SVM amplifier units) does not charge properly.	For details, refer to “FSSB Servo Alarms” on page 9-55.
x0B3	Low voltage on DC link.	The high voltage DC level (DC LINK) voltage is abnormally low.	Verify that the proper level of AC voltage is supplied to the line input (L1, L2 and L3) connections of the amplifier. For details, refer to “FSSB Servo Alarms” on page 9-55.
x0B4	Servo amplifier overheating	The temperature of the amplifier heat sink is too high or motor temperature is excessive. 1. Ambient temperature too high. 2. Motor is operating in violation of duty cycle restrictions. 3. Motor is overloaded.	1. If ambient temperature is too high, consider a cooling fan for the servomotor. GE Fanuc supplies fan kits for most FANUC motors. 2. Calculate the amount of cooling time needed based on the duty cycle curves published for the particular motor. 3. Check for excessive friction or binding in the machine. For all the above problems, allow ten minutes cooling of the amplifier with minimum or no motor loading then cycle amplifier power to reset.

Error No. (Hex)	Description	Cause	Recommended Correction
x0B5	Cooling fan failure – Internal Fan	The cooling fan has failed.	Perform the following checks: 1. Check the fan for obstructions or debris. With amplifier power removed, attempt to manually rotate the fan. 2. For SVM type amplifier systems the power supply module (PSM) and the servo amplifier module each include a cooling fan. The alarm code will indicate which unit failed. 3. Some amplifiers have field replaceable fan units. If a replacement fan unit is not available, replace the amplifier.
x0B6	SVM PSM IPM Alarm or Over Current	Over current detected by amplifier	The two-segment display on the SVM power supply module (PSM) intelligent power module (IPM) indicates power supply alarms. For details, refer to “FSSB Servo Alarms” on page 9-55. Amplifier alarms not cleared by power cycle of the amplifier will continue to be reported to the PMM.
x0B7	Over discharge alarm	Excessive discharge detected	For recommended corrections, refer to “FSSB Servo Alarms” on page 9-55. Amplifier alarms not cleared by power cycle of the amplifier will continue to be reported to the PMM.
x0B8	DC link high current alarm	Excessive DC current flows into the servo amplifier power inverter.	For recommended corrections, refer to “FSSB Servo Alarms” on page 9-55. Amplifier alarms not cleared by power cycle of the amplifier will continue to be reported to the PMM.
x0BC	Amplifier communication error	A communication error occurred between amplifiers.	For recommended corrections, refer to “FSSB Servo Alarms” on page 9-55. Amplifier alarms not cleared by power cycle of the amplifier will continue to be reported to the PMM.

Error No. (Hex)	Description	Cause	Recommended Correction
x0BD	Radiator fan failure, External Fan	Digital Servo alarm: The cooling fan has failed.	Perform the following checks: 1. Check the fan for obstructions or debris. With amplifier power removed attempt to manually rotate the fan. 2. For SVM type amplifier systems the power supply module (PSM) and the servo amplifier module each include a cooling fan. The alarm code will indicate which unit failed. 3. Some amplifiers have field replaceable fan units. If a replacement fan unit is not available, replace the amplifier.
x0BE	Abnormally high motor current	Motor current is excessive.	For possible causes and recommended corrective actions, refer to “FSSB Servo Alarms” on page 9-55.
x0C0	Servo not ready.	The servo is not showing ready.	For FANUC Digital servos, the amplifier E–Stop input may be activated or an amplifier fault may have occurred.
x0C1	Serial encoder battery low.	The battery on a serial encoder is low or has failed.	The battery must be replaced or the encoder can be configured for Incremental (instead of Absolute) operation.
x0C2	Serial encoder battery failed.	The battery on a serial encoder has failed.	The battery must be replaced or the encoder can be configured for Incremental (instead of Absolute) operation.
x0C3	Servo motor over temperature.	The servo motor or control firmware has reported an over temperature condition at the servo motor.	Check the motion application to ensure that the duty cycle rating for the motor is not being exceeded. Also check the motor mounting to ensure the heat sink and ventilation for the motor are adequate.
x0C4	Soft phase error.	Invalid feedback was detected from the encoder.	A possible cause of this alarm is noise on the encoder feedback cable. Check for other electrical noise sources in the area of the motor and encoder cabling. Isolate these sources from motor/encoder cabling if possible. If error persists consult factory.
x0C5	Loss of encoder.	The module is not communicating with the encoder.	Ensure the servo amplifier is on. Ensure encoder cable is connected. Additionally, check grounding to ensure that grounding is correct.

Error No. (Hex)	Description	Cause	Recommended Correction
x0C6	Error in encoder pulse detection.	The encoder pulse detection circuit has encountered an error.	Ensure the motor is properly grounded. If error persists consult factory.
x0C7	Encoder counter error	The encoder counter circuit has encountered an error.	Ensure the motor is properly grounded. If error persists consult factory.
x0C8	Encoder LED disconnected.	The encoder LED has become disconnected.	Contact GE Fanuc.
x0C9	Encoder CRC failure.	The encoder communications circuit has detected a cyclic redundancy check error.	Check the encoder cable grounding and the motor grounding for possible error sources. Check for other electrical noise sources in the area of the motor and encoder cabling. Isolate these sources from motor/encoder cabling if possible. If error persists consult factory.
x0CD	Encoder EEPROM alarm.	The encoder has reported a fault accessing the EEPROM.	Cycle the power on the amplifier. If error persists consult factory.
x0D1	Over current condition.	The Motor Control firmware detected an over current condition.	Possible sources for this error include: 1. Incorrect Motor Type selected in HWC. 2. Machine back driving motor excessively - Over Duty cycle conditions. If error persists consult factory.
x0D2	Current offset alarm.	The current offset (equivalent to the current value in the stopped state) of the current detector is too large.	Cycle the power on the amplifier. If the problem persists consult factory.
x0D3	Master FSSB data error.	Invalid data was detected on the master FSSB device.	Possible sources for this error include damaged FSSB cable or bad connection. If the problem persists consult factory.
x0D4	Slave FSSB data error.	Invalid data was detected on a slave FSSB device.	Possible sources for this error include damaged FSSB cable or bad connection. If the problem persists contact GE Fanuc.
x0D5	Amplifier illegal interface.	A non-supported amplifier was detected.	Check that the PMM335 supports the connected amplifier.
x0D6	FSSB disconnected.	The FSSB connection was lost.	Check amplifier power. Check FSSB cable connection.
x0D8	Amplifier ID write error.	The amplifier failed to transfer its ID data to the PMM335.	If the problem persists, contact GE Fanuc.
x0D9, x0DA, x0DC	PMM Module hardware error.	A hardware error was detected on the module.	Contact GE Fanuc.
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.

Error No. (Hex)	Description	Cause	Recommended Correction
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.
x0FB	Control loop execution exceeds specified maximum time.	The control loop exceeded its specified time limit.	Contact GE Fanuc.
x0FC	Control loop warning time limit exceeded.	The control loop exceeded its warning threshold.	Contact GE Fanuc.
x0FD	Severe software error.	Internal error.	Contact GE Fanuc.
x0FE	Unrecognized encoder.	An unrecognized encoder is attached.	This error can indicate a defective encoder cable – check cable. If cable checks out correctly, contact GE Fanuc.
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 GE Fanuc.
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. Note: If error causes an active DCS to be aborted then a Normal Stop on the axis will occur.	Correct the DCS track information.
x20B	DCS overlapping switch points.	Invalid inputs to MC_DigitalCamSwitch function block. Note: If error causes an active DCS to be aborted then a Normal Stop on the axis will occur.	Correct switch point information.
x20C	DCS invalid axis direction.	Invalid inputs to MC_DigitalCamSwitch function block. Note: If error causes an active DCS to be aborted, a Normal Stop on the axis will occur.	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. Note: If error causes an active DCS to be aborted, a Normal Stop on the axis will occur.	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 GE Fanuc.
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 powerup, 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.
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.

Error No. (Hex)	Description	Cause	Recommended Correction
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 GE Fanuc.
x228	Unable to communicate with FTB.	An FTB has been configured, but is not communicating with the PMM.	1) Check power to FTB. 2) 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.	1) Check power to FTB. 2) Check fiber cable connecting FTB and PMM.
x22A	FTB module hardware error.	FTB hardware is incompatible with PMM.	Contact GE Fanuc.
x22B	FTB module hardware error (stop axis).	FTB hardware is incompatible with PMM.	Contact GE Fanuc.
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 24V 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 24V 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 24V 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 24V outputs.	Check wiring, or disable FTB OUT 1–4 Power External Connection detection in HWC.
x247	Power failure for FTB outputs 5–8.	There is no external power source for the 24V 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 motor type 65535) must be on higher-numbered axes than any real motor.	Adjust the *Motor Type* settings on the *Axis* tabs in HWC. The default motor type of 65535 must be on higher numbered axes than real motors.

Error No. (Hex)	Description	Cause	Recommended Correction
x249	FANUC servos must be contiguous starting at Axis 1.	All real motors must be configured on lower numbered axes than any synthetic motor (configured as motor type 65535).	Adjust the *Motor Type* settings on the *Axis* tabs in HWC. The default motor type of 65535 must be on higher numbered axes than real motors.
x24A	Specified encoder resolution is not supported on this servo.	*Encoder Counts per Motor Revolution* exceeds the maximum allowed for motor specified in *Motor Type.*	Adjust the *Motor Type* or *Motor Encoder Counts per Motor Revolution* parameter setting in HWC for this axis. For a list of motor type codes with maximum encoder resolutions supported, see "Motor Type Codes" in Chapter 2.
x24B	Motor Velocity Limit is higher than supported by this motor type.	Motor Velocity Limit is set higher than supported by this motor type.	Reduce the Motor Velocity Limit to a value less than or equal to 1.1 × the maximum velocity supported by the motor type.
x24C–x250	Servo Control Board internal errors.	Internal error.	Contact GE Fanuc.
x251	Positive Software End of Travel value from HWC is invalid.	Internal error.	Contact GE Fanuc.
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 GE Fanuc.
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 GE Fanuc.
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, 0.2 ms.
x258	Too many IO Interrupts waiting to be processed	I/O Interrupts are occurring faster than they are being processed.	Contact GE Fanuc.
x259	IO Interrupt was rejected by the RX3i controller. Will try to send again.	I/O Interrupt was refused by the system CPU.	Contact GE Fanuc.

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 GE Fanuc.
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 GE Fanuc.
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 GE Fanuc.
x26D	Synchronization lost.	Module has lost synchronization.	Contact GE Fanuc.
x26E	Short circuit detected on FTB output.	A short circuit has been detected on the specified 24 volt 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 amplifier 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 16384 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.
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.

Error No. (Hex)	Description	Cause	Recommended Correction
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 ≥0 when used with a Rotary axis.
x300	Timeout while waiting for power feedback from the amplifier	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 amplifier and 500ms has elapsed without feedback indicating that the amplifier is powered up.	If the axis state does not transition to StandStill, there is a problem turning the amplifier power on. Examine the fault table and the event queue for errors indicating the nature of the problem powering up the amplifier.
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.

Error No. (Hex)	Description	Cause	Recommended Correction
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 GE Fanuc.
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.
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.

Error No. (Hex)	Description	Cause	Recommended Correction
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 GE Fanuc.
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.
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. ***Note:*** An MC_Reset or MC_ModuleReset must be performed in order for motion to resume after the RX3i returns to Run/Output Enabled mode.
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 GE Fanuc.
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 “Valid Buffer Modes” in Chapter 5.
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.	▪ 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. ▪ 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.
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.	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 “Axis Status Flags” in Chapter 6. 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)

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 GE Fanuc.
x31D	Slave axis failed to transition from synchronized motion.	The slave axis attempted to transition out of the Synchronized Motion state and failed.	Contact GE Fanuc.
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: 1) The master is linear, and already at the master sync position but moving away from it. 2) 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 pended/buffered/blended command.
x329	MC_CamIn with zero ramp distance but master/slave not already synchronized	MC_CamIn specified a zero ramp distance and the slave's position 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 half way to the end of the range during an MC_MoveVelocity	Motion has reached half way 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*** **i**n 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 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 GE Fanuc.
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 “Axis States” in Chapter 5 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.
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 “Axis States” in Chapter 5 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
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 “Axis States” in Chapter 5 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 “Axis States” in Chapter 5 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 “Axis States” in Chapter 5 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 “Axis States” in Chapter 5 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 “Axis States” in Chapter 5 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 “Axis States” in Chapter 5 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 “Axis States” in Chapter 5 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.
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 “Axis States” in Chapter 5 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.

Error No. (Hex)	Description	Cause	Recommended Correction
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.
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.

Error No. (Hex)	Description	Cause	Recommended Correction
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.
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.

Error No. (Hex)	Description	Cause	Recommended Correction
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.
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.

Error No. (Hex)	Description	Cause	Recommended Correction
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.
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.

Error No. (Hex)	Description	Cause	Recommended Correction
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.
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.

Error No. (Hex)	Description	Cause	Recommended Correction
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.	A write parameter command is being used to write an invalid value, based on the current hardware configuration. Examples: ▪ Attempting to use MC_WriteDWordParameter to write an FTB or Faceplate output point when that point is configured as an input. ▪ Attempting to use a write parameters command to write a max position lag that is greater than the max position error. Note: 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.	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.
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.

Error No. (Hex)	Description	Cause	Recommended Correction
x409	Invalid function block Axis parameter.	A function block instance has an invalid Axis parameter. Note 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 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 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.
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.

Error No. (Hex)	Description	Cause	Recommended Correction
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_DL_Configure 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.
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^2. 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^2. 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^3 the minimum jerk allowed is 1/60000 rev/sec^3 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%.

Error No. (Hex)	Description	Cause	Recommended Correction
x420	Invalid function block JerkUnits parameter.	A function block instance has an invalid JerkUnits parameter.	Valid JerkUnits are: *UserUnitsperSecondCubed* (UU/sec^3) 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 LimitSwitchRefPulse, RefPulse and RefPulseNeg. The Virtual Axis supports only RefPulse and RefPulseNeg.
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.
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).

Error No. (Hex)	Description	Cause	Recommended Correction
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.
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. 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.	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 by sync started need to be executed sooner or a longer timeout needs to be set.

Error No. (Hex)	Description	Cause	Recommended Correction
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.
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 (+/– 10V), 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.

Error No. (Hex)	Description	Cause	Recommended Correction
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.
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 “Parameter Errors Caused by Changes in Axis Scaling” on page 9-62.
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 × 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 × UU:Counts ratio).

Error No. (Hex)	Description	Cause	Recommended Correction
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.
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 “Parameter Errors Caused by Changes in Axis Scaling” on page 9-62.
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.

Error No. (Hex)	Description	Cause	Recommended Correction
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 D, "Tuning…" for details on how to use this parameter.
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 D, "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 D, "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 D, "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 D, "Tuning…" for details on how to use this parameter.
x460	DETQLM Deceleration Torque Limit.	An MC_WriteDwordParameters was issued with an invalid value for DETQLM.	The valid range for DETQLM is 0 to 32767. Refer to Appendix D, "Tuning…" for details on how to use this parameter.
x461	BITPA20 Disturbance Input Function	An MC_WriteDwordParameters was issued with an invalid value for BITPA20.	The valid range for BITPA20 is -32767 to 32767. Refer to Appendix D, "Tuning…" for details on how to use this parameter.
x462	DI_GAIN Disturbance input: gain	An MC_WriteDwordParameters was issued with an invalid value for DI_GAIN.	The valid range for DI_GAIN is 0 to 7282. Refer to Appendix D, "Tuning…" for details on how to use this parameter.
x463	S_FREQ Disturbance input: start frequency	An MC_WriteDwordParameters was issued with an invalid value for S_FREQ.	The valid range for S_FREQ is 0 to 2000. Refer to Appendix D, "Tuning…" for details on how to use this parameter.
x464	E_FREQ Disturbance input: end frequency	An MC_WriteDwordParameters was issued with an invalid value for E_FREQ.	The valid range for E_FREQ is 0 to 2000. Refer to Appendix D, "Tuning…" for details on how to use this parameter.
x465	MPOINTS Number of disturbance input	An MC_WriteDwordParameters was issued with an invalid value for MPOINTS.	The valid range for MPOINTS is -32767 to 32767. Refer to Appendix D, "Tuning…" for details on how to use this parameter.
x466	REF2C Res Filter 2 Center Freq (Hz)	An MC_WriteDwordParameters was issued with an invalid value for REF2C.	The valid range for REF2C is 0 to 2000. Refer to Appendix D, "Tuning…" for details on how to use this parameter.
x467	REF2W Res Filter 2 Bandwidth (Hz)	An MC_WriteDwordParameters was issued with an invalid value for REF2W.	The valid range for REF2W is 0 to 2000. Refer to Appendix D, "Tuning…" for details on how to use this parameter.

Error No. (Hex)	Description	Cause	Recommended Correction
x468	REF2D Res Filter 2 Damping (%)	An MC_WriteDwordParameters was issued with an invalid value for REF2D.	The valid range for REF2D is 0 to 100. Refer to Appendix D, "Tuning…" for details on how to use this parameter.
x469	REF3C Res Filter 3 Center Freq (Hz)	An MC_WriteDwordParameters was issued with an invalid value for REF3C.	The valid range for REF3C is 0 to 2000. Refer to Appendix D, "Tuning…" for details on how to use this parameter.
x46A	REF3W Res Filter 3 Bandwidth (Hz)	An MC_WriteDwordParameters was issued with an invalid value for REF3W.	The valid range for REF3W is 0 to 2000. Refer to Appendix D, "Tuning…" for details on how to use this parameter.
x46B	REF3D Res Filter 3 Damping (%).	An MC_WriteDwordParameters was issued with an invalid value for REF3D.	The valid range for REF3D is 0 to 100. Refer to Appendix D, "Tuning…" 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 milliseconds.
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
x472	Invalid DSC 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.

Error No. (Hex)	Description	Cause	Recommended Correction
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.
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.

Error No. (Hex)	Description	Cause	Recommended Correction
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.
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.
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 GE Fanuc.
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.

Error No. (Hex)	Description	Cause	Recommended Correction
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 GE Fanuc.
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 “Cam Operation.”
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 “Cam Operation.”
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 resolved the problem, contact GE Fanuc.
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 “Cam Operation.”

Error No. (Hex)	Description	Cause	Recommended Correction
x816	Absolute Slave, Periodic 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 "Cam Operation."
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 "Cam Operation" 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 GE Fanuc.
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.
x88A–x8A0	Internal error.	Internal error.	Contact GE Fanuc.

Error No. (Hex)	Description	Cause	Recommended Correction
x8A1	Wait for axes to stop before clearing configuration.	Axis is in motion when clear of configuration is attempted.	Wait until axis stops before clearing configuration.
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.
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.

Error No. (Hex)	Description	Cause	Recommended Correction
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 GE Fanuc.
xC10	A file transfer failure occurred while sending the datalog file to the CPU	Internal module error.	Contact GE Fanuc.
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.
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 is 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 GE-Fanuc for further assistance, providing the contents of the fault table and event queue.

Error No. (Hex)	Description	Cause	Recommended Correction
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 GE Fanuc.
xCC0	System Manager detected internal failure.	Internal error.	Contact GE Fanuc.
xCC1	Call to perform module reset operation failed.	An MC_ModuleReset function block issued by the application failed to reset the module.	Contact GE Fanuc.
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 GE Fanuc.
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.
xCC7	System Manager received invalid command.	Internal error.	Contact GE Fanuc.
xCC8	System Manager entered error state.	A system error has been detected that prevents the module from performing motion.	Contact GE Fanuc.
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.

Error No. (Hex)	Description	Cause	Recommended Correction
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 GE Fanuc.
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 FSSB 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 FSSB Servo Alarm
xCD3	The module is being shutdown because of a system error. The axis was fast stopped	Internal errors – Event Queue may have additional data concerning source.	Contact GE Fanuc.
xD00–xD09	Internal hardware error	Internal hardware error.	Contact GE Fanuc.
xD0A	Module exceeded high temperature threshold.	The temperature of the module exceeded the high temperature threshold. (Temp ≥ 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 >= 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 GE Fanuc.
xDC0	Powerup Event.	This event is generated every time the PMM is reset or powered-up.	None. This is an informational message.

Error No. (Hex)	Description	Cause	Recommended Correction
xDC1	Invalid EventID No Stop Response.	Internal error.	Contact GE Fanuc.
xDC2	Invalid EventID Normal Stop Response.	Internal error.	Contact GE Fanuc.
xDC3	Invalid Event ID Fast Stop Response	Internal error.	Contact GE Fanuc.
xDC4	An error occurred while sending the Event Queue file to the CPU.	Internal error.	Contact GE Fanuc.
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.
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.
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.
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 GE Fanuc.
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.

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 in 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.
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.

Error ID (Hex)	Description	Cause	Recommended Correction
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.

Note:	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.		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 "Cam File Format" in Chapter 7.
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.
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.

Error ID (Hex)	Description	Cause	Recommended Correction
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.
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 GE Fanuc.
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.

Error ID (Hex)	Description	Cause	Recommended Correction
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 Ion umber 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 GE Fanuc.
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.
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.

Error ID (Hex)	Description	Cause	Recommended Correction
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.

FSSB Servo Alarms

GE Fanuc α and β FSSB servo systems have built in detection and safety shut down circuitry for many potential error conditions. The following table lists alarms supported by the servo amplifiers and the corresponding error code.

To clear a servo alarm, amplifier power cycle reset is required. Amplifier alarms not cleared by power cycle of the amplifier will continue to be reported to the PMM. For details on diagnostics and corrective actions for these alarms, see page 9-56.

For details on the PACMotion error ID format and a complete listing of error codes, refer to refer to “PMM Error IDs” on page 9-2.

Note: Most of these servo alarm Error IDs will occur with the fast-stop response bit set (0x6xxx). For details, see the table of response types under “Error ID Reference” on page 9-3.

Digital Servo Alarm Error Codes

Error Code (Hexadecimal)	***Servo Alarm Name***	***Description***
x0B0	HV	Over-Voltage DC LINK
x0B1	LV	Low Voltage Control Power
x0B2		DC Link Charge Fault
x0B3	LVDC	Low Voltage DC LINK
x0B4	OH	Amplifier Over Heat
x0B5	FAL	Cooling Fan Failure
x0B6		SVM, PSM, or IPM Alarm or Over Current *
x0B7		Over Discharge Alarm
x0B8		DC Link High Current Alarm
x0BC		Amplifier Comm Error (Communication error occurred between amplifiers.)
x0BD	FAL	Radiator Fan Failure
x0BE	HC	Abnormally High Motor Current
	HCL	L axis
	HCM	M axis
	HCN	N axis
	HCLM	L & M axes
	HCMN	M & N axes
	HCNL	N & L axes
	HCLMN	L, M & N axes

* The two-segment display on the SVM power supply module (PSM) indicates power supply alarms.

Diagnostics for FSSB Servo Alarms

The guidelines below are intended to assist in isolating problems associated with various servo alarms. The appropriate amplifier and motor, maintenance manual or description manual will include more detailed trouble shooting procedures.

HV (High-voltage) Alarm: This alarm occurs if the high voltage DC level (DC LINK) is abnormally high.

1. The AC voltage supplied to the amplifier may be higher than the rated input voltage. The β Series amplifier, three-phase supply voltage should be between 200 VAC to 240 VAC.
2. The external regeneration resistor may be wired incorrectly. Carefully check the connections of the regeneration resistor to the amplifier. Check that the resistance of the regeneration resistor is within 20% of the rated value. Replace the regeneration unit if the resistance is out of tolerance.
3. The regeneration resistor may not be capable of dissipating excess generated voltage. Review the calculations for selecting the regenerative discharge unit and replace with a resistor of higher wattage rating as needed. Reducing acceleration values and position loop gains (larger value *Position Loop Time Constant*) will additionally reduce regenerated voltage levels.

LV (Low Voltage Control Power Alarm): The control voltage used to operate the low-voltage circuitry in the amplifier is too low.

1. Series SVU type amplifiers will be shipped with default jumpers to use a single phase of the 220 VAC power to the amplifier. Optionally the user may remove the jumpers and connect 220 VAC control power separately. Check that a minimum 200VAC is available on terminals L1C and L2C for default installation or on connector CX3 (Y Key) for separate control power.
2. Check the amplifier fuse. If the fuse is open replace with a new fuse after checking control power voltage. If the second fuse blows open, replace the amplifier.

DC Link Charge Fault: This alarm occurs on the PSM (Power Supply Module for SVM amplifier units) if the main circuit capacitor does not charge properly.

1. Check that the number of SVM units does not exceed the PSM specification.
2. The DC link from the PSM to SVM units may be short-circuited. Check the connection.
3. The PSM recharge current limiting resistor may be defective. Replace the distributing board.

LVDC (Low Voltage DC Link: This alarm occurs if the high voltage DC level (DC LINK) voltage is abnormally low.

The AC voltage supplied to the amplifier may be missing or lower in value than the rated input voltage. The β Series amplifier, three-phase supply voltage should be between 200 VAC to 240 VAC. Verify that the proper level of AC voltage is supplied to the line input (L1, L2 and L3) connections of the amplifier.

OH (Over-heat Alarm): The temperature of the amplifier heat sink is too high or motor temperature is excessive.

1. Ambient temperature may be too high, consider a cooling fan for the servomotor. GE Fanuc supplies fan kits for most FANUC motors.
2. The motor may be operating in violation of duty cycle restrictions. Calculate the amount of cooling time needed based on the duty cycle curves published for the particular motor.
3. The motor may be over loaded. Check for excessive friction or binding in the machine.
4. For all the above problems, allow ten minutes cooling of the amplifier with minimum or no motor loading then cycle amplifier power to reset.

FAL (Fan Alarm): The cooling fan has failed.

1. Check the fan for obstructions or debris. With amplifier power removed attempt to manually rotate the fan.
2. For SVM type amplifier systems the power supply module (PSM) and the servo amplifier module each include a cooling fan. The alarm code will indicate which unit failed.
3. Some amplifiers have field replaceable fan units. If a replacement fan unit is not available, replace the amplifier.

HC, HCL, etc. (High Current Alarm): Motor current is excessive.

For α Series amplifiers the suffix (L, M, N, etc.) indicates which axis is in alarm.

1. Motor power wiring (U, V and W) may be shorted to ground or connected with improper phase connections. Check the wiring and connections. Check the servomotor for shorts to motor frame. Replace the motor if shorted.
2. Improper motor type code may be configured or excessive values for tuning parameters. Confirm that the proper motor is configured and lower gain values.
3. Excessive noise on motor current signals. The amplifier maintenance manual will describe the procedure for monitoring motor current signals (IR and IS). If the waveforms are abnormal replace the amplifier. If excessive noise is observed check grounds and especially the cable shield grounds for the command cable (K1) to the amplifier.
4. The motor may be operating in violation of duty cycle restrictions. Calculate the amount of cooling time needed based on the duty cycle curves published for the particular motor.
5. The motor may be over loaded. Check for excessive friction or binding in the machine.
6. With some motor types, abnormally high motor current may occur when very high accelerations are commanded to negative velocities greater than –5000 RPM.

   To prevent the error from occurring, reduce the acceleration rate so the motor does not operate in torque limit. If the acceleration cannot be reduced, enter hardware configuration Advanced Tab parameter 10019 (EMFCMP) with data = 0.

For all the above problems, allow ten minutes cooling of the amplifier with minimum or no motor loading then cycle amplifier power to reset.

PML, IPMM, etc. (IPM Alarm): The Intelligent Power Module (IPM) is the high current switching device in the amplifier. The IPM can detect over-current, over-heat or low-voltage conditions in the power switching circuitry. The suffix (L, M, N, etc.) indicates which axis is in alarm.

1. Motor power wiring (U, V and W) may be shorted to ground or connected with improper phase connections. Check the wiring and connections. Check the servomotor for shorts to motor frame. Replace the motor if shorted.
2. Improper motor type code may be configured or excessive values for tuning parameters. Confirm that the proper motor is configured and lower gain values.
3. The amplifier maintenance manual will describe the procedure for monitoring motor current signals (IR and IS). If the waveforms are abnormal replace the amplifier. If excessive noise is observed check grounds and especially the cable shield grounds for the command cable (K1) to the amplifier.
4. The motor may be operating in violation of duty cycle restrictions. Calculate the amount of cooling time needed based on the duty cycle curves published for the particular motor.
5. The motor may be over loaded. Check for excessive friction or binding in the machine.

For all the above problems, allow ten minutes cooling of the amplifier with minimum or no motor loading then cycle amplifier power to reset.

Over Discharge Alarm: The connection to the external discharge resistor, the resistor itself, or the jumper that must be installed if no external resistor is used has been compromised or is missing.

1. The external regeneration resistor may be wired incorrectly. Carefully check the connections of the regeneration resistor to the amplifier. Check that the resistance of the regeneration resistor is within 20% of the rated value. Replace the regeneration unit if the resistance is out of tolerance.
2. If an external regeneration resistor is not used, check that the required jumper is properly installed.
3. Verify that the amplifier's front panel switches that configure the regeneration resistor are correctly set.

DC Link High Current Alarm: This alarm occurs if excessive DC current flows into the servo amplifier power inverter.

1. If the alarm only occurs when the motor is under a high acceleration or deceleration condition then increase the acceleration / deceleration time.
2. Check the insulation of the motor power leads and motor. If the insulation is deteriorated then replace the power leads and / or motor.
3. If the checks in 1 and 2 do not indicate a problem, replace the SVM unit.

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 MC_ReadEventQueue discussion in Chapter 6.

Event Queue Details

Relative Time	Time stamp of event. The time stamp is a counter that starts running at powerup. The resolution of the timer/counter is 50µsec. Thus, a value of 25 for relative time is 25 × 50µsec = 1250µsec.
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, see “Parameter Errors Caused by Changes in Axis Scaling” on page 9-62. 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.

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	0089175776	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	0x3800	0x0A5F0000	Normal Stop	PLC mode change aborted function block
78	0091434507	Axis 2	Informational	0x530F	0x3801	0x0A5F0000	Normal Stop	PLC mode change aborted function block
79	0091434507	Axis 3	Informational	0x030F	0x3802	0x0A580000	No Stop	PLC mode change aborted function block
80	0091434507	Axis 4	Informational	0x030F	0x3803	0x0A580000	No Stop	PLC mode change aborted function block
81	0091434507	Axis 5	Informational	0x030F	0x3804	0x0A580000	No Stop	PLC mode change aborted function block
82	0091434507	Axis 1	Informational	0x530F	0x3800	0x0A5F0000	Normal Stop	PLC mode change aborted function block
83	0091434507	Axis 2	Informational	0x530F	0x3801	0x0A5F0000	Normal Stop	PLC mode change aborted function block
84	0091434507	Axis 3	Informational	0x030F	0x3802	0x0A580000	No Stop	PLC mode change aborted function block
85	0091434507	Axis 4	Informational	0x030F	0x3803	0x0A580000	No Stop	PLC mode change aborted function block
86	0091434507	Axis 5	Informational	0x030F	0x3804	0x0A580000	No Stop	PLC mode change aborted function block
87	0092438905	Module	Informational	0x0F45	0x2D0F	0x001F0004	No Stop	Assume rack synch mastership
88	0092934514	Axis 1	Informational	0x0421	0x3300	0x13220003	No Stop	Invalid function block Direction parameter
89	0094572098	Axis 1	Informational	0x032D	0x3400	0x06750000	No Stop	Axis has reached half way to the end of the range during an MC_MoveVelocity
90	0094721936	Axis 1	Error	0x532B	0x3420	0x069B0000	Normal Stop	MC_MoveVelocity has reached the limit of the axis and has stopped
91	0094723975	Axis 1	Informational	0x032A	0x3400	0x03A40003	No Stop	Axis power forced off due to an ErrorStop

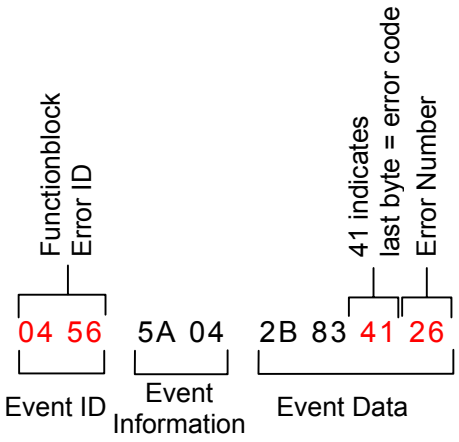
Parameter Errors Caused by Changes in Axis Scaling

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 on the next page. 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.



Scaling-Dependent Parameter Errors

Error Number	Failed Dependent Parameter	Likely Cause	Scaling	Scaling Parameter Numbers*
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
0x21	Command Low Position Limit	Command Position Range + Motor Encoder Low Position > Maximum Position	Command Position Resolution	1000

Error Number	Failed Dependent Parameter	Likely Cause	Scaling	Scaling Parameter Numbers*
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, see “Parameter Index” in Chapter 8.

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 CPU Reference Manual,* GFK-2222.

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

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 Chapter 6.

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.

Motor Encoder

The motor encoder provides feedback information via a serial interface that allows position information to be sampled every 125μs. The 125μs position sampling is time stamped when the actual touch probe input occurred. The time information allows the module to increase the positional accuracy of the touch probe data by estimating the actual position at the time of the touch probe trigger. The estimation uses the position read back from the encoder, the motor velocity and the time information to calculate the compensation. The velocity used for the calculation is derived from the difference in the previous encoder position readings before the touch probe event (see the formula below).

Given:

N – Current Sampling Period (125us)

V1 - 1st Previous Position Sample before touch probe event

V2 - 2nd Previous Position Sample before touch probe event

$$Velocity = \frac{V1 - V2}{N}$$

For touch probe events that occur when velocity is constant during the sampling period, the interpolation algorithm is accurate to within one count and the position capture inaccuracy is primarily determined by the filtering and sampling delays.

The compensation scheme does not compensate for acceleration. The following example can be used to calculate the worst-case inaccuracies due to motor acceleration.

Given the following values for this example:

Encoder Resolution = 65,536 counts/rev

A = Acceleration/deceleration during the touch probe event which is 2,000,000,000 counts/sec^2 (Assumed to be constant over the entire 125μs period. Larger acceleration values will increase the amount of error in the calculation.)

T_p = Position sampling period which is 125μs

V_I = Initial velocity just before the touch probe event, which will be 0 for this example.

The change in the number of encoder counts for a given amount of time (t) can be calculated using the following formula:

$$P_{act} = V_I\ t + \tfrac{1}{2} A\ t^2$$

Therefore the total number of counts to occur during the sampling period for this example is approximately 16 counts (actual calculated value is 15.625) or 0.08583 degrees of motor rotation.

The average velocity for the sample period given the change in position would be as follows:

$$V_{avg} = \frac{\text{Change in counts}}{\text{Sampling Period}} = \frac{15.625 \text{ counts}}{125\mu\text{sec}} = 125{,}000 \text{ counts/sec}$$

The following formula can be used to estimate the touch probe position using the velocity derived above:

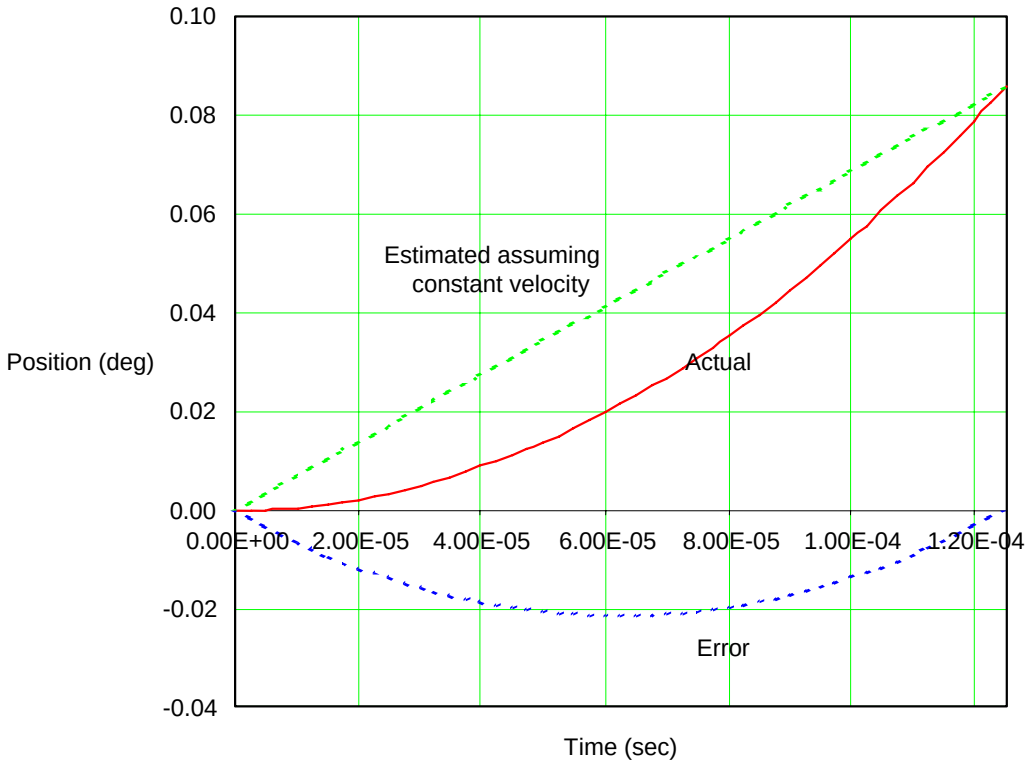
$$P_{est} = V_I\ t + V_{avg}\ t$$

The error between the estimated touch probe position and the actual touch probe position is as follows:

$$\text{Error} = P_{act} - P_{est}$$

Example Axis Position Capture Error due to Acceleration

The following graph contains plots of the actual position, the estimated position, and the resulting touch probe position count error for the 125µs sample period. The graph shows that the greatest count error occurs in the middle (i.e. at 62.5µs) of the period.



Since the initial velocity is equal to 0, the formula for calculating P_{act} can be manipulated to determine the time that the count actually occurred at (T_{act}) as follows:

$$T_{act} = \sqrt{\frac{2P_{act}}{A}}$$

Likewise, the formula for estimating the touch probe position (P_{est}) can be solved for time (T_{est}), assuming that the initial velocity is 0:

$$T_{est} = P_{est} / V_{avg}$$

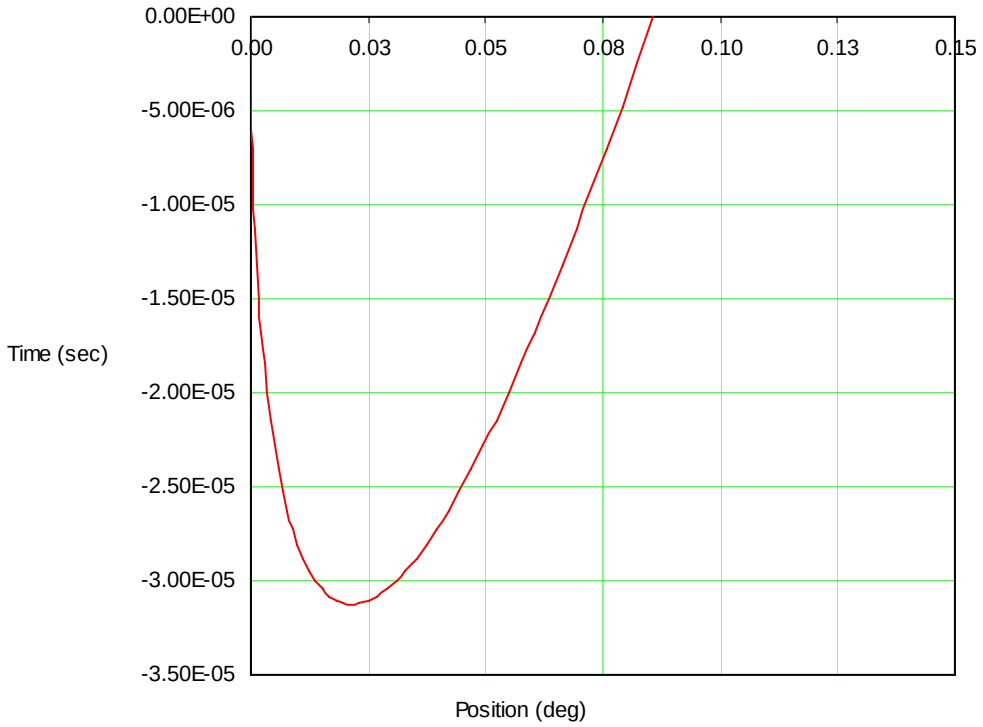
Using these formulas, the difference in time between when the touch probe occurred and when the reported count occurred (i.e. the effective delay) can be calculated as follows:

$$\text{Effective Delay} = T_{est} - T_{act}$$

The effective delay for the maximum touch probe position error (at 62.5 µs) is equal to -31.25µs. This value is negative because the estimated/reported touch probe position occurred prior to the actual position when the touch probe event happened.

Effective Response Time Delay

The following graph shows the effective delay that would be seen across the change in position for the sampling period in this example.



Therefore, in the example above, the worst-case error due to acceleration/deceleration can be expressed as +/-0.02146 degrees (approximately 4 counts of position) or as 31.25µs of delay — given that the initial velocity is 0).

The formulas above assume constant acceleration throughout the sampling period. The formulas for determining the error for the cases where acceleration is not constant during the sampling period are too complex for the context of this manual.

Note that an additional error of 100ns – 0.5ms may be seen due to hardware input filtering and sampling delays.

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.

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 are 125μs old – they were captured by hardware 125μs earlier.
- 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 625μs (0.625ms) after the associated input position was captured by hardware. A timer value of 1.0ms causes the DCS output to change state 1625μs (1.625 ms) after the associated input position was captured by hardware.
- Therefore the DCS firmware extrapolates or looks ahead by 625μs to 1625μs to control a digital output.

For details on the function blocks related to Digital Cam Switch operation, refer to Chapter 6.

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 5v output	< 1 us
Fiber Terminal Block 24v 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 / (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 = 10μs. This inaccuracy occurs because the encoder only provides new position data every 10μs.

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):

Velocity = constant

Position change = velocity * time

Linear Acceleration:

Velocity = Acceleration * time

Position change = 0.5 * Acceleration * 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 Acceleration * 0.5ms * 1.625ms.

   EP1 = A * 0.5ms * 1.625ms

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 * (1.625ms)^2.

   EP2 = 0.5 * A * (1.625ms)^2

   The total error in extrapolated position EPtotal = EP1 + EP2

   = [A * 0.5ms * 1.625ms] + [0.5 * A * (1.625ms)^2]

   = [0.5 * A * 1.0ms * 1.625ms] + [0.5 * A * 1.625ms * 1.625ms]

   = [0.5 * A * 1.625ms] * [1.0ms + 1.625ms]

   = [0.5 * A * 1.625ms] * [2.625ms]

   EPtotal = 2.133e-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 1.625ms extrapolation point is the measured DCS velocity plus the velocity change over an acceleration time of (0.5ms + 1.625ms) = 2.125ms.

   Vactual = Vmeas + (2.125e-3 * A)

   where:

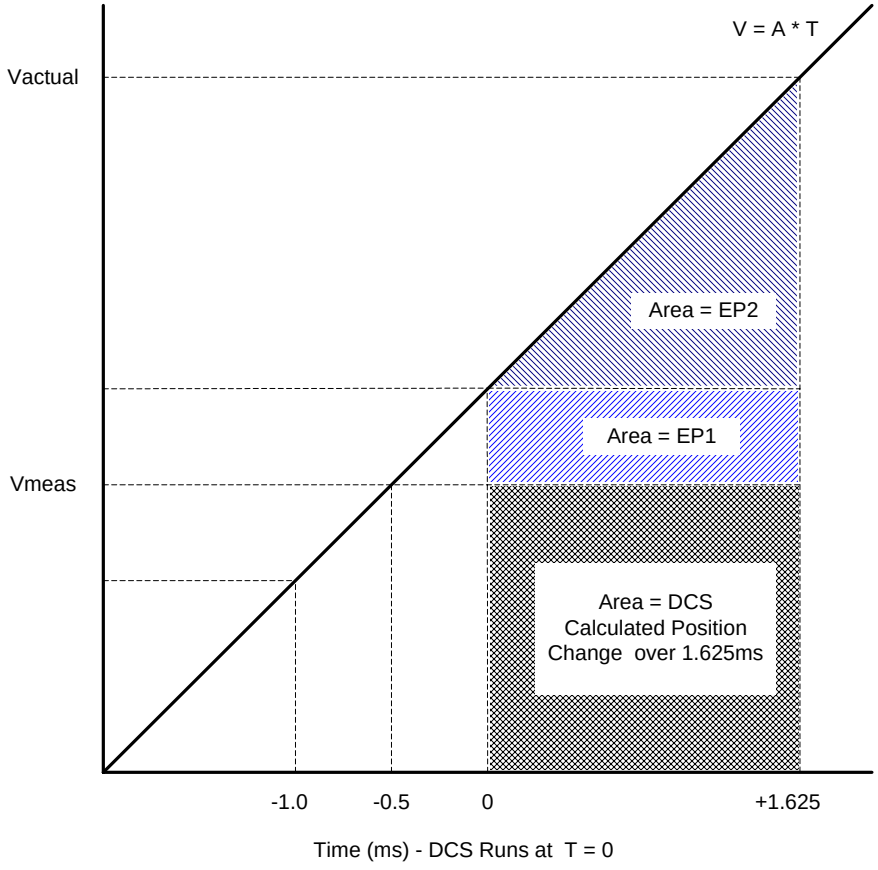
   A = acceleration in position units/sec/sec

   Vmeas = velocity in position units/sec

DCS Calculated Position Change and Position Error Terms

The following graph of velocity vs time shows:

- Velocities Vmeas and Vactual
- Area representing DCS calculated position change over 1.625ms
- Areas representing DCS position error terms EP1 and EP2



4. Dividing the worst case extrapolation error in position (EPtotal) by the actual velocity at the 1.625ms extrapolation point (Vactual) provides the DCS extrapolation error in units of time:

   Etime (seconds) = EPtotal / Vactual

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.

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.

DCS Error vs Position Source Acceleration and Velocity						
	Example 1	***Example 2***	***Example 3***	***Example 4***	***Example 5***	***Example 6***
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)	1,310,720	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
EPtotal = DCS Extrapolation Error (cts)	**186**	**93**	**23**	**23**	**9**	**93**
Etime = 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 EPtotal, Vmeas and Etime can be used to calculate the worst case DCS error:

Vactual = Vmeas + (2.125e-3 * A)

EPtotal = 2.133e-6 * A

Etime (seconds) = EPtotal / Vactual

where

A = acceleration in position units/sec/sec

Vmeas = velocity in position units/sec

Calculation Example

DCS position source acceleration = 100 user units/sec/sec

DCS position source velocity = 8 user units/sec

Vactual = 8 + (2.125e-3 * 100) = 8.2125 user units/sec

EPtotal = 2.133e-6 * 100 = 213.3e-6 user units

Etime = 213.3e-6 / 8.2125 = 25.97 microseconds

Note: The total DCS error is the sum of the acceleration related error and the errors discussed in "*DCS Accuracy at Constant Velocity*" on page A-5.

Appendix B Position Feedback Devices

Digital Serial Encoder Resolutions

The motor that is selected determines which encoder should be used. Please consult the motor specification for additional details regarding the specific encoder utilized for that motor.

64K	65,536 counts/rev
128K	131072 counts/rev
1024K	1048576 counts/rev

Digital Serial Encoder Modes

The FANUC digital serial encoders can be operated in either Incremental mode or Absolute mode. The mode is configured using the *Motor Encoder Mode* selection in the configuration software. Absolute mode requires an external battery pack connected to the servo amplifier. Refer to the appropriate amplifier manual for battery pack selection and installation.

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 A*ctual 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 digital serial encoder can be used as an absolute type encoder by adding a battery pack to retain servo position while system power is off. A *Find Home* cycle or *Set Position* command must be performed initially or whenever encoder battery power is lost with the servo amplifier also in a powered down state. *Motor Encoder Mode* must be set to *Absolute* in the hardware configuration for proper operation with a battery pack.

Absolute Encoder - First Time Use or Use After Loss of Encoder Battery Power

The absolute encoder temporarily provides incremental data during the first use or after restoring encoder battery power. The incremental data is lost when motor shaft rotation causes the encoder to pass a reference point (similar to a marker signal) within one revolution of the motor shaft. The Digital Absolute serial encoder must be rotated up to one full revolution after the absolute mode battery has been reattached to the amplifier. The encoder will reference itself within one revolution and report a referenced status to the PMM.

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* can be operated in Move (+), Move (–) or Home Switch mode. 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.

If a Set Position command is received before the encoder has been referenced, Error Code 53(hex) “MC_SetPosition absolute digital encoder has not passed through the zero reference” will be reported. This error code is only reported if the Motor Encoder Mode is set to Absolute. Serial Encoders configured for Incremental mode do not have this restriction.

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 battery pack attached to the servo subsystem maintains power to the encoder counter logic. Once the encoder has referenced through first time start up, the encoder automatically maintains the actual position, even if the axis is moved during servo power loss. The encoder monitors the status of the battery pack, and reports low battery power or loss of battery power to the PMM. If Motor Encoder Low Battery Alarm is enabled in hardware configuration, these conditions will be reported as errors 00C1 and 00C2 respectively.

The PMM completes a power-on diagnostics, and when configured for absolute encoder mode, interrogates the referenced status of the Digital serial 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 not rotated one revolution after SuperCap discharged and power was reapplied.
- 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: Low Limit + 0.5 * Range
Linear Mode: Negative SW EOT + 0.5 * (Positive SW EOT - Negative SW EOT)

External Quadrature Encoder

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.

Appendix C Startup and Tuning of a FSSB Servo Drive

This chapter provides an introduction to the basics required for tuning a FANUC FSSB servo drive. This introduction shows a method for servo drive tuning. The method may not work in all applications, and the approach should be modified based on the application.

Before tuning the drive, first make sure that the Configuration parameters discussed in Chapter 4 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.

There are two control loops in the PMM 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.

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, see the discussion of the MC_DL_ function blocks in Chapter 6.

Validating Axis Direction, Over Travel Switch and Home Switch Inputs C-2
Tuning a FANUC FSSB Servo Drive C-6

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.

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 a 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.

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 ½ 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.

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 Chapter 3). 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 a MC_ReadDwordParameters 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 a 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 a MC_ReadDwordParameters function block to monitor the bits in parameters 2107 (PMM Faceplate Inputs 1-8) and 2108 (FTB Inputs 1-28).

Validating the Home Switch Input

If a Home Switch is used, wire it to the correct PMM faceplate or FTB input point (refer to Chapter 3). The Home Switch must be assigned to the correct PMM faceplate or FTB 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 a 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 5V 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. Refer to Chapter 6 for details of the MC_Home function block. To confirm that the encoder reference pulse operates correctly, program a 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 a 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 then remove the Enable from MC_Power or use a 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 milliseconds before the encoder reference point is encountered.

FSSB Servo System Startup Diagnostics

1. A powered servo drive corresponding to every axis enabled as FSSB Servo on the Settings tab of the PMM hardware configuration must be present in the FSSB chain. Any axis that does not have a corresponding servo drive will be disabled and reported with an “FSSB Disconnected” error.
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 a 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 a MC_ReadStatus function block.
4. If an axis is in ErrorStop state, the error condition must be corrected and the error cleared using a 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. Check the error code words for Dxxx errors, which are documented in the “FSSB Servo Alarms” section of Chapter 9, “Diagnostics.” Also check the fault tables for reported configuration errors.

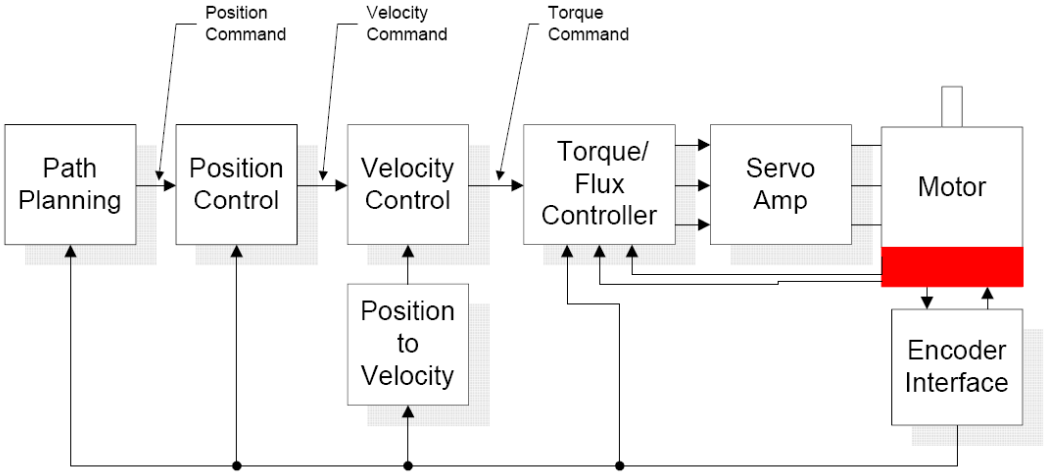
Tuning a FANUC FSSB Servo Drive

Tuning Requirements

The module has three main parameters that are adjusted during tuning. The parameters are the *Load Inertia Ratio, 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 the figure below, the position loop is the outer loop and sends velocity commands to the velocity loop.

Control Loops Block Diagram



Tuning the Velocity Loop

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.

Method #1

The *Force Servo Velocity* parameter (PN1311) will send a velocity command directly to the velocity loop. This command is different from the MC_JogAxis or MC_MoveVelocity function blocks, which uses 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 the user to generate a step change in the velocity. The velocity command step is then used to generate the velocity loop step response. The user should 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.

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* to 100 percent. You can then use MC_JogAxis or MC_MoveVelocity function blocks to generate velocity commands to the servo drive.

The first parameter that needs to be adjusted is the *Load Inertia Ratio.* The parameter adjusts the velocity loop bandwidth. As a starting point use the following formula:

Equation 1

$$\text{Load Inertia Ratio} = \frac{J_L}{J_M} \cdot 256$$

Where :

J_L = Load Inertia

J_M = Motor Inertia

The *Load Inertia Ratio* calculated in equation 1 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:

1. Use the Data Logger Window to display the *Servo Actual Velocity* parameter (PN1314) and the *Torque Command* parameter (PN1304).
2. Set the *Load Inertia Ratio* to zero. 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. *Load Inertia Ratio* can be set in the Hardware Configuration or by using a MC_WriteDword_Parameters function block to update the Load Inertia Ratio parameter (PN10032).

3. Generate a velocity command step change. At this point the step change should be relatively small compared to the full speed of the machine. Ten to 20 % of the rated machine speed is a good start.
4. 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.
5. 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 6 below.
6. 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 *Servo Actual Velocity* and *Torque Command* signals for any instability. If an instability or resonance is observed, reduce the *Load Inertia Ratio* and repeat the test.

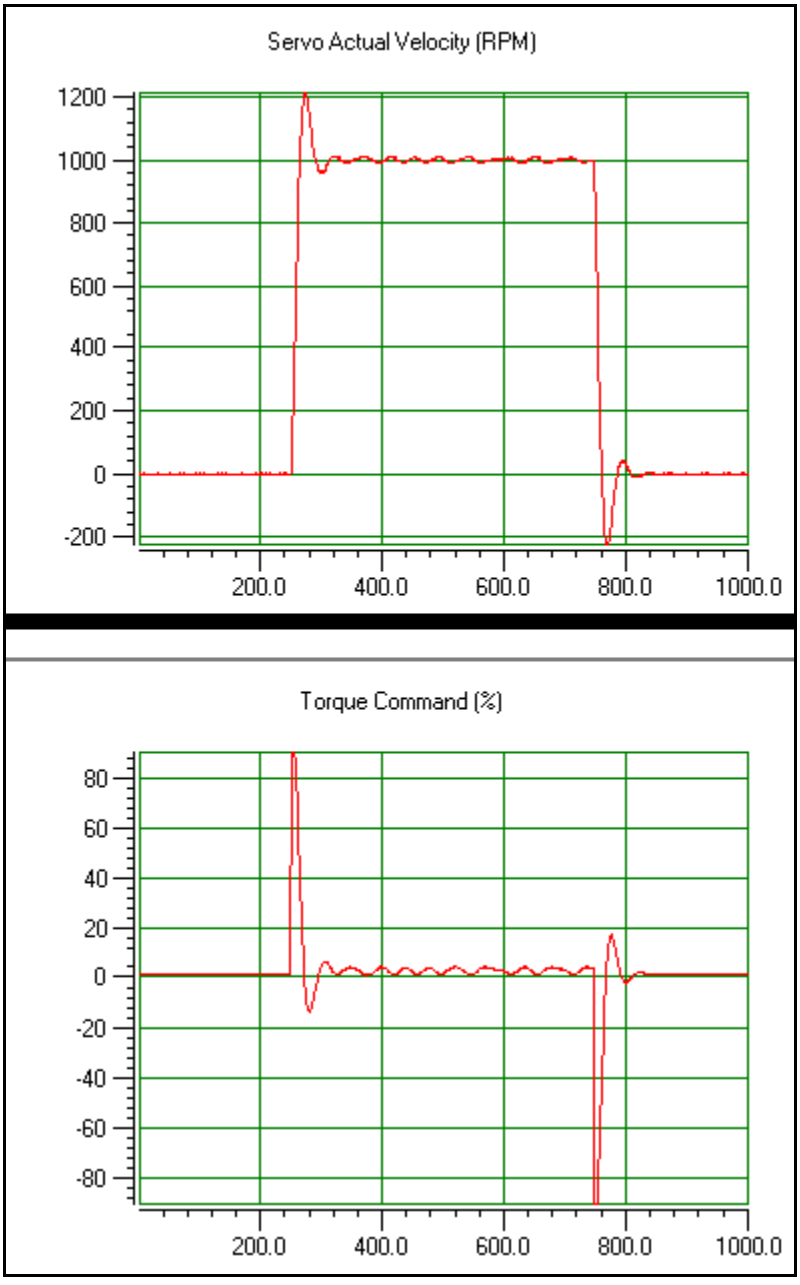
Sample Velocity Loop Tuning Session

A sample velocity loop tuning session is shown in the Data Logger plots that follow.

1. Start with Load Inertia Ratio = 0

In the following figures, the system does not have enough damping. In this case the controller does not have the required bandwidth and the Load Inertia Ratio must be increased.

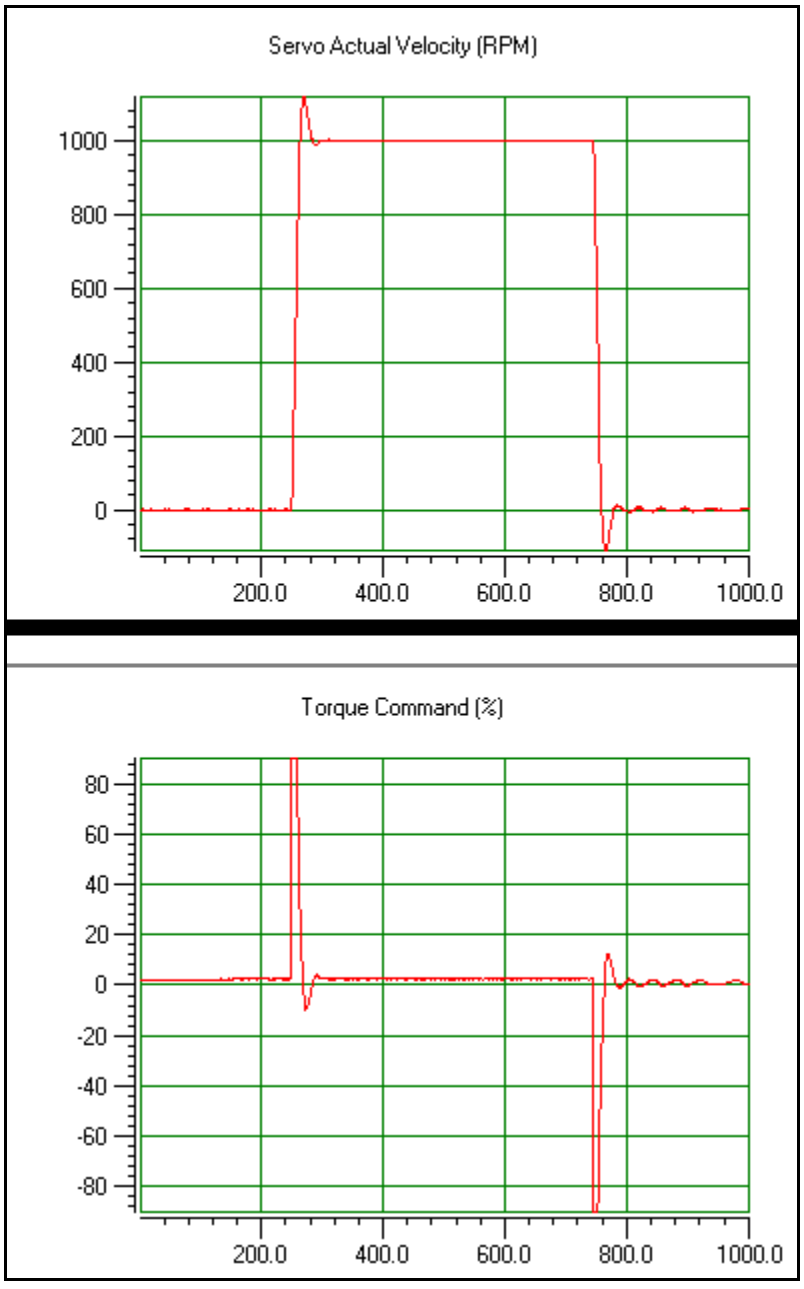
Servo Actual Velocity and Torque Command vs. Time (ms), Load Inertia Ratio = 0



2. *Increase Load Inertia Ratio to improve damping*

In the following figures, the system is beginning to look acceptable. The only problem is the velocity overshoot.

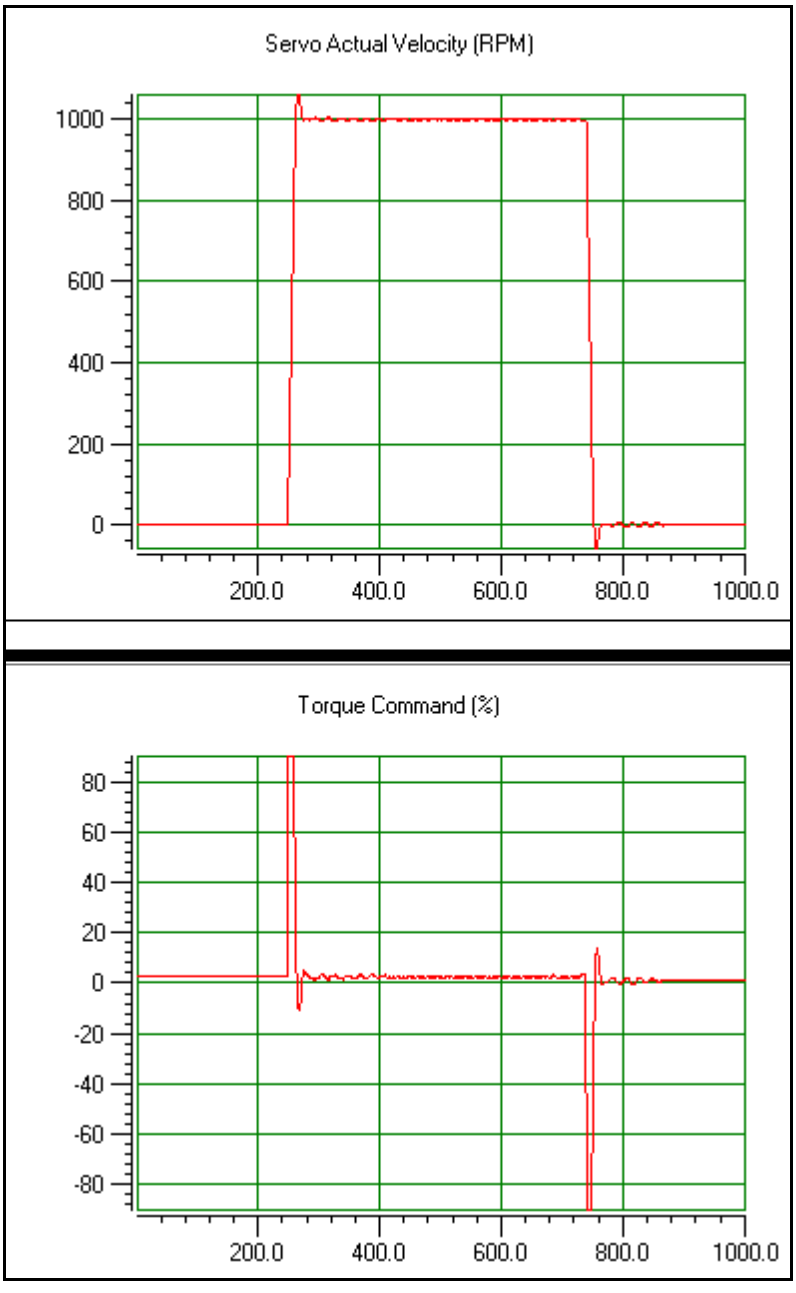
Servo Actual Velocity and Torque Command vs. Time (ms), Load Inertia Ratio = 128



3. *Adjust Load Inertia Ratio again to reduce velocity overshoot*

The response shown in the following figures is good.

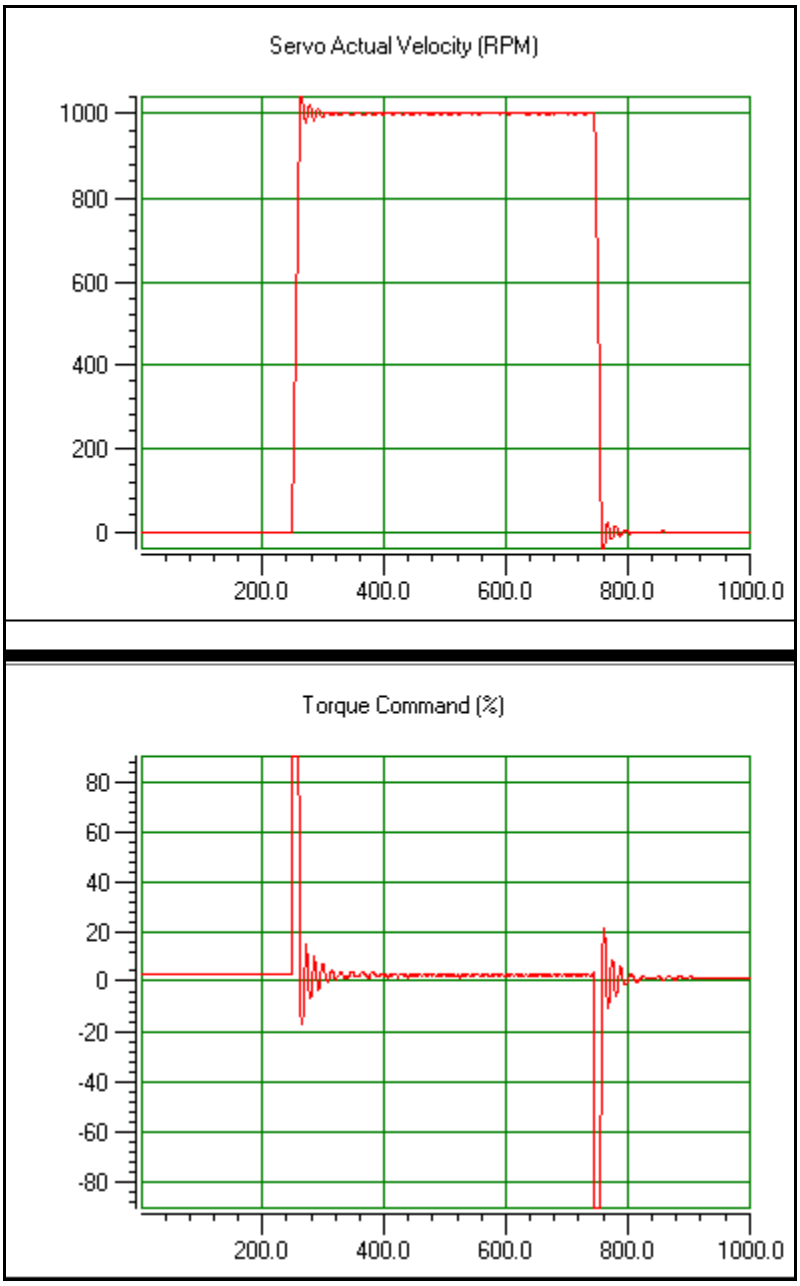
Servo Actual Velocity and Torque Command vs. Time (ms), Load Inertia Ratio = 384



4. Adjust Load Inertia Ratio again to reduce velocity overshoot

The response shown in the following figures has less velocity overshoot but excessive ringing is starting to appear in the Torque Command. The optimum Load Inertia Ratio is less than 768.

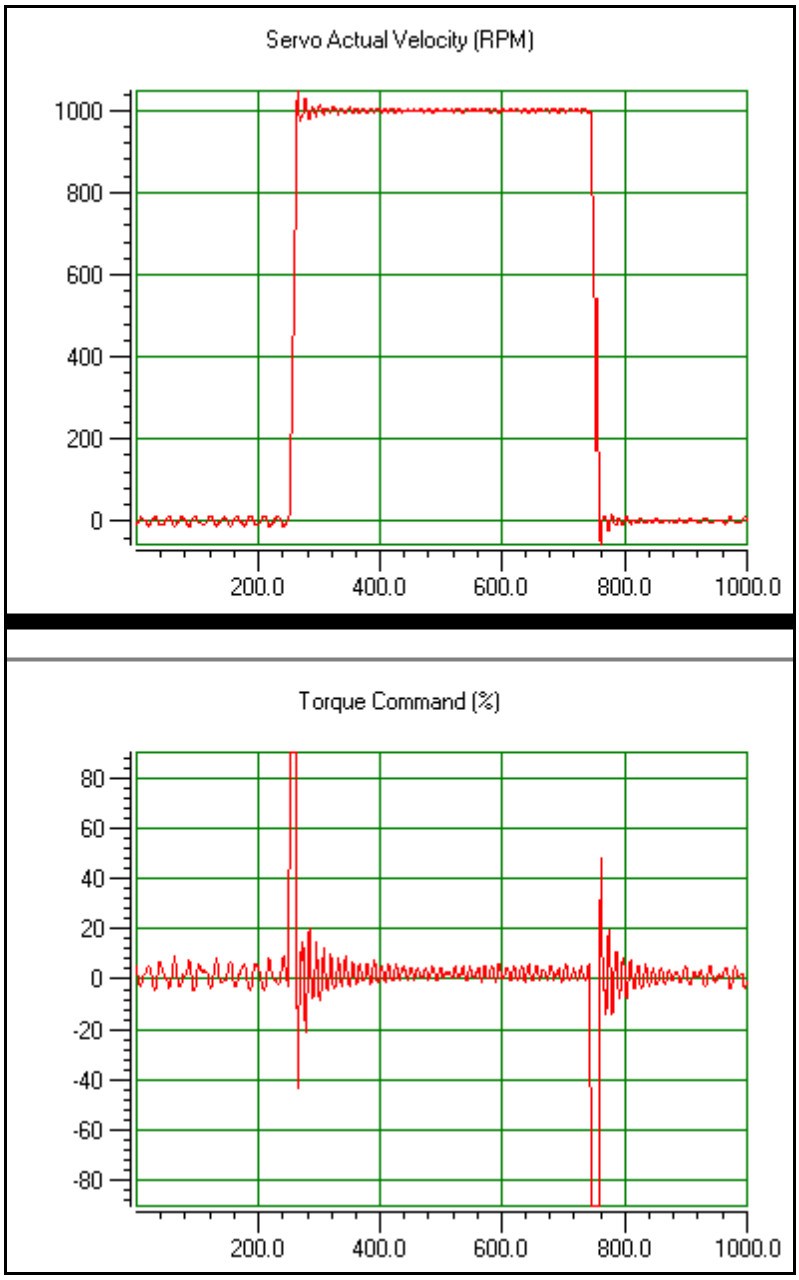
Servo Actual Velocity and Torque Command vs. Time (ms), Load Inertia Ratio = 768



5. *Do not over-adjust Load Inertia Ratio*

The response shown in the following figures is marginally stable and would be unacceptable in most applications. The plots are shown for reference only.

Servo Actual Velocity and Torque Command vs. Time, Load Inertia Ratio = 1536



Tuning the Position Loop

The first step in adjusting the tuning for the position loop is to insure 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

1. 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.
2. 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.
3. Set *Velocity Feedforward* % to zero.
4. 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 Chapter 4). *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 milliseconds. 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 second time constant will reduce 68% of the remaining command. Subsequent time constants will reduce 68% of remaining command. For example 100% - 68% (one time constant) = 32%, 32%(68%)=21.8%, 68% (first time constant) + 21.8% (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 mSec, the 200 mSec response to the change in command will be 98.9% complete in three time constants. Dividing the 200 mSec by 3 results in a time constant of about 67 mSec. **The *Position Loop Time Constant* parameter represents one time constant in ms.** In the example above one time constant is 67**ms**.

Position Loop Proportional Gain Method Method 2

This procedure is similar to the velocity loop tuning method described on page C-7. Using the Data Logging Window to monitor the *Servo Actual Velocity*, lower the *Position Loop Time Constant* value (increasing gain) to obtain the performance characteristics desired.

Optimizing Velocity Feedforward

Before setting *Velocity Feedforward*, the *Load Inertia Ratio* and *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 Chapter 4). *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, see Chapter 4.

Advanced Servo Tuning

Advanced Servo Tuning makes use of the digital servo parameters listed in the following table. These parameters can set in the hardware configuration Advanced Tab (refer to Chapter 4) or using the MC_WriteParameter command (refer to Chapter 6).

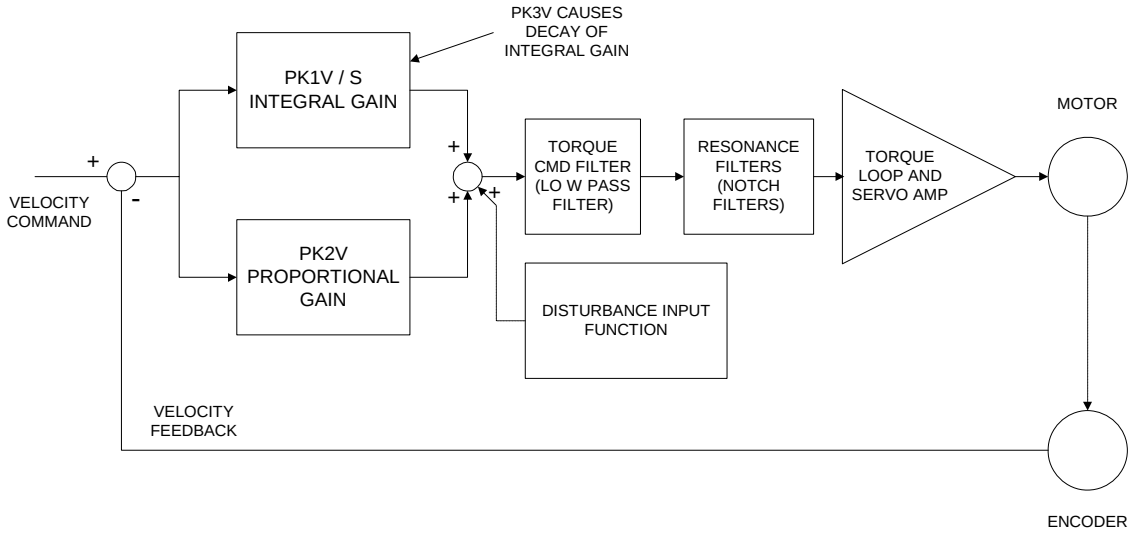
Digital Servo Parameters

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)
10019	-32768	0	EMFCMP, Current Loop Compensation
10031	0	2810	TCMD Torque Command Filter
10086	0	32767	DETQLM, Deceleration Torque Limit
10256	-32768	32767	BITPA20, Disturbance Input Function
10296	0	7282	DI_GAIN, Disturbance input: gain
10297	0	2000	S_FREQ, Disturbance input: start frequency
10298	0	2000	E_FREQ, Disturbance input: end frequency
10299	-32768	32767	MPOINTS, Number of disturbance input measurement points
10330	0	2000	REF2C, Resonance Filter 2 Center Freq (Hz)
10331	0	2000	REF2W, Resonance Filter 2 Bandwidth (Hz)
10332	0	100	REF2D, Resonance Filter 2 Damping (%)
10333	0	2000	REF3C, Resonance Filter 3 Center Freq (Hz)
10334	0	2000	REF3W, Resonance Filter 3 Bandwidth (Hz)
10335	0	100	REF3D, Resonance Filter 3 Damping (%)

The following illustration is a simplified block diagram of the digital servo velocity loop, showing these elements:

- Velocity loop gain parameters PK1V, PK2V, PK3V
- Torque Command Filter
- Resonance Filters

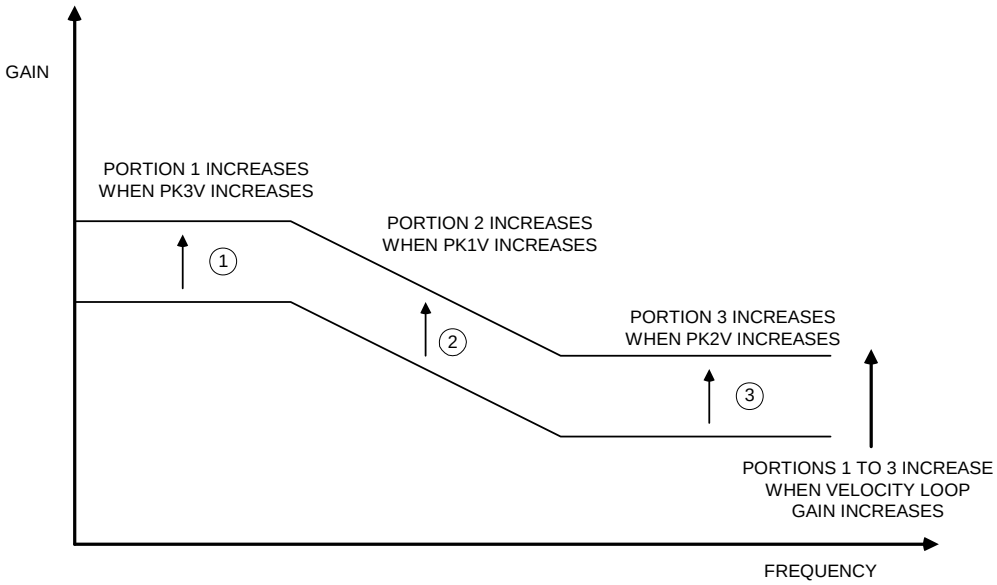
Digital Servo Velocity Loop Simplified Block Diagram



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



- **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

PK1V—PK3V are set to specific values for each motor type, as listed in “Parameters for HRV2 Control” in the *FANUC AC Servo Motor Parameter Manual,* B-65270EN. Normally the values of PK1V-PK3V should not be changed and the velocity loop should be tuned using the Load Inertia Ratio parameter.

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, refer to “Tuning the Velocity Loop” on page C-7.

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:

Tau (seconds) = 1.0 millisecond / ln (32768/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 milliseconds
30000	11 milliseconds
25000	3.7 milliseconds

EMFCMP Current Loop Compensation

This parameter should only be changed if an overcurrent alarm (BE error) occurs with Beta 0/5/6000is or Beta 1/6000is motors. The alarm may occur when torque limited acceleration is commanded to negative velocities faster than -5000 rpm. To prevent the error from occurring, reduce the acceleration rate so the motor does not operate in torque limit. If the acceleration cannot be reduced, enter Hardware Configuration Advanced Tab parameter 10019 (EMFCMP) with data value = 0. **Do not use the EMFCMP parameter for motors other than Beta 0/5/6000is or Beta 1/6000is. The only EMFCMP data value that should be used with these motors is zero.**

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 on page C-22, 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

* Values greater than 2400 are not recommended

DETQLM Deceleration Torque Limit.

The DETQLM parameter modifies the torque limit of the motor under deceleration conditions. DETQLM is set to a specific value for each motor type, as listed in “Parameters for HRV2 Control” of the *FANUC AC Servo Motor Parameter Manual*, B-65270EN. Normally the value of DETQLM should not be changed.

Disturbance Input Function

As shown in Digital Servo Velocity Loop Simplified Block Diagram on page C-18, the Disturbance Input Function is an internal function generator that applies vibration to an axis by entering a sinusoidal disturbance wave as a torque command. With this function, the frequency characteristics of the velocity loop (including machine sections) can be determined.

The Disturbance Input Function is controlled by five parameters:

Parameter Number	*Parameter Name*	*Description*	*Data Range*	*Default*
10256	BITPA20	BIT 15 = DSTIN BIT 14 = DSTTAN BIT 13 = DSTWAV	0 = Stop, 1 = Start 0 = 1 axis, 1 = 2 axis 0 = Sine Wave, 1 = Square Wave	0 0 0
10296	DI_GAIN	Disturbance Input Gain	0…7282	0
10297	S_FREQ	Disturbance Input Start Frequency	0…2000 Hz	10
10298	E_FREQ	Disturbance Input End Frequency	0…2000 Hz	200
10299	MPOINTS	Number of disturbance input measurement points	-32768…+32767	3

CAUTION

Read these CAUTIONS before using the Disturbance Input Function

1. **Do not activate the Disturbance Input Function unless the machine is in a state that will prevent damage from the oscillations caused by the function.**
2. **The Disturbance Input Function is superimposed on any other servo motion commands, so the servo should be stopped with the drive enabled before the Disturbance Input Function is activated.**
3. **Do not use data values for BITPA20 that set any bits other than DSTIN, DSTTAN and DSTWAV. Otherwise extreme servo oscillations may occur.**

BITPA20 – The only values that are recommended for this parameter are 0 and –32768 (8000 hex). A value of –32768 (8000 hex) sets bit DSTIN to start the disturbance input function.

DI_GAIN – A recommended gain value is 500. This represents a torque command with a peak value of 500/7282 = 6.8% of peak torque.

S_FREQ – Normally the default value of 10 Hz is a good start frequency.

E_FREQ – Normally the default value of 200 HZ is a good end frequency.

MPOINTS – Values of 10-100 work well for the number of points.

Note: A value for MPOINTS less than zero (such as –1) changes the frequency from swept mode to fixed mode. In that case the frequency is set by the S_FREQ parameter.

The values for DI_GAIN, S_FREQ, E_FREQ and MPOINTS can be set using the hardware configuration Advanced Tab or the MC_WriteParameter(s) function block. Values for BITPA20 should only be entered with the MC_WriteParameter(s) function block.

Resonance Elimination Filters

The PMM provides two notch filters that can be enabled to eliminate motor-machine resonance. As shown in the Digital Servo Velocity Loop Simplified Block Diagram on page C-18, these filters are in series with the torque command to the servo torque loop. Tuning Parameters 10330-10335 control the resonance elimination filters.

Resonance Elimination (RE) Filter Tuning Parameter 10330-10335 Data Values

Tuning Parameter Number	***Description***	***Tuning Parameter Data***
10330	REF2C -RE Filter 2 Attenuation center frequency	96 to 2000, (independent of the damping setting). Unit - Hz
10331	REF2W - RE Filter 2 Attenuation bandwidth	0 to attenuation center frequency (independent of the damping setting) Unit - Hz
10332	REF2D -RE Filter 2 Damping	0 to 100 (If it is 0, the attenuation ratio is maximized.) Unit - %
10333	REF3C -RE Filter 3 Attenuation center frequency	96 to 2000, (independent of the damping setting). Unit - Hz
10334	REF3W -RE Filter 3 Attenuation bandwidth	0 to attenuation center frequency (independent of the damping setting) Unit - Hz
10335	REF3D -RE Filter 3 Damping	0 to 100 (If it is 0, the attenuation ratio is maximized.) Unit - %

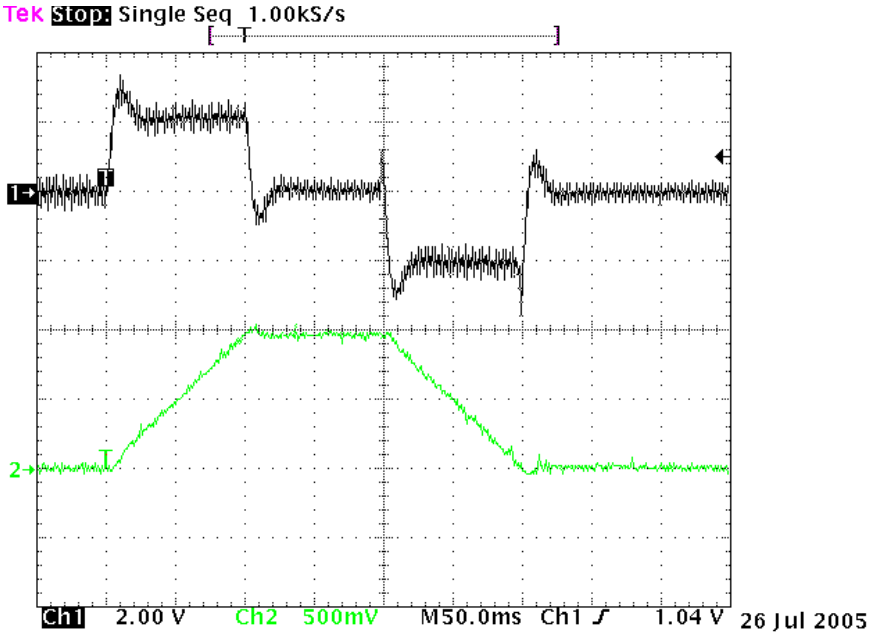
The resonance elimination filters are enabled if a nonzero value is set in the attenuation center frequency, bandwidth, or damping parameters. To disable a resonance elimination filter, do not enter the Tuning Parameters or set all three of its parameters to 0.

Resonance Elimination Filter Configuration Example

In this example, a motor has a large (3.25 times rotor inertia) inertial weight coupled to the shaft with no frictional damping. When the Load Inertia Ratio is set to an initial value of 32 x 3.25 = 104, a distinct whine can be heard when the axis drive is enabled.

1. To determine the frequency (or frequencies) involved in this resonance, the Torque Command and Actual Velocity for the axis of interest must be displayed in the Data Logger Window.
2. When an incremental move of 20,000 counts using a linear acceleration of 1,000,000 counts/sec² to a velocity of 100,000 counts/sec is executed, the following plot results.

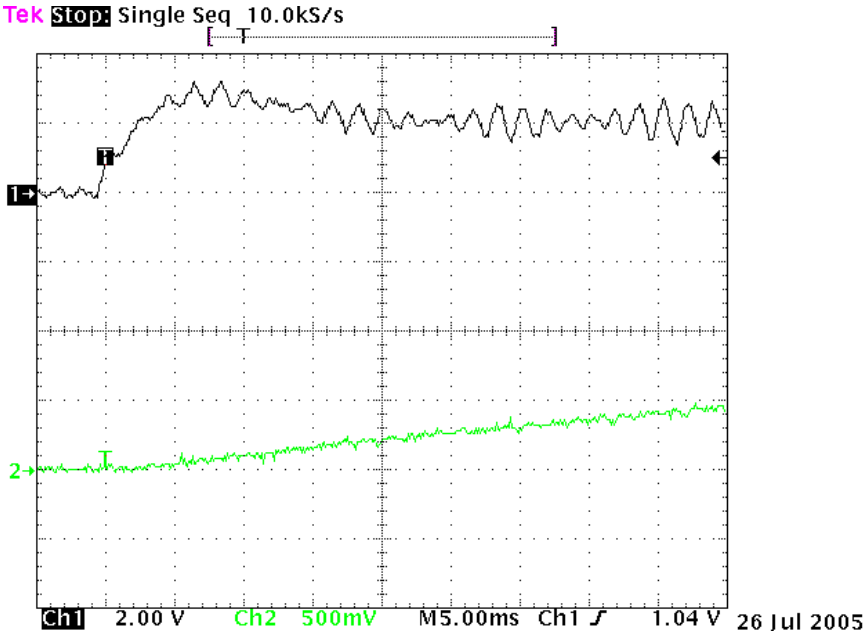
Axis 4 Torque Command and Actual Velocity Plots, 50ms/Division



The machine resonance shows up most clearly in this trace as a signal superimposed on the torque command plot. In more serious cases, the resonance may be visible in the actual velocity.

3. To get a better idea of the resonance frequency, the time base of the display is changed to 5 ms/division.

Axis 4 Torque Command and Actual Velocity Plots, 5ms/Division



4. Since slightly less than 6 cycles occur in 10 milliseconds, the resonant frequency can be approximated to 600 Hz, and a resonance filter configured accordingly. The center frequency (Axis 4 Par # 10330) is set to 600 (Hz), the bandwidth (Par # 10331) to 100 (Hz), and somewhat arbitrarily the damping (Par # 10332) to 10 (%).

   As shown in the following figure, the parameters for the second filter (Parameter #'s 10333, 10334, and 10335) are set to zero, so that the second filter is inactive but available if it is needed later.

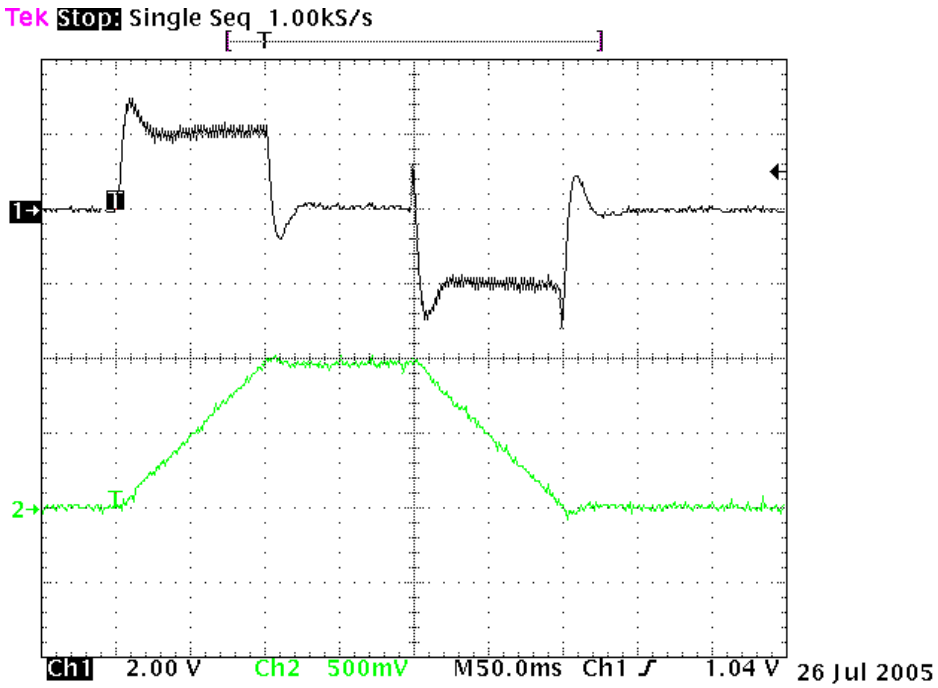
Sample RE Filter Configuration

Settings | FP I/O | FTB Inputs | FTB Outputs | I/O Interrupts | Axis 1 | Axis 2 | Axis 3 | Axis 4 | Axis 5 | Advanced | Power Consumption

Entry	Axis 1 Par #	Axis 1 Data	Axis 2 Par #	Axis 2 Data	Axis 3 Par #	Axis 3 Data	Axis 4 Par #	Axis 4 Data
Entry 1	0	0.0	10330	0.0	0	0.0	10330	600.0
Entry 2	0	0.0	10331	0.0	0	0.0	10331	100.0
Entry 3	0	0.0	10332	0.0	0	0.0	10332	10.0
Entry 4	0	0.0	10333	0.0	0	0.0	10333	0.0
Entry 5	0	0.0	10334	0.0	0	0.0	10334	0.0
Entry 6	0	0.0	10335	0.0	0	0.0	10335	0.0
Entry 7	0	0.0	0	0.0	0	0.0	0	0.0
Entry 8	0	0.0	0	0.0	0	0.0	0	0.0

5. When the drive is enabled after configuring the filter, the whine can no longer be heard. Running the same move as in step 2 results in the following plot.

Axis 4 Torque Command and Actual Velocity Plots with RE Filter Applied, 50ms/Division



The machine resonance signal, while not completely absent, is much reduced in amplitude.

In cases where the resonance contains multiple frequencies, rather than displaying signals in the time domain, the torque command and actual velocity signals may be better analyzed in the frequency domain to determine the predominant frequencies.

#

αHVis and αHVis Series
  amplifiers, 1-6
  motors, 1-6
β*i* and βHVi Series
  amplifiers, 1-5
  motors, 1-5

1

10V analog inputs, 3-23
10V analog output, 3-24

2

24V differential outputs, 3-17, 3-21
24V general purpose inputs, 3-16, 3-18
24V high speed inputs, 3-15

5

5V differential inputs, 3-20
5V differential outputs, 3-22
5V differential/single-ended inputs, 3-19
5V power, 3-24

A

Absolute encoder
  first time use, B-2
  use after loss of encoder battery power, B-2
Absolute encoder mode
  considerations, B-2
  find home cycle, B-3
  PMM power-up, B-3
  position initialization, B-2
  set position command, B-3
Absolute home switch mode. *See* Limit switch mode
Administrative instructions, 5-5
Advanced parameters, 4-52, C-17
Amplifiers
  αHVis and αHVis Series, 1-6
  β*i* and βHVi series, 1-5
Axis configuration data, 4-20
  data limits, 4-22
  parameters for Axes 1–4, 4-25
  parameters for Axis 5, 4-32
  user units to counts ratio, 4-20
Axis states, 5-27
  diagram, 5-29
Axis status bits, 6-135, 8-8

B

Bandwidth
  defined, C-14
Blending sectors (cam), 7-8
Boundary conditions (cam), 7-9
Buffer mode, 5-15
  blending, 5-16
Buffer Mode
  handling by motion function blocks, 5-17
  while jogging, 6-77

C

Cables, 3-8
Cam programming
  blending sectors, 7-8
  boundary conditions, 7-9
  cam types, 7-3
  overview, 7-1
  point limits in cam profiles, 7-2
  smoothing and curve fitting, 7-7
Configuration, 2-12
Connectors
  faceplate I/O, 3-2
  FTB I/O, 3-7
  ground, 3-28
Constants
  defined, 5-14
Continuous Motion
  state, 5-28
Customer care, 1-7

D

Data limits, 4-22
Data logging
  example, 6-45
  sweep impact, 6-44
Data types and structures, 5-24
Deferred response, 5-5
Diagnostic Logic Blocks (DLBs), 9-65
  cam profiles, 9-65
Diagnostics
  CPU error codes, 9-50
  Diagnostic Logic Blocks, 9-65
  digital servo alarms, 9-56
  Error IDs, 9-2
  LEDs, 3-13
  PMM error IDs:, 9-3
Digital cam switch
  accuracy calculations, A-5
Digital servo alarms, 9-55
Disabled state, 5-28
Discrete motion
  state, 5-27
Disturbance input function, C-20

E

Edge-triggered function blocks, 5-7
Enabled function blocks, 5-6
Encoder, quadrature
  operation, B-5
Error IDs
  CPU, 9-50
  format, 9-2
  PMM, 9-3
Errors
  due to changes to axis scaling, 9-62
  FSSB, 4-44
  FTB, 4-35
Errorstop state, 5-27
Event queue
  diagnostics, 9-60
Examples
  data logging, 6-45
  DCS Outputs input, 6-32
  DCS Switches input, 6-28
  DCS TrackOptions input, 6-33
  interrupt-driven executed MFBs, 5-11
  Jog to SW EOT with warning, 6-79
  MC_Power, 6-105
  move absolute, 6-90
  move additive, 6-94
  permissives with Execute input, 5-8
  position error calculation, 4-46
  read analog input, 6-109
  read bool parameter, 6-115
  read bool parameters, 6-117
  read digital input, 6-119
  read event queue, 6-126
  read parameters, 6-132
  scaling (UU/cts), 4-21
  touch probe, 6-152
  write analog output, 6-157
  write digital output, 6-164
  writing to a digital output, 5-9
Executed function blocks, 5-7

F

Feed hold, 6-142
Feedback devices
  quadrature encoder, B-5
Find home cycle, B-3
Find Home routine, 6-72
First time use, absolute encoder, B-2
Flow
  defined, 5-14
FSSB
  cables, 3-8, 3-25
  connector, 3-25
  servo drive tuning, C-6
  servo system diagnostics, C-5
FTB (Fiber I/O Terminal Block)
  configuration, 4-10
  error, 4-35
Functions and function blocks
  administrative, 5-5
  behavior, 5-4
  buffering, 5-15
  data types and structures, 5-24
  immediate vs. deferred response, 5-5
  input parameters, 5-15
  instance data, 5-4
  motion-generating, 5-5
  output parameters, 5-20
  parameters, 5-14
  triggering, 5-6

G

Grounding, 3-27
  FTB faceplate shield, 3-28
  I/O cable, 3-29
Grounding systems, 3-27

H

Hardware configuration
  adding a PMM, 4-3
  Advanced, 4-52, C-17
  Axis data, 4-20
  FP (faceplate) I/O, 4-8, 4-16
  FTB (Fiber I/O Terminal Block), 4-10
  FTB (fiber terminal block) I/O, 4-17
  I/O interrupts, 4-18
  parameters, 4-4
  Power consumption, 4-52
  Settings, 4-5
  Terminals, 4-52
  Virtual axis, 4-32
Home switch input
  validating, C-4
Homing
  state, 5-27
Human-machine interfaces, 2-5

I

I/O data reference numbers, 8-21
I/O Specifications, 3-15
  10V analog inputs, 3-23
  10V analog output, 3-24
  24V differential output, 3-17, 3-21
  24V general purpose input, 3-16, 3-18
  24V high speed input, 3-15
  5V differential, 3-20
  5V differential output, 3-22
  5V differential/single ended, 3-19
  5V power supply, 3-24
I/O status data, 4-7, 4-52

Immediate response, 5-5
Incremental encoder mode
considerations, B-2
Indicators
FTB, 3-12, 3-14
PMM, 3-5, 3-13
Input parameters, 5-15
Installation
FTB, 2-8
PMM, 2-6
Instance data, 5-4
Instructions
behavior, 5-4
instance data, 5-4
InSync
synchronized motion status, 7-13
Interface
machine, 2-5
operator, 2-5
PMM/RX3i, 2-3
Interpolation and smoothing, 7-7
Interrupts
calling executed MFBs, 5-11
hardware configuration, 4-18
maximjum frequency, 4-18, 9-14
procedure, 4-19

J

Jerk Units, 5-18
Jog
Jogging state, 5-28

L

LEDs
FTB, 3-12, 3-14
PMM, 3-5, 3-13
Limit, 6-72
Loss of encoder battery power, B-2

M

Max Position Lag, 4-45
Mode
absolute encoder, B-2
digital series encoder, B-2
incremental encoder, B-2
Mode, Limit switch, 6-72
Modes, serial encoders, B-2
Module status, 8-16
Module status bits, 4-7, 4-52
Module status code, 8-16
Motion Functions and Function Blocks
MC_ReadAxisError\f, 6-112
Motion instructions
behavior, 5-4
data types and structures, 5-24
error IDs, 9-2
immediate vs. deferred response, 5-5
instance data, 5-4
parameters, 5-14
triggering, 5-6
Motion-generatinginstructions, 5-5
Motor direction
validating, 4-40, C-2
Motors, 1-5
αHVis and αHVis Series, 1-6
βHV*i*s Series, 1-5
β*i*s Series, 1-5
Motor Type, 4-43
Motor Type codes, 2-17
synthetic, 2-22, 4-43
Move+ and Move– modes, 6-74
Multiple instances, 5-6

O

Output parameters, 5-20
Over travel limit switch inputs
validating, C-3

P

Parameter numbers
axis, 8-2
module, 8-16
Parameters
advanced, C-17
digital servo, C-17
resonance elimination, C-22
torque command filter, C-20
velocity loop gain, C-18
function and function block, 5-14
hardware configuration, 4-4
Part numbers
cables, 3-8
FSSB cables, 3-8, 3-25
terminal headers, 3-8
Pending synchronized motion status, 7-10
Permissives
defined, 5-14
with Execute input, 5-8
Pin assignments
faceplate I/O connector, 3-3
FTB terminal 1, 3-9
FTB terminal 2, 3-10
PLCOpen
axis states, 5-27
Position error
defined, 4-46
synthetic motor, 2-22
Position initialization, B-2
Power-up, B-3

Programmable limit switch (digital cam switch)
enabling outputs, 6-27
mask values, 6-27
operation, 6-24
Programming software
connecting, 4-2

Q

Quadrature encoder
operation, B-5

R

Ramping synchronized motion status, 7-12
Reference numbers, I/O, 8-21
Resonance
defined, C-14
elimination filters, C-22

S

Scaling
analog outputs, 8-17
errors caused by changing, 8-15, 9-62
example, 4-21
Serial encoder modes, B-2
Servos, 1-5
Motor Type, 4-43
Motor Type codes, 2-17
Set position command, B-3
Setup
state, 5-28
Severity
Error IDs, 9-2
Shared data
I/O status, 4-7, 4-52
Shield ground connection, 3-28
Specifications
Fiber I/O Terminal Block, 3-7
Standstill state, 5-27
Startup
basic configuration, 2-12
diagnostics, C-5
installation, 2-6
Status data
axis status bits, 6-135, 8-8
module status bits, 4-7, 4-52
Stop
fast vs. normal, 9-2
Stopping
state, 5-28
Superseded instance, 5-6, 9-27
Synchronized motion, 5-30
function block status, 7-10
state, 5-28
Synthetic motor, 2-22, 4-43

T

Technical support, 1-7
Terminal block
headers, 3-8
Touch probe
accuracy calculations, A-1
Tuning
advanced parameters, 4-52, C-17
position loop, C-14
requirements, C-6
velocity feedforward, C-16
velocity loop, C-7
Tuning parameters, 8-12

U

User Units to counts ratio, 4-20

V

Variables
defined, 5-14
linked, 5-18
reference ID, 5-18

W

Wiring connections
quadrature encoder, 4-40, C-2
Wiring diagrams
FTB, 3-9
PMM faceplate, 3-3

Motion Functions and Function Blocks

- MC_AbortTrigger, 6-2
- MC_CamFileRead, 6-4
- MC_CamFileWrite, 6-6
- MC_CamIn, 6-8
- MC_CamOut, 6-16
- MC_CamTableDeselect, 6-18
- MC_CamTableSelect, 6-20
- MC_DelayedStart, 6-22
- MC_DigitalCamSwitch, 6-24
- MC_DL_Activate, 6-35
- MC_DL_Configure, 6-37
- MC_DL_Delete, 6-42
- MC_DL_Get, 6-44
- MC_GearIn, 6-56
- MC_GearInPos, 6-61
- MC_GearOut, 6-65
- MC_Halt, 6-67
- MC_Home, 6-69, B-3
- MC_JogAxis, 6-76
- MC_LibraryStatus, 6-83
- MC_ModuleReset, 6-85
- MC_MoveAbsolute, 6-87
- MC_MoveAdditive, 6-91
- MC_MoveRelative, 6-95
- MC_MoveSuperimposed, 6-97
- MC_MoveVelocity, 6-99
- MC_Phasing, 6-101
- MC_Power, 6-103
- MC_ReadActualPosition, 6-106
- MC_ReadActualVelocity, 6-107
- MC_ReadAnalogInput, 6-108
- MC_ReadAnalogOutput, 6-110
- MC_ReadAxisError, 6-112
- MC_ReadBoolParameter, 6-114
- MC_ReadBoolParameters, 6-116
- MC_ReadDigitalInput, 6-118
- MC_ReadDigitalOutput, 6-120
- MC_ReadDwordParameters, 6-122
- MC_ReadEventQueue, 6-124
- MC_ReadParameter, 6-129
- MC_ReadParameters, 6-131
- MC_ReadStatus, 6-133
- MC_ReadTorqueCommand, 6-138
- MC_Reset, 6-139
- MC_SetOverride, 6-141
- MC_SetPosition, 6-144, B-3
- MC_Stop, 6-146
- MC_SyncStart, 6-148
- MC_TouchProbe, 6-150
- MC_WriteAnalogOutput, 6-156
- MC_WriteBoolParameter, 6-158
- MC_WriteBoolParameters, 6-160
- MC_WriteDigitalOutput, 6-162
- MC_WriteDwordParameters, 6-165
- MC_WriteParameter, 6-167
- MC_WriteParameters, 6-169